

**Name: Mini Mp3 Player**

**Team:**

1.Ghiba Răzvan-Constantin 1310A



2.Condrea Andreea 1305A



3.Flocea Ana-Gabriela 1305A



4.Marin Olivia 1305A



## **Elevator Pitch:**

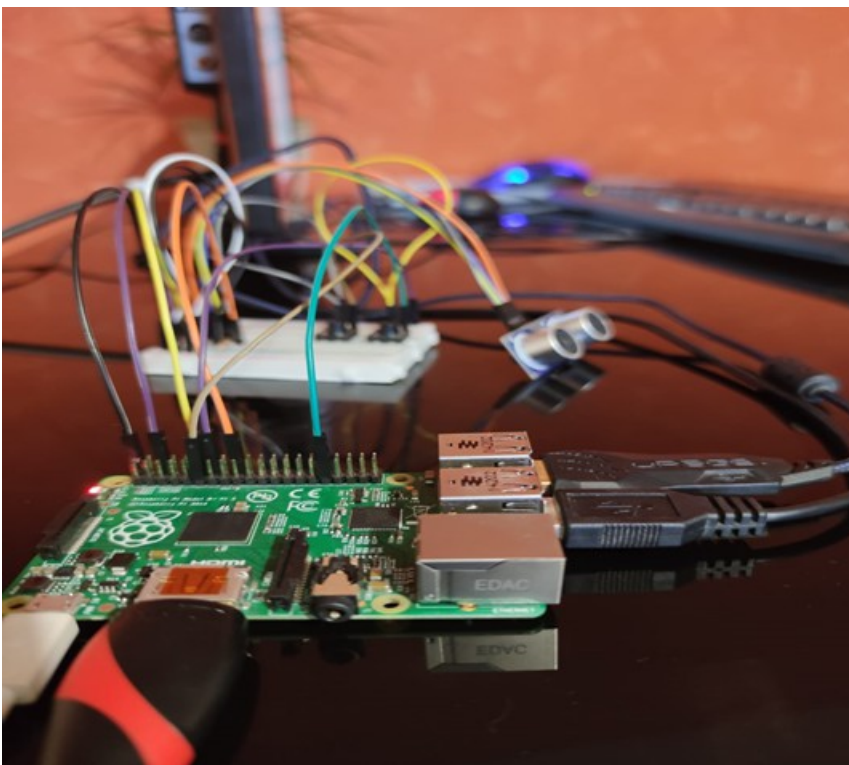
What would our life be without music? Boring, isn't it?

We are convinced that, at least once, while listening music on your new Mp3 Player, you asked yourself, ``Could I make something like that?`` Well, maybe we have an answer for you. You want to find out? Then, we invite you to follow our process of creating a mini Mp3 player. Enjoy!

## **Story:**

Our goal was to create a mini Mp3 player, using Raspberry Pi, with a list of funny songs, easy to use by users, but just as easy to create by any beginner in the field of microprocessors. The volume is controlled with a simple touch of the hand.

