Development of a Fixed IoT-Based Outdoor Noise Monitoring System

Wesley Angelo O. Lim, Ernest Angelo D. Valencia, Jan Angelo R. Viaña
Electrical and Electronics Engineering Institute
University of the Philippines, Diliman
Quezon City, Philippines
{wesley.angelo.lim, ernest.angelo.valencia, jan.angelo.viana}@eee.upd.edu.ph

Abstract—**Noise is the unwanted sound emitted from daily human activities that can cause harmful health effects from prolonged exposure. Existing noise monitoring solutions are limited by the type of data they collect and the number of edge devices being deployed. This restricts the potential applications that could be developed in conjunction with the collected noise data. The developed monitoring system measured the continuous equivalent A-weighted sound pressure level (L_{Aeq}) for noise level quantification, and the Mel-frequency Cepstrum Coefficients (MFCC) were extracted to enhance the system's analyzing capabilities for further in-depth studies such as noise classification. Four sensor nodes were deployed, each comprised of an INMP441 microphone, a Teensy 4.1 for edge computing, and an ESP32 and LPSTK-CC1352R for wireless communication. Wi-Fi, Bluetooth Low-Energy (BLE), and ZigBee protocols were explored to determine which is the most suitable network to use for the system. Overall, the system exhibited reliable L_{Aeq} measurements with an average error of 0.22 dBA when compared to a PAA3x audio analyzer and an R^2 value of 0.9977, corresponding to a strong linear response. The MFCC computation achieved acceptable accuracy with efficient runtime despite hardware constraints. The system was successfully deployed using Wi-Fi in two outdoor locations with varying noise levels and Wi-Fi signal strengths, with payload transmission success rates ranging from 90% to 94%.**

Index Terms—**L_{Aeq}, MFCC, environmental noise monitoring, wireless sensor networks**

I. Introduction

Noise, defined as unwanted sound that disturbs human hearing, has become a significant environmental pollutant in urban areas due to increasing traffic, industrial activities, and human social interactions [1]. The World Health Organization (WHO) recognizes noise pollution as the second largest environmental cause of health problems after air pollution, with long-term exposure linked to cardiovascular and metabolic complications [2].

Global regulations govern acceptable noise exposure levels, with WHO recommending a maximum of 70 dBA for 40-hour weekly exposure [3]. In the Philippines, the Department of Labor and Employment limits workplace noise to 90 dBA for 80-hour work shifts, while the Department of Environment and Natural Resources maintains standards for general environmental noise [4].

Internet of Things (IoT) technology enables innovative, low-cost noise monitoring systems by integrating sensors, software, and network connectivity [5]. These systems must comply with international standards like IEC 61672 for sound level meters to ensure measurement reliability [6]. Previous implementations of IoT-based noise monitoring systems have demonstrated various approaches. One approach from a recent study by Cidro et al. was a single-node deployment that is able to calculate L_{Aeq} and MFCC simultaneously with the use of two microcontrollers executing the respective tasks [7]. The use of multiple MCUs and single-node deployment may still be improved for better efficiency and precision of the collected data. Another approach is the multiple node deployment by Vidaña-Vila et al., taking advantage of the physical redundancy of the sensors, enhancing its robustness and accuracy [8]. However, their sensors were not able to calculate either L_{Aeq} or MFCC values and was mainly used for acoustic event classification only. While these systems have shown promise in noise measurement, limitations exist in simultaneous processing of multiple acoustic parameters and continuous data transmission from distributed nodes. This work addresses these challenges through a distributed IoT-based noise monitoring system that collects and analyzes environmental noise data through multiple sensor nodes.

II. Review of Noise Monitoring Systems

A. L_{Aeq} and MFCC

L_{Aeq} represents the average sound level over time using A-weighting to approximate human hearing perception. Mello et al. [9] demonstrated L_{Aeq} measurement implementation using MEMS microphones and microcontrollers, establishing the viability of low-cost noise monitoring systems. However, their system had notable limitations: indoor-only deployment suitability, reliance on short-range Bluetooth transmission, lack of real-time cloud connectivity, absence of validation against professional sound level meters, and restriction to basic noise metrics without advanced analysis capabilities.

