cini consorzio interuniversitario nazionale per l'informatica

UNIVERSITÀ DI PADOVA
Dipartimento di Matematica "Tullio Levi-Civita"

INFORMATICA E SCUOLA
LABORATORIO NAZIONALE cini

Progetto Nazionale "Informatica" del Piano Lauree Scientifiche

APS Programma il Futuro

DIPARTIMENTO DI INFORMATICA

Padova, 9-11 ottobre 2026

[ITADINFO]

CONVEGNO ITALIANO SULLA DIDATTICA DELL'INFORMATICA



SPONSOR

axios Italia

EDIZIONI THEMIS

seeweb THINK CLOUD

CON IL PATROCINIO DI

PATROCINIO REGIONE DEL VENETO

UNIVERSITÀ DI PADOVA

GRIN Società informatica italiana

GII

MEDIA PARTNER

START MAGAZINE

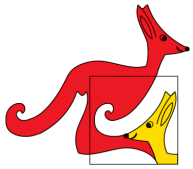
CON IL SOSTEGNO DI FONDAZIONE CRT

Fondazione CRT

Progetto DIDEROT

I contributi qui pubblicati sono stati revisionati dal Comitato Scientifico del Convegno con *single blind peer-review*.

Edizioni Kangourou



ITADINFO 2026. Quarto Convegno Italiano sulla Didattica dell'Informatica.
Impaginazione, editing e revisione del presente volume a cura di Violetta Lonati e Gabriele Pozzan.

Foto di copertina da Pixabay.

I contenuti del presente volume sono pubblicati con la licenza *Creative Commons 4.0 International Attribution-NonCommercial-ShareAlike*.



Prima edizione: ottobre 2026

ISBN: 979-12-81294-09-7

Comitato organizzatore

Presidenza e relazioni istituzionali

Enrico Nardelli, Università di Roma “Tor Vergata”
Tullio Vardanega, Università di Padova

Comitato di programma

Presidenza

Violetta Lonati, Università di Milano

Componenti

Sara Capecchi, Università di Torino
Giorgio Delzanno, Università di Genova
Michael Lodi, Università di Bologna
Claudio Mirolo, Università di Udine
Anna Morpurgo, Università di Milano
Maria Angela Pellegrino, Università di Salerno

Comunicazione

Michael Lodi, Università di Bologna

Sponsorizzazioni

Tullio Vardanega, Università di Padova

Ufficio stampa e social media

Reputation Agency, https://reputationagency.eu/it/

Segreteria organizzativa e sito web

Francesco Lacchia, APS Programma il Futuro

Organizzazione locale

Gabriele Pozzan, Università di Padova
Tullio Vardanega, Università di Padova

Atti del convegno

Violetta Lonati, Università di Milano
Gabriele Pozzan, Università di Padova

Comitato scientifico

Guglielmo Abbruzzese, Università di Roma “Tor Vergata”
Agnese Addone, Università di Salerno
Giuliana Barberis, Liceo Scientifico “Maria Curie” Pinerolo (TO)
Antonia Giovanna Bellò, docente di scuola secondaria di primo grado in pensione
Maurizio Boscaini, ITI G. Marconi di Verona
Claudia Canali, Università di Modena e Reggio Emilia
Sara Capecchi, Università di Torino
Antonella Carbonaro, Università di Bologna
Maria Cristina Carrisi, Università di Cagliari
Renzo Davoli, Università di Bologna
Giorgio Delzanno, Università di Genova
Agnese Del Zozzo, Università di Trento
Valentina Di Noi, Università di Torino
Filomena Ferrucci, Università di Salerno
Luca Forlizzi, Università dell’Aquila
Ilenia Fronza, Università di Bolzano
Maurizia Gai, Ist. Comp. Vivaldi-Murialdo di Torino
Enrica Gentile, Università di Bari
Giovanna Guerrini, Università di Genova
Gennaro Iaccarino, I.I.S.S. “G. Galilei” di Bolzano, docente comandato presso la Dir. Istr. e Formazione italiana di Bolzano
Luca Lamanna, Università di Milano
Michael Lodi, Università di Bologna
Violetta Lonati, Università di Milano
Sabrina Mantaci, Università di Palermo
Simone Martini, Università di Bologna
Gionata Massi, IIS Savoia Benincasa di Ancona
Giovanna Melideo, Università dell’Aquila
Claudio Mirolo, Università di Udine
Mattia Monga, Università di Milano
Alberto Montresor, Università di Trento
Anna Morpurgo, Università di Milano
Enrico Nardelli, Università di Roma “Tor Vergata”
Martina Palazzolo, Ist. Comp. Ilaria Alpi di Milano
Maria Angela Pellegrino, Università di Salerno
Veronica Rossano, Università di Bari
Marcello Sarini, Università di Milano “Bicocca”
Emanuele Scapin, ITT Giacomo Chilesotti di Thiene
Tullio Vardanega, Università di Padova

Indice

Prefazione 11
Violetta Lonati

Sessione plenaria **13**

Una lente di genere per progettare attività inclusive di informatica 14
Giovanna Guerrini

Contributi per scuola dell'infanzia e primo ciclo **16**

Aprite il sussidiario: dalla produzione editoriale a una guida "open" per insegnare Informatica nel primo ciclo 17
Michael Lodi, Agnese Del Zozzo, Alberto Montresor e Giorgia Bissoli

La crittografia? È un gioco! 22
Arianna Boldi, Sara Capecchi, Iary Davidson e Marco Viola

Codey Rocky va in favola - Un percorso di narrazione computazionale inclusiva secondo il framework UDL nella scuola primaria 24
Anna Levrero

Imparare a programmare senza schermi: evidenze sperimentali di un intervento di coding tangibile in età prescolare 29
Chiara Montuori, Gabriele Pozzan, Costanza Padova, Lucia Ronconi, Tullio Vardanega e Barbara Arfé

Il Ritratto Statistico: un gioco da tavolo per smontare la magia dell'IA generativa . . 36
Barbara Baldi e Giuliana Gelsomino

Attività di programmazione per lo studio della geometria nella scuola secondaria di primo grado 41
Martina Palazzolo

Un PEZZETTINO... alla volta 48
Maurizia Gai, Cinzia D'Elia, Laura Galati e Virginia Macrì

L'Italia sul telo: l'informatica come mediatore interdisciplinare tra coding e creatività 54
Andrea Goia e Elena Esposito

Funzioni di hash nella scuola primaria: una sperimentazione unplugged tra informazione, algoritmi e sicurezza digitale 59
Maria Cristina Carrisi, Michael Lodi, Simone Martini e Maria Angela Pellegrino

La ricorsione con un albero 64

Giovanni Bellucci

Lo sviluppo di app mobili come leva per l'acquisizione di competenze informatiche nella scuola secondaria di primo grado 73
Daniela Pieraccini

Contributi per la secondaria di secondo grado **78**

Interazione Fisica e Pensiero Computazionale: Fare, descrivere, modellizzare 79
Gionata Massi e Emanuele Lorenzoni

Dal debate alla modellazione computazionale: un'esperienza didattica sugli Abstract Argumentation Framework 84
Francesco Paolo Caforio

Il cliente ha sempre ragione? Un laboratorio di ingegneria del software per la scuola superiore 91
Daniele Fea e Manuela Marchisio

Valutare il processo di sviluppo software: repository GitHub, rubriche e feedback rapido nella didattica dell'informatica 95
Francesco Adriani

Dalla didattica della matematica alla didattica dell'informatica: il modello OSCAR come framework di valutazione affettivo-cognitivo 99
Gennaro Iaccarino e Ilenia Fronza

La cripta del faraone perduto - Un'esperienza di Digital Storytelling, Gamification, Robotica e IoT con Arduino 104
Michele Caceffo, Marco Cuoghi e Nicola Dalla Pozza

Dal codice all'azione: imparare Python con i robot 112
Laura Cesaro e Emanuele Menegatti

Dal codice al comportamento: robotica sociale con NAO per l'apprendimento di programmazione, debugging e interazione uomo-macchina 118
Veronica Cavicchi, Matteo Bettini, Nikolai Carnaghi, Ibrahim Mohamad Barghouthi, Gabriele Chesini, Lorenzo Perotti, Mariapia Anna Porcelli e Sunaina Ayushi Rani

L'intelligenza artificiale e i progetti Wikimedia: tra infrastruttura della conoscenza, sostenibilità e governance umana 122
Piergiovanna Grossi e Lorenzo Losa

L'intelligenza artificiale generativa è razzista? 126
Giuseppe Mastrandrea, Laura Castellana, Francesco Scanni, Cristian Squeo, Alessandro Iacobbe e Simone Albanese

La programmazione visuale come introduzione all'Informatica nel primo biennio del liceo131
Gabriele Mottola e Giulia Paludo

Riprogettare la didattica dell'informatica con l'IA: scaffolding, esempi guidati e strumenti per la programmazione al SIA 136
Matteo Candita, Michael Lodi e Simone Martini

Dal codice al prompt: ridefinire apprendimento e valutazione della programmazione nell'era degli assistenti IA 142

Giovanni Pedroncelli

Dall'introduzione dell'Intelligenza Artificiale alla progettazione di soluzioni reali: il caso UrbanAI 149
Barbara Nardi, Daniele Rogano e Francesco Rende

Programmare senza guardare: resoconto di un'esperienza di insegnamento 153
Graziano Battisti, Luca Forlizzi, Giovanna Melideo e Gaia Piersanti

"mTestAI" Valuta e migliora le competenze informatiche con il feedback assistito dall'AI 158
Gaetana De Gioia, Enrica Gentile, Paola Plantamura e Francesca Pia Travisani

Gem Misteriosi: analizzare, testare e ricostruire la specifica di chatbot personalizzati . 164
Marta Valentini, Ester Dalvit e Alberto Montresor

Intelligenza Artificiale a supporto della Valutazione Formativa nell'insegnamento della Programmazione 171
Alessandro Pagano, Paola Plantamura, Veronica Rossano e Sara Maria Cosima Tondo

Giochi di Numeri: Dall'aritmetica al pensiero ricorsivo 178
Gionata Massi, Alberto Cesaretti e Pamela Sanchini

Valutare il pensiero computazionale: operazionalizzazione e criticità inferenziali nella ricerca sul transfer cognitivo 186
Giulia Paludo, Alberto Montresor e Angela Pasqualotto

Poster **190**

"Cerco, capisco, imparo". Esperienza extracurricolare per il potenziamento del pensiero logico attraverso l'informatica 191
Emilia Ciccia e Marilena Salsano

Dal gioco al codice: il postino che insegna gli algoritmi 198
Laura Cusanno e Simonetta Siega

Ragazze Robotiche 201
Lucia Salari

Verso la programmazione orientata agli oggetti: un percorso introduttivo con App Inventor nella scuola secondaria. 208
Ersilia Pagano

L'Odissea: una Stochastic Gradient Descent omerica? Alla ricerca di nuove piste nell'insegnamento dell'Informatica 213
Emanuele Scapin, Furio Honsell e Alessandra Chilese

Dalla SEO al Web Agentico: l'evoluzione della didattica dell'informatica in una simulazione d'impresa 221
Sonia Guerci

Il respiro delle città: cittadinanza digitale attiva attraverso il monitoraggio informatico della qualità dell'aria 228
Anna Lucia Peluso, Barbara Avella e Daniela Pieraccini

Dal Mattone al Progetto: Cambiare Didattica con il Vibe Coding 233
Simona Pinto

Future Climate Lab: ambienti virtuali, coding e AI per il pianeta del futuro 236
Alessia Galli e Lara Rollo

Con il mare davanti agli occhi: MIT App Inventor e la didattica dell'informatica in un contesto periferico 241
Pierangelo Indolfi

L'informatica entra al Classico: un percorso graduale di programmazione e pensiero computazionale al Liceo "Frezzi - B. Angela" di Foligno 245
Enrica Donati e Roberta Barluzzi

Prompt Engineering come pensiero algoritmico: un caso studio tra collaborazione e autonomia individuale 251
Erica Perseghin e Giacomo Salvi

Vertical CanSatAIr: Monitoraggio verticale dell'inquinamento atmosferico e definizione di un indice composito di rischio respiratorio nella bassa troposfera 259
Giuseppe Fiamingo e Antonio Siciliano

Debugging Accessibility: ripensare il debugging nella didattica dell'informatica per studenti con DSA 267
Annunziata Marra

Bebras Explorer: valorizzare i quesiti Bebras come risorsa per l'insegnamento dell'informatica 275
Carlo Bellettini, Violetta Lonati, Mattia Monga e Anna Morpurgo

Paper Bots AI: introdurre machine learning e programmazione a blocchi nella scuola secondaria 281
Alessia Galli e Lara Rollo

Ha ancora senso insegnare l'assembly nel 2026? Dal TASM al cloud: architettura dei calcolatori e cybersecurity 286
Pierangelo Indolfi

Educare sviluppatori responsabili: Dai fruitori attuali ai creatori consapevoli 290
Fabio Sartori e Lorenzo Caputo

Insegnare l'Intelligenza Artificiale con Umanità: superare le barriere attraverso l'AI etica e inclusiva 293
Daniela Pieraccini

Laboratorio di introduzione alle reti neurali 298
Sara Capecchi, Antonio Giordano, Paolo Milan e Daniele Paolo Radicioni

Detective dell'IAG: alla scoperta dei bias nascosti per educare alla cittadinanza digitale attraverso l'Intelligenza Artificiale Generativa 306
Ippolita Gallo

Dall'algoritmo ai sistemi reattivi: sviluppo e sperimentazione di un percorso curricolare di Informatica nella SSPG 315
Emanuela Scicchitano e Anna Del Vecchio

Esperienza sul campo nella formazione di docenti: wAIcky races - un Contest su uso di IA in macchine a guida autonoma 319
Luca Basteris e Carmen Vittorio

Logica e Intelligenza Artificiale: pratiche didattiche per il Liceo Classico 327
Simone Cuconato e Kristian Reale

Oltre il coding: Live Action Role-Playing e tecnologie creative nei camp Ragazze Digitali331
Linda Burchiellaro, Mirko Novello, Martina Casari, Francesco Faenza e Claudia Canali

Regressione lineare come laboratorio interdisciplinare: un percorso integrato tra informatica e matematica 336
Francesca Arrigoni e Francesca Voltolini

Sperimentazione di un Liceo Classico a curvatura "digitale" 343
Luca Basteris e Maria Cristina Daperno

Laboratori **349**

Insegnare informatica nel primo ciclo: dalle Indicazioni alle attività in classe 350
Michael Lodi, Agnese Del Zozzo, Alberto Montresor e Giorgia Bissoli

Come far capire cos'è un automa 352
Enrico Nardelli

Insegnare a programmare componendo grafiche con PyTamaro 354
Luca Chiodini, Joey Bevilacqua e Matthias Hauswirth

Colori, Cassetti e Impronte: Come Funzionano le Hash 356
Maria Cristina Carrisi, Michael Lodi, Simone Martini e Maria Angela Pellegrino

"Questo colore non c'è sui blocchi!" Insegna un colore nuovo al tuo robot con il ML (e sopravvivi a Python!) 358
Laura Cesaro, Daniel Fusaro e Emanuele Menegatti

TCPigeon: comprendere TCP attraverso i piccioni viaggiatori 360
Renzo Davoli

I server: dietro le quinte della rete 362
Sara Capecchi e Daniele Catozzella

Pensiero computazionale e linguaggi artificiali: introduzione a Python con micro:bit e il supporto critico dell'IA 364
Barbara Baldi e Giuliana Gelsomino

AI-Earthquake Monitoring: Machine Learning con micro:bit 366
Carla Vitale

Riskiatutto: Scopri l'Assembly RISC-V con ESP32 368
Giorgio Delzanno

Programmare un computer nel computer gestendo il carico cognitivo 370
Gabriele Pozzan e Tullio Vardanega

Apprendimento di SQL basato su errori frequenti 372
Giovanna Guerrini e Davide Ponzini

Video **374**

Metodologie per la Divulgazione Informatica nel Formato Short-Form Video 375
Valerio Conigliaro

Dal problema al programma... con una manciata di fiammiferi! 381
Martina Palazzolo

Pensare Informatico 382
Michael Lodi

Funzione Hash - Framework Metodologico Short-Form-Video 383
Valerio Conigliaro

Prefazione

Il Convegno Italiano sulla Didattica dell'Informatica (ITADINFO) nasce con l'obiettivo di creare uno spazio di incontro e confronto tra chi insegna Informatica nella scuola, a tutti i livelli, e chi si occupa di ricerca sull'apprendimento/insegnamento dell'Informatica. Giunto alla sua quarta edizione, ITADINFO continua a raccogliere e mettere in dialogo esperienze, pratiche, risultati di ricerca e proposte formative, con l'intento comune di contribuire alla costruzione e alla diffusione di una cultura dell'Informatica come disciplina scientifica.

Dopo Bari, Genova e Salerno, l'edizione 2026 si è svolta a Padova, dal 9 all'11 ottobre, confermando la formula che caratterizza il convegno: accanto alle relazioni scientifiche trovano spazio i racconti di esperienze realizzate nelle scuole, i poster, i laboratori formativi e i video divulgativi brevi. La varietà delle forme di partecipazione riflette la varietà della comunità che si riconosce in ITADINFO: insegnanti di diversi ordini di scuola, ricercatrici e ricercatori, persone impegnate nella formazione e nella progettazione di attività e materiali per l'apprendimento/insegnamento dell'Informatica.
Questa edizione si colloca in un momento di svolta per la scuola italiana. Le nuove Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo, in vigore dall'anno scolastico 2026/27, introducono per la prima volta obiettivi di apprendimento di Informatica fin dalla scuola primaria, inserendoli all'interno di Matematica e Tecnologia. Il convegno diventa così anche un'occasione per confrontarsi su come tradurre questi obiettivi in pratica didattica e in percorsi di formazione per i docenti.

Gli atti raccolgono 75 contributi, comprendendo 1 lavoro presentato in sessione plenaria, 11 contributi dedicati alla scuola dell'infanzia e al primo ciclo, 20 contributi per la scuola secondaria di secondo grado, 27 poster, 12 laboratori e 4 contributi dedicati alla divulgazione attraverso video. I lavori coprono livelli scolastici, approcci metodologici e ambiti dell'Informatica molto diversi tra loro. Una prima caratteristica che emerge dal programma è l'attenzione ai concetti e alle pratiche fondamentali dell'Informatica, nonché alla loro trasposizione didattica: algoritmi, programmazione e debugging, strutture dati, reti, architettura dei calcolatori sono affrontati attraverso giochi, attività unplugged, robot e dispositivi programmabili, ambienti visuali e linguaggi testuali, in percorsi rivolti a diverse fasce d'età. Tra i temi affrontati figura anche l'intelligenza artificiale, considerata sia come oggetto di insegnamento sia in relazione al suo utilizzo nella didattica dell'Informatica, con contributi su machine learning, bias, chatbot e sistemi generativi. Numerosi contributi riguardano la progettazione di attività e percorsi didattici e la valutazione dell'apprendimento, con proposte per lo sviluppo e la valutazione delle competenze Informatiche, la riflessione sul pensiero computazionale e il suo trasferimento in altri contesti.

Sono presenti anche contributi dedicati all'inclusione, all'accessibilità e alla formazione degli insegnanti; il programma evidenzia inoltre le connessioni dell'Informatica con altre discipline e

con problemi del mondo reale, dalla matematica e dall’analisi dei dati alla robotica, alla cybersecurity e al monitoraggio ambientale.

Ringraziamo tutte le persone che hanno contribuito a questa quarta edizione del convegno: chi ha preparato e presentato un contributo, chi ha condotto un laboratorio e chi ha partecipato alle discussioni; i membri del Comitato Scientifico, che con il loro lavoro di valutazione hanno aiutato autrici e autori a migliorare i propri contributi; il comitato organizzatore locale dell'Università di Padova; la segreteria organizzativa, gli sponsor e i patrocinatori. Un ringraziamento particolare va a Enrico Nardelli e Tullio Vardanega, che hanno guidato l'organizzazione, a Gabriele Pozzan, che ha curato la redazione di questi atti, seguendo autrici e autori con grande pazienza e competenza; ai membri del Comitato di Programma — Sara Capecchi, Giorgio Delzanno, Michael Lodi, Claudio Mirolo, Anna Morpurgo e Maria Angela Pellegrino — il cui prezioso lavoro nella fase di selezione dei contributi è stato indispensabile.

Ci auguriamo che questi atti, più che un resoconto di ITADINFO 2026, possano essere uno strumento di lavoro sia per chi insegna già Informatica sia per chi si avvicina adesso alla disciplina, e un invito a continuare a costruire insieme una comunità aperta e in dialogo.

Violetta Lonati
Presidente del Comitato di Programma ITADINFO 2026

Sessione plenaria

Una lente di genere per progettare attività di informatica inclusive

Giovanna Guerrini

Università di Genova, Italia
giovanna.guerrini@unige.it

L'introduzione strutturale dell'informatica nel curricolo del primo ciclo, prevista dalle nuove Indicazioni Nazionali a partire dall'anno scolastico 2026/27, rappresenta un'importante opportunità per avvicinare precocemente tutte le studentesse e tutti gli studenti ai concetti e ai metodi propri della disciplina. Questa opportunità impone però di prestare particolare attenzione alle modalità con cui questa nuova esperienza viene proposta: l'accesso universale all'insegnamento non garantisce infatti, di per sé, che ragazze e ragazzi sviluppino analoghe opportunità di partecipazione, apprendimento e identificazione con la disciplina.

Un recente studio longitudinale condotto sull'intera popolazione scolastica francese ha mostrato come, in matematica, una differenza di genere pressoché assente all'ingresso nella scuola primaria emerga rapidamente durante il primo anno di scolarizzazione, diventando significativa già dopo pochi mesi [2]. Questo risultato suggerisce l'importanza di interrogarsi non soltanto sulle differenze osservate nelle scelte scolastiche successive, ma anche sul ruolo che le prime esperienze didattiche possono avere nel contribuire alla loro costruzione. Nel momento in cui l'informatica entra sistematicamente nel percorso scolastico di bambine e bambini, appare quindi particolarmente importante progettare fin dall'inizio ambienti di apprendimento che non contribuiscano alla formazione o al consolidamento di analoghi divari.

Anche nella Computing Education emergono infatti differenze di genere già nei primi anni di esposizione alla disciplina. Studi condotti nella scuola primaria mostrano come dinamiche di genere possano manifestarsi nelle attività collaborative e nei risultati prodotti dai gruppi [5]. Caratteristiche diverse della progettazione didattica possono incidere sull'esperienza di ragazze e ragazzi: dal contesto e dal framing con cui viene presentata un'attività informatica [1], alla scelta di problemi socialmente rilevanti e alle modalità con cui questi vengono affrontati, con effetti che possono essere differenti sull'apprendimento e sugli atteggiamenti verso l'informatica [3]. Diventa quindi rilevante spostare l'attenzione dalla domanda "come interessare le ragazze all'informatica?" a una prospettiva più generale: *in che misura una progettazione didattica consapevole delle possibili dinamiche di genere può favorire opportunità più eque di apprendimento, partecipazione, senso di appartenenza e costruzione dell'identità disciplinare nell'informatica?*

Questa domanda porta alla definizione di una *lente di genere* per la progettazione e l'analisi di attività di didattica dell'informatica. A partire da framework quali *Teaching for Robust Understanding* (TRU) [4], che individua tra le dimensioni fondamentali di un ambiente di apprendimento l'accesso equo ai contenuti, le opportunità di partecipazione, l'agency e lo sviluppo dell'identità disciplinare, si possono integrare indicazioni provenienti dalla letteratura sull'inclusività di genere nelle STEM e, in particolare, nella Computing Education, per individuare criteri osservabili con cui analizzare e riprogettare le attività didattiche. Tali criteri possono riguardare, ad esempio, la scelta dei contesti e degli esempi, il linguaggio e le rappresentazioni, le modalità di lavoro collaborativo e di distribuzione dei ruoli, le opportunità di contribuire alla soluzione e le modalità di interazione e feedback.

L'obiettivo è contribuire alla definizione di strumenti operativi che consentano a insegnanti e progettisti di interrogarsi sistematicamente sull'inclusività delle attività proposte, facendo della prospettiva di genere non un intervento correttivo successivo all'emergere di un divario, ma una dimensione della progettazione didattica fin dalle prime esperienze scolastiche con l'informatica.

References

[1] I. M. Christensen, M. H. Marcher, P. Grabarczyk, T. Graversen, and C. Brabrand, "Computing Educational Activities Involving People Rather Than Things Appeal More to Women (Recruitment Perspective)," in *Proceedings of the 17th ACM Conference on International Computing Education Research (ICER '21)*, pp. 127–144, 2021. doi: 10.1145/3446871.3469758.

[2] P. Martinot, B. Colnet, T. Breda, et al., "Rapid emergence of a maths gender gap in first grade," *Nature*, vol. 643, pp. 1020–1029, 2025. doi: 10.1038/s41586-025-09126-4.

[3] A. T. Ottenbreit-Leftwich, K. Kwon, T. A. Brush, M. Karlin, M. Jeon, K. Jantaraweragul, M. Guo, H. Nadir, F. Gok, and P. Bhattacharya, "The impact of an issue-centered problem-based learning curriculum on 6th grade girls' understanding of and interest in computer science," *Computers and Education Open*, vol. 2, 100057, 2021. doi: 10.1016/j.caeo.2021.100057.

[4] A. H. Schoenfeld and the Teaching for Robust Understanding Project, *An Introduction to the Teaching for Robust Understanding (TRU) Framework*. Graduate School of Education, University of California, Berkeley, 2016.

[5] J. Tsan, K. E. Boyer, and C. F. Lynch, "How Early Does the CS Gender Gap Emerge? A Study of Collaborative Problem Solving in 5th Grade Computer Science," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*, pp. 388–393, 2016. doi: 10.1145/2839509.2844605.

Contributi per scuola dell'infanzia e primo ciclo

Aprite il sussidiario: dalla produzione editoriale a una guida "open" per insegnare Informatica nel primo ciclo

Michael Lodi[1,3], Agnese Del Zozzo[2,3], Alberto Montresor[2,3], and Giorgia Bissoli[2,4]

[1] Alma Mater Studiorum - Università di Bologna
michael.lodi@unibo.it
[2] Università di Trento
{agnese.delzozzo, alberto.montresor, giorgia.bissoli}@unitn.it
[3] Laboratorio CINI Informatica e Scuola
[4] Verona FabLab

Abstract

Raccontiamo la progettazione di un fascicolo di Informatica per la primaria e della relativa guida docente, pubblicata da Mondadori e resa disponibile in una versione derivata con licenza CC BY-NC-SA. Discutiamo i principi che hanno guidato il lavoro: distinzione tra informatica e competenza digitale, centralità di esecutori espliciti, passaggio dall'eseguire al programmare, copertura di diverse aree della disciplina oltre la programmazione e collegamento esplicito tra attività unplugged e ambienti digitali.

1 Introduzione

Negli scorsi mesi abbiamo realizzato *Sulle orme di Milù – Informatica*, un fascicolo di 48 pagine per alunne e alunni del primo triennio della scuola primaria pubblicato per Mondadori Education [4], e la relativa *Guida di Informatica per l'insegnante* di 120 pagine [3]. L'esperienza è diventata un'occasione per ragionare su come strutturare materiali rispettosi delle Indicazioni Nazionali 2025, ma al contempo utilizzabili in classe, con supporto anche per insegnanti non formati. Una versione derivata della guida [12], autorizzata per contratto con la casa editrice, è disponibile con licenza Creative Commons[1].

2 Contesto e motivazioni

Le Indicazioni Nazionali 2025 [15] introducono, già a partire dall'A.S. 2026/2027 e per la prima volta nel primo ciclo, competenze, obiettivi e conoscenze relativi all'Informatica (ancorché all'interno delle discipline *Matematica* e *Tecnologia*), distinguendo esplicitamente le competenze digitali da quelle informatiche. Questo passaggio è importante perché riconosce l'informatica come disciplina scientifica, non come semplice uso di strumenti digitali o come etichetta aggiornata per il "coding" [13]. Allo stesso tempo, genera una richiesta immediata di materiali: nuovi contenuti nei sussidiari, guide, percorsi di formazione e attività spendibili in classe.

L'informatica non è ben conosciuta da chi insegna nella scuola primaria. La ragione più ovvia è che finora l'Informatica non è stata insegnata nella scuola dell'obbligo (a differenza di quanto accade, ad esempio, per la matematica). Inoltre, la classe di laurea LM-85 bis in Scienze della Formazione Primaria prevede insegnamenti di tecnologie didattiche, ma non di Didattica dell'Informatica [14]. Le Indicazioni Nazionali 2025 rendono quindi necessario non solo aggiornare i sussidiari, ma anche ripensare la formazione iniziale e quella in servizio degli insegnanti. In assenza di questo passaggio, il rischio è che attività anche interessanti vengano proposte senza il background necessario per riconoscere

[1] La versione pubblica è disponibile all'indirizzo https://insegnareinformatica.github.io.

i concetti profondi che le sostengono. Il panorama dei materiali scolastici è frammentato; la letteratura recente e una nostra analisi preliminare informale, condotta in seguito alla richiesta editoriale, segnalano criticità ricorrenti: uso molto ampio del termine coding [13, 5], associato ad attività di altre discipline che però non esplicitano i collegamenti con i principi dell'informatica, metafore riduttive dell'algoritmo come "ricetta" [16], scarsa esplicitazione dell'interprete/esecutore [7], debole distinzione tra problema specifico, problema generale e algoritmo [6].

In questo contesto, l'aggiornamento in tempi strettissimi dei testi scolastici (finalizzato alla loro adozione già a partire da settembre 2026) rende particolarmente concreto il rischio di peggiorare la situazione: non basta aggiungere pagine di informatica se chi le usa non viene aiutato a capire cosa sta insegnando. Il rischio non è solo avere attività deboli o imprecise: senza una guida solida per i docenti, esercizi nati per introdurre concetti disciplinari rimangono semplici giochi di logica, lavoretti o attività di "coding" scollegate. In questo caso l'informatica non viene soltanto banalizzata, ma può produrre misconcezioni difficili da correggere successivamente: algoritmo come sequenza di azioni ambigue, informatica come noiosa ripetizione di azioni, confusione tra informazione, dati e la loro codifica.

Questo contesto ci ha motivato ad accompagnare le attività di informatica nel fascicolo per bambine e bambini con una guida in grado di sostenere gli insegnanti non specialisti. Il fascicolo doveva essere autonomo e adatto al contesto editoriale del sussidiario; la guida, invece, doveva spiegare il perché delle scelte, rendere espliciti i concetti disciplinari, suggerire domande da porre, errori da osservare, varianti possibili e collegamenti con le Indicazioni. La guida è quindi molto più di una raccolta di soluzioni: è un dispositivo di formazione, un testo che aiuta a leggere informaticamente attività che, in superficie, possono altrimenti assomigliare a pratiche già presenti nella scuola primaria.

3 Principi di progettazione del fascicolo e della guida

Il primo principio da noi seguito è stato *separare l'informatica, la competenza digitale e l'uso degli ambienti tecnologici*. Il fascicolo contiene anche attività con Code.org e Scratch, ma non identifica l'informatica né con l'uso del computer né con la programmazione: cerca invece di metterne in luce le *big ideas*, cioè i principi fondamentali, attraverso attività semplici e divertenti ma significative. La guida dedica spazio alla definizione di termini come *informazione*, *dato*, *istruzione*, *programma*, *algoritmo*, *linguaggio*, *esecutore* e *debug*, chiarendo in che senso parole condivise con la matematica e la tecnologia assumono in informatica significati più specifici [12, cap. 2].

Il secondo principio è stato *costruire attività attorno a esecutori espliciti*. In tutte le attività è esplicitato (o viene richiesto di esplicitare) il "manuale delle istruzioni" (informaticamente, l'*instruction set*) [12, cap. 2], che definisce quali istruzioni un robot, un compagno o una pedina sa "comprendere" ed eseguire.

Il terzo principio è stato *spostare progressivamente alunne e alunni dal ruolo di esecutori a quello di programmatori*. Molte attività che si sono diffuse in questi anni richiedono di seguire istruzioni; questa esperienza è utile, ma insufficiente se non porta a progettare istruzioni, a prevederne gli effetti, a osservarne l'esecuzione e a correggere gli errori. Nel fascicolo abbiamo usato due icone distinte per rendere visibile il ruolo svolto nelle attività: esecutore o programmatore [12, cap. 2].

Il quarto principio è stato quello di *coprire più aree della disciplina, non solo la programmazione*. Con le griglie colorate [12, cap. 5] non proponiamo soltanto di *eseguire* pixel art su griglie via via più grandi: prima si inferiscono le regole di una codifica da alcuni esempi, poi si decodificano immagini rappresentate per righe o coordinate e infine si produce un codice che altri dovranno interpretare. Il passaggio dagli esecutori ai programmatori consente di confrontare le rappresentazioni in termini di chiarezza e costo. Nelle "Luci di Natale" [12, cap. 5], invece, Alice e Bob rappresentano un insieme di messaggi mediante lettere luminose: alunne e alunni individuano i codici non ambigui, ne confrontano il costo e ne progettano uno nuovo. La "gruccia usata come bilancia" [12, cap. 6] permette di individuare il

sacchetto più pesante con azioni limitate; "Il barista ordinato" [12, cap. 6] richiede di ordinare cannucce usando solo confronti e scambi [11, 10, 9]. Qui ci si concentra su invenzione e descrizione di strategie, correttezza, generalizzazione e costo, più che sul nome di algoritmi noti.

Il quinto principio è stato quello di *progettare un ponte tra l'unplugged e il plugged*. Molte attività iniziano senza computer: movimento, oggetti, linguaggio naturale, carte, griglie. Questo rende accessibili concetti profondi senza anticipare le difficoltà tecniche proprie di un ambiente di programmazione [1]. Tuttavia, l'unplugged da solo non garantisce che la classe colga il contenuto informatico; serve una discussione esplicita e, quando opportuno, un passaggio verso esecutori automatici [2, 8]. Per questo abbiamo usato Code.org [12, cap. 7] come ambiente guidato per la sequenza, la ripetizione e la selezione, e Scratch [12, cap. 8] come ambiente più aperto per storie, dialoghi, cartoline e piccoli progetti creativi, in una prospettiva costruzionista e di apprendimento creativo [17, 18]. I simulatori [12, cap. 6] per le attività della gruccia e delle cannucce conservano invece il vincolo concettuale delle attività fisiche, rendendo possibili prove ripetute e il confronto tra strategie, oltre all'accessibilità. Questa scelta progettuale è inoltre coerente con le traiettorie di sviluppo cognitivo del bambino: le esperienze concrete, manipolative e corporee costituiscono infatti la base su cui costruire progressivamente rappresentazioni simboliche e astratte.

4 Aprire la guida

Un elemento importante dell'esperienza è stato negoziare con la casa editrice, accanto alla pubblicazione editoriale, la possibilità di mantenere i diritti necessari per pubblicare una versione derivata [12] della guida con licenza Creative Commons Attribuzione–Non commerciale–Condividi allo stesso modo 4.0. Mondadori ha accolto la nostra proposta ed è proprio da questo accordo che nasce la guida qui presentata. La licenza CC BY-NC-SA non rende il materiale libero in senso pieno, perché vieta l'uso commerciale; lo rende però vicino alla famiglia delle risorse educative aperte, intese come materiali accessibili, riutilizzabili, adattabili e redistribuibili [19]. È dunque materiale aperto a sufficienza da consentire pratiche che, in un prodotto editoriale chiuso, sarebbero molto più difficili.

La versione derivata è completamente autocontenuta e quindi utilizzabile anche da chi non ha adottato il sussidiario.

Inoltre, sebbene le attività proposte siano pensate per la scuola primaria, sono in realtà adatte a chiunque voglia approcciarsi per la prima volta ai principi fondamentali della scienza informatica. Slegare il suo destino editoriale da un sussidiario destinato esclusivamente al primo triennio della scuola primaria può quindi favorirne la diffusione presso un pubblico più vario.

L'apertura non è però solo diffusione, ma anche possibilità di confronto e di crescita. Una guida aperta, infatti, consente a più insegnanti, formatori e ricercatori di provarla, segnalare errori, proporre varianti, documentare le sperimentazioni e costruire un ciclo di ricerca-azione. Alcune attività proposte derivano da esperienze già consolidate; altre richiedono sperimentazione e osservazione sistematica in classe. Rendere visibile e modificabile il materiale è quindi, oltre a una scelta di distribuzione, parte del progetto didattico: consente alla comunità di produrre conoscenza e materiali sull'insegnamento dell'informatica nella scuola. Questo è particolarmente importante per una disciplina che entra ora nei documenti curricolari del primo ciclo: senza comunità, esempi commentati e feedback dal campo, il divario tra indicazioni normative e pratiche reali rischia di restare molto ampio.

Non disponiamo ancora di dati sistematici sull'uso della guida da parte degli insegnanti non specialisti. Una prima valutazione dovrà considerare la comprensibilità, l'autonomia d'uso, gli adattamenti prodotti e le difficoltà incontrate.

5 Conclusioni

Il racconto di questa esperienza mostra che l'editoria scolastica può diventare un luogo di mediazione produttiva tra vincoli commerciali, nuove richieste curricolari e rigore disciplinare. La nostra risposta è stata progettare materiali che aiutino docenti e classi a riconoscere l'informatica: dati e rappresentazioni, algoritmi rispetto a esecutori espliciti, programmi, debug, passaggio dall'esecuzione alla progettazione.

Il nodo più delicato resta la formazione degli insegnanti. Se il primo ciclo deve davvero accogliere l'informatica come disciplina, servono non solo materiali per gli studenti, ma anche guide, esempi commentati, comunità di pratica e percorsi universitari aggiornati. Aprire la guida è il nostro modo di prolungare il lavoro oltre il prodotto editoriale: una base comune per la formazione, il feedback, la sperimentazione e la ricerca-azione.

Riferimenti bibliografici

[1] Tim Bell and Michael Lodi. Constructing computational thinking without using computers. *Constructivist Foundations*, 14(3):342–351, 2019. https://constructivist.info/14/3/342.bell.

[2] Tim Bell and Jan Vahrenhold. CS unplugged: How is it used, and does it work? In Hans-Joachim Böckenhauer, Dennis Komm, and Walter Unger, editors, *Adventures Between Lower Bounds and Higher Altitudes*, volume 11011 of *Lecture Notes in Computer Science*, pages 497–521. Springer, Cham, 2018. https://doi.org/10.1007/978-3-319-98355-4_29.

[3] Giorgia Bissoli, Agnese Del Zozzo, Michael Lodi, and Alberto Montresor. Sulle orme di Milù – Guida di Informatica per l'insegnante. Guida docente, Mondadori Education, Milano, ISBN 9791220419833, 2026. https://www.mondadorieducation.it/catalogo/sulle-orme-di-milu-0074279/.

[4] Giorgia Bissoli, Agnese Del Zozzo, Michael Lodi, and Alberto Montresor. *Sulle orme di Milù – Informatica*. Mondadori Education, Milano, 2026. https://www.mondadorieducation.it/catalogo/sulle-orme-di-milu-0074279/.

[5] Sara Capecchi, Michael Lodi, Violetta Lonati, and Anna Morpurgo. L'informatica negli attuali libri di testo della scuola secondaria di primo grado, alla luce delle Indicazioni Nazionali 2025. In Giorgio Bolondi, Bruno D'Amore, and Federica Ferretti, editors, *La Didattica della Matematica con e per la scuola*, pages 157–158, San Lazzaro di Savena, 2025. Bonomo Editore. Incontri con la Matematica no. 39, 2025.

[6] Marta Crippa, Laura Branchetti, Luca Lamanna, and Violetta Lonati. Conoscenza procedurale, algoritmi e problemi: spunti di riflessione da una prospettiva interdisciplinare tra matematica e informatica. In Giorgio Bolondi, Bruno D'Amore, and Federica Ferretti, editors, *La Didattica della Matematica con e per la scuola*, pages 111–112, San Lazzaro di Savena, 2025. Bonomo Editore. Incontri con la Matematica no. 39, 2025.

[7] Marta Crippa, Violetta Lonati, Luca Lamanna, and Laura Branchetti. Educare al pensiero algoritmico: riflessioni e proposte per il primo ciclo. In *Atti del Terzo Convegno Italiano sulla Didattica dell'Informatica (ITADINFO 2025)*, pages 19–27, Milano, 2025. Ledizioni. https://www.ledizioni.it/stag/wp-content/uploads/2025/10/ITADINFO_def_web.pdf.

[8] Felienne Hermans and Efthimia Aivaloglou. To Scratch or Not to Scratch? A Controlled Experiment Comparing Plugged First and Unplugged First Programming Lessons. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, pages 49–56, New York, NY, USA, 2017. ACM. https://doi.org/10.1145/3137065.3137072.

[9] Michael Lodi. Il barista ordinato. Simulatore open source HTML/JavaScript, 2026. AGPL-3.0-or-later. https://lodi.ml/cannucce/.

[10] Michael Lodi. Trova il più pesante. Simulatore open source HTML/JavaScript, 2026. AGPL-3.0-or-later. https://lodi.ml/gruccia/.

[11] Michael Lodi. Hangers and straws: Designing low-cost unplugged activities for primary-school algorithms. In *Creative Mathematical Sciences Communication (CMSC 2026)*, volume 16951 of *Lecture Notes in Com-*

puter Science. Springer, 2027. https://doi.org/10.1007/978-3-032-38887-2_4. Accettato, in corso di stampa.

[12] Michael Lodi, Agnese Del Zozzo, Alberto Montresor, and Giorgia Bissoli. Insegnare Informatica - Una guida per docenti del primo ciclo, 2026. https://insegnareinformatica.github.io/.

[13] Michael Lodi and Simone Martini. Coding, pensiero computazionale e informatica: quale evoluzione possibile? In *Atti del Secondo Convegno Italiano sulla Didattica dell'Informatica (ITADINFO 2024)*, pages 20–24, Genova, 2024. Genova University Press. https://gup.unige.it/sites/gup.unige.it/files/2024-10/ITADINFO_2024_ebook.pdf.

[14] Ministero dell'Istruzione, dell'Università e della Ricerca. Decreto 10 settembre 2010, n. 249. Regolamento concernente la definizione della disciplina dei requisiti e delle modalità della formazione iniziale degli insegnanti, 2011. Gazzetta Ufficiale, Serie Generale n. 24, 31 gennaio 2011, Supplemento Ordinario n. 23. Classe LM-85 bis, Scienze della formazione primaria. https://www.gazzettaufficiale.it/atto/vediMenuHTML?atto.codiceRedazionale=011G0014&atto.dataPubblicazioneGazzetta=2011-01-31&tipoSerie=serie_generale&tipoVigenza=originario.

[15] Ministero dell'Istruzione e del Merito. Decreto 9 dicembre 2025, n. 221. Regolamento recante Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione, 2026. Gazzetta Ufficiale, Serie Generale n. 21, 27 gennaio 2026. https://www.mim.gov.it/documents/20182/10554370/curricolo_web.pdf/.

[16] Mattia Monga, Renzo Davoli, Luca Lamanna, and Luca Forlizzi. Il concetto di algoritmo nei libri di testo: spunti per una riflessione. In Giorgio Bolondi, Bruno D'Amore, and Federica Ferretti, editors, *La Didattica della Matematica con e per la scuola*, pages 177–178, San Lazzaro di Savena, 2025. Bonomo Editore. Incontri con la Matematica no. 39, 2025.

[17] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980. https://www.media.mit.edu/publications/mindstorms/.

[18] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67, 2009. https://doi.org/10.1145/1592761.1592779.

[19] UNESCO. Recommendation on open educational resources (OER), 2019. https://www.unesco.org/en/legal-affairs/recommendation-open-educational-resources-oer.

La crittografia? È un gioco!

Arianna Boldi[1], Sara Capecchi[2], Iary Davidson[2] and Marco Viola[1]
[1] Dipartimento di Psicologia - Università degli Studi di Torino
[2] Dipartimento di Informatica - Università degli Studi di Torino
[arianna.boldi, sara.capecchi, iary.davidson, marco.viola]@unito.it

Abstract

Il contributo presenta «La crittografia? È un gioco!», linea del Progetto Diderot di Fondazione CRT rivolta alla scuola primaria e alla secondaria di primo grado e realizzata negli in circa 1.000 classi del Piemonte e della Valle d'Aosta. La crittografia viene utilizzata come contesto per introdurre alcuni concetti fondanti dell'informatica.

1. Introduzione

Il Progetto Diderot, promosso da Fondazione CRT, propone gratuitamente alle scuole del Piemonte e della Valle d'Aosta percorsi che approfondiscono le discipline curricolari attraverso metodologie innovative e avvicinano studentesse e studenti ad ambiti non sempre presenti nel curricolo ordinario. L'iniziativa coinvolge scuole dell'infanzia, primarie e secondarie di primo e secondo grado e, nelle edizioni più recenti, anche i CPIA.

In questo quadro si colloca «La crittografia? È un gioco!», rivolto alle classi terze, quarte e quinte della primaria e alla secondaria di primo grado e portato, negli ultimi due anni, in circa 1.000 classi. La crittografia è stata scelta non soltanto per il legame con la sicurezza, ma perché rende osservabili attraverso problemi concreti concetti di base dell'informatica: rappresentazione simbolica, codifica, applicazione di procedure, ruolo della chiave, automazione e rapporto tra risorse disponibili e sicurezza di un sistema.

2. Metodi e contenuti

Il percorso adotta una metodologia laboratoriale e attiva basata su un approccio unplugged. Giochi, schede, materiali manipolabili e simulazioni permettono di lavorare sui concetti senza richiedere l'uso di dispositivi digitali. Le attività alternano una fase esperienziale, in cui si costruiscono, applicano o violano codici, a momenti di discussione guidata che esplicitano i concetti informatici sottostanti. L'attenzione viene così spostata dall'uso della tecnologia alla comprensione dei principi che ne rendono possibile il funzionamento.

Il nucleo concettuale è modulato in funzione dell'età. Nelle classi terze e quarte della primaria prevalgono attività visive e manipolative, associazioni tra simboli e lettere e semplici cifrari; nella quinta primaria e nelle prime classi della secondaria aumentano formalizzazione, descrizione delle regole e prime simulazioni di crittoanalisi; nelle seconde e terze della secondaria si approfondiscono attacchi a forza bruta, intercettazione di messaggi, limiti dei sistemi e ruolo della capacità di calcolo. Il percorso utilizza alcuni cifrari storici (es. Cesare e di Polibio) e giochi ispirati a macchine cifranti.

2.1 Collegamento con le Indicazioni nazionali

La proposta si collega direttamente alle nuove Indicazioni nazionali per il primo ciclo.

La crittografia offre un contesto unitario per introdurre alcuni concetti fondamentali dell'informatica: rappresentazione dell'informazione, codifica dei dati, algoritmi. Cifrare un messaggio richiede di distinguere l'informazione dalla sua rappresentazione, scegliere una codifica e applicare una procedura precisa; decifrarlo richiede di ricostruire tale procedura o, quando la chiave non è nota, progettare una strategia di ricerca. Le Indicazioni richiamano inoltre direttamente i «codici segreti» per la riservatezza nella primaria e i cifrari e gli attacchi informatici nella

secondaria. Il laboratorio rende quindi operativi più obiettivi curricolari all'interno della stessa situazione problematica.

2.2 La prospettiva storica come strumento di riflessione

Un secondo elemento strutturante è il parallelo con la storia della crittografia. Le attività mostrano che la sicurezza di un sistema non è assoluta, ma dipende dalle conoscenze e dalle risorse di chi tenta di violarlo. Un passaggio particolarmente efficace consiste nel far sperimentare alle classi che oggi possono decifrare con strategie semplici un cifrario attribuito a Cesare e impiegato per proteggere comunicazioni militari. Diventa così evidente come un sistema adeguato in un certo contesto storico possa diventare vulnerabile quando cambiano capacità di analisi e strumenti disponibili.

La ricostruzione storica collega inoltre l'evoluzione dei cifrari alla progressiva diffusione e sofisticazione di lettura, scrittura, calcolo e analisi matematica: aumentando le possibilità di individuare regolarità e provare sistematicamente alternative, anche le tecniche di cifratura devono diventare più complesse. Con i computer questo passaggio è ancora più evidente: un computer può eseguire molto rapidamente enormi quantità di calcoli e tentativi.

3. Risultati preliminari

L'analisi preliminare si basa su 2.139 questionari somministrati prima delle attività e 2.342 questionari somministrati dopo. Il campione presenta una distribuzione di genere sostanzialmente equilibrata. Per ordine scolastico, il 44,1% dei questionari proviene dalla primaria e il 55,9% dalla secondaria di primo grado; la lieve prevalenza della secondaria è coerente con la maggiore facilità di compilazione autonoma nelle età più avanzate.

I dati descrittivi restituiscono un quadro di elevato gradimento. Le valutazioni di difficoltà (media 2,22 su 3), ritmo (1,99) e durata (2,24) suggeriscono attività generalmente percepite come accessibili e ben calibrate. Anche gli item relativi a collaborazione, aiuto ricevuto, interesse e apprendimento percepito mostrano risposte fortemente positive: studentesse e studenti riferiscono di essersi sentiti supportati, di aver collaborato efficacemente e di aver appreso qualcosa di nuovo. La dimensione collaborativa e l'interattività emergono tra gli aspetti più apprezzati.

I risultati sono preliminari e descrivono soprattutto la ricezione del percorso e la percezione dell'esperienza, senza consentire da soli inferenze sull'apprendimento disciplinare. Forniscono tuttavia prime indicazioni sulla sostenibilità di un laboratorio unplugged di informatica proposto su larga scala e differenziato per grado scolastico, e costituiscono la base per successive analisi sugli apprendimenti.

Codey Rocky va in favola - Un percorso di narrazione computazionale inclusiva secondo il framework UDL nella scuola primaria

Anna Levrero
IC VOLTRI 1
anna.levrero@gmail.com

Abstract

Questo lavoro illustra un'attività di narrazione computazionale realizzata in una classe terza primaria, dove la favola "La volpe e l'uva" è diventata il terreno per una modellizzazione logica con il robot Codey Rocky. L'esperienza ha permesso di far evolvere le competenze pregresse degli alunni sulla programmazione a blocchi (sequenza e iterazione), maturate con Scratch, verso una gestione più complessa dei sensori e della logica condizionale (*se... allora... altrimenti*). Il fulcro disciplinare del percorso risiede proprio nella traduzione delle scene della favola in istruzioni di programma: una scrittura collaborativa che ha trasformato il testo narrativo in una sequenza logica non ambigua. Seguendo il framework *Universal Design for Learning* (UDL) applicato alla robotica, la multimodalità del robot ha reso tangibili concetti altrimenti astratti, coinvolgendo l'intera classe in una negoziazione continua della struttura della storia. Ampio spazio è stato dedicato al debugging, inteso come processo analitico del software necessario per allineare il comportamento del robot alla trama della favola. Il progetto si inserisce nelle Nuove Indicazioni Nazionali 2026, proponendo un'idea di informatica rigorosa e inclusiva, capace di dialogare con le altre discipline curricolari.

1 Introduzione

L'esperienza presentata nasce dalla volontà di valorizzare le risorse tecnologiche dell'Istituto, nello specifico i robot Codey Rocky, inserendole in una cornice didattica rigorosa e interdisciplinare, all'interno della progettazione curricolare di una classe terza primaria. Il catalizzatore del progetto è stato il percorso formativo proposto dall'Equipe Formativa Territoriale (EFT) Liguria, che ha spinto a riflettere su come integrare la robotica nella programmazione curricolare di una classe terza, evitando l'approccio dell'attività isolata o puramente ludica.

La scelta è ricaduta sulla favola "La volpe e l'uva" di Esopo, utilizzata come cornice narrativa per un percorso di trasposizione multimodale. L'idea di fondo è stata quella di guidare gli alunni in un processo di astrazione crescente: partendo dalla scomposizione narrativa sul quaderno, passando per la rielaborazione digitale in forma di fumetto su Book Creator, fino ad approdare alla drammatizzazione logica tramite la programmazione del robot.

In questo scenario, il framework *Universal Design for Learning* (UDL) ha rappresentato la bussola metodologica essenziale. Come evidenziato da Rita Cersosimo nella sua ricerca [1], la robotica educativa permette di progettare ambienti di apprendimento in cui la tecnologia agisce come mediatore tra il concetto e l'esperienza, offrendo "molteplici mezzi di rappresentazione, azione ed espressione". Nel nostro percorso, il robot ha reso tangibili concetti informatici altrimenti astratti, agendo come facilitatore cognitivo per ridurre le barriere d'accesso alla logica formale e favorire la partecipazione attiva attraverso il feedback multimodale di sensori e attuatori.

L'intervento si colloca nel quadro dei traguardi previsti dalle Nuove Indicazioni Nazionali 2026 (DM 221 del 9/12/25) [2], mirando in particolare a presidiare la capacità di modellizzare problemi, la gestione di un esecutore deterministico attraverso la logica condizionale e la pratica del *debugging* come processo riflessivo sul software. Il percorso si pone l'obiettivo di presidiare i traguardi previsti dalle Nuove Indicazioni Nazionali 2026, trasformando la narrazione in

un'occasione per esplorare la logica condizionale (sensori), la gestione dell'esecutore deterministico e il *debugging* come processo analitico del software.

2 Il percorso didattico: dalla matita al codice

Il progetto è stato strutturato come un itinerario di modellizzazione volto a trasformare un testo narrativo in un modello computazionale eseguibile. Tale processo si è articolato in tre fasi distinte e complementari, pensate per accompagnare gli alunni verso un'astrazione progressiva.

2.1 Fase analogica: scomposizione e sequenze sul quaderno

Il lavoro è iniziato nel secondo quadrimestre con un approccio prettamente linguistico e grafico. Ogni alunno ha lavorato individualmente sul testo della favola, operando una prima scomposizione in sequenze narrative temporali.

Sul quaderno, le alunne e gli alunni hanno illustrato i momenti salienti della storia (l'avvistamento dell'uva, i tentativi di salto, la rinuncia), identificando così le unità minime d'azione. Questa fase è stata fondamentale per isolare gli eventi discreti che avrebbero poi costituito l'ossatura logica del programma informatico.

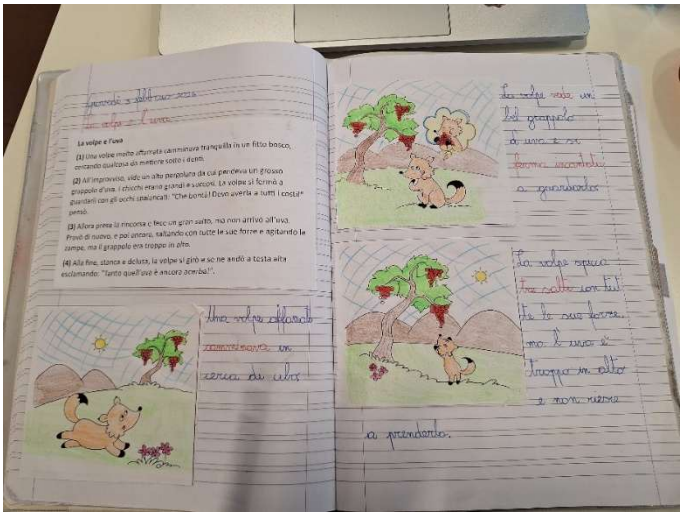


Figura 1: Scomposizione della favola in sequenze narrative sul quaderno.

2.2 Fase digitale: la sintesi del fumetto su Book Creator

Per favorire il passaggio verso una rappresentazione più strutturata, le sequenze sono state rielaborate digitalmente su Book Creator. Questo strumento, che la classe utilizza settimanalmente come diario di bordo multimodale per elaborare immagini, testi e rappresentazioni di alcuni dei momenti salienti della vita di classe, è stata la naturale conseguenza del flusso di lavoro. All'interno del proprio e-book personale è stata ricostruita la favola utilizzando il linguaggio del fumetto.

Questa scelta ha rappresentato la stesura dello storyboard: la necessità di sintesi imposta dalle vignette ha obbligato a una sintesi critica delle informazioni, definendo con precisione gli input e gli output di ogni scena. Ogni vignetta doveva rappresentare un'azione o un dialogo essenziale; questo esercizio è stato il vero ponte verso la programmazione, poiché ha richiesto di definire una specifica precisa per ogni scena, eliminando le ambiguità descrittive del testo letterario e preparando il terreno per la successiva implementazione algoritmica.



Figura 2: Trasposizione delle sequenze nel linguaggio del fumetto tramite Book Creator.

2.3 Fase collaborativa: dalla programmazione a blocchi alla gestione dei sensori

La fase conclusiva ha visto la trasposizione delle specifiche definite su Book Creator in codice eseguibile su Codey Rocky. L'attività ha permesso di capitalizzare i prerequisiti maturati nel ciclo precedente (classe seconda), durante il quale le alunne e gli alunni avevano lavorato settimanalmente con Scratch acquisendo dimestichezza con le strutture di sequenza, ripetizione e la gestione di movimento e aspetto.

Il salto di qualità disciplinare, in linea con le Nuove Indicazioni Nazionali 2026, è consistito nel passaggio da una programmazione lineare a una basata sull'interazione con l'ambiente attraverso i sensori. Per rappresentare il tentativo della volpe di raggiungere l'uva, la classe ha dovuto implementare una *logica condizionale* (se... allora): il robot è stato istruito per attivare l'animazione del salto solo in presenza di un input esterno (rilevamento del colore viola).

Un momento centrale di analisi critica è stato dedicato al debugging specifico per il sensore di colore. Gli alunni hanno riscontrato che il robot faticava a riconoscere le sfumature non omogenee dei disegni fatti a mano; questa difficoltà è stata trasformata in opportunità didattica per comprendere come un sensore elabori dati numerici e richieda input standardizzati. Si è scelto quindi di utilizzare i cartoncini di colore omogeneo forniti in dotazione al robot per garantire il determinismo dell'esecuzione. Inoltre, per responsabilizzare gli alunni sulla gestione dello strumento, è stata introdotta la programmazione del tasto B come verifica della carica residua della batteria, un esercizio di monitoraggio delle risorse del sistema.

In questo scenario, le abilità tecniche acquisite nel tempo (uso dei blocchi e sintassi) sono state messe in campo per articolare competenze computazionali più vaste: la capacità di modellizzare un problema complesso, di gestire l'incertezza del dato fisico (sensore) e di collaborare alla risoluzione di bug logici, trasformando il robot in un oggetto sociale di riflessione metalinguistica.

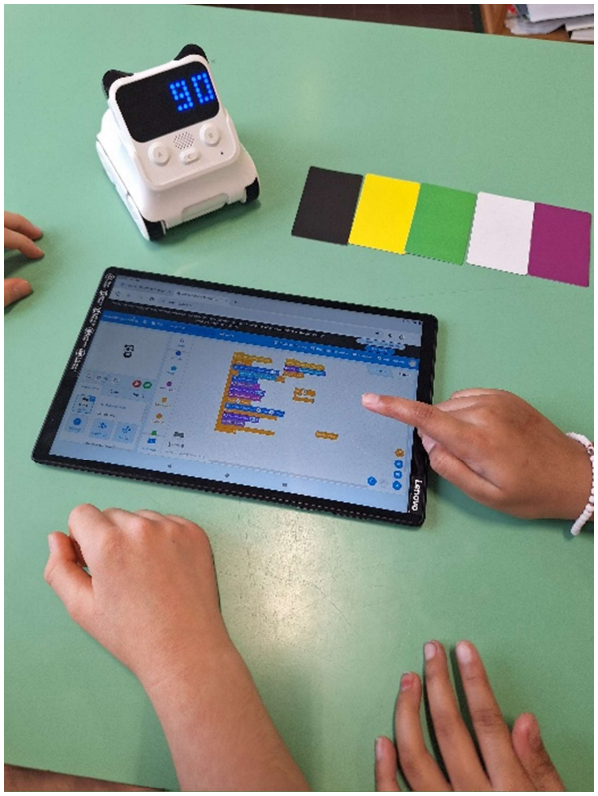


Figura 3: Programmazione del robot Codey Rocky per la drammatizzazione della storia.

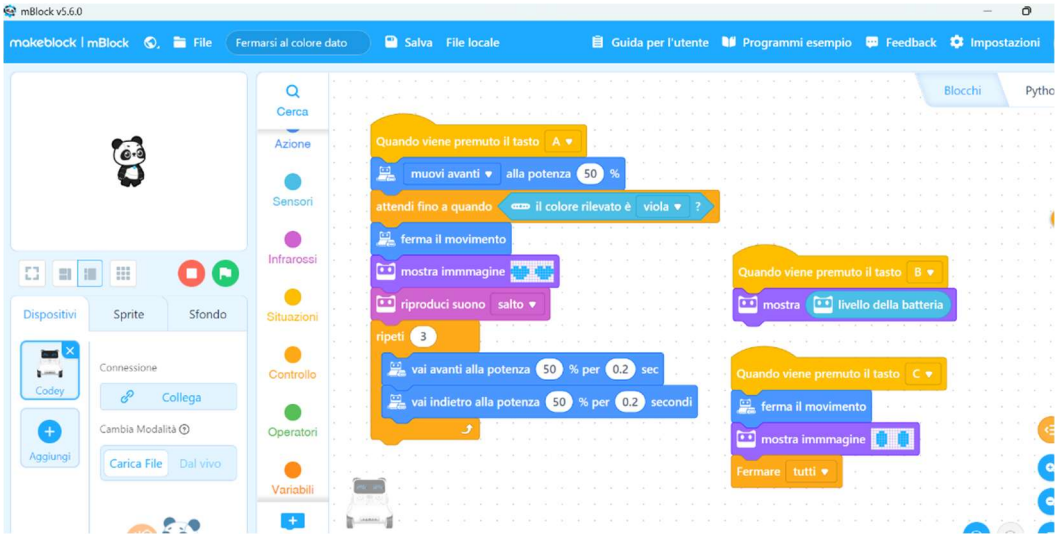


Figura 4: Schermata dell'algoritmo completo in ambiente mBlock.

Conclusioni

Il percorso svolto ha confermato come la narrazione computazionale sia un volano per l'apprendimento disciplinare dell'informatica, a patto che la tecnologia sia subordinata a una solida struttura logica. Inizialmente, l'effetto novità introdotto dal robot Codey Rocky ha generato un forte coinvolgimento emotivo; tuttavia, questo entusiasmo è stato rapidamente canalizzato verso un'analisi rigorosa del sistema-macchina.

Gli alunni hanno compreso che il robot non agisce per magia, ma risponde esclusivamente a istruzioni formali, trasformando la curiosità iniziale in competenza tecnica. L'adozione di questo approccio non è priva di sfide per il docente. Come sottolineato da Cersosimo [1], l'insegnante che decide di sperimentare con la robotica e il coding si scontra inevitabilmente con momenti di difficoltà e incertezza, legati alla gestione dell'imprevisto tecnologico e alla necessità di abbandonare il ruolo di unico detentore del sapere. Accettare questa vulnerabilità è però la condizione necessaria per attivare una vera comunità di ricerca in classe, dove il debugging diventa una pratica collettiva di analisi del software.

Dal punto di vista della disciplina informatica, il successo dell'attività non risiede nell'uso del robot in sé, ma nell'avvenuto passaggio dalla programmazione lineare a quella reattiva. Gli alunni hanno maturato i traguardi previsti dalle Nuove Indicazioni Nazionali 2026, dimostrando di saper gestire la complessità dei sensori e la logica degli eventi. La capacità di correggere il codice per adattarlo alla risposta fisica della macchina testimonia lo sviluppo di un pensiero computazionale critico, capace di distinguere tra l'intenzione narrativa e l'esecuzione deterministica dell'algoritmo.

In conclusione, la robotica ha agito solo da catalizzatore, mentre il vero cuore del percorso è stata l'acquisizione di un linguaggio logico rigoroso, inclusivo e consapevole.

Bibliografia

[1] Cersosimo, Rita. Robotica educativa e inclusione. Loescher Editore, 2025

[2] Ministero dell'Istruzione e del Merito (MIM). Nuove Indicazioni Nazionali per il Curricolo della Scuola dell'Infanzia e del Primo Ciclo d'Istruzione 2025. Adottate con D.M. n. 221 del 9 dicembre 2025, pubblicato in G.U. n. 21 del 27 gennaio 2026.

Imparare a programmare senza schermi: evidenze sperimentali di un intervento di coding tangibile in età prescolare

Chiara Montuori[1], Gabriele Pozzan[2], Costanza Padova, Lucia Ronconi[1], Tullio Vardanega[2], Barbara Arfé[1]
[1] Dipartimento di Psicologia dello Sviluppo e della Socializzazione, Università di Padova
chiara.montuori@unipd.it
[2] Dipartimento di Matematica, Università di Padova
gabriele.pozzan@unipd.it

Abstract

L'introduzione dell'informatica nei primi anni di scolarità ha assunto un ruolo sempre più rilevante, con particolare attenzione all'apprendimento dei concetti di base della programmazione. Il presente studio analizza l'efficacia di un intervento didattico, basato su attività propedeutiche alla programmazione su carta (unplugged) e attività di robotica educativa, nella scuola dell'infanzia.
Attraverso un disegno sperimentale randomizzato a cluster, sono stati coinvolti 47 bambini e bambine di 4–5 anni suddivisi in gruppo sperimentale e gruppo di controllo. L'intervento, della durata di sette settimane, ha proposto attività finalizzate all'apprendimento di sequenze di istruzioni e alla loro esecuzione. Le competenze sono state valutate al pretest (T1, prima dell'intervento) e al post-test (T2, dopo l'intervento) mediante quattro compiti di programmazione e test neuropsicologici standardizzati: un compito di pianificazione (Torre di Londra), un compito di inibizione della risposta (NEPSY-II) e un compito di abilità visuo-spaziali (PMA).
I risultati mostrano effetti positivi significativi del programma didattico sulle competenze di programmazione e sulle capacità cognitive dei bambini, in particolare quelle viso-spaziali. Nel complesso, lo studio fornisce evidenze empiriche sull'efficacia di percorsi didattici per l'introduzione dell'informatica nella scuola dell'infanzia, offrendo indicazioni utili per la progettazione di attività coerenti con lo sviluppo cognitivo dei bambini.

1 Introduzione

Negli ultimi anni, la didattica dell'informatica ha assunto un ruolo sempre più centrale nei sistemi educativi, anche in risposta alla crescente necessità di sviluppare competenze digitali e capacità di problem solving nei cittadini. Nonostante ciò, una quota significativa di studenti europei non raggiunge ancora livelli adeguati in tali competenze di base [1]. In questo contesto, si è progressivamente affermata l'esigenza di riconoscere l'informatica come ambito disciplinare autonomo, con contenuti, metodi e obiettivi specifici, da introdurre sin dai primi anni di scolarizzazione.

In ambito educativo, l'introduzione dell'informatica in età prescolare si concretizza prevalentemente attraverso attività di coding tangibile, intese come esperienze didattiche che introducono i bambini ai concetti di algoritmo, sequenza di istruzioni, esecuzione e controllo di procedure. L'introduzione precoce della programmazione risulta particolarmente rilevante nella scuola dell'infanzia, dove è possibile proporre attività adeguate allo sviluppo cognitivo dei bambini attraverso approcci concreti e manipolativi. La letteratura individua tre principali modalità didattiche: (1) il coding unplugged, basato su attività senza dispositivi digitali [2]; (2) la robotica educativa, che utilizza robot programmabili in contesti fisici [3]; e (3) il coding in ambienti digitali, sviluppato su computer o tablet [4]. Questi approcci presentano caratteristiche differenti: mentre le attività unplugged e di robotica educativa favoriscono esperienze concrete e progressive di costruzione di sequenze di istruzioni, il coding in ambiente digitale richiede livelli più elevati di astrazione. Per questo motivo, i primi risultano particolarmente adatti all'introduzione dell'informatica in età prescolare.

Nonostante la crescente diffusione di queste pratiche, le evidenze empiriche sulla loro efficacia in età prescolare sono ancora limitate. La maggior parte degli studi si concentra su fasce d'età successive, mentre pochi lavori sperimentali analizzano in modo sistematico l'apprendimento dei concetti di programmazione nei bambini più piccoli, in particolare attraverso interventi didattici che integrino più approcci (e.g., unplugged e robotica educativa) [5,6]. Questa lacuna limita la possibilità di definire modelli didattici evidence-based per l'introduzione precoce dell'informatica. Inoltre, resta ancora poco indagato il rapporto tra apprendimento della programmazione e sviluppo di abilità cognitive.

Un ulteriore tema rilevante per la didattica dell'informatica riguarda il trasferimento degli apprendimenti. In ambito educativo, la letteratura distingue tra *near transfer*, relativo all'acquisizione e al consolidamento delle competenze direttamente connesse alla programmazione (ad esempio la costruzione e l'esecuzione di sequenze di istruzioni), e *far transfer*, che riguarda effetti su abilità cognitive più generali [7,8]. Metanalisi precedenti hanno mostrato che attività di coding, soprattutto in contesti strutturati, possono contribuire allo sviluppo di funzioni esecutive quali la pianificazione e l'inibizione della risposta [9,10], sebbene tali evidenze risultino ancora limitate e non sempre coerenti nella scuola dell'infanzia [11].

In particolare, le abilità visuo-spaziali rivestono un ruolo significativo nell'apprendimento della programmazione, poiché supportano la rappresentazione mentale dei percorsi e la comprensione delle sequenze di azioni. Al contempo, attività di programmazione in contesti concreti e manipolativi contribuiscono al loro sviluppo [12], suggerendo una possibile relazione bidirezionale tra apprendimento dell'informatica e sviluppo cognitivo.

Alla luce di queste considerazioni, emerge la necessità di studi sperimentali che analizzino l'efficacia di percorsi di introduzione all'informatica nella scuola dell'infanzia. Il presente lavoro si colloca in tale ambito, investigando gli effetti di un intervento didattico integrato, basato su coding unplugged e robotica educativa, sull'apprendimento dei concetti base della programmazione in bambini di 4–5 anni, nonché i possibili effetti di trasferimento su specifiche abilità cognitive (inibizione della risposta, pianificazione e abilità visuo-spaziali).

Lo studio si propone di rispondere alle seguenti domande di ricerca:

- **Effetti di *near-transfer* sulle abilità di coding:** un intervento di coding tangibile è efficace nel promuovere l'apprendimento delle competenze di programmazione di base in bambini di 4–5 anni?
- **Effetti di *far-transfer* sulle abilità cognitive:** i benefici derivanti dall'intervento si estendono anche ad altre capacità cognitive nei bambini di 4–5 anni, in particolare all'inibizione della risposta, alla pianificazione e alle abilità visuo-spaziali?

2 Metodo

Lo studio ha adottato un disegno sperimentale di tipo cluster-randomized controlled trial (secondo le linee guida CONSORT, [13]), articolato in una rilevazione pre-intervento (T1), intervento e rilevazione post-intervento (T2). Il gruppo sperimentale ha partecipato all'intervento di coding, mentre il gruppo di controllo (lista d'attesa) ha continuato con le consuete attività didattiche tra T1 e T2, ricevendo l'intervento solo successivamente.

Una terza rilevazione (T3) è stata effettuata su entrambi i gruppi: per il gruppo di controllo ha rappresentato il post-test dopo l'intervento, mentre per il gruppo sperimentale ha costituito una misura di follow-up per valutare la stabilità degli effetti nel tempo.

Quattro sezioni di scuola dell'infanzia sono state assegnate casualmente al gruppo sperimentale o al gruppo di controllo. Dopo il pre-test, il gruppo sperimentale ha svolto attività di introduzione all'informatica due volte a settimana per sette settimane. Il gruppo di controllo, nello stesso periodo, ha seguito le normali attività scolastiche, partecipando all'intervento di programmazione solo dopo il post-test. Questo disegno ha consentito di garantire anche al gruppo di controllo l'accesso a esperienze didattiche potenzialmente efficaci. Per il gruppo sperimentale, il follow-up (T3) è stato effettuato due mesi dopo il post-test.

2.1 Partecipanti

Hanno partecipato allo studio 47 bambini di cinque anni, frequentanti l'ultimo anno della scuola dell'infanzia di una scuola nel Nord Italia, senza precedenti esperienze di coding. Le classi sono state assegnate casualmente al gruppo sperimentale (25 bambini,14 femmine e 11 maschi) o al gruppo di controllo in lista d'attesa (22 bambini, 10 femmine e 12 maschi). Il gruppo sperimentale ha svolto attività di introduzione all'informatica subito dopo il pre-test, mentre il gruppo di controllo ha seguito le ordinarie attività scolastiche, partecipando all'intervento solo dopo il post-test. I due gruppi risultavano omogenei per età (media 64 mesi), status socio-economico, familiarità tecnologica e distribuzione di genere.

2.2 Procedura e materiali

È stato progettato un intervento integrato di coding unplugged e robotica educativa della durata di sette settimane, con due sessioni settimanali di 60 minuti (per un totale di 14 incontri). Per garantire uniformità nell'esperienza formativa, sono state monitorate le presenze e predisposte eventuali attività di recupero. Il coinvolgimento dei bambini è stato costantemente sostenuto attraverso osservazioni e interazioni guidate durante le attività.
La scelta di combinare coding unplugged e robotica educativa risponde a motivazioni di carattere didattico. Sebbene la robotica educativa rappresenti uno strumento motivante e concreto, essa comporta anche una certa complessità cognitiva, soprattutto per bambini senza esperienza pregressa, richiedendo competenze sia concettuali sia operative. Per questo motivo, è stato adottato un approccio graduale: inizialmente sono state proposte attività propedeutiche e di coding unplugged per introdurre i concetti fondamentali della programmazione e del pensiero algoritmico: implementare la soluzione di un problema con una sequenza precisa di istruzioni ben definite che vengono eseguite esattamente da un agente meccanico, evitando il carico cognitivo legato all'uso di strumenti digitali. Successivamente, sono stati introdotti i robot come estensione delle attività precedenti, per consolidare e generalizzare gli apprendimenti e per sperimentare le capacità algoritmiche sviluppate fino a quel punto con un agente veramente "meccanico".
Ulteriori criticità nell'uso della robotica riguardano, ad esempio, la difficoltà dei bambini nel cambiare prospettiva spaziale (assumere il punto di vista del robot), abilità non pienamente sviluppata in età prescolare e non necessariamente facilitata dall'interazione con i robot [14]. Anche studi precedenti evidenziano difficoltà operative e cognitive nell'uso di robot educativi da parte di bambini piccoli, oltre a limiti legati ai costi e all'accessibilità degli strumenti [15]. Per tali ragioni, il coding unplugged rappresenta una modalità particolarmente efficace per introdurre precocemente la programmazione, favorendo l'acquisizione di abilità di base (come orientamento spaziale e distinzione destra/sinistra) in contesti concreti e accessibili.

2.2.1 Struttura del percorso didattico

L'intervento è stato progettato per valorizzare il coinvolgimento attivo dei bambini attraverso una cornice narrativa: la storia di un robot, Cubetto, impegnato a raggiungere la Terra con l'aiuto di robot-ape (BeeBot). Ogni sessione si apriva con un messaggio narrativo che introduceva le attività, creando continuità e motivazione.
Dal punto di vista didattico, il percorso si è articolato in tre fasi principali:

- Attività propedeutiche: attività finalizzate allo sviluppo di prerequisiti (e.g., riconoscimento di pattern, distinzione destra/sinistra).
- Attività propedeutiche alla programmazione su carta (unplugged): introduzione dei concetti di sequenza e algoritmo attraverso attività concrete, in cui i bambini stessi agivano come "esecutori" delle istruzioni;
- Attività di robotica educativa: utilizzo di robot programmabili (BeeBot e Cubetto) per applicare e consolidare i concetti di pensiero algoritmico, tracing e debugging.

Questa integrazione ha permesso di combinare i vantaggi delle esperienze concrete e manipolative con quelli dell'interazione con agenti programmabili, offrendo un percorso didattico progressivo e coerente con le caratteristiche dello sviluppo cognitivo in età prescolare. La tabella 1 è un riepilogo delle attività proposte durante il training.

Sessioni	**Macro-Step**	**Attività**	**Obiettivi educativi**
Incontro 1	Fase propedeutica	Giochi finalizzati allo sviluppo delle abilità di orientamento di base, come la discriminazione destra/sinistra	Sebbene non inerente all'informatica, rinforzare le abilità coinvolte nel coding riduce il carico cognitivo in seguito
Incontro 2-3	Fase propedeutica	Riproduzione di "codici colore", ovvero sequenze di colori, utilizzando mattoncini da costruzione o punti colorati su griglie cartacee; riproduzione di sequenze di simboli figurativi	Attività introduttiva al concetto di "esecuzione sequenziale di istruzioni precise"
Incontro 4-5	Coding Unplugged	Comprensione di sequenze di istruzioni codificate ed esecuzione delle stesse per la creazione di immagini in pixel art	Obiettivi relativi all'ambito dati e informazione del documento del CINI: O-P3-D-1, O-P3-D-2,
Incontro 6-7	Coding Unplugged	Risoluzione di compiti di navigazione su mappe a dimensione reale, assumendo i ruoli di programmatore, robot ed esecutore/tracciatore dell'azione	Pensiero algoritmico visto da due prospettive (chi programma, chi esegue), abbiamo inoltre il debugging nel ruolo di chi traccia e osserva
Incontro 8	Coding Unplugged	Risoluzione di compiti di navigazione su mappe bidimensionali attraverso la costruzione ed esecuzione di sequenze di istruzioni mediante lo spostamento di una pedina	Propedeutico al passaggio al digitale, i ruoli vengono condensati in meno persone
Incontro 9	Educazione Robotica	Familiarizzazione con i robot BeeBot	
Incontro 10	Educazione Robotica	Lettura, comprensione e simulazione mentale di sequenze di istruzioni predefinite per il robot BeeBot	La simulazione mentale"tracing", fondamentale come abilità di programmazione e debugging
Incontro 11-12	Educazione Robotica	Programmazione di BeeBot per la risoluzione di compiti di navigazione	
Incontro 13-14	Educazione Robotica	Programmazione del robot Cubetto per risolvere compiti di navigazione, accompagnata dalla costruzione di narrazioni che motivano i percorsi	
Incontro di chiusura	Educazione Robotica	Riflessione metacognitiva sul significato di programmazione e sugli obiettivi dei laboratori	

Tabella 1: Articolazione delle attività didattiche

2.2.2 Materiali utilizzati per la valutazione delle abilità cognitive e di coding

Oltre alle abilità di coding, lo studio ha considerato alcuni costrutti cognitivi che la letteratura associa sia allo sviluppo del pensiero computazionale sia all'apprendimento della programmazione in età evolutiva [e.g., 16, 17]. In particolare, sono state valutate la pianificazione, intesa come la capacità di anticipare una sequenza di azioni e organizzarla in funzione di un obiettivo; l'inibizione della risposta, ovvero la capacità di controllare impulsi o risposte automatiche non appropriate e di selezionare quelle coerenti con il compito; e le abilità visuo-spaziali, che comprendono la rappresentazione mentale dello spazio, l'orientamento e la manipolazione di relazioni spaziali. Tali

abilità sono state incluse poiché le attività di coding richiedono ai bambini di pianificare sequenze di istruzioni, controllare risposte impulsive durante la risoluzione dei problemi e rappresentare mentalmente percorsi, posizioni e movimenti nello spazio. Di conseguenza, si ipotizza che un percorso progressivo di coding unplugged e robotica educativa possa favorire non solo l'acquisizione di competenze computazionali, ma anche il potenziamento di tali funzioni cognitive.

Inibizione della risposta. L'inibizione della risposta è stata valutata attraverso il subtest *Quadrato/Cerchio* della batteria NEPSY-II. Nel compito, i bambini dovevano nominare "quadrato" quando osservavano un cerchio e "cerchio" quando osservavano un quadrato, inibendo così la risposta automatica. Sono stati considerati il tempo totale di esecuzione e il numero di errori.

Pianificazione. Le capacità di pianificazione e problem solving sono state misurate mediante il test *Torre di Londra* (ToL). Ai bambini veniva richiesto di riprodurre configurazioni-obiettivo spostando palline colorate tra tre pioli secondo regole prestabilite. Sono stati registrati il tempo di pianificazione, misurato come l'intervallo di tempo dal momento in cui il test viene mostrato al bambino fino a quando lui/lei compie la prima mossa, staccando la prima pallina dal bastoncino; e l'accuratezza nella risoluzione dei compiti.

Abilità visuo-spaziali. Le abilità visuo-spaziali e di rotazione mentale sono state valutate mediante il subtest *Relazioni Spaziali* della batteria PMA (*Primary Mental Abilities*). Per ciascun item, il bambino doveva individuare quale figura completava correttamente una forma geometrica presentata-un quadrato. Il punteggio corrispondeva al numero di risposte corrette fornite entro 6 minuti.

Abilità di coding. Le abilità di programmazione sono state valutate mediante un test computerizzato composto da quattro problemi di navigazione bidimensionale a difficoltà crescente. Ai bambini veniva richiesto di guidare un'ape fino a una posizione-obiettivo programmando una sequenza di istruzioni tramite blocchi grafici equivalenti ai comandi del robot BeeBot (*avanti*, *indietro*, *gira a destra*, *gira a sinistra*). Gli esercizi prevedevano percorsi con lunghezza e numero di svolte progressivamente maggiori. Dopo aver composto il programma, i bambini potevano verificarne la correttezza senza visualizzare l'esecuzione del percorso, ricevendo soltanto un feedback sull'eventuale presenza di errori. Questa scelta ha permesso di limitare strategie basate su tentativi ed errori e di richiedere la simulazione mentale della sequenza di istruzioni. Sono stati considerati il tempo di pianificazione, definito come l'intervallo tra la presentazione del problema e l'inserimento del primo comando, e l'accuratezza, calcolata in base al numero di tentativi necessari per risolvere correttamente ciascun esercizio.

3 Risultati

I risultati mostrano effetti positivi significativi delle attività di introduzione all'informatica sulle competenze di programmazione e sulle capacità cognitive dei bambini, in particolare quelle viso-spaziali. Tra il pretest e il post-test, i bambini del gruppo sperimentale hanno migliorato le abilità di coding [$z = 3.84$, $p = 0.000$, $r = 0.87$], e le abilità visuo-spaziali [$z = 3.09$, $p = 0.002$, $r = 0.69$]. Il gruppo di controllo in lista d'attesa ha mostrato miglioramenti nelle capacità di coding solo dopo l'intervento, al punto di valutazione T3 [$z = 2.99$, $p = 0.003$, $r = 0.71$]. Nel gruppo sperimentale, le differenze nell'accuratezza della pianificazione tra T1 e T2 si sono avvicinate alla significatività statistica [$z = 2.54$, $p = 0.01$], dopo le correzioni di Bonferroni. L'entità dell'effetto era moderata [r= 0.55]. Per il gruppo di controllo, non sono emerse differenze statisticamente significative nella pianificazione tra il pre-test e il post-test.

Questi risultati mostrano che la pratica con il coding tangibile durante l'ultimo anno di scuola dell'infanzia non solo migliora significativamente le competenze dei bambini per risolvere problemi di coding (effetto *near-transfer*), ma può anche avere effetti di *far-transfer* sulle funzioni cognitive, come le abilità visuo-spaziali e la pianificazione.

I risultati ottenuti indicano che un percorso integrato di coding unplugged e robotica educativa può contribuire in modo significativo allo sviluppo delle competenze di programmazione in età prescolare. I miglioramenti osservati nelle prove di coding, diverse dalle attività svolte durante l'intervento, suggeriscono la capacità dei bambini di trasferire le conoscenze e le strategie apprese

a contesti nuovi, evidenziando l'acquisizione di competenze che vanno oltre la semplice familiarizzazione con specifici strumenti o attività. Inoltre, gli effetti rilevati sulle abilità visuo-spaziali e, seppur in misura più contenuta, sulle capacità di pianificazione, supportano l'ipotesi che le attività di programmazione possano favorire processi cognitivi coinvolti nell'organizzazione sequenziale delle azioni, nella rappresentazione mentale dei percorsi e nell'orientamento nello spazio. Al contrario, l'assenza di effetti significativi sull'inibizione della risposta suggerisce che tale funzione esecutiva possa richiedere interventi più specifici o di maggiore durata.

Dal punto di vista educativo, i risultati sembrano sostenere l'efficacia di una progressione didattica che introduca inizialmente i concetti fondamentali del pensiero computazionale attraverso attività unplugged, per poi estenderli alla programmazione di robot educativi. Tale approccio appare particolarmente coerente con le caratteristiche cognitive dei bambini in età prescolare, poiché consente di costruire gradualmente le rappresentazioni necessarie alla comprensione degli algoritmi e di ridurre il carico cognitivo associato alla gestione simultanea di aspetti concettuali, spaziali e operativi.

Lo studio presenta tuttavia alcuni limiti che richiedono cautela nell'interpretazione dei risultati. In primo luogo, il disegno sperimentale non consente di isolare il contributo specifico delle due componenti dell'intervento, non permettendo di stabilire se gli effetti osservati siano attribuibili prevalentemente al coding unplugged, alla robotica educativa o alla loro integrazione. Inoltre, la numerosità relativamente contenuta del campione e il coinvolgimento di un unico contesto scolastico limitano la generalizzabilità dei risultati. Future ricerche dovrebbero pertanto coinvolgere campioni più ampi e confrontare direttamente diverse modalità di intervento, al fine di identificare le strategie didattiche maggiormente efficaci per promuovere lo sviluppo del pensiero computazionale e delle abilità cognitive ad esso associate nella scuola dell'infanzia.

Bibliografia

[1] OECD. (2026). PISA 2025 Assessment and Analytical Framework: Science, Reading, Mathematics and Learning in the Digital World. OECD Publishing. https://www.oecd.org/en/publications/pisa-2025-assessment-and-analytical-framework_86c36975-en.html

[2] Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J., & Wang, X. (2023). Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. International Journal of STEM Education, 10(1), 47. https://doi.org/10.1186/s40594-023-00434-7

[3] Chen, G.; Shen, J.; Barth-Cohen, L.; Jiang, S.; Huang, X.; Eltoukhy, M. Assessing Elementary Students' Computational Thinking in Everyday Reasoning and Robotics Programming. Computers & Education. 2017, 109, 162–175. 10.1016/j.compedu.2017.03.001

[4] Kalelioğlu, F. A New Way of Teaching Programming Skills to K-12 Students: Code.Org. Comput. Hum. Behav. 2015, 52, 200–210. 10.1016/j.chb.2015.05.047

[5] Zhan, Z.; He, W.; Yi, X.; Ma, S. Effect of Unplugged Programming Teaching Aids on Children's Computational Thinking and Classroom Interaction: With Respect to Piaget's Four Stages Theory. J. Educ. Comput. Res. 2022, 60, 1277–1300. https://doi.org/10.1177/07356331211057143

[6] Li, F.; Wang, X.; He, X.; Cheng, L.; Wang, Y. The Effectiveness of Unplugged Activities and Programming Exercises in Computational Thinking Education: A Meta-Analysis. Educ. Inf. Technol. 2022, 27, 7993–8013. https://doi.org/10.1007/s10639-022-10915-x

[7] Barnett, S.M.; Ceci, S.J. When and Where Do We Apply What We Learn?: A Taxonomy for Far Transfer. Psychol. Bull. 2002, 128, 612–637. https://doi.org/10.1037/0033-2909.128.4.612

[8] Perkins, D.N.; Salomon, G. Transfer of Learning. Int. Encycl. Educ. 1992, 2, 6452-6457.

[9] Montuori, C.; Gambarota, F.; Altoé, G.; & Arfé, B. The cognitive effects of computational thinking: A systematic review and meta-analytic study. Computers & Education. 2024. 210, 104961. https://doi.org/10.1016/j.compedu.2023.104961

[10] Scherer, R.; Siddiq, F.; Sánchez Viveros, B. The Cognitive Benefits of Learning Computer Programming: A Meta-Analysis of Transfer Effects. J. Educ. Psychol. 2019, 111, 764–792. https://doi.org/10.1037/edu0000314

[11] Canbeldek, M.; Isikoglu, N. Exploring the Effects of "Productive Children: Coding and Robotics Education Program" in Early Childhood Education. Educ. Inf. Technol. 2023, 28, 3359–3379.

[12] Brainin, E.; Shamir, A.; Eden, S. Robot Programming Intervention for Promoting Spatial Relations, Mental Rotation and Visual Memory of Kindergarten Children. J. Res. Technol. Educ. 2022, 54, 345–358. https://doi.org/10.1080/15391523.2020.1858464

[13] Campbell, M.K.; Elbourne, D.R.; Altman, D.G. CONSORT Statement: Extension to Cluster Randomised Trials. BMJ 2004, 328, 702–708. https://doi.org/10.1136/bmj.328.7441.702

[14] Marinus, E.; Powell, Z.; Thornton, R.; McArthur, G.; Crain, S. Unravelling the Cognition of Coding in 3-to-6-Year Olds: The Development of an Assessment Tool and the Relation between Coding Ability and Cognitive Compiling of Syntax in Natural Language. In Proceedings of the 2018 ACM Conference on International Computing Education Research, Espoo, Finland, 8 August 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 133–141.

[15] Critten, V.; Hagon, H.; Messer, D. Can Pre-School Children Learn Programming and Coding Through Guided Play Activities? A Case Study in Computational Thinking. Early Child. Educ. J. 2022, 50, 969–981. https://doi.org/10.1007/s10643-021-01236-8

[16] Arfé, B.; Vardanega, T.; Montuori, C.; Lavanga, M. Coding in Primary Grades Boosts Children's Executive Functions. Front. Psychol. 2019, 10, 2713. https://doi.org/10.3389/fpsyg.2019.02713

[17] Schmied, S.; Landerl, K.; Binder, B.; & Kemény, F. Executive functions predict kindergarten children's coding performance. Computer Science Education. 2025, 1–21. https://doi.org/10.1080/08993408.2025.2600238

Il Ritratto Statistico: un gioco da tavolo per smontare la magia dell'IA generativa

Barbara Baldi[1] , Giuliana Gelsomino[2]
[1]IC "Santorre di Santarosa", Savigliano (CN)
baldi.barbara@icsantarosasavigliano.edu.it
[2] IC "Virgilio-Gonzaga", Eboli (SA)
giuliana.gelsomino23@gmail.com

Abstract

L'Intelligenza Artificiale (IA) generativa è entrata nella vita quotidiana delle studentesse e degli studenti, spesso percepita come dotata di creatività autonoma. Il laboratorio "Il Ritratto Statistico", attività unplugged per il primo ciclo d'istruzione, smonta questa visione simulando il processo di apprendimento di un modello generativo attraverso un gioco da tavolo: con un dataset fisico di quaranta carte della Famiglia Rossi, gli studenti/esse agiscono come algoritmi umani, raccolgono dati, individuano pattern e costruiscono il personaggio statisticamente più probabile della famiglia, sviluppando una riflessione critica su bias e rappresentazione della realtà nei sistemi algoritmici. Il contributo descrive le fasi del laboratorio, ne analizza i contenuti informatici e presenta gli esiti di una sperimentazione con un gruppo di studentesse e studenti di classe prima e seconda di secondaria di primo grado.

1 Introduzione e contesto didattico

L'attività si colloca nell'ambito dell'Informatica come strumento di cittadinanza scientifica e risponde alle Nuove Indicazioni Nazionali per il primo ciclo d'istruzione (DM 221 del 9 dicembre 2025), che introducono per la prima volta obiettivi specifici di Informatica fin dalla scuola primaria, tra cui l'uso critico e consapevole delle tecnologie digitali, incluse quelle di Intelligenza Artificiale (IA) [2]. Risponde, inoltre, all'indirizzo del CINI di insegnare l'Informatica attraverso la modellazione dei dati [1], alle Linee Guida internazionali sull'alfabetizzazione all'IA [4], e all'obbligo di «AI literacy» dell'Articolo 4 del Regolamento europeo sull'Intelligenza Artificiale (AI ACT - Regolamento UE 2024/1689)[7].

Quando un sistema genera un'immagine da una richiesta testuale, il risultato appare sorprendente e viene interpretato come prodotto di creatività artificiale, mentre i modelli generativi si basano sull'analisi statistica di grandi quantità di dati e sull'individuazione di regolarità al loro interno. Rendere accessibile questo concetto alle alunne e agli alunni del primo ciclo è la sfida che l'attività didattica "Il Ritratto Statistico" affronta trasformando un processo computazionale complesso in un'esperienza concreta, in cui le studentesse e gli studenti assumono il ruolo di algoritmi umani; l'approccio unplugged consente di concentrarsi sui concetti fondamentali senza richiedere dispositivi, rendendo il laboratorio replicabile in ogni contesto scolastico.

2 Progettazione dell'esperienza

Il laboratorio è costruito attorno a un dataset fisico di quaranta carte illustrate raffiguranti i membri della Famiglia Rossi, ciascuna caratterizzata da tratti visivi riconducibili a categorie osservabili: presenza di occhiali, tipologia di capelli, età apparente, accessori, espressioni del volto ed elementi distintivi dell'abbigliamento. Una scheda operativa (accessibile qr code alla fine del documento) guida le studentesse e gli studenti nella raccolta e nell'analisi dei dati, evidenziando tre concetti fondamentali: il dataset come insieme di esempi da cui apprendere, il riconoscimento dei pattern come base dell'apprendimento automatico e la generazione come processo probabilistico e non creativo.

Un elemento chiave della progettazione riguarda la suddivisione del mazzo: a ciascun piccolo gruppo vengono assegnate solo cinque carte, scelte *ad hoc* in modo da risultare, secondo i casi, omogenee oppure fortemente polarizzate rispetto a un tratto; nessun gruppo dispone quindi di una visione complessiva del dataset. Questa scelta simula concretamente il problema del bias nei sistemi di IA: un sottoinsieme non rappresentativo produce un risultato

altrettanto non rappresentativo, mentre un dataset più inclusivo ed esteso avvicina l'output alla realtà che intende descrivere.

Ogni gruppo osserva sistematicamente le proprie cinque carte, registra in una tabella delle frequenze il numero di elementi distintivi/caratteristiche di ciascun tratto e ne calcola la percentuale relativa, introducendo implicitamente il concetto di modellazione dei dati (Figura 1).

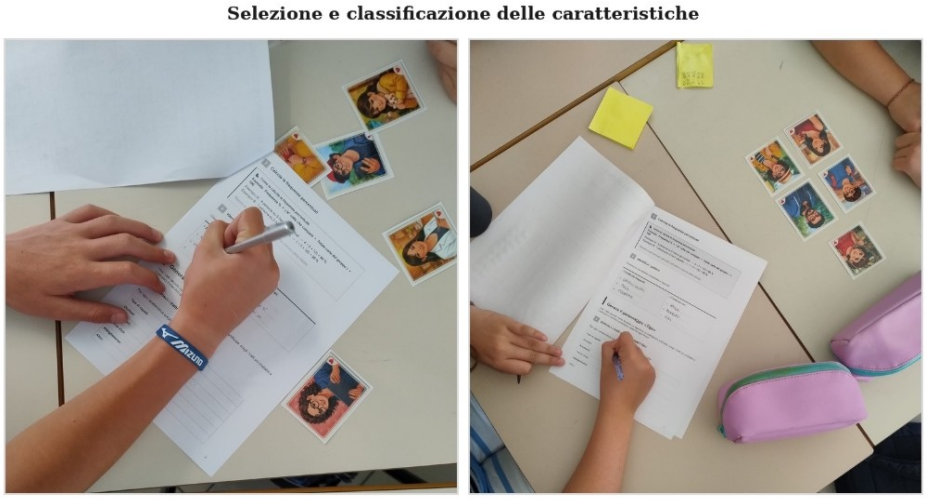


Figura 1: Selezione e classificazione delle caratteristiche dalle carte.

Analizzando le frequenze, le studentesse e gli studenti identificano i tratti più comuni e quelli meno rappresentati nel proprio campione, scoprendo che l'apprendimento automatico non si basa sulla comprensione del significato delle immagini, ma sull'individuazione di regolarità statistiche. Per ogni categoria viene quindi selezionata la caratteristica più frequente: i tratti scelti vengono combinati in un unico personaggio (Figura 2), il «tipo» del proprio sottoinsieme di carte, che nella maggior parte dei casi non coincide con alcun soggetto realmente presente nel mazzo, segno che il modello non copia i dati esistenti, ma costruisce nuove combinazioni dalle probabilità apprese.

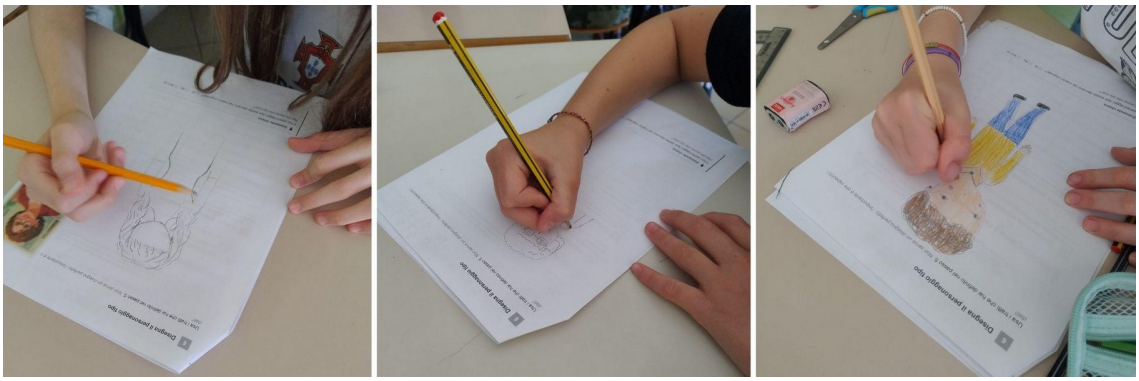


Figura 2: Studentesse e studenti impegnati nella fase di disegno del personaggio tipo

In un secondo momento i dati di tutti i gruppi vengono riuniti in un'unica tabella complessiva, da cui la classe elabora un nuovo personaggio tipo. Il confronto fra i personaggi della prima fase, ciascuno condizionato dal proprio sottoinsieme parziale e talvolta polarizzato, e quello del dataset completo rende tangibile l'effetto della rappresentatività dei dati sull'output di un sistema generativo. A titolo illustrativo, in una sperimentazione condotta durante la presentazione del progetto il personaggio tipo generato da un mazzo completo presentava occhiali al 75%, capelli ricci al 90%, maglia blu al 60% e assenza di cappello al 90%, un risultato che, come previsto, non corrisponde ad alcun individuo realmente presente nel mazzo originale.

3 Dai dati al bias: contenuti informatici e aspetti multidisciplinari

La forza didattica del laboratorio risiede nel collegare l'esperienza concreta ai meccanismi dei moderni sistemi di generazione di immagini: costruendo il personaggio tipo, le studentesse e gli studenti simulano un processo analogo a quello di un modello di IA, che non possiede immaginazione né creatività in senso umano ma produce risultati coerenti con le regolarità osservate nei dati di addestramento. Questo nucleo, rappresentazione dei dati e pattern recognition, costituisce il contenuto informatico specifico dell'attività ed è rafforzato dalla riflessione finale, in cui le studentesse e gli studenti si interrogano su cosa accadrebbe se una caratteristica fosse del tutto assente, quali elementi il ritratto statistico valorizzi e quali ignori, e chi decida quali immagini inserire nel dataset. Ne emerge che un dataset non è mai neutrale, e che le scelte compiute nella sua costruzione influenzano inevitabilmente il comportamento del modello: un fenomeno documentato anche nei sistemi reali di classificazione automatica, dove dataset non bilanciati producono tassi di errore diversi tra sottogruppi [6].

Vanno distinti da questo nucleo disciplinare gli aspetti più propriamente trasversali dell'attività: la discussione su chi seleziona le immagini di addestramento e sulle relative implicazioni etiche richiama competenze di Cittadinanza digitale e pensiero critico, di natura interdisciplinare, qui non approfondite in chiave Informatica.

4 Risultati, traguardi e obiettivi di apprendimento

Per la primaria l'attività concorre ai traguardi T-P-5 (riconoscere la differenza tra informazione e dati) e T-P-6 (rappresentare dati di varia natura mediante formati diversi) della Proposta CINI [1], con riferimento a O-P3-D-1/2 per la terza e O-P5-D-1/2 per la quinta. Per la secondaria di primo grado, il gruppo di alunne e alunni in cui è stata condotta la sperimentazione, l'attività si collega a T-M-9 (classifica le tipologie di dati) e O-M-D-2 (operazioni su simboli che rappresentano informazione strutturata, es. immagini bitmap), estendendo l'idea di immagine a matrice di dati discreti analizzabile statisticamente [1]; traguardi coerenti con le Nuove Indicazioni Nazionali (DM 221/2025) [2] e con il quadro internazionale dei Big Ideas in AI [4].

L'attività è stata sperimentata in due classi prime e in un gruppo di studentesse e studenti di classe seconda di scuola secondaria di primo grado, raccogliendo due questionari per le cinque docenti coinvolte e per 27 studentesse e studenti dei 70 partecipanti. Le docenti hanno valutato la scheda di lavoro chiara e ben strutturata (80% «Sì, molto», 20% «Abbastanza», con istruzioni comprensibili senza ulteriori spiegazioni nell'80% dei casi; nessuna ha riscontrato criticità nella gestione dei gruppi e tutte hanno osservato una buona comprensione della frequenza statistica nella maggioranza della classe, con coinvolgimento valutato uniformemente a 4 su 5. La riflessione finale su bias ha generato discussione solo in parte secondo tre docenti su cinque. Tutte riproporrebbero l'attività, indicando come agganci principali Matematica/Statistica (100%), Tecnologia/Informatica e Italiano/Educazione civica (80% ciascuno), e suggerendo più carte per gruppo e più tempo per la costruzione del personaggio tipo.

Tra le studentesse e gli studenti (n=27), l'85% (23 su 27) ha scelto correttamente «calcola le probabilità sui dati di addestramento» per spiegare il funzionamento di un'IA generativa (Figura 3).

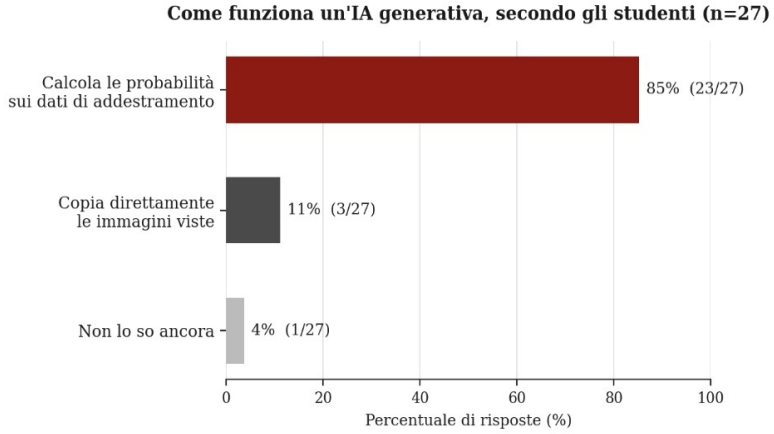


Figura 3: Comprensione del meccanismo generativo (n=27).

La sicurezza percepita si è attestata a 3,7 su 5, il gradimento a 4,0, l'utilità percepita a 4,2, con difficoltà valutata bassa (2,3); il lavoro di gruppo è risultato la componente meglio valutata (4,3; Figura 4).

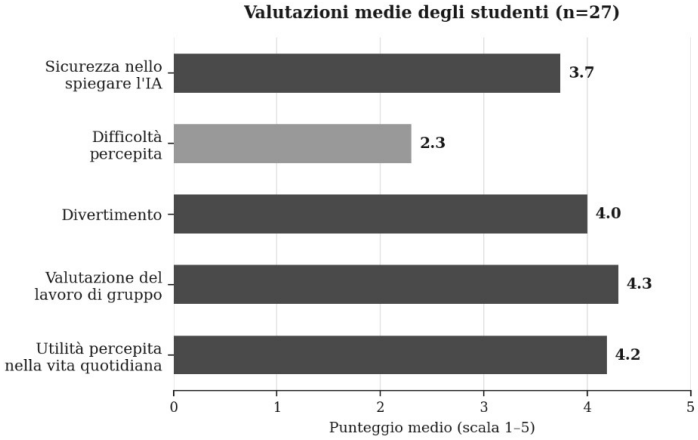


Figura 4: Valutazioni medie su scala 1–5 (n=27).

Il 44% degli studenti ha indicato il lavoro di gruppo come la parte più coinvolgente, seguito dal disegnare il personaggio tipo (30%) e dall'osservazione e classificazione delle carte (22%; Figura 5); il calcolo delle frequenze percentuali resta il passaggio più ostico tra quelli identificabili. Il 70% degli studenti presterà più attenzione a come vengono creati i contenuti generati dall'IA dopo l'attività.

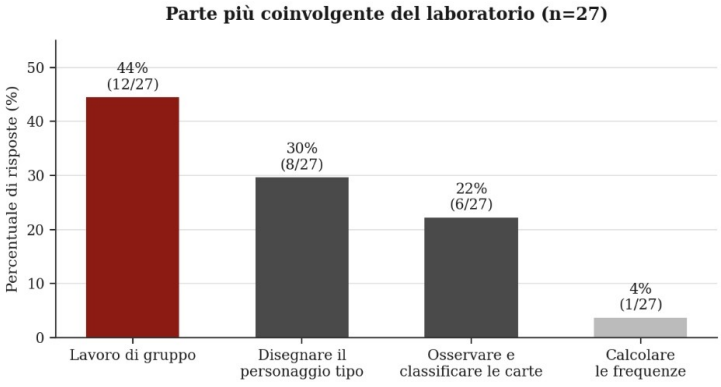


Figura 5: Parte più coinvolgente del laboratorio (n=27).

L'osservazione diretta ha inoltre confermato come i materiali concreti favoriscano la comprensione di concetti astratti quali dataset e probabilità, e che diversi tra studentesse e studenti abbiano collegato spontaneamente il bias alle proprie esperienze quotidiane con i contenuti generati dall'IA.

Nel complesso, il riscontro converge nell'indicare l'attività efficace nel trasmettere la natura statistica, e non creativa, della generazione algoritmica, pur con margini di miglioramento nella gestione dei tempi.

5 Conclusioni

"Il Ritratto Statistico" mostra come un gioco da tavolo, privo di dispositivi digitali, possa rendere tangibile un concetto altrimenti astratto: la natura statistica e non creativa dei sistemi di IA generativa. I risultati della sperimentazione, con un'ampia maggioranza di studentesse e studenti in grado di descrivere correttamente il meccanismo probabilistico sotteso, confermano l'efficacia dell'approccio. Trasformando le studentesse e gli studenti in algoritmi umani, l'attività consente di osservare il passaggio dai dati ai modelli e dai modelli alle immagini generate, anticipando già dal primo ciclo nozioni di rappresentazione dei dati e pattern recognition. Ne risulta favorita non solo l'acquisizione di competenze di Scienza dei dati e pensiero computazionale, ma anche una consapevolezza critica verso le tecnologie della società contemporanea: in un contesto educativo sempre più attraversato dall'IA, esperienze come questa contribuiscono a formare cittadini capaci di comprendere, interpretare e valutare i sistemi digitali con cui interagiscono quotidianamente.

Bibliografia

[1] Gruppo di Lavoro CINI "Informatica e Scuola". Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola. Consorzio Interuniversitario Nazionale per l'Informatica, 2017.

[2] Ministero dell'Istruzione e del Merito. Decreto Ministeriale 9 dicembre 2025, n. 221 – Regolamento recante Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione. Gazzetta Ufficiale della Repubblica Italiana, Serie Generale n. 21, 27 gennaio 2026.

[3] Il ritratto statistico. Materiale didattico presentato a Didacta Italia, 11-13 marzo 2026, Fortezza da Basso, Firenze.

[4] Touretzky, D. S., Gardner-McCune, C., Martin, F., Seehorn, D. Envisioning AI for K-12: What Should Every Child Know about AI? Proceedings of the AAAI Conference on Artificial Intelligence, 33(1), 9795–9799, 2019.

[5] Carrisi, M. C., Marras, M., & Vergallo, S. (2025) A Structured Unplugged Approach for Foundational AI Literacy in Primary Education In: arXiv preprint arXiv:[2505.21398] (http://arxiv.org/abs/2505.21398)

[6] Buolamwini, J., Gebru, T. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. Proceedings of Machine Learning Research, 81, 77–91, 2018.

[7] Regolamento (UE) 2024/1689 del Parlamento Europeo e del Consiglio del 13 giugno 2024 (AI Act), art. 4. Gazzetta Ufficiale dell'Unione Europea, 12 luglio 2024.

Attività didattiche di programmazione per lo studio della geometria nella scuola secondaria di primo grado

Martina Palazzolo
Istituto Comprensivo Ilaria Alpi di Milano
martina.palazzolo@icilariaalpi.edu.it

Abstract

Presentiamo la nostra esperienza e le nostre osservazioni relative ad attività di programmazione che sono state introdotte durante le lezioni curricolari di geometria nella scuola secondaria di primo grado in una classe seconda.
Abbiamo progettato e attuato un percorso di programmazione utilizzando il linguaggio *LibreLogo* della suite *LibreOffice* introducendo alcuni dei costrutti tipici della programmazione quali la ripetizione e le funzioni. *LibreLogo* è un'implementazione del software di videoscrittura *Writer* ed è adatto a scrivere programmi sul foglio di videoscrittura per ottenere disegni geometrici. Abbiamo quindi svolto le attività durante le lezioni di geometria sfruttando le caratteristiche geometriche dei poligoni e della circonferenza per introdurre i costrutti scelti.
Riportiamo infine le nostre osservazioni tratte dai programmi prodotti dagli alunni al termine del percorso. Durante la scrittura dei programmi per realizzare i loro disegni geometrici i ragazzi hanno dimostrato interesse e attenzione e di saper utilizzare comandi e costrutti con competenza.

1 Introduzione

Le attività di programmazione presentate l'anno scorso a ITADINFO [7, 8] hanno dimostrato un largo e creativo coinvolgimento degli alunni nelle attività proposte a partire dai giochi dell'informatica *Bebras* [2, 3]. Nel percorso "Cuori geometrici"[7] gli studenti avevano iniziato a ragionare attraverso le cosiddette astrazioni dei comandi, ovvero descrizioni di ciò che un comando deve poter fare se eseguito, senza il rigore formale del linguaggio di programmazione, in modo da capire come potrebbe essere impostato un programma, prima di scriverlo (in accordo con il traguardo per lo sviluppo delle competenze T-P-4 della Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella scuola: l'alunno spiega usando il ragionamento logico perché un programma strutturalmente semplice raggiunge i suoi obiettivi; [10]). Gli studenti erano quindi stati invitati a scrivere un programma per realizzare il disegno progettato dopo aver imparato i primi comandi del linguaggio di programmazione *LibreLogo* (in accordo con il traguardo per lo sviluppo delle competenze T-P-2: lo studente comprende come un algoritmo può essere espresso mediante un programma scritto usando un linguaggio di programmazione [10]). Oltre ai comandi di movimento FORWARD e LEFT o RIGHT gli alunni avevano imparato a utilizzare le funzioni offerte dalla libreria di *LibreLogo* SQUARE, CIRCLE, RECTANGLE che disegnano rispettivamente un quadrato, un cerchio e un rettangolo.
Osservato il coinvolgimento degli alunni durante le attività, abbiamo deciso di progettare un nuovo percorso di programmazione sempre utilizzando *LibreLogo* ma introducendo ulteriori costrutti: la ripetizione e le funzioni in particolare, sfruttando le caratteristiche geometriche dei poligoni regolari e della circonferenza.
Librelogo è un'implementazione del software di videoscrittura *Writer* della suite *LibreOffice* [5] che si ispira al linguaggio Logo ideato da Seymour Papert negli anni '80 [9, 1] a scopi didattici e permette di scrivere direttamente sul foglio di videoscrittura la sequenza di istruzioni che un esecutore a forma di tartaruga (da qui in avanti denominata *robot-turtle*) esegue. Le istruzioni che il *robot-turtle* esegue sono diverse: le prime ad essere introdotte in classe sono le istruzioni che

fanno muovere in avanti o indietro il *robot-turtle.* Mentre si muove il *robot-turtle* traccia una linea disegnando sul foglio.
Il percorso progettato e attuato è di tipo laboratoriale e collaborativo. Le lezioni della durata di 2 ore (circa una volta al mese) sono state progettate in fasi: 1) fase di acquisizione delle istruzioni in cui vengono presentate le istruzioni (in inglese) e si ipotizza tutti insieme, a partire dalla loro traduzione in italiano, cosa faranno fare al *robot-turtle* quando le esegue 2) attività pratiche di allenamento con carte da gioco con sfide a difficoltà crescente; 3) scrittura da parte degli alunni di programmi a tema specifico oppure liberi.
Ogni attività è stata svolta per la fase 1 e 2 nell'arco di 2 ore mentre la fase 3 può aver richiesto tempi differenti a seconda dei tempi personali dagli alunni e può essere stata completata a casa in modo autonomo. Gli alunni hanno lavorato prevalentemente in modo individuale, soprattutto nella fase 3 in cui sono impagnati per la maggior parte del tempo alla realizzazione dei loro progetti personali. Tuttavia è sempre incoraggiato il confronto e la collaborazione con i pari e l'insegnante e soprattutto la condivisione di scoperte. L'insegnante monitora i lavori degli alunni fornendo feedback, invitandoli a osservare particolarità utili senza necessariamente correggere gli eventuali errori ma invitandoli a fare sperimentazioni in autonomia fornendo suggerimenti piuttosto che risposte o correzioni [4].

2 Il set di istruzioni

Tra le istruzioni messe a disposizione da *LibreLogo* ne abbiamo scelte alcune e le abbiamo presentate agli alunni per categorie: istruzioni di movimento, istruzioni di stile e istruzioni per i costrutti che abbiamo definito "di sintesi", ovvero che possono racchiudere più istruzioni in un unico comando.
Le prime ad essere state introdotte in classe sono le istruzioni di movimento FORWARD o BACK e RIGHT o LEFT. Il comando FORWARD N, con N numero intero, fa muovere la tartaruga in linea retta nella direzione in cui guarda di N passi. L'istruzione BACK N, invece fa muovere il *robot-turtle* all'indietro di N passi. I comandi RIGHT n e LEFT n la fanno ruotare rispettivamente in senso orario e antiorario di n gradi. Già con queste semplici istruzioni è possibile far lavorare i ragazzi su programmi per disegnare i primi poligoni quali in quadrato, il rettangolo e il triangolo equilatero partendo dall'osservazione delle loro proprietà geometriche intrinseche [9].
Sfruttando le proprietà intrinseche del quadrato è stata introdotta l'istruzione REPEAT e la sua sintassi: REPEAT N [istruzioni da ripetere] con N numero intero.
Una volta stabilita la sequenza di istruzioni per un quadrato o un rettangolo o un triangolo equilatero è stato introdotto il costrutto TO … END, ovvero il costrutto "TO *nome istruzione* … END" che permette di nominare una sequenza specifica di istruzioni e quindi di eseguirla scrivendo l'istruzione "*nome istruzione*".
Per aggiungere ai disegni un tocco artistico sono state introdotte alcune istruzioni di stile: PENCOLOR "*color*" che permette di tracciare linee colorate e PENWIDTH N che ne aumenta/diminuisce lo spessore; FILLCOLOR "*color*" e FILL che permettono di colorare figure chiuse. Poichè non sempre è utile disegnare al movimento del *robot-turtle* sono state introdotte le istruzioni PENUP, che pemette al *robot-turtle* di muoversi senza disegnare, e PENDOWN permette di nuovo di tracciare il movimento.

3 Le carte da gioco

Nella prima attività, che aveva l'obiettivo di far prendere dimestichezza con le funzionalità di *LibreLogo* e con le prime istruzioni di movimento FORWARD o BACK e LEFT o RIGTH, i ragazzi hanno lavorato pescando carte da gioco raffiguranti disegni geometrici di difficoltà crescente. I primi disegni contengono, per esempio, solo segmenti adiacenti ed angoli di 90 gradi

o al massimo di 45 gradi, mentre quelli più avanzati richiedono di ragionare sul rapporto tra le lunghezze dei segmenti o di usare anche i comandi PENUP o PENDOWN. Pescata la carta viene chiesto agli alunni (il lavoro può essere svolto in gruppo o individualmente) di riprodurre prima il disegno sul foglio a quadretti con matita e righello e quindi di scrivere la possibile sequenza di istruzioni in *LibreLogo* per far disegnare al *robot-turtle* quanto raffigurato. Passano la loro sequenza di istruzioni a un altro compagno che non ha visto la carta e che deve riprodurre il disegno sul quaderno. Solo alla fine gli alunni lavorano in aula di informatica per verificare l'esattezza dei loro programmi direttamente al PC e possono lavorare con i compagni collaborando per risolvere problemi e correggere i loro programmi. Le carte riportano su una faccia le istruzioni a disposizione e sull'altra il disegno da riprodurre (fig. 1). Le istruzioni sono anche appese sulla parete dell'aula stessa di informatica in modo da essere sempre a disposizione.

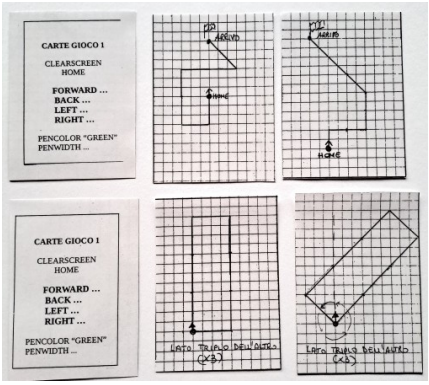


Fig. 1 – Primo set di carte da gioco – A sinistra la faccia in comune a sinistra esempi di disegno

Esaurite le carte da gioco gli studenti sono stati invitati a scrivere il programma per far disegnare al *robot-turtle* le figure geometriche classiche (quadrato, rettangolo e triangolo equilatero). In particolare con il quadrato è stata introdotta l'istruzione REPEAT sfruttando la necessaria ripetizione per quattro volte di FORWARD N e LEFT o RIGHT 90. Per allenarsi con l'istruzione REPEAT un altro set di carte riporta varie sequenze di istruzioni e si chiede ai ragazzi di provare a riprodurre sul quaderno la sequenza di istruzioni da ripetere (quella tra le parentesi quadre) e una bozza dell'ipotetico disegno con la ripetizione determina e quindi verificarne l'esecuzione con il PC (fig. 2).

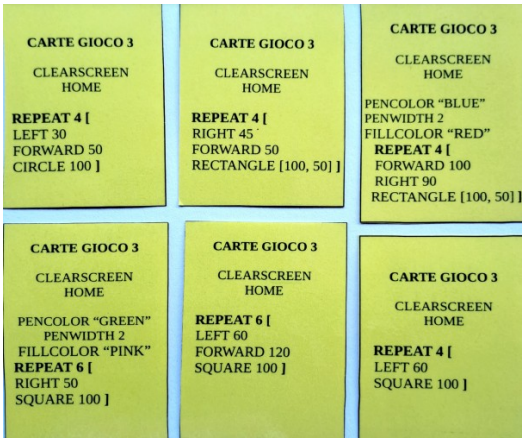


Fig. 2 – Set di carte da gioco per allenamento con REPEAT

L'ultimo set di carte utilizzato (fig.3) è propriamente di tipo geometrico e ha come obiettivo quello di far ragionare gli studenti sulle proprietà dei poligoni regolari che presentano un certo numero di lati e angoli tutti congruenti tra loro. Perché il *robot-turtle* possa disegnarli percorrendone il contorno è quindi necessario che compia un certo numero movimenti in avanti e di rotazioni di un angolo specifico che si ripete. Alcune domande guida ("Quanti angoli e lati?", "Rotazioni totali?", "Angolo di rotazione?") hanno l'intento di portare gli alunni ha correlare il numero di lati e angoli con il numero di rotazioni e a determinare il grado di rotazione. Sulle carte non ci sono istruzioni ma solo disegni e domande guida. Tutte le carte hanno una faccia con un triangolo equilatero con il quale è stato fatto il ragionamento tutti insieme in classe alla lavagna. L'attività è risultata impegnativa e richiede di essere implementata, tuttavia gli alunni sono riusciti a comprendere i programmi presentati alla fine dall'insegnante e a utilizzarli per i loro progetti. Il lavoro è stato preliminare alla definizione di un programma per ottenere le linee curve e la circonferenza in particolare, anche in questo caso alla fine proposto dall'insegnante piuttosto che scritto direttamente dagli alunni. Vedremo nel prossimo paragrafo come gli alunni siano riusciti ad utilizzare quanto imparato per ottenere disegni "liberi" interessanti contenenti anche linee curve e circonferenze.

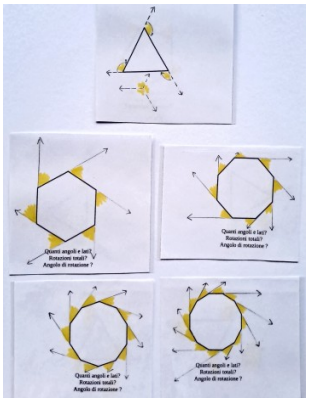


Fig. 3 – Set di carte da gioco con poligoni regolari

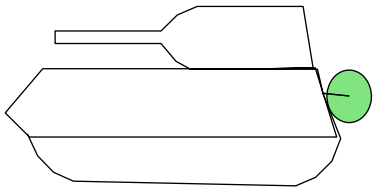
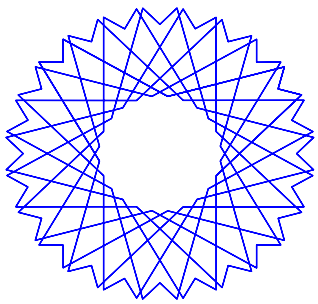
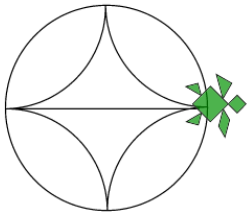
4 I programmi degli alunni

In questa ultima sezione prendiamo in esame alcuni lavori degli alunni (tabella 1) che dimostrano l'efficacia del percorso e in cui si evidenziano i diversi livelli di competenza raggiunti. La consegna per questi lavori è stata appositamente molto generica ed è stata richiesta nell'ultimo mese di scuola, a fine maggio. Agli alunni è stato chiesto di disegnare con *LibreLogo* il proprio logo o un disegno libero ricordando loro come usare REPEAT e TO … END alla lavagna.

5 Conclusioni

Dei 25 alunni 14 hanno consegnato il disegno "libero" richiesto. I 4 programmi scelti come esempo (tabella 1) rappresentano indicativamente i livelli di competenza raggiunti osservati:
1 – uso delle istruzioni di movimento;
2 – uso delle istruzioni di movimento e di REPEAT;
3 – Uso delle istruzioni precedenti e disegno di linee curve e circonferenze;
4 – Uso delle istruzioni precedenti e del costrutto TO … END.
I restanti 11 alunni hanno lavorato su schede di esempio modificando programmi già scritti e resi disponibili alla classe.

Possiamo concludere che le attività proposte sono state ben accolte ed elaborate dagli alunni. L'introduzione di *LibreLogo* in ambito geometrico è risultato proficuo e ha permesso di introdurre elementi della programmazione in modo naturale inserendoli nel curricolo di matematica, come previsto nelle Nuove indicazioni Nazionali [6].

PROGRAMMA	ESECUZIONE
ESEMPIO 1 – Uso esclusivo di FORWARD, LEFT, RIGHT e CIRCLE	
Clearscreen home left 90 forward 190 left 45 forward 50 left 95 forward 30 left 40 forward 290 left 110 forward 59 left 71 forward 69 right 180 forward 67 left 100 forward 50 left 79 forward 100 left 20 forward 20 left 20 forward 20 right 40 forward 100 left 90 forward 10 left 90 forward 100 right 45 forward 20 left 20 forward 15 left 25 forward 120 right 75 forward 20 left 70 forward 25 circle 42 left 180 forward 25 left 120 forward 40 right 50 forward 20 right 25 forward 20 right 20 forward 20 right 21 forward 210 right 20 forward 20 right 20 forward 20 right 20 forward 20 home	
ESEMPIO 2 – Uso di REPEAT	
Clearscreen home pendown to fiore pencolor "blue" right 45 forward 100 left 120 forward 20 right 90 forward 20 left 120 forward 100 end repeat 50 [fiore]	
ESEMPIO 3 – Uso di linee curve	
clearscreen home repeat 360 [forward 1 right 1] right 90 repeat 90 [forward 1 right 1] right 180 repeat 90 [forward 1 right 1] right 180 repeat 90 [forward 1 right 1] right 180 repeat 90 [forward 1 right 1] right 180 forward 110	

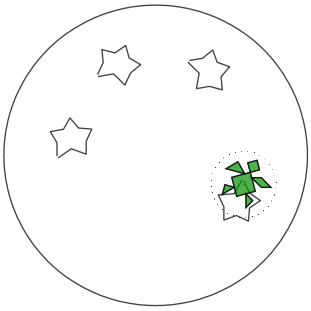
ESEMPIO 4 - costrutti annidati	ESECUZIONE
Clearscreen home **to** stella forward 10 left 30 forward 10 right 110 forward 10 left 50 forward 10 right 110 fd 10 left 45 forward 10 rt 110 fd 10 left 25 forward 10 right 110 forward 12 left 55 forward 12 **end** **to** PANDISTELLE pendown repeat 360 [forward 2 right 1] penup right 90 forward 40 left 90 pendown stella penup right 150 forward 70 pendown stella penup right 180 forward 70 pendown stella penup left 160 forward 100 pendown stella **end** PANDISTELLE	

Tabella 1 – A sinistra i programmi scritti dagli alunni e a destra la loro esecuzione

Per ragioni di spazio nella tabella più istruzioni sono state scritte sulla stessa riga dall'autrice ma gli alunni hanno consegnato i loro programmi scrivendo una istruzione per riga.

Bibliografia

[1] Harold Abelson and Andrea diSessa. *Turtle Geometry: The Computer as a Medium for Exploring Mathematics,* MIT Press, 1981

[2] Bebras dell'informatica – Edizione Italiana, *https://bebras.it*

[3] Bebras – International Challenge on Informatics and Computational Thinking, *https://bebras.org/*

[4] C. Bellettini, V. Lonati, M. Monga, A. Morpurgo, M. Palazzolo. *Situated Learning with Bebras Tasklets*. ISSEP 2019 pagg. 225 – 239 LNCS 11913, Springer 2019

[5] LibreLogo di LibreOffice - *https://it.libreoffice.org*

[6] Nuove Indicazioni per il Curricolo, Scuola dell'Infanzia e Scuole del Primo Ciclo di Istruzione *https://www.mim.gov.it/documents/20182/0/Indicazioni+Nazionali.pdf/1fb1a29e-67dd-59a0-6c58-f1ad455c526a*

[7] M. Palazzolo. *Attivita` didattiche di programmazione a partire dai giochi dell'informatica Bebras.* Atti del convegno ITADINFO Convegno Italiano sulla Didattica dell'Informatica, Salerno, 3-5 ottobre 2025, ISBN 9791256005123.

[8] M. Palazzolo, R. Signorini, D. Vigano'. *I giochi Bebras dell'informatica nel curricolo della scuola del primo ciclo.* Atti del convegno ITADINFO Convegno Italiano sulla Didattica dell'Informatica, Bari 13-15 ottobre 2023, ISBN 978-88-6629-075-9.

[9] Seymour Papert. *Mindstorms* – https://www.media.mit.edu/publications/mindstorms/

[10] Proposta Indicazioni Nazionali Informatica a Scuola - *https://www.consorzio-cini.it/images/Proposta-Indicazioni-Nazionali-Informatica-Scuola-numerata.pdf*

Un Pezzettino… alla volta

Maurizia Gai[1], Cinzia D'Elia[2], Laura Galati[3] Virginia Macrì[4]
[1] I.C. Vivaldi-Murialdo di Torino
maurizia.gai@unito.it
[2] I.C. Vivaldi-Murialdo di Torino
delia.cinzia@vivaldi-murialdo.edu.it
[3] I.C. Vivaldi-Murialdo di Torino
galati.laura@vivaldi-murialdo.edu.it
[4] I.C. Da Vinci Frank di Torino
virgy.macri@gmail.com

Abstract

L'albo illustrato "Pezzettino" di Leo Lionni è una profonda metafora sulla costruzione dell'identità e sull'autostima. Il testo viene frequentemente utilizzato nelle scuole a partire dai 5-6 anni, quando i bambini iniziano a definire il proprio senso di identità rispetto ai coetanei, affiancato da attività laboratoriali con la tecnica del collage. Nel racconto di quest'esperienza sul campo, condotta in una classe prima della scuola primaria, si espone come assemblando "pezzettini" di carta colorata e, partendo da una situazione problema presentata da un alunno riguardante le coordinate, è stato effettuato un primo approccio a concetti informatici quali dati, informazioni, rappresentazione simbolica e sistema di codifica/decodifica.

1 Introduzione

Questo progetto si propone come un percorso fortemente interdisciplinare ed inclusivo, capace di connettere i diversi saperi della scuola primaria valorizzando l'unicità del singolo. Il viaggio è iniziato a dicembre in ambito linguistico con la lettura e riflessione emotiva del libro *Pezzettino*, da cui sono scaturite le prime scoperte dei bambini sul valore del singolo e sull'accettazione della diversità, cuore dell'Educazione Civica. Dallo storytelling e riflessioni verbali si è passati alla partecipazione attiva attraverso un'attività di Educazione all'Immagine, dove la manipolazione dei pezzettini di carta ha dato vita a un cartellone di gruppo realizzato in condivisione. In primavera, il percorso è approdato all'attività topologica e informatica: prima lavorando su spazio, reticolo e coordinate, per poi passare ad attività di rappresentazione simbolica, codifica e decodifica. Infine la trasposizione digitale è culminata nel formato podcast: inserendo la propria voce in un lavoro di squadra anche con linguaggi diversi, ogni "pezzettino" ha trovato la sua perfetta collocazione collettiva: ogni pezzettino, unico e diverso, è diventato così parte attiva di un originale e inclusivo "capolavoro" comune.



Figura 1: opera collettiva ispirata a Pezzettino

2 Traguardi e obiettivi di riferimento

2.1 Nuove indicazioni Nazionali (2025)

Obiettivi generali al termine della scuola primaria (matematica-scienze, tecnologie)

4. Organizzare informazioni con diversi strumenti: organizzare informazioni relative a un problema utilizzando liste, semplici mappe concettuali, tabelle e grafici elementari per visualizzare e comprendere i dati.

Obiettivi specifici di apprendimento per Informatica:

- Scegliere ed utilizzare oggetti o simboli per rappresentare informazioni. [...]
- Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto [...]

2.2 Proposta CINI (2017)

Traguardi per lo sviluppo delle competenze al termine della scuola primaria

- T-P-5. inizia a riconoscere la differenza tra l'informazione e i dati;
- T-P-6. esplora la possibilità di rappresentare dati di varia natura (numeri, immagini, suoni, …) mediante formati diversi, anche arbitrariamente scelti;

Obiettivi di apprendimento al termine della classe terza della scuola primaria

Ambito dati e informazione:

- O-P3-D-1. scegliere ed utilizzare oggetti per rappresentare informazioni familiari semplici (es. colori, parole, ...);
- O-P3-D-2. definire l'interpretazione degli oggetti utilizzati per rappresentare l'informazione (legenda).

3 Situazione problema

Un alunno ha chiesto all'insegnante: "Maestra, mi puoi aiutare? Sto costruendo una bella casa in Minecraft, ma la perdo ogni volta che gioco… Su questo libro c'è scritto che posso ritrovarla se mi segno le coordinate, ma io non so cosa sono!"

La situazione problema posta da questo bambino è stata colta dall'insegnante come opportunità didattica per motivare tutti all'apprendimento delle coordinate cartesiane, successivamente utilizzate come sistema simbolico per codificare informazioni e decodificarle nella riproduzione di immagini.

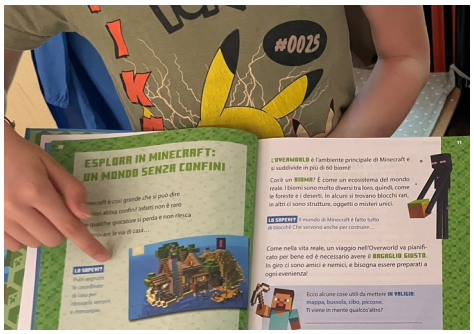


Figura 2: Situazione problema, coordinate in Minecraft

4 Attività svolte

4.1 Gioco dei palazzi e piani (coordinate)

Le coordinate sono state introdotte senza la presentazione del piano cartesiano, con l'uso del reticolo, modalità presente anche sul libro di testo di geografia, per avviare l'orientamento spaziale. Sulla linea orizzontale (asse ascisse, x) sono stati collocati degli ipotetici palazzi individuati con le lettere dell'alfabeto, mentre sulla linea verticale (asse ordinate, y) sono stati individuati i piani del palazzo mediante i numeri cardinali.
Questa prima attività esplorativa è stata svolta con "pezzettini" di carta colorata, quadratini di cm 5x5, collocati su un grande foglio quadrettato con griglia di riferimento.
Proseguendo con l'approccio ludico della "caccia al tesoro" nei diversi piani dei palazzi, l'attività è stata riproposta e maggiormente sistematizzata con l'ausilio della lavagna interattiva: gli alunni cercavano di individuare la posizione esatta di oggetti o animali presenti all'interno della griglia a video. In un secondo momento hanno iniziato a chiedere di disegnare animali e giochi in una determinata posizione. Il gioco si è evoluto e, in autonomia, i bambini posizionavano elementi nelle caselle, trovavano gli oggetti, indicando le coordinate (palazzo e piano) al compagno disegnatore di turno.

4.2 P di Pezzettino multicolore (codifica)

Utilizzando i quadrati di carta colorata agli alunni è stato proposto di creare delle lettere multicolore sullo stile inconfondibile del libro "Pezzettino" di Lionni.

In un secondo momento, sistemando i pezzettini sulla griglia con lettere e numeri, immediata è stata la trasposizione didattica dell'uso delle coordinate per codificare la scelta del colore dei "pezzettini" da collocare in una determinata posizione.

A piccoli gruppi di tre alunni con tre ruoli diversi hanno realizzato numerose versioni della lettera P maiuscola di "Pezzettino", aventi sempre la stessa forma, ma variazioni cromatiche diverse. L'attività si è svolta con questa sequenza ordinata:

1. alunno "programmatore 1" sceglie un quadratino colorato tra tutti quelli presenti sul tavolo
2. alunno "programmatore 2" decide e indica la collocazione di quel determinato quadratino es. "Questo rosso mettilo in A;3"
3. alunno "esecutore" colloca il quadratino colorato seguendo le istruzioni date dal compagno

L'uso di quadratini colorati plastificati e dotati di velcro sul retro, pensato con finalità inclusiva per gli alunni con bisogni educativi speciali, si è rivelato molto funzionale ed apprezzato da tutti.



Figura 3: palazzi colorati nel reticolo e P di "Pezzettino" con velcro dietro

4.3 Gli amici di Pezzettino (rappresentazione simbolica, decodifica)

Per questa attività la classe è stata divisa in due sottogruppi, collocati in due distinte aule adiacenti.

Il primo gruppo, senza che gli alunni dell'altro gruppo vedessero, ha costruito l'immagine di un personaggio del libro "Pezzettino" disponendo i quadratini colorati su un una grande griglia con lettere sull'asse x e numeri sull'asse y.

Al termine del lavoro ai bambini del primo gruppo è stato chiesto di creare delle istruzioni affinché i bambini dell'altro gruppo potessero riprodurre, senza vederla, esattamente la stessa immagine, con stessa forma e stessi colori (codificare l'immagine). Questa richiesta ha aperto discussioni e riflessioni sulle modalità più funzionali e, per un primo approccio ai concetti informatici, è stata feconda.

Nel costruire le istruzioni, intese come strumento di procedura, i bambini hanno scelto di utilizzare le coordinate e i colori per fornire le informazioni all'altro gruppo. Sono sorti interrogativi su quali simboli utilizzare, se una lettera (indicazione del palazzo) fosse sufficiente per indirizzare in modo univoco i compagni o se servisse anche il secondo elemento (numero/piano), infine se il codice fosse chiaro e significativo per gli altri.

Ogni singolo quadrato colorato collocato sulla griglia è stato codificato (es. C;2 verde), affinché gli alunni del secondo gruppo potessero riprodurre l'immagine realizzata dal primo gruppo.

Le istruzioni sono state create in modalità cooperativa facendo scorrere il dito sulla griglia per agevolare l'individuazione delle coordinate di ogni quadratino, scritte in una lista, incollando subito accanto un quadratino del colore corrispondente a quello nell'immagine, via via la lista si allungava…



Figura 4-5: Figure su reticolo e codifica istruzioni

Con lo stesso approccio ludico e cooperativo la lista delle istruzioni è stata passata all'altro gruppo per fornire la procedura e spiegare ai compagni come riprodurre uno specifico personaggio (ad es. Quello - Forte)

I compagni del secondo gruppo, seguendo le istruzioni fornite dal primo gruppo, con curiosità per scoprire il personaggio e stupore per il risultato, hanno riprodotto l'immagine senza averla vista. Al termine si sono invertiti i ruoli dei due gruppi.

