**Nume, Prenume (Grupa) :**     Bohotineanu Florin(1406A)

**Tema proiectului:** *Senzor de parcare*

Proiectul presupune calcularea distanței între un senzor și un obiect extern, iar în funcție de distanța calculată se va aprinde un led cu o rată și o intensitate care variază în funcție de distanță.

**Am folosit:** Raspberry Pi Zero v1.3

Senzor de distanță HC-SR04

Led de culoare roșie

**Rezumat:** În funcție de distanța calculată, se va aprinde și se va stinge un led roșu cu o rată și viteză care variază în funcție de distanță, dar nu mai mult de 0.5 secunde.

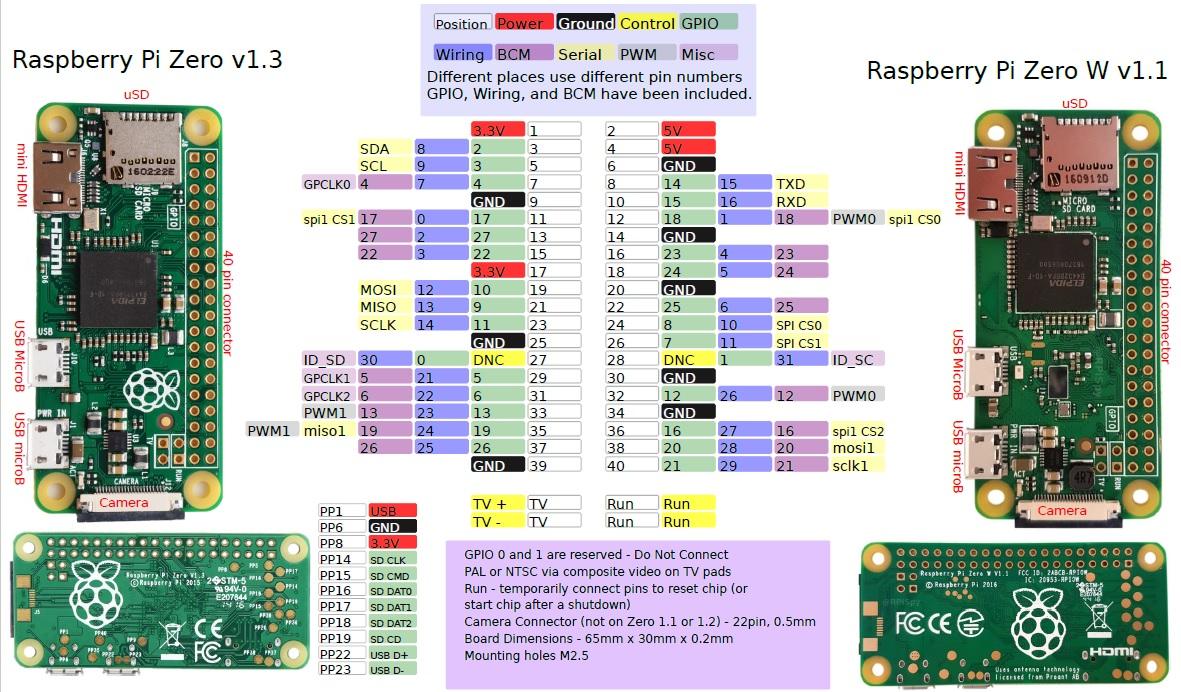
Calcul distanță: Timpul luat de puls este de fapt pentru și de la deplasarea semnalelor ultrasonice, în timp ce avem nevoie de doar jumătate din aceasta. Prin urmare, timpul este luat ca timp / 2.

Distanța = Viteza \* Timp / 2, în cazul nostru viteza este 34300 cm/s.

**Explorare documentara asupa temei,raport sintetic alternative solutie:**

HC-SR04 : <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>

Raspberry Pi Zero v1.3: <https://cdn.sparkfun.com/assets/learn_tutorials/6/7/6/PiZero_1.pdf>

<http://roboromania.ro/wp-content/uploads/2020/01/raspberry-pi-zero-w-roboromania-pini.jpg>

Ne-am conectat la plăcuță prin SSH la raspberrypi.local, port 22 unde s-a scris codul Python și de unde se execută acesta.

Plăcuța poate fi configurată să ruleze programul la boot astfel că fiind conectată la o mașină, la o sursă de tensiune de la aceasta, poate deveni astfel un senzor de parcare eficient și util oricărui șofer.

**Cod sursă Python:**

#Libraries

import RPi.GPIO as GPIO

import time

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 12

GPIO\_ECHO = 24

led = 11

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

GPIO.setup (led, GPIO.OUT)

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

if \_\_name\_\_ == '\_\_main\_\_':

pwm\_led = GPIO.PWM(led,50)

pwm\_led.start(100)

print("Distance:")

try:

while True:

dist = distance()

print ("Measured Distance = %.1f cm" % dist)

if int(dist) < 100 and int(dist) > 0:

duty=int(dist)

pwm\_led.ChangeDutyCycle(100-duty)

dist\_old=dist

if dist/50 > 0.5: # daca distanta este mai mare de 50 cm

temp=float(float(dist)/50)/0.5 #imparte in portiuni de 0.5 secunde distanta de mai sus

print("temp=%.lf",temp)

i=1

while i <= int(temp): #la fiecare 0.5s se citeste noua distanta si daca este mai mica decat 2/3

print(i) #din distanta veche atunci se incepe imediat calcularea noii distante

i=i+1 #si ledul va afisa noi valori corespunzatoare noii distante imediat

time.sleep(0.5)

dist=distance()

if dist < 2\*dist\_old/3:

pwm\_led.ChangeDutyCycle(0)

break

break

print("time.sleep extra: %.lf" % float(float(dist\_old)/50-temp\*0.5))

time.sleep(float(float(dist\_old)/50-temp\*0.5))

else:

time.sleep(dist/50) # timp in care ledul este aprins

pwm\_led.ChangeDutyCycle(0)

if 0.1\*dist/10 <= 0.5:

time.sleep(0.1\*dist/10) # timp in care ledul sta oprit si nu se calculeaza distanta, maxim 0.5 sta stins

else:

time.sleep(0.5)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

