SIGNALS & SYSTEMS (MCT-335L)

Semester Project Report

Streamlined Audio Filtration and Analysis

Submitted by:

Name: **Muawwiz Ali Yousuf** Registration No: **2021-MC-13**

Name: **Muhammad Saad Yaseen** Registration No: **2021-MC-37**

Name: **Muhammad Rafay** Registration No: **2021-MC-39**

Submitted to:

Ma'am Maliha Bakhshi

Department of Mechatronics & Control Engineering

University of Engineering & Technology, Lahore.

May-2024

Table of Contents

ABSTRACT

This project addresses the common issue of noise in audio signals, focusing specifically on the wireless transmission of audio from a microphone to a computer. The captured audio signals, often contaminated with noise, are processed using MATLAB for signal analysis and processing. Advanced filtration techniques are applied in both the time and frequency domains to denoise the audio signal effectively. The primary objective is to achieve a clear and crisp audio output with minimal noise interference. A user-friendly Graphical User Interface (GUI) has also been developed to facilitate seamless visualization, loading, and playback of audio signals. This interface enhances user interaction, providing an intuitive platform for real-time signal analysis and audio processing.

INTRODUCTION

Signals are the representation of information in the fields of science and engineering that are carried by physical quantities such as sound waves, voltage, or even temperature. Think of them as communications that are sent via several channels. In essence, systems are processors that respond to these signals and alter them in some manner. Consider them as the communications' recipients and interpreters, carrying out different tasks to determine their meaning or alter their format.

Noise in audio signals significantly impacts sound quality in various applications. This project addresses this issue by focusing on the wireless transmission of audio from a microphone to a computer, where noise often gets introduced during transmission. Using MATLAB for signal analysis and processing, the project applies advanced filtration techniques in both the time domain and frequency domain to effectively remove noise from the audio signal. This ensures a clearer and more intelligible audio output.

PROBLEM STATEMENT

Audio signal noise is a widespread issue that affects sound quality and intelligibility in many contexts. For instance, consider a content creator who uses an inexpensive microphone to record audio for videos. The microphone might pick up significant background noise from the environment, leading to distorted and unclear audio. This common problem not only diminishes the viewing experience but also hampers effective communication. In professional settings, such as live broadcasts or conference calls, poor audio quality due to noise can disrupt the flow of information and lead to misunderstandings. Addressing this widespread issue is crucial for enhancing audio clarity and ensuring effective communication.

SOLUTIONS AVAILABLE

Current solutions for noise removal in audio signals often rely on advanced AI-generated techniques, which are typically expensive and proprietary. These methods, while effective, may not be accessible to all due to their high costs. There are also expensive microphones available that filter noise due to their built-in features. This project offers a more cost-effective alternative by utilizing MATLAB's filtration techniques to denoise audio signals. MATLAB provides various filters, including Finite Impulse Response (FIR), low pass, high pass, Wiener, Chebyshev, and Butterworth (butter) filters. In this project, the Butterworth filter is favored which ensures minimal

signal distortion while effectively removing noise. This makes it ideal for applications requiring clear and undistorted audio, providing a balanced solution between complexity and performance. By leveraging the Butterworth filter, this project achieves high-quality noise reduction without the need for expensive AI solutions.

METHODOLOGY

The methodology to create this audio noise removal system using MATLAB is given below:

AUDIO TRANSMISSION

The audio signal is captured using a MEMS microphone interfaced with an ESP32 microcontroller. The ESP32 transmits the audio data to a PC where a local WebSocket server, built using Node.js (a JavaScript runtime environment), is running. The server processes and manages the audio data, which is then accessed via a webpage. The webpage is created using HTML and CSS and can be viewed in a web browser[1].

1. **INMP441 Interfacing with ESP32:**

   The INMP441 microphone is connected to an ESP32-S microcontroller. The microcontroller uses the I2S protocol to read digital audio data from the microphone. I2S is a method used to send high-quality digital sound. It's often used to connect audio devices like converters, processors, and microcontrollers.

