

Physical Computing

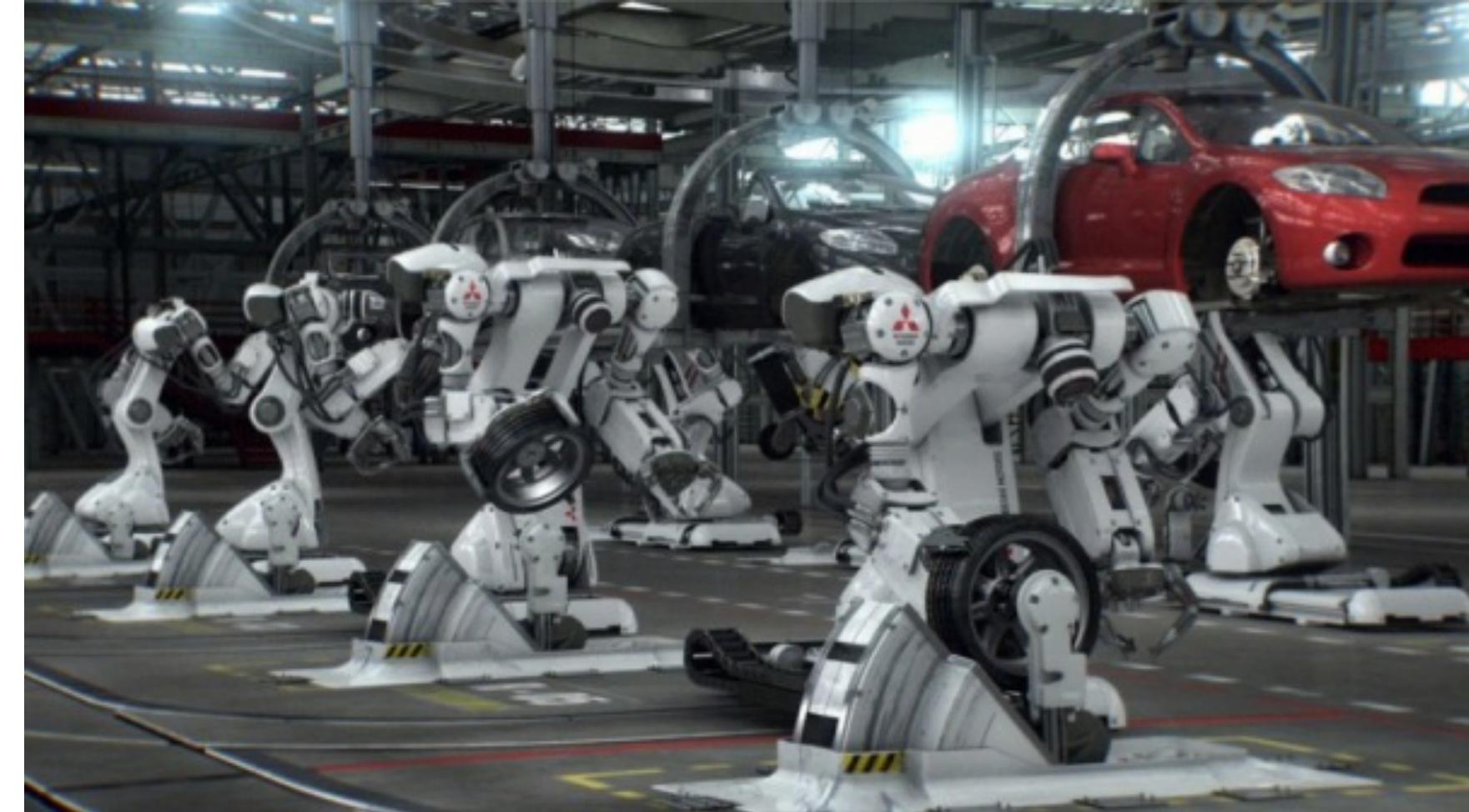
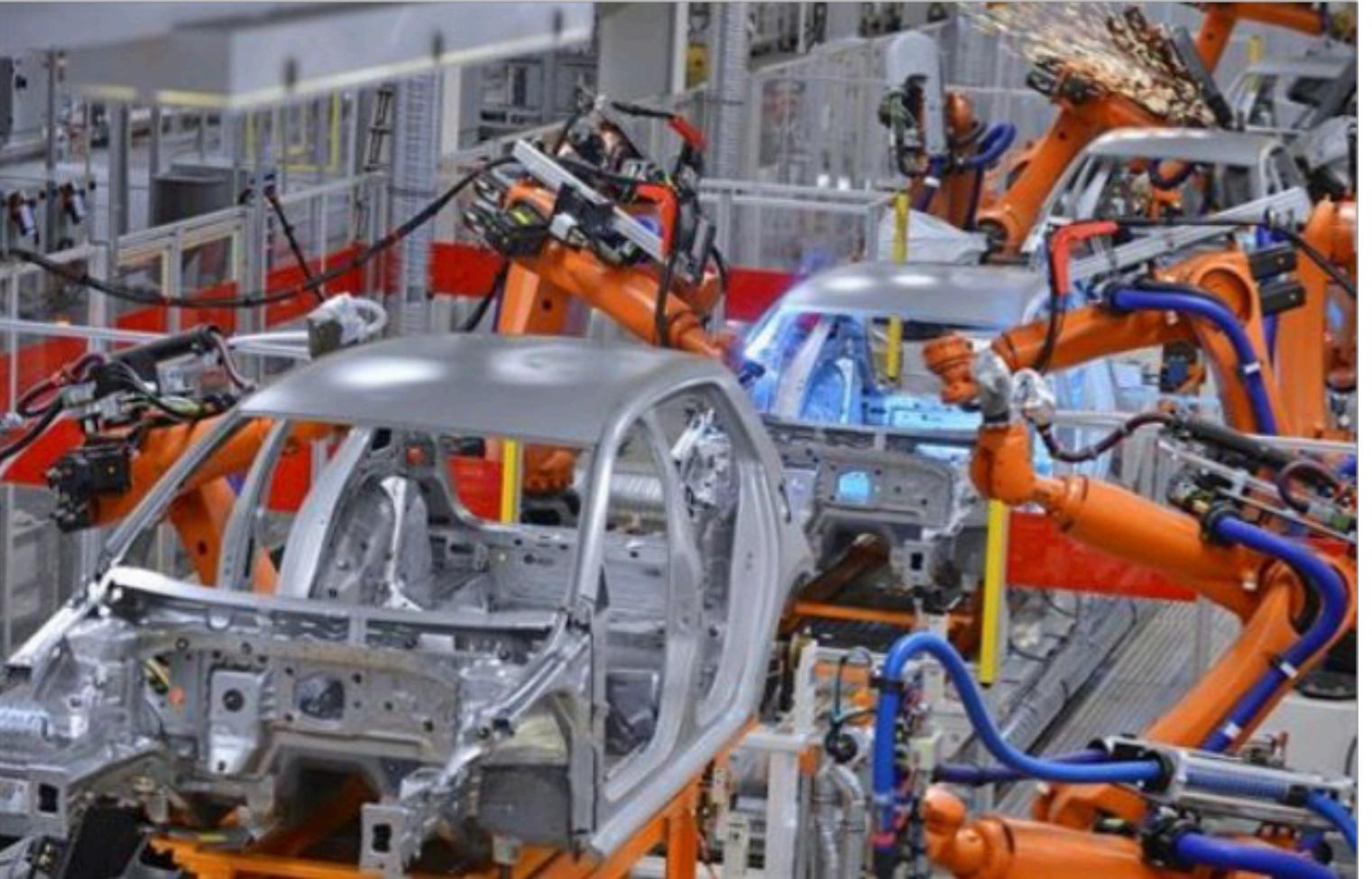
Controllare Minecraft con Raspberry Pi usando Python e dispositivi
elettronici

