**Wavepad - Gesture Controlled Raspberry Pi** **Music Player (With voice extension)**

**Team Members:**

(1) Sandeep Atkuri - 201501126 (2) Mihir Gajera - 201501127

(3) Shirish Jain - 201501128

(4) Pratik Wankhede - 201501129

**Explanation :**

Wavepad is a gesture and voice controlled music player. It is built using Raspberry pi board. With Wavepad we can increase and decrease the volume of the song, play next song, play previous song, pause and resume song. To recognise the gestures we are using 2 types of sensors. First one is Ultrasonic sensor which is used to control the volume and Other is IR proximity sensor x

2 which are used to change, pause and resume songs. We can also use voice commands to change the current playing song to the song which we wish to play.

**Real life application:**

Most of the music players have physical buttons to operate them, often requiring you to come physically and pause the song or play next song. Also if you want to play a specific track, you need to keep browsing through all the songs. However wavepad allows us to play a song by giving a voice command. Also many times, our hands are occupied doing other tasks, In that case voice commands are more convenient. Wavepad also enables you to play next, play previous, pause and increase or decrease volume through hand gestures. Physical buttons can get worn out after repeated use, but there is no such issue in gesture control. Overall it makes controlling the music player a lot more interactive.

**Equipments:**

Raspberry Pi 3 Model B Speaker

Ultrasonic Sensor

Photodiode (x2) IR LED (x2) Resistor 1K (x18) Resistor 10K (x2)

Resistor 100 Ohm (x2) Potentiometer 10K(x2) Bread Board

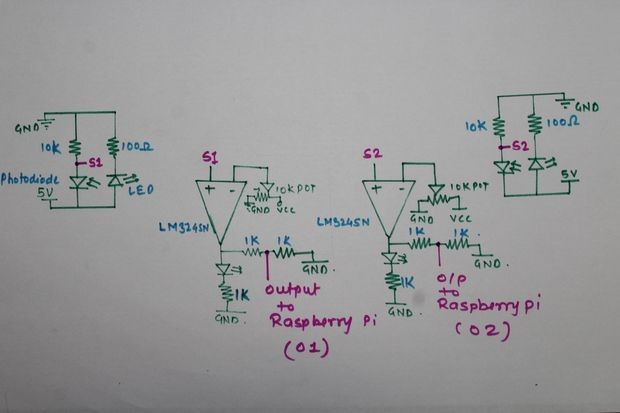
LED Blue (x12) Connecting Wires IC LM324SN (x1)

**Functionality:**

In our circuit there are three main parts.

1. ***IR Proximity sensor***

A proximity sensor is a sensor able to detect the presence of nearby objects without any physical contact. We are using 2 proximity sensors in our project it will be used to play next or previous and pause the song.



**2) *Ultrasonic Sensor***

We connect Ultrasonic Sensor as follows:

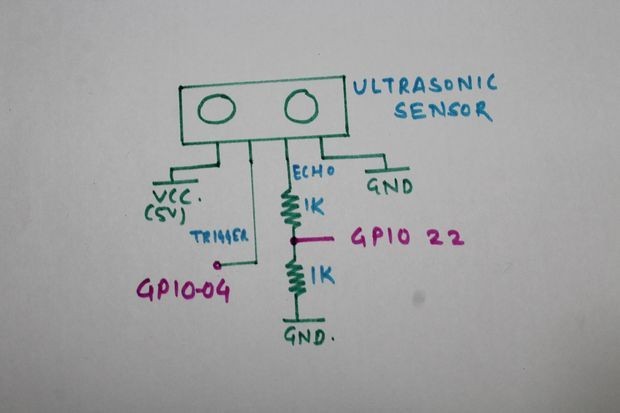
VCC -----> 5V GND -----> GND TRIGGER----> GPIO 04

ECHO -----> Voltage Divider Circuit------>GPIO 22

Echo gives output 5V but Raspberry pi needs 3.3V therefore we are using

Voltage Divider Circuit to make voltage equivalent to 3.3V.

It measures distance from obstacle and according to that we are changing volume.



**3) *Connecting LED’s***

LED's are used to Indicate the volume level. We are using 10 LED's. Each LED's Cathode is connected to common ground and Anode connected to 1K Resistor , then connected to GPIO Pins from Raspberry Pi.

**4) Coding**

The hardware is controlled by raspberry pi, and all the scripts are coded in python. The following are the libraries and software that we used to write the code:

1. Difflib
2. SpeechRecognition
3. Google Voice API
4. Omxplayer
5. Pyaudio

Code wavepad.py:

import os

import glob

import subprocess

import RPi.GPIO as GPIO

import time

from time import sleep

TRIGGER = 18

ECHO = 22

LEFT\_SENSOR=12

RIGHT\_SENSOR=16

GPIO.setmode(GPIO.BOARD)

GPIO.setup(LEFT\_SENSOR,GPIO.IN)

GPIO.setup(RIGHT\_SENSOR,GPIO.IN)

GPIO.setup(TRIGGER,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

os.chdir('/home/pi/Music')

f = glob.glob('\*mp3')

h = len(f)

status = 1

pointer = 0

start = 0

volume = 8

def distance():