MFCC is a feature extraction method used for audio, particularly speech signals. It is based on the Mel scale which has equally spaced frequency bands which approximate the human auditory system [10]. Labied and Belangour [11] compared different feature extraction techniques with MFCC and their results indicate that MFCC is less complex and is effective for isolated word speech, however, it is sensitive to noise. Lastly, Räni [12] implemented MFCC for noise classification on a low-power microcontroller and concluded that its' feasibility, however, the MCU's processing speed affects the accuracy of the model.

Recent work by Cidro et al. [7] demonstrated an embedded noise monitoring system capable of computing both L_{Aeq} and MFCC measurements. Their implementation required two separate hardware boards to process these parameters simultaneously, highlighting the computational challenges in real-time acoustic analysis. While effective for single-point measurements, their system was limited to one sensor node and faced challenges in continuous MFCC data transmission.

B. Networks

Wi-Fi has been integrated into different industries of WSNs due to its high-speed communication without sacrificing reliability and transmission range [13]. However, these promising properties come at a cost of high-power consumption [14]. For low-power applications, BLE and Zigbee have been a staple communication protocol for WSNs. BLE is often used for energy efficient projects that only require compact data transmission [15]. Zigbee, on the other hand, is often used for low-power mesh network projects for higher scalability that does not require high-bandwidth applications [16]. These protocols are selected based on the specific requirements of the project, balancing factors such as power efficiency, scalability, and data transmission needs.

III. Problem Statement and Objectives

Noise pollution is a pressing but overlooked environmental and health issue in the Philippines, with limited studies and monitoring systems in place. The lack of affordable, comprehensive monitoring solutions has made it difficult to quantify and address this issue effectively.

This project develops a fixed IoT-based outdoor noise monitoring system that analyzes environmental noise through L_{Aeq} and MFCC computation, with sound recording capabilities for verification purposes. The system transmits data primarily through Wi-Fi to the UP-CARE platform, with experimental BLE and ZigBee connectivity. System validation is conducted across both high-noise and low-noise environments.

Through sustained monitoring and data collection, this system aims to provide cost-effective, real-time noise monitoring while serving as a foundation for future noise analysis implementations and evidence-based policymaking. Specific objectives include the following:

1) Deploy multiple synchronized sampling nodes for better coverage of a wider sampling range
2) Extract necessary audio signal features through the Mel-Frequency Cepstrum Coefficient technique as a form of audio compression
3) Compare the performance of different wireless communication protocols in a noise monitoring WSN
4) Transmit noise level data and the corresponding MFCC Coefficients to the UP-CARE platform

IV. Methodology

The noise monitoring system, shown in Figure 1, utilizes distributed sensor nodes, each equipped with three main components: a Teensy 4.1 for L_{Aeq} and MFCC computation, an ESP32 for Wi-Fi connectivity and SD card module control, and a LPSTK-CC1352R for BLE/Zigbee transmission. Each node incorporates dual INMP441 MEMS microphones—one interfacing with the Teensy 4.1 for signal processing, and another connected to the ESP32 for raw audio recording to an external SD card module. The sensor nodes transmit node identifiers, timestamps, L_{Aeq}, and MFCC values through multiple wireless protocols. When using Wi-Fi, data is transmitted directly via an MQTT broker to a CARE Server; alternatively, when using BLE/Zigbee, data is first aggregated through a gateway before reaching the MQTT broker. All collected data is visualized through a Grafana dashboard for real-time monitoring and analysis.

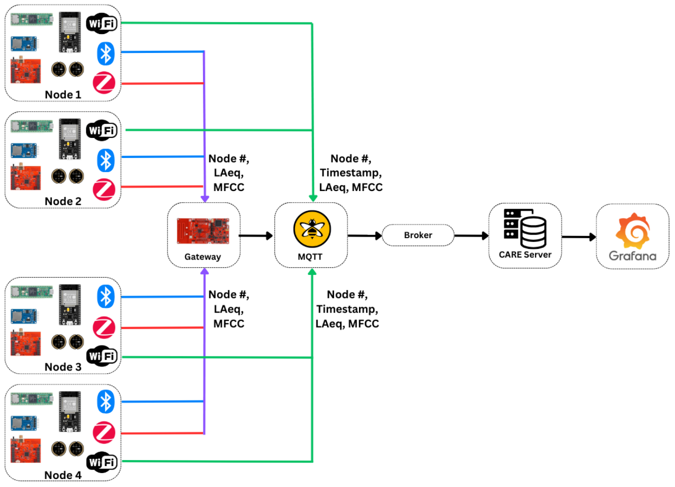


Fig. 1. Noise Monitoring System Diagram

A. L_{Aeq} Development and Audio Recording

The L_{Aeq} computation leveraged the Teensy audio library's design tools, implementing filter banks and A-weighting corrections based on ereuter's PJRC forum contribution [17]. The system collected measurements across nine octave bands (31.5 Hz to 8 kHz) with logarithmic averaging over 100ms intervals.