Fiera dell'Informatica elettronica e radiantismo
23 febbraio 2019

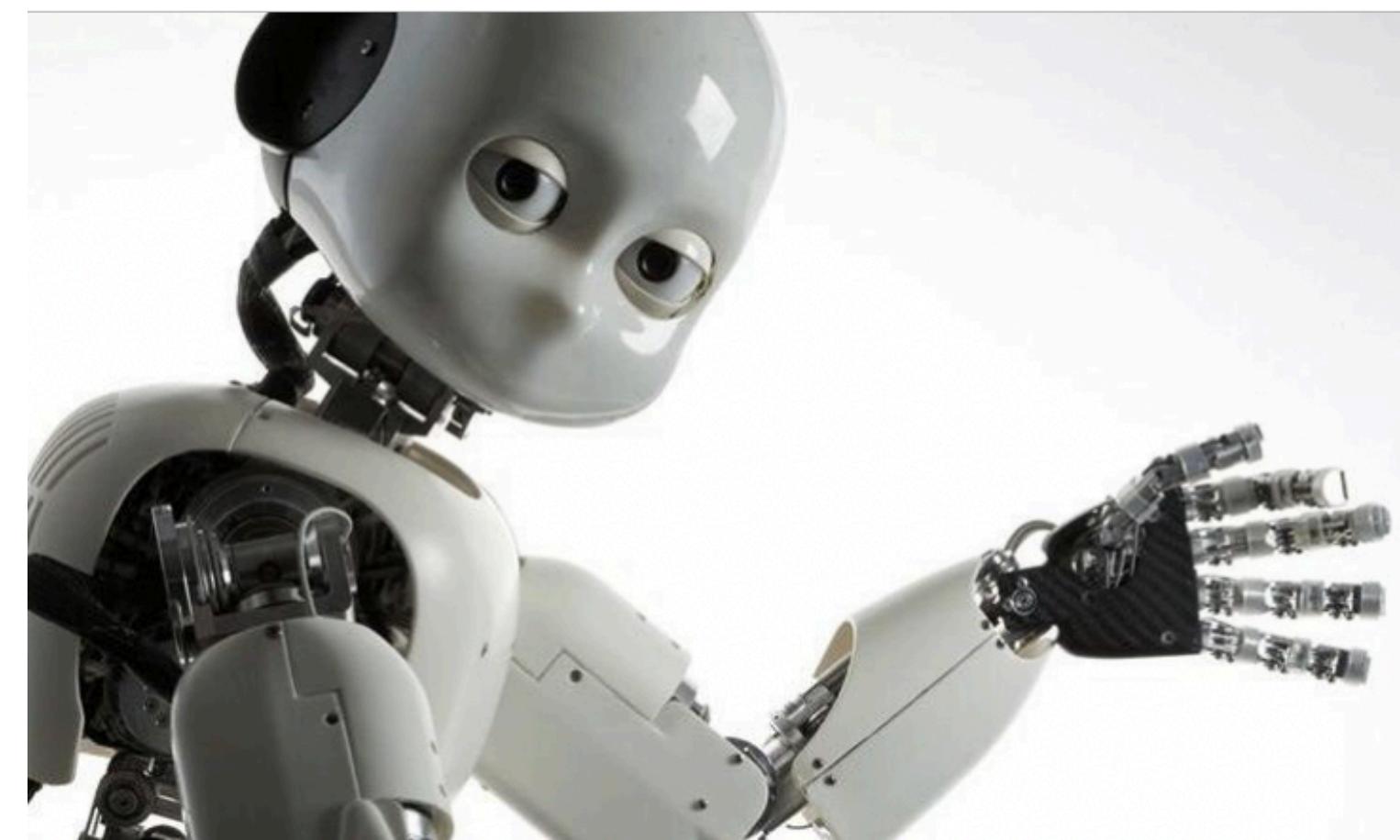
Introduzione

- Il robot è strumento didattico ad ampio spettro
- anche se la robotica è una disciplina a sé stante, **non è richiesta una elevata competenza tecnica, il robot come macchina non è il principale oggetto di studio**

Robot industriali

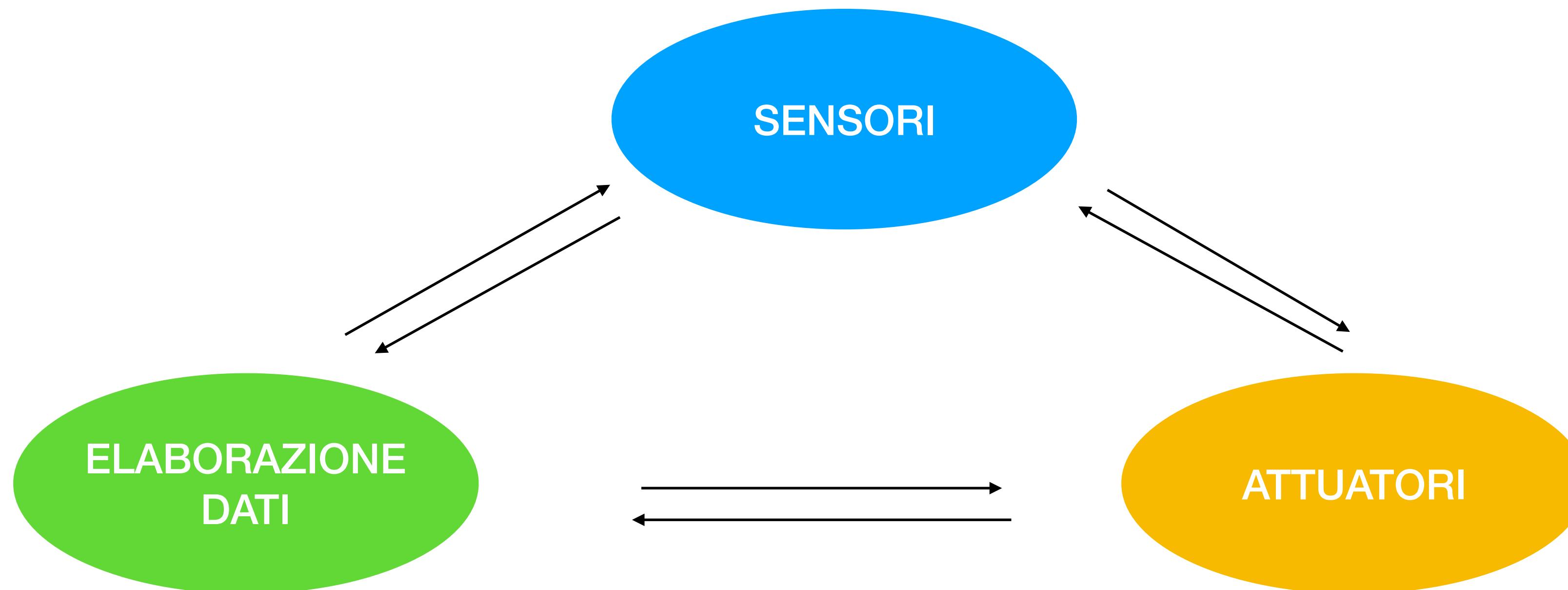


Robot umanoidi



Robotica laboratoriale

- Nella Robotica laboratoriale un robot è costituito da **tre elementi fondamentali**:

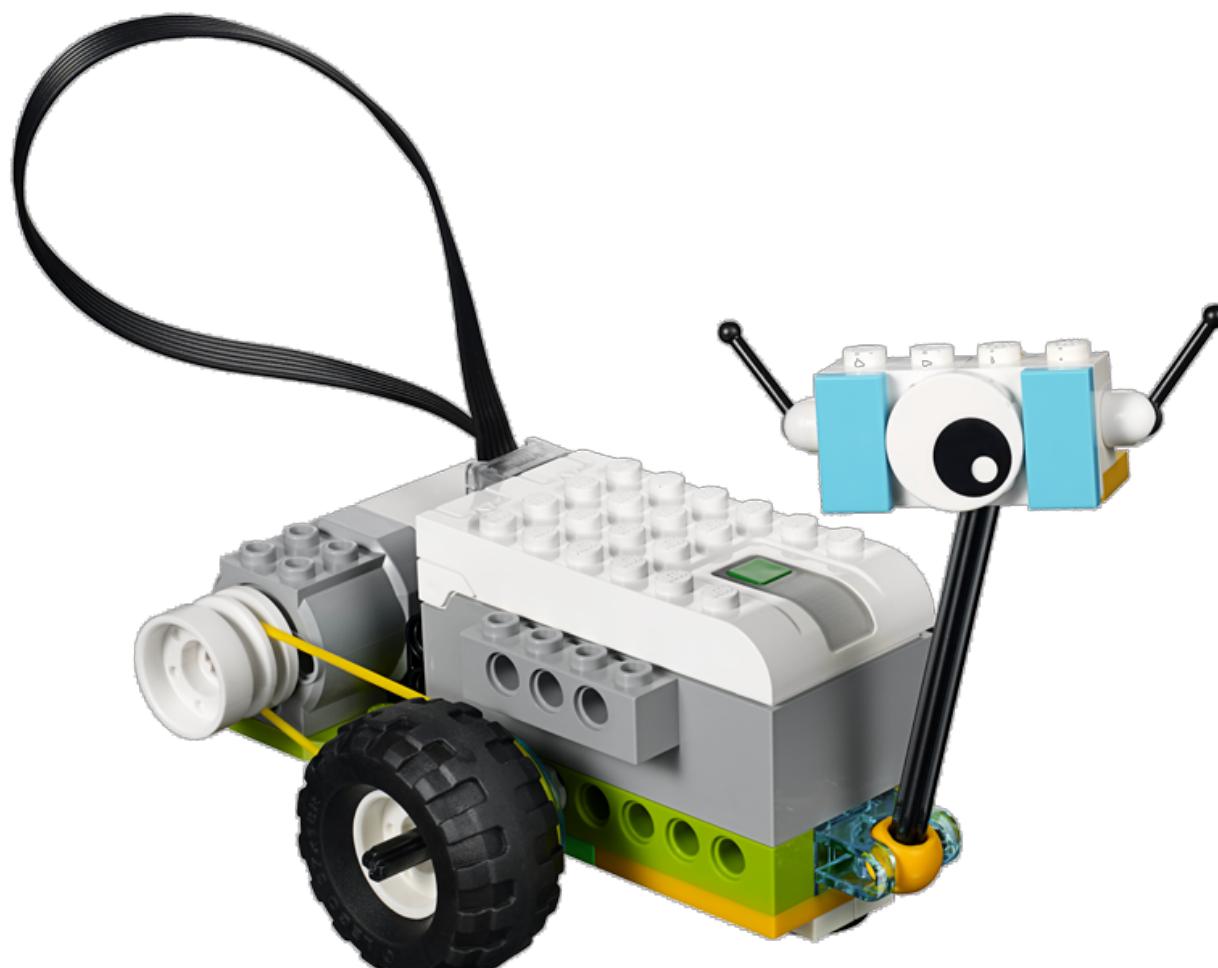


Robotica laboratoriale

- Abbiamo bisogno della parte **hardware**, ovvero la parte fisica costituita da tutte quelle parti elettroniche, elettriche, meccaniche, magnetiche, ottiche che ne consentono il funzionamento:
- Si suddivide in **sensori**, che prelevano i segnali di **input** dall'esterno, e in **attuatori**, che inviano segnali in **output** per creare dei movimenti, accendere led, ecc.
- La parte **software** si occupa della creazione di un programma che sovraintenda tutte le attività (**controllore**)

Lego WeDo 2.0

- Kit tradizionale programmabile da PC o tablet con linguaggio iconico



Lego Mindstorms EV3

- E' una linea di prodotti che combinano mattoncini programmabili con **motori elettrici, sensori**, pezzi di LEGO Technic (come ingranaggi, assi e parti pneumatiche) per costruire robot e altri sistemi automatici e/o interattivi.



