Erasmus+

2Play2Learn

Reviving hands-on educational play for learning skills of tomorrow

PROJECT N° 2019-1-UK01-KA201-061466

MODULO 3

Physical Computing – DIY Kit Elettronico

SVILUPPATO DA SCHOLE & EMPHASYS

CIVIC idec ΕΚΠΑΙΔΕΥΤΗΡΙΑ ΠΛΑΤΩΝ dlearn scholé Emphasys CENTRE CCS Digital Education

DIY Esercizio 1 – Il semaforo

Creare e programmare una sequenza luminosa usando 3 LED.

Co-funded by the
Erasmus+ Programme
of the European Union

DIY Esercizio 1 – Il semaforo

Soluzione– Circuito:

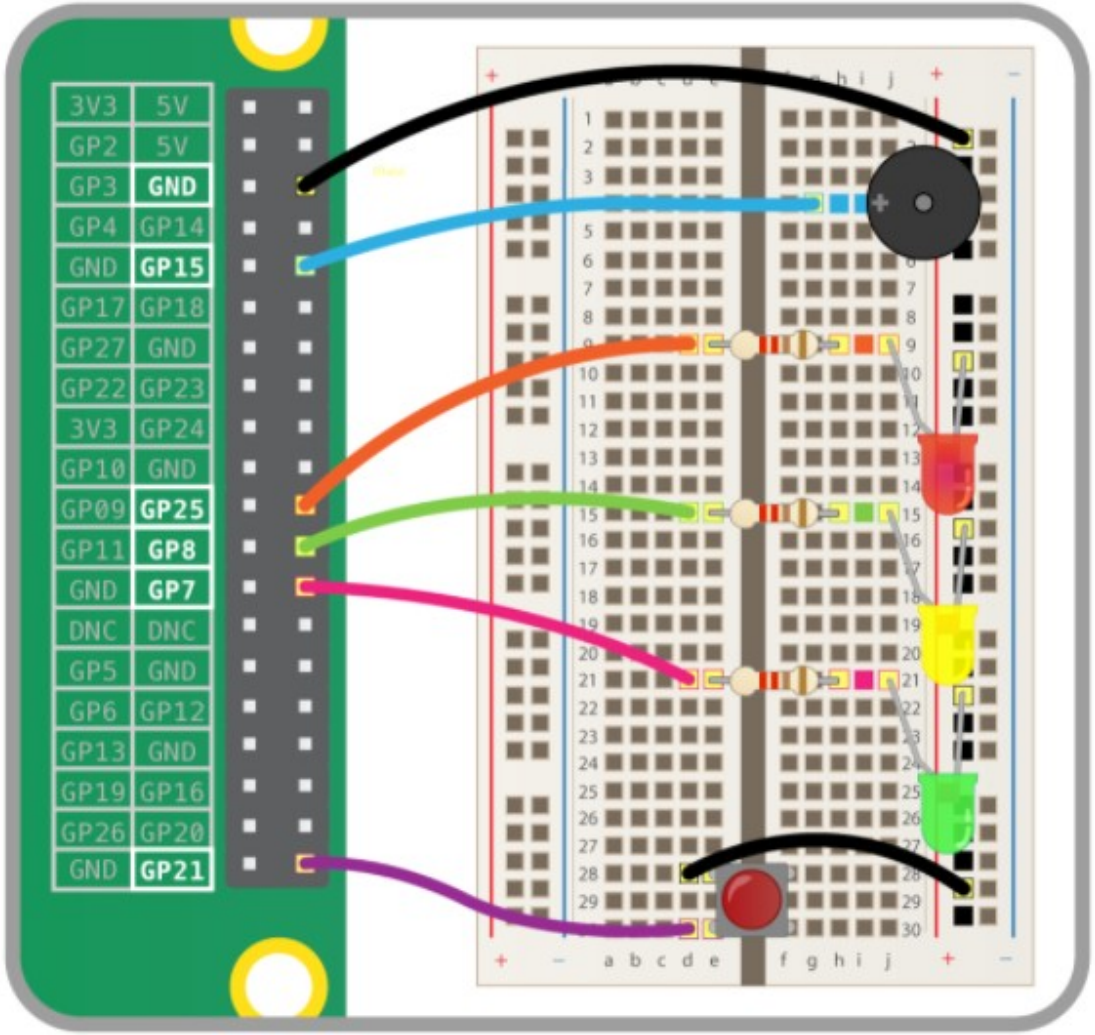


Image Source: https://projects.raspberrypi.org/en/projects/physical-computing/9

Co-funded by the Erasmus+ Programme of the European Union

Soluzione – Codice:

È necessario installare Mu. Aprire una finestra di terminale e digitare il seguente comando:

```
Sudo apt install mu-editor
```

Aprire Mu dal menu principale.