For audio recording, the ESP32's core 0 processes I2S input from the INMP441 microphone to create mono-channel WAV files at 44.1kHz sampling rate, with 60-second recording intervals. Recordings are timestamped and stored on an SD card with node identifiers for verification purposes.

System calibration was performed in a Whisper Room using a Phonic PAA3x Audio Analyzer as reference, with devices positioned 20cm from the source. Validation included ambient and pink noise measurements at multiple sound pressure levels, with linearity verified using 8kHz sine wave measurements at 5dBA increments.

B. MFCC

The MFCC computation in Teensy 4.1 was achieved by using the MFCC library created by Foued Derraz, which was available on GitHub [18]. The library featured functions of the individual processes required in computing MFCCs. For the required Fast Fourier Transform (FFT) operations, ArduinoFFT was imported from the available libraries in PlatformIO.

The sampling rate used was 44.1kHz, and the pre-emphasis constant was set to its typical value of 0.97. The frame length, which should be around 20-30ms, was set to 23.2ms, which is equivalent to 1024 audio samples. The Mel filter bank, whose size typically ranges from 20 to 40 filters, was set to 26 filters.

The MFCC task was set to continuously compute the MFCCs every 225 frames, which equates to 5.22 seconds. To start the MFCC computation, 1024 audio samples were acquired at a time from the microphone and were placed into a buffer to be normalized. Afterward, the buffer underwent pre-emphasis, hamming windowing, and application of FFT, Mel filter bank, and Discrete Cosine Transform (DCT). Finally, the resulting 13 coefficients are stored in an array. Each computed 13 coefficients constitute a single frame. This process was repeated 225 times to complete the 5.22 seconds worth of MFCC data.

C. Networks

1) Wi-Fi: Since the ESP32 handles the Wi-Fi transmission of data to the CARE database, the computed L_{Aeq} and MFCC were first received from Teensy using the UART protocol. To establish a synchronized sending and receiving of data, a handshake sequence was utilized at the beginning of the connection.

The NTPClient library was used to append timestamps to each data. A Geohash code was also included to provide the geographical location from which the data was collected. Additionally, node numbers were also assigned and added to identify the source of data. All of the mentioned information, along with transmission errors for L_{Aeq} and MFCC data which publish a value of 1 when errors occur, formed the payload that was being sent via MQTT. Publication of the payload to the MQTT broker via Wi-Fi were handled by the PubSubClient library.

In addition to the bare transmission of data, features were also added into the implementation of this network to improve its robustness. Worst-case scenarios, such as disconnections, were considered during the design process. Wi-Fi and MQTT reconnections were added into the implementation. Lastly, a signal strength scanning feature was integrated into the system that allowed each node to connect to an access point with the strongest signal strength relative to its position during initialization and reconnections.

2) Bluetooth Low Energy (BLE): The BLE peripheral nodes support dual operational modes: L_{Aeq} measurements and MFCC analysis. The CC1352R modules implement a proprietary BLE service protocol, facilitating transmission of either 32-bit floating-point L_{Aeq} values at 1-second intervals or 11,702-byte MFCC datasets. Data transfer between the Teensy and CC1352R components utilizes UART communication with a 7-byte packet structure, incorporating start/end markers and checksum validation mechanisms. For MFCC transmission, the system employs larger 244-byte chunks with integrated sequence numbers to ensure data integrity. The gateway infrastructure maintains simultaneous connections with up to 4 sensor nodes, coordinating device discovery and connection management protocols. This central component processes real-time L_{Aeq} measurements while continuously monitoring critical transmission metrics, including throughput rates, latency parameters, and packet loss statistics for MFCC data streams.

3) Zigbee: The TI CC1352R launchpad was used to implement the Zigbee protocol of the system. It uses Zigbee 3.0 certifiable stack that is compliant with the latest Zigbee standard which can be seamlessly integrated with other Zigbee devices. The launchpad has its own native software development kit that simplifies the development process. The Sonoff Zigbee Dongle-P served as the coordinator for the Zigbee devices. For direct access to MQTT services, it is integrated to Zigbee2MQTT which facilitates device configuration and enables the transmission of data to the server.