Touch Sensor

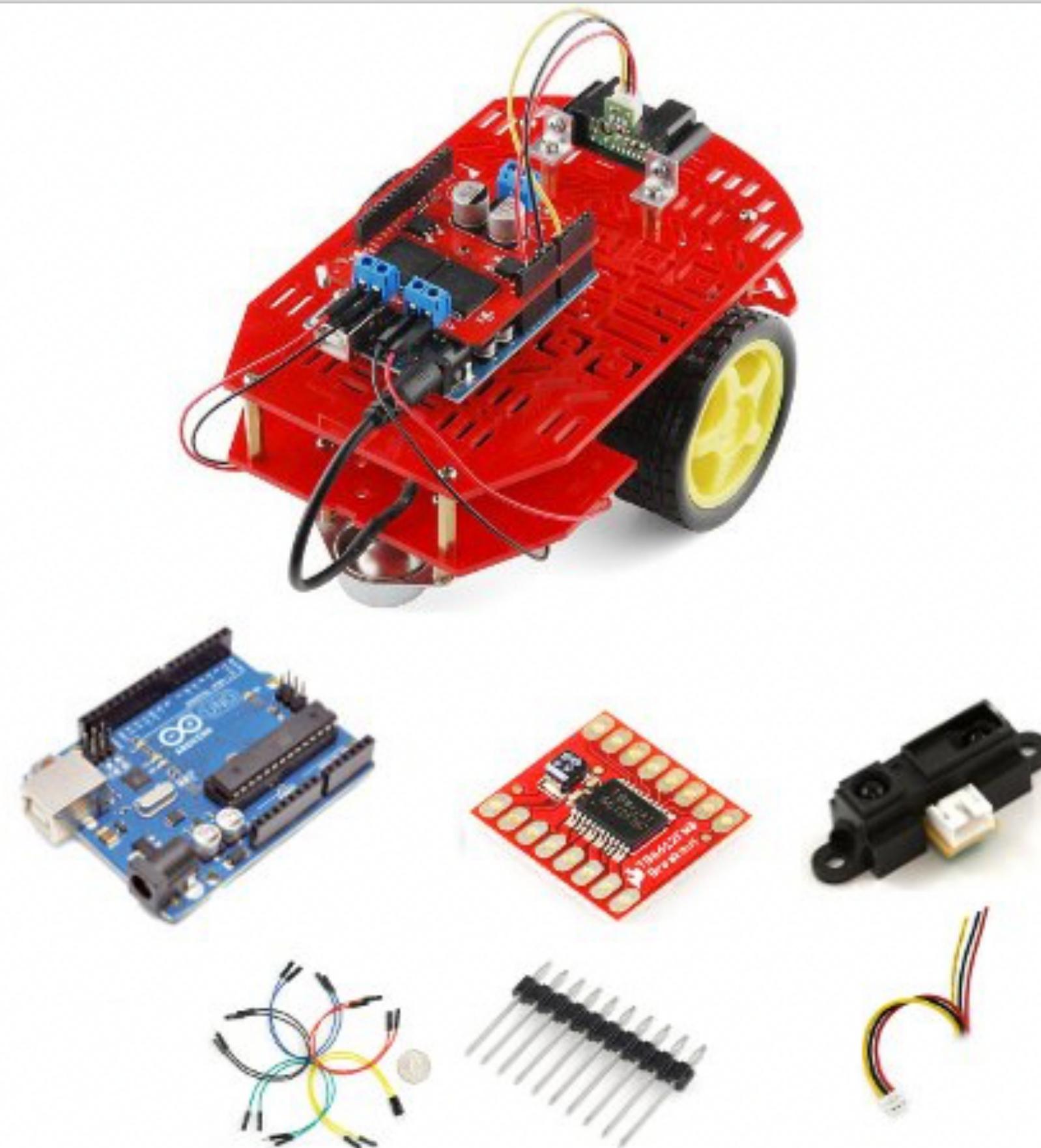
Ultrasonic Sensor

Light Sensor

Gyroscope Sensor

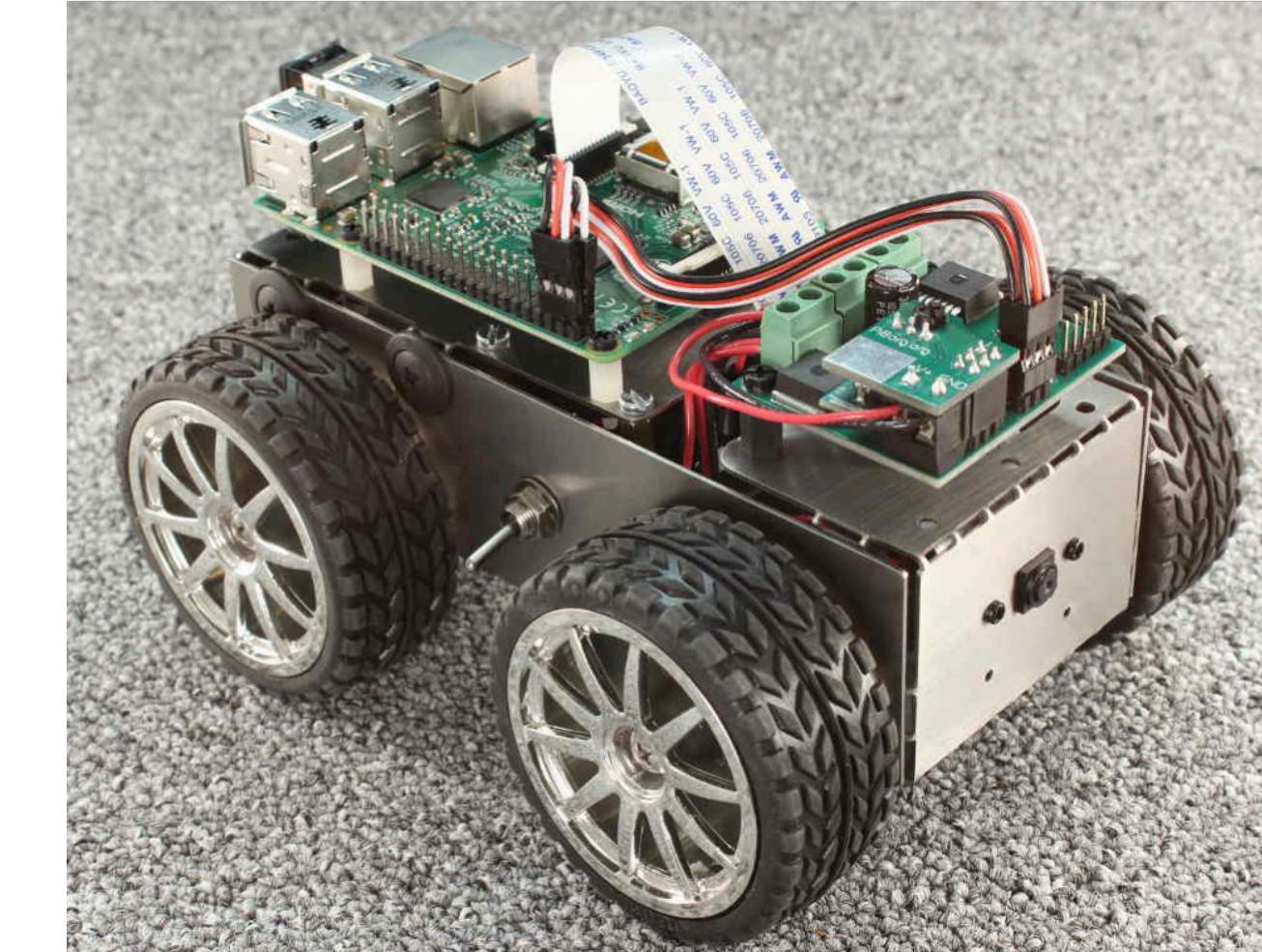
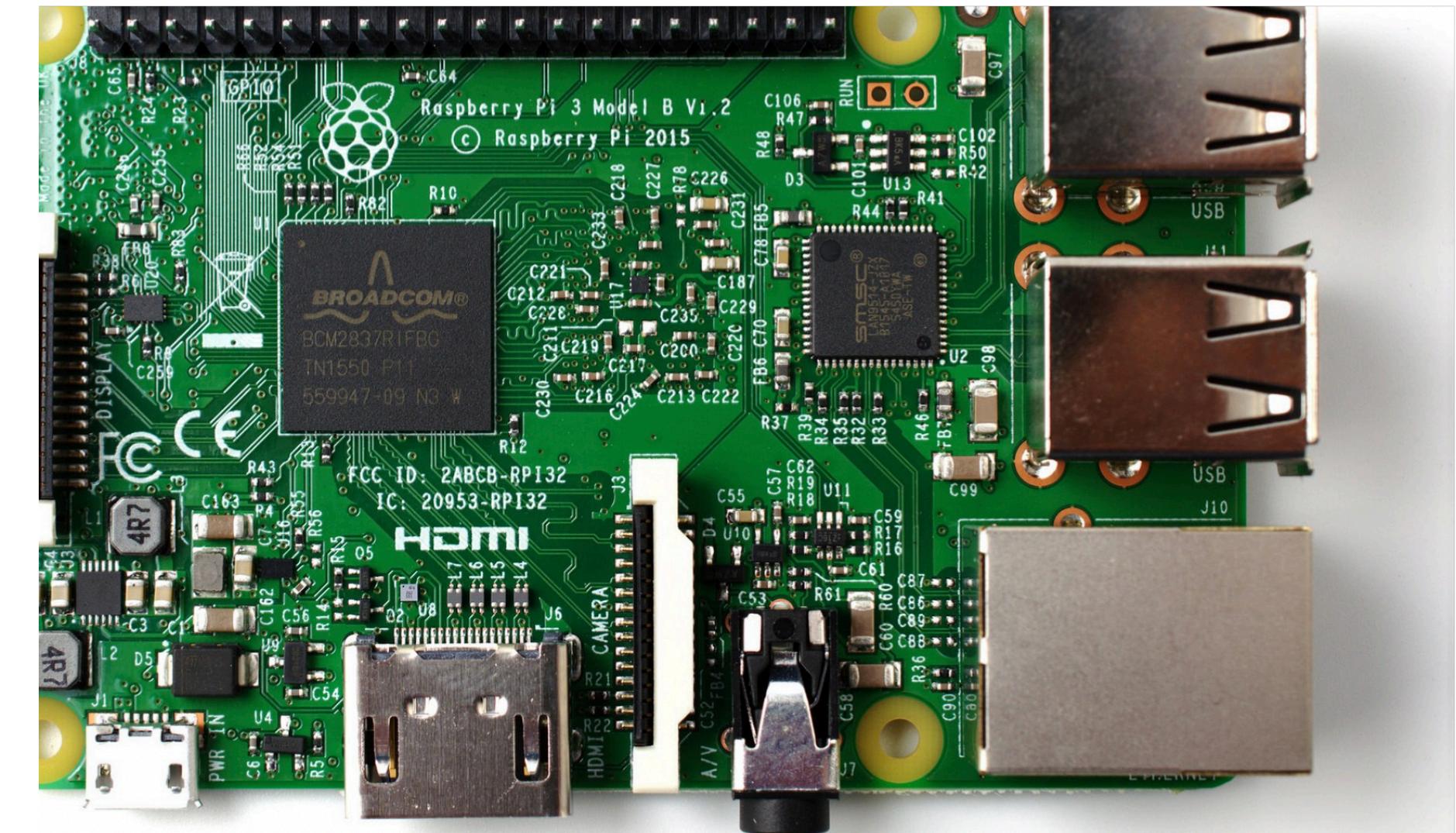
Arduino

