ESP32 Sound Signal Noise Reduction Using STFT

M. Adnan Bayu Firdaus
Faculty of Advanced Technology and Multidisciplinary
Universitas Airlangga
Mulyorejo, Surabaya
muhammad.adnan.bayu-2021@ftmm.unair.ac.id

Cahyan Irfan Syach
Faculty of Advanced Technology and Multidisciplinary
Universitas Airlangga
Mulyorejo, Surabaya
cahyan.fan.syach-2021@ftmm.unair.ac.id

Fatih Ulwan Annaufal
Faculty of Advanced Technology and Multidisciplinary
Universitas Airlangga
Mulyorejo, Surabaya
fatih.ulwan.annaufal-2021@ftmm.unair.ac.id

Timo Widyanvolta
Faculty of Advanced Technology and Multidisciplinary
Universitas Airlangga
Mulyorejo, Surabaya
timo.widyanvolta-2021@ftmm.unair.ac.id

***Abstract* (*Abstract*)**

Voice input processing is crucial in modern applications, especially within IoT systems, where clear communication is often hindered by background noise. This paper presents a real-time noise reduction system that captures voice input through a microphone module, transmits data using the ESP32 microcontroller and MQTT protocol, and processes it for noise reduction. The system integrates both hardware and software components to ensure efficient and low-latency voice signal processing, making it well-suited for noisy environments. Our approach demonstrates significant improvements in voice clarity, offering a practical solution for enhancing voice-based interactions in IoT applications.

Keywords—Voice Signal Processing, ESP32, Noise Reduction, FFT, STFT

I. Introduction

Voice input processing plays an increasingly important role in modern applications, powering technologies such as smart home devices, virtual assistants, and various IoT-based systems. As users rely more heavily on voice commands for seamless interaction with technology, the need for accurate and responsive voice input processing becomes crucial. However, one of the biggest challenges in this domain is managing the presence of background noise, which can significantly degrade the quality of voice signals and disrupt the performance of voice-based systems.

In noisy environments, especially within IoT ecosystems where devices must communicate effectively, traditional voice input systems often struggle to deliver clear, reliable sound. This highlights the need for efficient noise reduction techniques that can process voice input in real time, ensuring that commands are accurately captured and transmitted. As IoT devices increasingly depend on wireless communication, it is essential to address noise issues that can occur during data transmission and voice recognition tasks.

The primary objectives of this work are to develop a system that captures voice input through a microphone module, transmits the data using the ESP32 microcontroller via the MQTT protocol, processes the data for noise reduction, and ultimately provides a clean and intelligible voice signal. By addressing these goals, we aim to create a real-time noise reduction system that enhances the performance of voice-based IoT systems.

The key contribution of this work is the integration of both hardware and software components to enable efficient, real-time processing of voice data. The system not only captures and transmits voice input but also processes it on-the-fly to reduce noise, ensuring clear communication in challenging environments. This novel approach provides a practical solution for improving the clarity of voice signals in IoT applications, thereby enhancing user experience and system reliability.

II. Literature Review

A. Existing Noise Reduction Techniques

In image processing, audio enhancement and signal processing, noise reduction algorithms are very important to improve the quality of a signal by decreasing an unwanted selection of voltages and currents. Among the many known processes in audio processing, we can highlight the use of adaptive filters such as Wiener filter. It can be effective across a wide range of noise conditions, however, its primary drawback lies in the necessity for careful parameter tuning that prevents real-time execution.

Harmonic Regeneration Noise Reduction (HRNR) is a significant approach, proposed by Salmuthe and Agrawal (2017), for recovering lost harmonics after elimination of noise using conventional noise reduction techniques. Although HRNR delivers gain to improve speech clarity, it suffers from complicated tuning and poor generalizability toward different noise types. In sum, the main limitations of current noise reduction methods are adaptability and real-time deployment due to latent characteristics from diverse noises in speech signal, making it crucial to establish a model that can be flexible to different noise environments with high efficiency.