D. Data Monitoring and Visualization

To easily monitor the L_{Aeq} data computed by each node, a dashboard was created using the open-source platform, Grafana. A heat map was configured to see the L_{Aeq} value and visualize it at its exact position in the area of interest. In addition to the heatmap, time-series graphs were also added to monitor the transmission of the L_{Aeq} and MFCC data to the CARE database. These time-series graphs would display a value of 1 whenever a transmission error has occurred, which gives the users the ability to monitor the status of the system.

V. Results and Discussions

A. L_{Aeq} System Validation

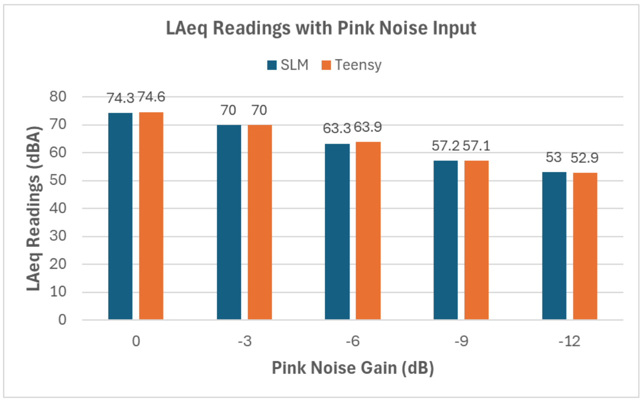


Fig. 2. Pink Noise

The system was validated through pink noise and linearity tests using a PAA3x audio analyzer as reference. For pink noise testing at various gains, the Teensy prototype demonstrated close values with the reference device, exhibiting an average error of only 0.22 dBA, well within the ±3 dBA human-perceptible difference threshold.

Linearity was evaluated using an 8 kHz sine wave with 5 dBA incremental increases from a 50 dBA reference. The prototype showed strong linear response ($R^2 = 0.9977$) with a slope of 4.93 compared to the reference's ideal slope of 5.0, and maintained an average error of 0.38 dBA across the tested range.

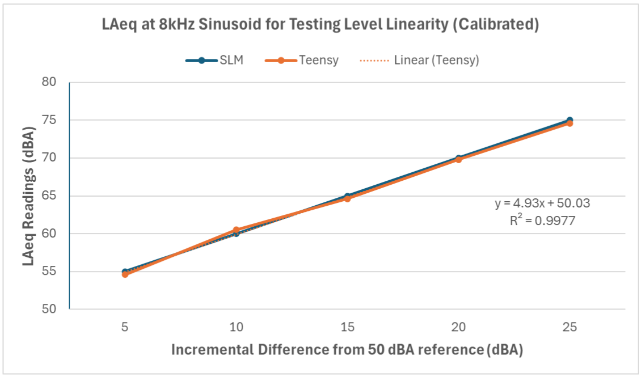


Fig. 3. Linearity Test

B. MFCC

In order to measure the performance of the MFCC algorithm developed for this system, it was compared with the MFCC implementation of MATLAB. The two were compared by varying the frequency of the input sinusoidal wave. The tested frequencies were 60 Hz, 600 Hz, 6000 Hz, and 15000 Hz. The frequencies were chosen because they covered the entire frequency range of INMP441.

TABLE I
MEAN PERCENTAGE ERROR AND NORMALISED ROOT MEAN SQUARE ERROR AT DIFFERENT FREQUENCIES

Sine Wave Frequency	MPE	NRMSE	NRMSE w/o 0th Coefficient
60Hz	154.5564014	0.4537524167	0.2122303374
600Hz	111.5918007	0.4419160456	0.2760441276
6000Hz	114.9961332	0.458267489	0.3617303239
15000Hz	381.834001	0.4458947418	0.3271052746

