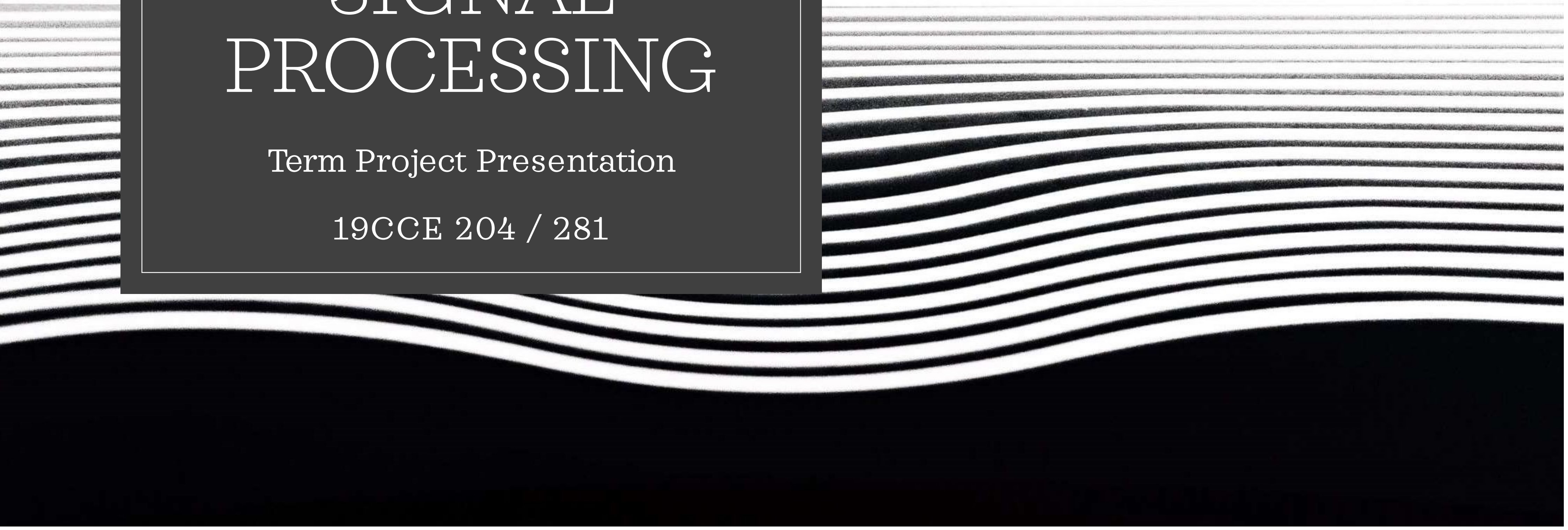


SIGNAL PROCESSING

Term Project Presentation

19CCE 204 / 281

Group Members :

M V S Krishna Pranav – CB.EN.U4CCE20030

Malavika Menon T - CB.EN.U4CCE20031

Manoj Parthiban - CB.EN.U4CCE20032

Narendran S - CB.EN.U4CCE20036

AUDIO SPECTRUM ANALYSER

Using Python

AIM OF THIS PROJECT

- This project focuses on developing an Audio spectrum analyzer using Python taking the user’s microphone as input. The spectrum analyzer measures the amplitude, magnitude of the input signal. With respect to the frequency. It is mainly used to analyze the amplitude of different frequencies of signals. The analyzer is able to sample the incoming spectrum in the time domain and convert the sampled information into frequency domain.
- In this project the algorithm that we did will be used to convert the time domain signal to frequency domain called Fast Fourier Transform (FFT). Several libraries such as NumPy, PyAudio, Struct, Matplotlib are used to generate the fft for the Python spectrum analyzer, and to build the audio spectrum analyzer we used Python (a multipurpose programming language) and this works out of the box with the microphone in windows.

