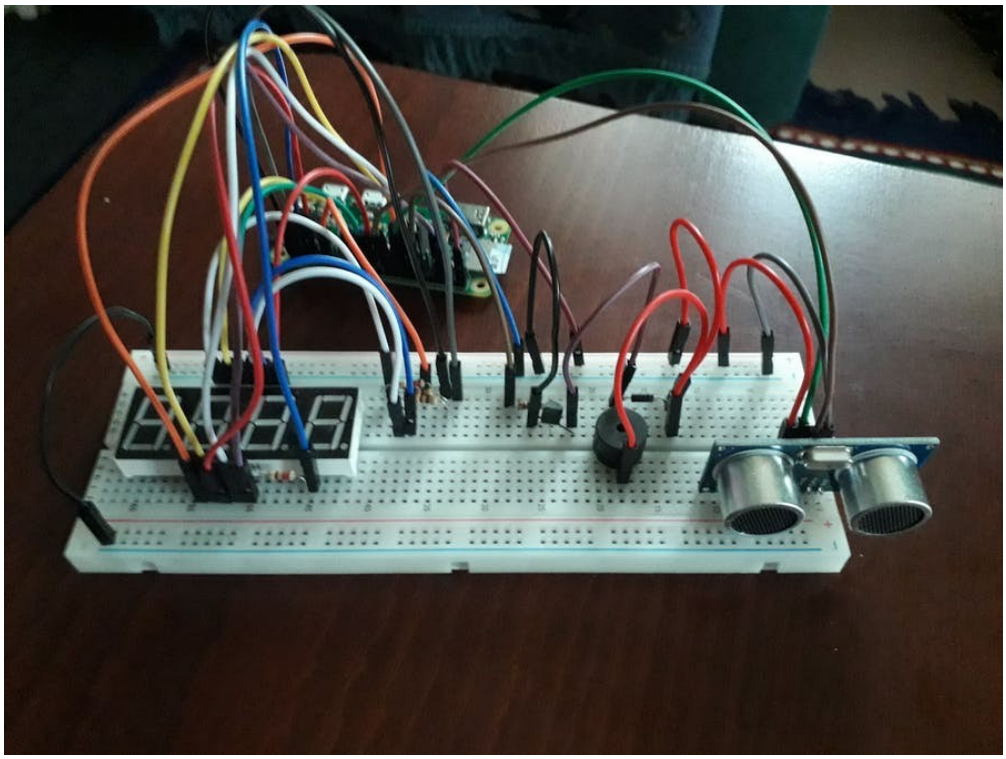
Nume: Buhai Ioan
Adresa e-mail: ioan.buhai@student.tuiasi.ro
Poza:

Hackster.io project link: https://www.hackster.io/ioan-buhai/rpi-distance-frequency-03bb33
Proiect: Detectare miscare si interpretarea distantei dintre senzor si obiect printr-un buzzer care-si va modifica frecventa in functie de distanta + afisarea distantei (cm) urmata de a frecventei(hz) pe un display.
Descriere Proiect (Conform Hackster.io rules)

RPi-Distance&Frequency.

The project is made using Python3 programming on a Raspberry Pi Zero W



Things used in this project:

Hardware components:

- **Raspberry Pi Zero Wireless x1**
- **Breadboard x1**
- **Buzzer x1**
- **4 Digit 7 Segment Display x1**
- **Ultrasonic Sensor - HC-SR04 x1**
- **1N4007 - High Voltage, High Current Rated Diode x1**
- **General Purpose Transistor NPN x1**
- **150 ohm Resistor x4**
- **1k ohm Resistor x1**
- **Male/Female Jumper Wires**
- **Male/Male Jumper Wires**

Software apps and online services:

- **Raspberry Pi Rasbian**
- **WinSCP (Used for files transfer between Laptop and RPi)**
- **PuTTY (Used for testing/compiling the code)**

Story:

This project is a good exercise for the python programming language lovers who want to extend their knowledge of the Embedded systems.

Small things good to know about the components:

The display in this project is used to show the distance of the object that's in front of the ultrasonic ranging HC-SR04 module and afterwards, a specific frequency that's also going to be used by the passive buzzer to produce a sound.

The buzzer is controlled by a Pulse Width Modulation (PWM), with a duty cycle of 50%.

The buzzer is powered by a 5V output voltage pin of the Raspberry Pi Zero W (see schematic).

Also, the buzzer generates a reverse voltage when unplugged, because of that we are using a diode to avoid any unexpected results.

For trying out this project you need the following things:

1. Installing NOOBS:

1.a. First of all, you need an atleast 4GB SD card, that needs to be formatted to FAT file system, not ExFat system (if the card is >16GB, the Microsoft Windows 10 format tool may not give the FAT option)

1.b. Download NOOBS at https://www.raspberrypi.org/downloads/noobs/ and unzip it on the freshly formatted SD card.