2. **URL:**

   Following will be the format for the url:

   http://<PC_IP_ADDRESS>:8000/audio

3. **Plotly.js:**

```
<script src="https://cdn.plot.ly/plotly-latest.min.js" charset="utf-8"></script>
```

- It is an open-source graphing library that makes interactive, publication-quality graphs online.
- It supports a wide range of chart types and is highly customizable, making it ideal for data visualization.

PCM Player:

```
<script src="https://unpkg.com/pcm-player"></script>
```

- It is a JavaScript library for playing PCM audio data directly in the browser.
- It supports various audio codecs and sample rates, enabling real-time audio playback from raw PCM streams.

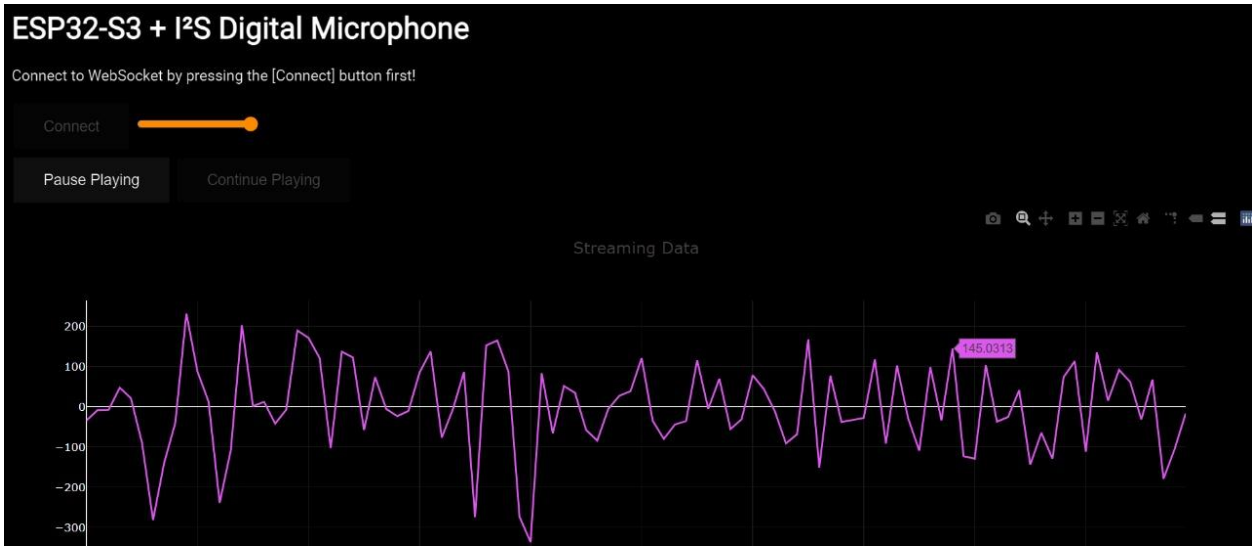


Figure 1: Web Page for Audio Transmission

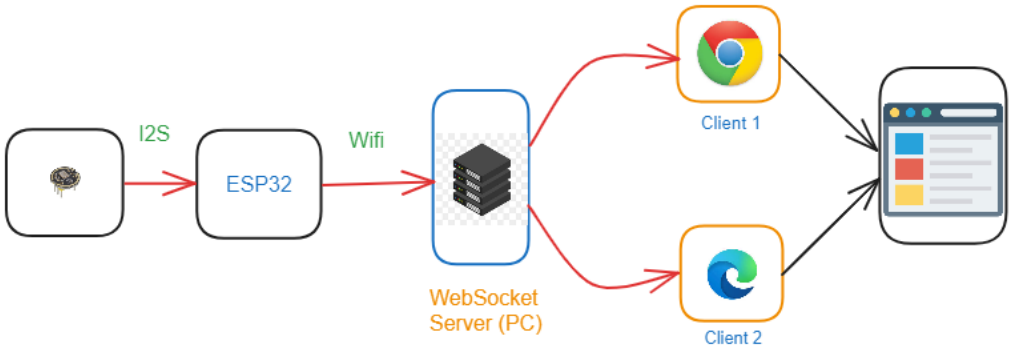


Figure 2: Audio Transmission Flowchart

AUDIO PROCESSING

Every audio has a different sampling frequency and amplitude which needs different types of filters and processing techniques to remove the background and distorted noise. So, to perform the signal processing on the audio using MATLAB, the steps are given below:

4. Load the Audio

The transmitted audio signal from the Web Server is saved in the PC using *Audacity* from where it is loaded into the MATLAB environment using:

```
[y, Fs] = audioread('noisy2_group2.wav');
disp(['Sampling Rate: ', num2str(Fs), ' Hz']);
```

5. Filtering the Data

The low pass filter is applied to this noisy signal to remove the background, distorted, and high amplitude from the noise. For this, the Butterworth filter is used, which is a type of low-pass filter.

It provides a smooth frequency response without sharp cutoffs, minimizing signal distortion while preserving the signal's essential components, resulting in clearer audio. In this code, we provide the order of the filter, the cutoff frequency, and the type of filter which is used[2].

```
order = 8;
cutoff_freq = 700;
[b, a] = butter(order, cutoff_freq/(Fs/2), 'high');
audio_filtered = filter(b, a, audio);
```

6. Plotting in time and Frequency Domain

The original noisy and filtered audio signal is plotted in time and frequency domain in MATLAB which shows the spectrum peaks of the signals. Firstly, the ***Fast Fourier transform*** is applied to the noisy signals in the frequency spectrum in which these signals are plotted. Also, the time domain plotting is shown.

```
Frequency Domain Representation of the Original Signal
f = (-Fs/2):(Fs/N):(Fs/2 - Fs/N);
Freq_audio = fftshift(fft(audio));
plot(f, abs(Freq_audio), 'b');
Time Domain Representation of the Original Signal
N = length(audio);
time = (0:N-1) / Fs;
plot(time, audio, 'b');
Frequency Domain Representation of the Filtered Signal
f = (-Fs/2):(Fs/N):(Fs/2 - Fs/N);
Freq_filtered = fftshift(fft(audio_filtered));
plot(f, abs(Freq_filtered), 'r');
Time Domain Representation of the Original Signal
N = length(audio);
time = (0:N-1) / Fs;
plot(time, audio_filtered, 'r');
```

7. Plotting the Spectrogram

The audio signal spectrogram is plotted in MATLAB, in which a time-frequency representation is given that shows how the signal's frequency content varies over time. The x-axis represents time, the y-axis represents frequency, and the color intensity indicates the magnitude of each frequency component. In this representation, brighter or warmer colors (such as yellow or red) depict higher magnitudes, indicating stronger frequency components, while darker or cooler colors (such as blue or black) indicate lower magnitudes, representing weaker frequency components.

```
Spectrogram Representation of the Original Signal
spectrogram(audio, 256, [], [], Fs, 'yaxis');
title('Spectrogram - Original Audio');

Spectrogram Representation of the Original Signal
spectrogram(audio_filtered, 256, [], [], Fs, 'yaxis');
title('Spectrogram - Filtered Audio');
```

8. Saving the Audio

At last, the filtered audio is saved in a *.wav* file after that it will be played in a separate file for future use.

```
soundsc(audio_filtered, Fs);
audiowrite('filtered_audio.wav', y_filtered, Fs);
```

