



CSE 499A Project Report

Project Topic: Spetial Microphone

Name: Md Mantasarul Elahi

ID: 1512518 042

Section: 10

GitHub Repository Link: <https://github.com/mantasarul4982/Spetial-Microphone>

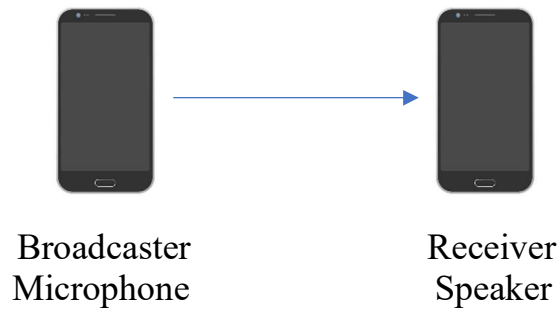
Content table

Project Description-----	Page 3
Application Layout/User Interface-----	Page 4
499A Working Progress-----	Page 6
Noise Cancellation With MATLAB-----	Page 7
Sound Recorder with Python-----	Page 10
Taking Multiple Audio Input (Microphone/s) and Plotting Graph in Real Time-----	Page 11
Graph Output-----	Page 13
Audio Visualization in CMD using Basic Python-----	Page 14
Graph Output-----	Page 16
Future Goal-----	Page 17
Conclusion-----	Page 18

Project Description

Introduction: This would be an Android based mobile application.

First Approach: At the primitive approach, there would be two cellphones. One would act as a microphone broadcaster. Another one would act as a speaker output.



Initially we will test with two android devices. When someone speaks at the Broadcast side, the application would scan via Bluetooth/hotspot technology and automatically connect to the other device which would act as a speaker output. After the first test is successfully done, we would proceed with multiple devices on the Receiver Side. In this case when someone speaks at Broadcast Microphone side the application would scan for nearby devices and automatically connect to all of them simultaneously.

Second Approach: In this stage a device would take audio with noises from nearby sources and produce better sound quality as output in the other device. Initially it would be done with two devices. Usually modern cellphones have numerous microphones. So, we can assume that a device would take sound input from its microphones and broadcast the sound with better quality audio to a nearby device which would be automatically connected when brought into the range of the broadcasting device. After this approach is successfully done, we can try with numerous android devices which would take audio input simultaneously and broadcast to another android phone. Here the number of cellphones in the receiving side may vary with project progress.

Application Layout/User Interface



In the first diagram we can see the main menu of the application. Here the user would have option of whether to use the application as a microphone, a speaker or try the beta feature. If the user selects the first option “Microphone”, then the application would move to the next diagram. Here the user would have to select how he/she would like to do the broadcast. There would be three options to choose from. The user can broadcast via the network he/she is signed in to, using Bluetooth technology or by making a hotspot network.

Now if someone is already broadcasting and another user wants to turn his/her device into a loud speaker in order to expand the broadcast range, then the user would have to select the speaker option from the first diagram. It would take the user to the third diagram. Here the interface is similar to the broadcast side. The user would have to select how he/she would like to connect with the microphone. Then in the fourth diagram there would be number of channel's list in case there are multiple microphones of multiple network. From here the user can select the appropriate one and turn his/her device into a live speaker device streaming live audio output.

In the beta category, the purpose would be the same but instead of only one microphone there would could be numerous microphones and numerous speakers. Now here is the twist. As there would be numerous devices acting as a microphone, the application would take all the audio input in real time and provide with a better-quality audio output to the speaker devices in real time.

499A Working Progress

499A was all about researching how to implement the audio processing part of the Beta version. Initial task was to choose which platform would be better to experiment with the audio processing part. Python seemed to have a lot of libraries so I chose Python in order to move on with the idea.

Most of the resources available in online about audio analysis and processing are all done in MATLAB. For audio source comparisons and graph generation MATLAB got most of the resources available. For python huge amounts of audio library are available.

Throughout the entire semester I have gathered resources about the project and my future goal is to properly utilize them all in the upcoming semester break and finish the project. All the gathered resources are given in the following.