SYSTEM DESCRIPTION

At first to develop an audio spectrum analyzer, where we sample the audio from the microphone and display it in the time domain. We will be needing the following Python Libraries.

SOFTWARE USED : Visual Studio Code (IDe for Python 3.8) / Sypder Ide.

HARDWARE USED : User's System Microphone.

◦ **Numpy** is a Python Library that adds support for large, multi-dimentional arrays and matrices.

◦ **PyAudio** is another Python Library that can be easily used to play and record audio with Python on a variety of platforms.

◦ **Matplotlib** is a library for the Python and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tintern, wxPython, Qt, or GTK+.

VARIALBES : Then, we initialize some variables.

◦ **CHUNK** is the number of samples that will be processed and displayed at any instance of time.

◦ **FORMAT** is the data type that the PyAudio library will output.

◦ **CHANNELS** is the number of channels our microphone has.

◦ **RATE** is the sampling rate. We will choose the most common one, that is 44100Hz or 44.1KHz.

We will be using the **(FFT) Fast Fourier Transform** algorithm to calculate the frequency for the spectrum analyzer. Now, FFT is an algorithm that computes the Discrete Fourier Transform.

In short, the DFT of a sequence of signal to represent it in frequency domain.

There are many different FFT algorithms based on a wide range of published theories, from simple complex-number arithmetic to group theory and number theory. The one we will be using is the python fft algorithm from the library Numpy.

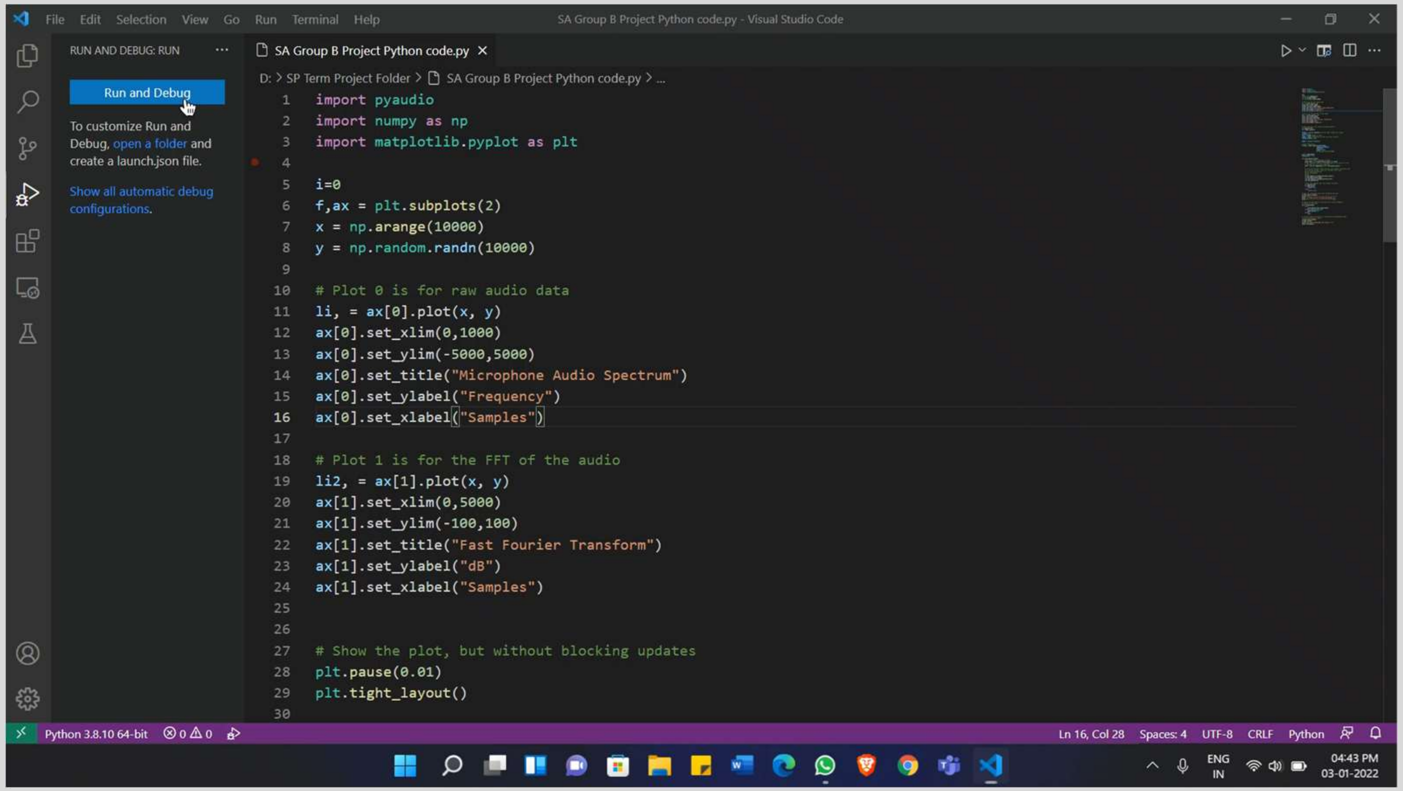
Now we will glance the methodology for this Audio Spectrum Analyzer.

