

UNIVERSITÀ DI PADOVA
Dipartimento
di Ingegneria
dell'Informazione

Edurobotics
IAS Lab

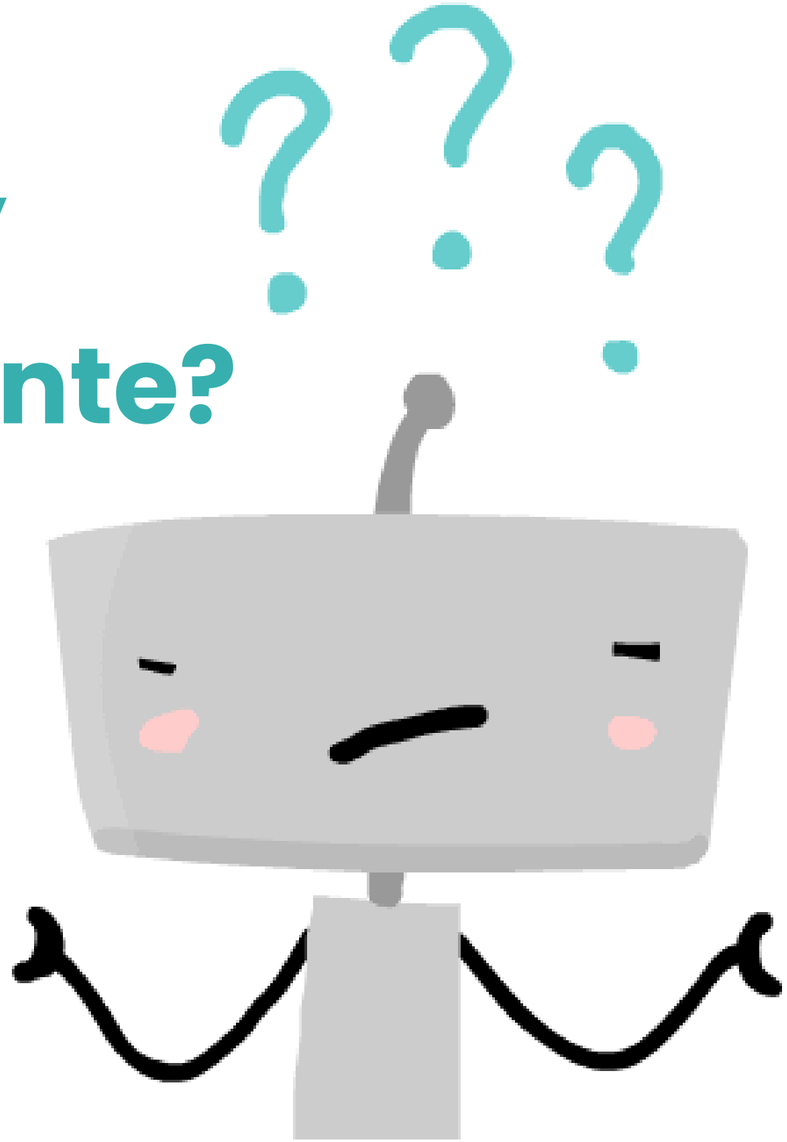
I robot possono imparare a riconoscere dei colori mai visti?

A cura di:
Dott. Daniel Fusaro
Dott.ssa Laura Cesaro
Prof. Emanuele Menegatti

[ITADINFO]

Padova, 10 ottobre 2026

Se dico “robot”
cosa vi viene in mente?



[ITADINFO]

Edurobotics
IAS Lab









[ITADINFO]

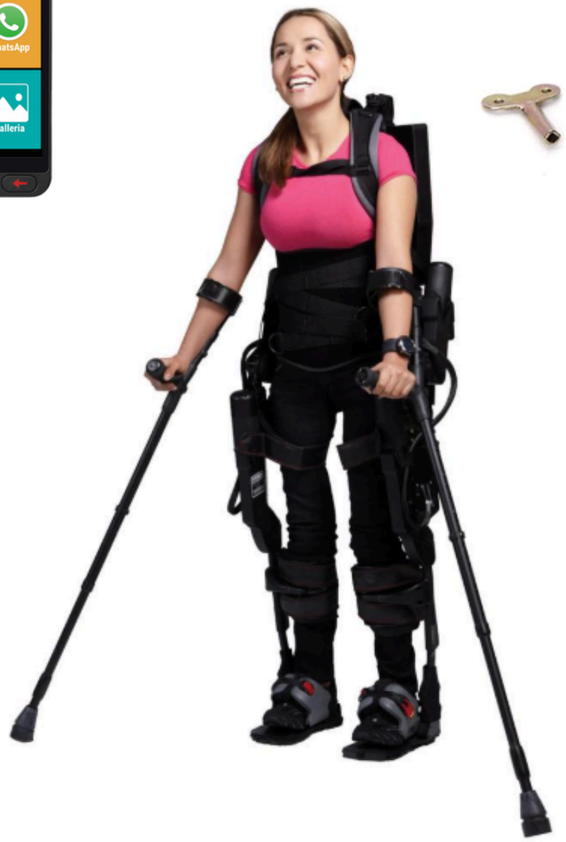
Edurobotics
IAS Lab

È un robot?











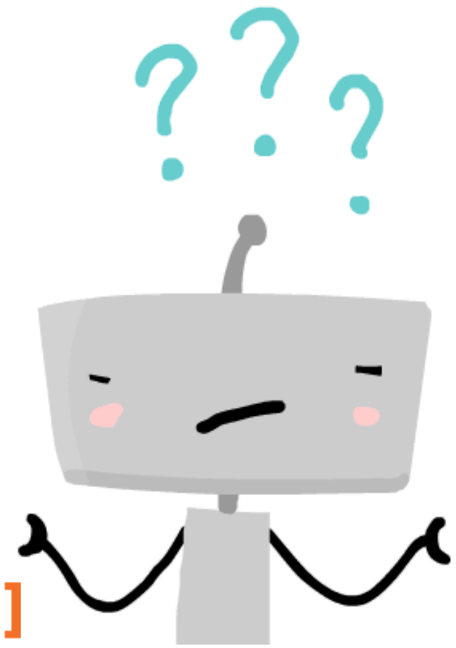


[ITADINFO]

Edurobotics
IAS Lab

Ma allora... che cos'è un robot?

