PROGETTO SENSORE MOVIMENTO



<https://drive.google.com/file/d/1Li3gJvFduj1TbZUDjDK_i--JycXCbQoZ/view?usp=drive_link>

Sommario

[Descrizione (scopo) 3](#_Toc166745880)

[Apparecchiatura e componenti utilizzati 3](#_Toc166745881)

[Raspberry 3](#_Toc166745882)

[Basetta 3](#_Toc166745883)

[Buzzer 4](#_Toc166745884)

[Procedimento 5](#_Toc166745885)

[Circuito reale 5](#_Toc166745886)

[Codice Arduino IDE 5](#_Toc166745887)

[Codice Thonny 7](#_Toc166745888)

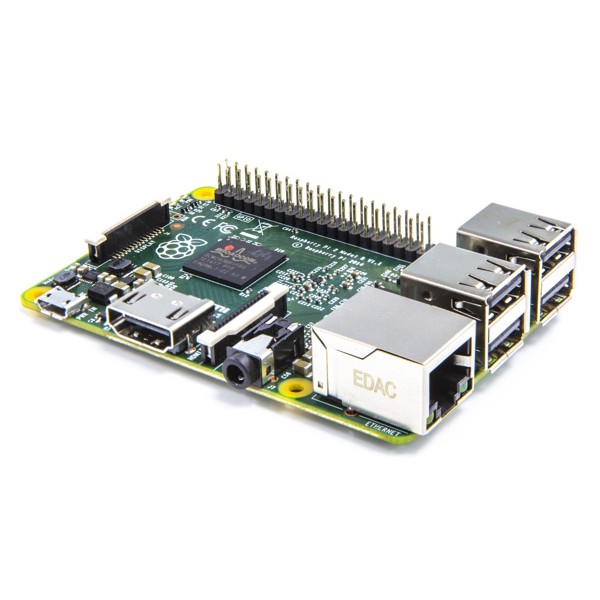
[Osservazioni, conclusioni, possibili sviluppi 8](#_Toc166745889)

# Descrizione (scopo)

Si realizzi un circuito che simuli il sensore delle auto. Più si avvicina un ostacolo e più il suono del buzzer sarà ripetitivo.

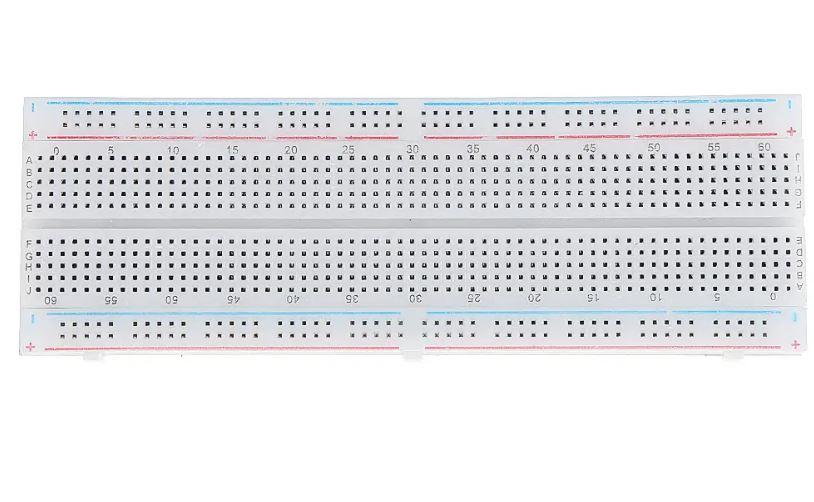
# Apparecchiatura e componenti utilizzati

## Raspberry

È il componente principale dell’esperienza. È progettato per essere economico, versatile e accessibile. Le schede Raspberry Pi sono dotate di CPU, RAM, porte di input/output e altre componenti necessarie per eseguire un sistema operativo e programmi. 

## Basetta

Una breadboard è uno strumento utilizzato per creare circuiti elettronici in modo temporaneo. È costituita da una base in plastica con fori allineati in modo regolare. I fori sono collegati tra loro in strisce metalliche sottostanti, che consentono di collegare i componenti elettronici. I fori superiori e inferiori sono collegati orizzontalmente mentre quelli centrali sono collegati verticalmente.



**Sensore ad ultrasuoni HC-SR04**

Il sensore ad ultrasuoni HC-SR04 è un dispositivo molto comune utilizzato per misurare la distanza da un oggetto. Funziona emettendo impulsi ad ultrasuoni e misurando il tempo impiegato per il loro ritorno dopo aver colpito un oggetto.

Per utilizzare l'HC-SR04 con Arduino o Raspberry Pi è necessario collegare quattro pin:

VCC: Alimentazione a +5V.

GND: Collegato a terra.

Trigger: Utilizzato per inviare il segnale di trigger al sensore per iniziare la misurazione della distanza.

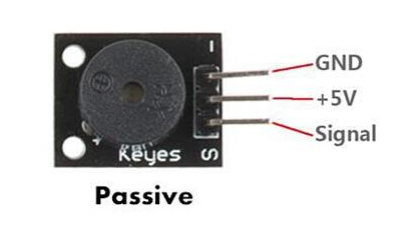
Echo: Restituisce un segnale proporzionale al tempo impiegato dall'impulso ad ultrasuoni per tornare.