- Arduino è un **microcontrollore**
- Si possono collegare sensori (pulsanti, potenziometri,...) e attuatori (LED, motori,...) e programmarne il funzionamento tramite l'**IDE proprietario**



Raspberry Pi

- Raspberry Pi è un **single-board computer** (un calcolatore implementato su una sola scheda elettronica)
- L'idea di base è la realizzazione di un **dispositivo economico**, concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole

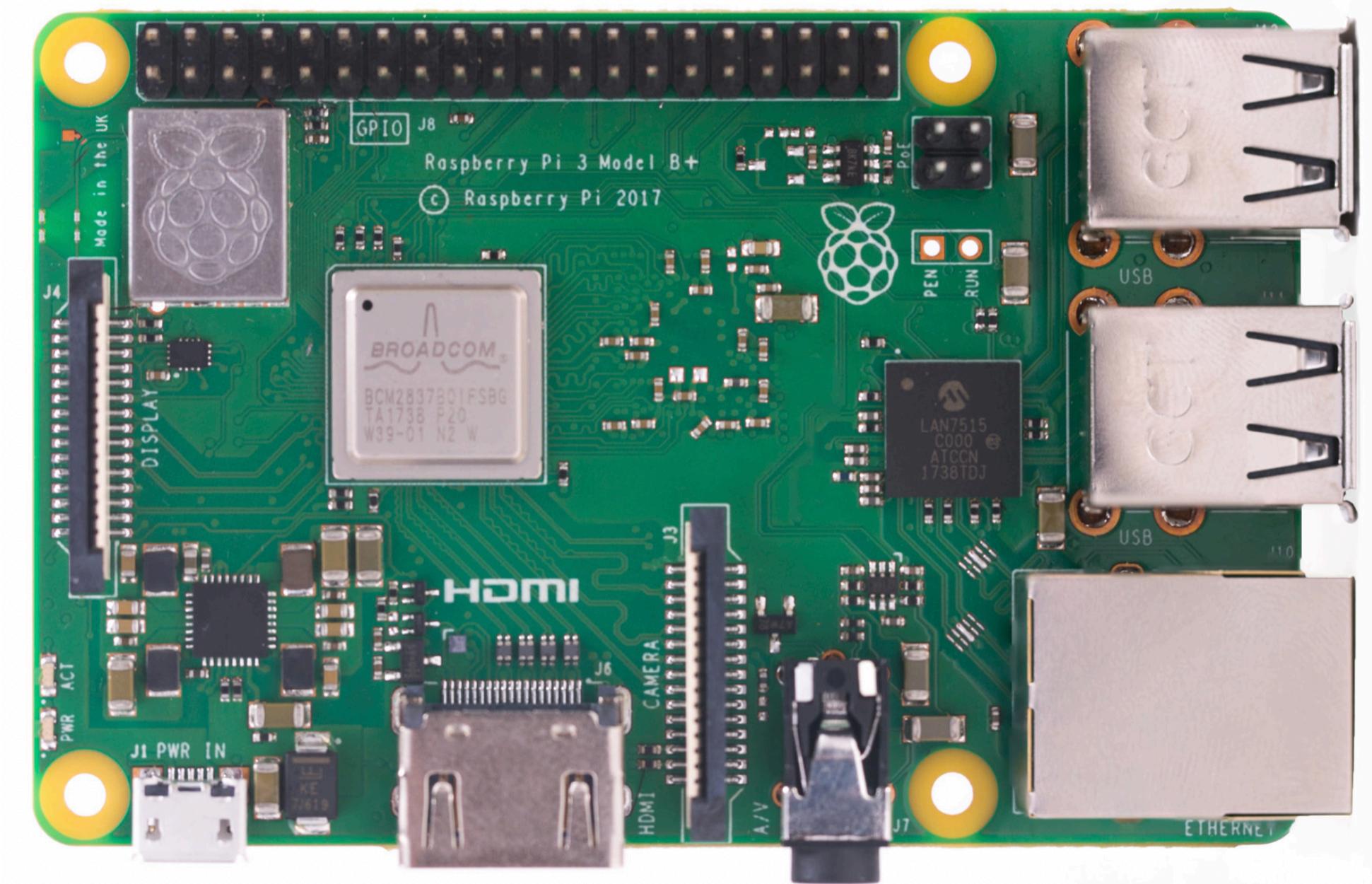


Raspberry PI 3B+



Raspberry PI 3B+

- L'ultima versione **Raspberry Pi 3 Model B+** ha un processore quad core a 64 bit con frequenza di clock 1.4GHz,
- **WLAN** dual-band a 2.4GHz e 5GHz
- Bluetooth 4.2/BLE

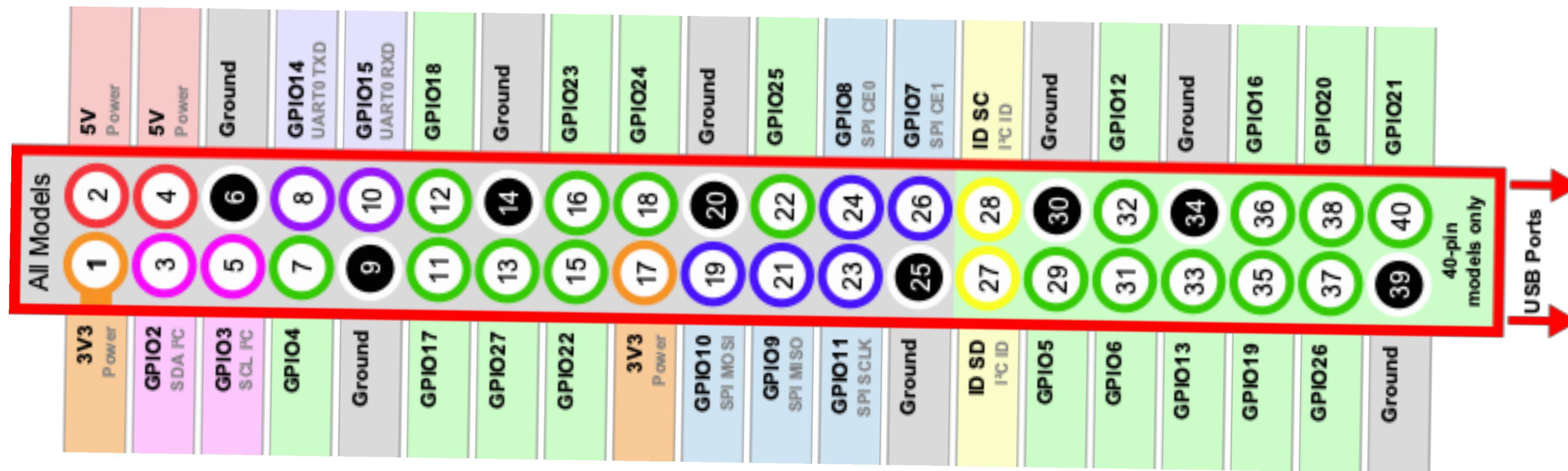


General Purpose Input/Output

- Il **General Purpose Input/Output** (anche noto come **GPIO**) è l'interfaccia di connessione con dispositivi e periferiche esterne.
- Queste possono agire come **input**, per leggere i segnali digitali dalle altre parti del circuito, o **output**, per controllare o segnalare agli altri dispositivi
- Raspberry PI 3B+ ha **40 pin GPIO**



General Purpose Input/Output



GPIO: caratteristiche

- Essendo in **diretta comunicazione** con il processore, e non avendo alcun tipo di protezione elettrica, l'interfaccia **GPIO** è anche una delle parti piú delicate del Raspberry Pi.
- L'interfaccia GPIO è un'interfaccia **digitale**: può inviare e ricevere soltanto segnali digitali, ovvero alternanze di **0 V** e **3.3 V**.
- Per utilizzare sensori e servomotori che abbiano interfaccia analogica (tipicamente quelli che si utilizzano in robotica) **è necessario utilizzare un ADC** (un convertitore da analogico a digitale) o i canali **PWM**
- i pin GPIO lavorano a 3.3 V e, a differenza di altri sistemi (Arduino) **non tollerano i 5 V**.

