NOISE REDUCTION IN AUDIO SIGNALS USING BANDPASS FILTER

TEAM MEMBERS

NAMES :

1.AURROBINDO TEJUSVAR P CB.EN.U4ECE23107

2.AVULA SURYA PHANINDRA CB.EN.U4ECE23108

3.VISHNU VARDHAN NAIDU CB.EN.U4ECE23117

AGENDA

- Hardware and Software Requirements
- Objectives
- Methodology
- Block Diagram
- Conclusion

Hardware and software requirements

HARDWARE:

Audio Amplifier: To amplify the output signal

Analog to Digital Converter: To convert audio signal into digital format

Digital Signal Processing: For real time filtering and noise reduction

Digital to Analog Converter: To convert

SOFTWARE:

Spyder IDE: For writing and executing the python code

Python libraries:

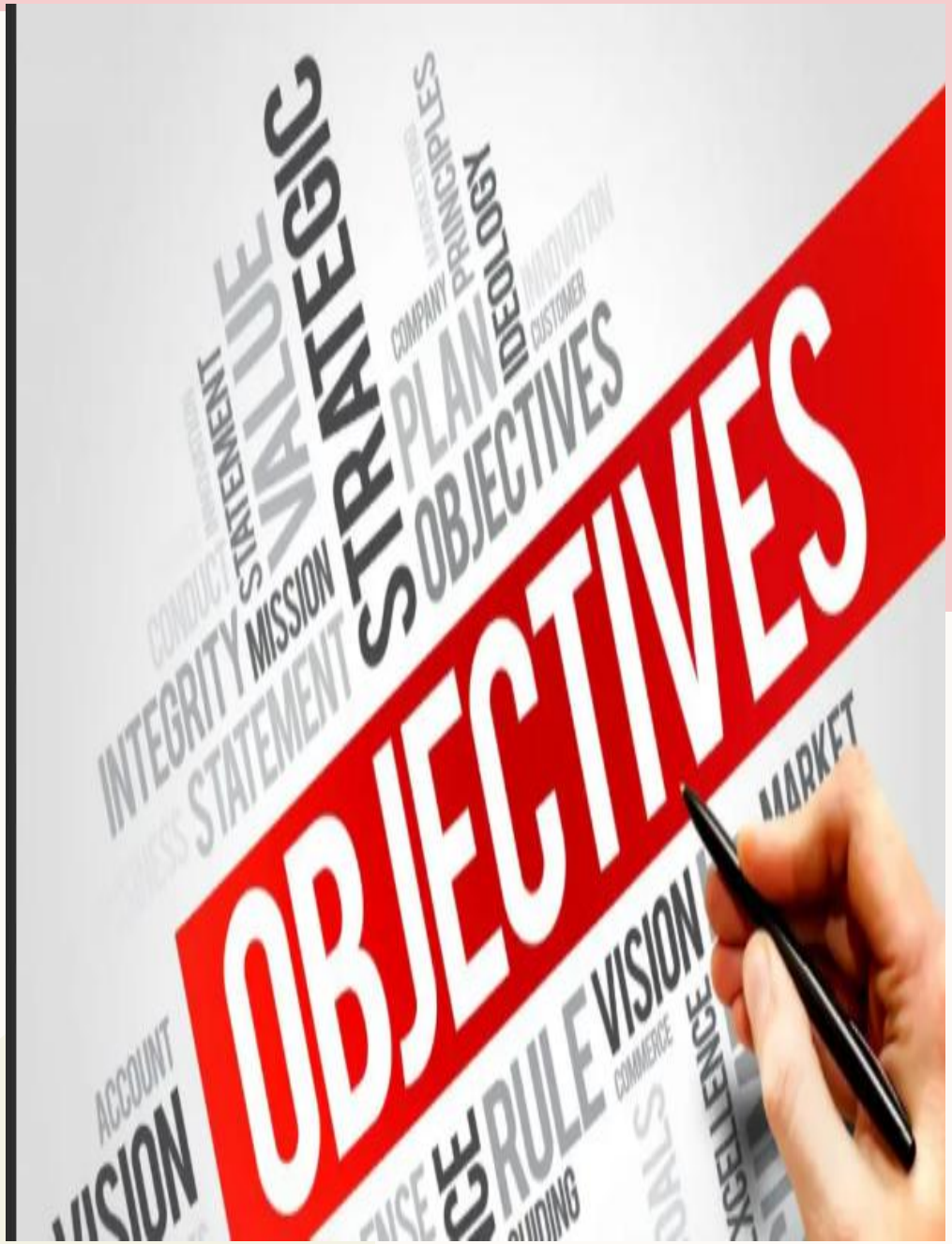
Numpy and scipy for signal processing

Soundfile for reading and writing .Wav (audio file)