Inserire il seguente codice:

```
from gpiozero import LED
red = LED(22)
amber = LED(27)
green = LED (17)

red.blink(1, 1)
amber.blink(2, 2)
green.blink (3, 3)
```

se si esegue il codice si dovrebbero vedere tre luci lampeggiare a differenti velocità

Co-funded by the Erasmus+ Programme of the European Union

Soluzione – Codice:

La funzione **ON** consente di accendere la luce. Si può usare **SLEEP** per creare una pausa tra un comando e l'altro:

```
from gpiozero import LED
red = LED(22)
amber = LED(27)
green = LED (17)


red.on()
sleep(1)                    # accendere le luci in sequenza
amber.on()
sleep(1)
green.on()
sleep(1)
```

Co-funded by the
Erasmus+ Programme
of the European Union

Soluzione – Codice:

La funzione **ON** consente di accendere la luce. Si può usare **sleep** per impostare una pausa tra un comando e l'altro:

```
from gpiozero import LED
red = LED(22)
amber = LED(27)
green = LED (17)


red.on()
sleep(1)
amber.on()                    # accendere le luci in sequenza.
sleep(1)
green.on ()
sleep(1)
```

Co-funded by the Erasmus+ Programme of the European Union

DIY Esercizio 1 – Il semaforo

Soluzione – Codice:

Per avere un semaforo funzionante:

```
red = LED(22)
amber = LED(27)
green = LED (17)

While True:
red.on()
sleep(1)
amber.on()
sleep(1)
green.on ()
sleep(1)
red.off()
sleep(1)
amber.off()
sleep(1)
green.off ()
```

accendere le luci in sequenza in loop

Co-funded by the
Erasmus+ Programme
of the European Union

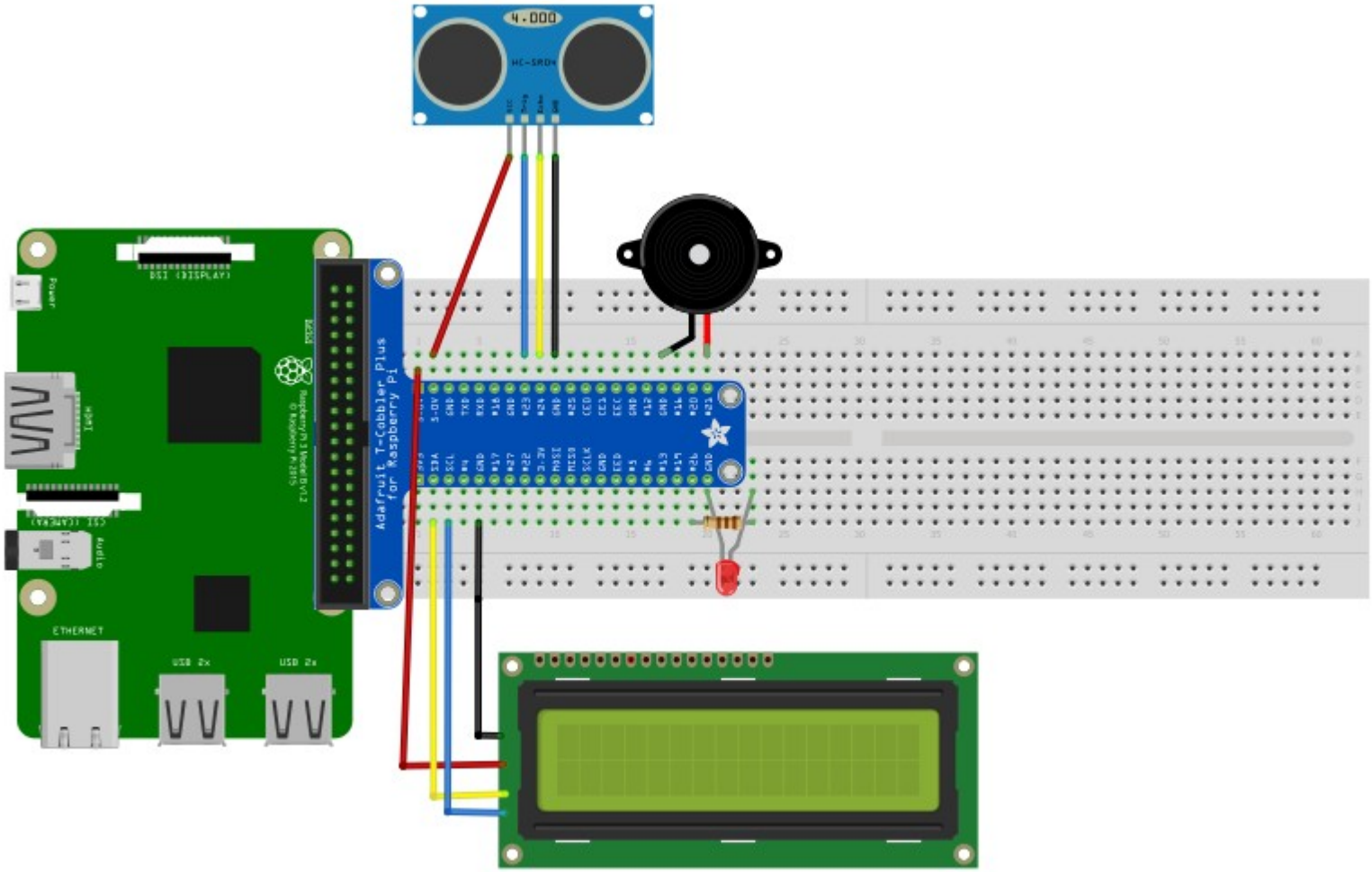
DIY Esercizio 2 – Sistema sensore di parcheggio

