A PROJECT REPORT
on
Build an Audio Recorder

Submitted to

A project submitted to **Central university of South Bihar** in partial fulfillment of the requirement for the award of Degree of **Master of Science.**

Submitted by

A. BRAHAMIAH (CUSB2202312003)

R. CHANDRASEKHAR(CUSB2202312024)

GOPICHAND (CUSB2202312009)

Under the Guidance of

Dr. Nemi Chandra Rathore

DEPARTMENT OF COMPUTER SCIENCE

Central University of South Bihar

SH-7, Gaya Panchanpur Road – Village – Karhara, Post, Fatehpur, Bihar 824236.

CERTIFICATE

This is to certify that the project titled **"Build an Audio Recorder"** is a Bonafede work done by **A. BRAHMAIAH, R. CHANRASEKHAR, GOPICHAND** under my guidance and supervision in partial fulfillment of the requirement for the award of degree of **Master of Computer Science** in Central university of South Bihar **Gaya**, during the academic year **2022-2024**.

Project Guide
Prof Dr. Nemi Chandra Rathore

Head of the Department
Prof Dr. Prabhat Rajan

DECLARATION

I hereby declare that this project titled "**Build an Audio Recorder**"**.** Is a bonafide work done by me under the guidance of **Prof Dr. Nemi Chandra Rathore** This project work is submitted to **Central University of South Bihar** Gaya, **in** partial fulfillment of the requirements for the award of **Master of Computer science** during the academic year **2022-2024**

I also declare that this project is a result of my own effort and that it has not been submitted to any other University for the award of any degree.

Date:_____

Place: CUSB

ACKNOWLEDGEMENT

It is great pleasure to take opportunity and express my gratitude to all those who helped me throughout my project work.

I also thank **Prof Dr. Nemi Chandra Rathore, Project Guide** and **Head of the Department of Computer Science , Central University of Bihar** for giving me opportunity to take upthe project work and helping me through out and finally completing theproject.

I would like to express my sincere and heartfelt thanks to all the teaching and non-teaching staff of the department for their continuous co-operation, which has given me the guidance to build up adamant aspiration over the completion of my project.

I extend my heartfelt thanks to my parents and friends for their constant support and cooperation in completing this project successfully.

Finally I thank one and all who directly and indirectly helped me to complete my project successfully.

Contents pg.no

1.Introduction

Python can be used to perform a variety of tasks. One of them is creating a audio recorder using python. We can use various types of libraries in python's sounddevice module to record and play audio. This module along with the wavio or the scipy module provides a way to save recorded audio.

2.Prerequisites

Before we start, make sure you have the following prerequisites:

Python Installed: If Python is not already installed, you can download it from the official website in python advanced version.

Required Libraries: We'll be using the sounddevice and tkiner,soundfile,threading and queue,libraries for audio recording and manipulation. You can install and import them using the following commands

- **import tkinter** : It is already available in python windows
- sudo apt-get install python3-tkinter, for ubuntu users
- sudo pacman -S tk, for arch linux users.

3.Build the audio recorder in step by step explanation

Step-1.Importing libraries
step-2. Defining the functions for threading, recording audio and providing a callback
step-3. Creating the audio recorder GUI
step-4. Create the components and the button

➢ step-1 Importing libraries

```
import sounddevice as sd
from tkinter import *
import queue
import soundfile as sf
import threading
from tkinter import messagebox
```

Code Explanation

- **Import sounddevice as sd:** Sounddevice contains the functions to record audio input using the microphone.
- **from tkinter import *:** Tkinter is the GUI library we use to create the GUI of the application.
- **import queue:** To use the data structure queue, we import it.
- **import soundfile as sf:** To save the audio recorded, we use soundfile.
- **from tkinter import messagebox:** Messagebox contains prompts to display to the user when there is a missing input or when the file does not exist.

- **import threading:** To simulate multithreading.

✓ Step -2 Defining the functions for threading, recording audio and providing a callback:

There are three functions in use.

1. threading_rec()
2. callback()
3. record_audio()

We will see each of the functions and its purpose in detail.