Noise Cancellation With MATLAB

When audio inputs are taken from two audio sources, one with clear sound another one with noises, the MATLAB code would take both the audio input and filter them out to a clear audible better-quality sound.

Code Snippets:

```
1
2
3  clear;
4
5  omega=1:1:1024;
6  coefficientTitle = 'Coefficient Weights';
7
8  noise_source=dsp.AudioFileReader('mp3filename', 'OutputDataType', 'double')
9  %Enter your n'(noise) source file in .mp3 format between the quotations
10 %above. Make sure your audio file is contained in the folder where this
11 %program lives on your computer. Matlab needs the path.
12
13 signal_source=dsp.AudioFileReader('mp3filename', 'OutputDataType', 'double')
14 %Enter your d[n](desired output) source file in .mp3 format
15 %between the quotations above. Make sure your audio file is contained in
16 %the folder where this program lives on your computer. Matlab needs the
17 %path. This desired output should have some of the correlated unwanted
18 %noise.
19
20 %Create and Configure an LMS adaptive filter System Object
21 LMS=dsp.LMSFilter(64, 'Method', 'LMS', 'StepSize', .01);
22
23 %Create and Configure an audio player system object with sample rate of
24 %44100 HZ to play the audio signal.
25
26 audioout = dsp.AudioPlayer('SampleRate', 44100)
27
28 %Set up a waterfall plot that displays 8 traces of the 64 filter
29 %coefficients
30
31 plotw=plotanddata(coefficientTitle, 8, 64)
32
33 playlength = 0;
34
35 %Play Original Audio for comparison
36
37 %waitfor(msgbox('Click Ok to Play Unfiltered Audio', 'Unfiltered Audio Playback'));
38 %warndlg('Playing Unfiltered Audio');
39 button = questdlg('Would you like to play the unfiltered audio?', 'Unfiltered Audio?');
40
```

```

41  switch button
42      case 'Yes'
43
44          while playlength < 1000
45
46              [signal, eof]=step(signal_source); %Signal source read from audio file
47              noise=step(noise_source);          %Noise source
48              desired=signal(:,1)+noise(:,1);    %Add Noise to Signal
49
50              step(audioout, desired);           %Audio Out
51              %numplays=numplays+eof;
52              playlength = playlength + 1;
53          end
54
55      case 'No'
56      case 'Cancel'
57          release(noise_source);
58          release(signal_source);
59          release(audioout);
60          return;
61  end
62
63  button = questdlg('Would you like to play the filtered audio?', 'Filtered Audio?');
64  fftbutton = questdlg('Would you like to see the FFT?', 'FFT');
65
66  switch button
67      case 'Yes'
68          noise_source=dsp.AudioFileReader('VA001 ground run.mp3', 'OutputDataType', 'double')
69
70          signal_source=dsp.AudioFileReader('MaryVoice.mp3', 'OutputDataType', 'double')
71
72          playlength = 0;
73
74          while playlength < 1000
75
76              [signal, eof]=step(signal_source); %Signal source read from audio file
77              noise=step(noise_source);          %Noise source
78              desired=signal(:,1)+noise(:,1);    %Add Noise to Signal
79              desiredfft = fft(desired);         %Compute FFT for display
80              [out, err, w]=step(LMS, noise(:,1), desired); %Filter using LMS filter
81              step(audioout, double(err));        %Audio Out
82              errfft=fft(double(err));            %Compute the FFT of Error Signal
83
84              if 1
85
86                  plotw(w);                      %Plot Coeffecients
87              end
88
89              if strcmp(fftbutton, 'Yes')
90                  figure(1);                     %Plot the FFT
91                  subplot(2,1,1)
92                  plot(omega,abs(desiredfft));
93                  axis([0 512 0 100]);
94                  title('Signal Plus Noise');
95                  subplot(2,1,2);
96                  plot(omega,abs(errfft));
97                  axis([0 512 0 100]);
98                  title('Signal');
99              end
100
101              %numplays=numplays+eof;
102              playlength = playlength + 1;
103          end

```



```
104
105     case 'No'
106     case 'Cancel'
107         release(noise_source);
108         release(signal_source);
109         release(audioout);
110         return;
111     end
112
113     release(noise_source);
114     release(signal_source);
115     release(audioout);
116
117     return
```

Sound Recorder with Python

Taking audio input was something new that I have never done before. The following code snippet is about how to record sound track in python in wav format. But this requires an additional python library in order to work with it.