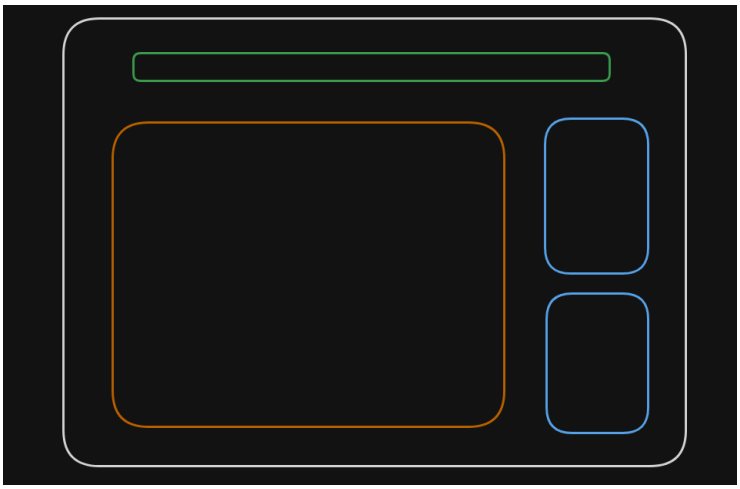
GUI

The Graphical User Interface is a user-friendly interface for visualizing data, enabling users to adjust parameters and instantly see results. This organized visualization enhances data comprehension, analysis, and decision-making by making complex data more accessible and easier to interpret[3].

In this project, a GUI is created in *App Designer* in which the app's various features are added, and in Code view, the logic of the buttons and features are implemented. This completely displays the plots of original noisy and filtered signals in both the time domain and frequency. The features that are given in the GUI are:

- Browsing the audio files in any format.
- Settings of the Parameters such as Cutoff frequency and Order of the filters.
- Specify the type of filter as 'high' and 'low'.
- Playing the Original and Filtered audio.
- Displaying of the plots in the Time and Frequency domain.
- Saving the audio in the directory.

Initial App Layout:



The final display of the GUI looks as:

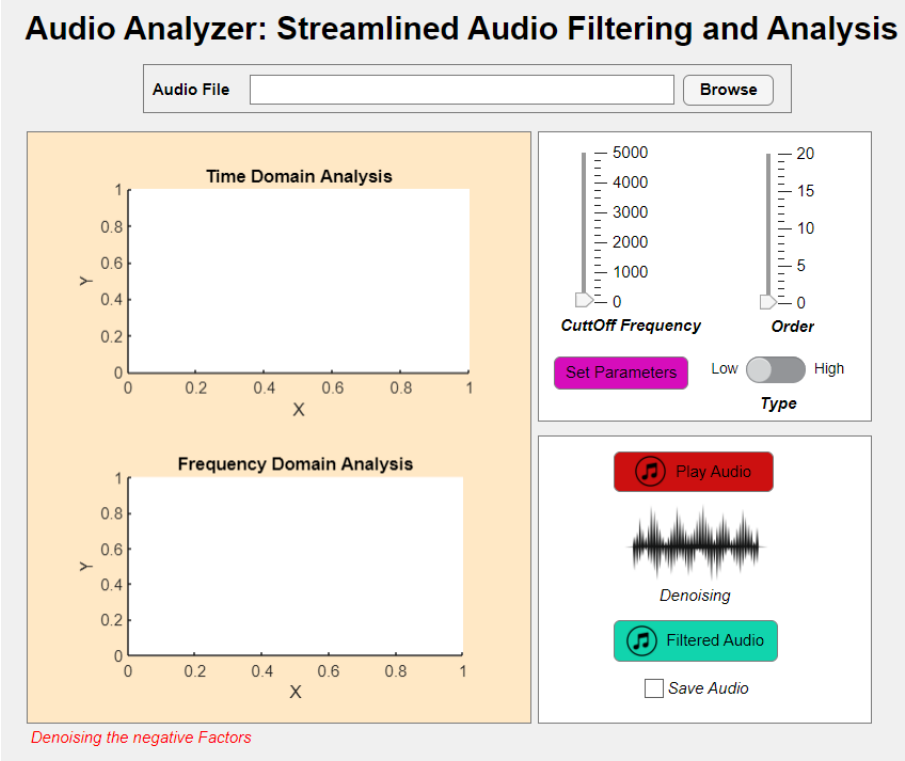


Figure 3: Graphical User Interface

RESULTS AND ANALYSIS

After the data processing, the results are displayed using the plots and audio is clearer without any disturbance and noise. The plots of the spectrogram and in the time and frequency domain of 2 different audio signals are given below:

Audio Signal 1: (Low-pass Filter)

1. **Time Domain Plots**

The top graph shows the amplitude of the original audio signal over time, which includes background noise and distortions. The bottom graph displays the filtered audio signal after applying a low-pass Butterworth filter, effectively reducing high-frequency noise and preserving the main components of the audio.