GPIO: librerie

- Il modo più semplice per programmare i pin GPIO è utilizzare **Python**, per fare ciò sono disponibili alcune librerie:
- **RPi.GPIO**: non supporta SPI, I2C, hardware PWM
- **GPIOZero**: fornisce una semplice interfaccia verso i dispositivi e sensori collegati e controllati tramite i pin GPIO
- **pigpio**: permette il controllo remoto dei pin GPIO da Raspberry, PC, Mac
- **Wiring.Pi**: permette programmazione in C/C++

GPIO: Python

- Python è un linguaggio di programmazione orientato agli oggetti.
- Python è un linguaggio di script pseudocompilato, similmente al Perl, ogni programma sorgente deve essere **pseudocompilato da un interprete**.
- L'interprete è un normale programma installato sulla macchina, e si occuperà di interpretare il codice sorgente e di eseguirlo. Quindi, diversamente dal C++, **non abbiamo un fase di compilazione - linking che trasforma il sorgente in eseguibile**
- Il principale vantaggio di questo sistema è la **portabilità**: lo stesso programma potrà girare su una piattaforma Linux, Mac o Windows purché vi sia installato l'interprete.

Esempi

Semaforo fisico e virtuale

- Edublocks è un **linguaggio grafico a blocchi**, simile a Scratch. Può essere utile per familiarizzare con le basi della programmazione ed i relativi costrutti fondamentali (sequenza, condizioni e cicli), concentrandosi sulle logiche di base, senza la preoccupazione di rispettare la sintassi di un linguaggio testuale.
- **Genera automaticamente sorgenti Python**

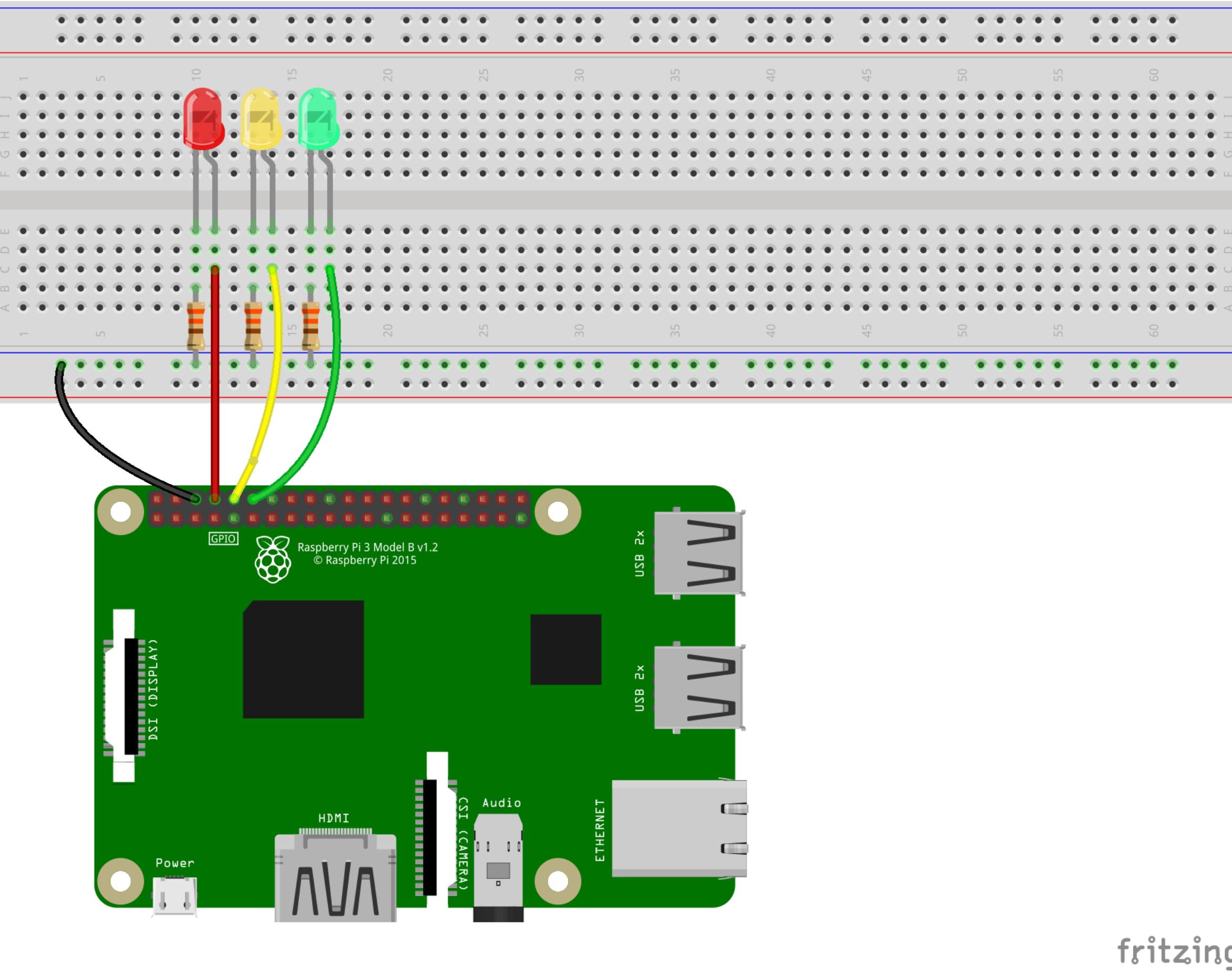


Semaforo fisico e virtuale

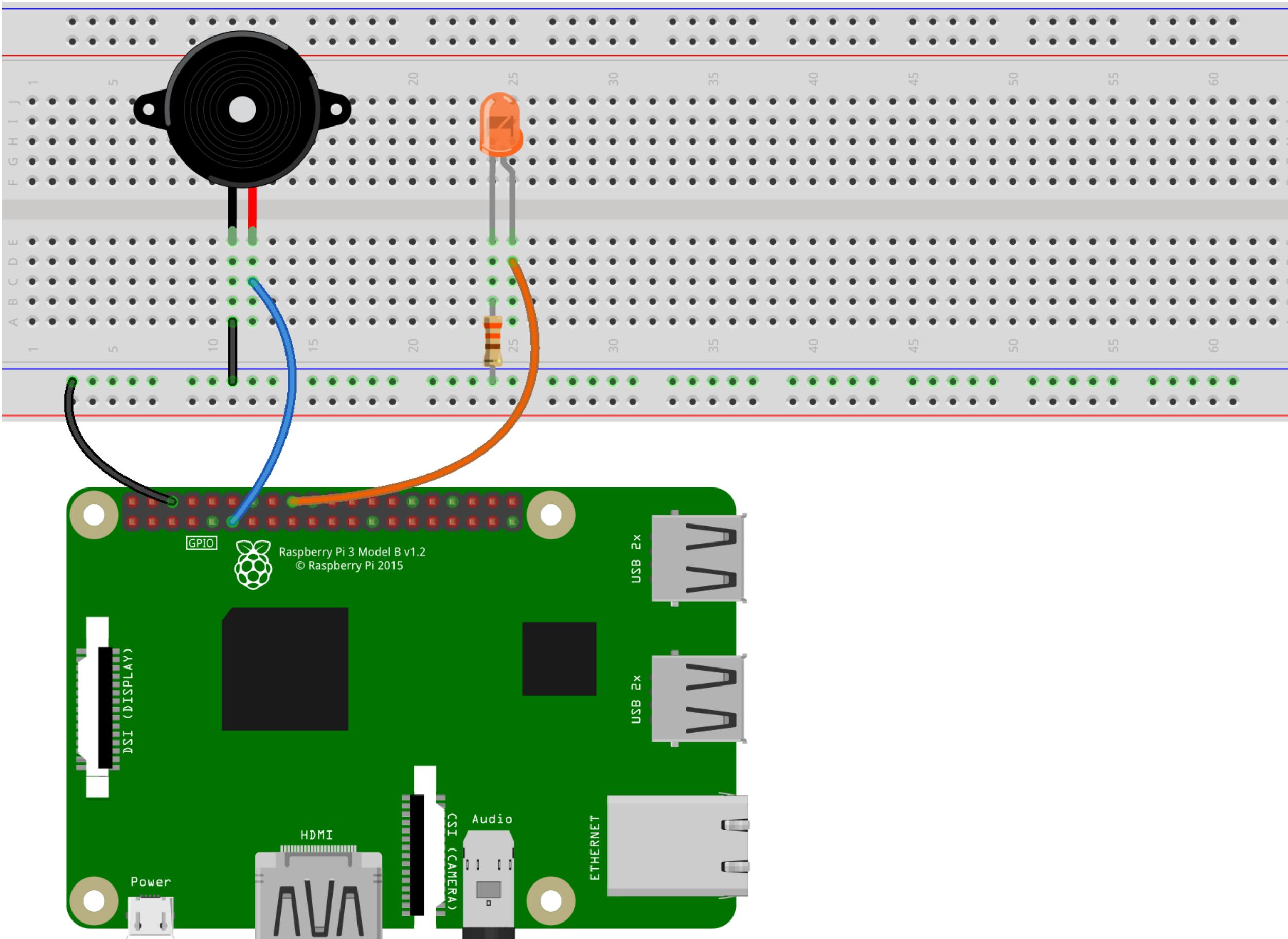
- **Minecraft Pi Edition** è una versione di Minecraft sviluppata per il Raspberry Pi. È basata sulla versione Alpha 0.6.1 della Pocket Edition
- Contiene una serie di funzioni rivisitate e il supporto a linguaggi di programmazione multipli
- La Pi Edition è intesa come uno strumento educativo ed è presente nella distribuzione Raspbian