B. Voice Processing in IoT Devices

Voice processing as a component in the Internet of Things (IoT) ecosystem enabling natural and intuitive interactions between users and devices. IoT devices with voice capabilities facilitate hands-free control, enhance accessibility, and provide a seamless user experience. Voice processing integration in IoT devices spans various applications, including smart homes, wearable technology, automotive systems, and industrial automation.

Communication protocols like MQTT (Message Queuing Telemetry Transport) are facilitating efficient data transmission in IoT applications. MQTT is a a lightweight, publish-subscribe network protocol ideal for devices with limited resources and unreliable networks. Research by Bandyopadhyay and Bhattacharyya (2013) demonstrated that MQTT effectively supports real-time data transmission in IoT environments, making it suitable for voice data transfer [1].

Recent research by Jin Hyun et al. (2021) introduced two optimization techniques for reducing the computational complexity of VAD algorithms tailored for edge-level devices. These techniques involve delta operations and feature extension, which exploit similarities between speech features at adjacent time steps. The delta operation reduces multiply-accumulate (MAC) operations by skipping elements that do not exhibit significant changes, while the feature extension technique substitutes similar features, reducing redundant computations in the fully connected layer [2].

C. *MQTT Protocol in Data Transmission*

With its lightweight minimalistic architecture, the capabilities for efficient data transmission around the globe, Message Queuing Telemetry Transport (MQTT) has emerged as one of the major preferred communication protocol for Internet of Things (IoT) applications. MQTT is ideal for low-power devices present in minimal-resource environments like sensors and custom hardware, which we often see in the context of IoT. Overview MQTT is an incredibly lightweight publish/subscribe protocol that was designed for low-bandwidth, high-latency networks by Thayyib [6], which makes it a good choice for use cases where data efficiency really matters. Moreover, Azzedin and Alhazmi [7] also highlight that MQTT high message distribution efficiency is inherited from the centralized broker architecture, as well as from the lightweight nature of its protocol that minimizes data transmission overhead; a very important aspect especially for environments with bandwidth constraints.

This aids in achieving greater scalability and flexibility in IoT applications, which is mostly due to the pub/sub communication mechanism facilitated by MQTT. Vaccari et al. Rivera et al. [8] describe the MQTT, an event-based protocol with multiple topics to which clients can subscribe, for conveying data across sensors and information gateways without requiring individual ICT device-to-device communication. It is especially valuable in large-scale IoT deployments when many devices have to communicate at once without flooding the network. In addition, MQTT offers QoS levels (QoS) that are applied dynamically to meet various needs for each message. MQTT persisted session support also makes the protocol more robust, if a client loses its connection it can reconnect and reestablish communication without any loss, just like before.

Though comes with a lot of benefits, MQTT is not perfect and has obviously limitations especially in terms of security (like vulnerable to denial-of-service attacks). Nevertheless, the benefits outweighing the risks as such research and development for improvement of MQTT security for IoT applications is still in progress [2]. Another benefit is that as an open-source, widely adopted platform after all, any technology gaps between different systems can be filled with MQTT itself which enables integration through the power of community. To sum up, MQTT stands out as a choice to consider due to its lightweight, scalable message patterns, reliable delivery mechanisms and vibrant community support — a sound combination applicable to anything between smart home setups to industrial automation use-cases.

III. Methodology

A. *System Architecture*

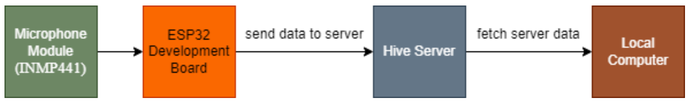


Fig.1. System block diagram

There are several components used in the system design including microphone module (INMP441), ESP32 development board, hive server, and a local computer. Microphone module acts as an input device with the input data being sound data and the sampling rate. The microphone module will be connected to an ESP32 development board, the ESP32 responsible for sending the data to the Hive server. In order to be able to send data to the Hive server, the ESP32 will be using MQTT protocol. The local computer will be used for fetching the sound data from the server and processing it later. Algorithm used for denoise the sound data is the digital band pass filter.