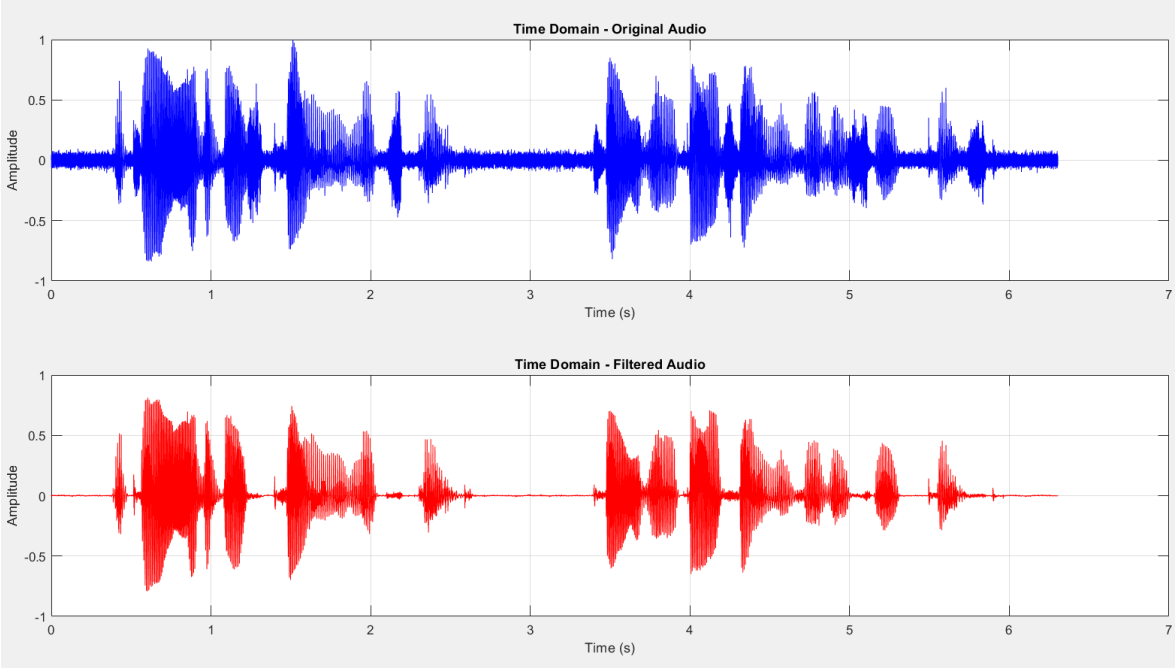


Figure 4: Time Domain Plot of Original and Filtered Audio

2. Frequency Domain Plots

In the frequency domain, a low-pass filter attenuates high-frequency components while allowing low-frequency components to pass through. This process reduces high-frequency noise and preserves the main low-frequency content of the signal. So, in the below graph, 1st one represents noisy data and 2nd graph is due to the low pass filter that gives smooth audio at the output.