Mathplotlib for visualizing the signals

OBJECTIVE

- To reduce the noise in the audio recording using digital filters
- Implement real time and post-processing noise reduction methods with software
- Produce clear output audio signal



METHODOLOGY

Step 1: Import Required Libraries

Purpose: Load essential libraries for audio processing, filtering, visualization, and playback.
numpy for numerical operations on arrays.
scipy.io.wavfile to read and save .wav files.
scipy.signal for signal processing functions, including filtering.
matplotlib.pyplot for plotting.
sounddevice for playing audio.

Step 2: Load Audio File

- **Code**: rate, audio_data = wav.read('path_to_file')
- **Purpose**: Read the audio file, obtaining:
  - **Sample Rate** (rate): Defines the number of samples per second.
  - **Audio Data** (audio_data): The actual audio signal as an array.

•**STEP 3: NORMALIZE AUDIO DATA**

CODE: AUDIO_DATA = AUDIO_DATA / NP.MAX(NP.ABS(AUDIO_DATA))

PURPOSE: NORMALIZE THE AMPLITUDE VALUES SO THEY RANGE BETWEEN -1 AND 1. THIS STEP ENSURES CONSISTENT AMPLITUDE LEVELS FOR PROCESSING AND PLAYBACK.

Step 4: Visualize Original Noisy Audio Signal

```
plt.figure(figsize=(10, 4))
plt.plot(audio_data)
plt.title("Original Noisy Audio Signal")
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.show()
```

Purpose: Display the waveform of the noisy audio to visually assess its characteristics and noise content.

•STEP 5: DEFINE BANDPASS FILTER FUNCTION

CODE: DEF BANDPASS_FILTER(DATA, LOWCUT, HIGHCUT, FS, ORDER=5)
PURPOSE: DEFINE A BUTTERWORTH BANDPASS FILTER TO ALLOW ONLY FREQUENCIES WITHIN A SPECIFIED RANGE (300-3400 HZ) AND REMOVE FREQUENCIES OUTSIDE THIS RANGE.
LOWCUT AND HIGHCUT: DEFINE THE FREQUENCY RANGE FOR THE BANDPASS.
FS: SAMPLING RATE OF THE AUDIO.
ORDER: DETERMINES THE SHARPNESS OF THE FILTER'S ROLL-OFF.
THE BUTTERWORTH FILTER (SIGNAL.BUTTER) IS DESIGNED FOR A SMOOTH FREQUENCY RESPONSE, MAKING IT SUITABLE FOR AUDIO SIGNALS.

•STEP 6: APPLY BANDPASS FILTER TO THE AUDIO SIGNAL

CODE: FILTERED_AUDIO = BANDPASS_FILTER(AUDIO_DATA, LOWCUT, HIGHCUT, RATE)

PURPOSE: PROCESS THE NOISY AUDIO SIGNAL TO REMOVE FREQUENCIES OUTSIDE THE DESIRED SPEECH RANGE, HELPING TO REDUCE BACKGROUND NOISE AND ENHANCE INTELLIGIBILITY.

Step 7: Plot Filtered Audio Signal

```
plt.figure(figsize=(10, 4))
plt.plot(filtered_audio)
plt.title("Filtered Audio Signal")
plt.xlabel("Sample")
plt.ylabel("Amplitude")
plt.show()
```

Purpose: Visualize the filtered signal to confirm that the noise reduction process has been applied.

•STEP 8: DEFINE POWER SPECTRUM PLOTTING FUNCTION

CODE: DEF PLOT_POWER_SPECTRUM(DATA, FS, TITLE)
PURPOSE: DEFINE A FUNCTION TO ANALYZE AND VISUALIZE THE FREQUENCY CONTENT OF THE AUDIO SIGNAL USING THE **POWER SPECTRAL DENSITY (PSD)**.
THE PSD PLOT SHOWS THE DISTRIBUTION OF POWER ACROSS DIFFERENT FREQUENCY COMPONENTS, HELPING TO CONFIRM THAT NOISE HAS BEEN REDUCED IN THE FILTERED AUDIO.