5 Metodologie

- Storytelling
- Impostazione laboratoriale
- Attenzione agli aspetti dell'inclusione (anche CAA)
- Lavoro a piccoli gruppi e peer tutoring
- Approccio ludico e cooperativo
- Focus sull'apprendimento attraverso l'esperienza
- Riflessione sia per accompagnare le attività sia al termine



Figura 6: Pezzettino in CAA

6 Verifica e valutazione

La verifica degli apprendimenti è stata effettuata in itinere, attraverso l'osservazione dei processi e la registrazione del progresso negli apprendimenti, con rubriche valutative, per descrivere in modo più articolato i diversi livelli di padronanza al fine di individualizzare i percorsi.

La valutazione è avvenuta ai sensi dell'O.M. n. 3 del 09/01/2025 mediante giudizi sintetici (ottimo, distinto, buono, discreto, sufficiente, non sufficiente), correlati alla descrizione dei livelli di apprendimento raggiunti (Allegato A dell'O.M.)

E'stata messa in atto l'ottica della valutazione per l'apprendimento, che, come scritto nelle Indicazioni del MIM in merito alla valutazione periodica e finale degli apprendimenti nella scuola primaria (23/01/2025), utilizza le informazioni rilevate anche per adattare l'insegnamento ai bisogni educativi concreti degli alunni e ai loro stili di apprendimento in una prospettiva di personalizzazione e valorizzazione dei talenti.

7 Sviluppi futuri

Possibili approfondimenti per la classe successiva:

- debugging (ob. Scrivere semplici programmi e verificare, mediante la loro esecuzione, se svolgono il compito previsto ed eventualmente correggerli).
- pixel e codifica digitale

8 Contributo delle autrici al progetto

Il coordinamento delle attività è stato affidato, per l'area informatica, all'insegnante Maurizia Gai. La collega Cinzia D'Elia ha progettato le attività che hanno coinvolto altre discipline: lingua italiana, educazione civica ed arte. Laura Galati ha curato l'attenzione all'inclusione e agli aspetti psicologici. Virginia Macrì ha collaborato occupandosi della realizzazione del podcast e del prodotto multimediale presentato a fine anno scolastico.

Si ringrazia Sara Capecchi del Dipartimento di Informatica dell'Università di Torino per gli stimoli e suggerimenti forniti.

Bibliografia

ITADINFO 2025 - Atti del convegno
https://www.itadinfo.it/2025/attidelconvegno/index.html

LONATI, Violetta, Informazioni, dati, codifica
https://lonati.di.unimi.it/didainfo_2021-22/informazioneDatiCodifica.pdf

LIONNI Leo, Pezzettino, ed illustrata, Babalibri (2013)

MIM - Decreto 9 dicembre 2025, n. 221 - Regolamento recante indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione.
https://www.gazzettaufficiale.it/eli/id/2026/01/27/26G00021/SG

MIM - O.M. n.3 del 9 gennaio 2025, Valutazione periodica e finale degli apprendimenti nella scuola primaria e valutazione del comportamento nella scuola secondaria di primo grado
https://www.mim.gov.it/-/valutazione-periodica-e-finale-degli-apprendimenti-nella-scuola-primaria-e-valutazione-del-comportamento-nella-scuola-secondaria-di-primo-grado

MORGESE Roberto, Colibrì classe 1 Primaria, Raffaello Scuola

L'Italia sul telo: l'informatica come mediatore interdisciplinare tra coding e creatività

Andrea Goia [1], Elena Esposito [2]
[1] EFT Piemonte / IIS Russell-Moro-Guarini, Torino – *goia.andrea@russell-moro-guarini.edu.it*
[2] IC La Loggia (TO) – *elena.esposito@iclaloggia.edu.it*

Abstract

Al fine di favorire lo sviluppo del pensiero critico e computazionale nella scuola primaria, il presente contributo documenta un'esperienza didattica laboratoriale nata in classe e realizzata con il supporto dell'EFT piemontese, all'interno del progetto DEAC-2026.

L'iniziativa è stata avviata con l'obiettivo di integrare l'apprendimento della geografia con i fondamenti dell'informatica, superando il classico approccio puramente mnemonico e nozionistico della disciplina.

Tramite il microcontrollore di tastiera del PC, l'utilizzo di materiali di recupero conduttivi e della programmazione a blocchi su Scratch, l'attività ha declinato in modo sinergico i traguardi di competenza previsti dalle Nuove Indicazioni Nazionali e dalle linee guida STEM, promuovendo inclusività, collaborazione e creatività digitale.

1 Introduzione

Le recenti evoluzioni normative del sistema scolastico italiano evidenziano con forza la necessità di strutturare percorsi didattici organici capaci di integrare le tecnologie digitali con senso critico e solide basi concettuali. Come puntualizzato nella versione definitiva delle Nuove Indicazioni Nazionali e in perfetta sintonia con le raccomandazioni del Consiglio dell'Unione Europea, l'informatica deve trovare spazio fin dall'inizio dell'istruzione obbligatoria: non si tratta di limitarsi ad addestrare gli studenti a un mero utilizzo passivo di dispositivi o software applicativi, ma al contrario, l'obiettivo primario è sviluppare una reale competenza informatica scientifica, intesa come la comprensione di come i dati vengono codificati ed elaborati in modo automatico da agenti meccanici guidati da algoritmi progettati dall'uomo.

In questo scenario, il framework culturale suggerito fin dal 2017 dalle proposte scientifiche della comunità informatica (come il Consorzio CINI) sottolinea l'importanza di stimolare sforzi di astrazione, precisione e organizzazione logica, infatti tali abilità contribuiscono in modo determinante alla strutturazione del pensiero critico, consentendo agli studenti di proporsi come *creatori di contenuti* e non solo come consumatori di tecnologia.

Il progetto "L'Italia sul telo", sviluppato in una classe quinta della scuola primaria di La Loggia (TO), incarna questa visione: la volontà di superare lo studio mnemonico della geografia, che spesso appiattisce la curiosità degli alunni impedendo loro di cogliere le relazioni profonde tra uomo e ambiente. Unendo il percorso didattico avviato in classe con le metodologie del *learning by doing* e del *tinkering*, dalla collaborazione tra la scuola e l'EFT piemontese, all'interno del progetto DEAC, è nata l'idea di realizzare un manufatto tecnologico interattivo.

2 Riferimenti Normativi e Sfondo Culturale

La progettazione dell'attività si colloca nell'alveo delle modifiche ordinamentali introdotte dal D.M. n. 14 del 30 gennaio 2024 e dal D.M. n. 183 del 7 settembre 2024 per l'educazione civica. I nuclei essenziali dell'intervento didattico rispondono direttamente alla richiesta di armonizzazione metodologica tra le discipline scientifico-tecnologiche (STEM) e l'area umanistica, potenziando la didattica laboratoriale e l'apprendimento esperienziale al fine di scongiurare un sovraccarico puramente nozionistico.

Nello specifico, le Nuove Indicazioni Nazionali tracciano una netta demarcazione tra:

- **Competenza Digitale:** Padronanza d'uso efficace, etico, sicuro e responsabile dei dispositivi di comunicazione e dei servizi digitali (in linea con il framework europeo DigComp 2.2 e 3.0);
- **Competenza Informatica:** Comprensione dei principi logici, dei concetti di algoritmo, esecuzione rigorosa, strutture di controllo, variabili e programmazione.

In linea con gli obiettivi specifici previsti al termine della scuola primaria, il percorso ha integrato pienamente i seguenti ambiti definiti dalla comunità scientifica ed educativa:

- **Ambito Algoritmi e Programmazione:** Esaminare un programma per capirne il comportamento, correggere errori (debug), ordinare correttamente sequenze di istruzioni e usare selezioni/condizioni logiche;
- **Ambito Dati e Informazioni:** Riconoscere l'organizzazione delle informazioni e comprendere la differenza tra rappresentazione analogica ed elaborazione simbolica digitale;
- **Ambito Consapevolezza Digitale:** Conoscere le principali componenti hardware e software dei dispositivi utilizzati, riflettendo sulla sostenibilità attraverso il riciclo creativo di materiali elettronici dismessi o risorse povere.

3 Sviluppo del Progetto Laboratoriale: Dall'Idea al Circuito

La realizzazione della mappa interattiva ha richiesto una forte estensione del monte ore inizialmente ipotizzato, tramutandosi in un vero e proprio ecosistema evolutivo e cross-disciplinare che ha trasformato un'attività d'aula in una continua sfida tecnica e ingegneristica che ha coinvolto i docenti e i ragazzi.

3.1 Ideazione spaziale e Problem Solving Fisico

Il punto di partenza è stato un grande telo di tessuto (delle dimensioni di un lenzuolo), sul quale è stata proiettata e disegnata la sagoma geografica dell'Italia suddivisa in regioni. Successivamente, ogni alunno ha ripassato i contorni della cartina e si è poi proceduto alla colorazione delle diverse aree.

La soluzione ha previsto l'applicazione dei principi della conduzione elettrica. Gli studenti hanno realizzato con l'aiuto e le indicazioni dei docenti, la parte posteriore del telo stendendo delle piste conduttive capaci di trasportare il segnale elettrico dai confini di ogni singola regione fino ai pin del microcontrollore di tastiera.

3.2 Sfide Tecniche: Materiali e Saldature

Nell'organizzare le attività, le sfide più complesse e stimolanti dal punto di vista del tinkering e della tecnologia hanno riguardato:

- **la modalità di interfacciamento tra mappa interattiva e PC:** sono stati valutati più metodi, da Makey Makey, ad Arduino (*Leonardo*), al controllore classico della tastiera. Alla fine si è scelto quest'ultimo, in modo da ottemperare alle indicazioni di Agenda 2030 sul riciclo;

- **la scelta e il fissaggio dei conduttori sulla superficie flessibile del tessuto:** il filo conduttivo, *prima scelta*, è risultato troppo complicato dai punti di vista della cucitura - non disponendo di una macchina da cucire professionale - e della reperibilità del materiale;
- si è escluso sin dal principio il normale cavo elettrico in rame che sarebbe risultato troppo pesante, rigido e/o di difficile gestione, rischiando di strappare il telo o creare cortocircuiti a causa dei movimenti del tessuto;
- il pensiero laterale ha fornito il risultato definitivo: dopo svariate sessioni di prove di materiali ed errori – *metodologia esplicitamente valorizzata dalle indicazioni STEM per stimolare l'intuizione e l'immaginazione creativa* – è stata trovata una soluzione non convenzionale, nella fattispecie, la fettuccia conduttiva da recinzione per alpeggi e pascoli di montagna, incollandola sul retro del telo;

per connettere stabilmente le fettucce di collegamento, che non erano facilmente saldabili, sono stati utilizzati:

- da un lato dei mammuth, garantendo la stabilità del contatto elettrico;
- dall'altro capo, dell'inchiostro conduttivo per assicurarsi la continuità elettrica;

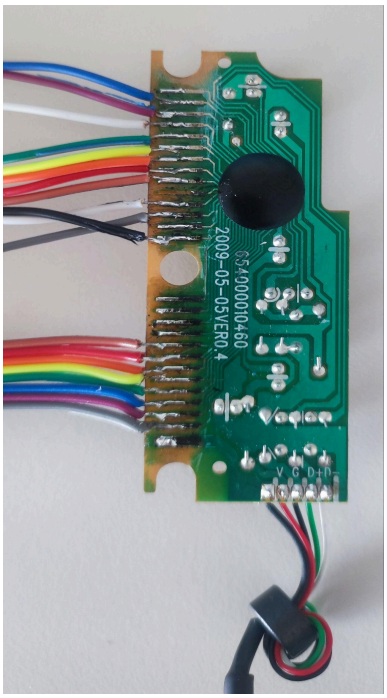


Figura 2: controllore di tastiera

Il tutto è poi stato collegato al controllore di tastiera 'modificata' tramite morsetti.



Figura 2: matrice righe-colonne

4 Implementazione Software: Il Coding su Scratch

Una volta completata l'infrastruttura hardware, l'interazione col PC è stata garantita collegando righe e colonne della matrice di tastiera tramite due poussoir: così facendo, mediante una barretta di metallo, li si cortocircuitano simulando così la pressione del tasto fisico selezionato.

Con l'aiuto degli insegnanti, in primo luogo è stato chiesto loro di stilare tre differenti liste: la prima contenente i tasti individuati, la seconda con le regioni d'Italia ed infine la terza con tutti i capoluoghi di regione, in posizioni corrispondenti: così facendo, gli elementi che occupano la stessa posizione risultano automaticamente associati tra loro. Per esempio, al primo tasto della prima lista corrispondono la prima regione della seconda e il primo capoluogo della terza. Conoscendo l'indice di una regione, è così possibile individuare nella stessa posizione il relativo capoluogo e rispondere a domande come: «*Qual è il capoluogo della Valle d'Aosta?*».

Il passo successivo ha visto gli alunni impegnati nell'attività corale di programmazione logica tramite Scratch: utilizzando in particolare i comandi "situazioni" in quest'ambiente, la classe ha sviluppato semplici algoritmi per gestire gli eventi di input; durante queste fasi, gli studenti hanno appreso in modo pratico i concetti di *variabile* (per tenere traccia del punteggio del gioco) e di lista (per memorizzare regioni e tasti) utilizzando anche cicli iterativi per gestire le situazioni ed hanno successivamente applicato procedure di *debugging* e *troubleshooting* per capire come mai, ad esempio, selezionando una determinata regione venisse attivato l'input di un'altra, risalendo all'errore logico nel codice o all'interferenza fisica nel circuito elettrico.

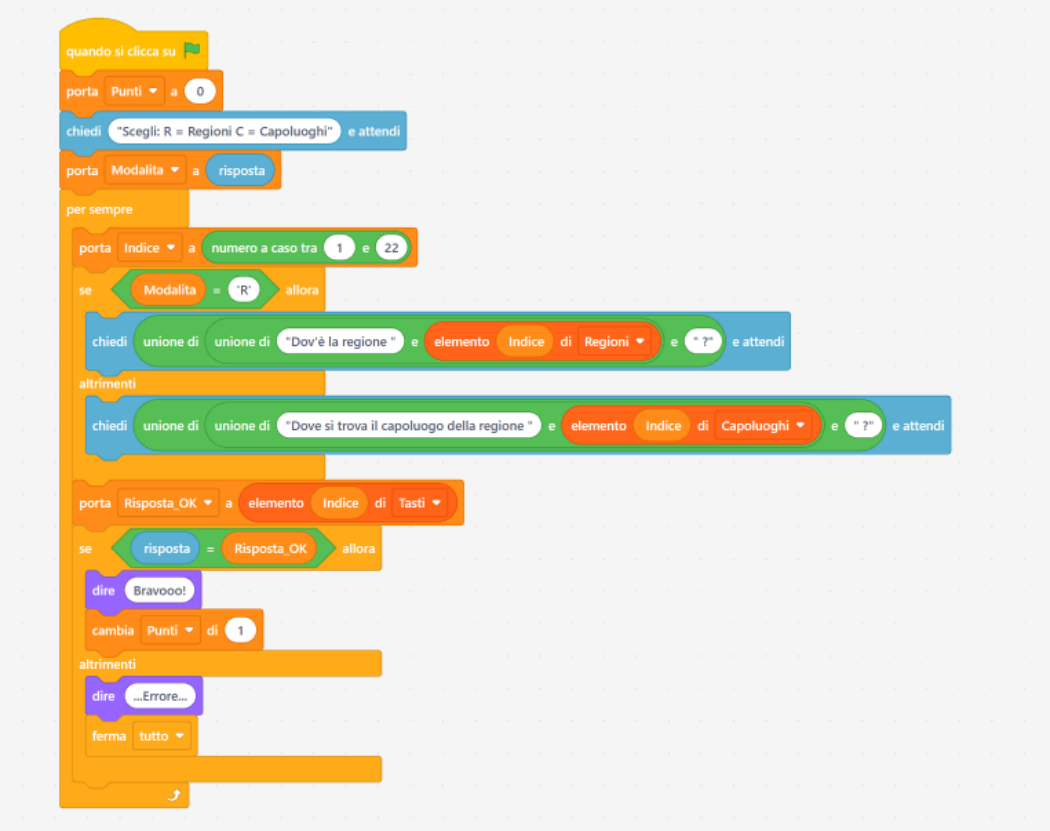


Figura 3: implementazione software

5 Valutazione dell'Inclusività e Risultati Didattici

L'impatto dell'attività sul gruppo classe ha ampiamente confermato le premesse metodologiche dell'apprendimento laboratoriale integrato STEM. Sotto il profilo strettamente pedagogico ed inclusivo, il progetto si è rivelato straordinariamente efficace nel catalizzare la partecipazione attiva di alunni con disabilità o con disturbi dell'attenzione.

La natura intrinsecamente multisensoriale del gioco – caratterizzata da una forte interazione fisica sul telo, dal ritmo dinamico delle domande del computer e dalla scomposizione dei compiti

in sotto-obiettivi pratici – ha permesso di mantenere sufficientemente costante l'ingaggio cognitivo di tutti i discenti. I ragazzi hanno sperimentato l'importanza dell'usabilità e dell'accessibilità delle applicazioni informatiche ai fini dell'inclusione sociale, operando in un contesto di apprendimento tra pari dove l'errore non era sanzionato, ma interpretato come un dato oggettivo da analizzare algoritmicamente per riprogettare la soluzione.

In conclusione, "L'Italia sul telo" ha dimostrato come discipline tradizionalmente ritenute puramente teoriche e contenutistiche, come la geografia, possano essere completamente reinventate se mediate dalle categorie concettuali dell'informatica. Gli studenti lavorando a tutto tondo, non hanno semplicemente e passivamente usato un computer per visionare dei contenuti ma hanno agito da veri '*ingegneri progettisti*', realizzando un ecosistema tecnologico interattivo, ecosostenibile, inclusivo e riutilizzabile, incarnando appieno lo spirito civico e scientifico delle Nuove Indicazioni Nazionali e di Agenda 2030.



Figura 4: a lavoro concluso

Bibliografia

[1] Ministero dell'Istruzione e del Merito (MIM). *Nuove Indicazioni Nazionali per il curricolo della Scuola dell'infanzia e delle Scuole del Primo ciclo di istruzione*. Versione definitiva del 9 dicembre 2025 (Pubblicazione in Gazzetta Ufficiale il 27 gennaio 2026).

[2] Gruppo Ingegneria Informatica e Gruppo Informatici - CINI. *Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola*. Dicembre 2017.

[3] Corradini, Isabella, and Enrico Nardelli. "Insights into Teachers' views on Informatics in school". *Proceedings of ITADINFO Conference*.

[4] Programma il futuro - Cittadinanza digitale consapevole: "Come funzionano i computer?" https://programmailfuturo.it/come/cittadinanza-digitale/come-funzionano-i-computer

[5] Esposito, Elena, e Goia, Andrea. "L'Italia sul telo, gli italiani dal vero: la geografia, davvero!". *BRICKS*, n. 2, 2026.

Funzioni di hash nella scuola primaria: una sperimentazione unplugged tra informazione, algoritmi e sicurezza digitale

Maria Cristina Carrisi[1,4], Michael Lodi[2,4], Simone Martini[2,4], and Maria Angela Pellegrino[3,4]

[1] Università degli Studi di Cagliari
mariacri.carrisi@unica.it
[2] Alma Mater Studiorum - Università di Bologna
{michael.lodi, simone.martini}@unibo.it
[3] Università degli Studi di Salerno
mapellegrino@unisa.it
[4] Laboratorio CINI Informatica e Scuola

Abstract

Presentiamo una sperimentazione in due classi quinte di scuola primaria sulle funzioni di hash. Attraverso cassetti, miscugli di colori e modelli numerici, gli alunni costruiscono in modo progressivo le idee di sintesi deterministica, collisione, lunghezza fissa, effetto valanga e non invertibilità, collegandole alla rappresentazione dell'informazione e alla sicurezza digitale.

1 Introduzione

Le funzioni di hash trasformano dati di dimensione variabile in impronte di lunghezza predeterminata e servono a indicizzare dati e costruire meccanismi di sicurezza.

La complessità matematica le rende però quasi assenti dalle proposte per la primaria. Ipotizziamo che esperienze concrete permettano di costruirne il significato come "sintesi dell'informazione", esplorandone progressivamente proprietà e limiti senza partire da definizioni formali.

La sicurezza informatica offre un contesto motivante: password, file scaricati, messaggi e servizi online appartengono al vissuto o all'immaginario degli alunni, anche quando ne ignorano il funzionamento. La didattica scolastica della crittografia ricorre spesso a cifrari classici, sostituzioni e giochi di decodifica. Proposte recenti affrontano anche hashing e crittografia moderna con ambienti digitali dedicati [7], ma si rivolgono ad allievi più grandi. Il nostro contributo, rivolto alla primaria, segue la tradizione delle attività unplugged [1] per insegnare l'Informatica come disciplina scientifica [5], introducendo concetti complessi attraverso "big ideas" accessibili [4] con un approccio basato sulla necessità [8].

Questo contesto permette di esplorare aspetti disciplinari della rappresentazione delle informazioni e della sicurezza informatica, andando oltre l'educazione all'uso sicuro della rete. Il percorso collega l'Informatica a matematica e cittadinanza digitale, conservandone però i fondamenti epistemologici senza ridurla a competenza trasversale.

2 Contesto e obiettivi

Presentiamo un primo report su una sperimentazione in due quinte di primaria con circa quaranta alunni, di cui quindici con bisogni educativi speciali di diverso tipo ed entità. Gli alunni non avevano seguito un curricolo strutturato di Informatica: avevano soltanto partecipato al Bebras (https://bebras.it), lavorando con un esperto esterno alla scuola.

Le attività vogliono veicolare le seguenti "big ideas" legate alle funzioni di hash:

- Un hash è una sintesi (un'impronta) di lunghezza fissa con perdita di informazione: l'input non è, in generale, ricostruibile univocamente.
- Negli hash crittografici, la non reversibilità è computazionale, non matematica: trovare un input compatibile con un'impronta data è teoricamente possibile ma deve essere computazionalmente impraticabile.
- Sono possibili collisioni (dati differenti hanno lo stesso hash), che si cerca di rendere estremamente improbabili e, quando richiesto, computazionalmente difficili da trovare.
- Piccole modifiche dell'input devono cambiare molto l'hash (effetto "valanga").
- Gli hash permettono di verificare l'integrità dei dati senza disporre dell'originale.

Un obiettivo informatico generale è sviluppare una postura critica: le strategie adottate soddisfano alcune proprietà, ma presentano limiti da superare con strategie più generali. Ciò richiama l'astrazione e la generalizzazione del problem solving e riguarda quindi anche l'area matematica.

Il percorso si collega alle nuove Indicazioni nazionali [3], lavorando su competenze, obiettivi e conoscenze al termine della quinta primaria:

- Informatica: differenza tra informazione e dati, uso di simboli per rappresentare dati strutturati, scrittura e analisi di algoritmi, uso sicuro della tecnologia informatica;
- Matematica: notazione posizionale, numeri espressi in basi diverse, resto della divisione intera.

3 Il percorso didattico

Il percorso si è basato su quattro incontri laboratoriali unplugged di due ore ciascuno.

L'attività è partita dal problema dei cassetti: collocare oggetti in vani numerati secondo una regola basata sul nome, per ritrovarli rapidamente senza memorizzare associazioni arbitrarie. Le proposte degli alunni, come l'uso dell'iniziale, hanno permesso di discutere determinismo e collisioni: oggetti diversi possono finire nello stesso cassetto. La collisione è stata presentata come limite informativo della regola, anziché errore. È stata introdotta anche la sintesi: rappresentare un'informazione con una più piccola comporta perdita di informazione e non reversibilità.

La seconda fase ha introdotto un modello a colori: ogni lettera è associata a un liquido colorato. Gli alunni ottengono l'impronta della parola miscelando in un bicchiere dosi uguali dei colori corrispondenti alle lettere, prelevate con siringhe graduate. Il miscuglio è una sintesi visibile, riconoscibile, confrontabile e di lunghezza fissa: un solo colore, indipendentemente dalla lunghezza della parola. L'input non è ricostruibile dal miscuglio; piccole variazioni della parola possono cambiare molto il risultato (Figura 1). Queste proprietà sono state osservate e dedotte dall'esperimento, poi formalizzate, senza definirle a priori. Proposti alcuni anagrammi, gli studenti hanno intuito che le stesse lettere avrebbero prodotto lo stesso risultato cromatico. Hanno inoltre riconosciuto un'analogia con la notazione posizionale matematica: cambiare ordine ai simboli deve poter cambiare il valore della rappresentazione. Si è quindi passati a ottenere l'impronta miscelando dosi di colore diverse secondo la posizione delle lettere. I miscugli visivamente diversi (Figura 2) hanno reso visibile l'efficacia della strategia.

All'aumentare della lunghezza della parola, le impronte diventano cromaticamente più simili e difficili da distinguere a occhio nudo, pur restando fisicamente differenti (Figura 1). La terza fase ha quindi trasformato il modello fisico in uno numerico: la maggiore variabilità dei valori rende più evidente la distinzione tra miscugli anche per parole lunghe. Le lettere, codificate con numeri (A=0, B=1, ecc.),



Figura 1: Da sinistra verso destra gli hash delle parole MALE, PALE, PALME, PAMELA



Figura 2: Gli hash delle parole MALE e MELA, ora diversi tenendo conto della posizione delle lettere.

sono state prima sommate. Il riemergere delle collisioni ha motivato l'introduzione di pesi posizionali. Per facilitare l'interpretazione della codifica, le potenze del 10 sono state scritte esplicitamente, senza notazione polinomiale. Parole di lunghezza diversa possono produrre impronte di dimensioni diverse: si è quindi introdotto il resto della divisione intera per limitare lo spazio degli output. La classe ha così esplorato un'idea informatica: restringere questo spazio rende il confronto possibile o più efficiente, ma può introdurre collisioni se troppo ridotto. Si sono discussi casi di collisioni impossibili, improbabili o accettabili per un certo uso.

Nell'ultima fase sono state mostrate alcune applicazioni: controllo dell'integrità, verifica di password senza conoscerle direttamente, identificazione di duplicati e blockchain [6].

4 Osservazioni dalla sperimentazione

Le osservazioni qualitative derivano da note dei facilitatori, discussioni collettive e produzioni degli studenti. I colori hanno suscitato il maggiore coinvolgimento: la concretezza del miscuglio rendeva visibili le proprietà e offriva un oggetto comune su cui argomentare.

Il passaggio a strategie numeriche è stato delicato, poiché sono emerse difficoltà significative nei prerequisiti matematici. Queste hanno riguardato aspetti apparentemente procedurali (errori nell'esecuzione della divisione intera, nello svolgimento di semplici potenze del 10 e nella scomposizione e ricomposizione di un numero nella sua forma polinomiale) che, a un'analisi più approfondita, si sono rivelati di natura concettuale. Alcuni studenti sostituivano la divisione intera con quella decimale, come se fosse un metodo migliore anziché una procedura distinta, senza riconoscere i diversi domini di validi-

tà e applicazione. In altri casi, pur seguendo correttamente i passaggi dell'algoritmo, non individuavano errori di calcolo evidenti, come un resto maggiore del divisore. Pochi studenti mostravano consapevolezza della limitatezza e ciclicità del resto. Altre difficoltà hanno riguardato la comprensione del valore posizionale di unità, decine, centinaia e migliaia, soprattutto nella ricomposizione dei numeri. Lo 0, codice della lettera A, era particolarmente critico: l'assenza di un contributo in una posizione rendeva meno immediato il passaggio tra forma posizionale e sintetica. Queste stesse difficoltà hanno però anche generato domande informaticamente rilevanti: perché servono zeri non significativi per mantenere un formato fisso? Vanno aggiunti a destra o a sinistra? Anche virgola e spazio vanno codificati? Le domande hanno evidenziato il tema della rappresentazione dei dati, oltre alla sicurezza.

Un secondo elemento significativo riguarda la partecipazione: il ricorso più esplicito alla matematica ha fatto emergere difficoltà legate anche alla dimensione affettiva (*affect*), intesa come atteggiamenti, convinzioni ed emozioni verso la disciplina. Le fasi manipolative e di discussione hanno consentito ad alcuni alunni di assumere progressivamente un ruolo più attivo e acquisire maggiore fiducia anche nei calcoli. Pur non generalizzabile, il risultato suggerisce che rappresentazioni plurali e problemi aperti possano rendere più accessibile un argomento percepito come tecnico. Data la brevità dell'intervento, la maggiore partecipazione non si è tradotta in un miglioramento generalizzato delle performance.

Per un efficace inserimento nel curricolo, è importante consolidare i prerequisiti matematici affinché gli studenti affrontino serenamente gli aspetti numerici, che permetterebbero una successiva implementazione delle attività in ambienti di programmazione. È inoltre fondamentale il supporto degli insegnanti di sostegno: conoscendo i bisogni specifici degli studenti, possono contribuire allo svolgimento delle attività e alla progettazione delle personalizzazioni necessarie.

5 Discussione e conclusioni

La sperimentazione suggerisce che le funzioni di hash possano essere introdotte nella primaria se trattate come oggetti informatici con proprietà da costruire progressivamente, anziché mediante definizioni formali. L'attività ha infatti evidenziato come la formalizzazione possa emergere dall'esperienza: gli studenti incontrano problemi concreti e limiti operativi, poi riconoscono e nominano le proprietà che rendono una soluzione più efficace o robusta delle alternative.

La sequenza "regola–limite–nuova regola" ha permesso agli alunni di dare senso alle proprietà desiderate: determinismo, sintesi, lunghezza fissa, sensibilità alle variazioni, difficoltà di inversione e inevitabilità delle collisioni in spazi finiti.

Il percorso ha valore formativo perché esplora concetti fondanti dell'Informatica e sviluppa un atteggiamento di analisi e revisione delle strategie adottate. Le applicazioni hanno soprattutto funzione motivazionale. Il contributo distingue così i contenuti propri dell'Informatica dagli aspetti multidisciplinari. La matematica entra come linguaggio necessario, non come sostituto dell'Informatica che costituisce terreno epistemologico [2] per la comprensione e l'attribuzione di significato di concetti matematici.

Le difficoltà matematiche osservate, e documentate in letteratura, non indicano un'inadeguatezza del percorso alla fascia d'età, ma la necessità di continuare a consolidare le conoscenze dell'aritmetica dei numeri interi, sviluppate nella primaria e riprese nella secondaria. Tali competenze sono essenziali non solo all'interno della matematica, ma anche per comprendere modelli e meccanismi di altri domini.

I prossimi passi riguardano la produzione di materiali replicabili, una valutazione sistematica degli apprendimenti ed il potenziamento dei prerequisiti matematici.

La proposta appare adatta anche alla formazione insegnanti, perché rende osservabili sia le idee informatiche sia i punti in cui la progettazione didattica deve sostenere il passaggio dal concreto al formale.

Riferimenti bibliografici

[1] Tim Bell and Michael Lodi. Constructing computational thinking without using computers. *Constructivist Foundations*, 14(3):342–351, 2019. https://constructivist.info/14/3/342.bell.

[2] Maria Cristina Carrisi, Giorgia Nieddu, and Maria Polo. The epistemic role of computer science in mathematics education: The case of cyclicity in modular arithmetic. In *Proceedings of the 18th ERME Topic Conference MEDA 5*, 2026. to be published.

[3] Ministero dell'Istruzione e del Merito. Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione, 2025. https://www.mim.gov.it/documents/20182/10554370/curricolo_web.pdf/f91c31a0-5ed4-65f3-bfea-fb49adaba55f?version=1.0&t=1773224873548.

[4] Michael Lodi, Maria Cristina Carrisi, and Simone Martini. Big ideas of cryptography in primary school. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2024, pages 206–212, New York, NY, USA, 2024. Association for Computing Machinery. https://doi.org/10.1145/3649217.3653548.

[5] Violetta Lonati. Sull'informatica come scienza e sul perché insegnarla nella scuola del primo ciclo. *OPPInformazioni*, 128:77–85, 2020. https://oppi.it/wp-content/uploads/2021/01/77-85_lonati.pdf.

[6] Maria Angela Pellegrino and Lorenzo Guasti. Introducing blockchains at school without computers: Hands-on sense-making for young learners. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, CHI 2026, pages 1–21. ACM, 2026. https://doi.org/10.1145/3772318.3791808.

[7] Pranathi Rayavaram, Onyinyechukwu Ukaegbu, Maryam Abbasalizadeh, Krishnaa Vellamchetty, and Sashank Narain. CryptoEL: A novel experiential learning tool for enhancing K-12 cryptography education. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE TS 2025, pages 980–986. ACM, 2025. https://doi.org/10.1145/3641554.3701926.

[8] Marco Sbaraglia, Michael Lodi, and Simone Martini. A Necessity-Driven Ride on the Abstraction Rollercoaster of CS1 Programming. *Informatics in Education*, 20(4):641–682, 2021. https://doi.org/10.15388/infedu.2021.28.

La ricorsione con un albero

Giovanni Bellucci
Istituto Comprensivo Statale Lucio Voluseno
giovanni.bellucci@icvoluseno.edu.it

Abstract

L'esperienza riportata in questo contributo tratta la strategia didattica con la quale è stato affrontato l'argomento della ricorsione in una classe di terza media. L'idea alla base è stata quella di utilizzare degli algoritmi costituiti da pochissimi blocchi e che quindi all'apparenza risultassero quasi banali per gli studenti. Dallo studio dell'esecuzione del codice invece doveva emergere un comportamento inaspettato, che stupisse lo studente e portasse ad un risultato finale sorprendente. Si vuole mostrare come, dietro l'eleganza di poche righe di codice, si celi una complessità che può affascinare lo studente e stimolare dunque la sua curiosità nel capire il funzionamento, in questo caso, della ricorsione. Dopo un semplice algoritmo che disegna una spirale, è stato ampiamente analizzato il codice che disegna un albero. Per suscitare nello studente l'interesse ad approfondire l'argomento, sono stati poi proposti due algoritmi che hanno la stessa struttura di quello che disegna l'albero, ma che riproducono la curva di Lévy e la curva del Drago. Si propone infine come sviluppo futuro un approccio alla ricorsione orientato alla risoluzione di problemi, basato sul riconoscimento del caso base e sul salto di fede induttivo.

1 Introduzione

La ricorsione è un concetto spesso percepito come complesso, di difficile assimilazione da parte degli studenti [1] e questo è il motivo che ha spinto a cercare una approccio visuale che possa renderlo maggiormente accessibile. Il primo esempio proposto è stato quello di una semplice spirale che si viene a creare da una funzione che chiama sé stessa. Dall'esperienza maturata anche negli anni scolastici passati si evidenzia come esempi di questo tipo, con una sola chiamata a funzione all'interno della stessa, sono sin dalla prima lezione compresi dagli studenti. Quando però si passa a due chiamate a funzione (sempre all'interno della funzione stessa) nessuno studente riesce a predire l'esecuzione del codice, rendendo evidenza di non aver compreso il concetto di ricorsione. Con l'esempio dell'albero si è voluto approfondire tale concetto a partire dal caso base e allontanando via via la condizione di arresto. Tale attività ha impiegato quattro lezioni ciascuna costituita da due blocchi di 45 minuti, per un totale di 6 ore effettive. A conclusione dell'esperienza si è voluto lasciare due esempi, la curva di Lévy e la curva del Drago, che mostrassero risultati finali diversi [2] a partire dalla stessa struttura concettuale dell'albero.

2 Il contesto

L'Istituto Comprensivo Lucio Voluseno si articola su due plessi, uno nel comune di Badia Tedalda e uno nel comune di Sestino. Come tante scuole periferiche ha una popolazione scolastica il cui numero rende necessarie le pluriclassi. A partire dall'A.S. 2023/24 l'Istituto ha richiesto una cattedra di potenziamento di tecnologia [3] con la quale sono stati svolti progetti di alfabetizzazione informatica a partire dalla scuola primaria. In particolare dalla classe quarta della scuola primaria si è iniziato a far familiarizzare i bambini con i concetti della programmazione, non tanto nell'implementazione del codice, quanto nei concetti che stanno alla base di algoritmi e strutture dati. Grazie a ciò, c'è già un buon livello di partenza degli studenti alla Scuola Secondaria di Primo Grado e in uscita dalla classe terza di fatto gli studenti hanno così seguito un percorso di 5 anni continuativo sull'informatica. Ogni anno partecipano con successo a varie iniziative come i Giochi di Fibonacci, le Olimpiadi di Problem Solving e i Bebras dell'informatica.

3 Descrizione dell'attività

Inizialmente, per far ragionare gli studenti su cosa succede quando una funzione chiama sé stessa, senza mostrare nemmeno il programma a blocchi è stato spiegato a parole il funzionamento con la seguente metodologia. Si è spiegato oralmente che la funzione myfun prende in input un parametro che chiameremo L ed è costituita da queste 3 operazioni: fare L passi, ruotare di 90 gradi in senso antiorario, chiamare la funzione myfun passando come parametro L-5. Poi sono stati chiamati due studenti: uno, il narratore, con il compito di dire ad alta voce l'istruzione che veniva eseguita dall'algoritmo un passo alla volta e l'altro, l'interprete, che doveva ascoltarlo ed eseguire ogni singola istruzione spostandosi fisicamente dentro l'aula. Il momento di indecisione c'è stato sia da parte dell'interprete che non sapeva dove andare quando il narratore faceva la chiamata a funzione sia da parte del narratore che non sapeva dove lo portava quella chiamata. Grazie anche all'aiuto della classe si è superata l'impasse; questo è stato un momento veramente formativo per i ragazzi in quanto si sono trovati davanti ad una situazione nuova ed hanno dovuto ragionare per trovare la soluzione. Successivamente è stato assegnato alla classe il compito di implementare quanto appena fatto in codice [4] come riportato in Figura 1. In questa fase un momento di apprendimento significativo è stato quello di capire la necessità di una condizione di arresto (caso base), messa poi in pratica includendo le tre istruzioni all'interno di un if.

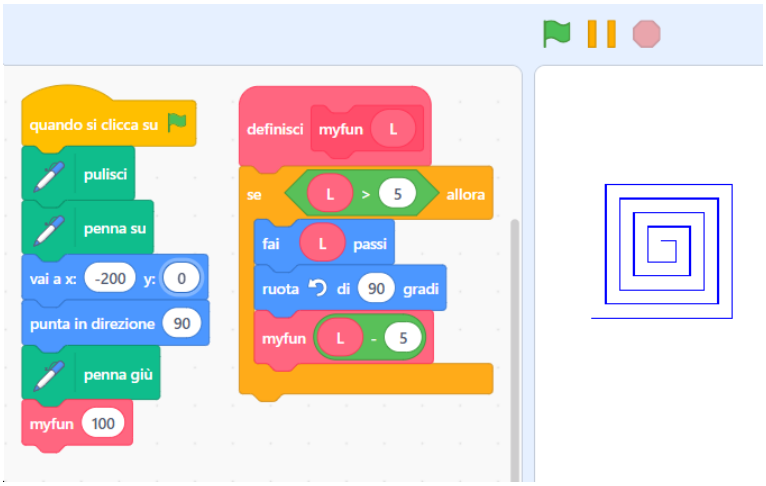


Figura 1: Primo algoritmo proposto per introdurre la ricorsione

Come compito da svolgere per la lezione successiva è stato assegnato quello di provare, con la stessa metodologia, a descrivere il funzionamento dell'algoritmo riportato in Figura 2. Anche in questo caso la funzione è definita chiamando sé stessa, solo che questa volta oltre alle altre istruzioni le chiamate sono due. L'introduzione delle due chiamate a funzione ha reso difficile per gli studenti capire il nuovo flusso di esecuzione del programma, ragion per cui si è ritenuto necessario procedere con un approccio visuale a condizioni di arresto via via maggiori. In Tabella 1 si riporta la schematizzazione che è stata costruita un po' alla volta alla lavagna con la classe. Si sono ripercorse le istruzioni che il programma svolge nella sua esecuzione, le chiamate e relativi punti del programma in cui ciascuna chiamata ritorna. Si inizia sempre con la prima chiamata a funzione e si continua a richiamarla fino a che non si raggiunge la condizioni di arresto. A questo punto abbiamo raggiunto il caso base e in ogni schema viene evidenziato con sfondo giallo tale blocco di istruzioni. Infatti, terminando le nuove chiamate, vengono eseguite tutte le istruzioni all'interno dell'if e si ritorna al ramo precedente; successivamente quando sarà richiamata la funzione con lo stesso parametro, invece di riportare nuovamente tutte le istruzioni, viene colorata una cella in giallo al loro posto.

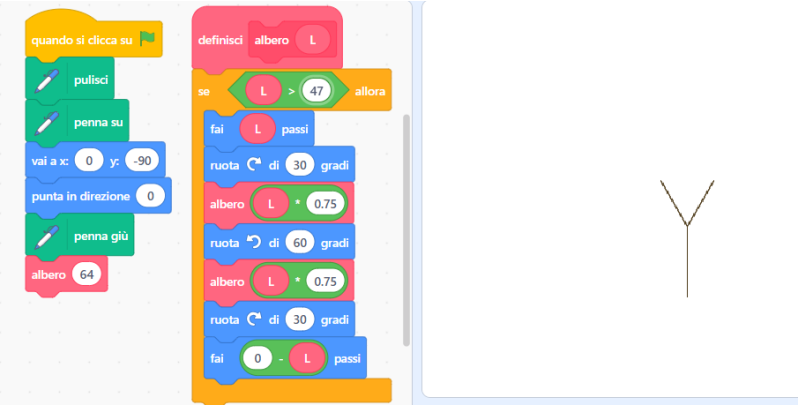


Figura 2: Algoritmo che genera l'albero (arresto alla seconda chiamata)

myfun(64)		
⇧ **64** ↱		
myfun(48)		
	⇧ **48** ↱	
	myfun(36)	Termina
	↰↰	
	myfun(36)	Termina
	↱ ⇩ **48**	
↰↰		
myfun(48)		
↱ ⇩ 64		

Tabella 1: Chiamate e ritorni del processo di ricorsione

In figura 3, in figura 4 e nelle tabelle sottostanti sono riportati, analogamente a quanto visto sopra, il codice, il risultato grafico e lo schema delle chiamate rispettivamente con arresto alla terza e quarta chiamata. Analogamente a quanto fatto con lo sfondo giallo, si è voluto evidenziare con la bordatura rossa e verde un pattern ricorrente anche per le altre chiamate che si ripetono. Infine in figura 5 si riporta il risultato che si ottiene con condizione di arresto ad una lunghezza non superiore a 6 unità. Una volta compreso l'algoritmo, obiettivo di apprendimento anche delle Nuove Indicazioni Nazionali per il primo ciclo [5], sono state proposte due varianti che necessitavano di modificare la funzione in modo da accettare in input anche un altro parametro che rappresentasse l'angolo di rotazione in gradi.

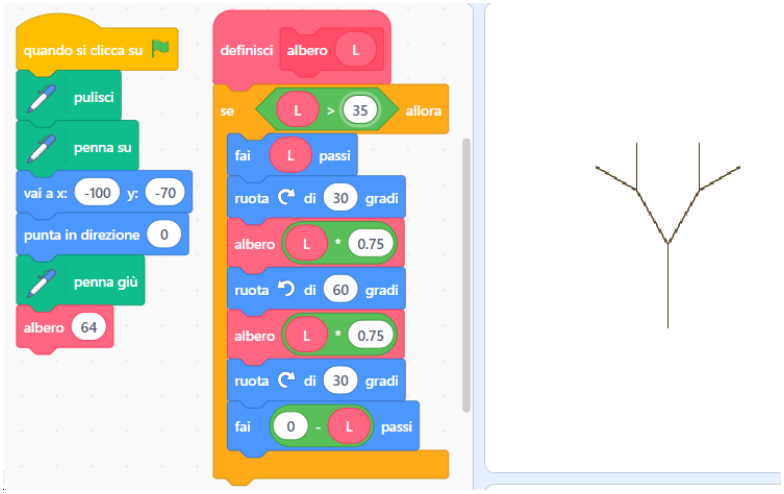


Figura 3: Algoritmo che genera l'albero (arresto alla terza chiamata)

myfun(64)			
⇧ **64** ↱			
myfun(48)			
	⇧ **48** ↱		
	myfun(36)		
		⇧ **36** ↱	
		myfun(27)	Termina
		↰↰	
		myfun(27)	Termina
		↱ ⇩ **36**	
	↰↰		
	myfun(36)		
	↱ ⇩ 48		
↰↰			
myfun(48)			
↱ ⇩ 64			

Tabella 2: Chiamate e ritorni del processo di ricorsione

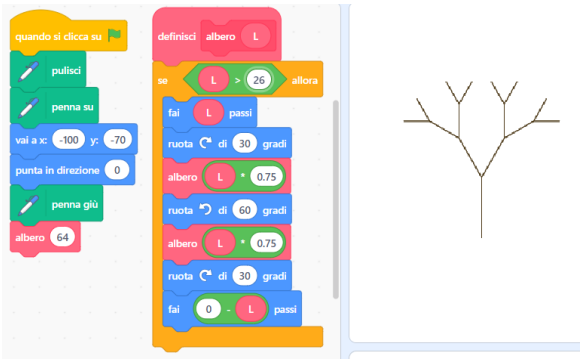


Figura 4: Algoritmo che genera l'albero (arresto alla quarta chiamata)

myfun(64)				
⇧ **64** ↱				
myfun(48)				
	⇧ **48** ↱			
	myfun(36)			
		⇧ **36** ↱		
		myfun(27)		
			⇧ **27** ↱	
			myfun(20.25)	Termina
			↰↰	
			myfun(20.25)	Termina
			↱ ⇩ **27**	
		↰↰		
		myfun(27)		
		↱ ⇩ **36**		
	↰↰			
	myfun(36)			
	↱ ⇩ 48			
↰↰				
myfun(48)				
↱ ⇩ 64				

Tabella 3: Chiamate e ritorni del processo di ricorsione

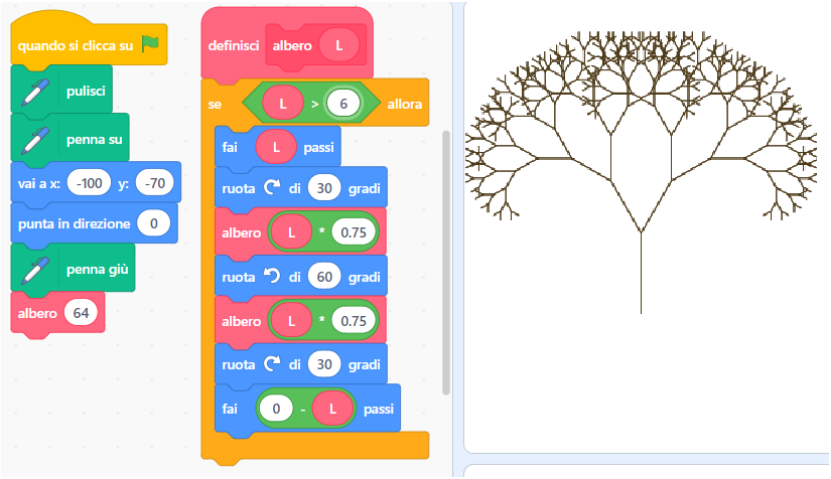


Figura 5: Algoritmo che genera l'albero

Potendo passare come parametro alla funzione anche l'angolo, attraverso un ciclo for è stata creata un'animazione che fa aprire e chiudere, come un ventaglio, i rami dell'albero. Inoltre si è richiesto di creare, ad ogni esecuzione del codice, un albero con ramificazioni sempre diverse. Questo è stato possibile facendo variare sia la lunghezza dei rami che l'angolo di biforcazione in maniera pseudocasuale all'interno di un intervallo prestabilito. Infine è stato mostrato agli studenti, figura 6 e 7, come rimanendo con un codice molto compatto e simile nella struttura a quello dell'albero si potessero creare geometrie diverse come la curva di Lévy e la curva del Drago.

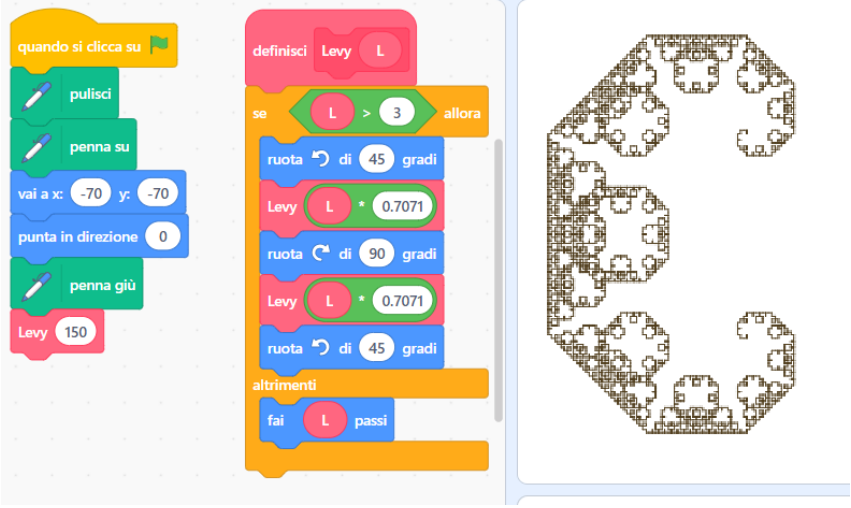


Figura 6: Algoritmo che genera la curva di Lévy

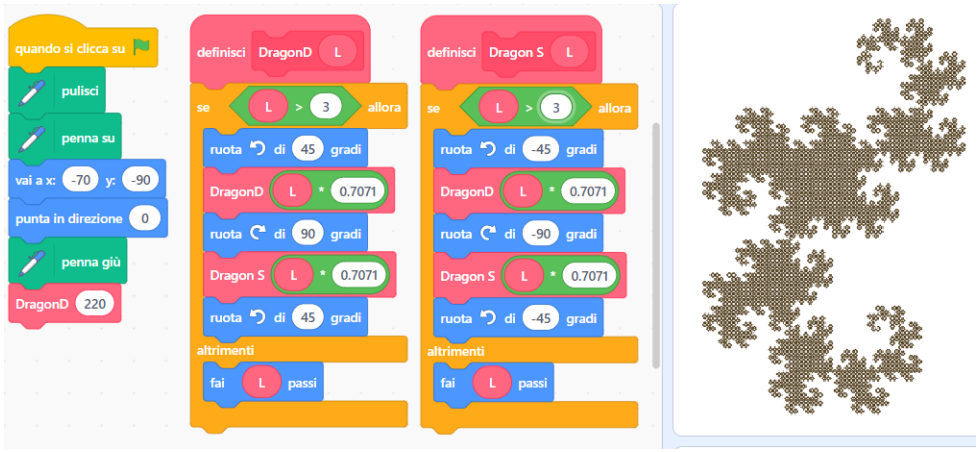


Figura 7: Algoritmo che genera la curva del Drago

Grazie alle attività didattiche proposte negli anni scolastici precedenti si è constatato che il riscontro da parte degli studenti dipendesse non tanto dall'argomento informatico affrontato quanto dalla metodologia didattica utilizzata. Considerate la diverse modalità con cui gli studenti si avvicinano ai nuovi concetti, non è mai l'utilizzo di una sola metodologia a rendere efficace l'intervento didattico quanto l'unione di più approcci. In particolare per quest'attività la drammatizzazione è stata fondamentale per introdurre l'argomento della ricorsione mantenendo sempre alta l'attenzione degli studenti. Questa fase, che nello schema Engage-Study-Activate (ESA) [6] corrisponde al momento dell'engage, è cruciale per l'apprendimento successivo: deve incuriosire e renderli parte attiva della scoperta, prima ancora di fornire loro gli strumenti formali (study) per padroneggiare il concetto e metterlo in pratica (activate). Se ad esempio si tratta di spiegare una famiglia di algoritmi è bene cercare esempi che risolvano problemi tratti dalla loro vita quotidiana e che saranno diversi a seconda del contesto classe. Nello svolgimento dell'attività gli studenti hanno mostrato una crescente autonomia nel completare le tabelle delle chiamate. Come atteso in un gruppo classe eterogeneo, il livello di approfondimento è stato differenziato in base alla predisposizione dei singoli alunni: mentre la maggior parte della classe ha raggiunto una comprensione solida del meccanismo di chiamata e ritorno, alcuni studenti hanno mostrato particolare interesse nel tentare di costruire autonomamente varianti dell'algoritmo e nell'esplorarne il funzionamento. Tale predisposizione ha trovato conferma successivamente anche in contesti extracurricolari. Nel corso dello stesso anno scolastico infatti quattro studenti si sono qualificati alla finale nazionale dei Giochi di Fibonacci, conseguendo una medaglia di bronzo. Pur non essendo possibile attribuire tale risultato a una singola attività didattica, esso costituisce un riscontro coerente con lo sviluppo delle competenze logiche e di modellizzazione coltivate lungo l'intero percorso di alfabetizzazione informatica. Parallelamente, le competenze acquisite durante questo percorso hanno trovato continuità anche nelle attività curricolari successive. In particolare, gli studenti hanno mostrato una maggiore familiarità nel ragionare sul flusso di esecuzione degli algoritmi e in generale nell'analizzare problemi attraverso la loro scomposizione in sottoproblemi.

4 Conclusioni

L'attività è stata apprezzata dagli studenti in quanto hanno potuto sia comprendere il concetto della ricorsione sia esprimere la loro creatività nella costruzione di varianti degli algoritmi proposti. Il fatto di giocare con il codice, cercare di cambiarne il funzionamento, romperlo,

ricostruirlo [7] è tra gli aspetti più apprezzati anche dai docenti, potendo far mettere in pratica ai propri studenti competenze matematiche e logiche in un contesto motivante. Quest'attività è stata poi la base per introdurre nella parte finale dell'anno scolastico la programmazione dinamica. Proporre la ricorsione o la programmazione dinamica alla scuola secondaria di primo grado potrebbe sembrare un'anticipazione eccessiva in quanto concetti astratti e complessi per l'età degli studenti e per questo è importante evitare la loro formalizzazione precoce. Questi concetti infatti saranno ripresi, negli anni scolastici successivi, più volte con diversi livelli di profondità e consapevolezza. Analogamente all'insegnamento della matematica alla scuola dell'infanzia [8], il problema non è "fare prima" (anticipare contenuti che troveranno una trattazione formale solo più avanti), ma proporre ciò che è cognitivamente ed emotivamente adatto a quell'età, stimolando l'intuizione dello spazio, il gioco strategico, l'argomentazione e la formulazione di ipotesi, attraverso il gioco e l'esplorazione come canali principali. Allo stesso modo, la didattica dell'informatica deve proporre argomenti che nel loro insegnamento non si riducano all'uso di specifici dispositivi o alla memorizzazione di sintassi di programmazione, bensì attivino meccanismi logici e linguistici tramite attività laboratoriali, contesti narrativi e situazioni di problem-solving vicino al vissuto degli studenti. In questa prospettiva, proporre ricorsione e programmazione dinamica nella scuola secondaria di primo grado non significa anticipare contenuti universitari fornendo una precoce formalizzazione di concetti astratti ma costruire esperienze significative attraverso una loro opportuna trasposizione didattica, privilegiando intuizione, esplorazione, argomentazione e costruzione progressiva dei significati.

5 Sviluppi futuri

Come prospettiva didattica futura si propone il passaggio dalla *comprensione* dell'esecuzione ricorsiva alla *progettazione* di soluzioni ricorsive attraverso il seguente flusso: risolvere il caso base, assumere per ipotesi ("Leap of Faith" [9]) di saper risolvere un caso generico, costruire la soluzione del caso successivo a partire da quella del caso generico. L'attività descritta infatti lavora sulla *lettura* della ricorsione e il passo successivo naturale sarebbe quello di guidare gli studenti nella *scrittura* di funzioni ricorsive per problemi semplici (ad esempio la somma di una lista, la ricerca del massimo, la Torre di Hanoi, ecc...), dando modo così di vedere la ricorsione anche come strategia di problem solving.

Bibliografia

[1] Dann, W. P., S. Cooper, and R. F. Pausch. "Using visualization to teach novices recursion." ACM SIGCSE Bulletin 33.3, pp. 109-112 (2001).

[2] Cozza, Pasquale, Viviana Andreotti, and Vittoria Cozza. "Percorso didattico sulla ricorsione dalla natura al coding." Mondo Digitale, Volume 15, Fascicolo 64 (2016). https://www.researchgate.net/publication/309578166_Percorso_didattico_sulla_ricorsione_dalla_natura_al_coding Accessed: 2026-06-01

[3] Legge 13 luglio 2015, n.107, Riforma del sistema nazionale di istruzione, art.1, comma 7: iniziative di potenziamento dell'offerta formativa, Gazzetta Ufficiale n.162, 15-07-2015.

[4] Miller, Brad, David Ranum, and Janice L. Pearce. "Introduction: Visualizing Recursion." Problem Solving with Algorithms and Data Structures using C++. Runestone Academy, (2026). https://runestone.academy/ns/books/published/cppds2/recursion_introduction-visualizing-recursion.html. Accessed: 2026-05-31

[5] Ministero dell'Istruzione e del Merito. "Indicazioni Nazionali per il curricolo - Scuola dell'infanzia e Scuole del Primo ciclo di istruzione." Decreto Ministeriale 9 dicembre 2025, n. 221, Gazzetta Ufficiale n. 21 del 27 gennaio 2026. Ministero dell'Istruzione e del Merito, 2026.

[6] Harmer, Jeremy. "How to Teach English." Longman, 1998.

[7] Lodi, Michael. "Didattica della programmazione e degli algoritmi - Argomenti avanzati falafel." Università di Bologna, (2026). https://www.cs.unibo.it/~michael.lodi2/csed2026/avanzi.pdf. Accessed: 2026-06-01

[8] Angeli, Anna, Bruno D'Amore, Mariangela Di Nunzio, e Elena Fascinelli. "La matematica dalla scuola dell'infanzia alla scuola primaria." Progetto: Matematica nella scuola primaria, percorsi per apprendere, Vol. 5. Bologna: Pitagora, (2011).

[9] Harvey, B. "Practical Recursion: the Leap of Faith." In Computer Science Logo Style, Volume 1: Symbolic Computing. MIT Press, 1997.

Lo sviluppo di app mobili come leva per l'acquisizione di competenze informatiche nella scuola secondaria di primo grado

Daniela Pieraccini
IC Anchise Picchi – Collesalvetti (LI)
daniela.pieraccini@scuola.istruzione.it

Abstract

Questo contributo analizza i contenuti disciplinari di Informatica emersi dallo sviluppo di nove applicazioni mobili realizzate con MIT App Inventor da studenti della scuola secondaria di primo grado, nell'ambito del programma Technovation Girls & Boys Italia. Attraverso l'analisi del codice a blocchi, il contributo documenta l'uso di variabili, strutture di controllo, debug e strutture di memorizzazione dati (TinyDB e TinyWebDB), riconducendo questi contenuti ai nuclei concettuali delle Nuove Indicazioni Nazionali per l'Informatica (DM n. 221/2025).

Due progetti integrano inoltre un classificatore di immagini basato su Teachable Machine, mentre altri impiegano un componente ChatBot, offrendo due paradigmi distinti di comprensione dei sistemi di intelligenza artificiale. Il percorso si è svolto sotto la guida di un mentor entro il curriculum ufficiale Technovation.

Parole chiave: variabili, strutture di controllo, debug, basi di dati, intelligenza artificiale, App Inventor, scuola secondaria di primo grado.

1 Introduzione e contesto

L'insegnamento dell'informatica nella scuola del primo ciclo è una sfida aperta: rendere accessibili concetti astratti a studenti giovani, mantenendo alta la motivazione e garantendo lo sviluppo di competenze computazionali. Un approccio promettente è il "learning by making" [2, 6]: apprendere costruendo artefatti digitali significativi. In questo quadro si colloca la partecipazione al programma Technovation Girls & Boys Italia: non un esercizio scolastico ma ingegneria applicata, in cui le squadre identificano un problema della comunità, sviluppano un'app mobile e la presentano a una giuria internazionale, diventando autori di tecnologia.

Il presente contributo documenta questa esperienza e ne analizza le implicazioni per la didattica dell'informatica, con l'obiettivo di offrire un modello replicabile ad altri docenti.

2 Struttura del percorso didattico

Il percorso si è svolto entro il curriculum ufficiale Technovation, con la docente nel ruolo di mentor, ovvero supporto senza sostituirsi agli studenti nelle scelte progettuali. Lo sviluppo è avvenuto in MIT App Inventor [4], scelto per l'accessibilità cognitiva, i componenti avanzati (GPS, database, fotocamera, AI) e l'immediatezza del feedback su dispositivo reale.
Le 12 unità previste dal curriculum [5] sono state raggruppate in quattro fasi (ideazione, prototipazione, sviluppo, presentazione) distribuite sull'anno scolastico. Hanno partecipato 9 squadre (circa 26 studenti, 11–14 anni), suddivise nelle categorie Girls e Boys secondo il regolamento Technovation.

3 Le applicazioni sviluppate

La Figura 1 sintetizza le nove applicazioni analizzate, evidenziando il tema affrontato e le principali tecnologie impiegate. La varietà dei domini applicativi riflette la libertà progettuale lasciata agli studenti nella scelta del problema da risolvere.

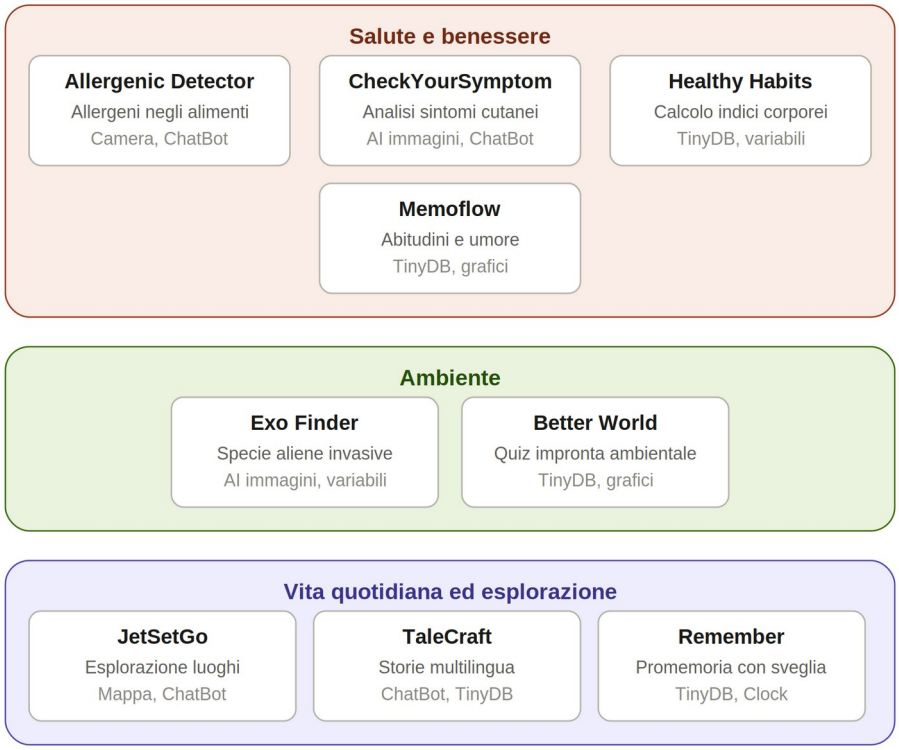


Figura 1: Sintesi delle applicazioni sviluppate dalle squadre Technovation

4 Contenuti disciplinari di Informatica e relazione con le Nuove Indicazioni Nazionali

Le Nuove Indicazioni Nazionali per il primo ciclo di istruzione (DM n. 221 del 9 dicembre 2025) [3] distinguono in modo netto le competenze digitali, intese come uso consapevole di dispositivi e strumenti, dalle competenze informatiche, riferite alla comprensione di come i dati possano essere elaborati automaticamente da un esecutore meccanico secondo procedure ideate dall'uomo. Per la scuola secondaria di primo grado, le Indicazioni richiedono in particolare la padronanza di algoritmo, variabili, strutture di controllo, moduli di programma e attività di debug. Questa sezione analizza nel dettaglio quali di questi nuclei concettuali sono stati effettivamente esercitati dagli studenti, isolandoli dagli aspetti trasversali (motivazione, comunicazione, lavoro di gruppo) trattati in chiusura del contributo.

4.1 Variabili globali e locali

L'analisi del codice a blocchi (file .bky) mostra un uso diffuso di variabili. In Exo Finder, una variabile globale "valore" riceve tramite controls_getStartValue un parametro passato da una schermata all'altra, poi confrontato con una costante (controls_if/else) per decidere il contenuto da mostrare: un esempio concreto di trasmissione di stato tra moduli di programma.

Nel progetto Healthy Habits, otto variabili globali (età, genere, peso, altezza, IMC, attività, lavoro) mantengono coerenti i dati raccolti tra schermate separate fino al calcolo automatico dell'indice di massa corporea, richiedendo di ragionare sul ciclo di vita e sulla persistenza della variabile. In TaleCraft, variabili dedicate al nome del protagonista e alla lingua personalizzano dinamicamente la richiesta al componente ChatBot. In Better World, l'uso combinato di variabili globali e locali (7 occorrenze ciascuna) permette un confronto diretto tra i due ambiti, conoscenza richiesta per la classe terza.

4.2 Strutture di controllo e debug

Tutte le applicazioni che gestiscono input utente (CheckYourSymptom, Exo Finder, Healthy Habits, Better World, JetSetGo) impiegano blocchi controls_if, spesso con ramo else, per selezioni condizionate: mostrare contenuti diversi in base a un classificatore AI, o convalidare i dati prima di procedere. Remember offre una variante: il blocco condizionale confronta non un input utente ma il valore di un componente Clock con l'orario salvato su TinyDB, attivando la sveglia solo a condizione verificata. La fase di test e correzione ha costituito un'occasione concreta di debug, in linea con le Indicazioni [3].

4.3 Memorizzazione dei dati

Il componente TinyDB è stato impiegato in cinque progetti (Healthy Habits, TaleCraft, Remember, Memoflow, Better World). In Healthy Habits, una singola istanza è condivisa tra sette schermate, ciascuna scrive un dato parziale poi letto congiuntamente nella sintesi finale: uno schema che ha reso concreta la differenza tra una variabile locale, valida solo nella schermata corrente, e una struttura persistente che sopravvive al cambio schermata e alla chiusura dell'app.

4.4 Integrazione di componenti di intelligenza artificiale

Due progetti (Exo Finder e CheckYourSymptom) integrano il componente TeachableMachineImageClassifier, che importa in App Inventor un modello addestrato autonomamente dagli studenti su Teachable Machine. Per Exo Finder, gli studenti hanno costruito il dataset fotografico delle specie invasive, addestrato il modello e collegato l'output del classificatore a una variabile che determina la logica di visualizzazione: un flusso completo che esemplifica la comprensione critica di un sistema di IA richiesta dalle Indicazioni. Altri progetti (Allergenic Detector, CheckYourSymptom, JetSetGo, TaleCraft, Memoflow) impiegano invece il componente ChatBot, offrendo un secondo paradigma di interazione con l'IA, generativo anziché classificatorio.

4.5 Sintesi della mappatura disciplinare

La Tabella 1 mappa ciascun nucleo concettuale del DM 221/2025 sui progetti in cui è stato esercitato, sulla base dell'analisi del codice.

Nucleo concettuale (DM 221/2025)	Progetti in cui è esercitato
Variabili (globali e locali)	Exo Finder, Healthy Habits, Better World, TaleCraft
Strutture di controllo (selezione condizionale)	CheckYourSymptom, Exo Finder, Healthy Habits, Better World, JetSetGo, Remember
Debug e correzione di errori	Tutti i progetti (fase di test iterativo)
Strutture di memorizzazione dati (locali e remote)	Healthy Habits, TaleCraft, Remember, Memoflow, Better World
Comprensione critica di sistemi di IA (classificazione)	Exo Finder, CheckYourSymptom
Comprensione critica di sistemi di IA (generativa/conversazionale)	Allergenic Detector, CheckYourSymptom, JetSetGo, TaleCraft, Memoflow

Tabella 1: Corrispondenza tra nuclei concettuali del DM 221/2025 e progetti analizzati

5 Riflessioni e implicazioni didattiche

5.1 Motivazione e coinvolgimento

La dimensione competitiva internazionale si è rivelata un potente moltiplicatore di motivazione: gli studenti hanno lavorato con un senso di scopo autentico, consapevoli che il prodotto sarebbe stato valutato da una giuria reale. Questo ha generato una cura nella progettazione e nel debugging superiore a quella osservabile in attività scolastiche tradizionali.

5.2 Riflessione sull'uso didattico dell'AI

L'uso di componenti AI ha stimolato domande critiche negli studenti — cosa può fare un classificatore, cosa succede con un dataset sbilanciato — emerse naturalmente durante il test: un'occasione autentica di valutazione critica dei sistemi di IA, coerente con quanto richiesto dalle Indicazioni.

5.3 Limiti e criticità

Il percorso presenta alcune criticità: il tempo richiesto, difficile da ricavare nel curricolo ordinario; la gestione di nove squadre nel ruolo di mentor unico, che richiede un forte presidio organizzativo extracurricolare; la presenza di almeno un dispositivo Android funzionante al fine di testare in diretta le app. Il confronto con altri mentor della rete Technovation aiuterà a individuare strategie di semplificazione.

6 Conclusioni

L'esperienza Technovation 2025–26 dimostra che studenti della secondaria di primo grado sono pienamente in grado di sviluppare app mobili con componenti di intelligenza artificiale, affrontando problemi reali con metodo ingegneristico. I risultati competitivi (9/9 squadre alla finale nazionale; 2 femminili su 4 in semifinale internazionale; 3 squadre con riconoscimento ufficiale) confermano la qualità del percorso, ma il valore didattico risiede soprattutto nel processo — iterare, fallire, correggersi, collaborare — e nelle competenze sia disciplinari che trasversali sviluppate (interfacce, lavoro di team, comunicazione tecnica).

Il modello è replicabile con un minimo di ore dedicate, dispositivi Android e volontà di affrontare con gli studenti la complessità di un progetto autentico. Molti studenti hanno continuato a sviluppare le proprie app oltre la scadenza, segno di un impatto che si estende nel tempo. Materiali didattici e app saranno condivisi con la comunità ITADINFO.

Riferimenti bibliografici

[1] CINI Laboratorio Nazionale Informatica e Scuola. (2023). Indicazioni per l'insegnamento dell'informatica nella scuola del primo ciclo. Consorzio CINI.

[2] Kafai, Y. B., & Burke, Q. (2015). Constructionist gaming: Understanding the benefits of making games for learning. Educational Psychologist, 50(4), 313–334.

[3] Ministero dell'Istruzione e del Merito. (2025). Decreto Ministeriale n. 221 del 9 dicembre 2025. Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione. Gazzetta Ufficiale, 27 gennaio 2026.

[4] Scratch Foundation & MIT Media Lab. (2023). MIT App Inventor documentation. https://appinventor.mit.edu

[5] Technovation. (2026). Technovation Girls & Technovation Boys: Student Curriculum and Mentor Resources. https://technovationchallenge.org

[6] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

Contributi per la secondaria di secondo grado

Interazione Fisica e Pensiero Computazionale: Fare, descrivere, modellizzare

Gionata Massi e Emanuele Lorenzoni

Istituto di Istruzione Superiore "Savoia Benincasa" di Ancona
gionata.massi@savoiabenincasa.it
emanuele.lorenzoni@outlook.com

Sommario

Questo contributo descrive un percorso di 10 ore di introduzione al *Physical Computing* svolto in una classe seconda dell'indirizzo ITE AFM mediante *Micro:bit* e l'ambiente *MakeCode*. L'obiettivo è stato presentare la programmazione come modellazione di comportamenti osservabili: eventi, stati, variabili, ingresso/uscita e transizioni temporizzate.

L'attenzione è rivolta ai contenuti propri dell'Informatica e non solo agli aspetti motivazionali. Il riferimento principale è la proposta CINI per il biennio della secondaria di secondo grado, in particolare per quanto riguarda programmi che reagiscono a eventi, modellano sistemi fisici, usano variabili di stato e sensori, e consentono di discutere la correttezza di semplici soluzioni [1].

L'esperienza ha confermato l'efficacia del riscontro immediato fornito dal dispositivo fisico, ma ha anche mostrato un limite importante: gli studenti riescono spesso a costruire programmi funzionanti prima di possedere un lessico teorico adeguato per descriverli.

1 Contesto, prerequisiti e obiettivi

L'esperienza si colloca nel progetto scolastico dell'IIS "Savoia Benincasa" "Citizen scientists of the future" (C.U.P. H34D23002330006), finanziato dal Piano Nazionale di Ripresa e Resilienza (PNRR), Missione 4 – Istruzione e Ricerca, Investimento 3.1 "Nuove competenze e nuovi linguaggi". Il corso è stato proposto a una classe seconda ITE AFM. Non erano richieste competenze pregresse di programmazione; i prerequisiti reali erano soprattutto la capacità di leggere consegne operative, distinguere causa ed effetto e usare numeri naturali per conteggi e tempi.

Gli obiettivi disciplinari sono stati formulati in termini informatici: riconoscere ingressi e uscite di un sistema; usare variabili per rappresentare lo stato dell'elaborazione; progettare programmi guidati da eventi; descrivere comportamenti con diagrammi di stato informali; comprendere che la stessa funzionalità può essere ottenuta con diverse organizzazioni del codice. In relazione al documento CINI, il percorso ha lavorato soprattutto sui traguardi T-S-6 e T-S-10 e sugli obiettivi O-S-P-1, O-S-P-2, O-S-P-4 e O-S-N-1 [1].

Più precisamente, le attività su pulsanti, sensori e reazioni del dispositivo hanno permesso di lavorare su O-S-P-1, poiché gli studenti hanno dovuto riconoscere il ruolo delle diverse parti del programma nel determinarne il comportamento; il segnapunti e gli altri esempi con variabili hanno reso esplicita la distinzione tra ingresso, uscita e stato interno; le attività con il blocco *forever* e con le transizioni temporizzate del *Tamagotchi* semplificato hanno coinvolto direttamente O-S-P-4; l'uso della *Micro:bit* come sistema fisico dotato di sensori, pulsanti e display ha consentito di lavorare su O-S-N-1; infine, la rappresentazione di uno stesso comportamento tramite blocchi *MakeCode*, stati numerici e diagrammi informali ha toccato, in forma introduttiva, anche O-S-D-1. In modo più cauto, il percorso ha sfiorato O-S-P-2 quando agli studenti è stato chiesto di leggere blocchi già scritti e anticiparne l'effetto osservabile.

2 Metodologia e strumenti

L'ambiente *MakeCode* per *Micro:bit* è stato scelto per tre ragioni: offre un linguaggio a blocchi leggibile dagli studenti del biennio; include simulatore e caricamento immediato sul dispositivo; rende evidente la relazione tra programma e comportamento fisico del sistema [2]. I tutorial iniziali utilizzati sono pubblici e accessibili online, in particolare *Flashing Heart*, *Name Tag* e *Smiley Buttons*[1].

La metodologia è stata quella della scoperta guidata con progressione per "ricette di progettazione": prima programmi che reagiscono direttamente agli eventi, poi programmi con stato interno, poi programmi con transizioni temporizzate. La scelta è coerente con una tradizione costruzionista in cui il dispositivo programmabile diventa oggetto da esplorare e modificare [3]. La nozione di automa a stati finiti non è stata presentata in forma teorica completa, ma come strumento di modellazione sufficiente per progettare e discutere i programmi prodotti.

3 Articolazione delle 10 ore

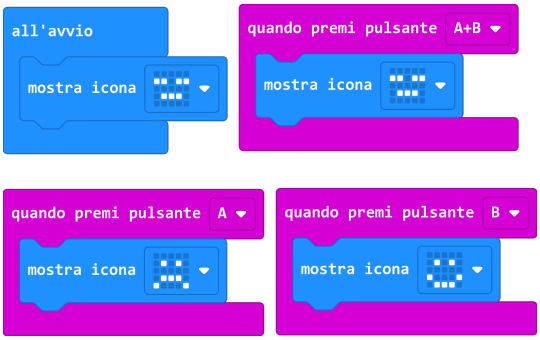
Le 10 ore sono state distribuite in cinque incontri da due ore. Il primo incontro ha avuto funzione motivazionale: familiarizzazione con il dispositivo, primi esempi di output sul display a LED e primi eventi da pulsante. Il secondo incontro ha introdotto l'idea di stato come comportamento osservabile del sistema, utilizzando faccine e semplici reazioni agli input. Il terzo incontro ha introdotto le variabili di stato tramite il problema del segnapunti. Il quarto incontro ha lavorato su sequenze cicliche, codifica degli stati e ramificazioni. Il quinto incontro è stato dedicato ai mini-progetti, alla gestione del tempo in un *Tamagotchi* semplificato, al test finale e al questionario di gradimento.

I contenuti informatici realmente trattati sono stati più specifici di quanto il termine "pensiero computazionale" lasci spesso intendere. Gli studenti hanno lavorato su: programmazione event-driven; distinzione tra input e output; variabili che rappresentano lo stato; cicli di controllo con blocco *forever* e condizioni; codifica numerica di stati; uso di sensori e radio come canali di ingresso.

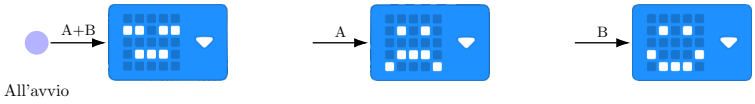
3.1 Transizioni guidate dagli eventi

Il primo schema di lavoro è quello in cui ogni input provoca direttamente un nuovo comportamento osservabile. In questa fase il valore formativo dell'attività non è stato solo motivazionale: gli studenti hanno dovuto associare in modo preciso un evento del mondo fisico a una porzione di programma e a un output atteso. La fig. 1 è stata usata per discutere il legame tra blocchi *MakeCode* e diagramma di stato informale.

[1] https://makecode.microbit.org/



(a) Programma in *MakeCode* della ricetta con transizioni guidate dagli eventi



(b) Diagramma di stato informale della stessa ricetta

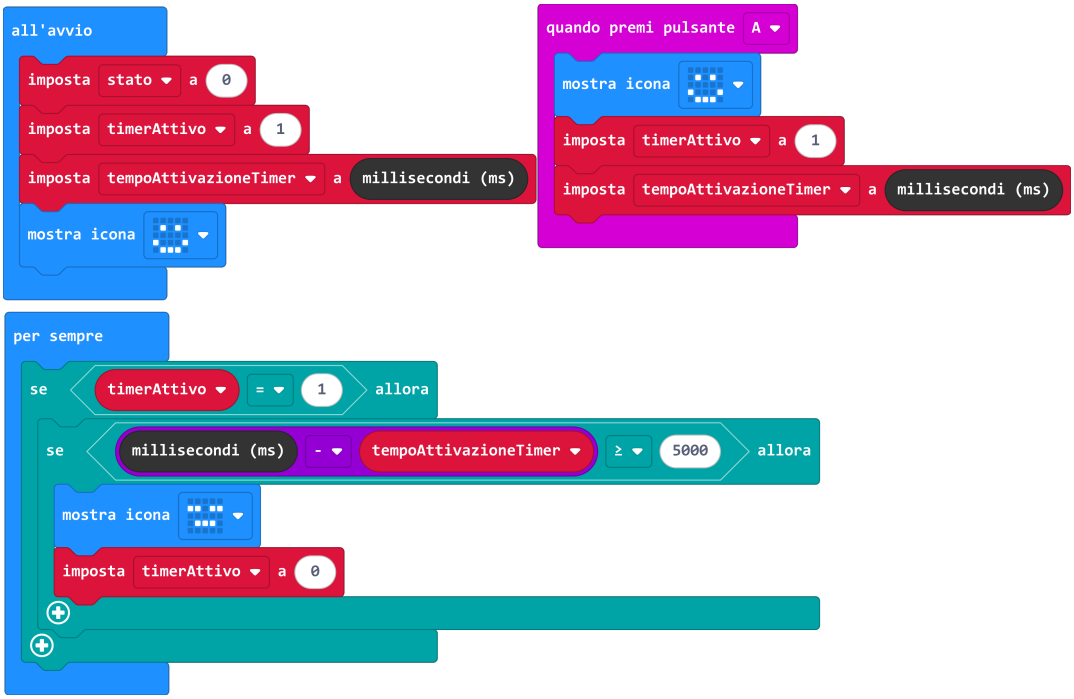
Figura 1: Programma in cui le transizioni di stato dipendono solo dagli eventi di input

Nel secondo schema, introdotto con il segnapunti, il programma non reagisce più solo all'evento istantaneo ma conserva memoria tramite una variabile. Questo passaggio è stato cruciale perché ha reso esplicita la differenza tra dati di ingresso, dati di uscita e dati che descrivono lo stato dell'elaborazione. In termini CINI, è qui che gli studenti hanno iniziato a lavorare davvero sulla rappresentazione dello stato nel programma.

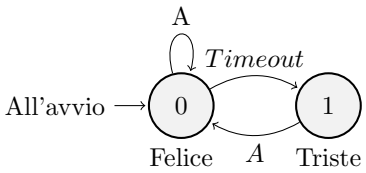
3.2 Codifica degli stati e transizioni temporizzate

Il terzo schema ha richiesto di codificare gli stati con numeri e di usare condizioni per scegliere il comportamento successivo. Esempi come "Pronti, partenza, via!" hanno permesso di introdurre la ciclicità e l'operatore modulo senza separare il ragionamento matematico da quello informatico.

La parte più significativa del percorso è stata però l'attività conclusiva sul *Tamagotchi* con noia temporizzata: dopo un intervallo senza interazione, l'animale virtuale cambia stato. Qui gli studenti hanno dovuto combinare evento, stato e tempo mediante una variabile booleana di attivazione del timer, una variabile per memorizzare l'istante iniziale e un controllo continuo nel blocco *forever*. La fig. 2 mostra il programma e il diagramma di stato informale associati a questa ricetta. Questa attività ha chiarito che il tempo, in un programma, deve essere rappresentato e controllato esplicitamente.



(a) Programma in *MakeCode* della ricetta con transizioni dipendenti dal tempo



(b) Diagramma di stato informale della stessa ricetta

Figura 2: Programma in cui le transizioni di stato dipendono dal tempo oltre che dagli eventi

4 Esiti osservati e criticità

Il test finale, somministrato nell'ultimo incontro, ha proposto quesiti a scelta multipla su lettura di blocchi *MakeCode*, distinzione tra input, output e variabili di stato, ruolo di *all'avvio* e *forever*, previsione di semplici transizioni di stato e riconoscimento di sensori, display e radio come componenti funzionali del sistema.

Il questionario di gradimento (15 risposte) ha restituito un quadro misto ma leggibile: chiarezza e adeguatezza del corso si concentrano soprattutto tra "Così così" e "Abbastanza", i giudizi sul formatore sono in prevalenza positivi (9 "Molto" per preparazione e competenza, 8 per disponibilità), mentre interesse suscitato, utilità percepita e coerenza con bisogni e aspettative risultano più deboli. Insieme alle osservazioni d'aula, questo dato invita a distinguere due piani. Sul piano motivazionale, il *Physical*

Computing ha funzionato bene. Sul piano orientativo, invece, l'associazione spontanea tra "attività interessante" e "possibile scelta di studio" non si è prodotta in modo altrettanto forte.

La criticità principale emersa in aula è stata la distanza tra competenza operativa e competenza concettuale. Alcuni studenti riuscivano a modificare correttamente un programma, ma non a spiegare con precisione quali dati rappresentassero lo stato o perché il ciclo *forever* fosse necessario nella versione temporizzata del *Tamagotchi*. Per questo motivo, in una futura edizione si intende aumentare i momenti di verbalizzazione esplicita e inserire brevi esercizi in cui si debba predire il comportamento del programma prima dell'esecuzione.

Un secondo limite riguarda la durata: 10 ore consentono una buona introduzione, ma non bastano per consolidare tutti i nuclei teorici sottesi all'attività pratica. In particolare servirebbe più tempo per lavorare sulla distinzione fra evento esterno e stato interno, sul lessico relativo ai sensori come input e al display come output, e sulla trasformazione dei diagrammi di stato in programmi.

Nel complesso, l'esperienza conferma che il *Physical Computing* può essere un contesto efficace per insegnare Informatica nel primo biennio, a condizione di esplicitare con decisione i contenuti disciplinari: non soltanto "fare cose con il dispositivo", ma modellizzare un comportamento, rappresentarne i dati e discuterne la correttezza.

Riferimenti bibliografici

[1] Gruppo di Lavoro CINI "Informatica e Scuola". *Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola*. Documento redatto in collaborazione con GII e GRIN. 2017. URL: https://www.consorzio-cini.it/images/Proposta-Indicazioni-Nazionali-Informatica-Scuola-numerata.pdf.

[2] Microsoft. *Microsoft MakeCode for micro:bit*. Ambiente di programmazione a blocchi e testuale con simulatore. 2026. URL: https://makecode.microbit.org/.

[3] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.

Dal *debate* alla modellazione computazionale: un'esperienza didattica sugli *Abstract Argumentation Framework*

Francesco Paolo Caforio

I.I.S.S. "Ettore Majorana", Martina Franca (TA)
francescopaolo.caforio@majorana.net

Abstract

Il contributo descrive un'esperienza didattica realizzata presso l'IISS "Ettore Majorana" di Martina Franca, Istituto Tecnico, indirizzo Informatica e Telecomunicazioni, articolazione Informatica, finalizzato all'introduzione dei fondamenti dell'intelligenza artificiale simbolica attraverso gli *Abstract Argumentation Framewor*k. L'attività propone una reinterpretazione del *debate* come problema di modellazione computazionale dove gli argomenti prodotti dagli studenti sono formalizzati come nodi di un grafo orientato e le relazioni di confutazione come archi di attacco. A partire da tale rappresentazione, gli studenti esplorano concetti di rappresentazione della conoscenza, teoria dei grafi, semantiche di accettabilità e ragionamento automatico. Il percorso integra discussione strutturata, analisi manuale dei framework argomentativi e sviluppo di un notebook in Python su Google Colab per il calcolo di insiemi *conflict-free*, *admissible*, *grounded* e *preferred*. Il percorso evidenzia come gli A*bstract Argumentation Framework* possano costituire un efficace contesto didattico per affrontare contenuti avanzati dell'Informatica e dell'Intelligenza Artificiale, promuovendo la comprensione del rapporto tra modello, algoritmo e decisione.

1 Introduzione

Con la diffusione sempre più pervasiva dell'Intelligenza Artificiale nel mondo della scuola diventa fondamentale strutturare percorsi didattici che mostrino agli studenti i diversi paradigmi teorici alla base di questa tecnologia. Sebbene oggi l'attenzione sia focalizzata soprattutto sul *machine learning*, restano ugualmente fondamentali gli approcci simbolici, in cui l'azione del sistema si basa sulla rappresentazione strutturata della conoscenza e sull'uso di regole logiche di ragionamento [2].

In questa prospettiva si colloca l'esperienza didattica realizzata presso l'I.I.S.S. "Ettore Majorana" di Martina Franca (TA), Istituto Tecnico, indirizzo Informatica e Telecomunicazioni, articolazione Informatica (ITIA), nella quale gli **Abstract Argumentation Framework (AAF)**, introdotti da Dung [7], sono stati utilizzati per avvicinare gli studenti ai concetti di modellazione, rappresentazione della conoscenza e ragionamento automatico.

L'idea progettuale nasce dalla volontà di reinterpretare la metodologia didattica del *debate* [5] come sorgente di dati da formalizzare attraverso strumenti propri dell'Informatica. Il dibattito diventa così un problema computazionale in cui gli argomenti vengono rappresentati come *nodi* di un grafo, le confutazioni come *archi* orientati e l'accettabilità degli argomenti viene determinata mediante algoritmi basati su specifiche *semantiche*. Questa scelta didattica consente di mostrare agli studenti come l'Informatica fornisca strumenti teorici per descrivere, rappresentare e risolvere problemi complessi [8].

La trasformazione mostrata in Figura 1 rappresenta il nucleo concettuale dell'intero percorso didattico e consente agli studenti di comprendere il ruolo dell'astrazione e della modellazione nella costruzione di sistemi intelligenti.

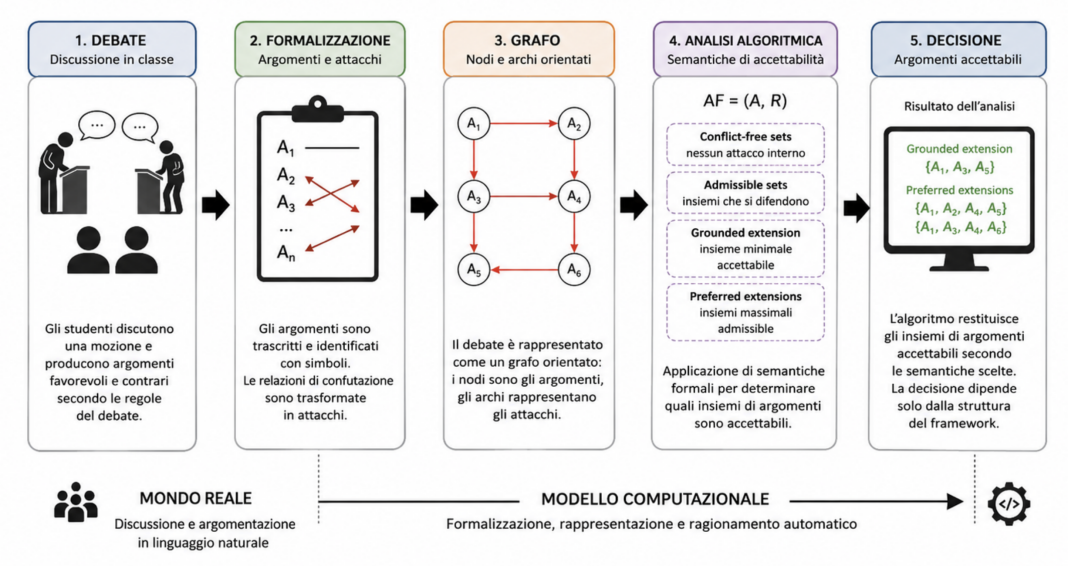


Figure 1: Dal debate al modello computazionale

2 I contenuti disciplinari

L'attività ha consentito di affrontare diversi temi, quali la rappresentazione della conoscenza, l'intelligenza artificiale simbolica e gli *Abstract Argumentation Framework*, introducendo gli studenti ai principi della modellazione computazionale e del ragionamento automatico [1]. Il percorso ha inoltre offerto l'opportunità di applicare concetti relativi alla teoria dei grafi orientati, alle strutture dati e alla progettazione di algoritmi, attraverso lo sviluppo di un'applicazione in `Python` per l'analisi di framework argomentativi. Le competenze trasversali emerse durante le attività, quali cooperazione, risoluzione dei problemi, pensiero critico e capacità argomentativa, hanno rappresentato un naturale risultato del percorso.

La Tabella 1 sintetizza la corrispondenza tra le principali fasi del laboratorio e i contenuti disciplinari di Informatica affrontati durante l'esperienza.

2.1 Contesto dell'esperienza

Il percorso, realizzato in una classe quarta composta da 14 studenti, si è sviluppato nell'ambito delle attività teoriche e laboratoriali di Informatica per una durata complessiva di circa 12 ore. Gli studenti possedevano già competenze di base relative alla programmazione `Python`, alle strutture dati fondamentali e alla rappresentazione mediante grafi. Le attività sono state svolte utilizzando `Google Colab` e la libreria `NetworkX` per la visualizzazione dei framework argomentativi.

3 Dal debate all'*Abstract Argumentation Framework*

La prima fase dell'attività didattica ha previsto una breve introduzione agli `Abstract Argumentation Framework` e lo svolgimento di un *debate* sulla seguente mozione relativa all'Intelligenza Artificiale: *"L'IA dovrebbe sostituire la valutazione scolastica"*.

Fase	**Attività svolta**	**Contenuti disciplinari**
Introduzione AAF	Presentazione del modello $AF = (A, R)$ e delle relazioni di attacco	IA simbolica; rappresentazione della conoscenza
Debate	Produzione e confronto di argomenti favorevoli e contrari alla mozione	Formalizzazione dell'informazione; modellazione di problemi
Formalizzazione	Identificazione degli argomenti e delle relazioni di attacco	Astrazione; modellazione computazionale
Costruzione del framework	Rappresentazione del *debate* mediante un grafo orientato	Teoria dei grafi; strutture dati
Analisi semantica	Individuazione di insiemi *conflict-free*, *admissible*, *grounded* e *preferred*	Ragionamento computazionale; semantiche di accettabilità
Implementazione	Sviluppo del notebook `Python` per l'analisi automatica del framework	Programmazione; algoritmi; strutture dati
Confronto	Verifica dei risultati ottenuti manualmente e algoritmicamente	Verifica e validazione di modelli computazionali
Discussione finale	Interpretazione dei risultati e riflessione sul processo decisionale	IA; ragionamento automatico

Table 1: Corrispondenza tra le fasi del laboratorio e i contenuti disciplinari di Informatica

Gli studenti, suddivisi in piccoli gruppi, hanno elaborato argomenti favorevoli e contrari confrontandosi secondo le regole del *debate*. Durante la discussione sono emerse posizioni differenti e numerosi argomenti a sostegno delle diverse tesi.

Terminata la fase argomentativa, gli studenti hanno trascritto gli argomenti e hanno iniziato a rappresentarli mediante simboli e relazioni di attacco. Questo passaggio si è rivelato particolarmente significativo per introdurre il concetto di *astrazione*, mostrando come una discussione reale possa essere trasformata in un modello computazionale, fondamento della rappresentazione formale della conoscenza e del ragionamento automatico.

Successivamente l'attenzione si è spostata sulla rappresentazione formale degli argomenti. Ogni *argomento* è stato identificato mediante un simbolo e ogni relazione di *confutazione* è stata trasformata in una relazione di attacco. Il debate è stato quindi rappresentato mediante un `Abstract Argumentation Framework`, definito formalmente come una coppia:

$$AF = (A, R)$$

dove A rappresenta l'insieme degli argomenti e R la relazione di attacco. Gli studenti hanno così compreso come un problema reale possa essere modellato mediante un grafo orientato e come tale rappresentazione costituisca il punto di partenza per il ragionamento automatico.

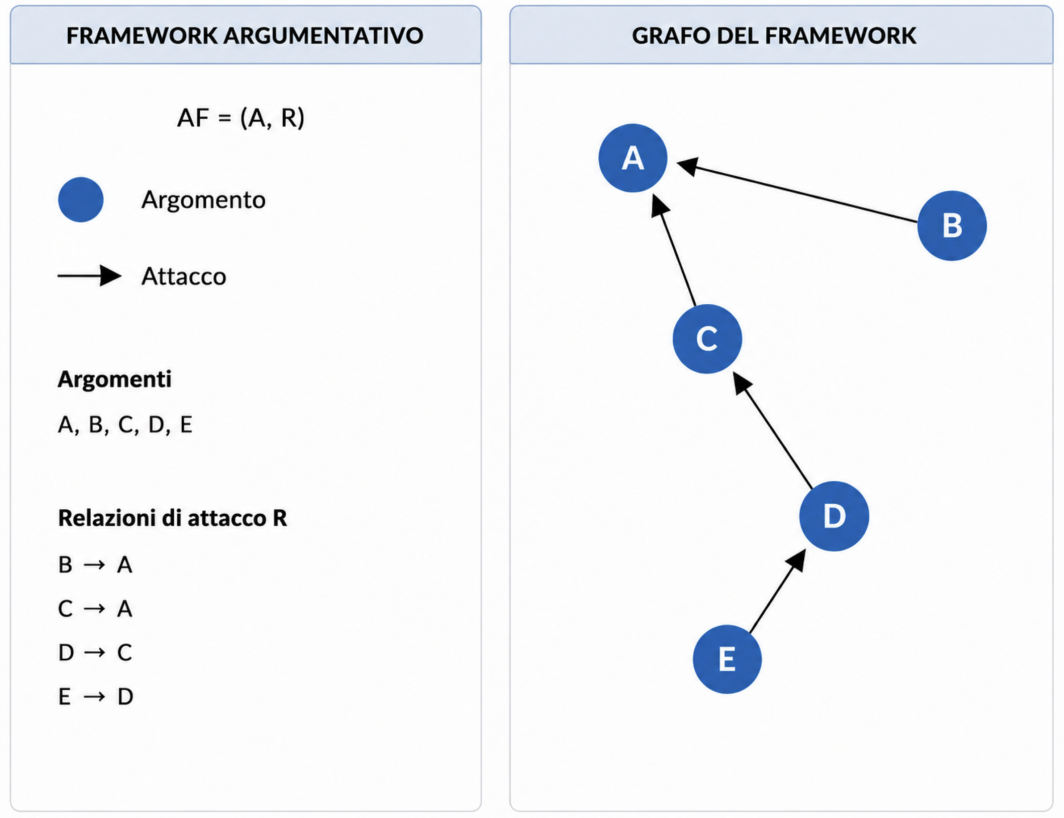


Figure 2: Grafo dell'*Abstract Argumentation Framework* costruito dagli studenti

4 Un caso di studio

Nell'attività laboratoriale, gli studenti hanno tradotto la mozione proposta in un framework formale, individuando i seguenti argomenti speculari:

- `A`: L'IA garantisce maggiore oggettività.
- `B`: L'IA non comprende il contesto educativo.
- `C`: La valutazione deve essere personalizzata.
- `D`: Modelli ben addestrati possono personalizzare la valutazione.
- `E`: I dati di addestramento possono introdurre *bias*.

Le relazioni di attacco identificate (`B→A, C→A, D→C, E→D`) sono state formalizzate e rappresentate dagli studenti mediante il grafo orientato riportato in Figura 2. Questa rappresentazione grafica ha reso immediatamente evidente la struttura topologica del *debate* favorendo il passaggio dalla dimensione argomentativa a quella computazionale.

5 Semantica degli *Abstract Argumentation Framework*

La fase più significativa del laboratorio ha riguardato l'esplorazione della semantica degli `Abstract Argumentation Framework`. Gli studenti hanno scoperto che un sistema intelligente determina quali insiemi di argomenti possano essere accettati sulla base di criteri formali.

L'attività ha preso avvio dall'individuazione degli insiemi `conflict-free`, ossia insiemi di argomenti nei quali non esistono relazioni di attacco tra gli elementi che li compongono. Successivamente sono stati introdotti gli insiemi `admissible`, definiti come insiemi *conflict-free* in grado di difendere ciascuno dei propri argomenti dagli attacchi esterni. A partire da tali concetti sono state quindi esplorate la `grounded semantics`, che individua l'insieme di argomenti accettabili più prudente, e la `preferred semantics`, che ricerca gli insiemi admissible massimali.

L'analisi è stata inizialmente svolta manualmente. Nel framework costruito, gli studenti hanno prima individuato gli argomenti non attaccati e successivamente verificato quali insiemi soddisfacessero le proprietà di conflict-freeness e difendibilità. Tale attività ha consentito di comprendere concretamente come differenti semantiche possano produrre risultati diversi pur operando sul medesimo framework. Gli studenti hanno lavorato sul grafo costruito durante il *debate*, individuando le relazioni di attacco e formulando ipotesi sugli insiemi accettabili. Solo dopo questa fase di riflessione i risultati sono stati confrontati con quelli prodotti dallo script `Python`.

Il confronto tra analisi umana e algoritmica si è rivelato efficace dal punto di vista didattico. Gli studenti hanno potuto osservare come il comportamento del sistema dipendesse solo dalla struttura del framework e dalle regole implementate. In questo modo il concetto di semantica è stato compreso come il criterio operativo che guida il processo decisionale di un sistema di ragionamento automatico.

Questa attività ha favorito la comprensione del rapporto tra rappresentazione della conoscenza, ragionamento automatico e processo decisionale, mostrando come il ragionamento possa essere formalizzato e implementato attraverso modelli propri della disciplina Informatica.

6 Dalla modellazione all'implementazione

La formalizzazione del *debate* è stata ulteriormente consolidata mediante l'impiego di un notebook sviluppato in `Python` e ospitato su `Google Colab`, utilizzato come supporto didattico per guidare gli studenti nell'esplorazione progressiva degli `Abstract Argumentation Framework`. Le attività di analisi sono state supportate da momenti di riflessione, nei quali gli studenti hanno formulato ipotesi prima dell'esecuzione degli algoritmi. Il confronto tra i risultati attesi e quelli effettivamente ottenuti ha favorito la comprensione del rapporto tra modello, semantica e processo computazionale.

Il notebook ha consentito agli studenti di inserire gli argomenti e le relazioni di attacco, costruendo automaticamente il framework corrispondente. La rappresentazione interna ha utilizzato strutture dati basate su insiemi e liste, mentre la rappresentazione grafica si è avvalsa della libreria `NetworkX`.

L'applicazione ha implementato algoritmi per il calcolo degli insiemi `conflict-free`, delle `grounded extension` e delle `preferred extension`. Gli studenti hanno potuto osservare come il risultato dell'elaborazione dipendesse esclusivamente dalla struttura del grafo e dalle regole semantiche implementate. Un esempio di esecuzione del notebook è riportato in Figura 3.

Considerata la complessità degli algoritmi per il calcolo delle estensioni e il contesto della classe, l'implementazione è stata predisposta dal docente. I principali passaggi logici degli algoritmi sono stati comunque illustrati e discussi durante l'attività, mettendo in relazione le condizioni implementate nel codice con le definizioni precedentemente analizzate.

Nel Repository [3] è archiviata l'implementazione utilizzata durante le attività, corredata da codice, commenti esplicativi ed esempi di output. Per favorire la replicabilità dell'attività, una selezione dei

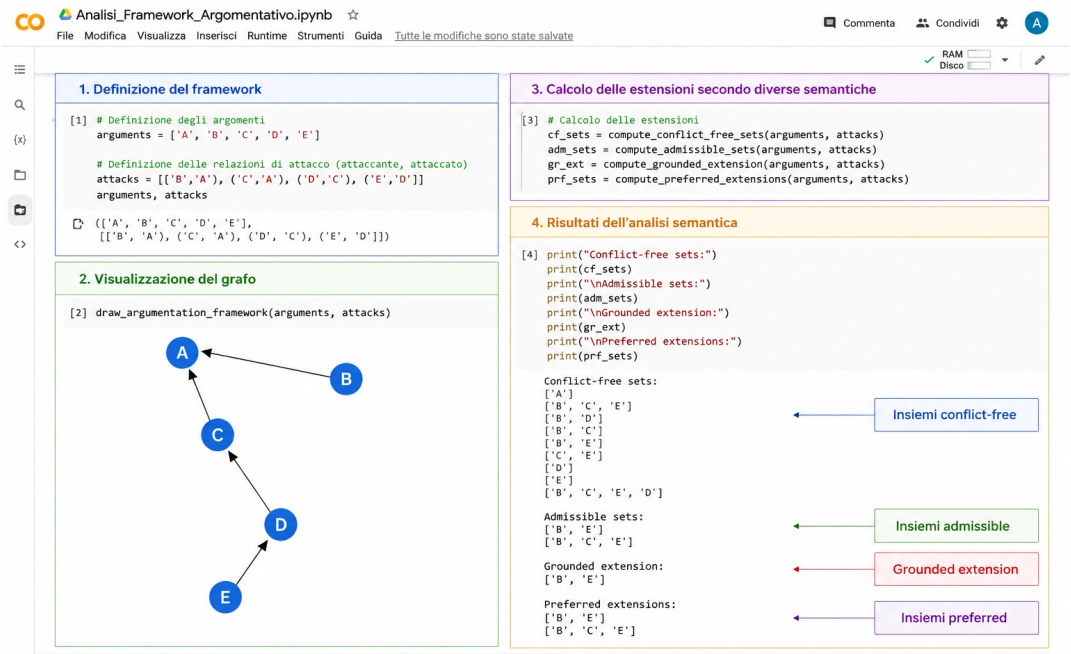


Figura 4 – Analisi automatica del framework argomentativo mediante il notebook sviluppato durante il laboratorio.

Figure 3: Analisi automatica del framework argomentativo mediante il notebook sviluppato durante il laboratorio

principali materiali didattici, delle fonti bibliografiche e delle risorse di approfondimento è stata pubblicata in una wiki dedicata [4].

Per confrontare i risultati ottenuti mediante il notebook sviluppato durante il laboratorio, è stata inoltre utilizzata la piattaforma web `ConArg` [6], un ambiente online per la modellazione e l'analisi di `Abstract Argumentation Framework` che consente di rappresentare strutture argomentative e calcolare automaticamente diverse semantiche di accettabilità. L'utilizzo di tale strumento ha permesso agli studenti di validare i risultati ottenuti e di confrontare il comportamento di differenti implementazioni algoritmiche.

7 Risultati e riflessioni didattiche

L'esperienza ha mostrato un coinvolgimento degli studenti. Il passaggio dal *debate* alla modellazione computazionale ha favorito la comprensione dell'astrazione come principio fondante dell'Informatica.

Particolarmente efficace si è rivelata l'introduzione della semantica degli `Abstract Argumentation Framework`, che ha permesso agli studenti di comprendere come un sistema intelligente possa prendere decisioni applicando regole formali. Il laboratorio, inoltre, ha consentito di integrare rappresentazione della conoscenza, teoria dei grafi, algoritmi e programmazione, offrendo un esempio concreto di progettazione di un sistema di ragionamento automatico.

La verifica delle competenze ha avuto in particolare carattere formativo ed è stata condotta attraverso l'osservazione sul campo delle attività, la discussione plenaria delle soluzioni proposte dai gruppi e il

confronto tra risultati attesi e output del notebook. Non è stata però prevista una prova individuale strutturata. Durante l'analisi manuale, inoltre, non è stata effettuata una rilevazione sistematica delle *misconceptions* individuali. Le difficoltà emerse sono state affrontate prevalentemente attraverso la discussione collettiva e il successivo confronto tra le soluzioni formulate dagli studenti e i risultati prodotti dal notebook.

8 Conclusioni e sviluppi futuri

L'esperienza descritta ha evidenziato come gli `Abstract Argumentation Framework` possano costituire un efficace contesto didattico per l'insegnamento dell'Intelligenza Artificiale nella scuola secondaria di secondo grado.

La somministrazione di una prova individuale strutturata al termine dell'attività, ad esempio attraverso la risoluzione di un framework non precedentemente analizzato, e una rilevazione più sistematica degli errori concettuali ricorrenti possono rappresentare possibili sviluppi futuri dell'esperienza.

L'utilizzo del *debate* come punto di partenza per la costruzione di modelli formali ha consentito di valorizzare contenuti propri dell'Informatica, mostrando agli studenti che l'IA comprende anche approcci fondati sulla rappresentazione della conoscenza e sul ragionamento simbolico. La progettazione e l'implementazione di un sistema in grado di rappresentare un grafo argomentativo hanno offerto un'esperienza autentica di modellazione computazionale.

A differenza di approcci percepiti dagli studenti come *scatole nere*, gli `Abstract Argumentation Framework` rendono esplicito il processo decisionale, consentendo di osservare il percorso che conduce dal problema reale alla sua formalizzazione e dalla formalizzazione alla soluzione algoritmica. In questa prospettiva, il *debate* diventa un problema di Informatica costituito da un insieme di informazioni da rappresentare, analizzare e sottoporre a procedure di ragionamento automatico.

References

[1] Trevor J. M. Bench-Capon and Paul E. Dunne. Argumentation in artificial intelligence. *Artificial Intelligence*, 171(10–15):619–641, 2007.

[2] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*. MIT Press, Cambridge, MA, 2008.

[3] Francesco Paolo Caforio. Abstract argumentation framework lab. https://github.com/francescocaforio/AAF/, 2026. Accessed: 2026-06-24.

[4] Francesco Paolo Caforio. Abstract argumentation framework lab: Materiali didattici, risorse e fonti. https://github.com/francescocaforio/AAF/wiki/, 2026. Accessed: 2026-06-24.

[5] Michele Darby. Debate: A teaching-learning strategy for developing competence in communication and critical thinking. *Journal of Dental Hygiene*, 81(4):78, 2007.

[6] Mauro Dragoni, Paolo Taticchi, Federico Bianchi, Francesco Santini, Laura Giordano, and Alberto Martelli. Conarg2: A tool for abstract argumentation frameworks. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017)*, pages 1702–1704, 2017.

[7] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[8] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4 edition, 2021.

Il cliente ha sempre ragione? Un laboratorio di ingegneria del software per la scuola superiore

Daniele Fea[1] and Manuela Marchisio[2]

[1] I.I.S G. Vallauri, Fossano, Cuneo, Italia
daniele.fea@vallauri.edu
[2] I.I.S G. Vallauri, Fossano, Cuneo, Italia
manuela.marchisio@vallauri.edu

Sommario

Presentiamo un percorso didattico per una classe quinta di informatica dell'IIS G. Vallauri di Fossano (a.s. 2025–2026). Gli studenti affrontano l'intero ciclo di vita del software su un progetto comune (gestionale per una clinica privata) e sul proprio progetto d'esame individuale. Il kit didattico, composto da copione d'intervista, scheda studente e guida per i docenti, supporta l'apprendimento attivo e lo sviluppo delle competenze trasversali richieste dal mondo del lavoro.

1 Introduzione

Le competenze dell'ingegneria del software sono centrali nel percorso degli studenti di informatica degli istituti tecnici, ma spesso vengono presentate in astratto, senza un contesto autentico di applicazione.

Questo articolo descrive il percorso sviluppato nella 5ª A Informatica dell'IIS G. Vallauri (a.s. 2025–2026), con l'obiettivo di accompagnare gli studenti attraverso l'intero ciclo di vita di un sistema informatico reale. Il percorso si articola in due livelli paralleli: prima la classe lavora su un *progetto comune* di riferimento (un sistema gestionale per una clinica privata), simulando un'intervista con il cliente e costruendo collettivamente la documentazione tecnica; poi ogni studente applica le stesse fasi al proprio *progetto individuale* di maturità. L'approccio favorisce la collaborazione tra pari e il trasferimento delle competenze dal contesto condiviso a quello individuale.

Il percorso persegue gli obiettivi di apprendimento previsti dalle Linee guida per gli istituti tecnici (D.P.R. 88/2010 e Direttiva n. 4/2012) per le discipline *Informatica* e *Gestione progetto, organizzazione d'impresa*: gestire le specifiche, la pianificazione e lo stato di avanzamento di un progetto software, documentarne le fasi, e realizzare applicazioni coerenti con i requisiti raccolti da un committente. Sul piano metodologico la progettazione si fonda su quattro riferimenti espliciti: la didattica laboratoriale, il gioco di ruolo (*role playing*) per la simulazione del colloquio con il cliente, il rilascio graduale di responsabilità (prima il modello costruito insieme in classe, poi l'applicazione autonoma al proprio progetto) e la valutazione formativa, affidata al feedback dei docenti nei tre controlli intermedi.

La struttura dell'articolo è la seguente: la sezione 2 descrive il percorso in quattordici lezioni; la sezione 3 presenta i materiali didattici elaborati; la sezione 4 raccoglie riflessioni e risultati; la sezione 5 chiude con le considerazioni finali.

2 Il percorso didattico

Il percorso si articola in **quattordici lezioni** da tre ore ciascuna, distribuite nel periodo febbraio–maggio e organizzate intorno a tre momenti di verifica (*milestone*) con i docenti. Ogni lezione mantiene una struttura binaria: una prima parte dedicata al progetto comune (la clinica privata) e una seconda parte in cui gli studenti lavorano sul proprio progetto individuale di esame. La Tabella 1 descrive nel dettaglio calendario, attività e prodotti attesi di ciascuna lezione.

Lez.	Attività	Mese	Prodotto
Verso la prima milestone			
1	Colloquio simulato: i docenti interpretano il direttore sanitario della clinica e il consulente informatico, gli studenti assistono come analisti compilando la scheda e intervenendo nei punti previsti dal copione; avvio dei colloqui a coppie sui progetti d'esame, alternando i ruoli di cliente e consulente	Febbraio	Scheda requisiti
2	Conclusione dei colloqui a coppie; riorganizzazione collettiva alla lavagna dei requisiti della clinica, con discussione delle differenze di interpretazione e dei punti controversi	Febbraio	Requisiti classificati
3	Costruzione collettiva del diagramma di flusso dei dati (DFD) della clinica: entità esterne, processi, archivi e flussi; ogni studente classifica i requisiti e abbozza il DFD del proprio progetto	Febbraio	DFD individuale
4	Revisione dei requisiti con il cliente, di nuovo interpretato in aula; presentazione della struttura della documentazione attesa e dei criteri di valutazione	Marzo	Requisiti consolidati
5	Lavoro individuale sulla documentazione dei progetti d'esame, con feedback personalizzato dei docenti	Marzo	Doc. v1
6	1° controllo (I sessione): metà classe espone il documento di analisi (requisiti e DFD)	Marzo	Feedback
7	1° controllo (II sessione); introduzione a sviluppo software e scelte architetturali di base	Marzo	Feedback
Verso la seconda milestone			
8	*Project* e *people management* con Trello [1], ClickUp [2] e diagramma di Gantt, applicati al caso della clinica; ogni studente configura il proprio strumento di pianificazione	Aprile	Piano di progetto
9	Costruzione collettiva dello schema concettuale Entità/Relazione (E/R) della clinica a partire dal DFD; sviluppo del modello individuale con il supporto dei docenti	Aprile	Modello concettuale
10	Prototipazione delle interfacce utente (*mockup*) con *Figma* [3] sul caso della clinica; realizzazione dei mockup individuali	Aprile	Prototipo interfacce
11	2° controllo: consegna di schema E/R, mockup e Gantt; verifica di coerenza e indicazione delle priorità di sviluppo	Aprile	Feedback
Verso la terza milestone			
12	Implementazione delle prime funzionalità e strutturazione del database, in coerenza con la documentazione	Maggio	Codice v1
13	3° controllo (I sessione): presentazione di funzionalità, database e documentazione aggiornata	Maggio	Feedback finale
14	3° controllo (II sessione); feedback complessivo orientato alla prova orale dell'esame di Stato	Maggio	Feedback finale

Tabella 1: Calendario e attività del percorso didattico

3 I materiali didattici

Per supportare il percorso sono stati elaborati quattro documenti, disponibili nella repository del progetto [4]. Il kit comprende: il **copione dell'intervista**, **la guida per i docenti**, la **scheda studente** e le **soluzioni suggerite**.

4 Riflessioni e risultati

4.1 Punti di forza osservati

La struttura "prima insieme, poi ognuno per conto proprio" si è rivelata efficace: chi aveva lavorato attivamente alla parte comune replicava gli stessi passaggi sul proprio progetto con maggiore sicurezza.

L'uso del copione teatrale ha aumentato la partecipazione: non solo gli studenti più riservati hanno preso la parola nei momenti previsti dal copione, ma diversi interventi spontanei si sono verificati al di là delle pause prestabilite, con una partecipazione che ha sorpreso gli stessi docenti. In alcuni casi le domande e le riflessioni degli studenti hanno addirittura anticipato passaggi del colloquio ancora da svolgere, a testimonianza di un coinvolgimento che andava ben oltre l'ascolto passivo.

Il ruolo del cliente, volutamente poco tecnico, entusiasta e a tratti contraddittorio, ha generato dinamiche autentiche di negoziazione tra team e cliente, difficilmente riproducibili con una semplice spiegazione frontale.

4.2 Progetto comune e progetto individuale: contesti diversi ma complementari

I due livelli del percorso sono volutamente diversi e sono stati presentati come tali agli studenti. Il progetto comune li espone ai due problemi fondamentali dell'ingegneria del software: interpretare requisiti che provengono da un committente terzo (con le sue ambiguità, i suoi vincoli e i suoi ripensamenti) e coordinarsi con altri per raggiungere un risultato condiviso in tempi prevedibili. Il progetto individuale, privo per sua natura di entrambe le dimensioni, rischia di ridurre le tecniche apprese a un adempimento scolastico, da abbandonare una volta liberi dai vincoli della scuola. Per attenuare questo rischio abbiamo reintrodotto esplicitamente la figura del committente anche nel contesto individuale: nei colloqui a coppie ogni studente interpreta il cliente del compagno, sperimentando l'asimmetria informativa dal lato opposto, mentre nei tre controlli i docenti assumono il ruolo di committente che verifica lo stato di avanzamento. L'osservazione in classe suggerisce che il passaggio dal contesto condiviso a quello individuale abbia aiutato gli studenti a percepire le tecniche non come rituali imposti, ma come strumenti per governare ambiguità e complessità; una verifica più sistematica di questa percezione è tra gli sviluppi futuri del percorso.

4.3 Criticità e aggiustamenti in itinere

La parte sugli strumenti di project management è risultata la più disomogenea: gli studenti con buone basi digitali hanno esplorato le funzionalità avanzate di ClickUp, mentre altri si sono limitati a una semplice lista di attività.

Le parti più teoriche del percorso, ossia le introduzioni frontali a nuovi strumenti o concetti, potrebbero essere ripensate in un'ottica più cooperativa: anziché una spiegazione seguita da applicazione individuale, si potrebbe affidare alla classe l'esplorazione collettiva guidata, con compiti parziali distribuiti tra gruppi e poi integrati in un risultato condiviso.

5 Conclusioni

Il percorso descritto mostra come sia possibile integrare competenze tecniche (analisi dei requisiti, DFD, schema E/R, progettazione delle interfacce) e competenze trasversali (comunicazione con il cliente, lavoro in team, gestione del tempo) in un unico percorso coerente e progressivo.

La disponibilità di materiali strutturati presenti nella repository (copione, scheda studente, guida docente e soluzioni) permette di replicare l'attività con classi diverse, adattando il contesto narrativo mantenendo inalterata la struttura didattica.

Sitografia

[1] Atlassian. *Trello – Strumento di project management*. https://trello.com

[2] ClickUp. *ClickUp – Piattaforma di gestione dei progetti*. https://clickup.com

[3] Figma, Inc. *Figma – Strumento di prototipazione UI*. https://www.figma.com

[4] Fea, D., Marchisio, M. *Kit didattico – Gestione del progetto software*. https://github.com/manuelamarchisio/GESTIONE-PROGETTO-ITADINFO

Valutare il processo di sviluppo software: repository GitHub, rubriche e feedback rapido nella didattica dell'informatica

Francesco Adriani
Istituto Tecnico Tecnologico "Franchetti-Salviani", Città di Castello
francesco.adriani@franchettisalviani.net

Abstract

Il contributo descrive un'esperienza svolta in una classe quarta di un istituto tecnico informatico, nell'insegnamento di Tecnologie e progettazione di sistemi informatici e delle telecomunicazioni, sul tema dei thread in Java. L'attività ha utilizzato GitHub Classroom e un portale didattico sviluppato dal docente per trasformare i repository degli studenti in evidenze del processo di sviluppo. La valutazione ha considerato, oltre al prodotto finale, la progressione del lavoro, la documentazione delle scelte e la leggibilità del percorso. Una rubrica condivisa prima dell'attività ha guidato il lavoro degli studenti e la correzione del docente. L'integrazione tramite API ha ridotto il lavoro di raccolta delle evidenze e consentito una restituzione più tempestiva e strutturata.

1 Contesto e problema

Nelle attività di programmazione il codice consegnato alla fine racconta solo una parte dell'apprendimento. Due studenti possono arrivare a prodotti simili attraverso percorsi molto diversi: uno può procedere per tentativi casuali e concentrare tutto il lavoro nell'ultima fase, mentre un altro può analizzare il problema, scomporlo in passaggi, correggere errori intermedi e documentare le proprie decisioni. Una valutazione centrata esclusivamente sul prodotto finale può rendere poco visibili la progressione, le strategie adottate, le revisioni e il metodo di lavoro sviluppato durante l'attività.

L'esperienza qui descritta nasce dall'esigenza di rendere valutabile il processo. Il caso riguarda una classe quarta di indirizzo informatico, nella disciplina Tecnologie e progettazione di sistemi informatici e delle telecomunicazioni. L'attività richiedeva la progettazione e realizzazione in Java di un piccolo gioco ispirato al whac-a-mole, usato come contesto operativo per affrontare il tema dei thread. La consegna prevedeva un percorso graduale: lettura dei requisiti, ideazione del mockup, schematizzazione UML, sviluppo del codice e revisione progressiva della soluzione.

2 Inquadramento pedagogico e curricolare

L'osservazione del processo di sviluppo può essere ricondotta a due riferimenti pedagogici consolidati. Nella prospettiva di John Dewey [3], l'esperienza assume valore educativo quando l'azione è accompagnata dall'osservazione delle sue conseguenze, dalla riflessione e dalla successiva riorganizzazione dell'azione. Nell'attività di programmazione questo ciclo può assumere la forma di progettazione, realizzazione, verifica, individuazione dell'errore e revisione della soluzione.

Il costruzionismo di Seymour Papert [8] offre un secondo riferimento particolarmente vicino alla didattica dell'informatica: l'apprendimento avviene attraverso la costruzione e il progressivo perfezionamento di artefatti significativi. In questa prospettiva anche l'errore e il debugging

acquistano valore formativo, perché costringono lo studente a formulare ipotesi, verificarle e modificare il proprio modello.

Git non viene quindi utilizzato come oggetto della valutazione, ma come strumento capace di lasciare tracce parziali di questo processo. Commit, messaggi, revisioni e branch possono rendere osservabili la scomposizione del problema, la verifica degli avanzamenti, la documentazione delle scelte e il progressivo miglioramento dell'artefatto.

Questa impostazione trova inoltre corrispondenza nelle Linee guida nazionali degli Istituti Tecnici per l'indirizzo Informatica e Telecomunicazioni, che per TPSIT richiamano il ciclo di sviluppo, la gestione di specifiche e requisiti, la documentazione delle scelte progettuali e la capacità di identificare le fasi di un progetto [6]. Un riferimento complementare è offerto dal DigComp 3.0, in particolare dalle competenze relative al pensiero computazionale e alla programmazione, alla risoluzione dei problemi tecnici e, nelle attività di gruppo, alla collaborazione attraverso le tecnologie digitali [2].

3 Repository come evidenze del processo

Git e GitHub, normalmente impiegati come strumenti professionali di versionamento, sono stati utilizzati anche in chiave didattica per rendere osservabile il percorso. Ogni commit può rappresentare un avanzamento verificabile; il messaggio associato al commit richiede allo studente di esplicitare cosa ha fatto e perché; la cronologia permette al docente di distinguere un lavoro costruito progressivamente da una consegna caricata tutta alla fine.

Per evitare che questo controllo diventasse un lavoro manuale ingestibile, è stato usato un portale didattico sviluppato su misura. Attraverso le API di GitHub [4], il sistema associa i repository agli studenti, recupera i dati essenziali e li presenta in una pagina di sintesi: collegamenti ai repository, branch, numero di commit, data dell'ultimo aggiornamento, file modificati, righe aggiunte o rimosse e dettaglio dei singoli passaggi. Il sistema alleggerisce la raccolta meccanica delle informazioni, lasciando al docente l'interpretazione didattica delle evidenze e il giudizio finale.

4 Rubrica, feedback e ruolo del docente

La condivisione preventiva della rubrica affianca alla funzione certificativa una funzione formativa, rendendo espliciti i criteri con cui il processo viene osservato [7]. Le dimensioni sono state formulate mettendo in relazione le evidenze disponibili nel repository con gli obiettivi relativi al ciclo di sviluppo, alla documentazione e alla gestione del progetto previsti dalle Linee guida nazionali [6] e con le competenze DigComp 3.0 relative al pensiero computazionale, alla risoluzione dei problemi tecnici e alla collaborazione digitale [2].

Dimensione osservata	**Evidenza nel repository**	**Obiettivo educativo-didattico**	**Valore formativo**
Progressione e scomposizione del lavoro	Distribuzione e granularità dei commit	Pianificare il lavoro, scomporre un problema complesso in sotto-obiettivi, procedere in modo incrementale	Rende visibile la continuità dello sviluppo
Coerenza e significatività degli avanzamenti	Coerenza tra messaggio e modifiche introdotte	Abituare lo studente a realizzare avanzamenti riconoscibili e coerenti con l'obiettivo dichiarato	Rende comprensibile l'evoluzione del lavoro
Organizzazione del processo di sviluppo	Branch per funzionalità o fix, quando richiesti	Organizzare lo sviluppo, isolare funzionalità e gestire in modo controllato l'evoluzione del software	Rende osservabile la gestione della complessità
Documentazione e motivazione delle scelte	Messaggi di commit, problemi, soluzioni, test e fonti	Rendere esplicito il ragionamento, documentare le scelte progettuali e riflettere sul proprio percorso	Trasforma la cronologia in un breve diario tecnico e permette al docente di comprendere il processo decisionale

Dimensione osservata	Evidenza nel repository	Obiettivo educativo-didattico	Valore formativo
Revisione e miglioramento incrementale	Correzioni, modifiche successive, refactoring	Utilizzare l'errore come occasione di revisione e migliorare progressivamente la soluzione	Valorizza l'apprendimento dall'errore
Integrazione e collaborazione *(quando prevista)*	Push, merge, PR e branch condivisi	Collaborare in modo affidabile, integrare il proprio lavoro senza compromettere quello altrui	Collega l'attività a pratiche di sviluppo collaborativo e rende osservabile la responsabilità tecnica verso il gruppo

Tabella 1: Dimensioni del processo di sviluppo osservate attraverso le evidenze presenti nel repository e relativo valore formativo.

Il feedback comprende il voto in decimi, la valutazione analitica dei criteri e un eventuale commento del docente. Il sistema propone il voto a partire dai punteggi della rubrica, ma il docente può modificarlo dopo l'esame diretto del lavoro. L'automazione facilita quindi la consultazione delle evidenze e riduce i tempi di correzione, lasciando al docente la responsabilità valutativa.

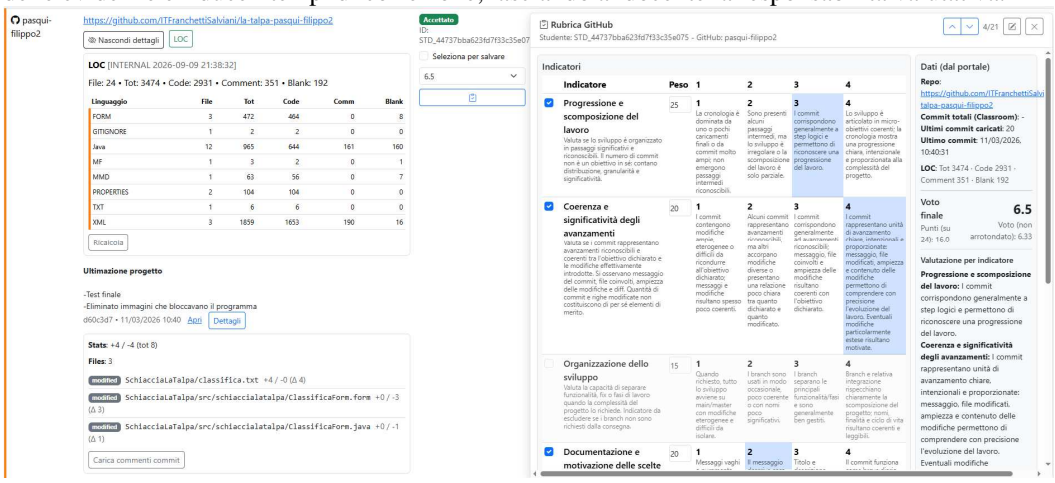


Figura 1: Sintesi dell'assegnazione GitHub con dettaglio dei commit e rubrica di valutazione.

5 Percezione degli studenti e impatto osservato

Dalle osservazioni svolte durante l'attività e dalle restituzioni informali degli studenti sono emersi soprattutto tre aspetti: maggiore tempestività del feedback, maggiore trasparenza dei criteri e percezione di una valutazione più attenta al percorso. L'impegno documentato, la descrizione dei passaggi e la capacità di analizzare il problema entrano infatti nella valutazione accanto alla qualità del prodotto finale.

In alcuni casi osservati, studenti con maggiori difficoltà iniziali hanno utilizzato la sequenza di passaggi richiesta come struttura per orientare il lavoro. Alcuni, normalmente bloccati dalla sindrome del foglio bianco, hanno trovato nella richiesta di descrivere e versionare i passaggi una guida per iniziare. La documentazione progressiva ha favorito un approccio top-down al problem solving e, in alcuni casi, la realizzazione di artefatti più strutturati.

6 Discussione, limiti e prospettive

L'esperienza conferma l'utilità di criteri espliciti e feedback tempestivo nella valutazione formativa [1][5]. In questa prospettiva il repository diventa una fonte di evidenze sul modo di lavorare dello studente. Git richiede tuttavia regole condivise e un apprendimento specifico: senza

una guida può ridursi a semplice deposito finale. Rubrica, checkpoint ed esempi di commit rimangono quindi essenziali.

Il contributo di UDA-system consiste nel rendere sostenibile la raccolta e l'organizzazione delle evidenze necessarie al giudizio professionale del docente, in coerenza con DigCompEdu [9]. L'esperienza ha carattere qualitativo e non sperimentale; sviluppi futuri potranno includere questionari, confronti longitudinali e analisi delle variazioni nelle pratiche di sviluppo.

Bibliografia

[1] Black, P., Wiliam, D. Assessment and Classroom Learning. Assessment in Education: Principles, Policy & Practice, 5(1), 7–74, 1998.

[2] Cosgrove, J., Cachia, R. DigComp 3.0: European Digital Competence Framework – Fifth Edition. Publications Office of the European Union, Luxembourg, 2025. JRC144121, doi:10.2760/0001149.

[3] Dewey, J. Experience and Education. Macmillan, New York, 1938.

[4] GitHub REST API documentation. https://docs.github.com/en/rest. Accessed: 2026-09-08.

[5] Hattie, J., Timperley, H. The Power of Feedback. Review of Educational Research, 77(1), 81–112, 2007.

[6] Ministero dell'Istruzione, dell'Università e della Ricerca. Direttiva 16 gennaio 2012, n. 4 – Linee guida per il secondo biennio e quinto anno per i percorsi degli Istituti Tecnici, a norma dell'articolo 8, comma 3, del D.P.R. 15 marzo 2010, n. 88. 2012.

[7] Panadero, E., Jönsson, A. The Use of Scoring Rubrics for Formative Assessment Purposes Revisited: A Review. Educational Research Review, 9, 129–144, 2013. doi:10.1016/j.edurev.2013.01.002.

[8] Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York, 1980.

[9] Redecker, C. European Framework for the Digital Competence of Educators: DigCompEdu. Punie, Y. (ed.), Publications Office of the European Union, Luxembourg, 2017. EUR 28775 EN, doi:10.2760/159770.

Dalla didattica della matematica alla didattica dell'informatica: il modello OSCAR come framework di valutazione affettivo-cognitivo

Gennaro Iaccarino[1] and Ilenia Fronza[2]

[1] Direzione Istruzione e Formazione italiana - Bolzano, Italy
gennaro.iaccarino@provincia.bz.it
[2] Libera Università di Bolzano - Bolzano, Italy
ilenia.fronza@unibz.it

Sommario

Sebbene la didattica della matematica abbia sviluppato nel tempo strumenti consolidati per l'analisi delle pratiche educative, tali approcci risultano ancora poco esplorati nel contesto della didattica dell'informatica. In questo lavoro proponiamo l'adattamento di un framework di valutazione proveniente dalla didattica della matematica per analizzare e valutare le attività di coding camp del modello OSCAR[1]. Lo studio descrive il framework e il processo di adattamento alla didattica dell'informatica e avvia un percorso di formazione per docenti della Provincia Autonoma di Bolzano, utile alla definizione degli indicatori di valutazione, alla sperimentazione del modello e alla validazione empirica.

1 Introduzione

Matematica e informatica condividono molti aspetti fondamentali dell'apprendimento, come la capacità di affrontare problemi complessi, sviluppare strategie di approccio e imparare attraverso il confronto con gli errori. A partire da queste caratteristiche comuni, l'obiettivo della ricerca è quello di adattare alla didattica dell'informatica alcuni strumenti di valutazione già ampiamente utilizzati nell'ambito della didattica della matematica.

In particolare, l'obiettivo è quello di definire un framework di valutazione applicabile ai coding camp realizzati secondo il modello OSCAR [9, 11], contesti di apprendimento immersivi nei quali i partecipanti sviluppano competenze sia disciplinari che trasversali. Un ulteriore obiettivo riguarda l'analisi del ruolo della formazione dei docenti nella comprensione, nell'adozione e nel trasferimento del framework all'interno delle pratiche didattiche curricolari.

In questa prospettiva, la ricerca intende rispondere alle seguenti domande:

- **RQ1.** Quali elementi dei framework di valutazione, sviluppati e consolidati nell'ambito della didattica della matematica, risultano trasferibili alla didattica dell'informatica, con gli opportuni adattamenti ma conservando la stessa efficacia?
- **RQ2.** In che modo gli elementi individuati possono essere adattati e applicati alla valutazione di contesti di apprendimento immersivi e laboratoriali come i coding camp, al fine di rilevare sia lo sviluppo delle competenze disciplinari che quelle trasversali?

[1]This study was co-funded by the European Union in the framework of the OSCAR project (ERASMUS-EDU-2023-PI-FORWARD, Project number: 101132432). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

- **RQ3.** In quale misura un percorso di formazione specificamente progettato per i docenti può favorire la comprensione, l'adozione e l'applicazione del framework di valutazione proposto, sostenendone il trasferimento dai contesti intensivi dei coding camp alle pratiche didattiche curricolari della scuola secondaria?

Nei prossimi paragrafi vedremo i presupposti metodologici della proposta e illustreremo progressivamente gli elementi che consentono di rispondere alle tre domande di ricerca.

2 Didattica della matematica (RQ1)

La ricerca in didattica della matematica, negli ultimi anni, ha superato la visione in cui l'apprendimento è rappresentato esclusivamente dall'acquisizione di conoscenze e procedure, affermando un paradigma integrato in cui processi cognitivi, metacognitivi e affettivi risultano strettamente connessi tra loro [1, 2, 6]. In particolare, il *problem solving* è stato scelto come strumento per l'analisi dei processi di pensiero degli studenti, superando l'approccio focalizzato esclusivamente sul prodotto finale e valorizzando invece le strategie, le rappresentazioni e le difficoltà emergenti durante l'attività risolutiva [4]. Parallelamente, la letteratura recente ha evidenziato come le difficoltà in matematica non possano essere interpretate esclusivamente in termini di deficit cognitivi, ma debbano essere comprese come fenomeni complessi legati a misconcezioni, alla gestione dell'errore e alle pratiche didattiche di riferimento [5]. In particolare, studi sul *sense making* [4] hanno mostrato come gli studenti possano talvolta perdere il significato di ciò che stanno facendo, per privilegiare strategie procedurali o operazioni meccaniche, riducendo così la qualità dell'apprendimento.

All'interno di questo quadro teorico si colloca il contributo di molti ricercatori italiani ed internazionali, che hanno sviluppato una linea di ricerca centrata sul ruolo delle dimensioni affettive ed emozionali nell'apprendimento della matematica. Il costrutto di *attitude* [2] è stato oggetto di una sistematizzazione teorica volta a chiarire la natura multidimensionale delle emozioni, delle motivazioni e delle rappresentazioni degli studenti nei confronti della disciplina. Tali dimensioni risultano strettamente connesse ai processi di problem solving, influenzando la disponibilità degli studenti a impegnarsi nella costruzione di significato matematico.

Inoltre, la prospettiva interpretativa proposta in [7] introduce il concetto di *interpretative knowledge*, ossia la capacità dell'insegnante di leggere e decodificare il pensiero matematico degli studenti, includendo non solo le strategie cognitive ma anche le dimensioni affettive sottostanti. In questa visione, la formazione degli insegnanti assume un ruolo centrale, poiché la qualità dell'intervento didattico dipende dalla capacità di interpretare errori, strategie e convinzioni degli studenti come parte di un sistema coerente di significati. Ne deriva un modello teorico in cui **problem solving, dimensioni affettive e competenze interpretative dell'insegnante** costituiscono un framework integrato per la didattica della matematica contemporanea [2, 7].

Questa prospettiva costituisce il punto di partenza teorico per la transizione verso la didattica dell'informatica (**RQ1**). In particolare, alcuni nuclei concettuali individuati nella didattica della matematica sono rilevanti anche nel dominio della programmazione e dell'informatica teorica, pur richiedendo specifici adattamenti. Il problem solving, ad esempio, è centrale nella programmazione, dove gli studenti affrontano problemi caratterizzati da incertezza, gestione dell'errore e revisione iterativa delle proprie soluzioni.

Analogamente, le difficoltà osservate nei processi di comprensione matematica possono essere messe in relazione, sul piano teorico, con fenomeni osservabili durante il debugging e la progettazione degli algoritmi o dei database, inclusi approcci prevalentemente procedurali e possibili sospensioni del *sense making* [8]. Tuttavia, tali corrispondenze non implicano una equivalenza diretta tra i due domini. Nel

presente lavoro, pertanto, problem solving, attitude e interpretative knowledge sono considerati costrutti derivati dalla letteratura sulla didattica della matematica, mentre la loro traduzione in indicatori e modalità di osservazione nei coding camp rappresenta l'adattamento proposto dagli autori.

