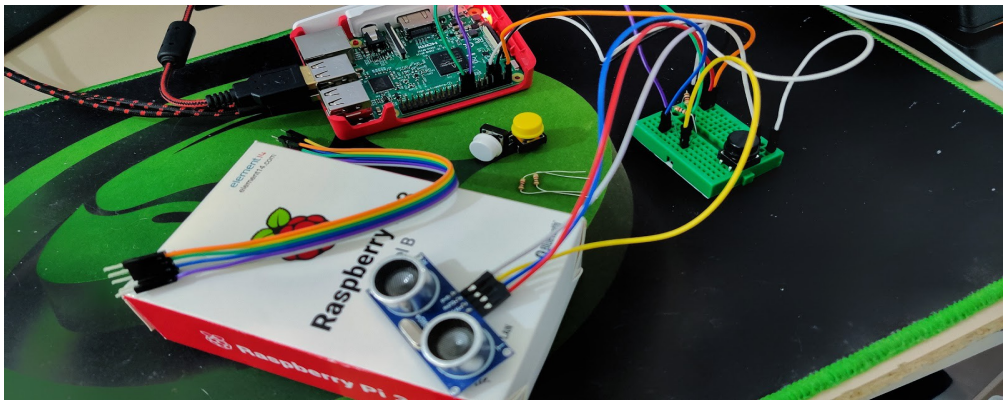
Turcu Ilinca Florinela 1305A
Marin Iulian Gabriel 1305A

Name: **Magic Music**

Elevator Pitch:

Have you ever thought about the hidden powers you have? Only one hand movement and you can have the World at your feet. Or in our case, you can become a magic DJ with a customizable “Magic Music” which lets you play different sounds when the button is pressed and controls the volume with a simple gesture of your hand.

Cover Image:



Components and apps:

Hardware:

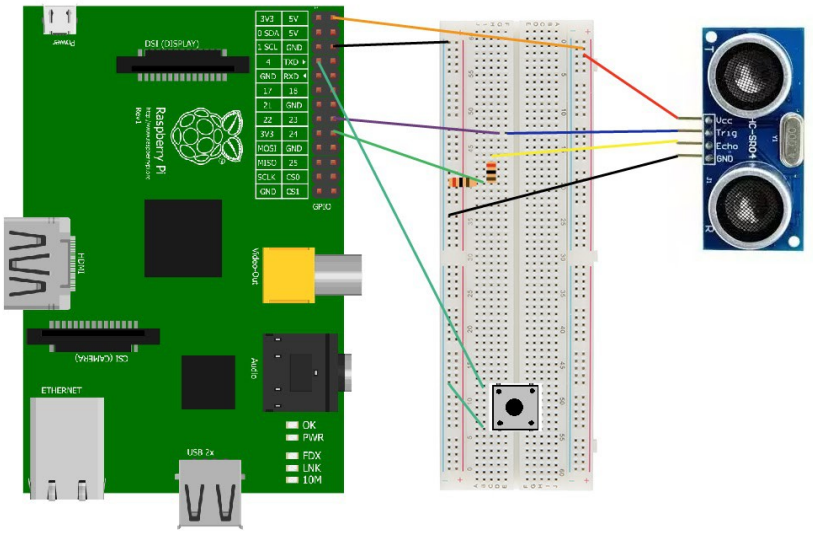
- **A Raspberry Pi computer**
- **A breadboard**
- **1x HC-SR04 sensor.**
- **One(1) tactile switches (to make a button)**
- **Five (5) male-to-female jumper leads**
- **Four (4) male-to-male jumper leads**
- **Speakers or headphones**
- **2x 1k ohm resistors.**

Software:

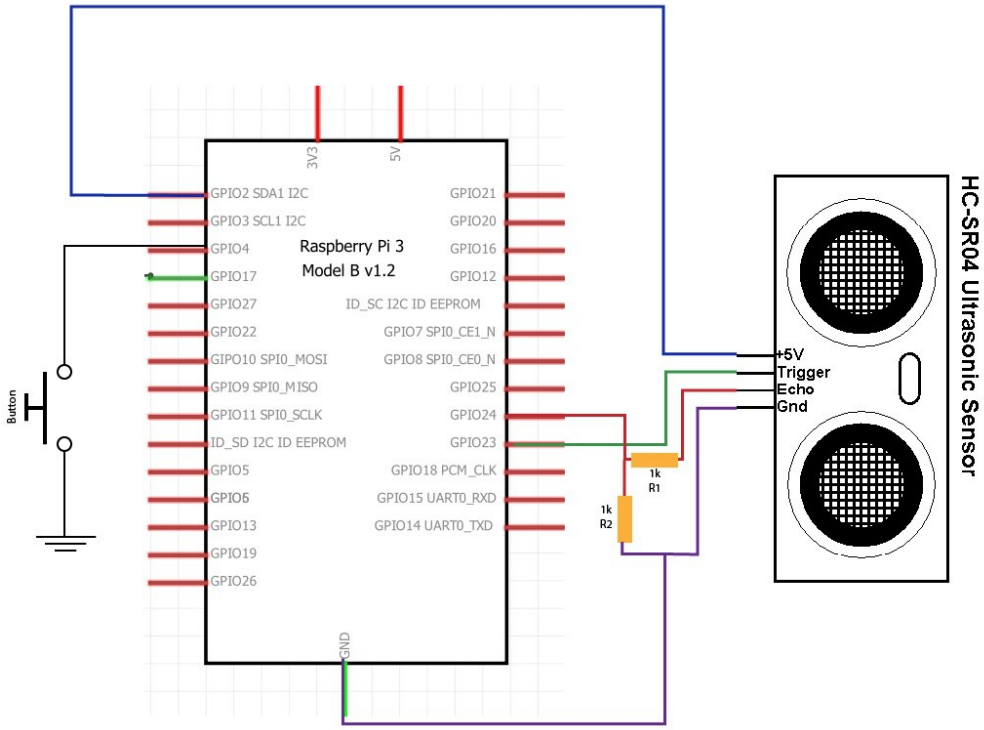
- **Raspberry PI Raspbian**
- **Python**

Code:

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

GPIO.setmode(GPIO.BCM)
m= alsaaudio.Mixer()
current_volume = m.getvolume()

pygame.init()
drum = pygame.mixer.Sound("/home/pi/gpio-music-box/sounds/disco-guitar-notes-3.wav")
btn_drum = Button(4)

TRIG = 23
ECHO = 24
print ("Distance Measurement In Progress")

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.output(TRIG, False)

print ("Waiting For Sensor To Settle")
time.sleep(2)

while 1:
    btn_drum.when_pressed = drum.play
    GPIO.output(TRIG, True)

