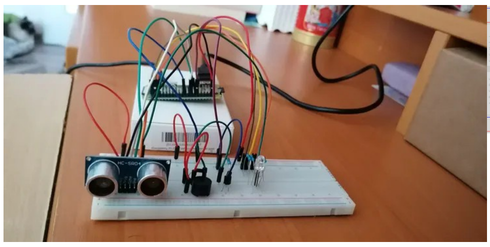
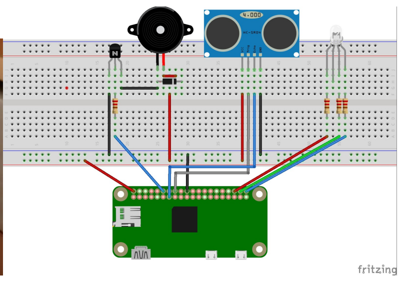
Titlu proiect: “Parking sensor alarm”

Gavril Alexandra, Daraba Sabina, Chiriloaie Cosmin, Ciobanu Andreea

Emails: alexandra-dumitrita.gavril@student.tuiasi.ro
sabina-sinziana.daraba@student.tuiasi.ro
cosmin.chiriloaie@student.tuiasi.ro
daiana-andreea.ciobanu@student.tuiasi.ro

https://www.hackster.io/342470/parking-sensor-alarm-ddd8d5



A system that can be used by anyone who drives a car. It allows them to park properly without causing any damage to their vehicle

Hardware components:

- Raspberry Pi Zero Wireless
- Solderless Breadboard Full Size
- Male/Female Jumper Wires
- Ultrasonic Sensor - HC-SR04 (Generic)
- Buzzer
- RGB LED
- Resistor 1k ohm, Resistor 100 ohm
- General Purpose Transistor NPN
- 1N4007 – High Voltage High Current Rated Diode

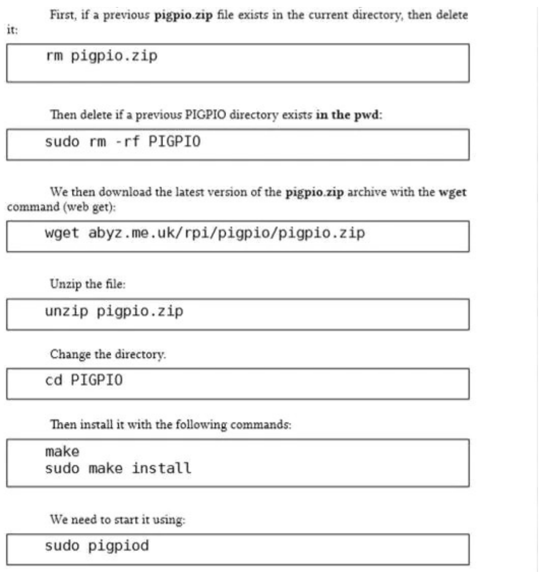
Our main goal in creating this project was to set up a system that can help people parking their vehicles properly and without any possible damage.

We built a simple circuit with Raspberry Pi and HC-SR04 ultrasonic range sensor, which will simply beep and display a different color according to how far it is from an obstacle.

- The system measures the distance between the obstacle and our sensor
- Blue light and no beeping will tell us that everything around us is far away.
- Green light and no beeping will start as we will be getting closer to the object.
- Red light and high beeping will start as we reach a dangerously close position to our obstacle.

Main steps into creating this project:

- **Raspberry Pi Zero configuration**
- **Hardware set up**
- **Install pigpio**
- **Developing the application**


First, if a previous **pigpio.zip** file exists in the current directory, then delete it:

```
rm pigpio.zip
```

Then delete if a previous PIGPIO directory exists **in the pwd**:

```
sudo rm -rf PIGPIO
```

We then download the latest version of the **pigpio.zip** archive with the **wget** command (web get):

```
wget abyz.me.uk/rpi/pigpio/pigpio.zip
```

Unzip the file:

```
unzip pigpio.zip
```

Change the directory.

```
cd PIGPIO
```

Then install it with the following commands:

```
make
sudo make install
```

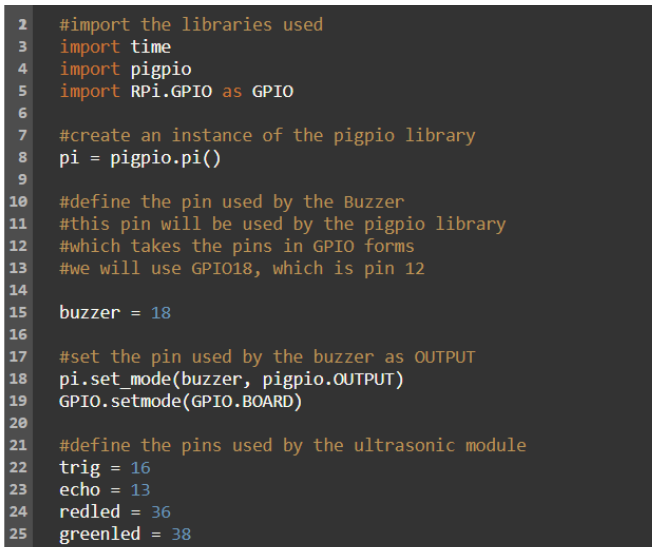
We need to start it using:

```
sudo pigpiod
```