B. *Hardware Components*

- Microphone Module

  The INMP441 omnidirectional microphone is a high-performance digital microphone used in this system to capture audio data [8]. It is well-suited for applications that require clear voice input, as it offers excellent sound quality and sensitivity. Unlike traditional analog microphones, the INMP441 outputs digital data, reducing the need for complex analog circuitry and making it ideal for use with microcontrollers like the ESP32.

- ESP32 Microcontroller

  The ESP32 is a versatile microcontroller that plays a crucial role in this system, acting as the central processing unit for capturing and transmitting voice data [9]. Known for its robust performance and integrated Wi-Fi and Bluetooth capabilities, the ESP32 is highly suited for IoT applications. In this project, the ESP32 manages the data collected from the INMP441 microphone, transmitting it efficiently over the MQTT protocol. Its dual-core architecture allows it to handle multiple tasks simultaneously, including real-time processing and data communication, ensuring low latency and efficient noise reduction. Additionally, the ESP32's low power consumption and small form factor make it an ideal choice for embedded systems and IoT devices.

C. Data Transmission

- MQTT Protocol Implementation

  In this project, MQTT is used as the communication protocol to transmit audio data from the ESP32 to a processing unit. The ESP32 is configured as an MQTT client, connecting to a broker via Wi-Fi. It captures voice input using the INMP441 microphone and publishes the data to a specific topic (e.g., "audio/input"). The broker then distributes the data to subscribed clients for noise reduction and further processing. MQTT's lightweight publish-subscribe architecture ensures efficient and reliable data transmission, making it ideal for real-time voice processing in IoT applications with limited bandwidth.

- Hive Dashboard as Database

  The Hive dashboard functions as a real-time data management platform, receiving and storing audio data transmitted via MQTT from the ESP32. It organizes the data with metadata like timestamps and device IDs, allowing easy access to historical and real-time data for monitoring and analysis. The dashboard also provides visual tools for users to track system performance, noise levels, and voice input processing. By offering an intuitive interface and scalable storage, the Hive dashboard ensures efficient management and accessibility of the data, supporting seamless noise reduction in IoT environments.

D. Software Implementation

- Sound Dataset

  The sample sound data length and sampling rate determine the analysis frequency, here is the data card for the sample sound.

No	Dataset Name	Dataset Sound Card
1	Sample-Sound-Sambutan-Jokowi-Aksi-212.wav	data length: 2693120 integer sampling rate: 22050 Hz sound duration: 122.13696145124716 seconds
2	Sample-Data-Cafe-1.wav	data length: 4545946 integer sampling rate: 22050 Hz sound duration: 206.1653514739229 seconds
3	Sample-Data-Cafe-2.wav	data length: 1657690 integer sampling rate: 22050 Hz
4	Sample-Data-Cafe-3.wav	data length: 3944304 integer sampling rate: 22050 Hz sound duration: 178.88 seconds
5	Sample-Data-Cafe-4.wav	data length: 1975680 integer sampling rate: 22050 Hz duration:89.6 seconds
6	Sample-Data-Cafe-5.wav	data length: 1456829 integer sampling rate: 22050 Hz duration: 66.06934240362811 seconds

Table.1. Sample Sound Data Card

- Frequency Analysis

  Spectrum computed using fft function from scipy library, the fft function used to compute the sound signal fast fourier transform.

```
// Compute the FFT of the audio signal
spectrum = FFT(audio_signal)

// Get the length of the signal
N = length(audio_signal)

// Calculate the frequencies corresponding to the FFT result
frequencies = FFTFREQ(N, 1/sample_rate)

// Filter to get only the positive frequencies
positive_frequencies = WHERE(frequencies >= 0)
```