## 

## Buzzer

Un buzzer per sistemi su Arduino o Raspberry Pi è un dispositivo elettronico che emette suoni quando è alimentato. È composto da un piccolo altoparlante e da un circuito che lo fa vibrare a una certa frequenza per produrre suoni. Può essere utilizzato per avvisi sonori, segnalazioni o per scopi artistici e ludici.



# Procedimento

* Procuro materiali necessari:
  + Scheda Raspberry
  + Breadboard
  + Cavi
  + Sensore ad ultrasuoni HC-SR04
* Creo il circuito sulla breadboard e lo collego con la scheda Arduino e poi Raspberry
  + Collego i vari il trigger e l’echo del sensore ai rispettivi pin su Arduino. Faccio i soliti collegamenti con 5V e GND.
* Creo il codice nel software Arduino IDE.
  + Collego i pin digitali di Arduino con quelli di RaspberryPi
* Creo il codice nel software Thonny.

# Circuito reale

# Codice Arduino IDE

// Definizione dei pin del sensore ad ultrasuoni

const int trigPin = A0;

const int echoPin = A1;

// Definizione del pin del buzzer

const int buzzerPin = A2;

const bool distanze = 5;

const bool distanze1 = 6;

void setup() {

  // Inizializzazione del sensore ad ultrasuoni

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

  // Inizializzazione del buzzer

  pinMode(buzzerPin, OUTPUT);

  // Inizializzazione della comunicazione seriale

  Serial.begin(9600);

}

void loop() {

  // Genera un impulso sull'uscita di trigger del sensore ad ultrasuoni

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  // Legge il tempo di ritorno dell'eco dal sensore ad ultrasuoni

  long duration = pulseIn(echoPin, HIGH);

  // Calcola la distanza in base al tempo di ritorno dell'eco

  // La velocità del suono è 343 m/s, quindi la distanza è la metà del tempo di ritorno moltiplicato per 0.0343

  float distance = (duration / 2) \* 0.0343;

  // Invia la distanza tramite comunicazione seriale al Raspberry Pi

  Serial.println(distance);

  // Calcola la frequenza proporzionale alla distanza

  int delayTime = map(distance, 0, 40, 10, 1000); // Mappa la distanza sul tempo di ritardo tra i suoni

  // Suona il buzzer con il tempo di ritardo calcolato

  if (distance < 40 && distance > 30) { // Modifica il valore della soglia a seconda della tua necessità

    digitalWrite(buzzerPin, HIGH);

    delay(delayTime);

    digitalWrite(buzzerPin, LOW);

    delay(50); // Aggiungi un piccolo ritardo tra un suono e l'altro

    digitalWrite(distanze, HIGH);

    digitalWrite(distanze1, LOW);

  } else if(distance < 30 && distance > 15){

    digitalWrite(buzzerPin, HIGH);

    delay(delayTime);

    digitalWrite(buzzerPin, LOW);

    delay(50); // Aggiungi un piccolo ritardo tra un suono e l'altro

    digitalWrite(distanze, LOW);

    digitalWrite(distanze1, HIGH);

  } else if(distance < 15){

    digitalWrite(buzzerPin, HIGH);

    delay(delayTime);

    digitalWrite(buzzerPin, LOW);

    delay(50); // Aggiungi un piccolo ritardo tra un suono e l'altro

    digitalWrite(distanze, HIGH);

    digitalWrite(distanze1, HIGH);

  } else{

    digitalWrite(buzzerPin, LOW);

    digitalWrite(distanze, LOW);

    digitalWrite(distanze1, LOW);

  }

  delay(100); // Aggiungi un piccolo ritardo per evitare la saturazione della comunicazione seriale

}

# Codice Thonny

import RPi.GPIO as GPIO

import time

# Imposta la modalità dei pin GPIO

GPIO.setmode(GPIO.BCM)

# Imposta il pin GPIO per ricevere i dati da Arduino

DISTANZE = 21

DISTANZE1 = 20

GPIO.setup(DISTANZE, GPIO.IN)

GPIO.setup(DISTANZE1, GPIO.IN)

GPIO.setup(16, GPIO.OUT)

while GPIO.input(DISTANZE)== GPIO.HIGH and GPIO.input(DISTANZE1) == GPIO.LOW:

GPIO.output(16,GPIO.HIGH)

time.sleep(2)

GPIO.output(16,GPIO.LOW)

time.sleep(2)

while GPIO.input(DISTANZE)== GPIO.LOW and GPIO.input(DISTANZE1) == GPIO.HIGH:

GPIO.output(16,GPIO.HIGH)

time.sleep(1)

GPIO.output(16,GPIO.LOW)

time.sleep(1)

while GPIO.input(DISTANZE)== GPIO.HIGH and GPIO.input(DISTANZE1) == GPIO.HIGH:

GPIO.output(16,GPIO.HIGH)

time.sleep(0.2)

GPIO.output(16,GPIO.LOW)

time.sleep(0.2)

# Osservazioni, conclusioni, possibili sviluppi

Durante lo svolgimento di questa esperienza abbiamo riscontrato alcune problematiche con il buzzer perché non funzionava e abbiamo dovuto cambiarlo. Abbiamo avuto una piccola difficoltà iniziale con il codice del programma su Thonny. È stato complicato eseguire il passaggio da Arduino a Raspberry con i pin. Per il resto è andato tutto a buon fine e abbiamo completato l’esperienza.