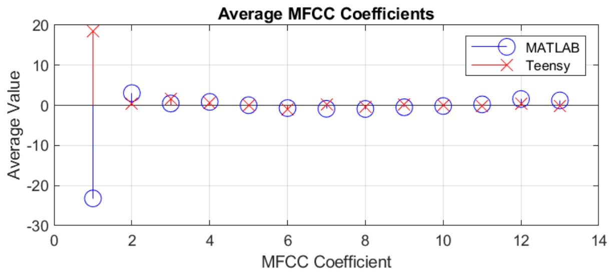


Fig. 4. Average MFCC Coefficients

As seen in Table I, the MPEs at different frequencies are relatively high, especially at the lower and higher end of the frequency range. On the other hand, the NRMSE across the frequencies are around 0.45, indicating that the error is 45% of the range of MFCC values from the MATLAB implementation. However, upon inspecting the values of each coefficient with a 600Hz sine input in Figure 4, the 0th coefficient contributes to the high error results. On the other hand, observing the values of the remaining 12 coefficients revealed that the values between the two implementations are significantly closer to one another. After excluding the 0th coefficient, the NRMSE dropped to 21-32%. Plotting the spectrograms also revealed the same observations, the 0th coefficient displayed the highest deviation when compared to its MATLAB counterpart. On the other hand, since the 0th coefficient of the MFCC contains no spectral information and is often discarded, this indicates that the use-case of MFCC as a means of data compression remains feasible.

The runtime performance of both implementations was evaluated, accounting for hardware specifications and audio input methods. For a 6-second sampling period, the MATLAB implementation achieved an average runtime of 26.102ms across 3 runs. The Teensy implementation averaged 483ms over the same number of runs. Despite the 457ms performance gap, the Teensy implementation demonstrated comparable efficiency, considering it operated on a 600MHz single-core microcontroller against MATLAB running on superior laptop hardware.

C. Networks

The system implements both Wi-Fi and BLE communication protocols, each offering distinct performance characteristics. For Wi-Fi communication between the Teensy 4.1 and ESP32, L_{Aeq} packet transmission requires 49.67ms, while MFCC data takes 2924.68ms.

Transmission to the MQTT broker from the ESP32 showed average latencies of 1.175 seconds for L_{Aeq} and 7.938 seconds for MFCC data. Power consumption during Wi-Fi operation was measured at 0.867W for L_{Aeq} and MFCC computation, with audio recording tasks consuming 0.9633W. The total system power consumption reaches 1W when all tasks are active. For memory utilization, the Teensy 4.1 consumed 141KB (13.52%) of its 1MB SRAM, while the ESP32 used 299KB (57.12%) of its 512KB memory when running both tasks.

For BLE communication, the system demonstrated consistent performance at a 4-meter range with 15-45ms communication intervals. Table II outlines the key metrics.

Memory utilization for BLE operation varies between components. The LPSTK device shows SRAM usage of 15,704 bytes (20%) for L_{Aeq} and 52,056 bytes (66%) for MFCC processing. The Launchpad CC1352R gateway utilizes 11,000 bytes (14%) and 34,808 bytes (44%) of SRAM for L_{Aeq} and MFCC respectively. Flash memory usage remains consistent at approximately 46% (162-163KB) on the LPSTK and 40% (142-144KB) on the gateway for both transmission modes.

The BLE implementation provides reliable connectivity with no transmission errors, though it cannot handle outdoor

TABLE II
BLE PERFORMANCE METRICS

Metric	L_{Aeq}	MFCC
Size	4B/sample	11,702KB
Rate	1/s	1.5810KB/s
Latency	1000.02±180ms	7.299s
Reliability	-	0% loss
Conn. Params	15-45ms	15-45ms
Power Consumption	0.483W	0.486W

noise monitoring applications and L_{Aeq} and MFCC transmissions simultaneously. This limitation aligns with the system's optimization for indoor deployment under typical noise level constraints.

D. *Deployment Results*

Two deployment locations were chosen to simulate different noise level conditions: E. Delos Santos St. (quiet location with high mobile data signal) and Roces Street – Osmeña Avenue intersection (high foot and vehicular traffic). Each deployment ran for 30 minutes with 3 Wi-Fi access points configured.

1) E. Delos Santos Street: In Figure 5, the time series of the nodes are superimposed into a single plot. It can be seen that the spikes and dips in the plot are nearly identical within the four nodes. It indicates that the nodes sample the same noise event at close or exact timestamps of one another. The signal strength of the three access points also proved to be high since only a few time skips were observed.

There is an observed correlation between the transmission errors of MFCC and L_{Aeq}, which can be seen in the error plots in Appendix C of this paper. This is because any significant delays in the transmission of the MFCC to the MQTT broker affect the transmission of the L_{Aeq} that follows it. Additionally, most of the error events occurred during the first 5 minutes of the deployment. Afterward, the number of errors dropped significantly. The average success rate in the transmission of MFCC was 94.1% while the L_{Aeq} was 93.2%. The high success rate is primarily attributed to the strong mobile data connection present in the location.