Code Snippets:

```
1  import pyaudio
2  import wave
3
4  FORMAT = pyaudio.paInt16
5  CHANNELS = 2
6  RATE = 44100
7  CHUNK = 1024
8  RECORD_SECONDS = 5
9  WAVE_OUTPUT_FILENAME = "file.wav"
10
11 audio = pyaudio.PyAudio()
12
13 # start Recording
14 stream = audio.open(format=FORMAT, channels=CHANNELS,
15                     rate=RATE, input=True,
16                     frames_per_buffer=CHUNK)
17 print "recording..."
18 frames = []
19
20 for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
21     data = stream.read(CHUNK)
22     frames.append(data)
23 print "finished recording"
24
25
26 # stop Recording
27 stream.stop_stream()
28 stream.close()
29 audio.terminate()
30
31 waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
32 waveFile.setnchannels(CHANNELS)
33 waveFile.setsampwidth(audio.get_sample_size(FORMAT))
34 waveFile.setframerate(RATE)
35 waveFile.writeframes(b''.join(frames))
36 waveFile.close()
```

Taking Multiple Audio Input (Microphone/s) and Plotting Graph in Real Time

The final task of 499A was to do graph comparison in order to show each audio in a same environment doesn't pick identical audio input. The following code snippet was done in python. It can plot graph in real time by taking one or multiple audio input and plot it into graph in real time. By looking at the graph we can do graph comparison which would help later on while continuing with the project work.