Semaforo fisico e virtuale



Caccia al tesoro



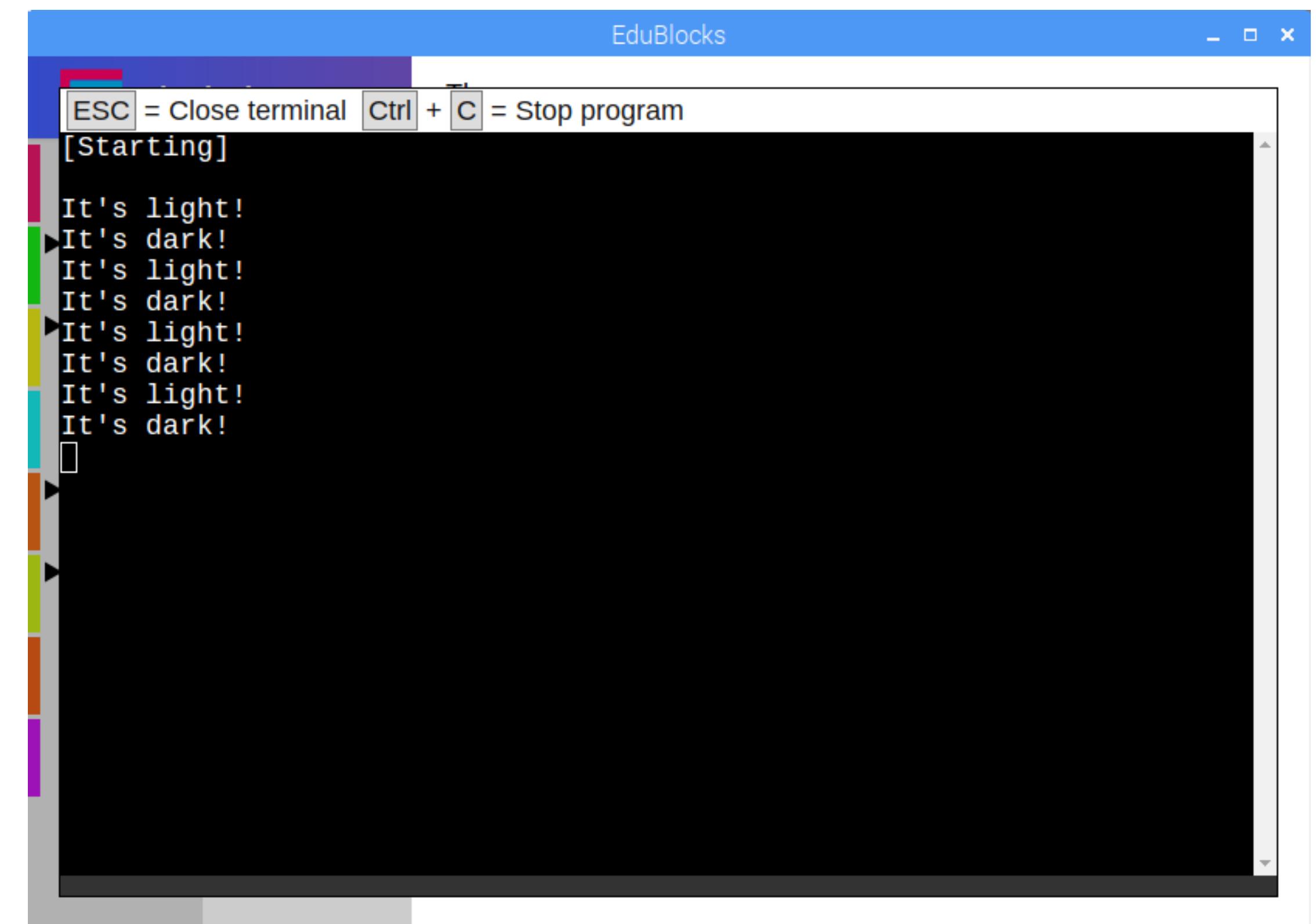
fritzing

Caccia al tesoro

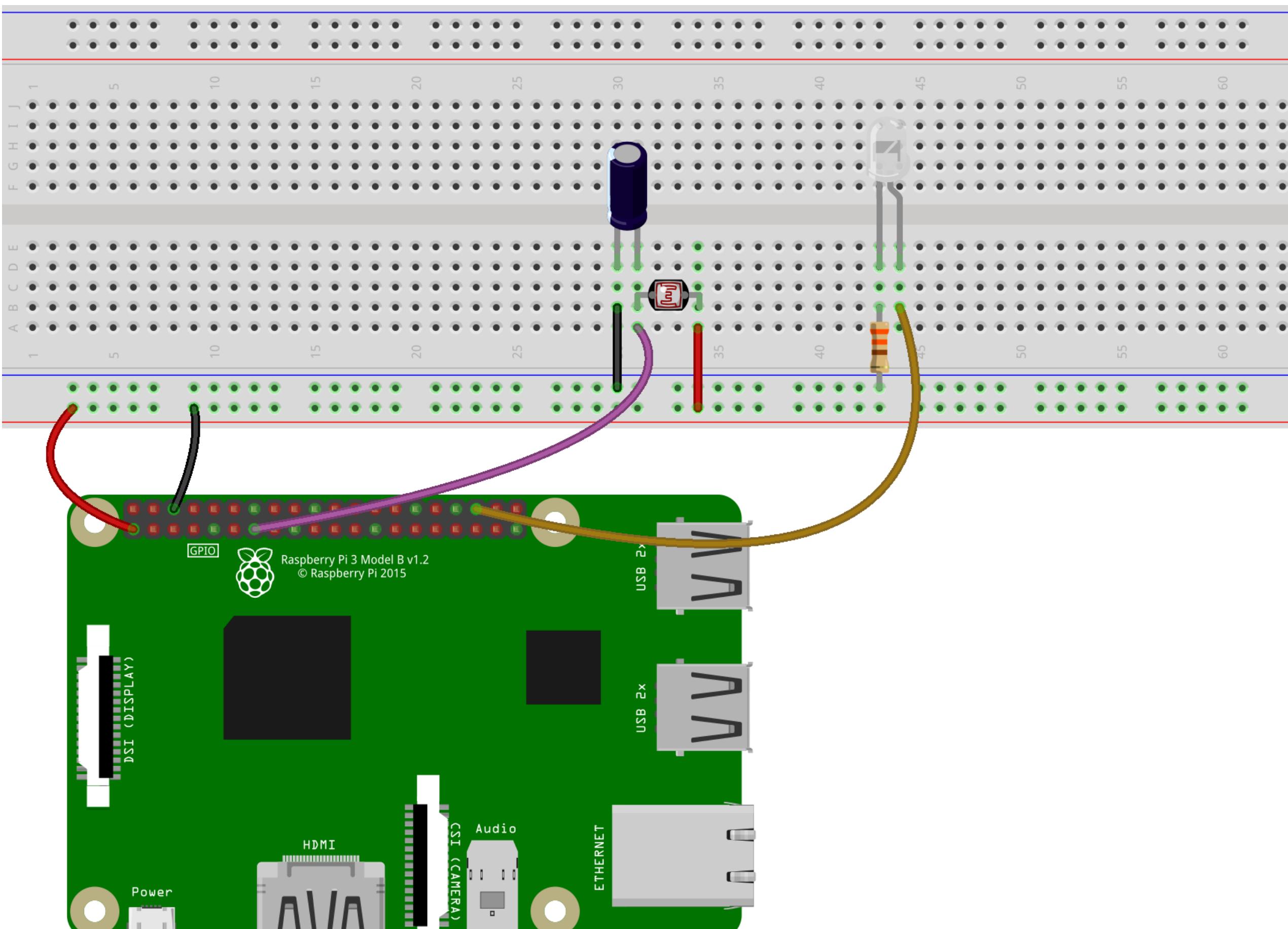


Sensore di luce - LDR 1

```
from gpiozero import *
sensor = LightSensor( 27 )
while True:
    sensor . wait_for_light( )
    print( "It's light!" )
    sensor . wait_for_dark( )
    print( "It's dark!" )
```