GPIO.output(TRIGGER,True)

time.sleep(0.00001)

GPIO.output(TRIGGER,False)

StartTime = time.time()

StopTime = time.time()

while GPIO.input(ECHO) == 0:

StartTime = time.time()

while GPIO.input(ECHO) == 1:

StopTime = time.time()

TimeElapsed = StopTime-StartTime

distance=(TimeElapsed\*34300)/2

return distance

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

try:

for counter in range(1,10):

volume = counter

sleep(0.1)

for counter in range(1,9):

volume = 10-counter

sleep(0.1)

for counter in range(1,8):

volume = counter

sleep(0.1)

while True:

dist = int(distance())

if(status==1):

player = subprocess.Popen(["omxplayer",f[pointer]],stdin=subprocess.PIPE)

fi = player.poll()

status = 0

start = 0

volume = 8

if(dist>=0 and dist<=35):

vol = dist/3

if(vol>volume):

for counter in range(1,vol-volume):

player.stdin.write("+")

volume = volume + 1

sleep(0.1)

elif(vol<volume):

for counter in range(1,volume-vol):

player.stdin.write("-")

volume = volume - 1

sleep(0.1)

if(GPIO.input(LEFT\_SENSOR)==True and GPIO.input(RIGHT\_SENSOR)==True):

sleep(0.5)

fi = player.poll()

if fi!=0:

player.stdin.write("p")

elif(GPIO.input(LEFT\_SENSOR)==True):

for counter in range(0,10000):

if(GPIO.input(RIGHT\_SENSOR)==True):

if start==0:

player.stdin.write("q")

status = 1

pointer = pointer +1

if(pointer>h-1):

pointer = 0

break

sleep(0.0001)

elif(GPIO.input(RIGHT\_SENSOR)==True):

for counter in range(0,10000):

if(GPIO.input(LEFT\_SENSOR)==True):

if(start==0):

player.stdin.write("q")

status = 1

pointer = pointer - 1

if(pointer<0):

pointer = h-1

break

sleep(0.0001)

else:

fi = player.poll()

if(fi==0 and start==0):

status = 1

pointer = pointer +1

if(pointer>h-1):

pointer = 0

sleep(0.1)

except KeyboardInterrupt:

print("Stopped by user")

GPIO.cleanup()

For voice extension:

voice\_text='snake if you'

os.chdir('/home/pi/Music')

f = glob.glob('\*mp3')

h = len(f)

status=1

pointer=0

start = 1

max\_match=float('-inf')

r = sr.Recognizer()

next='next.mp3'

previous='previous.mp3'

pause='pause.mp3'

stop='stop.mp3'

flag=0

while True:

if (flag==0):

with sr.Microphone() as source:

print ('Say Something!')

audio = r.listen(source)

print ('Done!')

try:

matched\_song=''

voice\_text=r.recognize\_google(audio)+".mp3"

for index in range(0,h):

seq=difflib.SequenceMatcher(None,f[index],voice\_text)

d=seq.ratio()\*100

if(d>max\_match):

max\_match=d

matched\_song=f[index]

pointer=index

print voice\_text

print matched\_song

print max\_match

except Exception as e:

print (e)

if(status==1):

player = subprocess.Popen(["omxplayer",f[pointer]],stdin=subprocess.PIPE)

status=0

fi = player.poll()

start = 0

volume = 8

sleep(0.1)

if(voice\_text=='increase volume.mp3'):

player.stdin.write("+")

volume = volume + 1

voice\_text=''

sleep(0.1)

elif(voice\_text=='decrease volume.mp3'):

player.stdin.write("-")

volume = volume - 1

voice\_text=''

sleep(0.1)

elif(voice\_text==next and flag==0) :

if start==0:

player.stdin.write("q")

status = 1

pointer = pointer +1

if(pointer>h-1):

pointer = 0

flag=1

elif(voice\_text==next and flag==1):

flag=0

elif(voice\_text==previous and flag==0):

if(start==0):

player.stdin.write("q")

status = 1

pointer = pointer - 1

if(pointer<0):

pointer = h-1

flag=1

continue

elif(voice\_text==previous and flag==1):

flag=0

elif(voice\_text==pause or voice\_text==stop):

sleep(0.5)

fi = player.poll()

if fi!=0:

player.stdin.write("p")

else:

fi = player.poll()

if(fi==0 and start==0):

status = 1

pointer = pointer +1

if(pointer>h-1):

pointer = 0

sleep(0.1)

Working:

For gestures, we have the ultrasonic sensors and the IR sensors which detect proximity, the inputs from these sensors is given to the GPiO pins of Pi. We are continuously scanning for any activity in front of the sensors, and whenever we find an activity, we take the desired action in our python code. For voice, we have a mic which is connected to pi, and the mic continuously asks for input, when the input is not null, we take some action depending on the command. We have used Google voice API (the one present in almost all the smartphones) which requires internet while we are operating the music player (as in our smartphones). Google API returns us the text, and then we can wish to do any processing we want, once we have got the text command.

**Work Distribution:**

Work was evenly distributed among all of us.