1.threading_rec():

Threading allows parallel execution of various processes or tasks at once. The main use of threads here is to keep the other buttons active even if a function is running because of another button click.

```
def threading_rec(x):
if x == 1:
#If recording is selected, then the thread is activated
t1=threading.Thread(target= record_audio)
t1.start()
elif x == 2:
#To stop, set the flag to false
global recording
recording = False
messagebox.showinfo(message="Recording finished")
elif x == 3:
#To play a recording, it must exist.
if file_exists:
#Read the recording if it exists and play it
data, fs = sf.read("Cusb.wav", dtype='float32')
sd.play(data,fs)
sd.wait()
else:
#Display and error if none is found
messagebox.showerror(message="Record something to play")
```

Code explanation:

- **def threading_rec(x):** Declaration of the function to initiate threads to record and stop. It contains a parameter, x.
- **if x == 1:** If x == 1, start recording the audio by creating a thread and allowing the thread to call the function. threading.Thread() creates a thread and target calls the function it is assigned with. start() starts the execution of the thread.
- **elif x == 2:** If x==2, then it indicates to stop the audio recording. It is done by reassigning a global variable to False. This global variable is a test condition in the recording function, record_audio. Setting it to False terminates the while loop which records this. Then notify the user with a prompt, messagebox.showinfo, to denote the completion of the audio recording.
- **x == 3:** To play the audio, it must be recorded. Hence we check if file_exists is True indicating a recording previously done by the user and it is played using sf.read() which accepts an input file and a

datatype to read it from. ‘float32’ is the default datatype. To play the audio, use play() where you play only the data and not the sample rate, fs. Use wait to create a delay to play the audio. The audio will not be played otherwise. If file_exists does not exist, raise a prompt using messagebox.showerror with a warning message.

2.callback():

```
def callback(indata, frames, time, status):
q.put(indata.copy())
```

Code explanation:

- **def callback(indata, frames, time, status):** Declare a function callback to record data into the queue. This method is necessary and exists in the documentation of sounddevice.
- **q.put(indata.copy()):** Queue is a data structure that follows FIFO. It means the first element to get into the queue is the first to exit the queue. A queue contains two methods: pop/get and push/put. Pop/get extracts the first content in the queue. Push/put puts the data into the queue.

3.record_audio():

```
def record_audio():
    #Declare global variables
    global recording
    #Set to True to record
    recording= True
    global file_exists
    #Create a file to save the audio
    messagebox.showinfo(message="Recording Audio. Speak into the mic")
    with sf.SoundFile("Cusb.wav", mode='w', samplerate=44100, channels=2) as file:
    #Create an input stream to record audio without a preset time
     with sd.InputStream(samplerate=44100, channels=2, callback=callback):
     while recording == True:
     #Set the variable to True to allow playing the audio later
      file_exists =True
     #write into file
     file.write(q.get())
```

Code explanation:

- **def record_audio():** Declaration of the recording function
- **global recording:** Declaration of a global variable. This global variable controls the audio recording loop

- **recording= True:** Setting record to True to initiate recording
- **global file_exists:** Declare a global variable to indicate the presence of a recording. A variable when made global, is accessible by all functions in the code irrespective of where it is declared.
- **messagebox.showinfo(message="Recording Audio. Speak into the mic"):** Display a prompt to the user to record the audio
- **sf.SoundFile():** Create a file object to save the audio recording. The parameters are: 1. Name of the file with .wav extension, 2. Mode which denotes the mode to write the file, 3. samplerate which is the frames per second, 4. Channels denoting the number of input and output channels.
- **sd.InputStream():** This function records the audio input without a preset duration. The additional parameter here is the callback which contains the array to put the input data
- while recording == True: Recording will happen only if it is set to true.
- **file_exists =True:** Marking the existence of a file if recoding happens
- **file.write(q.get()):** Writing the contents of the queue into the soundfile created, by extracting the queue contents using get()

✓ Step-3. Creating the audio recorder GUI:

```
#Define the user interface of the record_audio
audio_rec = Tk()
audio_rec.geometry("360x200")
audio_rec.title("Cusb Computer Science python mini project in AudioRecoder ")
audio_rec.config(bg="#107dc2")
#Create a queue to contain the audio data
q = queue.Queue()
#Declare variables and initialise them
recording = False
file_exists = False
```

Code explanation:

- **audio_recorder = Tk():** Creation of an object to use the widgets in Tkinter and define the window of the app
- **audio_rec.geometry("360×200"):** Define the dimensions of the window using geometry. It is of the format 'width x height'
- **audio_rec.title("Cusb Computer Science python mini project in AudioRecoder "):** Assign a title for the application
- **audio_rec.config(bg="#107dc2″):** To use a background colour, use config with the parameter bg to assign a colour. The colour can be given using the name or it can be specified using the colour code.
- **q = queue.Queue():** Initialize the queue to contain the audio data
- **recording = False, file_exists = False:** Declare and define the variable to False