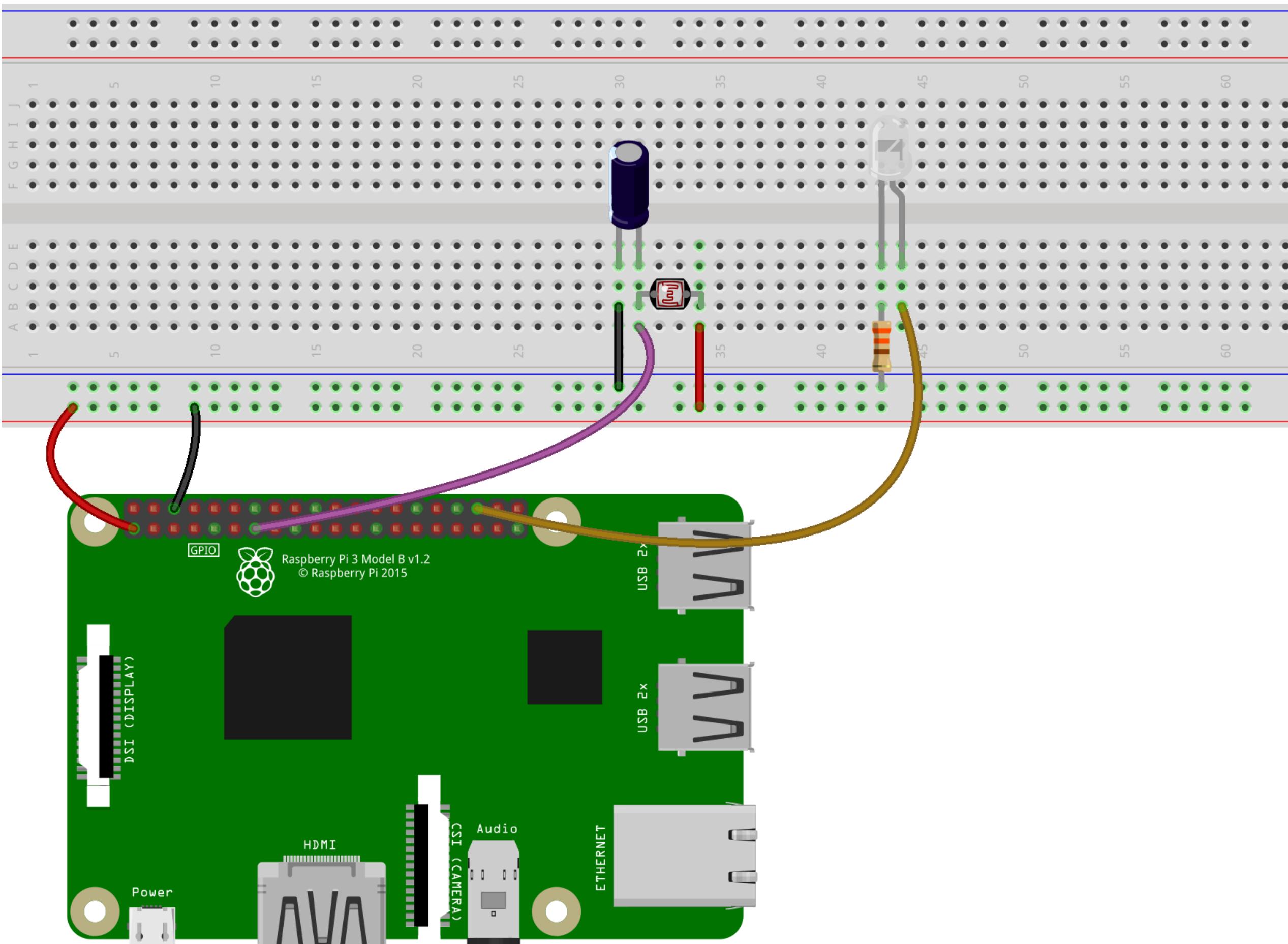
Sensore di luce - LDR 2



fritzing

```
from gpiozero import *
from signal import pause
sensor = LightSensor( 27 )
led = LED( 16 )
sensor . when_dark = led.on
sensor . when_light = led.off
pause()
```

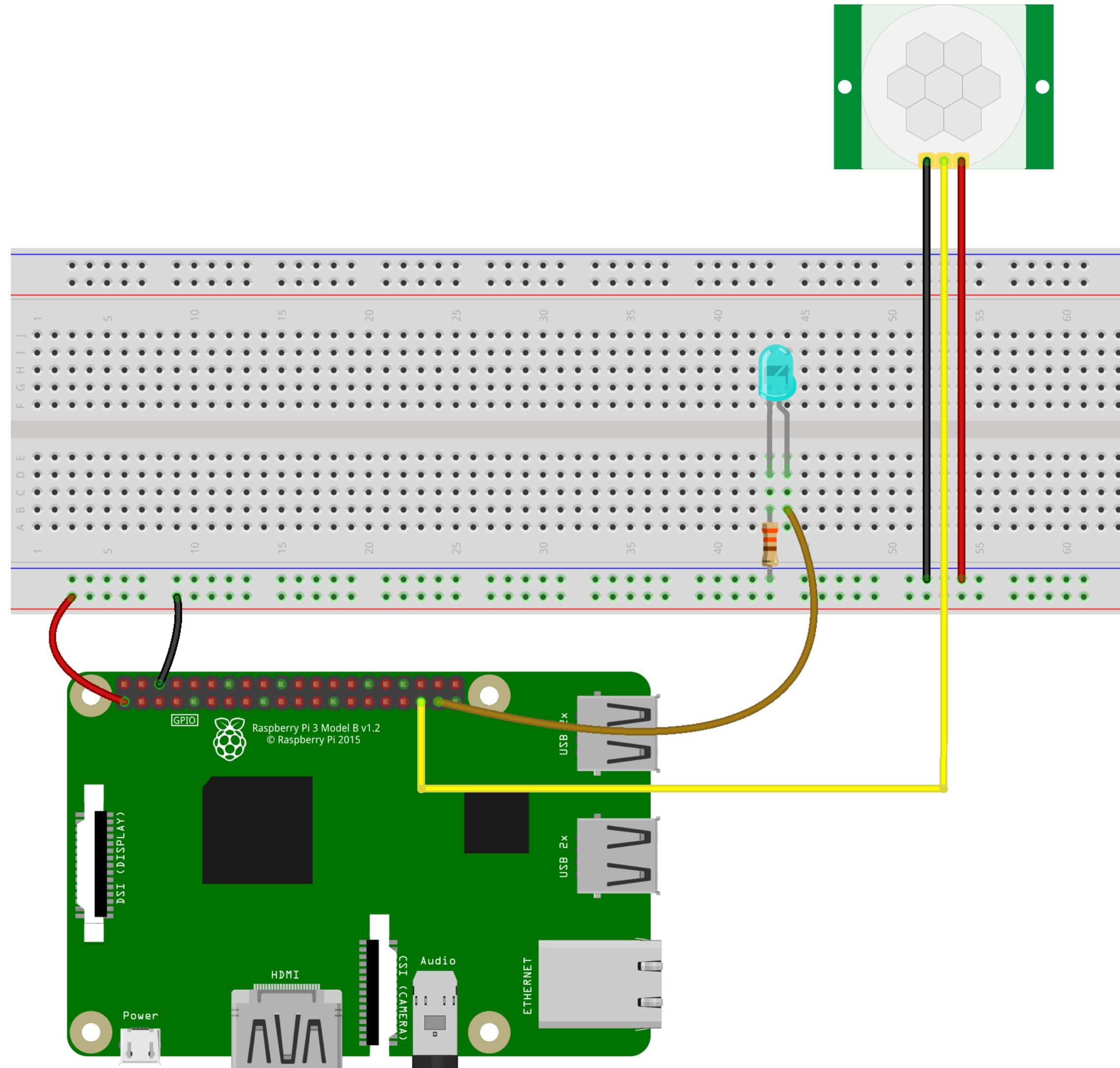
Sensore di luce - LDR 3



fritzing

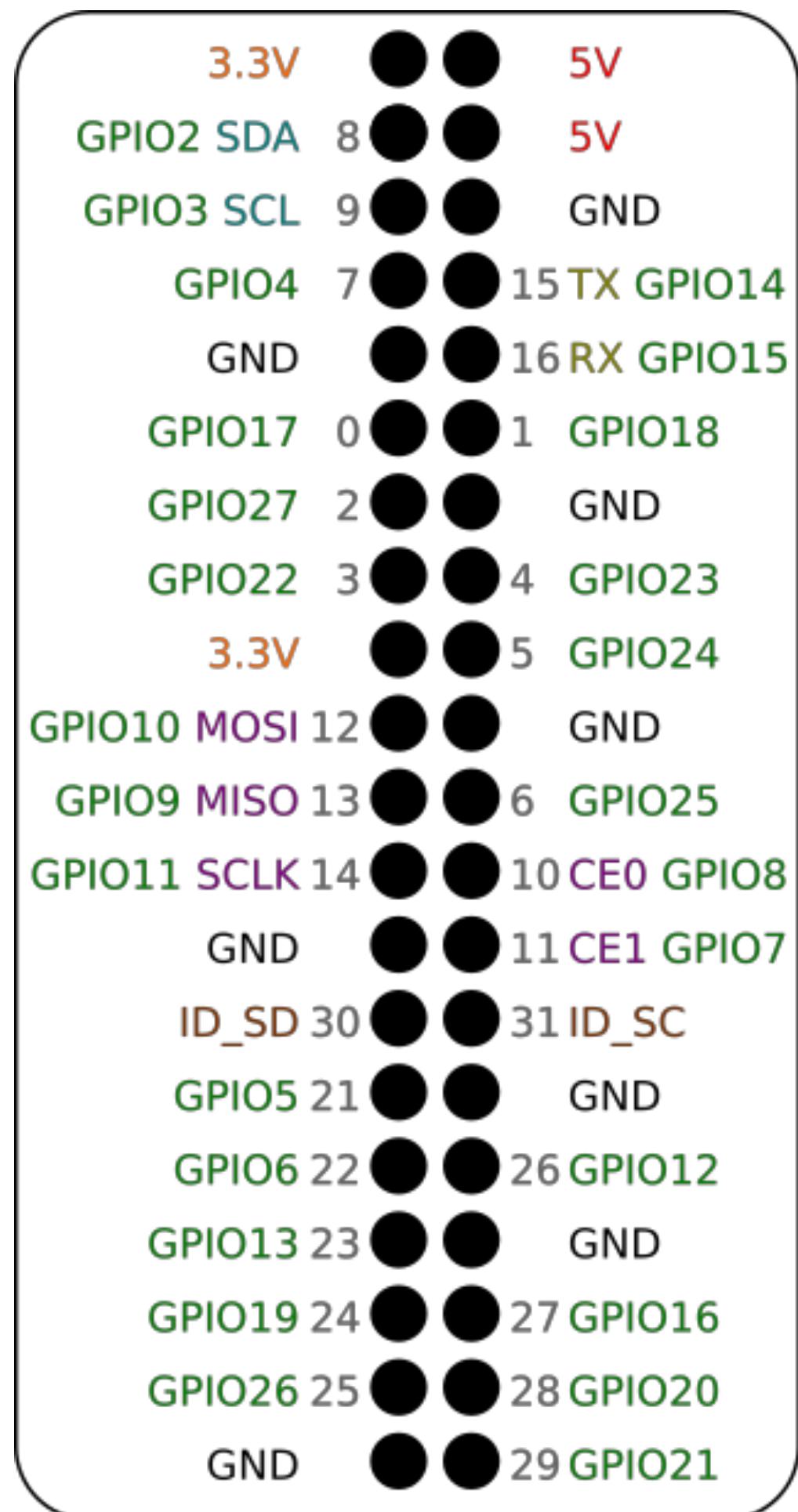
```
from gpiozero import *
from signal import pause
sensor = LightSensor( 27 )
led = PWMLED( 16 )
led . source = sensor
pause()
```

Pir -Led con Wiring PI



fritzing

Pir -Led con Wiring PI



Power (5 Volts)
Power (3 Volts)
Ground
WiringPi GPIO
BCM GPIO
I2C Interface
UART Interface
SPI Interface
ID EEPROM Interface