Step 9: Plot Power Spectrum of Original and Filtered Signals

```
plot_power_spectrum(audio_data, rate, "Power Spectrum of Original Noisy Audio")
plot_power_spectrum(filtered_audio, rate, "Power Spectrum of Filtered Audio")
```

Purpose: Compare the power spectrum of the original and filtered signals to assess the effectiveness of the noise reduction. The filtered signal should show reduced power outside the desired frequency range (300-3400 Hz).

Step 10: **Save Filtered Audio to a New File**

```
filtered_audio = np.int16(filtered_audio * 32767)
wav.write('filtered_audio.wav', rate, filtered_audio)
```

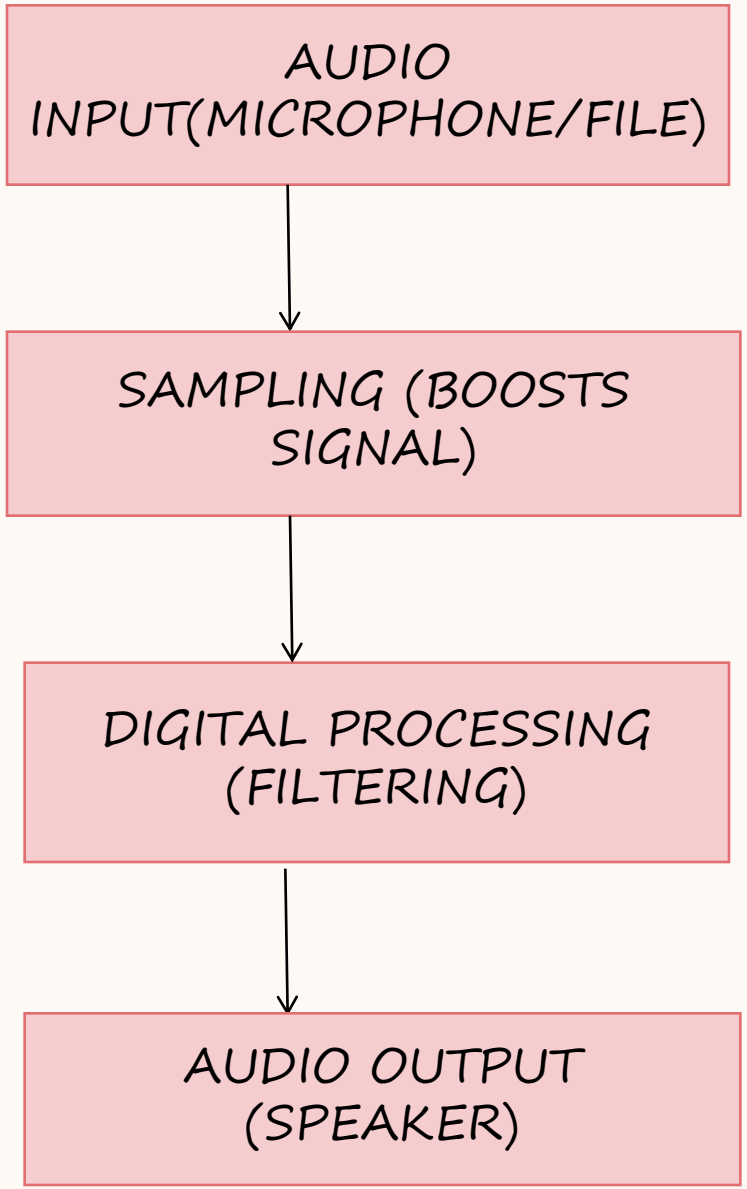
Purpose : convert the filtered audio back to int 16 format for saving and playback compatibility

Step 11: Play Original and Filtered Audio

```
print("Playing Noisy Audio...")
sd.play(audio_data, rate)
sd.wait()
print("Playing Filtered Audio...")
sd.play(filtered_audio, rate)
sd.wait()
```