    time.sleep(0.00001)
```


DSI (DISPLAY)
Raspberry Pi
GPIO
Video Out
Audio
ETHERNET
USB 2x
OK
PWR
FDX
LNK
10M
Vcc
Trig
Echo
GND
HC-SR04

Schematics



Story:

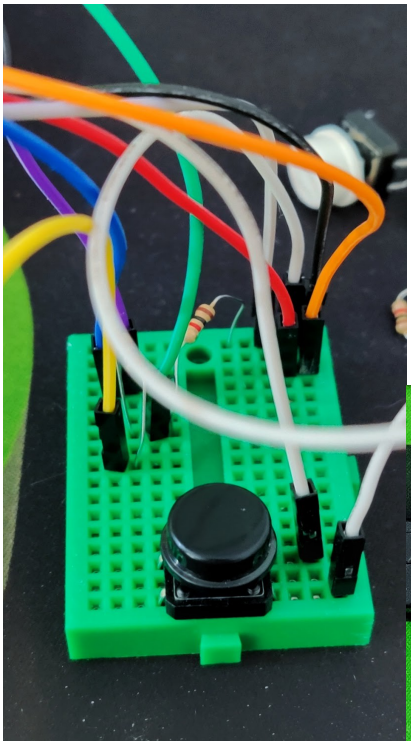
Our main objective was to make something which is fun to use and easy to customize by simply changing the tracks from the samples folder.

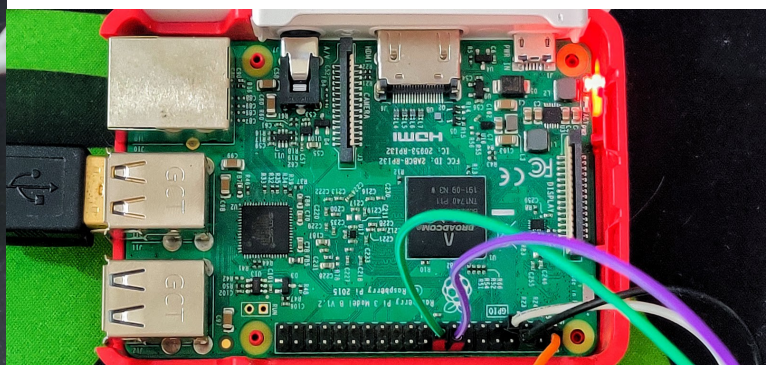
We built a simple circuit with Raspberry Pi, a breadboard, a proximity sensor and a button which

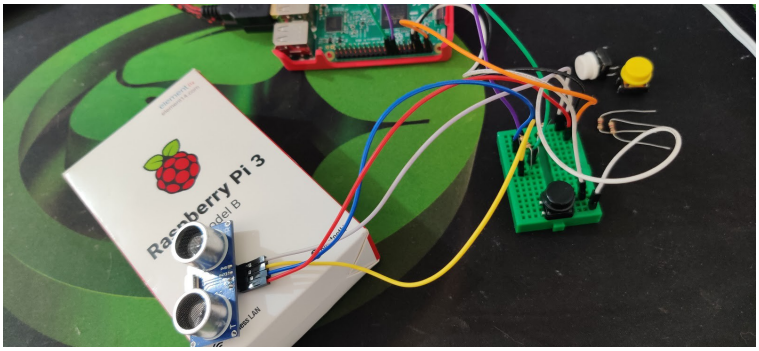
when pressed sends a signal to the Raspberry Pi, where the signal is interpreted and the track is played by the speaker, the volume being controlled by the distance of an object from the proximity sensor .

Main steps into creating this project:

- Get some audio files, a good bet is to take the sample files found in sonic-pi/samples folder. If python is used you need to convert the files from flac format to wav format.
- Set up your hardware.







Very important: The output signal of the HC-SR04 (ECHO) is rated 5V, while Raspberry Pi GPIO is rated at 3.3V. To not damage our GPIO pin you will need to construct a small voltage divider with 2 resistors as shown in the second picture.

- Develop the application.

If you decide to use python the first step is to import pygame module for playing sounds. Then, create a Python object that links to one of these sound files. Give the file its own unique name. Create your button objects according to the GPIO pins you put the physical buttons on.

```
pygame.init()
drum = pygame.mixer.Sound("/home/pi/gpio-music-box/sounds/disco-guitar-notes-3.wav")
btn_drum = Button(4)
```

Bind the button press events to their corresponding sound object.

```
while 1:
    btn_drum.when_pressed = drum.play
    GPIO.output(TRIG, True)