METHODOLOGY

- We create a PyAudio object from class **PyAudio** and store it in a variable **p**.
- Then, we use the **open** method on the object **p** and pass on the variables we initialized as parameters.
- Then, we will initialize the plot with random values and set the limits for each of the two axes. (Two subplots)
- Finally, we create an infinite loop and read the data from the microphone, convert the data into 16-bit numbers and further plot it using the matplotlib.pyplot function.
- But, To convert Time domain to Frequency domain we convert that binary data into 16-bit integers and displayed them on a Plot using the MatPlotLib's PyPlot.

- Next, The plotting function, we add the values from **0 uptill 44100** with number of parts equal to the size of **CHUNK**. And the next line sets the X axis as semilog since the frequency representation is always done on semilog plots. **(dB)**
- Since the output of the FFT computations are going to range from 0 to 1. We change the Y limits of the frequency plot. And also, since the computation produces a mirror of the spectrum after half the sampling rate. We don’t need the part after half the sampling rate, that is after 22050. And hence we change the X limit also.
- And then in the infinite loop, add the following lines to compute the FFT and plot it.
- Since the computed FFT contains both the real and the imaginary part. We take the absolute value of the returned FFT frequency in the Spectrum, multiply it by 2, and then divide it by 33000 times CHUNK to produce a plot with Y values ranging from 0 to 1.
- Finally, RUN the code for the flawless Simulation.

SIMULATION RESULTS


SA Group B Project Python code.py - Visual Studio Code
RUN AND DEBUG: RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.
D: > SP Term Project Folder > SA Group B Project Python code.py > ...
Python 3.8.10 64-bit
Ln 16, Col 28
Spaces: 4
UTF-8
CRLF
Python
04:43 PM
03-01-2022

```python
import pyaudio
import numpy as np
import matplotlib.pyplot as plt

i=0
f,ax = plt.subplots(2)
x = np.arange(10000)
y = np.random.randn(10000)

# Plot 0 is for raw audio data
li, = ax[0].plot(x, y)
ax[0].set_xlim(0,1000)
ax[0].set_ylim(-5000,5000)
ax[0].set_title("Microphone Audio Spectrum")
ax[0].set_ylabel("Frequency")
ax[0].set_xlabel("Samples")

# Plot 1 is for the FFT of the audio
li2, = ax[1].plot(x, y)
ax[1].set_xlim(0,5000)
ax[1].set_ylim(-100,100)
ax[1].set_title("Fast Fourier Transform")
ax[1].set_ylabel("dB")
ax[1].set_xlabel("Samples")


# Show the plot, but without blocking updates
plt.pause(0.01)
plt.tight_layout()