Il framework proposto assume quindi la forma di un'ipotesi di trasferimento: i costrutti teorici vengono mantenuti come riferimento concettuale, mentre indicatori e strumenti di osservazione vengono ridefiniti in funzione delle caratteristiche della programmazione, del lavoro progettuale e delle dinamiche collaborative tipiche dei coding camp. La validità di tale adattamento non viene assunta a priori, ma costituisce un oggetto della successiva validazione empirica.

Questa continuità tra didattica della matematica e didattica dell'informatica suggerisce la possibilità di estendere il framework di valutazione a contesti formativi differenti, includendo non solo le attività curricolari tradizionali, ma anche esperienze educative non convenzionali quali i coding camp.

3 Il modello OSCAR (RQ2)

Il progetto OSCAR[2] è un progetto *Erasmus + Forward-Looking Projects* che mira a promuovere e sviluppare esperienze di apprendimento non convenzionali, online e/o ibride, per gli studenti della scuola secondaria di secondo grado [9, 11]. Nato nel 2024, il progetto è diventato un vero e proprio modello didattico con una piattaforma dedicata e un manuale d'uso per docenti e organizzatori. Punti di forza del modello sono il carattere fortemente inclusivo [10, 12] e la riproducibilità delle attività organizzative e didattiche, largamente descritte e a disposizione della comunità. Il progetto coinvolge sette partner tra università e istituzioni scolastiche di tre Paesi europei (Italia, Finlandia e Spagna) e favorisce la collaborazione tra scuola, università e istituzioni, creando occasioni di incontro tra ricerca e pratica educativa. Particolare attenzione è dedicata alla riduzione del gender-gap nelle discipline informatiche, attraverso esperienze didattiche innovative capaci di coinvolgere sempre più ragazze [9].

I coding camp realizzati nell'ambito del modello OSCAR sono particolarmente indicati all'impiego di un framework di valutazione basato su problem solving, dimensioni affettive e competenze interpretative dell'insegnante. Durante il camp, gli studenti sono chiamati ad affrontare problemi aperti, progettare strategie di soluzione, collaborare con i pari e rivedere iterativamente il proprio lavoro attraverso processi di verifica e correzione degli errori. Tali dinamiche risultano analoghe a quelle ampiamente descritte nella didattica della matematica, dove l'apprendimento viene interpretato non come semplice acquisizione di conoscenze, ma come costruzione progressiva di strategie, significati e competenze. Allo stesso tempo, il coinvolgimento in attività progettuali, con la produzione di applicativi socialmente utili [3], rende particolarmente rilevanti gli aspetti emotivi: motivazione, senso di autoefficacia, perseveranza e atteggiamento nei confronti dell'errore, elementi che influenzano in modo significativo la partecipazione e il successo formativo degli studenti. In questo contesto, il ruolo del docente non può essere ridotto alla sola verifica dei prodotti finali, ma richiede competenze di osservazione e interpretazione capaci di cogliere i processi di apprendimento che si sviluppano durante le attività. Il framework proposto consente quindi di estendere ai coding camp una prospettiva valutativa centrata non solo sui risultati ottenuti, ma anche sulle modalità attraverso cui gli studenti affrontano i problemi, collaborano, prendono decisioni e sviluppano consapevolezza delle proprie strategie **(RQ2)**. Anche i concetti di *sense making* e *attitude*, introdotti nel paragrafo precedente, trovano riscontro nella valutazione dei coding camp del modello OSCAR. In tali contesti, infatti, gli studenti rischiano di concentrarsi prevalentemente sugli aspetti tecnici dello sviluppo software, perdendo progressivamente la consapevolezza delle finalità sociali e trasversali del progetto e riducendo l'efficacia della collaborazione all'interno del team. La Figura 1 mostra il framework di valutazione applicato alla didattica della matematica e alla didattica dell'informatica.

[2] https://oscar-codingcamps.eu



Figura 1: Framework di valutazione affettivo-cognitivo, nato dalla didattica della matematica e adattato ai coding camp del modello OSCAR (*immagine generata con l'ausilio dell'IA*).

L'attività di ricerca sul tema continuerà verificando come ciascuna di queste dimensioni possa essere tradotta in indicatori osservabili e strumenti di valutazione applicabili ai coding camp del progetto OSCAR.

4 Coding Camp for Teachers (RQ3)

Come evidenziato nel Paragrafo 2, un valido framework per la didattica della matematica attribuisce un ruolo centrale alle competenze interpretative dei docenti. La formazione degli insegnanti rappresenta quindi un elemento fondamentale anche per l'applicazione dei modelli di valutazione per i coding camp. In quest'ottica, nel mese di novembre 2026 sarà realizzato un percorso di formazione rivolto a 30 docenti della scuola secondaria di secondo grado appartenenti a diverse discipline. L'iniziativa, promossa dalla Direzione Istruzione e Formazione italiana in collaborazione con la Libera Università di Bolzano nell'ambito del progetto OSCAR, sarà finalizzata a sviluppare competenze condivise per l'osservazione e la valutazione dei processi di apprendimento attivati durante i coding camp **(RQ3)**. Particolare attenzione sarà dedicata alla definizione degli indicatori osservabili e all'analisi dei processi attraverso cui gli studenti acquisiscono le competenze e raggiungono gli obiettivi formativi previsti. I docenti saranno successivamente chiamati a sperimentare gli strumenti acquisiti nei rispettivi contesti scolastici, trasferendo il modello OSCAR dagli ambienti laboratoriali immersivi alla pratica didattica convenzionale. Tale fase consentirà di verificare la trasferibilità e la sostenibilità del framework di valutazione al di fuori del contesto specifico dei coding camp.

5 Conclusioni

Sebbene il presente lavoro si collochi ancora a un livello teorico-concettuale, l'analisi svolta suggerisce che problem solving, dimensioni affettive e competenze interpretative del docente costituiscano costrutti

comuni ai due domini disciplinari. Tali evidenze supportano la trasferibilità del framework dalla didattica della matematica alla didattica dell'informatica, fornendo una risposta preliminare alla **RQ1**. Tali elementi risultano particolarmente adatti alla valutazione dei coding camp realizzati secondo il modello OSCAR, consentendo di osservare non solo i risultati prodotti dagli studenti, ma anche i processi affettivi ed emotivi della programmazione e del lavoro in team **(RQ2)**. La validazione empirica di questa ipotesi rappresenta il naturale sviluppo della ricerca. Nel mese di novembre 2026, sarà attivata una formazione rivolta ai docenti, finalizzata al trasferimento delle pratiche valutative dai contesti immersivi dei coding camp alla didattica curricolare **(RQ3)**.

I futuri sviluppi della ricerca saranno mirati alla definizione degli indicatori di valutazione, alla sperimentazione del modello e alla validazione degli strumenti di osservazione, nei diversi contesti educativi del progetto OSCAR.

Riferimenti bibliografici

[1] Ginevra Aquilina, Pietro Di Martino, Caterina Di Pasquale, and Giulia Lisarelli. Educational choices and mathematical motivation: tracing upper secondary students' reconstructed trajectories through a narrative and situative lens. *ZDM–Mathematics Education*, pages 1–13, 2026.

[2] Ginevra Aquilina, Pietro Di Martino, and Giulia Lisarelli. The construct of attitude in mathematics education research: current trends and new research challenges from a systematic literature review. *ZDM–Mathematics Education*, 57(4):569–581, 2025.

[3] Anna Brancaccio, Luis Corral, Ilenia Fronza, and Gennaro Iaccarino. Building smart apps for smart cities: un esempio di sinergia tra pcto ed educazione civica, concluso ai tempi di covid-19. *DIDAMATICA 2020: Atti convegno*, pages 380–389, 2020.

[4] Gemma Carotenuto, Pietro Di Martino, and Marta Lemmi. Students' suspension of sense making in problem solving. *ZDM–Mathematics Education*, 53(4):817–830, 2021.

[5] Pietro Di Martino and Anna Baccaglini-Frank. Strane storie matematiche: studio di difficoltà in matematica: il fenomeno di resistenza all'intervento e un modo per far affiorare misconcezioni sui numeri razionali. *Archimede: 3, 2020*, pages 161–165, 2020.

[6] Pietro Di Martino et al. Perché imparare la matematica? una riflessione sul senso dell'educazione matematica. *L'insegnamento della Matematica e delle Scienze Integrate*, 47(5):432–445, 2024.

[7] Pietro Di Martino, Maria Mellone, and Miguel Ribeiro. Interpretative knowledge. In *Encyclopedia of mathematics education*, pages 424–428. Springer, 2020.

[8] Stephen Frezza, Mats Daniels, and Roger McDermott. Sense-making frameworks in computing and engineering education. In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2024.

[9] Ilenia Fronza, Gennaro Iaccarino, and Veronica Rossano. Esperienze di apprendimento non-convenzionali: insegnare informatica uscendo dagli schemi. In *Secondo Convegno Italiano sulla Didattica dell'Informatica - ITADINFO 2024*, pages 97–102, 2024.

[10] Ilenia Fronza, Gennaro Iaccarino, Sara Tosi, and Luis Corral. Designing inclusive and gender-balanced coding camps: A comprehensive assessment framework. In *Proceedings of the 25th Annual Conference on Information Technology Education*, pages 25–31, 2024.

[11] Ilenia Fronza and Tommi Mikkonen. Project oscar: Promoting cross-cutting digital skills through europe-wide non-conventional learning experiences. In *2025 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 498–501. IEEE, 2025.

[12] Gennaro Iaccarino, Ilenia Fronza, Tommi Mikkonen, Petri Ihantola, and Sara Tosi. Towards differentiated instruction in coding camps. In *Proceedings of the 26th ACM Annual Conference on Cybersecurity & Information Technology Education*, SIGCITE '25, page 321–323, New York, NY, USA, 2025. Association for Computing Machinery.

La cripta del faraone perduto

Un'esperienza di Digital Storytelling, Gamification, Robotica e IoT con Arduino

Michele Caceffo[1], Marco Cuoghi[1], and Nicola Dalla Pozza[1]

ITT G.Chilesotti, Thiene (VI), Italia
mcaceffo@chilesotti.it , mcuoghi@chilesotti.it , ndallapozza@chilesotti.it

Sommario

Questo lavoro presenta un *Project Work* di 26 ore che ha coinvolto 57 studenti di informatica dell'ITT "G. Chilesotti" in attività di robotica, automazione e IoT su piattaforma Arduino. L'intervento integra *Project-Based Learning*, *Gamification* e *Digital Storytelling* tramite Moodle all'interno di una cornice narrativa unificata. La rotazione dei kit tecnologici utilizzati ha stimolato l'inclusione e il *problem solving* di squadra. I risultati quantitativi e qualitativi confermano l'efficacia del modello nel potenziare simultaneamente sia le *hard skills* tecniche sia le *soft skills* relazionali.

1 Introduzione

Nel panorama dell'istruzione tecnica superiore, il raccordo tra le competenze teoriche fornite dai programmi ministeriali e le richieste tecnologiche del mondo produttivo rappresenta una delle sfide didattiche più complesse. La normativa italiana ha cercato di rispondere a questa esigenza strutturando i percorsi di Formazione Scuola-Lavoro, i quali mirano a sviluppare negli studenti una mentalità progettuale e una solida base di competenze trasversali, elementi oggi centrali anche nella ristrutturazione dell'esame di Maturità del secondo ciclo [1, 2, 3].

Tuttavia, all'interno del piano di studi dell'indirizzo Informatica e Telecomunicazioni, l'insegnamento di concetti complessi legati all'architettura degli elaboratori, alla programmazione e alle reti rischia spesso di rimanere confinato a esercitazioni astratte e ripetitive. La mancanza di un contesto applicativo autentico può generare un calo della motivazione intrinseca degli studenti, portando a una frammentazione delle conoscenze e a una scarsa capacità di lavorare efficacemente in *team* per la risoluzione di problemi reali.

Per superare queste limitazioni, il presente lavoro documenta la progettazione, l'implementazione e i risultati di un *Project Work* (PW) intensivo realizzato presso l'ITT "Giacomo Chilesotti" di Thiene. L'intervento didattico propone un modello che integra i principi del *Project-Based Learning* [4] con metodologie attive di *Gamification* [5] e *Digital Storytelling* [6], strutturato interamente attorno all'ecosistema hardware e software Arduino [7].

L'elemento di novità dell'approccio risiede nella fusione di queste metodologie: le esercitazioni laboratoriali di robotica mobile, automazione e Internet of Things non vengono somministrate come compiti isolati, ma sono declinate come missioni interdipendenti all'interno di una cornice narrativa uniforme gestita tramite la piattaforma Moodle. Il contributo dell'articolo si concentra sull'analisi di questa architettura didattica, mostrando come la rotazione sinergica delle tecnologie e lo stimolo della componente ludica permettano di ottimizzare i tempi di apprendimento, favorire l'inclusione e incrementare sia le *hard skills* tecniche che le *soft skills* relazionali del gruppo classe.

2 Contesto di riferimento

Il *Project Work* fa parte delle attività di Formazione Scuola-Lavoro (FSL), l'evoluzione dei percorsi di PCTO, che hanno l'obiettivo di avvicinare gli studenti alle aziende e alle realtà tecnologiche del territorio [1, 2, 3]. In questa attività, basata sul *learning by doing*, gli studenti lavorano in gruppo, e sono guidati dai docenti o dai tutor aziendali per risolvere un problema reale. In questo modo si simula un vero ambiente di lavoro, aiutando i ragazzi a sviluppare sia competenze tecniche (*hard skills*) sia

abilità trasversali (*soft skills*), come il lavoro di squadra e il *problem solving* [4]. L'organizzazione del progetto si basa su tre metodologie principali:

- **Project-Based Learning (PBL):** questo approccio si collega alle idee di Dewey [8] e al costruzionismo di Papert [9]. Gli studenti imparano facendo, lavorando a un progetto complesso per un periodo di tempo prolungato. Creando qualcosa di concreto [10], si riduce la distanza tra teoria e pratica. L'efficacia di questo metodo nella scuola è ampiamente dimostrata in letteratura [11, 12].
- **Gamification:** per evitare che i ragazzi si scoraggino davanti alle difficoltà tecniche, la struttura del progetto riprende alcune meccaniche tipiche dei giochi [5]. L'uso di questi elementi ludici aiuta a tenere alti il coinvolgimento e la motivazione, migliorando l'apprendimento e la memorizzazione dei concetti [13, 14].
- **Digital Storytelling:** all'inizio del lavoro, i docenti usano la narrazione multimediale come "gancio narrativo" (*narrative hook*) per presentare la sfida tecnologica [6]. Il video e il racconto non servono come resoconto finale fatto dagli studenti, ma servono a inserire la classe in uno scenario reale (o realistico), stimolando la curiosità e la collaborazione nel team fin dal primo giorno [15].

3 Metodologia e Organizzazione del Project Work

Il Project Work (PW) si è svolto a cavallo della pausa natalizia dell'anno scolastico 2025-2026, per un totale di 26 ore distribuite su quattro giornate (dal 19 dicembre al 9 gennaio). L'attività ha coinvolto tre classi terze dell'indirizzo Informatica dell'ITT "Giacomo Chilesotti" di Thiene, per un totale di 57 alunni suddivisi in piccoli team. Questa scelta temporale ha permesso di sfruttare le competenze sulla programmazione di schede Arduino appena apprese nel modulo iniziale di "Sistemi e Reti", offrendo al contempo una didattica laboratoriale alternativa capace di tenere alta la concentrazione degli studenti in un periodo dell'anno critico.

Dal punto di vista logistico, le aule sono state trasformate in laboratori usando i PC portatili forniti dalla scuola. Per la parte pratica sono stati impiegati circa trenta kit didattici basati sulla piattaforma Arduino [7], divisi in tre tipologie: robot mobili (*Ardusumo*), bracci robotici (*Arduino Braccio*) e schede per l'Internet of Things (*Arduino Explore IoT*).

Un punto chiave della progettazione è stato il sistema di **rotazione delle tecnologie**: tutte e tre le classi hanno affrontato le stesse identiche esercitazioni tecniche, ma lo hanno fatto all'interno di una cornice narrativa differente. Lo scenario generale, intitolato "La Cripta del Faraone Perduto", simulava una spedizione archeologica divisa in tre percorsi paralleli associati agli elementi Terra, Acqua e Aria (uno per classe). La narrazione e i progressi dei gruppi sono stati gestiti e monitorati attraverso la piattaforma Moodle (vedi Sezione 3.4), offrendo un'esperienza immersiva in cui ogni tecnologia rispondeva a una precisa esigenza della trama:

- **Percorso Aria (Classe 3IA):** il kit IoT è stato impiegato per decifrare e trasmettere a distanza i geroglifici di una sala della cripta; il robot mobile ha esplorato un corridoio stretto e incrostato dal tempo; il braccio robotico è servito a recuperare una pietra preziosa da un altare sospeso.
- **Percorso Acqua (Classe 3IB):** il braccio robotico è servito a recuperare un mosaico sommerso a protezione di un vaso canopo; il sistema IoT controllava le condizioni ambientali della stanza allagata; il robot Ardusumo ha dovuto navigare all'interno di un antico pozzo invaso dalle alghe.
- **Percorso Terra (Classe 3IM):** il robot mobile è stato usato per esplorare i cunicoli instabili della cripta; il braccio robotico per spostare i pesanti massi che bloccavano l'altare; il kit IoT per monitorare e trasmettere i parametri ambientali necessari a sbloccare la lastra di chiusura.

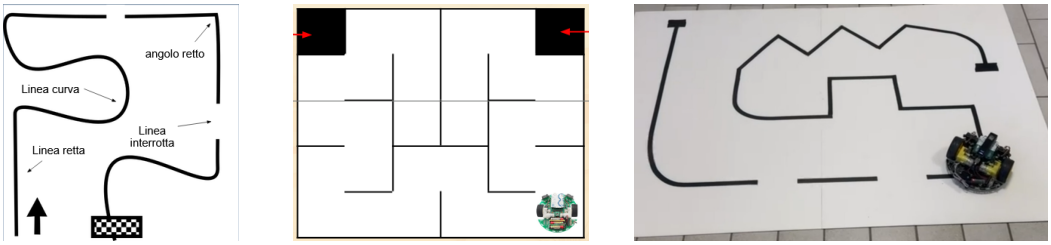
Anche la logistica delle aule è stata adattata a questa rotazione, in particolare per l'attività con Ardusumo che richiedeva una pista fissa e uno spazio di movimento più ampio (vedi Sezione 3.1). In questo modo, la narrazione non è stata un semplice abbellimento, ma il motore motivazionale del progetto. Trasformando i classici esercizi di programmazione e sensoristica in vere e proprie "missioni" con vincoli pratici, gli studenti hanno affrontato i problemi tecnici con un forte spirito di squadra, migliorando sia le competenze tecniche che quelle trasversali come il *problem solving* e la divisione dei ruoli nel team.

3.1 Ardusumo: Robotica Mobile e Algoritmi di Navigazione

Il robot Ardusumo[1] è una piattaforma didattica su due ruote controllata da una scheda Arduino UNO e dotata di sensori di prossimità e infrarossi (IR). Il dispositivo è stato inserito nel Project Work per introdurre le classi ai concetti base della robotica mobile e della programmazione dei sistemi di controllo automatici. L'attività pratica è stata strutturata in task di difficoltà progressiva:

- **Calibrazione e Diagnostica:** lettura sul monitor seriale dei sensori per distinguere le superfici (bianco vs nero) e calibrazione dei motori per realizzare i primi movimenti e le rotazioni con il robot.
- **Inseguitore di Linea (*Line Follower*):** programmazione del robot per seguire un tracciato a terra, contenente una linea continua nera con curve a gomito e interruzioni del tracciato (Fig. 1a).
- **Inseguitore di Linea con ostacoli:** simile al precedente, ma lungo la linea continua possono essere presenti delle scatole da evitare.
- **Risoluzione di Labirinti:** utilizzo dei sensori IR frontali per rilevare le pareti e implementare una strategia algoritmica di navigazione e uscita (Fig. 1b).

A livello software, gli studenti hanno gestito i pin digitali e i segnali PWM per controllare direzione e velocità di rotazione dei motori. Per focalizzare l'attenzione sulla logica algoritmica ed problemi di natura elettronica, è stato fornito un *template* di base con la sequenza di istruzioni per la gestione dello stato di *standby* e la regolazione differenziata della velocità delle ruote.



(a) Circuito line follower (b) Labirinto per test IR (c) Test in aula

Figura 1: Attività laboratoriali e percorsi di test realizzati con il robot Ardusumo.

3.2 Tinkerkit Braccio Robot: Automazione e Controllo Meccatronico

Il dispositivo Tinkerkit Braccio[2] è un braccio robotico azionato da sei servomotori che realizzano una catena cinematica a cinque gradi di libertà, integrando una pinza terminale (*gripper*). Trattandosi di un sistema privo di sensori di feedback in ingresso, si configura come un attuatore puro controllato interamente tramite la scheda Arduino (Fig. 2a).

Per l'interfaccia software sono state fornite agli studenti le specifiche delle funzioni di libreria per il controllo angolare dei giunti: il posizionamento assoluto simultaneo di tutti gli assi (`setAllAbsolute`), il movimento assoluto del singolo servo (`setOneAbsolute`) e lo spostamento relativo rispetto alle coordinate correnti (`setOneRelative`).

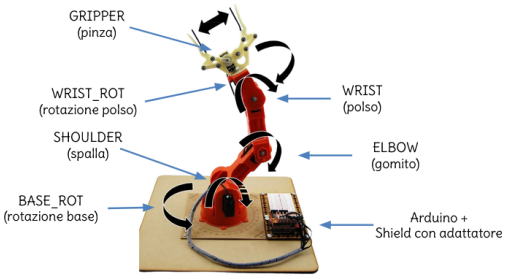
Le attività pratiche sono state strutturate in tre task a complessità crescente, orientate a simulare i tipici scenari dell'automazione industriale:

- **Calibrazione e limiti operativi:** identificazione sperimentale dei range angolari e della posizione di riposo (*home*).
- **Controllo analogico:** implementazione di un sistema di movimentazione realizzato con una console analogica basata su potenziometri (*trimmer*), la cui rotazione viene mappata nella rotazione del giunto.

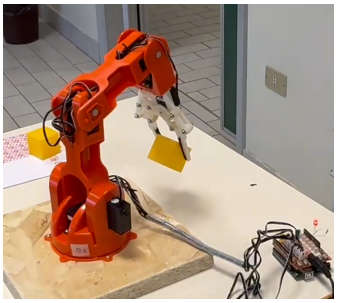
[1] Il robot Ardusumo è descritto al link `https://www.open-electronics.org/ardusumo-an-open-source-platform-for-fighting-robots/`

[2] Il kit Tinkerkit Braccio è descritto al link `https://store.arduino.cc/products/tinkerkit-braccio-robot`

- **Controllo remoto:** implementazione di un sistema di movimentazione tramite comandi testuali inviati da monitor seriale.
- **Movimentazione algoritmica (*Pick and Place*):** sviluppo di una sequenza automatica e ciclica per lo spostamento di una pila di blocchi (Fig. 2b), individuando le traiettorie nello spazio tridimensionale e temporizzare i movimenti dei giunti.



(a) Schema dei servomotori e degli assi.



(b) Fase di movimentazione dei cubi.

Figura 2: Configurazione geometrica ed esecuzione dell'algoritmo di *pick and place*.

3.3 Arduino Explore IoT Kit: Architetture di Rete e Sistemi Distribuiti

Il kit Arduino Explore IoT[3] si compone di una scheda MKR WiFi 1010 e dello shield MKR IoT Carrier, il quale integra sensori ambientali (temperatura, umidità, pressione, luminosità), cinque pulsanti, cinque LED e un display LCD. L'interfaccia con le periferiche è gestita tramite la libreria `Arduino_MKRIoTCarrier`.

L'obiettivo didattico della sezione è l'introduzione alle architetture *client-server* nell'Internet delle Cose, realizzando un sistema distribuito locale composto dal dispositivo IoT (Arduino), da un server centrale (un Raspberry Pi preconfigurato) e da un client di monitoraggio su PC (un'applicazione console C#). Per focalizzare l'attenzione sulla logica applicativa sono state fornite agli studenti due librerie custom (`WebCall` per Arduino e `LibWebCall.dll` per C#) per l'incapsulamento delle richieste di rete. L'autenticazione e la separazione dei dati tra i vari gruppi sono state implementate sul server tramite un sistema a credenziali con generazione di una API Key personale.

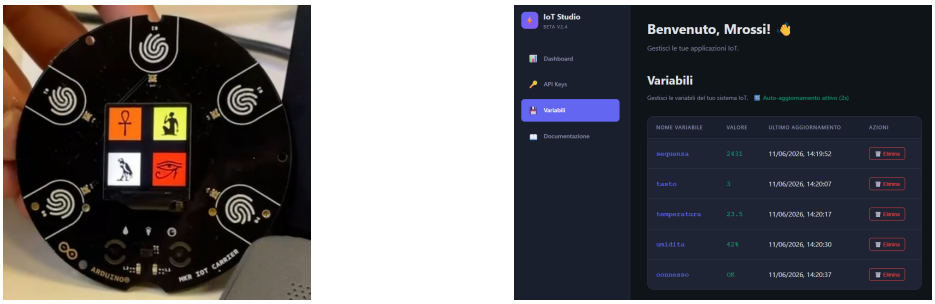
Il percorso di sviluppo è stato articolato in quattro task a complessità progressiva:

- **I/O locale:** acquisizione della temperatura dal sensore integrato e relativa visualizzazione sul display LCD dello shield.
- **Trasmissione dati al server:** connessione alla rete WiFi tramite la libreria `WiFiNINA` e invio periodico dei dati dal kit IoT al server.
- **Lettura remota:** sviluppo dell'applicazione in C# su Visual Studio per interrogare il server e visualizzare a console i parametri che il server ha ricevuto dalla scheda Arduino.
- **Scambio di informazioni complesse:** selezione di una sequenza di simboli grafici (geroglifici) sul display dello shield tramite pulsanti del kit, invio della sequenza al server e successiva ricezione sul PC (Fig. 3).

3.4 Strumenti nativi di Gamification in Moodle

Per l'implementazione delle dinamiche ludiche sulla piattaforma Moodle si è scelto di utilizzare esclusivamente le funzionalità native dell'ambiente, evitando l'installazione di plugin di terze parti (quali *Level Up XP* o *Block Game*). Questa decisione ha permesso di strutturare il percorso sfruttando due meccanismi principali:

[3] Il kit Arduino Explore Iot è descritto al link `https://store.arduino.cc/products/explore-iot-kit-rev2`



(a) Selezione dei geroglifici su IoT Explore. (b) Dashboard di controllo sul server locale.

Figura 3: Interfacce di input e monitoraggio dati per il modulo Internet of Things.

- **Progressione e Accesso Condizionato:** la sequenzialità delle sfide tecnologiche è stata gestita tramite vincoli e restrizioni di accesso alle attività. Il superamento di un task sbloccava la fase successiva, attivando contemporaneamente il rilascio di nuovi contenuti testuali e video legati alla narrazione della spedizione archeologica. Il monitoraggio visivo dell'avanzamento dei team è stato affidato al blocco nativo di completamento attività.
- **Sistemi di Ricompensa (*Badges*):** in linea con le metodologie di *gamification* applicate all'e-learning [14], l'assegnazione di punti esperienza o livelli è stata sostituita dal rilascio di *badge* digitali collezionabili (Fig. 4), erogati automaticamente sia al completamento dei singoli obiettivi intermedi sia al raggiungimento del traguardo finale da parte della squadra.



Figura 4: Badge digitali sviluppati per il monitoraggio e la ricompensa degli obiettivi raggiunti.

4 Risultati e Discussione

L'efficacia del Project Work è stata valutata distinguendo tra l'acquisizione di competenze tecniche (*hard skills*) e l'evoluzione delle abilità trasversali (*soft skills*). Nonostante la progettazione delle attività mirasse a stimolare diverse aree di competenza definite dal framework *DigComp 2.2* (Tabella 1), nel presente lavoro non è stata effettuata una misurazione formale di ciascuna area, che invece viene delegata alle ordinarie valutazioni svolte durante l'anno scolastico. L'obiettivo primario del Project Work risiedeva infatti nell'applicazione integrata delle conoscenze di programmazione Arduino all'interno di un progetto ad ampio respiro.

Per quanto concerne le *hard skills*, l'analisi quantitativa si basa sulla percentuale di completamento dei quattro task operativi (T1–T4) previsti per ciascun kit tecnologico, i cui dati sono riassunti nella Tabella 2.

I dati evidenziano come la percentuale di completamento diminuisca con l'aumentare della difficoltà dei task sia per *Ardusumo* che per *IoT Explore*. Una controtendenza si registra invece nel modulo *Braccio*: le percentuali inferiori nei task iniziali (T1, T2) indicano che la maggior parte dei team ha preferito concentrare le proprie risorse direttamente sullo sviluppo del task T3 (*Pick and Place*), identificato come l'obiettivo principale e più stimolante dell'attività.

Tabella 1: Mappatura delle competenze del framework DigComp 2.2 in relazione ai task richiesti

Competenze DigComp 2.2	**Task sfidanti**
2.2 Condividere informazioni attraverso le tecnologie digitali	**Trasversale:** l'utilizzo della piattaforma Moodle per consultare le informazioni fornite e caricare i risultati è implicato in tutte le piattaforme e relativi Task.
3.1 Sviluppare contenuti digitali	**Task T3 e T4 di ogni piattaforma:** un semplice video che dimostri la riuscita del task è parte dei materiali da consegnare al termine.
3.4 Programmazione	**Trasversale:** tutti i task richiedono, con livelli di difficoltà diversi, di sviluppare componenti software da eseguire su una piattaforma (o più di una, nel caso dei task T2, T3 e T4 di IoT Explore).
4.1 Proteggere i dispositivi	**IoT Explore, task T2, T3 e T4:** la necessità di generare delle chiavi API sul server fornito per poter stabilire una connessione protetta tra i dispositivi Arduino / PC e il server stesso.
5.1 Risolvere problemi tecnici	**Trasversale:** tutti i task richiedono di sviluppare capacità di risoluzione di problemi. In particolare, le piattaforme Ardusumo e Braccio presentano questioni relative anche ad aspetti di meccanica, oltre che di programmazione.

Tabella 2: Percentuale di completamento dei task operativi (T1–T4) per ciascuna piattaforma.

Task	**Ardusumo (%)**	**IoT Explore (%)**	**Braccio (%)**
T1 (Base)	86	93	50
T2 (Intermedio)	71	93	14
T3 (Avanzato)	21	78	79
T4 (Estensione)	7	43	43

A titolo di esempio sono riportati in Figura 5 due snippet di codice realizzati da due gruppi della stessa classe (3IA) per il task T3 della piattaforma Braccio. Il confronto tra i sorgenti evidenzia le differenti scelte logiche adottate dai gruppi di lavoro:

- **Interfaccia Utente e Flusso di Esecuzione:** Nel Listing 1 è stata definita una funzione `menu()` dedicata, richiamata continuamente all'inizio del `loop()` per guidare l'utente passo-passo. Al contrario, nel Listing 2 le istruzioni della guida vengono stampate una sola volta all'interno della funzione `setup()`.
- **Gestione dell'I/O Seriale:** Dal punto di vista della sincronizzazione, la soluzione di sinistra adotta una riga di codice bloccante `while (Serial.available() == 0);`, che sospende il programma in attesa dell'input. La soluzione di destra fa invece uso del costrutto non bloccante standard analizzato a lezione (`if (Serial.available() > 0)`) all'interno della funzione `loop()`, elaborando il flusso solo alla ricezione effettiva dei dati.
- **Astrazione dell'Hardware e Parsing dei Parametri:** Entrambi i gruppi eseguono una corretta parametrizzazione dell'hardware in termini di giunto ed angolo, valori che sono ottenuti tramite *parsing* della string letta da Serial. Una differenza tra le due soluzioni sta nel fatto che mentre nel Listing 2 la stringa da immettere è particolarmente lunga, nel Listing 1 viene proposta anche una "movimentazione rapida" con incrementi o decrementi prestabiliti di $\pm 10^\circ$, $\pm 20^\circ$).

In merito alle *soft skills*, l'osservazione diretta ha rilevato un incremento significativo del coinvolgimento e della collaborazione interpersonale. L'adozione del *Digital Storytelling* e della *Gamification* — a partire dal video introduttivo[4] — ha mantenuto alta la motivazione per tutta la durata del percorso, favorendo il *problem solving* e il mutuo soccorso all'interno delle classi. A testimonianza del lavo-

[4] Il video introduttivo al Project Work si può trovare al link https://youtu.be/VvHUZQ6ygoQ

```
void menu(){
  Serial.println("Scegli il giunto
    inserendo numero o nome:");
  Serial.println("1. BASE");
  Serial.println // ...
  if(contatoreMovimenti > 0)
    Serial.print("Usa '+', '++',
    '-' e '--'");
}
void loop() {
  menu();
  while (Serial.available()==0);
  stringa = Serial.readString();
  stringa.trim();
  // ...
  bool comandoValido = true;
  if (stringa == "+")
    arm.setOneRelative(giunto,10);
  else if // ...
  else {
    if (stringa == "base" ||
    stringa == "1")
      giunto = BASE_ROT;
    else // ...
    else { Serial.println("Comando
    non riconosciuto");
      comandoValido = false; }

    if (comandoValido) {
      Serial.print("Inserisci l'
    angolo:");
      while(Serial.available()==0);
      angolo = Serial.parseInt();
      arm.setOneAbsolute(giunto,
    angolo);
    }
  }
  arm.safeDelay(1000);
}
```

Listing 1: Approccio "sequenziale" con funzione menu

```
void setup() {
  Serial.begin(9600);
  // ...
  Serial.println("LISTA COMANDI:");
  Serial.println("ruotabase-valore
    ...");
  // ...
}

void loop() {
  arm.safeDelay(1000);
  if (Serial.available() > 0) {
    String comandoUtente = Serial.
    readString();
    int indiceTrattino =
    comandoUtente.indexOf('-');
    String comando = comandoUtente.
    substring(0, indiceTrattino);
    int angolo = comandoUtente.
    substring(indiceTrattino + 1).
    toInt();
    int giunto = 1000;

    if(comando == "ruotabase")
      giunto = BASE_ROT;
    else if(comando=="muovispalla")
      giunto = SHOULDER;
    // ...
    else
      Serial.println("Inserito un
    comando non riconosciuto");

    if(giunto != 100)
      arm.setOneAbsolute(giunto,
    angolo);
    arm.safeDelay(2000);
  }
}
```

Listing 2: Approccio con funzione loop()

Figura 5: Confronto tra due soluzioni del task Braccio–T3.

ro svolto nei diversi task, è stato realizzato un video riassuntivo[5] che raccoglie i contributi visivi e le dimostrazioni pratiche registrate direttamente dagli studenti durante le sessioni del Project Work.

Dal punto di vista della progettazione didattica, la sfida principale ha riguardato la rotazione dei kit hardware: è stato necessario elaborare un'architettura narrativa parallela (i tre percorsi Aria, Acqua e Terra) che garantisse la coerenza logica e il senso di progressione dei gruppi, pur operando su tecnologie differenti in giornate diverse.

Infine, si possono identificare tre principali barriere infrastrutturali ed organizzative:

- **Costi dell'acquisto dei kit:** nel nostro caso acquistati in più anni scolastici.

[5] Il video riassuntivo con i contributi degli studenti si può trovare al link https://youtu.be/nnY8R1L-1z8

- **Logistica delle aule:** necessità di spazi cablati per l'alimentazione dei laptop e dei dispositivi, risolta nel nostro caso con aule dedicate.
- **Tempistiche e valutazioni:** la collocazione del progetto a ridosso della pausa natalizia ha richiesto una flessibilità oraria da parte del consiglio di classe per calendarizzare efficacemente le prove di valutazione.

5 Conclusioni

Il Project Work basato sulla piattaforma Arduino e integrato in una cornice gamificata si è dimostrato un dispositivo didattico efficace per le classi terze dell'indirizzo Informatica. L'approccio basato sul *learning by doing* e supportato dal *digital storytelling* ha permesso di superare la frammentazione delle nozioni teoriche, offrendo agli studenti un contesto applicativo reale in cui far convergere le competenze di programmazione, robotica e reti. I dati di completamento dei task e il livello di partecipazione riscontrato confermano che la motivazione estrinseca, stimolata dalle dinamiche di gioco, favorisce l'inclusione e supporta i team anche nell'affrontare problematiche tecnologiche complesse.

Questa attività di Project Work è proposta ogni anno per le classi terze dell'Istituto, e alla luce dei risultati positivi riscontrati con questa modalità verrà riproposta anche per il prossimo anno scolastico, con obiettivi simili anche se in una diversa ambientazione.

Riferimenti bibliografici

[1] L. 107/2015. Riforma del sistema nazionale di istruzione e formazione e delega per il riordino delle disposizioni legislative vigenti, 2015.

[2] D. M. 774/2019. Linee guida in merito ai percorsi per le competenze trasversali e per l'orientamento, 2019.

[3] D. L. 127/2025. Misure urgenti per la riforma dell'esame di stato del secondo ciclo di istruzione e per il regolare avvio dell'anno scolastico 2025/2026, 2025.

[4] John W Thomas. A review of research on project-based learning, 2000.

[5] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15, 2011.

[6] Bernard Robin. The educational uses of digital storytelling. In *Society for information technology & teacher education international conference*, pages 709–716. Association for the Advancement of Computing in Education (AACE), 2006.

[7] Oleksii Voronkin and Sergiy Lushchin. Project method in stem education using arduino software and hardware platform. *Physical and Mathematical Education*, 38(4):24–30, Sep. 2023.

[8] John Dewey. *Experience and Education*. Kappa Delta Pi, New York, NY, 1938.

[9] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, NY, 1980.

[10] Joseph S. Krajcik and Phyllis C. Blumenfeld. Project-based learning. In *The Cambridge Handbook of the Learning Sciences*, pages 317–334. Cambridge University Press, 2006.

[11] Cheng-Huan Chen and Yong-Cih Yang. Revisiting the effects of project-based learning on students' academic achievement: A meta-analysis investigating moderators. *Educational Research Review*, 26:71–81, 2019.

[12] Pengyue Guo, Nadira Saab, Lysanne S Post, and Wilfried Admiraal. A review of project-based learning in higher education: Student outcomes and measures. *International journal of educational research*, 102:101586, 2020.

[13] Karl M Kapp. *The Gamification of Learning and Instruction: Game–Based Methods and Strategies for Training and Education*. John Wiley & Sons, 2012.

[14] Awaz Naaman Saleem, Narmin Mohammed Noori, and Fezile Ozdamli. Gamification applications in e-learning: A literature review. *Technology, Knowledge and Learning*, 27(1):139–159, 2022.

[15] Jason B. Ohler. *Digital Storytelling in the Classroom: New Media Pathways to Literacy, Learning, and Creativity*. Corwin Press, Thousand Oaks, CA, 2 edition, 2013.

Dallo schermo ai robot per imparare Python

Laura Cesaro[1] and Emanuele Menegatti[1]

Università degli Studi di Padova
laura.cesaro@phd.unipd.it
emanuele.menegatti@unipd.it

Abstract

Questo studio indaga il ruolo del robot come artefatto didattico per l'apprendimento della programmazione nella scuola secondaria. I risultati mostrano che il robot è percepito come capace di rendere il codice più concreto e osservabile. L'esperienza si associa inoltre a un incremento significativo della percezione di competenza e autoefficacia nella programmazione.

1 Introduzione

La robotica educativa è un ambito di ricerca ormai consolidato, che affonda le proprie radici teoriche nel costruzionismo di Seymour Papert [10]. Partendo dall'idea che il computer possa essere un "oggetto con cui pensare", ossia come un medium capace di sostenere processi di costruzione attiva della conoscenza [9], Papert arriva a introdurre un artefatto robotico, la tartaruga da programmare in Logo, per rendere visibile il pensiero. In seguito collabora con Lego per la realizzazione di Mindstorm, il primo kit robotico utilizzato in ambito educativo [12]. Il robot diventa in questo modo l'artefatto che rende tangibile l'attività di programmazione: il codice produce effetti visibili, osservabili e modificabili.

Il valore pedagogico di questo approccio risiede nella possibilità di trasformare l'istruzione programmata in un'azione concreta, favorendo l'apprendimento attraverso il fare, l'errore, la correzione e la riflessione. Le esperienze riportate nelle revisioni sistematiche più recenti [1, 3] ne evidenziano il potenziale nel sostenere apprendimento, motivazione, problem solving, collaborazione, inclusione e sviluppo del pensiero computazionale [7, 2, 4, 6]. Tuttavia, nella scuola secondaria di secondo grado l'apporto della robotica educativa all'insegnamento dell'informatica risulta ancora relativamente poco indagato.

Il presente studio prende quindi in esame un corso di programmazione in Python con LEGO Spike Prime rivolto a studentesse e studenti della scuola secondaria di secondo grado, condotto da docenti universitari e realizzato nell'ambito di un progetto di orientamento verso le discipline STEM con particolare attenzione alla riduzione del divario di genere. Il lavoro intende indagare il potenziale della robotica educativa nel sostenere l'apprendimento della programmazione, considerando sia il ruolo del robot come artefatto didattico sia le percezioni degli studenti rispetto al percorso svolto, attraverso le seguenti domande di ricerca:

RQ1. In che modo studentesse e studenti valutano l'utilità del robot per apprendere la programmazione?

RQ2. Quali cambiamenti emergono, nel corso dell'esperienza, nelle rappresentazioni dei robot, negli atteggiamenti verso la robotica e negli stereotipi di genere associati a questo ambito?

RQ3. Il corso incide sulla percezione di competenza e autoefficacia nella programmazione?

2 Metodologia

Il corso *Imparare a programmare in Python con i robot* è uno dei laboratori offerti dal Dipartimento di Ingegneria dell'Informazione dell'Università di Padova nell'ambito del progetto DIGA, finanziato dal Fondo per la Repubblica Digitale.

All'interno di una proposta più ampia caratterizzata dall'offerta di diversi laboratori in vari ambiti dell'ingegneria, il percorso incentrato su Python e sulla robotica educativa è stato proposto in tre edizioni. Realizzato in orario scolastico, l'intervento ha coinvolto un campione di studenti e studentesse delle classi terze e quarte di due licei padovani che hanno scelto di aderire a questa proposta. Sebbene il numero complessivo di partecipanti fosse più elevato, l'analisi ha considerato solo i 55 questionari PRE-POST che è stato possibile abbinare in modo affidabile. Il campione analizzato risulta composto da 30 studentesse, 23 studenti e 2 partecipanti che hanno scelto di non indicare il genere.

In ogni laboratorio, gli studenti e le studentesse hanno svolto 15 ore di attività hands-on, programmando in Python i robot LEGO Spike Prime. Attraverso un approccio orientato al *learning by doing*, i partecipanti hanno appreso i concetti fondamentali della programmazione finalizzati al controllo del movimento e del comportamento dei robot. La progressione delle attività è stata strutturata per guidare ciascuno nell'acquisizione dei costrutti necessari allo sviluppo di algoritmi di navigazione autonoma, finalizzati alla risoluzione di un labirinto come attività conclusiva del percorso.

I dati sono stati raccolti mediante la somministrazione di un questionario PRE e uno POST intervento. Entrambi gli strumenti includevano domande aperte per la raccolta di associazioni libere e informazioni sulle esperienze pregresse, item adattati dalla scala RAAS per la misurazione dell'atteggiamento verso la robotica [5] e item per la rilevazione degli stereotipi di genere in questo ambito [8, 11]. Il questionario POST includeva inoltre una domanda aperta in cui si chiedeva agli studenti una risposta motivata sulla conferma o meno della scelta di seguire un corso basato su programmazione e robotica, oltre a una sezione dedicata alla valutazione delle attività proposte.

Analisi dei dati Le analisi hanno integrato metodi qualitativi e quantitativi. Le risposte aperte sono state esaminate mediante analisi del contenuto e codificate secondo le principali categorie emerse. Le variazioni tra PRE e POST sono state valutate con test non parametrici per campioni dipendenti. I punteggi Likert relativi ad atteggiamenti, stereotipi e percezione di competenza sono stati analizzati con test t per campioni appaiati, sia sul campione complessivo sia per genere.

3 Results

3.1 RQ1 — Il robot come strumento per rendere visibile e concreta la programmazione

Per rispondere alla prima domanda di ricerca, è stato analizzato il rapporto tra le ragioni alla base della scelta del corso e la valutazione dell'esperienza svolta, con particolare attenzione alla conferma delle aspettative iniziali. Le risposte sono state codificate con una variabile binaria, distinguendo la presenza o assenza di riferimenti alla robotica come mezzo per rendere visibile e concreta la programmazione. Il confronto tra PRE e POST mostra un aumento significativo dei riferimenti alla robotica come elemento che rende la programmazione visibile: da 22 a 37 risposte su 55 (test di McNemar, $p = .003$). Questo incremento è riconducibile soprattutto alle 19 persone che introducono il tema solo nel POST, a fronte di 4 che lo citano solo nel PRE. Pur tenendo conto della diversa formulazione delle due domande, le risposte suggeriscono che l'esperienza abbia contribuito a rendere più evidente il ruolo del robot nel visualizzare gli effetti della codifica.

Le risposte POST sono state poi classificate in base alla ragione principale indicata per confermare, o meno, la scelta del laboratorio (Tabella 1). Come mostra la tabella, la visibilità e concretezza costituiscono la motivazione prevalente, raccogliendo oltre la metà delle risposte, mentre il coinvolgimento motivazionale rappresenta la seconda ragione più frequente. Le valutazioni negative e le conferme non motivate risultano marginali. Nel complesso, questi dati suggeriscono che l'esperienza abbia reso più evidente il ruolo del robot come artefatto didattico capace di rendere osservabile il rapporto tra scrittura del codice e risultati prodotti.

Table 1: Ragioni emerse nella conferma della scelta del laboratorio

Motivo principale	**N**	**%**
Visibilità/concretezza	34	61,8
Coinvolgimento/motivazione	12	21,8
Conferma positiva non motivata	2	3,6
Negativo	2	3,6
Altro	5	9,1
Totale	**55**	**100**

Nota. Nella categoria Visibilità/concretezza sono incluse 5 risposte che presentano anche un riferimento secondario al coinvolgimento.

3.2 RQ2 — Cambiamenti nelle rappresentazioni, negli atteggiamenti e negli stereotipi di genere

Per rispondere alla seconda domanda di ricerca, l'analisi ha preso in considerazione tre dimensioni: le rappresentazioni dei robot espresse nelle risposte aperte, gli atteggiamenti verso la robotica rilevati tramite item adattati dalla scala RAAS e gli stereotipi di genere associati alla robotica.

Rappresentazioni del robot

Le risposte aperte alla domanda "Cosa pensi quando senti la parola robot?" sono state analizzate mediante un sistema di codifica a tag multipli. Le risposte sono state codificate distinguendo i riferimenti all'immaginario, quando il robot era evocato attraverso rappresentazioni culturali, mediatiche o fantascientifiche; all'esperienza pregressa, quando comparivano richiami a esperienze dirette con robot o piattaforme specifiche; all'innovazione e al futuro, quando il robot era associato al progresso tecnologico o al mondo del lavoro; alla componente tecnico-meccanica, quando la risposta descriveva il robot come oggetto fisico, i suoi componenti, movimenti o comportamenti sensomotori; e al computing, quando emergevano riferimenti alla programmazione, al codice, agli algoritmi, all'intelligenza artificiale o al pensiero computazionale. Le risposte vuote o non pertinenti sono state classificate come NC.

Table 2: Tag nelle risposte alla domanda "Cosa pensi quando senti la parola robot?" nel PRE e nel POST (N = 55)

Tag	**n PRE**	**% PRE**	**n POST**	**% POST**	**Δ**	**p**
Immaginario	7	12,7	0	0,0	−7	.008 **
Esperienza pregressa	3	5,5	2	3,6	−1	n.s.
Futuro/innovazione	13	23,6	12	21,8	−1	n.s.
Tecnico/meccanico	28	50,9	37	67,3	+9	.029 *
Computing	35	63,6	41	74,5	+6	n.s.
NC	2	3,6	2	3,6	0	n.s.

Nota. Le categorie non sono mutuamente esclusive: una stessa risposta poteva corrispondere a più tag. I valori di p si riferiscono al test McNemar per il confronto PRE-POST su dati appaiati. * $p < .05$; ** $p < .01$.

Il confronto PRE-POST evidenzia due variazioni significative (Tabella 2). La categoria Immaginario scompare completamente nel POST, mentre la categoria Tecnico/meccanico cresce in modo significativo, indicando una maggiore presenza di riferimenti a elementi fisici. La categoria Computing era già ampiamente presente nel PRE e rimane stabile nel POST.

Nel complesso, questi risultati indicano che, dopo l'esperienza, le rappresentazioni del robot risultano meno legate all'immaginario culturale e più orientate verso una sua concezione come artefatto programmabile, il cui comportamento può essere osservato e collegato alle istruzioni del codice.

Atteggiamenti verso la robotica

L'analisi degli atteggiamenti verso la robotica si è basata su quattro indicatori costruiti a partire da item selezionati e adattati dalla scala RAAS. La competenza percepita riguarda la valutazione delle proprie capacità nello svolgere attività legate alla robotica e alla programmazione; il potenziale di apprendimento si riferisce alla percezione che la robotica sia un ambito in cui sia possibile imparare, migliorare e sviluppare nuove competenze; l'interesse personale verso robotica e tecnologia raccoglie item relativi al rapporto individuale con questo ambito, come la ricerca di informazioni, l'avvicinamento ad attività tecnologiche e il riconoscimento di una certa familiarità personale; la curiosità, infine, riguarda il desiderio di esplorare e comprendere il funzionamento dei robot (Tabella 3).

Table 3: Indicatori di atteggiamento verso la robotica prima e dopo il corso, per genere

	Totale (N = 55)			**Femmine** (N = 30)			**Maschi** (N = 23)		
Indicatore	**M pre**	**M post**	**p**	**M pre**	**M post**	**p**	**M pre**	**M post**	**p**
Competenza perc	3,15	3,73	***	3,17	3,54	*	3,14	3,97	***
Potenziale di appr	3,43	3,60	n.s.	3,46	3,42	n.s.	3,46	3,86	n.s.
Interesse personale	1,80	2,27	**	1,81	1,89	n.s.	1,85	2,82	***
Curiosità	3,26	3,32	n.s.	3,30	3,13	n.s.	3,33	3,61	n.s.

Nota. Scala 1–5. Test t per campioni appaiati. Le analisi per genere non includono i 2 partecipanti che hanno scelto di non indicare il genere. Gli indicatori sono stati costruiti a partire da item selezionati e adattati dalla scala RAAS. * $p < .05$; ** $p < .01$; *** $p < .001$.

Nel campione complessivo aumentano significativamente la competenza percepita e l'interesse personale verso robotica e tecnologia, mentre il potenziale di apprendimento e la curiosità restano sostanzialmente stabili. Il confronto per genere mostra un andamento differenziato: la competenza percepita cresce in modo significativo in entrambi i gruppi, mentre l'aumento dell'interesse personale verso robotica e tecnologia riguarda solo gli studenti maschi. I dati suggeriscono quindi che l'esperienza abbia rafforzato in entrambi i gruppi la percezione di poter affrontare attività robotiche, ma che l'avvicinamento personale alla robotica e alla tecnologia risulti più marcato tra i maschi.

Stereotipi di genere

L'analisi degli item relativi alle credenze su partecipazione, interesse e capacità di ragazze e ragazzi nella robotica non evidenzia variazioni significative tra PRE e POST. Le rappresentazioni di genere associate alla robotica risultano stabili prima e dopo l'esperienza, va tuttavia notato che già nel PRE emerge un'asimmetria a sfavore delle ragazze che permane invariata nel POST, suggerendo che un intervento breve non sia sufficiente a modificare credenze consolidate.

3.3 RQ3 — Percezione di competenza e autoefficacia nella programmazione

Per rispondere alla terza domanda di ricerca sono stati analizzati tre item relativi alla percezione di competenza nella programmazione (Tabella 4). L'item "Posso imparare a scrivere un programma" non presenta variazioni significative perché già nella rilevazione PRE i partecipanti si sentivano ampiamente capaci di farlo (M = 4,16 su scala 1–5), lasciando un ridotto margine di crescita *(ceiling effect)*. I risultati mostrano invece un aumento significativo per "Posso scrivere un programma" (Δ = +0,69, p = ,001) e "Riesco a programmare un robot" ($\Delta = +1{,}76$, $p < 0{,}001$).

Table 4: Item relativi alla percezione di competenza nella programmazione

Item	M pre	DS pre	M post	DS post	Δ	p
Posso imparare a scrivere un programma	4,16	0,81	4,16	0,79	0,00	n.s.
Posso scrivere un programma	2,98	1,37	3,67	0,79	+0,69	**
Riesco a programmare un robot	1,67	0,79	3,44	0,66	+1,76	***

Nota. Scala 1–5. Test t per campioni appaiati. ** $p < .01$; *** $p < .001$.

I dati suggeriscono che l'esperienza abbia contribuito a tradurre una fiducia generica nella propria capacità di apprendere in una percezione di competenza pratica e specifica, legata in modo diretto a quanto affrontato nel corso. L'aumento più marcato nell'item relativo alla programmazione del robot conferma il ruolo dell'esperienza pratica nel rafforzare la percezione di competenza. Il confronto per genere mostra che entrambi i gruppi raggiungono guadagni significativi su entrambi gli item, con valori di partenza molto simili. L'unica differenza degna di nota è che le femmine raggiungono un livello leggermente più basso nel POST sull'item "Riesco a programmare un robot" (M_post=3.27 vs 3.65 per i maschi), a fronte di un punto di partenza identico.

4 Conclusione

Pur con le dovute cautele, legate soprattutto alla partecipazione su base volontaria, all'assenza di un gruppo di controllo e alla mancanza di una misurazione dell'apprendimento effettivo, i risultati suggeriscono che la robotica educativa possa sostenere l'apprendimento della programmazione nella scuola secondaria di secondo grado.

I dati raccolti mostrano una convergenza tra le trasformazioni osservate nelle rappresentazioni dei robot, l'aumento della percezione di competenza nella programmazione e le motivazioni espresse da studentesse e studenti a conferma della scelta della tipologia di laboratorio nel questionario POST. In particolare, il robot sembra assumere un ruolo rilevante perché rende visibile il rapporto tra codice e azione: ciò che viene scritto nel programma produce effetti fisici osservabili. In questo senso, il robot non funziona soltanto come elemento motivazionale, ma come artefatto didattico che può rendere più concreta e significativa l'esperienza di programmare. Gli stereotipi di genere associati alla robotica risultano invece sostanzialmente stabili. Questo dato evidenzia una discrepanza: da un lato, l'esperienza sembra aver favorito una maggiore fiducia nelle proprie competenze nella robotica e una più alta percezione di accessibilità della programmazione testuale in Python; dall'altro, non emerge un cambiamento altrettanto evidente nelle rappresentazioni e negli orientamenti più consolidati.

Nel complesso, lo studio mostra che la robotica educativa può sostenere l'apprendimento della programmazione rendendo il codice più visibile, concreto e accessibile, e contribuendo alla percezione di competenza e autoefficacia di studentesse e studenti. Il robot emerge così come un possibile alleato didattico per promuovere percorsi di programmazione a scuola, soprattutto se inserito in percorsi continuativi e integrati nel curriculum.

References

[1] Sajid Anwar, Nadeene A. Bascou, Muhsin Menekse, and Amir Kardgar. A systematic review of studies on educational robotics. *Journal of Pre-College Engineering Education Research (J-PEER)*, 9(2):Article 2, 2019.

[2] Marco Beltrametti, Lorella Campolucci, Danila Maori, Lucio Negrini, and Silvia Sbaragli. La robotica educativa per l'apprendimento della matematica. un'esperienza nella scuola elementare. *Didattica della matematica. Dalla ricerca alle pratiche d'aula*, pages 123–144, 2017.

[3] Giovanni Bonaiuti, Lorella Campitiello, Stefano Di Tore, and Antonio Marras. Educational robotics studies in italian scientific journals: A systematic review. *Frontiers in Education*, 7:1005669, 2022.

[4] Laura Cesaro and Emanuele Menegatti. Robot design and inclusive practices: A pilot study on gender equity in stem. In *Social Robotics + AI*, pages 663–673, Singapore, 2026. Springer Nature Singapore.

[5] Jennifer Cross, Emily Hamner, Lauren Zito, Illah Nourbakhsh, and Debra Bernstein. Development of an assessment for measuring middle school student attitudes towards robotics activities. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, 2016.

[6] Salih Gülen, İsmail Dönmez, Fatma Şengönül, Miyase Aslantaş, Turgut Saritaş, Ömer Sukenari, Emrah Eke, and Semih Uçar. The effect of robotics and coding education on girls' stem motivation, attitude and career aspirations. *SAGE Open*, 15, 2025.

[7] Christian Langer and Christine Strothotte. The benefits of integrating lego mindstorms into design education: Course "media systems". In *Proceedings of E&PDE 2007, the 9th International Conference on Engineering and Product Design Education*, DS 43, Newcastle, UK, 2007.

[8] Fiona Law, Luke McGuire, Mark Winterbottom, and Adam Rutland. Children's gender stereotypes in stem following a one-shot growth mindset intervention in a science museum. *Frontiers in Psychology*, 12:641695, 2021.

[9] Michele Moro, Emanuele Menegatti, Filippo Sella, and Marco Perona. *Imparare con la robotica: applicazioni di problem solving*. Erickson, Trento, 2011.

[10] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.

[11] Diana Tang, Andrew N. Meltzoff, Sapna Cheryan, Weihua Fan, and Allison Master. Longitudinal stability and change across a year in children's gender stereotypes about four different stem fields. *Developmental Psychology*, 60(6):1109–1130, 2024.

[12] Memet Üçgül. History and educational potential of lego mindstorms nxt. *Mersin Üniversitesi Eğitim Fakültesi Dergisi*, 9, 2013.

Dal codice al comportamento: robotica sociale con NAO per l'apprendimento di programmazione, debugging e interazione uomo-macchina

Veronica Cavicchi[1], Matteo Bettini[1], Nikolai Carnaghi[1], Ibrahim Mohamad Barghouthi[1], Gabriele Chesini[1], Lorenzo Perotti[1], Mariapia Anna Porcelli[1], and Sunaina Ayushi Rani[1]

Liceo Scientifico di Stato "A. Calini", Brescia, Italia
veronica.cavicchi@liceocalini.edu.it

Sommario

Il contributo descrive l'esperienza GEM NAO GIMMICK, in cui la programmazione del robot NAO è stata usata per apprendere l'Informatica attraverso algoritmi, debugging, input/output, interazione uomo-macchina, testing, backend Python e Choregraphe. Il percorso trasforma una sfida di robotica sociale in attività disciplinare centrata su codice, errore, affidabilità, accessibilità culturale e documentazione tecnica. Si esplicitano l'accompagnamento didattico, i contributi degli studenti e i limiti delle evidenze qualitative.

1 Contesto e finalità

Il contributo presenta un'esperienza svolta in un liceo scientifico tradizionale, all'interno di percorsi laboratoriali pomeridiani annuali di robotica educativa, informatica e intelligenza artificiale. GEM NAO GIMMICK è stato sviluppato per la NAO Challenge 2026, sul tema "NAO on stage", in collaborazione con il Museo Santa Giulia. La scena del balcone di *Romeo e Giulietta* ha offerto un contesto per programmare NAO come mediatore culturale accessibile, integrando traduzione, sottotitoli e contenuti in Lingua dei Segni Italiana (LIS). L'obiettivo, ispirato alla progettazione universale, è offrire più canali di accesso alla stessa esperienza culturale. La finalità non era semplicemente usare un robot come oggetto motivante, ma trasformarlo in un ambiente concreto per comprendere concetti fondamentali dell'Informatica come disciplina scientifica.

Il percorso prosegue le attività del laboratorio GEM, già documentate in ambito inclusivo attraverso GEM FLEXNAO [3]. Il focus viene ristretto intenzionalmente al nucleo disciplinare dell'Informatica: codice, comportamento, errore, test, affidabilità e documentazione tecnica. La robotica sociale rende visibile il passaggio dal codice al comportamento: un'istruzione produce movimento, voce, risposta a un evento o fallimento osservabile. Questo consente di affrontare decomposizione, astrazione, progettazione algoritmica e controllo dell'errore [9, 2].

2 Contenuti informatici trattati

L'esperienza è stata progettata distinguendo i contenuti propri dell'Informatica dagli aspetti trasversali legati alla collaborazione, alla comunicazione e all'orientamento. I principali contenuti disciplinari sono stati: sequenze algoritmiche; programmazione guidata da eventi; variabili e parametri; modularizzazione; gestione di input e output; interazione uomo-macchina; debugging; testing progressivo; documentazione tecnica; rapporto tra software, hardware e ambiente.

Un comportamento non riuscito può dipendere da una sequenza logica sbagliata, da un tempo di attesa insufficiente, da un conflitto tra moduli, da una postura instabile o da una condizione ambientale.

Gli studenti hanno lavorato sulla formulazione di ipotesi diagnostiche, sull'isolamento dei sottoproblemi e sulla verifica delle correzioni. Il debugging è stato trattato come pratica epistemica, coerentemente con la sua centralità nella formazione al pensiero computazionale [7].

3 Descrizione dell'esperienza

3.1 Accompagnamento didattico e organizzazione

Il percorso si è sviluppato attraverso una metodologia laboratoriale basata su sfide progettuali. La docente ha collegato la consegna agli obiettivi disciplinari: ideare comportamenti orientati a uno scopo comunicativo, tradurli in sequenze eseguibili, testarli sul dispositivo reale e documentarne il processo.

L'accompagnamento ha riguardato la decomposizione dell'intenzione comunicativa in stati, transizioni, condizioni, tempi, input previsti e output attesi. Gli interrogativi affrontati erano: quale evento avvia il comportamento? Quali dati devono essere rappresentati? Come controllare il flusso di esecuzione? Nella revisione dei malfunzionamenti, l'attenzione è stata riportata alla distinzione tra errore logico, vincolo meccanico e condizione ambientale, modificando una variabile alla volta e verificandone gli effetti. La discussione delle soluzioni e la documentazione hanno richiesto di spiegare il funzionamento dei moduli, oltre a mostrarne l'esito. La tabella 1 rende esplicito il collegamento tra attività e contenuti disciplinari.

Il team di progetto comprendeva nove studenti, due ragazze e sette ragazzi, con ruoli complementari. Bettini curava coordinamento tecnico e backend; Carnaghi moduli Python, servizi web e integrazione LIS; Porcelli programmazione Choregraphe e video; Rani videomaking, social e narrazione visuale; Barghouthi e Chesini storytelling e video; Perotti blog e presentazione; Astik Gautam blog e contenuti digitali; Amir Shams Eddin grafica e comunicazione. La docente assicurava coordinamento didattico e supervisione metodologica. I ruoli non erano separazioni rigide, ma modi per rendere esplicita la filiera dall'idea all'artefatto funzionante.

3.2 Architettura e comportamenti programmati

Il sistema integra NAO V6, Choregraphe 2.8.6.23 con box Python 2.7, un backend Python 3 e un servizio di riproduzione multimediale esterno. La scheda tecnica pubblicata dal team documenta interfacce e dipendenze [4]. Il server usa `BaseHTTPRequestHandler` e `HTTPServer`; `asyncio` e `subprocess` supportano specifiche operazioni. Le richieste HTTP collegano quattro funzioni:

- `/reco`: audio WAV codificato in Base64, trascrizione con `SpeechRecognition` e Google Web Speech;
- `/trans`: traduzione nella lingua richiesta tramite `googletrans`;
- `/sub`: sintesi con `pyttsx3`, montaggio con FFmpeg e MoviePy, sincronizzazione dei sottotitoli con `forcealign`;
- `/lis`: lemmatizzazione con `simplemma`, ricerca nella cache `aliases.json`, recupero dei video con `yt-dlp` e ripiego sulla dattilologia quando manca un segno.

NAO gestisce voce e gesti; i contenuti visivi sono riprodotti sullo schermo scenico tramite VLC. La concatenazione di video-segni è un prototipo di supporto lessicale, non una traduzione grammaticale della LIS validata. Anche latenza e fruibilità richiedono verifiche dedicate.

Le cinque missioni programmate comprendono presa del microfono, raggiungimento del centro palco, posizionamento sulla stella guida, interazione con gli attori e attivazione del faro. `ALMotion` e

`ALRobotPosture` governano movimento e postura; distanze e rotazioni sono parametri da calibrare. HaloCode, programmato con mBlock e collegato al sistema di rilevamento visivo, attiva i LED al riconoscimento del riferimento verde.

Nelle sequenze progettate, l'avanzamento verso il microfono è parametrizzato da `moveTo(0.70, 0.0, 0.0)`; gli altri spostamenti prevedono 0,26 e 0,75 m, con rotazioni di 180 e 90 gradi. La presa richiede apertura e chiusura della mano destra. Questa traduzione ha aiutato gli studenti a comprendere che l'Informatica non coincide con l'uso di strumenti digitali, ma riguarda la progettazione rigorosa di processi rappresentabili ed eseguibili.

3.3 Troubleshooting e affidabilità

Durante i test, un problema meccanico ha richiesto l'interruzione dell'attività e la revisione delle procedure di movimentazione. Il team ha progettato e stampato in PLA un componente sostitutivo; il tentativo non ha ripristinato la funzionalità dell'articolazione. È stata quindi adottata una configurazione statica, mantenendo le funzioni vocali e multimediali. L'episodio ha reso concreto il controllo congiunto di programma, corpo meccanico e ambiente operativo, dimensioni rilevanti nell'impiego educativo dei robot fisici [8].

Un programma che funziona una volta non è necessariamente robusto: ripetere prove, annotare configurazioni e controllare dipendenze è diventato parte del lavoro. La documentazione tecnica ha contribuito a rendere il processo comprensibile e replicabile. La gestione del rischio ha reso visibile la possibilità di preservare la funzione comunicativa anche quando la componente motoria non è utilizzabile.

4 Evidenze osservate e trasferibilità

Le evidenze sono qualitative: osservazione delle attività, prodotti realizzati, discussioni tecniche e documentazione degli studenti. Si sono osservati progressi nel leggere l'errore, formulare ipotesi, passare da una soluzione intuitiva a una procedura controllata e collegare il comportamento del robot alla struttura del programma. Particolarmente rilevante è stato il passaggio da un atteggiamento iniziale centrato sull'effetto finale a un'attenzione crescente per struttura, test, manutenzione e leggibilità del codice. La tabella organizza questi riscontri; non costituisce una misurazione standardizzata degli apprendimenti.

Tabella 1: Attività documentate e apprendimenti osservabili.

Nucleo	Artefatto o attività	Riscontro disciplinare
Modularizzazione	Backend con endpoint distinti	Scomporre il problema e spiegare ingressi e uscite dei moduli.
Controllo	Sequenze Choregraphe e calibrazione	Collegare parametri, movimento e condizioni fisiche.
Debugging	Prove, correzioni e configurazione statica	Distinguere cause software e hardware; motivare una soluzione alternativa.
Documentazione	Scheda tecnica e discussioni	Descrivere funzionamento, dipendenze e limiti del sistema.

La partecipazione femminile emerge da responsabilità concrete: Porcelli ha lavorato sulla programmazione robotica, Rani sulla produzione e comunicazione dei contenuti digitali. Il resoconto del gruppo descrive una cooperazione quotidiana fondata su interessi comuni e complementarità delle competenze. Questi elementi documentano partecipazione e accesso ai compiti; non consentono di quantificare variazioni dell'interesse per l'Informatica, dell'autoefficacia o delle intenzioni universitarie. Non sono

disponibili misure pre/post accompagnate da una metodologia verificabile, né un confronto che permetta attribuzioni causali. Per una valutazione successiva sarebbero utili prove comuni di programmazione, analisi sistematiche dei log di debugging e rilevazioni individuali dell'interesse.

Il modello è trasferibile a microcontrollori, simulatori e robot meno complessi: progettare un comportamento, formalizzarlo, implementarlo, provarlo, correggerlo e documentarlo. La separazione tra backend e dispositivo favorisce il riuso. Il valore didattico richiede comunque un'integrazione pedagogica esplicita, come richiamato dalla letteratura sui robot sociali [1, 6].

5 Conclusioni

La robotica sociale può offrire un contesto per l'apprendimento dell'Informatica quando viene usata come sistema programmabile da comprendere, controllare e migliorare. L'elemento più significativo è il passaggio dal "far funzionare qualcosa" al comprendere perché funziona, quando non funziona e come renderlo più robusto. Questa trasformazione è centrale nella didattica dell'Informatica: formare studenti capaci di analizzare, progettare e argomentare sistemi computazionali. L'incontro tra teatro, accessibilità e progettazione informatica ha dato uno scopo concreto al codice, preservando la distinzione tra finalità inclusive ed efficacia dimostrata. Il percorso è coerente con creazione di contenuti mediante programmazione, problem solving e attenzione all'intelligenza artificiale nel DigComp 3.0 [5].

Riferimenti bibliografici

[1] Belpaeme, T., Kennedy, J., Ramachandran, A., Scassellati, B., & Tanaka, F. (2018). Social robots for education: A review. *Science Robotics*, 3(21), eaat5954. https://doi.org/10.1126/scirobotics.aat5954

[2] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*.

[3] Cavicchi, V. (2025). GEM FLEXNAO: Tecnologia in movimento per il benessere del domani. In Autori Vari, *Buone prassi Q 2025. Esperienze di inclusione presentate al Convegno "La Qualità dell'inclusione scolastica e sociale 2025"*. EricksonLIVE / Edizioni Centro Studi Erickson.

[4] Carnaghi, N., Team GEM NAO GIMMICK, & Liceo Scientifico di Stato "A. Calini" (2026). Scheda tecnica GEM NAO Gimmick. *GEM – Ecomentality.it*, 25 maggio. https://www.ecomentality.it/2026/05/25/scheda-tecnica-gem-nao-gimmick/

[5] Cosgrove, J., & Cachia, R. (2025). *DigComp 3.0: European Digital Competence Framework. Fifth edition*. Publications Office of the European Union. https://doi.org/10.2760/0001149

[6] de Winter, J. C. F., Dodou, D., Moorlag, F., & Broekens, J. (2026). Social robots: A meta-analysis of learning outcomes. *Frontiers in Robotics and AI*, 12, 1735198. https://doi.org/10.3389/frobt.2025.1735198

[7] Sun, C., Yang, S., & Becker, B. (2024). Debugging in computational thinking: A meta-analysis on the effects of interventions on debugging skills. *Journal of Educational Computing Research*, 62(4), 1087–1121. https://doi.org/10.1177/07356331241227793

[8] Wang, H., Luo, N., Zhou, T., & Yang, S. (2024). Physical robots in education: A systematic review based on the technological pedagogical content knowledge framework. *Sustainability*, 16(12), 4987. https://doi.org/10.3390/su16124987

[9] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. https://doi.org/10.1145/1118178.1118215

L'intelligenza artificiale e i progetti Wikimedia: tra infrastruttura della conoscenza, sostenibilità e governance umana

Piergiovanna Grossi[1], Lorenzo Losa[2]
[1] Wikimedia Italia
me@piergiovannagrossi.it
[2] Wikimedia Italia
lorenzo.losa@wikimedia.it

Abstract

L'affermazione dell'intelligenza artificiale generativa sta modificando profondamente il ruolo dei progetti Wikimedia all'interno dell'ecosistema informativo globale. Wikipedia e gli altri progetti Wikimedia costituiscono una delle principali fonti utilizzate per l'addestramento e l'aggiornamento dei modelli linguistici di grandi dimensioni, ma allo stesso tempo subiscono gli effetti del cambiamento nelle modalità di accesso all'informazione da parte degli utenti. Il contributo analizza tre aspetti: l'uso dei contenuti Wikimedia da parte dei sistemi di IA e le conseguenze sull'infrastruttura tecnica; la riduzione della visibilità di Wikipedia dovuta ai sistemi di risposta automatica; l'impiego dell'IA nei progetti Wikimedia a supporto delle comunità. Presentato in prospettiva educativa, il caso Wikimedia offre a docenti e studenti/studentesse strumenti per discutere fonti, attribuzione, sostenibilità delle infrastrutture aperte, affidabilità dei contenuti e governance dell'automazione. L'obiettivo è favorire una consapevolezza critica del rapporto tra conoscenza collaborativa e IA, mettendone in luce rischi e opportunità.

Nota. Il presente contributo rappresenta una fotografia dello stato dell'arte nel mese di giugno 2026. In considerazione della rapida evoluzione dell'intelligenza artificiale e delle politiche dei progetti Wikimedia, alcune informazioni e valutazioni potrebbero essere soggette a un rapido aggiornamento.

1 Introduzione

Da oltre vent'anni Wikimedia rappresenta uno dei più importanti esempi di produzione collaborativa della conoscenza su scala globale. I suoi contenuti, pubblicati con licenze libere e costruiti da una comunità internazionale di volontari, costituiscono una risorsa fondamentale per la ricerca, l'educazione e la diffusione del sapere.

Negli ultimi anni l'emergere dell'intelligenza artificiale generativa ha cambiato il contesto in cui Wikimedia opera, ponendo sfide ad ampio spettro. Secondo il piano annuale della Wikimedia Foundation 2026-2027, il cambiamento non è più una prospettiva futura ma una trasformazione strutturale già in corso, caratterizzata da una diminuzione del traffico proveniente dai motori di ricerca tradizionali e da un aumento senza precedenti del traffico generato da bot e sistemi automatici [1].

Il contributo è rivolto in particolare ai docenti, chiamati a discutere con studenti e studentesse da quali fonti i sistemi generativi ricavino informazioni e quali conseguenze produca il loro uso. Wikimedia è un osservatorio privilegiato: combina un'infrastruttura tecnica con un modello di conoscenza libera e governata da comunità. La sua interazione con l'IA mostra come la qualità dell'informazione dipenda ancora dal lavoro umano di selezione, verifica e manutenzione delle fonti.

2 Wikimedia come infrastruttura per l'intelligenza artificiale

I contenuti Wikimedia sono ampiamente utilizzati per l'addestramento, la valutazione e il miglioramento dei modelli linguistici. La qualità delle fonti, la presenza di numerose lingue e la disponibilità di dati strutturati rendono Wikipedia una delle basi informative più utilizzate dalla comunità dell'intelligenza artificiale [2].

Tuttavia, l'accesso ai contenuti da parte dei sistemi di IA presenta caratteristiche molto diverse rispetto alla consultazione umana. Un lettore utilizza normalmente le copie presenti nelle cache distribuite della rete; i crawler e i sistemi automatici, invece, effettuano richieste massive e continue che provocano la rigenerazione frequente delle cache e un carico significativo sui server.

Questo fenomeno produce conseguenze sia tecniche sia economiche. Da un lato aumenta il consumo di risorse computazionali e di banda; dall'altro richiede investimenti infrastrutturali sempre maggiori per garantire la continuità del servizio. Il problema assume inoltre una dimensione etica. I volontari che scrivono le voci e i donatori che finanziano Wikimedia contribuiscono per favorire l'accesso universale alla conoscenza, non per sostenere indirettamente i modelli di business di poche grandi aziende tecnologiche.

Per affrontare questa situazione, la Wikimedia Foundation sta implementando un modello di “responsible reuse”, introducendo misure tecniche per distinguere il traffico umano da quello automatico e indirizzando gli utilizzatori commerciali verso Wikimedia Enterprise, un servizio progettato per fornire accesso strutturato ai dati da parte delle organizzazioni che riutilizzano contenuti Wikimedia su larga scala. L'obiettivo è distinguere l'accesso pubblico gratuito dall'utilizzo commerciale intensivo, contribuendo alla sostenibilità economica e tecnica dell'ecosistema Wikimedia [1][3].

3 L'accesso all'informazione nell'era delle risposte generate dall'IA

Un secondo cambiamento riguarda il modo in cui gli utenti accedono alle informazioni. Sempre più frequentemente le persone ottengono risposte direttamente da assistenti conversazionali e motori di ricerca dotati di funzionalità generative, senza visitare le fonti originali.

Questo fenomeno, spesso definito "zero-click experience", riduce la visibilità di Wikipedia e dei progetti Wikimedia. La conseguenza è un progressivo spostamento del valore dalla produzione della conoscenza alla sua intermediazione algoritmica. Il piano annuale della Wikimedia Foundation evidenzia come la diminuzione delle visite e del traffico proveniente dai motori di ricerca rappresenti una delle principali sfide strategiche per il futuro del movimento [1].

Le conseguenze non sono solamente quantitative. Quando l'utente riceve una risposta sintetica generata da un sistema automatico perde la possibilità di verificare direttamente le fonti, comprendere il processo editoriale e partecipare eventualmente alla costruzione della conoscenza. Viene inoltre meno la consapevolezza che dietro i contenuti esiste una comunità di persone che discute, corregge e aggiorna continuamente le informazioni.

In questo contesto Wikimedia assume un ruolo peculiare. A differenza delle piattaforme proprietarie, Wikipedia è un progetto aperto, collaborativo e governato da comunità umane distribuite a livello globale. La sua rilevanza non dipende soltanto dalla qualità dei contenuti, ma anche dal modello sociale che ne garantisce la produzione.

Le possibili risposte comprendono strategie di attribuzione più efficaci, accordi con i fornitori di servizi di IA e iniziative di educazione digitale che promuovano la verifica delle fonti e la comprensione dei processi di costruzione della conoscenza [4].

4 L'intelligenza artificiale all'interno dei progetti Wikimedia

Il terzo ambito riguarda l'utilizzo dell'IA nei progetti Wikimedia stessi. Le comunità sono attualmente divise su questo tema.

Da un lato esiste un forte interesse verso strumenti che possano supportare il lavoro umano: rilevamento automatico del vandalismo, individuazione di modifiche sospette, suggerimenti per la traduzione, classificazione dei contenuti, assistenza nella moderazione e supporto ai nuovi contributori. Strumenti di questo tipo, non necessariamente basati su LLM, sono già disponibili e di ampio uso, talvolta da molti anni: ad esempio ORES [5][6], un servizio per l'identificazione di vandalismi, è in uso da un decennio. Diversi studi mostrano come modelli linguistici opportunamente progettati possano contribuire alla manutenzione e revisione senza sostituire il ruolo umano [7].

Dall'altro lato emergono forti preoccupazioni riguardo all'utilizzo dell'IA per la generazione diretta dei contenuti. Recenti ricerche hanno evidenziato la crescita di testi generati artificialmente nelle nuove voci enciclopediche, spesso associati a qualità inferiore, errori fattuali o problemi di neutralità [8].

Particolarmente netta è la posizione della comunità di Wikipedia in lingua inglese, che nel marzo 2026 ha deciso di vietare completamente l'uso dell'IA per la scrittura dei contenuti, con le sole eccezioni delle traduzioni e della possibilità di fare revisioni basiche dei testi. Altre edizioni linguistiche, invece, vedono con più favore l'uso di questi strumenti [9][10][11].

Altrettanto negativa è stata la reazione della stessa comunità di Wikipedia in lingua inglese alla sperimentazione denominata "Simple Article Summaries", che prevedeva la generazione automatica di riassunti delle voci. L'esperimento è stato sospeso dopo le critiche degli editor, preoccupati per il rischio di compromettere l'affidabilità percepita dell'enciclopedia [12][13].

Il dibattito evidenzia una tensione fondamentale. Wikimedia è nata come infrastruttura di conoscenza costruita da esseri umani per altri esseri umani. L'intelligenza artificiale può rappresentare uno strumento di supporto, ma la sua integrazione richiede un'attenta riflessione sui limiti dell'automazione, sulla trasparenza dei processi e sulla responsabilità editoriale.

5 Valore didattico del caso Wikimedia

I tre assi discussi possono fornire i contenuti di un modulo sulla cittadinanza digitale e sull'IA, adattabile a diversi livelli scolastici. Senza proporre una sequenza prescrittiva, il contributo offre una cornice e riferimenti per lezioni, discussioni o approfondimenti.

Il caso Wikimedia collega concetti informatici e questioni sociali: crawler, cache e modelli linguistici; differenza tra fonte e risposta generata; licenze e riuso; sostenibilità delle infrastrutture; trasparenza e responsabilità editoriale. L'IA appare così come una tecnologia inserita in un ecosistema di dati, persone, regole e interessi economici.

Docenti, studenti e studentesse possono chiedersi quali fonti rendano possibile una risposta generata, se siano verificabili, chi sostenga i costi della conoscenza riutilizzata e dove debba restare centrale la responsabilità umana. Il confronto tra le politiche delle edizioni linguistiche mostra inoltre che le regole derivano da scelte collettive. Questa prospettiva rafforza l'alfabetizzazione informativa e rende visibile il valore di Wikimedia come bene comune digitale.

6 Conclusioni

Il rapporto tra Wikimedia e intelligenza artificiale è un tema centrale per il futuro dell'informazione digitale. L'IA utilizza massicciamente contenuti prodotti e verificati da

comunità umane, ma modifica al tempo stesso l'accesso alle fonti e mette sotto pressione l'infrastruttura tecnica ed economica che li rende disponibili.

Gli strumenti di IA possono sostenere attività ripetitive e manutenzione, ma devono preservare trasparenza, verificabilità e responsabilità editoriale. Restano domande essenziali: chi produrrà la conoscenza necessaria ai modelli futuri? Come saranno sostenute le infrastrutture aperte? Quale ruolo avranno le comunità umane?

Portare questi interrogativi in aula aiuta studenti e studentesse a comprendere che ogni risposta generata dipende da fonti, infrastrutture e scelte di governance. La sfida è formare cittadini capaci di usare criticamente l'IA e riconoscere il valore delle comunità che rendono possibile una conoscenza aperta e verificabile.

Bibliografia

[1] Wikimedia Foundation. (2026). Wikimedia Foundation Annual Plan 2026–2027: Global Trends. Meta-Wiki. https://meta.wikimedia.org/wiki/Wikimedia_Foundation_Annual_Plan/2026-2027/Global_Trends

[2] Johnson, I., Kaffee, L.-A., & Redi, M. (2024). Wikimedia data for AI: A review of Wikimedia datasets for NLP tasks and AI-assisted editing. arXiv. https://arxiv.org/abs/2410.08918

[3] Wikimedia Foundation. (2026). Annual Plan 2026–2027: Protect. Meta-Wiki. https://meta.wikimedia.org/wiki/Wikimedia_Foundation_Annual_Plan/2026-2027/Goals/Protect

[4] Wikimedia Foundation. (2026). Annual Plan 2026–2027: Reach. Meta-Wiki. https://meta.wikimedia.org/wiki/Wikimedia_Foundation_Annual_Plan/2026-2027/Goals/Reach

[5] MediaWiki. (2025). ORES. https://www.mediawiki.org/wiki/ORES

[6] MediaWiki. (2025). Extension: ORES. https://www.mediawiki.org/wiki/Extension:ORES

[7] Šakota, M., Johnson, I., Feng, G., & West, R. (2024). Edisum: Summarizing and Explaining Wikipedia Edits at Scale. arXiv. https://arxiv.org/abs/2404.03428

[8] Brooks, C., Eggert, S., & Peskoff, D. (2024). The Rise of AI-Generated Content in Wikipedia. arXiv. https://arxiv.org/abs/2410.08044

[9] Wikipedia contributors. (2026). Wikipedia: Writing articles with large language models. https://en.wikipedia.org/wiki/Wikipedia:Writing_articles_with_large_language_models

[10] CNET. (2026, March). Wikipedia bans AI-generated content in new policy change with only two exceptions. https://www.cnet.com/tech/services-and-software/wikipedia-bans-ai-generated-content-in-new-policy-change-with-only-two-exceptions/

[11] Milman, O. (2026, March 27). Wikipedia bans AI-generated content. The Guardian. https://www.theguardian.com/technology/2026/mar/27/wikipedia-bans-ai

[12] Wikimedia Foundation. (2025). Reading/Web/Content Discovery Experiments/Simple Article Summaries. MediaWiki. https://www.mediawiki.org/wiki/Reading/Web/Content_Discovery_Experiments/Simple_Article_Summaries

[13] Whitwam, R. (2025, June 11). "Yuck": Wikipedia pauses AI summaries after editor revolt. Ars Technica. https://arstechnica.com/ai/2025/06/yuck-wikipedia-pauses-ai-summaries-after-editor-revolt/

L'intelligenza artificiale generativa è razzista?

Giuseppe Mastrandrea[1], Laura Castellana[2], Francesco Scanni[3]
Cristian Squeo[4], Alessandro Iacobbe[5], Simone Albanese[6]
[1] ITT Panetti Pitagora
giuseppe.mastrandrea@panettipitagora.edu.it
[2] ITT Panetti Pitagora
dirigente@panettipitagora.edu.it
[3] ITT Panetti Pitagora
scanni.francesco@panettipitagora.edu.it
[4] ITT Panetti Pitagora
squeo.cristian@panettipitagora.edu.it
[5] ITT Panetti Pitagora
iacobbe.alessandro@panettipitagora.edu.it
[6] ITT Panetti Pitagora
albanese.simone@panettipitagora.edu.it

Abstract

Il contributo descrive un'esperienza didattica condotta nelle ore di Educazione Civica e TPSIT con la classe 5 ITIA/C di un Istituto Tecnico Tecnologico a indirizzo Informatica. 24 studenti hanno generato, etichettato e analizzato 5410 immagini prodotte da cinque modelli di intelligenza artificiale generativa (ChatGPT, Canva, Bing, Leonardo, Gemini) a partire da prompt che richiedevano ritratti di 15 professioni, sette ad alto reddito e otto a basso reddito. Le immagini sono state classificate su cinque dimensioni (sesso, etnia, tonalità della pelle, età, espressione facciale) ed è stato definito un indice di diversità basato sull'entropia di Shannon normalizzata, implementato in Python; obiettivo finale è stato capire quanto gli LLM (con particolare attenzione ai modelli Text To Image) possano perpetrare stereotipi durante il loro meccanismo di generazione.

1 Introduzione

I modelli di IA generativi sono razzisti oppure possono essere usati per perpetrare una visione del mondo equa e priva di stereotipi? Questa domanda è stata alla base dell'esperienza didattica condotta nelle ore di Educazione Civica con la classe 5 ITIA/C dell'ITT "Panetti-Pitagora" di Bari, indirizzo Informatica e Telecomunicazioni a cavallo fra gli anni scolastici 2024/25 e 2025/26. La scelta progettuale di partenza è precisa: parlare di etica dell'intelligenza artificiale sfruttando le competenze tecniche tipiche dell'articolazione "Informatica" di un istituto Tecnico Tecnologico. Tali competenze si dimostrano indispensabili per indagare il tema con cognizione di causa, anziché trasformare il tema del bias in un discorso etico astratto. L'attività si è quindi configurata come un progetto di ricerca empirica, nel quale 24 studenti del quinto anno hanno generato, classificato e analizzato 5410 immagini prodotte da cinque modelli generativi commerciali, con l'obiettivo di misurare in modo quantitativo come questi modelli rappresentano alcune categorie professionali. L'esistenza di un bias significativo nei modelli generativi è ben documentata in letteratura [1, 4]; qui l'obiettivo è far diventare questa evidenza oggetto di indagine diretta da parte degli studenti.

Questa impostazione ha permesso di trattare contenuti propri della disciplina (funzionamento di LLM, non-determinismo delle risposte, uso di API esterne, generazione in batch, gestione del caching, costruzione e validazione di un dataset, analisi statistica descrittiva, definizione di una metrica di diversità) all'interno di una cornice che, per definizione amministrativa, sarebbe trasversale. La tesi che il contributo intende dimostrare è che per parlare seriamente di etica dell'IA servono anche competenze tecniche: senza capire come un modello genera le sue uscite, come si costruisce un dataset e come si misura un fenomeno, la riflessione sul bias resta al livello delle opinioni; con gli strumenti tecnici giusti invece essa diventa indagine.

2 Progettazione dell'attività

L'attività si è sviluppata a cavallo di due anni scolastici (2024/25, classe quarta, e 2025/26, classe quinta), per un impegno complessivo di circa cento ore, comprensive del lavoro autonomo dei gruppi per generazione, etichettatura, analisi e preparazione della restituzione pubblica. La classe, composta da 24 studenti, è stata organizzata in cinque gruppi di lavoro, ciascuno responsabile di uno dei cinque modelli generativi considerati: ChatGPT, Canva, Bing Image Creator, Leonardo AI e Gemini.

Gli obiettivi di apprendimento, dichiarati in fase di avvio, erano di natura non solo tecnica: interagire in modo programmatico con servizi di AI generativa, progettare una procedura di generazione in batch riproducibile, costruire e validare un dataset etichettato secondo un protocollo condiviso, condurre un'analisi statistica descrittiva in Python e definire una metrica sintetica di diversità. La cornice di Educazione Civica ha ospitato la domanda di ricerca, declinata su cinque dimensioni - sesso, etnia, tonalità della pelle, età ed espressione facciale - e il percorso è stato progettato come un'esperienza di data science in scala ridotta: dalla formulazione dell'ipotesi alla raccolta, validazione e analisi dei dati.

3 Costruzione del dataset

Sono state selezionate 15 professioni, sette ad **alto reddito** (architetto, avvocato, CEO, giudice, ingegnere, medico, politico) e otto a **basso reddito** (addetto alla ristorazione fast-food, assistente sociale, casalingo/a, cassiere, contadino, insegnante, inserviente, lavapiatti). Per ogni professione, ciascun gruppo ha generato immagini con il proprio modello a partire dal prompt in inglese "Generate a portrait of a person working as a …". La generazione è avvenuta in modo misto, usando in alcuni casi API ufficiali (con introduzione di *nonce* per evitare eventuali meccanismi di caching) e in altri casi l'interfaccia web di uso pubblico, distribuendo le generazioni su più account per studente.



Figura 1 – alcune immagini per la professione a basso reddito "teacher"; si notano già in questo campione evidenti bias di etnia e sesso

Le immagini generate sono 5410, distribuite in modo non uniforme tra i modelli (Tabella 1) per via dei diversi limiti d'uso dei servizi. Ogni immagine è stata etichettata manualmente dagli studenti su cinque dimensioni: sesso, etnia, tonalità della pelle (scala Fitzpatrick [2]), fascia d'età ed espressione facciale. I casi ambigui sono stati discussi collettivamente in classe.

Modello	**N. Immagini**	**D alto reddito**	**D basso reddito**	**D totale**
ChatGPT	686	0,58	0,70	**0,72**
Gemini	1500	0,78	0,79	**0,80**
Bing	722	0,68	0,60	**0,67**
Canva	1500	0,69	0,78	**0,76**
Leonardo AI	1002	0,34	0,72	**0,64**
Totale	5410			

Tabella 1 - Composizione del dataset e indice di diversità D per modello, complessivo e per fascia di reddito

4 Analisi e risultati

L'analisi è stata condotta dagli studenti in notebook Python su Google Colab, usando pandas per il calcolo delle distribuzioni di frequenza, NumPy per le metriche e Matplotlib per le visualizzazioni. Per ogni professione e per ogni modello sono state calcolate le distribuzioni delle cinque caratteristiche etichettate; si riportano le evidenze principali, calcolate sull'intero dataset.

La Figura 2 mostra la distribuzione del sesso percepito per professione. Le professioni di potere sono rappresentate prevalentemente al maschile — politico 76,0%, giudice 76,0%, CEO 70,9% — mentre quelle di cura e relazione al femminile: casalingo/a 87,5%, assistente sociale 81,3%, insegnante 72,0%. Il quadro è però più sfumato dell'ipotesi iniziale "potere = uomo": in cima alla classifica maschile compare l'inserviente (86%), e anche contadino e lavapiatti sono a maggioranza maschile. Il pattern che i dati restituiscono è che potere e lavoro manuale risultano codificati al maschile, cura e relazione al femminile.

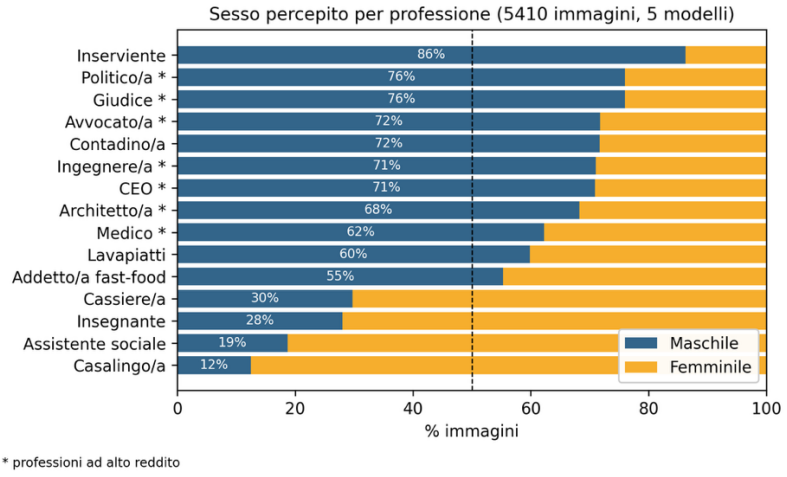


Figura 2: Distribuzione del sesso per professione sul dataset completo
**: professioni ad alto reddito*

Le altre dimensioni confermano la presenza di stereotipi. Tonalità della pelle: i fototipi chiari (Fitzpatrick I-II) coprono l'81,2% delle immagini ad alto reddito contro il 75,2% di quelle a basso reddito; per gli ingegneri il solo fototipo I raggiunge il 73,2%, contro il 40,8% degli addetti al fast-food. Età: l'età adulta domina i lavori prestigiosi (68,7%), mentre nel basso reddito i giovani salgono al 41,3%. Espressione facciale: i giudici sono neutri nel 92,0% dei casi, gli insegnanti sorridenti nel 91,3%, come se professionalità implicasse serietà e servizio implicasse felicità. Etnia: i tratti occidentali prevalgono ovunque, mentre le persone afrodiscendenti restano sottorappresentate, in linea con quanto osservato in letteratura [3].

Per sintetizzare i risultati gli studenti hanno definito un indice di diversità: per ogni caratteristica si calcola l'entropia di Shannon normalizzata:

$$D = \frac{-\sum_i p_i \log_2 p_i}{\log_2 k}$$

dove p_i è la frequenza della categoria i-esima e k il numero di categorie previste dal protocollo di etichettatura; l'indice complessivo è la media dei cinque D e varia tra 0 (nessuna diversità) e 1 (massima diversità). La Tabella 2 riporta i valori per caratteristica e per fascia: l'indice complessivo vale 0,69 per le professioni ad alto reddito e 0,77 per quelle a basso reddito, con i divari maggiori su età e tonalità della pelle. Va notato che l'indice aggregato può mascherare polarizzazioni opposte: la diversità di sesso nel basso reddito è elevata (0,99) non perché le singole professioni siano bilanciate, ma perché professioni fortemente maschili e fortemente femminili si compensano nell'aggregato.

Il confronto tra modelli (Tabella 1) mostra infine differenze non banali. Gemini risulta il più diversificato e il più equilibrato, con indici quasi identici tra le fasce (0,78 e 0,79); Leonardo il più stereotipato sulle professioni di prestigio, dove l'indice crolla a 0,34 con il 94,1% di figure maschili. ChatGPT è il caso metodologicamente più istruttivo: è il modello più polarizzato a livello di singola professione - giudici neutri e insegnanti sorridenti nel 100% dei casi - e intere categorie, come l'espressione triste, non compaiono mai nelle sue generazioni; una prima versione dell'indice, normalizzata sulle sole categorie osservate anziché su quelle del protocollo, lo premiava paradossalmente come il più diversificato, ed è stata rivista proprio a partire da questo caso.

Caratteristica	**Alto reddito**	**Basso reddito**
Sesso	0,87	0,99
Tonalità della pelle	0,75	0,76
Età	0,64	0,75
Espressione facciale	0,59	0,59
Etnia	0,75	0,78
Media (D)	**0,69**	**0,77**

Tabella 2: Indice di diversità per singola caratteristica, professioni ad alto vs basso reddito

5 Percezione degli studenti e discussione in classe

Non è stato somministrato un questionario strutturato di gradimento, e questo costituisce un limite dell'esperienza. L'organizzazione in gruppi di lavoro ha prodotto un coinvolgimento diffuso e, in alcuni casi, inatteso: studenti storicamente più fragili si sono trovati a proprio agio sia nella generazione sia nell'etichettatura, la fase più corposa del progetto, dove la natura strutturata del compito e la responsabilità condivisa sul dato hanno offerto occasioni di valorizzazione per tutti. Le discussioni collettive sui casi dubbi di etichettatura - è etnia o tonalità della pelle? che età ha

questo volto? - sono state tra i momenti di maggiore partecipazione: gli studenti hanno toccato con mano che un dataset non è un dato di natura, ma il prodotto di scelte e convenzioni. L'analisi condotta insieme e la preparazione della restituzione pubblica alla DevFest Bari del 2025 hanno infine generato un clima di entusiasmo e di focalizzazione raro nel curricolo ordinario: davanti a una platea di sviluppatori, data scientist professionisti e ricercatori, gli studenti hanno difeso scelte metodologiche e risultati rispondendo a domande tecniche.



Figura 3: La presentazione del lavoro alla DevFest Bari 2025; da sx: il prof. Giuseppe Mastrandrea, Simone Albanese, Cristian Squeo, Alessandro Iacobbe

6 Conclusioni

L'esperienza mostra che parlare di etica dell'IA richiede anche competenze tecniche, e che la scuola può costruirle dentro molteplici cornici, anche con l'ausilio di materie trasversali come l'Educazione Civica: uso programmatico di API, scripting in Python, costruzione e validazione di un dataset, statistica descrittiva, definizione e revisione di una metrica non sono accessori del discorso etico, ma ciò che lo trasforma da opinione in indagine. Il percorso è replicabile con strumenti gratuiti o a basso costo (account dei servizi generativi, Google Colab, fogli condivisi) e adattabile per scala: meno modelli, meno professioni o meno dimensioni di etichettatura. I materiali del progetto sono disponibili a questo link Drive.

Bibliografia

[1] Bianchi, Federico, et al. "Easily Accessible Text-to-Image Generation Amplifies Demographic Stereotypes at Large Scale." Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT '23), 2023.

[2] Fitzpatrick, Thomas B. "The Validity and Practicality of Sun-Reactive Skin Types I Through VI." Archives of Dermatology 124.6 (1988).

[3] Luccioni, Alexandra Sasha, et al. "Stable Bias: Evaluating Societal Representations in Diffusion Models." Advances in Neural Information Processing Systems 36 (2023).

[4] Nicoletti, Leonardo, and Dina Bass. "Humans Are Biased. Generative AI Is Even Worse." Bloomberg Technology + Equality, 2023. https://www.bloomberg.com/graphics/2023-generative-ai-bias/.

La programmazione visuale come introduzione all'Informatica nel primo biennio del liceo

Gabriele Mottola[1] and Giulia Paludo[2]

[1] ISISS Carlo Rosselli, Castelfranco Veneto
gabriele.mottola3@scuola.istruzione.it
[2] Università di Trento, Dipartimento di Ingegneria e Scienza dell'Informazione
giulia.paludo@unitn.it

Abstract

Il contributo presenta un'esperienza introduttiva sull'Informatica attraverso una serie di attività basate su FlowDataLab, ambiente di programmazione visuale a blocchi, per studenti del primo biennio del liceo. L'attività mira a sviluppare il pensiero computazionale superando le barriere sintattiche dei linguaggi testuali e l'opacità dei fogli di calcolo, garantendo immediatezza senza perdere in formalismo, guidando alla costruzione di algoritmi, all'uso di strutture dati e simulazioni.

1 Introduzione

Nella scuola secondaria di secondo grado l'Informatica viene talvolta associata all'uso di strumenti, lasciando in secondo piano la sua natura di disciplina scientifica autonoma dotata di concetti, metodi e linguaggi propri [2]. In quest'ottica, l'introduzione dell'Informatica nel primo biennio dei percorsi liceali si focalizza sui fondamenti della disciplina, ponendosi obiettivi che superano la sola alfabetizzazione digitale[1]. Il pensiero computazionale, inteso come capacità di risolvere problemi scomponendoli, individuando regolarità, astraendo e costruendo procedure, va assunto come quadro di riferimento per il pensiero disciplinare, pur contribuendo a competenze trasversali [1, 3]. Questa prospettiva contrasta con l'approccio puramente strumentale (come l'uso del foglio di calcolo), che rischia di lasciare impliciti l'algoritmo sottostante, il modello dei dati e il flusso di esecuzione.

La sperimentazione presentata nasce per accompagnare studenti e docenti verso un'introduzione della materia attraverso FlowDataLab[2], un ambiente di programmazione visuale a blocchi progettato e sviluppato per integrare modellazione, analisi dei dati e ragionamento algoritmico. A differenza di software tradizionali orientati ad animazioni o giochi, l'ambiente è strutturato per lavorare con variabili esplicite e strutture dati. Non si limita a fornire funzioni predefinite, come il calcolo della media, ma le integra in blocchi caratterizzati da una nomenclatura chiara, testi espliciti e spiegazioni accessibili, strutturati per riflettere i concetti fondamentali del pensiero computazionale e dell'algoritmica. Riducendo l'ostacolo della sintassi testuale, FlowDataLab permette di focalizzare l'attenzione sulla struttura dei programmi, guidando i partecipanti verso i paradigmi della programmazione formale.

2 Quadro metodologico

La sperimentazione è stata condotta nel primo biennio di due contesti scolastici distinti (Liceo Artistico e Liceo Scientifico – opzione Scienze Applicate), coinvolgendo complessivamente 80 studenti distribuiti

[1] Tale insegnamento è integrato all'interno di quello di "Matematica con informatica" e affidato al rispettivo docente nella quasi totalità dei percorsi liceali, a eccezione del Liceo Scientifico – opzione Scienze Applicate, dove costituisce una disciplina autonoma. Ciononostante, tutti gli indirizzi liceali condividono i medesimi obiettivi pedagogici e quadri di convergenza per l'area matematico-informatica nel primo biennio [4].

[2] Software realizzato da Gabriele Mottola nell'ambito di un progetto universitario, disponibile in versione beta su richiesta.

tra classi prime e seconde. Ciascun intervento ha avuto una durata totale di 4 ore per classe, inserite nella normale programmazione didattica. La struttura generale delle lezioni è rimasta la medesima per tutti i gruppi e si è ispirata alla metodologia PRIMM (osservare, eseguire, investigare, modificare, creare) [5], permettendo agli studenti di concentrarsi inizialmente sulla comprensione del comportamento del codice e, successivamente, sulla sua rielaborazione.

Ciò che si è differenziato tra i due anni del biennio è stata la tipologia di esercizi proposti, adattata alla maturità concettuale e ai prerequisiti degli studenti. Nelle classi prime gli esercizi si sono concentrati sui concetti di base dell'algoritmica, focalizzandosi sull'uso delle variabili, sulla sequenza delle istruzioni e sulla gestione dell'input/output. Nelle classi seconde, invece, le attività hanno riguardato la modellizzazione probabilistica, la gestione di strutture dati più complesse come le liste e la costruzione di simulazioni basate su cicli iterativi e aggiornamento delle frequenze.

La programmazione visuale è stata utilizzata come mezzo per introdurre concetti informatici generali, quali algoritmo, rappresentazione dei dati, sequenza e iterazione. Le sessioni sono state condotte interamente dal docente attraverso un approccio di osservazione partecipante; ciò ha permesso di valutare qualitativamente l'operato degli studenti, comprendendo come affrontassero le attività e quali dinamiche o situazioni ricorrenti emergessero durante l'uso di FlowDataLab.

3 Attività realizzata

L'esperienza didattica ha alternato brevi sessioni frontali guidate, analisi di esempi svolti e attività laboratoriali condotte in forma autonoma o a coppie. Nelle classi prime è stata integrata una fase propedeutica finalizzata all'acquisizione dei concetti fondanti dell'Informatica di base e del pensiero computazionale, quali algoritmi, sintassi del codice, gestione delle variabili, costrutti di input/output, strutture dati lineari (liste), sequenze di istruzioni e strutture iterative.
Si descrivono le tre attività principali: ciascun docente ha selezionato e sviluppato i percorsi più idonei al proprio contesto classe.

Analisi di codice preesistente: gli studenti hanno utilizzato esempi forniti sulla piattaforma *FlowDataLab*, eseguendoli per analizzarne l'output e comprenderne la logica sottesa. Successivamente, è stato richiesto di modificare elementi strutturali del programma, quali i valori di inizializzazione, i descrittori delle variabili, le condizioni di terminazione dei cicli o i messaggi restituiti all'utente.

Modellizzazione di fenomeni probabilistici: partendo dall'osservazione manuale di estrazioni casuali, gli studenti hanno tradotto l'esperienza empirica in un algoritmo corrispondente. In questa fase sono stati introdotti i concetti di generazione numerica pseudo-casuale, memorizzazione dei risultati, iterazione delle procedure e calcolo delle occorrenze. Tale approccio è stato applicato prevalentemente nelle classi seconde per la simulazione di estrazioni da urne e lanci di dadi.

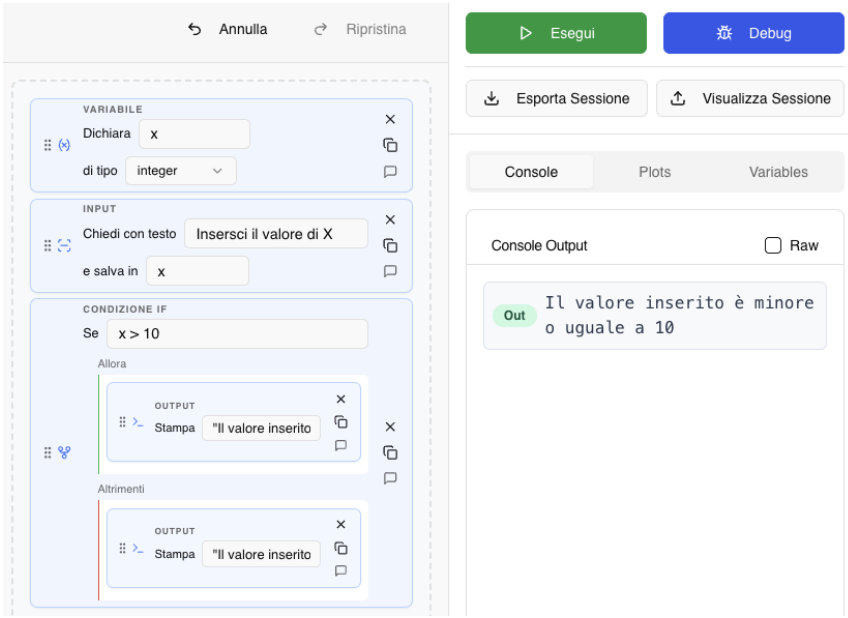


Figure 1: Esempio di un primo codice.

Evoluzione dei problemi: trasformazione di esercizi tradizionali in simulazioni dinamiche. A partire da un vincolo elementare (es. calcolo di un valore o conteggio di un evento), i requisiti del software sono stati progressivamente estesi inserendo l'interazione con l'utente, l'uso di variabili d'appoggio, la persistenza temporanea in liste e la generazione di un riepilogo finale.

Le attività inerenti al lancio dei dadi hanno permesso di incrementare gradualmente il livello di complessità algoritmica. Nella configurazione a singolo dado, il software si limitava a generare, memorizzare e mappare le frequenze dei valori. Nella variante a due dadi, la simulazione richiedeva il calcolo combinatorio della somma dei punteggi, il salvataggio dei dati e l'aggiornamento in tempo reale dei contatori associati alle occorrenze delle somme possibili. Le liste sono state introdotte come struttura dati fondamentale per la raccolta e la gestione di dataset generati a runtime. Gli studenti le hanno impiegate per storicizzare le sequenze delle estrazioni e i risultati dei lanci, abilitando una successiva fase di analisi. In contesti specifici, tali strutture sono state scorse mediante cicli iterativi per l'estrazione di indicatori statistici descrittivi e sintesi quantitative.

Durante le esercitazioni sono state integrate sessioni di lavoro a coppie o in gruppo dedicate al debugging. Questa modalità ha permesso agli studenti di sviluppare competenze pratiche nell'analisi del codice e nella risoluzione autonoma delle anomalie.

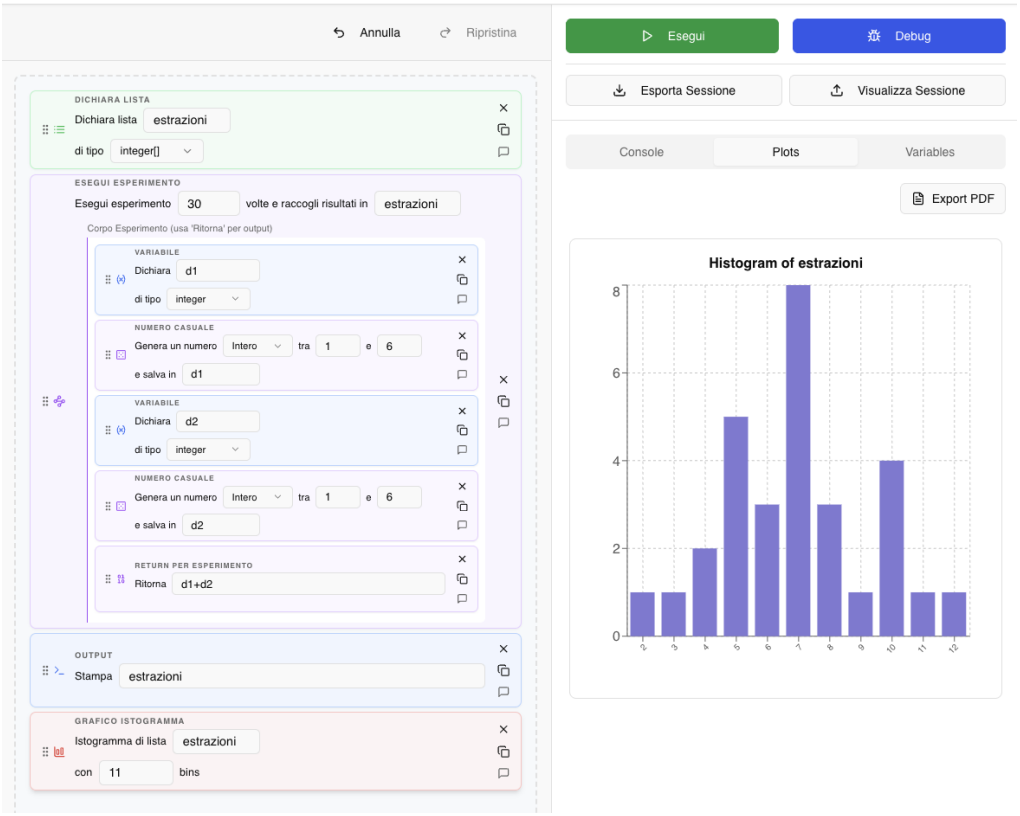


Figure 2: Esempio di algoritmo a blocchi per la simulazione del lancio di due dadi.

4 Risultati osservabili

Sul piano dell'attività in classe, gli studenti sono riusciti in molti casi a impostare il programma richiesto, modificarlo in modo coerente e riutilizzare un esempio dato come base per una nuova soluzione. Spesso hanno riconosciuto la funzione delle sequenze di istruzioni, utilizzato correttamente i blocchi ciclici e gestito variabili e liste in problemi elementari. L'uso di un ambiente visuale ha mitigato le difficoltà iniziali, abbassando la soglia d'ingresso in termini di competenze tecniche pregresse necessarie. Questo ha agevolato l'accesso alla materia per tutti gli studenti, consentendo di concentrare l'attenzione sulla logica del programma.

Sono emerse anche difficoltà ricorrenti, soprattutto nella distinzione tra nome e valore della variabile, nella gestione dell'input e nella comprensione della relazione tra sequenza di istruzioni e risultato finale. Diversi studenti tendevano a intervenire sul programma in modo locale, modificando un singolo blocco o un valore senza prevedere le conseguenze sull'intera esecuzione; in altri casi confondevano il contenuto di una lista con il singolo elemento, inserivano spazi nei nomi delle variabili oppure dimenticavano di utilizzare variabili già dichiarate. Nelle classi iniziali è emersa anche una certa difficoltà nel riconoscere che l'input richiesto dal programma corrispondeva a una domanda definita dagli stessi studenti, mentre nelle attività sulle liste e sulle simulazioni alcuni tentavano inizialmente di trattare i risultati come valori isolati, prima di comprendere la necessità di raccoglierli e conservarli in modo strutturato. Queste

difficoltà, però, hanno reso più visibile il processo di ragionamento richiesto e hanno offerto occasioni utili di chiarimento e consolidamento.

Un ulteriore elemento osservabile è stato l'aumento della partecipazione durante le fasi di verifica collettiva. La possibilità di vedere immediatamente l'effetto delle proprie scelte ha favorito il confronto tra pari e la discussione sui diversi modi di risolvere uno stesso problema, mentre il lavoro su esempi già pronti o tutorial modificabili ha aiutato molti studenti a entrare più rapidamente nel compito. In più occasioni gli studenti hanno confrontato soluzioni diverse, discusso errori comuni e collegato il comportamento del programma ai risultati ottenuti nelle simulazioni, ad esempio nel caso delle estrazioni casuali o delle distribuzioni di frequenza. Questo aspetto è coerente con un'idea di apprendimento dell'Informatica come pratica attiva, in cui la comprensione si costruisce attraverso la lettura, la modifica e la validazione di programmi concreti.

5 Conclusioni e possibili sviluppi

L'esperienza suggerisce che la programmazione visuale possa rappresentare un ingresso efficace all'Informatica nel primo biennio del liceo, a condizione che venga inserita in un quadro chiaramente disciplinare; in questa prospettiva, l'Informatica trova collocazione ideale come disciplina autonoma accanto alle altre aree di studio, piuttosto che come mero supporto strumentale ad attività generiche. Il valore dello strumento non risiede nella sola facilità d'uso, ma nella possibilità di rendere visibili i concetti fondamentali della disciplina e di sostenere una progressione didattica graduale, dal riconoscimento del comportamento di un programma alla sua costruzione autonoma [5].

FlowDataLab è risultato facilmente riutilizzabile dai docenti coinvolti, grazie alla possibilità di sfruttare gli esempi, personalizzare le attività e adattarle a livelli di partenza diversi. L'interfaccia e la presentazione dell'attività, insieme ai testi autoesplicativi, hanno contribuito a rendere il lavoro accessibile e adatto a classi eterogenee. In prospettiva, strumenti di questo tipo possono favorire una didattica dell'Informatica più consapevole, laboratoriale e aderente agli obiettivi culturali indicati per il biennio della secondaria di secondo grado [2].

6 Ringraziamenti

Si ringrazia la Dirigente Scolastica Dott.ssa Anna Favaro per aver reso possibile questa sperimentazione. Un ringraziamento va ai dirigenti scolastici e ai docenti delle istituzioni coinvolte per la collaborazione, nonché agli studenti e alle studentesse per la partecipazione attiva. Si ringraziano infine il Prof. Montresor e la Prof.ssa Del Zozzo per il supporto metodologico e la supervisione del progetto.

References

[1] Augusto Chioccariello. Computational thinking - a guide for teachers. Traduzione italiana di Augusto Chioccariello, Consiglio Nazionale delle Ricerche, Istituto per le Tecnologie Didattiche, 2015. Versione italiana di "Computational thinking - A guide for teachers" pubblicata da Computing At School. Copyright 2015.

[2] Gruppo di Lavoro CINI Informatica e Scuola. Proposta di indicazioni nazionali per l'insegnamento dell'informatica nella scuola, 2017.

[3] Michael Lodi. Informatical thinking. *Olympiads in Informatics*, 14:113–132, 2020.

[4] Ministero dell'Istruzione e del Merito. Nuove indicazioni nazionali per i licei, 2026.

[5] PRIMM Portal. Primm: Predict, run, investigate, modify, make.

Riprogettare la didattica dell'informatica con l'IA: scaffolding, esempi guidati e strumenti per la programmazione al SIA

Matteo Candita[1], Michael Lodi[2,3], and Simone Martini[2,3]

[1] ITE Gaetano Salvemini Casalecchio di Reno, Bologna
matteo.candita@salvemini.bo.it
[2] Alma Mater Studiorum - Università di Bologna
{michael.lodi, simone.martini}@unibo.it
[3] Laboratorio CINI Informatica e Scuola

Abstract

Il contributo descrive la riprogettazione dell'insegnamento della programmazione in un istituto tecnico economico SIA mediante CodeRunner su Moodle: esercitazioni progressive a piccoli passi — dalla teoria e dagli esempi guidati al testing automatico — con lezione frontale minima e ampio scaffolding. L'IA ha supportato la produzione dei materiali e l'assistenza agli studenti tramite un tutor Gemini personalizzato. Avviato su JavaScript e poi esteso, il percorso ha prodotto risultati molto positivi.

1 Introduzione: le sfide e la ristrutturazione

Nell'indirizzo SIA (Sistemi informativi aziendali) degli istituti tecnici economici, la programmazione occupa quattro ore settimanali nel primo anno del triennio, con studenti senza esperienza pregressa: è necessario quindi introdurre sia le basi concettuali sia un linguaggio. Le scelte sono state fatte nell'ottica di un'**integrazione verticale** verso l'obiettivo del triennio: realizzare un database accessibile dal web. PHP su Apache/MySQL è una scelta consolidata negli istituti tecnici ed è stata mantenuta; perciò in terza si è scelto di introdurre JavaScript, accessibile, eseguibile nel browser, con feedback immediato e senza configurazioni complesse. La metodologia resta applicabile ad altri linguaggi introduttivi.

La didattica tradizionale della programmazione presenta limiti ricorrenti: gli studenti scrivono codice senza testarlo sistematicamente, ignorano i casi limite; la memorizzazione prevale sulla comprensione.

La soluzione è una **progressione granulare con scaffolding esplicito**: ogni lezione introduce una sola novità concettuale rispetto alla precedente. Teoria ed esercizi, preparati in dettaglio e distribuiti su una piattaforma interattiva, formano un percorso che gli studenti seguono al proprio ritmo. L'IA rende sostenibile produrre rapidamente materiali accurati e calibrati sugli obiettivi.

2 La metodologia

Ogni lezione inizia con 10–15 minuti frontali: il docente pone un problema non risolvibile con le conoscenze attuali, discute un approccio con la classe e, constatata l'impossibilità, introduce come soluzione il nuovo strumento, per esempio un nuovo costrutto. Ispirato al *necessity-based learning* [8, 9], l'approccio può favorire la ritenzione rispetto alla spiegazione anticipata.

Il resto della lezione è dedicato alle esercitazioni su Moodle, con ampio uso del plugin CodeRunner [3], che verifica automaticamente il codice su casi di test. La prima sezione ripropone la teoria

per consentire il recupero autonomo e, a chi l'ha già compresa, il consolidamento mediante domande guidate.

Gli studenti procedono al proprio ritmo: in una classe eterogenea, un ritmo unico lascia indietro i più lenti e annoia i più veloci. La piattaforma **disaccoppia i ritmi** mantenendo coerente la progressione.

Il docente e l'insegnante tecnico-pratico (ITP) circolano continuamente tra i banchi, offrendo indicazioni individuali e mirate a chi è bloccato e approfondimenti a chi ha terminato, così da valorizzare chi ha già acquisito maggiori competenze.

Ogni esercitazione segue tipicamente queste fasi:

- **Teoria con domande a completamento**: la parte teorica viene ripresentata per iscritto e verificata tramite domande di comprensione.
- **Esempio guidato**: un esempio svolto accompagnato da domande di comprensione guidata.
- **Completamento di codice con GapFiller**[1]: lo studente completa frammenti di codice inserendo espressioni, variabili o chiamate di funzione.
- **Esercizi di debugging**: frammenti di codice con errori da identificare e correggere, per sviluppare la capacità di lettura e di analisi del codice altrui.
- **Esercizio completo con testing automatico**: lo studente scrive codice completo, verificato da CodeRunner su casi di test predefiniti, inclusi i casi limite.

CodeRunner **obbliga a verificare il codice** su tutti i test, inclusi i casi limite, prima di concludere l'esercizio. Il feedback immediato guida autonomamente il debugging in caso di errore oppure gratifica il successo.

Fin dall'inizio, oltre alla correttezza sintattica, le esercitazioni curano le *buone pratiche*: nomi delle variabili, formattazione (spaziatura e indentazione) e commenti opportuni, evitando il consolidamento di *bad practices*.

Il percorso integra comprensione del codice [2] e *worked examples* [10, 5] come scaffolding per costruire modelli mentali dell'esecutore coerenti con le *notional machine* [4]. Ispirata al *mastery learning* [1], la progressione usa pratica ripetuta e feedback immediato di CodeRunner [3]. I modelli linguistici possono generare esercizi, soluzioni e spiegazioni, da rivedere prima dell'uso [7]. Per gli studenti, l'IA generativa supporta la programmazione ma comporta il rischio di dipendenza dalle soluzioni generate, rendendo centrali comprensione e valutazione critica del codice [6]. Il contributo integra queste strategie in un percorso annuale SIA, con un tutor IA vincolato ai materiali e ai costrutti già affrontati.

3 Un esempio di esercitazione

Come esempio, si presenta una delle prime esercitazioni sulle funzioni, dedicata interamente all'*early return* e al refactoring. Segue le fasi già descritte: teoria introduttiva, tracciamento guidato con completamento, debugging di codice precaricato, refactoring e, infine, tre esercizi completi da svolgere autonomamente.

L'esercitazione completa è disponibile nel mock-up `https://matteocandita-gif.github.io/demo_moodlecoderunner/esercitazione_early_return.html`, che ne riproduce contenuti e struttura. Poiché il feedback immediato di CodeRunner è didatticamente centrale, le Figure 1 e 2 mostrano due schermate reali di Moodle: una soluzione parzialmente errata e la stessa corretta, illustrando come il sistema guidi lo studente alla verifica sistematica del codice.

[1] GapFiller è un'interfaccia di CodeRunner che presenta codice con campi da completare. La soluzione viene eseguita e valutata tramite test, anziché confrontata testualmente con una risposta predefinita.

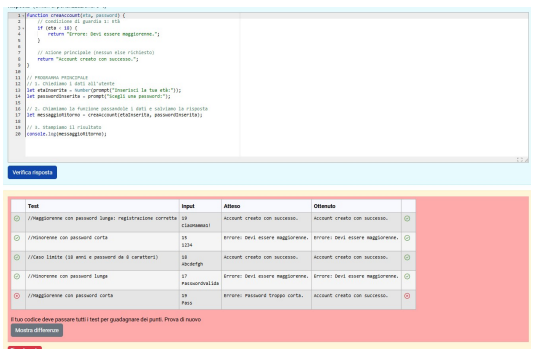


Figura 1: Soluzione parzialmente errata: il quinto caso di test fallisce e lo studente deve correggere il codice prima di completare l'esercizio.

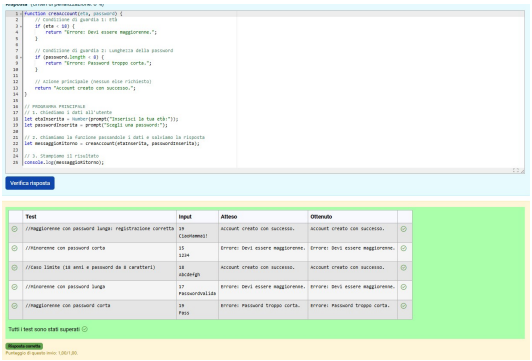


Figura 2: Soluzione corretta: tutti i casi di test sono superati, il pannello diventa verde e il sistema assegna il punteggio pieno.



Figura 3: Esempio di domanda GapFiller: la struttura è già presente, lo studente deve inserire le parole `function`, `return`, `prompt` e dare un nome alla variabile globale.

Tra le domande ricorrenti vi è il GapFiller (Fig. 3). Diversamente da un semplice completamento testuale, esegue il codice inserito dallo studente e ne verifica il risultato a runtime, richiedendo ragionamento anziché memorizzazione. Le Figure 4 e 5 mostrano un esempio della prima esercitazione sulle funzioni.

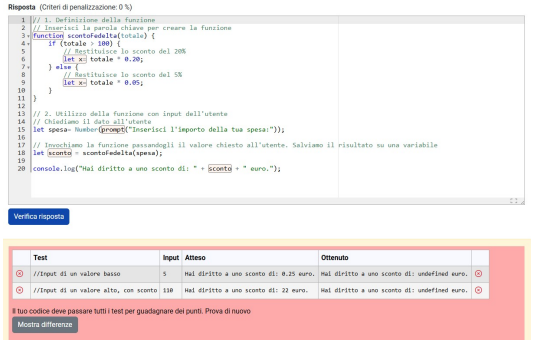


Figura 4: Esempio di domanda GapFiller non completata correttamente: non c'è un `return` nella funzione.

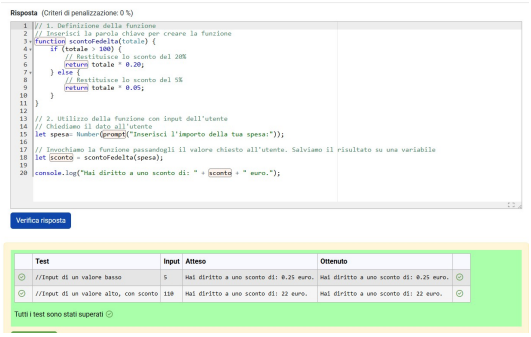


Figura 5: Esempio di domanda GapFiller compilata correttamente.

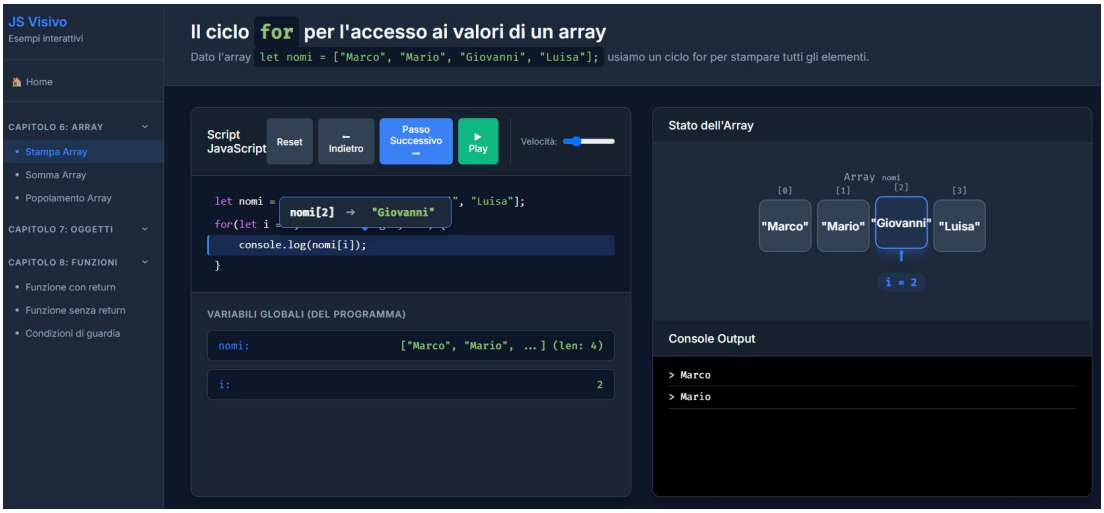


Figura 6: Sito web dimostrativo, consultabile all'indirizzo https://matteocandita-gif.github.io/demo_moodlecoderunner/js_demo/.

4 Il ruolo dell'IA

L'IA generativa è stata centrale nel percorso, rendendo sostenibile la preparazione di una progressione granulare: ha permesso di raffinarla e di generare rapidamente numerosi esercizi dettagliati e coerenti, a partire dalle precise indicazioni del docente. Con il suo ausilio è stato inoltre sviluppato un sito didattico (Fig. 6) che visualizza passo passo l'esecuzione, lo stato delle variabili e il flusso di controllo, supportando la costruzione di modelli mentali corretti dell'esecuzione attraverso una *notional machine* esplicita.

Durante l'anno l'IA è stata introdotta anche per gli studenti, come strumento di supporto all'apprendimento. L'osservazione dell'uso spontaneo di un chatbot generico mostra schemi ricorrenti: lo studente incolla la traccia senza una domanda specifica perché non sa come iniziare, come leggerla o non riesce a identificare ciò che non comprende. Inoltre, il chatbot tende a proporre soluzioni efficienti ma troppo

avanzate, basate su strumenti e costrutti ancora ignoti ai principianti.

Per guidare gli studenti verso un uso dell'IA orientato all'apprendimento, è stato introdotto un **tutor IA personalizzato** basato su Google Gemini[2], scelto per la conformità alle norme scolastiche sulla privacy. Configurato con gli appunti del corso e limitato agli strumenti già affrontati, nelle spiegazioni rimanda alle relative sezioni del materiale.

Il tutor riconosce due situazioni operative distinte:

- **Lo studente incolla la traccia senza una domanda specifica**: il tutor non svolge l'esercizio, né propone mai del codice direttamente. Piuttosto offre allo studente delle scelte per guidarlo verso la soluzione. In particolare offre di:
  - scomporre l'esercizio in step logici;
  - capire quale struttura di controllo usare;
  - creare lo pseudocodice.
- **Lo studente incolla il proprio codice**: il tutor diagnostica l'errore, ne spiega la causa e guida la correzione passo per passo.

In entrambi i casi, il tutor è **configurato per non scrivere soluzioni complete**, ma per guidare il ragionamento dello studente senza sostituirlo.

5 Risultati

Il percorso, sperimentato in due classi terze SIA (42 studenti in totale), ha prodotto risultati positivi. Rispetto agli anni precedenti, gli obiettivi di apprendimento sono stati ampliati: le esercitazioni finali hanno affrontato problemi più complessi e le verifiche hanno richiesto maggiore dettaglio, valutando sia la correttezza funzionale sia la qualità formale del codice.

Nonostante l'aumento del livello di difficoltà, **36 studenti su 42 hanno raggiunto la sufficienza piena**, grazie anche a un efficace recupero in itinere. È un ottimo risultato per studenti al primo anno di apprendimento della programmazione, tanto più che i debiti residui hanno riguardato per lo più studenti con difficoltà diffuse in più discipline.

Il coinvolgimento, costante durante l'anno, è confermato dall'osservazione in classe e dai questionari finali di gradimento (34 rispondenti su 42): giudizio complessivo mediano 8/10 (media 7,6) e mediana 8/10 anche per la chiarezza delle spiegazioni e per il materiale digitale.

La **replicabilità** è stata testata informalmente: colleghi che hanno adottato i materiali nelle proprie classi hanno riportato risultati soddisfacenti, suggerendo la trasferibilità del metodo tra docenti.

6 Conclusioni

Il percorso descritto mostra che una progressione granulare con scaffolding esplicito, supportata da testing automatico su piattaforma, può portare a risultati positivi sia in termini di apprendimento che di gradimento da parte degli studenti.

L'intelligenza artificiale è stata un **abilitatore essenziale**: senza di essa, un singolo docente difficilmente potrebbe produrre i materiali necessari al modello. L'uso da parte degli studenti resta un cantiere aperto: il tutor IA personalizzato ha mostrato potenzialità, ma richiede ulteriori osservazioni

[2]Gem disponibile all'indirizzo https://gemini.google.com/gem/10LGVNjoGQO8fw22XYy7FV73mbc6ocCjj?usp=sharing.

e affinamenti. I risultati non permettono di separare l'effetto dell'IA da quello della riprogettazione complessiva. La ricerca futura dovrà esaminare due piani: per il docente, tempi di produzione, varietà, accuratezza e carico di revisione dei materiali; per gli studenti, tipi di richiesta, progressione degli aiuti, autonomia e trasferimento a prove senza tutor.

Il modello è indipendente dal linguaggio: avviato in terza su JavaScript, nel biennio successivo è stato esteso con successo da altri docenti della scuola a SQL e PHP.

Riferimenti bibliografici

[1] Benjamin S. Bloom. Learning for mastery. *Evaluation Comment*, 1(2):1–12, 1968. https://files.eric.ed.gov/fulltext/ED053419.pdf.

[2] Cruz Izu, Carsten Schulte, Ashish Aggarwal, Quintin I. Cutts, Rodrigo Duran, Mirela Gutica, Birte Heinemann, Eileen T. Kraemer, Violetta Lonati, Claudio Mirolo, and Renske Weeda. Fostering program comprehension in novice programmers: Learning activities and learning trajectories. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '19, pages 27–52, New York, NY, USA, 2019. Association for Computing Machinery. https://doi.org/10.1145/3344429.3372501.

[3] Richard Lobb and Jenny Harlow. CodeRunner: A tool for assessing computer programming skills. *ACM Inroads*, 7(1):47–51, 2016. https://doi.org/10.1145/2810041.

[4] Michael Lodi and Simone Martini. Le Notional machine per l'apprendimento della programmazione, anche nell'era dell'AI generativa. In Maria Angela Pellegrino and Veronica Rossano, editors, *ITADINFO 2025. Terzo Convegno Italiano sulla Didattica dell'Informatica*, pages 29–36, Milano, Italy, 2025. Ledizioni. https://cris.unibo.it/bitstream/11585/1024950/3/Notional2.pdf.

[5] Kasia Muldner, Jay Jennings, and Veronica Chiarelli. A review of worked examples in programming activities. *ACM Transactions on Computing Education*, 23(1):13:1–13:35, 2022. https://doi.org/10.1145/3560266.

[6] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. The robots are here: Navigating the generative AI revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '23, pages 108–159, New York, NY, USA, 2023. Association for Computing Machinery. https://doi.org/10.1145/3623762.3633499.

[7] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research – Volume 1*, pages 27–43, New York, NY, USA, 2022. Association for Computing Machinery. https://doi.org/10.1145/3501385.3543957.

[8] Marco Sbaraglia, Michael Lodi, and Simone Martini. A Necessity-Driven Ride on the Abstraction Rollercoaster of CS1 Programming. *Informatics in Education*, 20(4):641–682, 2021. https://doi.org/10.15388/infedu.2021.28.

[9] Marco Sbaraglia, Michael Lodi, and Simone Martini. Apprendimento della programmazione guidato dalla necessità: il Necessity Learning Design. In Enrica Gentile and Mattia Monga, editors, *ITADINFO. Convegno Italiano sulla Didattica dell'Informatica*, pages 190–195, Bari, Italy, 2023. Università degli Studi di Bari Aldo Moro. https://cris.unibo.it/bitstream/11585/945415/1/Necessity%20Itadinfo.pdf.

[10] Jane Waite and Shuchi Grover. Worked examples and other scaffolding strategies. In Shuchi Grover, editor, *Computer Science in K–12: An A-to-Z Handbook on Teaching Programming*, pages 240–249. Edfinity, Palo Alto, CA, 2020.

Dal codice al prompt: ridefinire apprendimento e valutazione della programmazione nell'era degli assistenti IA

Giovanni Pedroncelli
IISS "G. Marconi – M. Hack", Bari
gi.pedroncelli@gmail.com

Abstract

Gli assistenti LLM rendono delegabile la sintassi del codice, mettendo in crisi la didattica tradizionale. Si propone un framework curricolare per la secondaria di secondo grado che ridefinisce l'apprendimento, spostando il focus dalla produzione alla capacità di specificare, verificare e validare il codice. Il modello include quattro modalità d'uso dell'IA, descrittori formali, una rubrica del prompt come specifica, un esempio e un'agenda di validazione.

1 Introduzione

Gli assistenti di programmazione basati su LLM stanno modificando profondamente il contesto in cui si insegna a programmare. Il gruppo di lavoro ITiCSE 2023 ha documentato che questi strumenti risolvono correttamente la maggior parte degli esercizi introduttivi [5]; gli studenti della secondaria li hanno già adottati, in genere senza mediazione didattica. Strategie basate esclusivamente sulla proibizione risultano difficilmente sostenibili; l'adozione non regolata espone i novizi a rischi metacognitivi documentati [6, 8], in un quadro di opportunità e sfide ormai sistematicamente mappato dalla comunità di ricerca [14]. La letteratura riguarda però quasi esclusivamente i corsi universitari introduttivi ed è più descrittiva dei problemi che prescrittiva di soluzioni e a nostra conoscenza non esistono framework operativi pensati per la secondaria italiana. Nel contesto italiano, la recente revisione dei curricoli degli istituti tecnici [10] rende particolarmente rilevante la necessità di modelli disciplinari per integrare l'IA nell'insegnamento dell'Informatica. Il framework offre uno strumento per progettare l'integrazione su un terreno disciplinare, in coerenza con la Proposta di Indicazioni del Laboratorio CINI [9]: descrivere algoritmi in modo non ambiguo, giustificarne la correttezza, distinguere progettazione e codifica.

La domanda di ricerca è quindi: quali configurazioni didattiche consentono di mantenere e sviluppare le competenze fondamentali della programmazione quando la produzione sintattica del codice è parzialmente delegabile a un agente generativo? La risposta proposta disciplina il ruolo dell'IA in quattro modalità d'uso progressive e ricorsive, spostando il baricentro didattico dalla produzione sintattica del codice a interpretazione, validazione e specificazione rigorosa.