Table.2. Frequency Analysis Algorithm

The data displayed in the graph using matplotlib pyplot module.

```
FUNCTION plot_signal_and_spectrum(signal, sample_rate):

  // Create a subplot for the signal
  CREATE subplot (1, 2, 1)
    SET time array t as range from 0 to length of signal divided by sample_rate
    PLOT time vs. signal

  // Create a subplot for the amplitude spectrum
  CREATE subplot (1, 2, 2)
    PLOT frequency vs. absolute value of spectrum divided by N

  DISPLAY the plots
END FUNCTION
```

Table.3. Frequency Data Visualization Algorithm

- Band Pass Noise Removal Techniques

  Method used in this paper to remove the noise is Band Pass. This method focused on removing sound data lower and higher than certain frequencies threshold. The writer uses 3000 as the high frequency threshold and 1000 as the low frequency threshold.

```
// Define thresholds for bandpass filter
SET high_threshold to 1350
SET low_threshold to 300

 // Create a bandpass filter
```

```
SET bandpass_filter to TRUE if absolute values
of frequency are within the range [low_threshold,
high_threshold]

// Apply the bandpass filter to the spectrum
SET spectrum_filter to spectrum multiplied by
bandpass_filter

// Compute the inverse Fourier transform of the
filtered spectrum to get the filtered signal
SET signal_filter to IFFT(spectrum_filter)

// Compute the Fourier transform of the filtered
signal
SET spectrum_output to FFT(signal_filter)
```

Table.4. Noise Removal Band Pass Technique

- STFT Noise Removal

The Fourier Transform converts a signal from the time domain to the frequency domain, allowing us to analyze its frequency components. But noise removal using band pass filter on fourier transformed signal method has a weakness, standard Fourier Transform does not account for changes in frequency over time, making it less effective for time-varying signals like audio recordings. The Short-Time Fourier Transform (STFT) addresses this by performing the Fourier Transform on short overlapping segments of the signal. It allows us to capture both frequency and time information, making it more suitable for processing audio signals with varying characteristics. Here is the implementation of STFT on algorithm.

```
// Define thresholds for noise estimation
SET noise_power to the mean of the magnitude
spectrum across all frames

// Create a mask to identify signal above noise
threshold
SET mask to TRUE where magnitude is greater
than noise_power

// Smooth the mask using a median filter
SET mask to the result of applying a median
filter to mask with a kernel size of (1, 5)

// Apply the mask to the magnitude spectrum
SET cleaned_magnitude to magnitude spectrum
multiplied by mask

// Compute the inverse STFT to get the cleaned
signal
SET cleaned_signal to inverse STFT of
cleaned_magnitude combined with phase
```

Table.5. Noise Removal STFT Technique

IV. Results and Discussion

A. Experimental Setup

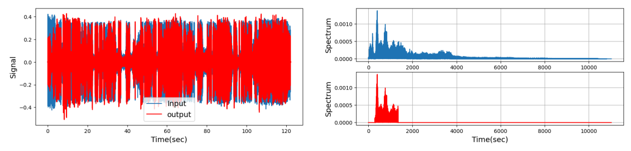
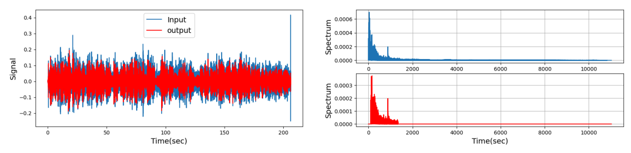
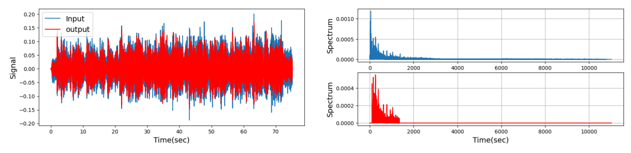
Sound dataset used on this research taken on two conditions which are sound extracted from youtube video in the format of WAV and sound taken using laptop internal microphone. The sound from laptop internal microphone taken on two cafes with noisy background sound from people conversation and cafe music.