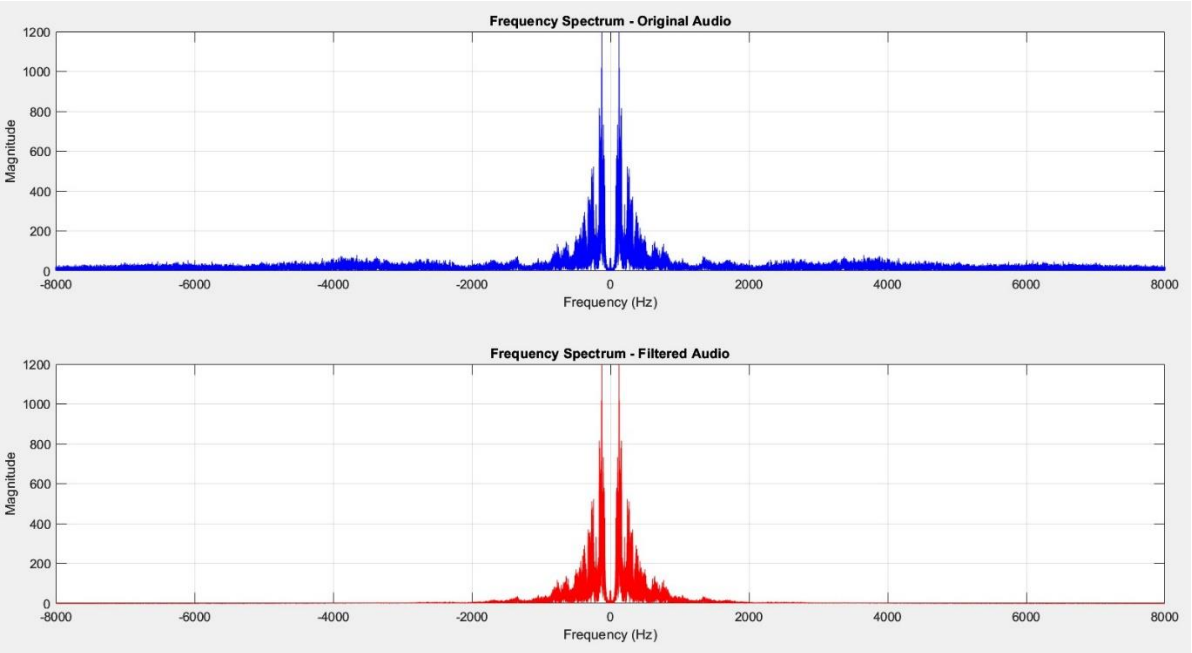


Figure 5: Frequency Domain Plot of Original and Filtered Audio

3. Spectrogram Plots

The 1st spectrogram shows that the high-frequency noise and distortions are visible as widespread energy across the spectrum. The 2nd spectrogram, exhibits a cleaner signal with reduced high-frequency content, concentrating the energy in the lower frequency bands. This indicates that the filter has effectively removed high-frequency noise, enhancing the clarity of the audio signal.

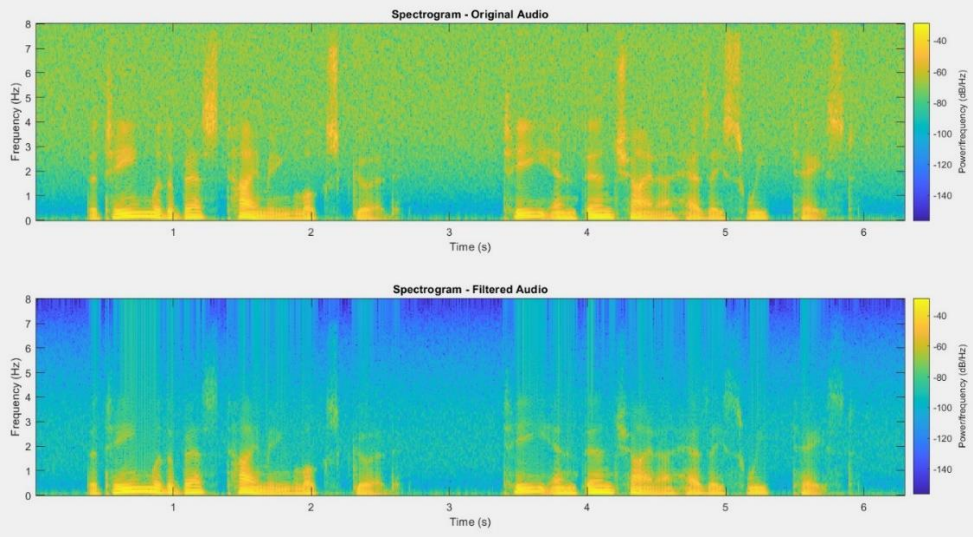


Figure 6: Spectrogram Plot of Original and Filtered Audio

Audio Signal 2: (High-pass Filter)

4. Time Domain Plots

In this graph, a high pass filter is applied to the audio, and the time domain plots of both the original and filtered audio are displayed which indicates high pass filter removes low-frequency sound signals by allowing high-frequency waves to pass through.