1.c. Insert the SD card in your RPi Zero W, power it on, attach a desktop device (ex. TV) by using the HDMI port on the board.

1.d. Install NOOBS.

2. pigpio library (For it, check out the next link: http://abyz.me.uk/rpi/pigpio/download.html)

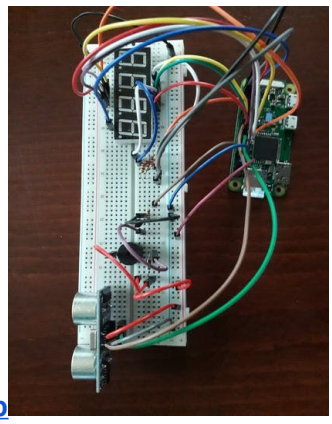
3. Use the code below, copy it on your RPi by using WinSCP or any other way you preffer.

4. Make the setup & Enjoy.

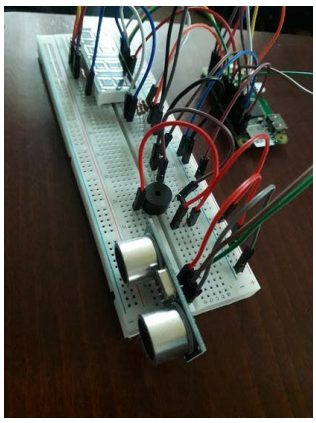
I must specify the way it works:

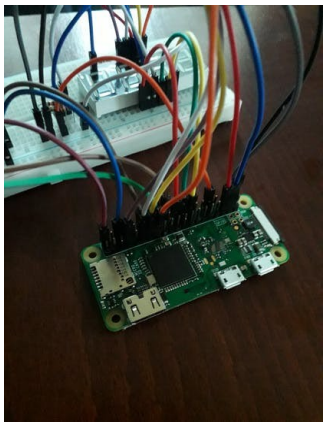
-- First, it gets the distance, displays it for 2 seconds, and afterwards the frequency and the buzzer rings for that amount of frequency.

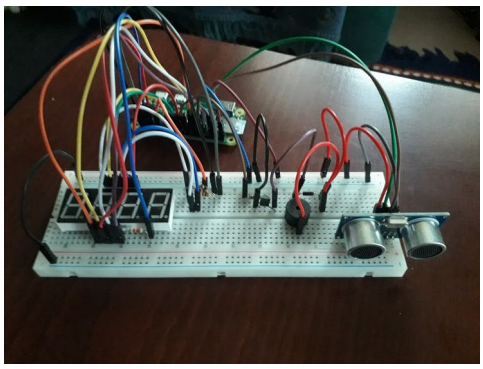
Below i've attached a video i recorded of how the project works and some pictures of the project.



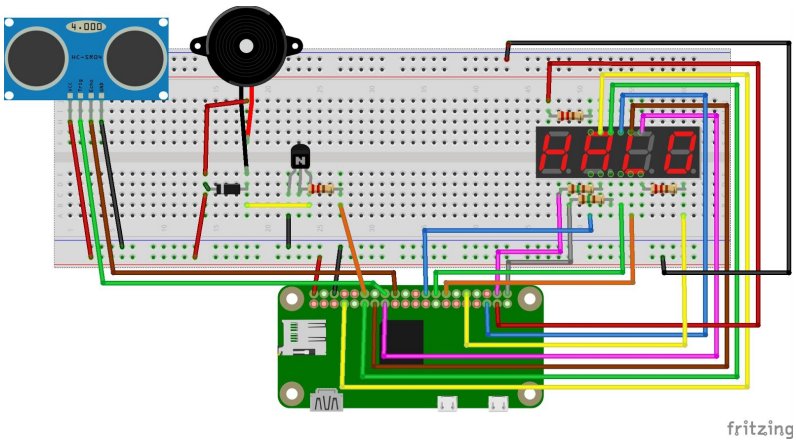
https://www.youtube.com/watch?v=UY1Jj1-QkA0&feature=youtu.b







Schematic:



Code:

```
import RPi.GPIO as GPIO
import time
import pigpio
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
pi = pigpio.pi()
# global variables
Segments = (a, b, c, d, e, f, g, dot) = (26, 22, 11, 13, 15, 24, 7, 35)
Digits = (D1, D2, D3, D4) = (32, 40, 38, 37)
Trig = 16
Echo = 18
Buzzer = 18
GPIO.setup(Trig, GPIO.OUT)
GPIO.setup(Echo, GPIO.IN)
for d in Digits:
        GPIO.setup(d, GPIO.OUT)
        GPIO.output(d, GPIO.HIGH)
for s in Segments:
        GPIO.setup(s, GPIO.OUT)
        GPIO.output(s, GPIO.LOW)
def display_segment(number, dot):
null = [0,0,0,0,0,0,0,0]
zero = [1,1,1,1,1,1,0,0]
one = [0,1,1,0,0,0,0,0]
two = [1,1,0,1,1,0,1,0]
three = [1,1,1,1,0,0,1,0]
four = [0,1,1,0,0,1,1,0]
five = [1,0,1,1,0,1,1,0]
six = [1,0,1,1,1,1,1,0]
seven = [1,1,1,0,0,0,0,0]
eight = [1,1,1,1,1,1,1,0]
nine = [1,1,1,1,0,1,1,0]
n = [0,1,1,0,1,1,1]
u = [0,1,1,1,1,1,0]
```

```
l = [0,0,0,1,1,1,0]
        if number == 1:
         for i in range(8):
GPIO.output(Segments[i], one[i])
    if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 2:
         for i in range(8):
         GPIO.output(Segments[i], two[i])
     if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 3:
         for i in range(8):
         GPIO.output(Segments[i], three[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 4:
         for i in range(8):
         GPIO.output(Segments[i], four[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 5:
         for i in range(8):
         GPIO.output(Segments[i], five[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 6:
         for i in range(8):
         GPIO.output(Segments[i], six[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 7:
         for i in range(8):
         GPIO.output(Segments[i], seven[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 8:
         for i in range(8):
         GPIO.output(Segments[i], eight[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 9:
         for i in range(8):
         GPIO.output(Segments[i], nine[i])
if i == 7 and dot == 1:
```

```
GPIO.output(Segments[i], 1)
        elif number == 0:
         for i in range(8):
         GPIO.output(Segments[i], zero[i])
if i == 7 and dot == 1:
GPIO.output(Segments[i], 1)
        elif number == 10:
for i in range(7):
GPIO.output(Segments[i], n[i])
        elif number == 11:
for i in range(7):
GPIO.output(Segments[i], u[i])
        elif number == 12:
for i in range(7):
GPIO.output(Segments[i], l[i])
def display_digit(digit, number, dot):

display_segment(number, dot)
GPIO.output(Digits[digit], GPIO.LOW)
time.sleep(0.0007)
GPIO.output(Digits[digit], GPIO.HIGH)

def display_distance(number, type):
#type = 0 => frecventa
#type = 1 => cm
if number < 1000:
n3 = (int)((number * 10) % 10)
n2 = (int)((number) % 10)
n1 = (int)((number // 10) % 10)
n0 = (int)(number // 100)
#print("n0 =",n0," n1=",n1," n2=",n2," n3=",n3)
if type == 1:
if n0 != 0:
display_digit(0,n0,0)
display_digit(1,n1,0)
display_digit(2,n2,1)
display_digit(3,n3,0)
elif type == 0:
if n0 !=0:
display_digit(0,n0,0)
elif n0 == 0:
display_digit(0,0,0)
display_digit(1,n1,0)
display_digit(2,n2,0)
display_digit(3,10,0) #hertz

if number>=1000:
```

```
display_digit(0,10,0)
display_digit(1,11,0)
display_digit(2,12,0)
display_digit(3,12,0)

def get_distance():
GPIO.output(Trig, GPIO.HIGH)
time.sleep(0.00001)
GPIO.output(Trig, GPIO.LOW)
start = time.time()
stop = time.time()
while GPIO.input(Echo) == 0:
start = time.time()
while GPIO.input(Echo) == 1:
stop = time.time()
duration = stop-start
distance = 34300/2 * duration
print ("Distance = %.2f" % distance)
return distance
try:
buzzer = 0
freq = 0
while True:
sleeper = 0
if buzzer == 0:
distance = get_distance()
if distance > 0 and distance < 150:
freq = distance * 3
elif distance > 500 and distance < 1000:
freq = distance / 2
elif distance > 1000:
freq = 250
elif distance > 150 and distance < 500:
freq = distance
while sleeper < 1500:
display_distance(distance,1)
sleeper = sleeper + 1
if buzzer == 1:
pi.hardware_PWM(Buzzer, freq, 500000)
while sleeper < 500:
display_distance(freq, 0)
sleeper = sleeper + 1
if buzzer == 0:
buzzer = 1
elif buzzer == 1:
buzzer = 0
pi.hardware_PWM(Buzzer,0,0)
```

```
except KeyboardInterrupt:
pass
pi.hardware_PWM(Buzzer, 0, 0)
GPIO.cleanup()
```