✓ Step-4. Create the components and the button:

```
#Label to display app title

title_lbl  = Label(audio_rec, text="Cusb Computer Science python mini project in Audio Recoder",
bg="#107dc2").grid(row=0, column=0, columnspan=3)

#Button to record audio

record_btn = Button(audio_rec, text="Record Audio", command=lambda m=1:threading_rec(m))

#Stop button

stop_btn = Button(audio_rec, text="Stop Recording", command=lambda m=2:threading_rec(m))

#Play button

play_btn = Button(audio_rec, text="Play Recording", command=lambda m=3:threading_rec(m))

#Position buttons

record_btn.grid(row=1,column=1)

stop_btn.grid(row=1,column=0)

play_btn.grid(row=1,column=2)

audio_rec.mainloop()
```

Code explanation:

- **title_lbl:** To display non-editable text which cannot be copied, we use Label. It contains the window of the application and the text to be displayed. Add an optional 'bg' equivalent to the window's background colour or ignore the tag to allow default value. Positioning a widget makes it visible on the app. Hence we use grid, which divides the window into rows and columns. To position the label, we use row 0, column 0 and set a columnspan of 3 to make it appear across the other columns.

- **record_btn, stop_btn, play_btn:** To activate the record function, stop or play the audio, we use buttons. Create the buttons using Button() widget and specify the parameters such as main window of the app, text of the button and the function the button must activate. Here we use only one function for all the buttons, threading_rec. Thus we use lambda to pass different parameters, x values, to the threading_rec function.

- **grid(row=X,column=X):Place** the buttons using grid function. Specify the row and column to place the widget

- **audio_rec.mainloop():** To activate the window and execute the application mainloop is used. When the user exits the application, the control flows to after mainloop thus terminating the app

4.Programming of Audio recorder

```
# Audio Recording project on Python
#Import necessary modules
import sounddevice as sd
from tkinter import *
import queue
import soundfile as sf
import threading
from tkinter import messagebox
#Define the user interface of the record_audio
audio_rec = Tk()
audio_rec.geometry("360x200")
audio_rec.title("Cusb Computer Science python mini project in Audio Recoder ")
audio_rec.config(bg="#107dc2")
#Create a queue to contain the audio data
q = queue.Queue()
#Declare variables and initialise them
recording = False
file_exists = False
#Fit data into queue
def callback(indata, frames, time, status):
    q.put(indata.copy())
#Functions to play, stop and record audio
#The recording is done as a thread to prevent it being the main process
def threading_rec(x):
    if x == 1:
        #If recording is selected, then the thread is activated
        t1=threading.Thread(target= record_audio)
        t1.start()
    elif x == 2:
        #To stop, set the flag to false
```

```
        global recording
        recording = False
        messagebox.showinfo(message="Recording finished")
    elif x == 3:
        #To play a recording, it must exist.
        if file_exists:
            #Read the recording if it exists and play it
            data, fs = sf.read("Cusb.wav", dtype='float32')
            sd.play(data,fs)
            sd.wait()
        else:
            #Display and error if none is found
            messagebox.showerror(message="Record something to play")
#Recording function
def record_audio():
    #Declare global variables
    global recording
    #Set to True to record
    recording= True
    global file_exists
    #Create a file to save the audio
    messagebox.showinfo(message="Recording Audio. Speak into the mic")
    with sf.SoundFile("Cusb.wav", mode='w', samplerate=44100,
                      channels=2) as file:
    #Create an input stream to record audio without a preset time
            with sd.InputStream(samplerate=44100, channels=2, callback=callback):
                while recording == True:
                    #Set the variable to True to allow playing the audio later
                    file_exists =True
                    #write into file
                    file.write(q.get())
```

#Label to display app title

title_lbl = Label(audio_rec, text="Cusb Computer Science python mini project in Audio Recoder", bg="#107dc2").grid(row=0, column=0, columnspan=3)

#Button to record audio

record_btn = Button(audio_rec, text="Record Audio", command=lambda m=1:threading_rec(m))

#Stop button

stop_btn = Button(audio_rec, text="Stop Recording", command=lambda m=2:threading_rec(m))

#Play button

play_btn = Button(audio_rec, text="Play Recording", command=lambda m=3:threading_rec(m))