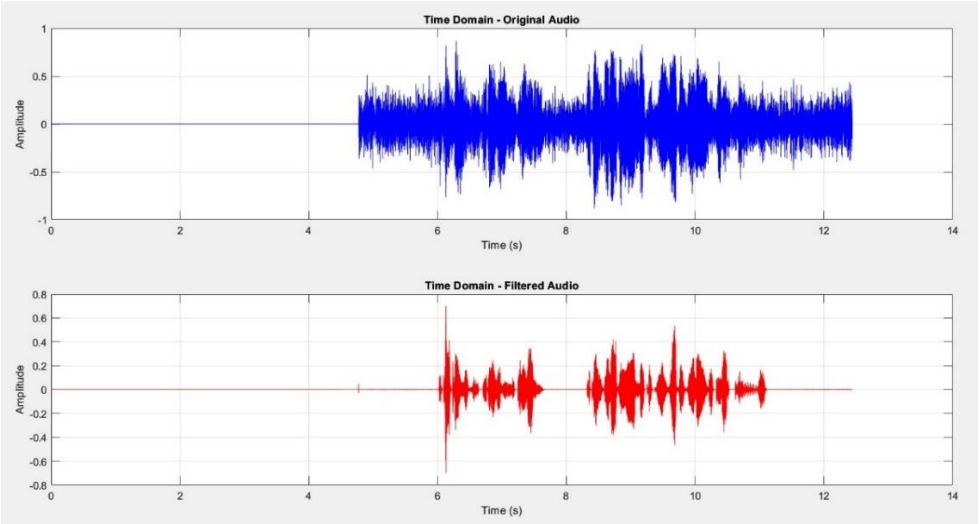


Figure 7: Time Domain Plot of Original and Filtered Audio

5. Frequency Domain Plots

The original audio's frequency spectrum shows dominant low-frequency components, while the filtered audio's spectrum indicates these low frequencies have been attenuated, emphasizing higher frequencies. This confirms the high-pass filter's effectiveness in removing low-frequency noise.

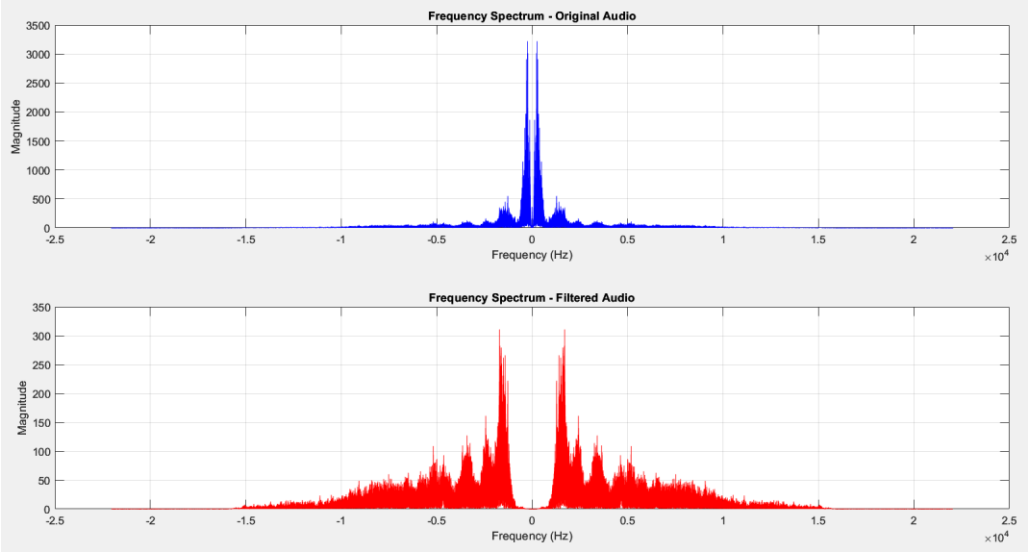


Figure 8: Frequency Domain Plot of Original and Filtered Audio

6. Spectrogram Plots

The spectrograms show the frequency content of the audio over time. In the original audio (top graph), high-frequency noise is present, especially above 700 Hz. In the filtered audio (bottom graph), this high-frequency noise has been significantly reduced, indicating that the high-pass filter effectively removed low-frequency components below the cutoff frequency. The color intensity represents the power level of frequencies, with yellow being higher and blue being lower.