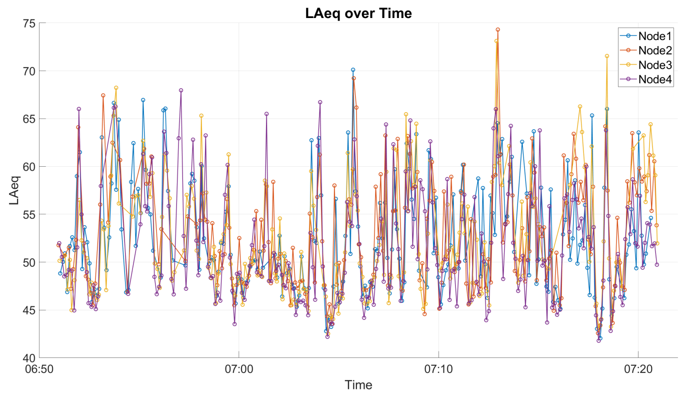


Fig. 5. Superimposed L_{Aeq} Readings in E. Delos Santos Street

2) Roces Street – Osmeña Avenue intersection: The L_{Aeq} readings of the four nodes were mostly identical to each other, as seen in Figure 6. However, due to the weaker mobile data connection in this location, disconnections occurred on multiple nodes. The node closest to the strongest access point had the most consistent data. At the halfway mark of the deployment, the data transmission appeared to be more consistent for all of the nodes because they were connected to the strongest access point by that time.

The error plots for the L_{Aeq} and MFCC for the second deployment location can be viewed in Figure 18 in Appendix C. Due to more transmission errors, the success rates for MQTT transmissions were lower compared to the first deployment location. The transmission of MFCC was at 92.5% while the L_{Aeq} was at 90.8%.

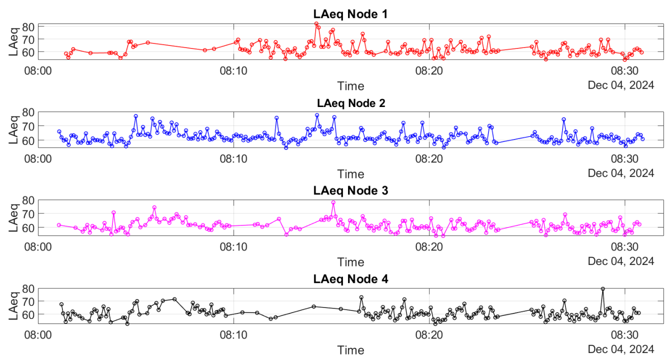


Fig. 6. L_{Aeq} Readings in Roces-Osmeña Intersection

Comparing the measured noise levels between the two locations, it was expected that the readings at E. Delos Santos would be lower due to it being a low-traffic area. The measured L_{Aeq} ranged from 41.78 to 74.31 dBA and had an average of 52.397 dBA. On the other hand, the readings at Roces-Osmeña intersection were higher with a mean L_{Aeq} of 61.714 dBA and ranged between 52.55 to 82.30 dBA.

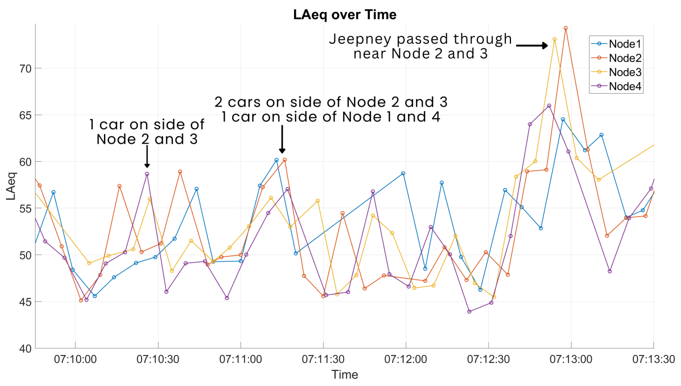


Fig. 7. 3-Minute L_{Aeq} Readings in E. Delos Santos Street

Zooming into the plot for further analysis, the 3-minute time series shows that the spikes and dips are connected to the event happening near the nodes as seen in Figure 7. For example, the first event at which a car passed near nodes 2 and 3 resulted in a spike of readings through the respective nodes. The second event, in which cars passed near all nodes, resulted in identical spikes throughout all the nodes as they picked up all the noise from the passing cars. The readings peaked when a loud jeepney passed near nodes 2 and 3, resulting in an almost 75 dBA reading through the respective nodes.