Progettare e creare un sistema di sensori di parcheggio che può essere utilizzato in un'automobile. Questo sistema:

- Avrà un sensore di distanza che misurerà la distanza da un oggetto
- Mostrerà la distanza attuale sullo schermo LCD
- Se la distanza è meno di 20 cm avrà:
  - Suono del Buzzer
  - Luce LED lampeggiante
  - Invierà un messaggio di avviso visibile sullo schermo LED

Co-funded by the Erasmus+ Programme of the European Union

Soluzione– Circuito:



Co-funded by the
Erasmus+ Programme
of the European Union

Soluzione– Codice:

```
from PCF8574 import PCF8574_GPIO              #Importare il setup del pin dal programma PCF8574.py
from Adafruit_LCD1602 import Adafruit_CharLCD   #Importare il metodo dal programma Adafruit_LCD1602.py

from time import sleep, strftime
from datetime import datetime

import RPi.GPIO as GPIO #Give access to the GPIO pins
import time             # Dare accesso al modulo tempo

def setup():
    GPIO.setmode(GPIO.BOARD)       # Usare PHYSICAL GPIO Numbering
    GPIO.setup(trigPin, GPIO.OUT)   # Configurare trigPin in modalità OUTPUT
    GPIO.setup(echoPin, GPIO.IN)    # Configurare echoPin in modalità INPUT
    GPIO.setup(buzPin,GPIO.OUT)     # Configurare buzPin in modalità OUTPUT

    GPIO.setup(ledPin, GPIO.OUT)   # Configurare il ledPin in modalità OUTPUT
    GPIO.output(ledPin, GPIO.LOW)  # Portare l'uscita ledPin al livello LOW
```

Co-funded by the Erasmus+ Programme of the European Union

```
# Per il buzzer
buzPin=40       # Definisci il pin del buzzer
# For LED
ledPin = 37     # Definisci il pin del  Led

trigPin = 16    # La posizione del pin trigger
echoPin = 18    # La posizione del pin echo
maxDistance = 400          # Questa è la distanza massima misurata in centimetri che il sensore può misurare:
timeOut = maxDistance*58.82     # Calcolare il timeout in base alla misurazione massima di distanza /tempo in cui il sensore non sarà più in attesa
                        # per un segnale se la distanza da un oggetto è maggiore del massimo

def pulseIn(pin,level,timeOut): # Ottenere il tempo di impulso di un pin durante il timeOut
    t0 = time.time()
    while(GPIO.input(pin) != level):
        if((time.time() - t0) > timeOut*0.000001):
            return 0;                       # Ritorna 0 se il tempo è superiore al tempo del timeout
    t0 = time.time()
    while(GPIO.input(pin) == level):
        if((time.time() - t0) > timeOut*0.000001):
            return 0;                       # Ritorna 0 se il tempo è di più del timeout
    pulseTime = (time.time() - t0)*1000000          # Calcolare il tempo di impulso
    return pulseTime
```