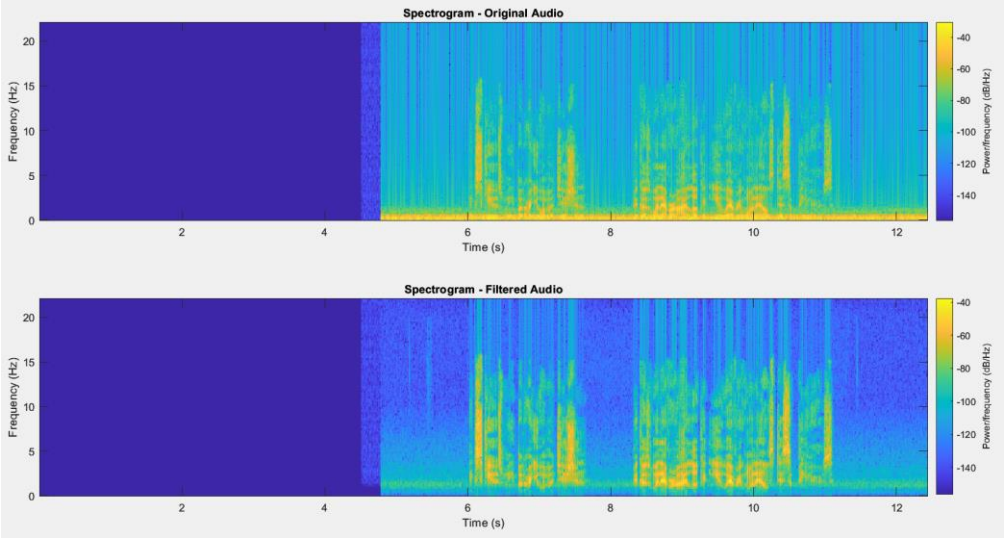


Figure 9: Spectrogram Plot of Original and Filtered Audio

ENHANCEMENTS

The future enhancements for this project are:

- Implementing better and faster filtering techniques such as spectral subtraction and Kalman filters to improve audio quality and reduce noise.
- Leveraging machine learning algorithms for advanced noise removal, which can adapt to various audio environments and improve the clarity of the recorded sound.
- Developing robust reconstruction techniques to recover any audio data that might be lost during the filtering process, ensuring the integrity and completeness of the audio.
- Developing different audio enhancement techniques more than filtration like amplification, equalization

LIMITATIONS

Some limitations of this project include:

- For data transmission, the local PC and ESP32 should be connected to the same Wi-Fi connection.
- A high-speed WI-FI connection should be required for greater-quality audio transmission. Otherwise, a delay would occur during transmission.
- These filtration processes cannot filter the audio which has a large amount of noise in it.

CONCLUSION

In general audio transmission, filtration, and processing allow users to modify audio data by adjusting factors like cutoff frequency and filter order, and visualizing their influence. This capability addresses real-life challenges faced by content creators, enabling them to enhance audio quality by removing unwanted noise and improving clarity. By providing an interactive interface, these tools empower creators to optimize their audio content, thereby meeting the demands of their audience and ensuring a better listening experience.

REFERENCES

1. Project, T. *Broadcasting Your Voice with ESP32-S3 & INMP441*. 2024 May 17, 2023 [cited 2024 28-05-2024]; Available from: https://www.youtube.com/watch?v=qq2FRv0lCPw&t=424s.
2. MathWorks. *Filtering noise from an audio signal*. 2024 [cited 2024 29-05-2024]; Available from: https://www.mathworks.com/matlabcentral/answers/1608710-filtering-noise-from-an-audio-signal.
3. Hurayrah, T.b. *MATLAB speech Enhancement using Filters (GUI)*. 2024 Apr 8, 2023 [cited 2024 29-05-2024]; Available from: https://www.youtube.com/watch?v=jnmzv48xNak.