Code Snippets:

```
test1.py - C:\Users\Md Mantasarul Elahi\Desktop\test1.py (3.6.6)
File Edit Format Run Options Window Help

import argparse
import queue
import sys

def int_or_str(text):
    """Helper function for argument parsing."""
    try:
        return int(text)
    except ValueError:
        return text

parser = argparse.ArgumentParser(description=__doc__)
parser.add_argument(
    '-l', '--list-devices', action='store_true',
    help='show list of audio devices and exit')
parser.add_argument(
    '-d', '--device', type=int_or_str,
    help='input device (numeric ID or substring)')
parser.add_argument(
    '-w', '--window', type=float, default=200, metavar='DURATION',
    help='visible time slot (default: %(default)s ms)')
parser.add_argument(
    '-i', '--interval', type=float, default=30,
    help='minimum time between plot updates (default: %(default)s ms)')
parser.add_argument(
    '-b', '--blocksize', type=int, help='block size (in samples)')
parser.add_argument(
    '-r', '--samplerate', type=float, help='sampling rate of audio device')
parser.add_argument(
    '-n', '--downsample', type=int, default=10, metavar='N',
    help='display every Nth sample (default: %(default)s)')
parser.add_argument(
    'channels', type=int, default=[1], nargs='*', metavar='CHANNEL',
    help='input channels to plot (default: the first)')
args = parser.parse_args()
if any(c < 1 for c in args.channels):
    parser.error('argument CHANNEL: must be >= 1')
mapping = [c - 1 for c in args.channels] # Channel numbers start with 1
q = queue.Queue()

def audio_callback(indata, frames, time, status):
    """This is called (from a separate thread) for each audio block."""
    if status:
        print(status, file=sys.stderr)
    # Fancy indexing with mapping creates a (necessary!) copy:
    q.put(indata[:,args.downsample, mapping])

def update_plot(frame):