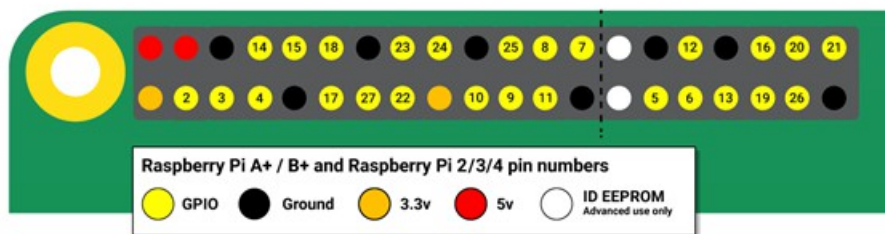
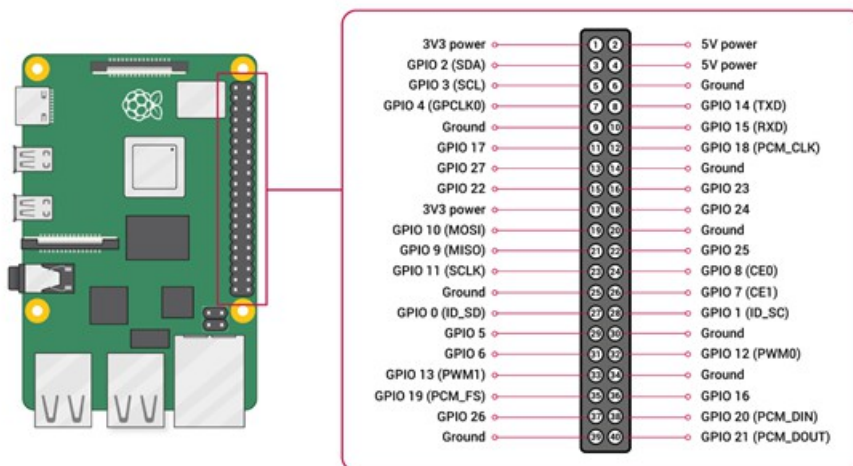
## **Cover Image:**



## **Components:**

- **Hardware:**
  - o A Raspberry Pi computer
  - o A breadboard
  - o 1x HC-SR04 sensor.
  - o Two tactile switches ( buttons )
  - o 11 male-to-female jumper leads
  - o 4 male-to-male jumper leads
  - o 2x 1k ohm resistors
  - o Tv as monitor
- **Software:**
  - o Raspberry PI Raspbian
  - o Python 2

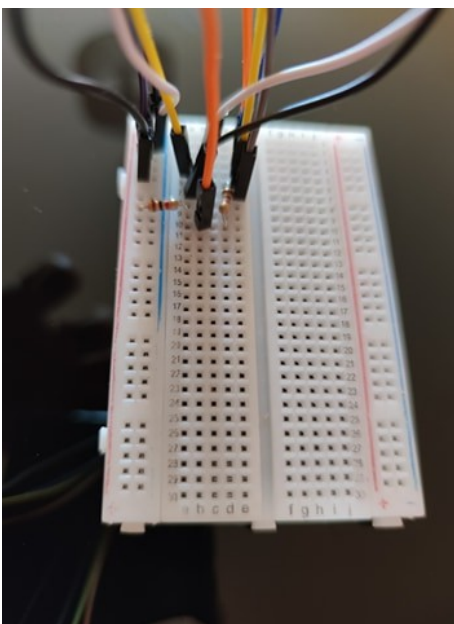
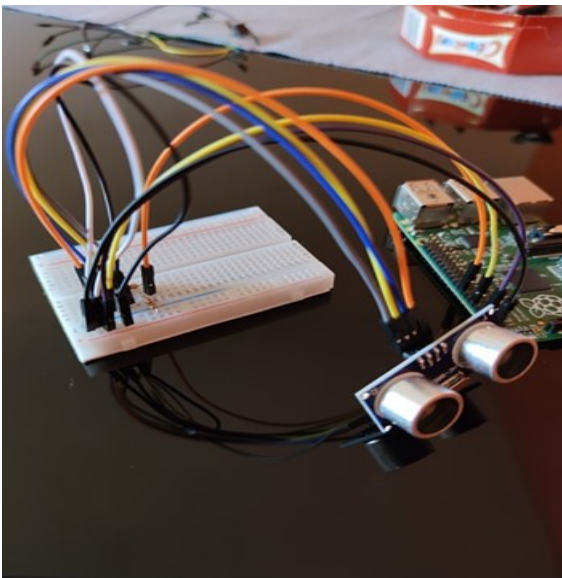
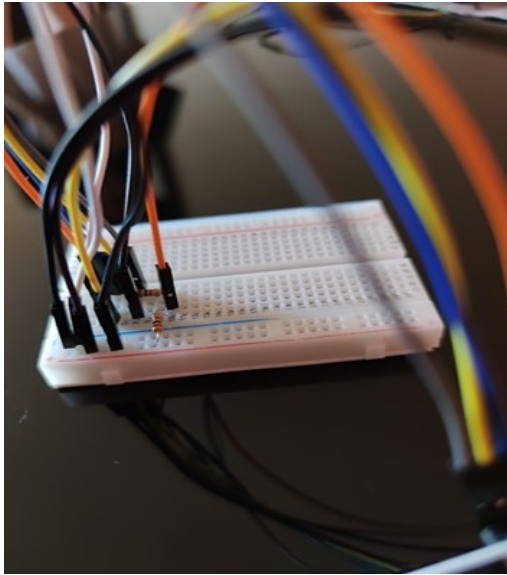
**Schematics:**