E. Roadblocks

1) BLE: Several limitations were encountered during BLE implementation using the SIMPLELINK-LOWPOWER-F3-SDK with TI-RTOS. Concurrent transmission of L_{Aeq} and MFCC data proved unfeasible due to MFCC's large payload size exceeding BLE's efficient transmission capabilities, combined with unresolved task scheduling implementation in TI-RTOS. The gateway functionality using LAUNCHXL-CC1352R1 could not relay data to the UP CARE Server without reverting to redundant Wi-Fi transmission through ESP32. Additionally, the LPSTK-CC1352R's inability to access an NTP server prevented accurate timestamping of measurements over BLE communication.

2) Zigbee: Attempts to transmit L_{Aeq} and MFCC values between the CC1352R launchpad (end device) and another launchpad (coordinator) were unsuccessful due to Code Composer Studio's lack of monitoring capabilities for error checking and debugging. To mitigate potential communication errors, a Sonoff Zigbee Dongle-P, supporting Zigbee 3.0 like the CC1352R, was implemented as the coordinator after being flashed with Z-stack firmware based on CC2652P/CC1352P2. Z2M served as the bridge between devices, hosted on Home Assistant with direct UP CARE MQTT server access.

Testing revealed the launchpad was incompatible with the coordinator, preventing dummy data transmission. Multiple troubleshooting approaches, including ZHA implementation instead of Z2M and manual CC1352R firmware extraction for dongle flashing, proved ineffective. While the TI forum provided additional suggestions and relevant discussions, progress was interrupted by the developers' Thanksgiving break. This led to the termination of Zigbee network development in favor of deploying alternative networks.

VI. Conclusion and Recommendations

This work successfully developed and validated a distributed IoT-based noise monitoring system capable of collecting environmental noise data through multiple synchronized sensor nodes. The system demonstrated reliable L_{Aeq} measurements with an average error of only 0.22 dBA compared to professional equipment, and achieved efficient MFCC computation despite hardware limitations. While the Wi-Fi implementation proved most successful with transmission success rates above 90%, the BLE implementation showed promise for low-power applications despite concurrent transmission constraints. The deployments in both quiet and high-traffic locations validated the system's ability to capture varying noise conditions, with nodes successfully synchronizing measurements and adapting to different signal strengths.

Several areas for future improvement emerged from this work. The Zigbee implementation faced technical challenges that could be addressed in future iterations. Additionally, the system could benefit from enhanced power optimization, improved MFCC computation efficiency, and more robust error handling during network transitions. Nevertheless, this work demonstrates the viability of low-cost IoT noise monitoring systems and provides a foundation for future urban noise pollution studies and evidence-based policymaking in the Philippines.

References

[1] K. M. de Paiva Vianna, M. R. A. Cardoso, and R. Rodrigues, "Noise pollution and annoyance: An urban soundscapes study," Noise & Health, vol. 17, pp. 125–133, 2015.

[2] A. Angelo, "Could everyday noise be affecting your health?," UC Davis Environmental Health Sciences Center, 2023.

[3] World Health Organization, "WHO-ITU global standard for safe listening devices and systems," WHO, 2019.

[4] "Philippine regulation on noise," Power City Electrical Supply Inc., 2015.

[5] P. Patil, "Smart IoT based system for vehicle noise and pollution monitoring," in Proc. Int. Conf. Trends Electron. Inform. (ICEI), 2017, pp. 322–326, doi: 10.1109/ICOEI.2017.8300941.

[6] "IEC 61672-1:2002 Electroacoustics - Sound level meters - Part 1: Specifications," International Electrotechnical Commission.

[7] A. C. Cidro et al., Development of an Embedded Noise Monitoring System with ESP32 and nRF52840. Philippines, 2023.

[8] E. Vidaña-Vila et al., "Multilabel acoustic event classification using real-world urban data and physical redundancy of sensors," Sensors, vol. 21, no. 22, p. 7470, Nov. 2021. doi:10.3390/s21227470

[9] F. Mello, W. Fonseca, and P. Mareze, "Autonomous noise monitoring system based on digital mems microphones: Development of a smartphone application for remote communication", Aug. 2022. DOI: 10.3397/IN_2022_0831.