Purpose : listen to both filtered and original audio to compare audio levels

BLOCK DIAGRAM



ANALOG TO DIGITAL CONVERTER

DIGITAL TO ANALOG CONVERTER

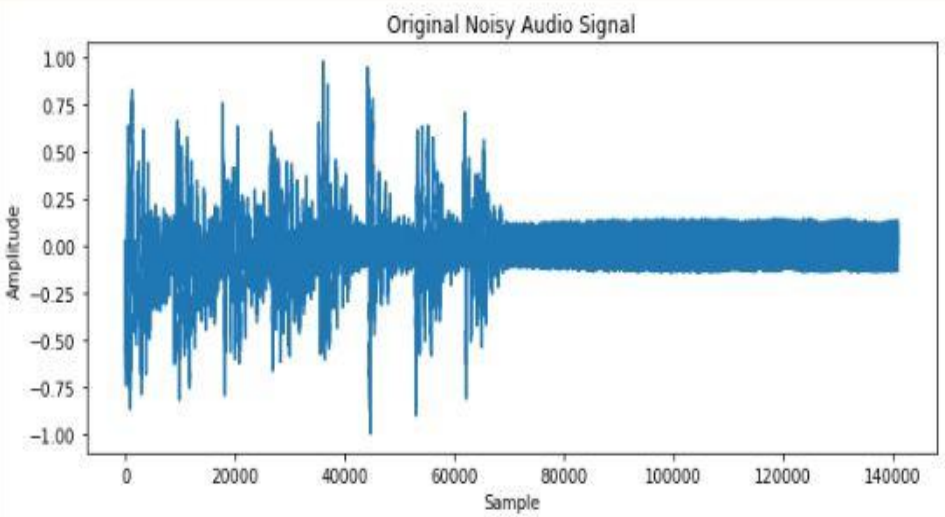
TIMELINE

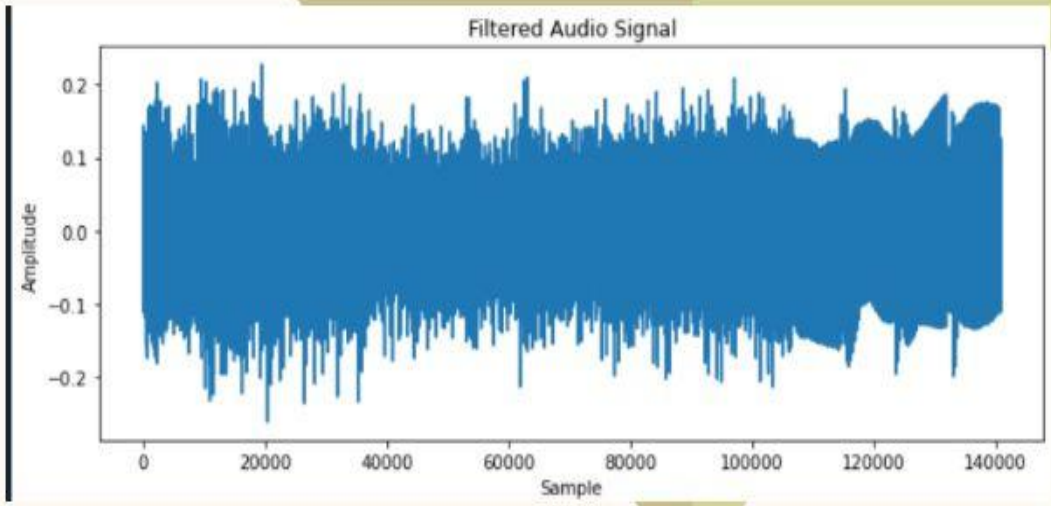
MONTH	WEEK	PROGRESS
July	Week -4	Group formation for project
August	Week - 1	Title selection and research on the project
	Week - 2	Learning about the filters
	Week - 3	Discussed with group members and formulated algorithm
	Week - 4	Implemented the algorithm in spyder IDE
September	Week - 1	Assigned the filtered value and verified the output spectrum
	Week - 2	Import audio signal using audacity
	Week -3	Made necessary and minor changes in code
	Week - 4	Import sound file to code to get audio signal as output
October	Week - 1	Include the power and the spectrum display to the code
	Week -2	Discussed the topic to cover for PPT
	Week -3	Final corrections and checking for errors and adding comments
	Week - 4	Prepare PPT for presentation
November	Week - 1	End semester final lab evaluation

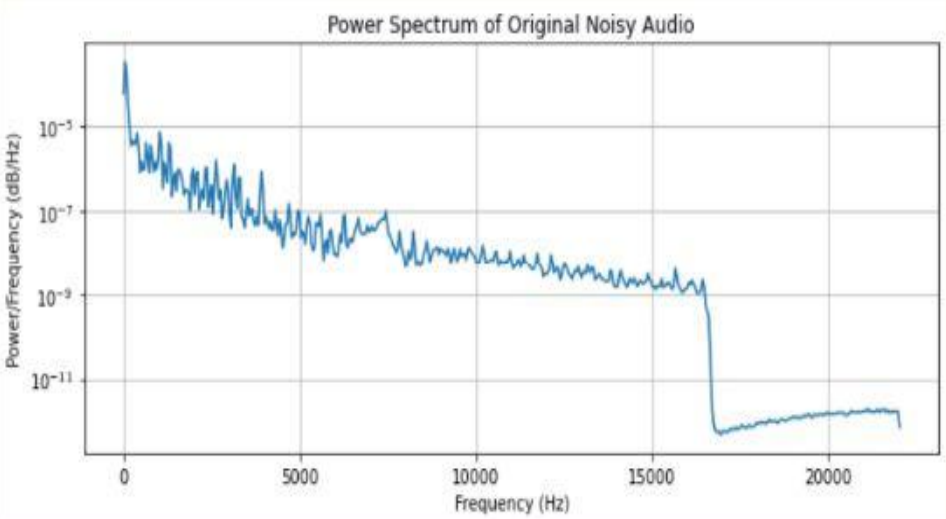
CONCLUSION

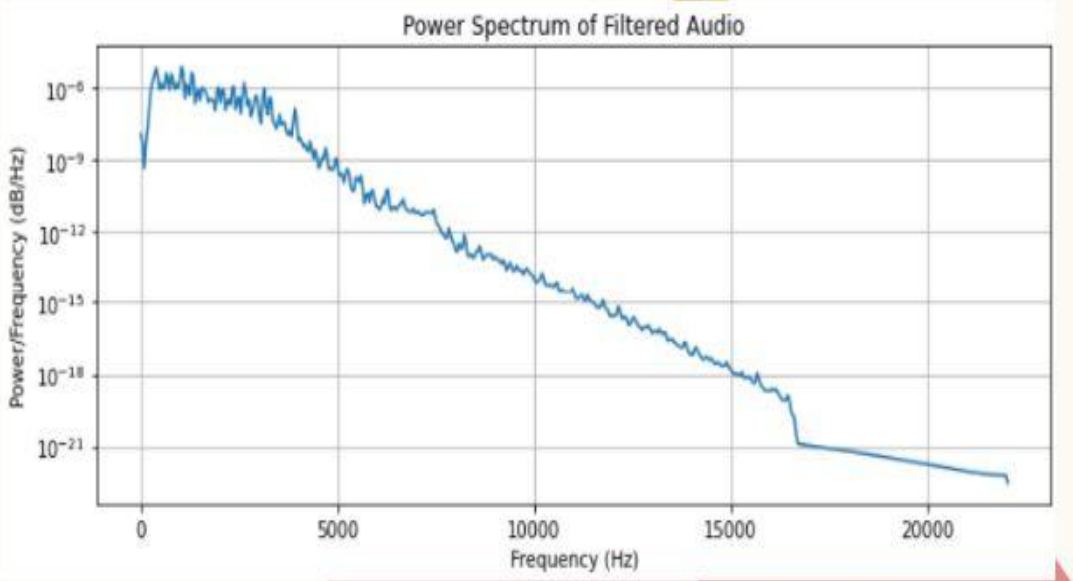
The code successfully reduces noise in an audio file by applying a bandpass filter, which limits frequencies to a specified range. This range targets common voice frequencies (300–3400 Hz) while reducing higher and lower frequency noise. By plotting both the waveform and power spectrum of the original and filtered signals, the code provides a clear view of how the bandpass filter impacts the audio data.

RESULT


Original Noisy Audio Signal
Amplitude
Sample
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20000
40000
60000
80000
100000
120000
140000


Filtered Audio Signal
Amplitude
Sample
0.2
0.1
0.0
−0.1
−0.2
0
20000
40000
60000
80000
100000
120000
140000


Power Spectrum of Original Noisy Audio
Power/Frequency (dB/Hz)
Frequency (Hz)
10^{-5}
10^{-7}
10^{-9}
10^{-11}
0
5000
10000
15000
20000


Power Spectrum of Filtered Audio
Power/Frequency (dB/Hz)
Frequency (Hz)
10^{-6}
10^{-9}
10^{-12}
10^{-15}
10^{-18}
10^{-21}
0
5000
10000
15000
20000

THANK YOU