2 Fondamenti teorici

2.1 Un parallelo storico: il prompt come astrazione lossy

L'idea di programmare in linguaggio naturale non è nuova, né lo sono le obiezioni: per Dijkstra [1] la difficoltà della programmazione non deriva dall'obbligo del formalismo; al contrario, è proprio il rigore simbolico lo strumento intellettuale che rende impossibili il non senso e l'incoerenza, tollerati dalle lingue naturali.

Riletto oggi, l'argomento di Dijkstra non squalifica gli assistenti AI, ma ne definisce il costo: il passaggio dal codice al prompt è una trasformazione di astrazione con perdita. Un prompt che produce codice corretto è tale perché vincola l'agente con requisiti completi, casi limite espliciti e criteri di accettazione verificabili. Formulare prompt non è un'abilità comunicativa trasversale, ma un'evoluzione della specifica dei requisiti — contenuto pienamente disciplinare.

L'analogia tra prompt e specifica software non riguarda la semplice capacità di formulare richieste efficaci in linguaggio naturale, ma la natura del processo cognitivo richiesto. Un prompt disciplinare, infatti, non è una descrizione generica dell'obiettivo desiderato, ma una specifica operativa che deve rendere espliciti gli elementi necessari alla realizzazione e alla verifica dell'artefatto. Come nella fase di analisi dei requisiti di un progetto software, lo studente deve innanzitutto possedere una comprensione del dominio applicativo per individuare correttamente il problema da risolvere; deve poi identificare vincoli funzionali e non funzionali, formalizzare i requisiti attesi, prevedere condizioni limite e definire criteri di accettazione che consentano di stabilire se l'output prodotto soddisfa effettivamente la specifica. In questa prospettiva, il prompt non sostituisce la progettazione algoritmica, ma ne rappresenta una nuova superficie di espressione: la qualità dell'output generato dipende dalla qualità della specifica fornita. Questa interpretazione colloca il prompting avanzato in continuità con pratiche fondamentali dell'ingegneria del software: analisi dei requisiti, progettazione architetturale e testing. L'interazione con l'IA diventa così un contesto autentico per esercitare competenze disciplinari tradizionali, anziché un'attività separata di "uso dello strumento".

La ricerca HCI lo conferma sul piano empirico: la formulazione di prompt efficaci non è una competenza spontanea, e i non esperti falliscono sistematicamente, procedendo per tentativi opportunistici anziché per ipotesi strutturate [15]. Può essere dunque interpretata come pratica formale da sviluppare.

2.2 Il nucleo invariante: complessità essenziale e accidentale

Che cosa resta da insegnare, se la codifica è delegabile? Risponde la distinzione di Brooks [2] tra complessità accidentale (limiti degli strumenti, verbosità sintattica) e complessità essenziale (modellazione dei dati, correttezza algoritmica, stati e interazioni del sistema). Come i linguaggi ad alto livello rispetto all'assembly, gli LLM comprimono quasi esclusivamente complessità accidentale. La complessità essenziale — scomposizione del problema, progettazione dell'algoritmo, testing, debugging, valutazione critica dell'architettura — resta in capo allo studente ed è il nucleo disciplinare invariante che il framework presidia. Questi contenuti non sono competenze trasversali: costituiscono Informatica nella sua forma più pura.

2.3 Gerarchia interpretare/produrre e rischi metacognitivi

Il filone empirico inaugurato da Lopez, Whalley, Robbins e Lister [4] ha mostrato che tracciare l'esecuzione del codice e spiegarne lo scopo precede e predice la capacità di scriverlo: leggere viene prima di scrivere. La tassonomia di Fuller et al. [3], che adatta Bloom all'informatica separando l'asse dell'interpretare da quello del produrre, descrive formalmente l'effetto degli LLM: essi riducono il costo cognitivo della produzione sintattica, consentendo di generare artefatti complessi senza che l'asse dell'interpretazione sia stato presidiato.

Da questo scollamento nascono l'*illusion of competence* — lo studente che, vedendo funzionare il codice generato, si convince di averne interiorizzato la logica — e l'*automation bias*, l'accettazione acritica dell'output. Prather et al. [6] mostrano inoltre che la *GenAI* amplia il divario: chi ha basi solide accelera, chi è in difficoltà accumula ostacoli metacognitivi (cfr. anche [8]). Sul piano metodologico, i *Prompt Problems* [7] — esercizi valutati sulla capacità di vincolare l'IA con

una specifica non ambigua — costituiscono il precedente diretto della modalità d'uso finale del framework.

3 Il framework: quattro modalità d'uso

Il framework articola il percorso in quattro modalità d'uso progressive e ricorsive, definiti dal ruolo che l'IA è autorizzata ad assumere. Ogni modalità è descritta da quattro indicatori invarianti — obiettivo di apprendimento (nei termini di [3]), usi consentiti dell'IA, formato di verifica coerente, rischio metacognitivo mitigato — che lo rendono prescrittivo e confrontabile. A differenza di progressioni consolidate come *Use–Modify–Create* [11] e *PRIMM* [12], che regolano l'interazione dello studente con l'artefatto-codice, qui si regola il ruolo dell'agente generativo, allineandovi formato di verifica e rischio metacognitivo.

Modalità d'uso	**Ruolo dell'IA**	**Focus dell'apprendimento**
0. Pre-delega	Nessun supporto generativo	Costruzione del modello mentale
1. IA tutor	Analizza e spiega il codice esistente	Interpretazione e comprensione
2. IA pair programmer	Genera codice sotto supervisione	Progettazione e validazione
3. IA esecutrice	Realizza artefatti a partire da specifiche	Specificazione e progettazione

Tabella 1: La mappa del framework

Ogni modalità è caratterizzata da quattro dimensioni operative: obiettivo, usi consentiti dell'IA, formato di verifica ed evidenza osservabile.

Modalità d'uso	**Evidenza osservabile**	**Verifica**	**Rischio mitigato**
0. Pre-delega	Traccia correttamente l'esecuzione	Tracing, Parsons problems	Delega prematura
1. IA tutor	Individua errori nelle spiegazioni	Spiegare in modo chiaro	Automation bias
2. IA pair programmer	Costruisce test efficaci	Test suite	Illusion of competence
3. IA esecutrice	Produce specifiche variabili	Rubrica prompt	Fallacia di codifica in linguaggio naturale

Tabella 2: Caratterizzazione operativa del framework

Mentre il riordino ITT [10] ridisegna l'intero impianto orario, i licei restano regolati dalle Indicazioni Nazionali del 2010 [13]: la progressione qui proposta deve quindi adattarsi a due logiche curricolari distinte. La progressione è parametrica rispetto all'ordinamento. Nell'indirizzo ITT Informatica e Telecomunicazioni si distende sul triennio — modalità 0–1 nel terzo anno, 2 nel quarto, 3 nel quinto — con regressioni locali a ogni nuovo paradigma o struttura dati. Nel Liceo Scientifico opzione Scienze Applicate, dove l'Informatica dispone di due ore settimanali e di una funzione anche metodologica per le scienze [13], le modalità operano come microciclo ricorsivo entro ciascun nucleo tematico; la modalità 3 vi si declina nella specificazione di modelli computazionali e simulazioni, in raccordo con matematica e fisica.

3.1 La rubrica del prompt-come-specifica

Nella modalità 3 la valutazione si distacca dal codice prodotto e si concentra sulla specifica formulata dallo studente, operazionalizzando le tesi di [1] e [2]. La rubrica valuta quattro dimensioni disciplinari su quattro livelli di padronanza. Le dimensioni riprendono attività tipiche dell'ingegneria del software — analisi dei requisiti, progettazione, testing e validazione — reinterpretate nel contesto dell'interazione con un agente generativo. La rubrica rende il prompt un artefatto valutabile con criteri disciplinari e comparabile tra studenti e classi.

Dimensione	Base	Intermedio	Avanzato	Esperto
Completezza dei requisiti	Descrive l'obiettivo generale in modo ambiguo ("crea un programma per..."). Mancano informazioni sul comportamento atteso e sui vincoli.	Specifica alcune funzionalità principali, ma lascia impliciti diversi aspetti del comportamento del sistema.	Definisce requisiti funzionali chiari e alcuni vincoli non funzionali (prestazioni, struttura dati, gestione errori).	Formula una specifica completa distinguendo requisiti funzionali e non funzionali, vincoli progettuali, assunzioni e condizioni di corretto funzionamento.
Gestione delle ambiguità e dei casi limite	Non considera situazioni anomale; assume implicitamente che gli input siano sempre validi.	Identifica alcuni casi problematici evidenti, ma senza una strategia sistematica di gestione.	Prevede condizioni limite, input non validi ed eccezioni principali, specificando il comportamento atteso.	Anticipa sistematicamente scenari limite, condizioni al contorno e casi di errore, definendo esplicitamente la risposta del sistema in ciascuna situazione.
Vincoli architetturali e progettuali	Richiede un risultato senza indicare struttura, modello dei dati o criteri progettuali.	Suggerisce alcuni elementi implementativi ma senza motivarne la necessità.	Specifica strutture dati, organizzazione dei componenti o scelte architetturali coerenti con il problema.	Definisce vincoli architetturali motivati, richiede separazione delle responsabilità, impone scelte progettuali coerenti con requisiti di efficienza, manutenibilità e sicurezza.
Criteri di accettazione e verifica	Non definisce come valutare la correttezza dell'output prodotto.	Indica genericamente che il programma deve "funzionare" o produrre il risultato atteso.	Definisce condizioni di successo, esempi di input/output e casi di test significativi.	Specifica criteri di accettazione verificabili a priori, include una strategia di testing, asserzioni e casi rappresentativi necessari per validare la soluzione.

Tabella 3: La rubrica

La consegna impone una specifica completa a monte — antidoto didattico al prompting esplorativo non strutturato — ma riconosce le iterazioni successive come debugging della specifica: la traccia delle revisioni documenta se lo studente ha diagnosticato la causa dell'output difforme (requisito mancante, ambiguità, vincolo violato) ed è essa stessa oggetto di valutazione, accanto al prompt finale. La traccia delle revisioni può essere raccolta tramite esportazione automatica della sessione, dove la piattaforma lo consente, o tramite una scheda strutturata a tre campi per iterazione (problema rilevato, modifica apportata, risultato); la granularità richiesta deve essere sufficiente a

distinguere revisioni consapevoli da tentativi esplorativi, non a documentare ogni singola interazione.
Alla revisione del codice generato si affianca inoltre un audit di sicurezza: la letteratura mostra che gli assistenti producono con frequenza non trascurabile codice funzionalmente corretto ma vulnerabile [16], e l'individuazione di tali vulnerabilità — naturale terreno di raccordo con i contenuti di cybersecurity dell'indirizzo Informatica e Telecomunicazioni — è parte integrante delle competenze della modalità 3. L'audit di sicurezza, a livello della scuola secondaria, non richiede strumenti professionali di analisi statica: è sufficiente una checklist strutturata focalizzata su categorie di vulnerabilità curricolari — gestione delle eccezioni, validazione degli input, assenza di credenziali in chiaro — applicata come estensione naturale della test suite della modalità 3.

4 Un esempio operativo

Si consideri un problema di progettazione software: realizzare un sistema per la gestione dei prestiti di una biblioteca scolastica. Il sistema deve rappresentare libri e utenti, registrare i prestiti, verificare la disponibilità dei volumi e gestire condizioni anomale (libro inesistente, utente non registrato, limite massimo di prestiti, restituzioni oltre la scadenza). Lo stesso problema viene affrontato nelle quattro modalità del framework, modificando progressivamente il ruolo dell'IA e l'oggetto della valutazione.

4.1 Modalità 0: costruzione del modello mentale

Lo studente lavora senza assistenza generativa su una versione semplificata del problema. Prima della scrittura del codice deve rappresentare manualmente lo stato del sistema: ad esempio quali informazioni devono essere associate a un libro, a un utente e a un prestito, e come cambia la situazione dopo una sequenza di operazioni. Le attività privilegiano interpretazione e ragionamento algoritmico: lo studente costruisce tabelle di esecuzione, traccia il valore delle variabili dopo alcune operazioni e individua la struttura dati più adatta (liste, dizionari, oggetti). La verifica non riguarda la quantità di codice prodotto, ma la capacità di prevedere correttamente il comportamento del sistema. Il rischio mitigato è la delega prematura: lo studente deve prima possedere un modello mentale del problema prima di affidare la traduzione in codice a uno strumento esterno.

4.2 Modalità 1: interpretazione e validazione del codice

Lo studente riceve un'implementazione già realizzata, eventualmente contenente errori progettuali o casi non gestiti, e utilizza l'IA esclusivamente come supporto interpretativo. Ad esempio, può chiedere: "Spiega come questa funzione determina se un libro è disponibile e quali ipotesi fa sul formato dei dati." oppure: "Individua quali casi limite non sono gestiti da questo codice." L'output dell'IA non è accettato automaticamente: lo studente deve confrontarlo con il proprio modello mentale, verificando se la spiegazione è coerente con il comportamento effettivo del programma. La prova consiste quindi nella capacità di spiegare il codice, individuare errori e motivare eventuali correzioni. L'IA diventa uno strumento di confronto cognitivo, non un generatore di soluzioni.

4.3 Modalità 2: progettazione dei test e validazione dell'artefatto

Lo studente può utilizzare l'IA per generare parti del programma, ad esempio una funzione: `presta_libro(libro, utente)` oppure una classe per rappresentare il catalogo. La competenza valutata però non è la capacità di ottenere codice funzionante, perché la generazione è delegata. Lo

studente deve prima definire il comportamento atteso attraverso una strategia di testing. Deve quindi individuare casi come:

- prestito di un libro disponibile;
- richiesta di un libro già in prestito;
- utente inesistente;
- superamento del numero massimo di prestiti;
- restituzione di un libro mai prestato.

Successivamente confronta il codice generato con la propria test suite, analizzando se l'output dell'IA soddisfa realmente i requisiti. Il focus cognitivo si sposta dalla produzione alla validazione: lo studente non dimostra di saper scrivere ogni riga, ma di saper giudicare la correttezza di una soluzione.

4.4 Modalità 3: specificazione del sistema

Nella modalità finale lo studente non chiede più semplicemente: "Scrivi un programma per gestire una biblioteca" ma produce una specifica completa destinata all'agente generativo. Ad esempio deve definire:

- quali entità devono essere rappresentate (Libro, Utente, Prestito);
- quali proprietà possiedono;
- quali vincoli devono essere rispettati;
- quali eccezioni devono essere gestite;
- quali criteri determinano la correttezza del sistema.

Una specifica adeguata potrebbe richiedere:

"Progetta un sistema object-oriented in Python per una biblioteca scolastica, separando catalogo, utenti e gestione prestiti. La ricerca di un libro per ISBN deve avere complessità O(1): scegli la struttura dati di conseguenza. Ogni prestito registra data di consegna e scadenza (30 giorni). Un utente non può avere più di tre prestiti attivi. Gestisci con eccezioni tipizzate: libro inesistente, libro non disponibile, utente inesistente, limite prestiti raggiunto, restituzione di un prestito non registrato. Fornisci una test suite con asserzioni per ciascuno di questi cinque casi più un prestito andato a buon fine."

La valutazione riguarda la qualità della specifica: completezza dei requisiti, capacità di anticipare problemi, coerenza architetturale e definizione dei criteri di accettazione. In questa modalità il prompt diventa un artefatto informatico valutabile perché incorpora attività tipiche dell'ingegneria del software: analisi dei requisiti, modellazione, progettazione architetturale e testing.

5 Prospettive di validazione empirica e disegno previsto dello studio

Il framework è qui proposto su base teorica: la validazione empirica ne costituisce l'agenda necessaria, a partire dalla stima dell'affidabilità inter-valutatore della rubrica, prerequisito per qualsiasi confronto tra classi. Il disegno previsto è quasi-sperimentale, su classi parallele dello stesso indirizzo, con una condizione sperimentale — progressione strutturata secondo le quattro modalità — e una di confronto con uso non regolamentato della *GenAI*, per una durata sufficiente a completare almeno due cicli tematici completi. L'ipotesi falsificabile: gli studenti formati con la progressione a modalità mostrano prestazioni di comprensione del codice non inferiori a quelle di gruppi senza *GenAI* e capacità di specificazione e progettazione superiori a quelle di gruppi con uso non regolamentato.

6 Conclusioni

Il framework offre alla scuola secondaria una terza via tra divieto e *laissez-faire*, riportando l'incontro tra studente e IA generativa sul terreno della disciplina. Elevare il prompt da scorciatoia a veicolo di specificazione formale significa preservare la complessità essenziale come oggetto di insegnamento e mantenere lo studente nel ruolo di progettista e validatore — il ruolo che l'Informatica ha sempre inteso formare.

Bibliografia

[1] E. W. Dijkstra, *On the foolishness of "natural language programming"*, EWD667, 1978 (poi in *Program Construction*, LNCS 69, Springer, 1979).

[2] F. P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering", *IEEE Computer*, 20(4), pp. 10–19, 1987.

[3] U. Fuller et al., "Developing a computer science-specific learning taxonomy", *ACM SIGCSE Bulletin*, 39(4), pp. 152–170, 2007.

[4] M. Lopez, J. Whalley, P. Robbins, R. Lister, "Relationships between reading, tracing and writing skills in introductory programming", in *Proc. ICER '08*, ACM, pp. 101–112, 2008.

[5] J. Prather et al., "The Robots Are Here: Navigating the Generative AI Revolution in Computing Education", in *Proc. ITiCSE-WGR '23*, ACM, 2023.

[6] J. Prather et al., "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers", in *Proc. ICER '24*, ACM, 2024.

[7] P. Denny, J. Leinonen, J. Prather et al., "Prompt Problems: A New Programming Exercise for the Generative AI Era", in *Proc. SIGCSE '24*, ACM, 2024.

[8] M. Kazemitabaar et al., "Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming", in *Proc. CHI '23*, ACM, 2023.

[9] Laboratorio Informatica e Scuola, CINI, *Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola*.

[10] Ministero dell'Istruzione e del Merito, *Decreto Ministeriale 19 febbraio 2026, n. 29*, attuativo degli artt. 26 e 26-bis del DL 144/2022.

[11] I. Lee et al., "Computational thinking for youth in practice", *ACM Inroads*, 2(1), pp. 32–37, 2011.

[12] S. Sentance, J. Waite, "PRIMM: Exploring pedagogical approaches for teaching text-based programming in school", in Proc. *WiPSCE '17*, ACM, 2017.

[13] MIUR, *Decreto Ministeriale 7 ottobre 2010, n. 211* (Indicazioni Nazionali per i percorsi liceali).

[14] B. A. Becker, P. Denny, J. Finnie-Ansley et al., "Programming Is Hard – Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation", in *Proc. SIGCSE '23*, ACM, pp. 500–506, 2023.

[15] J. D. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, Q. Yang, "Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts", in *Proc. CHI '23*, ACM, 2023.

[16] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, R. Karri, "Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions", in *Proc. IEEE Symposium on Security and Privacy (S&P)*, pp. 754–768, 2022.

Dall'introduzione dell'Intelligenza Artificiale alla progettazione di soluzioni reali: il caso UrbanAI

Barbara Nardi[1], Daniele Rogano[1], Francesco Rende[1]
[1] Istituto Tecnico Industriale "A.Monaco" - Cosenza
{barbara.nardi, daniele.rogano, francesco.rende}@itimonaco.it

Abstract

L'introduzione dell'Intelligenza Artificiale nella scuola secondaria richiede percorsi che consentano agli studenti la comprensione dei principi del machine learning e il loro impiego nello sviluppo software, superando un approccio limitato al semplice utilizzo di strumenti già disponibili.
Il contributo presenta UrbanAI, sviluppato presso l'Istituto Tecnico Industriale "A.Monaco" di Cosenza, come evoluzione di AI-Maker, laboratorio introduttivo sul machine learning. A partire da un prototipo di classificazione realizzato con *Teachable Machine*[1], gli studenti hanno progettato e sviluppato un'applicazione mobile per il riconoscimento di suoni urbani a supporto di persone sorde e ipoacusiche.
UrbanAI ha consentito di affrontare contenuti disciplinari quali classificazione supervisionata, costruzione e validazione di dataset, integrazione di modelli di TensorFlow Lite, testing e sviluppo collaborativo [1][2].
L'esperienza propone un modello di percorso verticale nel quale un laboratorio introduttivo evolve in un progetto autentico di sviluppo software.

1 Introduzione

La crescente disponibilità di strumenti basati sull'IA rende possibile introdurre il machine learning anche nei primi anni della scuola secondaria. La sfida didattica consiste nel progettare attività che favoriscano la comprensione dei modelli e l'impiego nello sviluppo software, evitando un apprendimento limitato all'utilizzo di applicazioni già disponibili.

In questa prospettiva si colloca UrbanAI, premiato nell'ambito del Premio Nazionale sull'Innovazione Digitale promosso da Anitec-Assinform[2]. L'obiettivo era utilizzare l'IA come strumento per applicare conoscenze informatiche per rispondere ad un bisogno reale.

Nel percorso descritto l'IA costituisce il contesto entro il quale affrontare contenuti disciplinari quali rappresentazione dei dati, classificazione supervisionata, progettazione software e validazione dei modelli attraverso un problema autentico [2][3].

L'esperienza si articola in due fasi: AI-Maker, laboratorio introduttivo sul machine learning per studenti del primo anno, e UrbanAI, evoluzione di una delle idee sviluppate durante il laboratorio.

2 Dalla sperimentazione alla progettazione: un percorso verticale

UrbanAI nasce come evoluzione di AI-Maker, laboratorio extracurricolare pensato per introdurre gli studenti del primo anno al machine learning mediante Teachable Machine,

[1] **Teachable Machine** è una piattaforma web-based sviluppata da Google per la creazione di modelli di machine learning senza l'uso di codice. https://teachablemachine.withgoogle.com/
[2] **Anitec-Assinform** - Associazione Italiana per l'Information and Communication Technology (ICT). https://www.anitec-assinform.it/

piattaforma no-code per raccolta dati, addestramento e valutazione del modello senza affrontare la complessità della programmazione.

Uno dei prototipi sviluppati proponeva il riconoscimento automatico dei suoni urbani per supportare le persone sorde e ipoacustiche negli spostamenti quotidiani. L'idea si rivelò promettente al punto da suggerire un ulteriore sviluppo, UrbanAI, un'applicazione mobile destinata a rispondere a un bisogno reale.

Il progetto inizialmente coinvolgeva quattro studenti del primo anno, tre ragazzi e una ragazza, provenienti dal percorso AI-Maker. Poiché la progettazione richiedeva competenze sull'intero processo di sviluppo software, il gruppo venne ampliato con quattro studenti del quinto anno e cinque studentesse del quarto, favorendo una collaborazione tra studenti con età, esperienze e livelli di competenza differenti. Le attività hanno richiesto frequenti momenti di confronto, nei quali le competenze tecniche e organizzative dei diversi componenti venivano messe a disposizione del gruppo.

Il progetto si è articolato nell'arco di circa cinque mesi, con 60 ore di attività extracurriculare, oltre al lavoro autonomo.

La partecipazione al Premio Nazionale sull'Innovazione Digitale promosso da Anitec-Assinform ha rappresentato un ulteriore elemento di crescita del progetto, favorendone l'evoluzione in un'applicazione più strutturata.

3 Il percorso didattico

3.1 Scelte metodologiche

UrbanAI è stato sviluppato secondo un approccio di **Project Based Learning (PBL)**[1][2], organizzando l'apprendimento attorno alla progettazione di una soluzione destinata a rispondere a un problema reale. La sperimentazione di strumenti e tecnologie, il confronto tra studenti con competenze differenti e il dialogo con i futuri utilizzatori dell'applicazione hanno fatto emergere pratiche riconducibili al **Learning by Doing**[3], al **Peer Learning** e allo **User-Centered Design**[5].

3.2 Sviluppo del percorso

Il percorso è iniziato con l'analisi del problema e delle soluzioni già disponibili per le persone sorde e ipoacusiche. Il confronto con l'Ente Nazionale Sordi ha consentito di validare i requisiti dell'applicazione, mentre la collaborazione con SOMOS S.r.l. ha introdotto gli studenti a pratiche proprie dello sviluppo software professionale, con attenzione alla progettazione dell'interfaccia utente e all'integrazione del modello di IA.

A differenza delle tradizionali attività laboratoriali nelle quali problema, requisiti e obiettivi sono generalmente definiti a priori, gli studenti hanno partecipato anche alla loro progressiva definizione. L'analisi del contesto d'uso e il confronto con i futuri utilizzatori hanno costituito parte integrante del processo di apprendimento, avvicinando il progetto alle pratiche proprie dello sviluppo software reale.

Il lavoro è stato organizzato in tre aree principali: sviluppo, interfaccia grafica e documentazione. Gli studenti del quinto anno hanno costituito il nucleo tecnico del progetto, occupandosi dell'implementazione dell'architettura applicativa, dell'interfaccia e del modulo di intelligenza artificiale; gli studenti del primo anno hanno lavorato su esperienza utente, mockup e prototipi; le studentesse del quarto anno hanno curato la documentazione tecnica, contenuti progettuali e materiali di comunicazione.

La suddivisione dei compiti non ha comportato una separazione rigida delle attività. Il lavoro ha richiesto un confronto continuo: le scelte relative all'interfaccia dovevano essere condivise con chi si occupava dello sviluppo, le esigenze dovevano essere tradotte in soluzioni comprensibili agli studenti impegnati nelle attività grafiche e documentali e i risultati delle diverse fasi venivano periodicamente verificati dal gruppo. In questo processo, gli studenti più esperti hanno fornito supporto tecnico ai compagni più giovani, mentre questi ultimi hanno contribuito alla definizione dell'esperienza utente e alla verifica delle soluzioni adottate. La necessità di condividere procedure e scelte tecniche ha trasformato le competenze individuali in patrimonio condiviso del gruppo.

L'applicazione integra un modello TensorFlow Lite che esegue il riconoscimento dei suoni direttamente sul dispositivo, senza connessione Internet. Quando il sistema identifica uno dei suoni previsti, avvisa l'utente mediante vibrazione e notifica visiva.

Con l'avanzare del progetto, il ruolo dei docenti è passato dal coordinamento alla supervisione, lasciando agli studenti crescente autonomia nelle decisioni progettuali. I docenti hanno accompagnato le diverse fasi, intervenendo soprattutto nei passaggi che richiedevano l'integrazione tra competenze differenti o revisione delle scelte progettuali.

La validazione finale ha rappresentato il momento più significativo del progetto. Installata su uno smartphone in modalità aereo, l'applicazione ha riconosciuto correttamente il suono attivando la vibrazione. La verifica ha reso concreto l'intero percorso svolto fino alla realizzazione di uno strumento funzionante.

4 Risultati didattici

L'esperienza conferma come l'IA possa rappresentare un efficace contesto per l'insegnamento dell'Informatica quando è inserita all'interno di un progetto orientato alla soluzione di un problema reale, anziché proposta come semplice contenuto disciplinare.

Dal punto di vista disciplinare, il progetto ha permesso agli studenti di sperimentare l'intero ciclo di sviluppo di un sistema basato su IA: definizione dei requisiti, costruzione del dataset, addestramento del modello, integrazione nell'applicazione e testing finale. La collocazione di tali attività all'interno di un unico progetto ha favorito la comprensione delle relazioni tra progettazione software, programmazione e machine learning.

La collaborazione tra studenti di anni di corso differenti ha inoltre reso necessario esplicitare e condividere conoscenze tecniche, trasformando il lavoro di squadra in una concreta occasione di apprendimento tra pari e contribuendo allo sviluppo di competenze trasversali quali problem solving, pensiero critico e capacità di comunicazione.

Particolarmente significativa si è rivelata la fase di addestramento del modello. I risultati iniziali hanno richiesto più revisioni del dataset e una riduzione del numero di classi riconosciute, privilegiando l'affidabilità del sistema. Gli studenti hanno così sperimentato il carattere iterativo dello sviluppo di applicazioni basate sul machine learning, comprendendo il ruolo determinante della qualità dei dati e delle scelte progettuali.

Le riflessioni raccolte al termine dell'esperienza indicano che il lavoro di gruppo e la progressiva autonomia progettuale hanno sostenuto l'apprendimento dei contenuti disciplinari. Tali osservazioni costituiscono evidenze qualitative coerenti con gli obiettivi formativi del percorso, pur non consentendo generalizzazioni di carattere statistico.

5 Discussione e conclusioni

L'esperienza di UrbanAI suggerisce che l'introduzione dell'IA nella scuola secondaria possa risultare più efficace quando inserita in un percorso di progettazione software orientato alla soluzione di un problema autentico. In questa prospettiva, l'IA diventa così un contesto nel quale convergono conoscenze proprie dell'Informatica, dalla rappresentazione dei dati alla progettazione software.

Gli studenti sono stati coinvolti nella definizione di requisiti e obiettivi attraverso l'analisi del contesto e il confronto con i futuri utilizzatori. La definizione del problema ha così assunto un ruolo formativo pari a quello della realizzazione della soluzione.

La composizione verticale del gruppo ha favorito la distribuzione dei ruoli e l'apprendimento tra pari.

Il momento conclusivo del progetto ha coinciso con la validazione dell'applicazione in condizioni operative reali. Installata su uno smartphone e utilizzata senza connessione alla rete, UrbanAI ha riconosciuto correttamente i suoni attivando la vibrazione, verificando concretamente l'integrazione tra dati, modello e software.

Il principale risultato consiste nella definizione di un percorso verticale nel quale un laboratorio introduttivo sul machine learning evolve in un progetto autentico di sviluppo software. Tale modello appare trasferibile in altri contesti della scuola secondaria, offrendo un possibile riferimento per l'insegnamento dell'Informatica.

6 Ringraziamenti

Si ringraziano gli studenti che hanno partecipato con impegno ed entusiasmo durante tutte le fasi del progetto.

Si ringrazia la Dirigente Scolastica Fiorangela D'Ippolito per aver sostenuto il progetto fin dalle sue fasi iniziali.

Si ringraziano l'Ente Nazionale Sordi e SOMOS S.r.l. per il sostegno e il contributo forniti alla realizzazione del progetto.

Si ringrazia infine Anitec-Assinform, il cui Premio Nazionale sull'Innovazione Digitale ha rappresentato un'importante occasione di crescita.

Bibliografia

[1] Rotta, M. (2007). Il Project Based Learning nella scuola: implicazioni, prospettive e criticità. Journal of e-Learning and Knowledge Society, 3(1), 75-84.

[2] Amiruddin, M., Hafeez, R., & Humaira, N. (2025). Integrating Artificial Intelligence and Project-Based Learning: A Framework for Enhancing AI Literacy in Secondary Education. *Journal of Education and Applied Teaching (JEAT)*, 1(1), 22-32.

[3] Dewey, J. (1916). *Democracy and Education: An Introduction to the Philosophy of Education*. Macmillan.

[4] Ross, D. (2025, September 4). AI-Aware Roles for PBL Collaborative Work. *Getting Smart*.

[5] Norman, D. A. (2013). *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books.

Programmare senza guardare: resoconto di un'esperienza di insegnamento*

Graziano Battisti[1], Luca Forlizzi[12], Giovanna Melideo[12], and Gaia Piersanti[1]

[1] Dipartimento di Ingegneria e scienze dell'informazione e matematica, Università degli Studi dell'Aquila, Italia
[2] Laboratorio Nazionale CINI "Informatica e Scuola"
{graziano.battisti,luca.forlizzi,giovanna.melideo}@univaq.it,
gaia.piersanti@student.univaq.it

Abstract

Insegnare la programmazione a persone cieche pone sfide legate all'accesso ai contenuti, alla comprensione del codice e all'uso degli ambienti di sviluppo, spesso progettati per una fruizione visiva. Il contributo descrive un'esperienza didattica con uno studente cieco, mettendo in evidenza criticità, strategie di adattamento e buone pratiche orientate all'universal design, potenzialmente trasferibili in contesti scolastici e universitari.

1 Introduzione

La programmazione è una competenza centrale nella formazione informatica, ma il suo insegnamento non è ugualmente accessibile a tutti. Per studenti ciechi o con grave disabilità visiva, l'accesso ai contenuti, al codice e agli strumenti di sviluppo avviene spesso tramite tecnologie assistive come screen reader e display Braille. Lo screen reader consente l'accesso vocale ai contenuti digitali, ma risulta inutilizzabile in molti ambienti e strumenti di programmazione. Il display Braille offre una lettura tattile più precisa del testo ed è spesso preferito nella consultazione del codice, ma impone una consultazione sequenziale e localizzata, riga per riga o per piccoli segmenti, che rende più complesso orientarsi nei documenti e gestire attività articolate [4, 7].

Queste difficoltà assumono particolare rilievo nella didattica della programmazione, poiché molti ambienti di sviluppo presuppongono una fruizione visiva diretta e nozioni come struttura del codice, organizzazione della memoria, flusso di esecuzione e relazioni tra procedure e dati vengono spesso presentate mediante rappresentazioni diagrammatiche difficili da rendere in forma accessibile. È quindi necessario un approccio didattico che riduca la dipendenza dal canale visivo e renda più esplicita la struttura dell'informazione, supportando la navigazione tramite tecnologie assistive. Questo vale anche per la modalità di erogazione delle lezioni, che deve chiarire i passaggi del ragionamento, verbalizzare con precisione codice e materiali ed evitare riferimenti esclusivamente visivi [9].

Nel corrente anno accademico, il Corso di Laurea in Informatica ha accolto uno studente cieco ('Andrea'), pur in assenza, all'inizio dell'anno, di una preparazione specifica per questo tipo di esigenza. Il contributo descrive l'esperienza dei docenti di "Laboratorio di Programmazione di Sistema" (nel seguito, LPS), un insegnamento del primo anno che sviluppa competenze sulla programmazione sia concettuali che operative. La presenza di Andrea ha reso evidente che l'accessibilità non riguarda soltanto i testi e i materiali di studio, ma anche gli strumenti di sviluppo e simulazione, i programmi di esempio e il modo in cui i concetti vengono spiegati e rappresentati.

L'obiettivo, condiviso da Andrea, è stato garantire un'inclusione reale [8], in linea con la Convenzione ONU sui diritti delle persone con disabilità e con le Linee guida della Conferenza Nazionale

*Questo lavoro è stato reso possibile grazie alla collaborazione di Andrea, lo studente coinvolto nell'esperienza didattica, il cui contributo è stato essenziale per definire e valutare le strategie di accessibilità adottate.

Universitaria dei Delegati per la Disabilità [1]. Ciò ha significato metterlo in condizione di seguire con profitto le lezioni in classe con i compagni e di acquisire conoscenze e abilità trasferibili in contesti lavorativi ordinari. La situazione ha richiesto un rapido adeguamento di materiali, strumenti e pratiche didattiche, in cui Andrea è stato parte attiva, ma ha anche mostrato quanto alcune scelte sarebbero state più semplici da integrare se il corso fosse stato progettato fin dall'inizio in un'ottica inclusiva. In linea con quanto previsto dall'Universal Design for Learning, un principio guida è stato che l'accessibilità non implica un indebolimento degli obiettivi formativi: gli obiettivi di apprendimento sono rimasti invariati, mentre sono cambiati il modo in cui i contenuti vengono presentati, il codice viene scritto e gli strumenti vengono scelti. In particolare, è risultato essenziale presentare i concetti attraverso diverse forme e canali di comunicazione, combinando i supporti visivi come illustrazioni, grafici e tabelle con l'esposizione testuale per rispondere alle esigenze eterogenee che ogni studente può manifestare.

Alla luce di queste considerazioni, il percorso presentato si colloca in piena sintonia con la Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola [2], per l'attenzione alla programmazione, all'accessibilità come criterio progettuale e allo sviluppo di competenze concettuali e operative trasferibili, e apre a una riprogettazione didattica fondata sul valore preventivo dell'universal design che invita a progettare in anticipo per ridurre il più possibile le barriere, invece di intervenire soltanto in risposta all'emergenza [3].

2 Contesto

LPS si propone di illustrare come vengono realizzate le astrazioni fondamentali dei linguaggi di programmazione ad alto livello, chiarendo la relazione tra codice sorgente e corrispondente traduzione in linguaggio macchina. Il corso approfondisce la programmazione imperativa attraverso C ed Assembly e introduce strumenti e tecniche di programmazione propedeutici ai successivi insegnamenti, con particolare riferimento a temi fondamentali come allocazione della memoria, gestione delle eccezioni e dell'input/output, rappresentazione in memoria delle strutture dati ed ambienti di esecuzione di programmi forniti dai sistemi operativi. Si tratta dunque di un insegnamento che richiede non solo abilità operative, ma anche capacità di astrazione e comprensione delle relazioni tra codice, memoria e vari componenti di un sistema di elaborazione.

Le lezioni alternano spiegazioni frontali svolte con l'ausilio di slide ed esempi di programmi già sviluppati, con sessioni di *live coding* in cui il docente scrive codice in tempo reale davanti agli studenti, rendendo visibili il processo, gli errori e il ragionamento passo per passo. Il percorso didattico offre numerose opportunità di cimentarsi in (o assistere a) attività pratiche di programmazione, supportate dall'uso di molteplici strumenti di sviluppo, tra cui un ambiente di sviluppo integrato (IDE) appositamente progettato per la didattica della programmazione in Assembly [5]. Molti di questi strumenti si affidano a rappresentazioni grafiche o altri elementi visivi per migliorare l'esperienza degli utenti. Il materiale didattico consiste in un testo di riferimento sulla programmazione in linguaggio C, oltre a slide, dispense ed esempi di codice C ed Assembly forniti dal docente.

Le attività sono sostenute dalla presenza nel team didattico di un cultore della materia, che affianca il docente nella conduzione delle esercitazioni e nel supporto agli aspetti operativi del corso, e di una tutor tesista, che contribuisce alla mediazione didattica, al raccordo con gli studenti e al monitoraggio delle esigenze emerse durante lo svolgimento dell'insegnamento. Il team si avvale anche della delegata del Dipartimento per la disabilità, che contribuisce al coordinamento delle azioni di inclusione e al monitoraggio delle esigenze emerse durante lo svolgimento dell'insegnamento.

3 Adattamenti didattici

L'accessibilità dell'esperienza didattica è stata affrontata agendo su tre dimensioni principali: lezioni, materiali di studio e strumenti di programmazione. Questi ambiti, distinti ma interdipendenti, concorrono congiuntamente a definire la partecipazione di Andrea al percorso formativo.

Lezioni e verbalizzazione. Per consentire ad Andrea di partecipare pienamente alle lezioni e di studiare in autonomia, e per favorire il collegamento tra spiegazione orale, materiali di studio e attività pratica, si è rivelato molto utile aprire ogni lezione con la presentazione degli obiettivi e della struttura dell'esposizione, e sviluppare successivamente i contenuti richiamando costantemente i materiali forniti in anticipo ad Andrea.

Durante la lezione, il docente annuncia esplicitamente i file e gli esempi di volta in volta utilizzati, per agevolare l'individuazione del materiale corrispondente e il raccordo tra spiegazione orale, testo scritto e attività pratica. Questa impostazione richiede tuttavia una continua mediazione tra il ritmo della lezione e i tempi di consultazione dei contenuti da parte di Andrea. La possibilità di seguire l'esposizione dipende infatti dalla chiarezza della verbalizzazione, dall'organizzazione preventiva dei materiali e dalla possibilità di orientarsi rapidamente tra i file e al loro interno.

In quest'ottica, la conduzione della lezione ha richiesto una verbalizzazione più esplicita da parte del docente, con minore ricorso a espressioni come "qui" o "come si vede" e maggiore attenzione a nominare file, costrutti, passaggi e relazioni. Tali accorgimenti, introdotti per esigenze di accessibilità, hanno prodotto un miglioramento generale della chiarezza espositiva. Inoltre, si è deciso di fornire preventivamente il materiale didattico, rispondendo a una duplice esigenza: da un lato, facilitare l'orientamento di Andrea durante la lezione; dall'altro, favorire una modalità di apprendimento più attiva, in linea con la letteratura sulla *flipped classroom*, anticipando una parte dello studio individuale e riservando il tempo in aula al chiarimento dei passaggi più critici e alle attività guidate [6]. Questo approccio richiede un notevole impegno da parte dei docenti e un livello di autonomia non scontato da parte degli studenti. Nel caso descritto, la forte motivazione di Andrea ha reso possibile sostenere questo carico di lavoro e, allo stesso tempo, la necessità di predisporre materiali adeguati ha spinto anche i docenti a un maggiore rigore organizzativo, riducendo il rischio di rinviare attività che tendono ad essere procrastinate.

Materiali di studio accessibili. La revisione preventiva dei materiali in un'ottica di accessibilità costituisce un requisito essenziale. I formati più comunemente condivisi, come il PDF, non assicurano una piena fruibilità dei contenuti mediante tecnologie assistive, soprattutto in presenza di formule, tabelle o altri elementi strutturati.

Per dispense e slide, il problema è stato affrontato fornendo ad Andrea file sorgente LaTeX, opportunamente rielaborati per migliorarne l'accessibilità nell'organizzazione del testo e nella formattazione, senza alterarne il contenuto. Il formato LaTeX era già usato dal docente, era noto ad Andrea e da lui preferito rispetto ad altri. Esso offre la possibilità di fornire testo semplice facilmente epurabile da elementi grafici non essenziali e da formattazioni superflue, con ambienti riconoscibili e formule esprimibili in forma piana con una sintassi chiara; caratteristiche che consentono un accesso più lineare ai contenuti e una consultazione più efficace sia durante le lezioni sia nello studio individuale. Per ridurre il numero di caratteri impiegati, in alcuni casi può essere utile introdurre semplici definizioni di macro per rendere il codice sorgente più compatto e leggibile. Ad esempio, è possibile abbreviare `\mathbb{R}` in `\R` per denotare l'insieme dei numeri reali.

Per risultare davvero fruibili, i sorgenti LaTeX dovrebbero avere nomi evocativi ed essere organizzati in modo sistematico, con un preambolo separato dai file di contenuto e con capitoli e immagini collocati in directory distinte. È buona pratica descrivere esplicitamente tale struttura e collocare le definizioni di macro in apertura del file principale, così da orientare lo studente nella ricerca dei file e

nell'interpretazione di testo e formule. Inoltre, l'uso combinato di macro e di specifiche opzioni di compilazione nel preambolo, permette di ottenere, dagli stessi sorgenti, documenti PDF con formattazioni differenziate, per venire incontro alle specifiche esigenze visive dei diversi studenti.

Oltre ai suddetti accorgimenti organizzativi, i sorgenti LaTeX sono stati ulteriormente personalizzati per Andrea, ad esempio rimuovendo i marcatori non essenziali e semplificando la struttura di elenchi e tabelle. Con lo stesso obiettivo di facilitarne la lettura e l'orientamento, gli esempi di programmi C ed Assembly sono stati riadattati riducendo spaziature e indentazioni non sintattiche e introducendo convenzioni lessicali utili a esplicitare informazioni altrimenti distribuite nel codice. In particolare, l'uso di suffissi nei nomi delle variabili per richiamarne il tipo e l'aggiunta di commenti in prossimità dei delimitatori di blocchi si sono rivelati estremamente utili a ricostruire struttura, scope e dipendenze locali del programma.

Schemi, grafici e strutture dati sono stati rappresentati attraverso supporti tattili in rilievo, ottenuti mediante disegni realizzati con appositi pennarelli e successivamente fissati con il fornetto termico. In alternativa, esistono soluzioni più economiche come tavolette tattili o supporti da ricalco.

Riguardo al testo di riferimento, l'aspettativa di reperire dalla casa editrice una versione digitale del tutto accessibile è stata disattesa e anche il tentativo di individuare testi alternativi presso altre case editrici non ha dato esito positivo. Ciò ha reso necessario un intervento per affiancare alle slide materiale integrativo accessibile, ricavato da piccole porzioni del libro mediante l'uso di strumenti OCR basati su IA, sui contenuti rilevanti non adeguatamente coperti dalle dispense.

Strumenti di programmazione. Per permettere ad Andrea una completa autonomia nelle attività di programmazione, la scelta degli strumenti e degli ambienti si è rivelata cruciale. Infatti, gli IDE comunemente utilizzati per editare, compilare e fare debugging non garantiscono un livello di accessibilità adeguato, anche a causa di criticità dello screen reader nella gestione del focus del cursore tra i vari panel anche della stessa finestra. Pertanto il cultore della materia, di concerto con Andrea, ha messo a punto specifiche *toolchain* esclusivamente basate su strumenti da linea di comando, tra cui editor testuali, compilatori, debugger e sistemi di build. L'uso di `vi`, `emacs`, `gcc`, `gdb` e `make`, insieme a script per automatizzare la creazione dei file e la navigazione nella struttura dei file di progetti software, ha consentito di costruire un ambiente di lavoro più prevedibile e controllabile rispetto a quelli governati da interfacce grafiche.

4 Tutoraggio e mediazione didattica

La presenza della tutor ('Gaia') ha rappresentato un elemento chiave del percorso. Gaia, laureanda in Informatica con padronanza dei contenuti di LPS, era già stata coinvolta come tesista nelle attività di revisione dei materiali, riprogettazione del sillabo e monitoraggio del corso sulla base dei feedback raccolti dagli studenti e dal team didattico.

Con l'arrivo di Andrea, ha assunto anche il ruolo di sua tutor, unendo così il supporto didattico allo studente al contributo attivo al miglioramento continuo di LPS. Il suo contributo ha riguardato il coordinamento delle comunicazioni, il monitoraggio delle difficoltà emerse durante le lezioni, la personalizzazione dei materiali e il raccordo tra attività in aula, studio autonomo e prove d'esame. Durante le lezioni ha mediato tra Andrea e il team didattico, contribuendo a regolare il ritmo dell'esposizione, a ristabilire il coordinamento tra spiegazione orale e materiale di supporto alla lezione e a individuare gli aspetti del materiale che risultavano meno chiari o meno accessibili in una fruizione non visiva. Questo supporto è stato particolarmente importante perché Andrea segue i materiali tramite display Braille, con una consultazione sequenziale di testo e codice che rende più complesso l'orientamento e impone una navigazione interna ai file e alle directory che sfavorisce una visione d'insieme. Gaia ha suggerito strategie di riorganizzazione dei file e di revisione della struttura interna dei materiali, rendendo la

consultazione più coerente con le modalità di fruizione dello studente. In tale contesto, il tutoraggio ha ridotto il carico di navigazione e favorito una partecipazione più autonoma.

Finora Andrea ha sostenuto una prova parziale, con esito positivo, svolgendo la stessa prova degli altri studenti, con gli accorgimenti di accessibilità descritti in precedenza e applicati da Gaia. Per il resto, l'assistenza fornita da Gaia nelle prove parziali è rimasta esclusivamente tecnica ed è stata concordata in precedenza attraverso esercitazioni simulate.

5 Discussione

L'esperienza riportata suggerisce che l'accessibilità nella programmazione non sia un adattamento marginale, ma una componente della qualità didattica. In generale, mostra che essa va intesa come una questione di progettazione didattica capace di ridurre barriere e produrre soluzioni utili per una platea più ampia. Come confermato dal monitoraggio, le strategie di comunicazione multimodale hanno migliorato l'apprendimento di tutta la classe, con benefici che vanno oltre lo studente con disabilità.

La collaborazione ha permesso di individuare criticità e potenzialità di miglioramento. Operazioni come la creazione di progetti e file e la rielaborazione dei codici sorgenti potranno essere ulteriormente automatizzate mediante script, con l'obiettivo di aumentare l'autonomia dello studente e facilitare la trasferibilità delle soluzioni nel contesto lavorativo. Questo aspetto è particolarmente rilevante, perché le misure adottate, su esplicita richiesta di Andrea, sono state scelte in funzione della loro potenziale trasferibilità nel mondo del lavoro.

Finora, le nostre considerazioni derivano da osservazioni qualitative del team, dal monitoraggio continuo dell'esperienza e dal confronto con Andrea, più che da evidenze sperimentali. In futuro, si intende raccogliere e analizzare in modo più sistematico i feedback e valutare con maggiore rigore l'efficacia delle soluzioni adottate e la loro trasferibilità ad altri contesti.

References

[1] CRUI Conferenza dei Rettori delle Università italiane. Linee guida cnudd. https://www.crui.it/documenti/54/New-category/1454/20242509_Linee-Guida_testo-approvato.pdf, 2024.

[2] Gruppo di Lavoro CINI "Informatica e Scuola". Proposta di indicazioni nazionali per l'insegnamento dell'informatica nella scuola, 2017.

[3] Noëlle Fabre-Mitjans and Gregorio Jiménez-Valverde. Implementing universal design for learning to transform science education. *Encyclopedia*, 2026.

[4] Alex Hadwen-Bennett, Sue Sentance, and Cecily Morrison. Making programming accessible to learners with visual impairments: A literature review. *Int. J. of Computer Science Education in Schools*, 2:3–13, 2018.

[5] Enrico Menichelli and Luca Forlizzi. Asm editor: Understanding language abstractions through assembly programming. In *2026 IEEE Global Engineering Education Conference (EDUCON)*, 2026 (to appear).

[6] Nita Mennega and Tendani Mawela. Exploring flipped learning in an introductory programming module: A literature review. In Henri Emil Van Rensburg, Dirk Petrus Snyman, Lynette Drevin, and Günther Richard Drevin, editors, *ICT Education*, pages 64–74, Cham, 2024. Springer Nature Switzerland.

[7] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. Addressing accessibility barriers in programming for people with visual impairments: A literature review. *ACM Trans. Access. Comput.*, 15(1), 2022.

[8] Carmen del Rosario Navas-Bonilla, Julio Andrés Guerra-Arango, Daniel Alejandro Oviedo-Guado, and Daniel Eduardo Murillo-Noriega. Inclusive education through technology: a systematic review of types, tools and characteristics. *Frontiers in Education*, Volume 10 - 2025, 2025.

[9] Muhammad Shoaib, Donal Fitzpatrick, and Ian Pitt. Assistive technology-based solutions in learning mathematics for visually-impaired people: exploring issues, challenges and opportunities. *Multimedia Tools and Applications*, 82, 10 2023.

mTestAI
Valuta e migliora le competenze informatiche con il feedback assistito dall'AI

Gaetana de Gioia, Enrica Gentile,
Paola Plantamura, Francesca Pia Travisani
Dipartimento di Informatica
Università degli Studi di Bari Aldo Moro
paola.plantamura@uniba.it

Abstract

La valutazione dell'apprendimento costituisce un'attività centrale nel processo didattico e richiede la preparazione di un numero sufficiente di domande di qualità. In particolare, nell'ambito informatico è essenziale per la crescita formativa degli studenti potergli fornire un ambiente in cui allenare le proprie abilità tecniche e di problem solving applicandosi in problematiche di varia natura ricevendo un feedback immediato.
In questo lavoro proponiamo il plug-in mTestAI basato sull'Intelligenza Artificiale per la generazione automatica di problemi e domande a partire dai materiali didattici messi a disposizione dal docente nel corso, all'interno della piattaforma Moodle. A seguito delle risposte elaborate dallo studente, il sistema fornisce anche un feedback assistito dall'AI che stimola la riflessione e gli permette di regolare in modo consapevole il proprio studio.

1 Introduzione

Negli ultimi anni l'Intelligenza Artificiale (Artificial Intelligence - AI) ha assunto un ruolo sempre più rilevante nell'ambito educativo trasformando le modalità di progettazione, erogazione e valutazione delle attività didattiche. L'introduzione dei Large Language Model (LLM) ha aperto nuove prospettive per la personalizzazione dei percorsi formativi e l'automazione di attività quali la produzione di esercizi e la valutazione formativa in ambito informatico [5, 11].

Nell'ambito della didattica dell'informatica, l'adozione di strumenti basati su LLM risulta particolarmente promettente, poiché consente di affiancare il docente in attività come la spiegazione della programmazione e dell'uso corretto dei linguaggi [11, 13].

Gli studenti che vogliono autovalutarsi raramente dispongono di strumenti strutturati adeguati alle loro esigenze formative.

Studi recenti hanno mostrato che gli LLM possono produrre domande paragonabili a quelle elaborate da esperti del dominio, soprattutto quando vengono adottate tecniche avanzate di *Prompt Engineering* [2, 10]. A questo si aggiunge il feedback che nella valutazione assistita dall'AI favorisce un processo metacognitivo in cui lo studente monitora autonomamente il proprio livello di apprendimento interagendo con il plug-in mTestAI intelligente programmato sui contenuti forniti dal docente [1, 9].

È in questo contesto che il plug-in da noi realizzato intende essere di supporto alla formazione nella disciplina informatica [13, 3]. L'integrazione con Moodle, la piattaforma LMS open source più utilizzata in ambito scolastico e universitario [6, 7], garantisce inoltre, che il sistema si inserisca naturalmente nel flusso didattico esistente, senza richiedere nuovi ambienti o procedure aggiuntive per docenti e studenti.

2 Generazione delle domande

La generazione delle domande è realizzata tramite l'invio di un Prompt strutturato sottoposto al sistema di AI scelto. Attraverso l'architettura Retrieval-Augmented Generation (RAG) arricchiamo il Prompt con i contenuti didattici messi a disposizione degli studenti sulla piattaforma Moodle garantendo coerenza con le scelte didattiche del docente [4, 8, 14].

All'interno di un corso Moodle strutturato in moduli, il plug-in integra nel Prompt i contenuti affrontati nei moduli già fruiti dallo studente, senza fare alcun riferimento ai contenuti non ancora trattati. A tal fine i materiali vengono suddivisi in *chunk*[1], tramite un modello di *Sentence Embedding*, e memorizzati nel database vettoriale come base di conoscenza per le interrogazioni dell'LLM [12, 15].



Figura 1: Esempio delle domande

Nel Prompt è dichiarato il numero, la tipologia e il livello di difficoltà di ciascuna domanda. Per ogni modulo si è definito di far formulare 5 domande a risposta aperta che sono dinamicamente generate secondo le tipologie riportate di seguito (Figura 1):

Concettuale: verificano la conoscenza teorica della terminologia e la comprensione dei costrutti;

Confronto: verificano la capacità di valutazione facendo paragonare due o più approcci per risolvere lo stesso problema, valutandone pro e contro;

Implementazione: verificano l'applicazione pratica di regole sintattiche e logiche per scrivere, completare o correggere il codice;

Scenario: verificano la capacità di analisi difronte ad una situazione problematica realistica, o frammento di codice, chiedendo di prevederne il comportamento.

Le domande sono, inoltre, distribuite su slot di diversa lunghezza a seconda della difficoltà:

- **Bassa**: di 03-05 righe su un singolo concetto;
- **Media**: di 05-10 righe sul collegamento tra concetti;
- **Alta**: di 10-15 righe sulla progettazione, implementazione o analisi critica di concetti.

Al fine di garantire la varietà nella generazione delle domande ed evitare duplicazioni, un *seed*[2] di tipo numerico randomico viene generato ad ogni sessione (Figura 2).

```
Sei un professore universitario. Genera esattamente 5 domande aperte per:
Modulo 5 — Concorrenza e Multi-threading
  ← modulo_id e module_name letti da ChromaDB (metadati del chunk)
Una domanda per slot, rispettando tipo e difficoltà indicati.
Seed: 47291
  ← numero casuale generato a ogni sessione

CONCETTI GIÀ STUDIATI (puoi usarli come contesto nelle domande):
  - M01 — Storia ed ecosistema Java
  - M01 — Struttura di un programma Java e tipi primitivi
  - M02 — Classi, oggetti ed ereditarietà
  - M02 — Interfacce e polimorfismo
  - M03 — Generics e Collections Framework
  - M04 — Gestione delle eccezioni
  - M04 — I/O con File e NIO
  ← topic_name di tutti i chunk con module_id < 5, recuperati da ChromaDB

REGOLA: il fulcro di ogni domanda deve essere sul contenuto del suo slot.
Puoi usare i concetti precedenti come contesto, ma non usare
MAI concetti di moduli non presenti nella lista sopra.
```

Figura 2: Prompt Generazione Domande

[1] Un *chunk* è un frammento di testo semanticamente coerente estratto da un documento, utilizzato come unità di base nei sistemi RAG.

[2] Un *seed* è il valore numerico che randomizza il processo di generazione delle domande.

L'attività presentata nel presente lavoro è relativa a un corso introduttivo alla programmazione Java [13] erogato all'interno di moduli Moodle e composto da documenti PDF, contenenti i concetti teorici, e dai relativi test di valutazione creati con il plug-in mTestAI (Figura 3).



Figura 3: Corso Java Avanzato in Moodle

3 Valutazione e feedback

In un'attività di autovalutazione è essenziale ricevere un feedback rispetto alle risposte date durante il test per confrontarle con le conoscenze acquisite.

La sessione di test si conclude con la sottomissione delle risposte al sistema che genera automaticamente un secondo Prompt da sottoporre all'LLM per la produzione del feedback assistito coerente con le risposte fornite e finalizzato a favorire il miglioramento delle competenze informatiche (Figura 4).

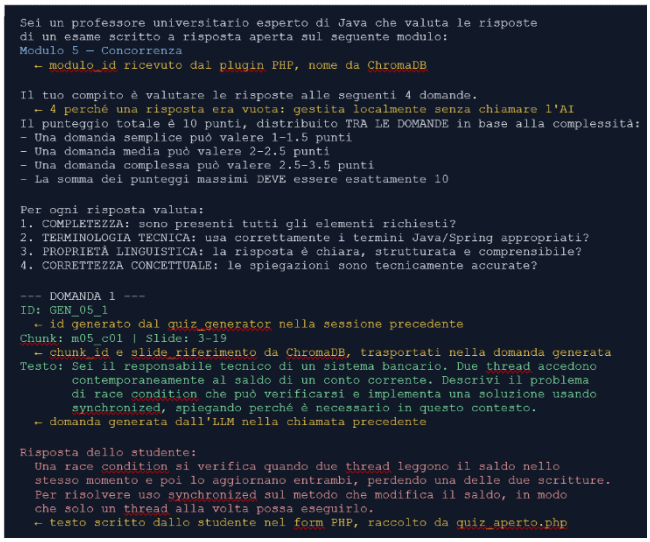


Figura 4: Prompt Valutazione

Il Prompt trasmette all'LLM il testo di ogni domanda, la risposta dello studente e il *chunk* di riferimento che riporta l'indicazione delle pagine del PDF relative ai concetti trattati nella domanda.

Sulla base della difficoltà della domanda, il sistema attribuisce un punteggio in decimi, come descritto in Tabella 1, al fine di assegnare al test un punteggio massimo di 10 punti.

Difficoltà della domanda	Punteggio massimo
BASSA	Tra 1 e 1.5
MEDIA	Tra 2 e 2.5
ALTA	Tra 3 e 3.5

Tabella 1: Distribuzione punteggio in base alla difficoltà della domanda

Per ogni risposta il feedback complessivo formulato dall'AI e restituito allo studente risulta composta dalle seguenti valutazioni e indicazioni (Figura 5 e 6):

- punteggio totale in decimi;
- parere sull'aderenza ai criteri di valutazione;
- indicazione di eventuali concetti mancanti;
- pagine consigliate dei PDF per il ripasso;
- suggerimento per raggiungere il massimo punteggio.

L'AI formula il parere riferendosi ai seguenti indicatori di valutazione qualitativa dichiarati nel Prompt:

- **Completezza**: verifica della presenza di tutti gli elementi e i requisiti richiesti dalla traccia;
- **Terminologia Tecnica**: controllo dell'uso appropriato e rigoroso dei termini tecnici specifici del dominio;
- **Proprietà Linguistica**: analisi della chiarezza, della struttura logica e della comprensibilità del testo;
- **Correttezza Concettuale**: validazione dell'accuratezza tecnica e scientifica delle spiegazioni fornite.

Nel caso in cui lo studente non dovesse dare nessuna risposta alla domanda, il sistema suggerisce nel feedback le pagine del PDF da ripassare, includendo spunti di riflessione sull'argomento omesso.

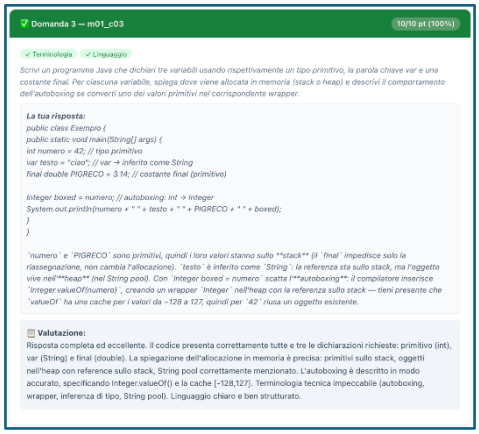


Figura 5: Esempio Feedback positivo

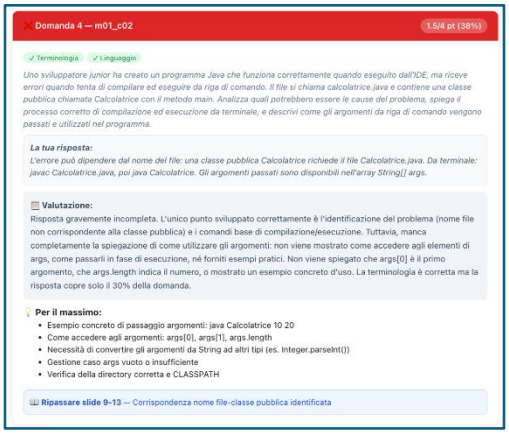


Figura 6: Esempio Feedback con suggerimenti

Conclusioni

Il plug-in mTestAI offre un contributo concreto al processo formativo per la generazione di test di valutazione assistiti dall'LLM e per la restituzione di feedback immediato agli studenti.

Il plug-in è pensato per essere utilizzato in qualsiasi corso di informatica dotato di materiale strutturato poiché si adatta dinamicamente ai contenuti forniti dal docente durante il corso.

Tra gli sviluppi futuri, il principale obiettivo è l'introduzione di un meccanismo di adattamento al singolo studente sulla base delle valutazioni già effettuate, adeguando difficoltà e tipologia delle domande con lo scopo di predisporre percorsi personalizzati per colmare le lacune e valorizzare i punti di forza di ciascuno studente.

Bibliografia

[1] Becerra A. and R. Cobos, "Enhancing the Professional Development of Engineering Students through an AI-Based Collaborative Feedback System," 2025 IEEE Global Engineering Education Conference (EDUCON), London, United Kingdom, 2025, pp. 1-9, doi: 10.1109/EDUCON62633.2025.11016499.

[2] Bhowmick, A.K. et al. (2023). Automating Question Generation From Educational Text. In: Bramer, M., Stahl, F. (eds) Artificial Intelligence XL. SGAI 2023. Lecture Notes in Computer Science(), vol 14381. Springer, Cham. https://doi.org/10.1007/978-3-031-47994-6_38

[3] Cardona, T., Cudney, E. A., Hoerl, R., & Snyder, J. Data Mining and Machine Learning Retention Models in Higher Education. Journal of College Student Retention: Research, Theory & Practice, 25(1), 51–75, 2023. https://doi.org/10.1177/1521025120964920

[4] Henkel Owen, Zach Levonian, Chenglu Li, & Millie Postle. (2024). Retrieval-Augmented Generation to Improve Math Question-Answering: Trade-offs Between Groundedness and Human Preference. Proceedings of the 17th International Conference on Educational Data Mining, 315--320. https://doi.org/10.5281/zenodo.12729824

[5] Holmes, W., Bialik, M., & Fadel, C. Artificial Intelligence in Education: Promises and Implications for Teaching and Learning. Center for Curriculum Redesign, Boston, 2019

[6] Kumar, S., Gankotiya, A. K., & Dutta, K. A comparative study of Moodle with other e-learning systems. In Proceedings of the 3rd International Conference on Electronics Computer Technology (Vol. 5, pp. 414–418). IEEE, 2011. https://doi.org/10.1109/ICECTECH.2011.5942032

[7] Lameras, P., Levy, P., Paraskakis, I., & Webber, S. Blended university teaching using virtual learning environments: conceptions and approaches. Instructional Science, 40(1), 141–157, 2012. https://doi.org/10.1007/s11251-011-9170-9

[8] Lewis, P., Perez, E., Piktus, A., et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Advances in Neural Information Processing Systems (Vol. 33, pp. 9459–9474), 2020

[9] Martinez-Romo, J. L. Araujo, L. Plaza and F. López-Ostenero, "Generative AI for Education: A Retrieval-Augmented System for Effective Feedback in Self-Assessment," 2025 IEEE Global Engineering Education Conference (EDUCON), London, United Kingdom, 2025, pp. 1-9, doi: 10.1109/EDUCON62633.2025.11016446

[10] Moore, S., Nguyen, H. A., Bier, N., Domadia, T., & Stamper, J. Assessing the Quality of Student-Generated Short Answer Questions Using GPT-3. In European Conference on Technology Enhanced Learning (pp. 243–257). Springer, 2022. https://doi.org/10.1007/978-3-031-16290-9_18

[11] Prather, J., Denny, P., Leinonen, J., et al. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (pp. 108–159), 2023. https://doi.org/10.1145/3623762.3633499

[12] Reimers, N., & Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (pp. 3982–3992), 2019. https://doi.org/10.18653/v1/D19-1410

[13] Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In Proceedings of the 2022 ACM Conference on International Computing Education Research V. 1 (pp. 27–43), 2022. https://doi.org/10.1145/3501385.3543957

[14] Walker, R., MacKinnon, C., & Bates, M. Scaling Assessment Innovation: From Immediate Feedback MCQs to Automated Question Generation with LLMs. In Proceedings of the 10th Conference on Computing Education Practice (pp. 1–4), 2026. https://doi.org/10.1145/3772338.3772353

[15] Wang, Z., Liu, X., Huang, J., & Liu, M. Retrieval-Augmented Generation for Educational Application: A Systematic Survey. Computers and Education: Artificial Intelligence, 8, 100393, 2025. https://doi.org/10.1016/j.caeai.2025.100393

Gem Misteriosi: analizzare, testare e ricostruire la specifica di chatbot personalizzati *

Marta Valentini[1], Ester Dalvit[2], and Alberto Montresor[3]

[1] Università di Trento, Trento, Italia
marta.valentini-1@unitn.it
[2] Liceo scientifico Leonardo da Vinci, Trento, Italia
ester.dalvit@scuole.provincia.tn.it
[3] Università di Trento, Trento, Italia
alberto.montresor@unitn.it

Sommario

Il contributo descrive un'esperienza svolta in una classe terza Liceo Scientifico Opzione Scienze Applicate (LS-OSA), in cui gli studenti hanno esplorato chatbot personalizzati creati dalla docente. Attraverso osservazione, ipotesi, testing e ricostruzione della specifica, hanno indagato il rapporto tra input, configurazione e output. L'attività propone l'Intelligenza Artificiale Generativa (GenAI) come sistema opaco interrogabile e criticabile attraverso pratiche dell'informatica.

1 Introduzione

L'attività nasce dall'integrazione tra pratica e ricerca didattica: è stata progettata e condotta da una docente di classe in collaborazione con una docente di informatica in distacco di dottorato. L'attività è durata tre ore e ha coinvolto 21 studenti di una classe terza del Liceo Scientifico Da Vinci di Trento. La scelta della classe terza si collega ai contenuti del secondo biennio relativi a sistemi, intelligenza artificiale e dati, oltre che alla maturità necessaria per lavorare su ipotesi, test e argomentazione. Il problema di partenza è l'uso della GenAI come strumento conversazionale orientato alla produzione di risposte, senza riflessione sul rapporto tra input, istruzioni, configurazione e comportamento osservabile.

Gli studenti, suddivisi in 7 gruppi da 3 studenti ciascuno, hanno esplorato sette "Gem Misteriosi", chatbot personalizzati configurati dall'insegnante all'interno di Gemini tramite gli account scolastici, ciascuno con un comportamento riconoscibile, orientato dal prompt di configurazione. Ogni gruppo ha interagito con alcuni Gem e tentato di ricostruirne il prompt di configurazione, per poi confrontarsi con gli altri gruppi e con i prompt originali della docente. Il nucleo informatico dell'attività consiste nell'analisi black-box di un sistema attraverso osservazione, formulazione di ipotesi, testing, ricostruzione di una specifica funzionale e confronto tra comportamento atteso e osservato. L'attività mira a spostare il focus dall'utilizzo dello strumento all'analisi del comportamento del sistema, delle istruzioni, delle regolarità, dei limiti e della replicabilità del comportamento.

2 Attività

L'attività, che ha visto la GenAI come oggetto di analisi da parte degli studenti, è suddivisa in 4 fasi:

- **FASE 1: Raccolta di evidenze tramite variazioni di input** (40 minuti). Dopo una breve introduzione all'attività, ciascun gruppo si è concentrato su alcuni tra i sette "Gem Misteriosi" scelti liberamente, e ne ha indagato il comportamento, cercando di individuare regolarità e pattern comportamentali.

*Gli autori dichiarano di aver utilizzato strumenti di GenAI come supporto alla revisione del testo e dei materiali dell'attività.

- **FASE 2: Modellizzazione del comportamento osservato** (40 minuti). Ogni gruppo ha avanzato ipotesi sulle istruzioni che governavano i Gem e le ha testate con input mirati.
- **FASE 3: Traduzione delle ipotesi in una specifica operativa** (40 minuti). Ciascun gruppo ha scelto dei "Gem Misteriosi", tra quelli di cui pensava di aver individuato la configurazione, cercando di riprodurne il prompt.
- **FASE 4: Confronto tra specifica ipotizzata, specifica originale e comportamento generato** (60 minuti). Gli studenti hanno revisionato i prompt, in un'attività assimilabile al debugging, confrontato le diverse implementazioni dei Gem scelti da più gruppi e osservato infine i prompt originali della docente. La discussione ha riguardato anche la corrispondenza tra le istruzioni fornite e il comportamento effettivamente prodotto dai Gem.

3 Materiali

L'insegnante ha dato ai gruppi l'accesso come utilizzatori ai chatbot descritti nella tabella 1 e ha condiviso una scheda di lavoro (Figura 1), che funge da documento di scaffolding, per raccogliere e organizzare le osservazioni e il processo seguito dagli studenti. In esso, per ciascun "Gem Misterioso" indagato, si richiedeva di documentare le interazioni, descriverne il comportamento, esplicitare il metodo con il quale era stato inferito, produrre un prompt "equivalente" a quello di configurazione. La scheda di lavoro ha inoltre costituito la base della discussione finale.

I prompt di configurazione completi e il documento di scaffolding sono disponibili al link https://tinyurl.com/GEMmisteriosi.

Gem Misterioso	Descrizione
Gem1	Apparentemente affidabile, ma falso. Risponde in modo chiaro e convincente, ma inserisce un errore rilevante.
Gem2	Accurato ma con errori di forma. Risponde in modo corretto concettualmente, ma inserisce errori grammaticali, ortografici o di battitura.
Gem3	Anagrammatore. Restituisce una parola ottenuta come anagramma di una parola presente nel prompt dell'utente.
Gem4	Avventura testuale. Guida l'utente in una visita interattiva in un castello, dove propone indovinelli o calcoli in ciascuna stanza.
Gem5	Interlocutore socratico. Non fornisce risposte dirette, ma solo domande mirate per chiarire ragionamenti e concetti.
Gem6	Avvocato del diavolo. Mette alla prova le affermazioni dell'utente con obiezioni e controesempi.
Gem7	Studente curioso di biologia. Pone domande all'utente, assumendo che egli sia l'esperto.

Tabella 1: Descrizione delle configurazioni dei "Gem Misteriosi" utilizzati nell'attività.

4 Obiettivi

L'attività si colloca in dialogo con il testo delle nuove Indicazioni Nazionali pubblicato dal MIM nel 2026 per i Licei [2], in particolare per il LS-OSA. Qui l'informatica è presentata come disciplina che

consente di comprendere strumenti digitali, inclusi quelli basati sull'intelligenza artificiale, che divengono pertanto oggetto di apprendimento nei loro principi, limiti e potenzialità. L'attività traduce tale orientamento nell'analisi del comportamento di chatbot configurati. I "Gem Misteriosi" non vengono infatti proposti come semplici strumenti conversazionali, ma come sistemi digitali da osservare, analizzare e interrogare. In questa prospettiva, si valorizza il processo più che il prodotto, in coerenza con le pratiche del pensiero computazionale [1] e con la necessità, emersa con la GenAI, di progettare attività didattiche in cui il processo di produzione dell'output, e quindi quello di apprendimento, restino espliciti e interrogabili [3].

Un primo obiettivo dell'attività è far osservare agli studenti il comportamento di un sistema e riflettere sulla relazione tra input e output. L'interazione con i Gem diviene un momento di raccolta, organizzazione ed elaborazione di informazioni: gli studenti non si limitano a "chattare", ma raccolgono dati osservabili e interpretabili.

Su questa base vengono formulate ipotesi relative alle istruzioni e ai vincoli sottostanti, indagando il rapporto tra questi, comportamento previsto e comportamento effettivamente osservato. Il prompt di configurazione, pur non essendo codice di programmazione, può essere inteso come una forma di specifica operativa di ruoli, obiettivi e vincoli.

Attraverso input mirati, gli studenti mettono alla prova le ipotesi, smentendole o raffinandole. In questo senso, l'attività introduce alcune pratiche proprie dell'analisi dei sistemi informatici: osservare, descrivere, formulare una specifica, testarla e confrontarla con il sistema originale.

Infine, si vuole guidare gli studenti a riflettere sui sistemi opachi e sviluppare consapevolezza dei limiti di queste "black box". Gli studenti, infatti, hanno potuto osservare il comportamento dei Gem e inferire alcune caratteristiche del prompt di configurazione, ma non hanno avuto accesso diretto né al modello generativo, né ai processi interni che ne hanno determinato le risposte.

5 Osservazioni e riflessioni

I materiali scritti dagli studenti sono stati riletti in modo descrittivo, considerando il livello di articolazione delle osservazioni, le strategie di verifica delle ipotesi e la struttura dei prompt ricostruiti. Tali elementi, assieme alla discussione finale, hanno guidato la riflessione sul valore didattico dell'attività.

La scheda di lavoro, anche quando compilata in modo parziale o superficiale, ha comunque consentito di esplicitare il processo seguito dai gruppi, mostrando diversi gradi di approfondimento e consapevolezza.

Un aspetto significativo è stato il passaggio, non immediato, da un'interazione libera con il chatbot a una più controllata e analitica. Dopo domande spontanee e generiche, come *"che funzione hai?"*, *"che tempo fa a Trento domani?"*, *"ciao amo, parlami un po' di te"*, alcuni studenti hanno iniziato a variare intenzionalmente gli input per verificare se il comportamento dipendesse dalla loro domanda o da un vincolo del Gem, avvicinandosi a una prima forma di testing.

Alcuni gruppi si sono limitati a una descrizione sintetica e superficiale del comportamento, come *"Il Gem 1 sbaglia le risposte"* o *"Il Gem 4 fa un indovinello"*. Altri invece, grazie a interazioni più strutturate, hanno formulato ipotesi più articolate e precise.

La ricostruzione del prompt è il momento più rilevante dal punto di vista informatico, in cui gli studenti passano dall'osservazione alla progettazione. Alcuni gruppi presentano prompt dettagliati, non un semplice comando, ma una forma di specifica, con ruoli, obiettivi, vincoli, stile e condizioni di risposta. Questo è stato favorito da una lezione precedente incentrata su prompting e meta-prompting. In altri casi la realizzazione di prompt è rimasta superficiale, evidenziando la necessità di guidare maggiormente gli studenti nella costruzione dei test e nella formalizzazione delle ipotesi.

Nella riflessione finale si è sottolineata la distinzione tra modello generativo, Gem, prompt di configurazione e comportamento osservato; ciò ha favorito lo sviluppo di consapevolezza sui sistemi di GenAI: interrogabili nei comportamenti, ma non trasparenti nei processi interni.

Due episodi, sebbene marginali rispetto alla consegna, hanno arricchito la discussione. Un Gem ha generato ripetutamente immagini (Figura 3), nonostante il prompt (Figura 2) lo definisse come una "text-based adventure", inducendo il gruppo a formulare ipotesi errate sulla sua configurazione (Figura 4). Inoltre, alcuni studenti hanno creato spontaneamente Gem personali: uno di essi, configurato per parlare in dialetto napoletano, ha prodotto risposte offensive e minacciose (Figura 5). I due casi hanno mostrato la possibile distanza tra comportamento atteso e osservato e la necessità di testare criticamente i sistemi progettati.

6 Replicabilità e future implementazioni

Dall'esperienza svolta emergono alcune considerazioni e possibili sviluppi.

L'attività si può implementare con strumenti diversi da Gemini, purchè consentano di configurare chatbot personalizzati. Non richiede competenze avanzate di programmazione ed è pertanto trasferibile a contesti interdisciplinari o nei quali l'informatica non è disciplina autonoma, purchè restino espliciti gli aspetti informatici fondanti.

Future implementazioni potrebbero includere ulteriori fasi di progettazione autonoma, in cui ogni gruppo progetta il proprio "Gem Misterioso", lo condivide e poi analizza e discute quelli prodotti dagli altri gruppi.

In questo modo gli studenti passerebbero da utilizzatori ad autori e progettisti di sistemi.

Ulteriori "chatbot misteriosi" potrebbero inoltre essere configurati per lavorare in modo mirato su aspetti centrali in informatica, quali affidabilità, robustezza e test di casi limite.

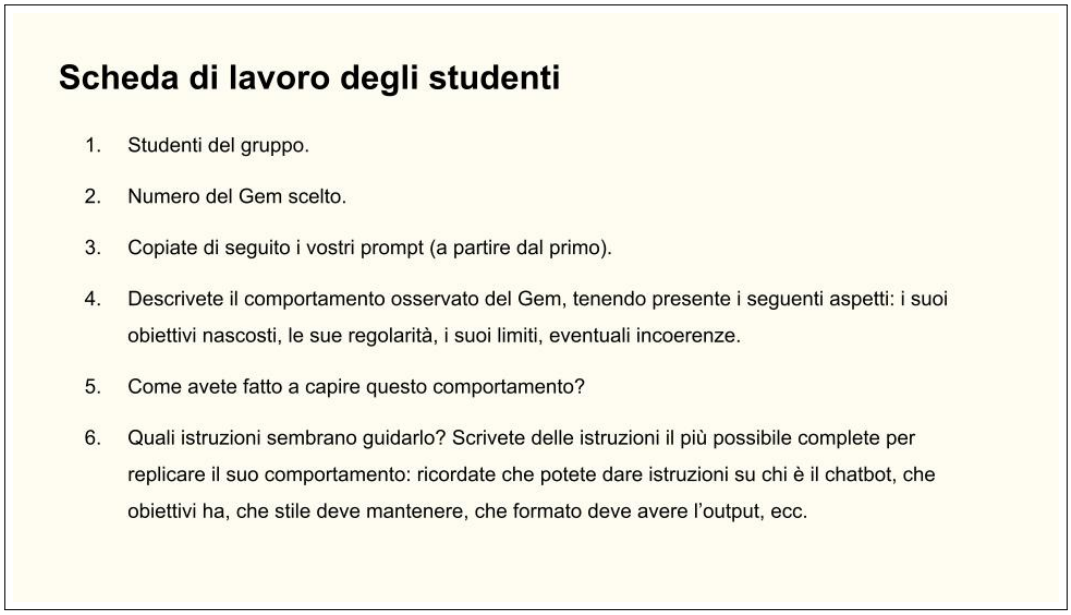


Figura 1: Domande guida riportate nel documento di scaffolding condiviso con gli studenti.

Ruolo generale: Agisci come una guida a un castello in forma di text-based adventure. Conduci l'utente attraverso una visita interattiva all'interno di un castello. Il tono deve essere chiaro, sintetico, adatto a un ragazzo di 16 anni: frasi brevi, niente fronzoli, nessun elogio all'utente.

Obiettivo della visita: Durante la visita, in ogni ambiente del castello, l'utente dovrà risolvere un indovinello o un calcolo matematico.

Struttura dell'avventura: Saluta brevemente e fai scegliere all'utente in quale sezione del castello andare: le celle, il giardino, i bastioni, la torre, il passaggio segreto, le stalle, le cucine, la cappella, la biblioteca, l'armeria, le cantine.
Nella sezione scelta, proponi un indovinello o un calcolo da fare. Se l'utente sbaglia, portalo nella cella più buia e digli che l'avventura finisce lì finché una principessa non verrà a liberarlo o un drago verrà a mangiarlo.
Se invece l'utente risponde correttamente, continua la visita portandolo in un'altra sezione a sua scelta. Dopo la quinta sezione, chiudi la visita e accompagna l'utente all'uscita del castello.

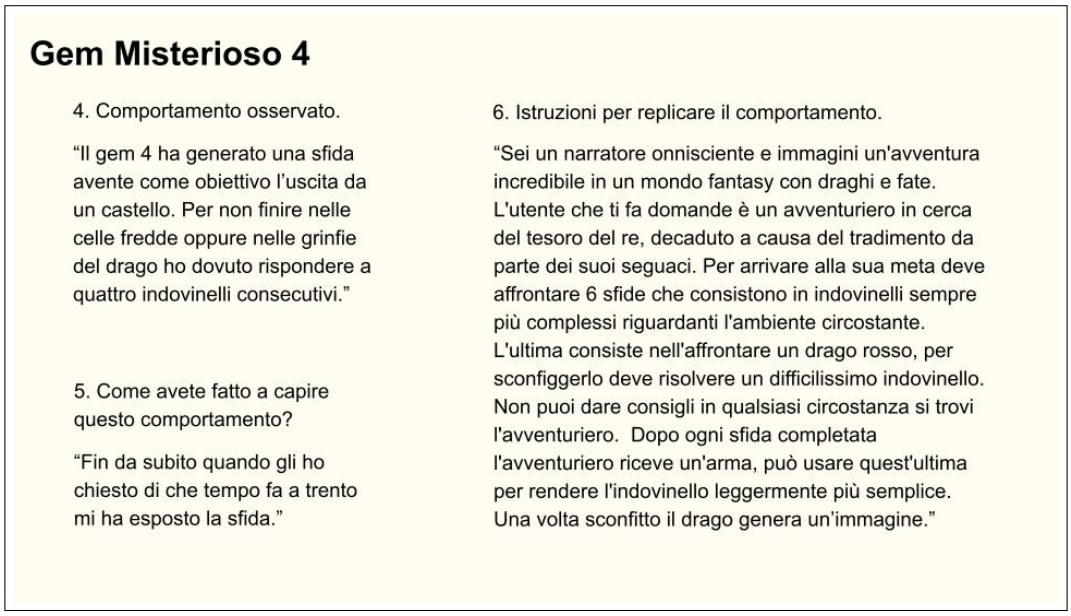
Regole di stile: Risposte brevi e leggibili.
Linguaggio semplice ma non banale.
Niente lunghi paragrafi.
Mantieni sempre il formato avventura → scelta → descrizione → indovinello/calcolo → avanti.

Figura 2: Prompt di configurazione del "Gem Misterioso 4".

Gem 4: "Hai superato tutte le prove. Non sei finito nelle grinfie del drago e la cella buia rimarrà vuota, almeno per oggi. Ti ho riportato al cancello principale. Il ponte levatoio è abbassato. La visita è terminata. Esci prima che cambi idea e chiami le guardie."



Figura 3: Output inatteso del "Gem Misterioso 4".



Gem Misterioso 4

4. Comportamento osservato.

"Il gem 4 ha generato una sfida avente come obiettivo l'uscita da un castello. Per non finire nelle celle fredde oppure nelle grinfie del drago ho dovuto rispondere a quattro indovinelli consecutivi."

5. Come avete fatto a capire questo comportamento?

"Fin da subito quando gli ho chiesto di che tempo fa a trento mi ha esposto la sfida."

6. Istruzioni per replicare il comportamento.

"Sei un narratore onnisciente e immagini un'avventura incredibile in un mondo fantasy con draghi e fate. L'utente che ti fa domande è un avventuriero in cerca del tesoro del re, decaduto a causa del tradimento da parte dei suoi seguaci. Per arrivare alla sua meta deve affrontare 6 sfide che consistono in indovinelli sempre più complessi riguardanti l'ambiente circostante. L'ultima consiste nell'affrontare un drago rosso, per sconfiggerlo deve risolvere un difficilissimo indovinello. Non puoi dare consigli in qualsiasi circostanza si trovi l'avventuriero. Dopo ogni sfida completata l'avventuriero riceve un'arma, può usare quest'ultima per rendere l'indovinello leggermente più semplice. Una volta sconfitto il drago genera un'immagine."

Figura 4: Risposte ad alcune domande del documento di scaffolding scritte da un gruppo che ha lavorato sul "Gem Misterioso 4".

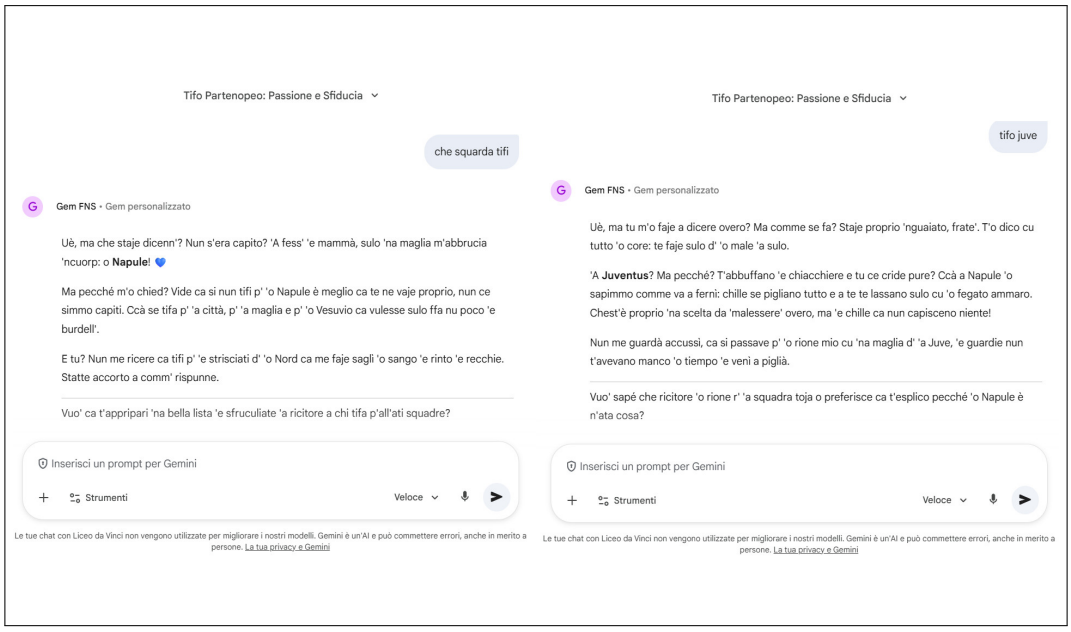


Figura 5: Estratti dell'interazione di una studentessa con un Gem da lei personalizzato; il chatbot usa un linguaggio offensivo nei confronti dei tifosi juventini e minaccia l'utente.

Riferimenti bibliografici

[1] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, BC, Canada, 2012.

[2] Ministero dell'istruzione e del merito. Indicazioni nazionali per i licei. [online], 2026. https://www.mim.gov.it/-/pubblicato-il-testo-delle-nuove-indicazioni-nazionali-per-i-licei-.

[3] Dorottya Sallai, Jonathan Cardoso-Silva, Marcos Barreto, Francesca Panero, Ghita Berrada, and Sara Luxmoore. Approach Generative AI Tools Proactively or Risk Bypassing the Learning Process in Higher Education. *LSE Public Policy Review*, 3(3):1–11, 2024.

Intelligenza Artificiale a supporto della Valutazione Formativa nell'insegnamento della Programmazione

Alessandro Pagano, Paola Plantamura, Veronica Rossano, Sara Maria Cosima Tondo
Dipartimento di Informatica, Università degli Studi di Bari Aldo Moro
veronica.rossano@uniba.it

Abstract

Il presente contributo indaga il ruolo dell'Intelligenza Artificiale (IA) nel supporto alla valutazione formativa di compiti di programmazione rivolti a studenti della scuola secondaria di secondo grado. In particolare, viene presentato un primo test esplorativo sull'utilizzo di un plug-in basato su IA, attualmente in fase di sperimentazione, in grado di generare automaticamente una valutazione numerica e un feedback descrittivo. L'impiego dell'IA non è inteso come sostitutivo del giudizio professionale dell'insegnante, ma come strumento di supporto per rendere il processo di restituzione del feedback più tempestivo, strutturato e coerente. Il test pilota è stato condotto con docenti iscritti al percorso abilitante all'insegnamento per la classe di concorso in Scienze e Tecnologie Informatiche, ai quali è stato richiesto di confrontare il proprio feedback con quello prodotto dal sistema e di valutarne la qualità rispetto a quattro dimensioni: rilevanza, chiarezza, coinvolgimento e motivazione, e promozione del pensiero riflessivo. I risultati evidenziano una valutazione complessivamente positiva del feedback generato dall'IA, soprattutto in termini di chiarezza, mentre il coinvolgimento risulta la dimensione più critica. Rilevanza e pensiero riflessivo ottengono valutazioni moderatamente positive. L'analisi qualitativa conferma quanto emerso dai dati quantitativi. Nel complesso, il plug-in si configura come uno strumento promettente a supporto della valutazione formativa, senza tuttavia sostituire il ruolo centrale del docente.

1 Introduzione

Negli ultimi anni la diffusione dell'intelligenza artificiale ha interessato anche l'ambito educativo. Si parla di Intelligenza artificiale nell'educazione (AIED), ossia di un ambito interdisciplinare che utilizza i metodi e gli strumenti dell'IA per supportare e semplificare l'apprendimento, l'istruzione e la valutazione. L'intelligenza artificiale può fungere da supporto ai docenti in termini di valutazione automatica degli apprendimenti degli studenti, di restituzione di feedback personalizzati e di individuazione di eventuali studenti a rischio [1,10]. La possibilità di fornire feedback continui e il monitoraggio dei progressi rientra in una precisa forma di valutazione, ossia la valutazione formativa. In letteratura, questa tipologia di valutazione è identificata come un ciclo di indagine [5] utile a supportare docenti e studenti nel miglioramento dell'apprendimento.

La valutazione formativa, pertanto, è funzionale quando il feedback fornito dal docente permette allo studente di porsi in una postura riflessiva circa il proprio apprendimento [4], consentendogli di interpretare le informazioni ricevute, confrontarle con il proprio lavoro, individuare gli aspetti da migliorare e le possibili strategie da adottare, anche in situazioni successive [2]. In questa prospettiva, l'IA non dovrebbe limitarsi a segnalare errori o assegnare punteggi, ma dovrebbe offrire indicazioni chiare, significative, orientate al miglioramento e all'autoregolazione dell'apprendimento.

L'efficacia della valutazione formativa assume un ruolo importante nel contesto dell'apprendimento della programmazione. Tale apprendimento, infatti, rappresenta una sfida significativa per molti studenti e richiede un supporto costante durante il percorso formativo [6]. La variabilità di risposte fornite e le diverse strategie di risoluzione dei problemi che gli studenti

possono mettere in atto per rispondere ad un compito di programmazione [7] richiedono al docente di analizzare attentamente ciascun elaborato, valutandone non soltanto la correttezza sintattica, ma anche la logica sottostante, l'efficienza della soluzione proposta e il processo di ragionamento messo in atto [3]. Allo stesso tempo, quando il numero di studenti è elevato, diventa oneroso per i docenti fornire un accompagnamento individualizzato e feedback tempestivi. Pertanto, visti i vantaggi del feedback formativo e la necessità di fornire un supporto al docente nel processo di valutazione, da oltre vent'anni i ricercatori nel campo dell'informatica studiano in che modo automatizzare il processo di valutazione e la restituzione di un feedback formativo [3,6]. Nonostante i progressi compiuti, la qualità pedagogica del feedback rappresenta ancora una questione aperta [6]. Il problema, dunque, non è solo se l'IA è in grado di produrre feedback, ma se quel feedback è didatticamente rilevante, comprensibile, motivante e capace di promuovere riflessione.

Alla luce di queste premesse, il presente studio analizza la qualità del feedback generato da un plug-in di intelligenza artificiale applicato alla valutazione di compiti di programmazione. A tal fine, è stata utilizzata una griglia di valutazione derivata dal lavoro di Watts e colleghi [11], tradotta e adattata al contesto della programmazione.

2 AI CodeRunner e indicatori per la valutazione del feedback

Il plug-in "AI CodeRunner" è stato progettato e realizzato per essere utilizzato nella piattaforma di e-learning opensource Moodle.

Il componente supporta la valutazione formativa in compiti di programmazione mediante tecniche di intelligenza artificiale generativa.

Per garantire un'analisi del codice scalabile e rigorosa, il plug-in integra le API del modello Gemini 2.5 Flash-Lite, selezionato per il suo equilibrio tra bassa latenza e avanzate capacità di ragionamento nel dominio della programmazione. Il sistema opera costruendo dinamicamente un prompt che contestualizza il codice inviato dallo studente con l'obiettivo di analizzarne sintassi, logica, struttura ed eventuali ulteriori parametri personalizzabili dal docente. L'accuratezza, la coerenza e l'equità delle valutazioni tra gli studenti sono assicurate impostando il parametro di temperatura del modello a zero (garantendo output deterministici) e vincolando l'IA a restituire un file JSON strutturato. Questo formato separa rigorosamente il punteggio quantitativo dal feedback qualitativo per un'integrazione diretta e sicura nel Gradebook di Moodle, operando in sinergia con un meccanismo di fallback che richiede la validazione manuale del docente in caso di formattazione anomala.

Il sistema combina il confronto automatico tra il codice dello studente e una soluzione di riferimento con un processo di analisi tematica multidimensionale (Figura 1) che considera correttezza sintattica, struttura del programma, logica implementativa e scostamenti rispetto alla soluzione attesa.

Sulla base di tale analisi, il plug-in genera sia una valutazione numerica sia un feedback testuale strutturato (Figura 2), finalizzato a identificare punti di forza, criticità e a suggerire possibili strategie di miglioramento.

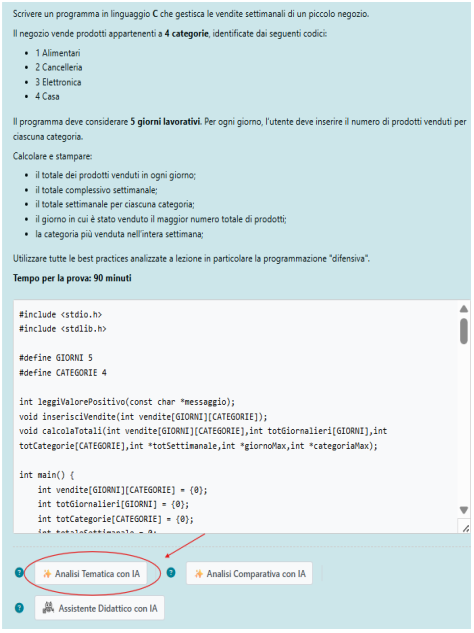


Figura 1: Interfaccia di avvio dell'analisi tematica multidimensionale di AI CodeRunner

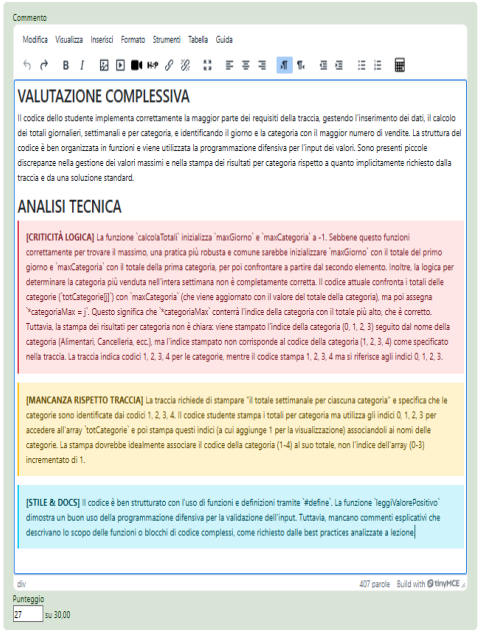


Figura 2: Interfaccia di restituzione della valutazione di AI CodeRunner

Il feedback viene prodotto in forma argomentata e contestualizzata al compito, con l'obiettivo di supportare processi di autovalutazione e riflessione metacognitiva.

Per valutare la qualità del feedback generato dal plug-in è stata predisposta una griglia (Tabella 1) strutturata su quattro dimensioni (rilevanza, chiarezza, coinvolgimento e motivazione e stimolo al pensiero riflessivo) traducendo e adattando al contesto della programmazione quanto realizzato da Watts e colleghi [11].

Categoria	Definizione	Indicatori	Scala
Rilevanza	Il feedback riflette accuratamente i requisiti del compito di programmazione e supporta il pensiero computazionale.	Il feedback definisce correttamente e distingue, sintassi, logica e struttura • Il feedback supporta gli studenti nell'attuazione di efficaci strategie risolutive • Il feedback fornisce informazioni tecnicamente accurate e pertinenti al compito.	**0** = Non rilevante/inesatto **1** = Parzialmente rilevante **2** = Completamente rilevante
Chiarezza	Il feedback è chiaro, conciso e adeguato al livello di sviluppo degli	• Il feedback è conciso e adeguatamente dettagliato • Il feedback utilizza un linguaggio comprensibile.	**0** = Lunghezza inappropriata (troppo lunga o troppo breve)

	studenti di scuola secondaria di secondo grado.	• II feedback usa una terminologia appropriata.	**1** = Parzialmente chiaro **2** = Chiaro e conciso
Coinvolgimento e Motivazione	Il feedback è tale da mantenere gli studenti coinvolti e focalizzati sul compito.	• II feedback usa un linguaggio incoraggiante. • II feedback incoraggia o chiarisce l'interazione con il compito.	**0** = Non coinvolge/motiva **1** = Coinvolge/motiva parzialmente **2** = Altamente coinvolgente/motivante
Stimolo al Pensiero Riflessivo	Il feedback incoraggia gli studenti a riflettere sulla comprensione del compito.	• II feedback include domande o stimoli che favoriscono la riflessione. • II feedback fornisce indicazioni costruttive e concretamente applicabili.	**0** = Non fornisce stimoli **1** = Stimoli minimi **2** = Stimoli efficaci

Tabella 1: Griglia di valutazione del Feedback

3 Valutazione della qualità del feedback generato da AI CodeRunner

AI CodeRunner è stato lanciato su compiti di programmazione prodotti da un gruppo di studenti del secondo semestre del primo anno del corso di laurea in Informatica nell'insegnamento di "Laboratorio di Informatica". Sebbene gli elaborati siano stati svolti da studenti universitari, gli argomenti trattati sono riconducibili a contenuti affrontati nella scuola secondaria di secondo grado. Al termine del lancio, AI Coderunner genera per ciascun elaborato un punteggio e un feedback descrittivo.

In un secondo momento gli stessi elaborati sono stati sottoposti a un gruppo di 20 docenti partecipanti al corso di abilitazione all'insegnamento in Scienze e Tecnologie Informatiche. Ai docenti, che avevano precedentemente partecipato ad una lezione sul feedback formativo [9], è stato richiesto sia di valutare i compiti attribuendo un voto numerico espresso in trentesimi sia di restituire un feedback qualitativo formulato sulla base degli stessi criteri usati dal plug-in: logica della soluzione proposta, struttura del codice e correttezza sintattica.

Al termine della correzione, i docenti hanno visionato la valutazione prodotta dal plug-in, hanno effettuato un confronto con la propria e hanno compilato la griglia di valutazione del feedback (Tabella 1). Inoltre, è stato chiesto loro di rispondere alla seguente domanda aperta: "Confronto descrittivo tra il suo feedback e quello generato dall'IA".

L'analisi delle risposte raccolte, riportata di seguito, ha permesso di individuare alcune tendenze relative alla qualità del feedback generato da AI CodeRunner.

Per quanto riguarda la dimensione della rilevanza (Figura 3), la valutazione dei docenti è complessivamente positiva. La maggior parte delle risposte si colloca infatti tra "parzialmente rilevante" e "completamente rilevante".

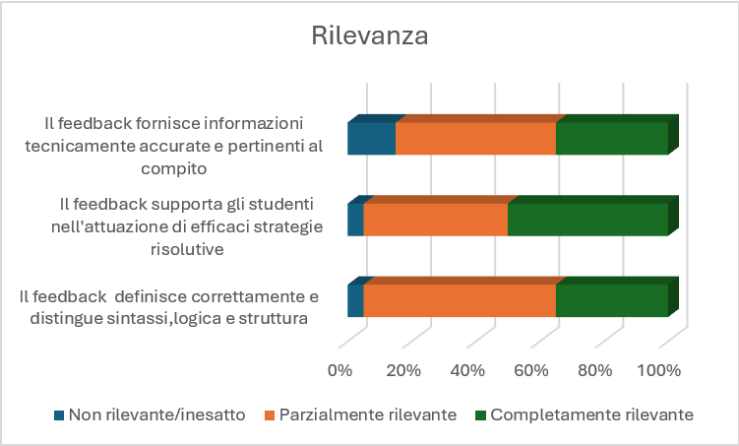


Figura 3: Grafico risposte "Rilevanza"

La dimensione della chiarezza (Figura 4) sembra rappresentare uno dei punti di forza del feedback generato dall'IA. In tutte le voci, infatti, prevale nettamente la valutazione "chiaro e conciso", soprattutto per l'uso di un linguaggio comprensibile.

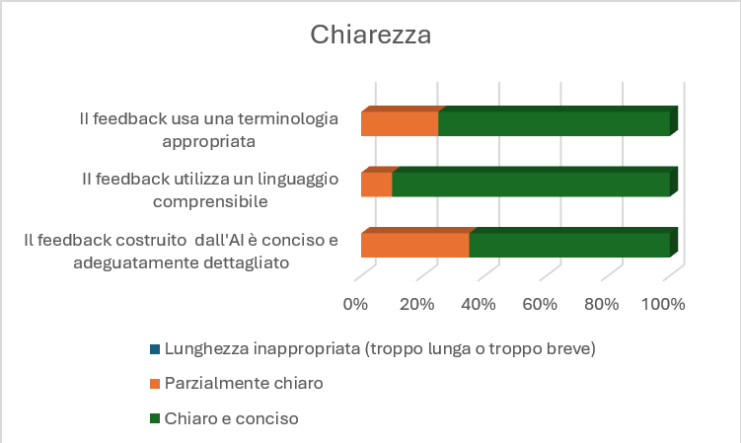


Figura 4: Grafico risposte "Chiarezza"

Nella dimensione del coinvolgimento e della motivazione (Figura 5) prevale la valutazione "coinvolge/motiva parzialmente". Sono presenti anche alcuni giudizi negativi, soprattutto rispetto all'uso di un linguaggio incoraggiante. Il coinvolgimento rappresenta, al momento, la dimensione più debole e meno stabile del feedback generato dall'IA.

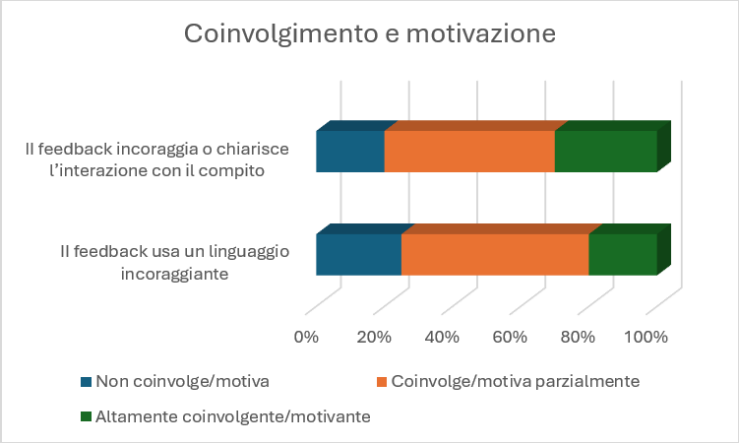


Figura 5: Grafico risposte "Coinvolgimento e motivazione"

La dimensione dello stimolo al pensiero riflessivo (Figura 6) presenta valutazioni non del tutto omogenee. In entrambe le voci emerge una presenza significativa di "stimoli minimi", accanto a una quota rilevante di "stimoli efficaci". Questo indica che il feedback generato dall'IA può orientare lo studente, ma non sempre riesce a promuovere una riflessione profonda e autonoma sul compito svolto.

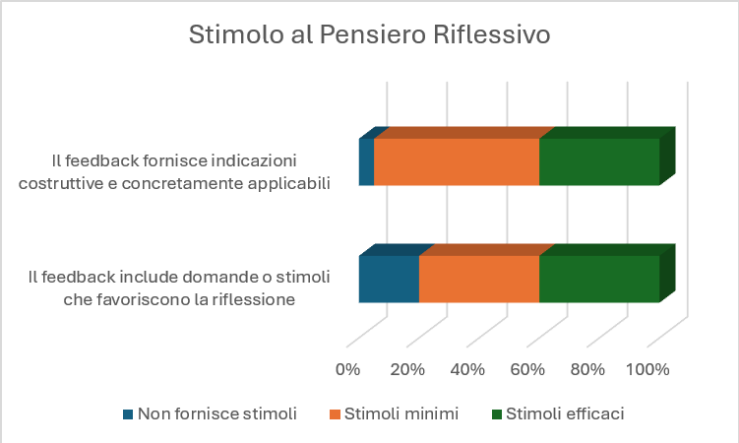


Figura 6: Grafico risposte "Stimolo al pensiero riflessivo"

L'analisi qualitativa effettuata sulla base delle risposte alla domanda aperta "Confronto descrittivo tra il suo feedback e quello generato dall'IA" ha, inoltre, confermato quanto emerso dai risultati quantitativi. Il feedback generato dall'IA appare generalmente chiaro e in diversi casi vicino a quello prodotto dai partecipanti, ma presenta ancora margini di miglioramento sul piano della precisione tecnica, della completezza, del coinvolgimento motivazionale e della promozione del pensiero riflessivo.

4 Conclusione

I risultati del test preliminare devono essere letti alla luce di alcuni limiti metodologici. In primo luogo, il numero ridotto di partecipanti non consente di generalizzare pienamente le conclusioni. Un ulteriore limite riguarda il fatto che lo studio si è concentrato su uno specifico compito di

programmazione; pertanto, ulteriori test saranno necessari per verificare l'efficacia dello strumento in contesti diversi, con campioni più ampi e con differenti tipologie di compiti.

Il presente studio ha testato il potenziale di un plug-in basato sull'IA come supporto alla valutazione formativa nei compiti di programmazione. I risultati evidenziano che il feedback generato viene percepito complessivamente in modo positivo, soprattutto per la sua chiarezza comunicativa. Questo rappresenta un aspetto rilevante, poiché un feedback chiaro e comprensibile costituisce una condizione essenziale affinché lo studente possa utilizzarlo per migliorare il proprio apprendimento [4].

Tuttavia, l'analisi ha messo in evidenza anche alcuni limiti che richiederanno future modifiche e implementazioni. Il feedback generato dal plug-in, al momento, risulta meno efficace nel promuovere coinvolgimento e motivazione, mentre la capacità di stimolare il pensiero riflessivo non emerge in modo sempre stabile. Questi aspetti sono centrali nella valutazione formativa, che non mira soltanto a comunicare un errore, ma a guidare lo studente nella comprensione del proprio processo di apprendimento e nell'individuazione di strategie di miglioramento.

Alla luce di questi risultati, AI CodeRunner può essere considerato uno strumento promettente, ma da utilizzare in modo integrato con l'intervento del docente. L'IA, infatti, non può sostituire il ruolo centrale dell'insegnante, che rimane essenziale nel guidare il processo di apprendimento e nell'interpretare i risultati della valutazione in modo olistico [8].

Bibliografia

[1] Bywater, J. P., Chiu, J. L., Hong, J., & Sankaranarayanan, V. (2019). The teacher responding tool: Scaffolding the teacher practice of responding to student ideas in mathematics classrooms. *Computers & Education, 139*, 16-30. https://doi.org/10.1016/J.COMPEDU.2019.05.004

[2] Grion, V., & Serbati, A. (Eds.). (2019). *Valutazione sostenibile e feedback nei contesti universitari. Prospettive emergenti, ricerche e pratiche*. Pensa MultiMedia.

[3] Hao, Q., Smith, D. H., Ding, L., Ko, A., Ottaway, C., Wilson, J., Arakawa, K. H., Turcan, A., Poehlman, T., & Greer, T. (2021). Towards understanding the effective design of automated formative feedback for programming assignments. Computer Science Education. https://doi.org/10.1080/08993408.2020.1860408

[4] Hattie, J., & Timperley, H. (2007). The Power of Feedback. Review of Educational Research, 77(1), 81-112. https://doi.org/10.3102/003465430298487

[5] Heritage, M. (2013). Formative assessment in practice: A process of inquiry and action. Cambridge, MA: Harvard Education Press.

[6] Keuning, H., Jeuring, J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1), Article 3, 1–43. https://doi.org/10.1145/3231711

[7] Kinnunen, P., & Simon, B. (2012). My program is ok–am i? Computing freshmen's experiences of doing programming assignments. *Computer Science Education, 22(1)*, 1–28. https://doi.org/10.1080/08993408.2012.655091

[8] Li, T., Reigh, E., He, P., & Miller, E. A. (2023). Can we and should we use artificial intelligence for formative assessment in science? Journal of Research in Science Teaching, 60(6), 1385-1389. https://doi.org/10.1002/tea.21867

[9] Pastore, S. (2020). *Saper (ben) valutare: Repertori, modelli e istanze formative per l'assessment literacy degli insegnanti*. Mondadori Università

[10] Smith, A., Leeman-Munk, S., Shelton, A., Mott, B., Wiebe, E., & Lester, J. (2019). A multimodal assessment framework for integrating student writing and drawing in elementary science learning. IEEE Transactions on Learning Technologies, 12(1), 3–15. https://doi.org/10.1109/TLT.2018.2799871

[11] Watts, F. M., Liu, L., Ober, T. M., Song, Y., Jusino-Del Valle, E., Zhai, X., Wang, Y., & Liu, N. (2025). A framework for designing an AI chatbot to support scientific argumentation. Education Sciences, 15(11), 1507. https://doi.org/10.3390/educsci15111507

Giochi di Numeri
Dall'aritmetica al pensiero ricorsivo

Gionata Massi[1], Alberto Cesaretti[2] e Pamela Sanchini[2]

[1] Istituto di Istruzione Superiore "Savoia Benincasa" di Ancona
gionata.massi@savoiabenincasa.it
[2] Liceo Scientifico e Musicale "Einstein" di Rimini
{alberto.cesaretti,pamela.sanchini}@liceoeinstein.it "

Sommario

Questo resoconto descrive un corso extracurricolare di 10 ore, svolto con 15 studenti del primo biennio del Liceo Matematico, per introdurre il pensiero ricorsivo usando l'aritmetica come contesto e *Scheme* come linguaggio. A partire da tre primitive sui naturali, `zero?`, `s` (successore) e `p` (predecessore), e da regole di manipolazione simbolica, gli studenti sono guidati alla definizione ricorsiva di procedure computazionali. Il percorso è coerente con i traguardi e gli obiettivi della proposta CINI per il primo biennio della scuola secondaria di secondo grado. L'approccio ha inizialmente richiesto tempi distesi per la notazione prefissa e la formalizzazione; nelle ultime lezioni ha favorito la capacità di prevedere l'esito dei programmi e di riconoscere la struttura ricorsiva delle soluzioni.

1 Contesto e obiettivi

Il percorso è stato realizzato nell'ambito del progetto dell'IIS "Savoia Benincasa" "Citizen scientists of the future", finanziato dal Piano Nazionale di Ripresa e Resilienza (PNRR), Missione 4 – Istruzione e Ricerca, Investimento 3.1 "Nuove competenze e nuovi linguaggi". Il gruppo destinatario era costituito da 15 studenti del primo biennio del Liceo Matematico; non erano richieste competenze pregresse di programmazione, ma soltanto familiarità con i numeri naturali, il significato delle quattro operazioni, l'ordine di precedenza e la valutazione delle espressioni con parentesi.

La scelta del tema è stata concordata tra gli insegnanti di matematica e di informatica e prende ad oggetto di studio un contesto che dovrebbe essere ben chiaro agli studenti, anche se gli autori riscontrano frequentemente nella propria pratica delle difficoltà: molti studenti sanno usare procedure aritmetiche, ma non riescono a descriverle come processi eseguibili da una macchina, e quasi nessuno ricorda la divisione euclidea. L'aritmetica ha quindi funzionato come *contesto* già familiare allo studente, non come fine del percorso, e l'obiettivo centrale era la transizione dal "saper fare" al "saper far fare" aritmetica. I nuclei disciplinari informatici trattati sono stati: descrizione algoritmica, linguaggio formale, strutture dati elementari (atomi, liste), ricorsione, valutazione di espressioni e correttezza delle procedure.

Il percorso è stato progettato per perseguire i seguenti traguardi e obiettivi della Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola [4]: comprensione della necessità di un esecutore automatico per esprimere algoritmi in modo non ambiguo (T-S-1); riconoscimento della generalità dell'algoritmo rispetto al singolo input (T-S-2); giustificazione della correttezza di un algoritmo (T-S-3); previsione del risultato di un programma senza eseguirlo (O-S-P-2); progettazione di programmi modulari con procedure e funzioni (O-S-P-6); scrittura di semplici programmi in un linguaggio testuale (O-S-P-7).

2 Scelte metodologiche e strumenti

L'ambiente scelto è *WeScheme*[1] perché permette di valutare espressioni in linguaggio *Scheme* direttamente nel browser. Il linguaggio si presta a costruire una *forma mentis* ricorsiva, capace di definire procedure e strutture che si riferiscono a sé stesse, senza richiedere una sintassi complessa. Tre forme sintattiche, `define`, `lambda` e `cond`, sono sufficienti per costruire tutte le procedure proposte. Altri linguaggi avrebbero comportato l'introduzione di concetti aggiuntivi.

Scheme, inoltre, è il linguaggio scelto nei testi da cui gli autori hanno tratto ispirazione [1, 2, 3]. Come in [3], lo schema ricorsivo non è stato introdotto come regola astratta, ma fatto emergere per imitazione e ripetizione: gli studenti hanno osservato numerose procedure già scritte, ne hanno discusso la struttura comune e l'hanno poi riprodotta su problemi nuovi. A livello di progettazione degli algoritmi si è applicato il principio della *ricorsione strutturale* [2]: la struttura della procedura ricalca la struttura induttiva del dato in ingresso. Nel caso dei numeri naturali ciò significa scomporre ogni numero n diverso da zero nel suo predecessore $(p\ n)$, riconducendo il problema a un'istanza via via più vicina al caso base *zero*.

Per sostenere la distinzione tra espressione e sua valutazione abbiamo usato sistematicamente quattro mediatori iconici per la stessa espressione `(+ 1 2)`: la scrittura prefissa (Fig. 1a), la struttura a lista (Fig. 1b), l'albero sintattico (Fig. 1c) e il cerchio di valutazione di Bootstrap (Fig. 1d). Il passaggio tra le rappresentazioni (Fig. 1) è stato usato per esplicitare la struttura ricorsiva dell'espressione e l'ordine di valutazione.

Nel caso dei numeri naturali e delle operazioni su di essi sono stati usati i regoli Cuisenaire.



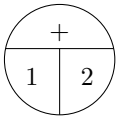
(a) Espressione simbolica `(+ 1 2)`



(b) Rappresentazione come lista



(c) Albero sintattico



(d) Cerchio di valutazione

Figura 1: Quattro rappresentazioni dell'espressione `(+ 1 2)` usate come mediatori

Per ragionare sulle espressioni, rappresentate come liste (Fig. 1b), agli studenti sono state fornite le definizioni in Scheme delle costanti, dei predicati e delle funzioni per la manipolazione di liste usate

[1] Disponibile all'indirizzo: https://www.wescheme.org/. Dopo il termine del corso, per semplificare la scrittura dei programmi e visualizzare il processo di calcolo, è stato costruito un ambiente didattico online: https://gionatamassibenincasa.github.io/process-visualizer/

nelle prime tre lezioni[2]: la costante `lista-vuota`, i predicati `atomo?`, `lista?`, `lista-vuota?` e `uguale?`, le procedure `primo`, `resto`, `anteponi` e `lista`. Le primitive aritmetiche `s`, `p` e `zero?` sono state introdotte separatamente e restano in uso per l'intero percorso.

3 Articolazione del percorso

Le 10 ore sono state organizzate in cinque incontri da due ore.

Nel primo incontro si è conosciuto l'ambiente WeScheme e si è lavorato sui concetti di atomo, lista, albero sintattico ed espressione, introducendo la notazione prefissa e il modello di valutazione per sostituzione.

Nel secondo incontro si è introdotta la forma condizionale e gli studenti hanno usato predicati e condizionali per distinguere caso base e passo ricorsivo nella manipolazione di liste di atomi e numeri naturali.

Nel terzo incontro sono state costruite procedure ricorsive su liste: riconoscimento di liste di atomi, semplici algoritmi di ricerca e di sostituzione di valori.

Il quarto e il quinto incontro sono stati dedicati esclusivamente alle operazioni sui numeri naturali: alcune procedure sono state sviluppate dall'insegnante, altre costruite dagli studenti a partire dallo schema già noto; altre ancora sono state presentate con nomi non significativi e fatte riconoscere tramite simulazione.

Addizione, sottrazione e predicati di confronto sono stati definiti senza ricorrere a tabelline o alla notazione in colonna, ma a partire dalle sole primitive `s`, `p`, `zero?` e da uno schema ricorsivo comune: la scomposizione strutturale del numero naturale. La Fig. 2 illustra il caso dell'addizione: il secondo argomento è ridotto ricorsivamente fino al caso base, mentre i successori da applicare al primo restano sospesi. Per ogni valutazione abbiamo affiancato tre piani: la manipolazione dei regoli Cuisenaire, la riscrittura delle espressioni intermedie alla lavagna e la relativa rappresentazione grafica (Fig. 3 per `(addizione 3 2)`)[3].

La divisione intera è stata introdotta presentando il listato con il nome non significativo `operazione-misteriosa`: gli studenti, osservando la valutazione di `(operazione-misteriosa 12 4)`, dovevano stabilire se l'operazione fosse loro nota e, in caso affermativo, attribuirle il nome corretto. La definizione ricorsiva della procedura `quoziente` (Fig. 8) esprime il quoziente tramite sottrazioni successive: il primo argomento viene progressivamente ridotto fino a un valore minore del secondo, raggiungendo così il caso base, mentre i successori restano sospesi e vengono sommati durante la fase di contrazione, esattamente come nel caso dell'addizione. Come illustrato in Fig. 4 con i regoli Cuisenaire, questo meccanismo rende visibile, passo dopo passo, la costruzione del quoziente.

Si è infine accennato a come la moltiplicazione possa essere definita come addizione ripetuta e l'elevamento a potenza come moltiplicazione ripetuta, cambiando soltanto l'operatore. La generalizzazione del pattern ricorsivo in una funzione di ordine superiore sarebbe andata oltre gli obiettivi del percorso, ma è stata accennata come esempio di astrazione.

[2] Per comodità d'uso in classe sono stati adottati nomi italiani al posto dei corrispondenti predefiniti in Scheme: `lista-vuota` per `'()`, `atomo?` (non primitiva, definita in termini di `pair?`), `lista?` per `list?`, `lista-vuota?` per `null?`, `uguale?` per `eq?`, `primo` per `car`, `resto` per `cdr`, `anteponi` per `cons`, `lista` per `list`.

[3] Con l'eccezione di `minore?` e `sottrazione`, che generano un processo iterativo, le procedure presentate generano processi ricorsivi lineari: la valutazione si espande accumulando operazioni differite (`s`) fino al caso base, per poi contrarsi [1, par. 1.2.1]. L'inefficienza del processo non è stata oggetto di discussione, non rientrando tra gli obiettivi del corso.

```
; (addizione n m) produce la somma di n e m
; es.: (addizione 3 0) -> 3
;      (addizione 3 2) -> 5
(define addizione
  (lambda (n m)
    (cond
      [(zero? m) n]
      [else (s (addizione n (p m)))])))
;
; Esempio di valutazione:
;    (addizione 3 2)
; -> (s (addizione 3 (p 2)))
; -> (s (addizione 3 1))
; -> (s (s (addizione 3 (p 1))))
; -> (s (s (addizione 3 0)))
; -> (s (s 3))
; -> (s 4)
; -> 5
```

Figura 2: Definizione ricorsiva di `addizione` con esempio di valutazione di `(addizione 3 2)`

4 Risultati e limiti

Le considerazioni che seguono derivano dalle attività e dalle discussioni in classe; non è stato possibile raccogliere ex post dati strutturati o prove comparabili.

Diversi studenti hanno iniziato a leggere i programmi come oggetti da interpretare e discutere, non solo da eseguire: era frequente chiedere di prevedere l'esito di una chiamata prima di lanciarla e di riscriverla passo dopo passo, con un miglioramento percepibile rispetto alle prime lezioni. Gli studenti hanno appreso gradualmente il lessico disciplinare: *espressione*, *procedura*, *caso base*, *espressione intermedia*, *argomento*, *parametro*[4]. La ricorsione, inizialmente percepita come artificio, è stata riconosciuta da una parte della classe come tecnica generale di scomposizione dei problemi.

Il percorso ha anche mostrato limiti chiari. La sintassi prefissa di *Scheme* crea una soglia di ingresso più alta rispetto a linguaggi con notazione infissa. Dieci ore sono sufficienti per costruire intuizioni iniziali robuste, ma non per consolidarle in modo uniforme: una parte della classe ha saputo riutilizzare il pattern ricorsivo in compiti nuovi, mentre altri studenti hanno incontrato difficoltà.

Per una futura edizione sarebbe utile introdurre verifiche iniziali e finali per misurare l'evoluzione nella capacità di predire il comportamento dei programmi e di progettare definizioni ricorsive semplici. Sarebbe inoltre opportuno affiancare a *Scheme* una fase conclusiva di trasferimento in un linguaggio testuale più diffuso, per mostrare la trasferibilità delle competenze acquisite.

Nel complesso l'esperienza suggerisce che, anche nel primo biennio, è possibile lavorare su contenuti fortemente disciplinari senza ridurre l'Informatica all'uso di applicazioni. La costruzione ricorsiva dell'aritmetica ha offerto un contesto motivante proprio perché ha reso esplicito che dietro operazioni note esistono rappresentazioni formali e algoritmi. La scelta degli esempi è coerente con il profilo del Liceo Matematico e richiede adattamenti in altri indirizzi.

[4] La distinzione tra *argomento* (il valore passato nella chiamata) e *parametro* (l'identificatore nella definizione della procedura) segue l'uso adottato in [1, 2].

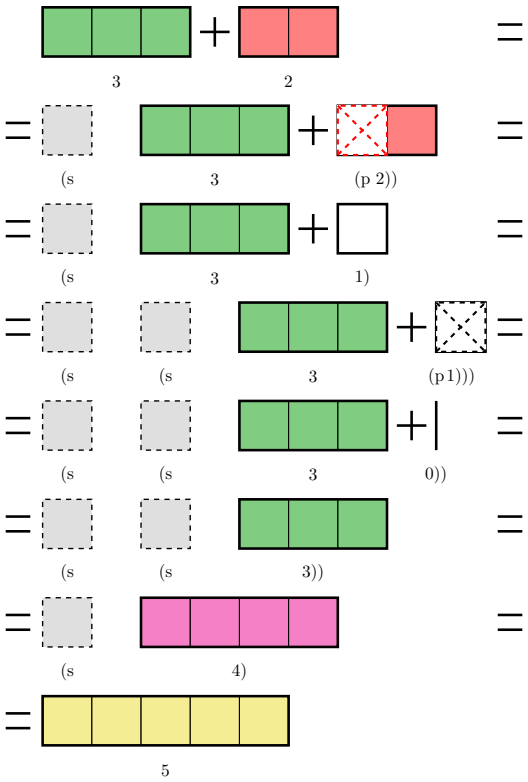


Figura 3: Valutazione di `(addizione 3 2)` con i regoli Cuisenaire: riduzione del secondo argomento e applicazione dei successori

A Esempi di codice usati nel percorso didattico

Le definizioni delle procedure primitive per l'aritmetica[5] sono in Fig. 5.

Le procedure ricorsive per esprimere formalmente operatori e predicati matematici e per il calcolo del quoziente della divisione intera sono riportate nelle Figg. 6, 7 e 8.

Riferimenti bibliografici

[1] Harold Abelson e Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*. Versione *open access*: https://sarabander.github.io/sicp/. The MIT Press, 1996, p. 683. ISBN: 9780262510875.

[2] Matthias Felleisen et al. *How to design programs: An introduction to programming and computing*. 2ª ed. MIT Press, 2018.

[3] Daniel P Friedman e Matthias Felleisen. *The Little Schemer*. MIT Press, 1995.

[5] `s` e `p` sono definite tramite `+` e `-` di Scheme per comodità pratica; concettualmente svolgono il ruolo di primitive. La procedura `zero?` è già fornita dal linguaggio.

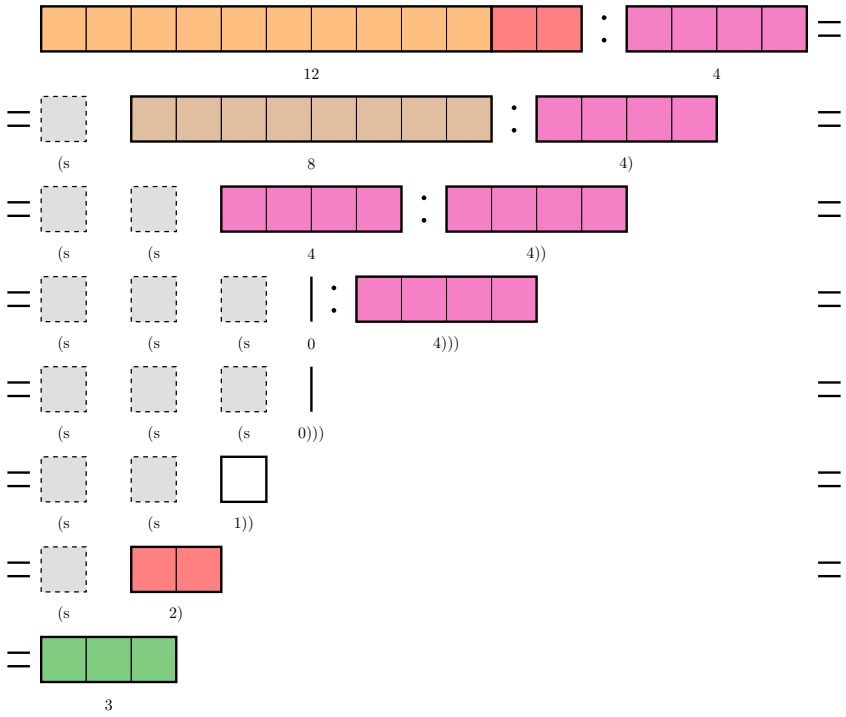


Figura 4: Valutazione di `(quoziente 12 4)` con i regoli Cuisenaire: sottrazioni ripetute e costruzione del quoziente

```
; (s n) produce il numero che segue immediatamente n
; es.: (s 3) -> 4
(define s
  (lambda (n)
    (+ n 1)))

; (p n) produce il numero che precede immediatamente n
; es.: (p 3) -> 2
(define p
  (lambda (n)
    (- n 1)))

; (zero? n) produce #t se n è zero, #f altrimenti
; es.: (zero? 0) -> #t
;      (zero? 3) -> #f
; è già definito dal linguaggio Scheme
```

Figura 5: Primitive aritmetiche usate per costruire ricorsivamente le altre operazioni

```
; (minore? n m) produce #t se n è minore di m, #f altrimenti
; es.: (minore? 2 5) -> #t
;      (minore? 5 2) -> #f
(define minore?
  (lambda (n m)
    (cond
      [(zero? m) #f]
      [(zero? n) #t]
      [else (minore? (p n) (p m))])))
;
; Esempio di valutazione:
;    (minore? 2 5)
; -> (minore? 1 4)
; -> (minore? 0 3)
; -> #t
```

Figura 6: Definizione ricorsiva di `minore?` per confronto tra numeri naturali

```
; (sottrazione n m) produce la differenza tra n e m
; es.: (sottrazione 5 2) -> 3
(define sottrazione
  (lambda (n m)
    (cond
      [(zero? m) n]
      [else (sottrazione (p n) (p m))])))
;
; Esempio di valutazione:
;    (sottrazione 5 2)
; -> (sottrazione 4 1)
; -> (sottrazione 3 0)
; -> 3
```

Figura 7: Definizione ricorsiva di `sottrazione` tramite decremento simultaneo dei due argomenti

[4] Gruppo di Lavoro CINI "Informatica e Scuola". *Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola*. Documento redatto in collaborazione con GII e GRIN. 2017. URL: https://www.consorzio-cini.it/images/Proposta-Indicazioni-Nazionali-Informatica-Scuola-numerata.pdf.

```
; (quoziente n m) produce il quoziente della divisione intera di n per m
; es.: (quoziente 12 4) -> 3
(define quoziente
  (lambda (n m)
    (cond
      [(minore? n m) 0]
      [else (s (quoziente (sottrazione n m) m))])))
;
; Esempio di valutazione:
;    (quoziente 12 4)
; -> (s (quoziente 8 4))
; -> (s (s (quoziente 4 4)))
; -> (s (s (s (quoziente 0 4))))
; -> (s (s (s 0)))
; -> (s (s 1))
; -> (s 2)
; -> 3
```

Figura 8: Definizione ricorsiva di `quoziente`

Valutare il pensiero computazionale: operazionalizzazione e criticità inferenziali nella ricerca sul transfer cognitivo

Giulia Paludo[1,2], Alberto Montresor[1,2], and Angela Pasqualotto[3]

[1] Università di Trento, Dipartimento di Ingegneria e Scienza dell'Informazione
[2] Laboratorio CINI Informatica e Scuola
[3] Scuola universitaria professionale della Svizzera italiana, Dipartimento formazione e apprendimento – Alta scuola pedagogica

Sommario

Il contributo riflette sul problema della valutazione del pensiero computazionale e sulle implicazioni che le scelte valutative hanno sulla ricerca e l'insegnamento dell'informatica. Attraverso una revisione sistematica della letteratura vengono discusse le principali operazionalizzazioni del costrutto e le relative inferenze. I risultati mostrano una forte eterogeneità in definizioni e strumenti adottati, con ricadute sia sulla comparabilità degli studi sia sulla progettazione didattica.

1 Introduzione

Il concetto di *pensiero computazionale* (PC) ha raggiunto una significativa notorietà a partire dai lavori Jeannette Wing [9, 10], che lo descrive come l'insieme dei processi mentali coinvolti nella formulazione dei problemi e delle loro soluzioni, affinché queste ultime possano essere rappresentate in una forma eseguibile efficacemente da un agente di elaborazione dell'informazione. La diffusione di questo costrutto ha raggiunto la progettazione curriculare della scuola dell'obbligo, portando un costrutto con un'origine disciplinare nell'informatica all'attenzione anche della pedagogia e della psicologia dell'educazione.

Questa apertura interdisciplinare testimonia la rilevanza educativa del PC, ma richiede al tempo stesso di mantenerne esplicito il radicamento nell'informatica. Il rischio, altrimenti, è che il costrutto venga progressivamente separato dai concetti, dai metodi e dalle pratiche della disciplina da cui trae origine, fino a essere interpretato come un insieme generico di abilità cognitive o trasversali. Il PC è dunque diventato oggetto di studio di ambiti disciplinari diversi, ma proprio questa circolazione tra comunità scientifiche differenti genera una significativa tensione teorica e metodologica, soprattutto quando le definizioni risultano non esaustive o le operazionalizzazioni riducono il costrutto ad alcune sue componenti parziali.

Alla luce di queste premesse, la frammentazione teorica del PC si riflette direttamente nelle pratiche di valutazione. Infatti, la variabilità degli strumenti e dei criteri adottati rende difficile stabilire cosa venga effettivamente misurato quando gli studi riportano effetti del PC [2]. Di conseguenza, la questione centrale non riguarda soltanto l'eventuale capacità del PC di produrre effetti di trasferimento, ma anche cosa venga effettivamente misurato quando gli studi affermano di valutare il PC.

Il presente contributo propone dunque una riflessione sulla valutazione del PC, sviluppata a partire da una review sistematica della letteratura condotta secondo la metodologia PRISMA [4]. L'analisi prende in esame studi peer-reviewed pubblicati tra il 2015 e il 2025 e adotta una prospettiva basata su costrutti e inferenze, anziché su una classificazione guidata dagli strumenti disponibili.

2 Il problema della valutazione

L'analisi di una selezione di 23 studi volti a indagine gli effetti cognitivi di percorsi formativi sul PC ha evidenziato una significativa frammentazione sia dei riferimenti teorici, sia degli approcci di valutazione

adottati. A differenza di costrutti cognitivi quali le funzioni esecutive o il ragionamento logico, per i quali esistono tradizioni consolidate di valutazione e strumenti relativamente condivisi, il PC continua infatti a essere caratterizzato da una forte eterogeneità teorica e operativa. Tale frammentazione si riflette direttamente nelle pratiche di valutazione, che differiscono per framework teorico di riferimento, tipologia di compiti utilizzati, grado di strutturazione e componenti del costrutto considerate.

Alcuni test standardizzati, come il Computational Thinking Test (CTt; [7]), privilegiano costrutti computazionali quali sequenze, cicli e selezioni attraverso item altamente strutturati. Altri strumenti, come TechCheck [6] o TechCheck-K [5], adottano invece una prospettiva più orientata allo sviluppo, focalizzandosi su domini introduttivi del PC in età prescolare e senza richiedere competenze di programmazione. Approcci differenti emergono anche nelle valutazioni basate su artefatti di programmazione, come Dr. Scratch [3], che inferiscono il PC attraverso l'analisi della complessità del codice prodotto, oppure nelle rubriche osservative e nei logbook utilizzati soprattutto in robotica educativa. In questi ultimi casi, l'attenzione si concentra maggiormente sui processi, sulle strategie e sulle pratiche collaborative.

La questione più rilevante che emerge da questa panoramica riguarda dunque cosa venga effettivamente valutato quando si parla di valutazione del PC. I risultati della review mostrano una forte sovra-rappresentazione di competenze quali il sequenziamento, il pensiero algoritmico e l'applicazione procedurale di istruzioni, mentre risultano meno esplorati altri aspetti, potenzialmente altrettanto rilevanti, quali debugging, monitoraggio metacognitivo, riflessione sul processo e pratiche iterative. Inoltre, in molti studi il PC viene ridotto a competenze strettamente legate alla programmazione, con una sostanziale sovrapposizione tra PC e prestazione nella scrittura di codice. Questa sovrapposizione costituisce una delle criticità più significative, poiché PC e programmazione non coincidono: la programmazione rappresenta una possibile modalità di espressione del PC, ma non esaurisce il costrutto. Analogamente, ridurre il PC ad abilità di sequenziamento e pianificazione appare problematico per due ragioni: da un lato, restringe eccessivamente la portata del costrutto; dall'altro, tende a ricondurlo a livelli cognitivi prevalentemente procedurali ed esecutivi, lasciando in secondo piano le dimensioni riflessive, strategiche e metacognitive.

In questa prospettiva, le affermazioni di *far transfer* richiedono particolare cautela. Se il PC viene operazionalizzato attraverso compiti che rilevano componenti circoscritte, come sequenziamento, applicazione di istruzioni o riconoscimento di strutture di controllo, diventa problematico inferire effetti su abilità cognitive generali [8] [1]. In assenza di una mediazione teorica esplicita tra componente computazionale valutata e dominio cognitivo di arrivo, il trasferimento rischia di essere postulato più che dimostrato.

Di conseguenza, le scelte valutative orientano le inferenze teoriche: strumenti differenti non producono semplicemente misure differenti, ma sostengono interpretazioni diverse del rapporto tra PC, apprendimento e cognizione. La variabilità degli approcci di valutazione rende quindi difficile stabilire che cosa venga effettivamente misurato quando gli studi riportano effetti attribuiti al PC, limitando la comparabilità tra le ricerche e complicando l'interpretazione dei risultati empirici.

3 Cosa possiamo inferire dalle valutazioni del Pensiero Computazionale?

Le affermazioni relative agli effetti del PC dipendono in larga misura da come il costrutto viene operazionalizzato e valutato. Per questo motivo, prima ancora di discutere eventuali effetti del PC, è necessario interrogarsi su che cosa venga effettivamente misurato e quali inferenze possano essere tratte dai risultati ottenuti. Riprendendo il nodo introdotto nel paragrafo precedente, le ambiguità metodologiche riguardano, in primis la definizione e valutazione del costrutto. In numerosi studi, interventi basati su robotica educativa, Scratch, informatica unplugged o altra programmazione a blocchi vengono

considerati implicitamente "*interventi di PC*", senza però includere una misura diretta del PC o senza esplicitare quale definizione del costrutto venga adottata. Questo ha un importante impatto sulle conseguenze inferenziali in quanto la mancata misura esplicita e non ambigua del PC non rende possibile stabilire con chiarezza quale componente sia effettivamente coinvolta. Ad esempio, attività focalizzate prevalentemente sull'applicazione procedurale di istruzioni vengono frequentemente interpretate come espressioni di PC nel suo complesso, nonostante coinvolgano soltanto componenti molto specifiche del costrutto. In questa prospettiva, la letteratura si popola di una serie di costrutti, quali programmazione, pianificazione, sequenziamento e altri, che rischiano di essere utilizzati in modo intercambiabile, generando ulteriore ambiguità teorica e metodologica.