Ogni robot è una **MACCHINA** che ha questi componenti di base:



SENSORI

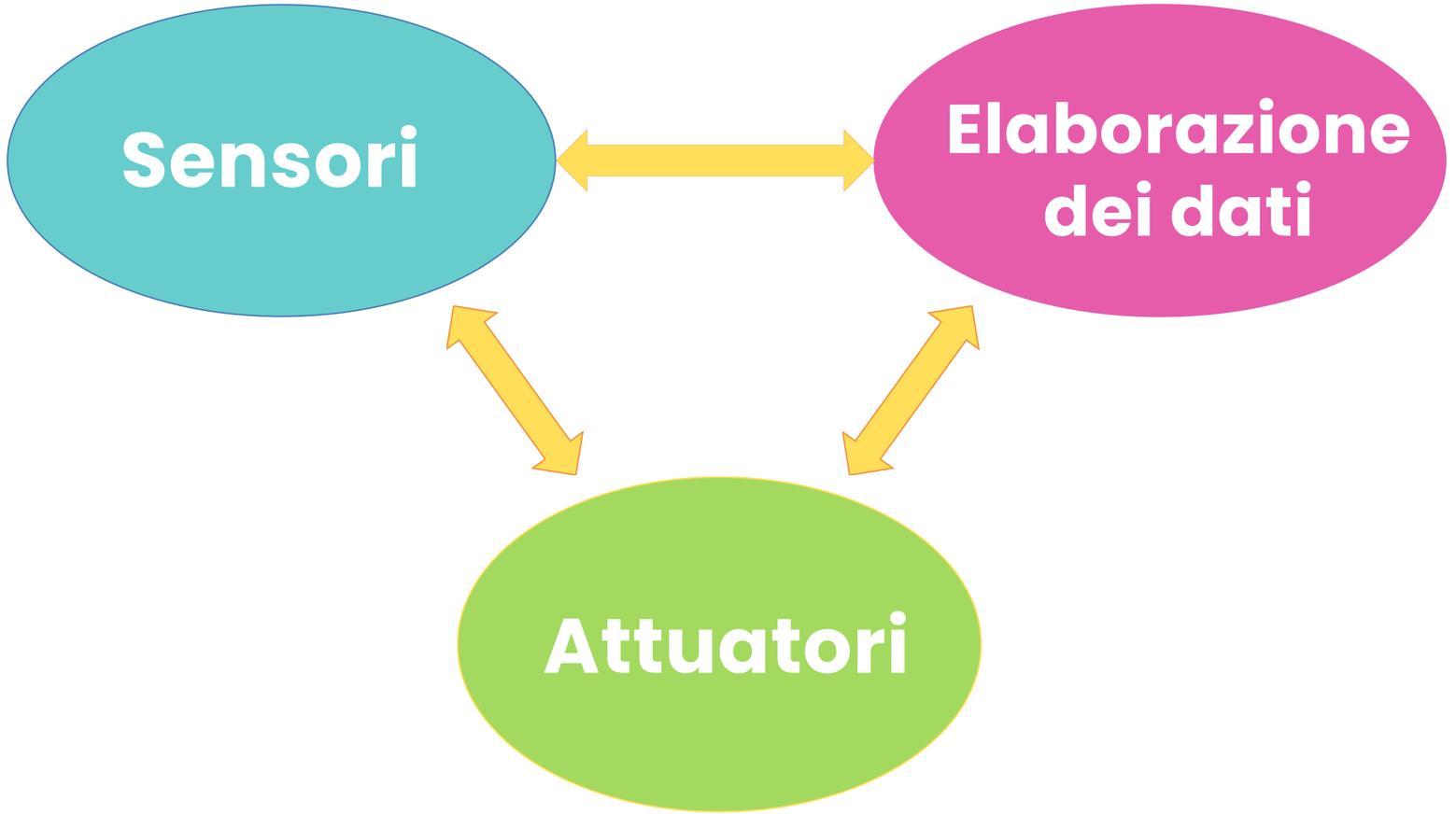
SCHEDA DI CONTROLLO

ATTUATORI

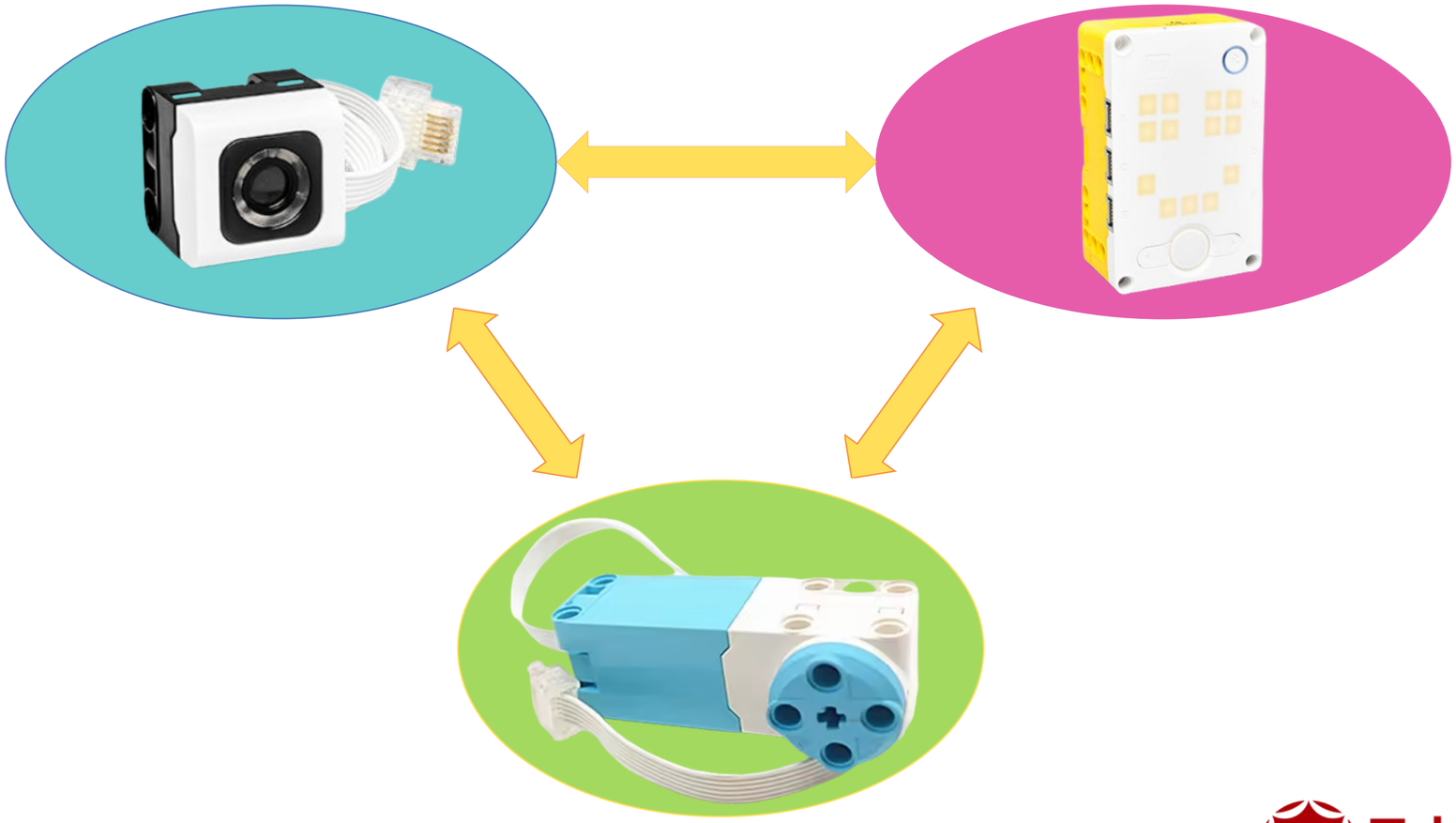
[ITADINFO]