    time.sleep(0.00001)
```

We need to name the input and output pins, the output pin, the one that triggers the sensor GPIO 23 [Pin 16] will be named TRIG and the input pin,the one that receives the response GPIO 24 [Pin 18] will be named ECHO, and set them as either input or output.

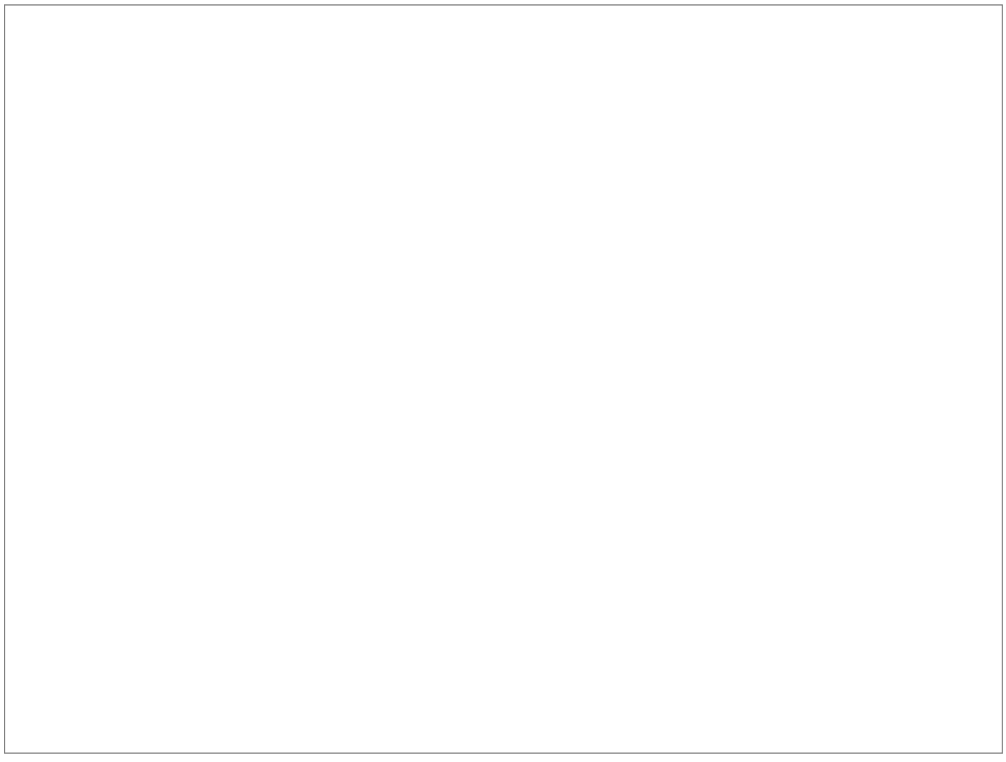
```
15  TRIG = 23
16  ECHO = 24
17  print ("Distance Measurement In Progress")
18
19  GPIO.setup(TRIG,GPIO.OUT)
20  GPIO.setup(ECHO,GPIO.IN)
21  GPIO.output(TRIG, False)
```

The proximity sensor needs a 10uS pulse to start the ranging program, in order to do that we set the TRIG high for 10us and then set it low. After this we need to calculate how much time the reflected sound waves need to come back to the sensor. ECHO will be high for the duration of this cycle. Finally with some clever math we calculate the distance based on the pulse duration.

```
    GPIO.output(TRIG, True)

    time.sleep(0.00001)

    GPIO.output(TRIG, False)

    while GPIO.input(ECHO)==0:

      pulse_start = time.time()

    while GPIO.input(ECHO)==1:

      pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration*17150
    distance = round(distance, 2)
```

After that you just need to set the volume based on distance in any way you like.

```
    time.sleep(0.00001)

    GPIO.output(TRIG, False)

    while GPIO.input(ECHO)==0:

      pulse_start = time.time()

    while GPIO.input(ECHO)==1:

      pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration*17150
    distance = round(distance, 2)

    print ("Distance:",distance,"cm")
    if int(round(distance /2.5, 0)) > 100 :
        distance_volume = 100
    distance_volume = int(round(distance /2.5, 0))
    m.setvolume(distance_volume)
    print ("     Volume:",distance_volume,"%")
    time.sleep(0.2)

GPIO.cleanup()
```



Funny version of the official video: https://drive.google.com/open?id=1L9tcaIKElPlVhnOmNfbrDhDOUOc9a1hx

---