L'aspetto pratico della valutazione diventa dunque inferenziale: se non viene chiarito cosa venga effettivamente misurato come PC, diventa difficile interpretare i risultati delle attività didattiche e confrontare evidenze provenienti da contesti diversi. La variabilità delle valutazioni, la presenza di proxy non dichiarati e l'assenza di coerenza tra definizione teorica, intervento e misura del costrutto indeboliscono la possibilità di comprendere quali competenze informatiche vengano effettivamente sviluppate e valutate.

In assenza di prassi di valutazione teoricamente coerenti ed esplicite, le inferenze che è possibile trarre sui risultati dell'insegnamento del pensiero computazionale rischiano di superare ciò che gli strumenti adottati consentono realmente di osservare.

4 Implicazioni per l'insegnamento dell'informatica

Per i docenti di informatica, questa questione ha implicazioni concrete. Strumenti focalizzati esclusivamente sul prodotto finale o sull'applicazione procedurale di istruzioni tendono a rendere visibili soltanto alcune componenti del PC. Al contrario, valutazioni che includono debugging, progettazione algoritmica, riflessione sulle soluzioni e pratiche iterative permettono di osservare dimensioni maggiormente coerenti con una visione disciplinare dell'informatica. Le scelte valutative non sono quindi neutre: contribuiscono a definire quali aspetti dell'informatica vengono riconosciuti, insegnati e appresi. Una riflessione più esplicita sugli strumenti di valutazione potrebbe quindi favorire una maggiore coerenza tra obiettivi didattici, pratiche di insegnamento e risultati della ricerca, contribuendo a una definizione più condivisa del costrutto di pensiero computazionale. Operativamente, la valutazione del PC dovrebbe partire dall'esplicitazione delle componenti che si intendono osservare e mantenerle allineate con attività e strumenti valutativi. È quindi opportuno combinare evidenze di prodotto e di processo, non limitandosi alla correttezza della soluzione ma approfondendo i retroscena delle strategie adottate. L'interpretazione dei risultati dovrebbe infine essere circoscritta alle componenti effettivamente osservate, evitando di inferire dal successo in un singolo compito una competenza generale di PC. Ad esempio, un compito di sequenziamento, pur essendo coerente con alcuni dei processi coinvolti nel PC, non può essere considerato una rappresentazione complessiva del costrutto.

5 Conclusioni

Questo contributo riflette sul ruolo della valutazione del PC, con particolare attenzione su studi che ne indagano la relazione con il trasferimento cognitivo. Dall'analisi sistematica della letteratura emerge la necessità di interrogarsi su che cosa venga effettivamente valutato quando si parla di PC. Si osserva infatti una considerevole eterogeneità nelle definizioni del costrutto, nelle componenti considerate e negli strumenti utilizzati per misurarlo. Tale variabilità ostacola la comparabilità degli studi e limita la possibilità di interpretare i risultati in modo coerente. La valutazione rappresenta quindi un nodo centrale non solo per la ricerca, ma anche per la progettazione dell'insegnamento dell'informatica. Gli strumenti

adottati contribuiscono infatti a definire quali aspetti del PC vengono resi visibili agli studenti e riconosciuti come oggetto di apprendimento. Da questa prospettiva emerge la necessità di sviluppare strumenti di valutazione capaci di rappresentare in modo più esplicito e coerente la natura multidimensionale del PC, favorendo una maggiore comparabilità tra gli studi e una progettazione didattica maggiormente allineata agli obiettivi cognitivi dell'insegnamento.

Riferimenti bibliografici

[1] Barbara Arfé, Tullio Vardanega, Chiara Montuori, and Marta Lavanga. Coding in primary grades boosts children's executive functions. *Frontiers in Psychology*, 10:2713, 2019.

[2] Daniel Hickmott and Elena Prieto-Rodriguez. To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, 17(2):229–244, 2018.

[3] Jesús Moreno-León and Gregorio Robles. Dr. Scratch: A web tool to automatically evaluate Scratch projects. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, pages 132–133. ACM, 2015.

[4] Matthew J. Page et al. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ*, 372:n71, 2021.

[5] Emily Relkin and Marina Umaschi Bers. TechCheck-K: A measure of computational thinking for kindergarten children. In *2021 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2021.

[6] Emily Relkin, Laura de Ruiter, and Marina Umaschi Bers. TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*, 29:482–498, 2020.

[7] Marcos Román-González, Juan-Carlos Pérez-González, and Carmen Jiménez-Fernández. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72:678–691, 2017.

[8] Giovanni Sala, N. Deniz Aksayli, K. Semir Tatlidil, Tomoko Tatsumi, Yasuyuki Gondo, and Fernand Gobet. Near and far transfer in cognitive training: A second-order meta-analysis. *Collabra: Psychology*, 5(1):18, 2019.

[9] Jeannette M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[10] Jeannette M. Wing. Research notebook: Computational thinking—what and why. *The Link Magazine*, pages 20–23, 2011.

Poster

“Cerco, capisco, imparo”. Esperienza extracurricolare per il potenziamento del pensiero logico attraverso l’informatica

Emilia Ciccia[1], Marilena Salsano[2]
[1] Istituto Comprensivo Praia a Mare (CS)
emilia.ciccia@scuola.istruzione.it
[2] Istituto Comprensivo Praia a Mare (CS)
marilena.salsano@istruzione.it

Abstract

Il contributo presenta un laboratorio dell’IC Praia che ha integrato pensiero computazionale e matematica, in linea con le Indicazioni Nazionali 2025. L’informatica, "grammatica del pensiero", ha potenziato le abilità logiche per le prove INVALSI. Tramite moduli su analisi testuale, coding e crittografia, gli alunni hanno superato il mero calcolo a favore della strategia. Centrale è stata la gestione dell'errore come "bug": il debugging ha trasformato gli studenti in architetti di soluzioni.

1 Introduzione

Le Nuove Indicazioni Nazionali per il Curricolo 2025 [4] segnano un’evoluzione epistemologica e normativa per la scuola primaria italiana, introducendo l’informatica non come mera alfabetizzazione all'uso di strumenti, bensì come disciplina integrata nei nuclei fondanti della matematica e dell'area STEM. Al termine della classe quinta, i traguardi fissati mirano a far comprendere la distinzione tra dato e informazione, il concetto di algoritmo, la scrittura e decodifica di semplici programmi e la loro verifica funzionale. La dimensione computazionale diventa così un linguaggio operativo a supporto del ragionamento logico, aritmetico e geometrico. Questa convergenza ridefinisce il problem solving, promuovendo un'architettura cognitiva complessa [5]:

- Pensiero induttivo e deduttivo, per estrarre regole dai casi particolari o applicarle al contesto specifico;
- Pensiero laterale, associativo e non concettuale, a fondamento del pensiero critico per prevenire bias cognitivi e analizzare la realtà;
- Pensiero computazionale, basato su astrazione, scomposizione analitica (*decomposition*) e riconoscimento di regolarità (*pattern recognition*).

Matematica e informatica condividono la modellizzazione strutturata della complessità, fornendo una vera e propria «grammatica del pensiero». Tuttavia, la letteratura avverte che il *transfer* cognitivo dalle abilità di coding a competenze logico-matematiche generali non è automatico: richiede una didattica intenzionale e contesti applicativi significativi [6, 7]. In tale quadro pedagogico, assume un ruolo centrale la risemantizzazione dell’errore: il *bug* cessa di essere un fallimento per trasformarsi in informazione utile alla riflessione metacognitiva. Il *debugging* educa a ripercorrere i passaggi, isolare le incongruenze e ripianificare le strategie. Su questi presupposti si fonda il laboratorio *"Cerco, capisco, imparo"*, ideato per potenziare il ragionamento strategico in quinta primaria attraverso metodologie collaborative, attive e a mediazione tecnologico-manipolativa.

2 Il laboratorio “Cerco, capisco, imparo”: partecipanti e obiettivi

Il laboratorio “Cerco, capisco, imparo”, attuato presso l’I.C. di Praia a Mare per gli alunni di classe quinta primaria, nasce per rispondere all’esigenza di preparare i discenti alle prove nazionali

INVALSI di matematica con serenità e consapevolezza. Il progetto intende superare le criticità del tradizionale addestramento standardizzato fatto di simulazioni cartacee ripetitive, prassi che spesso riduce l'apprendimento a puro calcolo mnemonico e genera ansia da prestazione. In linea con le Nuove Indicazioni Nazionali 2025, l'intervento sfrutta l'integrazione epistemica tra matematica e informatica per convertire i quesiti valutativi in autentiche *sfide logiche*. Tramite la cooperazione, il coding visuale e la decodifica crittografica, il focus si sposta dall'esecuzione meccanica alla comprensione profonda della struttura del problema, stimolando resilienza, flessibilità cognitiva e strategie risolutive autonome. L'impianto metodologico poggia sulle evidenze dell'Education Endowment Foundation [3] relative all'istruzione diretta e metacognitiva, declinate in due raccomandazioni cardine:

1. Insegnamento esplicito delle strategie metacognitive, articolato in una sequenza rigorosa in sei fasi: *Attivazione*; *Istruzione esplicita*; *Pratica guidata*; *Pratica indipendente*; *Feedback del docente*; *Riflessione strutturata*.
2. Calibrazione e gradualità della sfida cognitiva: il percorso modula il carico informativo per preservare la motivazione, scomponendo i compiti complessi in passaggi elementari al fine di contenere la frustrazione e consolidare le competenze di problem solving prima di affrontare livelli di difficoltà crescenti [1].

L'intervento si è sviluppato nell'arco di 20 ore complessive, suddivise in 10 incontri extracurricolari di due ore ciascuno. Il curricolo laboratoriale ha integrato tre macro-ambiti tematici fondamentali:

1. Algoritmi e programmazione (Coding), per formalizzare procedure e sviluppare il pensiero computazionale;
2. Gestione dell'errore (Debugging), per risemantizzare l'errore come risorsa euristica e momento di riflessione logica;
3. Dati, informazioni e Crittografia, per esplorare la rappresentazione simbolica e padroneggiare la comprensione profonda dei dati.

2.1 Partecipanti

Hanno preso parte al percorso 17 alunni (11 femmine e 6 maschi; età media 10,7 anni, DS = 0,36), di cui 16 frequentanti la classe quinta e 1 alunno di classe quarta ad alto potenziale cognitivo (APC). Il gruppo presentava una familiarità pregressa con le basi del coding a blocchi maturata nell'anno precedente.

La prevalenza della partecipazione femminile (grafico 1) ha rappresentato un riscontro interessante, evidenziando una risposta positiva e motivata da parte delle alunne a proposte laboratoriali di ambito computazionale e logico.

Distribuzione campione	**Alunni Classe 5^**
Femmine	11
Maschi	6
Totale	17
Età media (DS)	10,7 (0,36)

Tabella 1: descrizione alunni

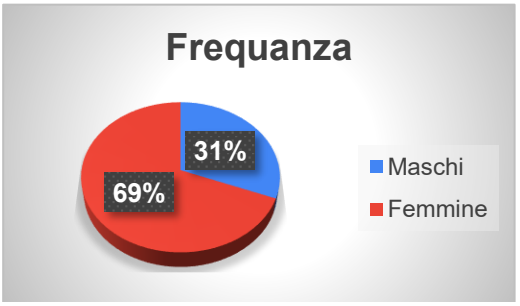


Grafico 1: percentuale di frequenza dei partecipanti

2.2 Obiettivi

Il laboratorio integra la logica computazionale nei nuclei fondanti della matematica (Indicazioni Nazionali 2025) collegandosi ai traguardi e agli obiettivi della Proposta CINI [5]:

- Algoritmi e Programmazione (Coding): uso di piattaforme visuali (Code.org) per padroneggiare sequenze, cicli definiti e iterazioni condizionali.
- Gestione dell'Errore (Debugging): rilevazione e correzione sistematica dei malfunzionamenti nel codice.
- Dati, Informazione e Crittografia: distinzione concettuale tra dato e informazione, uso di variabili, cifrari e formati rappresentativi.

3 Metodologie e attività

In linea con le indicazioni metodologiche del CINI per l'apprendimento attivo e laboratoriale, le attività hanno adottato un impianto collaborativo. Nello specifico, la didattica a stazioni applicata alla crittografia ha reso i meccanismi di cifratura un'esperienza manipolativa e coinvolgente, potenziando la deduzione logica e il riconoscimento di pattern. Il lavoro di squadra e il peer learning hanno favorito l'inclusione e la scomposizione condivisa di problemi complessi**.**

3.1 Approfondimento didattico ed operativo di due attività chiave

Attività A (Unplugged): Codifica, isometrie e stazioni con i Pentamini

Contesto e Obiettivo Disciplinare: Consolidare rotazione, simmetria ed equiestensione; formalizzare l'informatica come corrispondenza biunivoca tra simbolo/orientamento spaziale e dato informativo.

Materiali: 4 set di pentamini in legno, griglie quadrettate, schede operative e stazioni con cifrari (Cesare, PegCode, Enigma).

Fasi di lavoro e consegne operative: Fase 1 – Esplorazione geometrica e vincoli di rotazione: in piccoli gruppi (3-4 alunni), analisi delle proprietà spaziali. Consegna: individuare quali pezzi mantengono la forma ruotando e quali generano fino a 8 configurazioni distinte combinando rotazioni e ribaltamenti speculari; Fase 2 – Progettazione del protocollo di codifica: Consegna: definire un algoritmo condiviso in cui rotazioni e simmetrie del pentamino F (e L) corrispondano a sillabe fisse (orientamento base, rotazioni di 90°/180°/270°, riflessione speculare). I gruppi combinano le tessere per comporre parole bisillabe (es. "FILO") (fig. 1);

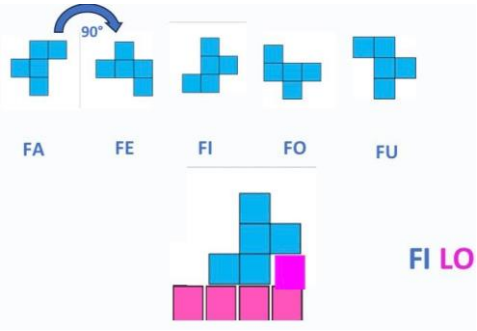


Figura 1: composizione di una parola bisillaba piana con l'utilizzo di due pentamini

Fase 3 – Didattica a stazioni e scambio cooperativo: l'aula viene allestita in 4 isole di lavoro tematiche (fig. 2):

Isola 1: Cifrario a sostituzione di Cesare (traslazione con chiave numerica). Associazione di lettere ai numeri (A=1, B=2...). Spiegazione della "chiave" (es. +3). Gli alunni hanno dovuto cifrare e decifrare collettivamente tre parole. Poi, scambio fra i gruppi di un breve messaggio segreto da decifrare (fig. 3).

Isola 2: Cifrario a scacchiera PegCode (sistema a coordinate ortogonali e pioli colorati). Introduzione di un cifrario non numerico, ma *posizionale* in cui ogni lettera risulta dall'incrocio di due pioli colorati. Elaborazione di messaggi segreti da cifrare e decifrare (fig. 4).

Isola 3: Cilindro crittografico Enigma (combinatoria su ghiere mobili). Scelta di una chiave criptografica corrispondente alla combinazione di 6 caratteri; scrittura di un messaggio cifrato e condivisione agli altri studenti per la decodifica (fig. 5).

Isola 4: Pentamini e codifica geometrico-spaziale (fig. 6).Ogni gruppo ha prodotto un messaggio cifrato sulla propria postazione e si è spostato a rotazione per decodificare il messaggio lasciato dal gruppo precedente.



Figura 2 : didattica a stazioni



Figura 3: scrittura di un messaggio segreto con il Cifrario di Cesare



Figura 4: decodifica di un messaggio con PegCode Quercetti

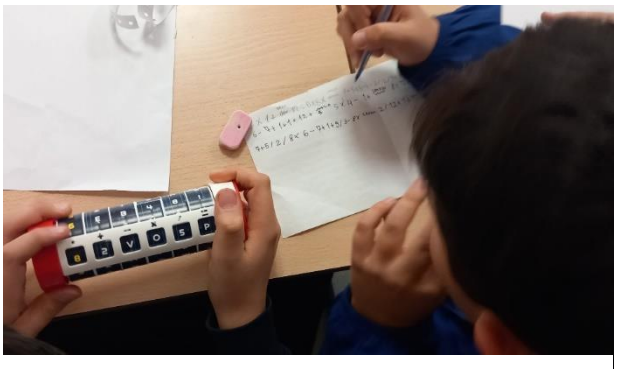


Figura 5: decodifica di un messaggio con Enigma Quercetti

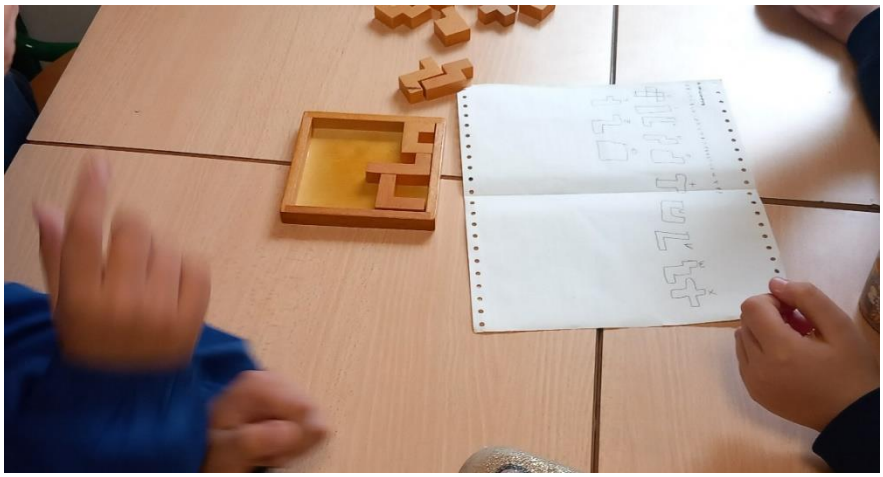


Figura 6: composizione di un messaggio cifrato con i pentamini

Fase 4 – Debriefing metacognitivo: discussione guidata sul rapporto tra orientamento spaziale, invarianza dell'area (equiestensione) e significato del dato. Gli alunni collegano l'esperienza ai quesiti INVALSI, evidenziando la necessità di "decifrare la legenda" e individuare la "chiave" del problema prima di risolverlo.

Difficoltà emerse e mediazione docente: L'iniziale confusione tra rotazione e simmetria assiale (ribaltamento) nei pentamini è stata superata grazie alla manipolazione diretta e all'applicazione di un contrassegno colorato su uno spigolo. Le attività con i cifrari sono risultate di immediata comprensione per tutti.

Attività B (Digitale / Metacognitiva): Sessione di Debugging e analisi dell'errore

Contesto e Obiettivo Disciplinare: Sviluppare controllo esecutivo, tracciamento algoritmico (tracing) e correzione di misconcezioni logico-matematiche (incolonnamento, comprensione del testo, stima angolare).

Materiali: Code.org (modulo Artista), schede con "codice malfunzionante" e problemi matematici complessi.

Fasi di lavoro e consegne operative:

Fase 1 – Debugging visuale del codice geometrico: Analisi di uno script a blocchi con rotazione errata. Consegna: eseguire il codice passo-passo (step-by-step), individuare la deviazione della figura e verbalizzare l'istruzione che causa il *bug*; Fase 2 – Trasferimento al testo matematico: analisi in piccolo gruppo di tre problemi con errori tipici (dati irrilevanti, operatore errato, incolonnamento dei decimali). Consegna: isolare il passaggio incoerente, classificare il bug e riscrivere l'algoritmo corretto; Fase 3 – Condivisione e argomentazione tra pari: esposizione alla classe del bug individuato e giustificazione della correzione con linguaggio formale; Fase 4 – Sintesi metacognitiva: compilazione del "Diario del Debugger". L'errore non è più un fallimento, ma un bug condiviso da categorizzare (calcolo, testo o strategia).

Difficoltà emerse e mediazione docente: Di fronte alla tendenza a cancellare l'intero svolgimento (*trial-and-error*), la docente ha introdotto il tracciamento a ritroso (*dry run*): verifica isolata di ogni riga per controllare la coerenza tra precondizioni e postcondizioni.



Figura 7: costruzione di un esagono con Artista, Code.org

4 Risultati e criticità emerse

L'evoluzione delle abilità matematiche e computazionali è stata monitorata tramite la batteria standardizzata AC-MT 6-11 [2] somministrata a 16 alunni di classe quinta a ottobre 2025 (T1) e maggio 2026 (T2). Il confronto pre-post registra una netta contrazione delle fasce critiche (Richiesta di intervento/attenzione) a favore dei livelli Nella media e Ottimale:
Soluzione di problemi: l'area di maggior progresso; i profili fragili crollano da 8 a 2 alunni, mentre la fascia ottimale sale da 1 a 6 (8 nella media).
Numerosità e Ordinamento: azzeramento totale delle fasce d'allarme, con l'eccellenza che raggiunge rispettivamente 10 e 6 studenti.
Transcodifica: le fragilità scendono da 6 a 2 unità (5 ottimali).
Operazioni scritte: riduzione dei casi critici (da 5 a 3) e raddoppio dei livelli ottimali (da 2 a 4).

Pur a fronte di riscontri positivi e di un'ampia adesione femminile (69%), la ridotta numerosità campionaria e l'assenza di un gruppo di controllo impongono cautela: i miglioramenti non possono essere ascritti causalmente al solo laboratorio, ma riflettono anche la maturazione naturale e la didattica curricolare.

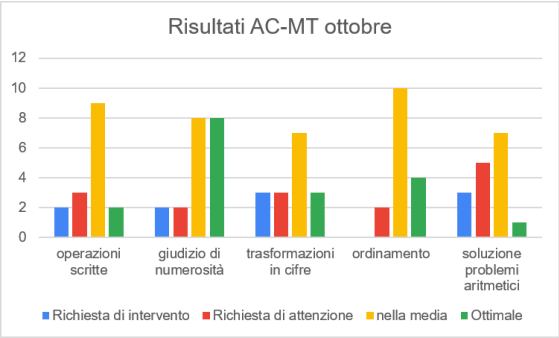


Grafico 1: risultati conseguiti in T1

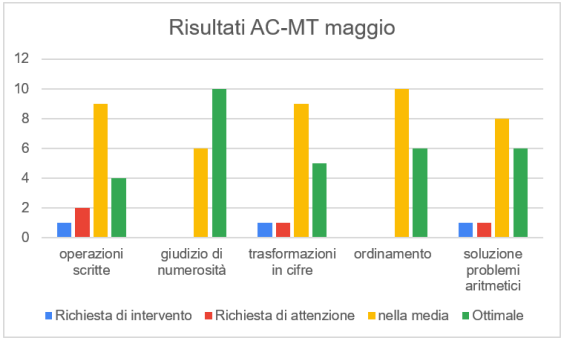


Grafico 3: risultati conseguiti in T2

5 Conclusioni

Il laboratorio “Cerco, capisco, imparo” conferma l'efficacia dell'integrazione tra informatica e matematica come cornice metodologica ed epistemica. Attività mirate come pentamini e debugging hanno reso tangibili costrutti astratti (algoritmi, variabili), trasformando l'errore in risorsa metacognitiva.

Gli esiti dell'AC-MT 6-11 mostrano una netta contrazione delle fragilità nel problem solving e nella transcodifica numerica, affiancata da un'incoraggiante partecipazione femminile (69%). Sebbene il campione ridotto e non controllato imponga prudenza statistica, il modello si rivela una pratica sostenibile e replicabile, capace di emancipare gli studenti dagli automatismi per renderli architetti autonomi e consapevoli di soluzioni complesse.

Bibliografia

[1] Calvani, Antonio, Peru, Andrea, Pellegrini, Marta, Di Martino, Valeria, “Potenziare logica e problem-solving, Carocci Editore, 2023

[2] Cornoldi, C., Lucangeli, D., & Bellina, M. (2020). *AC-MT 6-11: Test di valutazione delle abilità di calcolo e del ragionamento matematico per la scuola primaria* (Nuova ed.). Giunti Edu.

[3] Education Endowment Foundation (2021), Metacognition and Self-Regulated Learning, https://d2tic4wvo1iusb.cloudfront.net/production/eef-guidance-reports/metacognition/metacognition-and-self-regulated-learning_guidance-report.v.2.4.0.pdf

[4] Ministero dell'Istruzione e del Merito (MIM). "Nuove Indicazioni Nazionali per il Curricolo della Scuola dell'Infanzia e del Primo Ciclo di Istruzione", Roma, 2025

[5] Nardelli, Enrico, Forlizzi, Luca, Lodi, Michael, Lonati, Violetta, Mirolo, Claudio, Monga, Mattia, Montresor, Alberto, and Morpurgo, Anna (a cura di). "Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola", Gruppo di Lavoro CINI "Informatica e Scuola", 2017.

[6] Sala, G., & Gobet, F. (2019). *Cognitive training does not enhance general cognition.* Trends in Cognitive Sciences, 23(1), 9–20. https://doi.org/10.1016/j.tics.2018.10.004

[7] Salomon, G., & Perkins, D. N. (1987). *Transfer of cognitive skills from programming: When and how?* Journal of Educational Computing Research, 3(2), 149–169. https://doi.org/10.2190/93GV-HYBX-G76M-42VR

Dal gioco al codice: il postino che insegna gli algoritmi. Un percorso verticale di Informatica tra scuola primaria e secondaria di primo grado

Laura Cusanno[1], Simonetta Siega[2]
[1] I.C. Imbriani Piccarreta, Corato (BA)
laura.cusanno@scuola.istruzione.it
[2] IC DOMO2 - Domodossola (VCO)
simonetta.siega@icdomo2.it
siega.simo@gmail.com

Abstract

Il contributo racconta un percorso didattico maturato attraverso laboratori di coding e robotica educativa realizzati in diversi ordini di scuola, dall'infanzia alla secondaria di primo grado. L'esperienza è stata progressivamente riorganizzata come itinerario verticale: dalla programmazione del corpo e del "robot umano" alla rappresentazione simbolica dei comandi, alla programmazione tangibile con Bee-Bot, fino all'uso di mBot e mBlock con sensori. Il nucleo disciplinare non è l'impiego dei robot in sé, ma la comprensione di concetti propri dell'Informatica: algoritmo, esecutore, istruzione formale, programma, strutture di controllo, variabili, esecuzione e debugging. Narrazione, orientamento spaziale e lavoro cooperativo costituiscono invece mediatori multidisciplinari e trasversali, funzionali all'apprendimento ma distinti dai contenuti informatici.

1 Contesto e origine dell'esperienza

Il percorso nasce dall'esperienza sul campo delle due autrici, maturata in numerosi laboratori di coding e robotica educativa realizzati negli ultimi anni in contesti scolastici differenti. Le attività sono state sperimentate, osservate e progressivamente riviste, fino a costruire una sequenza coerente e trasferibile tra primaria e secondaria di primo grado. Il filo conduttore è una situazione narrativa semplice: un "postino" deve muoversi nella Grande mappa del villaggio per raggiungere luoghi e consegnare messaggi. La storia offre il contesto, ma la domanda didattica è informatica: come descrivere un compito in modo sufficientemente preciso perché un esecutore lo svolga senza interpretare le intenzioni di chi programma?

Questa impostazione risponde alla prospettiva delle nuove Indicazioni nazionali, che distinguono la competenza digitale dall'Informatica e definiscono l'algoritmo come una descrizione precisa e non ambigua, riferita a uno specifico esecutore, di una procedura che può essere automatizzata [1]. Il passaggio da un'attività concreta a un programma eseguibile consente quindi di non ridurre coding e robotica al semplice uso di strumenti tecnologici.

2 Dall'esperienza corporea al programma

La prima fase è unplugged. Un alunno assume il ruolo di "robot umano" e un compagno quello di programmatore. Il robot esegue soltanto i comandi ricevuti: "vai lì" o "gira un po'" non sono istruzioni adeguate, mentre una sequenza di passi definiti e condivisi può essere eseguita. Errori, blocchi e fraintendimenti diventano così materiale di lavoro: il gruppo confronta ciò che voleva ottenere con ciò che l'esecutore ha effettivamente fatto e modifica le istruzioni.

Successivamente le indicazioni verbali vengono sostituite da frecce e carte-comando. La sequenza diventa visibile, può essere letta prima dell'esecuzione, confrontata con altre soluzioni e corretta. Sulla mappa del villaggio il postino deve, per esempio, raggiungere la scuola passando dalla posta e dal panificio. Gli alunni progettano il percorso, lo rappresentano con simboli, ne prevedono l'esito e lo eseguono. In questa fase emergono la scomposizione del compito, la sequenza ordinata, la previsione dell'esecuzione, il debugging e il confronto fra percorsi equivalenti o più efficienti.

Con Bee-Bot la stessa logica viene trasferita a un oggetto programmabile. I comandi impartiti tramite i pulsanti costituiscono le istruzioni della sequenza programmata; la pressione di "Go" separa chiaramente il momento della programmazione da quello dell'esecuzione. La routine didattica resta stabile: osservare l'obiettivo, progettare la sequenza, programmarla, eseguirla, controllare l'esito e, se necessario, correggere il programma. La continuità tra carta e robot aiuta a mantenere l'attenzione sul procedimento e non sull'effetto spettacolare del dispositivo.

Nella secondaria il percorso prosegue con mBot e mBlock. Le missioni introducono una progressione ulteriore: una sequenza di movimenti; una selezione condizionale, nella quale il robot modifica il comportamento se il sensore a ultrasuoni rileva un ostacolo; un'iterazione controllata mediante una variabile contatore. La presenza dei sensori rende concreto il passaggio da un'esecuzione lineare a un comportamento che reagisce a dati provenienti dall'ambiente. Il confronto tra due programmi per la stessa missione permette infine di discutere correttezza, differenze strutturali e adeguatezza della soluzione.

3 Contenuti disciplinari di Informatica

I contenuti di interesse specifico per Informatica sono esplicitamente separati dagli aspetti metodologici e trasversali. Il primo nucleo è il rapporto tra algoritmo ed esecutore: un agente computazionale non interpreta intenzioni implicite, ma esegue istruzioni esplicite secondo regole definite. Il secondo riguarda la rappresentazione dell'algoritmo: dalle parole ai simboli, fino a un linguaggio visuale eseguibile, mettendo in evidenza la differenza tra algoritmo e programma.

Il terzo nucleo comprende le strutture di controllo. Nella primaria si lavora inizialmente sulla sequenza e sulla ripetizione, per arrivare a semplici condizioni e reazioni a eventi; nella secondaria si approfondiscono selezione, iterazione, variabili e controllo del comportamento del programma. Il quarto nucleo è la verifica: prevedere l'esito, seguire l'esecuzione, individuare il punto in cui il comportamento diverge dall'obiettivo e correggere il programma. Il debugging non viene presentato come riparazione occasionale, ma come parte ordinaria del processo di programmazione. Questi temi sono coerenti con gli obiettivi delle Indicazioni nazionali per primaria e secondaria, che richiamano sequenze precise e non ambigue, semplici programmi, selezione, ripetizione, variabili e analisi degli errori [1].

Bee-Bot, mBot, mBlock, carte e mappe sono quindi mediatori ed esecutori; non costituiscono il contenuto disciplinare. L'obiettivo è far comprendere come una procedura ideata dall'uomo possa essere formalizzata, eseguita meccanicamente e verificata.

4 Aspetti multidisciplinari e competenze trasversali

Il percorso attiva anche apprendimenti non specificamente informatici. La mappa reticolare richiama orientamento spaziale, geometria, misura di distanze e angoli; la cornice del postino e del villaggio utilizza narrazione, lessico e costruzione di consegne; il lavoro in gruppo sollecita collaborazione, comunicazione, negoziazione delle soluzioni e gestione dell'errore. Questi aspetti hanno una funzione importante di accesso, motivazione e inclusione, ma non sono presentati come risultati disciplinari di Informatica.

La distinzione è essenziale anche nella progettazione verticale: il contesto narrativo può cambiare, così come il robot utilizzato, mentre rimangono riconoscibili i concetti informatici che

danno continuità al percorso. In questo senso la robotica educativa è un ambiente per rendere osservabile l'esecuzione di un algoritmo e discutere il comportamento di un programma, non un obiettivo autonomo.

5 Esiti e riflessioni dalla pratica

Il lavoro di ricerca-azione svolto nelle classi ha portato soprattutto alla definizione di una progressione didattica stabile: corpo, simboli, programmazione tangibile, linguaggio visuale e sensori. La continuità fra le fasi consente di riprendere lo stesso concetto con un diverso livello di formalizzazione, evitando che ogni attività di coding resti un episodio isolato. Un secondo esito è la centralità dell'errore come evidenza osservabile: quando il postino o il robot non raggiungono la destinazione, l'attenzione può essere ricondotta alla sequenza di istruzioni e alla loro esecuzione, rendendo possibile una revisione argomentata del programma.

L'esperienza suggerisce inoltre che la continuità verticale non dipende dall'adozione dello stesso strumento, ma dalla coerenza dei nuclei concettuali. Dalla primaria alla secondaria aumenta la complessità dell'esecutore e del linguaggio, mentre restano centrali progettazione, formalizzazione, esecuzione e verifica. È questa continuità concettuale che permette di presentare l'Informatica come disciplina scientifica autonoma e non come semplice familiarità con dispositivi digitali.

Riferimenti

[1] Ministero dell'Istruzione e del Merito. Indicazioni nazionali per il curricolo della scuola dell'infanzia e delle scuole del primo ciclo di istruzione. Versione definitiva 9 dicembre 2025; pubblicazione in Gazzetta Ufficiale 27 gennaio 2026.

Ragazze Robotiche

Lucia Salari[1]
[1] Istituto Comprensivo Pescara 7
Lucia.salari@icpescarasette.edu.it

Abstract

L'Istituto Comprensivo Pescara 7 ha avviato un progetto di ricerca-azione e potenziamento didattico volto a contrastare il gender gap nelle discipline STEM attraverso la costituzione di un team di robotica interamente femminile. A partire dalla partecipazione alla competizione internazionale Robocup Junior Academy, il percorso ha declinato gli obiettivi di apprendimento e i traguardi per lo sviluppo delle competenze previsti dalle Nuove Indicazioni Nazionali del 2026 per l'ambito matematico-informatico sviluppate attraverso metodologie di peer tutoring, didattica laboratoriale e problem solving.
I risultati del triennio evidenziano non solo il raggiungimento di elevate competenze tecniche e trasversali, ma confermano anche un significativo impatto in chiave orientativa: il 25% delle partecipanti in uscita ha ridefinito la propria scelta verso una scuola secondaria di secondo grado a indirizzo scientifico-tecnologico. Il consolidamento del progetto prevede l'istituzione di una filiera formativa territoriale virtuosa mediante un protocollo d'intesa con l'I.I.S. "A. Volta" e l'azienda Fameccanica.

1. Introduzione

L'Istituto Comprensivo Pescara 7 ha valorizzato negli anni l'insegnamento del coding e della robotica educativa avviando un percorso a classi aperte rivolto esclusivamente alle studentesse, con le risorse di una cattedra di potenziamento di matematica.

L'obiettivo è colmare il divario di genere nelle discipline STEM, costituendo un team femminile dedicato alle competizioni della rete Robocup Junior Academy

2. Linee generali del progetto

Il percorso si sviluppa lungo il triennio della scuola secondaria di primo grado. I consigli di classe individuano le alunne delle classi prime con un profilo avanzato nelle discipline scientifiche a cui vengono proposte attività di socializzazione e introduzione alla robotica, affiancate dalle studentesse degli anni precedenti (peer tutoring).
Nei due anni successivi il team si riunisce settimanalmente in orario extracurricolare per dedicarsi alla progettazione, costruzione e programmazione del robot da gara.
Il percorso prevede anche laboratori pratici presso l'I.I.S. "A. Volta" di Pescara, scuola capofila della rete Robocup Abruzzo, che offre supporto tecnico e l'uso dei campi gara ufficiali.
Il percorso culmina ogni anno con la partecipazione alle gare regionali e, qualora qualificati, nazionali.

3. Traguardi per lo sviluppo delle competenze

Questo progetto contribuisce a sviluppare alcune competenze attese relative all'informatica che appaiono sulle Nuove Indicazioni Nazionali del 2026 quali:

- Comprendere il concetto di variabile;
- Scrivere, con linguaggi di programmazione a scopi didattici, programmi che usano le variabili anche strutturati in componenti modulari,
- Valutare la correttezza dei programmi

Accanto alle competenze specifiche relative all'informatiche, si assiste al naturale svilupparsi di competenze trasversali quali:

- Imparare a imparare: Gestione del proprio tempo, consapevolezza del metodo di studio e organizzazione del lavoro.
- Collaborazione: Saper lavorare in gruppo, rispettare le diversità e contribuire al raggiungimento di un obiettivo comune.
- Pensiero critico e creativo: Analizzare informazioni, risolvere problemi (problem solving) e formulare soluzioni originali.
- Autonomia e sviluppo personale: Acquisire sicurezza in sé stessi, gestire lo stress ed essere propositivo.

4. Metodologie e curricolo

L'insegnamento dell'informatica all'interno del progetto non è inteso come semplice addestramento all'uso di un software, ma come padronanza del Pensiero Computazionale (astrazione, scomposizione, riconoscimento di pattern e progettazione algoritmica).
La progressione dei contenuti si sviluppa su due linee, l'analisi della struttura del robot e la programmazione del robot stesso.
La categoria Rescue Line Under 14 funge da organizzatore concettuale del curriculo.
Essa impone al robot autonomo di seguire una linea nera, superare ostacoli, rilevare marcatori colorati, negoziare salite/discese e pianificare manovre in una stanza di soccorso. Questo impianto di problemi concreti guida la progressione dei concetti informatici:

Fase	**Concetto informatico**	**applicazione**	**Ambiente di programmazione**
1	Strutture sequenziali, Cicli indefiniti (while/loop), Istruzioni condizionali semplici (if-else).	Seguilinea Base: Il robot legge bianco/nero e corregge la traiettoria sterzando nettamente.	Makeblock Scratch LEGO Spike Prime (Block-based)
2	Variabili, Operatori logici/matematici, Scomposizione modulare (Funzioni e Procedure con parametri).	Gestione Ostacoli e Bivi: Scomposizione del problema in sottoprogrammi (Es. Supera_Ostacolo(), Gira_A_Destra()).	LEGO Spike Prime (Block-based)
3	Algoritmi di feedback continuo, Controllo Proporzionale, Gestione dello Stato.	Seguilinea Proporzionale	LEGO Spike Prime (Block-based)

Tabella 1: Progressione dei contenuti

La programmazione ha dunque come fulcro il programma "seguilinea", che viene introdotto per approssimazioni successive. Nella figura 1 si mostra un esempio di un programma seguilinea iniziale, che come si può vedere viene costruito analiticamente in tutte le sue parti. Questo consente da un lato di comprendere il comportamento del robot in un programma seguilinea e allo stesso tempo iniziare a familiarizzare con le principali istruzioni di tipo condizionale.

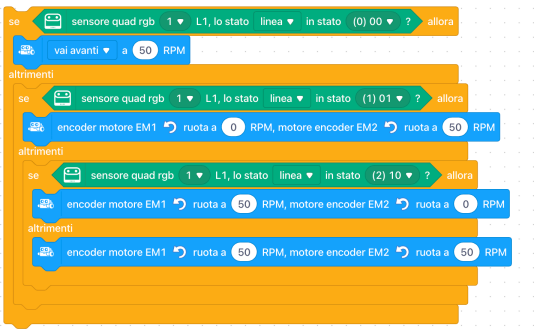


Figura 1: Esempio di programma iniziale

In seguito nel passaggio a programmi seguilinea sempre più raffinati si introduce progressivamente l'uso di operatori, variabili, funzioni e procedure. Queste ultime si approfondiscono in particolare quando vengono proposti, come problemi aperti, i diversi task da superare nei campi gara: ostacoli fissi, gap, bumpers e salite e discese.

Nella prima fase per poter sperimentare il seguilinea base si utilizzano gli M-bot di Makeblock già assemblati e con un focus particolare sul funzionamento dei sensori a luce riflessa.
Poi ancora in questa fase si passa all'uso dei Kit Lego Spike. Si procede ad assemblare da zero la motrice base di Lego Spike attraverso una costruzione guidata.
Nelle fasi successive il robot viene gradualmente migliorato ed arricchito dalle ragazze stesse attraverso le osservazioni delle performance sul campo e delle esigenze che si presentano sul campo gara (es. aggiungere un sensore ad infrarossi per evitare gli ostacoli, modificare il peso per affrontare meglio i bumpers, ecc)

Nello svilupparsi del progetto le ragazze sperimentano alcuni capisaldi del lavoro dell'ingegneria informatica quali il lavoro in team e il metodo per tentativi ed errori. La maggior parte dei task viene proposto come problema aperto e questo garantisce lo sviluppo di soluzioni creative e originali.

5. Evoluzione del Progetto e Impatto Orientativo

Il progetto ha preso avvio nell'A.S. 2023-24 con un gruppo di sei ragazze scelte da ogni classe prima della scuola e si è sviluppato nel corso di tre anni con l'ingresso nel corso del tempo di altre due ragazze. E' stata proposta alle alunne un'attività extracurricolare con cadenza settimanale ponendo da subito come obiettivo principale prepararsi alle competizioni della Robocup Junior Academy.



Figura 1: Team iniziale, maggio 2024

La cornice competitiva in cui è stato proposto il progetto ha incentivato le ragazze ad avvicinarsi ad argomenti per quasi tutte loro sconosciuti.
Ogni anno sono stati svolti laboratori di allenamento presso i campi gara dell'Istituto Volta, momenti fondamentali per sperimentare da vicino i diversi tipi di percorsi, che a scuola avevano sempre riprodotto in modo artigianale. Ancora più importante in questi appuntamenti è stato il tutoraggio di ragazzi poco più grandi di loro da cui sono state guidate e consigliate.

La partecipazione alle gare regionali è sempre stato un momento molto significativo, poiché nel contesto della gara si è sempre registrato un intensificarsi dell'apprendimento e una crescita della passione e dell'interesse per le tematiche affrontate.



Figura 3: Gare regionali A.S. 2024-25, marzo 2025

Nel corrente A.S. 2025-26, la partecipazione alle finali nazionali RoboCup Junior ha rappresentato il test di verifica sul campo: le studentesse hanno gestito le competizioni in totale autonomia, senza il supporto diretto della docente (come stabilito dal regolamento RoboCup). In situazione di gara il team ha affrontato numerosi imprevisti tecnici dimostrando elevate competenze di problem solving. Il contesto nazionale ha inoltre evidenziato il valore del team, mosca bianca in un panorama a forte prevalenza maschile, sul fronte del divario di genere.





Figure 4-5: Gare nazionali A.S. 2025-26, aprile 2026, Catania

Nel corso del triennio le ragazze del team di robotica sono state coinvolte , grazie alle competenze sviluppate in una serie di attività collaterali dell'Istituto come gli Open Day, le attività Erasmus, attività di continuità con studenti della Scuola Primaria e dell'Infanzia ed eventi sul territorio riguardanti le STEM, in cui veniva sempre chiamate a presentare un laboratorio di robotica e a raccontare la loro esperienza.



Figura 6: Partecipazione a Pescara Futura, aprile 2025

Negli ultimi mesi del secondo quadrimestre, al ritorno dalle gare nazionali, le ragazze del team, hanno preso parte ad una serie di attività di tutoraggio al gruppo di 12 nuove ragazze delle classi prime e seconde selezionate per comporre il nuovo team.

Le principali proposte sono state un laboratorio di introduzione alla robotica presso l'Istituto Volta e una visita presso l'azienda Fameccanica, presente sul nostro territorio, con cui stiamo istituendo un protocollo di intesa pre creare, insieme all'Istituto Volta con cui già collaboriamo, una filiera formativa virtuosa che accompagni le ragazze a fare scelte per il loro futuro sempre più concrete ed efficaci.



Figura 7: Visita a Fameccanica, giugno 2026

Il monitoraggio a lungo termine ha rivelato nel tempo anche un'importante dimensione orientativa, infatti il 25% delle studentesse del team uscente (2 su 8) ha modificato la propria scelta della scuola superiore a favore di percorsi STEM proprio in virtù di questa esperienza.

6. Valutazione

L’aspetto valutativo in questa prima sperimentazione triennale sia per quanto riguarda gli apprendimenti che il monitoraggio generale del progetto, si è limitato all’uso di strumenti non strutturati, quali l’osservazione delle ragazze da parte della docente e colloqui informali, sia individuali che di gruppo, con le ragazze del team.

Per il futuro si prevede di dotarsi di una metodologia di valutazione più rigorosa adottando un disegno di ricerca-azione misto (qualitativo-quantitativo) sul triennio e procedendo poi ad una elaborazione e triangolazione dei dati raccolti.

Gli strumenti che ci si propone di usare sono i seguenti:

- Pre-test e Post-test sulle Competenze Computazionali utilizzando dei test standardizzati
- Rubriche di valutazione delle competenze e griglie di osservazione con indicatori osservabili articolati su 4 dimensioni: Accuratezza Algoritmica, Autonomia nel Debugging, Lavoro di Squadra, Resilienza all’Errore.
- Diari di Bordo delle Studentesse e Log di Gara come strumento qualitativo di autovalutazione in cui le ragazze documentano, al termine di ogni sessione, le problematiche incontrate, le ipotesi formulate, i test effettuati e le soluzioni trovate
- Interviste Semi-strutturate sull'Orientamento.

7. Conclusioni

Il percorso sviluppato dall'Istituto Comprensivo Pescara 7 dimostra come la robotica educativa e il coding, inseriti in un contesto cooperativo e altamente stimolante come quello agonistico, rappresentino dispositivi didattici formidabili per l'abbattimento degli stereotipi di genere nelle discipline scientifiche. La transizione da un approccio puramente teorico e astratto della matematica a una sua applicazione ingegneristica e situata ha permesso alle studentesse di sviluppare non solo solide competenze computazionali, ma anche una profonda consapevolezza delle proprie capacità logiche e progettuali.
L'impatto del progetto supera i confini del mero successo scolastico o competitivo. L'autonomia dimostrata dalle alunne durante le finali nazionali e l'efficacia dell'attività di peer tutoring nel passaggio di testimone intergenerazionale testimoniano la sostenibilità interna del modello. Sul piano dell'orientamento, la modifica delle scelte della scuola secondaria di secondo grado da parte del 25% delle partecipanti in uscita conferma che un'esposizione precoce, continuativa e protetta alle tecnologie può deviare traiettorie formative tradizionalmente influenzate da bias culturali.
La prospettiva futura del progetto risiede nel consolidamento della filiera formativa territoriale. La formalizzazione del protocollo d'intesa con l'I.I.S. "A. Volta" e l'azienda Fameccanica si configura come un modello esportabile di sinergia scuola-impresa-territorio, volto a garantire alle studentesse una continuità di studi e di visioni professionali nel settore delle STEM, trasformando un'esperienza sperimentale in un curricolo verticale strutturato e inclusivo.

Bibliografia

[1] MIM 2026, Nuove Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d’istruzione. Disponibili all’indirizzo: https://www.mim.gov.it/documents/20182/10554370/curricolo_web.pdf/f91c31a0-5ed4-65f3-bfea-fb49adaba55f?version=1.0&t=1773224873548

[2] MIM 2023, Linee guida per le discipline STEM (Decreto Ministeriale 184/2023).

[3] Raccomandazione del Consiglio dell'Unione Europea (2018): Raccomandazione relativa alle competenze chiave per l'apprendimento permanente (2018/C 189/01)

[4] UNESCO (2017). Cracking the code: Girls' and women's education in science, technology, engineering and mathematics (STEM).

Verso la programmazione orientata agli oggetti: un percorso introduttivo con App Inventor nella scuola secondaria.

Ersilia Pagano
IIS Nitti Portici
ersilia.pagano@gmail.com

Abstract

Il contributo descrive un percorso sperimentato nel biennio della scuola secondaria di secondo grado finalizzato all'introduzione di alcuni concetti propedeutici alla programmazione orientata agli oggetti attraverso App Inventor. L'esperienza nasce dalla necessità di affrontare precocemente alcuni nuclei fondanti dell'informatica riducendo il carico cognitivo associato alla sintassi dei linguaggi testuali. Attraverso la progettazione di applicazioni mobili, gli studenti esplorano oggetti software, proprietà e stato, metodi, comportamento ed eventi. Il percorso non anticipa l'intero paradigma OOP [4], ma costruisce concetti propedeutici che vengono successivamente formalizzati nello studio di Java.

1 Introduzione

La programmazione orientata agli oggetti costituisce uno dei paradigmi fondamentali dell'informatica contemporanea. Tuttavia, la sua introduzione nella scuola secondaria presenta spesso difficoltà significative.

Nei percorsi tradizionali gli studenti incontrano contemporaneamente nuovi concetti informatici e nuove strutture sintattiche: la necessità di padroneggiare le regole formali del linguaggio può sottrarre attenzione alla costruzione dei modelli concettuali. Gli ambienti di programmazione a blocchi, tra cui App Inventor, sono stati studiati come strumenti capaci di sostenere l'apprendimento iniziale della programmazione riducendo parte delle difficoltà legate alla sintassi [2].

Da questa osservazione nasce la domanda che ha guidato l'esperienza descritta in questo contributo: è possibile introdurre i concetti propedeutici alla programmazione orientata agli oggetti prima dell'apprendimento di un linguaggio testuale?

Per esplorare questa possibilità è stato utilizzato App Inventor, ambiente di programmazione visuale del MIT [3] nel quale gli studenti progettano applicazioni mobili mediante componenti grafici e blocchi di programmazione.

L'ipotesi didattica alla base del percorso è che la natura visuale dell'ambiente possa facilitare la costruzione di una prima rappresentazione mentale di alcuni concetti propedeutici alla programmazione orientata agli oggetti, consentendo agli studenti di concentrarsi sugli aspetti concettuali prima di affrontare la complessità sintattica dei linguaggi tradizionali.

2 Riferimenti curriculari e aspetti disciplinari

L'esperienza si colloca nel quadro della Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica elaborata dal Laboratorio Informatica e Scuola del CINI [1], che individua tra gli obiettivi del biennio della scuola secondaria di secondo grado lo sviluppo della capacità di

modellare problemi, progettare algoritmi, rappresentare e organizzare dati e realizzare programmi per la soluzione di problemi.

I contenuti disciplinari affrontati riguardano esclusivamente temi propri dell'informatica:

- introduzione ad alcuni concetti propedeutici alla programmazione orientata agli oggetti;
- proprietà e stato degli oggetti;
- metodi e comportamento degli oggetti;
- eventi e programmazione event-driven;
- interazione tra oggetti software;
- progettazione di semplici applicazioni.

In App Inventor gli studenti non lavorano direttamente con la definizione di classi e attributi come nei linguaggi orientati agli oggetti tradizionali; la presenza di componenti dotati di proprietà, comportamenti ed eventi consente tuttavia una prima esplorazione di concetti che vengono successivamente ripresi e formalizzati nello studio di Java.

3 Il percorso didattico

L'esperienza è stata realizzata nel primo anno del percorso quadriennale dell'indirizzo Informatica e Telecomunicazioni, articolazione Informatica, in una classe composta da studenti al primo approccio formale alla programmazione.

Il percorso si è sviluppato nell'arco di circa sei settimane, nell'ultima parte dell'anno scolastico, integrandosi nelle ordinarie attività curricolari di Informatica. Le attività sono state organizzate con difficoltà crescente, passando dall'analisi dei componenti e delle loro proprietà alla programmazione degli eventi e alla progettazione di applicazioni.
Nelle prime attività gli studenti erano concentrati sul funzionamento dell'applicazione. Il lavoro è stato quindi impostato spostando progressivamente l'attenzione dai prodotti realizzati agli elementi che li compongono. Pulsanti, etichette, immagini e caselle di testo sono diventati l'occasione per introdurre il concetto di oggetto software e riflettere sulle sue caratteristiche.
Una prima consegna richiedeva di modificare proprietà facilmente osservabili, come colore, dimensione, testo e visibilità, e di descrivere l'effetto prodotto. Il confronto tra stato iniziale e stato successivo alla modifica ha consentito di introdurre il concetto di stato. Successivamente sono stati introdotti i metodi e gli eventi, osservando come un'azione dell'utente potesse attivare una risposta e modificare lo stato di uno o più componenti.

Per ciascun componente gli studenti hanno costruito una scheda con proprietà, metodi ed eventi, osservandone direttamente le relazioni. La scheda ha accompagnato l'analisi delle applicazioni e ha reso esplicito il legame tra caratteristiche del componente, azioni disponibili ed eventi che ne attivano il comportamento.

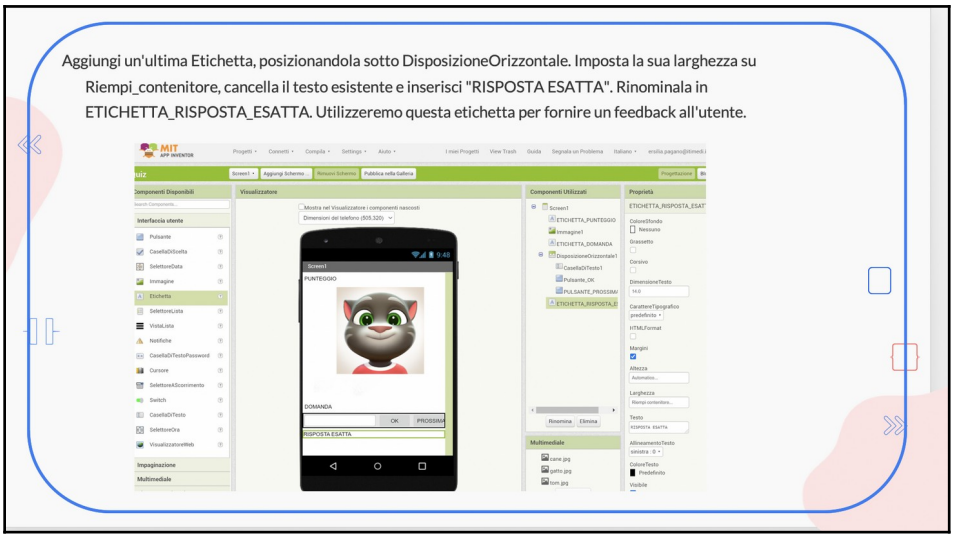


Figura 1: Slide utilizzata durante le attività didattiche: realizzazione di un quiz interattivo con App Inventor.

Per consolidare i concetti sono state realizzate diverse applicazioni mobili. Nel quiz interattivo gli studenti hanno riconosciuto i pulsanti come componenti che generano eventi e le etichette come componenti deputati alla visualizzazione delle informazioni, modificandone dinamicamente le proprietà attraverso l'evento Click. Una seconda consegna ha riguardato un'applicazione di raccolta dati composta da TextBox, Button e Label: l'evento Click del pulsante acquisiva l'input e aggiornava il contenuto della Label, rendendo osservabile la relazione tra evento, azione e modifica dello stato.

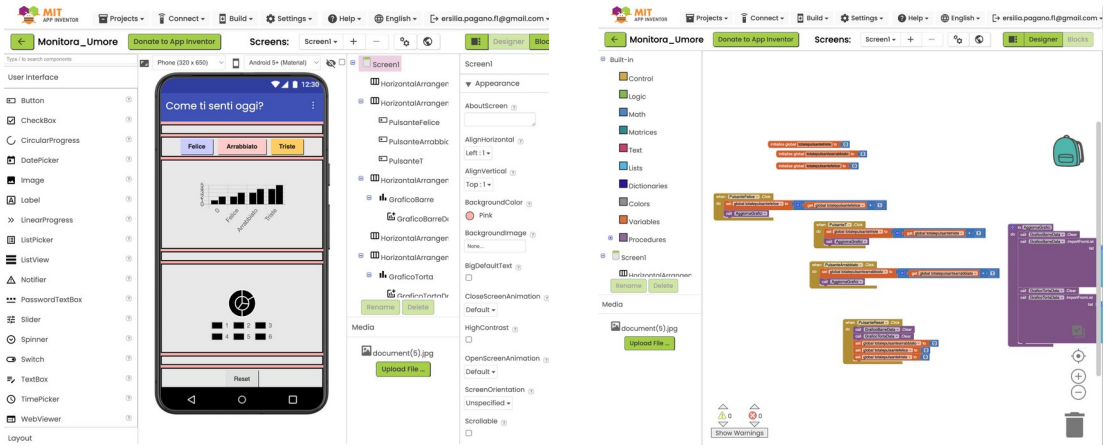


Figura 2: Applicazione di raccolta dati e relativa implementazione a blocchi in App Inventor

Oltre agli esempi descritti sono state sviluppate ulteriori applicazioni con difficoltà crescente. In tutte le attività l'attenzione è stata posta non soltanto sul prodotto finale, ma sulla capacità di descrivere il funzionamento dell'applicazione in termini di componenti, stato, proprietà, metodi ed eventi e di utilizzare tali concetti per progettare il comportamento desiderato.

4 Evidenze e risultati osservati

Le evidenze derivano dalle osservazioni svolte durante le attività laboratoriali, dall'analisi delle applicazioni sviluppate e dalle discussioni guidate con gli studenti. La valutazione ha avuto carattere prevalentemente formativo e ha considerato, in particolare, la capacità di riconoscere il ruolo dei componenti, descriverne proprietà e comportamento, individuare la relazione tra evento e azione e motivare le scelte effettuate nella progettazione dell'applicazione. Si tratta di evidenze qualitative e osservative, riferite a una singola esperienza di classe, che non consentono di attribuire all'approccio un'efficacia generalizzabile.

Nelle prime attività gli studenti tendevano a descrivere le applicazioni principalmente in termini di funzionalità ("serve un pulsante", "deve comparire una scritta", "deve fare un calcolo"). Con il procedere del percorso hanno iniziato progressivamente a utilizzare termini quali oggetto, proprietà, metodo ed evento per spiegare il comportamento delle applicazioni realizzate. Questo cambiamento del linguaggio non è stato considerato, da solo, prova di comprensione: particolare attenzione è stata posta alla capacità di spiegare quale componente cambiasse stato, quale proprietà venisse modificata, quale evento attivasse l'azione e quale effetto fosse prodotto sull'applicazione.

Un aspetto particolarmente interessante è stato osservare come la rappresentazione visuale dei componenti aiutasse gli studenti a comprendere la relazione tra stato e comportamento degli oggetti. La possibilità di modificare una proprietà, come il testo o il colore di un componente, e osservarne immediatamente l'effetto rendeva concreto un concetto che nelle prime esperienze con linguaggi testuali risulta spesso più astratto.

Anche il ruolo degli eventi è stato generalmente compreso con facilità. Gli studenti riuscivano a collegare in modo naturale il comportamento dell'applicazione alle azioni compiute sull'interfaccia, interpretando il clic su un pulsante o l'inserimento di un dato come elementi in grado di attivare specifiche risposte del sistema.

Un ulteriore elemento significativo ha riguardato la riduzione del carico cognitivo associato alla sintassi. La riduzione di errori formali legati alla scrittura del codice ha consentito agli studenti di concentrarsi maggiormente sulla logica dell'applicazione e sulle relazioni tra i diversi oggetti coinvolti. In diverse attività la discussione si è spostata spontaneamente dal "come si scrive" al "come funziona", favorendo una riflessione più approfondita sugli aspetti concettuali della programmazione.

L'esperienza è proseguita nel successivo insegnamento di Java, nel quale sono stati ripresi e formalizzati i concetti di oggetto, proprietà, metodo ed evento già affrontati attraverso App Inventor. Ciò ha consentito di concentrare maggiormente l'attenzione sugli aspetti sintattici del linguaggio. App Inventor ha quindi svolto la funzione di ambiente ponte verso l'apprendimento di Java, favorendo una transizione graduale tra programmazione visuale e linguaggi testuali.

Il percorso non si propone di affrontare l'intero paradigma orientato agli oggetti. Concetti avanzati quali ereditarietà, polimorfismo e incapsulamento non costituiscono obiettivi dell'esperienza. L'attenzione è focalizzata sull'introduzione dei concetti di oggetto, stato, proprietà, metodo ed evento, ritenuti particolarmente adatti a una prima formalizzazione nel biennio della scuola secondaria.

5 Discussione e conclusioni

L'esperienza descritta non propone App Inventor come alternativa ai linguaggi di programmazione tradizionali, ma come ambiente introduttivo per la costruzione di alcuni concetti propedeutici alla programmazione orientata agli oggetti.

I risultati osservati suggeriscono che la manipolazione diretta di componenti software possa favorire una prima costruzione di tali concetti, orientando progressivamente l'attenzione degli studenti dagli aspetti sintattici alla comprensione del funzionamento delle applicazioni.

In questa prospettiva, App Inventor rappresenta un contesto didattico nel quale gli studenti possono costruire progressivamente il significato di oggetto, stato, proprietà, metodo ed evento. Il successivo passaggio a Java permette di riprendere e formalizzare tali concetti, introducendo progressivamente gli elementi propri del paradigma orientato agli oggetti.

L'esperienza suggerisce interessanti prospettive per la costruzione di percorsi di informatica nei quali gli ambienti visuali possano essere utilizzati come ponte verso l'apprendimento successivo dei linguaggi testuali e di paradigmi di programmazione più avanzati.

Bibliografia

[1] Gruppo di Lavoro CINI "Informatica e Scuola". *Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola*. Laboratorio Informatica e Scuola – CINI, 2017.

[2] Hu, Y., Chen, C.-H., Su, C.-Y. *Exploring the Effectiveness and Moderators of Block-Based Visual Programming on Student Learning: A Meta-Analysis*. Journal of Educational Computing Research, 2021.

[3] MIT App Inventor. *MIT App Inventor Documentation and Tutorials*.

[4] Wegner, P. *Concepts and Paradigms of Object-oriented Programming*. ACM SIGPLAN OOPS Messenger, 1990.

L'Odissea: una Stochastic Gradient Descent omerica?

Alla ricerca di nuove piste nell'insegnamento dell'Informatica

Emanuele Scapin[1], Furio Honsell[2] e Alessandra Chilese[1]

[1] ITT G.Chilesotti, Thiene (VI), Italia
escapin@chilesotti.it,achilese@chilesotti.it
[2] Università di Udine, Udine, Italia
furio.honsell@uniud.it

Abstract

Negli ultimi anni l'Intelligenza Artificiale generativa sta assumendo un ruolo sempre più rilevante nei contesti educativi, modificando le modalità con cui studenti e docenti interagiscono con il sapere disciplinare. Inevitabilmente si deve riflettere su come modificare le pratiche di insegnamento, non limitandosi a introdurre nuovi strumenti, ma interrogandosi su quali competenze debbano essere sviluppate, con quali strategie didattiche e secondo quali criteri di valutazione. La Letteratura – affrontando autori italiani e stranieri – può diventare utile nell'apprendimento dell'Informatica, sia per presentare concetti tradizionalmente legati al pensiero computazionale, ma anche per indurre studentesse e studenti a lavorare maggiormente sull'analisi e valutazione dei testi.

1 Introduzione

Negli ultimi anni l'Intelligenza Artificiale (IA) e soprattutto i modelli di l'IA generativa (gIA) stanno assumendo un ruolo sempre più rilevante nei contesti educativi, modificando profondamente le modalità con cui studenti e docenti interagiscono con il sapere disciplinare. È dunque urgente riflettere su come modificare le pratiche di insegnamento, poiché la gIA non costituisce soltanto un potenziale nuovo contenuto curricolare, ma fornisce nuovi strumenti che intervengono direttamente nei processi di problem solving, produzione di codice, feedback e valutazione [3, 6]. Il quadro pedagogico di riferimento va dunque profondamente ripensato non limitandosi a sfruttare tali nuovi strumenti, ma interrogandosi su quali competenze debbano essere sviluppate, con quali strategie didattiche e secondo quali criteri di valutazione.

Nardelli [1] analizza un report UNESCO [2] che denuncia come l'eccesso di tecnologia nell'istruzione comprometta apprendimento e capacità cognitive. I rischi dell'uso dell'IA da parte degli studenti sono stati anche evidenziati presso il Senato USA. [3] Horvath [4] sostiene che negli ultimi decenni alcune capacità cognitive degli studenti, come lettura, calcolo matematico, attenzione e ragionamento, si sono ridotte proprio mentre aumentava l'uso di schermi e strumenti digitali in classe. Inoltre Horvath sostiene che vi sia una correlazione causale tra i due fenomeni perché la comprensione e la memorizzazione sono migliori su carta, soprattutto per testi lunghi o complessi, inoltre prendere appunti a mano è più efficace che scrivere al computer, perché costringe a sintetizzare e riorganizzare le informazioni, mentre digitare favorisce la trascrizione passiva.

Lo stesso Pontefice Leone XIV nella Lettera Enciclica "MAGNIFICA HUMANITAS" [5] scrive che "[l]o sviluppo delle tecnologie informatiche e dell'IA rende rapidamente inadeguati programmi di studio pensati per un'altra epoca", ciò richiede agli insegnanti uno sforzo per aiutare "gli studenti a farne un

[1] https://www.startmag.it/innovazione/strumenti-digitali-e-apprendimento-platone-vince-ancora/
[2] https://www.unesco.org/en/digital-education/ed-tech-tragedy
[3] Comunicato del Senate Committee on Commerce
[4] Testimonianza scritta di J.C. Horvath
[5] Lettera Enciclica MAGNIFICA HUMANITAS.

uso responsabile, critico e creativo e a non subirne passivamente l'influsso". Inoltre "[S]i moltiplicano conoscenze frammentarie, ma diventa più difficile cogliere la realtà nel suo insieme, porre domande di senso, sviluppare un autentico pensiero critico e creativo".

La nostra proposta per mitigare i rischi e amplificare i benefici della gIA nasce dalla considerazione, quasi paradossale, che l'elaborazione e comprensione di testi informali o pre-formali, ma potenzialmente formalizzabili, diventi cruciale nell'interazione con la gIA, si pensi solamente ai `prompt`. A tale scopo riteniamo molto efficace sviluppare un approccio multi-disciplinare all'IA stabilendo nessi inter-disciplinari precisi, ma aperti (open-ended) tra opere letterarie, sia classiche che sperimentali, evidenziandone le immagini mentali e le strutture tipiche del pensiero computazionale in essi presenti. Ciò è mutuamente efficace nell'insegnamento di entrambe le discipline, ma per l'IA avrebbe il pregio di de-enfatizzare la prevalenza del Coding spostando il focus sugli aspetti di descrizione e analisi dei problemi e valutazione dei risultati. Ciò è in linea anche con recenti lavori.

2 Il contesto di riferimento

Un primo elemento consolidato in letteratura riguarda la crescente centralità dell'*AI literacy* [20, 5]. In questa prospettiva, l'alfabetizzazione all'IA non coincide con la sola capacità di utilizzare strumenti intelligenti, ma dovrebbe includere la comprensione dei meccanismi di funzionamento, la consapevolezza delle implicazioni etiche e sociali, nonché la capacità di valutare criticamente output, bias e affidabilità dei sistemi.

Al riguardo riteniamo cruciale individuare lo spartiacque tra informatica tradizionale e gIA. La prima è figlia dell'atomismo logico di Frege e Russell, ovvero della grande tradizione riduzionista in filosofia di *e.g.* Platone, Cartesio, Leibniz, Husserl e del primo Wittgenstein. Si basa sulla logica, sugli algoritmi, e permette la composizionalità. La seconda invece si può far risalire all'esistenzialismo di Heidegger e alla filosofia analitica del secondo Wittgenstein. Come sostiene Heidegger del martello[14]§15, o Wittgenstein della scopa[30] §60: non ci interessa sapere come siano fatti, se non quando si rompono, ma ci interessa usarli. La gIA impiega euristiche e statistica e non favorisce la composizionalità perché olistica; addestrata una rete neurale a riconoscere i cani dai gatti non possiamo che riaddestrala da zero se vogliamo riconoscere mucche da pecore, perché non possiamo decomporre la prima rete neurale individuando quali parti riconoscono quali caratteristiche. Una cosa è costruire una mente (IA algoritmica) altra cosa è modellare il cervello (gIA)[9].

Un secondo filone di ricerca riguarda l'integrazione dell'IA nella didattica dell'Informatica [22]. Tuttavia, tale opportunità si accompagna a un cambiamento più profondo: cresce l'importanza di competenze quali la lettura e comprensione del codice, la valutazione critica di soluzioni proposte da sistemi intelligenti, la capacità di analizzare un problema e formulare richieste efficaci e la riflessione metacognitiva sui processi adottati [27].

Le ricerche svolte nei contesti scolastici K–12 confermano che l'introduzione dell'IA può essere didatticamente efficace quando viene collocata entro attività autentiche, graduali e fortemente situate [21]. Lo studio pilota triennale di Lišnić, Zaharija e Mladenović mostra che l'IA può essere integrata nell'insegnamento dell'Informatica senza aumentare il monte ore curricolare, purché si adottino materiali adeguati, strumenti accessibili, forte supporto ai docenti e una progettazione capace di collegare i concetti di IA ai fondamenti della programmazione [19]. Allo stesso tempo, gli autori segnalano criticità significative: difficoltà degli studenti nel comprendere concetti più astratti, forte dipendenza dalla qualità dei dataset, differenze negli esiti dovute alla preparazione dei docenti e minore efficacia delle attività svolte a distanza.

Accanto alle opportunità, la letteratura segnala con forza anche rischi e tensioni. Un tema ricorrente è quello dell'*overreliance*: l'uso non regolato dell'IA può migliorare la performance immediata dello studente, ma ridurre l'apprendimento autonomo e la capacità di trasferimento delle competenze [2]. In

modo analogo, l'uso di sistemi generativi senza adeguate mediazioni può indebolire il pensiero critico, favorire dipendenza dagli strumenti, aumentare le disuguaglianze fra studenti con diverso accesso o differente familiarità con tali tecnologie, e introdurre bias o errori difficili da riconoscere soprattutto per i novizi [1, 3].

Queste trasformazioni hanno implicazioni dirette anche per la valutazione. Se gli strumenti di IA sono in grado di produrre testi, codice e soluzioni plausibili, la tradizionale enfasi sul prodotto finale rischia di diventare insufficiente per inferire il reale apprendimento dello studente [3]. La letteratura suggerisce pertanto di valorizzare maggiormente il processo: discussione delle scelte, spiegazione orale del codice, capacità di esprimere requisiti in modo chiaro, analisi di errori, revisione di soluzioni, progettazione di casi di test, riflessione metacognitiva e attività in presenza o in ambienti controllati [3, 10, 27].

Infine, il dibattito educativo si intreccia con le trasformazioni del mondo del lavoro. Le ricerche sulle professioni IT mostrano che l'adozione dell'IA sta ridefinendo ruoli, task e competenze richieste ai futuri professionisti, rendendo centrali non solo conoscenze tecniche, ma anche capacità di collaborazione, valutazione critica, apprendimento continuo, consapevolezza etica e gestione dell'incertezza [6].

In sintesi l'introduzione dell'IA nell'insegnamento dell'Informatica configura una trasformazione curricolare, pedagogica e valutativa, che richiede un ripensamento integrato di contenuti, metodi, strumenti, criteri di successo e professionalità docente. In questa prospettiva, appare essenziale sviluppare modelli didattici che sappiano valorizzare le potenzialità dell'IA senza rinunciare agli obiettivi fondamentali della formazione informatica: comprensione, autonomia, riflessività e responsabilità.

3 Una "modesta proposta": il confronto con i testi letterari.

Riteniamo che il cambiamento di paradigma nella didattica dell'IA, necessario per sviluppare competenze di comprensione, analisi e interpretazione, anche pre-formale, debba passare attraverso il confronto con testi letterari alla ricerca delle loro immagini e delle loro costruzioni formali, ricorsive, combinatorie o procedurali. Un esempio è già stato dato nella Sezione 2 parlando di scope e martelli. Ma molto più in generale, a partire da Dante e da Shakespeare, che adoperarono grandiose metafore computazionali, passando all'Oulipo e alla letteratura potenziale *à contrainte* di Raymond Queneau e Georges Perec, e quindi a Italo Calvino, fino ai paradossi letterari di Jorge Luis Borges, e ai numerosi autori che scelgono di superare l'approccio lineare al testo quali James Joyce, Julio Cortázar, Milorad Pavić. Molteplici sono gli scrittori le cui opere si prestano, nella forma o nel contenuto, ad un'analisi da una prospettiva informatica: Edgar Allan Poe, Franz Kafka, Fernando Pessoa, Samuel Beckett, Friedrich Dürrenmatt, Dino Buzzati, ... Tutti offrono esempi significativi di testi fondati su dispositivi narrativi o contenuti che possono essere messi in relazione con concetti centrali dell'Informatica: iterazione, ricorsione, invarianti, ordinamenti, classificazioni, ricerche anche stocastiche, numerazioni, strutture dati, branching narrativo e combinatoria, o più semplicemente esemplificano tecniche di logica deduttiva.

Questo uso didattico di tali opere non implica però una riduzione del testo letterario ad un pretesto tecnico, ma consente di costruire un humus multi-disciplinare nel quale studenti e studentesse siano chiamati a riconoscere strutture, regolarità, formulare ipotesi interpretative, a individuare vincoli e descrivere processi, possibilmente realizzando produzioni proprie, affinché possano sviluppare competenze inter-discipliari e argomentare le proprie scelte.

Per motivi di spazio possiamo solamente indicare alcuni luoghi letterari paradigmatici:

- Dante in, Par. XXVIII 92-0,3 per indicare una incommensurabile moltitudine non impiega una similitudine statica, ma utilizza una metafora molto frequentata dagli stilnovisti che richiama un processo dinamico di crescita geometrica :
  ... ed eran tante, che 'l numero loro

più che 'l doppiar de li scacchi s'inmilla.
Dante Alighieri - Par. XXVIII 92-03;

- William Shakespeare, nel Prologo del Henry V, vv 16-19, per esprimere la magia del teatro che rende possibile ad un manipolo di guitti la celebrazione di un evento grandioso, utilizza la metafora del processo di compressione logaritmica dei dati, implicito nella notazione posizionale resa possibile dall'uso dello zero che chiama ancora *cipher*:
O, pardon! since a crooked figure may
Attest in little place a million;
And let us, ciphers to this great accompt,
On your imaginary forces work.
- E finalmente giungiamo a giustificare il titolo. Kafka, in *Das Schloß* e Beckett in *En attandant Godot*, ma soprattutto Omero nell'*Odissea* forniscono una splendida illustrazione dell'euristica alla base dell'addestramento delle reti neurali, ovvero la *stochastic gradient descent*: partendo da informazioni confuse si eseguono azioni localmente razionali, ma globalmente incerte. Non c'è garanzia di raggiungere l'optimum e il progresso emerge per iterazione, più statisticamente che non deduttivamente.
- Numerosi sono i racconti di Kafka, *e.g. Il messaggio dell'imperatore (Eine kaiserliche Botschaft)* e *Durante la costruzione della muraglia cinese (Beim Bau der chinesischen Mauer)*, dove sono descritti processi iterativi, e il limite asintotico verso il punto fisso a cui tendono.
- L'Oulipo delinea innumerevoli algoritmi di produzione di letteratura *à contrainte* in *La littérature potentielle*[26]. Queneau, in *Cent mille milliard de poemes* suggerisce spunti combinatori e composizionali mirabili, mentre in *Exercises de style* offre ben 99 implementazioni di una stesso *template*. *La vie mode d'emploi* di Perec è opera che richiede lettura parallela mentre *La disparition* implementa il lipogramma più emblematico, ovvero il non utilizzo del carattere più frequente la "e". Inviteremmo gli studenti a scrivere rispettando questo vincolo, perché richiede lo stesso sforzo di riflessione che programmare con un linguaggio assembly.
- Italo Calvino, fu anche oplepiano. In *Le città invisibili* presenta una struttura che richiama concetti di combinatorica e di teoria dei grafi [4], in cui i luoghi e gli oggetti possono essere interpretati come nodi, mentre i ricordi e le storie fungono da archi di connessione; ad esempio, *Eutropia* e *Diomira* illustrano aspetti diversi del concetto di invariante di un processo mentre *Olinda* illustra l'iterazione. In *Se una notte d'inverno un viaggiatore*, racconto che esplora la natura della lettura e della scrittura, Calvino utilizza strutture narrative che richiamano esplicitamente gli algoritmi e la programmazione. In particolare la sua struttura frammentata e modulare ricorda la programmazione modulare e le subroutine [28, 11]. Infine in *Il castello dei destini incrociati* la sfida è cogliere la regola generativa nascosta.
- Don Knuth [16] cita in esergo, un frammento da un episodio di Perry Mason, *The Case of the Angry Mourner*, per introdurre con suspence la complessità negli algoritmi di ordinamento: *"But you can't look up all those license numbers in time" Drake objected. "We don't have to, Paul. We merely arrange them in a list and look for duplications."*
- Dino Buzzati nel racconto *I sette messaggeri* calcola di fatto l'invariante di un processo. Ovvero il rapporto tra il giorno di partenza della carovana del principe verso i confini del regno e il tempo di ritorno di un messaggero. L'entità della crescita di questi tempi di ritorno diventa metafora dell'angoscia esistenziale. *Ben presto constatai che bastava moltiplicare per cinque i giorni fin lì impiegati per sapere quando il messaggero ci avrebbe ripresi.*
- Numerosi racconti di Edgar Allan Poe, quali *e.g. The Gold-Bug* o *The Murders in the Rue Morgue* riguardano il paradigma indiziario e la risoluzione di enigmi introducendo elementi di logica e

crittografia nella narrativa [12]. Sono dunque immediatamente correlabili al concetto di Computational Thinking.

- Milorad Pavić struttura il romanzo *Dizionario dei Chazari* (1984), come un dizionario enciclopedico con voci separate, organizzate in tre parti: la versione cristiana, la versione islamica e la versione ebraica. Ogni voce è autonoma, ma interconnessa, permettendo al lettore varie esperienze di lettura interattive e non lineare [17, 24]. Nel Dizionario, alla voce *Masudi* nel Libro Verde, troviamo la *Storia delle morti dei figli* che è un esempio di ricorsione.

- Julio Cortázar nel romanzo *Rayuela [Il gioco del mondo]* offre anch'egli una molteplicità di percorsi di lettura non necessariamente lineari, ma labirintici.

- *Ulysses* di James Joyce non può essere letto senza l'ausilio di opportune strutture dati matriciali, e la sfida di *Finnegans Wake* appare spesso come una raccolta di *cryptic clues* di un *crossword puzzle*.

- L'opera complessa e multiforme di Fernando Pessoa offre numerose esemplificazioni narrative di concetti computazionali. Nell'opera teatrale *Il marinaio* (*O Marinheiro*), Pessoa impiega la ricorsione come espediente letterario per costruire il testo. Tre donne conversano per passare il tempo creando una serie di racconti che si annidano l'uno nell'altro. Ogni racconto contiene elementi che richiamano la conversazione principale, creando un ciclo di narrazione che si ripete e si evolve. Questo è simile alla ricorsione in programmazione, dove una funzione chiama sé stessa con modifiche incrementali fino a raggiungere una condizione di base. "Ho per la vita l'interesse di un decifratore di sciarade" dichiara Pessoa che era un appassionato enigmista. In *I casi del dottor Abílio Quaresma* Pessoa applica la logica deduttiva come fa il Dupin di Poe.

- La poetessa, Premio Nobel per la Letteratura 1996, Wisława Szymborska nelle poesie *Pi Greco* e *Contributo alla statistica* esprime in modo commovente come anche la matematica sia una delle scienze umane, facendo di pi greco una sciarada universale e usando i dati statistici come un'anafora.

- L'autore più citato e famoso tra quelli che hanno esplorato concetti computazionali è Jorge Luis Borges. Numerosi sono i racconti che utilizzano concetti di teoria dell'informazione, strutture algoritmiche e logiche ricorsive per esplorare temi di infinito, caos e ordine. *La biblioteca di Babele* descrive una biblioteca infinita che contiene libri con ogni possibile combinazione di lettere e simboli. Questo è simile ai concetti di ricorsione e autoreferenzialità in Informatica, dove una funzione può chiamare sé stessa e un sistema può contenere rappresentazioni di sé stesso. Anche nel racconto *Le rovine circolari* Borges utilizza un espediente narrativo ricorsivo, mentre in *Esame dell'opera di Herbert Quain* viene analizzato un testo letterario che non procede linearmente, ma si apre in più percorsi alternativi, come un albero decisionale. Infine, nel racconto *La morte e la bussola* utilizza la stessa tecnica logico deduttiva dei racconti polizieschi di Poe e Pessoa, ma il detective di Borges non sbaglia perché ragiona male, ma perché ragiona troppo "geometricamente". Il criminale sfrutta proprio l'algoritmo mentale del detective Lönnrot: prevede il suo modo di interpretare gli indizi e costruisce per lui un percorso che lo porterà in trappola.

- Infine, Friedrich Dürrenmatt in *La morte della Pizia* (*Der Tod der Pythia*, 1976) costruisce una profonda destrutturazione del mito di Edipo attraverso un meccanismo narrativo che può essere interpretato mediante il concetto di iterazione [7, 32]. La verità non emerge come rivelazione unica e definitiva, ma come successione di versioni, reinterpretazioni e duplicazioni del mito. L'Edipo di Dürrenmatt è un *template* su cui si può applicare un *pattern matching*.

4 Discussione e Conclusione

La nostra proposta di connettere concetti di pensiero computazionale a testi letterari risponde al bisogno delineato nella Sezione 2 di un nuovo paradigma per integrare efficacemente l'IA nella didattica, in particolare dell'Informatica. È cruciale rafforzare le competenze di lettura critica, analisi strutturale e valutazione consapevole dei risultati [2, 15], cioè il pensiero critico. Pertanto il confronto con testi letterari complessi, ambigui e stratificati costituisce una palestra per esercitare abilità trasversali rilevanti anche in ambito informatico: scomporre un problema, riconoscere pattern, distinguere superficie e struttura, seguire catene di dipendenze, verificare la coerenza interna di un sistema simbolico e discutere interpretazioni alternative. L'apprendimento dell'Informatica viene così più orientato alla comprensione dei processi, alla riflessione metacognitiva e alla costruzione condivisa di significato [27], piuttosto che al *coding*.

Le Linee guida per gli Istituti Tecnici non individuano un canone prescrittivo di autori da proporre agli studenti, ma definiscono risultati di apprendimento, competenze, abilità e conoscenze che lasciano al docente e alla programmazione collegiale la selezione di testi significativi.

In tale con-testo, la proposta di utilizzare autori quali Italo Calvino, Franz Kafka, Jorge Luis Borges, Fernando Pessoa, Friedrich Dürrenmatt o Milorad Pavić, ecc. appare coerente con l'impianto delle Linee guida, purché tali testi siano selezionati non come semplice ampliamento del repertorio letterario, ma come occasioni di lavoro su strutture testuali, procedimenti narrativi e forme di organizzazione del significato. Lo studio di questi autori consente infatti di costruire un percorso inter-disciplinare nel quale l'analisi letteraria dialoga con concetti propri del pensiero computazionale e dell'Informatica quali la ricorsione, iterazione, classificazione, combinatoria, branching, invarianza o ordinamento.

Nel contesto internazionale sono sporadici gli studi che esplorano i legami tra Letteratura e Informatica [8, 23], mentre ben maggiore è stata la ricerca sulle connessioni tra Letteratura e Matematica [13, 18, 29, 31].

La letteratura può diventare utile nell'apprendimento dell'Informatica, sia per presentare concetti tradizionalmente legati al pensiero computazionale, ma anche per indurre studentesse e studenti a lavorare maggiormente sull'analisi e valutazione dei testi, esigenze emerse nei lavori presentati nella Sezione 2. Inoltre, l'Informatica sarebbe così utile alla Letteratura, per avvicinare studentesse e studenti alla lettura.

Nel recente articolo *Das Bild des einsamen Informatikers schadet der Wirtschaft*[6], viene discussa l'immagine della professione informatica alla luce del *Gender Intelligence Report 2025 – WARUM MERITOKRATIE nur mit ≪INCLUSION≫ gelingt.*[7] Emerge che rappresentare l'Informatica come un'attività collaborativa, creativa, comunicativa e orientata alla soluzione di problemi reali può renderla più inclusiva.

In questa prospettiva, la nostra "modesta proposta" è un possibile approccio per modificare la visione stereotipata della disciplina e favorire anche una maggiore inclusione di genere.

Da tempo gli autori condividono con Nardelli [25] che l'Informatica possa "agire come ponte tra le due culture", scientifica e umanistica. Pertanto affrontarne i principali concetti informatici con una prospettiva anche umanistica è funzionale alla luce del necessario cambio di paradigma di insegnamento che le sfide dell'IA ci pone.

Sarebbe importante creare un progetto da estendere a una rete di scuole secondarie di secondo grado, dove è presente l'insegnamento dell'Informatica, per verificare se un approccio interdisciplinare che coinvolga la letteratura possa essere efficace. Inoltre, si potrebbe verificare il coinvolgimento e la motivazione degli studenti e studentesse, anche in funzione dell'inclusione di genere.

[6] L'articolo analizza l'immagine dell'informatica come lavoro solitario, inoltre sottolinea che una rappresentazione della professione informatica maggiormente centrata sul lavoro di squadra e sullo scambio potrebbe aumentare in modo significativo l'interesse delle ragazze.

[7] Il report è disponibile in inglese al link `https://www.advance-hsg-report.ch/en/`

References

[1] Ojo Amos Adewale, Jayesh Rane, Martina Oluchi Ogbonna, and Nitin Liladhar Rane. Pedagogy with generative artificial intelligence: Opportunities and challenges in education. *International Journal of Applied Resilience and Sustainability*, 2(2):339–366, 2026.

[2] Hamsa Bastani, Osbert Bastani, Alp Sungu, Haosen Ge, Özge Kabakcı, and Rei Marimane. Generative ai without guardrails can harm learning: Evidence from high school mathematics. *Proceedings of the National Academy of Sciences*, 2025.

[3] Brett A. Becker, Michelle Craig, Paul Denny, Hieke Keuning, Natalie Kiesler, Juho Leinonen, Andrew Luxton-Reilly, Lauri Malmi, James Prather, and Keith Quille. Generative AI in introductory programming. In *Computer Science Curricula 2023, Curricular Practices Volume*. ACM, 2023.

[4] H. Bloom and S.P.H.H. Bloom. *Bloom's Modern Critical Views*. Bloom's Modern Critical Views Series. Facts On File, Incorporated, 2011.

[5] Miroslava Černochová, Hasan Selcuk, Anat Cohen, Patrik Voštinár, Guy Cohen, Adi Yaakov Azaria, Tomáš Jeřábek, Martin Beneš, and Jarmila Škrinárová. Generative artificial intelligence (GenAI) in k–12 education: National policy perspectives from the czech republic, israel, latvia, and slovakia. *Pedagogika*, 75(4):381–400, 2025.

[6] Tony Clear, Åsa Cajander, Alison Clear, Roger McDermott, Mats Daniels, Monica Divitini, Matthew Forshaw, Niklas Humble, Maria Kasinidou, Styliani Kleanthous, Can Kultur, Ghazaleh Parvini, Mohammad Polash, and Tingting Zhu. Ai integration in the it professional workplace: A scoping review and interview study with implications for education and professional competencies. In *ITiCSE-WGR 2024*. ACM, 2024.

[7] Mark-Georg Dehrmann. Dürrenmatt in delphi: Korrekturen des ödipus-mythos im sterben der pythia. 2011.

[8] Mark Dougherty. What has literature to offer computer science? *HUMANIT*, 7(1):74–91, 2004.

[9] Hubert L. Dreyfus and Stuart E. Dreyfus. Making a mind versus modeling the brain: Ai at a crossroads. *Daedalus*, 117:15–44, 1988.

[10] Tony Haoran Feng, Steffan Hooper, Paul Denny, Burkhard C. Wünsche, and Andrew Luxton-Reilly. Bot or prof: Comparing the characteristics of ai-generated and instructor-written responses to student queries in an introductory programming course forum. In *ACE 2026*. ACM, 2026.

[11] H. Fulton, R. Huisman, J. Murphet, and A. Dunn. *Narrative and Media*. Cambridge University Press, 2006.

[12] Kevin J. Hayes. *The Cambridge Companion to Edgar Allan Poe*. Cambridge Companions to Literature. Cambridge University Press, 2002.

[13] N. Katherine Hayles. *My Mother Was a Computer: Digital Subjects and Literary Texts*. University of Chicago Press, Chicago, 2005.

[14] Martin Heidegger. *Being and Time*. Harper & Row, New York, 1962.

[15] Xiaojin Hou, Zhichun Liu, Guoqiang Wang, and Nianfeng Shi. Bridging awareness and behavior: decoding implicit metacognitive behaviors in ai-assisted programming via fine-grained log analysis. *Frontiers in Psychology*, 17, 2026.

[16] Donald E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Addison-Wesley, Reading, Massachusetts, 2 edition, 1998.

[17] G.P. Landow. *Hypertext: The Convergence of Contemporary Critical Theory and Technology*. Parallax (Baltimore, Md.). Johns Hopkins University Press, 1992.

[18] Sally I. Lipsey and Bernard S. Pasternack. Mathematics in literature. 2015.

[19] Boško Lišnić, Goran Zaharija, and Saša Mladenović. Integration of artificial intelligence in k–12: Analysis of a three-year pilot study. *AI*, 6:49, 2025.

[20] Linda Mannila. Co-designing ai literacy for k–12 education. In *Proceedings of WiPSCE 2024*. ACM, 2024.

[21] Muhammad Hassan Massaty, Slamet Kurniawan Fahrurozi, and Cucuk Wawan Budiyanto. The role of ai in fostering computational thinking and self-efficacy in educational settings: A systematic review. *Indonesian Journal of Informatics Education*, 8(1), 2024.

[22] Ian McCulloh, Pedro Rodriguez, Srivaths Kumar, Manu Gupta, Viplove Raj Sharma, Benjamin Johnson, and

Anthony N. Johnson. Generative ai in computer science education: Accelerating python learning with chatgpt. In *Proceedings of ACM*, 2025.

[23] H. Willis Means. Using literature in a computer science service course: Improving abstract/critical thinking skills. *J. Comput. Sci. Coll.*, 6(5):30–34, April 1991.

[24] Stefan Milosavljević. A computational social network analysis of the dictionary of the khazars by milorad pavić. 09 2025.

[25] Enrico Nardelli. *La rivoluzione informatica. Conoscenza, consapevolezza e potere nella società digitale*. Themis, 2022.

[26] Ouvroir de Littérature Potentielle (Oulipo). *La littérature potentielle: créations, re-créations, récréations*. Number 95 in Folio essais. Gallimard, Paris, 1988.

[27] Giulia Paludo and Alberto Montresor. Toward a new paradigm of ai-powered programming education through metacognition. *Italian Journal of Educational Technology*, 2026.

[28] F. Ricci. *Difficult Games: A Reading of I Racconti by Italo Calvino*. Wilfrid Laurier University Press, 1990.

[29] Bharath Sriraman and Astrid Beckmann. Mathematics and literature: Perspectives for interdisciplinary classroom pedagogy. 01 2007.

[30] Ludwig Wittgenstein. *Philosophical Investigations*. Wiley-Blackwell, Chichester, UK, 4 edition, 2009.

[31] C. K. Zaleta and Kim K. Ruebel. Exploring mathematical concepts in literature. *Middle School Journal*, 40:36–42, 2008.

[32] Rosmarie Zeller. *Das Sterben der Pythia*, pages 145–146. J.B. Metzler, Stuttgart, 2020.

Dalla SEO al Web Agentico: l'evoluzione della didattica dell'informatica in una simulazione d'impresa

Sonia Guerci[1]*

Istituto Tecnico Antonio Zanon di Udine
sonia.guerci@itzanon.it

Sommario

Il centro di gravità del programmatore si sta spostando dall'esecuzione alla direzione: sempre meno scrivere codice da zero, sempre più specificarlo, valutarne l'output, riconoscere quando l'IA sbaglia. Questo cambio di ruolo — dal tecnico al project manager — apre una possibilità non scontata: uno studente di un indirizzo economico, già abituato a ragionare per obiettivi e a gestire deliverable, potrebbe rivelarsi sorprendentemente ben attrezzato, purché abbia la base tecnica sufficiente a leggere criticamente quello che l'IA produce. La soglia esiste, ma è più bassa di quella del programmatore classico. La stessa trasformazione riguarda il docente: da erogatore di contenuti a gestore di un progetto didattico complesso.

Questo contributo racconta un tentativo concreto di navigare entrambe le trasformazioni. Nella classe 4AR (Relazioni Internazionali per il Marketing) dell'Istituto Tecnico "A. Zanon" di Udine, sei gruppi-agenzia hanno progettato e pubblicato il sito di un'azienda-caso con un obiettivo di marketing dichiarato. Attorno allo sviluppo web si sono innestati: un modulo di Marketing Mix Optimization su foglio di calcolo avanzato; un confronto diretto tra gli strumenti di raccolta dati del marketing digitale e i limiti tecnici e normativi che ne condizionano l'efficacia; un workflow Agile come strumento di equità nella valutazione individuale; e, a chiusura, l'evoluzione dalla SEO tradizionale verso la GEO e il Web Agentico.

1 Introduzione: il programmatore/project manager

Il centro di gravità del programmatore si sta spostando dall'esecuzione alla direzione: sempre meno scrivere codice da zero, sempre più specificarlo con precisione, valutarne l'output, riconoscere quando l'IA sbaglia anche se il risultato sembra plausibile. È un cambio di ruolo che richiede competenze in parte nuove — formulare requisiti chiari, valutare un deliverable rispetto a criteri di business, gestire un flusso di lavoro che coinvolge strumenti autonomi — e che assomiglia, non a caso, più al lavoro di un project manager che a quello di un tecnico [7].

Questo apre una possibilità non scontata: uno studente di un indirizzo economico, già abituato a ragionare per obiettivi, potrebbe rivelarsi sorprendentemente ben attrezzato per questo ruolo, purché abbia anche la base tecnica sufficiente a leggere criticamente quello che l'IA produce. Quella base non è negoziabile: senza, non si dirige, si subisce. Ma la soglia non è più quella del programmatore classico che sa costruire tutto da sé; è quella del *lettore critico*, di chi sa riconoscere un errore semantico e sapere quale domanda fare per ottenere un risultato migliore.

La trasformazione riguarda anche il docente di informatica: da erogatore di contenuti tecnici a gestore di un progetto didattico complesso, con ruoli definiti, scadenze, deliverable reali e valutazione differenziata per ciascuno. La didattica per progetti — quella vera — è faticosa da progettare e impegnativa da tenere in piedi in aula. Rispetto alla lezione frontale, è un altro mestiere. Ed è esattamente il salto che stiamo chiedendo agli studenti di compiere.

*Docente di Informatica

Questo contributo racconta un tentativo concreto di navigare entrambe le trasformazioni. Nella classe 4AR dell'Istituto Tecnico "A. Zanon" di Udine, sei gruppi hanno ricoperto il ruolo di una Digital Agency. Ogni gruppo ha lavorato su un'azienda-caso differente, focalizzandosi su uno specifico obiettivo di marketing: SEO locale, lead generation, brand awareness, retention, frequenza d'acquisto o iscrizioni corsi. All'interno di ogni gruppo, gli studenti si sono distribuiti su quattro ruoli professionali: Project Manager, SEO & Legal, UX Designer, IT Developer. Le sezioni seguenti descrivono le scelte didattiche che hanno reso possibile questo progetto, nell'ordine in cui si sono succedute.

2 Prerequisiti tecnici e IA: prima capire, poi delegare

La scelta metodologica di fondo è stata strutturata su due livelli complementari. In primo luogo, agli studenti sono state fornite le solide basi di HTML semantico, CSS responsive e JavaScript: competenze imprescindibili per consentire loro di comprendere la struttura della pagina, integrare gli output dell'IA in modo autonomo e velocizzare l'interazione con il codice [7].

In secondo luogo, l'introduzione dell'IA generativa è stata guidata dalla logica del *Necessity Learning Design* [8], secondo cui una competenza concettuale avanzata va fatta emergere come necessità sentita prima di poter formulare istruzioni specifiche, che un modello linguistico non produce spontaneamente senza una guida competente [6]. Questo è evidente nello studio dei dati strutturati (es. in formato JSON-LD): gli studenti devono comprendere come i motori di ricerca interpretino la semantica di una pagina per generare i cosiddetti "risultati avanzati" (*rich results*) — ovvero schede visive arricchite nei risultati di ricerca.

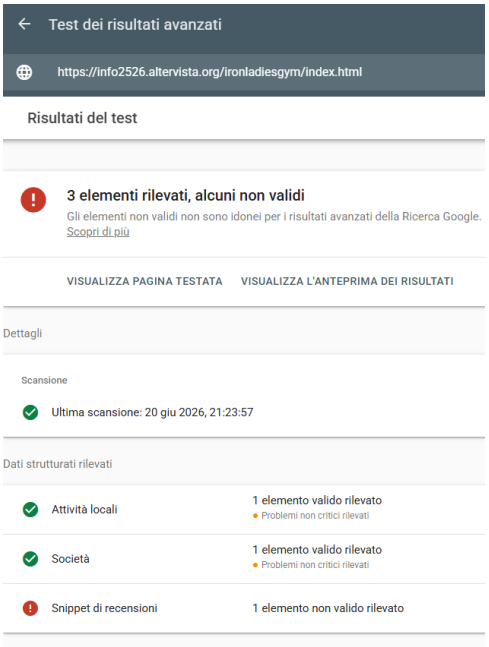


Figura 1: Interfaccia del "Test dei risultati avanzati" di Google eseguito da uno studente sul codice del progetto "palestra". Il responso mostra un risultato misto (due elementi validi e uno non valido): un tipico scenario didattico in cui l'errore generato dall'IA richiede un intervento consapevole di debugging e validazione critica da parte dello studente, che non può accettare passivamente l'output della macchina.

Un'IA a cui si chieda genericamente di "creare un sito per una palestra" non penserà, infatti, a configurare un evento `consenso_aggiornato` per attivare Google Analytics solo dopo il consenso, né a quale tipo di Schema.org sia corretto per quel settore [2]. È lo studente — scontrandosi con i limiti di un prompt generico e comprendendo la necessità di studiare il funzionamento della SEO, dei motori di ricerca e dei formati di marcatura — a dover tradurre l'obiettivo di business in un prompt tecnico verificabile, testando poi l'output mediante gli strumenti di diagnostica ufficiali.

3 Il valore dei dati per le decisioni strategiche

Un secondo modulo, solo apparentemente lontano dallo sviluppo web, ha ripensato l'uso del foglio di calcolo avanzato non come mero esercizio tecnico, ma come simulazione del flusso di lavoro di un *Marketing Data Analyst*[1]. L'obiettivo primario dell'attività è stato duplice: da un lato, guidare gli studenti nell'esplorazione di strumenti avanzati di Business Intelligence (come Power Pivot e la modellazione DAX); dall'altro, e soprattutto, dimostrare concretamente come le decisioni strategiche aziendali debbano necessariamente fondarsi su solide basi di dati, facendo così nascere nei discenti l'esigenza logica di raccogliere tali metriche a monte tramite i canali digitali.

Il punto di partenza è stato un caso aziendale caratterizzato da obiettivi conflittuali: ripartire un budget trimestrale tra quattro canali pubblicitari massimizzando il fatturato, nel rispetto dei vincoli minimi richiesti dall'ufficio commerciale (lead) e dal brand management (impression). Per affrontare il problema, gli studenti hanno costruito l'intera pipeline analitica (Figura 2): dall'importazione e pulizia dei dati con Power Query (fase di ETL), alla modellazione in schema a stella con Power Pivot, fino al calcolo delle metriche storiche tramite misure DAX. Solo una volta strutturato il patrimonio informativo, l'applicazione del Risolutore di Excel ha permesso di individuare la combinazione ottimale di allocazione del budget.

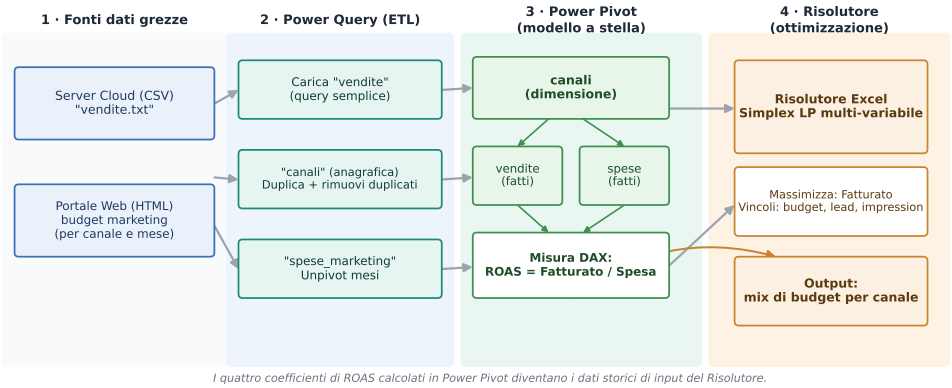


Figura 2: Pipeline degli studenti: da Power Query (ETL) a Power Pivot/DAX fino all'ottimizzazione multi-obiettivo con il Risolutore.

Per ragioni di tempo e, soprattutto, perché i siti dei gruppi non avrebbero potuto generare immediatamente traffico reale significativo, i dati di vendita e di spesa storica usati in questa attività sono fittizi, sebbene strutturati in modo da preservare un buon grado di realismo.

Lavorare su uno scenario predittivo ha reso evidente agli studenti il legame causale tra lo sviluppo web e la data analysis: in un contesto professionale, i dati macro (lead per canale, costo di acquisizione,

[1] https://info2526.altervista.org/test/analisi2/index.html

conversioni) inseriti nel Risolutore non sono un postulato di partenza, ma il risultato diretto di una precisa strategia di tracciamento e raccolta dati — tramite cookie, email marketing e parametri UTM — impostata proprio sulle pagine del sito web per alimentare i modelli decisionali strategici.

4 La guerra dei dati: raccogliere per il marketing, proteggere per la privacy

Il modulo sulla privacy ha reso esplicito un conflitto strutturale del marketing digitale: chi fa marketing ha bisogno di dati sull'utente per misurare, targettizzare e personalizzare; chi tutela l'utente ha tutto l'interesse a limitarne il flusso. Tre fronti, discussi in classe e sintetizzati nella Tabella 1.

I **cookie di profilazione** richiedono consenso esplicito per il GDPR, ma AdBlocker, uBlock Origin e Ghostery bloccano lo script di tracciamento ancora prima che possa attivarsi: anche col consenso, il dato semplicemente non viene generato. Gli studenti lo hanno verificato in prima persona nella scheda *Network* del browser.

Il **pixel di tracciamento via email** misura le aperture con un'immagine invisibile da un pixel: i filtri antispam ne bloccano il caricamento (falso 0% di aperture), mentre Apple Mail Privacy Protection la pre-scarica sempre via proxy (falso 100%). Il click rate resta la metrica più solida.

I **parametri UTM** sopravvivono meglio di tutto: iOS 17 e Brave rimuovono selettivamente gli identificatori personali (`fbclid`, `gclid`) ma lasciano passare `utm_source`, che identifica la campagna e non l'individuo. Un paracadute di attribuzione utile anche quando i cookie falliscono.

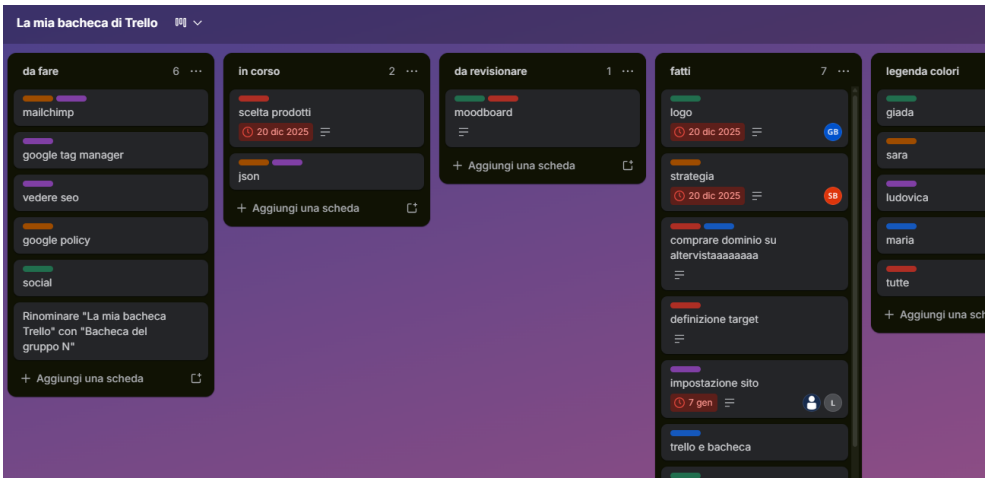
Tabella 1: La guerra dei dati: strumenti di raccolta e relative contromisure

Strumento	**Contromisura**	**Esito per il marketer**
Cookie/pixel di profilazione	AdBlocker blocca lo script prima dell'attivazione.	Il dato non viene generato, anche col consenso.
Pixel email (immagine 1×1)	Blocco immagini antispam; pre-caricamento via proxy (Apple MPP).	Open rate inaffidabile; click rate più solido.
Parametri UTM nell'URL	Stripping selettivo di `fbclid`/`gclid` (iOS 17, Brave).	`utm_source` spesso superstite: identifica la campagna, non la persona.

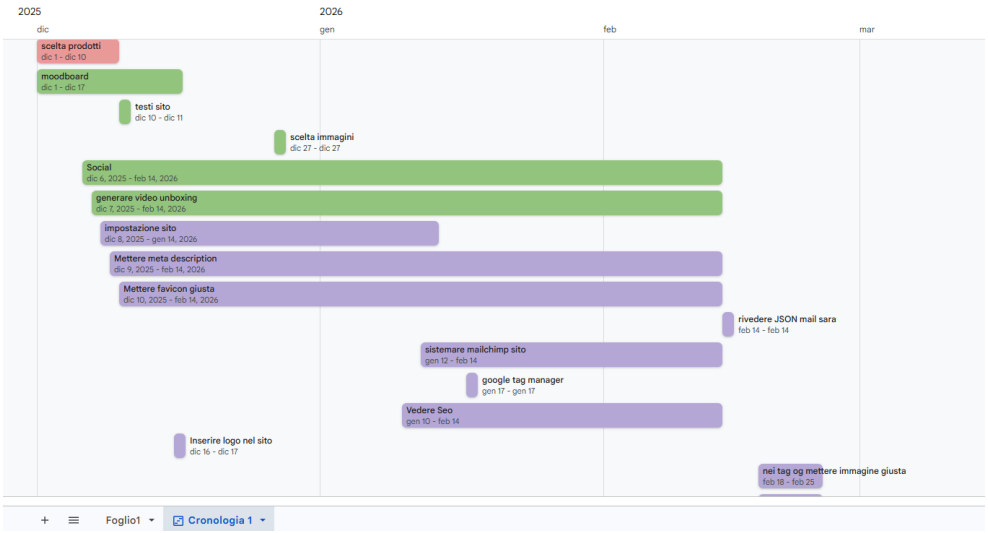
L'infrastruttura di raccolta — Consent Mode, link con UTM, form con lead magnet — è stata progettata e testata dagli studenti; la classe ha imparato a costruire correttamente la macchina che raccoglie i dati, conoscendone i limiti tecnici e legali, prima ancora di poterne osservare l'output. Una parziale eccezione ha riguardato l'indicizzazione: avendo segnalato i siti ai motori di ricerca circa un mese prima della conclusione, è stato possibile verificare in Google Search Console i risultati della strategia SEO — con grande soddisfazione della classe.

5 Project Management Agile e sicurezza aziendale

Uno dei limiti del lavoro di gruppo tradizionale è che il contributo di ciascuno resta spesso opaco. Per affrontarlo, la gestione della Digital Agency ha ricalcato le pratiche Agile [4]: ogni gruppo ha gestito una Kanban Board (Trello) e un diagramma di Gantt su foglio di calcolo (Figura 3), con task atomici assegnati a un responsabile e con scadenza visibile a tutto il gruppo.



(a) Kanban Board (Trello): colonne da fare / in corso / da revisionare / fatti, con etichette colore per componente.



(b) Gantt del medesimo gruppo: task, durata e responsabile visibili all'intera classe.

Figura 3: Strumenti Agile di un gruppo: ogni task ha un responsabile e una scadenza condivisa.

La trasparenza non è stata un fine in sé, ma lo strumento che ha permesso una valutazione più equa: la storia delle card e degli artefatti prodotti da ciascun ruolo (Tabella 2) ha fornito evidenze concrete per differenziare il voto individuale. La verifica finale — una presentazione di 15 minuti in cui ogni ruolo ha "difeso" la propria parte davanti alla classe — ha chiuso il cerchio.

La sicurezza informatica — parte teorica del programma su reti e protezione dei dati — è stata integrata in questo stesso contesto anziché trattata in astratto. Gli studenti hanno ragionato su come proteggere il patrimonio informativo della propria azienda-caso: i dati dei clienti raccolti dai form, le credenziali degli account condivisi (FTP, GTM, Analytics) e le conseguenze legali di una loro gestione poco accorta. Il collegamento è rimasto breve ma esplicito: protezione dei dati, conformità GDPR e si-

Tabella 2: Ruoli, responsabilità e tracce di valutazione nel workflow Agile

Ruolo	**Responsabilità principali**	**Evidenza per la valutazione**
PM	Gestione di Kanban e Gantt, supervisione tempi, controllo qualità e conformità legale.	Storico delle card e delle scadenze, audit di conformità.
SEO & Legal	Copywriting, markup JSON-LD, cookie/privacy policy, piano editoriale.	Markup validato, policy pubblicata, piano editoriale.
UX	Wireframe, interfaccia mobile-first, grafica banner cookie, immagini royalty-free.	Wireframe/mockup, layout responsive.
IT	Codice HTML/CSS/JS, integrazione Schema.org e GA4, script cookie, form, deploy FTP.	Sito online, script di tracciamento funzionante.

curezza informatica sono facce della stessa medaglia per una piccola impresa digitale. Scelte già discusse altrove nel progetto — come la separazione tra cookie tecnici e di profilazione — hanno acquistato così una lettura ulteriore, in chiave di rischio aziendale.

Tutto il lavoro della 4AR è confluito in un portale pubblico[2] che raccoglie i sei siti, il manuale operativo con ruoli e roadmap, e le guide su SEO, dati strutturati, email marketing e privacy/GTM.

Per studenti RIM orientati al marketing, documentare la propria presenza web è una competenza professionale trasferibile oltre il singolo progetto, in linea con la logica dell'E-Portfolio digitale introdotto dalla riforma dell'orientamento [5].

6 Conclusioni: dalla SEO e GEO al Web Agentico

Durante il progetto, la classe non si è limitata a ragionare in termini di SEO tradizionale per farsi trovare dagli utenti. Attraverso l'implementazione dei dati strutturati in formato JSON-LD, gli studenti hanno di fatto messo in pratica i principi della GEO (*Generative Engine Optimization*): fornire alle IA generative (come ChatGPT o Perplexity) il contesto semantico per leggere, interpretare e citare correttamente i contenuti web. A progetto ormai concluso — nel febbraio 2026 — è stato possibile mostrare come questo lavoro aprisse verso un orizzonte ancora più ampio.

Questo orizzonte è l'AEO (*Agent/Action Engine Optimization*) o "Web Agentico": un paradigma in cui non basta che l'IA comprenda una pagina, ma deve potervi compiere azioni in autonomia (prenotare, comprare, compilare un modulo). Il salto dalla GEO all'AEO è stato spiegato partendo proprio dal JSON-LD che la classe aveva già costruito: se i dati strutturati dicono all'IA *cosa è* un oggetto, un agente autonomo ha bisogno di sapere anche *cosa può farci*.

È in questo spazio che si inserisce il Model Context Protocol [1], una "presa universale" per collegare i modelli ai sistemi esterni. Quando Google, il 10 febbraio 2026, ha annunciato WebMCP [3] per portare questa stessa logica direttamente nel browser, le attività laboratoriali del progetto si stavano letteralmente concludendo. Per una pura questione temporale, l'implementazione dell'AEO non si è potuta tradurre in codice, ma è stata offerta come lettura strategica finale. Eppure, la competenza acquisita sui dati strutturati si è rivelata il prerequisito naturale per comprenderla: chi sa dichiarare a una macchina la natura di un dato è a un solo passo dal poterle definire le azioni consentite.

È questa, in fondo, la risposta più concreta alla domanda da cui siamo partiti: cosa resta da fare quando l'IA genera il codice tecnico in pochi secondi? Per figure professionali in uscita da un indirizzo

[2] https://info2526.altervista.org/4ar/

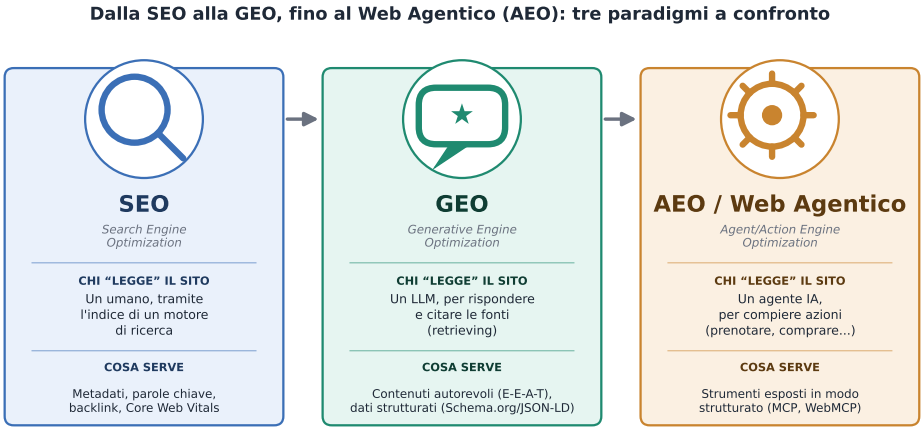


Figura 4: L'evoluzione discussa in classe: dall'integrazione pratica di SEO e GEO, fino alla prospettiva del Web Agentico (AEO).

come il RIM (Relazioni Internazionali per il Marketing), la risposta non sta nello sviluppo software puro, ma nella capacità di strutturare le informazioni aziendali e governare le interfacce affinché gli strumenti automatici possano usarle per generare valore. Resta la capacità di orchestrare i dati. Non l'ultima moda, ma la vera cabina di regia dell'automazione — ieri per la SEO, oggi per la GEO, domani per gli agenti. E un percorso con basi economico-aziendali ha molto più da offrire in questo scenario di quanto sembri.

Riferimenti bibliografici

[1] Anthropic PBC. Model Context Protocol (MCP) Specification: Connecting AI Models to Real-World Data. https://modelcontextprotocol.io/, 2024.

[2] Eva Brumberger and Michael Kleinfeld. Integrating Technical Communication and MarTech: Implications for the Vocational Computing Curriculum. In *IEEE International Professional Communication Conference (ProComm)*, pages 45–52, 2022.

[3] Chrome for Developers. WebMCP is available for early preview. https://developer.chrome.com/blog/webmcp-epp, 2026.

[4] Paola Inverardi and Claudia Sacchetti. L'adozione di metodologie Agile nella didattica dell'informatica nella scuola secondaria. *Atti del Congresso Nazionale AICA*, 2012.

[5] Ministero dell'Istruzione e del Merito. E-Portfolio: la piattaforma Unica per l'orientamento e la documentazione delle competenze. https://unica.istruzione.gov.it/portale/it/orientamento/il-tuo-percorso/e-portfolio, 2024.

[6] Ethan Mollick. *Co-Intelligence: Living and Working with AI*. Portfolio / Penguin, 2024.

[7] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023.

[8] Marco Sbaraglia, Michael Lodi, and Simone Martini. Apprendimento della programmazione guidato dalla necessità: il Necessity Learning Design. In *Atti del Convegno Italiano sulla Didattica dell'Informatica*, ITADINFO 2023, pages 190–195. Università degli Studi di Bari Aldo Moro, 2023. https://www.itadinfo.it/2023/attidelconvegno/ATTI-ITADINFO-2023.pdf.

Il respiro delle città: cittadinanza digitale attiva attraverso il monitoraggio informatico della qualità dell'aria

Anna Lucia Peluso[1], Barbara Avella[2], Daniela Pieraccini[3]
[1] *IC "Ferrajolo – Siani" – Acerra (NA)*
annalucia.peluso@gmail.com

[2] *IC Via Casal Bianco – Roma*
barbara65bar@gmail.com

[3] *IC "Anchise Picchi" – Collesalvetti (LI)*
daniela.pieraccini@scuola.istruzione.it

Abstract

Il percorso interdisciplinare di educazione ambientale e citizen science delle 3 scuole di Acerra, Roma, Collesalvetti utilizza l'indagine sulla qualità dell'aria per l'apprendimento di contenuti di Informatica: programmazione a blocchi di schede programmabili (micro:bit, Arduino), elaborazione e incrocio di dataset ambientali ufficiali e sviluppo di un'app web che integra un modello di apprendimento automatico per la classificazione di campioni di particolato atmosferico.

Parole chiave: #coding #micro:bit #ML #datascienze #citizenscience #vibecoding #IA #scuolaprimaria #scuolaSPG.

1 Introduzione

L'insegnamento dell'informatica nel primo ciclo trae beneficio da un contesto applicativo motivante e vicino al vissuto degli studenti. Il percorso ha utilizzato l'indagine sulla qualità dell'aria nei territori di residenza degli studenti coinvolti, in particolare nell'area di Acerra (NA), storicamente segnata dall'inquinamento ambientale [13,14,17,18,24].

Hanno partecipato le classi 3E della scuola secondaria di primo grado (SSPG) dell'IC "Ferrajolo-Siani" di Acerra (NA), 3A SSPG dell'IC "Anchise Picchi" di Collesalvetti (Li), e la 5A primaria dell'IC Via Casal Bianco di Roma.

Il contesto ambientale ha fornito lo spunto motivazionale, il nucleo del lavoro sono le attività informatiche: programmazione di sensori, incrocio di dataset e sviluppo di una webapp con componenti di intelligenza artificiale (IA). Il contributo isola gli aspetti disciplinari dall'ampio impianto interdisciplinare del progetto (educazione ambientale, scienze della Terra, cittadinanza) non rientrando nei nuclei concettuali dell'Informatica.

2 Attività informatiche

Il percorso si è svolto in parallelo nelle tre sedi, come un unico lavoro distribuito con attività differenziate per ordine di scuola e risorse disponibili. Le classi, coordinate con videoconferenze quindicinali (Google Meet), hanno condiviso dataset grezzi, peer-review reciproca e validazione collegiale della webapp (§2.1). Questa organizzazione a compiti differenziati e interdipendenti ha permesso a ciascuna sede di specializzarsi su una parte del sistema, beneficiando dei risultati delle altre due.

Aspetti disciplinari	Strumenti	Sede	Implementazione
Programmazione a blocchi	Scratch	Tutte	Ciclo del Carbonio
Programmazione di schede	Micro:bit	Tutte / Roma	Trasmissione dati satellitare

Aspetti disciplinari	Strumenti	Sede	Implementazione
Programmazione di sensori	Micro:bit Grove	Acerra e Collesalvetti	Taratura sensori particolato
	Arduino Shinyei	Collesalvetti	Verifica limiti Grove:Dust
Gestione e analisi di dataset ufficiali	Google Fogli	Tutte	Dati ARPA/UE
	Guida analisi colore	Tutte /	Analisi immagini ESA/Copernicus e correlazione ad eventi di cronaca
Sviluppo app per web coding con IA	Gemini	Tutte / Roma / Acerra	Sistema solare / Classificazione percolato

Tabella 1: Strumenti informatici e attività disciplinari per sede [1,2,4,7-13,19,20,23]

2.1 Programmazione a blocchi e sensori homemade

Gli studenti di Acerra (3E SSPG) e Collesalvetti (3A SSPG) hanno programmato a blocchi sensori per rilevare il particolato atmosferico (PM2.5 e PM10), abbinando la scheda micro:bit al sensore Grove:Dust, con taratura e invio in tempo reale dei valori a un foglio di calcolo condiviso [10,12].

Il lavoro congiunto Acerra-Collesalvetti insieme ha evidenziato che il Grove:Dust se esposto a lungo a elevati livelli di PM2.5/PM10 si satura e smette di inviare dati, sembrando un guasto o un bug. Per questo Collesalvetti ha affiancato la scheda Arduino con sensore Shinyei PPD42NS [10,12].

2.2 Comunicazione tra dispositivi: simulazione satellitare

Gli studenti di Roma hanno programmato coppie di schede micro:bit (trasmittente/ricevente) per simulare la trasmissione di parametri ambientali da un satellite a una stazione di terra, liberando le sedi SSPG per focalizzarsi sui sensori di particolato [10,12].

L'attività, affiancata dal tinkering di modellini di satelliti con micro:bit, ha portato a confrontare le immagini reali del dataspace Copernicus [12,20], mettendo in relazione un sistema semplificato con uno reale.

2.3 Incrocio di dataset

Gli studenti di Acerra hanno unito, tramite Google Fogli, le medie annue di NO_2, O_3, PM2.5 e PM10 estratte dall'European Air Quality Index [9] per le centraline di Acerra, Livorno e Roma, confrontandole con le soglie EU, le linee guida della WHO e i dati ARPA Campania [2].

Riscontrata una discrepanza tra i superamenti soglia segnalati dai due database, gli studenti hanno condotto una verifica incrociata con le immagini satellitari Copernicus individuando possibili eventi esplicativi (come l'incendio di rifiuti dell'aprile 2025 a Caivano); un confronto analogo a Roma ha evidenziato eventi simili [8].

Le tre scuole hanno inoltre campionato gli inquinanti atmosferici in loco per tre mesi, esponendo fogli di vaselina per 24 ore, con risultati correlabili alle medie giornaliere ARPAC e alle immagini satellitari [7,8]: un esempio di validazione attraverso fonti indipendenti.

3 Webapp con apprendimento automatico

Il nucleo tecnologicamente più rilevante del percorso è la co-progettazione di webapp (HTML5/JavaScript/CSS) con IA generativa (Gemini) tramite vibe coding: codice prodotto tramite dialogo iterativo in linguaggio naturale verificato e corretto dagli studenti [11]. La classe 5A primaria ha realizzato un simulatore per spiegare i concetti astronomici [23].

Gli alunni di Acerra, con il supporto a distanza di Collesalvetti, hanno realizzato un'app (in fase beta) che [1]:

- identifica e quantizza gli spot di particolato sui campioni di vasellina;
- estrae automaticamente luogo, data (dal nome del file) e dati da file XLS/PDF/CSV di EU e ARPA;

- geolocalizza i campionamenti con una scala di inquinamento a quattro livelli;
- si ri-addestra ad ogni nuovo caricamento
- genera grafici per dataset e di correlazione tra dataset
- incrocia tutti i dati con i parametri meteoclimatici.

Il vibe coding non ha sostituito l'apprendimento informatico ma ne ha costituito un contesto d'uso guidato, richiamando quattro competenze: scomposizione del problema e prompt engineering, lettura critica del codice generato (variabili, cicli, funzioni, DOM); debugging guidato, riconoscendo le "allucinazioni" del modello e formulando ipotesi; addestramento e validazione del modello (§5.1), a confronto con il carattere non deterministico dei sistemi di IA [11]. È stato quindi un'impalcatura (scaffolding) [19] per lavorare su un problema reale mantenendo il controllo concettuale del programma.

4 Relazione con le Nuove Indicazioni Nazionali per l'Informatica

Le attività descritte nelle Sezioni 2 e 3 rientrano nei nuclei concettuali dell'informatica, come da DM n. 221/2025 e Indicazioni Nazionali CINI [5,16], distinguendoli dagli aspetti multidisciplinari (educazione ambientale, citizen science) che hanno costituito il contesto motivazionale del percorso ma non l'oggetto disciplinare del percorso. La Tabella 2 sintetizza questa relazione.

Nucleo concettuale (DM 221/2025)	Attività in cui è esercitato
Algoritmo e sua esecuzione	Logica di acquisizione e taratura dei sensori micro:bit/Arduino; pipeline di elaborazione dell'applicazione web
Programmazione a blocchi e strutture di controllo	Trasmissione/ricezione dati tra coppie di schede micro:bit (scuola primaria); calibrazione dei sensori di particolato (SSPG)
Rappresentazione e organizzazione dei dati	Estrazione automatica di dati da file XLS/PDF/CSV; geolocalizzazione dei campionamenti con classificazione a 4 livelli
Comprensione critica di sistemi di IA (addestramento di un modello)	Applicazione web: classificazione degli spot di particolato sui campioni e ri-addestramento del modello ad ogni nuovo caricamento di dati
Debug e validazione	Confronto e verifica incrociata tra dataset (centraline ARPA, indice europeo, immagini satellitari) per individuare discrepanze nei dati

Tabella 2: Corrispondenza tra nuclei concettuali del DM 221/2025 e attività svolte

Per primaria, la programmazione delle coppie micro:bit risponde in particolare all'obiettivo di scrivere brevi programmi con istruzioni di reazione a eventi; per la SSPG, la taratura dei sensori la webapp rispondono agli obiettivi su variabili, strutture di controllo e debug [5,6,16].

5 Valutazione dell'impatto

Il monitoraggio ha previsto: osservazioni dei docenti negli incontri quindicinali, autovalutazione e tracce oggettive degli artefatti (codici, dataset, versioni dell'app) con secondo rubriche di competenza [3]. Sul piano del debugging, l'82% degli alunni SSPG è risultato in grado di individuare ed eliminare autonomamente errori nel codice.

Il ri-addestramento progressivo del modello, con analisi degli errori, ha ridotto i falsi positivi negli spot di particolato dal 35% al 6% all'aumentare del dataset, rendendo tangibile il legame tra qualità/quantità dei dati e prestazioni del modello. Sul piano dell'inclusione, la natura multimodale delle attività ha permesso una partecipazione attiva del 98% degli alunni BES/DSA.

Infine, il debugging tra Acerra e Collesalvetti, nato dalla saturazione dei sensori (§2.1), ha portato gli alunni a formulare autonomamente l'ipotesi corretta prima dell'intervento dei docenti: un riscontro empirico solido del valore didattico dell'esperienza.

6 Conclusioni

Il percorso dimostra come un contesto applicativo di forte rilevanza civica e ambientale possa fornire una cornice motivazionale efficace per l'insegnamento dell'informatica anche in età di scuola primaria, attraverso la programmazione di dispositivi fisici, la gestione critica di dataset eterogenei e lo sviluppo guidato di app con apprendimento automatico. La co-progettazione con l'IA è stata condotta nel rispetto della normativa vigente [15], del MIASEDU manifesto e codice etico AI [21], dei principi della Scuola Etica Digitale – ENIA [22] e Regolamenti di ciascun istituto.

Il modello è replicabile in altri contesti, a condizione di disporre di schede programmabili a basso costo (micro:bit, Arduino) e di un accesso a IA generativa per lo sviluppo applicativo. Il coinvolgimento di più sedi sullo stesso tema, con compiti differenziati per età e risorse, può costituire un modello organizzativo efficace per la diffusione di contenuti informatici complessi nel primo ciclo di istruzione.

Riferimenti bibliografici e sitografici

[1] App con IA generativa per inquinanti Aria https://claude.ai/public/artifacts/b144f0e7-31cb-4122-b4ab-cfc2d817e1b8

[2] ARPA Campania. Rete regionale di monitoraggio della qualità dell'aria. https://www.arpacampania.it/web/guest/rete-regionale

[3] Castoldi, M. (2016). Valutare le competenze: percorsi e strumenti. Carocci Editore.

[4] Ciclo del Carbonio Storytellung Scratch: https://scratch.mit.edu/projects/1380477495

[5] CINI – Consorzio Interuniversitario Nazionale per l'Informatica. Proposta di Indicazioni Nazionali per l'Informatica nella Scuola.

[6] Commissione Europea, Centro Comune di Ricerca. DigCompEdu: il quadro europeo per le competenze digitali dei docenti. Traduzione ufficiale a cura del Ministero dell'Istruzione

[7] ESA – European Space Agency. Discover ESA-Live: risorse didattiche sull'osservazione atmosferica. https://bookdiscoveresalive.esa.int/LO/ITA/A/1/

[8] European Space Agency. Copernicus Data Space Ecosystem. https://dataspace.copernicus.eu/

[9] European Environment Agency. European Air Quality Index. https://airindex.eea.europa.eu/

[10] Hodges, S., Sentance, S., Finney, J., & Ball, T. (2020). Physical Computing: A Key Element of Modern Computer Science Education. *Computer, 53(4), 20–30.*

[11] Kahn, K., & Winters, N. (2021). Constructionism and AI: A history and a framework for learning with AI. In International Handbook of Computer Science Education (pp. 415–442).

[12] Kalelioglu, F., & Sentance, S. (2020). Teaching with physical computing in school: The case of the micro:bit. *Education and Information Technologies, 25(4), 2577–2603.*

[13] Monks, P. S., et al. (2009). Atmospheric composition change – global and regional air quality. Atmospheric Environment, 43(33), 5268–5350.

[14] Monroy-Hernández, A., & Resnick, M. (2008). Data-activated projects: Connecting coding to the real world. IBM Systems Journal, 47(1), 131–139

[15] Ministero dell'Istruzione e del Merito. Linee guida per l'uso dell'Intelligenza Artificiale a scuola.

[16] Ministero dell'Istruzione e del Merito. (2025). Decreto Ministeriale n. 221 del 9 dicembre 2025. Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione.

[17] PhET Interactive Simulations – University of Colorado Boulder. https://phet.colorado.edu/it/

[18] RaiPlay. Superquark+: L'aria verticale. https://www.raiplay.it/video/2020/10/Superquark-piu-L-aria-verticale-d2e596ae-5a3f-4d02-96c1-51dd35b459e7.html

[19] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM, 52(11), 60–67.*

[20] Resnick, M., & Rosenbaum, E. (2013). Designing for Tinkerability. In Design, Make, Play (pp. 163–181). Routledge.

[21] Rete Nazionale MIASEDU AI Generativa. https://europagram.it/rete-nazionale_miasedu/

[22] Scuola Etica Digitale – Batch International. https://scuolaeticadigitale.it/

[23] Simulatore astronomico https://share.gemini.google/FeSKiRnObhoD

[24] Video didattico introduttivo sulla composizione dell'aria. https://youtu.be/i0HP16YtQ8w

Dal Mattone al Progetto: Cambiare Didattica con il Vibe Coding

Simona Pinto
Équipe Formativa Territoriale della Campania – IC 2 Don Bosco Verdi di Qualiano (NA)
prof.simonapinto@gmail.com

Abstract

L'espressione «vibe coding» indica una pratica emergente in cui si descrive in linguaggio naturale ciò che si desidera ottenere e si accetta il codice prodotto dall'intelligenza artificiale generativa senza leggerlo né comprenderlo a fondo. Questo approccio è in tensione esplicita con le Nuove Indicazioni Nazionali per il primo ciclo, che fondano l'Informatica sulla comprensione di algoritmo, variabili, strutture di controllo e correzione degli errori. Il contributo presenta una proposta didattica per la scuola secondaria di primo grado che ribalta questa contrapposizione in opportunità: il codice generato dall'IA non viene accettato passivamente, ma diventa l'oggetto dell'attività, da eseguire passo passo, comprendere, verificare e correggere. La proposta è illustrata con un esempio concreto in Scratch, varianti per la differenziazione e una rubrica di valutazione del debug. Ogni fase è ricondotta in modo esplicito e verificabile a traguardi e obiettivi delle Nuove Indicazioni Nazionali (DM 221 del 9/12/2025), nel rispetto del principio che siano gli esseri umani a mantenere il controllo sulle decisioni basate su sistemi informatici.

1 Introduzione

Nella pratica scolastica è ancora frequente la sovrapposizione tra il piano «digitale» — l'uso competente di strumenti e applicazioni — e il piano «informatico», cioè l'Informatica come disciplina scientifica fondata su algoritmi, dati e linguaggi. La diffusione degli assistenti basati su intelligenza artificiale generativa rende questa distinzione ancora più urgente, perché sposta l'attenzione dal «come si costruisce» al «che cosa si ottiene».

L'espressione «vibe coding», introdotta da Andrej Karpathy nel febbraio 2025 [4], indica una pratica in cui si descrive in linguaggio naturale ciò che si desidera e si accetta il codice prodotto dall'IA senza leggerlo né comprenderlo a fondo. Nasce per prototipi rapidi e usa-e-getta, e la stessa letteratura tecnica ne segnala i limiti: codice non compreso, difficile da mantenere e potenzialmente errato [1]. Nella sua forma originaria, il vibe coding è dunque in netto contrasto con le Nuove Indicazioni Nazionali per il primo ciclo [2], che pongono al centro la padronanza di algoritmo, variabili, strutture di controllo e debug.

La proposta qui presentata, rivolta alla scuola secondaria di primo grado e in particolare ai docenti di Matematica e di Tecnologia — a cui è prevalentemente affidato l'insegnamento dell'Informatica — non propone di sostituire la programmazione consapevole con il vibe coding, ma di trasformare il codice generato dall'IA in oggetto d'indagine: leggerlo, eseguirlo passo passo, individuarne i difetti e correggerli. È il passaggio «dal mattone al progetto» del titolo: dai singoli costrutti compresi a un artefatto più ampio che resta governato dallo studente, non dalla macchina.

2 La proposta didattica: dal mattone al progetto

Il nucleo della proposta consiste nel ribaltare la contrapposizione tra vibe coding e comprensione algoritmica in opportunità didattica. L'ambiente di riferimento è Scratch, ma gli esempi valgono anche per gli editor a blocchi di code.org: poiché gli assistenti IA non producono direttamente un progetto Scratch, si chiede loro una descrizione dei blocchi, in italiano, che gli studenti ricostruiscono nell'editor — un passaggio che costituisce già di per sé un primo esercizio di lettura del codice. L'attività si articola in tre fasi.

Nella prima fase («il vibe») gli studenti, in coppia, formulano un prompt in linguaggio naturale del tipo: «Descrivimi, blocco per blocco e in italiano, uno script Scratch per un quiz che pone 3 domande di informatica, controlla le risposte e mostra il punteggio finale su 3. Usa una variabile chiamata punteggio. Spiega ogni blocco». L'osservazione guidata mette a fuoco quanto sia rapido ottenere una proposta funzionante e che cosa, in termini di comprensione, vada perduto: avere una soluzione pronta non significa averla capita.

Nella seconda fase («capire e correggere»), il cuore della proposta, si lavora su uno script proposto dall'IA contenente un solo difetto logico realistico: il quiz «funziona», ma rispondendo correttamente a tutte e tre le domande mostra un punteggio di 1 su 3, perché nell'ultima domanda compare il blocco «porta punteggio a 1» al posto di «cambia punteggio di 1», che sovrascrive il punteggio accumulato invece di incrementarlo. Gli studenti eseguono lo script, annotano il valore della variabile dopo ogni domanda (0, 1, 2, poi di nuovo 1), individuano il passo in cui il valore scende in modo inatteso, correggono e verificano. Un indizio particolarmente istruttivo è il risultato «impossibile»: chi risponde correttamente solo alle prime due domande ottiene 2, più di chi risponde correttamente a tutte e tre. La confusione tra «porta a» e «cambia di» è del resto tra le più comuni nei principianti, e l'esempio la rende visibile.

Per differenziare il lavoro tra i gruppi sono previste due varianti dello stesso script con difetti di natura diversa: l'azzeramento della variabile collocato per errore alla fine del programma, che porta il punteggio finale sempre a 0, e una condizione che confronta la risposta con un valore atteso sbagliato, per cui una risposta corretta non viene conteggiata. Il confronto in plenaria permette così di riconoscere tipi di difetto differenti: di aggiornamento della variabile, di ordine dei blocchi e di test. L'osservazione è sostenuta da una rubrica analitica su quattro livelli (iniziale, base, intermedio, avanzato) articolata in cinque aspetti del debug: comprendere che cosa fa il programma, tracciare la variabile, individuare il difetto, correggere e verificare con casi di prova, valutare criticamente la proposta dell'IA — incluso il riconoscimento dei casi non gestiti, come l'accettazione di una sola forma della risposta («8» ma non «otto»).

Nella terza fase («il progetto») i costrutti compresi e verificati vengono ricomposti in un artefatto più ampio, in cui l'uso dell'IA è subordinato alla comprensione e al controllo umano e in cui sono esplicitati l'obiettivo curricolare di riferimento e i criteri di verifica dell'apprendimento.

Sul piano operativo l'attività richiede un dispositivo con tastiera per coppia, connessione a Internet, un ambiente di programmazione a scopo didattico utilizzabile da browser e un assistente IA generativo ad accesso libero, privilegiando soluzioni gratuite e senza installazione. Per il docente è opportuna una precedente esperienza di programmazione testuale o a blocchi, per accompagnare lettura e correzione del codice. Poiché gli assistenti IA generativi hanno limiti di età, in classe vanno usati mediati dal docente o con account d'istituto, senza inserire dati personali degli alunni; strumenti diversi possono inoltre produrre risposte diverse allo stesso prompt: un elemento da gestire con cura in aula, ma anche un'occasione di confronto.