Edurobotics
IAS Lab

Elementi essenziali di un robot



Elementi essenziali di un robot

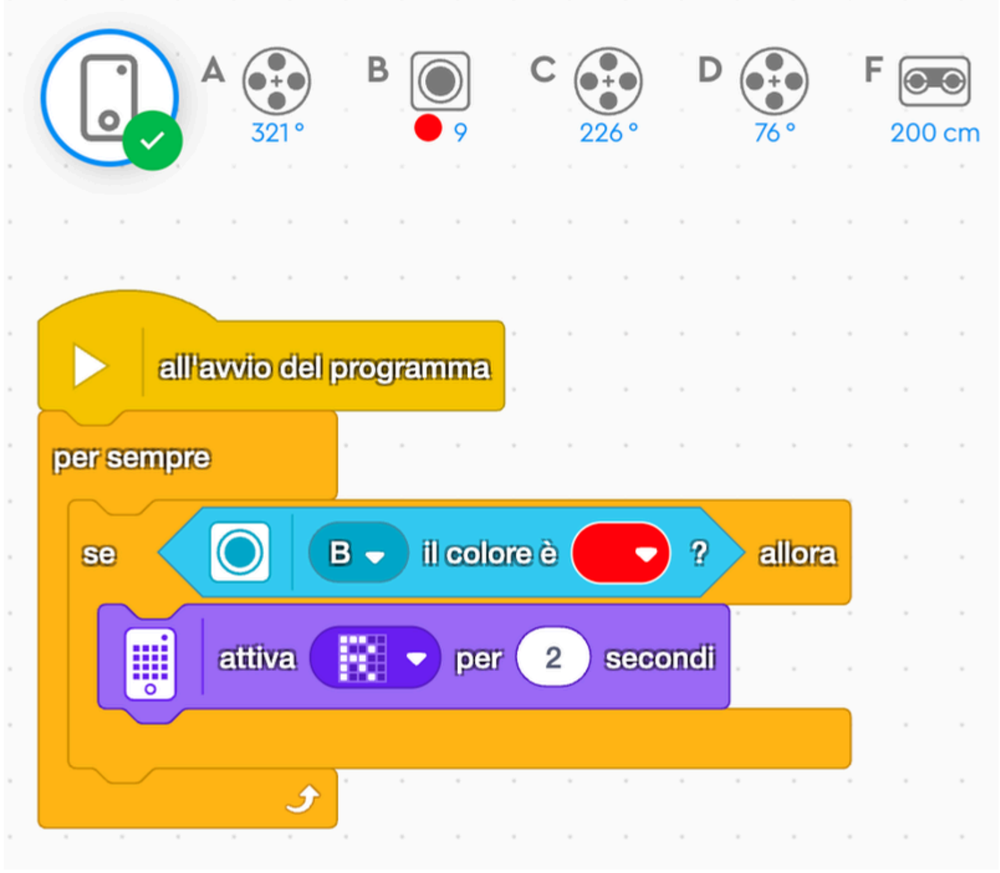


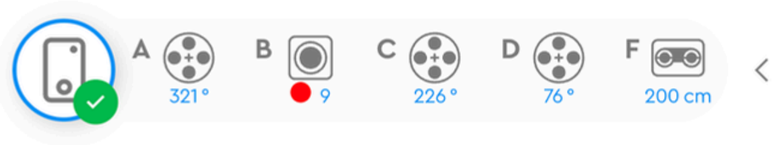
[ITADINFO]

Edurobotics
IAS Lab

Come programmiamo un robot?

BLOCCHI PAROLE





PYTHON

```python
from hub import port, sound
from hub import light_matrix
import color
import color_sensor
import runloop

async def main():
    # write your code here
    while True:
        if color_sensor.color(port.B) == color.RED:
            light_matrix.write("R")
        else:
            light_matrix.write(" ")
        await runloop.sleep_ms(5)

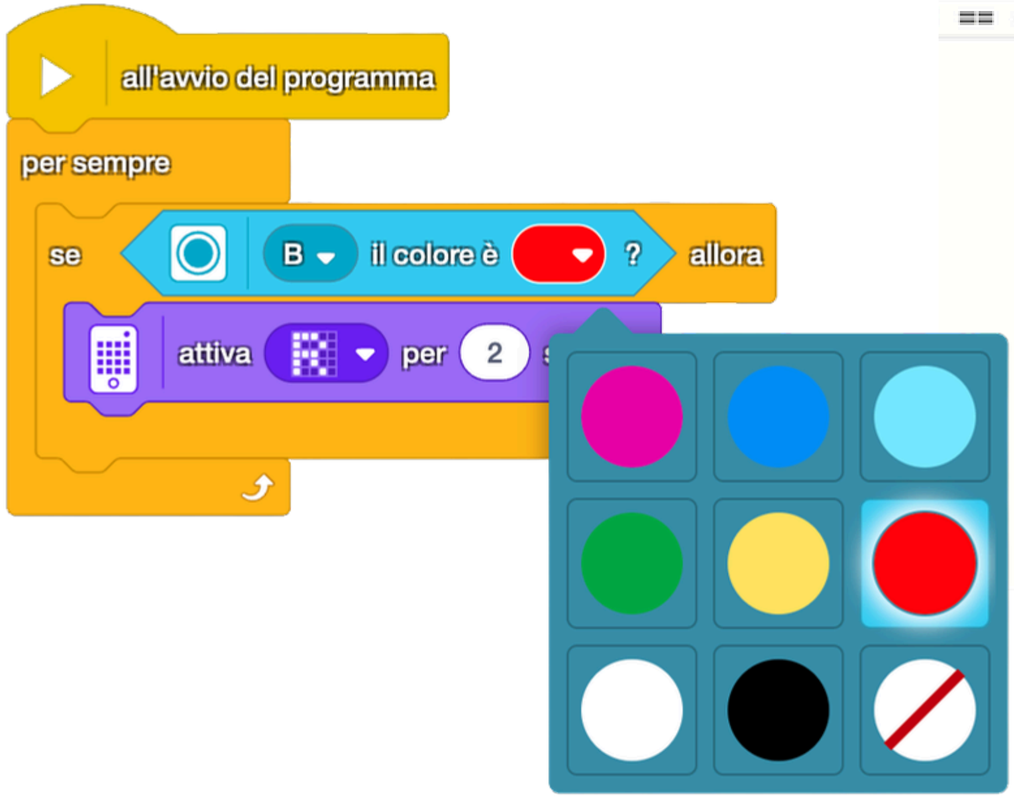
runloop.run(main())
```

[ITADINFO]

Edurobotics
IAS Lab

Quanti colori riconosce il nostro robot?