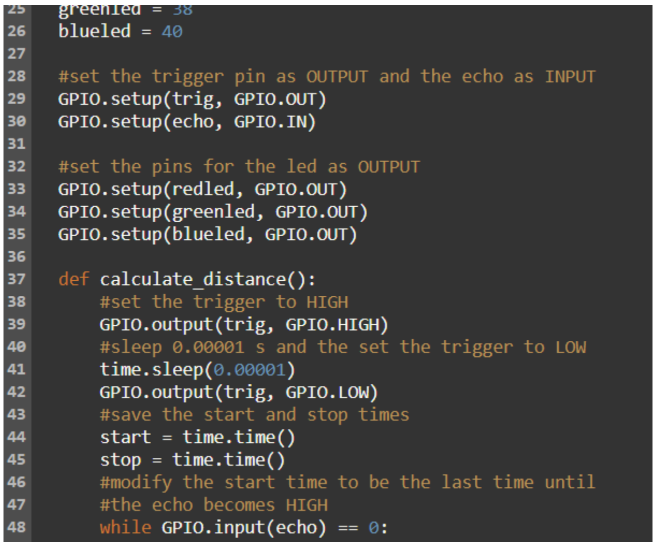
How does this project function:

https://youtu.be/efr54I0cDG8

https://www.youtube.com/watch?time_continue=5&v=gSNHBCtoOg4&feature=emb_logo

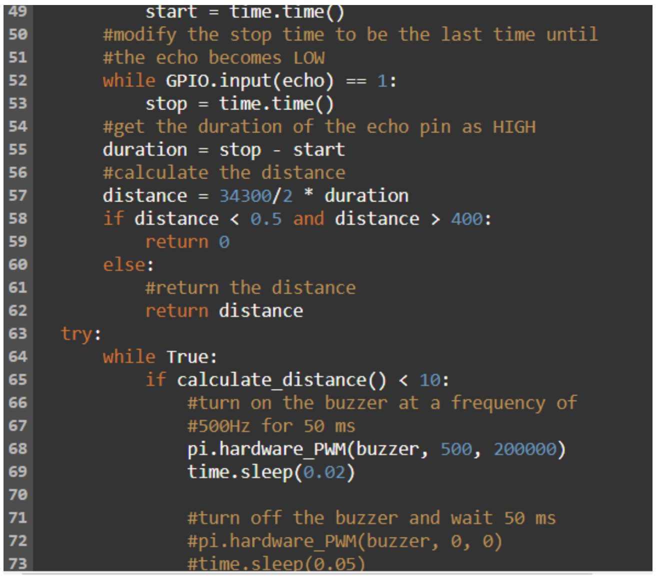


```python
#import the libraries used
import time
import pigpio
import RPi.GPIO as GPIO

#create an instance of the pigpio library
pi = pigpio.pi()

#define the pin used by the Buzzer
#this pin will be used by the pigpio library
#which takes the pins in GPIO forms
#we will use GPIO18, which is pin 12

buzzer = 18

#set the pin used by the buzzer as OUTPUT
pi.set_mode(buzzer, pigpio.OUTPUT)
GPIO.setmode(GPIO.BOARD)

#define the pins used by the ultrasonic module
trig = 16
echo = 13
redled = 36
greenled = 38
```