    """This is called by matplotlib for each plot update.

    Typically, audio callbacks happen more frequently than plot updates,
    therefore the queue tends to contain multiple blocks of audio data.

    """
```

```

global plotdata
while True:
    try:
        data = q.get_nowait()
    except queue.Empty:
        break
    shift = len(data)
    plotdata = np.roll(plotdata, -shift, axis=0)
    plotdata[-shift:, :] = data
    for column, line in enumerate(lines):
        line.set_ydata(plotdata[:, column])
    return lines

try:
    from matplotlib.animation import FuncAnimation
    import matplotlib.pyplot as plt
    import numpy as np
    import sounddevice as sd

    if args.list_devices:
        print(sd.query_devices())
        parser.exit(0)
    if args.samplerate is None:
        device_info = sd.query_devices(args.device, 'input')
        args.samplerate = device_info['default_samplerate']

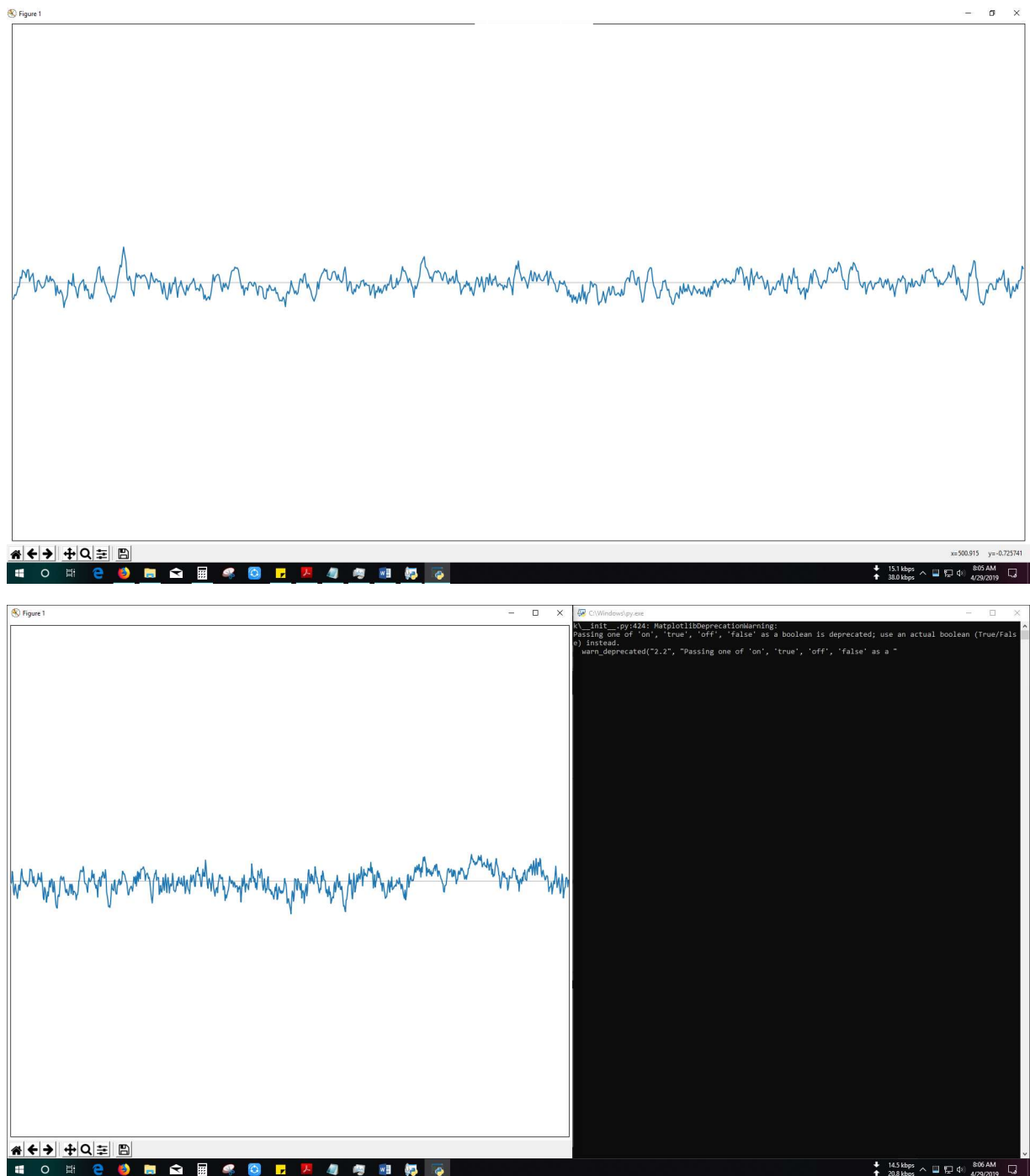
    length = int(args.window * args.samplerate / (1000 * args.downsample))
    plotdata = np.zeros((length, len(args.channels)))

    fig, ax = plt.subplots()
    lines = ax.plot(plotdata)
    if len(args.channels) > 1:
        ax.legend(['channel {}'.format(c) for c in args.channels],
                  loc='lower left', ncol=len(args.channels))
    ax.axis((0, len(plotdata), -1, 1))
    ax.set_yticks([0])
    ax.yaxis.grid(True)
    ax.tick_params(bottom='off', top='off', labelbottom='off',
                  right='off', left='off', labelleft='off')
    fig.tight_layout(pad=0)

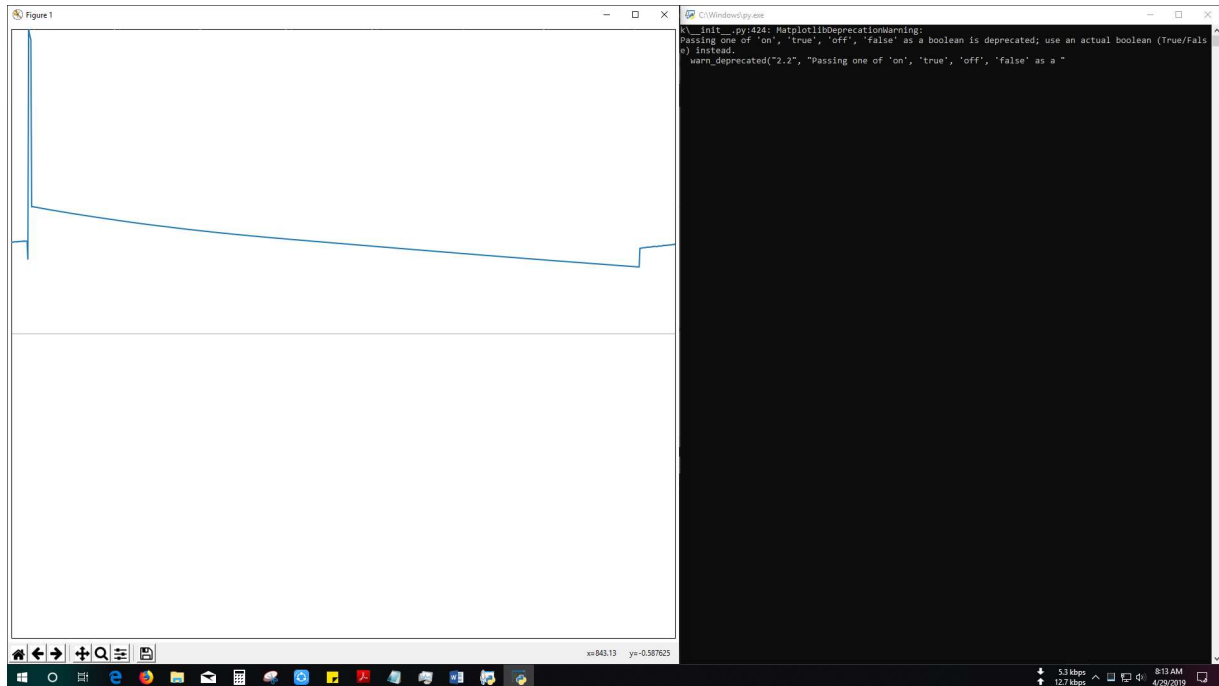
    stream = sd.InputStream(
        device=args.device, channels=max(args.channels),
        samplerate=args.samplerate, callback=audio_callback)
    ani = FuncAnimation(fig, update_plot, interval=args.interval, blit=True)
    with stream:
        plt.show()
except Exception as e:
    parser.exit(type(e).__name__ + ': ' + str(e))