BLOCCHI PAROLE




```
== color.:
```

- AZURE
- BLACK
- BLUE
- GREEN
- MAGENTA
- ORANGE
- PURPLE
- RED
- TURQUOISE
- UNKNOWN
- WHITE
- YELLOW

PYTHON

[ITADINFO]

Edurobotics
IAS Lab

Ma il mondo è pieno di sfumature di colori!



Giorgione, Pala di Castelfranco, 1503 ca.

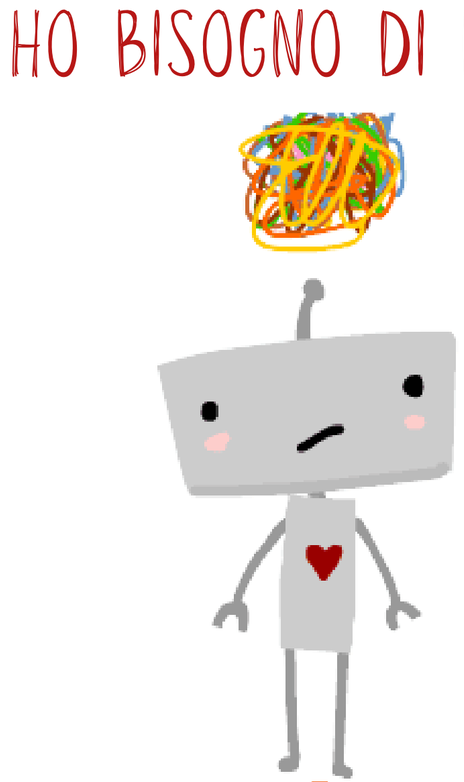
Quante sfumature di **rosso**?

Quante sfumature di **verde**?

[ITADINFO]

Edurobotics
IAS Lab

E se avessimo bisogno che il robot imparasse colori diversi?





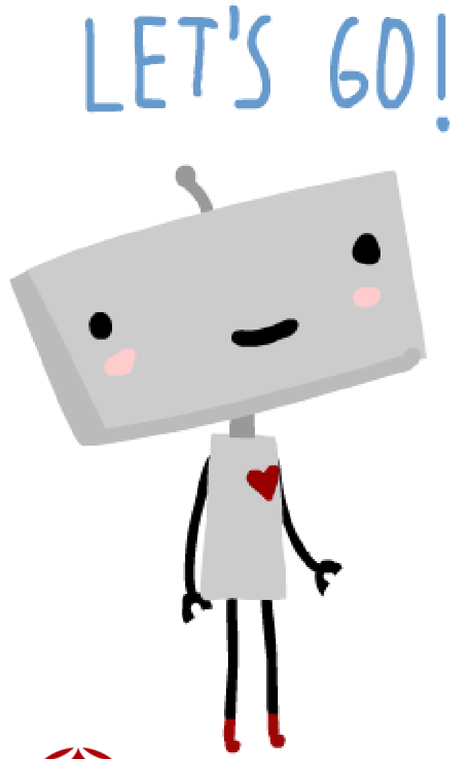
[ITADINFO]

Edurobotics
IAS Lab

Cosa possiamo fare?

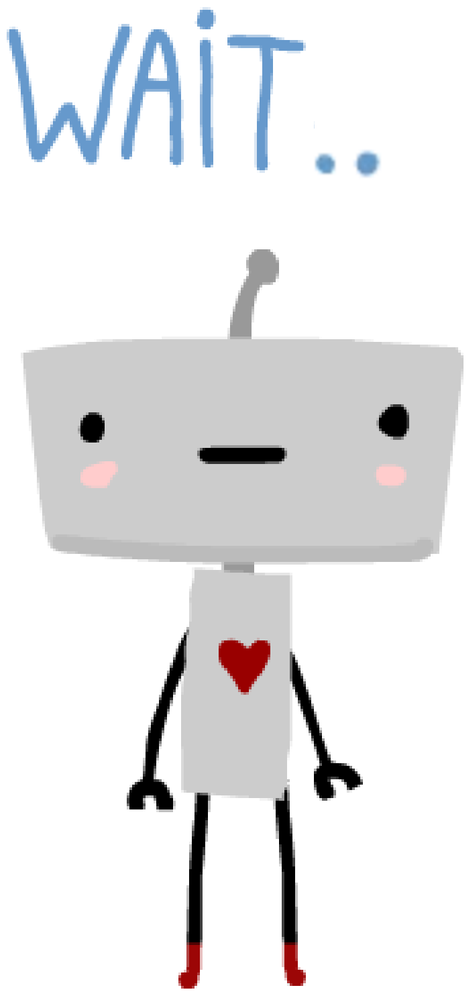
Dobbiamo costruire un **algoritmo che impara** ad **associare** i numeri **letti da un sensore** di colore **ai colori corrispondenti**.

In questo modo, il modello impara a riconoscere un colore partendo solo dai valori numerici forniti dal sensore.



[ITADINFO]

Edurobotics
IAS Lab



E come si fa?

Possiamo usare l'Intelligenza Artificiale!

In particolare usiamo una tecnica chiamata:
Machine Larning (Apprendimento Automatico)

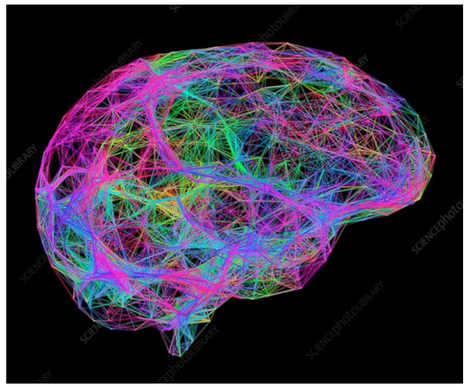
Tra tutte le tecniche di Machine Learning che esistono, oggi useremo le **Reti Neurali.**

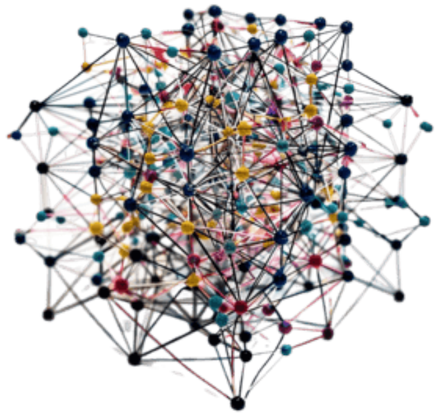
Ma... cosa sono le Reti Neurali?