Co-funded by the
Erasmus+ Programme
of the European Union

```
def getResponce():                    # Ottenere i risultati della misurazione del modulo ultrasonico in cm
    GPIO.output(trigPin,GPIO.HIGH)    # Configurare trigPin in output 10us al livello HIGH
    time.sleep(0.00001)               # Aspetta per 10us
    GPIO.output(trigPin,GPIO.LOW)     # Configurare trigPin in output al livello LOW
    pingTime = pulseIn(echoPin,GPIO.HIGH,timeOut)   # Memorizzare il tempo di impulso dell'echoPin
    distance = pingTime * 340.0 / 2.0 / 10000.0   # Calcolare la distanza con il tempo del suono 340m/s
    return distance

def ledBlink():
        GPIO.output(ledPin, GPIO.HIGH)    # Configurare il ledPin in output HIGH e accendere il led
        print ('led turned on >>>')       # Visualizzare le informazioni sullo schermo
        time.sleep(0.2)                   # Aspettare 0.2 secondi
        GPIO.output(ledPin, GPIO.LOW)     # Configurare il ledPin in output LOW e spegnere la luce
        print ('led turned off <<<')      # Visualizzare le informazioni sullo schermo
        time.sleep(0.2)                   # Aspettare 0.2 secondi
```

Co-funded by the Erasmus+ Programme of the European Union

```
def main_loop():
    mcp.output(3,1)     # Accendere la retroilluminazione LCD
    lcd.begin(16,2)     # Configurare un numero di linee e colonne  LCD
    while(True):
        lcd.setCursor(0,0)  # Confiigurare la posizione iniziale del cursore
        sleep(1)            # Aspettare 1 secondi
        lcd.clear()         # Cancellare il contenuto dello schermo LCD
        sleep(1)            # Aspettare 1 secondo

        distance = getResponce()                    # Ottenere la distanza da un oggetto
        print ("The distance is : %.2f cm"%(distance))  #Visualizzare la distanza sullo schermo del computer
        distInString = str(distance)                # Trasformare la distanza in una stringa variabile  da usare con lo schermo LED
        lcd.message('Distance:'+(distInString)+'\n')    # Visualizzare la distanza sullo schermo LED
        time.sleep(0.2)

        if distance<20:                 #Controllare se la distanza è meno di 20cm
            GPIO.output(buzPin,GPIO.HIGH)   #Accendere il buzzer
            ledBlink()                  #Chiamare il metodo ledBlink()
            lcd.message('DANGER!!!')        # Visualizzare il messaggio sullo schermo LCD

        else:
            GPIO.output(buzPin,GPIO.LOW)    #Spegnere il buzzer
```

Co-funded by the Erasmus+ Programme of the European Union

```
def destroy():
    lcd.clear()                 # Sgomberare lo schermo LCD
    GPIO.cleanup()                  # Rilasciare tutti i GPIO

PCF8574_address = 0x27  # Indirizzo I2C del chip PCF8574
PCF8574A_address = 0x3F  # Indirizzo I2C del chip PCF8574A
# Creare un adattatore PCF8574 GPIO
try:
    mcp = PCF8574_GPIO(PCF8574_address)
except:
    try:
        mcp = PCF8574_GPIO(PCF8574A_address)
    except:
        print ('I2C Address Error !')
        exit(1)
# Creare un LCD, passare ad un adattatore MCP GPIO
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)

if __name__ == '__main__':
    print ('The Play2Learn Parking Sensor Program is starting ... ')
    setup()
    try:
        main_loop()
    except KeyboardInterrupt:
        destroy()
```