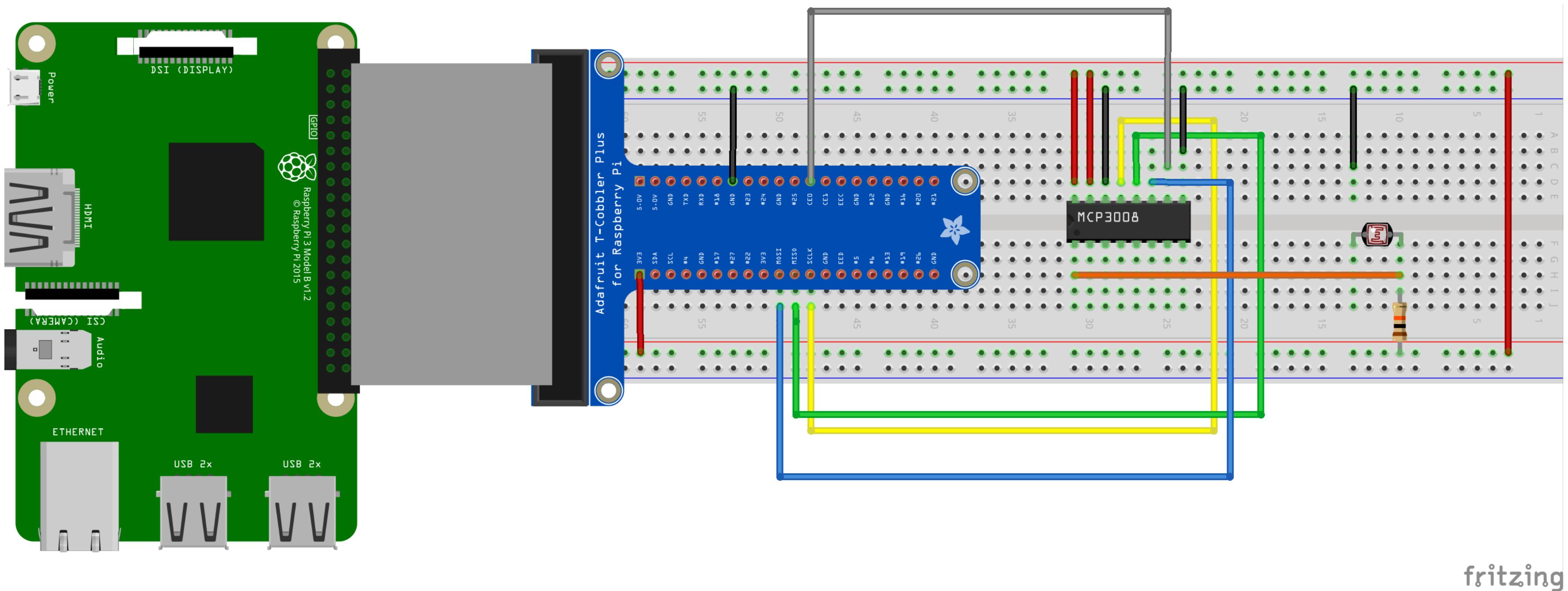
The code editor shows two files:

- blink.c**: A C program that includes the `wiringPi.h` header and defines a main function. It sets up the wiringPi library, configures pins 25 as an output and 24 as an input, and enters a loop. Inside the loop, it reads the state of pin 24 and writes to pin 25 based on the reading, with a 10ms delay between each state change.

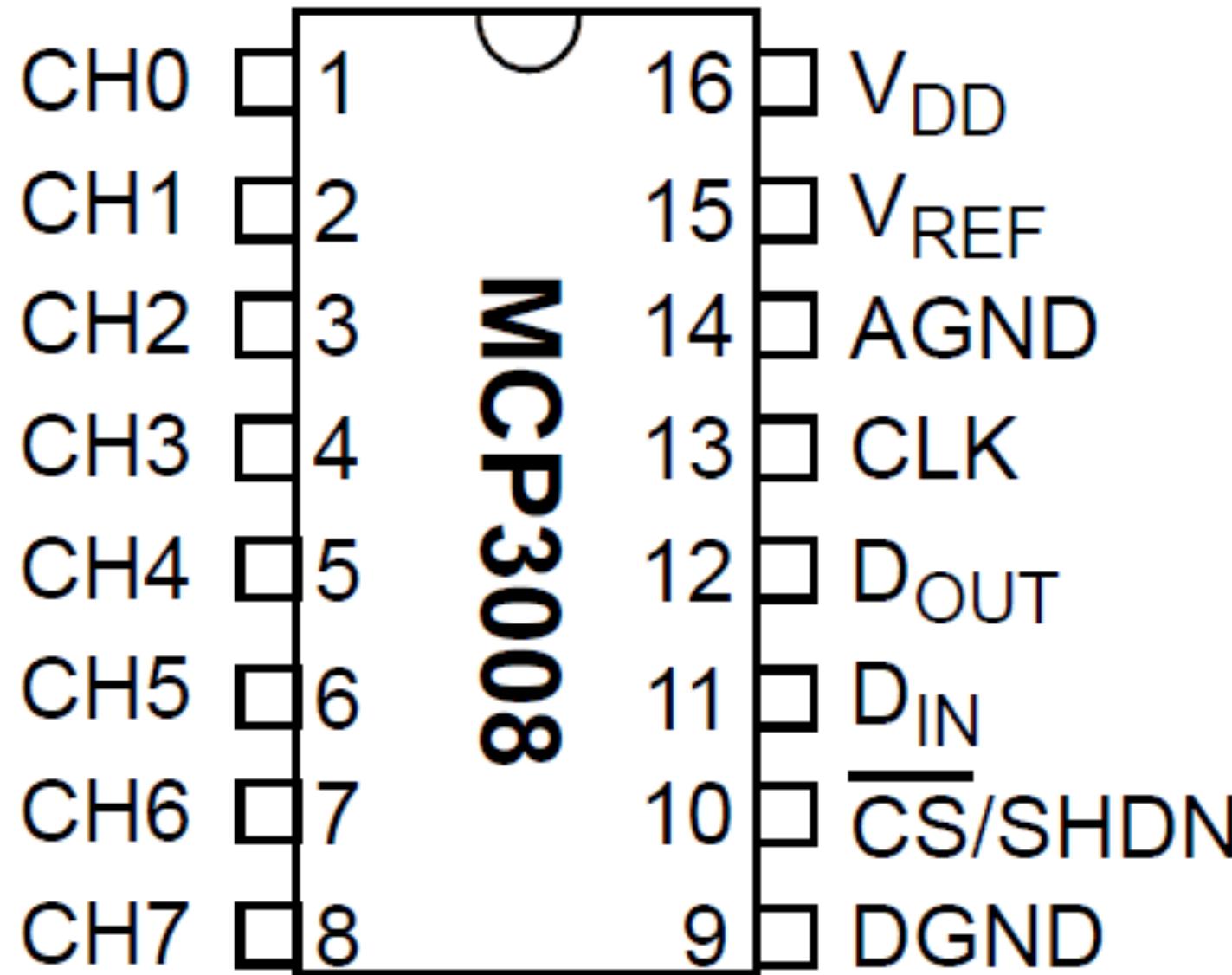
```
#include <wiringPi.h>
int main(void){
    wiringPiSetup();
    pinMode(25, OUTPUT);
    pinMode(24, INPUT);
    for(;;){
        if(digitalRead(24)==HIGH){
            digitalWrite(25, HIGH);
            delay(10);
        } else {
            digitalWrite(25, LOW);
            delay(10);
        }
    }
}
```
- adc-code.py**: A Python script that is currently empty.

```
gcc -Wall -o blink blink.c -lwiringPi
sudo ./blink
```

LDR - ADC con Wiring PI



LDR - ADC con Wiring PI



A screenshot of a code editor showing two files: 'adc-code.py' and 'blink.c'. The 'adc-code.py' file contains Python code for reading SPI data from the MCP3008. The 'blink.c' file contains C code for setting up the SPI interface and printing the LDR value.

```
adc-code.py
1 #!/usr/bin/python
2
3 import spidev
4 import time
5
6 #Define Variables
7 delay = 0.5
8 ldr_channel = 0
9
10 #Create SPI
11 spi = spidev.SpiDev()
12 spi.open(0, 0)
13 spi.max_speed_hz = 7629
14
15 def readadc(adcnum):
16     # read SPI data from the MCP3008, 8 channels in total
17     if adcnum > 7 or adcnum < 0:
18         return -1
19     r = spi.xfer2([1, 8 + adcnum << 4, 0])
20     data = ((r[1] & 3) << 8) + r[2]
21     return data
22
23
24 while True:
25     ldr_value = readadc(ldr_channel)
26     print "-----"
27     print("LDR Value: %d" % ldr_value)
28     time.sleep(delay)
29
```

```
blink.c
1 #include <sys/types.h>
2 #include <sys/stat.h>
3 #include <sys/conf.h>
4 #include <sys/conf.h>
5 #include <sys/conf.h>
6 #include <sys/conf.h>
7 #include <sys/conf.h>
8 #include <sys/conf.h>
9 #include <sys/conf.h>
10 #include <sys/conf.h>
11 #include <sys/conf.h>
12 #include <sys/conf.h>
13 #include <sys/conf.h>
14 #include <sys/conf.h>
15 #include <sys/conf.h>
16 #include <sys/conf.h>
17 #include <sys/conf.h>
18 #include <sys/conf.h>
19 #include <sys/conf.h>
20 #include <sys/conf.h>
21 #include <sys/conf.h>
22 #include <sys/conf.h>
23 #include <sys/conf.h>
24 #include <sys/conf.h>
25 #include <sys/conf.h>
26 #include <sys/conf.h>
27 #include <sys/conf.h>
28 #include <sys/conf.h>
29 #include <sys/conf.h>
```

```
gcc -Wall -o add-code adc-code.c -lwiringPi
sudo ./blink
```

Link

- <https://gpiozero.readthedocs.io/en/stable/index.html>
- <https://www.raspberrypi.org/>
- <https://www.raspberrypi.org/magpi/>
- <https://github.com/fpizzardo/physicalComputing>