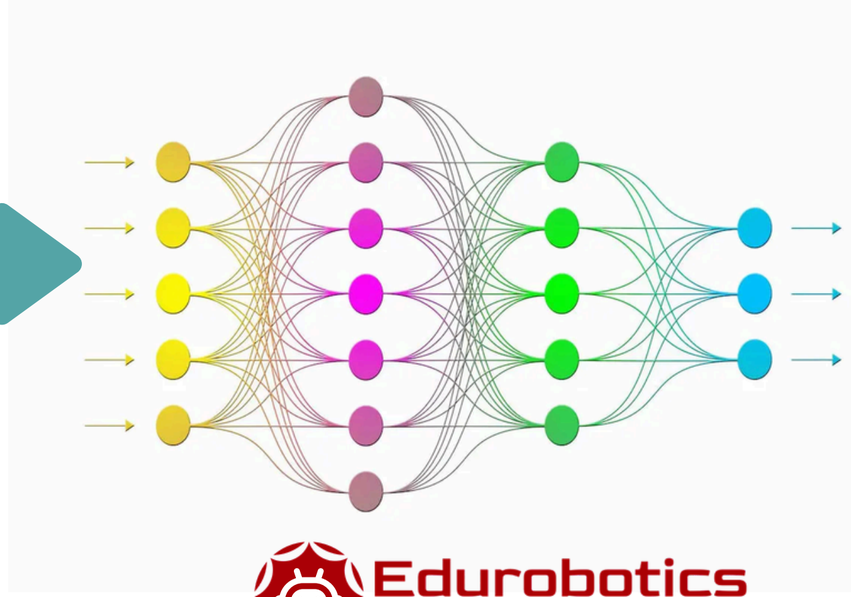
[ITADINFO]

Edurobotics
IAS Lab

Cos’è una rete neurale?







Una **rete neurale** è

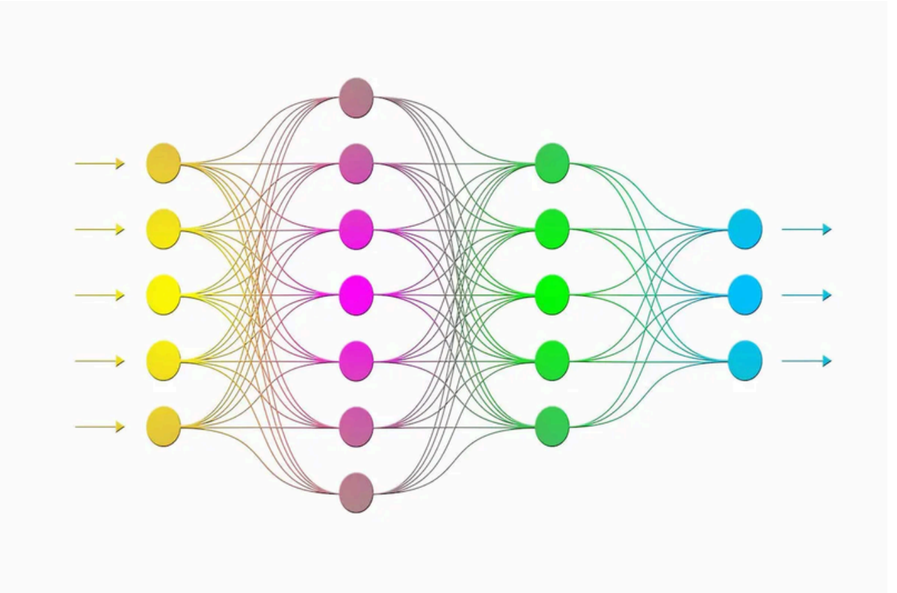
- un **modello** matematico
- ispirato al **cervello umano**
- costituito da **neuroni interconnessi** organizzati in **strati**.

[ITADINFO]

Edurobotics
IAS Lab

Cos’è una rete neurale?

- I **neuroni** sono nodi interconnessi che ricevono input, li elaborano e producono un output.
- Il primo strato è **l’input** della rete
- l’ultimo strato è **l’output** della rete
- gli strati intermedi rappresentano **risultati parziali** calcolati dalla rete
- Le connessioni tra i nodi sono chiamate “**pesi**”, e rappresentano l’importanza del nodo di partenza nel calcolo del nodo di arrivo

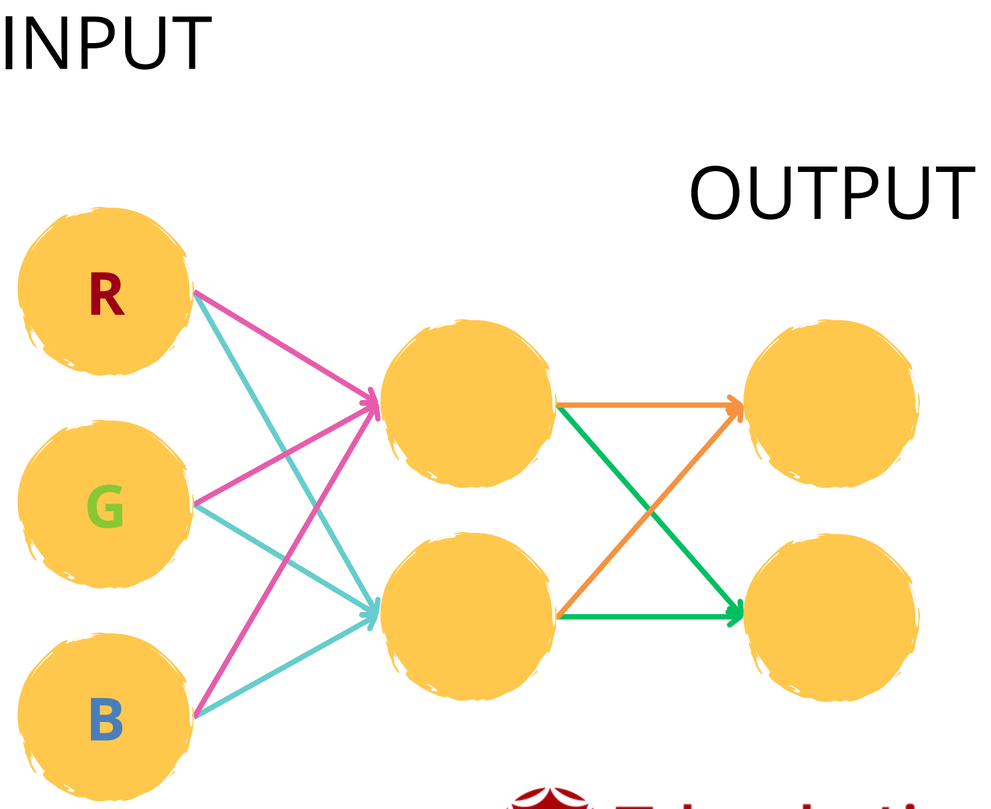


[ITADINFO]