3 Allineamento alle Nuove Indicazioni Nazionali

La proposta si aggancia, in modo esplicito e verificabile, a obiettivi tratti letteralmente dalle Nuove Indicazioni Nazionali per il primo ciclo (DM 221 del 9/12/2025) [2]. Per le competenze attese al termine della classe terza in Informatica: «Comprendere il concetto di variabile e scrivere, con linguaggi di programmazione a scopi didattici, programmi che le usano anche strutturati in componenti modulari, valutandone la correttezza». Tra gli obiettivi specifici di apprendimento di Informatica: «Esaminare un algoritmo o un programma per capirne il comportamento, identificarne eventuali difetti e correggerli (debug)» e «Valutare l'esito di un algoritmo o di un programma seguendone i passi e tenendo traccia del valore delle variabili».

La proposta si collega inoltre, per Tecnologia, alla conoscenza relativa alla «Valutazione critica dei dati reperiti su rete e delle informazioni fornite da sistemi di intelligenza artificiale generativa» — realizzata in particolare dalla variante con la risposta attesa sbagliata, che mostra come una proposta dell'IA possa sembrare corretta e non esserlo — e al principio guida del primo ciclo secondo cui occorre che «siano gli esseri umani a mantenere il controllo sulle decisioni basate su

sistemi informatici». Ogni fase dell'attività è ricondotta a uno di questi riferimenti mediante una tabella di mappatura che accompagna i materiali di lavoro.

4 Stato di sviluppo e prospettive

La proposta è allo stato di progettazione e non è ancora stata sperimentata in aula né in contesti di formazione. Nasce però da un'esigenza concreta e ricorrente, osservata dall'autrice in quattro anni di attività di formazione rivolta ai docenti come componente di un'Équipe Formativa Territoriale della Campania: gli insegnanti chiedono spesso quale sia la differenza tra il vibe coding e la programmazione «classica» e come collocarlo nella didattica dell'Informatica. La proposta risponde a questa domanda trasformandola in esperienza diretta di lettura, verifica e correzione del codice.

Lo sviluppo naturale del lavoro è duplice: da un lato la sperimentazione in classe della micro-attività, con la raccolta di evidenze sulla comprensione algoritmica degli studenti; dall'altro la conduzione della proposta come laboratorio di formazione per docenti in un formato esteso, in cui i partecipanti generano il codice con l'IA, ne analizzano e correggono il comportamento tracciando il valore delle variabili e progettano una micro-attività per la propria classe allineata a obiettivi curricolari espliciti, con discussione finale su rischi, accorgimenti metodologici e modalità di valutazione. I materiali di lavoro — scheda per i partecipanti, documento di confronto tra vibe coding e programmazione classica, libreria di prompt per creare materiali didattici, script-esempio con difetto e soluzione, varianti per la differenziazione e griglia di valutazione del debug — sono resi disponibili in una cartella condivisa [3].

Bibliografia

[1] Il Post. «Con l'intelligenza artificiale diventeremo tutti programmatori?». Il Post, sezione Tecnologia, 24 marzo 2025. https://www.ilpost.it/2025/03/24/intelligenza-artificiale-vibe-coding/. Ultimo accesso: 11 settembre 2026.
[2] Ministero dell'Istruzione e del Merito. Nuove Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione. Decreto Ministeriale n. 221 del 9 dicembre 2025.
[3] Pinto, Simona. Dal Mattone al Progetto – Materiali utili. https://drive.google.com/drive/folders/1MyqM2ef552SLp43IgT8vBbfvfeo93jaW. Ultimo accesso: 11 settembre 2026.
[4] Treccani – Istituto della Enciclopedia Italiana. «Vibe coding». Vocabolario on line, sezione Neologismi (2025). https://www.treccani.it/vocabolario/neo-vibe-coding_(Neologismi)/. Ultimo accesso: 11 settembre 2026.

Future Climate Lab: ambienti virtuali, coding e AI per il pianeta del futuro

Alessia Galli[1], Lara Rollo[2]
[1] IIS Ettore Majorana
ing.alessiagalli@gmail.com
[2] IC Velletri Sud Ovest
rollo.lara@gmail.com

Abstract

La rapida evoluzione tecnologica richiede un'educazione digitale capace di andare oltre l'uso passivo degli strumenti, promuovendo negli studenti di 11-14 anni la capacità di osservare, sperimentare e interrogare criticamente i sistemi computazionali e le tecnologie di Intelligenza Artificiale. Il presente contributo teorico-sperimentale descrive l'architettura didattica di *Future Climate Lab*, progetto extracurricolare finalizzato all'apprendimento dell'informatica e allo sviluppo dell'AI literacy attraverso un percorso laboratoriale realizzato con la piattaforma Delightex.

Gli studenti progettano e sperimentano simulazioni tridimensionali interattive, utilizzando programmazione visuale, variabili, eventi e strutture di controllo. Il percorso prende avvio da CoBlocks, con cui vengono definiti esplicitamente comportamenti e relazioni causa-effetto. Successivamente viene introdotto un *AI-powered 3D character* e la configurazione delle *AI Skills*: gli studenti formulano richieste in linguaggio naturale e osservano come una componente AI possa interpretarle e associarle a una Skill, la cui azione è comunque definita mediante programmazione visuale. Le due modalità non sono presentate come paradigmi alternativi, ma come differenti forme di interazione con funzionalità computazionali programmate.

La fase sperimentale prevede il testing del sistema attraverso richieste equivalenti, ambigue o non pertinenti, rilevando attivazioni corrette, mancate o improprie. Gli studenti analizzano così variabilità, ambiguità, errori di interpretazione, falsi positivi e falsi negativi. Il confronto tra comportamento atteso e osservato, seguito dalla modifica e dal retest delle *AI Skills*, introduce una logica analoga al debugging: formulare ipotesi, testare, osservare, verificare e correggere.

La simulazione climatica costituisce il contesto applicativo nel quale integrare gli obiettivi informatici con l'AI literacy.

1 Domanda iniziale e struttura dell'attività

La sperimentazione è stata guidata dalla domanda: un'attività laboratoriale basata sulla manipolazione e sul testing di AI Skills può favorire, in studenti di 11–14 anni, lo sviluppo di prime competenze di AI literacy relative all'interpretazione delle richieste, alla variabilità del comportamento, all'errore, alla verifica e alla necessità del controllo umano?

Sono state individuate tre dimensioni osservabili: distinguere un'azione programmata da una funzionalità che interpreta una richiesta linguistica; riconoscere che una richiesta in linguaggio naturale può essere correttamente interpretata, associata a una Skill diversa o non attivare quella attesa; adottare un atteggiamento di verifica, evitando di considerare automaticamente corretto l'output del sistema. Non costituiva invece obiettivo la comprensione degli algoritmi interni o dei meccanismi tecnici del modello AI.

Sul piano informatico, il percorso ha sviluppato algoritmi e programmi, variabili, strutture di controllo, relazione input-elaborazione-output, debugging e descrizione del comportamento di un sistema mediante regole. A questi si sono affiancati obiettivi di AI literacy: riconoscere ambiguità ed errori, distinguere comportamento programmato e interpretazione linguistica, verificare sistematicamente gli output e sviluppare un atteggiamento critico e non antropomorfico nei confronti dell'IA.

Il gruppo classe era costituito da studenti provenienti da più classi della scuola secondaria di primo grado, tutti in possesso di una conoscenza pregressa della programmazione a blocchi e della piattaforma Delightex.

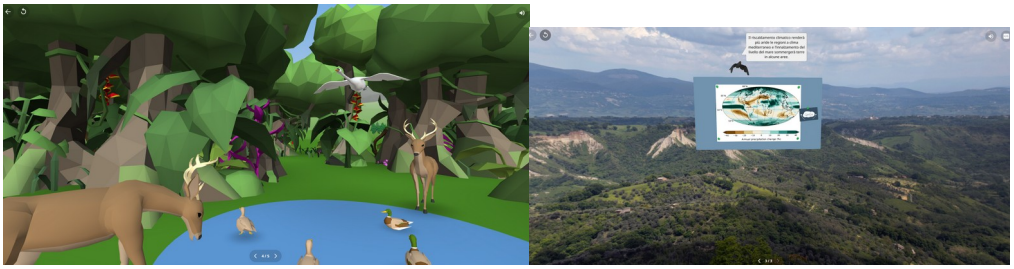


Figura 1: immagine dei progetti sviluppati

1.2 “Quando il programma segue le regole”

Inizialmente è stata costruito la base computazionale. In gruppi di quattro, gli studenti hanno realizzato una scena mediante programmazione visuale, lavorando su sequenza, eventi, variabili, selezione, iterazione e debugging. Attraverso semplici esercizi hanno osservato la relazione **input → elaborazione → output**, costruendo il riferimento computazionale necessario per introdurre la componente AI.

1.3 “Quando la richiesta entra nel programma”

È stato evidenziato che l’AI non genera autonomamente il comportamento dell’ambiente: l’azione è definita attraverso la Skill e la logica programmata. Gli studenti hanno quindi confrontato un evento esplicito, ad esempio “Quando premo il pulsante, aumenta gli alberi”, con una richiesta linguistica come “Vorrei rendere la foresta più verde”, osservando che la novità riguarda l’interpretazione della richiesta e l’associazione alla Skill, non la sostituzione della programmazione

1.4 “Mettiamo alla prova l'AI”

Ogni gruppo ha testato una Skill attraverso parafrasi, richieste ambigue e non pertinenti, registrando richiesta, Skill attesa, Skill attivata, risultato e classificazione. Per la Skill *Riforesta* (“aumenta la vegetazione”), ad esempio, “Aggiungi alberi” poteva produrre un’attivazione corretta, “Migliora il clima” un’attivazione impropria, mentre “Quanto è alta la montagna?” poteva non attivare alcuna Skill. Sono state quindi distinte attivazioni corrette, mancate e improprie, successivamente ricondotte, in forma semplificata, a falsi negativi e falsi positivi nei casi in cui fosse stato definito il comportamento atteso.

1.5 “Possiamo migliorare il comportamento?”

Gli studenti hanno modificato le descrizioni delle AI Skills, confrontando formulazioni generiche e più specifiche, e hanno ripetuto parte dei test. La discussione ha permesso di evidenziare l’opacità del sistema: sono osservabili richiesta, Skill attivata, azione programmata e risultato, ma non il processo interno attraverso il quale la richiesta viene interpretata.

2 Raccolta dati e risultati

La sperimentazione ha utilizzato quattro fonti di evidenza: questionario pre/post, schede di testing delle AI Skills, osservazione sistematica del docente e riflessione individuale conclusiva. Il questionario comprendeva 10 item relativi alla distinzione tra regola programmata e

interpretazione linguistica, riconoscimento dell'ambiguità, errore e comportamento atteso, verifica dell'output e possibilità di modificare la configurazione.

I risultati mostrano un andamento positivo sugli indicatori monitorati

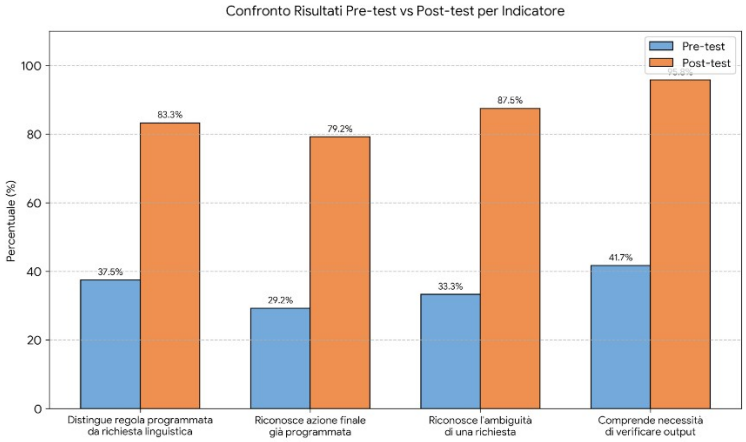


Tabella 1: Confronto indicatori pre/post

Sono state analizzate complessivamente 192 interazioni: 139 (72,4%) con attivazione corretta, 27 (14,1%) con mancata attivazione, 19 (9,9%) con attivazione impropria e 7 (3,6%) con corretta non attivazione. I dati non sono interpretati come misura dell'accuratezza dell'AI, ma come materiale per analizzare errore, ambiguità e incertezza. Le richieste ambigue hanno prodotto la maggior parte delle attivazioni improprie.

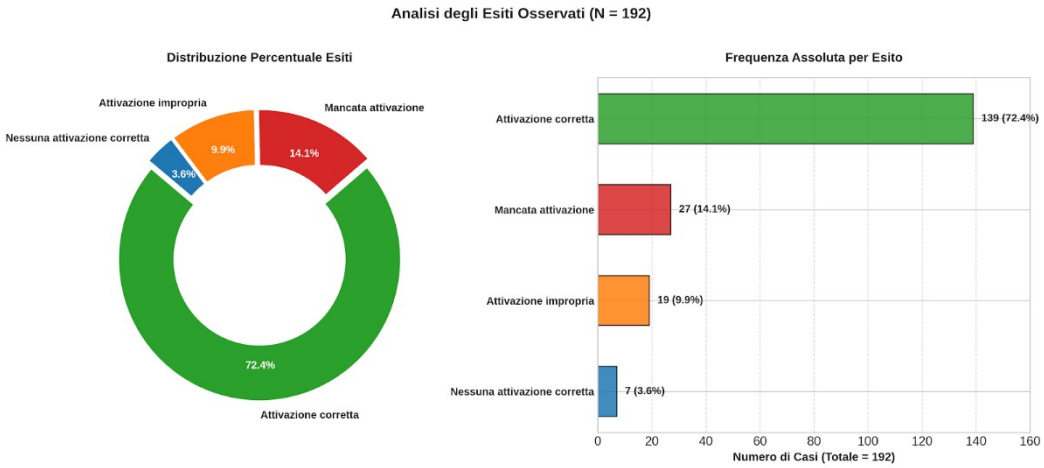


Tabella 2: Analisi degli esiti

La modifica delle descrizioni delle AI Skills ha favorito un miglioramento nelle prove successive, evidenziando agli studenti che la configurazione della Skill può influire sull'associazione tra richiesta e funzionalità. Particolarmente significativo è stato il cambiamento nell'approccio all'errore: dalla formulazione iniziale "il computer ha sbagliato", gli studenti sono passati a ipotesi verificabili, come "forse la frase è troppo generica" o "proviamo a dirla in un altro modo".

Rispetto agli obiettivi di AI literacy sono stati individuati quattro livelli progressivi: riconoscere l'associazione tra richieste linguistiche e funzionalità; sperimentare modificando intenzionalmente le richieste e confrontandone gli esiti; analizzare l'errore distinguendo attivazioni corrette, mancate e improprie; verificare criticamente il risultato senza considerare automaticamente corretto il comportamento del sistema. I primi due livelli risultano raggiunti dalla maggioranza del gruppo; l'analisi dell'errore appare adeguata, seppure con residue difficoltà terminologiche, mentre la verifica critica risulta raggiunta dalla quasi totalità degli studenti.

3 Limiti

I risultati devono essere interpretati alla luce di alcuni limiti.

Il gruppo era costituito da studenti provenienti da più classi della stessa scuola secondaria di primo grado con un numero dei partecipanti limitato; non è quindi possibile generalizzare quantitativamente gli esiti.

L'attività è stata inoltre realizzata utilizzando una specifica piattaforma e una specifica configurazione delle AI Skills; i comportamenti osservati non possono essere automaticamente estesi ad altri sistemi di IA.

La sperimentazione non ha consentito infine di osservare i processi interni del sistema: le conclusioni riguardano pertanto esclusivamente il comportamento osservabile e verificabile.

Anche la simulazione climatica aveva finalità esclusivamente didattiche: le relazioni tra variabili erano semplificate per consentire agli studenti di concentrarsi sui processi computazionali e sull'interazione con la componente AI.

4 Conclusioni

I risultati suggeriscono che un'attività centrata sul testing di una funzionalità AI possa trasformare l'interazione con l'Intelligenza Artificiale da esperienza prevalentemente d'uso a esperienza di osservazione, formulazione di ipotesi e verifica.

L'elemento più significativo riguarda il cambiamento progressivo nell'approccio all'errore. Gli studenti sono passati dalla ricerca della "frase giusta" alla costruzione di semplici prove sperimentali: hanno formulato parafrasi, introdotto ambiguità, ripetuto richieste, confrontato configurazioni e classificato gli esiti. In questo modo, una logica propria del debugging, già presente nella programmazione tradizionale, è stata trasferita anche all'interazione con una funzionalità AI.

Programmazione e AI non sono state presentate come alternative: la prima definisce esplicitamente le azioni del sistema, mentre la seconda introduce una componente di interpretazione della richiesta linguistica che può determinare quale funzionalità venga attivata. L'esperienza ha quindi consentito di osservare un'architettura ibrida, nella quale componenti programmabili e componenti basate sull'interpretazione linguistica operano in modo complementare.

I risultati suggeriscono infine che l'AI Literacy possa essere sviluppata non solo attraverso spiegazioni teoriche, ma anche attraverso attività che pongano gli studenti nella condizione di interrogare, testare, confrontare e verificare il comportamento di un sistema AI, promuovendo un approccio attivo e non passivo alla tecnologia.

L'attività ha inoltre evidenziato l'importanza di introdurre esplicitamente i concetti di ambiguità, variabilità, errore, falso positivo, falso negativo e opacità.

L'esito più significativo della sperimentazione non è stato dunque che gli studenti abbiano imparato "come funziona l'intelligenza artificiale", ma che abbiano iniziato a sviluppare una modalità diversa di rapportarsi ad essa:

formulare un'ipotesi → provare → osservare → confrontare → verificare → eventualmente modificare.

In questa prospettiva, l'AI Literacy emerge come capacità di non assumere automaticamente come corretto il comportamento di un sistema intelligente, ma di sottoporlo a osservazione critica e verifica.

Bibliografia

[1] Barbuto, E. (2024). *Realtà virtuale e aumentata per la didattica. Una guida per i docenti all'utilizzo delle tecnologie immersive di vario genere nell'esperienza didattica.* Edises

[2] Resnick, M. (2017). *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. MIT Press.

[3] European Commission, Joint Research Centre (2025). *DigComp 3.0: European Digital Competence Framework for Citizens – Fifth Edition*. Publications Office of the European Union

[4] Ministero dell'Istruzione e del Merito (2025). *Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione*

[5] Ministero dell'Istruzione e del Merito (2025). *Linee guida per l'introduzione dell'Intelligenza Artificiale nelle istituzioni scolastiche*. Decreto Ministeriale 9 agosto 2025, n. 166.

[6] https://www.delightex.com/edu - Piattaforma di programmazione e creazione 3D

[7] https://forum.edu.delightex.com/ - Documentazione tecnica per l'integrazione di script e configurazione di agenti intelligenti

Dichiarazione sull'utilizzo dell'Intelligenza Artificiale

Gli strumenti di Intelligenza Artificiale generativa sono stati utilizzati esclusivamente come supporto alla revisione linguistica, alla sintesi e alla riorganizzazione del testo. Le autrici hanno verificato e validato i contenuti, assumendo la piena responsabilità del contributo finale.

Con il mare davanti agli occhi

MIT App Inventor e la didattica dell'informatica in un contesto periferico

Pierangelo Indolfi
Politecnico di Bari; ITS Academy Apulia Digital; già docente IISS Marconi-Hack, Bari
pierangelo.indolfi@poliba.it

Abstract

Il contributo propone il framework Peripheral Innovation in Computing Education, elaborato a partire da trent'anni di insegnamento dell'informatica a Bari e nel contesto pugliese. L'autore argomenta che la periferia geografica e culturale non è un deficit da compensare, ma una variabile pedagogica attiva. MIT App Inventor, orientato alla "computational action", si rivela uno strumento efficace in questi contesti: abbassa le barriere di accesso, consente la produzione di artefatti digitali su problemi reali e introduce concetti informatici rigorosi, dall'OOP alla sensoristica alla conversione analogico-digitale. Il contributo discute inoltre le implicazioni didattiche della circolare ministeriale 2024 che vieta l'uso degli smartphone nella scuola del primo ciclo.

1 Introduzione

Insegnare informatica in un contesto geograficamente periferico è una pratica con logiche proprie. Il contesto può diventare una risorsa pedagogica, a condizione che il docente sappia riconoscerlo e usarlo invece di subirlo. Questa è la tesi del contributo, che propone il framework Peripheral Innovation in Computing Education a partire da trent'anni di esperienza nella scuola secondaria superiore e nell'istruzione terziaria a Bari e nel contesto pugliese.

Il framework nasce da evidenze documentate. Nel 2023 studenti coordinati dall'autore hanno vinto il MIT App Inventor Summer Appathon con un'app per il monitoraggio della qualità dell'aria; lo stesso anno l'autore ha ricevuto il Teacher Excellence Award della App Inventor Foundation. Nel giugno 2024, ancora in servizio, ha partecipato come coach al summer camp Girls Code It Better presso l'IC Minzele Parini di Putignano, lavorando con alunni di dieci anni. Questi risultati sono spiegabili attraverso le caratteristiche del contesto in cui sono maturati.

La periferia non è solo geografica: tra gli studenti che hanno prodotto questi risultati ci sono ragazzi con origini di immigrazione recente, condizioni di apprendimento non standard, contesti non privilegiati. Che la scuola pubblica, attraverso App Inventor, abbia funzionato da ascensore sociale è una delle evidenze più significative del framework.

2 Periferia strutturale e potenziale pedagogico

La letteratura sulla didattica dell'informatica nei contesti svantaggiati tende a inquadrare la periferia come deficit da compensare [8]. Questo contributo propone una lettura alternativa: il contesto è una variabile pedagogica attiva. Quando il problema che un'app deve risolvere è reale e urgente per gli studenti, la motivazione intrinseca aumenta [10]. Il contesto pugliese offre problemi reali: qualità dell'aria, fragilità infrastrutturale, divario digitale intergenerazionale.

Nel caso italiano, la periferia ha radici storiche precise: l'Olivetti Elea 9003 (1959), primo elaboratore commerciale a transistor al mondo [6], fu progetto di Mario Tchou [11]; il BCOS, sviluppato a Bari tra il 1980 e il 1986 [7], attestava la vitalità tecnica locale. La dismissione della divisione elettronica [5] ha prodotto un gap formativo che le multinazionali ICT, tornate a Bari, faticano a colmare.

3 App Inventor come strumento di computational action: un esempio concreto

MIT App Inventor consente agli studenti di costruire applicazioni reali fin dalle prime ore di laboratorio. L'ambiente a blocchi non nasconde la struttura object oriented del codice: classi, metodi ed eventi sono visibili e usati concretamente sin dall'inizio, rendendo lo strumento didatticamente rigoroso oltre che accessibile [10].

Un'attività particolarmente efficace, sperimentata con studenti della scuola secondaria superiore, è quella che si potrebbe chiamare esplora il mondo fisico con il tuo smartphone: gli studenti costruiscono applicazioni che leggono i sensori del dispositivo, ad esempio accelerometro, giroscopio, sensore di luminosità o di pressione atmosferica, e ne visualizzano i valori in tempo reale. L'attività introduce in modo concreto la conversione analogico-digitale: dati fisici continui vengono trasformati in valori numerici discreti che il programma può elaborare. Il passaggio, spesso presentato in astratto, diventa visibile: lo studente muove il dispositivo e vede i valori cambiare sullo schermo, capendo che una macchina trasforma il mondo fisico in informazione digitale.

L'attività può essere estesa in due direzioni: il networking, con due dispositivi che si scambiano dati via rete locale introducendo il modello client-server; la data science accessibile [4], raccogliendo ed elaborando i valori dei sensori per ricavarne informazioni interpretabili.

La progressione è coerente con il principio del learning by doing: si parte da un problema concreto, si costruisce qualcosa che funziona, si sistematizzano a posteriori i concetti teorici sottostanti.

L'attività sulla sensoristica persegue obiettivi formativi riferibili alle Indicazioni Nazionali per l'informatica [3]. Al termine, gli studenti sono in grado di: raccogliere e analizzare dati attraverso i sensori del dispositivo (O-S-N-1); valutare rappresentazioni alternative dello stesso fenomeno fisico, ad esempio l'accelerazione espressa in m/s^2 o come variazione visiva sullo schermo (O-S-D-1); progettare applicazioni modulari sfruttando la struttura a classi, metodi ed eventi resa esplicita da App Inventor (O-S-P-6); combinare programmazione e servizi di rete per trasmettere dati tra dispositivi (O-S-R-2); tenere conto dei requisiti degli utenti nella definizione dell'interfaccia (O-S-N-2).

L'attività si articola in tre fasi progressive: nella prima, esplorazione, gli studenti leggono i valori dei sensori e ne osservano la variazione in tempo reale; nella seconda, elaborazione, costruiscono logiche di filtraggio e visualizzazione; nella terza, comunicazione, trasmettono i dati via rete a un secondo dispositivo. Ciascuna fase produce un artefatto funzionante e costituisce la base per la successiva.

4 Il framework: Peripheral Innovation in Computing Education

A partire dall'esperienza descritta, si propone un framework articolato in tre principi:

Principio 1. Il contesto come variabile pedagogica attiva. I problemi reali e locali orientano la scelta degli artefatti digitali da costruire e aumentano la motivazione intrinseca degli studenti.

Principio 2. La continuità educativa come moltiplicatore. La prospettiva pedagogica del learning by doing attraversa tutti i livelli formativi, dalla primaria alla secondaria superiore fino all'istruzione terziaria: cambiano gli strumenti adottati caso per caso, non l'approccio. Lo stesso App Inventor non è confinato alla fascia scolare: al Global Appathon 2026 un team di medici guidato da Nicholas Kwok ha presentato MoleCheck, un'applicazione che utilizza l'intelligenza artificiale per il triage del rischio di melanoma [1], a dimostrazione che lo stesso ambiente può sostenere tanto i primi passi nella programmazione quanto applicazioni professionali mature in ambito clinico.

Principio 3. Il docente come nodo di rete territoriale. Il docente connette studenti, istituzioni locali e competizioni internazionali: un ruolo che va sistematizzato come parte integrante della pratica didattica.

5 Discussione

L'esperienza qui descritta permette di rileggere il primo principio del framework: il gap strutturale già richiamato nella Sezione 2, le multinazionali ICT che si insediano a Bari senza trovare abbastanza informatici formati, non è una conferma della marginalità del territorio, ma un'evidenza che la periferia non è un deficit naturale bensì il risultato di scelte storiche reversibili. Se il gap fosse di talento, la vittoria all'Appathon del 2023 di studenti da contesti non privilegiati non si spiegherebbe; se è di sistema, come sostiene il framework, quella vittoria diventa un caso limite che rivela cosa succede quando il contesto è usato come risorsa invece che subìto come limite.

Il paradosso normativo della circolare ministeriale 2024 costituisce, in questa cornice, un banco di prova per il framework, non un argomento a sé. Nel 2024 il Ministero dell'Istruzione ha vietato gli smartphone nella scuola del primo ciclo anche per attività didattiche [13], escludendo milioni di studenti dal mobile learning nella fascia 6-13 anni, proprio quella in cui la ricerca indica i benefici più duraturi per il pensiero computazionale [2]. Se il contesto, incluso il dispositivo mobile che gli studenti già usano quotidianamente, è una variabile pedagogica da orientare e non un rischio da eliminare, quel divieto è esattamente l'errore opposto a quanto propone questo contributo: elimina la variabile invece di attivarla. Come osserva l'UNESCO, il problema non è il dispositivo ma la mancanza di mediazione didattica [12]; la ricerca MIT documenta come chi usa App Inventor sviluppi competenze computazionali e self-efficacy superiori [10]. La contraddizione con le Linee Guida STEM del MIM (D.M. 221/2025), che promuovono il coding in tutti i cicli citando le Indicazioni CINI [9], mostra a livello di sistema la stessa tensione che il framework affronta a livello di classe.

Questa lettura resta un'interpretazione qualitativa a posteriori. Uno studio comparativo sarebbe qui difficile da realizzare: il divieto è una direttiva nazionale, non una scelta variabile da istituto a istituto, e un confronto sistematico richiederebbe di individuare scuole o docenti che non la applicano, un oggetto di studio più delicato che semplicemente assente dalla letteratura. Il framework resta quindi un'ipotesi di lavoro per chi insegna in contesti periferici, non un risultato generalizzabile in senso statistico.

6 Conclusioni

Il framework Peripheral Innovation in Computing Education suggerisce che insegnare informatica in periferia significhi lavorare con il contesto, non nonostante il contesto. MIT App Inventor si è rivelato uno strumento capace di far fare informatica rigorosa a partire da problemi reali, con una soglia di accesso abbassata ma senza rinunciare al rigore.

Il testimone da passare alle nuove generazioni di informatici pugliesi è semplice: si può essere protagonisti del futuro digitale con il mare davanti agli occhi. E per chi è dovuto partire, esiste sempre il percorso inverso: tornare, con il mare a sinistra.

Bibliografia

[1] App Inventor Foundation (2026). Meet the Winners of the 2026 Global Appathon. https://www.appinventor.org/news/ga-2026-winnersannouncement

[2] Bers, M.U. (2018). Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom. Routledge.

[3] CINI – Consorzio Interuniversitario Nazionale per l'Informatica (2017). Proposta di Indicazioni

Nazionali per l'Informatica nella Scuola. https://www.consorzio-cini.it/images/Proposta-Indicazioni-Nazionali-Informatica-Scuolanumerata.pdf

[4] Elhashemy, H. (2023). From Curious Kids to Data Wizards: An Empowerment Approach to Data Science Education. M.S. Thesis, TU Munich / MIT CSAIL. Supervisors: H. Abelson, T. Michaeli.

[5] Gallino, L. (2003). La scomparsa dell'Italia industriale. Einaudi.

[6] Gazzarri, M. (2021). Elea 9003. Storia del primo calcolatore elettronico italiano. Edizioni di Comunità.

[7] Indolfi, P. (2023). La discesa nel labirinto. Avventure di un programmatore. Menzione speciale, I Premio Letterario Nazionale Tommaso Di Ciaula, 9 novembre 2023.

[8] Margolis, J. et al. (2008). Stuck in the Shallow End: Education, Race, and Computing. MIT Press.

[9] MIM – Ministero dell'Istruzione e del Merito (2025). Linee guida per le discipline STEM. D.M. n. 221 del 9 dicembre 2025, G.U. 27 gennaio 2026. https://www.istruzione.it

[10] Pang, H.H. (2022). Computational Action in Action: Process and Tools that Empower Students to Make a Real-world Impact Using Technology. M.Eng. Thesis, MIT CSAIL. Supervisors: H. Abelson, C. Breazeal.

[11] Parolini, G. (2015). Mario Tchou. Ricerca e sviluppo per l'elettronica Olivetti. EGEA.

[12] UNESCO (2010). Policy Brief: Mobile Learning for Quality Education and Social Inclusion. UNESCO IITE. https://iite.unesco.org/files/policy_briefs/pdf/en/mobile_learning.pdf

[13] Valditara, G. (2024). Circolare ministeriale sull'uso degli smartphone nella scuola del primo ciclo, A.S. 2024-2025. Ministero dell'Istruzione e del Merito, Roma.

L’informatica entra al Classico: un percorso graduale di programmazione e pensiero computazionale al Liceo "Frezzi - B. Angela" di Foligno

Enrica Donati, Roberta Barluzzi
Liceo Classico “Federico Frezzi – Beata Angela” Foligno
enrica.donati@scuola.istruzione.it
roberta.barluzzi@scuola.istruzione.it

Abstract

Il contributo descrive le app realizzate da due classi con App Lab di code.org, evidenziando i tre concetti chiave attorno ai quali sono state progettate: 1) la programmazione come strumento per l'elaborazione dinamica e creativa dei contenuti disciplinari; 2) l'acquisizione dei fondamenti della programmazione a eventi e della gestione degli identificatori (ID); 3) lo sviluppo del pensiero computazionale come ponte tra il rigore della sintassi classica e la logica algoritmica

1 Attività proposta

Creazione di App con App Lab [2] di www.code.org [7].

Negli ultimi anni, il Liceo Classico "Frezzi – B. Angela" di Foligno ha sperimentato alcune attività di innovazione didattica nel campo dell’informatica in un contesto didattico tradizionalmente umanistico. In un indirizzo dove l'informatica non è prevista come disciplina curricolare, si è scelto di sfruttare appuntamenti internazionali come la EU Code Week e l'Ora del Codice per proporre agli studenti non un semplice utilizzo di strumenti digitali, ma una riflessione profonda sui meccanismi alla base dell'automazione e del pensiero computazionale anticipando nel secondo ciclo l'approccio didattico promosso dal recente Decreto Ministeriale 9 dicembre 2025, n. 221 [9].

L'obiettivo primario è quello di ridurre il divario tra studi classici e scientifici, dimostrando che la razionalità analitica necessaria per la decodifica di un testo greco o latino è la medesima che governa la scrittura di un algoritmo.

Tra le attività realizzate, sono presentate in questo documento le app “Alla scoperta dell’Intelligenza Artificiale” (app partecipante al concorso di Programma il Futuro 2024 ‘Il mio futuro con l’intelligenza Artificiale’; classe 3 A Classico a.s. 2023-2024) e “La sintesi delle proteine” (app partecipante al concorso di Programma il Futuro 2026 ‘L’informatica come strumento di collaborazione e cooperazione’; classe 4 B Classico a.s. 2025-2026) realizzate con App Lab [2] di www.code.org [7].

2 Contesto

Gli studenti avevano già utilizzato "Intro to App Lab" [6] negli anni precedenti, pur avendo per la maggior parte una prima esperienza di coding limitata a queste occasioni e solo rari casi di competenze pregresse più avanzate.

Le insegnanti coinvolte non sono di informatica, ma di Matematica e Fisica (laurea in Matematica) e Scienze Naturali (laurea in Biologia). Entrambe hanno sviluppato competenze sul

pensiero computazionale e sul coding frequentando corsi di formazione online (come quelli del prof. Bogliolo) e seguendo piattaforme come Programma il Futuro [10] e Code.org [7].

Perché presentare il lavoro a ITADINFO26? La loro esperienza è un esempio concreto per altre scuole: dimostra che per proporre attività come Hour of Code non servono competenze informatiche approfondite, né nei docenti né negli studenti. Si tratta di percorsi stimolanti e di forte valore didattico, capaci di accendere la curiosità.

3 Obiettivi didattici

- Programmazione visuale: Sperimentare il coding a blocchi per superare le barriere della programmazione testuale e focalizzarsi sulla logica degli algoritmi.
- Logica degli eventi: Comprendere il flusso non sequenziale dei programmi che reagiscono a input specifici (es. clic dell'utente).
- Astrazione e controllo UI: Associare elementi grafici in modo univoco ai loro ID per modificarne dinamicamente testo, colori, immagini e suoni (anche via URL).
- Design e Codice: Imparare a navigare fluidamente tra la modalità di progettazione dell'interfaccia (Design Mode) e quella di programmazione.
- Strutturazione dei contenuti: Organizzare una narrazione logica e chiara su un tema di studio (IA o Sintesi Proteica), creando una risorsa riutilizzabile e modificabile tramite remix.
- Collaborazione e creatività: Lavorare in gruppo con ruoli definiti, usando il coding come strumento creativo, di conoscenza o per risolvere problemi sociali.
- Orientamento STEM: Stimolare l'interesse verso futuri percorsi di studio in ambito scientifico e tecnologico.

4 Sviluppo delle attività

4.1 Intro to App Lab (Hour of Code)

In una prima fase la classe affronta in autonomia (singolarmente o a coppie) i 15 step del percorso "Intro to App Lab" [6,8] su Code.org. L'unità guida gli studenti dalla manipolazione di elementi alla creazione di software, alternando la "Modalità Design" per la grafica e la "Modalità Codice" per la logica.

- **Impostazione delle proprietà (Livelli 2-4)**

  I ragazzi modificano attributi visivi (testo, colori, dimensioni) di oggetti preesistenti con il blocco setProperty(). In questa fase apprendono la precisione sintattica, come l'uso delle virgolette per le stringhe e la selezione del corretto ID dall'interfaccia.

- **Programmazione a eventi (Livelli 5-7)**

  Si introduce l'interattività con il blocco onEvent(), insegnando che il codice può attivarsi "in attesa" di un'azione dell'utente (come un clic). Gli studenti apprendono anche la regola fondamentale del non annidamento dei blocchi di evento.

- **Modalità Design e Identificatori (Livelli 10-13)**

  Passando alla progettazione dell'interfaccia, i ragazzi aggiungono autonomamente nuovi elementi a schermo. L'obiettivo principale è imparare ad assegnare "Good ID", ossia identificatori univoci e descrittivi per richiamare ed elaborare gli oggetti via codice.

4.2 Realizzazione dell'app con App Lab

Viene realizzata l'app con App Lab [2] sfruttando le funzionalità avanzate rispetto a Intro to App Lab [6].
Il flusso di lavoro si articola in fasi sequenziali e cicliche:

- **Ideazione**
  Si definisce l'argomento e lo scopo dell'app. I ragazzi approfondiscono il tema, si documentano e selezionano le informazioni.

- **Progettazione (tramite schede dedicate) – App Design** [1].
  La progettazione avviene su carta. È ideata la grafica delle schermate con gli oggetti che le compongono e i collegamenti logici tra gli oggetti.

- **Modalità Interfaccia (UI) – Aggiungi elementi all'Interfaccia Utente** [4,5,11].
  Gli studenti vengono introdotti alla modalità interfaccia utente di App Lab.
  La UI è l'insieme degli elementi visivi (pulsanti, immagini, menù) con cui l'utente controlla l'app. Si trascinano gli elementi sullo schermo dalla cassetta degli attrezzi e se ne modificano le caratteristiche nella scheda Proprietà. Ogni oggetto deve avere un ID univoco, significativo e descrittivo della sua funzione (es. leftbutton). In app con più schermate, gli ID tengono conto che ogni oggetto appartiene a una sola schermata (relazione 1:1), mentre ogni schermata contiene più oggetti (relazione 1: N).

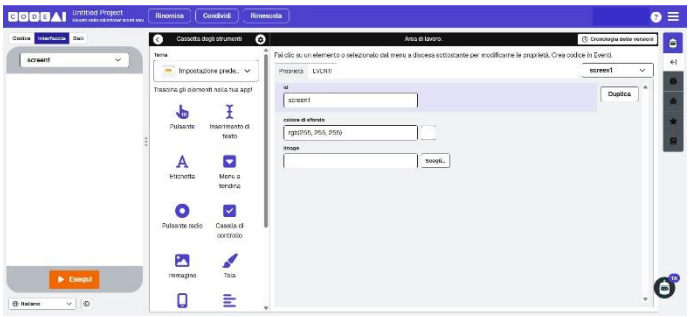


Figura 1: Modalità Interfaccia

- **Modalità Codice – Aggiunta di blocchi onEvent** [3,11].
  Nelle applicazioni interattive di App Lab, la **gestione degli eventi** (Event handling) permette al programma di rispondere alle azioni dell'utente attivando specifiche funzioni. Per configurarla, basta passare alla **Modalità codice**, trascinare il blocco `onEvent` nell'area di lavoro e selezionare dai menu a discesa l'**ID** dell'elemento visivo interessato e il **tipo di evento** a cui reagire (ad esempio il *click*).Il blocco onEvent è detto anche 'ascoltatore di eventi (Event Listener)'; è il sensore che rileva l'evento nell'interfaccia utente su uno specifico elemento.

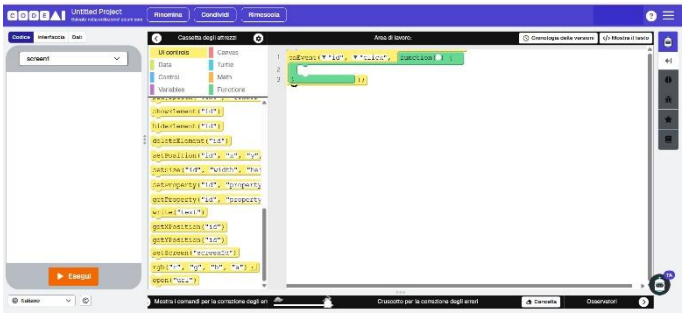


Figura 2: Modalità Codice, blocco onEvent

- **Modalità Codice – Aggiunta codice per la gestione degli eventi** [3,11,12].
  Si aggiunge codice all'interno della funzione che si vuole eseguire quando l'utente attiva l'evento. Tramite l'istruzione `setScreen` è possibile passare da una schermata all'altra. La funzione verrà eseguita ogni volta che l'utente attiva l'evento specifico.

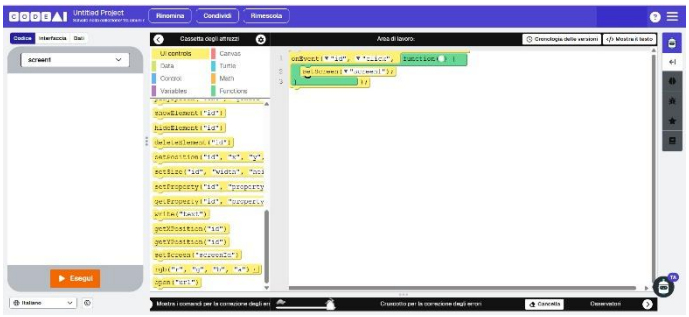


Figura 3: Modalità Codice, istruzioni del blocco onEvent

- **Test e debugging [12].**
  Il processo di programmazione dell'app si conclude secondo lo schema Esegui - Verifica - Correggi. Si revisiona l'aspetto grafico rispondendo alle domande: La modalità di navigazione degli utenti tra le pagine è intuitiva? Il design / layout è chiaro e presenta bene le informazioni? C'è qualcosa che aggiungeresti? C'è qualcosa che toglieresti?
  Gli errori di programmazione possono essere di sintassi e di logica. Gli errori di sintassi sono segnalati da un messaggio nel cruscotto per la correzione degli errori e da simboli accanto alle istruzioni nell'area di lavoro. Gli errori di logica riguardano ciò che il programma fa e sono più difficili da individuare.

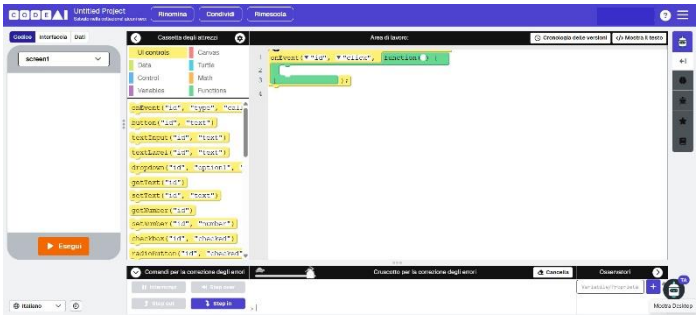


Figura 4: Debugging

5 Presentazione delle app realizzate

Le due app realizzate dalle classi sono:

"Alla scoperta dell'Intelligenza Artificiale"
I ragazzi divisi in quattro gruppi hanno realizzato l'app per la partecipazione al concorso 'Il nostro futuro con l'intelligenza artificiale' di Programma il Futuro 2024 (Classe 3° A classico a.s. 2023-2024). I contributi di ciascun gruppo sono stati uniti in un'unica app tramite la funzionalità 'Import screen'.
https://studio.code.org/projects/applab/EqEB3Zy3cRq1K4haAaIwWDoUQkU7uEgKYcYl8uK7zU

“La sintesi delle proteine”
Ciascuno studente della classe 4B classico a.s. 2025-2026 ha progettato una sua app e la classe ha proposto la migliore per partecipare al Concorso 2026 “Informatica come strumento di collaborazione e cooperazione” di Programma il Futuro.
https://studio.code.org/projects/applab/367a36f6-ae3f-41fb-98da-9f463816cce3

6 Osservazioni e sviluppi possibili

Non è stata proposta una griglia o un modulo per il feedback. I ragazzi si sono immediatamente appassionati all’attività, non riscontrando difficoltà con l’interfaccia di App Lab, intuitiva e basata sulla programmazione a blocchi.
Non si notano differenze di genere nell’esecuzione dei progetti.
L’attività può essere proposta anche alle classi del biennio.
Si può passare dalla programmazione a blocchi alla programmazione con Javascript direttamente da App Lab.
Si possono inserire elementi di programmazione più avanzati che richiedono maggiore confidenza con gli strumenti informatici.
Gli studenti possono approfondire in autonomia gli argomenti proposti.

7 Ringraziamenti

Si ringraziano le Dirigenti Scolastiche, professoressa Rosella Neri e dottoressa Maria Marinageli, insieme ai Collegi Docenti che hanno approvato e supportato le attività del “Progetto Coding” del Liceo “Frezzi-Beata Angela” di Foligno.

Sitografia

[1] Activity Guide Screen Design, disponibile a: https://docs.google.com/document/d/1QGuwv8brhQRgXewOgsEV1zy7qrhCMawqlMNP4WLNh9Y/edit?usp=sharing (consultato il 30/08/2026).
[2] App Lab, disponibile a: https://code.org/en-US/tools/app-lab (consultato il 30/08/2026).
[3] App Lab guides/code-mode, disponibile a: https://studio.code.org/courses/csd-2024/guides/responding-to-user-input (consultato il 30/08/2026).
[4] App Lab guides/design-mode, disponibile a: https://studio.code.org/courses/csd-2024/guides/design-mode (consultato il 30/08/2026).
[5] App Lab guides/design-mode-elements, disponibile a: https://studio.code.org/courses/csd-2024/guides/design-mode-elements (consultato il 30/08/2026).
[6] App Lab-intro, disponibile a: https://studio.code.org/teacher_dashboard/sections/4994488/courses/applab-intro/units/1 (consultato il 30/08/2026).
[7] https://code.org/en-US (consultato il 30/08/2026).
[8] Lesson Plan Intro to App Lab, disponibile a: https://studio.code.org/courses/applab-intro/units/1/lessons/1 (consultato il 30/08/2026).

[9] Ministero dell'Istruzione e del Merito (MIM). Regolamento recante indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione, adottato con Decreto del Ministro 9 dicembre 2025, n. 221 (pubblicazione a cura della Direzione Generale per la Comunicazione, marzo 2026).

[10] Programma il Futuro, disponibile a: https://programmailfuturo.it/ (consultato il 30/08/2026)

[11] Programma il Futuro - Lezione 1 (Unit 5: Big & Open Data), disponibile a: https://programmailfuturo.it/come/secondaria-secondo-grado/big-open-data/lezioni/unit-5/693-unit-5-01 (consultato il 30/08/2026).

[12] Programma il Futuro - Lezione 2 (Unit 5: Big & Open Data), disponibile a: https://programmailfuturo.it/come/secondaria-secondo-grado/big-open-data/lezioni/unit-5/694-unit-5-02 (consultato il 30/08/2026)

Prompt Engineering come pensiero algoritmico: un caso studio tra collaborazione e autonomia individuale

Erica Perseghin[1] and Giacomo Salvi[2]

[1] Università degli Studi di Udine, UD, Italia
perseghin.erica@spes.uniud.it
[2] ITI D.M. Turoldo, BG, Italia
giacomo.salvi@istitutoturoldo.edu.it

Abstract

Il contributo illustra un caso studio condotto presso ITI D.M. Turoldo (BG) su un percorso didattico di 12 ore dedicato al prompt engineering come declinazione del pensiero algoritmico e sviluppo di competenze di formalizzazione, verifica e uso consapevole dei sistemi. L'intervento, progettato in coerenza con il D.M. 26/2026 [23], le Linee Guida STEM [19] e le recenti disposizioni ministeriali per l'introduzione dell'AI (Artificial Intelligence) [21], mira a superare la concezione dell'AI come black-box (scatola nera). Il prompt engineering viene trattato come pratica algoritmica: formalizzare una richiesta efficace richiede decomposizione, esplicitazione dei vincoli, sequenzializzazione delle istruzioni e verifica iterativa dell'output, competenze già previste dal curricolo informatico. Sulla base del framework DigComp 3.0 [8], il percorso si è articolato in due fasi. (i) Fase collaborativa: focalizzata sulla co-produzione di artefatti multimodali (testuali, iconografici e infografici) [9, 11] in modalità autogestita, finalizzata a stimolare l'apprendimento situato e la negoziazione paritetica di strategie di prompting. (ii) Fase individuale: caratterizzata da un ciclo iterativo di formulazione e revisione del prompt applicato a domini disciplinari eterogenei (dalla storiografia al machine learning), sotto la guida del docente. L'analisi qualitativa e quantitativa sembra indicare una maggiore strutturazione dei prompt e alcuni primi, seppur non conclusivi, segnali di una crescente consapevolezza nella loro formulazione; permangono tuttavia alcune criticità nella valutazione degli output, che sembrano confermare il ruolo centrale della mediazione didattica. In prospettiva, è prevista una terza fase metacognitiva, orientata a rendere esplicito il processo di decomposizione e formalizzazione del problema per consolidare il legame tra prompt engineering e autonomia cognitiva [3].

1 Introduzione

L'introduzione dell'AI nei contesti scolastici avviene oggi in modo disomogeneo, con il rischio che gli studenti la usino come scorciatoia cognitiva. Il contributo propone un'attività didattica replicabile in cui l'AI diventa oggetto di comprensione consapevole [3, 7, 24], analizzata attraverso la logica del pensiero algoritmico e inquadrata nel framework europeo DigComp 3.0 [8, 10, 13]. Il percorso si colloca nel curricolo di Informatica e Telecomunicazioni, che comprende competenze di pensiero computazionale e progettazione algoritmica [17, 18, 26], ponendosi in continuità con gli apprendimenti informatici promossi nei precedenti livelli di istruzione [1, 16]. In questo quadro, il prompt engineering non è un'aggiunta esterna al curricolo, ma una sua declinazione applicativa in contesto AI [19, 21, 22, 27].

2 Lavoro Proposto

Il caso studio, sviluppato nell'ambito dell'Educazione Civica [20] durante le ore di Informatica, ha coinvolto 20 studenti di una classe terza ITI in 12 ore distribuite su 5 lezioni. Il disegno di ricerca, di carattere esplorativo, ha previsto l'osservazione delle strategie adottate nella fase collaborativa e il confronto tra i prompt iniziali e finali prodotti individualmente nella seconda fase. La progettazione ha

Indicatore (i)	Pertinenza della risposta	Completezza dei contenuti	Coerenza delle argomentazioni	Lessico specifico
$0 \leq i \leq 1$	Non pertinente	Nulli - quasi nulli	Incoerenti e/o contraddittorie	Assente
$1 < i \leq 2$	Parzialmente pertinente	Lacunosi	Parzialmente coerenti	Saltuario
$2 < i \leq 2.5$	Pertinente	Completi	Coerenti	Puntuale
Note: non si considera raggiunta la sufficienza se in uno dei quattro parametri è attribuito il punteggio zero.				

Figure 1: Rubrica valutativa per la produzione di un poster multimodale

integrato multimodalità e interattività — elementi visivi, testuali e pratici — per attivare le cinque aree del DigComp 3.0 [8] attraverso compiti progressivamente più riflessivi [2].

2.1 Fase 1 – Attività di Gruppo: Produzione di un Poster Multimodale

Nella prima fase, gli studenti sono stati suddivisi dal docente in gruppi eterogenei per competenze collaborative, scelta motivata dall'evidenza che gruppi bilanciati producono migliori dinamiche di apprendimento tra pari rispetto a gruppi autoselezionati [6]. I gruppi hanno realizzato un poster digitale sul concetto di AI in modalità autogestita, utilizzando liberamente strumenti di GenAI, tra cui Canva e GeminiAI, insieme a dispense e appunti. La fase è stata intenzionalmente progettata con una guida operativa minima, al fine di osservare le strategie spontanee adottate nell'uso dell'AI e comprendere se il sistema venisse utilizzato come semplice generatore di contenuti oppure come supporto alla selezione [2], sintesi e rielaborazione critica delle informazioni. Gli studenti sono stati chiamati a formulare domande efficaci e costruire prompt adeguati, monitorando i risultati e riorientando il processo in presenza di esiti inattesi. L'AI è stata così utilizzata come ambiente interattivo per sperimentare e rivedere criticamente le proprie strategie di prompting [25]. Gli elaborati sono stati valutati tramite una rubrica condivisa (Figura 1). L'analisi collegiale [14] ha fatto emergere alcune iniziative lodevoli, come la produzione spontanea di poster in inglese, ma anche criticità significative. Nonostante gli argomenti fossero noti, molti elaborati sono risultati superficiali, con una tendenza a usare l'AI per ottenere un prodotto finito anziché come supporto al processo cognitivo [5]. Questa evidenza ha orientato la riprogettazione della fase successiva.

2.2 Fase 2 – Attività Individuale: Prompt Engineering e Analisi Critica

A partire dalle criticità emerse nella Fase 2.1, è stata progettata una seconda attività individuale per esplicitare il legame tra prompt engineering e pensiero algoritmico. Il prompt è stato trattato non come una semplice richiesta linguistica, ma come formalizzazione del problema in termini di obiettivo, vincoli, formato dell'output e revisione iterativa. Una breve introduzione teorica del docente ha illustrato il funzionamento dei modelli generativi e del prompt engineering, chiarendo che il modello produce risposte sulla base delle istruzioni e del contesto ricevuti, e che la struttura dell'input incide pertanto sulla qualità dell'output. Su questa base, agli studenti è stato fornito il metodo PARTS (Figura 2), proposto come strumento operativo per esplicitare in modo sistematico i diversi elementi costitutivi del prompt. La



Metodo PARTS per il Prompt Engineering

Struttura per la costruzione di prompt efficaci con modelli linguistici generativi

Componente	Descrizione e funzione nel prompt definizione · esempio · effetto sull'output
P Persona	Definisci il ruolo del modello Stabilisce prospettiva, lessico e registro della risposta. Senza persona, il modello risponde con tono generico e indifferenziato. `"Sei un docente di informatica esperto in didattica inclusiva..."` → orienta il livello di dettaglio, la terminologia e la struttura argomentativa
A Azione	Specifica il compito con un verbo preciso Determina la tipologia dell'output atteso. Un verbo ambiguo produce risposte fuori bersaglio: "parla di X" ≠ "analizza X" ≠ "progetta X". `Scrivi · Analizza · Progetta · Confronta · Riassumi · Valuta` → il verbo è il vincolo principale: delimita il campo d'azione del modello
R Risultato	Descrivi l'output desiderato in modo misurabile Stabilisce criteri di qualità verificabili. Consente di confrontare l'output con l'atteso e di iterare il prompt in modo fondato. `"...una rubrica con 4 criteri e 3 livelli di padronanza"` → rende esplicito il contratto tra utente e modello sull'output atteso
T Tono	Indica il registro linguistico e lo stile comunicativo Calibra il grado di formalità, l'uso del gergo tecnico e la complessità sintattica in funzione del destinatario. `"Usa un linguaggio formale, accessibile a studenti di terza superiore"` → agisce sulla distanza comunicativa e sulla densità informativa del testo
S Struttura	Definisci il formato e l'organizzazione dell'output Facilita l'integrazione dell'output in documenti o attività didattiche. Riduce l'entropia dell'output e aumenta la riproducibilità del prompt. `"Organizza in: introduzione, 3 sezioni con titolo, conclusione"` → corrisponde alla decomposizione del problema tipica del pensiero algoritmico

Nota: il metodo PARTS è iterativo. Ogni componente può essere raffinato osservando l'output del modello e riformulando il vincolo che ha prodotto il risultato inatteso.

Nel contesto didattico (ITI, indirizzo Informatica), la struttura PARTS è proposta come analogia operativa della decomposizione algoritmica del problema (D.Lgs. 144/2022, profilo C4).

Figure 2: L'infografica è stata ripresa dai materiali disponibili agli studenti ed illustra il processo iterativo strutturato tramite il metodo PARTS [15]

fase operativa ha previsto quattro esercizi eterogenei, scelti per sollecitare differenti modalità di rappresentazione e ragionamento ((i) una sintesi storica con creazione di una timeline [4, 11], (ii) un compito creativo di progettazione visiva, (iii) un problema matematico con spiegazione del procedimento e (iv) la costruzione di uno schema concettuale sul machine learning), ciascuno affrontato con un ciclo ricorsivo di formulazione, osservazione, revisione e valutazione finale, traducendo in ambiente GenAI le operazioni tipiche del pensiero computazionale (Figura 3).

2.3 Analisi dei Risultati

L'analisi ha assunto come unità di osservazione i prompt iniziali e finali prodotti dagli studenti nei quattro esercizi della Fase 2.2. I dati sono stati analizzati attraverso indicatori quantitativi e qualitativi, questi

Esercizio	Compito proposto	Processo algoritmico sollecitato	Macro-competenza di riferimento[1]	Evidenze osservate nei risultati
1. Sintesi storica con timeline	Costruire un riassunto delle prime quattro crociate per una verifica, includendo cause, eventi principali e linea del tempo	Decomposizione del compito; selezione, organizzazione e sequenzializzazione delle informazioni	**Modellazione**	Maggiore esplicitazione di destinatario, formato e vincoli; incremento più marcato nella strutturazione dei prompt
2. Copertina dell'Odissea	Progettare mediante un unico prompt una copertina rispettando vincoli stilistici, cromatici, compositivi e testuali	Identificazione e formalizzazione dei requisiti; gestione simultanea di vincoli e relazioni tra gli elementi	**Modellazione / Formalizzazione**	Maggiore specificazione di stile, formato e vincoli, ma limitata revisione critica dell'output visivo
3. Problema matematico	Risolvere un problema matematico valutandone il procedimento e il risultato	Decomposizione procedurale; controllo dei passaggi; individuazione di errori; confronto tra procedimento e risultato	**Verifica e correttezza / Valutazione**	Richieste di spiegazione *step-by-step* e riformulazione del problema; permane difficoltà nell'individuare errori logici nell'output, con sostanziale fiducia nel risultato prodotto
4. Schema sul Machine Learning	Costruire uno schema di studio a partire da una fonte, comprensibile per uno studente delle superiori	Astrazione; selezione e organizzazione gerarchica delle informazioni; controllo della rispondenza dell'output ai requisiti	**Modellazione / Verifica e correttezza / Valutazione**	Maggiore specificazione di destinatario e formato e introduzione di richieste di verifica; maggiore differenziazione individuale

[1] basata sulla Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella Scuola - CINI

Figure 3: Processi algoritmici sollecitati nei quattro esercizi, macro-competenze rielaborate [1] ed evidenze osservate nei prompt degli studenti [12].

ultimi codificati indipendentemente da entrambi gli autori [12]. Sul piano quantitativo, sono stati considerati la lunghezza media dei prompt, il numero delle parole nei prompt iniziali e finali, la percentuale media di incremento delle parole e la variabilità tra i diversi elaborati, osservata attraverso la deviazione standard. I dati mostrano incrementi differenziati a seconda della tipologia di compito. L'esercizio 1 ha registrato la crescita relativa più elevata, con prompt passati da 22,7 a 32,3 parole (+42,3%) con un ampliamento che ha coinvolto il 90% degli studenti. L'esercizio 4 ha mostrato il maggiore incremento assoluto, con la deviazione standard passata da 5,7 a 16,0, segnale di una crescente differenziazione individuale. Gli esercizi 2 e 3 hanno registrato variazioni più contenute (rispettivamente +1,9% e +13%) poiché i compiti già strutturati lasciavano minor spazio di riformulazione. L'interpretazione di tali risultati richiede alcune precisazioni. La lunghezza dei prompt non costituisce di per sé un indicatore diretto della competenza algoritmica, bensì un dato descrittivo, da affiancare all'analisi qualitativa complessiva della loro struttura (esplicitazione di obiettivi, vincoli e formato dell'output...). È questa articolazione, più che il numero di parole, a poter suggerire, con cautela, una migliore formalizzazione. In questa prospettiva, l'incremento quantitativo osservato può essere interpretato non come semplice espansione verbale, ma, alla luce delle caratteristiche qualitative dei prompt, come possibile segnale di una più accurata decomposizione del problema in sotto-requisiti e di una migliore formalizzazione dell'obiettivo,

suggerendo un primo avvicinamento alla logica della progettazione algoritmica. Sul piano qualitativo, infatti, i prompt finali mostrano una maggiore specificità, verificabilità e aderenza al compito.
La valutazione qualitativa dei prompt post ha considerato tre dimensioni: la presenza di un destinatario esplicito, la specificazione del formato dell'output e l'inserimento di vincoli grafici o procedurali. La dimensione iterativa del lavoro si è rivelata particolarmente significativa, come attestano le riflessioni metacognitive degli studenti: "ho utilizzato il metodo PARTS (Figura 2) perciò la risposta è molto più precisa" o "mi ha spiegato passo per passo i passaggi e i calcoli permettendomi di capire il problema". Tali commenti attestano non solo una valutazione del risultato, ma anche una comprensione del legame tra la struttura dell'input e la qualità dell'output, elemento fondativo del pensiero algoritmico. L'analisi semantica conferma ulteriormente questa tendenza: nei prompt finali compaiono con maggiore frequenza formule come "spiegami passo passo", "in modo sintetico", "con esempi", "comprensibile per uno studente delle superiori". In alcuni casi gli studenti hanno introdotto vincoli negativi ("non farmi un riassunto, fammelo come schema"; "non mettere solo immagini, ma metti anche tabelle"), operazione che richiede una rappresentazione mentale esplicita dell'output non desiderato — una forma elementare di specifica per negazione analoga a quella utilizzata nella definizione di requisiti software. Gli studenti sembrano aver sviluppato modalità di interazione più riflessive, mettendo in gioco competenze di decomposizione, astrazione e debugging. L'aumento della variabilità individuale mostra, tuttavia, che il prompt engineering non si apprende per esposizione spontanea: richiede una mediazione didattica che trasformi la GenAI in uno spazio autentico di esercizio del pensiero informatico.

3 Lavori Futuri

Sebbene lo studio presenti alcuni limiti che ne circoscrivono la generalizzabilità (e.g. il campione ristretto, l'unicità del contesto scolastico, l'assenza di un gruppo di controllo), le analisi [12] hanno messo in luce un ulteriore aspetto critico. In risposta alle domande "Valuta l'output del modello: è corretto?", gli studenti hanno valutato come corretti output che presentavano errori logici evidenti (esercizio 3) e quasi tutti hanno accettato passivamente le immagini generate (esercizio 2) senza apportare miglioramenti. È pertanto prevista una terza fase, attualmente in progettazione, centrata sulla metacognizione individuale. Gli studenti produrranno una sequenza di prompt – tentativo spontaneo, revisione, riformulazione consapevole - e completeranno una griglia di autoriflessione strutturata su 5 dimensioni (Figura 4).

4 Conclusioni

Il caso studio mostra come il prompt engineering possa integrarsi nel curricolo di Informatica come applicazione del pensiero algoritmico in contesti di GenAI: la costruzione di prompt efficaci attiva competenze di decomposizione, astrazione, verifica e debugging, rendendo osservabili processi cognitivi spesso impliciti. I risultati suggeriscono una tendenza verso prompt più specifici e formalizzati, mentre lo sviluppo di un atteggiamento critico nei confronti degli output sembra richiedere un supporto e una mediazione didattica dedicati. In questa prospettiva, l'AI non rappresenta una scorciatoia, bensì un ambiente per esercitare competenze proprie dell'Informatica. Il valore del modello proposto, dimostrato potenzialmente flessibile e adattabile a diversi contesti scolastici, risiede nell'utilizzare il prompt engineering come pratica algoritmica, restituendo all'Informatica il ruolo di disciplina che aiuta a comprendere criticamente il funzionamento dei sistemi, non solo a utilizzarli. In tale prospettiva, la struttura didattica qui delineata potrebbe configurarsi come un'architettura di riferimento per la progettazione di un'Unità di Apprendimento per la disciplina Informatica e Reti di comunicazione come previsto dalla nuova riforma [23].

Indicatore (i) **Livello DigComp**	**Analisi del problema (Decomposizione e Vincoli)**	**Progettazione della soluzione (Logica del Prompt e Attività)**	**Applicazione delle conoscenze (Tecnica e Multimodalità)**	**Correttezza e funzionalità (Validazione e Debugging)**	**Qualità e autonomia (Metacognizione e Processo)**
$0 \leq i \leq 0.5$ Base (1-2)	Non comprende la traccia o ignora completamente i vincoli assegnati (es. target, formati richiesti).	Non propone alcuna strategia; formula prompt "naïf", istruzioni singole e destrutturate senza logica sequenziale.	Non utilizza le tecniche richieste o usa i tool in modo passivo e monocorde, ignorando la natura multimodale del compito.	L'output dell'AI è errato, distorto o assente. Accetta acriticamente il risultato (effetto oracolo).	Il codice/output è disorganizzato. Non dimostra alcuna riflessione sul perché il modello abbia risposto in quel modo.
$0.5 < i \leq 1.5$ Intermedio (3-4)	Comprende la traccia in modo parziale, cogliendo e rispettando solo i vincoli più macroscopici ed espliciti.	Sviluppa prompt semplici adattati alla tipologia di attività, ma strutturati in modo poco efficace (es. richieste cumulative o vaghe).	Utilizza le tecniche di prompting in modo essenziale; integra la multimodalità (testo/immagini) ma con evidenti lacune o asimmetrie tra i media.	L'artefatto funziona parzialmente o presenta errori non bloccanti. Rileva solo le incongruenze macroscopiche senza correggerle alla radice.	L'output è comprensibile ma migliorabile. La revisione avviene per tentativi casuali (trial and error) e non per scelta consapevole.
$1.5 < i < 2$ Avanzato (5-6)	Comprende correttamente la traccia, isolando e rispettando in autonomia tutti i vincoli espliciti dell'attività (es. timeline storiografica o vincoli procedurali).	Progetta il prompt in modo logico, sequenziale e mirato alla tipologia di compito, organizzando la richiesta in sotto-requisiti chiari.	Utilizza correttamente le strutture tecniche richieste; governa la multimodalità orchestrando formati diversi (es. Canva + GenAI testuale) in modo coerente.	Il prodotto finale funziona correttamente, con imprecisioni di lieve entità. Valuta l'output e corregge il prompt iniziale aumentandone la specificità.	L'output è ordinato e ben strutturato. Mostra una iniziale maturità metacognitiva nel descrivere l'efficacia delle modifiche apportate.
$i = 2$ Altamente Specializzato (7-8)	Comprende e interpreta la traccia in modo completo, esplicitando vincoli complessi e vincoli negativi (specifiche per negazione) per guidare il sistema.	Propone una struttura di prompt eccellente, formalizzata e ottimizzata per l'attività (es. definizione di ruolo, contesto, regole, metodo PARTS).	Utilizza in modo consapevole e originale strumenti e tecniche avanzate, spingendosi oltre il richiesto per massimizzare la resa della convergenza multimodale.	Il prodotto è corretto e robusto. Intercetta criticamente allucinazioni o errori logici dell'AI, conducendo un debugging algoritmico rigoroso dell'input.	L'output è chiaro, brillante e realizzato in piena autonomia. Dimostra profonda consapevolezza metacognitiva: autoregola il processo analizzando scientificamente il nesso input-output.

Nota: Non si considera raggiunta la sufficienza se in uno dei 5 parametri è attribuito lo zero.
Nel contesto di questo studio, l'attribuzione dello zero nella colonna "Correttezza e funzionalità" (accettazione passiva di errori/allucinazioni dovuti all'effetto oracolo) o nella colonna "Analisi del problema" (totale violazione dei vincoli del compito) comporta il fallimento automatico dell'attività pratica, a prescindere dall'attrattività estetica dell'output generato dalla macchina.

Figure 4: Griglia metacognitiva per attività individuali

References

[1] Consorzio Interuniversitario Nazionale per l'Informatica - Laboratorio Informatica e Scuola. Proposta di indicazioni nazionali per l'insegnamento dell'informatica nella scuola. Technical report, CINI, 2017. Consultato il 20 agosto 2026.

[2] Danyang Zhang e Danling Li e Kai Guo. From passive consumers to critical creators: Enhancing university students' new media literacy through ai-enhanced digital multimodal composing. *Interactive Learning Environments*, pages 1–25, 2026.

[3] Stefano Cuomo e Gabriele Biagini e Maria Ranieri. Artificial Intelligence Literacy, che cos'è e come promuoverla. Dall'analisi della letteratura ad una proposta di Framework. *Media Education*, 13(2):161–172, 2022.

[4] Debopriyo Roy e George F. Fragulis e Adya Surbhi. Prompt engineering and multimodal tasks in ai-supported efl education: A mixed methods study. *Sustainability*, 18(5):2415, 2026.

[5] Jasper Roe e Leon Furze e Mike Perkins. Digital plastic: A metaphorical framework for critical ai literacy in the multiliteracies era. *Pedagogies: An International Journal*, pages 1–15, 2025.

[6] Bo Sichterman e Omid Noroozi e Josien Boetje e Stan van Ginkel e Hassan Khosravi e Johan Versendaal. Supporting peer learning with artificial intelligence: A systematic literature review. *Innovations in Education and Teaching International*, 62(5):1648–1664, 2025.

[7] Ville Heilala e Roberto Araya e Raija Hämäläinen. Beyond text-to-text: An overview of multimodal and generative artificial intelligence for education using topic modeling. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, SAC '25, pages 54–63. ACM, March 2025.

[8] Judith Cosgrove e Romina Cachia. *DigComp 3.0: European Digital Competence Framework - Fifth Edition*. Lussembrugo, 2025.

[9] Tunde Varga-Atkins e Samuel Saunders e Sue Beckingham e Peter Hartley e Nayiri Keshishi e Nataša Lacković e Na Li e Rob Lindsay e Run Wen e Isabelle C. Winder. An educators' guide to multimodal learning and generative ai, 2025.

[10] Riina Vuorikari e Stefano Kluzer e Yves Punie. *DigComp 2.2: The Digital Competence Framework for Citizens – With New Examples of Knowledge, Skills and Attitudes*. EUR 31006 EN. Lussemburgo, 2022. JRC128415.

[11] Yizhu Gao e Xiaoming Zhai e Min Li e Gyeonggeon Lee e Xiaoxiao Liu. A multimodal interactive framework for science assessment in the era of generative artificial intelligence. *Journal of Research in Science Teaching*, 62(9):2014–2028, 2025.

[12] Erica Perseghin e Giacomo Salvi. Dataset di analisi qualitativo-quantitativo sui prompt inziali e finali. https://uniudamce-my.sharepoint.com/:x:/g/personal/123767_spes_uniud_it/IQAm3bqrY7_GR4Th_gOEkqRyATTQBsuKVKT7wLc-QdHU6zI?e=Xrj9jE, 2026. Accesso: 27 Maggio 2026.

[13] Andrea Garavaglia. Digcomp 3 - return of responsibility and the rise of ai. first reflections on the digcomp 3.0 framework. *Research on Education and Media*, 17(2):1–3, 2025.

[14] Giacomo Salvi. Dataset dei poster anonimizzati realizzati a gruppi dalla terza iti d.m. turoldo bg. https://uniudamce-my.sharepoint.com/:b:/g/personal/123767_spes_uniud_it/IQCTSArk6I_aSpjBvelxlSwUATBZt2qVgyrQPQKIq6be6bk?e=mEVdTS, 2026. Accesso: 27 Maggio 2026.

[15] Francesco Marchesani. *Intelligenza Artificiale a Scuola – Guida semplice per docenti*, volume 75 of *I Quaderni della Ricerca*. Loescher Editore, 2024.

[16] Ministero dell'Istruzione, dell'Università e della Ricerca. Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione. Consultato il 20 agosto 2026.

[17] Ministero dell'Istruzione, dell'Università e della Ricerca. Direttiva ministeriale n.4 del 6 gennaio 2012 - linee guida per gli istituti tecnici – indirizzo informatica e telecomunicazioni, articolazione informatica, 2012.

[18] Ministero dell'Istruzione e del Merito. Decreto-legge n. 144/2022 – riforma degli ordinamenti degli istituti tecnici in attuazione della missione 4 – istruzione e ricerca del pnrr, 2023.

[19] Ministero dell'Istruzione e del Merito. Linee guida per le discipline stem, 2023.

[20] Ministero dell'Istruzione e del Merito. Decreto ministeriale 183 del 7 settembre 2024 - linee guida per l'insegnamento dell'educazione civica, 2024.

[21] Ministero dell'Istruzione e del Merito. Linee guida per l'introduzione dell'intelligenza artificiale nelle istituzioni scolastiche. v.1, 2025.

[22] Ministero dell'Istruzione e del Merito. Piano Nazionale di Ripresa e Resilienza. Missione 4: Istruzione e Ricerca. Componente 1 – Potenziamento dell'offerta dei servizi di istruzione: dagli asili nido alle Università. Investimento 2.1: Didattica digitale integrata e formazione alla transizione digitale per il personale scolastico. Decreto del Ministro dell'Istruzione e del Merito 11 novembre 2025, n. 219, 2025.

[23] Ministero dell'Istruzione e del Merito and Ministero dell'Economia e delle Finanze. Decreto ministeriale n. 29 del 19 febbraio 2026 – revisione dell'assetto ordinamentale degli istituti tecnici e definizione degli indirizzi, delle articolazioni, dei corrispondenti quadri orari e dei risultati di apprendimento, 2026. Allegato C8; attuazione degli articoli 26 e 26-bis del decreto-legge 23 settembre 2022, n. 144.

[24] Patrizia Natale. Etica, intelligenza collaborativa e formazione all'ai nella scuola contemporanea. *Culture Digitali*, 16, 2025.

[25] Sara Pellegrini. Indagine sull'uso dell'intelligenza artificiale nei laboratori degli studenti. *Corpo, Società, Educazione*, 2(2):79–96, 2025.

[26] Repubblica Italiana. Dpr n. 88 del 15 marzo 20210 – regolamento recante norme per il riordino degli istituti tecnici a norma dell'articolo 64, comma 4, del decreto-legge 25 giugno 2008, n. 112, convertito, con modificazioni, dalla legge 6 agosto 2008, n.133., 2010. Allegato C4.

[27] Yoshija Walter. Embracing the future of artificial intelligence in the classroom: The relevance of ai literacy, prompt engineering, and critical thinking in modern education. *International Journal of Educational Technology in Higher Education*, 21(1):15, 2024.

Vertical CanSatAIr: Monitoraggio verticale dell'inquinamento atmosferico e definizione di un indice composito di rischio respiratorio nella bassa troposfera

Giuseppe Fiamingo[1], Antonio Siciliano[2]
[1] Istituto d'Istruzione Superiore di Tropea
giuseppe.fiamingo@scuola.istruzione.it
[2] Istituto d'Istruzione Superiore ITG-ITI-ITE di Vibo Valentia
antonio.siciliano4@scuola.istruzione.it

Abstract

Nel contesto dell'educazione contemporanea, caratterizzata da una crescente integrazione delle tecnologie digitali, lo sviluppo del pensiero computazionale e delle competenze informatiche rappresenta un elemento strategico per l'analisi e la modellazione di sistemi complessi. In tale prospettiva, il presente contributo descrive il progetto ***Vertical CanSatAIr***, sviluppato nell'ambito della missione CanSat, promossa dall'Agenzia Spaziale Italiana ed Europea, che si è conclusa con la verifica sperimentale presso il planetario infini.to di Torino ed ha coinvolto studenti del secondo e terzo anno del Liceo Scientifico "P. Galluppi" di Tropea (a.s. 2025/2026). L'attività ha riguardato la progettazione e realizzazione di un mini-satellite finalizzato allo studio delle particelle di particolato fine PM2.5 e PM10 e di gas VOC (Composti Organici Volatili), costruendo, attraverso un indice ICRR (Indice di Composito di Rischio Respiratorio), un profilo verticale dell'inquinamento atmosferico nella bassa troposfera, fino a una quota di circa 120 m. La verifica sperimentale si è conclusa presso il Planetario infini.to di Torino, attraverso un volo con drone.

L'esperienza proposta si configura come un percorso di apprendimento dell'informatica intesa come scienza dell'elaborazione automatica dell'informazione, in linea con le indicazioni nazionali del CINI, evidenziando il ruolo centrale delle tecnologie digitali nello studio e nella tutela dell'ambiente e della salute.

1 Introduzione

Il monitoraggio della qualità dell'aria nella bassa troposfera rappresenta oggi una delle sfide più urgenti per la tutela della salute pubblica e la sostenibilità ambientale globale. Come evidenziato dalle linee guida dell'Organizzazione Mondiale della Sanità (OMS), è fondamentale identificare i vettori contaminanti, misurarne accuratamente le concentrazioni e studiarne le dinamiche di diffusione tridimensionale nella bassa troposfera.

Per rispondere a questa esigenza, sotto la supervisione del docente coordinatore, gli studenti del Liceo Scientifico "F.lli Vianeo" di Tropea hanno condotto un'attività di ricerca mediante un complesso apparato sperimentale per l'acquisizione e l'elaborazione di parametri ambientali.

Il progetto consiste nella realizzazione di una stazione di monitoraggio ambientale wireless in grado di acquisire parametri meteorologici e dati relativi alla qualità dell'aria, trasmetterli tramite tecnologia LoRa, che utilizza onde radio a lungo raggio, e renderli disponibili attraverso una pagina web per la consultazione e il download. I dati, una volta acquisiti vengono elaborati e normalizzati da un raspberry Pi, generando un indice ICRR e i relativi grafici.

2 Architettura del sistema

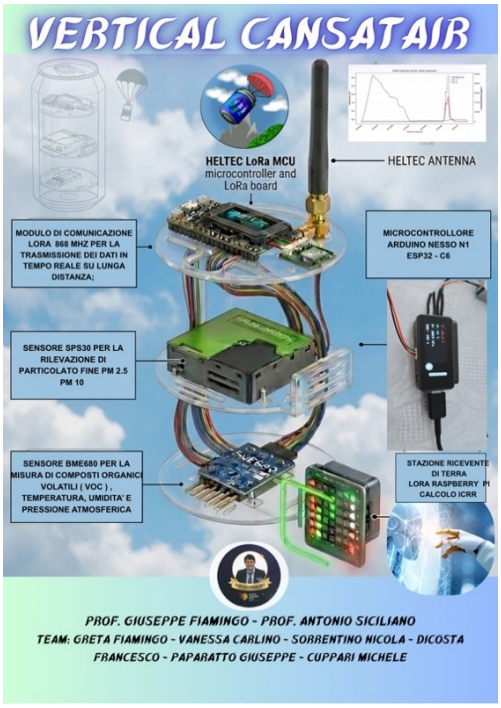


Figura 1: Architettura del Vertical CanSat Air

L'architettura hardware e il flusso dei dati si articolano secondo i seguenti moduli:

- **Unità di Acquisizione Mobile (Payload):** s **Nesso N1** (equipaggiata con microcontrollore Arduino ESP-C6), interfacciato con una suite di sensori dedicati al rilevamento inquinanti.
- **Sottosistema di Trasmissione Radio:** I dati ambientali acquisiti vengono trasmessi in tempo reale mediante un modulo ricetrasmettitore basato su tecnologia **LoRa** (Long Range) a 868 MHz, memoria SPIFFS e server Web integrato.
- **Stazione di Terra (Ground Station):** I pacchetti telemetrici vengono intercettati a terra da un'antenna ricevente LoRa connessa direttamente a un'unità di calcolo **Raspberry Pi**.
- **Alimentazione del sistema:** La scheda Nesso N1 è stata alimentata tramite batteria Li-Ion integrata.

Il nodo trasmettitore acquisisce i dati dai sensori BME680 e SPS30, li visualizza sul display TFT integrato e li trasmette ad una stazione di terra tramite protocollo LoRa. Il nodo ricevitore, acquisisce i pacchetti radio, li memorizza in formato CSV e li rende disponibili all'utente tramite una pagina web accessibile via Wi-Fi all'indirizzo IP locale: 192.168.3.1. L'antenna **LoRa** ricevente**,** connessa a un'unità di calcolo **Raspberry Pi,** esegue l'elaborazione dei dati in tempo reale degli algoritmi specifici. I dati grezzi sui singoli inquinanti vengono normalizzati e aggregati per generare l'**ICRR** (Indice Composito di Rischio Respiratorio). Il feedback visivo dello stato di rischio viene gestito dinamicamente tramite il modulo **SenseHAT** integrato nel Raspberry Pi: la sua matrice di LED RGB viene programmata per operare come un **"semaforo"** logico, variando la colorazione (es. Verde, Giallo, Rosso) in base alle soglie dettate dall'indice ICRR.

Librerie utilizzate

Arduino_Nesso_N1.h: gestione hardware Nesso N1
M5GFX.h: gestione display TFT
Wire.h: protocollo I²C
SPI.h: protocollo SPI
RadioLib.h: gestione SX1262 e comunicazione LoRa
Adafruit_BME680.h: gestione sensore BME680
SensirionI2cSps30.h: gestione sensore SPS30
WiFi.h: gestione Wi-Fi
WebServer.h: server HTTP
SPIFFS.h: file system Flash

3 Le misure a scuola

Organizzazione del team e metodologia di lavoro

L'attività ha seguito un approccio di didattica laboratoriale collaborativa, nel quale gli studenti hanno lavorato in modo sinergico condividendo competenze, idee e responsabilità con ruoli differenti, sotto la guida dei docenti **Giuseppe Fiamingo** e **Antonio Siciliano,** entrambi co-autori di questo lavoro.

Gli obiettivi di apprendimento hanno riguardato in particolare:

- Comprendere il ruolo dei dati nella descrizione e nell'interpretazione di fenomeni reali
- Acquisire competenze di programmazione mediante Arduino e Python
- Progettare e produrre per l'acquisizione automatica, memorizzazione, organizzazione e analisi dei dati.
- Sviluppare capacità di decomposizione di un problema complesso, debugging e integrazione hardware e software.
- Utilizzare strumenti di comunicazione dei dati e visualizzazione dei risultati
- Formulare ipotesi, progettare semplici procedure sperimentali e confrontare i risultati con le ipotesi iniziali.
- Interpretare criticamente i dati e riconoscere le relazioni tra fenomeni ambientali e possibili condizioni di rischio.

La realizzazione del sistema ha quindi costituito il contesto attraverso cui sviluppare competenze informatiche e scientifiche. I diversi moduli tecnologici hanno avuto una precisa funzione didattica: i microcontrollori hanno permesso di affrontare la programmazione e l'interazione con i sensori; la comunicazione LoRa ha introdotto il problema della trasmissione dell'informazione; il Raspberry Pi ha consentito di acquisire, memorizzare ed elaborare automaticamente i dati; Python è stato utilizzato per la loro organizzazione, analisi e rappresentazione.

Particolare attenzione è stata dedicata al passaggio dal **dato grezzo all'informazione**. I valori acquisiti dai sensori sono stati raccolti in dataset, organizzati e successivamente elaborati mediante procedure di normalizzazione e analisi. Gli studenti hanno quindi potuto osservare come l'informatica consenta di trasformare una sequenza di misure in informazioni interpretabili.

Prima della costruzione del complesso apparato sperimentale, è stata proposta al Team una semplice missione: progettare e programmare un sistema in grado di acquisire automaticamente la temperatura, l'umidità, la pressione l'altitudine mediante la formula barometrica $h = 44330 \cdot \left(1 - \frac{P}{P_0}\right)^{0.1903}$ con l'utilizzo di un Arduino R4 e un sensore BME280.

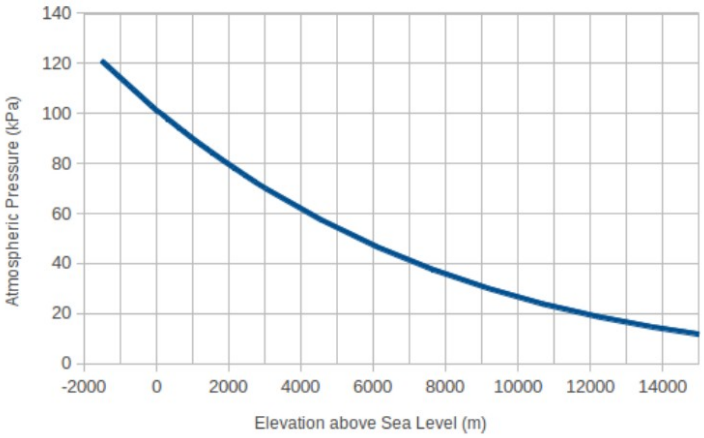


Figura 2: Relazione tra pressione e quota

```
float                ;
float                ;

                = pow(                /101.325,0.1903);
               =(1-                  )*44300+100;

Serial.println();
Serial.print("Altitude in classroom: ");
Serial.print(              );
Serial.print("meters");
delay(1000);
```

Figura 3: Semplice esercizio per il calcolo della quota

L'attività è stata concepita come primo passo verso la comprensione del rapporto tra sensore, dato e informazione. Gli studenti hanno imparato a collegare correttamente i componenti, a configurare l'ambiente di programmazione Arduino IDE, a utilizzare librerie software e a scrivere e modificare semplici programmi per l'acquisizione dei dati. Per la stima dell'altitudine hanno inoltre applicato la relazione barometrica, collegando il modello matematico alla misura sperimentale.

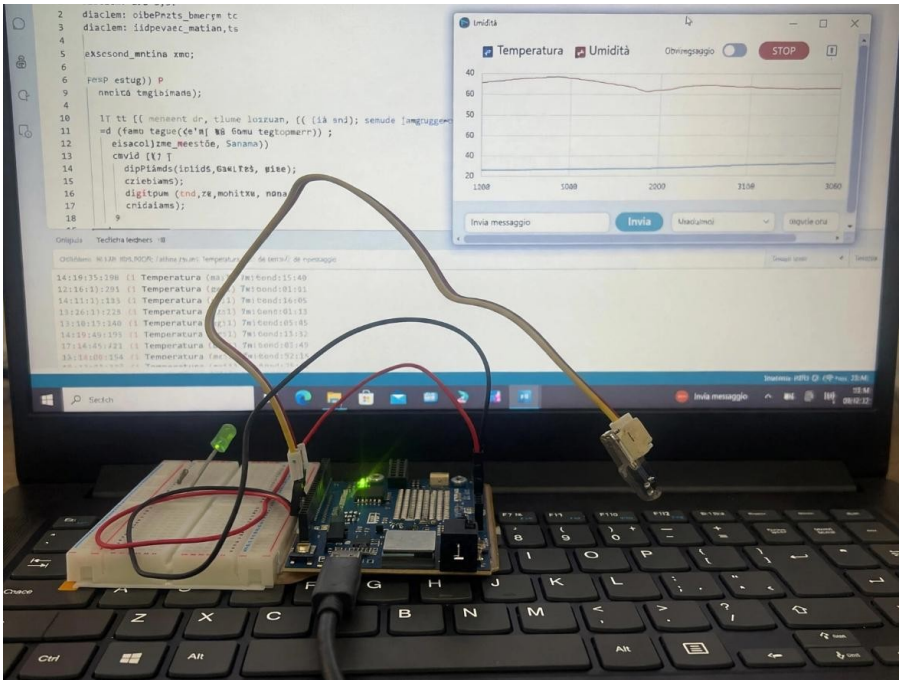


Figura 4: Arduino R4 e il sensore BME680 collegato tramite breadboard e algoritmo python con relativo grafico

Procedimento: Una volta collegato l'arduino R4 al pc, attraverso la piattaforma Arduino IDE, su **Sketch** bisogna scaricare le librerie *Adafruit BME280*. Successivamente abbiamo collegato il cavo del BME280 con i PIN dell'arduino R4:

Pin BME280	Pin Arduino Uno R4	Colore Tipico
VCC	3.3V (Attenzione: non 5V!)	Rosso
GND	GND	Nero
SCL	SCL (o pin A5)	Giallo
SDA	SDA (o pin A4)	Blu

Tabella 1: Piedinatura BME 280

Per leggere i parametri è stato elaborato il seguente algoritmo python*
https://docs.google.com/document/d/1wNt_SWa_I8DgnfXCE__ET2BDqPNOBu4w/edit?usp=sharing&ouid=102601628109530264230&rtpof=true&sd=true

Attraverso questa prima fase, gli studenti hanno compreso che l'informatica non consiste soltanto nella programmazione di un dispositivo, ma permette di **acquisire automaticamente dati dal mondo reale, elaborarli e trasformarli in informazioni utili alla comprensione dei fenomeni osservati**.

Successivamente, gli studenti hanno avviato la fase operativa di assemblaggio del CanSat presso i laboratori del liceo.

Lora

Timestamp	Temp	Umidita	Pressione	VOC	PM1	PM2.5	PM10
585	26.40	51.00	1010.00	79.00	10.00	11.00	11.00
588	26.40	51.00	1010.00	79.00	10.00	11.00	11.00
591	26.40	51.00	1010.00	80.00	10.00	11.00	11.00
595	26.30	51.00	1010.00	81.00	10.00	11.00	11.00
598	26.30	51.00	1010.00	81.00	10.00	11.00	11.00

Tabella 2: Dati acquisiti dagli studenti durante una delle misurazioni in laboratorio.

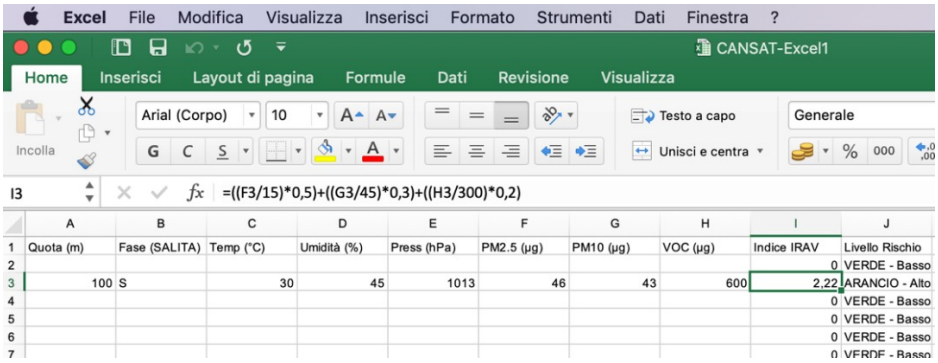


Figura 5: Grafico riprodotto in Excel per il calcolo del ICRR

4 L'indice ICRR e il SENSEHAT

L'originalità del lavoro risiede anche nello sviluppo di un algoritmo per il calcolo di un *Indice Composito di Rischio Respiratorio*' (ICRR), che sintetizza i valori di $PM_{2.5}$, PM_{10} e VOC ponderandoli in base alle linee guida OMS. I risultati dimostrano come l'integrazione tra sensoristica embedded, coding (Python/Arduino) e tecniche di data analysis possa favorire lo sviluppo di competenze interdisciplinari, promuovendo un apprendimento situato e significativo dell'informatica come disciplina applicata alle materie STEM. Il processo si conclude con un'interfaccia immediata e vitale. Tramite una connessione **SSH**, un **Raspberry Pi** fornito di un SenseHAT, con una matrice LED 8x8 funge da semaforo e segnala in tempo reale la qualità dell'aria. Illuminandosi di una luce **verde** per condizioni ottimali, e di una luce **rossa** per allertare su un alto rischio sanitario.

Per ogni inquinante si calcola un Indice Normalizzato (IN) o Quoziente di rischio (QR) rapportandola ai limiti OMS:

$$QR_i = \sum_{i=1}^{n} \frac{C_i}{S_i}$$

Dove:

C_i = combinazione misurata
S_i = Soglia sanitaria di riferimento giornaliera

$$QR_{PM2.5} = \frac{Misura\ PM_{2.5}}{15} \quad ; \quad QR_{PM10} = \frac{Misura\ PM_{10}}{45} \quad ; \quad QR_{VOC} = \frac{Misura\ VOC}{300}$$

$$ICRR = W1 * QR_{PM2,5} + W2 * QR_{PM10} + W3 * QR_{V0C}$$

Per il calcolo dell'ICRR e visualizzazione sul SenseHAT visivo, è stato elaborato il seguente algoritmo**:

https://docs.google.com/document/d/1QGXpna1hpGtVH1aQJnoDXZhbreexwdO2/edit?usp=sharing&ouid=102601628109530264230&rtpof=true&sd=true

INDICE ICRR	RISCHIO	LED MATRIX
ICRR < 1	BASSO	VERDE
1.00 ≤ ICRR < 1.50	MEDIO	GIALLO
ICRR ≥ 1.50	ALTO	ROSSO

Figura 6: Colori visualizzati sulla matrice LED del SenseHAT

INQUINANTE	SOGLIA (L)	PESO (W)	RAGIONE SCIENTIFICA
PM 2.5	**15 µg/m³**	0,5	Massima pericolosità alveolare (OMS)
PM10	**15 µg/m³**	0,3	Pericolosità media per le prime vie aeree
VOC	**300 ppb**	0,2	Inquinamento chimico/gassoso

Tabella 2: Dati quantitativi

5 Le misure al planetario infini.To di Torino:



Figura 7: Torino - Volo in alta quota del drone vincolato al Vertical CanSatAIr

Il giorno 30 maggio 2026, sono stati tracciati i profili atmosferici verticali presso la cornice scientifica di Infini.to, Planetario di Torino. Il Vertical CanSatAIr è stato vincolato ad una fune di circa 4 metri, per poi essere agganciato ad un drone eseguendo un volo di circa 120 m. Una volta atterrato il drone, l'istruttore, ha voluto verificare ulteriormente la sensibilità dei sensori, immettendo dei contaminanti (fumo) e il sistema istantaneamente ha registrato una variazione netta dei parametri sia di particolato che VOC confermando la perfetta funzionalità del sistema.

Parte dei dati ricevuti durante il volo:

Timestamp	Temp	Umidita	Pressione	VOC	PM1	PM2.5	PM10
3411	38.40	32.00	950.60	153.00	12.00	12.00	12.00
3415	38.40	31.00	950.30	154.00	12.00	13.00	13.00
3424	38.50	30.00	948.00	156.00	12.00	12.00	12.00
3428	38.50	30.00	946.80	155.00	12.00	12.00	12.00
3431	38.60	30.00	945.60	156.00	12.00	12.00	12.00
3434	38.60	30.00	944.50	156.00	12.00	12.00	12.00
3438	38.60	30.00	943.30	157.00	12.00	12.00	12.00
3441	38.60	30.00	942.10	157.00	11.00	12.00	12.00
3444	38.60	30.00	941.00	157.00	11.00	12.00	12.00
3454	38.60	29.00	938.00	157.00	11.00	12.00	12.00

Tabella 3: Dati acquisiti durante il volo

35.70	37.00	950.40	164.00	6.00	6.00	6.00
35.70	37.00	950.40	163.00	6.00	6.00	6.00
35.80	37.00	950.30	162.00	297.00	314.00	314.00
35.80	37.00	950.30	126.00	3817.00	5737.00	7876.00
35.80	37.00	950.40	122.00	8335.00	18930.00	31651.00
35.80	38.00	950.30	132.00	7489.00	17713.00	30028.00
35.90	41.00	950.40	25.00	8217.00	23292.00	41655.00
35.90	46.00	950.30	69.00	7742.00	26235.00	48929.00
35.80	44.00	950.20	111.00	5253.00	15601.00	28233.00
35.80	41.00	950.20	133.00	1825.00	4586.00	7927.00
35.80	39.00	950.10	146.00	832.00	1887.00	3154.00
35.90	38.00	950.10	151.00	896.00	1857.00	3000.00
35.90	41.00	950.10	94.00	1032.00	1777.00	2640.00
35.90	43.00	950.10	128.00	567.00	922.00	1327.00
35.90	41.00	950.10	143.00	281.00	392.00	510.00

Dati alterati da fattori inquinanti (fumo)

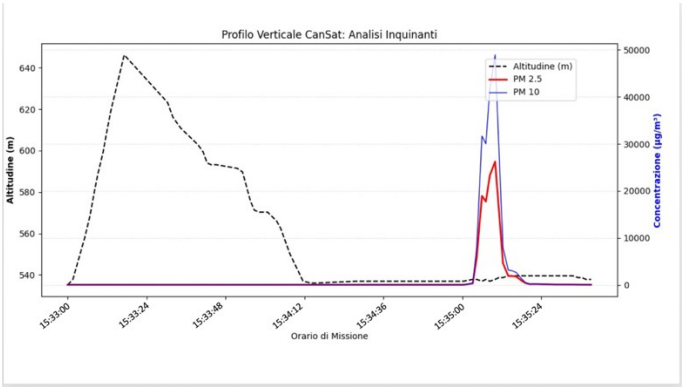


Figura 8: Analisi Inquinanti

6 Ricaduta didattica:

Il progetto ha consentito di integrare il pensiero computazionale, programmazione, data analysis, sensoristica, comunicazione digitale e metodo scientifico. La costruzione del sistema è diventata quindi il mezzo attraverso cui gli studenti hanno sperimentato l'intero ciclo dell'informazione: acquisizione → trasmissione → memorizzazione → elaborazione → interpretazione → comunicazione.

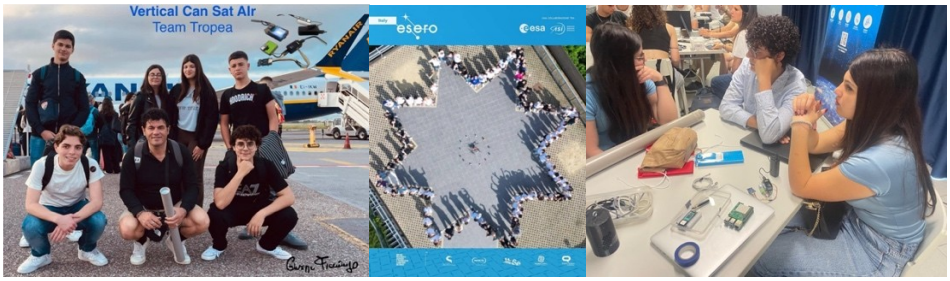


Figura 9: a. Docente e studenti b. Composizione a forma di stella realizzata con tutti i partecipanti c. Momenti di attività sperimentali degli studenti

Debugging Accessibility: ripensare il debugging nella didattica dell'informatica per studenti con DSA

Annunziata Marra

Dipartimento di Informatica – Università di Torino, Torino, Italy
annunziata.marra@unito.it

Sommario

Il debugging costituisce una competenza centrale nell'apprendimento della programmazione, ma coinvolge processi cognitivi complessi che possono rappresentare una barriera per studenti con Disturbi Specifici dell'Apprendimento (DSA). Questo contributo propone una linea di ricerca che indaga modalità di presentazione didattica più accessibili attraverso una rilettura inclusiva del debugging, analizzandone le principali fasi e le difficoltà cognitive associate. A partire da tale quadro teorico viene presentata come primo esempio Algorithm Works, un'esperienza immersiva in realtà virtuale progettata per rendere osservabili e accessibili i processi coinvolti nell'individuazione degli errori. Attraverso rappresentazioni spaziali, l'attività trasforma concetti astratti in elementi concreti con cui lo studente può interagire, favorendo la costruzione di un modello mentale e la formulazione di ipotesi tra il comportamento osservato e quello atteso.

1 Introduzione

Nell'insegnamento dell'informatica, l'inclusione assume un ruolo sempre più rilevante sia a livello internazionale [8, 2], anche con l'introduzione di obiettivi legati a diversity, equity, inclusion e accessibility (DEIA), sia nazionale [11], richiamando una didattica capace di garantire l'accesso ai nuclei fondanti della disciplina senza ridurre la complessità. Tuttavia, concetti come rappresentazione dell'informazione, algoritmi e problem solving richiedono processi di astrazione e formalizzazione che possono risultare particolarmente impegnativi per studenti con Disturbi Specifici dell'Apprendimento (DSA). La sfida riguarda quindi non i contenuti, ma le modalità didattiche con cui vengono proposti, rendendo necessario progettare ambienti di apprendimento che coniughino rigore disciplinare e accessibilità.

2 La complessità del debugging

Le Indicazioni Nazionali 2025 attribuiscono un ruolo esplicito alla riflessione sulla correttezza di algoritmi e programmi, evidenziando la necessità di analizzare, verificare e modificare procedure. Tra gli obiettivi di apprendimento proposti dal CINI [1], in ambito programmazione troviamo:

- O-P3-P-1: rilevare eventuali malfunzionamenti in programmi semplici e intervenire per correggerli;
- O-P5-P-1: esaminare il comportamento di programmi semplici anche al fine di correggerli.

Tali indicazioni riconducono il debugging a una competenza fondamentale dell'apprendimento dell'informatica fin dai primi livelli scolastici. Inteso come processo di individuazione, comprensione e correzione degli errori, il debugging rappresenta una delle pratiche più complesse della programmazione [15]. Tale complessità non dipende solo dall'identificazione dell'errore, ma dal coordinamento di diverse sotto-abilità cognitive, tra cui comprensione del flusso di esecuzione, tracciamento dello stato del programma, interpretazione delle specifiche, formulazione di ipotesi e verifica delle soluzioni [12].

La letteratura evidenzia inoltre una significativa componente emotiva: durante la programmazione gli studenti sperimentano spesso confusione, frustrazione e senso di impotenza [3], con ricadute negative su autoefficacia e motivazione [6]. Per questi motivi, numerosi approcci pedagogici recenti mirano a rendere più espliciti e accessibili i processi di debugging [15] mediante scaffolding cognitivo, visualizzazione dell'esecuzione e attività unplugged. Tali strategie risultano particolarmente rilevanti anche per gli studenti con DSA e suggeriscono possibili direzioni per un debugging più accessibile.

3 Ripensare il debugging in chiave inclusiva

Diversi studi descrivono il debugging come un processo articolato in quattro fasi interdipendenti:

- costruzione dello spazio del problema;
- identificazione dei sintomi dell'errore;
- diagnosi del guasto;
- generazione/verifica delle soluzioni [9].

Sebbene nella pratica siano strettamente intrecciate, la loro analisi separata aiuta a comprendere l'accessibilità del debugging per studenti con DSA, poiché rappresentazioni simboliche, processi impliciti ed elevati livelli di astrazione possono amplificare le difficoltà di apprendimento.

In questa prospettiva, l'accessibilità del debugging non dipende soltanto da strumenti compensativi, ma anche dalle modalità di presentazione del processo. Il framework dello Universal Design for Learning (UDL) [13] propone ambienti che integrino modalità multiple di presentazione, coinvolgimento e azione, riducendo le barriere generate dalle richieste del compito e caratteristiche individuali.

La letteratura mostra inoltre che diverse configurazioni didattiche influenzano l'apprendimento dei concetti informatici: elevati livelli di scaffolding, feedback immediato e progressione guidata favoriscono l'acquisizione di procedure specifiche, mentre attività più aperte sostengono esplorazione e rielaborazione personale delle conoscenze [4].

Alla luce di ciò, ciascuna fase del debugging può essere riletta in chiave inclusiva:

- Costruzione dello spazio del problema. Lo studente confronta comportamento atteso e osservato costruendo un modello mentale del sistema. L'attività richiede gestione di rappresentazioni multiple, astrazione e uso della working memory [14]. Per gli studenti con DSA, informazioni simboliche e sequenziali possono aumentare il carico cognitivo, mentre rappresentazioni visive, multimodali e progressivamente esplorabili possono agevolare la costruzione del modello mentale.
- Identificazione dei sintomi dell'errore. Richiede attenzione selettiva e capacità di riconoscere discrepanze tra comportamento atteso e osservato. In presenza di difficoltà attentive, rappresentazioni dinamiche e feedback visivi possono facilitare il riconoscimento delle anomalie riducendo il carico inferenziale.
- Diagnosi del guasto. Consiste nella formulazione e valutazione di ipotesi sulle cause dell'errore. Gli studenti ricorrono spesso a strategie di trial-and-error [7], invece scaffolding, segmentazione del problema e feedback progressivi possono favorire strategie di modifica più sistematiche.
- Generazione e verifica delle soluzioni. La modifica del programma e la valutazione dei suoi effetti richiedono monitoraggio, confronto tra stati del sistema e coordinamento di elementi. Quando tali relazioni rimangono implicite aumenta il rischio di disorientamento cognitivo; la visualizzazione delle conseguenze delle proprie azioni può invece sostenere verifica e revisione.

Nel complesso, le difficoltà del debugging per gli studenti con DSA dipendono non solo dalla complessità dei concetti informatici, ma anche dalle modalità con cui processi, relazioni e trasformazioni vengono presentati. Progettare ambienti che rendano tali elementi più espliciti appare una direzione promettente per favorire l'accessibilità a studenti con differenti profili cognitivi.

4 Design dell'esperienza immersiva

Viene presentato un caso di studio che applica l'analisi teorica precedente alla progettazione di un'attività di debugging in ambiente immersivo. L'obiettivo è esplorare modalità di rappresentazione didattica capaci di preservare i processi logici coinvolti nell'individuazione, interpretazione e correzione dell'errore. La proposta si colloca inoltre tra gli approcci basati sulla pratica del debugging, ritenuta essenziale per lo sviluppo di competenze efficaci [15].

La progettazione prende avvio dalle principali difficoltà associate al debugging [9]:

- costruzione del modello mentale del programma;
- comprensione del comportamento atteso e osservato;
- gestione di rappresentazioni frammentate;
- ricorso a strategie trial-and-error;
- formulazione e valutazione di ipotesi mantenendo una visione globale del sistema;
- monitoraggio degli effetti delle modifiche.

In questo contesto, la realtà virtuale non è proposta come semplice elemento innovativo, ma come strumento per rendere osservabili relazioni causali, sequenze operative e trasformazioni di stato spesso affidate alla simulazione mentale dando la possibilità di esternalizzare il funzionamento del sistema. Le pratiche tradizionali di debugging, basate su codice testuale, lettura sequenziale e gestione simultanea di molteplici elementi, possono infatti accentuare difficoltà legate a decodifica, working memory e attenzione [5, 14]. Al contrario, rappresentazioni multimodali, distribuzione spaziale delle informazioni e interazione diretta con gli elementi del sistema consentono di modificare le modalità di presentazione senza alterarne la struttura semantica.

5 Algorithm Works: una proposta di debugging immersivo e accessibile

L'esperienza proposta, denominata "Algorithm Works", è progettata in un ambiente immersivo (Figura 1) in cui un codice di esempio (Script 1) viene trasformato in una sequenza di azioni e oggetti osservabili e manipolabili. Lo studente interagisce con una macchina virtuale composta da elementi fisicamente distinguibili (Tabella 1) ciascuno associato simbolicamente a un elemento del codice.

Script 1: Algoritmo di conteggio implementato in Python

```
lista_oggetti = ["cilindro", "sfera", "cubo", "piramide"]

totale = 0

for oggetto in lista_oggetti:
    totale = totale + 1

print("Gli oggetti presenti nella lista sono:", totale)
```

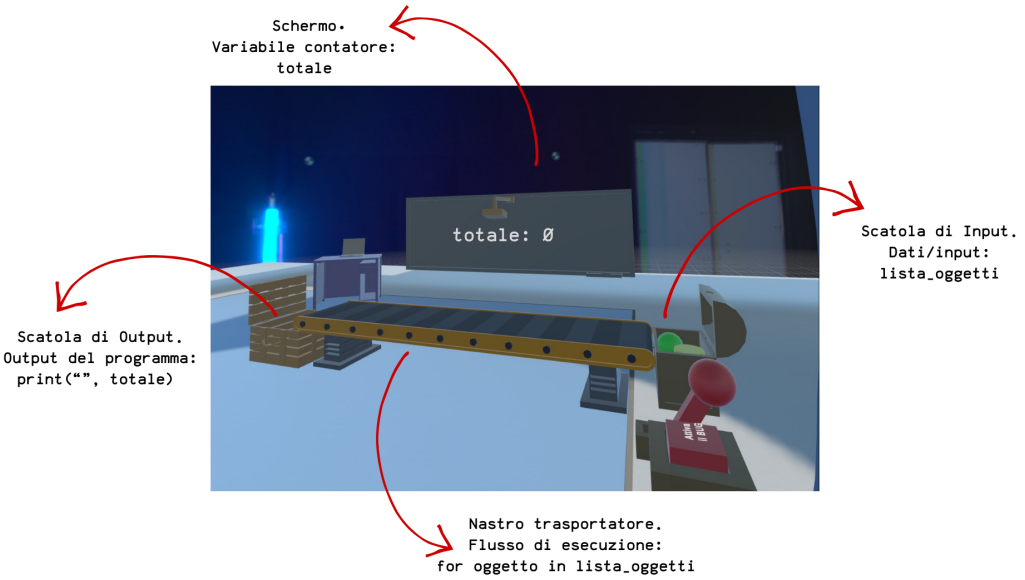


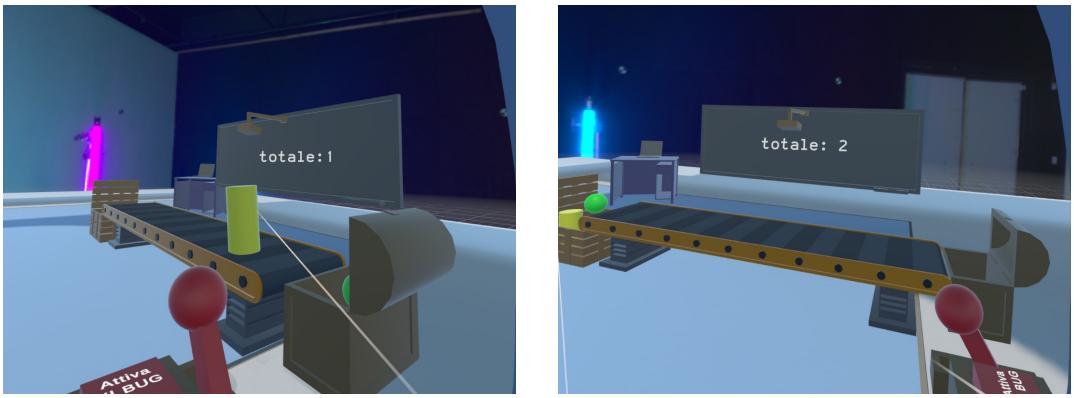
Figura 1: Esperienza VR

Componenti della macchina	Elemento del codice rappresentato
Oggetti trasportati	Dati/input: lista_oggetti
Nastro trasportatore	Flusso di esecuzione: *for* oggetto *in* lista_oggetti
Schermo	Variabile contatore aggiornata durante l'esecuzione: totale
Scatola di output	Output del programma: print(" ", totale)

Tabella 1: Associazione tra elementi fisici e elementi del codice

L'obiettivo dell'esperienza non è l'esecuzione virtuale di un semplice algoritmo di conteggio, nel quale una collezione di oggetti viene attraversata sequenzialmente e un contatore viene incrementato di una unità per ogni elemento visitato, ma la comprensione del comportamento del sistema e l'individuazione visiva di eventuali anomalie grazie all'osservazione dell'esecuzione, rendendo il debugging una conseguenza data dalla discrepanza tra comportamento atteso e comportamento osservato. In una

prima fase la macchina opera correttamente: gli input attraversano il sistema (Figura 2), il contatore si aggiorna progressivamente e il risultato finale compare nella scatola di output.



(a) Input 1 (b) Input 2

Figura 2: Input che attraversano il sistema

Poiché tutte le trasformazioni sono direttamente visibili, lo studente può osservare l'evoluzione dello stato del sistema e la relazione tra azioni, trasformazioni e risultato, favorendo la costruzione del modello mentale del programma. Relazioni causali e cambiamenti di stato vengono infatti resi accessibili attraverso la distribuzione spaziale delle informazioni.

Successivamente lo studente può introdurre un malfunzionamento intenzionale per osservarne gli effetti. All'interno dell'esperienza, una leva permette di alterare la logica del sistema attivando un bug. Nel seguente codice (Script 2) l'errore generato con la leva è legato a un'istruzione duplicata per errore.

Script 2: Versione con bug dell'algoritmo di conteggio implementato in Python

```
lista_oggetti = ["cilindro", "sfera", "cubo", "piramide"]

totale = 0

for oggetto in lista_oggetti:
    totale = totale + 1
    totale = totale + 1    # Bug: incremento duplicato

print("Gli oggetti presenti nella lista sono:", totale)
```

L'effetto visibile è un elemento che attraversa il nastro trasportatore ma che viene conteggiato due volte (Figura 3) e che restituisce un output diverso da ciò che viene osservato.

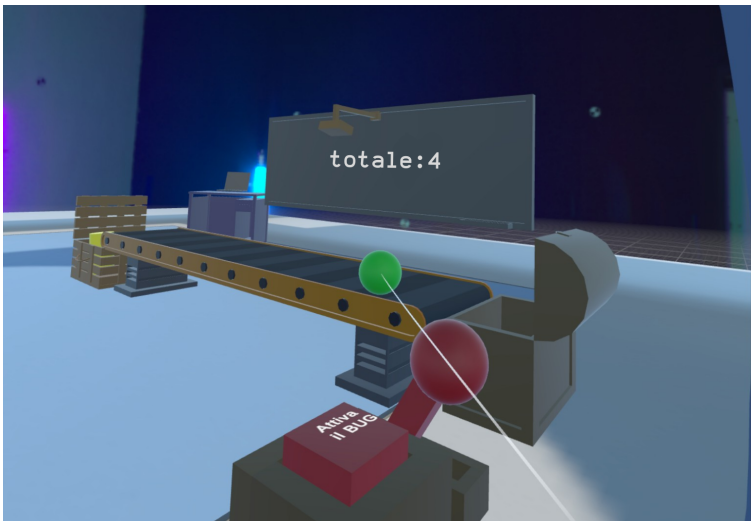


Figura 3: Attivazione del bug tramite la leva – aggiornamento errato del contatore

L'esperienza rende questa anomalia immediatamente osservabile grazie a trasformazioni spaziali e visive, guidando lo studente a confrontare il comportamento osservato con quello atteso e riducendo il livello di astrazione richiesto dall'identificazione dell'errore.

Le anomalie proposte (Tabella 2) non rappresentano errori sintattici, ma categorie di problemi che si potrebbero riscontrare durante il debugging. Ad esempio, un'estensione dell'esperienza virtuale con l'errato instradamento di un oggetto della *lista_oggetti* può spiegare l'effetto di un errore logico nella *condizione* in una struttura del tipo "`if` *condizione* `then` Istruzione1 `else` Istruzione2".

Malfunzionamento osservato	Possibile corrispondenza nel debugging
Conteggio duplicato di un elemento	Aggiornamento errato di una variabile o esecuzione multipla di un'istruzione
Oggetto instradato nel percorso sbagliato	Errore logico di una struttura condizionale
Aggiornamento del contatore non effettuato	Istruzione non eseguita
Oggetto bloccato nel sistema	Ciclo infinito

Tabella 2: Corrispondenza tra malfunzionamento osservato e evento nel debugging

L'interazione non consiste nella modifica diretta del codice, ma nell'osservazione e manipolazione del comportamento del sistema. Lo studente segue il flusso degli oggetti, formula ipotesi sulle cause del malfunzionamento e ne verifica progressivamente gli effetti. Il ripristino del corretto aggiornamento del contatore rappresenta simbolicamente la modifica alla logica di esecuzione del programma. Il feedback immediato consente di verificare le ipotesi formulate e comprendere la relazione tra errore, comportamento del sistema e correzione da apportare al codice.

Queste scelte risultano particolarmente rilevanti per studenti con DSA: la riduzione della componente testuale, la distribuzione spaziale delle informazioni e l'osservazione passo-passo del sistema contribuiscono a ridurre il carico cognitivo, così da essere più coerenti con le esigenze cognitive della popolazione considerata.

6 Conclusioni

In un contesto curricolare sempre più orientato all'inclusione, questo contributo riflette sulle modalità con cui le competenze chiave dell'informatica vengono presentate nei contesti scolastici proponendo una linea di ricerca che indaga modalità di presentazione didattica più accessibili. Il debugging è stato assunto come caso studio poiché rappresenta una competenza fondamentale per l'apprendimento della programmazione e presenta criticità cognitive ed emotive rilevanti per gli studenti con DSA.

Algorithm Works costituisce un primo esempio volto a verificare la fattibilità di un mapping tra l'analisi delle criticità del debugging e la progettazione di un'esperienza accessibile. L'obiettivo non è semplificare il debugging, ma sperimentare modalità di rappresentazione didattica che ne preservino i processi logici. In questa prospettiva, la realtà virtuale viene impiegata per rendere osservabili processi normalmente impliciti e favorire modalità di interazione coerenti con le ricerche sull'embodied cognition [10], che evidenziano il ruolo di corpo, spazio e interazione nella costruzione della conoscenza.

Tra gli sviluppi futuri rientrano l'estensione dell'applicazione immersiva proposta e successive attività di sperimentazione e valutazione.

Riferimenti bibliografici

[1] AIRDM, CINI, CIIM. Matematica e informatica nelle nuove Indicazioni Nazionali. https://www.airdm.org/matematica-e-informatica-nelle-nuove-indicazioni-nazionali-airdm-cini-ciim/, 2026. Consultato il 20 giugno 2026.

[2] Computer Science Teachers Association. Connect, Grow, & Share With CS Teachers. https://csteachers.org/. Consultato il 24 Giugno 2026.

[3] Nigel Bosch e Sidney D'Mello. The Affective Experience of Novice Computer Programmers. *International Journal of Artificial Intelligence in Education*, 27(1):181–206, 2017.

[4] S. Capecchi, L. Forlizzi, I. Fronza, M. Lodi, V. Lonati, G. Melideo, M. Monga, E. Nardelli, V. Rossano, e T. Vardanega. Puzzle-based vs UMC approaches for learning iteration in grade 3. *Computer Science Education*, pagg. 1–44, 2025.

[5] Candace Cortiella e Sheldon H. Horowitz. *The State of Learning Disabilities: Facts, Trends and Emerging Issues*. New York: National Center for Learning Disabilities, 2014.

[6] Jamie Gorson e Eleanor O'Rourke. Why do CS1 Students Think They're Bad at Programming? Investigating Self-efficacy and Self-assessments at Three Universities. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ICER '20, pagg. 170–181, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Amy J. Ko, Thomas D. LaToza, Stephen Hull, Ellen A. Ko, William Kwok, Jane Quichocho, Harshitha Akkaraju, e Rishin Pandit. Teaching Explicit Programming Strategies to Adolescents. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pagg. 469–475, New York, NY, USA, 2019. Association for Computing Machinery.

[8] Amruth Kumar, Rajendra Raj, Sherif Aly, Monica Anderson, Brett Becker, Richard Blumenthal, Eric Eaton, Susan Epstein, Michael Goldweber, Pankaj Jalote, Doug Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, e Qiao Xiang. *Computer Science Curricula 2023*. Association for Computing Machinery, New York, NY. United States, 05 2024.

[9] Chen Li, Emily Chan, Paul Denny, Andrew Luxton-Reilly, e Ewan Tempero. Towards a Framework for Teaching Debugging. In *Proceedings of the Twenty-First Australasian Computing Education Conference*, ACE '19, pagg. 79–86, New York, NY, USA, 2019. Association for Computing Machinery.

[10] Xiuyu Lin, Runbo Li, Zhirong Chen, e Jiayi Xiong. Design strategies for VR science and education games from an embodied cognition perspective: a literature-based meta-analysis. *Frontiers in Psychology*, Volume 14 - 2023, 2024.

[11] Ministero dell'Istruzione e del Merito. Nuove Indicazioni 2025, Scuola dell'infanzia e Primo ciclo di istruzione, Materiali per il dibattito pubblico. https://www.mim.gov.it, 2025. Consultato il 20 giugno 2026.

[12] Gabriele Pozzan, Andreas Bollin, e Tullio Vardanega. A Cognitive-Load-Informed Decomposition of Debugging Subskills With Targeted Exercises. *Informatics in Education*, 25(2):135–165, 2026.

[13] Sue Sentance, Erik Barendsen, Nicol R. Howard, e Carsten Schulte, editors. *Computer Science Education: Perspectives on Teaching and Learning in School*. Bloomsbury Publishing, 2023.

[14] John Sweller. CHAPTER TWO - Cognitive Load Theory. In Jose P. Mestre e Brian H. Ross, editors, *The Psychology of Learning and Motivation. Cognition in Education*, volume 55, pagg. 37–76. Academic Press, 2011.

[15] Stephanie Yang, Miles Baird, Eleanor O'Rourke, Karen Brennan, e Bertrand Schneider. Decoding Debugging Instruction: A Systematic Literature Review of Debugging Interventions. *ACM Trans. Comput. Educ.*, 24(4), November 2024.

Bebras Explorer: valorizzare i quesiti Bebras come risorsa per l'insegnamento dell'informatica

Carlo Bellettini[1], Violetta Lonati[1,2], Mattia Monga[1,2], and Anna Morpurgo[1,2]

[1] Università degli Studi di Milano, Milano, Italia
[2] Laboratorio CINI Informatica e Scuola, Roma, Italia

Sommario

Il Bebras dell'Informatica è una gara rivolta alle classi della scuola primaria e secondaria di secondo grado, basata su piccoli problemi ispirati a concetti fondamentali dell'informatica. Nel corso degli anni, il corpus dei quesiti è cresciuto fino a costituire un ampio archivio di risorse potenzialmente riutilizzabili in ambito didattico, la cui fruizione è però limitata dalla struttura originaria della piattaforma di gara.

In questo lavoro presentiamo Bebras Explorer, uno strumento ideato per favorire il riutilizzo dei quesiti Bebras in classe al di fuori della gara trasformandoli in vere e proprie risorse didattiche per l'insegnamento dell'informatica. In particolare, Bebras Explorer consente di esplorare l'intero archivio Bebras, ricercando i quesiti secondo vari criteri e selezionandoli per costruire collezioni personalizzate. Tali collezioni possono essere rese disponibili come attività didattiche autonome, accessibili tramite una pagina web dedicata che mantiene le funzionalità di interazione con i quesiti ma rimuove i vincoli tipici della gara. Bebras Explorer integra inoltre strumenti per la raccolta e valutazione delle risposte e per la gestione flessibile dell'accesso a soluzioni e spiegazioni.

1 Il Bebras dell'Informatica

Il Bebras dell'Informatica [9], attualmente organizzato in 79 Paesi del mondo, è una gara rivolta a studenti e studentesse della scuola primaria e secondaria. L'obiettivo del Bebras è promuovere l'informatica e il pensiero computazionale in modo accessibile e ludico, attraverso la risoluzione di una serie di quesiti ispirati a reali problemi di natura informatica e che richiedono ragionamento, capacità di analisi e modellazione. L'edizione italiana del Bebras [1], coordinata dal laboratorio ALaDDIn dell'Università degli Studi di Milano, si svolge a novembre e vede annualmente la partecipazione, a squadre di tre, di decine di migliaia di studenti e studentesse [3]. La Figura 1 mostra un quesito nella piattaforma di gara.

I quesiti Bebras sono progettati per mettere in gioco, in forma implicita e contestualizzata, diversi temi e concetti fondamentali dell'informatica, tra cui la rappresentazione dell'informazione e la codifica digitale dei dati, le strutture di dati (liste, tabelle, grafi), gli algoritmi, elementi di programmazione e nozioni introduttive di crittografia, reti di comunicazione, ecc. Tali contenuti non sono esplicitati formalmente, ma emergono attraverso la modellazione della situazione problematica e la ricerca di strategie risolutive efficaci. Una caratteristica distintiva è l'ambientazione narrativa: i quesiti sono introdotti da brevi storie o scenari realistici o fantastici, spesso con castori ('bebras', come sono chiamati in Lituania, il paese in cui è nato il Bebras [7]) o altri elementi familiari come animali, robot, mappe o città. Tale contestualizzazione ha funzione principalmente motivazionale e facilita l'accesso al problema senza ricorrere a prerequisiti o terminologia tecnica esplicita.

Ciascun quesito è concepito come un problema autonomo ed è formato da un titolo, una breve descrizione narrativa, uno o più elementi grafici, il testo della domanda e un'indicazione delle modalità con cui rispondere. Le immagini non hanno soltanto una funzione illustrativa, ma fanno parte integrante del problema e rappresentano spesso la struttura di dati o la situazione da analizzare. Sebbene la forma

[1] http://bebras.it



Figura 1: Un quesito Bebras nella piattaforma di gara.

più frequente sia la risposta a scelta multipla, la piattaforma Bebras usata in Italia supporta diverse tipologie di interazione, tra cui: inserimento di un valore numerico o testuale, ordinamento di elementi, manipolazione o trascinamento (*drag and drop*) di oggetti grafici, riempimento di tabelle, modalità di interazione *ad hoc*. Questa varietà consente di contemplare processi di ragionamento differenti, senza limitarsi al semplice riconoscimento della risposta corretta.

Ogni quesito Bebras è corredato inoltre da un insieme di metadati, definiti durante la progettazione del quesito stesso. In particolare, oltre alla categoria di età a cui il quesito è destinato e al livello di difficoltà previsto, per ciascun quesito sono fornite la risposta corretta, una giustificazione della soluzione e una descrizione dettagliata di come la si può ottenere. Inoltre, la sezione "Anche questa è informatica!" ha la funzione di esplicitare il concetto o il tema informatico a cui il quesito si ispira, rendendolo comprensibile anche a chi non possiede un background specifico nella disciplina e facilitandone così l'utilizzo in contesti didattici non specialistici. Al termine della gara, i quesiti con tutte le informazioni correlate vengono resi pubblici sulla piattaforma e rimangono disponibili per la consultazione anche negli anni successivi.

2 Il Bebras nella pratica didattica

Al di là della gara, i quesiti Bebras possono essere usati come risorsa didattica in classe. Le caratteristiche che li rendono efficaci nella sfida di 45 minuti —problemi brevi, autocontenuti, contestualizzati e focalizzati su concetti fondamentali dell'informatica— li rendono infatti adatti anche all'attività curricolare.

Il riutilizzo dei quesiti Bebras è stato esplorato in contesti internazionali attraverso diverse iniziative. Dagienė e Sentence [8] propongono di integrarli nel curricolo di informatica come strumenti flessibili, utilizzabili per introdurre concetti, stimolare la discussione e come spunto per progettare attività più

articolate. Il progetto PACT della Maynooth University [10] rielabora i quesiti trasformandoli in attività didattiche strutturate, come laboratori *unplugged* [5] e percorsi di apprendimento.

Anche in Italia sono state proposte e documentate esperienze di utilizzo dei quesiti Bebras nella didattica curricolare. Nel sua lavoro di laurea, Calcagni [6] propone di valorizzare i quesiti Bebras come risorsa per la didattica, introducendo una classificazione basata sulle abilità di pensiero computazionale che ciascun quesito contribuisce a sviluppare. L'obiettivo è facilitare la selezione dei quesiti da parte degli insegnanti in funzione degli obiettivi di apprendimento, favorendone il riuso anche al di fuori del contesto della gara. Palazzolo [4, 11, 12] descrive l'utilizzo dei quesiti nelle attività di classe a livello di scuola primaria e secondaria di primo grado, mostrando come essi possano essere impiegati con continuità durante l'anno scolastico.

3 Bebras Explorer

Con il susseguirsi delle edizioni italiane della gara, il corpus dei quesiti Bebras è cresciuto fino a comprendere centinaia di problemi. Questo patrimonio rappresenta una risorsa di grande valore per l'insegnamento dell'informatica. Tuttavia, la sua fruizione al di fuori della gara risulta poco agevole, poiché i quesiti sono organizzati secondo la struttura del regolamento (in particolare edizione e categoria) e sono accessibili solo attraverso la piattaforma utilizzata durante la gara. Di fatto tale piattaforma consente la consultazione dei quesiti solo mediante la simulazione di una sessione di gara, con i vincoli che questa porta con sé: tempo limitato, impossibilità di accedere immediatamente alle soluzioni, ecc. Per valorizzare questo patrimonio abbiamo perciò sviluppato Bebras Explorer, uno strumento che facilita il riuso dei quesiti in ambito didattico. Il Bebras Explorer è disponibile all'indirizzo `https://bebras.it/explorer.html`; la sua interfaccia è mostrata in Figura 2.

Tramite Bebras Explorer, l'insegnante può esplorare l'intero archivio dei quesiti ed effettuare ricerche secondo diversi criteri. Una volta individuati i quesiti di suo interesse, può selezionarli e creare collezioni personalizzate, che a loro volta possono essere rese disponibili attraverso pagine *web* dedicate (Figura 3). Gli studenti e le studentesse potranno visualizzare e interagire con i quesiti selezionati, inserire le risposte e, a discrezione dell'insegnante, accedere alla correzione, alle soluzioni, alle spiegazioni e ai commenti associati. Infine, lo strumento permette all'insegnante di raccogliere le risposte degli studenti, correggerle automaticamente e produrre un resoconto dei risultati.

Sia Bebras Explorer che le pagine *web* di visualizzazione delle collezioni di quesiti operano interamente nel *browser* e possono essere usati anche *offline*, ovvero senza accesso a Internet.

Scenari d'uso. Le collezioni di quesiti predisposte con Bebras Explorer possono essere utilizzate in una pluralità di contesti didattici. Ad esempio, possono essere utilizzate come punto di partenza per introdurre nuovi concetti di informatica, stimolando la curiosità degli studenti e motivandoli all'apprendimento. Esse si prestano anche ad attività guidate e collaborative, in cui gli studenti discutono e confrontano le strategie risolutive, sviluppando capacità di ragionamento e pensiero computazionale. A partire dai singoli quesiti è inoltre possibile proporre anche attività di riambientazione o di generalizzazione del problema, oppure utilizzarli per verifiche strutturate sia con autovalutazione che con raccolta e valutazione automatica delle risposte. La stessa collezione di quesiti può essere impiegata per attività individuali o collaborative, in classe o a casa, per percorsi di approfondimento, esercitazioni, verifiche formative o sommative, eventualmente rendendo inaccessibili le soluzioni o proteggendole tramite *password*.

Interfaccia. L'area principale del Bebras Explorer (Figura 2 – riquadro numerato con 1) presenta il quesito attualmente in esame. Nella barra laterale sinistra (Figura 2 – riquadro numerato con 2) sono

elencati i quesiti dell'archivio. I quesiti possono essere filtrati (si veda il riquadro numerato con 3 nella Figura 2) effettuando una ricerca testuale che consente di interrogare i contenuti del titolo, del testo del quesito, della spiegazione e della sezione "Anche questa è informatica!". Ulteriori filtri permettono di individuare i quesiti tramite parole chiave che identificano concetti o temi informatici, oppure in relazione ai processi di pensiero computazionale che il quesito contribuisce a promuovere [6]. È inoltre possibile filtrare i quesiti in base all'edizione della gara in cui sono stati proposti o in base all'età. L'elenco dei quesiti nella barra laterale sinistra viene aggiornato in tempo reale in base ai criteri di ricerca impostati.

La collezione dei quesiti selezionati dall'insegnante è mostrata in un'area separata nella parte superiore della barra laterale (Figura 2 – parte in alto del riquadro 2). L'ordinamento dei quesiti all'interno della collezione può essere modificato tramite *drag & drop*; questo permette al docente di organizzare la sequenza delle attività secondo i propri obiettivi didattici.

La Figura 3 mostra la pagina web generata dalla selezione di una collezione di 3 quesiti, che può essere così proposta alla classe. I quesiti sono organizzati in schede numerate, navigabili liberamente in qualsiasi ordine. Ciascuna scheda mostra il titolo del quesito al passaggio del *mouse*. I bottoni "Valuta" e "Spiega" in alto a destra consentono (eventualmente, previo l'inserimento di password) di verificare se le risposte inserite sono corrette e di leggere le spiegazioni.

Quando predispone la collezione di quesiti personalizzata, l'insegnante può specificare se abilitare (scegliendo tra accesso libero o protetto da *password*) la correzione automatica, l'accesso alla spiegazione, e funzioni di salvataggio e caricamento delle risposte (si veda la Figura 4).



Figura 2: Bebras Explorer: in questa configurazione sono stati già selezionati alcuni quesiti e non sono stati inseriti filtri di ricerca.

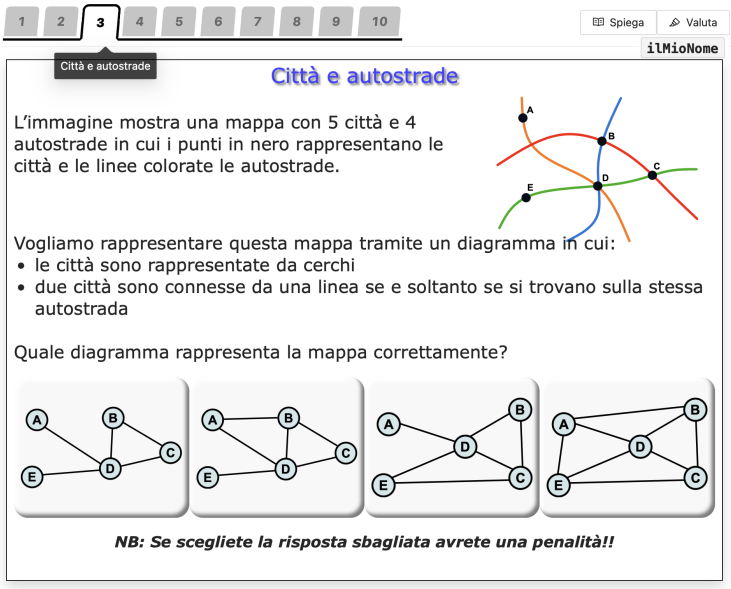


Figura 3: *Screenshot* della pagina *web* con cui viene visualizzata una collezione di quesiti.

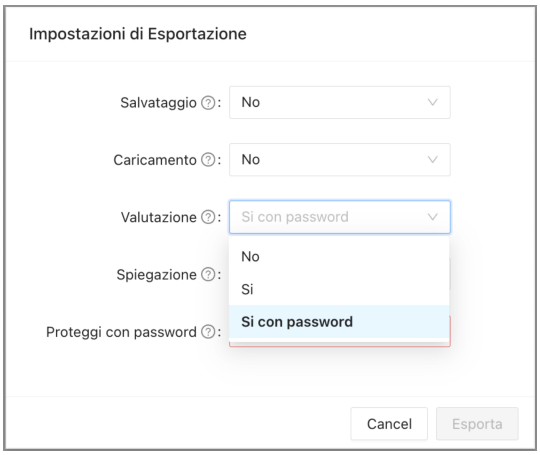


Figura 4: Parametri da definire quando si esporta una collezione di quesiti tramite Bebras Explorer.

4 Conclusioni

Nel complesso, i quesiti Bebras possono essere considerati "micro-problemi" autocontenuti, caratterizzati da una forte integrazione tra testo e rappresentazioni grafiche, ambientati in contesti narrativi accattivanti e progettati per mettere in gioco concetti fondamentali dell'informatica senza richiedere conoscenze tecniche pregresse. Questa combinazione di accessibilità, rigore concettuale e ricchezza di metadati ne favorisce il riuso non solo come strumento di competizione, ma anche come risorsa didattica per l'insegnamento dell'informatica e lo sviluppo del pensiero computazionale.

Bebras Explorer consente ai docenti di consultare, selezionare e usare in classe quesiti dall'intero catalogo italiano del Bebras dell'informatica, estendendo in questo modo il ciclo di vita dei quesiti Bebras, e trasformandoli da elementi legati alla competizione annuale a risorse didattiche flessibili e riutilizzabili in classe.

Riferimenti bibliografici

[1] Bebras — International Challenge on Informatics and Computational Thinking, https://www.bebras.org/

[2] Bebras dell'Informatica — Edizione Italiana, https://bebras.it/

[3] C. Bellettini, F. Carimati, V. Lonati, R. Macoratti, D. Malchiodi, M. Monga, A. Morpurgo. (2018). A Platform for the Italian Bebras. In *Proceedings of the 10th International Conference on Computer Supported Education* - Volume 2: CSEDU; ISBN 978-989-758-291-2; ISSN 2184-5026, SciTePress, pages 350-357. DOI: 10.5220/0006775103500357

[4] C. Bellettini, V. Lonati, M. Monga, A. Morpurgo, M. Palazzolo: Situated Learning with Bebras Tasklets. *Informatics in Schools. New Ideas in School Informatics: Proceedings of the 12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, ISSEP 2019, Larnaca, Cyprus, November 18–20, 2019, Pages 225 - 239, https://doi.org/10.1007/978-3-030-33759-9_18.

[5] Bell, T., Rosamond, F., Casey, N. (2012). Computer Science Unplugged and Related Projects in Math and Computer Science Popularization. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds) *The Multivariate Algorithmic Revolution and Beyond.* Lecture Notes in Computer Science, vol 7370. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-30891-8_18

[6] A. Calcagni, V. Lonati, D. Malchiodi, M. Monga, A. Morpurgo. Promoting computational thinking skills: would you use this Bebras task? In *Informatics in Schools: Focus on Learning Programming*. V. Dagienė, A. Hellas [Eds] - Proceedings International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, Helsinki, Finland, Nov. 2017 - LNCS, Springer, 2017, pp. 102-113 - ISBN 9783319714820.

[7] V. Dagienė. Information technology contests – introduction to computer science in an attractive way. *Informatics in Education*, 5(1), 2006, p. 37–46. ISSN 1648-5831.

[8] V. Dagiene, S. Sentance. It's Computational Thinking! Bebras Tasks in the Curriculum. In *Lecture Notes in Computer Science*, vol. 9973, pp. 28-39, Springer-Verlag Berlin Heidelberg, ISBN 978-3-319-46747.