#Position buttons

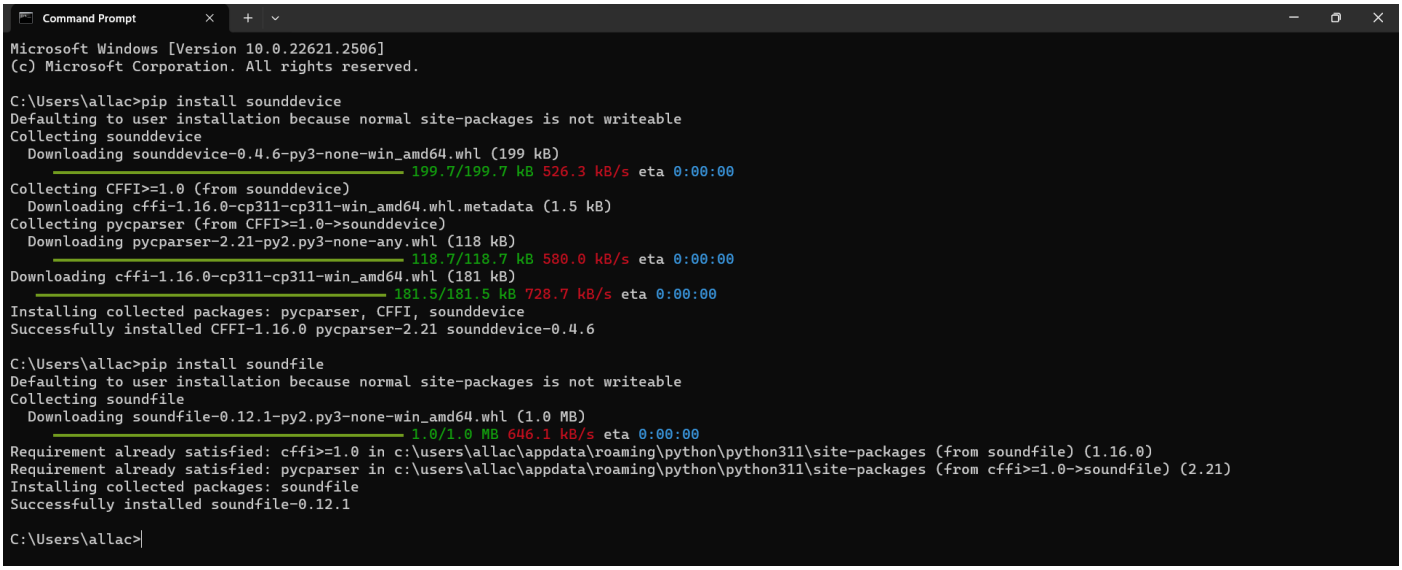
record_btn.grid(row=1,column=1)

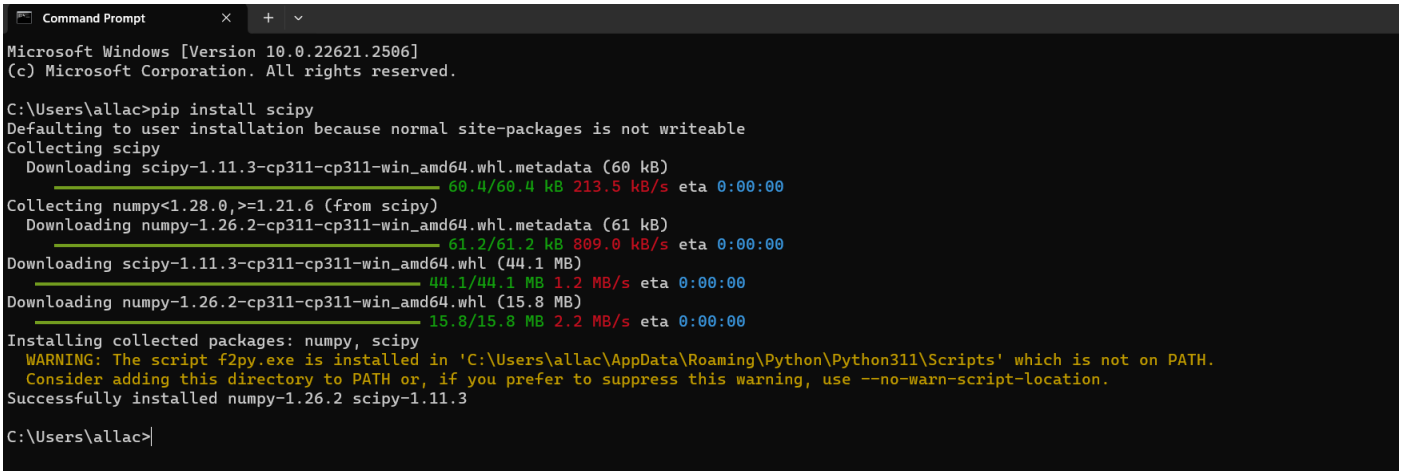
stop_btn.grid(row=1,column=0)

play_btn.grid(row=1,column=2)

audio_rec.mainloop()