Edurobotics
IAS Lab

Facciamo un esempio

- Creiamo una rete che classifica il colore di un pixel dell'immagine. Abbiamo solo due classi: "**Rosso**" oppure "**Non rosso**"
- **input** della rete: 3 neuroni (R, G, B)
- **output** della rete: 2 neuroni (Rosso, Non rosso)
- **hidden** layers: 1 strato da 2 neuroni



[ITADINFO]

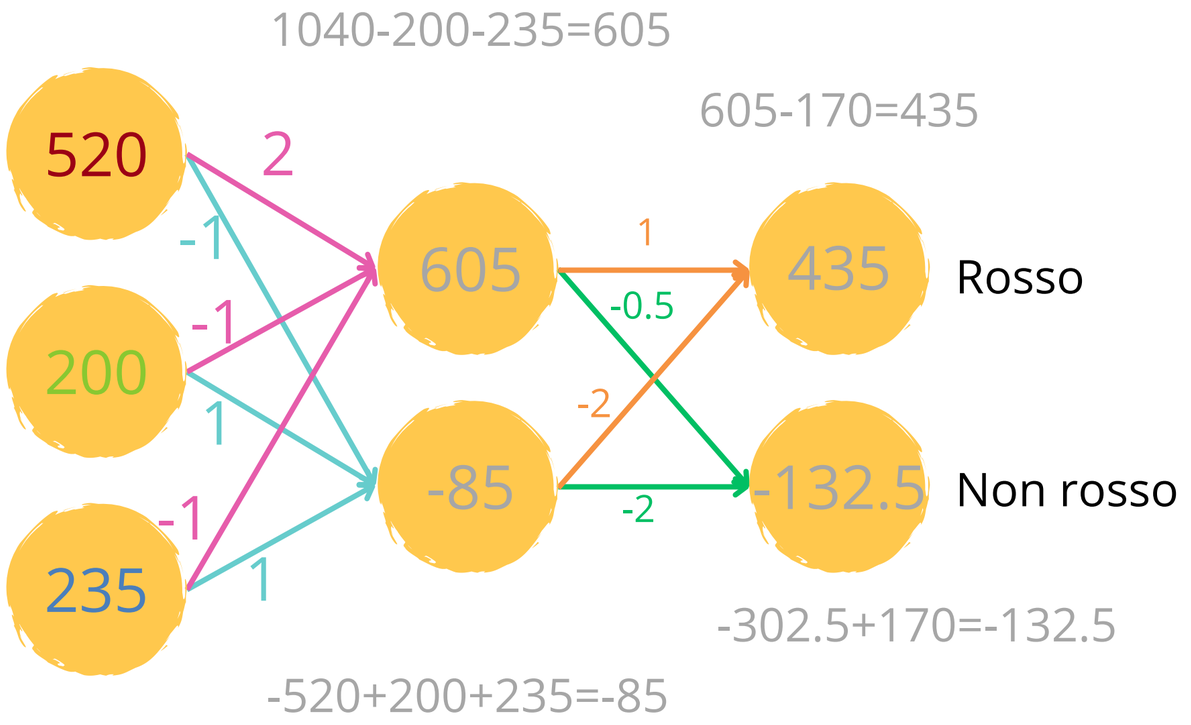
Edurobotics
IAS Lab

Facciamo un esempio

INPUT

R **G** **B**

520, **200**, **235**



435 > -132.5

...e quindi

ROSSO

[ITADINFO]

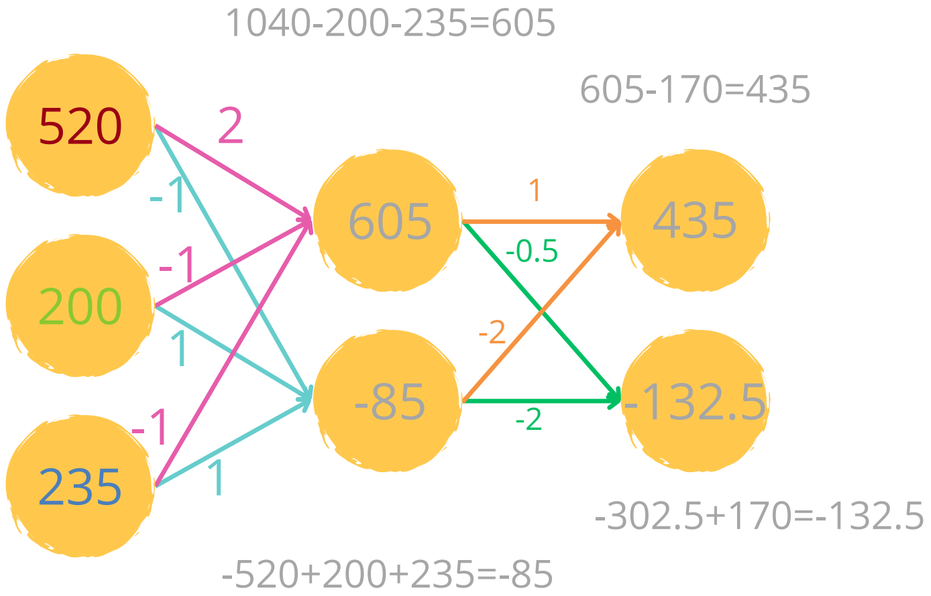
Edurobotics
IAS Lab

Ma i *pesi* da dove vengono?

I pesi si calcolano **allenando (training)** la rete:

1. raccogliamo tanti esempi (**dataset**) di letture RGB
2. ogni lettura RGB deve essere associata al valore corretto di **rosso**/**non rosso** (**ground truth**)
3. Attraverso un **algoritmo** (**back-propagation**) possiamo trovare i pesi migliori per risolvere il nostro problema

Durante l'allenamento, la rete commette un errore (**loss**) sempre più piccolo.



Una volta allenata la rete, possiamo usarla per riconoscere i colori!

[ITADINFO]

Edurobotics
IAS Lab

Imparare con le reti neurali







Materiali:

- Lego Spike Hub
- Sensore di colore
- Motori

Obiettivo:

Creare un **modello** (rete neurale) capace di associare dei **numeri** (letture del sensore di colore) a una serie di **colori**



R G B

520, 200, 235





[ITADINFO]

Edurobotics
IAS Lab

Mettiamoci al lavoro!