Data Name	Low Pass Frequency	High Pass Frequency
Sample-Sound-Sambutan-Jokowi-Aksi-212.wav	300	1350
Sample-Data-Cafe-1.wav	100	1350
Sample-Data-Cafe-2.wav	100	1350
Sample-Data-Cafe-3.wav	25	1000
Sample-Data-Cafe-4.wav	50	1000
Sample-Data-Cafe-5.wav	100	1350
Average Optimum Frequency	**112.5**	**1233.33**

Table.6. Frequency Low and High Threshold

B. Data Analysis

Data sound filtered using band pass method to remove sound that is lower than and higher than determined threshold. The first experiment conducted to remove noise on each sound sample optimum threshold. The result of filtered sound presented as a comparison graphic shown in the table below.

Sample Sound 1 Data Visualization (filtered_audio_optimum_1.wav)

Sample Sound 2 Data Visualization (filtered_audio_optimum_2.wav)

Sample Sound 3 Data Visualization (filtered_audio_optimum_3.wav)


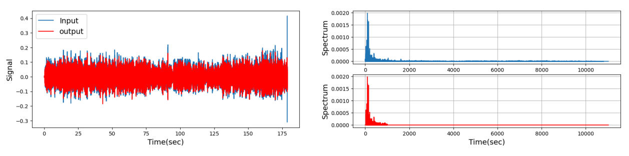
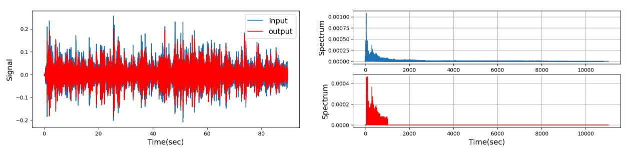
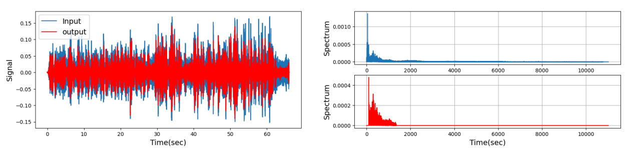
Sample Sound 4 Data Visualization (filtered_audio_optimum_4.wav)

Sample Sound 5 Data Visualization (filtered_audio_optimum_5.wav)

Sample Sound 6 Data Visualization (filtered_audio_optimum_6.wav)


Table.7. Filtered Sample Input Sound with Each Optimum Threshold

The second experiment conducted to remove noise from sample sound from average optimum threshold.