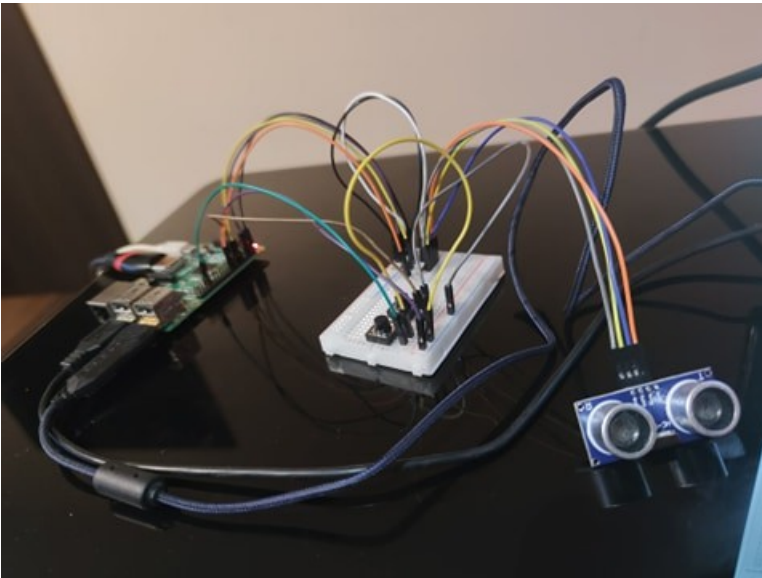


**Steps:**

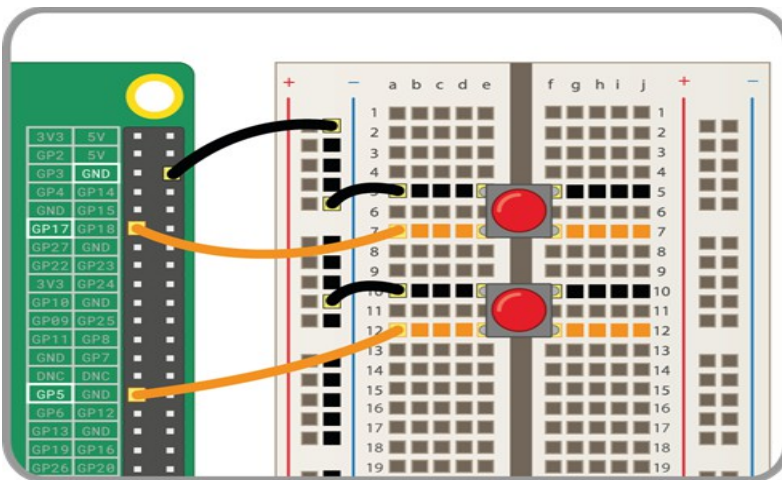
- First of all, we have to assemble the circuit. Below we have some images suggestive for assembly, very easy to follow and understand.

- o The sensor aims to take the distance from the user's hand and adjust the volume accordingly.





- o We have two buttons. The first is to turn on the mp3 player and change the song. The second stops the current song.



- Now that we have assembled the circuit, we can move on to the software development part.

- o Firstly, we will download a few songs to make the list.