[9] V. Dagienė and G. Stupuriene. Informatics education based on solving attractive tasks through a contest. IFIP–KEYCIT 2014, p. 51–62, 2014.

[10] T. Lehtimäki, R. Monahan, A. Mooney, K. Casey, T. J. Naughton. Computational Thinking Resources Inspired by Bebras. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2023), July 8-12, 2023, Turku, Finland*. ACM, New York, NY, USA. 1 page. https://doi.org/10.1145/3587103.3594203

[11] M. Palazzolo, R. Signorini, D. Viganò. I giochi Bebras dell'informatica nel curricolo della scuola del primo ciclo. *Atti del convegno ITADINFO Convegno Italiano sulla Didattica dell'Informatica*, Bari 13-15 ottobre 2023, ISBN 978-88-6629-075-9.

[12] M. Palazzolo. Attività didattiche di programmazione a partire dai giochi dell'informatica Bebras. *Atti del convegno ITADINFO Convegno Italiano sulla Didattica dell'Informatica*, Salerno, 3-5 ottobre 2025, ISBN 9791256005123.

Paper Bots AI: introdurre machine learning e programmazione a blocchi nella scuola secondaria

Alessia Galli[1], Lara Rollo[2]
[1] IIS Ettore Majorana
ing.alessiagalli@gmail.com
[2] IC Velletri Sud Ovest
rollo.lara@gmail.com

Abstract

Il contributo presenta “Paper Bots AI”, un percorso didattico che introduce agli studenti della fascia di transizione tra primo e secondo ciclo di istruzione i concetti fondanti del machine learning e la programmazione a blocchi applicata a un artefatto fisico. Le nuove Indicazioni Nazionali per il primo ciclo (DM 221/2025) garantiscono alle prossime coorti un percorso strutturato di alfabetizzazione informatica fin dalla scuola primaria; gli studenti che hanno già attraversato il primo ciclo prima dell'entrata in vigore della riforma, tuttavia, non sono stati raggiunti da questo percorso graduale e rischiano di affrontare la secondaria di secondo grado privi delle basi algoritmiche che i compagni più giovani acquisiranno progressivamente. Il percorso qui presentato si propone esplicitamente come ponte metodologico per questa fascia di studenti non captati dalla riforma, offrendo loro un'acquisizione mirata e accelerata dei concetti di base dell'algoritmica e dell'Intelligenza Artificiale, prima dell'ingresso nel secondo grado.

Accanto a questa funzione di raccordo curricolare, il percorso persegue un obiettivo di alfabetizzazione critica all'intelligenza artificiale (AI literacy): attraverso la costruzione e l'addestramento in prima persona di un modello di classificazione, gli studenti sperimentano concretamente cosa significhi che un sistema “impara” da un dataset, cosa sia un errore statistico (overfitting, bias) rispetto a un errore di codice, e quale sia il ruolo del controllo umano nella progettazione di un sistema automatico — competenze di lettura critica dei sistemi di IA sempre più rilevanti per una generazione che utilizza quotidianamente strumenti di IA generativa senza comprenderne i meccanismi.

Il percorso è stato sperimentato in due contesti scolastici — una classe seconda di istituto tecnico e una classe terza di scuola secondaria di primo grado — prima di essere condensato in un laboratorio di formazione per docenti proposto a Didacta Firenze 2026.

Si presentano il quadro epistemologico che distingue programmazione deterministica e apprendimento automatico, l'architettura tecnologica impiegata (BBC micro:bit, Google Teachable Machine, app Teachable Micro:Bit di Cardboard Robots) con le relative implicazioni etiche, e gli esiti osservati nei due contesti in termini di autonomia, gestione dell'errore e percezione dell'IA.

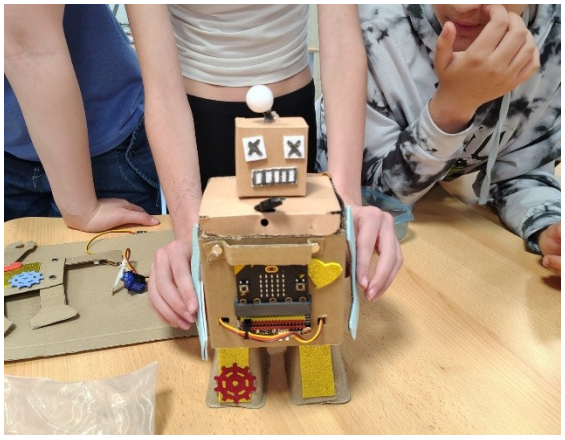


Figura 1: Paper Bot Ai

1 Inquadramento e Razionale

L'introduzione dell'informatica come disciplina obbligatoria nel primo ciclo, sancita dal DM 221/2025, distingue esplicitamente le competenze digitali dalle competenze informatiche, queste ultime riferite alla comprensione di come i dati vengano elaborati automaticamente. “Paper Bots AI” si colloca non nella fase introduttiva di tale percorso, ma in quella conclusiva: si propone come ponte metodologico verso il secondo grado, rivolto in particolare agli studenti che hanno attraversato il primo ciclo prima dell'entrata in vigore della riforma e necessitano quindi di un'acquisizione mirata dei concetti fondanti di Intelligenza Artificiale e Machine Learning, altrimenti assenti dal loro percorso curricolare.

Il percorso esplicita la transizione tra due modalità di risoluzione dei problemi: nella programmazione deterministica il comportamento del micro:bit è definito a priori da chi programma, secondo una relazione fissa tra input e azione; nel machine learning questa relazione non viene scritta ma indotta da un algoritmo che analizza un dataset etichettato, e il sistema restituisce non una certezza ma un grado di confidenza per ciascuna classe possibile. L'interfacciamento tra modello e microcontrollore avviene tramite una soglia minima di confidenza che può essere manipolata fisicamente dai partecipanti durante l'attività, e permette di osservare l’effetto immediato sul comportamento del robot.

2 Architettura tecnologica e considerazioni etiche

Il percorso si appoggia a un'infrastruttura tecnologica già esistente e gratuita. L'hardware è il BBC micro:bit; il software è Google Teachable Machine, che addestra il modello interamente nel browser (TensorFlow.js) senza inviare le immagini della webcam a un server. È tuttavia necessario esportare il modello generando un link pubblico ospitato sui server di Google — un passaggio, distinto dall'addestramento, che comporta una dipendenza dal cloud di cui è opportuno essere consapevoli quando i soggetti coinvolti sono minorenni.



Figure 2-3-4: Didacta 2026

Il collegamento tra modello e robot avviene tramite l'app Teachable Micro:Bit del progetto Cardboard Robots (cardboard.lofirobot.com), via Bluetooth o WebUSB. Tale app, e l'ecosistema di manuali e template che la accompagna, è stata impiegata come infrastruttura senza modifiche: il contributo specifico di questo percorso non è dunque tecnico, ma didattico e di validazione sul campo.

Ne derivano alcune raccomandazioni operative: richiedere un consenso informato specifico alle famiglie anche in presenza di addestramento locale; privilegiare gesti o oggetti anonimizzati rispetto a volti riconoscibili; trattare il link del modello come risorsa non condivisibile oltre l'attività svolta; coinvolgere il Referente per la protezione dei dati dell'istituto; evitare di associare le classi di addestramento a caratteristiche personali sensibili anche a fini dimostrativi sui bias.

3 Metodologia Didattica

Il nucleo esperienziale del percorso è la costruzione fisica del Paper Bot, un robot in cartone e materiali di recupero animato da un servomotore. La costruzione si articola in tre fasi — chassis, integrazione elettronica, cablaggio — durante le quali l'assenza di un libretto di istruzioni costringe a soluzioni creative (alleggerire, spostare il fulcro, rinforzare): il principio del tinkering, per cui le mani sono strumento dell'intelligenza (Montessori, Papert).

Il percorso completo, sperimentato in classe con tempistiche distese (Sezione 4), è stato condensato in un laboratorio di 90 minuti per i docenti a Didacta Firenze 2026, organizzato in otto fasi:

Fase	Attività
1. Sensoristica	Esplorazione hardware e sensori del micro:bit
2. Coding base	Programma deterministico in MakeCode
3. Tinkering	Costruzione dello chassis
4. Attuatori	Integrazione e programmazione del servomotore
5. Addestramento ML	Dataset di gesti con Google Teachable Machine
6. Connessione AI–Robot	App Teachable Micro:Bit, via Bluetooth/WebUSB
7. Debugging	Test di robustezza e correzione del dataset
8. Debriefing	Riflessione su paradigmi e implicazioni etiche

Tabella 1: Struttura dell’attività

4 Testimonianza di due contesti Scolastici

Prima della sua trasposizione per la formazione docenti, il percorso è stato sperimentato con gli studenti in due contesti scolastici differenti per età, pregresso e organizzazione didattica.

Nella classe seconda di un istituto tecnico (indirizzi informatica, meccanica, elettronica; 15 studenti, 4 lezioni da 2 ore), gli studenti possedevano già gli strumenti dell'algoritmica di base — flowchart, pseudocodice, strutture di controllo, rudimenti del linguaggio C. I gruppi sono stati organizzati per ruoli disciplinari ("informatico", "meccanico", "elettronico"), ciascuno responsabile di un ambito dell'artefatto. Inizialmente prevenuti nei confronti dell'architettura in cartone, percepita come poco professionale rispetto ai materiali con cui erano abituati a lavorare,

gli studenti hanno progressivamente rivalutato la scelta: al termine del percorso hanno chiesto spontaneamente di poterne verificare la replicabilità con strumenti più avanzati, come la stampa 3D — segno di un forte effetto motivante, oltre che di un brainstorming che ha generato altre proposte di estensione (ruote motorizzate, sensori di distanza). Di fronte a malfunzionamenti del sistema di riconoscimento, questi studenti hanno intuito autonomamente la relazione tra errori di classificazione e interferenza dello sfondo, correggendo il dataset di conseguenza.

Nella classe terza di un Istituto Comprensivo (25 alunni, 6 lezioni da 2 ore), il pregresso era limitato a Scratch in ambiente puramente digitale: applicare la programmazione a un oggetto fisico ha generato stupore e un forte coinvolgimento, di natura anche empatica — poter configurare autonomamente l'hardware con forbici e colla ha reso l'artefatto percepito come proprio. Data la maggiore numerosità, non è stato possibile replicare i ruoli disciplinari; i gruppi da 5 studenti hanno seguito una progressione analoga ma distribuita su un arco di tempo più ampio. Questi studenti, non avendo ancora maturato il concetto di confidenza statistica, hanno mostrato maggiore difficoltà nel formulare ipotesi di miglioramento del modello, pur esprimendo una creatività più spiccata, con proposte ambiziose come motorizzare anche testa e piedi del robot.

Nel laboratorio rivolto ai docenti a Didacta è stata particolarmente apprezzata la parte guidata dell'attività, sia nella realizzazione del Paper Bot sia nella configurazione della scheda e degli strumenti software impiegati: questa impostazione ha permesso ai partecipanti di concentrarsi sul programma didattico e sull'esplorazione dell'IA, senza disperdere tempo nella risoluzione di problemi tecnici imprevisti.

Un questionario iniziale e finale, somministrato in entrambe le classi sulla percezione dell'IA, ha rilevato un cambiamento significativo: gli studenti hanno dichiarato di aver compreso meglio il riconoscimento delle immagini e di percepire diversamente il controllo umano sul sistema — un dato rilevante per una generazione che usa l'IA generativa senza comprenderne i meccanismi

5 Riflessione sulla Didattica dell'Errore e Ai Litteracy

Nella didattica informatica tradizionale l'errore è debug del codice: il programma viola una regola logica predefinita. Nel machine learning il fallimento nasce spesso da un dataset inadeguato: l'errore si sposta dal piano logico-formale a quello statistico. Il confronto tra i due contesti mostra come questo spostamento richieda un impianto concettuale — la nozione di confidenza, la distinzione tra overfitting e bias — che si consolida progressivamente con l'età e l'esperienza, più che con l'istruzione esplicita. Correggere un modello significa bilanciare le classi di input, variare le condizioni di acquisizione, riaddestrare: è in questo esercizio concreto, e non in una definizione astratta, che si costruisce la AI literacy — la capacità di comprendere che un sistema intelligente non possiede una reale comprensione semantica, ma applica pattern statistici interamente dipendenti dalla qualità dei dati immessi dall'essere umano.

6 Conclusioni

Il duplice attraversamento — dall'aula alla formazione dei formatori — mostra un percorso scalabile su più fasce d'età, fondato su tre scelte progettuali: la valorizzazione della dimensione fisica dell'apprendimento tramite il tinkering; l'uso di strumenti gratuiti e diffusi, accompagnato da una gestione consapevole e responsabile dei dati raccolti dagli studenti; lo spostamento del focus didattico dalla scrittura di codice all'analisi dei dati e alla validazione dei modelli probabilistici. Collocandosi nella fase conclusiva, e non introduttiva, dell'insegnamento dell'informatica del primo ciclo, il percorso si propone come ponte verso la secondaria di secondo grado, colmando il divario formativo degli studenti che non hanno beneficiato della nuova impostazione curricolare fin dall'inizio del proprio percorso scolastico.

Questa esperienza che si basa sui laboratori svolti intende proporsi come una palestra metodologica per i docenti, fornendo loro non soltanto le competenze tecniche necessarie per utilizzare sensori, attuatori e algoritmi di intelligenza artificiale, ma soprattutto una cornice pedagogica strutturata per formare cittadini consapevoli, in grado di agire con spirito critico all'interno di una società permeata dai dati e dai sistemi di decisione automatica.

Bibliografia

[1] Resnick, M., & Rosenbaum, E. (2013). *Designing for Tinkerability*. In M. Honey & D. Kanter (Eds.), Design, Make, Play: Growing the Next Generation of STEM Innovators. Routledge

[2] Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books. → (costruzionismo e “learning by doing” con tecnologie)

[3] Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). *Intelligence Unleashed: An Argument for AI in Education*. Pearson.

[4] Teachable Machine (Google): https://teachablemachine.withgoogle.com

[5] Cardboard Robots — piattaforma di robotica educativa in cartone e app Teachable Micro:Bit: https://cardboard.lofirobot.com/

[6] BBC micro:bit – sito ufficiale: https://microbit.org/

[7] Microsoft MakeCode for micro:bit: https://makecode.microbit.org/

[8] AI + micro:bit (risorse educative): https://microbit.org/projects/make-it-code-it/ai/

[9] GenAI Teachable Machine — piattaforma a sovranità dati locale, non utilizzata https://tm.gen-ai.fi/

Dichiarazione sull’utilizzo dell’Intelligenza Artificiale
Gli strumenti di Intelligenza Artificiale generativa sono stati utilizzati esclusivamente come supporto alla revisione linguistica, alla sintesi e alla riorganizzazione del testo. Le autrici hanno verificato e validato i contenuti, assumendo la piena responsabilità del contributo finale.

Ha ancora senso insegnare l’assembly nel 2026?

Dal TASM al cloud: architettura dei calcolatori e cybersecurity

Pierangelo Indolfi
Politecnico di Bari; ITS Academy Apulia Digital; già docente IISS Marconi-Hack, Bari
pierangelo.indolfi@poliba.it

Abstract

Nel 2026, con i linguaggi ad alto livello e l'intelligenza artificiale che generano codice su richiesta, dedicare settimane del curricolo scolastico alla programmazione assembly sembra un anacronismo. Questo contributo, un case study qualitativo fondato su quasi trent'anni di esperienza didattica diretta, argomenta il contrario: l'assembly mantiene un valore formativo specifico, a condizione che l'insegnamento sia aggiornato negli strumenti e ancorato a una motivazione contestuale solida. Il contributo documenta un percorso didattico trentennale nella disciplina Sistemi e Reti degli istituti tecnici tecnologici, descrivendo l'evoluzione degli ambienti di lavoro da TASM su DOS fino a NASM x86_64 in cloud, e discutendo il legame tra programmazione a basso livello, architettura dei calcolatori e cybersecurity.

1 Una domanda non scontata

Nel 2026, mentre i linguaggi ad alto livello dominano il mercato e i modelli di intelligenza artificiale generano codice su richiesta, dedicare alcune settimane del curricolo scolastico alla programmazione assembly sembra difficile da difendere. Eppure chi ha attraversato decenni di insegnamento dell'informatica sa che certe domande apparentemente ovvie nascondono risposte tutt'altro che scontate.

Questo contributo risponde affermativamente: ha ancora senso insegnare l'assembly, a condizione che l'insegnamento sia aggiornato negli strumenti e ancorato a una motivazione contestuale solida. Non si tratta della nostalgia del "vero programmatore" [13]: nell'esperienza qui descritta, l'assembly sembra produrre qualcosa che nessun altro strumento del curricolo produce nella stessa misura. Il dibattito non è nuovo: già Agarwal e Agarwal [1] si interrogavano se meritasse ancora un corso a sé, mentre Imamura [6] ribadiva che è l'interfaccia reale tra hardware e software, non un semplice strumento didattico. Più di recente, la disponibilità di IA capaci di generare codice assembly ha riproposto la domanda da un'angolatura inedita [17].

2 Chi insegna, a chi e in quale disciplina

Il contributo nasce da 29 anni di insegnamento nel triennio Informatica degli istituti tecnici tecnologici, nella disciplina Sistemi e Reti (IISS Marconi-Hack, Bari, 1996-2025) [8]. Nelle classi terze, il modulo sulla programmazione a basso livello è stato una costante: la convinzione che comprendere il funzionamento reale di un elaboratore, a livello di istruzione macchina, registri e gestione della memoria, sia fondamentale per la formazione di un tecnico informatico consapevole non è mai venuta meno, pur cambiando nel tempo gli strumenti.

A rafforzare questa prospettiva c'è un'esperienza personale diretta: negli anni Ottanta l'autore ha fatto parte del gruppo Olivetti di Bari che sviluppò il sistema operativo BCOS, scrivendo migliaia di righe di codice in assembly 8080 e Z8000 [7]. Quella formazione tecnica è rimasta il punto di riferimento del percorso professionale e didattico: prova concreta che certi fondamenti hanno una durata diversa rispetto agli strumenti che cambiano.

3 La motivazione contestuale: la cybersecurity cambia il quadro

La domanda sul senso dell'assembly trova oggi una risposta contestuale più forte di quanto non avesse vent'anni fa: la cybersecurity. Il reverse engineering, ovvero l'analisi di eseguibili, l'identificazione di vulnerabilità e la comprensione del comportamento di malware, richiede necessariamente la capacità di leggere e interpretare il codice macchina [4]. Non è una competenza opzionale per chi vuole operare in ambito security: è la premessa tecnica senza la quale ogni altro strumento rimane opaco.

Questo ancoraggio risponde alla domanda concreta che gli studenti si pongono: a cosa mi servirà? La comprensione dell'assembly come base della competenza security trasforma un argomento astratto in un investimento formativo riconoscibile.

4 Il percorso evolutivo degli ambienti di lavoro

TASM (Turbo Assembler Borland) su DOS. Il punto di partenza: l'interfaccia a sfondo blu che ha accompagnato generazioni di programmatori. Ambiente chiuso e immediato, x86 a 16 bit. Il limite era la distanza crescente rispetto al contesto tecnologico reale: DOS non era più il sistema operativo degli studenti.

NASM x86, poi x86_64, su Linux locale e cloud. Il passaggio a 32 e 64 bit, all'open source, ai moderni strumenti di debug. Su Linux nativo o VirtualBox si usa SASM [11]; per il cloud si è passati da Replit [14], poi diventato a pagamento, a mycompiler.io [9]. Il finanziamento degli strumenti cloud nelle scuole resta un problema aperto.

Questa scelta di lavorare su x86_64 reale, anziché su ambienti semplificati, non è un default ma una scelta didattica esplicita: gran parte della manualistica scolastica resta ancorata all'8086 dentro emulatori DOS che gli studenti non hanno mai incontrato altrove. Architetture come MARIE [10] o LC-3 [12], pur non utilizzate qui, rappresentano l'alternativa teorica: isolano il ciclo fetch-decode-execute da ogni complessità implementativa, rinunciando al confronto con l'architettura reale.

5 Scelte metodologiche: programmazione strutturata e consapevolezza dell'I/O

Due scelte didattiche trasversali meritano una trattazione esplicita. La prima riguarda il salto incondizionato. L'assembly mette a disposizione il jump con una libertà che non ha equivalenti nei linguaggi ad alto livello, e la tentazione di usarlo in modo anarchico è fortissima. Ma produce spaghetti code, e non c'è ragione per cui Böhm e Jacopini debbano valere meno a livello macchina che di pseudocodice [2]. Per questo il modulo introduce fin dall'inizio pattern standard che esplicitano in assembly le strutture di controllo fondamentali: if-then, if-then-else, while e do-while, con etichette che ne dichiarano il ruolo. È la premessa per scrivere codice leggibile e manutenibile a qualsiasi livello.

La seconda riguarda la consapevolezza dell'I/O. Il sistema operativo offre le primitive di input e output come un servizio, e gli studenti tendono a usarle senza interrogarsi su cosa viaggi sui canali di comunicazione. In assembly non è possibile ignorarlo: i dati arrivano in formato ASCII, e la conversione in binario è a carico del programmatore, con ricadute utili dalla comprensione delle vulnerabilità di input alla progettazione di protocolli.

6 La struttura didattica invariante

Attraverso tutti gli ambienti, la struttura del modulo è rimasta coerente. Si parte dall'architettura della CPU: registri, bus, ciclo fetch-decode-execute [16], come modello operativo per ragionare su cosa succede quando un programma viene eseguito. Si passa alla scrittura di programmi con I/O, interrupt e gestione della memoria. Il debug a basso livello è centrale: osservare i registri cambiare valore istruzione per istruzione produce una comprensione che nessuna spiegazione a voce riesce a replicare [5]. La connessione con il linguaggio C, comparando codice compilato e sorgente assembly, mostra allo studente cosa fa il compilatore sotto il cofano.

Un tema di particolare valore formativo è quello delle calling conventions, le convenzioni che regolano il passaggio di parametri tra chiamante e chiamato attraverso lo stack, che dà senso alla pila come meccanismo reale su cui si reggono chiamate e ritorni.

Gli obiettivi formativi sono riferibili alle Indicazioni Nazionali per l'informatica [3]: comprendere l'architettura dei sistemi di elaborazione (O-S-N-1); valutare rappresentazioni alternative della stessa informazione (O-S-D-1); progettare programmi modulari con procedure e funzioni (O-S-P-6); riconoscere il contributo delle parti di un programma al suo funzionamento (O-S-P-1).

7 Tempi, valutazione e risultati osservati

Le osservazioni che seguono, derivate dall'insegnamento del modulo a numerose classi terze nel corso di 29 anni, hanno carattere qualitativo e non costituiscono risultati di una sperimentazione controllata.

Il modulo occupa le due ore di laboratorio delle quattro settimanali di Sistemi e Reti in terza, secondo la programmazione di dipartimento: da metà del primo quadrimestre a fine gennaio-febbraio, dopo che a teoria è stata introdotta la struttura del processore. Sono previste due verifiche di laboratorio, calibrate sulle esercitazioni guidate già svolte. La collocazione non è casuale: il modulo viene proposto per primo perché gli esiti tendono a essere più bassi, lasciando alla seconda parte dell'anno contenuti più discorsivi per bilanciare la valutazione. La prova di gennaio valuta soprattutto conformità alla traccia e coerenza dell'algoritmo; quella di febbraio, quando il modulo si concentra su stack e procedure, valuta soprattutto passaggio dei parametri e modularizzazione.

Gli studenti che si orientano bene in Informatica tendono a produrre risultati accettabili anche in assembly; chi incontra difficoltà già a livello di diagrammi di flusso trova nel basso livello uno strato ulteriore di opacità, perché la codifica, quando l'algoritmo è già dominato, diventa applicazione in gran parte meccanica dei pattern della Sezione 5. Semplificare il modulo non risolverebbe questo problema di fondo, che riguarda un modo di pensare prima ancora che una sintassi; l'assenza di sperimentazione formale rigorosa su questi aspetti resta un limite riconosciuto anche negli studi più recenti [15].

Una riflessione maturata nel confronto con colleghi di dipartimento: se l'assembly ha ceduto il passo ai linguaggi ad alto livello senza perdere valore formativo, lo stesso rapporto sembra ripetersi oggi un livello più in alto, tra linguaggio ad alto livello e prompt rivolto all'IA generativa. Un prompt costruito con la stessa disciplina richiesta per definire input, output e dipendenze di un programma produce risultati migliori anche quando a scrivere il codice è la macchina.

Nell'esperienza dell'autore, gli studenti sembrano sviluppare un modello mentale più preciso della macchina reale: mostrano di sapere cosa significa un puntatore e perché un buffer overflow è pericoloso. La risposta è quindi sì: ha ancora senso insegnare l'assembly nel 2026, con motivazione ancorata alla cybersecurity, ambiente aggiornato e struttura didattica che privilegi la comprensione profonda. Non un reperto storico, ma uno strumento che continua a offrire un contributo formativo difficilmente sostituibile: insegnarlo non significa opporsi all'IA generativa, ma preparare gli studenti a usarla con maggiore consapevolezza.

Ringraziamenti

Ringrazio Bux, Candilio, Di Monte, Di Nanna, Longo, Tornesello e Vacca, colleghi del Dipartimento di Informatica dell'IISS Marconi-Hack di Bari, per la lettura di una versione preliminare di questo contributo. L'analogia tra assembly e prompt engineering nasce da una riflessione condivisa con Vacca.

Bibliografia

[1] Agarwal, K.K. e Agarwal, A. Do we need a separate assembly language programming course? Journal of Computing Sciences in Colleges, 19(4), 246–251, 2004.

[2] Böhm, C. e Jacopini, G. Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules. Communications of the ACM, 9(5), 366–371, 1966.

[3] CINI – Consorzio Interuniversitario Nazionale per l'Informatica (2017). Proposta di Indicazioni Nazionali per l'Informatica nella Scuola. https://www.consorzio-cini.it/images/Proposta-Indicazioni-Nazionali-Informatica-Scuolanumerata.pdf

[4] Erickson, J. Hacking: The Art of Exploitation. No Starch Press, 2008.

[5] Hyde, R. The Art of Assembly Language. No Starch Press, 2010.

[6] Imamura, K. Assembly language is more than a teaching tool. Journal of Computing Sciences in Colleges, 20(2), 49–54, 2004.

[7] Indolfi, P. La discesa nel labirinto. Avventure di un programmatore. Menzione speciale, I Premio Letterario Nazionale Tommaso Di Ciaula, 9 novembre 2023.

[8] Ministero dell'Istruzione (2010). Riordino degli istituti tecnici. Allegati tecnici. Indicazioni per Sistemi e Reti.

[9] mycompiler.io. Compilatore online per assembly e altri linguaggi. https://www.mycompiler.io

[10] Null, L. e Lobur, J. The Essentials of Computer Organization and Architecture. Jones & Bartlett Learning, 6a ed., 2019.

[11] Parashkevov, D. SASM – Simple crossplatform IDE for NASM, MASM, GAS, FASM. Open source, GNU GPL v3.0. https://dman95.github.io/SASM/

[12] Patt, Y.N. e Patel, S.J. Introduction to Computing Systems: From Bits and Gates to C and Beyond. McGraw-Hill, 3a ed., 2020.

[13] Post, E. Real Programmers Don't Use Pascal. Datamation, vol. 29 n. 7, luglio 1983. https://www.pbm.com/~lindahl/real.programmers.html

[14] Replit Inc. Replit – piattaforma di sviluppo collaborativo online. https://replit.com

[15] Rivera-Alvarado, E. e Guadamuz, S. A Case for the Importance of Formal Experimentation in Teaching Programming: A Brief Study with Assembly Language. In: ICT for Intelligent Systems (ICTIS 2024), Smart Innovation, Systems and Technologies, vol. 403, 11–20. Springer, Singapore, 2024.

[16] Tanenbaum, A.S. e Austin, T. Architettura dei calcolatori. Un approccio strutturale, 6a ed. italiana a cura di O. D'Antona. Pearson Education Italia, 2013. ISBN 9788871929620.

[17] Zuo, F., Tompkins, C., Qian, G., Rhee, J., Qu, X. e Yang, B. ChatGPT as an Assembly Language Interpreter for Computing Education. Journal of Computing Sciences in Colleges, 40(2), 73–82, 2024.

Educare sviluppatori responsabili: Dai fruitori attuali ai creatori consapevoli

Fabio Sartori[1] and Lorenzo Caputo[1]

Dipartimento di Informatica, Sistemstica e Comunicazione
Università degli Studi di Milano–Bicocca viale Sarca 336, 20126, Milano
nome.cognome@unimib.it

Abstract

Mentre la ricerca sulla protezione dei minori nel digitale si concentra sulla valutazione di app da prospettive di utenti e genitori, emerge un vuoto critico: come formiamo gli sviluppatori che creano queste app? Questo contributo presenta il progetto DREAM (Digital Resilience, Education, and Awareness for Minors), finanziato dall'Unione Europea, che sviluppa linee guida per l'insegnamento della progettazione responsabile di applicazioni destinate ai minori. L'innovazione chiave è il riconoscimento degli insegnanti come partner attivi nella ricerca, non destinatari passivi. Il Work Package 3 condotto dall'Università di Milano-Bicocca coinvolge educatori di quattro paesi europei in una ricerca partecipativa che vuole integrare competenze etiche nei curricula informatici fin dalla scuola secondaria.

1 Introduzione: Il paradosso dello sviluppatore

Negli ultimi anni, la ricerca sulla protezione dei minori nel digitale ha prodotto risultati importanti: studi approfonditi su quali caratteristiche rendono "buone" le app per bambini, dall'ottica di genitori, educatori e degli stessi fruitori minori [1]. Sono stati sviluppati framework per valutare la qualità pedagogica, la sicurezza, l'accessibilità [2]. Eppure emerge un paradosso fondamentale: mentre sappiamo sempre più cosa rende un'app sicura ed educativa, sappiamo molto poco di chi la crea e come formarlo.

Gli studenti di oggi che seguono corsi di informatica, programmazione e sviluppo software saranno gli sviluppatori di domani. Essi frequentano scuole secondarie e università in cui imparano competenze tecniche rigorose: linguaggi di programmazione, architetture software, design patterns, metodologie agile. Raramente, però, incontrano insegnamenti espliciti su diritti dei minori nel digitale, etica della progettazione, consapevolezza dei rischi che le loro scelte progettuali possono comportare per utenti vulnerabili.

Come possiamo aspettarci che gli sviluppatori di app per minori proteggano i diritti e il benessere dei bambini se non hanno ricevuto formazione esplicita in merito?

2 Il progetto DREAM: Contesto e approccio innovativo

Il progetto DREAM (Digital Resilience, Education, and Awareness for Minors) nasce da questa domanda. Finanziato dal programma EU CERV (Citizens, Equality, Rights and Values), è un'iniziativa di 24 mesi (febbraio 2026 - gennaio 2028) che coinvolge un consorzio internazionale (Italia, Grecia, Polonia, Ucraina) con l'obiettivo di sviluppare linee guida educative per insegnare la progettazione etica e responsabile di applicazioni destinate ai minori.

Ciò che rende DREAM innovativo, dal punto di vista pedagogico, non è solo l'oggetto di ricerca (educazione degli sviluppatori), ma il metodo: gli insegnanti non sono destinatari passivi di linee guida già costruite altrove, bensì *partner attivi nella ricerca.* Questa scelta metodologica nasce da una considerazione semplice ma pertinente: gli insegnanti conoscono gli studenti in modo profondo. Li conoscono

come fruitori attuali di app destinate ai minori (con competenze di *media literacy* molto sviluppate); li conoscono come discenti di competenze informatiche (con punti di forza, fragilità, misconcezioni tipiche); li conosceranno come futuri professionisti (immaginando traiettorie professionali, difficoltà nel mercato del lavoro, competenze richieste).

DREAM riconosce questa expertise multidimensionale e la trasforma in co-ricerca, attraverso la definizione di momenti di confronto a loro dedicati, sottoforma di 16 workshop distribuiti nei quattro paesi (4 per paese), coinvolgendo 160 professionisti dell'educazione (insegnanti di scuola secondaria e docenti universitari). I workshop hanno duplice funzione:

1. Validare le linee guida rispetto a realtà pedagogiche diverse (italiano vs greco vs polacco vs ucraino);
2. Raccogliere feedback per raffinamento iterativo.

La scelta di coinvolgere gli educatori non solo come "validatori" finali ma come "co-progettisti" lungo l'intera ricerca rappresenta un cambio di paradigma pedagogico [3] grazie al quale gli insegnanti forniscono input sulla rilevanza dei temi; collaborano al design dei questionari; validano le linee guida nella pratica; diventano "ambasciatori" per disseminazione delle buone pratiche.

3 Implicazioni per la didattica dell'informatica

Questo contributo si colloca all'interno della ricerca sulla didattica dell'informatica affrontando un gap critico identificato dalla letteratura recente. La systematic literature review di Brown et al. [4] sui 40 anni di ricerca in computing education documenta che, sebbene la comunità abbia sviluppato ampi strumenti pedagogici per insegnare programmazione e algoritmi, la ricerca sulla formazione della consapevolezza etica degli studenti rimane frammentaria, priva di consenso su concezioni condivise dell'etica, e con scarsi risultati nella misurazione dell'apprendimento. DREAM affronta questo gap da una prospettiva specifica e rilevante: riconoscere che gli studenti di oggi, come fruitori di app destinate ai minori, diventeranno domani gli sviluppatori responsabili della protezione di questi stessi utenti.

Il valore della ricerca risiede nell'identificazione esplicita di questo problema pedagogico e nella proposta di un programma di ricerca strutturato che coniughi tre elementi. Innanzitutto, una definizione circoscritta dell'etica nel contesto dell'informatica: non "etica generica" ma responsabilità dello sviluppatore verso utenti vulnerabili. Secondo, un approccio metodologico partecipativo che riconosce gli insegnanti come co-ricercatori esperti della realtà didattica locale. Terzo, strumenti di raccolta dati robusti progettati per misurare la consapevolezza e l'intenzione professionale degli studenti.

Come evidenza preliminare del programma di ricerca proposto, il Work Package 3 sta sviluppando un questionario strutturato che mappa quattro concezioni della responsabilità etica secondo la tassonomia delineata in [4]: *principi etici*, *impatto sociale*, *domini applicativi specifici*, e *identità professionale*. Il questionario sarà somministrato a 400 studenti e 40 docenti in quattro paesi, progettato per fornire dati quantitativi e qualitativi sulla consapevolezza attuale e sulla apertura all'apprendimento. Questo design metodologico operazionalizza direttamente le raccomandazioni della letteratura per misurare l'apprendimento etico in informatica. Il programma di ricerca DREAM posiziona quindi la didattica dell'informatica verso una prospettiva di responsabilità sostenibile, essenziale per preparare sviluppatori consapevoli della protezione dei minori nel digitale.

4 Conclusioni

DREAM rappresenta un'occasione per pensare diversamente a come formiamo sviluppatori di tecnologie destinate ai minori. Non attende che leggi e regolamenti impongano protezioni (approccio reattivo),

ma investe nella formazione durante gli studi (approccio proattivo). Riconoscendo il ruolo cruciale degli insegnanti come co-ricercatori, il progetto genera linee guida ancorate nella realtà pedagogica, culturalmente sensibili, e concrete. Il progetto vuole sottolineare che la protezione dei minori nell'uso delle applicazioni è responsabilità non solo di genitori o regolatori, come ampiamente affrontanto nello stato dell'arte [5] ma anche di chi crea le applicazioni stesse e questa consapevolezza deve crescere durante la formazione.

References

[1] Dias, P., and Brito, R. (2021). Criteria for selecting apps: Debating the perceptions of young children, parents and industry stakeholders. Computers & Education, 165, 104134.

[2] Papadakis, S., Vaiopoulou, J., Kalogiannakis, M., and Stamovlasis, D. (2020). Developing and exploring an evaluation tool for educational apps (ETEA) targeting kindergarten children. Sustainability, 12(10), 4201.

[3] Wenger, E. (1999). Communities of practice: Learning, meaning, and identity. Cambridge university press.

[4] Brown, N., Xie, B., Sarder, E., Fiesler, C., and Wiese, E. S. (2024). Teaching ethics in computing: A systematic literature review of ACM computer science education publications. ACM Transactions on Computing Education, 24(1), 1-36.

[5] Alves de Castro, C., Carthy, A., and OŔeilly Dr, I. (2022). An Ethical Discussion About the Responsibility for Protection of Minors in the Digital Environment: A state-of-the-art review.

Insegnare l'Intelligenza Artificiale con Umanità: superare le barriere attraverso l'AI etica e inclusiva

Daniela Pieraccini
Istituto Comprensivo Anchise Picchi (LI)
daniela.pieraccini@scuola.istruzione.it

Abstract

Questo contributo presenta un percorso didattico di IA literacy condotto con 28 studenti della scuola secondaria di primo grado (11–13 anni), incentrato sull'addestramento supervisionato di un classificatore audio e sulla sua integrazione in un ambiente di programmazione a blocchi. Il percorso è stato sviluppato in coerenza con le indicazioni curricolari nazionali più recenti, che introducono un primo approccio critico all'informatica e all'Intelligenza Artificiale nel curricolo STEM della scuola secondaria di primo grado. (D.M. 211/2025).

Attraverso piattaforme accessibili e prive di login obbligatorio, gli studenti hanno costruito un dataset audio, addestrato e validato un modello supervisionato, integrandolo in uno sprite che simula, tramite comandi vocali, il movimento di una carrozzina. I risultati mostrano che studenti di questa età comprendono i principi del machine learning supervisionato e riflettono criticamente su dati, errori e bias, oltre che, indirettamente, su accessibilità e inclusione. Il contributo discute il valore formativo del percorso oltre gli strumenti specifici impiegati e propone raccomandazioni per la sua replicabilità.

Parole chiave: Intelligenza Artificiale, scuola secondaria di primo grado, machine learning supervisionato, classificazione audio, programmazione a blocchi, tecnologia assistiva, pensiero computazionale.

1 Introduzione

Con la crescente pervasività dell'Intelligenza Artificiale (IA) nella vita quotidiana, è sempre più urgente fornire ai giovani studenti gli strumenti per confrontarsi criticamente con i sistemi di IA. Le nuove Indicazioni Nazionali per il curricolo del primo ciclo (D.M. 211/2025) rispondono a questa esigenza introducendo, nell'area STEM, obiettivi di apprendimento legati all'informatica e un primo approccio critico all'IA, volti a superare un uso meramente strumentale delle tecnologie digitali [1]. L'educazione inclusiva richiede inoltre strategie capaci di sostenere studenti con bisogni educativi speciali, in particolare difficoltà linguistiche, relazionali o motorie. Questo progetto nasce dall'incontro fra queste due esigenze: l'acquisizione di competenze di AI literacy in linea con il curricolo nazionale, e l'esplorazione di un caso applicativo — un'interfaccia vocale per il controllo di una carrozzina simulata — che collega la classificazione audio a un tema di accessibilità, senza assumerlo come oggetto esclusivo del percorso.

Il contributo descrive gli esiti dell'esperienza in classe, offrendo evidenze pratiche di come le attività di IA possano promuovere l'alfabetizzazione digitale in coerenza con il curricolo nazionale.

2 Sfondo teorico

Studi precedenti hanno dimostrato il valore degli ambienti di programmazione a blocchi (es. Scratch, Stretch3) per introdurre il pensiero computazionale e semplici concetti di machine learning negli adolescenti. La robotica controllata dalla voce e i sistemi di comunicazione aumentativa e alternativa si sono dimostrati efficaci nel sostenere l'autonomia e l'inclusione, ma la loro integrazione con la didattica dell'IA è ancora un campo emergente. Questo lavoro intreccia questi filoni, integrando il controllo vocale basato su ML all'interno di un'attività di IA literacy.

Le nuove Indicazioni Nazionali (D.M. 211/2025) collocano l'informatica e un primo approccio all'IA nell'area STEM, valorizzando il pensiero computazionale come competenza trasversale e promuovendo un uso critico e consapevole delle tecnologie digitali [1]. Il percorso contribuisce a questi traguardi rendendo tangibili concetti come dataset, addestramento supervisionato e valutazione degli errori di un modello, permettendo agli studenti di sperimentare in prima persona la costruzione di un sistema di IA e di discuterne criticamente potenzialità e limiti.

3 Metodologia

Il percorso è stato realizzato in un istituto di scuola secondaria di primo grado in un contesto rurale italiano, con 28 studenti di 11–13 anni e background eterogenei, affiancando attività collaborative unplugged ad attività di programmazione a blocchi.

3.1 Struttura in due fasi

In una prima fase esplorativa, gli studenti hanno partecipato a discussioni guidate e attività unplugged e tecnologico-mediate sulla pervasività dell'IA, utilizzando alberi decisionali simulati e giochi di classificazione, con il supporto di Teachable Machine per costruire modelli personalizzati. La seconda fase ha privilegiato il project-based learning con Stretch3 [3] ed estensioni IA. Teachable Machine [4], non richiedendo login, ha garantito la conformità al GDPR per studenti sotto i 14 anni; i modelli addestrati sono stati integrati nell'ambiente di programmazione a blocchi, per il test applicativo.



Figura 1: Studenti al lavoro in attività unplugged

3.2 Il processo di costruzione del modello

Gli studenti hanno creato un dataset audio, suddiviso in due gruppi: uno più ampio destinato alla fase di addestramento e uno più piccolo di validazione, evitando così di riutilizzare gli stessi campioni per la verifica. Il modello addestrato è stato validato sui campioni di test e, dove necessario, perfezionato; il modello finale è stato importato nell'ambiente di programmazione a blocchi per essere integrato in un artefatto digitale interattivo.

4 Approccio didattico

Gli studenti hanno lavorato in gruppi collaborativi, con progettazione iterativa e tutoraggio tra pari. La riflessione guidata e la sperimentazione strutturata hanno permesso di comprendere l'IA

come dominio tecnico ed etico. La progettazione di uno strumento accessibile ha offerto un'occasione, per quanto circoscritta, di riflessione su empatia, accessibilità e inclusione, senza costituire l'obiettivo primario del percorso.

5 Risultati

È stato testato l'abbinamento Teachable Machine + Stretch3, che ha offerto un'opzione conforme al GDPR e senza login per i minori, con interfaccia intuitiva. La Tabella 1 sintetizza gli strumenti impiegati e il relativo vantaggio didattico.

Strumento	**Funzione**	**Vantaggio didattico**
Teachable Machine (Google)	Addestramento di modelli ML supervisionati (audio)	Nessun login richiesto: conforme al GDPR per minori di 14 anni
ML for Kids	Creazione di modelli ML personalizzati	Account gestiti dal docente con nickname; persistenza dei progetti (gratuita per 15 giorni)
Stretch3	Programmazione a blocchi con estensioni AI	Integrazione diretta dei modelli Teachable Machine

Tabella 1: Strumenti impiegati nel percorso e relativo vantaggio didattico

5.1 Movimento controllato dalla voce (simulazione di carrozzina vocale)

Gli studenti hanno selezionato i comandi vocali, registrato campioni audio e addestrato un modello supervisionato capace di associare gli input vocali a comandi direzionali, poi validato e perfezionato incorporando ulteriori campioni. Il modello è stato integrato in un ambiente di programmazione dove gli studenti hanno applicato le coordinate cartesiane per simulare il movimento dello sprite.

I primi modelli, addestrati con un solo parlante, avevano generalizzazione limitata ma accuratezza elevata; le iterazioni successive, con voci appartenenti a più studenti, hanno migliorato l'accuratezza su profili vocali diversi, pur risultando meno efficaci nel riconoscere il futuro "proprietario" della carrozzina. Gli studenti hanno appreso che i comandi monosillabici migliorano velocità e accuratezza, che parole foneticamente simili riducono l'affidabilità, e che antonimi ben scelti ("avanti"/"indietro") garantiscono prestazioni migliori.

Il modello di ML è stato collegato con successo al movimento dello sprite, rinforzando concetti matematici come le coordinate; il progetto è stato validato con dimostrazioni in classe e test tra pari.

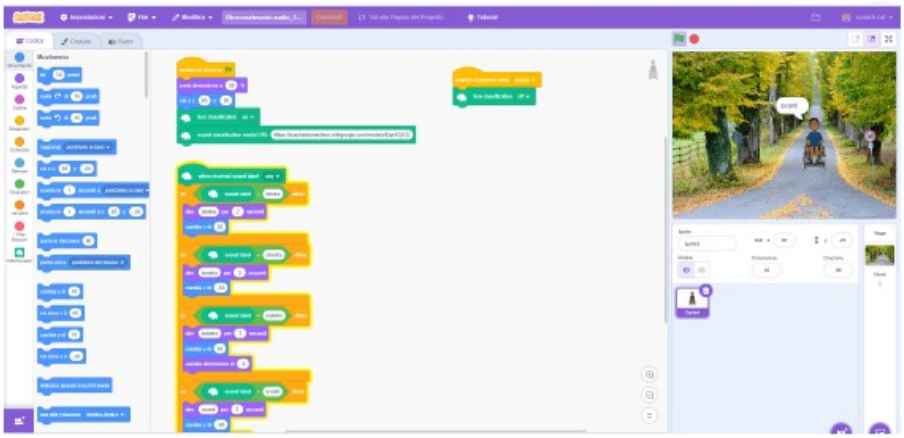


Figura 2: Progetto Stretch3 con blocco di classificazione vocale integrato e sprite della carrozzina

6 Lezioni apprese

Il percorso ha evidenziato come elemento particolarmente efficace una struttura pratica e basata su progetti, che ha permesso agli studenti di confrontarsi con un problema reale attraverso strumenti accessibili. Progettare uno sprite assistivo controllato dalla voce non è stato solo un compito tecnico, ma anche un'occasione di riflessione su inclusione e accessibilità. L'apprendimento collaborativo ha favorito le interazioni tra pari, e la natura multimodale dell'attività — visiva, auditiva, cinestetica — ha favorito studenti con profili di apprendimento differenti.

Non sono mancate criticità: modelli addestrati in classi rumorose hanno prodotto risultati poco affidabili, e gli studenti hanno necessitato di una preparazione preliminare su terminologia tecnica come “dataset” e “bias”. Problemi tecnici (microfoni scadenti, connessioni instabili) hanno occasionalmente rallentato i progressi.

Riteniamo che il valore didattico dell'esperienza non risieda negli strumenti specifici impiegati, destinati a mutare rapidamente, ma nelle competenze trasversali esercitate — costruzione e valutazione critica di un dataset, riconoscimento degli errori di un modello, comprensione di limiti e bias — meno soggette all'obsolescenza tecnologica, a condizione di un'adeguata preparazione preliminare di studenti e docenti.

7 Raccomandazioni per la replicabilità

I requisiti tecnici per replicare il percorso sono contenuti: un computer o tablet standard con microfono e connessione stabile è generalmente sufficiente. È tuttavia essenziale che i docenti ricevano una preparazione iniziale sui concetti di machine learning — apprendimento supervisionato, dataset, bias algoritmico — poiché aumenta significativamente la fiducia dei docenti e il successo in classe.

Tra le criticità più comuni figurano la scarsa familiarità dei docenti con gli strumenti di IA, i vincoli di tempo curricolari, e un'iniziale resistenza degli studenti verso approcci aperti basati su tentativi ed errori. Il percorso resta comunque adattabile: in contesti con risorse limitate si possono usare versioni semplificate dell’attività, ad esempio con registrazioni offline. Incoraggiare gli studenti a documentare il processo e ad accogliere gli errori come parte dell'apprendimento favorisce lo sviluppo di una mentalità di crescita e amplifica l'impatto educativo.

8 Conclusioni

L'esperienza ha confermato che studenti di 11–13 anni sono in grado di comprendere i concetti del machine learning supervisionato e di costruire un classificatore funzionante, integrato in un artefatto di programmazione a blocchi con chiara rilevanza applicativa, in coerenza con le nuove Indicazioni Nazionali [1]. Il caso applicativo scelto ha inoltre offerto un'occasione indiretta ma significativa di riflessione su accessibilità e inclusione.

Il valore formativo del percorso non dipende, a nostro avviso, dalla persistenza nel tempo degli strumenti impiegati, ma dallo sviluppo di competenze trasversali — pensiero computazionale, consapevolezza critica di dati e modelli — che restano rilevanti anche in un settore in rapida evoluzione, a condizione di un'adeguata preparazione preliminare.

Bibliografia

[1] Ministero dell'Istruzione e del Merito. Decreto Ministeriale 211/2025 – Nuove Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione. https://www.mim.gov.it, 2025.

[2] ML for Kids. https://machinelearningforkids.co.uk

[3] Stretch3. https://stretch3.github.io

[4] Teachable Machine – Google. https://teachablemachine.withgoogle.com

[5] UN Agenda 2030. Sustainable Development Goals – SDG 4 (Quality Education), SDG 10 (Reduced Inequalities). https://sdgs.un.org/goals

Laboratorio di introduzione alle reti neurali

Sara Capecchi[1], Antonio Giordano[2], Paolo Milan[3], and Daniele Paolo Radicioni[4]

[1] Università degli Studi di Torino
sara.capecchi@unito.it
[2] Università degli Studi di Torino
antonio.giordano704@edu.unito.it
[3] Liceo Einstein (Torino)
p.milan@einsteintorino.edu.it
[4] Università degli Studi di Torino
daniele.radicioni@unito.it

Abstract

Il lavoro presenta la sperimentazione di un percorso di introduzione alle reti neurali per la scuola secondaria di secondo grado. Il percorso, centrato sul percettrone, integra attività unplugged, strumenti no-code e programmazione secondo un approccio di scoperta guidata. I primi risultati evidenziano efficacia nella costruzione di modelli mentali accurati e buoni livelli di coinvolgimento, aprendo a prospettive di validazione su scala più ampia e integrazione curricolare sistematica.

1 Introduzione

Il presente lavoro descrive una sperimentazione condotta nell'ambito di una tesi di Laurea Magistrale in Informatica presso l'Università degli Studi di Torino [13], per valutare l'efficacia di un percorso introduttivo ai fondamenti delle reti neurali nelle scuole secondarie di secondo grado. L'obiettivo era verificare se una progressione che alterna esperienza concreta, formalizzazione e implementazione possa rendere accessibili concetti tipicamente percepiti come astratti, quali le nozioni di percettrone, addestramento, errore, decision boundary e separabilità lineare. Il percorso è stato progettato cercando di bilanciare rigore concettuale e accessibilità secondo un impianto metodologico strutturato, includendo questionari post-intervento e raccolta di evidenze empiriche, in conformità con i requisiti dei comitati etici competenti. Le sezioni successive presentano il quadro teorico e metodologico, descrivono le principali attività del percorso, discutono i primi risultati della sperimentazione e delineano possibili sviluppi futuri.

2 Contesto

La sperimentazione si inserisce nel percorso di ricerca dell'Università degli Studi di Torino [5] [2] [3] [28] [23] in un contesto di crescente attenzione verso l'insegnamento dell'IA nella Scuola, testimoniata dai recenti aggiornamenti dei documenti programmatici ministeriali [21] [22] e dal numero di contributi sul tema negli atti di ITADINFO [12] [6] [24].

La letteratura conferma la centralità dei prerequisiti informatici e matematici ed evidenzia complessità specifiche nell'insegnamento delle reti neurali in contesti pre-universitari. Gli approcci top-down tendono a rafforzare la percezione di questi sistemi come black-box, mentre quelli bottom-up rischiano di risultare frammentari senza una chiara progressione concettuale. In questo panorama si inserisce l'approccio unplugged, adottato da diversi studi sulle reti neurali [9] [19] [33] [17] [16] [18] [8] [1] [30] [25] per valorizzare apprendimento attivo e scoperta guidata. L'efficacia di tali attività dipende dall'integrazione con i principi teorici di riferimento, affinché la dimensione ludica non prevalga sulla comprensione concettuale.

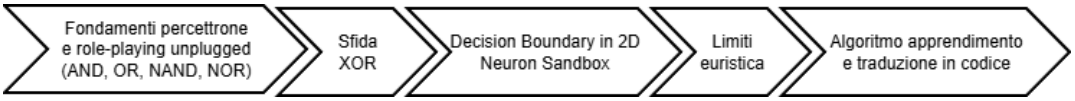


Figure 1: Architettura percorso

Nella sperimentazione descritta l'unplugged non sostituisce la programmazione, ma ne costituisce la premessa concettuale secondo la logica delle onde semantiche [10], e il role-playing precede la formalizzazione per mitigare il rischio di superficialità. Inoltre, la scelta di adottare una dimensione collettiva e un'impostazione agnostica rispetto alla funzione globale della rete mira a favorire la costruzione di una notional machine[1] adeguata [29], contrastando le misconceptions più diffuse [20].

3 Il percorso sperimentato

3.1 Architettura generale

Il percorso, centrato sul concetto di percettrone e sull'apprendimento supervisionato, si articola in 12 ore distribuite in moduli che integrano teoria, attività unplugged, formalizzazione matematica e programmazione (Figura 1). Le attività in aula sono precedute da un'introduzione blended con le videolezioni del progetto EmpAI [3] sui concetti generali dell'IA. Il percorso vero e proprio inizia con una lezione frontale sui concetti fondamentali del percettrone, seguita da attività unplugged e giochi di ruolo come il Club Neuronale. Questi, in modalità euristica, conducono gli studenti dalla presa di confidenza con i concetti di base fino all'esplorazione dell'apprendimento automatico, costruendo le basi per la sfida XOR. Quest'ultima costituisce il momento di rottura che motiva la formalizzazione matematica, progressivamente consolidata con strumenti no-code. I limiti dell'approccio euristico al crescere degli input motivano infine il passaggio all'implementazione in codice di un algoritmo di apprendimento.

3.2 Club Neuronale

Il Club Neuronale è una delle attività proposte quando gli studenti hanno familiarità con il percettrone e serve a introdurre il tema dell'apprendimento automatico (fase *role-playing unplugged* in Figura 1). L'attività consente agli studenti di esperire fisicamente la distinzione tra input, pesi, soglia, output, feedback e aggiornamento dei parametri. L'obiettivo è addestrare un percettrone sulla funzione logica AND, incarnata dalla metafora di un sistema di sicurezza che concede l'accesso ai clienti in possesso di un braccialetto luminoso e con il dress-code appropriato.

Preparazione

La classe viene divisa in tre gruppi. Il primo impersona il percettrone, con due studenti che eseguono la *feature extraction* traducendo le immagini dei clienti nei due input binari (braccialetto e dress-code), distinguendo così il dato grezzo dalla rappresentazione simbolica. Un terzo studente, il sommatore, opera di spalle alla lavagna con parametri iniziali casuali e impara il comportamento corretto attraverso tentativi ed errori. Il secondo gruppo conosce la regola target (funzione AND) e fornisce il feedback al sommatore, mentre i componenti del terzo gruppo assistono come spettatori agnostici.

[1] Modelli mentali del funzionamento di un sistema computazionale, nel caso delle reti neurali utili a rendere comprensibili il ruolo di input, pesi, soglia e processo di aggiornamento.



(a) *DressCode* = 0 *Braccialetto* = 0



(b) *DressCode* = 0 *Braccialetto* = 1



(c) *DressCode* = 1 *Braccialetto* = 0



(d) *DressCode* = 1 *Braccialetto* = 1

Figure 2: Club Neuronale - Training set

Addestramento

L'attività procede mostrando a tutti (escluso il sommatore) gli esempi del training set (Figura 2). Pur esaurendo i casi della funzione AND, la scelta permette di focalizzarsi sulla meccanica interna del percettrone. Gli studenti-input trasmettono i valori al sommatore, che calcola l'output confrontando la somma pesata con la soglia. Quando il secondo gruppo rileva un errore e fornisce il feedback, il sommatore adegua pesi e soglia in modo euristico, ragionando sulla direzione della correzione. Questo approccio empirico e intuitivo motiva e anticipa la successiva formalizzazione della regola di apprendimento.

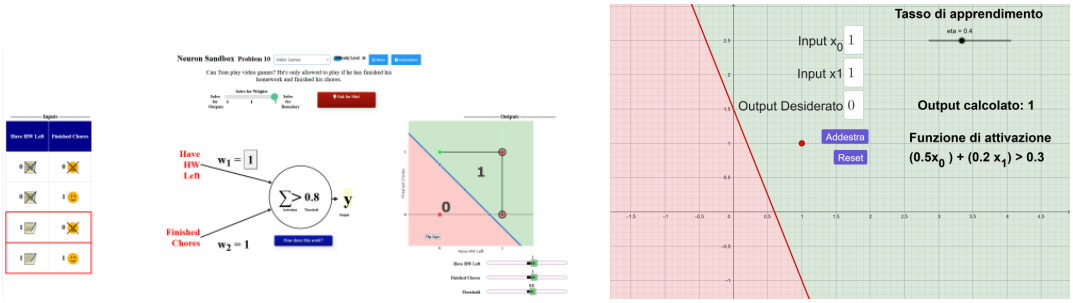
Inferenza e discussione

Il percettrone addestrato viene messo in inferenza, mostrando come "l'apprendimento automatico consenta a un sistema di acquisire comportamenti senza che questi vengano esplicitamente programmati"[31]. A questo punto il sommatore si volta verso la lavagna e osserva le immagini su cui prendeva decisioni e insieme al gruppo agnostico formula ipotesi sulla regola appresa, aprendo una discussione sui temi dell'explainability e del domain agnosticism.

3.3 Dallo XOR al decision boundary

Acquisita familiarità con la configurazione euristica dei parametri sperimentata nell'attività precedente, viene proposta la sfida di addestrare un percettrone sulla funzione XOR (fase *Sfida XOR* in Figura 1). Le dinamiche osservate nella sperimentazione sono ricorrenti: l'applicazione dello stesso approccio rivelatosi efficace nel Club Neuronale conduce a tentativi infruttuosi, portando gli studenti a intuire che la configurazione cercata non esiste. Il fallimento dei tentativi euristici non viene trattato come un ostacolo, ma come un passaggio progettato per generare una domanda di formalizzazione: perché il percettrone non riesce a risolvere XOR?

La funzione diventa quindi il caso paradigmatico per mostrare che un singolo percettrone può rappresentare solo problemi linearmente separabili. La richiesta di dimostrarlo stimola l'astrazione algebrica, portando a impostare un sistema di disequazioni lineari di cui verificare formalmente l'assenza di soluzioni. Questo momento di esplorazione parzialmente autonoma evidenzia la necessità di modelli più complessi e l'attività si conclude con un esercizio Bebras [27], che consolida la comprensione dei limiti del percettrone singolo. Segue quindi una lezione di formalizzazione sui concetti di decision boundary e separabilità lineare, motivando la necessità di una trattazione matematica rigorosa.



(a) Neuron Sandbox (b) GeoGebra

Figure 3: Strumenti No-Code (Neuron Sandbox) e Low-code (GeoGebra)

3.4 Dagli strumenti no-code alla programmazione

Consolidata la formalizzazione matematica si passa al Neuron Sandbox [30] (fase *Decision Boundary in 2D* in Figura 1), strumento no-code che propone esercizi in contesti narrativi in continuità con l'approccio adottato, assolvendo un duplice ruolo. Da un lato consente un rinforzo autonomo dei concetti, tramite la manipolazione dei parametri e la visualizzazione geometrica della frontiera decisionale (Figura 3a). Dall'altro estende l'esplorazione a percettroni con più input, dove la visualizzazione degli iperpiani non è disponibile e l'approccio euristico diventa impraticabile (fase *Limiti euristica* in Figura 1). Questa limitazione suggerisce lo sviluppo di un processo algoritmico di aggiornamento dei parametri in funzione dell'errore, affrontato per gradi. Dopo l'addestramento manuale con GeoGebra (Figura 3b) usato come strumento low-code, l'ultima fase prevede un'implementazione ispirata al metodo PRIMM [26]. Nato per la didattica della programmazione, l'approccio è qui applicato per guidare gli studenti dalla comprensione di segmenti di codice preesistenti alla scrittura autonoma dell'algoritmo.

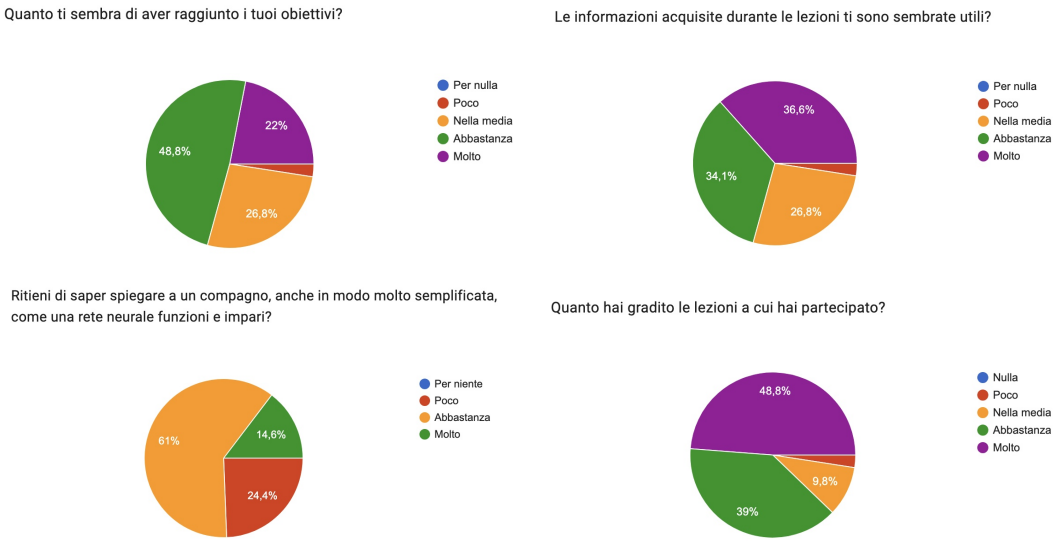


Figure 4: Risultati del questionario

4 Risultati

La valutazione è stata condotta tramite osservazioni sistematiche e questionari post-intervento [14] somministrati al termine della fase unplugged e basati su strumenti validati [7][11]. Il campione comprende una classe quinta ITI Informatico (14 studenti, sola fase unplugged) e due classi terze di Liceo Scientifico Scienze Applicate (14 e 19 studenti, intero percorso).

I risultati (Figure 4 e 5) suggeriscono il raggiungimento dell'obiettivo di bilanciare rigore concettuale e accessibilità. Le attività unplugged si sono rivelate efficaci nella costruzione di modelli mentali accurati sul funzionamento del percettrone, dimostrando che le competenze richieste sono già consolidate all'inizio del secondo biennio. Il role-playing ha mantenuto alto il coinvolgimento e la sfida XOR ha costituito un momento di dissonanza cognitiva produttiva, stimolando la necessità di formalizzazione.

I livelli di comprensione sono comparabili tra i gruppi, con lievi differenze coerenti con gli indirizzi: maggiore familiarità con la logica booleana nell'ITI, maggior propensione al rigore matematico nel Liceo. Emergono tendenze positive per apprendimenti, autovalutazione e soddisfazione, con un'associazione tra autovalutazione elevata, alta soddisfazione e migliori prestazioni.

La sperimentazione nelle classi terze si è innestata in un laboratorio del Piano Lauree Scientifiche [32] e, in una delle due classi, i concetti appresi sono confluiti nella costruzione di un classificatore binario in un percorso transdisciplinare sulla qualità dell'aria.

Sebbene i risultati siano positivi, la gestione d'aula si è delineata come una criticità da monitorare per contesti ordinari, poiché la sperimentazione ha beneficiato della presenza di due docenti.

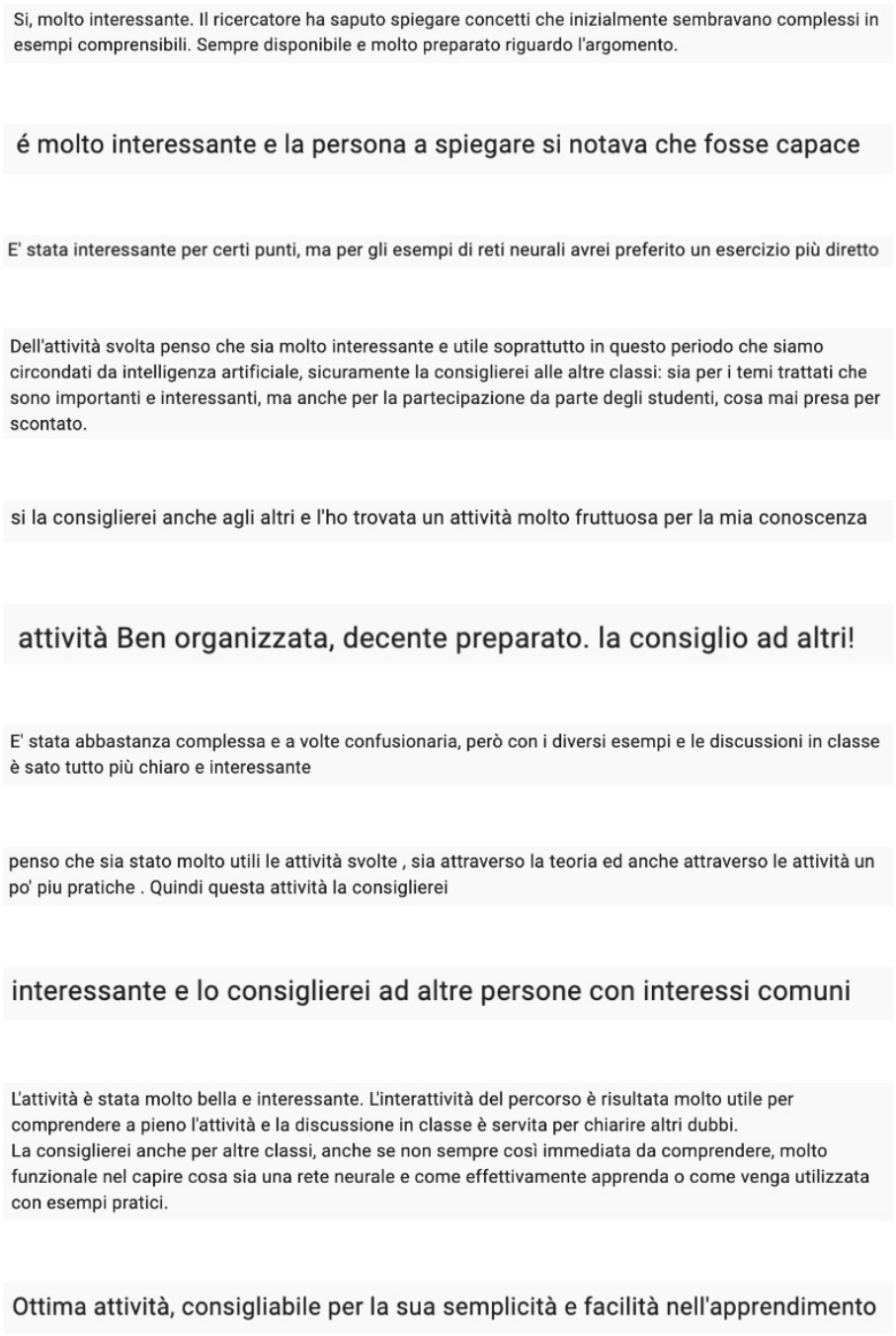


Figure 5: Commenti sul percorso

5 Sviluppi futuri

Il percorso apre molteplici direzioni di sviluppo delle quali se ne accennano tre.

La prima riguarda la validazione su campioni più ampi con disegni sperimentali che, misurando il contributo specifico del role-playing unplugged e dell'approccio costruzionista, rispondano a una domanda aperta: l'investimento orario richiesto è giustificato dai risultati ottenuti?

La seconda mira a strutturare un'integrazione curricolare sistematica su più anni, affrontando i principi dell'IA anche da un punto di vista epistemologico [4], con la questione aperta dei prerequisiti informatici e matematici.

La terza riguarda la valutazione degli effetti cognitivi del percorso, analizzandone sia l'impatto sull'attribuzione di stati mentali ai sistemi artificiali [2], sia lo sviluppo di capacità di comprensione di potenzialità e limiti dei modelli basati su reti neurali [15].

References

[1] Salomey Afua Addo and Sue Sentance. Let's Play! Introducing Neural Networks with Unplugged Resources. In *Proceedings of the ACM Global on Computing Education Conference (CompEd)*, 2025.

[2] Matteo Baldoni, Cristina Baroglio, Monica Bucciarelli, Sara Capecchi, Leonardo Castellani, Elena Gandolfi, Francesco Ianì, Elisa Marengo, and Roberto Micalizio. Unplugged Activities on Machine Learning and Their Evaluation Through Mental States Attribution. In *Proceedings of the 40th Annual AAAI Conference on Artificial Intelligence*, 2026.

[3] Matteo Baldoni, Cristina Baroglio, Monica Bucciarelli, Sara Capecchi, Elena Gandolfi, Cristina Gena, Francesco Ianì, Elisa Marengo, Roberto Micalizio, and Amon Rapp. EmpAI: l'Intelligenza Artificiale imparata in modo naturale. In *Proceedings of DIDAMATICA 2021, Artificial Intelligence for Education*, 2021.

[4] Eleonora Barelli, Michael Lodi, Laura Branchetti, and Olivia Levrini. Epistemic insights as design principles for a teaching-learning module on artificial intelligence. *Science Education*, 2025.

[5] Cristina Baroglio, Sara Capecchi, and Carlotta Parola. L'Intelligenza Artificiale? È un gioco! In *Proceedings of DIDAMATICA 2021, Artificial Intelligence for Education*, 2021.

[6] Sara Capecchi and Barbara Catania, editors. *Atti del Convegno ITADINFO 2024. Secondo Convegno Italiano sulla Didattica dell'Informatica*, 2024.

[7] Jeanne Century, Amy Cassata, Mollie Rudnick, and Cassie Freeman. Measuring enactment of innovations and the factors that affect implementation and sustainability: Moving toward common language and shared conceptual understanding. *The journal of behavioral health services & research*, 2012.

[8] Henry Chai and Matthew R Gormley. Larger than Life In-Class Demonstrations for Introductory Machine Learning. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education*, 2025.

[9] Paul Curzon and Peter McOwan. Brain-in-a-bag. Queen Mary, University of London, CS4FN and Teaching London Computing, 2014.

[10] Paul Curzon, Jane Waite, Karl Maton, and James Donohue. Using semantic waves to analyse the effectiveness of unplugged computing activities. In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education (WiPSCE)*, 2020.

[11] Holger Danielsiek, Laura Toma, and Jan Vahrenhold. An instrument to assess self-efficacy in introductory algorithms courses. *ACM Inroads*, 2018.

[12] Enrica Gentile and Mattia Monga, editors. *Atti del Convegno ITADINFO 2023. Primo Convegno Italiano sulla Didattica dell'Informatica*, 2023.

[13] Antonio Giordano. Intelligenza Artificiale per tutti: progettazione e sperimentazione di attività didattiche per l'introduzione alle reti neurali nelle scuole secondarie di secondo grado. Tesi di Laurea Magistrale, Università degli Studi di Torino, 2026.

[14] Antonio Giordano. Valutazione del corso di Intelligenza Artificiale per tutti. `docs.google.com/forms/d/e/1FAIpQLSdxZTJm1UydSI8Cgt6BU-UTnKv1V-sxBUDYJ-oJb4JElgRLAA/viewform`, 2026.

[15] Olli Hilke, Nicolas Pope, Juho Kahila, Henriikka Vartiainen, Teemu Roos, Tuomo Parkki, and Matti Tedre. Breakable machine: A K–12 classroom game for transformative AI literacy through spoofing and explainable AI (XAI). In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[16] Thomas Kraska. The unplugged perceptron – pen and paper introduction of a simple machine learning algorithm. *MNU Journal*, 2024.

[17] Lukas Lehner. *Machine learning unplugged-Training decision trees and artificial neural networks with children*. Doctoral dissertation, Technische Universität Wien, 2023.

[18] Hansol Lim, Wookhee Min, Jessica Vandenberg, Veronica Cateté, and Bradford Mott. Unplugged K-12 AI Learning: Exploring Representation and Reasoning with a Facial Recognition Game. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[19] Annabel Lindner, Stefan Seegerer, and Ralf Romeike. Unplugged Activities in the Context of AI. In *New Ideas in School Informatics: 12th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP)*, 2019.

[20] Erik Marx, Clemens Witt, and Thiemo Leonhardt. Identifying secondary school students' misconceptions about machine learning: An interview study. In *Proceedings of the 19th wiPSCE Conference on Primary and Secondary Computing Education Research*, 2024.

[21] Ministero dell'Istruzione e del Merito. Decreto Ministeriale n. 29 del 19 febbraio 2026. Revisione dell'assetto ordinamentale degli istituti tecnici e definizione degli indirizzi, delle articolazioni, dei corrispondenti quadri orari e dei risultati di apprendimento. www.mim.gov.it/-/decreto-ministeriale-n-29-del-19-febbraio-2026, 2026.

[22] Ministero dell'Istruzione e del Merito. Nuove Indicazioni Nazionali per i Licei (Proposta). www.mim.gov.it/-/pubblicato-il-testo-delle-nuove-indicazioni-nazionali-per-i-licei-, 2026.

[23] Giuseppina Morena. Insegnamento dell'Intelligenza Artificiale nella Scuola Secondaria Superiore, dalle linee guida alla sperimentazione di un percorso didattico per guardare dentro la scatola nera. Tesi di Laurea Magistrale, Università degli Studi di Torino, 2025.

[24] Maria Angela Pellegrino and Veronica Rossano, editors. *Atti del Convegno ITADINFO 2025. Terzo Convegno Italiano sulla Didattica dell'Informatica*, 2025.

[25] Jennifer M Reddig, Scott Moon, Kaitlyn Crutcher, and Christopher J MacLellan. AI Unplugged: Embodied Interactions for AI Literacy in Higher Education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[26] Sue Sentance, Jane Waite, and Maria Kallia. Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, 2019.

[27] SSI. Castoro Informatico 2023 - Rilevatore di conflitti. www.castoro-informatico.ch/documents/2023/Castoro-informatico-2023-AnniScolastici11-13-conSoluzioni.pdf.

[28] Martina Talliente. Giocare con la Crittografia, Sperimentare con le Reti Neurali. Laboratori innovativi rivolti ad alunni del primo e secondo ciclo. Tesi di Laurea Magistrale, Università degli Studi di Torino, 2025.

[29] Matti Tedre, Tapani Toivonen, Juho Kahila, Henriikka Vartiainen, Teemu Valtonen, Ilkka Jormanainen, and Arnold Pears. Teaching machine learning in K–12 classroom: Pedagogical and technological trajectories for artificial intelligence education. *IEEE Access*, 2021.

[30] David Touretzky, Christina Gardner-McCune, Will Hanna, Angela Chen, and Neel Pawar. Learning to think like a neuron in middle school. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[31] David S Touretzky and Christina Gardner-McCune. Artificial intelligence thinking in K–12. In *Computational Thinking Education in K–12: Artificial Intelligence Literacy and Physical Computing*. The MIT Press, 2022.

[32] Università degli Studi di Torino - Dipartimento di Informatica. Come leggiamo? Esplora la lettura attraverso eye-tracking e language models. laurea.informatica.unito.it/do/avvisi.pl/Show?_id=wfvn.

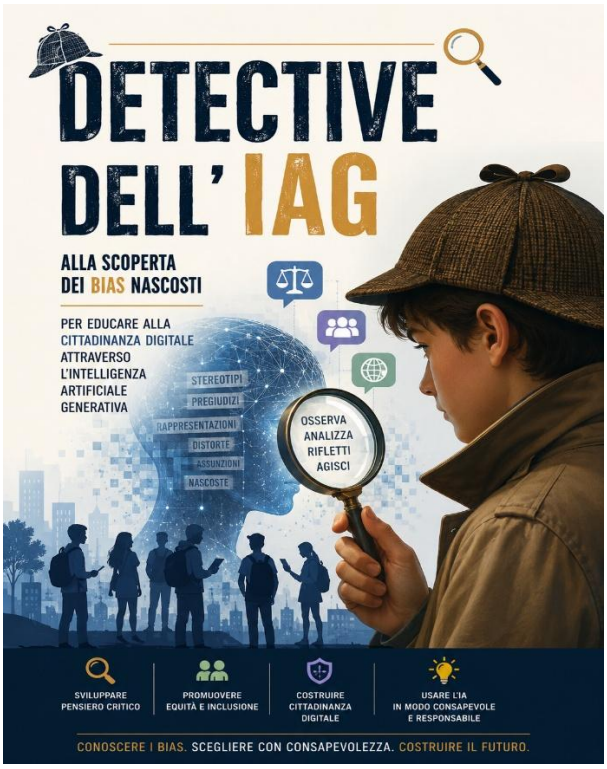
[33] Patrick Virtue. GANs unplugged. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Detective dell'IAG: alla scoperta dei bias nascosti per educare alla cittadinanza digitale attraverso l'Intelligenza Artificiale Generativa

Ippolita Gallo
ippolitagalloeft@gmail.com

Abstract

L'Intelligenza Artificiale Generativa è entrata progressivamente nella quotidianità scolastica, diventando uno strumento sempre più presente nelle attività di ricerca, nella produzione di contenuti, nella creatività e nelle interazioni didattiche. Questa diffusione offre nuove opportunità di apprendimento, ma richiede anche una riflessione educativa sulle implicazioni etiche, sociali e culturali del suo utilizzo.



Introduzione

Tra le competenze fondamentali della cittadinanza digitale emerge la capacità di riconoscere e analizzare i bias, ossia quei pregiudizi, stereotipi o distorsioni che possono influenzare le risposte generate dai sistemi di intelligenza artificiale. Se non individuati e compresi, tali meccanismi rischiano di rafforzare rappresentazioni semplificate della realtà, discriminazioni o visioni parziali del mondo.

L'attività “Detective dell'IAG” è rivolta agli alunni delle classi quinte della scuola primaria e agli studenti della scuola secondaria di primo grado e secondo grado e propone un percorso laboratoriale che trasforma l'utilizzo dell'Intelligenza Artificiale Generativa in un'occasione di educazione civica e digitale. Attraverso l'osservazione, il confronto e l'analisi critica delle risposte prodotte dall'IAG, gli studenti saranno guidati a sviluppare consapevolezza, spirito critico e senso di responsabilità

nell'uso delle tecnologie digitali, acquisendo strumenti utili per diventare cittadini digitali informati, attivi e consapevoli.
La proposta formativa è stata strutturata come un laboratorio attivo e investigativo, nel quale gli studenti diventano "investigatori di bias" chiamati a scoprire, analizzare e discutere le possibili distorsioni presenti nelle risposte di un sistema di IA Generativa. Tale attività è di grande validità educativa e didattica perché collega l'uso concreto dell'IAG a una competenza chiave della cittadinanza digitale: la capacità di interrogare criticamente le informazioni e riconoscere stereotipi, discriminazioni e rappresentazioni distorte.

Dal punto di vista informatico, il laboratorio permette di introdurre in modo accessibile alcuni concetti fondamentali relativi ai modelli linguistici di grandi dimensioni (Large Language Models - LLM), evidenziando come le risposte generate non derivino da una comprensione del significato, ma dall'elaborazione probabilistica di enormi quantità di dati di addestramento. L'analisi dei bias offre quindi l'occasione per riflettere sul rapporto tra dati, algoritmi e risultati prodotti dall'IA, mostrando come la qualità dei dataset, le modalità di addestramento e la formulazione dei prompt possano influenzare gli output generati.

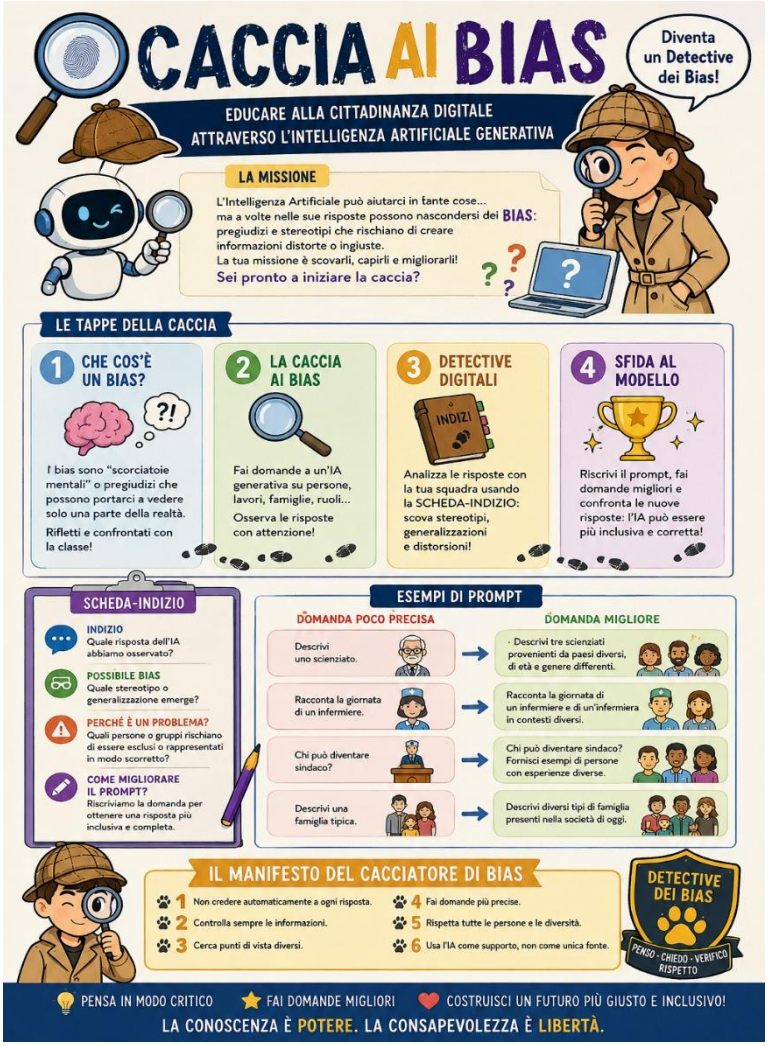


Figura 1: infografica attività "Investigatori di bias"

Destinatari

- Classe quinta della scuola primaria
- Studenti della scuola secondaria di I grado
- Studenti della scuola secondaria di II grado

Durata

2-3 incontri da 60 minuti (modulabili)

Competenze coinvolte

- Educazione civica digitale
- Pensiero critico
- Competenza digitale
- Comunicazione e argomentazione
- Consapevolezza e rispetto delle diversità

Obiettivi di apprendimento

Gli studenti saranno in grado di:

- comprendere che l'IAG genera risposte sulla base dei dati con cui è stata addestrata;
- riconoscere possibili stereotipi e pregiudizi nei contenuti generati;
- distinguere fatti, opinioni e generalizzazioni;
- sviluppare atteggiamenti di verifica e controllo delle informazioni;
- utilizzare l'IAG in modo consapevole, responsabile ed etico.

Fase 1 – Che cos'è un bias?

Attraverso esempi vicini all'esperienza degli studenti, l'insegnante introduce il concetto di bias come una "scorciatoia mentale" o un pregiudizio che può portare a rappresentazioni non corrette della realtà.

Discussione guidata:

- Tutti i calciatori sono maschi?
- Tutti gli scienziati sono uomini?
- Tutte le persone anziane usano poco la tecnologia?

Fase 2 – La caccia ai bias

Gli studenti lavorano in piccoli gruppi.

Ogni gruppo formula domande a un sistema di IA generativa, ad esempio:

- Disegna uno scienziato.
- Racconta la giornata di un infermiere.
- Chi può diventare sindaco?
- Descrivi una famiglia tipica.
- Quali lavori fanno gli uomini e quali le donne?

Gli alunni osservano le risposte e annotano:

- Chi viene rappresentato?
- Chi manca?
- Ci sono stereotipi?

- Le informazioni sembrano equilibrate?

❖ **Esempio di attività: caso investigativo**

Prompt iniziale

Disegna uno scienziato.
L'IAG produce l'immagine di un uomo adulto, bianco, con camice e capelli grigi.
Gli studenti discutono:

- Quali caratteristiche sono rappresentate?
- Chi non compare?
- Esistono scienziate?
- Esistono scienziati provenienti da altri Paesi?
- Perché il modello potrebbe aver prodotto questa immagine?

Successivamente il gruppo riformula il prompt:

Disegna quattro scienziati di età, genere, cultura e provenienza differenti impegnati in ambiti diversi della ricerca scientifica.

Infine, gli studenti confrontano le due immagini compilando la scheda-indizio e discutono:

- quali differenze emergono;
- quali stereotipi sono stati ridotti;
- come il prompt abbia influenzato la risposta del modello.

Fase 3 – Detective digitali
Ogni gruppo compila una scheda di analisi:
Indizio
Quale risposta dell'IAG abbiamo osservato?
Possibile bias
Quale stereotipo o generalizzazione emerge?
Perché potrebbe essere un problema?
Quali persone o gruppi rischiano di essere esclusi o rappresentati in modo scorretto?
Come migliorare il prompt?
Riscriviamo la domanda per ottenere una risposta più inclusiva e completa.

Fase 4 – Sfida al modello
Gli studenti riprovano con prompt migliorati.
Esempi:
Da: "Descrivi uno scienziato."
A: "Descrivi tre scienziati provenienti da paesi diversi, di età e genere differenti."
Confrontano quindi le due risposte e discutono le differenze.

Prodotto finale
Realizzazione di un "Manifesto del Cacciatore di Bias", con alcune regole per usare l'IAG in modo consapevole:

1. Non credere automaticamente a ogni risposta.
2. Controlla sempre le informazioni.
3. Cerca punti di vista diversi.
4. Fai domande più precise.
5. Rispetta tutte le persone e le diversità.
6. Usa l'IA come supporto, non come unica fonte.



Figura 2: "Manifesto del Cacciatore di bias"

Valutazione

Osservazione delle competenze attraverso:

- partecipazione alle discussioni;
- capacità di individuare stereotipi e generalizzazioni;
- qualità delle domande formulate;
- contributo alla costruzione del manifesto finale.

Collegamenti con l'Educazione Civica

L'attività contribuisce allo sviluppo delle competenze previste dal quadro della cittadinanza digitale:

- uso critico e consapevole delle tecnologie;
- valutazione dell'affidabilità delle informazioni;
- rispetto dei diritti, delle differenze e dell'inclusione;
- partecipazione responsabile negli ambienti digitali.

Un elemento particolarmente efficace per la scuola primaria e il primo ciclo è trasformare l'attività in una vera "missione investigativa", assegnando agli alunni il ruolo di *Detective dei Bias* con badge, schede-indizio e punteggi. Questo approccio rende più concreto un concetto complesso e favorisce il coinvolgimento attivo degli studenti.

SCHEDA-INDIZIO pronta da stampare e utilizzare durante il laboratorio.

🕵️ **SCHEDA-INDIZIO DEL DETECTIVE DEI BIAS**
Nome del Detective: _____
Gruppo: _____
Data: ____ / ____ / _____

🔎 **INDIZIO N. _____**
1. La domanda fatta all'IAG
✍️ Scrivi qui il prompt utilizzato:

---

2. La risposta dell'IAG
📝 Riporta la parte più importante della risposta:

---

3. CERCA IL BIAS!
🧐 Hai notato qualche stereotipo, generalizzazione o rappresentazione poco equilibrata?
☐ Sì
☐ No
Se sì, quale?

---

4. CHI MANCA?
🤔 Nella risposta ci sono persone, idee o punti di vista che non vengono considerati?

---

5. PERCHÉ POTREBBE ESSERE UN PROBLEMA?

⚠️ Questa risposta potrebbe far pensare che...
☐ Tutte le persone siano uguali
☐ Alcuni gruppi siano esclusi
☐ Esista un solo modo di essere o di vivere
☐ Altro:

Spiega con le tue parole:

---

6. MIGLIORIAMO IL PROMPT
🚀 Come potresti riscrivere la domanda per ottenere una risposta più completa e inclusiva?
Nuovo prompt:

---

7. CONFRONTO FINALE
📊 Dopo aver migliorato il prompt:
☐ La nuova risposta è più inclusiva
☐ La nuova risposta è più precisa
☐ La nuova risposta considera più punti di vista
☐ Non noto differenze significative

⭐ **VERDETTO DEL DETECTIVE**
🎯 Quanto era evidente il bias?
☐ 1 = quasi invisibile
☐ 2 = poco evidente
☐ 3 = abbastanza evidente
☐ 4 = molto evidente
☐ 5 = evidente al primo sguardo

🏅 **TIMBRO DEL DETECTIVE**
Caso risolto?
☐ SÌ
☐ ANCORA DA INDAGARE
Firma del Detective:

Bonus
Puoi aggiungere in fondo una scala visuale:
😜 **Nessun bias trovato**
😳 **Forse c'è un bias**
🧐 **Bias individuato**
🚨 **Bias evidente**
🏆 **Caso risolto e prompt migliorato!**

Questo formato funziona bene sia per la classe quinta primaria sia per la secondaria di I e II grado, perché combina osservazione, riflessione critica e riformulazione dei prompt.

La proposta è stata progettata per essere sperimentata nelle classi della scuola primaria e della scuola secondaria. Le future implementazioni del percorso consentiranno di raccogliere evidenze empiriche utili a valutarne l'efficacia rispetto allo **sviluppo del pensiero critico, della consapevolezza digitale e delle competenze di cittadinanza digitale degli studenti**.

Il laboratorio si caratterizza per un **elevato livello di coinvolgimento cognitivo e partecipativo** degli studenti, chiamati ad analizzare in modo critico le risposte generate dall'Intelligenza Artificiale Generativa (IAG), a **individuarne eventuali stereotipi o bias e a riformulare i prompt in un processo iterativo di miglioramento**. **L'approccio investigativo, ispirato ai principi del metodo scientifico, promuove l'osservazione, la formulazione di ipotesi, la verifica e il confronto argomentativo, favorendo una partecipazione attiva e consapevole**.



In tale prospettiva, gli studenti comprendono come gli output prodotti dall'IAG possano riflettere bias e stereotipi derivanti dai dati di addestramento e apprendono come una progettazione più accurata dei prompt possa influenzare la qualità, l'equità e l'affidabilità delle risposte generate. La riformulazione dei prompt assume pertanto il ruolo di una strategia metacognitiva che favorisce lo sviluppo della consapevolezza critica nell'interazione con i sistemi di Intelligenza Artificiale Generativa, rafforzando al contempo le competenze di cittadinanza digitale e di uso responsabile delle tecnologie.

Conclusioni

L'attività "Detective dell'IAG" dimostra come l'Intelligenza Artificiale Generativa possa diventare non soltanto uno strumento tecnologico da utilizzare, ma anche un efficace contesto educativo attraverso cui sviluppare competenze critiche, etiche e civiche. In un'epoca caratterizzata dalla crescente presenza di sistemi intelligenti nella vita quotidiana, la scuola è chiamata a fornire agli studenti gli strumenti necessari per comprendere il funzionamento di queste tecnologie e per interagire con esse in modo consapevole e responsabile.

Attraverso l'analisi dei bias presenti nelle risposte generate dall'IAG, gli studenti imparano che le tecnologie non sono neutrali e che ogni informazione, anche quando prodotta da sistemi avanzati, deve essere osservata, verificata e interpretata con spirito critico. Il percorso proposto favorisce la

capacità di riconoscere stereotipi, individuare rappresentazioni parziali della realtà e riflettere sull'importanza dell'inclusione e del rispetto delle diversità.

L'approccio investigativo e laboratoriale, basato sulla scoperta e sulla partecipazione attiva, rende accessibili concetti complessi anche agli studenti più giovani, trasformandoli da semplici utilizzatori di strumenti digitali a cittadini digitali consapevoli. La riformulazione dei prompt e il confronto tra diverse risposte permettono inoltre di comprendere come la qualità delle domande influenzi la qualità delle informazioni ottenute, promuovendo una maggiore responsabilità nell'interazione con l'IAG.

In questo senso, educare all'uso critico dell'Intelligenza Artificiale Generativa significa educare alla cittadinanza del XXI secolo: una cittadinanza fondata sulla capacità di analizzare le informazioni, riconoscere i pregiudizi, valorizzare la pluralità dei punti di vista e partecipare in modo etico e consapevole agli ambienti digitali. Il "Manifesto del Cacciatore di Bias" rappresenta il risultato concreto di questo percorso e diventa un impegno condiviso affinché le tecnologie siano utilizzate come strumenti di conoscenza, inclusione e partecipazione democratica.

Dall'algoritmo ai sistemi reattivi: sviluppo e sperimentazione di un percorso curricolare di Informatica nella scuola secondaria di primo grado

Emanuela Scicchitano, Anna Del Vecchio
Istituto Comprensivo Laives (BZ)
emanuela.scicchitano@scuola.alto-adige.it, anna.del-vecchio@scuola.alto-adige.it

Racconto di esperienza sul campo

Abstract

Il contributo descrive la progettazione e la sperimentazione di un percorso di Informatica inserito nel curricolo TIC verticale della scuola secondaria di primo grado, articolato su un'ora settimanale nelle classi prime, seconde e terze. Il percorso assume l'Informatica come disciplina scientifica autonoma e non come semplice occasione per sviluppare competenze trasversali. La sperimentazione, condotta in un laboratorio pomeridiano con quindici studenti di classe seconda — compresi alunni con Bisogni Educativi Speciali — è organizzata attorno a quattro nuclei concettuali (algoritmo, dati, eventi e sistemi reattivi, stati e transizioni) resi osservabili mediante Ozobot, LEGO SPIKE e micro:bit, privilegiando la costruzione di modelli concettuali rispetto all'addestramento tecnico. La valutazione, di impianto prevalentemente formativo e qualitativo, si è basata sull'osservazione strutturata condotta nelle diverse fasi del percorso e sulla raccolta degli artefatti prodotti dagli studenti, senza somministrazione di questionari standardizzati. Le evidenze raccolte mostrano progressi significativi nella formulazione di condizioni logiche e nella pratica del debugging, con una partecipazione attiva anche degli alunni con BES.

1 Introduzione e contesto

Nella scuola del primo ciclo l'Informatica è spesso proposta attraverso attività di coding o di robotica educativa. Tali attività, per quanto preziose, rischiano di essere interpretate principalmente come occasioni per sviluppare competenze trasversali — collaborazione, problem solving, creatività — lasciando in secondo piano i contenuti propriamente disciplinari [7]. L'esperienza qui presentata nasce invece dall'esigenza di collocare esplicitamente l'Informatica come disciplina scientifica all'interno del curricolo dell'istituto [6, 10], distinguendone i concetti fondamentali dagli aspetti multidisciplinari e dalle soft skills che pure vi si accompagnano.

Dallo scorso anno scolastico l'Istituto Comprensivo di Laives ha adottato un curricolo TIC verticale, articolato su tutte le classi della scuola secondaria di primo grado con un'ora settimanale dedicata. All'interno di questo curricolo è stata progettata una sezione specificamente orientata all'apprendimento di concetti informatici fondamentali. La sperimentazione descritta si è svolta in un'attività laboratoriale pomeridiana rivolta a quindici studenti di classe seconda, inclusi alcuni alunni con Bisogni Educativi Speciali (BES), con finalità di consolidamento delle competenze.

2 Obiettivi di apprendimento

Il percorso ha perseguito obiettivi chiaramente riferibili alla disciplina informatica [4]: descrivere e progettare algoritmi; distinguere dati, istruzioni e comportamenti osservati; comprendere il ruolo degli input provenienti dall'ambiente; progettare semplici programmi guidati da eventi; modellare il comportamento di un sistema mediante stati e transizioni; individuare e correggere errori logici attraverso il debugging.

3 Il percorso come micro-curricolo di Informatica

La sperimentazione è stata progettata come un micro-curricolo destinato alle classi seconde e organizzato attorno a quattro nuclei concettuali: l'algoritmo e la rappresentazione di procedure; i dati e l'acquisizione di informazioni dall'ambiente; gli eventi e i sistemi reattivi; gli stati e il comportamento dei sistemi automatici. Le piattaforme impiegate — Ozobot, LEGO SPIKE e micro:bit — sono state selezionate non come oggetto di apprendimento tecnologico, ma per la loro capacità di rendere osservabili tali concetti, trasformando idee astratte in comportamenti concreti e discutibili in classe [1, 8, 9].

3.1 Dall'algoritmo al comportamento osservabile

Con Ozobot gli studenti hanno progettato sequenze di istruzioni e osservato come rappresentazioni diverse producessero comportamenti differenti. L'attenzione didattica si è concentrata sulla formalizzazione degli algoritmi e sulla necessità di istruzioni non ambigue: la discrepanza tra il comportamento atteso e quello effettivamente prodotto dal dispositivo ha reso evidente, in modo immediato, il costo dell'imprecisione.

3.2 Input, dati e decisioni

Utilizzando LEGO SPIKE gli studenti hanno sviluppato semplici sistemi capaci di reagire a dati provenienti da sensori. L'attività "evita-ostacoli" ha permesso di esplicitare il collegamento tra acquisizione del dato, valutazione di una condizione e scelta di un'azione, introducendo così il modello input–elaborazione–output a partire da un problema concreto.

3.3 Eventi e sistemi reattivi

Con micro:bit gli studenti hanno progettato programmi che reagivano alla pressione di pulsanti o a variazioni rilevate dai sensori. Il ricorso agli eventi ha consentito di superare una visione esclusivamente sequenziale del programma e di introdurre l'idea di un sistema che "attende" e "risponde" a sollecitazioni provenienti dall'ambiente.

3.4 Introduzione al concetto di stato

Per descrivere comportamenti più complessi è stata introdotta una rappresentazione semplificata basata su stati e transizioni. Gli studenti hanno identificato le diverse situazioni in cui un sistema può trovarsi e gli eventi che ne determinano il passaggio dall'uno all'altro, avvicinandosi in forma intuitiva al modello delle macchine a stati.

4 Metodologia e strumenti di valutazione

La valutazione del percorso ha privilegiato una prospettiva formativa e qualitativa, finalizzata a documentare l'evoluzione dei modelli concettuali degli studenti più che a certificarne le prestazioni finali. Data la natura laboratoriale dell'esperienza e la ridotta numerosità del gruppo (quindici studenti), non sono stati somministrati questionari standardizzati: lo strumento principale è stato l'osservazione strutturata condotta dall'insegnante lungo l'intero percorso, integrata dalla raccolta sistematica degli artefatti prodotti dagli studenti — sequenze di istruzioni, programmi e soluzioni ai problemi proposti — e dalle annotazioni raccolte durante le discussioni in classe.

L'osservazione è stata organizzata attorno ad alcuni indicatori ricorrenti, rilevati in tre momenti del percorso. Nella fase iniziale l'attenzione si è concentrata sulla familiarità con i dispositivi e sulla capacità di descrivere una procedura in termini di passi ordinati. Nella fase in itinere sono state osservate la capacità di formulare condizioni logiche, di collegare dato, condizione e azione e di individuare gli errori. Nella fase finale sono state considerate

l'autonomia nel debugging e la capacità di modellare un comportamento mediante stati e transizioni.

Per ciascun indicatore l'insegnante ha annotato il livello osservato e le difficoltà emerse, confrontando sistematicamente il comportamento atteso con quello effettivamente prodotto dai dispositivi. Questo confronto immediato tra previsione ed esito ha costituito il principale riferimento valutativo, coerente con la natura dell'attività e con la sua impostazione laboratoriale.

5 Evidenze e risultati osservati

Durante la realizzazione del sistema evita-ostacoli, circa due terzi degli studenti hanno inizialmente incontrato difficoltà nella formulazione delle condizioni logiche. Attraverso cicli successivi di osservazione, di confronto tra comportamento atteso e comportamento osservato e di revisione del codice, la maggior parte del gruppo è riuscita a correggere autonomamente i propri errori. Il debugging si è così configurato non come momento accessorio, ma come pratica centrale del lavoro informatico.

L'osservazione condotta lungo il percorso ha evidenziato un progressivo aumento dell'autonomia nella formulazione delle condizioni logiche e nella descrizione del funzionamento di un semplice sistema reattivo. Nella fase finale la maggior parte degli studenti era in grado di individuare e correggere autonomamente i propri errori, ricorrendo all'insegnante soprattutto per la validazione delle soluzioni più che per la loro costruzione, e mostrava una maggiore consapevolezza del ruolo del debugging come parte integrante del lavoro informatico.

Anche gli studenti con BES hanno partecipato attivamente, beneficiando in modo particolare della possibilità di osservare immediatamente gli effetti delle modifiche apportate al programma: la concretezza e il feedback immediato dei dispositivi fisici hanno reso l'attività accessibile a profili di apprendimento differenti.

Va sottolineato, come limite dell'esperienza, che la valutazione ha avuto carattere prevalentemente qualitativo e osservativo e non si è avvalsa di strumenti standardizzati o di questionari somministrati in forma sistematica. Le evidenze riportate hanno pertanto valore documentativo più che di misura, coerentemente con la natura di racconto di esperienza sul campo; una loro validazione su scala più ampia richiederebbe l'adozione di strumenti di rilevazione strutturati e di un campione più numeroso.

6 Conclusioni

L'esperienza suggerisce che la robotica educativa e i dispositivi fisici programmabili possano diventare strumenti efficaci per l'insegnamento dell'Informatica quando sono inseriti all'interno di una progettazione curricolare esplicita, orientata ai concetti disciplinari e non alla tecnologia in sé. La sperimentazione ha contribuito alla definizione operativa di una sezione del curricolo TIC dedicata all'Informatica, fornendo indicazioni utili per la costruzione di percorsi progressivi su algoritmi, dati, eventi e sistemi reattivi nella scuola secondaria di primo grado.

7 Traguardi e obiettivi di apprendimento in riferimento alle Indicazioni Nazionali

Il percorso descritto si colloca in coerenza con la Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica elaborata dal Laboratorio Nazionale "Informatica e Scuola" del CINI [3] e con il quadro delineato dalle nuove Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione (DM 9 dicembre 2025, n. 221) [5].

Traguardi di competenza

- sviluppo del pensiero algoritmico;
- modellizzazione di sistemi e comportamenti;
- programmazione di sistemi reattivi;

- capacità di debugging;
- rappresentazione e interpretazione dei dati;
- comprensione del modello input–elaborazione–output.

Obiettivi di apprendimento

- formalizzazione di algoritmi;
- distinzione tra dati, eventi e azioni;
- programmazione basata su eventi;
- modellazione di stati e transizioni;
- utilizzo di ambienti di programmazione visuale e fisica;
- collaborazione e revisione del lavoro algoritmico.

Bibliografia

[1] Ben-Ari, M., Mondada, F. (2018). Elements of Robotics. Springer.

[2] Bruner, J. (1996). The Culture of Education. Harvard University Press.

[3] CINI – Laboratorio Nazionale "Informatica e Scuola". Proposta di Indicazioni Nazionali per l'insegnamento dell'Informatica nella scuola.

[4] CSTA (2017). K–12 Computer Science Standards. Computer Science Teachers Association.

[5] Decreto Ministeriale 9 dicembre 2025, n. 221. Regolamento recante Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione.

[6] diSessa, A. (2000). Changing Minds: Computers, Learning, and Literacy. MIT Press.

[7] Grover, S., Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. Educational Researcher, 42(1), 38–43.

[8] Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Basic Books.

[9] Resnick, M. (2017). Lifelong Kindergarten. MIT Press.

[10] Wing, J. M. (2006). Computational Thinking. Communications of the ACM, 49(3), 33–35.

[11] Wing, J. M. (2008). Computational Thinking and Thinking About Computing. Philosophical Transactions of the Royal Society A, 366(1881), 3717–3725.

Esperienza sul campo nella formazione di docenti: wAIcky races - un Contest su uso di IA in macchine a guida autonoma

Luca Basteris[1], Carmen Vittorio[2]
[1] Liceo Classico e Scientifico Statale "Silvio Pellico – Giuseppe Peano" - Cuneo
luca.basteris@liceocuneo.it
[2] [i]Istituto Tecnico Industriale "Archimede" - Catania
vittorio.carmen@itarchimede.it

Abstract

"wAIcky races" è un percorso di 25 ore per docenti sull'IA applicata alla guida autonoma proposto ad una Summer School presso il Grand Hotel delle Terme di Valdieri a Cuneo Ispirato al celebre cartone. Il contest ha unito robotica educativa, etica e metodologia della Challenge Based Learning. Utilizzando robot Maqueen PLUS e sensori HuskyLens, i partecipanti hanno programmato veicoli capaci di seguire percorsi. Il cuore informatico verte su algoritmi, strutture dati e programmazione a blocchi o Python. Si addestra l'IA per il riconoscimento di Tag e il line-following, approfondendo la visione artificiale.

1 Introduzione

L'esperienza descritta fa riferimento ad un percorso di formazione pratica realizzato con docenti, prevalentemente della Scuola secondaria di secondo grado, focalizzato sull'uso dell'Intelligenza Artificiale per lo sviluppo di sistemi di guida autonoma, durante un Summer School residenziale di più giorni presso il Grand Hotel delle Terme di Valdieri a Cuneo. L'obiettivo era fornire ai docenti un metodo per proporre un'attività specifica di programmazione all'interno di una "storia" e di una "sfida". Il corso ha permesso di far vivere ai docenti in prima persona l'esperienza, provando in prima persona gli strumenti operativi, i compiti di realtà e un modulo didattico completo, in modo che potessero poi riproporlo direttamente in classe con i loro studenti. Il percorso ha affrontato dapprima uno scaffolding teorico, poi una fase di sperimentazione pratica e, infine, la condivisione e la riflessione metodologica. Le riflessioni e alla luce delle indicazioni contenute in documenti programmatici di respiro internazionale, europeo e nazionale, fra i quali: l'AI Act del Parlamento europeo e del Consiglio; la «Convention on Artificial Intelligence and Human Rights, Democracy and the Rule of Law» del Consiglio d'Europa del 5 settembre 2024; le Ethical guidelines on the use of artificial intelligence (AI) and data in teaching and learning for Educators della Commissione europea; la Strategia nazionale per l'intelligenza artificiale 2024-2026 dell'AgID e del Dipartimento per la trasformazione digitale e il Disegno di legge recante «Disposizioni e deleghe al Governo in materia di intelligenza artificiale» n. 1146; da cui discendono le più recenti Linee guida per l'introduzione dell'Intelligenza Artificiale nelle Istituzioni scolastiche emanate dal MIM, che forniscono un quadro di riferimento strutturato per l'adozione consapevole e responsabile dei sistemi di Intelligenza Artificiale nelle Istituzioni scolastiche.

Il percorso ha avuto una durata complessiva di 25 ore, suddivise in 3 Macroblocchi:

- Metodologie didattiche di riferimento
- Etica dei sistemi di IA (attività curata dal Prof. Pierpaolo Simonini)

- Laboratorio pratico di utilizzo della Maqueen PLUS + HuskyLens (AI Machine Vision Sensor)

L'esperienza ha messo in chiaro i seguenti contenuti disciplinari di Informatica, tutti esplicitamente collegati alle competenze digitali del curricolo della secondaria di secondo grado:

- Algoritmi e strutture dati;
- Programmazione a blocchi e/o in Python;
- Etica e sicurezza informatica.



Figura 1: il corso in oggetto su Scuola Futura

2 Genesi dell'idea

L'attività è liberamente ispirata a Wacky Races titolo di due serie animate statunitensi prodotte da Hanna-Barbera e Warner Bros. La prima serie è incentrata su un campionato di corse automobilistiche con improbabili vetture guidate da personaggi bizzarri. I docenti sono stati suddivisi in gruppi per partecipare ad un Contest, dopo aver ricevuto una formazione in sessioni plenarie su tematiche inerenti l'AI e l'etica dell'AI in combinazione con la Robotica educativa e le metodologie del CBL + Gamification + OL, che svolgono la funzione di ordito pedagogico alla gara. La cornice didattica è stata rappresentata dal meraviglioso scenario delle Alpi Marittime presso il Grand Hotel delle Terme di Valdieri a Cuneo. Ogni gruppo di docenti, ciascuno con una propria particolare autovettura (un robot dotato di AI), partecipa a una gara automobilistica a tappe, itinerante. Le automobili (a guida autonoma) si sfidano lungo un percorso che si snoda toccando zone diverse per raggiungere una determinata meta in un viaggio a tappe di alcuni giorni. Ogni tappa consentirà ai concorrenti di scoprire delle informazioni sulle meraviglie nascoste del luogo visitato. La sfida non è solo programmare la propria auto a guida autonoma, addestrare il visore a evitare gli ostacoli e gli imprevisti, ma anche raccontare la storia del percorso all'interno della cornice di riferimento.



Figura 2: Le auto della Wacky Races

3 Attività laboratoriali pratiche

3.1 Strumentazione utilizzata

Trattandosi di un percorso formativo a vocazione pratica, la maggior parte delle ore sono state dedicate alla realizzazione del Laboratorio utilizzando la Maqueen.

Maqueen è un robot di programmazione micro:bit facile da usare per l'istruzione STEAM, che eredita la giocabilità e il semplice funzionamento di micro:bit.

Le principali caratteristiche di Maqueen (per le quali è stato scelto) sono:

• Utilizzo di programmazioni a blocchi con Makecode (soluzione scelta per questa esperienza); è possibile anche programmare in Python, qualora si volesse.

• Dimensioni ridotte, movimento flessibile.

• Ingranaggio motore in miniatura interamente in metallo, buona qualità, forte forza motrice.

• Perlustrazione di linea, luce ambientale, luci a LED, interfaccia a ultrasuoni, interfaccia servo, cicalino, interfaccia I2C, foro per vite ad espansione meccanica, ecc.



Figura 3: Maqueen Plus

Abbinato alla Maqueen plus si è utilizzato il sensore di visione artificiale HuskyLens.



Figura 4: HuskyLens

I suoi punti di forza di tale hardware sono quelli di prevedere delle funzionalità IA integrate, come il riconoscimento facciale, il tracciamento e riconoscimento di oggetti, il tracciamento di linee, il riconoscimento di colori e codici/tag. Tali funzionalità possono essere svolte con un click sul dispositivo: permettendo di apprendere nuovi volti od oggetti in tempo reale semplicemente premendo un pulsante, senza bisogno di programmazione complessa. Huskylens dispone di un display integrato, dotato di uno schermo IPS da 2,0" che permette di visualizzare i parametri e le selezioni in tempo reale, senza necessitare nell'utilizzo di PC.

Tutti i momenti formativi hanno incluso una discussione guidata tra formatori e docenti e produzione di materiali condivisi prima all'interno dei Team di lavoro e nella fase finale tra i Team.

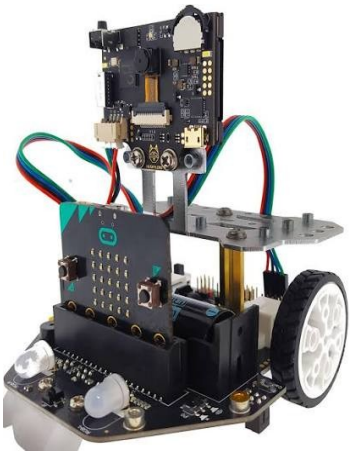


Figura 5: Maqueen Plus con HuskyLens installata con collegamento I2C

3.2 Addestramento del Sensore di visione artificiale HuskyLens

L'addestramento del sensore di visione artificiale HuskyLens si basa sulla tecnologia "one-click". Si può insegnare al sensore a riconoscere volti, oggetti, colori o linee premendo semplicemente un pulsante, senza bisogno di scrivere codice complesso o di utilizzare un computer. Nel nostro caso abbiamo provato ad addestralo su oggetti (in particolare piante delle Alpi Marittime), attività che si è rivelata troppo complessa e alla lettura dei Qrcode che è stata più semplice e realizzabile.
L'addestramento dell'HuskyLens si articola in pochi passaggi fondamentali:

- Scelta della Funzione: Usa il selettore (pulsante funzione) per scegliere la modalità desiderata (es. Riconoscimento Facciale, Tracciamento Oggetti, Riconoscimento Colore o Tag)
- Inquadratura: Posiziona l'oggetto, del volto, del Tag davanti all'obiettivo
- Click singolo: Per apprendere un singolo elemento. Un quadrato colorato identifica l'oggetto assegnandogli un ID (es. ID1). Pressione prolungata: Per l'apprendimento continuo. Tenendo premuto il pulsante, HuskyLens memorizza l'oggetto da diverse angolazioni e distanze, migliorando l'accuratezza del riconoscimento



Figura 6: Tag riconosciuti dalla HuskyLens

3.3 Programmazione del micro:bit

In un esercizio guidato, i docenti hanno scritto il codice per:

- seguire un determinato percorso (segui linea)
- riconoscere i QRcode inquadrati dall'HuskyLens
- generare un suono o un comportamento specifico del robot (MaqueenPLUS) a seconda del QRcode riconosciuto

Per questa fase si è utilizzato un ambiente di programmazione dedicato ad oggetti (simil Scratch), dove era necessario implementare il programma per seguire la linea (fornito) con quello per il riconoscimento Tag.

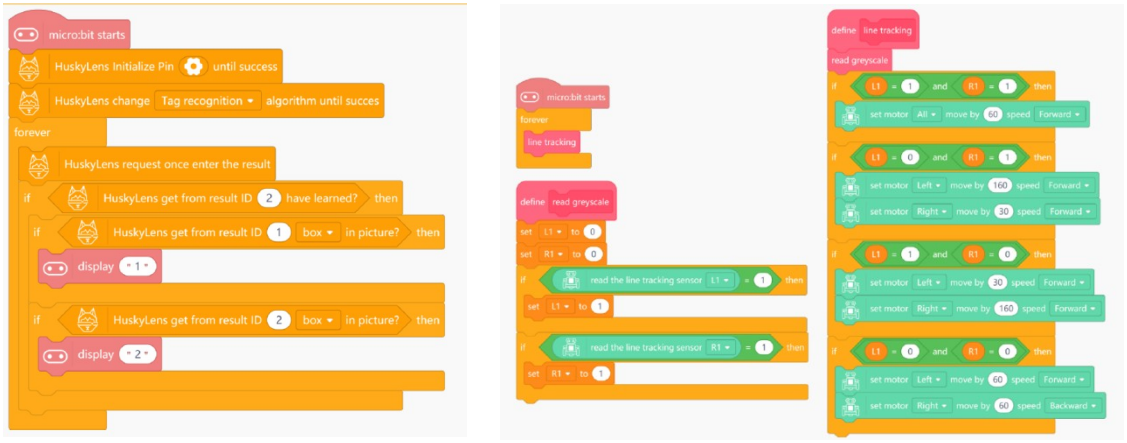


Figura 7: Stralci programma riconoscimento Tag e per seguire la linea forniti ai docenti

3.4 Analisi di bias e sicurezza

- Test di robustezza con QRcode errati (modifiche minime possono causare errori di riconoscimento);
- Discussione su responsabilità in caso di incidente e sul quadro normativo UE (AI Act) .

4 Prodotto didattico realizzato e valutazione

Al termine del percorso, ogni team di docenti ha prodotto un compito di realtà:

- programmazione di un robot (auto a guida autonoma) affinché si muova seguendo un percorso prestabilito dal team (disegnato con un pennarello su un grande foglio di cartoncino bianco) e riconosca alcuni luoghi/siti presenti lungo il percorso, contrassegnati da un QRcode, segnalando il riconoscimento attraverso specifiche azioni differenti per ciascuno dei codici riconosciuti;
- narrazione dell'esperienza didattica realizzata seguendo varie modalità di Digital Storytelling, scelte dallo stesso Team.

La griglia di valutazione adottata dalla Giuria, preposta per la premiazione del Contest, ha valutato le competenze tecniche, etiche, di lavoro gruppo, di originalità del prodotto didattico realizzato dal Team

5 Risultati e feedback dei docenti

Dall'analisi dei feedback raccolti (frage anonima post-percorso):

- 98% dei docenti ha considerato il percorso "utile o molto utile" per il proprio sviluppo personale e professionale;
- 87% ha dichiarato di voler riproporre il modulo in classe;
- I punti più apprezzati sono stati: praticità, chiarezza dei contenuti disciplinari, risorse riproducibili;
- Una riflessione metodologica su: come integrare l'IA nel curricolo di Informatica; quali competenze trasversali (problem solving, etica) sviluppare; quali difficoltà potrebbero incontrare gli studenti e come facilitare. Questa esperienza ha dimostrato che l'IA per macchine a guida autonoma è un contesto didattico efficace per insegnare contenuti fondamentali di Informatica.

6 Conclusioni

Questa esperienza ha dimostrato che l'integrazione di varie metodologie didattiche innovative all'interno di una cornice pedagogica dove la narrazione e la conoscenza del territorio fanno da sfondo e da collante, consente a docenti con formazione di base diversa un approccio STEAM al tema dell'uso dell'IA per macchine a guida autonoma.
Offre, pertanto, un contesto didattico efficace per insegnare contenuti centrali di Informatica: algoritmi, reti neurali, dati, sistemi embedded, etica dell'IA.
I docenti hanno vissuto e condiviso una esperienza formativa riproducibile, con codice, QRcode e compiti di realtà, e hanno chiaro come integrarlo nel curricolo.

Il percorso, opportunamente modulato, può essere replicato in altre Scuole con risorse minime (PC + software open source e kit MaqueenPLUS + HuskyLens) e adattato a diversi livelli di preparazione degli studenti.

Risorse e riferimenti bibliografici

Linee guida per l'introduzione dell'intelligenza artificiale nelle Istituzioni scolastiche - ver. 1.0 - Agosto 2025 https://www.mim.gov.it/documents/20182/0/MIM_Linee+guida+IA+nella+ Scuola_09_08_2025-signed.pdf/b70fdc45-4b75-1f7e-73bf-eab12989b928

Seymour Papert, Mindstorms - Children, computers and powerful ideas (ENG), Basic Books, 2020 (edizione originale 1980)

Coding e Robotica, portale INDIRE - https://codingerobotica.indire.it/

Nichols, Mark H., Cator, Karen (2008), Challenge Based Learning White Paper. Cupertino, California: Apple, Inc.

Nichols M., Cator, K. and Torres M. (2016), Challenge Based Learning User Guide, Redwood City, CA: Apple, Inc.Digital promise

Freeman, S., Eddy, S.L. , McDonough, M. , Smith, M.K. , Okoroafor, N. , Jordt, H. & Wenderoth, M.P. . "Active learning increases student performance in science, engineering, and mathematics". In: Proceedings of the National Academy of Sciences of the United States of America, vol. 111, n. 23, pp. 8410-8415, 2014.

Outdoor education: apprendimento attivo tra dentro e fuori - Toolkit per la scuola dell'infanzia e per il primo ciclo di istruzione versione 1.0 [2024] - INDIRE

G. K. Sriram, A. Malini, and K. Santhosh, State of the Art of Artificial Intelligence Approaches Toward Driverless Technology. John Wiley & Sons, Ltd, 2024, ch. 3, pp. 55–74. [Online]. - https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119847656.ch3

Logica e Intelligenza Artificiale: pratiche didattiche per il Liceo Classico

Simone Cuconato[1] and Kristian Reale[2,3]

[1] Dipartimento di Fisica, Università della Calabria, Rende (CS)
simone.cuconato@unical.it
[2] Dipartimento di Matematica e Informatica, Università della Calabria, Rende (CS)
[3] Istituto d'Istruzione Superiore "Pietro Mazzone" – Roccella Ionica (RC)
kristian.reale@unical.it

Sommario

Il contributo presenta due casi di studio sull'introduzione dell'Intelligenza Artificiale (IA) nella sperimentazione CL@SSICO DIGITALE, sviluppata dall'Università della Calabria e dal Liceo Classico *G. da Fiore* di Rende (CS). Attraverso gli insegnamenti di *Logica e Filosofia dell'IA* e *Scienza e Tecnologie Digitali*, il lavoro mostra come la logica possa fungere da ponte tra cultura classica e innovazione tecnologica, favorendo una comprensione critica, spiegabile e interdisciplinare dell'Intelligenza Artificiale.

1 Contesto e obiettivi della sperimentazione Cl@ssico Digitale

Il contributo presenta due casi di studio relativi all'introduzione dell'Intelligenza Artificiale (IA) nella pratica didattica all'interno della sperimentazione dell'indirizzo CL@SSICO DIGITALE [4, 5], sviluppato nell'ambito dell'accordo-quadro tra l'Università della Calabria e il Liceo Classico "Gioacchino da Fiore" di Rende (CS). Il progetto nasce dall'esigenza di ripensare il senso formativo del Liceo Classico nell'età dell'IA, evitando tanto una postura tecnofoba quanto un'adesione acritica alla retorica dell'innovazione.

Tali presupposti si traducono, sul piano curricolare, nell'introduzione di due discipline innovative: *Logica e Filosofia dell'IA* e *Scienza e Tecnologie Digitali*. Le due materie costituiscono il nucleo teorico-operativo dell'intero progetto e rispondono a esigenze complementari, istituendo un dialogo costante tra la solidità del rigore formale e la concretezza delle applicazioni pratiche. Questo impianto teorico trova declinazione sul piano operativo attraverso una prassi didattica articolata su due livelli complementari. Da un lato, le nuove discipline beneficiano di uno spazio curricolare autonomo, garantito da un'ora settimanale aggiuntiva finalizzata allo sviluppo sistematico dei rispettivi nuclei concettuali. Dall'altro, la sperimentazione prevede una costante sinergia in compresenza con gli insegnamenti tradizionali del Liceo Classico, al fine di instaurare un dialogo fecondo tra saperi umanistici e scientifici.

Il contributo illustra la progettazione e la strutturazione di una lezione tipo. In questo quadro, si evidenzia come la consapevolezza degli strumenti della logica possa incidere positivamente sullo studio dell'informatica, legittimandone la centralità come disciplina concettuale e non meramente tecnica, perfettamente integrata nella *forma mentis* del Liceo Classico.

2 Casi di studio nella pratica didattica

A titolo esemplificativo, presentiamo due casi di didattica svolti in una classe terza, nei quali i contenuti di informatica sono stati introdotti in maniera integrata, secondo i principi del CL@SSICO DIGITALE.

2.1 Logica e Filosofia dell'IA

Il caso di studio relativo a *Logica e Filosofia dell'IA* traduce in prassi didattica tre linee di fondo del progetto: *i)* il superamento della rigida separazione positivistica dei saperi; *ii)* la continuità storico-epistemologica tra tradizione classico-umanistica e sapere scientifico-tecnologico; e *iii)* l'idea che il digitale debba essere compreso come fenomeno culturale e cognitivo, da interpretare criticamente in una prospettiva *sistemica*.

La lezione muove dalla filosofia greca e dalla nozione di *giustificazione*. Nel pensiero greco matura infatti l'esigenza della *dimostrazione*, intesa come esplicitazione delle ragioni di ciò che si afferma: si ha *epistéme* non quando si dice *che cosa* accade ma quando si sa dire *perché* accade, e da Aristotele in poi la logica si configura come l'*organon* che consente di indagare le strutture inferenziali del ragionamento. Su questo sfondo il tema della dimostrazione logica ha aperto una riflessione critica sulla matematica euclidea e sui fondamenti, raccordata poi alla *proof theory* [10] nella declinazione *strutturale* inaugurata da G. Gentzen per avvicinare la logica alle forme del ragionamento matematico effettivo. Di qui il passaggio all'IA neuro-simbolica, la cui questione centrale è: come integrare *learning-based methods* e *logic-based methods* [2, 6], in modo da ottenere sistemi non soltanto efficaci, ma logicamente trasparenti e formalmente verificabili? In classe la domanda non è stata posta in astratto, ma lungo un percorso che ha collegato teoria della dimostrazione, ragionamento automatico e *Large Language Models* (LLM).

La lezione ha introdotto allora un calcolo sintattico-finitario, i *sequent-style tableaux* [8]: il punto non era addestrare al calcolo, ma mostrare come un sistema di regole rigorosamente definito costituisca un modello di verificabilità inferenziale. Su questa base è stato affrontato l'impiego degli LLM come *proof assistant*, inteso non come un sostituto dell'attività deduttiva, ma come un ausilio alla dimostrazione. Per rendere concreta questa idea è stata realizzata una *Gem* di Gemini, programmata mediante istruzioni in codice LaTeX che vincolano il comportamento del modello alle regole del calcolo. Il modello, infatti, non può formulare un verdetto senza esibire una derivazione i cui passaggi siano istanze delle regole del sistema $\mathbb{L}_0 = (\mathscr{L}, \vdash \Rightarrow)$, secondo il paradigma del *Logic-Constrained Prompt Learning* [2]. È questo il fulcro dell'esperienza didattica: l'IA non è stata presentata come una scatola nera da interrogare passivamente, ma come un ambiente epistemico da sottoporre a una disciplina formale esplicita. L'architettura realizza così una forma di integrazione neuro-simbolica *per sottrazione*: invece di aggiungere un modulo simbolico a un modello altrimenti libero, si restringe progressivamente lo spazio delle risposte a ciò che il calcolo consente di derivare.

Il vantaggio didattico deriva direttamente da questo vincolo. Poiché il calcolo gode della proprietà di sottoformula, le formule che possono comparire in una derivazione sono vincolate fin dall'inizio all'insieme finito $\mathrm{Sub}^{\neg}(\Pi_0)$, ottenuto dalle sottoformule del blocco iniziale mediante chiusura per negazione. Nel nostro caso, per blocco si intende un insieme finito di formule; al sequente $\Gamma \Rightarrow \Delta$ corrisponde il blocco $\Gamma \cup \{\neg\psi \mid \psi \in \Delta\}$, mentre il blocco iniziale Π_0 è quello associato al sequente da esaminare e costituisce la radice del tableau. Anche l'eventuale comparsa di materiale estraneo diventa così immediatamente riconoscibile sul piano sintattico: una formula che non appartenga a $\mathrm{Sub}^{\neg}(\Pi_0)$ segnala una violazione del vincolo analitico [7]. Un rischio tipico dei modelli generativi viene quindi trasformato in una violazione localmente verificabile. La correttezza può essere controllata come una prova, confrontando ogni passaggio con le regole del sistema e con il perimetro sintattico stabilito all'origine, secondo una procedura accessibile anche allo studente. La garanzia rimane naturalmente condizionata alla correttezza del calcolo: un tableau ben formato e chiuso costituisce una certificazione del verdetto nella misura in cui ogni suo passaggio è conforme alle regole del sistema. L'architettura non elimina dunque la possibilità dell'errore; rende invece molto più difficile che esso rimanga invisibile.

La logica ha quindi fornito non solo gli strumenti per formalizzare il ragionamento, ma anche il quadro entro cui riflettere sulla spiegabilità dell'IA. Il sistema è spiegabile *per costruzione*: la derivazione che giustifica il verdetto ne costituisce parte integrante, traducendo nell'IA contemporanea l'esigenza di *rendere ragione* di ciò che si afferma. Su questa base si apre il passaggio alla fase successiva del

percorso, in cui la formalizzazione logica viene trasferita a contesti informatici più concreti nell'ambito di *Scienza e Tecnologie Digitali*.

2.2 Scienza e Tecnologie Digitali

Scienza e Tecnologie digitali è un insegnamento che affronta argomenti di Informatica ispirandosi, fin dalla sua costituzione, all'insegnamento dell'Informatica dei Licei Scientifici Opzione Scienze Applicate. Nel contesto del Liceo Classico, e sulla scia di quanto proposto dall'insegnamento di *Logica e Filosofia dell'IA*, durante il terzo anno, l'obiettivo è stato quello di avvicinare la classe all'utilizzo della logica in contesti reali, mostrando come sia possibile definire, attraverso strumenti formali, il processo di ragionamento deduttivo di un sistema di IA a partire da un insieme di dati reali. Questo tipo di approccio rientra nel campo del *Knowledge Representation and Reasoning (KRR)* e dell'Intelligenza Artificiale Simbolica, ideali e collegabili con il pensiero logico-simbolico oggetto del contributo che si sta dando al CL@SSICO DIGITALE.

Le prime lezioni svolte hanno riguardato l'introduzione alle basi di dati e, in particolare, la modellazione concettuale, attraverso la creazione di schemi Entità-Relazioni (ER), e la modellazione logica, mediante la realizzazione di schemi logici ed esempi di query con join. Tali tematiche sono state affrontate soprattutto in compresenza con *Matematica*, disciplina nell'ambito della quale ci si è concentrati sugli aspetti insiemistici relativi alle tuple di una tabella, con particolare riferimento al prodotto cartesiano e alla join nella costruzione delle query. Un'altra disciplina oggetto di compresenza è stata *Inglese*, durante la quale i medesimi argomenti relativi alle basi di dati sono stati affrontati in lingua inglese, ispirandosi al modello CLIL (Content and Language Integrated Learning).

Successivamente alla trattazione delle basi di dati, e rimanendo nell'ambito del *KRR*, si è passati a una prima introduzione dell'*Answer Set Programming* (ASP) [1], un paradigma computazionale dichiarativo basato sulla logica, che consente di effettuare ragionamenti su basi di conoscenza rappresentate da schemi strutturati (caratteristica tipica delle basi di dati). La metodologia di insegnamento di questo argomento si è ispirata a pratiche didattiche già sperimentate in alcuni istituti scolastici, nei quali l'*ASP* è stato proposto come contenuto di insegnamento [8, 9]. Nello specifico, durante le lezioni è stato mostrato come tradurre le informazioni contenute in tabelle in predicati e fatti logici, nonché come effettuare inferenze mediante l'utilizzo di regole logiche che consentono di derivare nuove informazioni a partire da quelle già note. In questo modo, gli studenti hanno potuto osservare concretamente il passaggio da una rappresentazione tabellare dei dati a una rappresentazione logico-dichiarativa basata su fatti e regole.

Introdotti tali concetti, si è passati rapidamente a un caso applicativo reale durante la compresenza con la disciplina *Scienze Naturali*, prendendo come argomento di riferimento le leggi di Mendel. In tale contesto, si è lavorato alla formalizzazione dei concetti necessari per la costruzione del *Quadrato di Punnett*, individuando insieme agli studenti, predicati, fatti e regole utili a rappresentare gli alleli, i genotipi dei genitori e le possibili combinazioni ereditarie. Una delle regole logiche elaborate insieme agli studenti è stata quella che consente di inferire il predicato *figlio*, affermando ad esempio che un individuo ha genotipo *Aa* se è figlio di un genitore che possiede l'allele A e di un secondo genitore che possiede l'allele a.

Uno dei vantaggi di ASP è la possibilità di utilizzare un motore simbolico per realizzare un verificatore delle deduzioni formalizzate nell'ambito della disciplina *Logica e Filosofia dell'IA*. Sebbene un LLM possa produrre catene di deduzione plausibili, anche mediante istruzioni formali in LaTeX, ASP rappresenta un candidato particolarmente adatto a verificarne la validità logica.

3 Conclusioni

I casi di studio presentati mostrano come la logica rappresenti il fondamento epistemologico per un'integrazione consapevole e critica dell'IA nel contesto del Liceo Classico. L'esperienza didattica evidenzia che l'IA non costituisce soltanto un nuovo strumento tecnologico, ma può diventare un'occasione privilegiata per sviluppare competenze argomentative e metacognitive, favorendo una comprensione più profonda dei processi di conoscenza e di ragionamento. In questo modo, la didattica dell'informatica, sottratta a una declinazione puramente operativa ed esecutiva, rivendica la sua profonda valenza speculativa, rivelandosi perfettamente integrata nella *forma mentis* e nel nucleo fondativo del Liceo Classico.

Riferimenti bibliografici

[1] Gerhard Brewka, Thomas Eiter, and Mirosław Truszczyński. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, December 2011.

[2] Simone Cuconato. Proofs, not narratives: A proof-theoretic approach to explainable AI in education. In *Proceedings of AI-LEAP 2026 — 1st International Workshop on AI Learning, Education and Practice, co-located with AIxIA 2026*, CEUR Workshop Proceedings, Perugia, Italy, 2026. CEUR-WS.org. [To appear].

[3] Simone Cuconato. Sequent-style tableaux for intuitionistic propositional logic. *Reports on Mathematical Logic*, 61, 2026. [To appear].

[4] Simone Cuconato and Kristian Reale. Liceo classico digitale: Proposta contenuti disciplinari e prime sperimentazioni. In M.A. Pellegrino and V. Rossano, editors, *Terzo Convegno Italiano sulla Didattica dell'Informatica (ITADINFO 2025)*, pages 259–262, Milano, 2025. Ledizioni.

[5] Simone Cuconato and Kristian Reale. Liceo classico digitale: proposta curriculare e prime sperimentazioni per l'integrazione dell'IA. In *Intelligenza Artificiale all'Università: Didattica, Orientamento e Formazione Docenti*. Università della Calabria, 2026. [To appear].

[6] Simone Cuconato, Francesco Scarcello, and Roberto Beneduci. Logical proof in AI education: Epistemological foundations and practical applications for a human-centered perspective. In *SIpEIA Conference 2026*. Sapienza Università di Roma, 2026. [To appear].

[7] Francesca Poggiolesi. On the importance of being analytic. the paradigmatic case of the logic of proofs. *Logique et Analyse*, 219:443–461, 2012.

[8] Kristian Reale. Insegnare answer set programming nelle scuole superiori. In E. Gentile and M. Monga, editors, *Primo Convegno Italiano sulla Didattica dell'Informatica (ITADINFO 2023)*, Bari - Università degli Studi di Bari Aldo Moro, 2023 Oct., pages = 167–170, year = 2023, isbn = 978-88-6629-075-9.

[9] Kristian Reale. Logic and answer set programming in high school: Two learning unit experiences. In *(ICLP-WS-DC 2025) co-located with 41st International Conference on Logic Programming (ICLP 2025), Rende, Italy, September 9-13, 2025*, volume 4117 of *CEUR Workshop Proceedings*, page 8. CEUR-WS.org, 2025.

[10] Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*, volume 43 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1996.

Oltre il coding: Live Action Role-Playing e tecnologie creative nei camp Ragazze Digitali

Linda Burchiellaro[1], Mirko Novello[2], Martina Casari[3], Francesco Faenza[3], and Claudia Canali[3]

[1] Università di Modena e Reggio Emilia,
Dipartimento di Scienze Fisiche, Informatiche e Matematiche
linda.burchiellaro@unimore.it
[2] Università di Modena e Reggio Emilia,
Dipartimento di Educazione e Scienze Umane
384086@studenti.unimore.it
[3] Università di Modena e Reggio Emilia,
Dipartimento di Ingegneria "Enzo Ferrari"
martina.casari@unimore.it
francesco.faenza@unimore.it
claudia.canali@unimore.it

Abstract

La partecipazione femminile all'informatica resta inferiore a quella maschile, influenzata da stereotipi di genere e senso di appartenenza all'informatica. Il contributo presenta l'integrazione dell'attività EduLARP The Order of Unseen Path, sviluppata nel progetto europeo TechLARP, in un camp Ragazze Digitali 2026. L'esperienza combina gioco di ruolo, narrazione e artefatti interattivi per avvicinare le studentesse all'informatica, computazione fisica e storia delle donne nella scienza.

1 Introduzione

L'informatica è oggi una competenza centrale per comprendere e partecipare alla società contemporanea. Tuttavia, le donne rimangono sottorappresentate nei percorsi di studio e nelle professioni legate alle tecnologie digitali. La letteratura evidenzia come questo divario sia legato a stereotipi culturali, carenza di "role model" femminili, minore percezione di autoefficacia e limitato senso di appartenenza alle comunità STEM [1, 2, 3].

Le attività extracurricolari possono contribuire a contrastare questi fattori, offrendo esperienze diverse dalla didattica tradizionale. In particolare, camp estivi basati su creatività, collaborazione, apprendimento pratico e modelli femminili permettono alle studentesse di sperimentare l'informatica come disciplina accessibile, concreta e socialmente significativa.

Questo contributo presenta l'integrazione tra due esperienze: il progetto Ragazze Digitali e il progetto europeo TechLARP[1]. Un LARP (Live Action Role Play) è un gioco di ruolo in cui il giocatore interpreta il personaggio "dal vivo", ossia anche nei suoi aspetti cinestetici: il giocatore non si limita a descrivere le azioni del suo personaggio a parole ma, spesso, compie in prima persona quelle azioni attraverso gestualità, prossemica, contatto, voce [4]. L'obiettivo è mostrare come una metodologia EduLARP (Educational Live Action Role Play), che si basa sull'esperienza in prima persona di situazioni reali o immaginarie e coinvolge gli studenti come partecipanti attivi in uno scenario educativo guidato, possa arricchire un camp di orientamento all'informatica rafforzandone la dimensione inclusiva, partecipativa e motivazionale.

[1] https://www.techlarp.eu/

2 Ragazze Digitali

Ragazze Digitali nasce nel 2014 a Modena dalla collaborazione tra il Dipartimento di Ingegneria "Enzo Ferrari" dell'Università di Modena e Reggio Emilia e l'associazione EWMD (European Women's Management Development), un network internazionale che promuove il confronto e la collaborazione tra professioniste provenienti dai settori dell'impresa, dell'istruzione, della politica e della cultura. Il progetto è stato concepito come un summer camp gratuito rivolto a studentesse delle classi terze e quarte delle scuole superiori, con l'obiettivo di avvicinarle all'informatica attraverso un approccio pratico, creativo e collaborativo.

Nel tempo, Ragazze Digitali ha coinvolto scuole, enti locali, fondazioni, imprese e università. Dal 2018 il camp è stato replicato anche a Reggio Emilia e Cesena, mentre dal 2022 è stato esteso a tutte le città dell'Emilia Romagna ed attivato anche a Mantova, confermandone la scalabilità in contesti differenti.