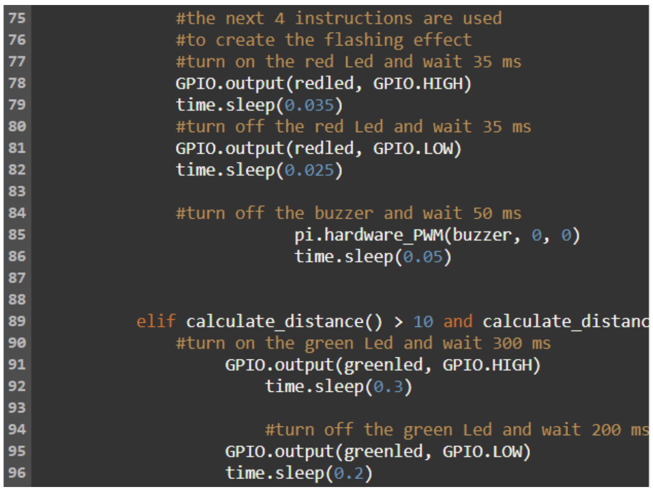


greenled = 38

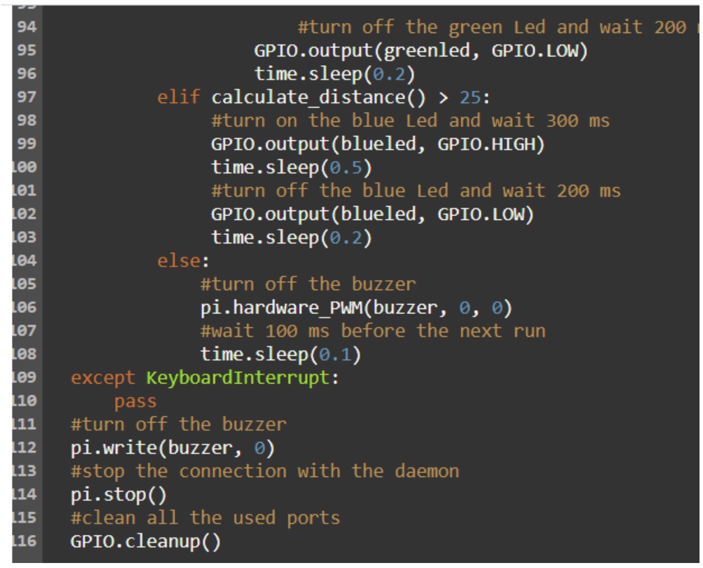
```python
blueled = 40

#set the trigger pin as OUTPUT and the echo as INPUT
GPIO.setup(trig, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)

#set the pins for the led as OUTPUT
GPIO.setup(redled, GPIO.OUT)
GPIO.setup(greenled, GPIO.OUT)
GPIO.setup(blueled, GPIO.OUT)

def calculate_distance():
    #set the trigger to HIGH
    GPIO.output(trig, GPIO.HIGH)
    #sleep 0.00001 s and the set the trigger to LOW
    time.sleep(0.00001)
    GPIO.output(trig, GPIO.LOW)
    #save the start and stop times
    start = time.time()
    stop = time.time()
    #modify the start time to be the last time until
    #the echo becomes HIGH
    while GPIO.input(echo) == 0:
```



```python
        start = time.time()
    #modify the stop time to be the last time until
    #the echo becomes LOW
    while GPIO.input(echo) == 1:
        stop = time.time()
    #get the duration of the echo pin as HIGH
    duration = stop - start
    #calculate the distance
    distance = 34300/2 * duration
    if distance < 0.5 and distance > 400:
        return 0
    else:
        #return the distance
        return distance
try:
    while True:
        if calculate_distance() < 10:
            #turn on the buzzer at a frequency of
            #500Hz for 50 ms
            pi.hardware_PWM(buzzer, 500, 200000)
            time.sleep(0.02)

            #turn off the buzzer and wait 50 ms
            #pi.hardware_PWM(buzzer, 0, 0)
            #time.sleep(0.05)
```



```
            #the next 4 instructions are used
            #to create the flashing effect
            #turn on the red Led and wait 35 ms
            GPIO.output(redled, GPIO.HIGH)
            time.sleep(0.035)
            #turn off the red Led and wait 35 ms
            GPIO.output(redled, GPIO.LOW)
            time.sleep(0.025)

            #turn off the buzzer and wait 50 ms
                        pi.hardware_PWM(buzzer, 0, 0)
                        time.sleep(0.05)


        elif calculate_distance() > 10 and calculate_distanc
            #turn on the green Led and wait 300 ms
                 GPIO.output(greenled, GPIO.HIGH)
                     time.sleep(0.3)

                     #turn off the green Led and wait 200 ms
                 GPIO.output(greenled, GPIO.LOW)
                 time.sleep(0.2)
```



```
                     #turn off the green Led and wait 200 
                 GPIO.output(greenled, GPIO.LOW)
                 time.sleep(0.2)
```

```
        elif calculate_distance() > 25:
             #turn on the blue Led and wait 300 ms
             GPIO.output(blueled, GPIO.HIGH)
             time.sleep(0.5)
             #turn off the blue Led and wait 200 ms
             GPIO.output(blueled, GPIO.LOW)
             time.sleep(0.2)
        else:
            #turn off the buzzer
            pi.hardware_PWM(buzzer, 0, 0)
            #wait 100 ms before the next run
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
#turn off the buzzer
pi.write(buzzer, 0)
#stop the connection with the daemon
pi.stop()
#clean all the used ports
GPIO.cleanup()
```