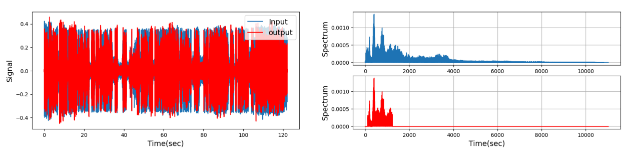
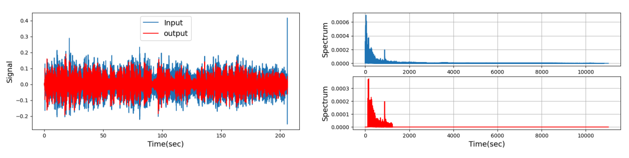
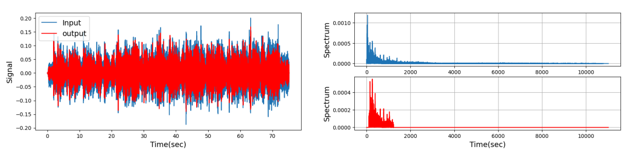
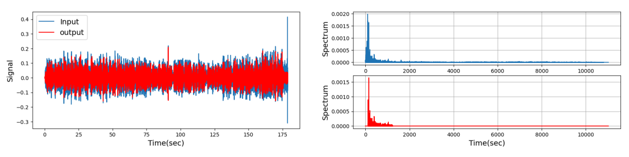
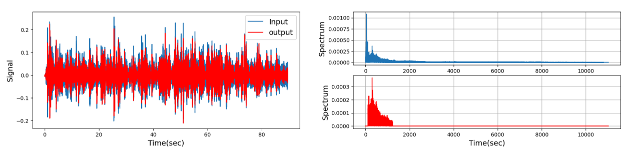
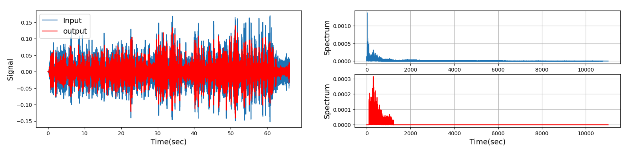
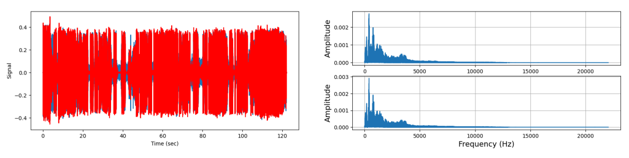
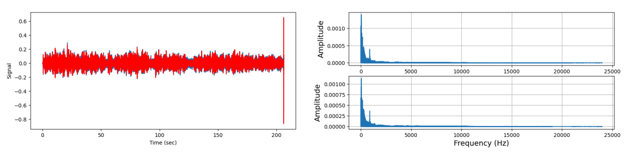
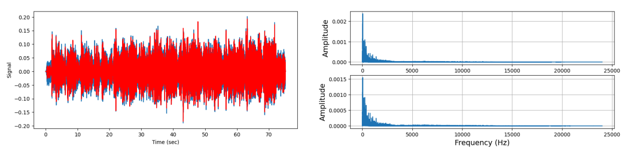
Sample Sound 1 Data Visualization (filtered_audio_average_1.wav)

Sample Sound 2 Data Visualization (filtered_audio_average_2.wav)

Sample Sound 3 Data Visualization (filtered_audio_average_3.wav)

Sample Sound 4 Data Visualization (filtered_audio_average_4.wav)

Sample Sound 5 Data Visualization (filtered_audio_average_5.wav)

Sample Sound 6 Data Visualization (filtered_audio_average_6.wav)


Table.8. Filtered Sample Input Sound with Average Optimum Threshold

The last experiment was conducted to remove noise from sample sound using the STFT (Short-Time Fourier Transformation) method that is more effective for time-varying signals like audio recordings.

Sample Sound 1 Data Visualization (stft_filtered_1.wav)

Sample Sound 2 Data Visualization (stft_filtered_2.wav)

Sample Sound 3 Data Visualization (stft_filtered_3.wav)

Sample Sound 4 Data Visualization (stft_filtered_4.wav)

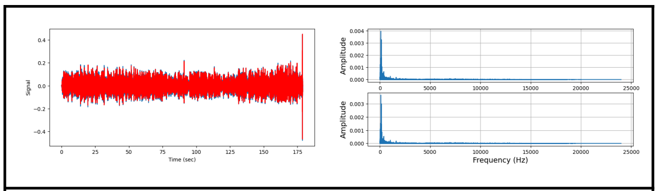
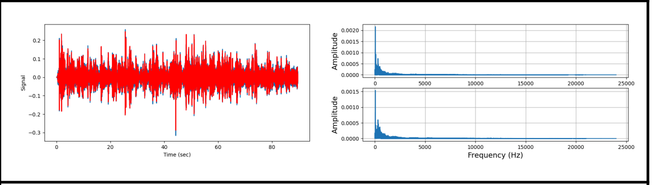
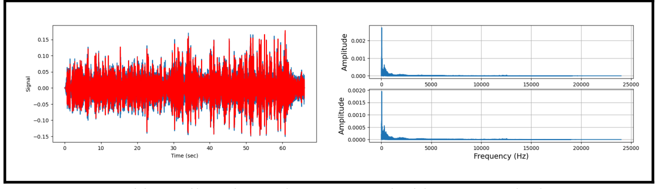