[10] D Anggraeni, W. S. M. Sanjaya, M. Y. S. Nurasyidiek, and M Munawwaroh, "The implementation of speech recognition using mel-frequency cepstrum coefficients (mfcc) and support vector machine (svm) method based on python to control robot arm", IOP Conference Series: Materials Science and Engineering, vol. 288, no. 1, p. 012 042, 2018. [Online]. Available: https://dx. doi.org/10.1088/1757-899X/288/1/012042.

[11] M. Labied and A. Belangour, "Automatic speech recognition features extraction techniques: A multi-criteria comparison", International Journal of Advanced Computer Science and Applications, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:239089801.

[12] J. Räni, "Implementation of urban environment noise classification application on a low power microcontroller", [Online]. Available: https://www.etis.ee/Portal/Mentorships/Display/0172a8f9-d8fc-4b97-83cc-5dd440d01909.

[13] L. Li, H. Xiaoguang, C. Ke and H. Ketai, "The applications of WiFi-based Wireless Sensor Network in Internet of Things and Smart Grid," 2011 6th IEEE Conference on Industrial Electronics and Applications, Beijing, China, 2011, pp. 789-793, doi: 10.1109/ICIEA.2011.5975693.

[14] M. Khan, Ali, A. Khan, and M. Kabir, "Comparison among short range wireless networks: Bluetooth, zigbee, wi-fi", Advances in Computer Science and Engineering, vol. 4, pp. 19–28, Jan. 2016.

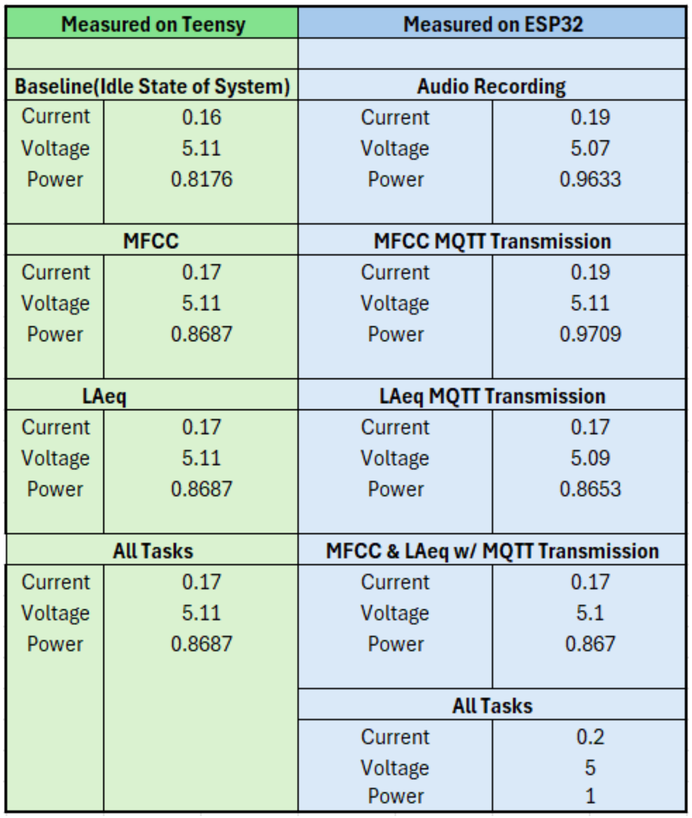
[15] J. Hughes, J. Yan, and K. Soga, "Development of wireless sensor network using bluetooth low energy (ble) for construction noise monitoring", International Journal on Smart Sensing and Intelligent Systems, vol. 8, no. 2, pp. 1379–1405, 2015. [Online]. Available: https: //doi.org/10.21307/ijssis-2017-811.

[16] Z. Rasin and M. R. Abdullah, "Water Quality Monitoring System Using Zigbee Based Wireless Sensor Network," International Journal of Engineering & Technology, vo l. 9, pp. 24–28, May 2012.

[17] ereuter, "Teensy 3.2 Sound Level Meter (sound measurement device)," PJRC Forum, Dec. 19, 2019. [Online]. Available: https://forum.pjrc.com/index.php?threads/teensy-3-2-sound-level-meter-sound-measurement-device.34371/post-224228.

[18] F. