AI-Earthquake Monitoring: Machine Learning con micro:bit



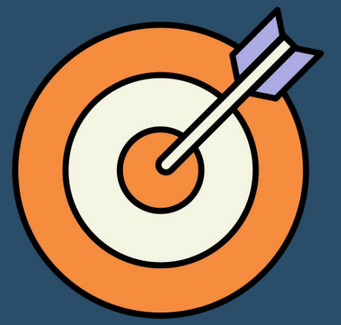
Carla Vitale

Scuola secondaria di I grado

9-11 ottobre 2026, Padova

[ITADINFO]
CONVEGNO ITALIANO
SULLA DIDATTICA DELL'INFORMATICA

Obiettivo del laboratorio



Finalità: Avvicinare al **Machine Learning** attraverso un'esperienza concreta di costruzione, **addestramento** e utilizzo di un **modello di classificazione** con **micro:bit.**

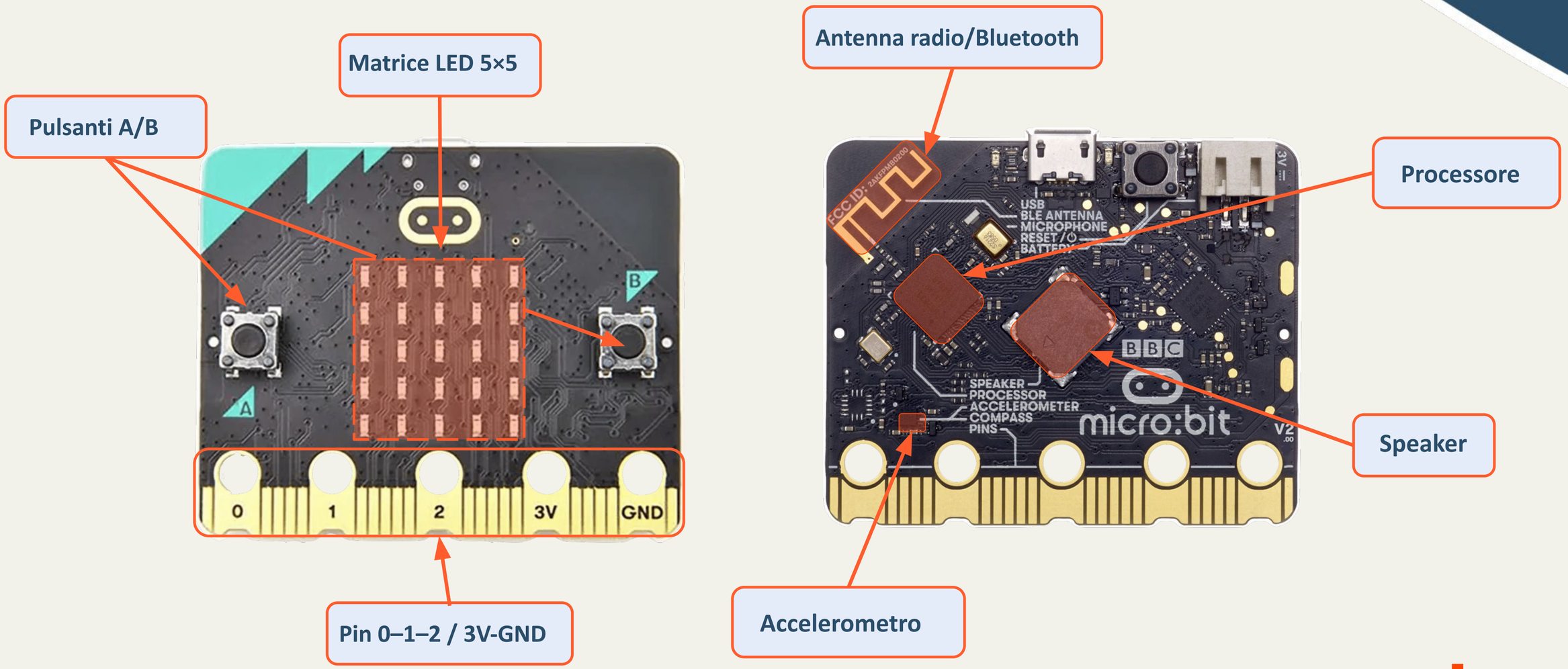
Nucleo informatico

- comprendere i concetti di **classe, dataset, modello e previsione**;
- addestrare e verificare un **modello di Machine Learning** in CreateAI;
- integrare il modello nella **programmazione a blocchi** con MakeCode;
- usare i **sensori della micro:bit** come input del sistema;
- **programmare output visivi, sonori e LED esterni** in base alla classe riconosciuta;
- mettere in **comunicazione radio due micro:bit**, una come rilevatore e una come stazione ricevente.

Scenario applicativo

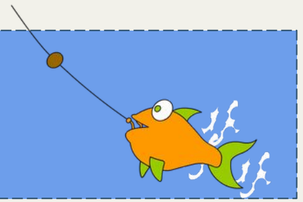
Simulazione del monitoraggio di un terremoto attraverso un prototipo didattico di "sismografo in scatola", utilizzato per riconoscere **quiete, movimento ondulatorio e movimento sussultorio.**

La micro:bit: sensore, elaborazione e comunicazione



Il percorso didattico: IBSL e modello 5E

Engage



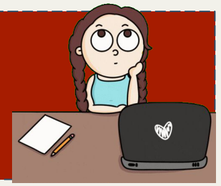
Domanda guida: «Come può un sensore digitale distinguere tra movimenti diversi?»

Explore



Definizione delle classi e raccolta dei campioni in CreateAI.

Explain



Formalizzazione dei concetti di Machine Learning e dei concetti disciplinari relativi alle onde sismiche P e S.

Elaborate



Integrazione del modello in MakeCode, programmazione degli output e comunicazione radio tra due micro:bit.

Evaluate



Test del modello, osservazione degli errori, miglioramento e debriefing finale

Il problema

Engage

Come può un sensore digitale distinguere movimenti diversi e riconoscere un pattern significativo?

Osserviamo come vengono raccolti e interpretati i segnali sismici.



Quali dati servono a una macchina per distinguere un evento da un altro?

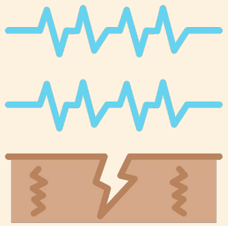
Dal terremoto alla simulazione didattica

Tradurre un fenomeno complesso in pattern di movimento riconoscibili dalla micro:bit

Terremoto → vibrazioni del suolo → simulazione dei movimenti → classi ML → CreateAI

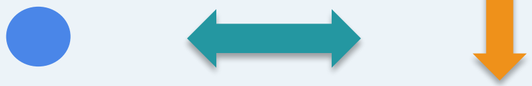
1 Fenomeno reale

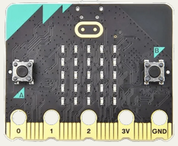
Durante un terremoto il terreno è attraversato da onde sismiche che producono vibrazioni e spostamenti del suolo.



2 Simulazione didattica

Nel laboratorio non riproduciamo le onde sismiche reali: simuliamo movimenti semplici rilevabili dall'accelerometro e distinguibili dal modello ML.





Sensore: accelerometro a 3 assi (X, Y, Z)

3 Classi del modello ML

QUIETE	micro:bit ferma
ONDULATORIO	movimento laterale (piano X–Y)
SUSSULTORIO	movimento verticale (asse Z)

Le classi non descrivono tipi di onde sismiche: sono una semplificazione didattica dei movimenti osservabili, usata per costruire il dataset.

Machine Learning su CreateAI

Explore

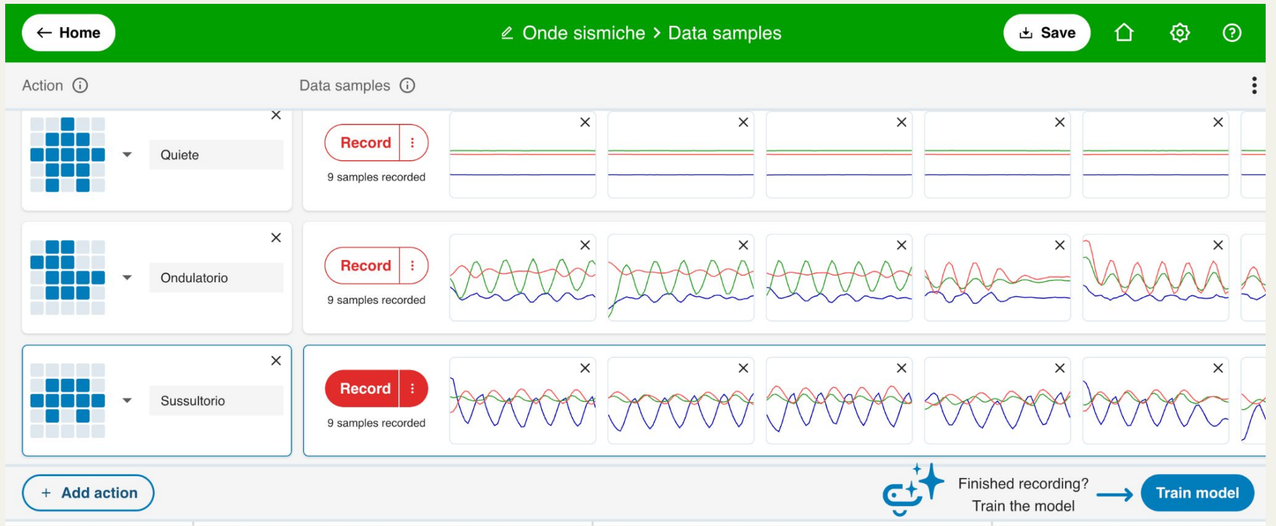
Capire come il modello apprende dai dati

dati → modello → test → miglioramento

Workflow CreateAI: Collega → Raccogli i dati → Addestra il modello→Testa il modello → Programma → Usa

Definizione delle classi

- **Quiete** → micro:bit ferma
- **Ondulatorio** → oscillazione laterale regolare (piano X-Y)
- **Sussultorio** → brevi movimenti verticali (asse Z)



Dataset di addestramento

[ITADINFO]
CONVEGNO ITALIANO
SULLA DIDATTICA DELL'INFORMATICA

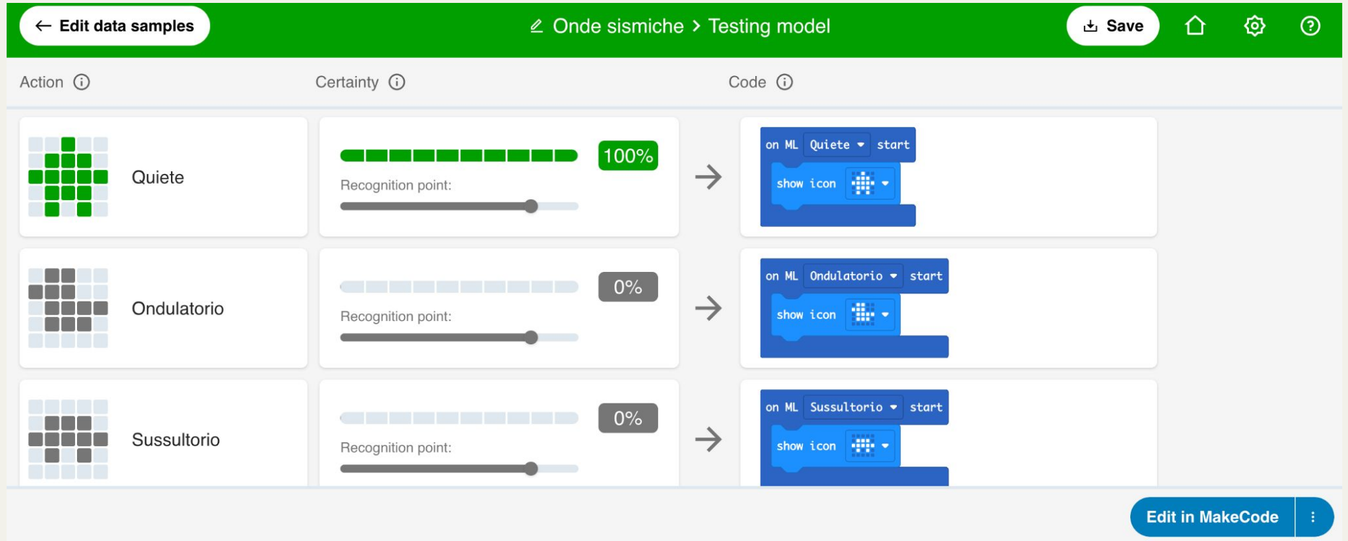
Dal dataset al modello addestrato

Explore

Testare il riconoscimento prima di passare alla programmazione

Dataset → Training → Test del modello → Edit in MakeCode

- Il modello assegna una percentuale di certezza a ciascuna classe;
- il Recognition point definisce la soglia oltre la quale una classe viene riconosciuta;
- ogni classe è già associata a un evento ML che poi sarà programmato in MakeCode.



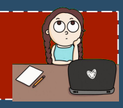
Test del modello addestrato

Test non soddisfacente → miglioramento del dataset → nuovo training

[ITADINFO]
CONVEGNO ITALIANO
SULLA DIDATTICA DELL'INFORMATICA

Machine learning

Explain

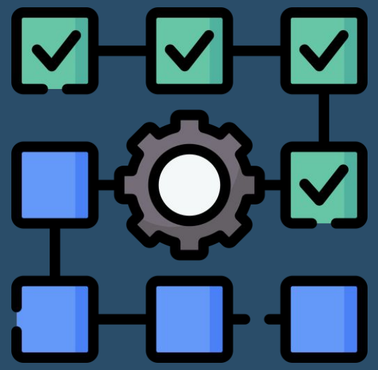


Due momenti diversi: prima il modello impara, poi viene messo alla prova con dati nuovi.

FASE 1: Addestramento

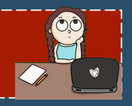


FASE 2: Test



Dal dataset al modello

Explain



Fase 1: Addestramento

1 Dataset

Campioni dell'accelerometro etichettati come Quiete, Ondulatorio e Sussultorio.

2 Training

L'algoritmo analizza gli esempi e cerca regolarità utili a distinguere le classi.

3 Modello addestrato

Il risultato del training è un classificatore che incorpora le regolarità apprese dai dati (pattern)

Fase 2: Test

4 Dati di test

Un campione non usato nell'addestramento viene fornito al modello per verificarne il comportamento.

5 Applicazione del modello

Il modello analizza i nuovi dati in ingresso.

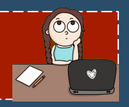
6 Output/ previsione

Il modello assegna il nuovo campione a una classe e restituisce una percentuale di certezza per ciascuna classe.

Dataset di addestramento ≠ dataset di test

Test, errore e miglioramento

Explain



Il test serve a capire se il modello generalizza su esempi nuovi

Cosa abbiamo osservato

- **Movimenti rapidi** — pattern più distinti → riconoscimento più stabile
- **Movimenti lenti** — segnali meno marcati → maggiore incertezza o confusione
- **Dati di training** — qualità e varietà degli esempi influenzano il comportamento del modello

Ciclo di miglioramento



Il modello non “riconosce un terremoto reale”: classifica pattern di movimento appresi dagli esempi.

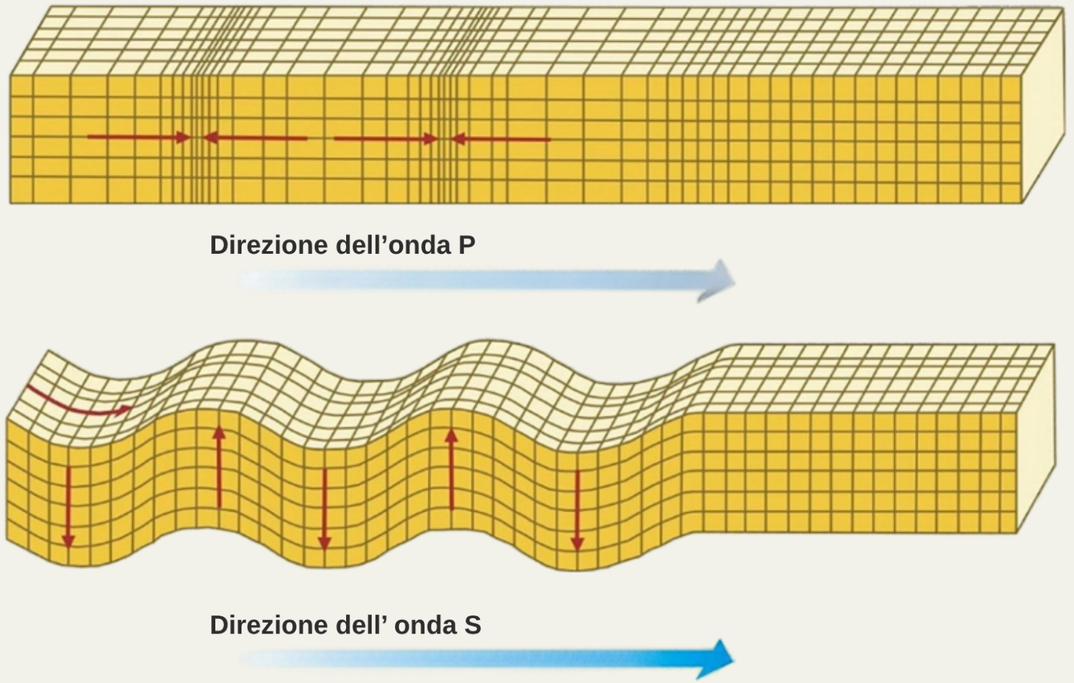
Explain

Il contesto disciplinare:

Onde sismiche P e S

Onde P le particelle **vibrano parallelamente** alla direzione di propagazione.

Onde S le particelle **vibrano perpendicolarmente** alla direzione di propagazione.



Collegamento al laboratorio:
Nel nostro modello ML, "**ondulatorio**" e "**sussultorio**" rappresentano due pattern didattici di movimento distinti, laterale sul piano X–Y e verticale lungo l'asse Z, rilevati dall'accelerometro della micro:bit.

Elaborate

Dal modello ML al programma

Esportare/integrare il modello in **MakeCode** e programmare micro:bit affinché:

- **mostri un’icona** per ciascuna classe (Q / O / S)
- **emetta un suono** diverso
- **invii via radio** alla “centrale” (un altro micro:bit) il tipo di movimento.

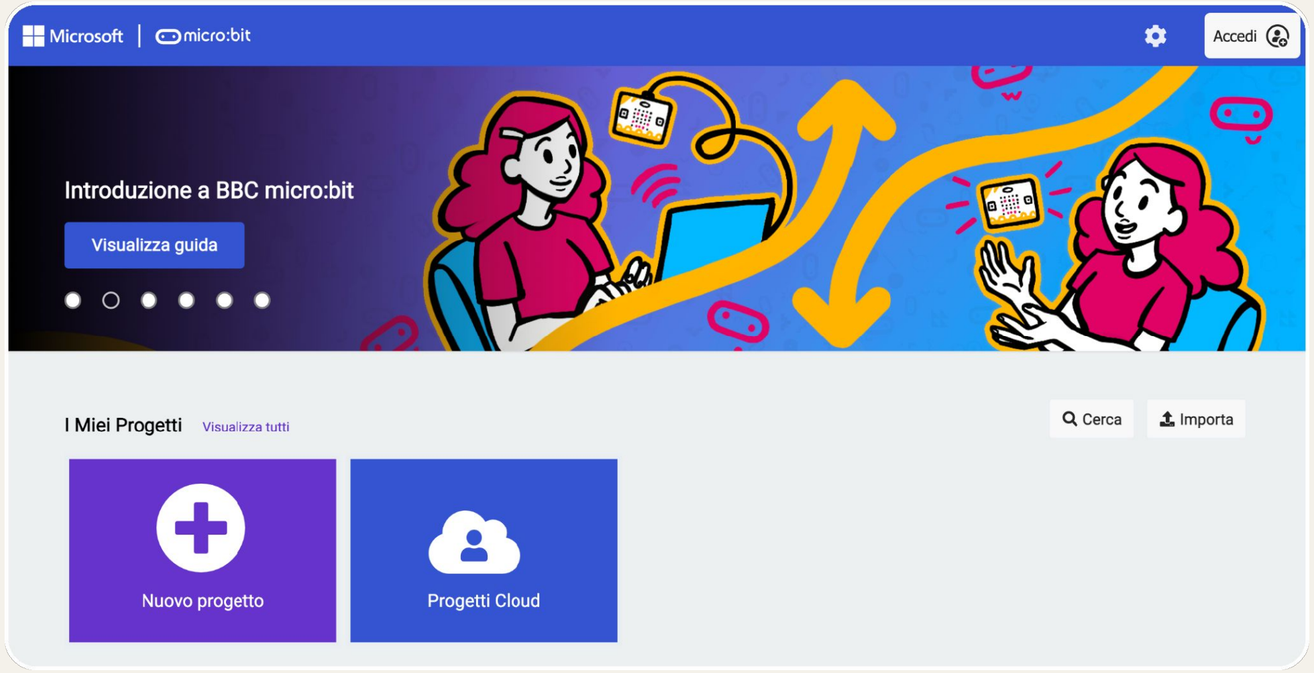


Microsoft MakeCode for micro:bit

Elaborate

L’ambiente di programmazione del sistema

- **Programmazione visuale a blocchi**
- Programmazione testuale in **Python**
- **Ambiente integrato per micro:bit**: simulatore, download del programma sulla scheda, gestione di input/output e radio



Tutorial e risorse operative

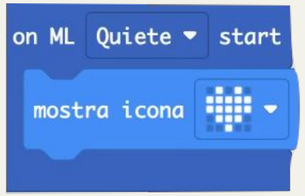
Un unico ambiente per programmare la micro:bit e controllare l’hardware.

Programmare gli eventi di classificazione

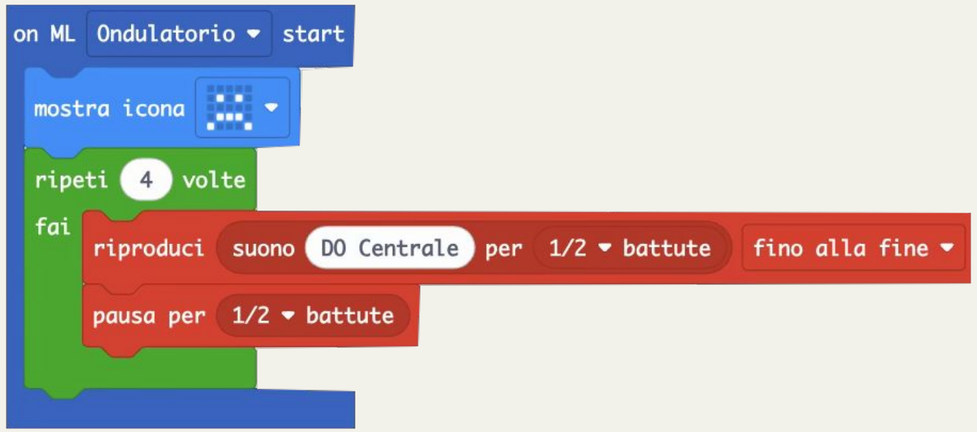
Elaborate

Ogni classe riconosciuta attiva un evento ML e un output specifico

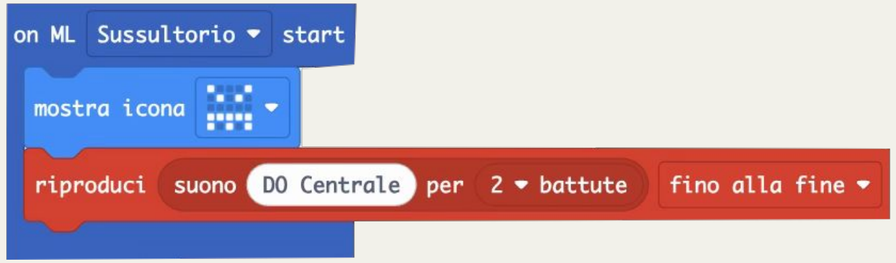
Quiete ⟶ nessun suono ⟶



Ondulatorio ⟶ bip intermittente ⟶



Sussultorio ⟶ segnale prolungato ⟶



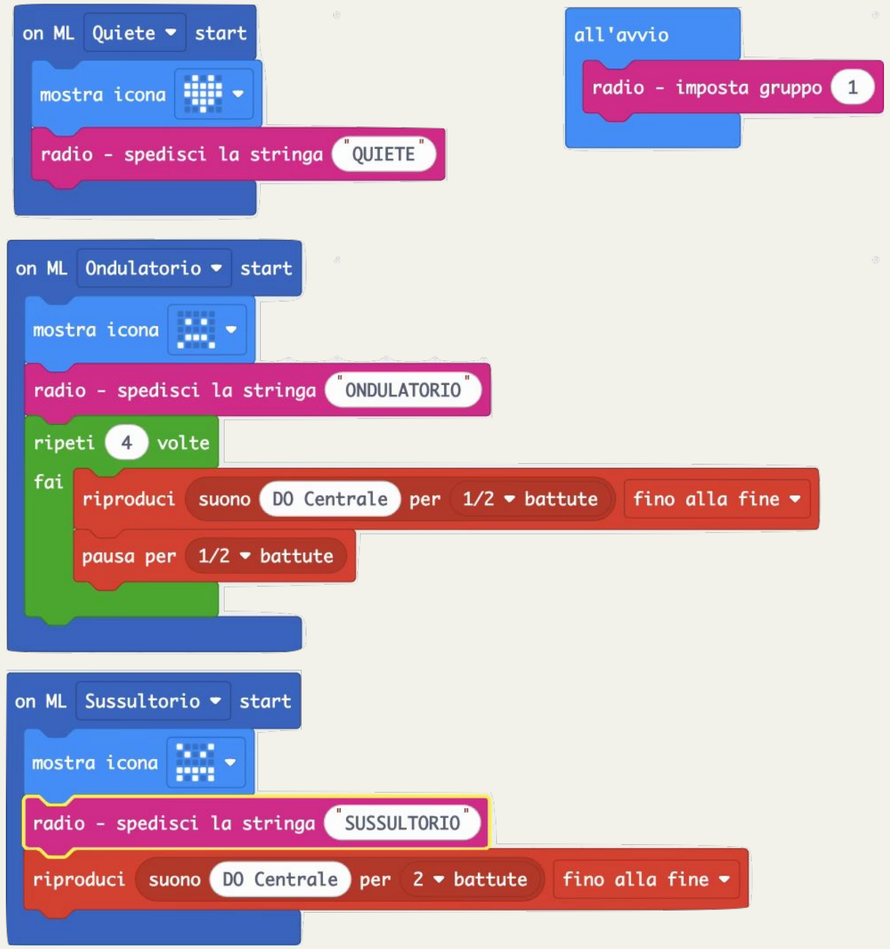
Il modello ML genera eventi; MakeCode associa a ciascun evento un’azione.

Programmazione radio tra le due micro:bit

Elaborate

La prima micro:bit riconosce la classe e invia il risultato alla seconda micro:bit, che funziona come stazione ricevente

- imposta radio gruppo 1
- invio di "QUIETE"
- invio di "ONDULATORIO"
- invio di "SUSSULTORIO"



Entrambe le micro:bit devono usare lo stesso radio group.

Programma completato → Download sulla micro:bit → Test locale → Programmazione della ricevente

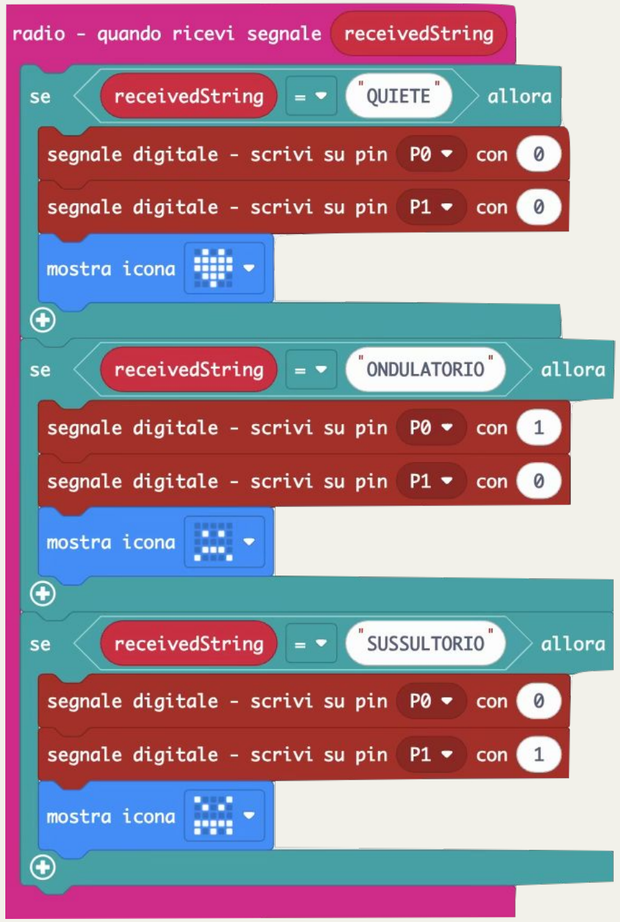
Programmazione stazione ricevente

Elaborate

La seconda micro:bit riceve il messaggio radio e attiva l'output corrispondente

Messaggio ricevuto → output

- **"QUIETE"** → LED spenti
- **"ONDULATORIO"** → LED giallo
  • P0 = 1
- **"SUSSULTORIO"** → LED rosso • P1 = 1





Trasmettitore e ricevitore: radio group 1

→ Download sulla ricevente
→ avvio delle due micro:bit
→ verifica di icone e LED

Messaggio ricevuto → confronto della stringa → uscita digitale → icona + LED

Dal prototipo al sistema completo

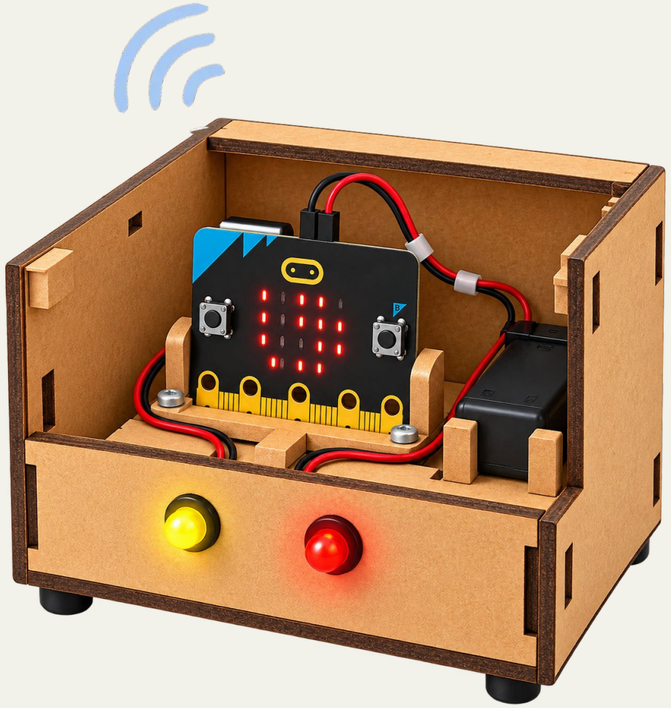
Elaborate

Una micro:bit rileva il movimento, l’altra riceve il segnale e attiva gli output

- **Stazione trasmittente**: micro:bit in posizione **orizzontale** all’interno della scatola, per rilevare i pattern di movimento simulati.
- **Stazione ricevente**: seconda micro:bit con **LED esterni** per visualizzare il tipo di segnale riconosciuto.
- Le due schede comunicano via **radio group**, realizzando un semplice sistema di monitoraggio.



Stazione trasmittente



Stazione ricevente

Tinkering

Evaluate

Valutazione del processo didattico

Osservare, documentare e riflettere sull’esperienza di laboratorio

- **Osservazione in itinere**: partecipazione, collaborazione, comprensione del funzionamento del sistema.
- **Prodotto finale**: correttezza del modello, programmazione in MakeCode, funzionamento della comunicazione radio.
- **Riflessione metacognitiva**: cosa ho imparato, quali difficoltà ho incontrato, come migliorerei il prototipo.



scheda metacognitiva studenti
scheda di osservazione docente

La valutazione considera sia il prodotto finale sia il percorso di progettazione, test e revisione.

Possibili sviluppi e curvature

Dal laboratorio con micro:bit a percorsi progressivamente più avanzati

Secondaria di I grado

Possibili ampliamenti

- **Coding:** dai blocchi a Python per gli alunni più esperti
- **Output:** LED, buzzer, display e altri dispositivi
- **Dataset:** nuovi campioni, test e miglioramento del modello
- **Prototipo:** estensione con nuovi output, display e più nodi radio

Secondaria di II grado

Curvatura embedded + ML

- Arduino Nano 33 BLE Sense Rev2;
- Acquisizione dati accelerometrici X, Y, Z
- Training, test e Live classification in **Edge Impulse**
- deployment del modello;
- programmazione nell'IDE Arduino;
- output su display OLED.

Approfondimento: confrontare window size diverse (1 s • 3 s • 5 s)

Nota metodologica sull'uso dell'intelligenza artificiale
Alcune immagini della presentazione sono state generate con strumenti di intelligenza artificiale. Progettazione didattica, contenuti, verifica e controllo finale restano a cura della docente.