**PORTABLE SELF ASSESSMENT AUDIOMETER USING RASPBERRY PI**

**SOURCE CODE**

**PYTHON CODE**

**import argparse**

**from datetime import** datetime, timedelta

**from time import** sleep

**import numpy as np**

**import pyaudio**

**import threading**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**from matplotlib.backends.backend\_tkagg import** FigureCanvasTkAgg

**from pynput.mouse import** Listener, Button

**import tkinter as tk**

# Use interactive backend for displaying plots in a separate window

plt.switch\_backend('TkAgg') # You may need to install TkAgg backend if not already installed

**class HearingTest**:

**def \_\_init\_\_**(self):

self.signal = None

self.right\_data = []

self.detected = False

self.start\_time = None

**def display\_instructions**(self):

"""Display instructions in a new window"""

instructions\_window = tk.Tk()

instructions\_window.title("Instructions")

instructions\_label = tk.Label(instructions\_window, text="Portable Self Assessment Audiometer", font=("Arial", **16**, "bold"))

instructions\_label.pack(padx=**10**, pady=**10**)

instructions\_text = "1. Put on your headphones.**\n\n**2. Click the left mouse button when you hear pulsing sounds.**\n\n**3. The test will run for the right ear, playing sounds at different frequencies and volumes.

instructions\_text\_label = tk.Label(instructions\_window, text=instructions\_text, font=("Arial", **12**), justify=tk.LEFT)

instructions\_text\_label.pack(padx=**10**, pady=**10**)

start\_button = tk.Button(instructions\_window, text="Start Test", command=self.start\_test)

start\_button.pack(padx=**10**, pady=**10**)

instructions\_window.mainloop()

**def start\_test**(self):

"""Start the hearing test"""

self.display\_instructions()

self.run\_test()

**def player**(self, p, repeat=**1**, ear='right'):

"""Plays sounds with different frequencies and volume levels"""

volumes = [**0**, **10**, **20**, **30**, **40**, **50**, **60**, **70**, **80**, **90**] # Adjusted volume levels in dB

frequencies = [**125**, **250**, **500**, **1000**, **2000**, **4000**, **8000**] # Adjusted frequencies in Hz

# Repeat each frequency based on the provided argument

frequencies = np.repeat(frequencies, repeat)

stream = p.open(format=pyaudio.paFloat32,

channels=**1**,

rate=**44100**,

output=True)

sleep(**0.1**)

**for** freq **in** frequencies:

self.detected = False

**for** vol **in** volumes:

**print**(f"Playing frequency: {freq} Hz at volume: {vol} dB for {ear} ear")

self.signal = [freq, vol, datetime.now()]

audio\_data = (np.sin(**2** \* np.pi \* np.arange(**44100** \* **0.5**) \* freq / **44100**)).astype(np.float32)

audio\_data = audio\_data \* **10**\*\*(vol / **20**)

stream.write(audio\_data.tobytes())

sleep(**2**) # Adding 2-second pause after playing each volume level

**if** self.detected:

**break**

sleep(**2**) # Adding 2-second pause after playing each frequency

stream.stop\_stream()

stream.close()

**def on\_click**(self, x, y, button, pressed):

"""Callback function for mouse clicks"""

**if** button == Button.left **and** pressed:

**if** self.signal:

d = self.signal + [datetime.now()]

**print**(f'Recording event: {d}')

self.right\_data.append(d)

self.detected = True

**def listener**(self):

"""Listens to mouse clicks"""

**with** Listener(on\_click=self.on\_click) **as** listener:

listener.join()

**def analyse\_results**(self, data, ear):

"""Stores and visualizes results"""

now = datetime.now()

# Load data to DataFrame

df = pd.DataFrame(data, columns=['frequency', 'volume', 'played', 'heard'])

df['reaction\_time'] = (df['heard'] - df['played']).dt.microseconds // **1000**

# Create audiogram chart

audiogram\_fig = plt.figure()

ax1 = audiogram\_fig.add\_subplot(**111**)

ax1.plot(df['frequency'], df['volume'], marker='x', linestyle='-', color='black')

ax1.set(title=f"Audiogram for {ear} ear", ylim=[**90**, -**10**], yticks=[**90**, **80**, **70**, **60**, **50**, **40**, **30**, **20**, **10**, **0**, -**10**])

ax1.grid(True)

ax1.set\_ylabel('Hearing Level in decibels (volume in dB)')

# Add x-axis ticks and labels at the top of the chart

ax2 = ax1.twiny()

ax2.set\_xlim(ax1.get\_xlim())

ax2.set\_xticks(df['frequency'])

ax2.set\_xticklabels(df['frequency'])

ax2.set\_xlabel('Pitch (frequency in Hz)')

ax2.xaxis.tick\_top()

# Add colored rows for different hearing loss stages

ax1.axhspan(-**10**, **15**, facecolor='green', alpha=**0.3**, label='Normal Hearing (0-15 dB)')

ax1.axhspan(**16**, **25**, facecolor='yellow', alpha=**0.3**, label='Slight Hearing Loss (16-25 dB)')

ax1.axhspan(**26**, **40**, facecolor='orange', alpha=**0.3**, label='Mild Hearing Loss (26-40 dB)')

ax1.axhspan(**41**, **55**, facecolor='red', alpha=**0.3**, label='Moderate Hearing Loss (41-55 dB)')

ax1.axhspan(**56**, **70**, facecolor='purple', alpha=**0.3**, label='Moderately Severe Hearing Loss (56-70 dB)')

ax1.axhspan(**71**, **90**, facecolor='brown', alpha=**0.3**, label='Severe Hearing Loss (71-90 dB)')

ax1.axhspan(**91**, **120**, facecolor='black', alpha=**0.3**, label='Profound Hearing Loss (91 dB or greater)')ax1.legend()