**Be careful: Files must be in .wav format!**

- o In order to work with such files we need the pygame library:

**sudo apt-get install python-pygame**

We'll also need the alsaaudio library, to set the volume:

**sudo -H pip3 install pyalsaaudio**

and GPIO library, to connect the microcontroller to our electronic devices:

**sudo apt install python3-gpiozero**

- o After we download all the libraries, we can start to write the code:
  - we import all the libraries

```
import alsaaudio
import RPi.GPIO as GPIO
import time
from gpiozero import Button
import pygame
```

Ø we set our GPIO pin numbering using **setmode()**. We'll name our output pin(trigger the sensor) GPIO\_TRIGGER( Pin 18) and our input pin GPIO\_ECHO(Pin 24).

**btn** and **btn1** are corresponding to the two buttons: btn to stop the song, and btn1 change the song.



```

pygame.init()
GPIO.setmode(GPIO.BCM)
GPIO_TRIGGER = 18
GPIO_ECHO = 24
m = alsaaudio.Mixer('PCM')
volum = m.getvolume()
btn=Button(17)
btn1=Button(5)

```

- **distance()** is the function which calculate the distance between the user's hand and the sensor. Also, in this function we set the volume in correspondence with the calculated distance

```

15 def distance():
16     GPIO.setup(GPIO_TRIGGER,GPIO.OUT)
17     GPIO.setup(GPIO_ECHO,GPIO.IN)
18     GPIO.output(GPIO_TRIGGER,False)
19     time.sleep(2)
20     GPIO.output(GPIO_TRIGGER,True)
21     time.sleep(0.00001)
22     GPIO.output(GPIO_TRIGGER,False)
23     while GPIO.input(GPIO_ECHO) == 0:
24         pulse_start = time.time()
25     while GPIO.input(GPIO_ECHO) == 1:
26         pulse_end = time.time()
27     pulse_duration = pulse_end - pulse_start
28     distance = pulse_duration *17150
29     distance = round(distance,2)
30     print("Distance : ", distance, "cm")
31     if distance > 100:
32         new_volume = 90
33     elif distance<20:
34         new_volume=70
35     if distance>=20 and distance<=100:
36         new_volume=80
37     m.setvolume(new_volume)

```

∅ we create a list with all of our songs.

```

39 my_sound=[pygame.mixer.Sound("/home/pi/Desktop/song0.wav"),
40           pygame.mixer.Sound("/home/pi/Desktop/song1.wav"),
41           pygame.mixer.Sound("/home/pi/Desktop/song2.wav"),
42           pygame.mixer.Sound("/home/pi/Desktop/song3.wav"),
43           pygame.mixer.Sound("/home/pi/Desktop/song4.wav")]

```

- the most important part of the program is located in while(1) loop. The first while check the state of each button and begins to calculate the distance. If btn is pressed then the song is stopped. We press btn1 after stopping the previous song, and we move to the next song.

```

44 while 1:
45     while btn.is_pressed==False and btn1.is_pressed==False:
46         distance()
47     if index > 5:
48         index = 0
49     if btn.is_pressed == True:
50         print("Stopped")
51         print(index)
52         if index>0:
53             my_sound[index-1].stop()
54     if btn1.is_pressed == True:
55         print(index)
56         print("Next song")
57         if index<5:
58             my_sound[index].play()
59         index = index + 1
60         while pygame.mixer.get_busy() and btn.is_pressed==False:
61             distance()
62     GPIO.cleanup()

```

In the end, we must clean our GPIO pins to ensure that all inputs/outputs are reset.

**DEMO:** [https://drive.google.com/file/d/1-3vOaNoz0fSG9gBzLKcrkjuPFrUEePIJ/view?fbclid=IwAR12R7sZ28b\\_RhjVmubkKL8JEhJQFp8UrCFszKLeCqN9Chx4zyfg4QASGx4](https://drive.google.com/file/d/1-3vOaNoz0fSG9gBzLKcrkjuPFrUEePIJ/view?fbclid=IwAR12R7sZ28b_RhjVmubkKL8JEhJQFp8UrCFszKLeCqN9Chx4zyfg4QASGx4)