Co-funded by the
Erasmus+ Programme
of the European Union

Esecuzione di circuiti e programmi

Mu 1.0.2 - PLAY2LEARN_EXERCISE_SOLUTION.py

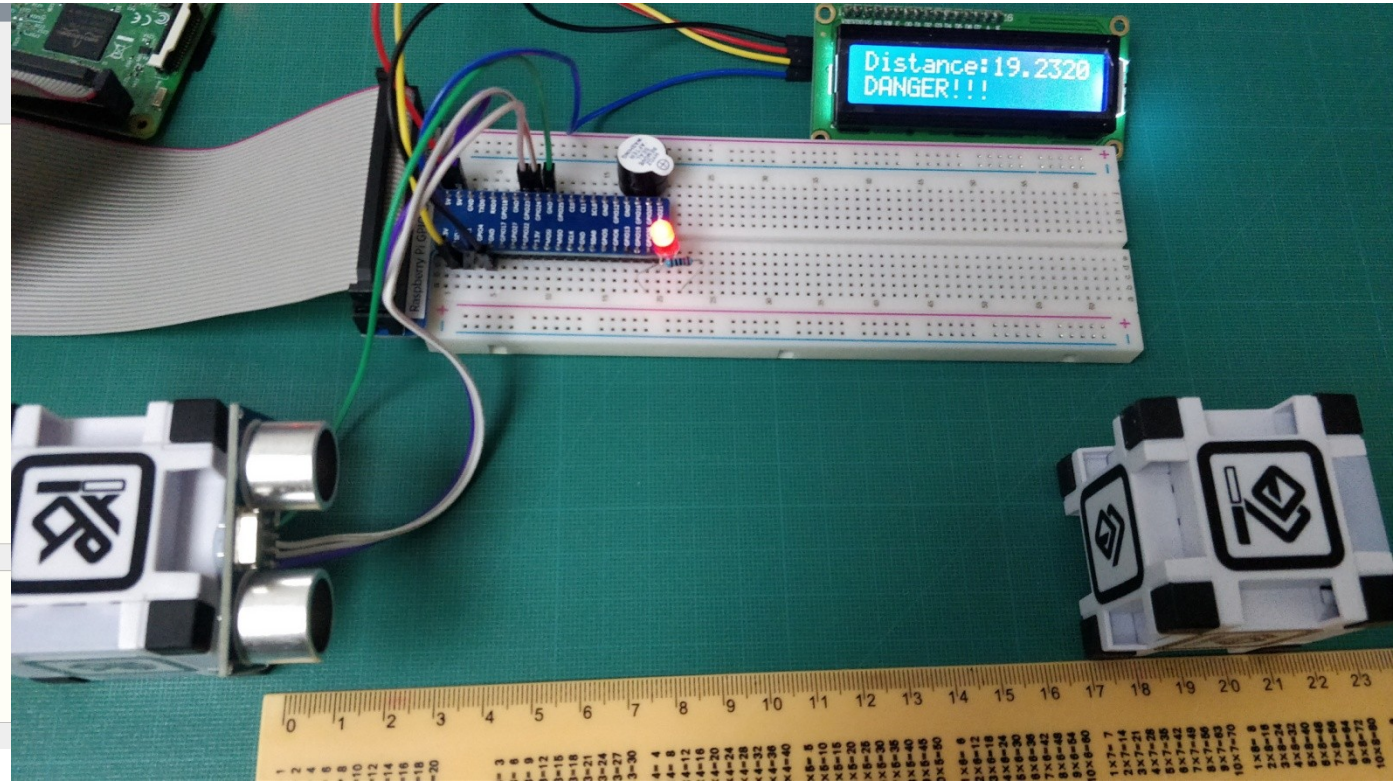
Mode | New | Load | Save | Stop | Debug | REPL | Plotter | Zoom-in | Zoom-out | Theme | Check | Help | Quit

untitled | PLAY2LEARN_EXERCISE_SOLUTION.py

```
from PCF8574 import PCF8574_GPIO               #Import the pin setup from the PCF8574.py program
from Adafruit_LCD1602 import Adafruit_CharLCD   #Import the methods from the Adafruit_LCD1602.py program

from time import sleep, strftime
from datetime import datetime

import RPi.GPIO as GPIO #Give access to the GPIO pins
import time             # Give access to the time module

def setup():
    GPIO.setmode(GPIO.BOARD)        # Use PHYSICAL GPIO Numbering
    GPIO.setup(trigPin, GPIO.OUT)   # Set trigPin to OUTPUT mode
    GPIO.setup(echoPin, GPIO.IN)    # Set echoPin to INPUT mode
    GPIO.setup(buzPin,GPIO.OUT)     # Set buzPin to OUTPUT mode

    GPIO.setup(ledPin, GPIO.OUT)   # Set the ledPin to OUTPUT mode
    GPIO.output(ledPin, GPIO.LOW)  # Make ledPin output LOW level

# For buzzer
buzPin=40       # Define the buzzer pin
# For LED
ledPin = 37     # Define the Led pin
```

Running: PLAY2LEARN_EXERCISE_SOLUTION.py

```
led turned on >>>
led turned off <<<
The distance is : 13.74 cm
led turned on >>>
led turned off <<<
The distance is : 13.75 cm
led turned on >>>
led turned off <<<
```



Co-funded by the
Erasmus+ Programme
of the European Union

Erasmus+

Reviving hands-on educational play for learning skills of tomorrow

PROJECT N° 2019-1-UK01-KA201-061466

CIVIC idec ΕΚΠΑΙΔΕΥΤΗΡΙΑ ΠΛΑΤΩΝ dlearn scholé Emphasys CENTRE CCS Digital Education