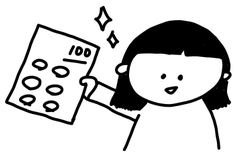
Obiettivo:

Creiamo una **rete neurale** capace di associare letture **RGBI** a una serie di **colori** (4 + lo sfondo *(background)*)

Metodo:



1. Raccogliamo il **dataset** con cui allenare la rete



2. **Alleniamo** la rete 
3. **Valutiamo** il modello
4. **Usiamola** per un problema pratico



Faremo tutto utilizzando il **Lego Spike Prime**

tiny.cc/s4a25

[ITADINFO]

Edurobotics
IAS Lab

1. Raccolta del dataset

Connettiamo l'hub e usiamo il file

1_create_dataset

Per ogni colore che abbiamo deciso di insegnare al robot, dobbiamo raccogliere degli esempi ***etichettandoli***, cioè scrivendo il colore corrispondente al posto di "**color**" alla riga 36 e premendo ***play*** ogni volta. Annotiamo su un foglio i nomi delle etichette (consigliato usare solo minuscole)

PS. raccogliete per primo il background ;)

```
7  ######## PARAMETERS ########
8  N_samples  = 50
9  move_robot = True
10 ############################
11
12 async def create_dataset(color):
13     global N_samples, move_robot
14     filename = "/flash/{:s}.csv".format(color)
15     f_out = open(filename, "w")
16     for i in range(N_samples):
17         rgbi = color_sensor.rgbi(port.B)
18         s = "{:d} {:d} {:d} {:d}\n".format(rgbi[0], rgbi[1], rgbi[2], rgbi[3])
19         f_out.write(s)
20         if i % 10 ==0:
21             print("{:.2f}% ".format(i/N_samples*100), end="")
22         await runloop.sleep_ms(100)
23     print("100.00 %")
24     print("Fatto! Ho salvato il dataset per il colore", color, "qui:", filename)
25     move_robot = False
26
27 async def move():
28     global move_robot
29     degrees = 180
30     while move_robot:
31         await motor_pair.move_for_degrees(motor_pair.PAIR_1, degrees, 0)
32         await runloop.sleep_ms(100)
33         degrees*=-1
34
35 # colori di esempio: verde, marrone, arancio, blu, background
36 color = "background"
37
38 runloop.run(move(), create_dataset(color))
```



[ITADINFO]

Edurobotics
IAS Lab

Un esempio di dataset

s4a > data > arancio.csv

1	1018 371 363
2	1002 366 357
3	1004 364 356
4	1000 367 358
5	987 358 348
6	994 362 356
7	1003 360 352
8	995 359 352
9	1014 371 364
10	1018 379 372
11	1018 391 383
12	1018 389 383
13	1020 392 386

s4a > data > blu.csv

1	14 17 21
2	14 17 20
3	13 16 19
4	14 17 20
5	13 16 19
6	13 16 19
7	13 17 20
8	14 18 21
9	13 17 20
10	13 17 20
11	14 17 21
12	14 18 21
13	14 17 21

s4a > data > background.csv

1	918 960 982
2	907 951 968
3	896 938 949
4	897 939 955
5	903 946 964
6	909 952 973
7	896 935 951
8	880 921 940
9	899 940 960
10	906 952 974
11	906 946 964
12	902 945 962
13	907 950 967

2. Alleniamo il modello



Usiamo il file **2_train**

Questi sono i **parametri**, cioè le impostazioni del modello.
Le **EPOCHS** (epoche) sono il numero di volte in cui il modello rivede tutti i dati di allenamento.
Se il modello non funziona come dovrebbe, provate a modificare **INITIAL_LR**

```
colors = ["arancio", "marrone", "verde", "blu", "background"]

######### PARAMETERS #########
INPUT_SIZE= 3                # il numero di dati di input che il modello riceve (RGB => 3)

NUM_CLASSES= len(colors)        # il numero di colori che il modello sarà in grado di riconoscere

INITIAL_LR= 0.02                # più alto è, più impara in fretta, ma meno "attentamente"
                                # mantenetelo minore di 1, ma maggiore di 0.0001

EPOCHS_NUM= 50                # questo è il numero massimo di epoche per cui alleneremo il modello

TRAIN_SIZE, VAL_SIZE = 0.5, 0.2    # con 0.5 e 0.2 vogliamo dire che useremo il 50% dei dati per allenare,
                                    # il 20 % per validare e il restante (30%) per testare il modello

MODELNAME    = "/flash/simple_nn_model_new.ckpt"# salveremo qui il modello allenato
##############################
```

[ITADINFO]

Edurobotics
IAS Lab

2. Alleniamo il modello



I dati raccolti vengono divisi in 3 parti:

1. train
2. validation
3. test

Durante l'allenamento, viene usato il **train set** per trovare i pesi migliori e il **validation set** per verificare le prestazioni di questi pesi, in modo da commettere un **errore** sempre più piccolo.

Il **train set** serve per "studiare"

Il **validation set** serve a "verificare" le conoscenze

```
Epoch 1 | Train Loss: 1.0811 | Valid Loss: 1.1951 * | Train Acc: 68.00 | Val Acc: 48.00
Epoch 2 | Train Loss: 0.8758 | Valid Loss: 0.9841 * | Train Acc: 47.20 | Val Acc: 24.00
Epoch 3 | Train Loss: 0.7746 | Valid Loss: 0.8600 * | Train Acc: 47.20 | Val Acc: 24.00
Epoch 4 | Train Loss: 0.6912 | Valid Loss: 0.7566 * | Train Acc: 64.00 | Val Acc: 56.00
Epoch 5 | Train Loss: 0.6161 | Valid Loss: 0.6651 * | Train Acc: 96.00 | Val Acc: 92.00
Epoch 6 | Train Loss: 0.5498 | Valid Loss: 0.5880 * | Train Acc: 100.00 | Val Acc: 100.00
Fermo l'allenamento: abbiamo raggiunto 100% di accuracy in validation.
Accuracy finale del modello: 100.0
```

PS. La accuracy misura quante risposte giuste vengono date, in percentuale.

3. Valutiamo il modello

Usiamo il file **3_test**

Avviando questo programma, usiamo il **test set**, composto da dati non usati per l'allenamento, per misurare le prestazioni del modello (**accuracy**) e la sua **confusione**.

```python
def test():
    # carichiamo il dataset
    all_data, all_labels = read_data(files)

    # dividiamo il dataset in train, validation e test
    (_, _), (_, _), (X_test, y_test) = \
            split_data(all_data, all_labels, TRAIN_SIZE, VAL_SIZE, test=True)

    del all_data, all_labels
    descrivi(y_test, "test", NUM_CLASSES)

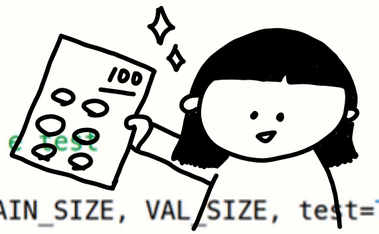
    # carichiamo il modello
    model = SimpleNN(input_size=INPUT_SIZE, num_classes=NUM_CLASSES)
    read_model(model, MODELNAME)
    print("Ho caricato il modello da qui:", MODELNAME)

    # misuriamo le prestazioni del modello
    print("Misuriamo le prestazioni del modello...")

    acc_test  = accuracy(model, X_test, y_test)
    print("Prestazioni in  test: {:.2f} %".format(acc_test * 100))

    confusion_matrix = get_confusion_matrix(model, X_test, y_test, NUM_CLASSES)
    stampa_confusion_matrix(confusion_matrix)

test()
```



[ITADINFO]

Edurobotics
IAS Lab

3. Valutiamo il modello

La **confusion matrix** è una tabella che mostra quante previsioni un modello ha indovinato e quante ne ha sbagliate, distinguendo i vari tipi di errore.