5.Slides





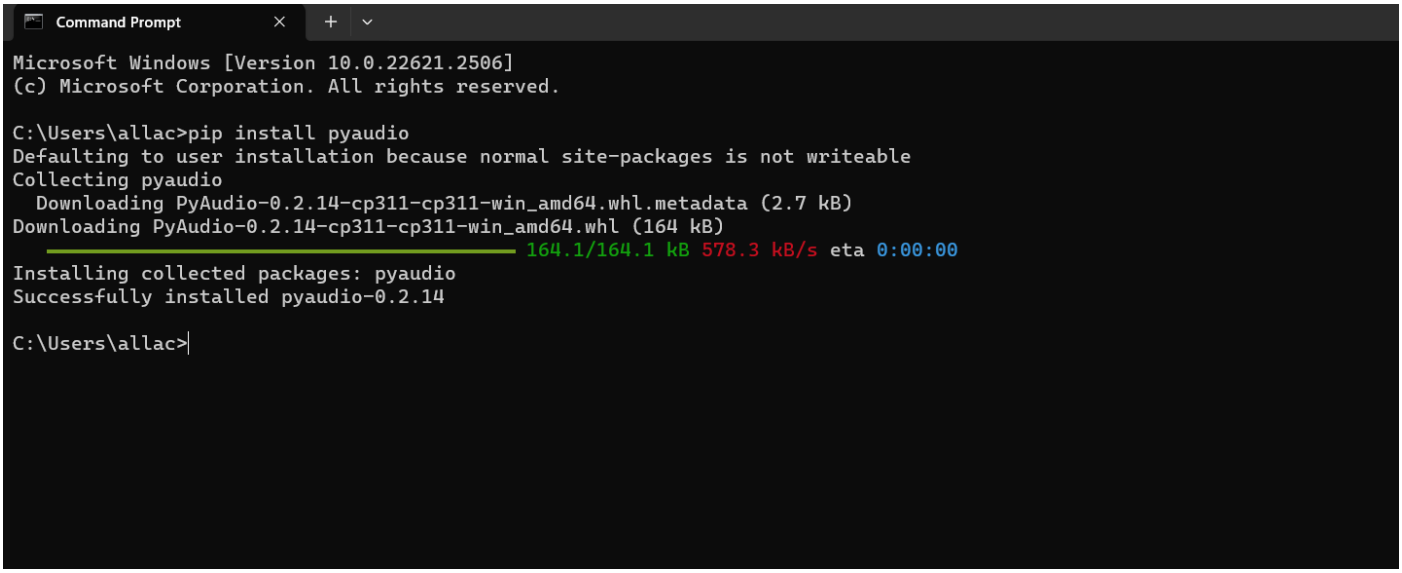


Fig-1 Install the import and libraries

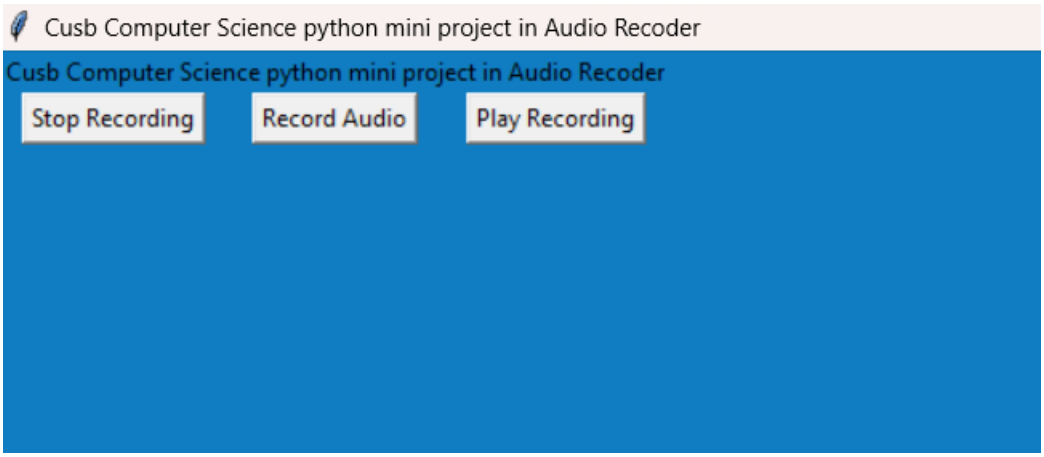


Fig-2 Output of Audio Recorder

6.Summary

Thus we created a simple Audio recorder from scratch using python. The project introduces audio libraries such as sounddevice and soundfile. We also came across a simple example of threading concept and its usage, along with a data structure.