```

Graph Output



P.S. I didn't have additional microphone so initially can't show more than one output at a time.



This diagram shows the output moment after the microphone was removed from the microphone port. The graph froze there and no change was seen.

Important Observation:

I did the execution of the program in my desktop. For desktop computers there are separate port for microphone and headphone. As I have used a mobile headphone which had a built-in microphone, I had to use a one input two output converter in order to use it in my desktop computer. When I have pulled off the headphone cord from the converter while the converter was still connected to my desktop, change in the frequency of the graph could be seen. If the converter is moved using hand (it was still connected to the pc), fluctuation in the graph could be seen. Later on when I finally pulled off the converter, the graph froze still with no change to the audio frequency.

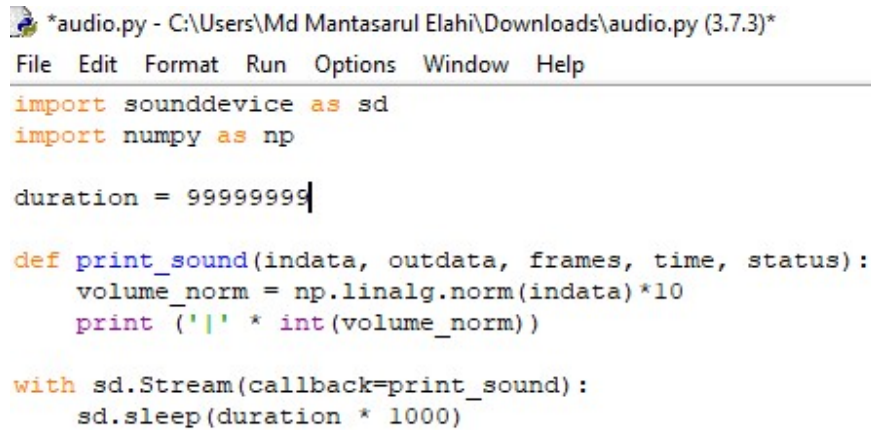
Required Python Library List:

- Matplotlib
- Numpy
- Sound Device

Audio Visualization in CMD using Basic Python

Using simple Python code audio visualization could be done in CMD. For this python library sound device needs to be installed.

Code Snippets:

A screenshot of a text editor window titled '*audio.py - C:\Users\Md Mantasarul Elahi\Downloads\audio.py (3.7.3)*'. The editor has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code is as follows:

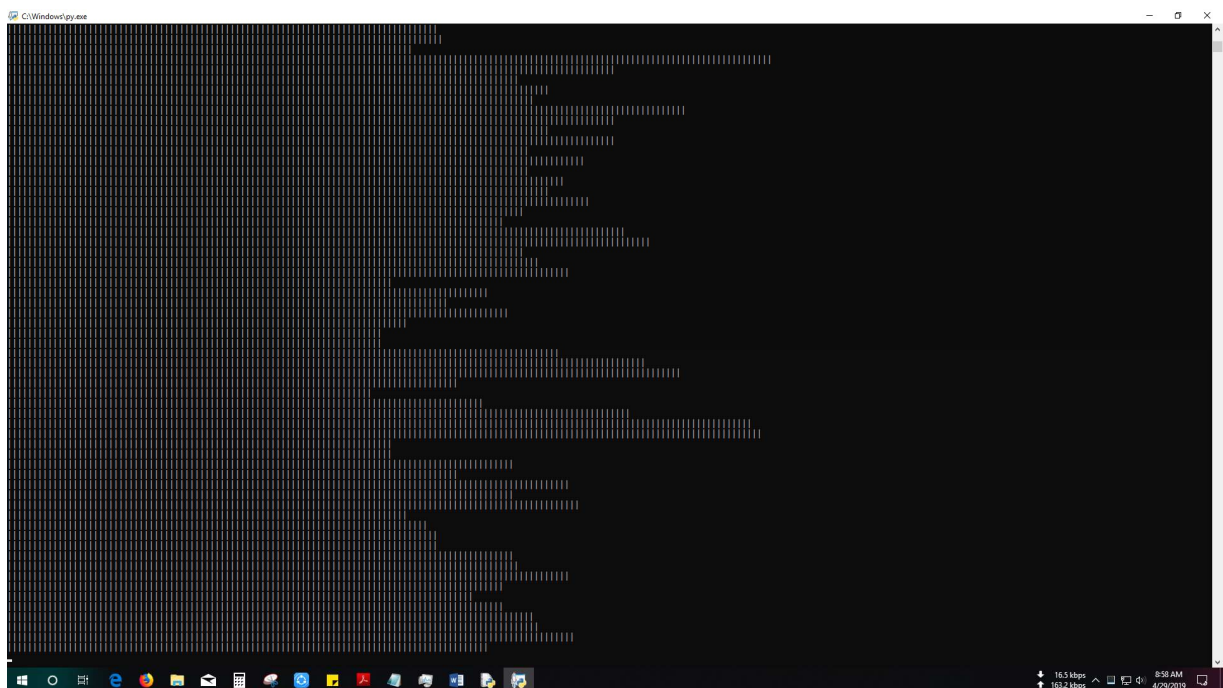
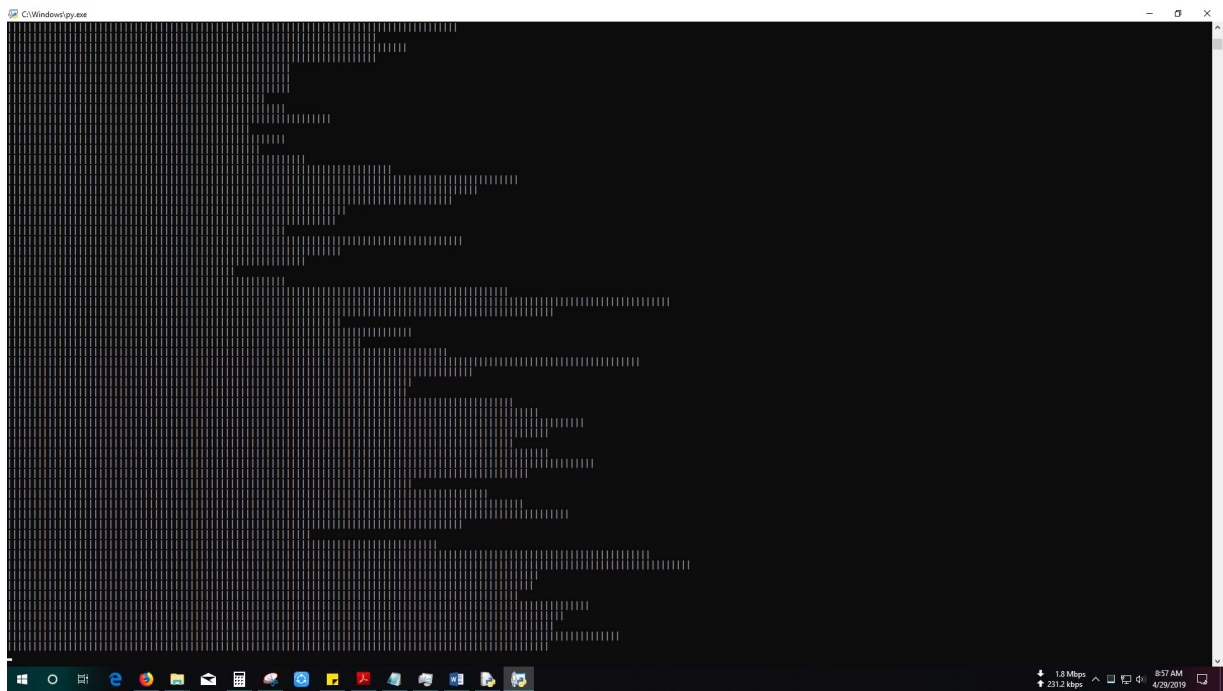
```
import sounddevice as sd
import numpy as np

duration = 99999999

def print_sound(indata, outdata, frames, time, status):
    volume_norm = np.linalg.norm(indata)*10
    print ('|' * int(volume_norm))

with sd.Stream(callback=print_sound):
    sd.sleep(duration * 1000)
```

Graph Output



Future Goal

- Properly use all these resources and modify them when needed to get better expected result and carry out the research.
- Start building the Android application in order to use this research in the actual project

Conclusion

Special thanks to Mohammad Ashrafuzzaman Khan for selecting my project idea which I never expected would get this much priority and for enlightening me in this interesting field of work and for all the opportunities.

The End