# Save audiogram chart as image

audiogram\_fig.savefig(f'./results\_{ear}\_{now:%Y%m%d%H%M%S}\_audiogram.png')

# Display audiogram chart in a new window

audiogram\_window = tk.Tk()

audiogram\_window.title(f"Audiogram for {ear} ear")

canvas = FigureCanvasTkAgg(audiogram\_fig, master=audiogram\_window)

canvas.draw()

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=**1**)

# Display Excel table in a new window

excel\_window = tk.Tk()

excel\_window.title(f"Portable Self Assessment Audiometer - {ear} ear")

excel\_table = tk.Frame(excel\_window)

excel\_table.grid(row=**0**, column=**0**, padx=**10**, pady=**10**)

table\_label = tk.Label(excel\_table, text="Portable Self Assessment Audiometer", font=("Arial", **16**, "bold"))

table\_label.grid(row=**0**, columnspan=**5**, sticky="w")

headers = ['Sl. No.', 'Pitch (Frequency Hz)', 'Hearing Level (Volume dB)', 'Hearing Loss Range']

**for** i, header **in** enumerate(headers):

col\_label = tk.Label(excel\_table, text=header, font=("Arial", **12**, "bold"))

col\_label.grid(row=**1**, column=i, padx=**5**, pady=**5**)

**for** i, row **in** df.iterrows():

sl\_no = tk.Label(excel\_table, text=i+**1**, font=("Arial", **12**))

sl\_no.grid(row=i + **2**, column=**0**, padx=**5**, pady=**5**, sticky="w")

freq\_label = tk.Label(excel\_table, text=row['frequency'], font=("Arial", **12**))

freq\_label.grid(row=i + **2**, column=**1**, padx=**5**, pady=**5**)

vol\_label = tk.Label(excel\_table, text=row['volume'], font=("Arial", **12**))

vol\_label.grid(row=i + **2**, column=**2**, padx=**5**, pady=**5**)

range\_label = tk.Label(excel\_table, text=self.get\_hearing\_loss\_range(row['volume']), font=("Arial", **12**))

range\_label.grid(row=i + **2**, column=**3**, padx=**5**, pady=**5**)

excel\_window.mainloop()

# Create CSV file

df.to\_csv(f'./results\_{ear}\_{now:%Y%m%d%H%M%S}.csv', index=None)

# Create Excel sheet

df\_excel = pd.DataFrame(data, columns=['Sl. No.', 'Pitch (Frequency Hz)', 'Hearing Level (Volume dB)', 'Hearing Loss Range'])

df\_excel['Hearing Loss Range'] = df\_excel['Hearing Level (Volume dB)'].apply(self.get\_hearing\_loss\_range)

df\_excel.to\_excel(f'./results\_{ear}\_{now:%Y%m%d%H%M%S}.xlsx', index=None)

**print**("Audiogram chart, CSV file, and Excel sheet created successfully.")

**return** df

**def get\_hearing\_loss\_range**(self, volume):

"""Determines the hearing loss range based on volume level"""

**if** volume <= **15**:

**return** 'Normal Hearing (0-15 dB)'

**elif** volume <= **25**:

**return** 'Slight Hearing Loss (16-25 dB)'

**elif** volume <= **40**:

**return** 'Mild Hearing Loss (26-40 dB)'

**elif** volume <= **55**:

**return** 'Moderate Hearing Loss (41-55 dB)'

**elif** volume <= **70**:

**return** 'Moderately Severe Hearing Loss (56-70 dB)'

**elif** volume <= **90**:

**return** 'Severe Hearing Loss (71-90 dB)'

**else**:

**return** 'Profound Hearing Loss (91 dB or greater)'

**def run\_test**(self):

parser = argparse.ArgumentParser()

parser.add\_argument('-r', '--repeat', help='Number of times each frequency is repeated', type=int, default=**1**) # Change default value to 1

args = parser.parse\_args()

self.start\_time = datetime.now()

p = pyaudio.PyAudio()

# Start listener

p2 = threading.Thread(target=self.listener, daemon=True)

p2.start()

# Run test for the right ear

**print**('Testing right ear...')

self.player(p, repeat=args.repeat, ear='right')

# Analyse and visualize results for the right ear

right\_df = self.analyse\_results(self.right\_data, 'right')

self.right\_data = []

**print**('Test is finished. Please check visualizations and files.')

self.display\_date\_time\_duration()

**def display\_date\_time\_duration**(self):

now = datetime.now()

duration = now - self.start\_time

# Display test information in a new window

info\_window = tk.Tk()

info\_window.title("Test Information")

date\_label = tk.Label(info\_window, text=f"Date: {self.start\_time.strftime('%Y-%m-%d')}", font=("Arial", **12**))

date\_label.pack()

start\_time\_label = tk.Label(info\_window, text=f"Start Time: {self.start\_time.strftime('%H:%M:%S')}", font=("Arial", **12**))

start\_time\_label.pack()

duration\_label = tk.Label(info\_window, text=f"Duration: {duration}", font=("Arial", **12**))

duration\_label.pack()

info\_window.mainloop()

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

test = HearingTest()

test.start\_test()