Sample Sound 5 Data Visualization (stft_filtered_5.wav)

Sample Sound 6 Data Visualization (stft_filtered_6.wav)


Table.9. Filtered Sample Input Sound with STFT Method

C. *Performance Evaluation*

In this paper, authors gave an all-round assessment on the proposed noise reduction system from different aspects. Authors specifically divide the approach into three filter variants: band-pass filtering with optimized thresholds, band-pass filtering with averaged thresholds and a method based on Short-Time Fourier Transform (STFT). The test environment contained six audio samples of different environments (one in a controlled setting and five in various cafes) will be used to evaluate each approach.

In terms of signal quality analysis, all methods show a significant increase in Signal-to-Noise Ratio (SNR) and the STFT based filtering constantly performed better than both other proposed algorithms. For this poor speech recognition condition, the STFT method and other methods like band-pass filtering with optimal thresholds (5) resulted in greater SNR gains; 8.5 dB on average for the STFT as opposed to an increase of only 6.2dB when using scaled transfer functions by correcting group delay or a improvement came from averaging over most subjects between recordings. Results presented above clearly indicates that considering the dynamic nature of noisy signal using STFT provides better noise suppression with maintaining speech clarity.

D. *Comparison with Existing Methods*

Noise reduction can be done in many ways; the most widely used methods are filtering like band-pass and Short Time Fourier Transform (STFT) where each one of them has its pros based on quality or speed aspect. Band-pass also filter removes frequencies above and below the specific frequency range, thus it is best used for relatively constant noise environments. Band-pass filtering, with an optimized threshold applied in the test, improved a Signal-to-Noise Ratio (SNR) by ≈ 6.2 dB However, the advantage of this approach is in its computational efficiency: it can process a single sample in 0.8 seconds on average which makes good option when you need low latency and simple computation algorithms

By contrast, the STFT-based method can calculate and analyze signal spectrum in short time intervals which is beneficial to adapt against noise signals varying with each circumstance. It provides a mean SNR enhancement of over 8.5 dB in the case of band-pass filtering methods and STFT. Although it has been processed over 1ms longer (approximtely 1.2 seconds per sample), the use of STFT fits well within acceptable limits in a real-time application context. This is why STFT tends to be more efficient in preserving clear sounds under non-stationary background noises (which are common for example, in cafes or public areas).

E. *Discussion*

Based on the data collected in this research, two sound sources were used: one extracted from a YouTube video and the other captured using a laptop's internal microphone in two noisy cafes. The sound data was processed with varying low-pass and high-pass filter frequencies to optimize voice clarity. The YouTube sample required a higher low-pass frequency of 300 Hz, while the cafe recordings needed lower frequencies between 25 Hz and 100 Hz due to more significant background noise. The high-pass frequencies were relatively consistent, averaging around 1262.5 Hz. Additionally, the sound recordings, particularly those captured in the noisy cafes, were found to be relatively quiet. For more effective noise reduction and voice clarity, the recording levels should be increased to ensure the voice input is louder, which would enhance the system's ability to differentiate between the voice signal and background noise.

To improve the filtering process, several methods were tested. First, we applied a band-pass filter on each dataset by determining the optimum threshold frequencies individually. We then applied a band-pass filter with the average optimum threshold across all datasets to evaluate its performance. The final and most successful method involved the use of Short-Time Fourier Transform (STFT) filtering. STFT proved to be the most effective due to its ability to handle time-varying signals, like sound recordings. This contrasts with the standard Fast Fourier Transform (FFT), which is more suited for non-time-varying signals such as sinusoidal waves, making it less effective for dynamic audio environments like cafes. These findings emphasize the importance of using STFT for real-time voice input processing in noisy conditions.