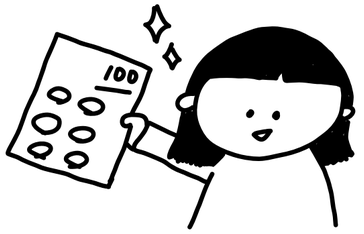
Nella diagonale ci sono tutte le previsioni corrette, mentre fuori dalla diagonale ci sono gli eventuali errori.

Descrizione del valid set: (che contiene 50 elementi)

Class	0	1	2	3	4
Train	6	14	11	9	10

Matrice della confusione del modello:

[6, 0, 0, 0, 0]
[0, 14, 0, 0, 0]
[0, 0, 5, 6, 0]
[0, 0, 0, 9, 0]
[0, 0, 0, 0, 10]



4. Usiamo il modello per un problema pratico

Usiamo il file **4_task**

A questo punto il nostro modello è pronto per essere messo alla prova:

1. scriviamo nella riga dei parametri i nomi che abbiamo dato ai colori costruendo il dataset
2. associamo ad ogni colore il comportamento che dovrà avere il nostro robot “vedendo” quel colore.

```
async def test():
    # carichiamo il modello
    model = SimpleNN(INPUT_SIZE, NUM_CLASSES)
    read_model(model, MODELNAME)

    while True:
        runloop.sleep_ms(200)

        # leggi dal sensore
        rgbi = get_color()

        # usiamo il modello per capire il colore
        color_idx, confidence = model.predict(rgbi)
        colore_predetto = colors[color_idx]

        if confidence >0:
            print("Ho visto il colore", colore_predetto, \
                  "(con fiducia {:.2f}%)".format(confidence))

            if colore_predetto=="orange":
                await vai_dritto()
            elif colore_predetto=="green":
                await vai_indietro()
            elif colore_predetto=="blue":
                await gira_a_destra()
            elif colore_predetto=="brown":
                await gira_a_sinistra()
```



[ITADINFO]

Edurobotics
IAS Lab

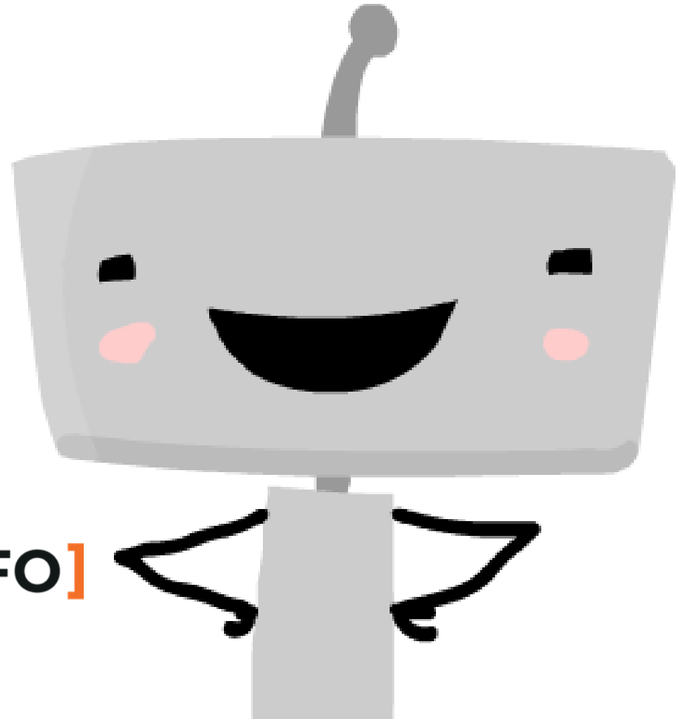
RECAP

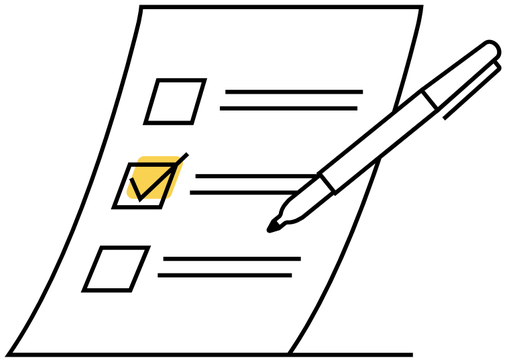
Cosa abbiamo imparato?

- I computer (e quindi anche i robot) possono **imparare** dai **dati in input**
- Il **Machine Learning** è una branca dell'Intelligenza Artificiale
- Le **Reti Neurali** sono una delle tecniche del Machine Learning
- Una rete neurale è un modello matematico ispirato al cervello umano che:
    1. impara cambiando i valori dei **pesi** (parametri)
    2. modifica i propri parametri in base ai **dati in ingresso**
    3. una volta allenata può **classificare** dati mai visti prima

Edurobotics
IAS Lab

prima di andare...

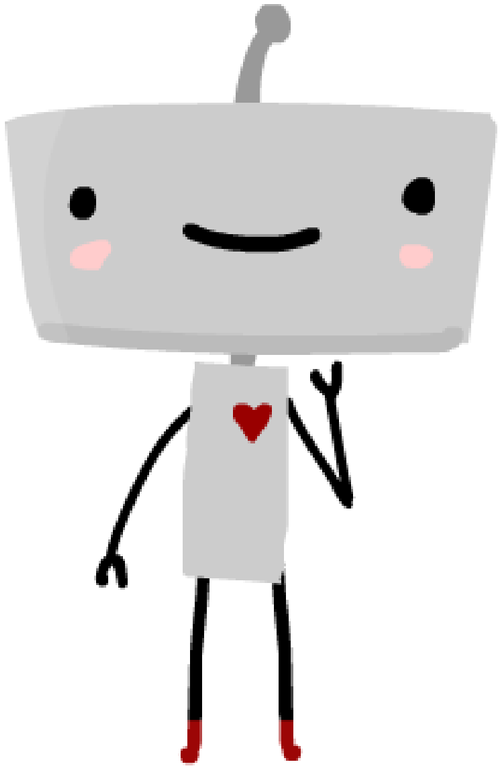




[ITADINFO]

fusarodani@dei.unipd.it
laura.cesaro@phd.unipd.it

GRAZIE!



[ITADINFO]

Edurobotics
IAS Lab

UNIVERSITÀ DEGLI STUDI DI PADOVA