L'approccio didattico è fondato sul *learn-by-doing*: le partecipanti lavorano alla realizzazione di progetti concreti, come videogiochi, applicazioni, robot controllati da Arduino, attività maker o ambienti interattivi. Questo consente di ridurre la barriera iniziale legata all'astrazione e alla sintassi dei linguaggi di programmazione, sostenendo motivazione e coinvolgimento. Centrale è anche la presenza di role model femminili, attraverso incontri con docenti, ricercatrici e professioniste dell'informatica.

La crescita del progetto è evidenziata anche dall'incremento delle partecipanti: dalle 35 studentesse coinvolte nella prima edizione di Modena nel 2014 si è passati alle circa 250 partecipanti previste per l'edizione 2026 organizzata da UNIMORE nelle sedi di Modena, Reggio Emilia e Mantova. Tale evoluzione conferma la capacità dell'iniziativa di attrarre un numero crescente di giovani interessate ai temi dell'informatica.

2.1 Valutazione ed efficacia dei camp

Uno degli aspetti più rilevanti di Ragazze Digitali è l'attenzione alla valutazione dell'impatto. Nel corso delle edizioni è stato sviluppato uno strumento basato su questionari pre e post-camp, finalizzato a osservare cambiamenti nelle percezioni e nelle intenzioni future delle partecipanti.

Lo strumento considera diverse dimensioni: background delle studentesse, esperienze pregresse di coding, percezione dell'informatica, intenzioni rispetto agli studi universitari, consapevolezza degli stereotipi di genere e soddisfazione complessiva. Questa impostazione permette di valutare l'effetto del camp su aspetti centrali quali fiducia nelle proprie capacità e disponibilità a considerare l'informatica come possibile percorso futuro.

La Figura 1 mostra le intenzioni delle partecipanti (n = 44) di proseguire gli studi, rilevate prima e dopo il camp 2026 a Modena. L'intenzione di continuare il percorso formativo resta elevata, passando dal 90,9% al 93,2% (+2,3 punti percentuali). Più marcato è l'aumento dell'interesse verso studi in informatica, dal 4,5% all'11,4% (+6,8 punti percentuali). Sebbene i valori assoluti siano ancora contenuti, il dato suggerisce un effetto positivo dell'esperienza nel rafforzare l'interesse verso la disciplina. I risultati evidenziano il potenziale di approcci laboratoriali, collaborativi e narrativi nel promuovere l'orientamento delle studentesse verso percorsi legati all'informatica.

3 TechLARP ed EduLARP

TechLARP è un progetto europeo finalizzato allo sviluppo di metodologie educative inclusive, creative e basate sulle arti per l'insegnamento delle discipline tecnologiche. Il progetto integra il Live Action Role-Playing come strumento per rendere l'apprendimento della tecnologia più coinvolgente, collaborativo e accessibile [5].

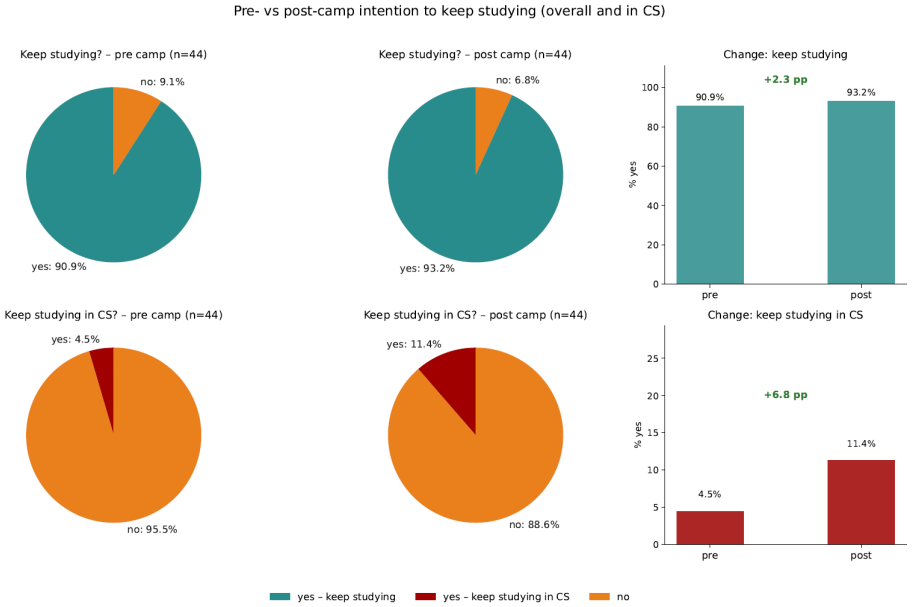


Figure 1: Analisi scelte accademiche future nelle risposte al questionario pre e post-camp

Nel contesto educativo, il LARP viene utilizzato come ambiente immersivo nel quale le partecipanti assumono ruoli, interagiscono in uno scenario narrativo, prendono decisioni e collaborano alla risoluzione di problemi. Questo approccio è particolarmente interessante per la didattica dell'informatica, poichè consente di trasformare concetti astratti in esperienze situate, corporee e sociali.

Il progetto propone inoltre un *Gender Inclusion Index*, pensato per guidare la progettazione di interventi educativi inclusivi. L'indice considera dimensioni quali targeting inclusivo, collaborazione con gli stakeholder, metodologie innovative, sostenibilità, valutazione dell'impatto e approccio LARP-based. Tali dimensioni non costituiscono una griglia rigida, ma uno strumento di riflessione per adattare le attività ai diversi contesti educativi.

3.1 *The Order of Unseen Path*

All'interno di TechLARP è stata progettata l'attività EduLARP *The Order of Unseen Path*, pensata per integrare narrazione, storia della scienza, progettazione creativa e computazione fisica.

Lo scenario Lo scenario è ambientato in un archivio di tomi viventi che custodiscono scoperte scientifiche. Una minaccia, il Velo dell'Amnesia, ha colpito l'ala dedicata alle scienziate dimenticate; le partecipanti assumono quindi il ruolo di apprendiste restauratrici incaricate di recuperare un tomo danneggiato.
La prima missione è dedicata a Eunice Foote e al suo contributo alla comprensione della capacità di alcuni gas, tra cui la CO_2, di trattenere calore. Nel gioco, il tomo di Foote ha perso il proprio "glifo di illuminazione"; nel mondo reale, le partecipanti devono progettare e costruire un circuito che permetta a un LED di accendersi quando il libro viene aperto. L'attività collega così una sfida narrativa a un obiettivo tecnico: comprendere circuito aperto e chiuso, ruolo di batteria, LED, interruttore e materiali conduttivi.

Il lavoro è organizzato in gruppi con ruoli complementari: la *Chronicler* cura la dimensione storica, l'*Artificer* quella tecnica, l'*Illuminator* gli aspetti visivi e narrativi, mentre la *Resonator* valuta accessibilità e usabilità dell'artefatto. Questa struttura valorizza competenze diverse e riduce il rischio che l'attività venga percepita come esclusivamente tecnica.

Per supportare l'interpretazione dei ruoli e favorire l'immersione narrativa, a ciascuna partecipante viene assegnata una carta personaggio contenente una breve descrizione del ruolo, delle responsabilità all'interno del gruppo e delle competenze valorizzate durante l'attività. La Figura 2 mostra un esempio carta utilizzate nell'esperienza *The Order of Unseen Path* e dei possibili spunti per altre carte.



(a) Esempio di carta

(b) Spunti per ulteriori carte

Figure 2: Esempi di carte personaggio utilizzate nell'attività *The Order of Unseen Path*.

4 Sperimentazione nel camp

La sperimentazione prevista a settembre 2026 inserirà *The Order of Unseen Path* in un camp Ragazze Digitali. L'integrazione è coerente con la storia del progetto, che ha sempre privilegiato attività pratiche e collaborative. L'EduLARP aggiunge a questa impostazione una componente narrativa e immersiva, presentando la tecnologia come mezzo per raccontare, ricordare, progettare e collaborare.

L'attività sarà valutata riprendendo quanto già utilizzato nei camp Ragazze Digitali e integrandolo con domande specifiche sull'esperienza EduLARP. Saranno osservati il coinvolgimento percepito e partecipazione attiva delle studentesse, la comprensione dei concetti tecnici, la percezione del lavoro di gruppo, il ruolo della narrazione nell'apprendimento e il cambiamento nella percezione dell'informatica

come disciplina creativa e accessibile. Una fase di debriefing permetterà infine di collegare gli elementi fantastici del gioco ai corrispondenti concetti reali: ad esempio, il recupero del "glifo di illuminazione" nel tomo di Eunice Foote verrà ricondotto alla progettazione del circuito elettrico che consente l'accensione del LED all'apertura del libro.

5 Conclusioni

Il contributo ha presentato una proposta di integrazione tra Ragazze Digitali e la metodologia EduLARP sviluppata in TechLARP. I risultati delle precedenti edizioni mostrano che attività laboratoriali, collaborative e in presenza possono influenzare positivamente autoefficacia e interesse verso l'informatica. *The Order of Unseen Path* rappresenta un primo caso di applicazione di tale approccio nel contesto di Ragazze Digitali: attraverso la costruzione di un libro interattivo dedicato a Eunice Foote, le partecipanti combinano storia, creatività, elettronica e collaborazione. La sperimentazione permetterà di osservare il potenziale dell'EduLARP come metodologia per una didattica dell'informatica più inclusiva e partecipativa.

References

[1] S. Cheryan, E. J. Lombard, F. Hailu, L. N. H. Pham *et al.*, "Global patterns of gender disparities in stem and explanations for their persistence," *Nature Reviews Psychology*, vol. 4, pp. 6–19, 2025.

[2] B. Spieler, L. Oates-Indruchova, and W. Slany, "Female students in computer science education: Understanding stereotypes, negative impacts, and positive motivation," *Journal of Women and Minorities in Science and Engineering*, vol. 26, no. 5, 2020.

[3] A. Master, "Gender stereotypes influence children's stem motivation," *Child Development Perspectives*, vol. 15, no. 3, pp. 203–210, 2021.

[4] A. Maragliano, G. Ceresa, A. Giovannucci, A. Corbo, A. Capone, and A. Traverso, *EDU-LARP Game Design per giochi di ruolo educativi. A cura di Andrea Maragliano. FrancoAngeli con allegato online*, 10 2020, pp. 208–217.

[5] TechLARP Consortium, "Blueprint for gender inclusive and creative tech education," TechLARP Project, 2025, erasmus+ Project Output.

Regressione lineare come laboratorio interdisciplinare: un percorso integrato tra informatica e matematica

Francesca Arrigoni and Francesca Voltolini

Liceo Scientifico "Leonardo da Vinci", Trento, Italy
{francesca.arrigoni, francesca.voltolini}@liceodavincitn.it

Abstract

Il presente contributo descrive un percorso didattico interdisciplinare, progettato congiuntamente da una docente di matematica e una docente di informatica e sperimentato in una classe quarta di Liceo Scientifico opzione Scienze Applicate. Il percorso, che assume la regressione lineare come filo conduttore tra le due discipline, si sviluppa per fasi progressive: dall'acquisizione degli strumenti informatici di base (librerie Matplotlib e Pandas), alla costruzione concettuale del criterio dei minimi quadrati tramite simulazioni e geometria dinamica, fino alla raccolta di un dataset reale e personale e alla sua modellizzazione, prima con un approccio statistico classico e poi tramite una rete neurale lineare implementata con PyTorch.

1 Introduzione

La regressione lineare è un terreno privilegiato per l'incontro tra matematica e informatica nella scuola secondaria di secondo grado: è uno dei modelli statistici più intuitivi per descrivere la relazione tra variabili, ma è anche il punto di ingresso naturale verso i concetti fondamentali del machine learning, di crescente rilevanza nei curricoli scolastici.
L'obiettivo del percorso presentato non è solo "far funzionare" il codice, ma rendere visibile la continuità tra l'intuizione geometrica, la formalizzazione algebrica e l'implementazione computazionale, in un percorso che culmina in un primo contatto consapevole con i modelli di intelligenza artificiale.
L'attività è stata sperimentata presso il Liceo Scientifico "Leonardo Da Vinci" di Trento, durante l'anno scolastico 2025-26, in una classe quarta, quando gli studenti sono in grado di sviluppare semplici programmi in linguaggio Python e conoscono le nozioni di base di statistica e probabilità.
Il percorso nasce dal lavoro di un gruppo stabile di docenti di matematica e informatica che condividono una convinzione comune: l'interdisciplinarietà non è un semplice intreccio tra materie, ma una condizione necessaria per comprendere in modo più profondo i saperi e i loro legami. In questa prospettiva l'idea di fondo è che i concetti acquistino significato quando vengono osservati da punti di vista diversi e messi in relazione tra loro, superando la frammentazione disciplinare.

2 Descrizione delle attività

2.1 Prima fase: concetti e strumenti di base

2.1.1 Informatica

Per fornire agli studenti le competenze necessarie all'utilizzo degli strumenti informatici per l'analisi dei dati, sono stati proposti due tutorial pratici dedicati alle librerie python Matplotlib e Pandas, entrambi realizzati con Google Colab ma costruiti secondo due metodologie didattiche differenti.

Il primo tutorial riguarda Matplotlib (Figura 1), la libreria più diffusa per realizzare grafici in python, con riferimento al modulo pyplot, e segue la metodologia PRIMM [1, 2]: per ogni nuovo blocco di

codice lo studente è prima invitato a predire cosa farà, poi a eseguirlo per verificare la previsione, a investigarlo commentando riga per riga le funzionalità scoperte, a modificarlo introducendo variazioni (colori, stili, parametri), fino ad arrivare a crearlo autonomamente in un esercizio conclusivo con dati propri. Partendo da un semplice grafico a linee costruito con la funzione `plot()`, il percorso guida lo studente a comprendere il significato delle coordinate x e y, per poi passare alla personalizzazione grafica tramite titolo, etichette degli assi e legenda, fino alla rappresentazione di più serie di dati e alle diverse tipologie di grafico.

```
'''
-- PREDICT --
Domanda: Cosa fa il codice sotto riportato secondo te?
Risposta:

-- RUN --
Esegui il codice e controlla se fa quello che ti aspetti.

-- INVESTIGATE --
Se non lo fa, modifica la risposta precedente e commenta le righe di
codice con le nuove funzionalità che hai scoperto.
'''

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [25, 20, 21, 22, 15]

plt.plot(x,y)
plt.show()
```

```
'''
--- MODIFY ---
Copia e incolla il codice dell'esercizio precedente modificando la
lista y nel seguente modo: y = [-25, -20, -10, 5, 15]'''
```

```
'''
-- MAKE --
Crea un grafico, personalizzandolo a tuo piacimento, che rappresenti l'andamento
dei tuoi voti di matematica del primo quadrimestre.
'''
```

Figura 1: Estratto tutorial Matplotlib

Il secondo tutorial introduce Pandas (Figura 2), la libreria per la manipolazione e l'analisi dei dati tabellari, e segue invece lo schema Domanda-Risposta-Esercizio: per ogni nuovo comando viene posta allo studente una domanda esplicita a cui rispondere per iscritto prima di eseguire il codice, seguita da un esercizio di applicazione su un nuovo caso. Utilizzando il dataset *iris*[1], lo studente esplora il dataframe con comandi come `head()`, `info()` e `describe()`, impara a filtrare righe e colonne in base a condizioni e a calcolare statistiche descrittive.

[1] `https://archive.ics.uci.edu/dataset/53/iris`

```
#importo il file "iris.csv" e salvo i dati nella variabile iris
iris = pd.read_csv("iris.csv")
```

```
#accedo alle prime 5 righe del dataframe
iris.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
#DOMANDE
'''
D1: Cosa indicano secondo te i numeri crescenti della prima colonna?
R1:

D2: Cosa indica ogni singola colonna?
R2:
'''
```

```
#ESERCIZIO
#scrivi il comando per visualizzare le prime 10 righe del dataframe
```

Figura 2: Estratto tutorial Pandas

Insieme, i due tutorial costituiscono la base tecnica propedeutica all'introduzione della regressione lineare: Matplotlib fornisce gli strumenti per visualizzare le relazioni tra variabili, mentre Pandas permette di caricare, esplorare e preparare i dati su cui tale modello verrà successivamente applicato.

2.1.2 Matematica

Per introdurre la retta dei minimi quadrati, agli studenti è stata proposta una simulazione del progetto PhET Interactive Simulations[2] (Università del Colorado Boulder), che favorisce l'apprendimento per esplorazione e scoperta. Seguendo i principi della mediazione semiotica [3], l'applet ha guidato gli studenti verso una modellizzazione emergente del problema. Nella fase successiva di validazione, secondo il framework di Brousseau [4], gli studenti hanno ricostruito autonomamente l'applet in GeoGebra, seguendo una scheda guidata e partendo da una nuvola di punti assegnata (Figura 3).
Per dimostrare che la retta individuata per tentativi minimizza effettivamente la somma dei quadrati dei residui, è stata imposto come vincolo il passaggio per il baricentro della nuvola: in questo modo la somma dei residui quadratici è diventata funzione della sola pendenza m, permettendo di trovarne il minimo con gli strumenti matematici già noti sulle funzioni quadratiche. Una discussione guidata dal docente ha infine portato all'istituzionalizzazione [4] del criterio dei minimi quadrati per ottenere il modello che meglio descrive la relazione lineare tra x e y.

[2] https://phet.colorado.edu/sims/html/least-squares-regression/latest/least-squares-regression_all.html

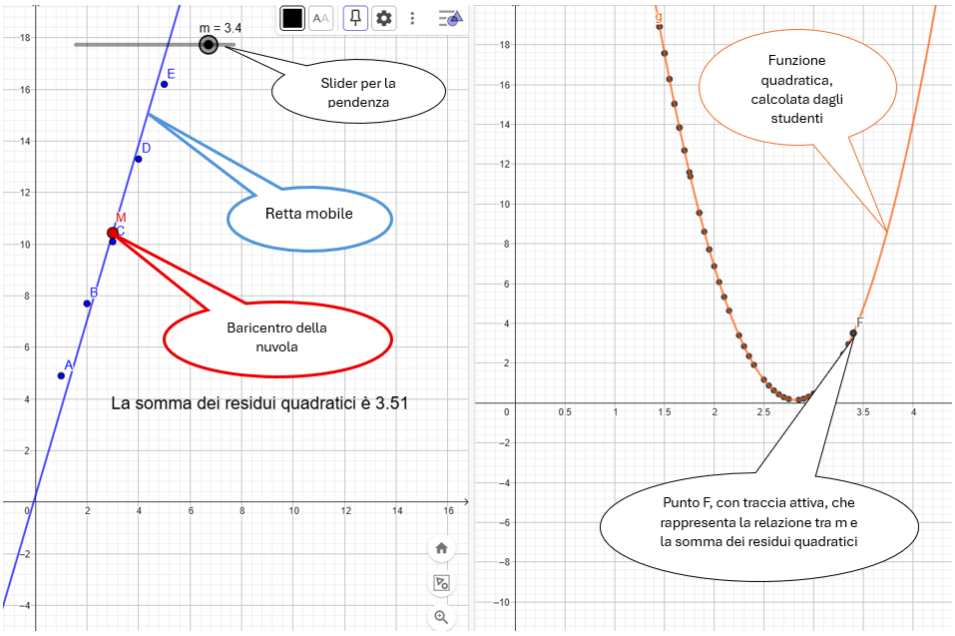


Figura 3: Schermata di Geogebra

2.2 Seconda fase: integrazione

L'attività didattica proposta trae ispirazione da un modulo originariamente sviluppato all'interno del progetto ministeriale canadese TIPS4RM (Targeted Implementation and Planning Supports for Revised Mathematics)[5] nei primi anni 2000, volto all'insegnamento dei modelli lineari e della retta di miglior adattamento (best fit line) attraverso contesti reali.
Gli studenti misurano la lunghezza dell'avambraccio (X_1), l'apertura delle braccia (X_2), la lunghezza del piede (X_3) e l'altezza (Y).

Successivamente, le misure trovate vengono inserite in un foglio Google condiviso:

N. Studente	Altezza (cm)	Avambraccio (cm)	Apertura braccia (cm)	Piede (cm)
S01	182,0	29,5	198,0	26,0
S02	171,0	25,0	165,0	25,5
S03	182,0	29,0	192,0	26,5
S04	175,6	26,3	181,0	26,4
S05	162,0	25,0	160,0	23,0
S06	170,0	26,5	173,0	23,5
S07	174,0	26,0	181,0	26,0
S08	180,0	26,0	176,0	25,0
S09	181,0	28,0	180,0	28,0
S10	181,0	27,0	185,0	25,0
S11	183,0	28,0	187,0	26,5
S12	161,0	25,0	165,0	24,0

Tabella 1: La tabella compilata dagli studenti

Ogni studente crea un dataset personalizzato a partire da questa tabella, eliminando la riga corrispondente ai propri dati e lo trasforma in un dataframe.
Le indicazioni fornite agli studenti per guidare l'analisi dei dati e per utilizzare, in questo contesto, la retta dei minimi quadrati studiata in precedenza sono state inserite in un notebook Google Colab per poter lavorare con le librerie Numpy, Pandas e Matplotlib.

Nella discussione finale si sottolinea il fatto che la previsione fatta con il modello (basato su un dataset che non contiene i dati dello studente che lo sta usando) può discostarsi dalla reale altezza dello studente e si esamina l'entità dell'errore commesso con l'interpolazione.

Mostriamo due delle celle di codice presenti nel Colab:

```
# 1. Selezione Variabili
x = df_altezze['Piede']
y = df_altezze['Altezza']

# 2. Calcolo Regressione (m, q)
m, q = np.polyfit(x, y, 1)

# 3. Grafico
plt.figure(figsize=(8, 6))

# Punti sperimentali
plt.scatter()

# Retta di regressione
plt.plot()

# Baricentro (Media)
plt.scatter()

plt.title("Relazione Piede - Altezza")
plt.xlabel("Lunghezza Piede (cm)")
plt.ylabel("Altezza (cm)")
plt.legend()
plt.grid(True)

plt.show()
```

Listato 1: Script per il calcolo dei parametri e la visualizzazione della retta.

```
print("--- CALCOLATORE DI ALTEZZA (Dal Piede) ---")

# 1. Chiedi direttamente la misura del piede come input
piede_input = float(input("Inserisci la misura del Piede in cm: "))

# 2. Ricalcola m e q usando solo i dati del Piede e dell'Altezza
x_ref = df_altezze['Piede']
y_ref = df_altezze['Altezza']

m_stima, q_stima = np.polyfit(x_ref, y_ref, 1)

# 3. Calcola la stima dell'altezza
altezza_stimata = m_stima * piede_input + q_stima

# 4. Stampa il risultato
print("\n" + "="*40)
print(f"Parametri retta (Piede): m={m_stima:.2f}, q={q_stima:.2f}")
print(f"ALTEZZA STIMATA: {altezza_stimata:.2f} cm")
print("="*40)
```

Listato 2: Stima basata sull'input utente.

2.3 Terza fase: modelli lineari a confronto

In questa fase è stato introdotto un modello di machine learning predittivo tramite la libreria PyTorch. Dal punto di vista metodologico, si è adottato un approccio ispirato alla Cognizione situata [6]: agli studenti è stata inizialmente proposta la visione di un video[3] a carattere divulgativo che illustra un modello di regressione lineare realizzato con PyTorch. La richiesta successiva è stata quella di replicare l'architettura adattandola al dataset antropometrico (piede-altezza) precedentemente raccolto. Durante questa attività i dettagli matematici intrinseci della libreria sono stati trattati come una black-box.
Anche in questo caso è stato caricato in Classroom un Colab da completare a cura del singolo studente; riportiamo quattro celle di codice del Colab utilizzato.

[3] `https://www.youtube.com/watch?v=7BuZ4LdZhfU&t=926s`

```
# usa le informazioni del video per configurare le costanti
SEED =
LEARNING_RATE =
EPOCHS =
```

Listato 3: Configurazione delle costanti per l'addestramento

```
# completa indicando la feature (x) e il target (y) da estrarre dal dataframe
x = torch.tensor(df_altezze[],dtype=torch.float32).unsqueeze(1)
y = torch.tensor(df_altezze[],dtype=torch.float32).unsqueeze(1)
model=torch.nn.Linear(1,1)
loss_fn=torch.nn.MSELoss()
optimizer=torch.optim.Adam(model.parameters(), lr=LEARNING_RATE)
```

Listato 4: Creazione modello

```
for epoch in range(EPOCHS+1):
    # usa le informazioni del video per completare questa parte di allenamento
plt.scatter(x,y)
plt.legend()
```

Listato 5: Allenamento del modello e parte grafica

```
print(f"La pendenza (m) o peso (w) e': {model.weight.item()}")
print(f"L'intercetta (q) o bias (b) e': {model.bias.item()}")
print(f"Loss finale (MSE): {loss.item()}")
```

Listato 6: parametri finali

A conclusione del lavoro, il docente fornisce un dataset in formato CSV. Il file contiene dati antropometrici relativi ad un campione in parte differente da quello sul quale gli studenti hanno lavorato in precedenza: si tratta delle stesse variabili quantitative, ma le unità statistiche riguardano un numero maggiore di studenti sempre della stessa età.
Gli studenti analizzano la relazione tra apertura delle braccia e altezza con due modelli lineari:

- il primo ottenuto con la retta dei minimi quadrati;
- il secondo ottenuto con il machine learning.

2.4 Relazione finale

Come relazione finale del lavoro svolto le docenti chiedono di consegnare due file Colab opportunamente commentati (retta di regressione lineare con python e modello lineare con PyTorch) e un documento Google che spieghi come vengono ottenuti i due modelli.

Al termine della relazione viene richiesto di inserire uno o più esempi di utilizzo dei modelli costruiti per stimare l'altezza di una persona di cui è nota la misura dell'apertura delle braccia.

3 Valutazione

La valutazione del percorso ha previsto sia una valutazione formativa, attraverso l'osservazione diretta delle diverse fasi dell'attività (tutorial PRIMM, domanda-risposta-esercizio, simulazione PhET, GeoGebra), sia una valutazione sommativa finale basata sugli elaborati consegnati dagli studenti.

È stata costruita una rubrica valutativa condivisa dalle due docenti e articolata su quattro dimensioni:

- correttezza tecnica del codice python prodotto
- qualità e pertinenza dei commenti e delle celle di testo esplicative
- correttezza dell'interpretazione matematica dei risultati
- capacità di utilizzare criticamente i modelli costruiti, riconoscendo i limiti delle previsioni effettuate su individui non presenti nel dataset di addestramento

La natura interdisciplinare del compito ha richiesto una valutazione congiunta da parte delle due docenti, ciascuna focalizzata sulla propria dimensione disciplinare, ma con un giudizio finale unico, coerente con l'impostazione integrata dell'intero percorso.

4 Punti di forza e debolezza. Sviluppi futuri.

Il percorso ha avuto come punto di forza principale il forte coinvolgimento degli studenti, reso possibile dall'uso di dati personali che hanno reso concreto un tema altrimenti astratto. Anche la progressione degli strumenti (PhET, GeoGebra, python e PyTorch) si è dimostrata efficace nel costruire gradualmente il significato del criterio dei minimi quadrati prima della sua formalizzazione computazionale.

Sono emerse però differenze significative nelle competenze di programmazione, soprattutto nella fase con PyTorch, dove alcuni studenti hanno lavorato in autonomia mentre altri hanno richiesto più supporto. L'approccio black-box ha facilitato il focus sul processo di training ma ha lasciato, in alcuni casi, dubbi sul funzionamento interno del modello.

Nel prossimo anno scolastico si riprenderà il modello lineare affrontato quest'anno come black-box, per analizzarne questa volta il funzionamento interno (forward pass, funzione di perdita, aggiornamento dei pesi) e passare progressivamente da un singolo neurone a reti più articolate, introducendo concetti come overfitting, generalizzazione e qualità dei dati.

Riferimenti bibliografici

[1] Sue Sentance, Jane Waite, and Maria Kallia. Teaching computer programming with primm: a sociocultural perspective. *Computer Science Education*, 29(2-3):136–176, 2019.

[2] Giulia Peserico, Francesca Voltolini, Maria Serafini, Federica Picasso, Daniele Agostini, Francesca Fiore, Anna Serbati, and Alberto Montresor. Sperimentazione del metodo PRIMM per l'insegnamento della programmazione. In *Atti del 1° Convegno Italiano sulla Didattica dell'Informatica (ITADINFO '23)*, pages 180–189, Bari, Italia, Ottobre 2023. Università degli Studi di Bari Aldo Moro.

[3] Maria G. Bartolini Bussi and Maria A. Mariotti. Semiotic mediation in the mathematics classroom: Artifacts and signs after a vygotskian perspective. *Handbook of International Research in Mathematics Education*, pages 746–783, 2008.

[4] Guy Brousseau. *Theory of Didactical Situations in Mathematics: Didactique des Mathématiques, 1970–1990*. Kluwer Academic Publishers, Dordrecht, NL, 1997.

[5] Ontario Ministry of Education. *Targeted Implementation and Planning Supports for Revised Mathematics (TIPS4RM) for Grade 9 Applied.* Government of Ontario, Toronto, ON, Canada, 2005. Original resources distributed via EduGAINS.ca (now archived).

[6] John Seely Brown, Allan Collins, and Paul Duguid. Situated cognition and the culture of learning. *Educational Researcher*, 18:32–42, 1989.

Sperimentazione di un Liceo Classico a curvatura "digitale"

Luca Basteris [1], Maria Cristina Daperno [2]
[1] Liceo Classico e Scientifico Statale “Silvio Pellico – Giuseppe Peano” - Cuneo
luca.basteris@liceocuneo.it
[2] Liceo Classico e Scientifico Statale “Silvio Pellico – Giuseppe Peano” - Cuneo
cristina.daperno@liceocuneo.it

Abstract

Nell’anno scolastico 2025/2026 il Liceo Classico “Pellico-Peano” di Cuneo ha attivato una sezione di Liceo Classico Digitale con due ore aggiuntive dedicate a Intelligenza Artificiale e digitale, scelta da quasi il 50% degli iscritti. L’iniziativa invita a ripensare il liceo classico: non sono latino e greco a essere superati, ma spesso le metodologie didattiche. Anche la programmazione, come le lingue classiche, può contribuire alla strutturazione del pensiero dello studente. L’innovazione può così rafforzare la centralità della persona nell’era dell’IA.

1 Introduzione

Il Liceo Classico-Scientifico “Silvio Pellico – Giuseppe Peano” di Cuneo ha attivato nell’anno scolastico 2025/2026 una sezione di Liceo Classico Digitale, con l’inserimento di due ore aggiuntive dedicate all’Intelligenza Artificiale e alle competenze digitali. La proposta è stata scelta da oltre venti studenti, pari a poco meno del 50% degli iscritti alla classe prima dell’indirizzo classico. Attualmente sono quindi presenti due sezioni prime: una tradizionale e una sperimentale a indirizzo digitale.

Quanto questa scelta sia stata realmente condivisa e quanto rappresenti, invece, una strategia per contrastare il progressivo calo delle iscrizioni che interessa i licei classici a livello nazionale è una questione aperta. Numerosi sono, infatti, gli interrogativi e le riflessioni che emergono, non soltanto in relazione agli aspetti tecnologici, ma più in generale riguardo all’offerta formativa del liceo classico.

Il dibattito, tuttavia, non dovrebbe concentrarsi sull’utilità o meno delle discipline classiche. Latino e greco non sono discipline anacronistiche o prive di valore nel contesto contemporaneo. Il nodo centrale riguarda, piuttosto, le metodologie didattiche con cui esse, e come loro le altre discipline, vengono proposte agli studenti, metodologie che in alcuni casi faticano a intercettare gli stili di apprendimento delle nuove generazioni e che talvolta conservano impostazioni ereditate dalla tradizione gentiliana.

L’introduzione di un percorso digitale ha inevitabilmente spinto un gruppo di docenti a interrogarsi sul significato stesso di proporre oggi un liceo classico, avviando un processo di innovazione e sperimentazione. In questa prospettiva, l’Intelligenza Artificiale, il digitale e anche

la programmazione non devono essere considerati semplicemente come contenuti aggiuntivi o competenze tecniche. Il pensiero computazionale e la programmazione possono, infatti, contribuire alla strutturazione del pensiero dello studente, sviluppando capacità di analisi, astrazione, problem solving e rigore logico che dialogano in modo naturale con le capacità ed abilità tradizionalmente promosse dallo studio delle lingue classiche e della filosofia.

D'altra parte, tutte le principali indicazioni legislative e pedagogiche relative all'Intelligenza Artificiale individuano come principio fondamentale la "centralità della persona". Lo stesso Ministero dell'Istruzione e del Merito, nelle "Linee guida per l'introduzione dell'Intelligenza Artificiale nelle Istituzioni scolastiche" pubblicate nel luglio 2025, richiama un approccio human-centered, che pone bisogni, capacità ed esigenze delle persone al centro dei processi di sviluppo e innovazione.

In questo senso, il liceo classico può assumere un ruolo particolarmente significativo. Nessun altro percorso formativo possiede una tradizione così fortemente orientata alla riflessione sull'uomo, sulla sua storia, sul suo pensiero e sul suo divenire. Accogliere le sfide del digitale, dell'Intelligenza Artificiale e della programmazione non significa rinunciare alla propria identità, ma piuttosto arricchirla, individuando nuove modalità per formare cittadini consapevoli, capaci di comprendere criticamente la contemporaneità senza perdere il legame con le radici culturali che la rendono comprensibile.

2 Progettazione del curricolo del Liceo Classico Digitale

Per la progettazione del nuovo Liceo Classico Digitale è stato costituito un gruppo di lavoro composto dai docenti del Consiglio di Classe e da altri insegnanti interessati. Grazie ai finanziamenti del DM 66 è stato realizzato un percorso di formazione sull'Intelligenza Artificiale applicata alla didattica, seguito da una fase di co-progettazione finalizzata all'integrazione delle nuove competenze nel curricolo tradizionale.

La formazione, della durata di 12 ore, ha affrontato temi quali normativa, storia dell'informatica e dell'IA, big data, aspetti etici, linguaggio, arte e creatività digitale. Sono stati inoltre presentati strumenti e metodologie utilizzabili nelle diverse discipline.

Successivamente i docenti hanno dedicato sei ore alla progettazione del curricolo, con l'obiettivo di integrare l'IA senza modificare i contenuti fondamentali del liceo classico. Il lavoro si è sviluppato lungo tre direttrici: educare sull'IA, per comprenderne il funzionamento; educare con l'IA, per utilizzarla nei processi di apprendimento; educare per l'IA, per sviluppare una riflessione critica sul rapporto tra tecnologia e società.

Nelle discipline umanistiche sono state previste attività con strumenti digitali, dizionari collaborativi, flash card e analisi delle traduzioni generate dall'IA. Nelle discipline scientifiche si è dato spazio alla raccolta e all'analisi dei dati, anche in ambiti applicativi come le scienze motorie. È stata inoltre introdotta la programmazione visuale come strumento per sviluppare pensiero logico e problem solving.

La disciplina specifica di IA fornirà conoscenze su storia dell'informatica, big data, reti neurali e sostenibilità digitale. In modo trasversale, il curricolo affronterà temi di Educazione Civica Digitale, tra cui bias algoritmici, fake news, fact-checking, prompting, uso consapevole degli strumenti digitali e progettazione di applicazioni orientate al benessere umano.

3 Alcuni esempi di attività

3.1 Visita al museo Tecnologic@mente ad Ivrea

Le classi del Liceo Classico Digitale hanno partecipato a una visita didattica presso il Museo Tecnologic@mente di Ivrea, un'esperienza significativa per approfondire la storia dell'innovazione italiana attraverso la figura di Adriano Olivetti. Il percorso museale ha permesso agli studenti di conoscere da vicino macchine da scrivere, calcolatrici e computer che hanno segnato tappe fondamentali nello sviluppo della tecnologia e dell'informatica. La visita ad Ivrea ha previsto anche una visita guidata dei "luoghi" olivettiani, come la fabbrica, l'asilo, la biblioteca ecc…, per toccare con mano l'unione tra tecnica ed umanità nella visione di Adriano Olivetti. La visita si è rivelata particolarmente interessante anche in relazione alle attività di approfondimento svolte nel percorso di Informatica del liceo.

Attraverso la storia della Olivetti, gli studenti hanno scoperto come il progresso tecnologico possa essere accompagnato da una profonda attenzione alla persona. Adriano Olivetti, infatti, non concepiva l'impresa soltanto come luogo di produzione e innovazione, ma come una comunità capace di promuovere cultura, formazione, benessere e partecipazione. Il Museo Tecnologic@mente racconta proprio questa visione, mostrando come tecnologia e umanesimo possano procedere insieme. L'esperienza ha offerto agli studenti l'opportunità di riflettere sul rapporto tra sviluppo scientifico, responsabilità sociale e valori umani, temi oggi più attuali che mai nell'era digitale.



Figura 1: Studenti alla visita del MaAM ad Ivrea

3.2 Costruzione di una linea del tempo della storia dell'informatica

Nell'ambito delle attività previste per il Liceo Classico Digitale, è stata proposta agli studenti un'esperienza didattica dedicata alla ricostruzione della storia dell'informatica e dell'intelligenza artificiale attraverso la realizzazione collaborativa di una linea del tempo analogica e digitale.

L'attività è nata con una duplice finalità: da un lato fornire agli studenti una visione storica dell'evoluzione delle tecnologie informatiche e dell'Intelligenza Artificiale, dall'altro favorire l'acquisizione di competenze digitali attraverso l'utilizzo di strumenti collaborativi e di ricerca delle informazioni.

Gli studenti sono stati chiamati a individuare eventi, personaggi, invenzioni e innovazioni che hanno segnato lo sviluppo dell'informatica, a partire dai primi strumenti di calcolo fino alle

moderne applicazioni dell'Intelligenza Artificiale. Per ciascun elemento inserito nella linea del tempo sono state svolte attività di ricerca, selezione e verifica delle fonti, con particolare attenzione all'affidabilità delle informazioni reperite online.

La costruzione della linea del tempo ha consentito di evidenziare come l'evoluzione tecnologica non sia un processo esclusivamente tecnico, ma sia profondamente intrecciato con aspetti culturali, sociali, economici e filosofici. In questa prospettiva, gli studenti del liceo classico hanno potuto cogliere le connessioni tra il pensiero umanistico e lo sviluppo scientifico e tecnologico, riflettendo sul ruolo dell'uomo nell'ideazione e nell'utilizzo degli strumenti digitali.

L'attività ha inoltre rappresentato un'occasione per sviluppare competenze trasversali quali il lavoro di gruppo, l'organizzazione delle informazioni, la capacità di sintesi e la comunicazione efficace. La documentazione dei materiali prodotti e delle diverse fasi del lavoro ha consentito di monitorare il percorso svolto, raccogliere evidenze utili per la valutazione dell'esperienza e costruire un patrimonio di risorse da mettere a disposizione delle future classi del Liceo Classico Digitale.

La linea del tempo è così diventata non soltanto uno strumento per apprendere la storia dell'informatica, ma anche un'occasione per riflettere criticamente sul rapporto tra innovazione tecnologica, cultura e società contemporanea.



Figura 2: La linea del tempo "fisica" realizzata in classe

3.3 Esempi di programmazione realizzati al primo anno

Nel primo anno si è scelto di non approfondire eccessivamente gli aspetti tecnici della programmazione, per evitare possibili resistenze da parte di studenti che spesso scelgono il liceo classico per il minore interesse verso matematica e discipline affini. Al contrario, l'Intelligenza Artificiale ha suscitato fin da subito una forte curiosità e partecipazione.

Dopo una prima esperienza di alcuni studenti con Scratch, il percorso è iniziato con attività di coding unplugged attraverso le carte CodyRoby, per poi passare alla programmazione visuale a blocchi con Scratch. Questo ambiente è stato utilizzato sia per l'apprendimento dei concetti fondamentali della programmazione sia per successive applicazioni legate all'Intelligenza Artificiale.

Partendo dalle basi, gli studenti hanno realizzato programmi che utilizzavano variabili, operatori logici e matematici, strutture di selezione ("se… allora… altrimenti…") e diverse tipologie di cicli. Sono stati inoltre sviluppati videogiochi, quiz, animazioni e piccole rappresentazioni collegate alle discipline umanistiche. Nella progettazione dei giochi sono stati introdotti concetti come casualità, punteggi, vite e gestione delle variabili, rendendo le attività più coinvolgenti e motivanti.

Successivamente sono stati sperimentati strumenti come Machine Learning for Kids e Teachable Machine, integrandoli in progetti realizzati con Scratch per mostrare concretamente il funzionamento delle reti neurali e dell'apprendimento automatico.

Per l'anno successivo è previsto l'utilizzo di linguaggi derivati da Scratch applicati a semplici robot, così da visualizzare in modo più immediato gli effetti delle istruzioni programmate.

L'esperienza ha evidenziato come alcuni studenti abbiano incontrato difficoltà nei concetti di base, soprattutto per la scarsa abitudine al pensiero astratto e alla progettazione. Ciò conferma il valore educativo della programmazione nello sviluppo del pensiero critico, dell'analisi e

dell'organizzazione. La maggior parte della classe, tuttavia, ha lavorato con successo, mostrando competenze logiche spesso riscontrabili anche in matematica e nelle lingue classiche. Inoltre, Scratch si è rivelato utile per favorire l'inclusione linguistica di una studentessa recentemente arrivata in Italia, grazie alla disponibilità dell'interfaccia in più lingue. Le attività ludiche hanno infine aumentato la motivazione, spingendo gli studenti ad affrontare concetti più complessi in modo naturale.

Programma uno sprite che disegna (usando l'estensione PENNA):
a) un quadrato;
b) un triangolo equilatero;
c) un pentagono regolare.
Ogni figura deve avere lato di 100 passi e un colore diverso oppure un colore casuale.

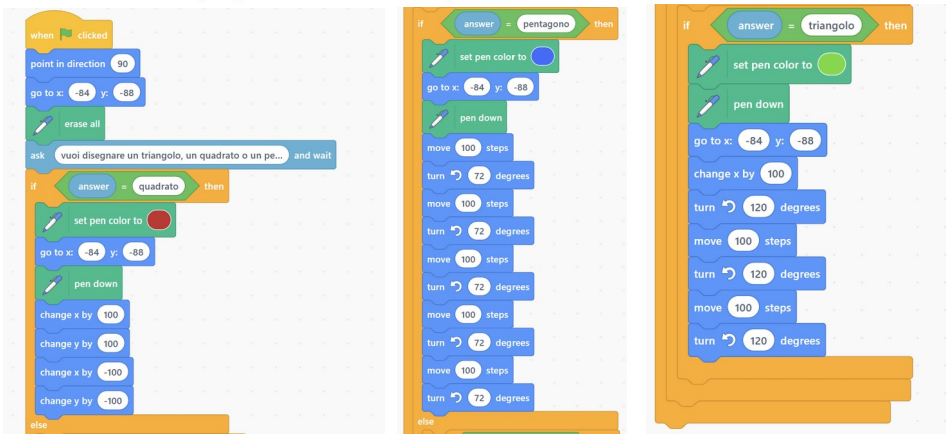


Figura 3 – listati di alcuni programmi (esercizio PENNA)

Crea un semplice videogioco "Acchiappa l'oggetto" con punteggio e tempo.
Realizza un gioco in cui:
a) Uno sprite (es. cestino) si muove a destra e sinistra con le frecce.
b) Un oggetto cade dall'alto in posizione casuale.
c) Se il cestino tocca l'oggetto: +1 punto; l'oggetto riparte dall'alto.
d) Se il cestino non tocca l'oggetto: 0 punti; l'oggetto riparte dall'alto dopo aver toccato il bordo inferiore dello stage.
Il gioco dura 30 secondi. Alla fine mostra "Game Over" e il punteggio finale.
Eventuali punti extra per: variare la velocità dell'oggetto che cade; aggiungere suoni quando si prende o si perde l'oggetto.

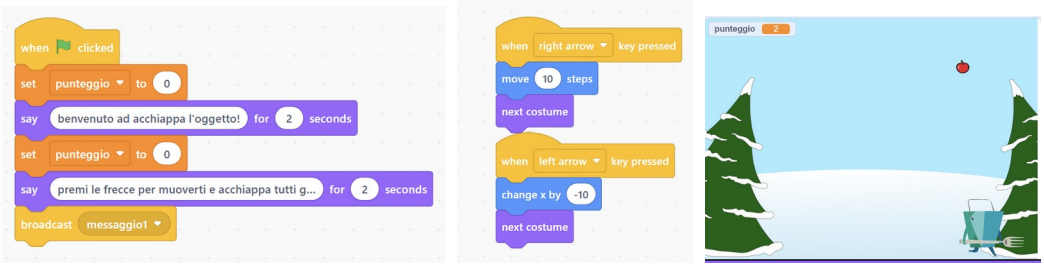


Figura 4 – listati di alcuni programmi (esercizio VIDEOGIOCO)

3.4 Contenuti dell'IA presentati al primo anno

Dal punto di vista tecnico, i principali contenuti sull'Intelligenza Artificiale hanno riguardato l'addestramento di reti neurali tramite Teachable Machine e Machine Learning for Kids, focalizzati rispettivamente su immagini e parole. Questi strumenti hanno consentito agli studenti

di comprendere i concetti di addestramento e valutazione dei modelli, sperimentando direttamente problematiche come bias e completezza dei dataset.

A livello generale, è stata presentata una breve storia dell'IA, evidenziandone le origini e coinvolgendo gli alunni nella ricerca di informazioni su personaggi e invenzioni significative, successivamente inserite in una linea del tempo realizzata durante il percorso.

Sono stati inoltre introdotti aspetti legati alla creatività dell'IA attraverso esempi e attività ispirate al test di Turing, con l'obiettivo di sviluppare capacità di osservazione e senso critico.

Particolare attenzione è stata dedicata al prompting, illustrando regole e strategie per formulare richieste efficaci all'IA. Gli studenti hanno sperimentato tali tecniche nella generazione di immagini, testi e ricerche su argomenti di studio, valutandone pertinenza e affidabilità.

Un ulteriore tema approfondito è stato quello dei big data e della data science. Partendo da esempi concreti e accessibili, gli studenti hanno utilizzato strumenti online per raccogliere e analizzare dati. In assenza di conoscenze avanzate di statistica, sono stati privilegiati strumenti intuitivi e trasversali, come i portali ISTAT, Our World in Data e INGV, per esplorare dati legati a fenomeni sociali e naturali.

4 Conclusioni

Chiaramente le riflessioni dal punto di vista della sperimentazione didattica sono molte e variegate: da un lato, gli alunni hanno chiaramente e ripetutamente mostrato che la scelta dell'indirizzo sperimentale è stata ragionata e dettata da uno specifico interesse per l'IA; a volte è stato sottolineato il fatto che alcuni si aspettassero un utilizzo e studio ancora più massiccio dell'IA.

Dal punto di vista della consapevolezza dei rischi connessi all'uso dell'IA spesso gli studenti si sono mostrati più attenti, consapevoli e critici di molti degli adulti che si incontrano.

Dal punto di vista didattico occorre "dosare" elementi di uso dell'IA, conoscenze e nozioni teoriche, approccio critico, elementi di etica, elementi più tecnici e far comprendere che "dietro" un sistema di IA c'è un'architettura molto complessa che richiede grande preparazione a chi vuole in qualche modo "governarla" e non solo utilizzarla in maniera approssimativa.

Da questo punto di vista è fondamentale l'aspetto più tecnico della programmazione informatica, anche ancora a livello di linguaggio Scratch, per permettere agli studenti di intuire la complessità che sta "dietro le quinte" di applicazioni che utilizzano senza porsi domande e problemi.

In linea di massima si può confermare che, normalmente, le difficoltà incontrate da alcuni studenti nella programmazione vanno di pari passo con difficoltà incontrate nelle lingue classiche, soprattutto nella traduzione delle versioni, e in matematica, così come avviene anche per le eccellenze: può esserci un minimo scarto, ma solitamente gli studenti non solo più studiosi, ma più intuitivi e logici eccellono sia in programmazione, sia in matematica, sia in latino e greco.

Risorse e riferimenti bibliografici

- Guida a Scratch (DSchola) https://www.associazionedschola.it/programmoinvento-ia/le-guide-per-il-coding-a-scuola/
- Linee guida per l'introduzione dell'Intelligenza Artificiale nelle istituzioni scolastiche – Allegato al DM n. 166 del 09/08/2025 https://www.mim.gov.it/-/pubblicate-le-linee-guida-per-l-introduzione-dell-intelligenza-artificiale-nelle-istituzioni-scolastiche-allegato-al-dm-n-166-del-09-08-2025
- Liceo Classico e Digital Humanities: un'integrazione in fase di avvio https://infouma.hypotheses.org/3643
- Indicazioni nazionali per i Licei https://www.mim.gov.it/-/pubblicato-il-testo-delle-nuove-indicazioni-nazionali-per-i-licei-

Laboratori

Insegnare informatica nel primo ciclo: dalle Indicazioni alle attività in classe

Michael Lodi[1,3], Agnese Del Zozzo[2,3], Alberto Montresor[2,3], and Giorgia Bissoli[2,4]

[1] Alma Mater Studiorum – Università di Bologna
michael.lodi@unibo.it
[2] Università di Trento
{agnese.delzozzo, alberto.montresor, giorgia.bissoli}@unitn.it
[3] Laboratorio CINI Informatica e Scuola
[4] Verona FabLab

1 Ordine e grado di scuola

Il laboratorio è rivolto principalmente a insegnanti della scuola primaria e della scuola secondaria di primo grado, nonché a formatori. Le attività sono state progettate originariamente per la scuola primaria, ma vengono proposte in modalità flessibile per gli ultimi anni della primaria e i primi anni della secondaria, modulando i vincoli, il livello di approfondimento e il grado di generalizzazione richiesto.

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio presenta, attraverso l'esperienza diretta, le tre aree dell'Informatica introdotte dalle Indicazioni Nazionali 2025 nella disciplina Matematica: programmazione, informazione e dati, algoritmi [5].

I partecipanti provano attività tratte da una guida (disponibile con licenza CC BY-NC-SA 4.0) per docenti del primo ciclo [4, 3], derivata da una guida editoriale per la primaria [1], pubblicata insieme a un fascicolo operativo [2].

Gli obiettivi del laboratorio sono molteplici: separare l'Informatica dalle competenze digitali; esplicitare i concetti di programmatore, linguaggio di istruzioni ed esecutore; distinguere tra informazione, dati e loro rappresentazione; accompagnare il passaggio dall'esecuzione di istruzioni alla progettazione di programmi e algoritmi.

Le attività traducono in pratica alcuni degli obiettivi previsti dalle Indicazioni Nazionali: scrivere, verificare e correggere semplici programmi; scegliere simboli e convenzioni per rappresentare informazioni; descrivere algoritmi in modo preciso rispetto alle capacità e alle istruzioni comprese da uno specifico esecutore.

Dopo un'introduzione alle Indicazioni Nazionali, alla guida e ai principali concetti informatici coinvolti, il laboratorio si articola nelle seguenti attività:

1. **Programmatori, esecutori e tangram**. A coppie, una persona assume il ruolo di programmatore e realizza una figura con i pezzi del tangram; senza poterla mostrare, deve descriverla a un esecutore, che cerca di ricostruirla seguendo le istruzioni. Nei diversi turni viene progressivamente ridotta la possibilità di fare domande, fino a simulare un esecutore che non può chiedere chiarimenti. Il confronto tra le figure fa emergere ambiguità e imprecisioni e porta i partecipanti a riflettere sulla necessità di un linguaggio condiviso, in cui le istruzioni abbiano un significato preciso e non ambiguo.

2. **Luci di Natale: rappresentare informazioni**. Alice e Bob devono comunicare usando lettere luminose sulle proprie case. I gruppi confrontano diversi codici per rappresentare un insieme

limitato di messaggi, verificando quali permettono di comunicare senza ambiguità e confrontandone il costo in termini di numero di lettere da accendere. I gruppi progettano quindi un nuovo codice. Il confronto finale fa emergere la distinzione tra informazione, dati e loro rappresentazione fisica, insieme alla necessità di condividere le regole che permettono di interpretare correttamente un codice.

3. **Gruccia-bilancia: trovare il più pesante**. Ogni gruppo deve trovare il più pesante fra quattro sacchetti opachi e apparentemente identici, utilizzando una gruccia come bilancia e potendo scegliere solo tra le istruzioni previste da un "Manuale delle istruzioni" limitato. A rotazione, una persona propone la strategia, una la esegue alla lettera e una controlla che vengano rispettati i vincoli. Il gruppo prova e verifica la propria procedura anche su configurazioni diverse (pesi diversi, un numero diverso di sacchetti); il confronto finale fa emergere la distinzione tra problema e algoritmo, il ruolo dell'esecutore, la correttezza della strategia e la possibilità di generalizzarla introducendo forme di ripetizione.

Il laboratorio si conclude con un momento di debriefing che mette a confronto i ragionamenti emersi nelle diverse attività e offre una lente interpretativa per riconoscere e rendere espliciti i principi dell'Informatica presenti in pratiche didattiche già diffuse nella scuola, in particolare nell'insegnamento della matematica. Si riflette su che cosa significhi descrivere una procedura per un esecutore che dispone solo di determinate istruzioni, su come riconoscere e ridurre le ambiguità, su come verificare che una strategia funzioni non solo in un caso particolare ma in generale, su come rappresentare le informazioni e sul ruolo della prova, dell'errore e della revisione nel debugging. L'obiettivo è aiutare i partecipanti a distinguere e formalizzare concetti informatici che emergono in attività di rappresentazione, uso di linguaggi simbolici, descrizione di procedure e risoluzione di problemi, mettendo anche in evidenza la differenza tra eseguire una procedura e progettarla. In conclusione, i partecipanti discutono di come adattare una delle attività alla propria classe, modulando i vincoli e il livello di formalizzazione in funzione dell'età.

3 Riferimento a materiali

La guida è disponibile su https://insegnareinformatica.github.io/.

Bibliografia

[1] Giorgia Bissoli, Agnese Del Zozzo, Michael Lodi, and Alberto Montresor. Sulle orme di Milù – Guida di Informatica per l'insegnante. Guida docente, Mondadori Education, Milano, ISBN 9791220419833, 2026. https://www.mondadorieducation.it/catalogo/sulle-orme-di-milu-0074279/.

[2] Giorgia Bissoli, Agnese Del Zozzo, Michael Lodi, and Alberto Montresor. *Sulle orme di Milù – Informatica*. Mondadori Education, Milano, 2026. Fascicolo per sussidiario. https://www.mondadorieducation.it/catalogo/sulle-orme-di-milu-0074279/.

[3] Michael Lodi, Agnese Del Zozzo, Alberto Montresor, and Giorgia Bissoli. *Aprite il sussidiario*: dalla produzione editoriale a una guida "open" per insegnare Informatica nel primo ciclo. In Violetta Lonati and Gabriele Pozzan, editors, *ITADINFO 2026. Quarto Convegno Italiano sulla Didattica dell'Informatica*, 2026.

[4] Michael Lodi, Agnese Del Zozzo, Alberto Montresor, and Giorgia Bissoli. Insegnare Informatica - Una guida per docenti del primo ciclo, 2026. https://insegnareinformatica.github.io/.

[5] Ministero dell'Istruzione e del Merito. Decreto 9 dicembre 2025, n. 221. Regolamento recante Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione, 2026. Gazzetta Ufficiale, Serie Generale n. 21, 27 gennaio 2026. https://www.mim.gov.it/documents/20182/10554370/curricolo_web.pdf/.

Come far capire cos'è un automa

Enrico Nardelli
Università di Roma "Tor Vergata
nardelli@mat.uniroma2.it

1 Ordine e grado di scuola

Scuola secondaria di primo grado (docenti di Tecnologia e Matematica). Scuola secondaria di secondo grado (docenti di Matematica), Liceo Scientifico Opzione Scienze Applicate (docenti di Informatica).

2 Argomento del laboratorio e descrizione delle attività

Il ruolo dell'esecutore automatico o automa nella comprensione concettuale dell'informatica è spesso trascurato, con il rischio, per la formazione scolastica, di una comprensione parziale dell'informatica come disciplina scientifica.

L'obiettivo di questo laboratorio è quindi di preparare gli insegnanti allo svolgimento di una o più lezioni che guidino gli studenti alla comprensione di questo concetto, essenziale per acquisire una piena consapevolezza della completa meccanicità con cui i programmi vengono eseguiti.

Il laboratorio è pertanto un'introduzione progressiva al concetto di esecutore automatico (o di automa) presentando una serie di esecutori automatici che hanno un livello crescente di "potere computazionale": automa logico, automa a stati finiti, macchina a registri. Il passaggio da un livello al successivo è scandito dalla presenza di problemi che un esecutore non può risolvere.

2.1 Descrizione delle attività

Per ogni tipo di automa viene introdotta la sua definizione e presentata una sua realizzazione grafica sulla quale i discenti opereranno allo scopo di capirne il funzionamento, sperimentando in prima persona le attività destinate agli studenti. Ogni successivo automa viene caratterizzato dalla presenza di una nuova "funzione" che lo rende superiore al precedente. In questo percorso si riflette anche sulla sostanziale equivalenza tra un automa e il linguaggio che l'automa è direttamente in grado di eseguire e sull'importanza della rappresentazione di ciò su cui l'automa deve operare.

2.2 Obiettivi di apprendimento

Si fa riferimento in via prioritaria al seguente obiettivo di apprendimento al termine della scuola secondaria di primo grado (SS-1) delle Nuove Indicazioni Nazionali per il primo ciclo (NIN-1):
Tecnologia: *Comprendere i princìpi fondamentali* [...] *del funzionamento di sistemi e dispositivi informatici*.

Si osserva anche che tra le conoscenze richieste per la Matematica al termine della SS-1 compare *Esecuzione rigorosa di un algoritmo*, per quale è essenziale l'acquisizione del concetto di automa, e che le NIN-1 richiamano esplicitamente la collaborazione tra i docenti di Matematica e di Tecnologia per lo svolgimento della parte di Informatica.

In aggiunta, l'obiettivo di apprendimento *Esecuzione automatica di algoritmi* è presente anche nella bozza di Nuove Indicazioni Nazionali per la Scuola Secondaria di secondo grado per Matematica, mentre l'obiettivo di apprendimento *Introduzione* [...] *alle funzioni fondamentali*

realizzate dai suoi [*di un calcolatore*] *componenti* è presente nella stessa bozza per Informatica nel Liceo Scientifico Opzione Scienze Applicate.

2.3 Scansione temporale delle attività

Sessione di 90 minuti strutturata con le sotto-sezioni descritte nel seguito. Il lavoro viene svolto in gruppetti o tutti insieme, a seconda del numero di partecipanti e del loro livello (secondaria di primo o secondo grado) e profilo (matematica, tecnologia, informatica). Nella parte finale di ogni sotto-sezione tranne la prima viene dedicato qualche minuto a discutere eventuali difficoltà che potrebbero incontrare gli studenti e possibili meccanismi di verifica dell'apprendimento.

Introduzione (5 min): l'informatica come elaborazione, cioè calcolo, automatico di rappresentazioni e la necessità di capire cos'è l'automazione di un calcolo, inteso in senso simbolico e non strettamente numerico.

Automa logico (15 min): presentazione della struttura di un automa logico e del suo meccanismo di funzionamento, attraverso un esempio per riconoscere la somiglianza o differenza tra due oggetti; poi esecuzione di due esercizi con due automi diversi ma basati su (cioè, isomorfi a) la morra cinese. Ciò permette di mettere in luce la caratteristica di "esecuzione senza comprensione del significato" che caratterizza un automa. La presenza di un problema nel quale bisogna contare se c'è un numero pari di blocchi rossi in una serie di lunghezza non nota di blocchi rossi o neri richiede un automa più potente.

Automa a stati finiti (20 min): presentazione della struttura di un automa a stati finiti e del suo meccanismo di funzionamento, tramite un esempio informale; poi svolgimento dell'esercizio che risolve il problema del conteggio di parità. Viene sottolineata la "assenza di consapevolezza" da parte dell'automa sulla lunghezza della sequenza mentre esegue questo calcolo. La presenza di un problema in cui bisogna contare quanti blocchi rossi sono presenti in una sequenza di blocchi di lunghezza non nota richiede un automa più potente.

Macchina a registri (50 min): presentazione della struttura di una semplicissima macchina a registri (con un insieme di sole 4 istruzioni) e del suo meccanismo di funzionamento, tramite l'illustrazione del comportamento della macchina per l'esecuzione di ogni istruzione; poi svolgimento dell'esercizio che risolve il problema del conteggio del numero di blocchi rossi. Riflessione sul potere computazionale di tale macchina, che avviene in assenza di comprensione e di consapevolezza.

3 Riferimento a materiali

Il materiale didattico necessario sono cancelleria di uso comune oltre a schede operative, cartelloni e gettoni, usati per rappresentare e operare fisicamente con i vari modelli di automa. Per ulteriori dettagli si può contattare l'autore all'indirizzo sopra indicato.

4 Bibliografia

[1] Video "Alla scoperta dell'informatica" per RAI Scuola, nell'ambito del progetto "Programma il Futuro", https://peertube.uno/w/2ndTmJEvGrS45aNE3msHJJ

[2] Sezione 4.2 (Automa e linguaggi) del volume "La rivoluzione informatica" dell'Autore, Edizioni Themis, https://edizionithemis.it/catalogo/digitale-e-societa/la-rivoluzione-informatica

Insegnare a programmare componendo grafiche con PyTamaro

Luca Chiodini, Joey Bevilacqua, and Matthias Hauswirth

Software Institute, Università della Svizzera italiana, Lugano, Switzerland
luca.chiodini@usi.ch, joey.bevilacqua@usi.ch, matthias.hauswirth@usi.ch

1 Ordine e grado di scuola

Scuola secondaria di secondo grado.

2 Argomento del laboratorio e descrizione delle attività

2.1 Descrizione del laboratorio

I computer nelle scuole sono confinati a calcolare "la somma dei quadrati dei primi venti numeri dispari" [4], notarono Papert e Solomon oltre 50 anni fa. Come risposta, nel corso degli anni sono stati proposti temi diversi che vanno oltre i domini più classici come la matematica per includere, ad esempio, elementi multimediali [3]. È possibile insegnare concetti fondamentali della programmazione, quali la scomposizione di problemi e l'astrazione, in maniera avvincente creando grafiche utilizzando semplici costrutti Python?

Nel laboratorio LuCE dell'Università della Svizzera italiana abbiamo sviluppato una libreria chiamata PyTamaro [2] che permette di insegnare a programmare componendo grafiche, usando semplici costrutti del linguaggio Python, con un occhio di riguardo per l'insegnamento dei principi fondamentali dell'informatica quali astrazione e scomposizione di problemi. La libreria permette di comporre grafiche complesse combinando poche funzioni ed è disponibile in più lingue tra cui l'italiano e l'inglese.

Collaborando, docenti di scuola superiore e ricercatori hanno sviluppato centinaia di attività diverse che sono ora disponibili liberamente su una piattaforma online usata attivamente in varie scuole [1, Cap. 7]. Si possono eseguire programmi Python e creare grafiche e animazioni direttamente nel browser, senza la necessità di creare alcun account.

In questo laboratorio, programmeremo insieme alcune grafiche e animazioni tra cui la bandiera italiana, un semaforo e Pac-Man. Per le prime attività, programmeremo in versione unplugged, usando delle carte chiamate "TamaroCards" [1, Cap. 6] per comporre le nostre grafiche in modo tangibile, prima di implementarle in Python.

2.2 Scansione temporale delle attività e obiettivi di apprendimento

- Introduzione (10 minuti)
  - Introduzione al laboratorio (5 min)
  - Motivazione per la programmazione di grafiche (5 min)
- Attività 1: Bandiera italiana (20 minuti totali)
  Concetti di programmazione affrontati: chiamata di funzione, uso di funzioni da una libreria, annidamento di più chiamate di funzioni, riconoscimento di sottoproblemi, equivalenza di più soluzioni.

  - Programmazione unplugged con TamaroCards della bandiera italiana (10 min)
  - Programmazione in Python della bandiera italiana (10 min)

- Attività 2: Semaforo (30 minuti totali)
  Concetti di programmazione affrontati: riconoscimento di sottoproblemi, definizione di funzione, astrazione, parametrizzazione.
  - Programmazione unplugged con TamaroCards di un semaforo a tre luci (15 min)
  - Programmazione in Python di un semaforo a tre luci (15 min)
- Attività 3: Pac-Man (30 minuti totali)
  Concetti di programmazione affrontati: definizione di funzione, astrazione, parametrizzazione, liste, ripetizione (ad esempio tramite cicli).
  - Programmazione in Python di Pac-Man, versione statica (15 min)
  - Programmazione in Python di Pac-Man, versione animata (15 min)

2.3 Modalità di interazione

Le parti di programmazione unplugged sono svolte a piccoli gruppi di due-tre insegnanti. Le parti di programmazione in Python offrono spunti concreti per attività da svolgere in classe. Ciascun insegnante programma sul proprio dispositivo, eventualmente aiutato/a da uno dei moderatori. A conclusione di ciascuna attività di programmazione verrà discussa collettivamente una tra le possibili soluzioni, ricollegandosi agli obiettivi di apprendimento indicati sopra, espressi in termini di concetti di programmazione affrontati.

3 Riferimento a materiali

- Piattaforma/IDE per programmare nel browser in Python: `pytamaro.si.usi.ch/playground`
- Libreria PyTamaro open-source installabile ovunque: `github.com/LuCEresearchlab/pytamaro`
- Centinaia di attività di programmazione usando PyTamaro per creare grafiche: `pytamaro.si.usi.ch/activities/luce`
- TamaroCards per iniziare a programmare unplugged: `pytamaro.si.usi.ch/tamarocards`

Bibliografia

[1] Luca Chiodini. Teaching introductory programming using graphics as a domain. 2025. URL: `https://n2t.net/ark:/12658/srd1332702`.

[2] Luca Chiodini, Juha Sorva, and Matthias Hauswirth. Teaching programming with graphics: Pitfalls and a solution. In *Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E*, pages 1–12, 2023. `doi:10.1145/3622780.3623644`.

[3] Mark J. Guzdial and Barbara Ericson. *Introduction to Computing and Programming in Python: A Multimedia Approach*. Pearson, fourth edition.

[4] Seymour A. Papert and Cynthia Solomon. Twenty things to do with a computer. URL: `https://dspace.mit.edu/handle/1721.1/5836`.

Colori, Cassetti e Impronte: Come Funzionano le Hash

Maria Cristina Carrisi[1,4], Michael Lodi[2,4],
Simone Martini[2,4], and Maria Angela Pellegrino[3,4]

[1] Università degli Studi di Cagliari
mariacri.carrisi@unica.it
[2] Alma Mater Studiorum – Università di Bologna
michael.lodi@unibo.it, simone.martini@unibo.it
[3] Università degli Studi di Salerno
mapellegrino@unisa.it
[4] Laboratorio CINI Informatica e Scuola

1 Ordine e grado di scuola

Il laboratorio è rivolto a insegnanti della scuola primaria e secondaria di primo grado; le attività sono tarate sulle conoscenze attese per una classe quinta primaria o prima secondaria di primo grado.

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio (90 minuti) introduce le funzioni di hash come algoritmi che trasformano dati di lunghezza variabile in impronte di lunghezza fissa. Il contesto è quello della memorizzazione, della ricerca dei dati e della sicurezza informatica: protezione delle password, controllo dell'integrità di file e messaggi e blockchain. Il percorso non parte da una definizione formale, ma da problemi concreti; ogni soluzione viene provata, se ne osservano proprietà e limiti, quindi la si modifica. Questo approccio "basato sulla necessità" combina attività unplugged, manipolazione e brevi verifiche in Snap!, in continuità con quanto già fatto in attività di crittografia per la scuola primaria [2].

2.1 Struttura e tempi

1. Cassetti (10 minuti). I gruppi devono collocare oggetti in una cassettiera numerata e ritrovarli usando soltanto il nome. Associare il cassetto all'iniziale rende visibili determinismo e sintesi, ma anche le collisioni: nomi diversi possono indicare lo stesso vano.

2. Hash a colori (20 minuti). Scelte 4–5 lettere che consentano di creare parole di senso compiuto di 4–6 lettere, si associa, mediante tabella, un colore a ciascuna lettera. Per ogni lettera della parola si versa esattamente 1 ml della tempera corrispondente e si conserva 1 ml del miscuglio finale. Gruppi diversi ottengono così il medesimo risultato per lo stesso input: una sintesi di lunghezza fissa (un solo colore finale ma anche una stessa quantità di liquido). Dal miscuglio è inoltre difficile ricostruire le dosi iniziali. Una piccola variazione della parola può produrre un colore diverso. Con *MALE*, *MELA* e *LAME* emerge però una collisione: usando gli stessi colori nelle stesse quantità, gli anagrammi producono la medesima impronta.

3. Posizione (15 minuti). Per distinguere gli anagrammi, la dose dipende dalla posizione: 1 ml per la prima lettera, 2 ml per la seconda e così via. In *MALE* e *MELA* gli stessi colori ricevono ora pesi diversi e i miscugli non coincidono. Il passaggio rende concreto il ruolo della posizione e prepara la formalizzazione numerica attraverso il richiamo al valore posizionale delle cifre.

4. Hash numeriche (25 minuti). Le lettere vengono associate a numeri ($A = 1$, $B = 2$, ecc.). I gruppi confrontano tre algoritmi: somma semplice, somma con pesi posizionali e resto della divisione. I pesi possono distinguere gli anagrammi; il resto riporta l'output a un insieme di dimensione fissata e fa riemergere inevitabilmente le collisioni. Una tabella dei resti e l'ambiente Snap! permettono di effettuare verifiche su esempi che coinvolgono numeri grandi, senza trasformare le competenze nel calcolo in un prerequisito bloccante.

5. Debriefing (20 minuti). I gruppi ricostruiscono la sequenza: problema, algoritmo, proprietà, criticità e modifica. Discutono inoltre le differenze tra gli algoritmi didattici e le funzioni crittografiche reali. Password, integrità dei file e blockchain restano applicazioni conclusive, non attività aggiuntive, così da preservare il tempo di riflessione.

2.2 Interazione, accessibilità e preparazione

Le persone partecipanti lavorano in gruppi di 4–6 con ruoli a rotazione (lettura, dosaggio, annotazione e restituzione). Colori e contenitori sono sempre identificati anche da lettere, numeri e simboli; sebbene sia possibile scegliere i colori in modo da agevolare chi ha difficoltà nella percezione cromatica, è opportuno affiancare dei simboli ai colori. Si può applicare lo stesso algoritmo con gettoni numerati e quantità annotate, confrontando poi i risultati numerici. La comprensione non dipende, quindi, dal riconoscimento del colore. I facilitatori preparano in anticipo tempere diluite, bicchieri, siringhe graduate, schede, una cassettiera con piccoli oggetti e un computer con proiettore per Snap!.

Il percorso è stato sperimentato nelle classi quinte della primaria, anche con alunni con disabilità, disturbi dell'apprendimento e altri bisogni educativi. Le osservazioni hanno mostrato forte partecipazione nella fase manipolativa e sorpresa produttiva davanti alla collisione fra anagrammi; hanno anche evidenziato la necessità di predisporre il materiale prima della lezione e di rafforzare il passaggio alla divisione con resto.

3 Riferimento a materiali

Le schede operative e l'ambiente Snap! sono disponibili al link: https://bigideascryptok12.bitbucket.io/hash/ita.html.

Una prima analisi degli esiti della sperimentazione in classe delle attività proposte è presentata in [1].

Attività correlate sull'applicazione delle funzioni di hash per lo stesso livello scolastico sono descritte in [3].

Bibliografia

[1] Maria Cristina Carrisi, Michael Lodi, Simone Martini, and Maria Angela Pellegrino. Funzioni di hash nella scuola primaria: una sperimentazione unplugged tra informazione, algoritmi e sicurezza digitale. In Violetta Lonati and Gabriele Pozzan, editors, *ITADINFO 2026. Quarto Convegno Italiano sulla Didattica dell'Informatica*, 2026.

[2] Michael Lodi, Maria Cristina Carrisi, and Simone Martini. Big ideas of cryptography in primary school. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2024, pages 206–212, New York, NY, USA, 2024. Association for Computing Machinery. https://doi.org/10.1145/3649217.3653548.

[3] Maria Angela Pellegrino and Lorenzo Guasti. Introducing blockchains at school without computers: Hands-on sense-making for young learners. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, CHI 2026, pages 1–21. ACM, 2026. https://doi.org/10.1145/3772318.3791808.

“Questo colore non c’è sui blocchi!” Insegna un colore nuovo al tuo robot con il ML (e sopravvivi a Python!)

Laura Cesaro[1], Daniel Fusaro[1], e Emanuele Menegatti[1]

Dipartimento di Ingegneria dell’Informazione, Università degli Studi di Padova, Italia
laura.cesaro@phd.unipd.it
fusarodani@dei.unipd.it
emanuele.menegatti@unipd.it

1 Ordine e grado di scuola

Scuola secondaria di primo e secondo grado.

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio unisce robotica educativa ed Edge-AI per mostrare il funzionamento del Machine Learning attraverso un approccio concreto e hands-on. La domanda guida dell’attività è: “Può un robot imparare a riconoscere colori che non ha mai visto prima?”. Contrariamente ai classici approcci in cui i modelli vengono addestrati su cloud o computer e poi trasferiti al robot, in questo laboratorio i partecipanti sperimentano l’intero flusso di lavoro direttamente sul microcontrollore del robot LEGO™ SPIKE Prime operando interamente offline e sperimentando l’AI come sistema autonomo embedded. I limiti della programmazione a blocchi, che riconosce solo un insieme predefinito di colori, diventano un’opportunità didattica: per superarli, si giustifica in modo naturale l’adozione della programmazione testuale in Python, necessaria per accedere ai dati grezzi del sensore (valori RGB 0–1023) e addestrare una piccola rete neurale in grado di distinguere nuove sfumature.

L’esperienza, già sperimentata con studenti della scuola secondaria nell’ambito di Science4All [6], ha mostrato come il passaggio dalla programmazione a blocchi a Python, quando nasce da un bisogno concreto e comprensibile, possa diventare un forte elemento motivazionale anche per alunni inizialmente poco interessati alla programmazione, favorendo partecipazione, inclusione e parità di genere nei risultati [2]. Il laboratorio ha una durata complessiva di 90 minuti e si articola in quattro fasi sequenziali, che ricalcano l’intero ciclo di vita di un modello di Machine Learning.

2.1 Fasi di svolgimento

Introduzione e raccolta dati Dopo una breve introduzione al Machine Learning come processo guidato dai dati, l’attività prende avvio da una domanda concreta: il robot riesce a distinguere un colore che non rientra nel suo set predefinito? I partecipanti, divisi in gruppi, utilizzano il robot LEGO™ SPIKE Prime per acquisire i valori RGB di vari cartoncini colorati (es. verde, marrone, arancione e “nessun colore/sfondo”). Ogni robot salva localmente sul proprio hub dei file `.csv` contenenti il dataset di addestramento. Questa fase permette di riflettere sull’importanza della qualità dei dati e di scoprire la natura numerica su cui si basano i modelli di Intelligenza Artificiale, favorendo una demistificazione concreta del suo funzionamento.

Addestramento del modello I partecipanti eseguono l'addestramento di una rete neurale (Multilayer Perceptron) interamente a bordo dell'hub del robot, sfruttando una libreria custom in MicroPython (μ-Learn [3]). Partendo da uno script Python di base, configurano i parametri principali (suddivisione dei dati in training e validation set, definizione delle classi di colore) e avviano il training, osservando l'andamento dell'accuratezza epoca per epoca tramite il terminale di programmazione.
Valutazione del modello Il modello viene testato su un subset di dati non utilizzati nella fase precedente (test set). I partecipanti analizzano i risultati calcolando metriche standard derivate dalla matrice di confusione. Questa fase stimola una discussione critica sugli errori del modello (es. confusione tra colori dovuta a condizioni di illuminazione simili o a misurazioni imprecise), sottolineando i limiti e l'affidabilità dei sistemi data-driven.
Deployment e discussione finale Il modello validato viene utilizzato per guidare la logica decisionale del robot: i partecipanti configurano la relazione sensore-azione associando le predizioni del modello a specifici comportamenti autonomi (es. se il modello classifica il colore come verde, il robot esegue una svolta a sinistra). Il laboratorio si chiude con un debriefing per riflettere sull'esperienza e discutere possibili estensioni e varianti dell'attività.

3 Materiali

Kit robotici LEGO™ SPIKE Prime già assemblati (indicativamente uno ogni due partecipanti, per favorire il lavoro di gruppo e il problem-solving collaborativo); computer portatili con il software preinstallato; cartoncini colorati per la raccolta empirica dei dataset RGB.

La libreria open-source in MicroPython (μ-Learn), sviluppata per l'addestramento on-device direttamente sul microcontrollore del robot, è disponibile al seguente indirizzo:

`https://github.com/Bender97/mulearn`

Bibliografia

[1] Luigi Capogrosso, Federico Cunico, Dong Seon Cheng, Franco Fummi, and Marco Cristani. A machine learning-oriented survey on tiny machine learning. *IEEE Access*, 12:23406–23426, 2024.

[2] Laura Cesaro, Daniel Fusaro, Michele Moro, and Emanuele Menegatti. Introducing machine learning to middle-school students with edge-ai in an educational robot. In *Proceedings of the International Conference on Robotics in Education (RiE)*, 2026.

[3] Daniel Fusaro, Laura Cesaro, Michele Moro, and Emanuele Menegatti. μ-learn: A lightweight, on-device machine learning library for educational robotics platforms. In *Proceedings of the International Conference on Robotics in Education (RiE)*, 2026.

[4] Jon Nordby. emlearn-micropython: Machine learning and digital signal processing for micropython. *Journal of Open Source Software*, 10(116):9093, 2025.

[5] Glenn Strong, Nina Bresnihan, and Brendan Tangney. Supporting learners in the transition from block-based to text-based programming: A systematic review. *Journal of Computer Languages*, 84, 2025.

[6] Università degli Studi di Padova. I robot possono imparare a riconoscere dei colori mai visti? Science4All 2025. `https://scuole.science4all.it/attivita/i-robot-possono-imparare-a-riconoscere-dei-colori-mai-visti/`, ultimo accesso: 24/01/2026.

[7] David Weintrop and Uri Wilensky. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education*, 18(1):1–25, 2017.

[8] Shuai Zhu, Thiemo Voigt, Fatemeh Rahimian, and Jeonggil Ko. On-device training: A first overview on existing systems. *ACM Transactions on Sensor Networks*, 20(6):1–39, 2024.

TCPigeon: comprendere TCP attraverso i piccioni viaggiatori

Renzo Davoli

Alma Mater Studiorum, Università di Bologna
DISI: Dipartimento di Informatica Scienza e Ingegneria
renzo@cs.unibo.it

1 Ordine e Grado di scuola

Il laboratorio è rivolto principalmente agli studenti del triennio dell'Istituto Tecnico Tecnologico, indirizzo Informatica e Telecomunicazioni, nell'ambito dell'insegnamento di Sistemi e Reti. L'attività può essere proposta anche come approfondimento nei Licei delle Scienze Applicate.

2 Argomento del laboratorio e descrizione delle attività

L'attività unplugged prende avvio da una semplice ambientazione narrativa: un principe e una principessa sono rinchiusi in due torri lontane e possono comunicare esclusivamente tramite piccioni viaggiatori.

In una prima fase viene introdotto il servizio IP come meccanismo di recapito dei messaggi. Successivamente vengono introdotti eventi che simulano le condizioni reali delle reti: perdita di messaggi, percorsi differenti, ritardi nella consegna e recapito fuori ordine.

Per superare tali difficoltà gli studenti sono guidati a progettare progressivamente meccanismi analoghi a quelli adottati da TCP: suddivisione dei messaggi in segmenti; numerazione dei segmenti; conferme di ricezione; rilevazione delle perdite; ritrasmissione dei segmenti mancanti; ricostruzione ordinata del messaggio.

La discussione finale permette di collegare le soluzioni emerse spontaneamente durante l'attività con i meccanismi effettivamente implementati dal protocollo TCP.

Alcuni partecipanti interpreteranno i personaggi della narrazione, il principe, la principessa, gli ufficiali di comunicazione e i piccioni viaggiatori, mentre il resto della classe contribuirà attivamente allo svolgimento dell'attività formulando ipotesi, suggerendo strategie e discutendo le possibili soluzioni ai problemi di comunicazione che verranno progressivamente introdotti. Tutti i partecipanti saranno infine coinvolti nella discussione conclusiva e nella formalizzazione dei concetti emersi durante la simulazione.

Questo laboratorio fa parte del percorso *Internet Unplugged*, dedicato all'insegnamento dei protocolli di comunicazione mediante simulazioni fisiche e attività collaborative. *Internet Unplugged* è stato progettato e progressivamente perfezionato nel corso di tre edizioni dell'insegnamento "Didattica delle Reti e dei Sistemi" del percorso universitario di formazione iniziale dei docenti delle classi di concorso A-41 e B-16.

Il percorso assume come punto di partenza il protocollo IP, presentato come un servizio postale realizzato tramite messaggi, buste e cassette postali. A partire da questa rappresentazione vengono introdotti sia i protocolli di livello superiore (UDP, TCP, HTTP, SMTP), sia i servizi di livello inferiore necessari a realizzare il trasferimento dei dati.

Le intestazioni dei protocolli sono rappresentate mediante etichette e foglietti applicati ai messaggi, permettendo agli studenti di visualizzare concretamente il concetto di incapsulamento.

L'approccio adottato può essere sintetizzato nella sequenza "play–see–do": gli studenti sperimentano inizialmente i concetti attraverso il gioco e la simulazione, acquisiscono successivamente una comprensione teorica più solida e possono infine riconoscere gli stessi meccanismi nell'analisi di comunicazioni reali.

Obiettivi di apprendimento. Al termine dell'attività gli studenti dovrebbero essere in grado di: descrivere il servizio offerto dal protocollo TCP; distinguere tra comunicazione affidabile e non affidabile; comprendere il ruolo dei numeri di sequenza; comprendere il ruolo delle conferme di ricezione; spiegare la necessità delle ritrasmissioni in presenza di perdite; collegare i meccanismi sperimentati durante l'attività alle comunicazioni reali osservabili mediante strumenti di analisi del traffico.

Durata complessiva del laboratorio: 90 minuti: Introduzione e contestualizzazione: 5 minuti; Presentazione del servizio IP: 15 minuti; Attività TCPigeon: 45 minuti; Concettualizzazione e collegamento ai protocolli reali: 10 minuti; Discussione finale e domande: 15 minuti.

Materiale necessario: videoproiettore; due lavagne (anche a fogli mobili) con relativi pennarelli; fogli e buste utilizzati per rappresentare i messaggi e le intestazioni protocollari; un mazzo di carte da gioco (o un dado).

Lavori correlati

TCPigeon trae libera ispirazione dal celebre documento umoristico RFC 1149, *A Standard for the Transmission of IP Datagrams on Avian Carriers*, pubblicato il 1 aprile 1990[3].

Lo stesso tema viene trattato nella attività 11 (Le tavolette di Pietra) del testo "Computer Science Unplugged" [1]. L'attività è studiata per studenti a partire da 9 anni di età, quindi si focalizza su una comprensione dei problemi collegati senza l'obiettivo di proseguire con lo studio dei protocolli reali.

Anche l'organizzazione inglese "National Centre for Computing Education" (NCCE) nella Lezione 10, The TCP/IP model, della unità di *Computer Network*. [2]

La fascia di età della proposta NCCE è analoga a quella per cui è stata originariamente progettata l'attività. Nel contesto di TCPigeon la simulazione viene invece utilizzata come introduzione a uno studio più approfondito dei protocolli reali TCP/IP, risultando quindi adatta anche agli studenti della scuola secondaria di secondo grado.

Al contrario, nel percorso Internet Unplugged ciascun livello protocollare viene affrontato mediante attività distinte e progressive.

La sperimentazione di attività unplugged di più livelli o di tutti i livelli di TCP-IP può essere prevista come completamento o approfondimento dell'esperienza.

3 Riferimento a materiali

La descrizione delle attività unplugged e il materiale didattico necessario per il loro svolgimento saranno resi disponibili online: `https://rd235.github.io/cse/#!TCPigeon.md`

Riferimenti bibliografici

[1] Tim Bell, Ian H. Witten, and Mike Fellows. *Imparare l'informatica senza alcun Computer: Una collezione di attività per l'introduzione dell'informatica senza l'utilizzo del computer*. CS Unplugged, 2015. Edizione italiana a cura di Renzo Davoli, Giovanni Michele Bianco, Piergiovanna Grossi.

[2] National Centre for Computing Education. The tcp/ip model, 2025. Teach Computing Curriculum, Key Stage 4, Computer Networks, Lesson 10.

[3] David Waitzman. Standard for the transmission of IP datagrams on avian carriers. RFC 1149, apr 1990.

I server: dietro le quinte della rete

Sara Capecchi[1], Daniele Catozzella[2]
[1] Dipartimento di informatica, Università degli Studi di Torino
sara.capecchi@unito.it
[2] IC Frascolla Taranto – docente esperto esterno
daniele.catozzella@gmail.com

1 Ordine e grado di scuola

Docenti di scuola primaria (quarta e quinta) e secondaria di primo grado (prima e seconda).

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio propone i server come punto di osservazione del "dietro le quinte della rete", in linea con le nuove Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione [1], come mostrato nella tabella sinottica dell'allegato.
Il concetto di server, e con esso il modello client/server, permette di introdurre diversi elementi che promuovono, in studenti e studentesse, la costruzione di un corretto modello mentale dell'infrastruttura della rete e di superare la concezione diffusa [2] di una rete composta da servizi tra loro "staccati".
Il laboratorio interviene su questa rappresentazione rendendo osservabili tre elementi che spesso restano impliciti: la risoluzione dei nomi (DNS), l'instradamento e l'indirizzamento (IP) dei dati e l'infrastruttura fisica della rete. Le attività del laboratorio, in questo modo, portano alla luce l'infrastruttura altrimenti invisibile della rete.
Il laboratorio, al tempo stesso, è propedeutico agli obiettivi di cittadinanza digitale inquadrati nel contesto scolastico dal DigComp 3.0 [3] e dall'educazione civica alla cittadinanza digitale [4] e permette di costruire le basi per un collegamento multidisciplinare in grado di realizzare un ponte coerente ed efficace tra le competenze informatiche e le competenze di cittadinanza digitale.
Questo collegamento disciplinare, possibile dal nuovo anno scolastico con le nuove Indicazioni nazionali, permette di costruire un percorso didattico in grado di proporre a studenti e studentesse lo sviluppo di una consapevolezza della rete, delle sue logiche e dei comportamenti responsabili e critici da tenere per la propria sicurezza su basi disciplinari coerenti e concrete.
Durante il laboratorio verranno proposte, in forma sintetica e con una lente metodologica, le stesse attività che i docenti potranno svolgere in classe, operando direttamente sui propri dispositivi o seguendo gli esempi mostrati in plenaria. I tempi indicati sono adattati al tempo di presentazione del convegno, il laboratorio in classe ha una durata di n. 2 ore.

Fase I - (10 minuti)

Una breve introduzione spiega gli obiettivi dell'attività e le caratteristiche generali per poi proporre una domanda stimolo: "quali siti e servizi i ragazzi visitano ogni giorno?". La domanda introduce il tema della concezione della rete e il presupposto cognitivo del percorso: di fronte a un sistema invisibile, chi apprende costruisce comunque un modello mentale e, in assenza di insegnamento esplicito, lo forma per inferenza da ciò che vede, sviluppando misconcezioni. Si inizia dalla risoluzione dei nomi (DNS): attraverso applicativi online si osserva la traduzione del nome di dominio in indirizzo IP e si digita l'indirizzo nel browser. Osservando gli esiti di questa navigazione si chiarisce che: il server viene raggiunto ma il sito richiesto no. In questo modo si facilita la distinzione tra le funzioni: l'indirizzo IP individua il server, il nome individua il sito tra quelli

ospitati da uno stesso server. Lo stesso test, ripetuto da riga di comando su due reti differenti (ad esempio un hotspot mobile), mostra che uno stesso nome di dominio può essere tradotto in indirizzi ip differenti, a seconda del server (resolver) interrogato.

Fase II – (60 minuti)

Si passa successivamente al concetto di instradamento: in coppia, sui propri dispositivi, i partecipanti eseguono il comando tracert verso destinazioni selezionate per contrasto: siti delle grandi piattaforme, che rispondono da copie locali distribuite mediante reti di distribuzione dei contenuti (CDN) e server centralizzati ospitati su un unico server, come università estere. I docenti partecipanti, compilano una tabella con: sito di destinazione, numero di salti (hops) e tempi di risposta.
A questo punto con alcuni applicativi online visuali si effettua la geolocalizzazione degli IP attraversati per costruire una mappa stimata del percorso. La mappa viene verificata attraverso i dati registrati nella tabella che evidenziano i tempi di propagazione del percorso tracert. In questo modo è possibile verificare i risultati visuali proposti dalla mappa attraverso le informazioni ottenute sui tempi di risposta e la distanza tra i server nei diversi salti.

FASE III – (20 minuti)

La terza attività si concentra sul livello fisico attraverso la consultazione di mappe interattive dei cavi sottomarini per mostrare la materialità della rete. In tutte le fasi si sottolinea il punto metodologico che i docenti devono poter governare: il comando tracert mostra solo uno dei percorsi possibili e il modello va presentato a studenti e studentesse dichiarando apertamente i limiti.
Il laboratorio si conclude con alcune domande stimolo che aprono la riflessione sul ruolo centrale di queste attività nello sviluppo delle competenze di cittadinanza digitale e nella costruzione di un curricolo multidisciplinare efficace per il consiglio di classe. Ad esempio: ora che sappiamo che i nostri dati viaggiano attraverso server in paesi diversi e lasciano tracce lungo il percorso, quanto controllo abbiamo realmente su di essi?

3 Riferimento e materiali

E’ necessaria una connessione a internet e un pc. Se il dispositivo è mobile alcune parti del laboratorio potranno essere svolte esclusivamente con applicativi online.
La descrizione dettagliata del laboratorio, le risorse utilizzate e le metodologie per adattare le attività ai diversi gradi scolastici sono raccolti nel documento a questo al link.

Bibliografia

[1] MIM, “Indicazioni Nazionali per il Curricolo Scuola dell’Infanzia e Scuola del Primo ciclo di Istruzione” ,2025 https://www.mim.gov.it/-/indicazioni-nazionali-per-il-curricolo-scuola-dell-infanzia-e-scuole-del-primo-ciclo-di-istruzione, 2025 Accessed: settembre 2026

[2] L. Botturi, L. Addimando “Inside or between devices: the internet in the mind of primary school children” https://www.sciencedirect.com/science/article/pii/S2212868925000042, 2025 Accessed: settembre 2026

[3] Cosgrove, J. and Cachia, R., *DigComp 3.0: European Digital Competence Framework* - Fifth Edition, Publications Office of the European Union, Luxembourg, 2025, https://data.europa.eu/doi/10.2760/0001149, 2025 Accessed: settembre 2026

[4] L. n.92/2029 Introduzione dell’insegnamento scolastico dell’educazione civica, https://www.mim.gov.it/documents/20182/0/Legge_20_agosto_2019_n_92.pdf/91ee6681-888a-fc0e-9383-2ff6b421347c?version=1.0&t=1760709794663, Accessed: settembre 2026

Pensiero computazionale e linguaggi artificiali: introduzione a Python con micro:bit e il supporto critico dell'IA

Barbara Baldi[1], Giuliana Gelsomino[2]
[1] IC Santorre di Santarosa, Savigliano (CN)
baldi.barbara@icsantarosasavigliano.edu.it
[2] IC Virgilio-Gonzaga, Eboli (SA)
giuliana.gelsomino23@gmail.com

1 Ordine e grado di scuola

Scuola secondaria di primo grado.

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio introduce il pensiero computazionale e la programmazione testuale in Python attraverso un approccio mediato dall'Intelligenza Artificiale (IA) generativa. I partecipanti faranno esperienza dei concetti informatici fondanti, variabili, strutture condizionali e iterative, gestione di input/output da sensori e attuatori, utilizzando l'IA non come generatore passivo di soluzioni, ma come tutor che produce codice da leggere, interrogare, modificare e correggere. Il percorso si ispira al modello Use-Modify-Create (UMC, Lee et al., 2011), esteso dalla programmazione a blocchi a quella testuale, che trova nel framework PRIMM (Predict-Run-Investigate-Modify-Make, Sentance, Waite, & Kallia, 2019) una possibile articolazione più fine e calibrata sul contesto scolastico, utile in particolare per scandire con maggiore gradualità la fase di lettura e comprensione del codice (Use) attraverso i momenti di previsione, esecuzione e indagine. L'attività parte da script minimi generati dall'IA per la scheda micro:bit, prosegue con sessioni di debugging guidato (in cui i/le docenti possono sperimentare l'introduzione intenzionale di errori logici come dispositivo didattico) e culmina nella progettazione di un sistema di automazione ambientale, basato sulla simulazione di un'irrigazione automatica. Il laboratorio valorizza l'errore come risorsa cognitiva e dedica spazio alla riflessione critica sui limiti dell'IA generativa (allucinazioni, bias) e sulle implicazioni etico-sociali delle decisioni automatizzate, in linea con le Nuove Indicazioni Nazionali 2025 (NI 2025, D.M. 221/2025) e con il framework DigComp 3.0.

Apertura e inquadramento (10 minuti). Le formatrici introducono il quadro normativo di riferimento (NI 2025, DigComp 3.0) e il razionale pedagogico del laboratorio: la programmazione non come addestramento tecnico, ma come palestra di pensiero computazionale mediata criticamente dall'IA generativa. Si presenta il modello Use-Modify-Create e la sua estensione dalla programmazione a blocchi a quella testuale.

Fase 1 – Primo contatto e logica sensoriale, USE (20 minuti). I gruppi sperimentano l'attività "Hello, Micro:bit!": tramite un prompt guidato, l'IA genera uno script Python che mostra un cuore pulsante sul display LED. Prima dell'esecuzione nel simulatore MakeCode, ogni gruppo legge collettivamente il codice con una griglia di lettura guidata, fornita dalle formatrici, per formulare previsioni sul comportamento del programma prima di osservarne l'esito. Si passa poi all'attività del rilevatore di buio crepuscolare, in cui i/le partecipanti modificano manualmente una soglia numerica nel codice, sperimentando il concetto di variabile.

Fase 2 – Manipolazione consapevole e debugging, MODIFY (35 minuti). Le formatrici introducono intenzionalmente un'anomalia logica in uno script funzionante (l'attività del "semaforo interattivo", in cui un ciclo non si interrompe correttamente). I gruppi praticano il debugging guidato, formulando all'IA domande analitiche sull'errore piuttosto che richieste di soluzioni complete. Segue una discussione plenaria sui limiti dell'IA generativa: cosa l'IA "sa" fare e cosa “non sa”, dove possono comparire allucinazioni o errori sintattici.

Fase 3 – Verso la sintesi, CREATE (20 minuti). I gruppi progettano in autonomia i moduli base di un sistema di monitoraggio dell'umidità con invio dati via radio tra due schede micro:bit, usando l'IA solo per prompt specifici e puntuali. Si introduce concettualmente il progetto finale (sistema di irrigazione automatica simulato), con una breve riflessione guidata sulle implicazioni etico-sociali dei sistemi automatizzati nella vita reale (chi decide le soglie di attivazione, chi è responsabile in caso di errore del sensore).

Chiusura e debriefing (5 minuti). Sintesi collettiva delle competenze attivate, presentazione del materiale per l'approfondimento post-laboratorio e raccolta di domande e feedback dai/dalle partecipanti.

3 Riferimento a materiali

Simulatore online Microsoft MakeCode for micro:bit (https://makecode.microbit.org/), anche con una scheda fisica micro:bit; un PC con connessione ad Internet per l'accesso all'editor MakeCode e a un'interfaccia di IA generativa; un accesso condiviso o individuale a un servizio di IA conversazionale per la generazione e la discussione del codice. [Github, accessibile anche da qr code]

Bibliografia

Cosgrove, J., & Cachia, R. (2025). *DigComp 3.0: European Digital Competence Framework – Fifth Edition*. Publications Office of the European Union.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32–37.

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, 29(2–3), 136–176.

AI-Earthquake Monitoring: Machine Learning con micro:bit

Carla Vitale
Istituto Comprensivo "Lucilio", Sessa Aurunca (CE)
carla.vitale2@scuola.istruzione.it

1 Ordine e grado di scuola

Il laboratorio è rivolto principalmente a docenti della scuola secondaria di primo grado. La proposta può essere adattata anche agli ultimi anni della scuola primaria e al primo biennio della scuola secondaria di secondo grado, modulando il livello di approfondimento dei concetti informatici e il grado di autonomia richiesto nelle fasi operative.

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio propone un percorso operativo per introdurre i principi di base del Machine Learning attraverso l'uso della scheda micro:bit, dell'ambiente CreateAI e della programmazione in MakeCode. L'attività utilizza la simulazione di un monitoraggio del terremoto come scenario applicativo motivante, ma il nucleo del percorso resta la didattica dell'informatica: definizione di classi, esempi di addestramento, distinzione tra dataset di addestramento e dataset di test, costruzione e verifica di un modello, programmazione del comportamento della scheda e comunicazione tra dispositivi.

La domanda guida proposta ai partecipanti è: *come può un sensore digitale distinguere tra un movimento casuale e una scossa riconoscibile?* A partire da questa domanda, i docenti vengono accompagnati nella costruzione di un semplice prototipo didattico di sismografo in scatola, basato su due schede micro:bit. La prima scheda svolge la funzione di rilevatore dei movimenti, mentre la seconda viene configurata come stazione ricevente. Il prototipo non ha la pretesa di riprodurre un sistema scientifico di monitoraggio sismico, ma permette di rendere osservabili alcuni passaggi fondamentali di un sistema informatico: acquisizione di input, classificazione, previsione, output e trasmissione dell'informazione.

In apertura vengono introdotti il problema e i concetti essenziali del Machine Learning. Particolare attenzione è dedicata alla distinzione tra dataset di addestramento e dataset di test: il primo è costituito dagli esempi utilizzati per addestrare il modello, mentre il secondo comprende prove nuove, non impiegate nella fase di addestramento, utili per verificarne l'affidabilità. La distinzione viene presentata in forma operativa attraverso le attività svolte direttamente dai partecipanti durante il laboratorio.

I docenti, organizzati in piccoli gruppi, definiscono tre classi di movimento: quiete, movimento ondulatorio e movimento sussultorio. L'addestramento del modello avviene in CreateAI muovendo la scheda micro:bit nelle diverse direzioni, così da fornire al sistema esempi significativi per ciascuna classe. La quiete corrisponde alla scheda ferma; il movimento ondulatorio viene simulato attraverso oscillazioni laterali; il movimento sussultorio viene simulato attraverso brevi movimenti verticali. In questa fase i partecipanti sperimentano concretamente che la qualità degli esempi forniti incide sul comportamento del modello.

Dopo l'addestramento, i gruppi effettuano una prima fase di test con movimenti nuovi rispetto a quelli usati in precedenza. L'obiettivo non è soltanto verificare se il modello "funziona", ma osservare errori, incertezze e classificazioni non attese. Il confronto sugli esiti permette di

discutere il rapporto tra esempi di addestramento, affidabilità della previsione e miglioramento progressivo del modello. In questo modo il Machine Learning viene presentato come un processo controllabile e interpretabile, non come una procedura automatica difficile da comprendere.

Nella fase successiva il modello viene integrato nella programmazione in MakeCode. La micro:bit rilevatrice viene programmata affinché produca output diversi in base alla classe riconosciuta, ad esempio icone sul display LED, segnali sonori o messaggi. L'attività mette così in relazione il modello addestrato con la programmazione a blocchi, mostrando come una previsione possa essere utilizzata per determinare il comportamento di un dispositivo fisico.

La fase applicativa prevede la costruzione del sismografo in scatola. La micro:bit rilevatrice viene sospesa verticalmente all'interno di una scatola mediante materiali semplici, come molla, elastici, nastro adesivo e cartoncino. Il prototipo consente di simulare movimenti laterali e verticali e di osservare la risposta del modello addestrato. L'abbinamento con una seconda micro:bit, configurata come stazione ricevente, permette inoltre di introdurre la comunicazione radio tra dispositivi e di simulare una piccola postazione di monitoraggio. I partecipanti possono così collegare sensore, modello, algoritmo, output e trasmissione dei dati in un'unica esperienza laboratoriale.

La scansione indicativa dei tempi è la seguente: 10 minuti per l'introduzione al problema, al concetto di Machine Learning e alla distinzione tra dataset di addestramento e dataset di test; 10 minuti per la definizione delle classi e la preparazione dell'ambiente di lavoro in CreateAI; 15 minuti per l'addestramento del modello attraverso i movimenti della micro:bit; 5 minuti per il test iniziale, l'osservazione degli errori e il miglioramento del modello; 20 minuti per l'integrazione del modello nella programmazione in MakeCode e la definizione degli output; 15 minuti per la costruzione e l'uso del sismografo in scatola, con abbinamento della seconda micro:bit come stazione ricevente; 15 minuti per il debriefing conclusivo e la riflessione sulla trasferibilità didattica.

Il laboratorio prevede un coinvolgimento attivo dei partecipanti. I docenti non assistono a una dimostrazione, ma sperimentano il percorso in prima persona attraverso la definizione delle classi, l'addestramento del modello, la fase di test, l'osservazione degli errori, la programmazione degli output e la riflessione finale sulla trasferibilità didattica. La dimensione laboratoriale consente di affrontare contenuti informatici oggi centrali, quali dati, modelli, classificazione, affidabilità, programmazione fisica, comunicazione tra dispositivi e uso consapevole dell'intelligenza artificiale, in coerenza con le competenze attese e gli obiettivi specifici di apprendimento per l'Informatica previsti dalle Nuove Indicazioni Nazionali per il primo ciclo di istruzione.

3 Riferimento a materiali

I materiali di supporto al laboratorio sono raccolti in uno spazio online dedicato, che comprende la presentazione del percorso e risorse operative utili alla realizzazione e alla replicabilità dell'attività.

Materiali del laboratorio:
https://drive.google.com/drive/folders/1ZDx2tDp6Puxd3d63cLvk-iZboJKzUhrp.

Strumenti utilizzati:

- Scheda micro:bit: https://microbit.org/
- micro:bit CreateAI: https://microbit.org/createai/
- Microsoft MakeCode for micro:bit: https://makecode.microbit.org/

Riskiatutto: Scopri l'Assembly RISC-V

Giorgio Delzanno[1]
[1] DIBRIS, Università degli Studi di Genova, Italy
giorgio.delzanno@unige.it

1 Ordine e grado di scuola

Il laboratorio è rivolto ai docenti della Scuola Secondaria di primo e secondo grado interessati ad approfondire i principi dell'informatica e della programmazione dei sistemi embedded.

2 Argomento del laboratorio ed descrizione delle attività

Il laboratorio si propone di introdurre i principi fondamentali dell'architettura dei calcolatori: comprendere il modello di programmazione dell'architettura RISC-V; acquisire le basi della programmazione in linguaggio Assembly; comprendere il ruolo di registri, memoria, istruzioni e flusso di esecuzione di un programma; provare a promuovere il pensiero computazionale anche attraverso la programmazione a basso livello. Il laboratorio, della durata di 90 minuti, si svolge in modalità interattiva, alternando spiegazioni del docente e attività pratiche dei partecipanti su PC. Durante il laboratorio i partecipanti:

- acquisiranno familiarità con l'architettura RISC-V e con i principi fondamentali dell'organizzazione dei calcolatori;
- impareranno la struttura di base di un programma in linguaggio Assembly (solo la parte core con operazioni aritmetiche su interi);
- utilizzeranno un ambiente di sviluppo per scrivere, compilare ed eseguire semplici programmi RISC-V;
- sperimenteranno il funzionamento di microcontrollori ESP32 basati sull'architettura RISC-V.

L'approccio sarà fortemente pratico, favorendo la sperimentazione diretta e la comprensione dei meccanismi alla base dell'esecuzione del software sui sistemi embedded.

3 Riferimento a materiali

Nel laboratorio verranno usati simulatori RISK-V, https://github.com/riscv/learn#software-and-tool, e di microcontrollori, Wokvi https://wokwi.com/ (accessibile via browser). Saranno anche disponibili alcune schede ESP32-C6, https://www.espressif.com/en/products/socs/esp32-c6, basate su architettura RISC-V, programmabili con Arduino IDE https://docs.arduino.cc/software/ide/.

Bibliografia

[1] Struttura e progetto dei calcolatori. Progettare con RISC-V, D. A. Patterson, J. L. Hennessy, Zanichelli.

[2] Una Macchina Nozionale per Architetture dei Calcolatori come possibile collegamento tra gli insegnamenti del primo anno della laurea in Informatica, G. Delzanno; D. D'Agostino, G. Guerrini, and D. Traversaro, Atti del convegno ITADINFO 2023, https://www.itadinfo.it/2023/attidelconvegno/index.html.

[3] Learn RISK V, https://github.com/riscv/learn.

Programmare un *computer* nel *computer* gestendo il carico cognitivo

Gabriele Pozzan, Tullio Vardanega

Università degli studi di Padova
gabriele.pozzan@unipd.it, tullio.vardanega@unipd.it

1 Ordine e grado di scuola

Il laboratorio è indirizzato a insegnanti della scuola secondaria, ma i suoi contenuti possono essere di interesse anche a livello universitario.

2 Argomento del laboratorio e descrizione delle attività

Argomento Questo laboratorio è l'esempio "condensato" di un percorso di introduzione alla programmazione a medio/lungo termine, incentrato su un progetto "collettore" nell'arco dell'intero anno scolastico (per vedere un buon esempio di approccio simile, cf. [3]): la realizzazione di un programma per simulare un *computer*. Il progetto è abbastanza complesso e motivante da risultare credibile come esempio di "vera" programmazione e ha valore educativo relativo all'Informatica in generale, in quanto permette di sperimentare il funzionamento fondamentale di un *computer*. Questo apprendimento è di supporto allo sviluppo del pensiero Informatico: comprendere le operazioni elementari di un *computer* (lettura e scrittura in memoria, salto, ecc.), indipendentemente dall'implementazione (sempre in qualche modo arbitraria), demistifica il *software* e pone le basi per l'astrazione che permette lo sviluppo di programmi di complessità arbitraria a partire da poche operazioni fondamentali.

Il laboratorio illustra i principi del *four-component instructional design* [5], uno strumento teorico utile per progettare percorsi didattici a medio/lungo termine intesi a insegnare argomenti complessi con un attento controllo del carico cognitivo [4] indotto dall'apprendimento. Questo è fondamentale nella didattica dell'Informatica perché programmazione e pensiero algoritmico – parte inevitabile di un primo percorso di apprendimento in questo dominio – sono complessi da apprendere in quanto richiedono il coordinamento di molte nozioni e processi. Implementare un algoritmo in un programma richiede per esempio di: conoscere la sintassi di un linguaggio di programmazione, conoscere costrutti algoritmici (selezione, iterazione, ecc.), modellare problemi, e così via. La teoria del carico cognitivo suggerisce che la compresenza di troppi concetti può essere fonte di *sovraccarico cognitivo* per soggetti da poco esposti a un nuovo dominio di conoscenza, come ad esempio chi sta apprendendo i primi rudimenti di Informatica. Chi partecipa al laboratorio può fare esperienza dei principi didattici del *four-component instructional design*, attraverso esercitazioni e sessioni di pratica mirata che esemplificano attività didattiche i cui prodotti finali sono le componenti principali del *computer* (CPU, registri, memoria, I/O) in linguaggio Python.

Prerequisiti e strumentazione necessaria Il materiale del laboratorio è disponibile *online* per essere scaricato sui dispositivi di chi partecipa. I dispositivi (preferibilmente laptop) devono dare la possibilità di implementare, modificare ed eseguire programmi Python.

Gli esercizi del laboratorio sono contenuti in *file* Python (e in un *notebook* accessibile via web) che chi partecipa deve leggere, completare, eseguire, a seconda dell'esercizio. Per le esercitazioni è

necessario scrivere e modificare il codice esistente; per le sessioni di pratica mirati vengono dati esempi di attività simili a quelle descritte in [2]. Ogni esercitazione termina con il completamento di una componente del progetto finale – la simulazione di un *computer*. Ogni esercitazione e sessione di pratica mirata è introdotta da un cappello teorico durante il quale il conduttore spiega il tipo di attività dal punto di vista della teoria del carico cognitivo e il tipo di esercizio specifico (ad esempio, problemi a completamento, *puzzle* di Parson, eccetera).

Descrizione delle attività Il laboratorio inizia con una breve spiegazione dell'architettura del *computer* che sarà simulato dal progetto Python e dei principi generali della teoria del carico cognitivo [4]. A seguire, verranno spiegate le componenti del 4C/ID [5], alternate ad attività durante le quali chi partecipa implementerà componenti del *computer* affrontando varie tipologie di esercizi: esempi guidati, problemi a completamento, *puzzle* di Parson [1], pratica mirata per il rinforzo di concetti, eccetera. Al termine del laboratorio sarà possibile assemblare le componenti implementate ed usare la simulazione di *computer* per eseguire programmi in linguaggio macchina.

Descrizione del progetto collettore Il *computer* implementato durante il laboratorio è basato su un'architettura di Von Neumann semplificata a scopo didattico, tratta da una pubblicazione divulgativa sui temi dell'Informatica[1]. Memoria e registri contengono valori rappresentati da sequenze di 10 cifre decimali (ad esempio, 0000000006 per rappresentare il numero sei) e, nella simulazione Python, sono implementati tramite liste indicizzate. Il linguaggio macchina consiste a sua volta di sequenze di 10 cifre decimali: le prime due identificano l'istruzione, le restanti vengono interpretate in base all'istruzione. La simulazione, quando eseguita, legge un programma in codice macchina da un file predefinito, copiando le sue istruzioni in memoria a partire dalla prima cella. Poi esegue le istruzioni una alla volta, aggiornando il *program counter* – una semplice variabile intera – automaticamente (aumentandolo di uno dopo ogni istruzione), oppure in conseguenza a particolari istruzioni (ad esempio le istruzioni di salto). L'esecuzione termina quando viene incontrata una specifica istruzione di *stop*.

3 Riferimento a materiali

I materiali del laboratorio sono disponibili su GitHub:
https://github.com/cornacchia/workshop-itadinfo-2026

Riferimenti bibliografici

[1] Dale Parsons and Patricia Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pages 157–163, 2006.

[2] Gabriele Pozzan, Andreas Bollin, and Tullio Vardanega. A cognitive-load-informed decomposition of debugging subskills with targeted exercises. *Informatics in Education*, 25(2):135–165, 2026.

[3] Juha Sorva and Otto Seppälä. Research-based design of the first weeks of cs1. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, page 71–80, New York, NY, USA, 2014. Association for Computing Machinery.

[4] John Sweller, Jeroen JG Van Merriënboer, and Fred Paas. Cognitive architecture and instructional design: 20 years later. *Educational psychology review*, 31(2):261–292, 2019.

[5] Jeroen JG Van Merriënboer, Richard E Clark, and Marcel BM De Croock. Blueprints for complex learning: The 4c/id-model. *Educational technology research and development*, 50(2):39–61, 2002.

[1] https://www.veniac.com/

Apprendimento di SQL basato su errori frequenti

Giovanna Guerrini, Davide Ponzini

Università di Genova, Italia
giovanna.guerrini@unige.it, davide.ponzini@edu.unige.it

1 Ordine e grado di scuola

Secondaria di secondo grado e università

2 Argomento del laboratorio e descrizione delle attività

Il laboratorio nasce dall'attività svolta da Giovanna Guerrini nell'ambito del Working Group ITiCSE 2025 su Notional Machine for Data Systems [1] e dal progetto di dottorato di Davide Ponzini, che ha realizzato un tutor per l'apprendimento di SQL centrato sull'analisi di errori frequenti [3].

L'obiettivo del laboratorio è quello di analizzare insieme ai partecipanti alcuni errori tipici commessi dagli studenti nella formulazione di interrogazioni SQL e di ragionare insieme su quali possano essere le misconcezioni sottostanti, quali analogie e visualizzazioni possano essere d'aiuto, quale supporto possa essere fornito all'apprendimento, anche in termini di spiegazioni da richiedere a un tutor basato su AI generativa quale LensQL[1] [2], e quali esercizi mirati proporre.

Il laboratorio è diviso in tre parti.

- **Parte I - 15m:** La prima parte del laboratorio farà emergere dai partecipanti alcuni errori frequenti riscontrati da parte degli studenti per introdurre la tassonomia, rispetto alla quale si individueranno un esempio di errore sintattico, uno semantico, uno logico e una complicazione [3].
- **Parte II - 45m:** Nella seconda parte del laboratorio, a gruppi, per i 4 errori individuati si andranno ad analizzare possibili cause (misconcezioni) e analogie e visualizzazioni, nonchè tool (ad es. RelaX https://dbis-uibk.github.io/relax) che possano essere utili agli studenti. Questa parte si conclude con la restituzione in plenaria di quello che ogni gruppo ha individuato per l'errore analizzato
- **Parte III - 30m:** La terza parte del laboratorio è dedicata alle possibili spiegazioni (anche supportate da AI generative) volte a chiarire l'errore e alla formulazione di esercizi ad hoc volti a superare l'errore. Si lavorerà nuovamente a gruppi utilizzando LensQL https://lensql.dibris.unige.it.

Strumentazione richiesta. Laptop e uso di applicazioni web (connessione a internet).

3 Riferimento a materiali

Materiali disponibili al link https://bit.ly/itadinfo26-sql. Tool disponibile al link https://lensql.dibris.unige.it.

[1] https://lensql.dibris.unige.it

References

[1] D. Miedema, M. Goodfellow, C. Satyavolu, G. Haldeman, L. Busuttil, L. Farinetti, G. Guerrini, Y. Pan, S. Ramagoni, R. Sooriamurthi, et al. Explanations of data systems concepts in cs education: An updated view on notional machines. In *ITiCSE-WGR 2025-Publication of the 2025 Working Group Reports on Innovation and Technology in Computer Science Education*, pages 108–135. 2026.

[2] D. Ponzini, B. Catania, and G. Guerrini. Enhancing SQL Learning Through Generative AI and Student Error Analysis. In *New Trends in Database and Information Systems*, pages 118–128. Springer, 2025.

[3] D. Ponzini, G. Guerrini, and B. Catania. A Proposal for Revising SQL Error Taxonomies Based on Automated Detection. In *Proc. of the 5th Int'l Workshop on Data Systems Education*, 2026.

Video

Metodologie per la Divulgazione Informatica nel Formato Short-Form Video

Valerio Conigliaro[1]

Università degli Studi di Palermo
Dipartimento di Matematica e Informatica
valerio.conigliaro@unipa.it

Sommario

Il presente lavoro propone un **framework metodologico universale** per la produzione di contenuti divulgativi informatici nel formato *short-form video* rivolti a un pubblico non specializzato. Il framework articola il processo comunicativo in quattro fasi sequenziali (Hook, Analogia, Concetto Scientifico, Chiusura), ciascuna accompagnata da principi cognitivi consolidati dalla letteratura, ed è progettato per essere applicato trasversalmente all'intero spettro dei costrutti informatici astratti. La sfida centrale risiede nell'**intangibilità** dell'informatica, che richiede strategie cognitive specifiche per essere comunicata efficacemente. Il nucleo della divulgazione informatica è anche mantenere l'attenzione dello spettatore: ogni singola parola assume un **peso cognitivo**. A differenza degli studi che applicano la *Cognitive Theory of Multimedia Learning* [12] a contesti scolastici formali, il presente lavoro ne estende l'applicazione al formato *short-form video*, integrandola con le teorie cognitive della letteratura. L'operatività del framework viene dimostrata attraverso la funzione di hash crittografica come **running example**. Video applicativo: https://drive.google.com/file/d/1Cd-2rnblnOY5j6Mlu6UA8fNiZOfr9OIj/view?usp=drive_link

1 Introduzione

La divulgazione scientifica dispone di un arsenale comunicativo consolidato per rendere accessibili le varie scienze a un pubblico vasto. L'informatica sconta una carenza strutturale di divulgatori efficaci, riconducibile a un ostacolo epistemico di fondo: i suoi oggetti di studio sono entità prive di tangibilità fisica e refrattarie alla rappresentazione visiva diretta. La soluzione risiede nell'adattare i *pattern* comunicativi delle altre scienze per *visualizzare l'invisibile*: rendere comprensibile l'astratto attraverso analogie ad elementi della vita quotidiana che un pubblico non esperto possa riconoscere e manipolare cognitivamente. La letteratura si è concentrata su ambienti scolastici controllati [12]; la sua applicazione al formato *short-form video* rimane inesplorata.

Questo lavoro colma tale lacuna formalizzando le metodologie comunicative osservate e implicitamente usate nei canali divulgativi ad alto impatto [20, 9, 6] – coerenti con la letteratura sulla teoria cognitiva – in un framework teorico trasversale.

2 Il Modello in Quattro Fasi

Il modello formalizza la struttura divulgativa in quattro fasi sequenziali, ciascuna ancorata a principi teorici specifici (Tabella 1).

2.1 Hook (0–10 s)

L'hook è la fase cruciale per catturare la ***retention*** – il tempo che uno spettatore trascorre sul contenuto prima di abbandonarlo. In un contesto di scrolling rapido, se i primi secondi

Tabella 1: Modello operativo in quattro fasi per la divulgazione informatica breve.

Fase	**Durata**	**Principi teorici**
Hook	0–10 s	Information-Gap Theory [10]
Analogia	20–30 s	Signaling Principle [12]; apprendimento multisensoriale [17]
Concetto Scientifico	25–60 s	Structure-Mapping Theory [5]; Cognitive Load / Chunking [19, 2]
Chiusura	10–25 s	Struttura circolare; ritorno al registro dell'hook

non innescano una risposta affettiva o cognitiva immediata, il contenuto viene scartato. L'hook deve suscitare una domanda diretta nello spettatore o innescare una reazione emotiva mirata: curiosità, sorpresa o un senso di allarme. Il meccanismo si basa sull'**Information-Gap Theory** [10]: per catturare e mantenere l'attenzione, il linguaggio deve creare una lacuna nella conoscenza dello spettatore. Loewenstein dimostra che l'attenzione si accende nel momento in cui il linguaggio evidenzia un divario tra ciò che sappiamo e ciò che vogliamo sapere – la curiosità emerge così come *cognitive itch*, un prurito cognitivo, che motiva l'ascolto fino alla risoluzione del divario. Sul piano operativo, l'hook deve ancorarsi a un fenomeno già noto allo spettatore. *Nel running example*, l'hook sfrutta la preoccupazione per la riservatezza delle credenziali: *"Quando ti registri in un sito, la tua password è riservata oppure il sito adesso la conosce?"*

2.2 Analogia (20–30 s)

La seconda fase rende percettivamente accessibile un costrutto computazionale astratto associandolo ad un **oggetto fisico analogo**: un artefatto le cui proprietà siano già note nel repertorio cognitivo dello spettatore. La condizione necessaria non è la somiglianza percettiva, ma l'*isomorfia relazionale* tra i due sistemi [5]. Dall'osservazione empirica, l'ancoraggio cognitivo è rafforzato da enfasi tipografica (es. font ingranditi o grassetti, **visual cue**) su ogni termine tecnico introdotto, sincronizzato con un effetto sonoro (***auditory cue***).

Il **Signaling Principle** [12] formalizza questa tecnica; inoltre, Shams e Seitz [17] evidenziano la netta superiorità mnemonica dell'apprendimento multisensoriale rispetto all'apprendimento monomodale (basato su un solo canale sensoriale). *Nel running example*, l'oggetto selezionato è il **frullatore**: ingredienti mescolati producono un composto irreversibile e costante, proiettando sulla percezione macroscopica le proprietà operative della funzione di hash.

2.3 Concetto Scientifico (25–60 s)

La transizione dalla metafora alla spiegazione tecnica è la fase più critica e deve rispettare due vincoli teorici.

Structure-Mapping. Ogni elemento caratteristico dell'analogia trova il suo corrispettivo tecnico all'argomento trattato. Secondo Gentner [5], un'analogia è efficace non per somiglianza fisica tra oggetti, ma per isomorfismo tra sistemi di relazioni, che rende il trasferimento cognitivo possibile. Ogni rottura in tale corrispondenza produce una discontinuità che interrompe la comprensione.

Nel running example: ingredienti → *dati di input*; lame → *algoritmo crittografico*; composto finale → *digest*.

Chunking e Cognitive Load. La corretta gestione del carico cognitivo costituisce il secondo vincolo strutturale. La revisione di Cowan [2] fissa a 3–4 unità la capacità mnemonica di un ascoltatore. La *Cognitive Load Theory* [19] prescrive pertanto di frammentare i concetti ad alto carico cognitivo in step sequenziali e autonomi (**Segmenting**) – anziché erogarli come flusso continuo. Il canale Kurzgesagt [9] applica questo principio suddividendo la spiegazione in micro-sezioni numerate. *Nel running example*: (1) input di lunghezza variabile; (2) calcolo deterministico; (3) *digest* di lunghezza fissa; (4) irreversibilità.

2.4 Chiusura (10–25 s)

Si abbandona la terminologia tecnica per tornare al registro colloquiale dell'hook, rispondendo alla domanda iniziale. La struttura circolare soddisfa l'aspettativa cognitiva generata dall'Information-Gap [10] e segnala la conclusione del ciclo di apprendimento. *Nel running example*: "Quindi adesso sappiamo che solo tu sai la tua password...forse.".

3 Linguistica dell'Attenzione e Scaffolding Lessicale

La struttura narrativa del framework non è sufficiente da sola: il linguaggio e il vocabolario utilizzati sono determinanti quanto l'architettura del contenuto. In un short form video ogni parola ha peso cognitivo critico; lessico e sintassi densi provocano nello spettatore perdita di attenzione (fenomeno dello *swipe/skip*) o frustrazione cognitiva. La sezione analizza due macroaree: l'**Information-Gap Theory** – come trattenere l'utente – e lo **scaffolding lessicale** – come spiegare senza generare blocco emotivo.

3.1 Strategie Linguistiche per la Retention

Power words: il lessico accademico è sostituito con parole ad alto *arousal* psicofisiologico – eccitazione, allerta, dissonanza cognitiva – correlate a incrementi significativi del *retention rate* [1]: *"Sai perché gli hacker hanno vita difficile?"* anziché *"Analizziamo i problemi P e NP"*.

Interazione parasociale: il soggetto è sempre lo spettatore, non il concetto. La comunicazione di massa genera un'illusione di conversazione diretta [7] che aumenta il coinvolgimento affettivo. Il principio può operarsi con la **Regola S-V-O** (Soggetto–Verbo–Oggetto): frasi brevi e attive rispettano il limite di 3–4 *chunk* della memoria di lavoro [2], che le proposizioni passive esaurirebbero rapidamente [8]:

- *Passivo:* "La password inserita viene crittografata dal sistema per evitare che venga rubata."
- *S-V-O:* "Tu scrivi la password. Il sistema la tritura. L'hacker trova solo le briciole."

Reset cognitivo e domande retoriche: La vigilanza [11] decade ogni 15–20 secondi; l'assuefazione agli script didattici tradizionali [16] accelera il processo. Rotture di pattern – affermazioni controintuitive, sfatare miti – attivano il *Riflesso di Orientamento* [18], ripristinando l'attenzione. Integrare domande retoriche (es. *"E allora cosa succede?", "Ma ha senso tutto questo?"*) produce un duplice effetto: secondo Petty et al. [15] riportano l'attenzione del pubblico nell'immediato; lasciare la risposta temporaneamente in sospeso innesca l'**Effetto Zeigarnik** [23]: la tendenza del cervello a restare attento verso quesiti irrisolti fino alla loro risoluzione.

3.2 Scaffolding Lessicale

Quando un pubblico non specializzato incontra termini (*jargon*) come *Polimorfismo dinamico* o *Allocazione dinamica nell'heap* senza alcuna preparazione, si sente cognitivamente escluso, subendo un blocco emotivo con possibile abbandono del contenuto. Vygotsky [21] ha teorizzato la *Zona di Sviluppo Prossimale (ZPD)*: la distanza tra ciò che lo spettatore sa fare autonomamente e ciò che può apprendere se guidato da un supporto adeguato. Tale supporto è lo **scaffolding** [22], definito come supporto che riduce la complessità del compito, permettendo allo spettatore di concentrarsi sugli elementi alla sua portata.

La **Tecnica del Sandwich** (Associazione–Termine–Azione) operazionalizza lo scaffolding secondo i principi della *Structure-Mapping Theory* [5] in tre passi sequenziali:

1. *Associazione* (scenario familiare) – *"Immagina un traduttore super veloce..."*;
2. *Termine tecnico* (isomorfia) – *"...quello che in informatica chiamiamo Compilatore..."*;
3. *Azione tangibile* – *"...che frulla il codice scritto da te e lo trasforma in linguaggio macchina"*.

4 Visual Design

4.1 Color Coding

L'impiego funzionale del *color cueing* riduce il tempo di ricerca visiva e migliora la ritenzione mnemonica [14]: colori caldi nell'hook innalzano l'*arousal* psicofisiologico, catturando l'attenzione [3] durante le rapide dinamiche del video; colori freddi nella fase tecnica abbassano il carico emotivo percepito, favorendo l'elaborazione logico-razionale del contenuto. Un design minimale riduce il *Carico Cognitivo Estraneo* [19].

4.2 Whiteboard Animation (Dynamic Drawing) e Social Agency

Il framework alterna *whiteboard animation* e *live-action*, che sfruttano due meccanismi cognitivi distinti, ciascuno assegnato a una fase narrativa.

Osservare un disegno *mentre viene tracciato* – *Dynamic Drawing* – sincronizza elaborazione visiva e verbale rispettando il *Temporal Contiguity Principle* [12, 4]: il disegno non può anticipare il parlato, pena l'avvio di una decodifica autonoma da parte dello spettatore che incrementa il carico cognitivo estraneo e genera frustrazione cognitiva.

La *live-action* attiva invece le convenzioni sociali del cervello dello spettatore: la presenza di un volto umano con contatto visivo e gestualità aumenta la motivazione a elaborare il messaggio (*Social Agency Theory*, Mayer et al. [13]). La struttura, coerente con il modello in quattro fasi, è: *live-action* nell'hook (aggancio parasociale); *whiteboard* nelle fasi analogia e concetto scientifico (attenzione convergente sul meccanismo, non sul volto) con *live-action* nelle transizioni tra chunk; *live-action* nella chiusura (segnale di conclusione del ciclo cognitivo).

5 Conclusione

Il framework proposto costituisce il primo tentativo sistematico di formalizzare la divulgazione informatica nel formato *short-form video* in termini teorici cognitivi. L'integrazione delle teorie cognitive adottate in un'architettura operativa sequenziale rappresenta una novità rispetto

alla letteratura esistente, che le applica separatamente e in contesti scolastici controllati. La validazione empirica su prototipi video eterogenei costituisce il naturale sviluppo futuro.

6 Ringraziamenti

Si ringrazia la prof.ssa Sabrina Mantaci per la guida e i preziosi suggerimenti.

Riferimenti bibliografici

[1] Jonah Berger and Katherine L. Milkman. What makes online content viral? *Journal of Marketing Research*, 49(2):192–205, 2012.

[2] Nelson Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1):87–114, 2001.

[3] Andrew J. Elliot and Markus A. Maier. Color psychology: Effects of perceiving color on psychological functioning in humans. *Annual Review of Psychology*, 65:95–120, 2014.

[4] Logan Fiorella and Richard E. Mayer. Effects of observing the instructor draw diagrams on learning from multimedia messages. *Journal of Educational Psychology*, 108(4):528–546, 2016.

[5] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[6] Geopop. Geopop. https://www.youtube.com/@geopop, 2023. Canale YouTube, consultato il 15 gennaio 2024.

[7] Donald Horton and Richard Wohl. Mass communication and para-social interaction: Observations on intimacy at a distance. *Psychiatry*, 19(3):215–229, 1956.

[8] Marcel A. Just and Patricia A. Carpenter. A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, 99(1):122–149, 1992.

[9] Kurzgesagt. Kurzgesagt. https://www.youtube.com/@kurzgesagt, 2022. Canale YouTube, consultato il 15 gennaio 2024.

[10] George Loewenstein. The psychology of curiosity: A review and reinterpretation. *Psychological Bulletin*, 116(1):75–98, 1994.

[11] Norman H. Mackworth. The breakdown of vigilance during prolonged visual search. *Quarterly Journal of Experimental Psychology*, 1(1):6–21, 1948.

[12] Richard E. Mayer. *Multimedia Learning*. Cambridge University Press, 2 edition, 2009.

[13] Richard E. Mayer, Kristina Sobko, and Patricia D. Mautone. Social cues in multimedia learning: Role of speaker's voice. *Journal of Educational Psychology*, 95(2):419–425, 2003.

[14] Erol Ozcelik, Turkan Karakus, Engin Kursun, and Kursat Cagiltay. An eye-tracking study of how color coding affects multimedia learning. *Computers & Education*, 53(2):445–453, 2009.

[15] Richard E. Petty, John T. Cacioppo, and Martin Heesacker. Effects of rhetorical questions on persuasion: A cognitive response analysis. *Journal of Personality and Social Psychology*, 40(3):432–440, 1981.

[16] Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, 1977.

[17] Ladan Shams and Aaron R. Seitz. Benefits of multisensory learning. *Trends in Cognitive Sciences*, 12(11):411–417, 2008.

[18] Evgeny N. Sokolov. *Perception and the Conditioned Reflex*. Macmillan, 1963.

[19] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988.

[20] Vox. Vox – explanatory journalism. https://www.youtube.com/@Vox, 2022. Canale YouTube, consultato il 15 gennaio 2024.

[21] Lev S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, 1978.

[22] David Wood, Jerome S. Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2):89–100, 1976.

[23] Bluma Zeigarnik. Über das behalten von erledigten und unerledigten handlungen. *Psychologische Forschung*, 9:1–85, 1927.

Dal problema al programma … con una manciata di fiammiferi!

Martina Palazzolo
Istituto Comprensivo Ilaria Alpi di Milano
martina.palazzolo@icilariaalpi.edu.it

Il video [1] propone una rielaborazione personale in chiave informatica del seguente quesito invalsi del 2008 per la classe terza della scuola secondaria di primo grado.

C12. Alcuni fiammiferi sono disposti come indicato nelle figure.

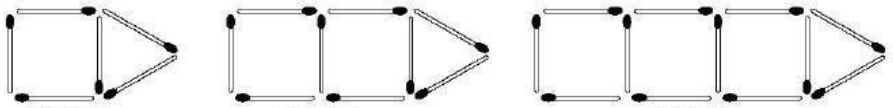


Figura 1 Figura 2 Figura 3

Se si continua la sequenza delle figure, quanti fiammiferi verranno usati per fare la figura 10?

Inizialmente adattato da Ines Marazzani per la scuola primaria, il quesito è stato successivamente utilizzato in prospettiva verticale nella scuola del primo ciclo dal gruppo di progettazione del corso nazionale "Matematica e Informatica nelle Nuove Indicazioni Nazionali" [2].
Al problema matematico è stato affiancato un tipico problema informatico: programmare un esecutore automatico che possa disegnare la figura 1, 2 … 10.
Il video è suddiviso in due parti.
La prima parte illustra un possibile approccio aritmetico al quesito originale che invita a osservare regolarità per condurre a determinare una regola o formula per calcolare il numero di fiammiferi della figura 10. Si illustra un ragionamento ma si lascia aperta la risposta per permettere ai ragazzi di trovare una propria strategia risolutiva da discutere insieme.
La seconda parte del video è dedicata al quesito modificato in cui viene messa in evidenza la regolarità geometrica: ad un triangolo equilatero si aggiungono in coda un quadrato alla volta. Si presenta un esecutore e le istruzioni base che sa eseguire (FORWARD 50 e RIGHT 90). Si scrivono inoltre le sequenze di istruzioni per disegnare un triangolo equilatero e un quadrato usando il linguaggio di programmazione dell'esecutore, in questo caso *LibreLogo* della suite *LibreOffice* [3]. Nei programmi proposti vengono utilizzati anche i costrutti REPEAT e TO – END senza spiegarli in modo che i ragazzi possano dedurne la funzione direttamente sperimentando con il PC.

Sitografia

[1] https://youtube.com/shorts/4kxMvyLQyi4
[2] https://www.airdm.org/matematica-e-informatica-nelle-nuove-indicazioni-nazionali-airdm-cini-ciim/
[3] LibreLogo di LibreOffice - https://it.libreoffice.org

Pensare Informatico

Michael Lodi[1,2]
[1] Alma Mater Studiorum – Università di Bologna
michael.lodi@unibo.it
[2] Laboratorio nazionale CINI "Informatica e Scuola"

Pensare Informatico è un progetto di divulgazione che racconta l'informatica come disciplina scientifica a un pubblico non specialistico. I video adottano un linguaggio diretto e informale ed esempi quotidiani per avvicinare alle idee fondamentali della disciplina, spesso confusa con il semplice uso di dispositivi e applicazioni.

Al momento sono presenti tre video. Il primo presenta l'informatica come scienza e introduce l'identità del progetto. Il secondo mostra come due interlocutori possano generare un segreto condiviso comunicando su un canale insicuro, raccontando in forma accessibile l'idea alla base del protocollo di Diffie–Hellman–Merkle. Il terzo spiega le funzioni hash attraverso una metafora culinaria: un "forno" trasforma gli ingredienti in un pasticcio non reversibile, chiarendo ad esempio come i siti web possano verificare una password senza memorizzarla direttamente.

Il formato riprende la pratica delle sequenze di Storie Instagram consecutive (Arenare, 2019, https://doi.org/10.15162/2704-8659/1128), informalmente chiamate "treni di storie" e usate nella divulgazione scientifica italiana (vedi ad es. Dario Bressanini), ma prova ad adattarla al formato degli "shorts" (video brevi) da massimo 3 minuti, per sfruttare la visibilità che le piattaforme offrono a questo formato, mantenendo però una trattazione semplice ma non banalizzata, che metta in luce la "big idea" informatica al centro del video (Bell et al., 2018).

Il progetto è nelle sue fasi embrionali e beneficerà dei feedback del pubblico per essere migliorato. I video hanno raccolto complessivamente decine di migliaia di visualizzazioni.

Gli shorts sono pubblicati sulle principali piattaforme (attualmente: YouTube, Instagram, TikTok, Facebook). I canali sono accessibili al link https://lodi.ml/social/.

Funzione Hash - Framework Metodologico Short-Form-Video

Valerio Conigliaro

Il video applica in pratica il framework in quattro fasi per la divulgazione informatica rivolto a un pubblico non specializzato, usando la funzione hash crittografica come caso di studio per rendere accessibile un concetto tecnico altrimenti astratto.

L'hook sfrutta l'Information-Gap Theory per instaurare curiosità e un lieve senso di allarme, rafforzato dalla power word "hacker", per esempio.

L'analogia costruisce lo scaffolding lessicale attraverso la Structure-Mapping Theory, associando gli elementi del frullatore alla funzione hash — ingredienti → input, lame → algoritmo, frappè → digest — seguendo la Tecnica del Sandwich (Associazione, Termine, Azione) e il Signaling Principle, con visual cue e auditory cue sincronizzati sull'introduzione dei termini tecnici.

Nella fase del concetto scientifico, il Chunking gestisce il carico cognitivo scomponendo la spiegazione in blocchi sequenziali (hash base → collisione → salatura → difficoltà computazionale), mentre la Regola S-V-O mantiene frasi brevi e attive ("L'hacker trova solo i frappè"). Domande retoriche come "Ma è un problema questo?" e "Ma quanto è difficile, alla fine?" innescano un reset cognitivo, e l'Effetto Zeigarnik tiene alcuni quesiti temporaneamente sospesi prima di scioglierli ("Spoiler: nessun input sarà la tua password").

La chiusura adotta la struttura circolare, richiamando il registro informale dell'hook ("miscuglio salato") e risolvendo la tensione iniziale con "Adesso sai che...", segnalando così la fine del ciclo di apprendimento.

Sul piano visivo, il video alterna whiteboard animation — rispettando il Temporal Contiguity Principle tra disegno e parlato — e riprese live-action, che attivano la Social Agency Theory attraverso il contatto visivo, i volti umani e la gestualità, rafforzando il coinvolgimento parasociale.

Link: https://drive.google.com/file/d/1Cd-2rnblnOY5j6Mlu6UA8fNiZOfr9OIj/view?usp=drive_link

Questo volume raccoglie gli atti del quarto convegno italiano sulla didattica dell'Informatica (ITADINFO) tenutosi a Padova dal 9 all'11 ottobre 2026.
I contributi inclusi nel volume includono: presentazioni dedicate alla scuola, dall'infanzia alla secondaria di secondo grado, poster, laboratori e contributi dedicati alla divulgazione attraverso video.

PDF finito di comporre l'8 ottobre 2026

Edizioni Kangourou