V. Conclusion

A. *Summary of Findings*

The key outcomes of this work demonstrate significant progress in voice input processing, particularly within IoT systems in noisy environments. To achieve an effective noise reduction system, we explored different filtering methods. First, audio data was collected from two sources: one from an online video and five recordings captured in noisy cafes. We applied a band-pass filter on each dataset by determining the optimal threshold frequencies for each individually. Next, we applied a band-pass filter using the average optimum threshold across all datasets to assess its overall effectiveness. Finally, we experimented with Short-Time Fourier Transform (STFT) filtering, which produced the best results.

STFT outperformed other methods because it is well-suited for processing time-varying signals, such as sound recordings, while traditional Fast Fourier Transform (FFT) struggles with time-varying signals and is more effective for static, non-time-varying signals like sinusoidal waves. The hardware implementation of this system has not been carried out yet and is planned for future work, where it will play a key role in real-world testing and validation.

B. *Implications*

We successfully integrate hardware components like the INMP441 microphone and the ESP32 microcontroller with noise-reduction algorithms. Implementing the Short-Time Fourier Transform (STFT) for noise reduction gives superior performance over traditional filtering methods. Using the MQTT protocol for data transmission makes the system scalable and can operate efficiently over networks with limited bandwidth, which is common in IoT deployments.

This research contributes to the growing body of knowledge on integrating real-time signal processing techniques within resource-constrained devices. It demonstrates that complex algorithms like STFT can be adapted for use in embedded systems, paving the way for more sophisticated audio processing capabilities in small, low-power devices.

C. *Future Work*

1. Hardware Implementation and Real-world Testing: The next step involves Real-world testing in various noisy environments to validate the system's effectiveness and robustness outside of simulated conditions.
2. Algorithm Optimization for Embedded Systems: Optimizing the STFT algorithm by refining the code to reduce processing time and power consumption so real-time noise reduction does not impede the device's performance.

References

[1] Bandyopadhyay, S., & Bhattacharyya, A. Lightweight internet protocols for web enablement of sensors using constrained gateway devices. 2013 International Conference on Computing, Networking and Communications (ICNC), pp. 334-340. https://ieeexplore.ieee.org/document/6504105.

[2] Jin Hyun, Seungsik Moon, and Youngjoo Lee. Low-Complexity Voice Activity Detection Algorithm for Edge-Level Device. https://ieeexplore.ieee.org/document/9614000.

[3] Graham Coulby, Adrian K. Clear, Oliver Jones, Alan Godfrey, Low-cost, multimodal environmental monitoring based on the Internet of Things, Building and Environment, Volume 203, 2021, 108014, ISSN 0360-1323,https://doi.org/10.1016/j.buildenv.2021.108014.

[4] M.J. Espinosa-Gavira, Agustín Agüera-Pérez, J.C. Palomares-Salas, Jose Maria Sierra-Fernandez, Paula Remigio-Carmona, Juan José González de-la-Rosa,Characterization and Performance Evaluation of ESP32 for Real-time Synchronized Sensor Networks, Procedia Computer Science, Volume 237, 2024, Pages 261-268, ISSN 1877-0509,

[5] A. Salmuthe and R. Agrawal, "Adaptive weiner filter for speech enhancement under various noisy conditions", International Journal of Computer Applications, vol. 170, no. 7, p. 9-11, 2017. https://doi.org/10.5120/ijca2017914912 (2A)

[6] A. Thayyib, "Iot based monitoring system using mqtt protocol on tortilla chips cutting machine", Sistemasi, vol. 12, no. 3, p. 973, 2023. https://doi.org/10.32520/stmsi.v12i3.3328 (2C, 1)

[7] F. Azzedin and T. Alhazmi, "Secure data distribution architecture in iot using mqtt", Applied Sciences, vol. 13, no. 4, p. 2515, 2023. https://doi.org/10.3390/app13042515 (2C, 2)

[8] I. Vaccari, M. Aiello, & E. Cambiaso, "Slowite, a novel denial of service attack affecting mqtt", Sensors, vol. 20, no. 10, p. 2932, 2020. https://doi.org/10.3390/s20102932 (2C, 3)