```

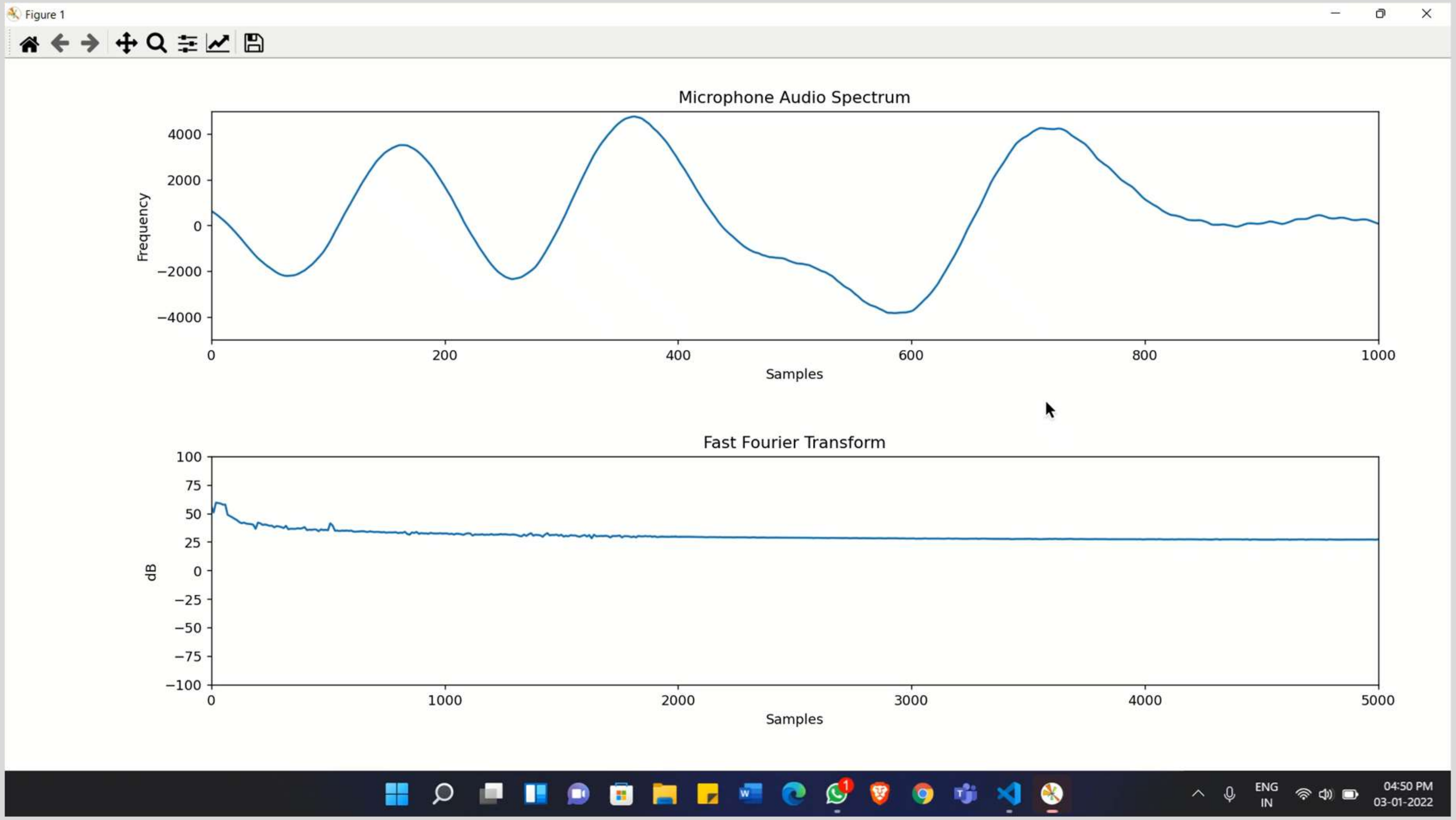


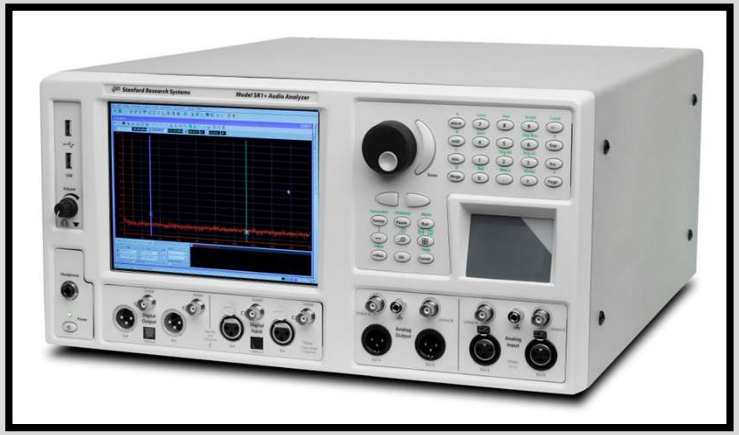
Figure 1
Microphone Audio Spectrum
Frequency
4000
2000
0
−2000
−4000
0
200
400
600
800
1000
Samples
Fast Fourier Transform
dB
100
75
50
25
0
−25
−50
−75
−100
0
1000
2000
3000
4000
5000
Samples
ENG
IN
04:50 PM
03-01-2022

APPLICATIONS

- **Where is this Audio Spectrum Analyser used ?**

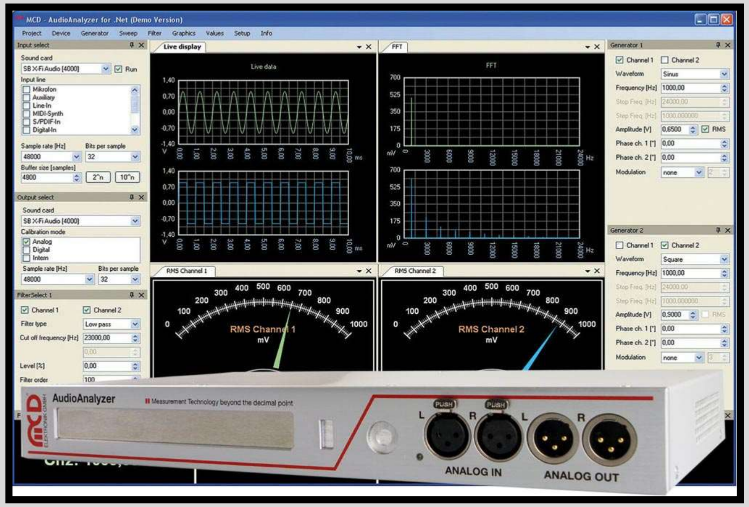
Spectrum analyzer gives insight into your sounds and deconstructs the audio spectrum, showing the levels of the various frequencies in your audio signal. The built-in sound meter provides high-quality measurement results when measuring ambient noise levels in a multitude of real-life scenarios.

1. DJ Visualization displays
2. Airplane Cockpit display – MCD Audio Analyzer
3. Spectriod
4. In some Medical purpose devices.


Stanford Research Systems
Model SR1+ Audio Analyzer

PHONIC
PAA3X
PROFESSIONAL AUDIO ANALYZER

Audio Analyzer
MacBook Pro


MCD - AudioAnalyzer for .Net (Demo Version)
Live display
FFT
RMS Channel 1
RMS Channel 2
AudioAnalyzer
Measurement Technology beyond the decimal point
ANALOG IN
ANALOG OUT

Audio Analyser
Peak
-2.3 dB
Max dB
-10
Min dB
-100
Playback Delay (ms)
250
Audio Meters
1
Window size (zoom level)
1

Thank you!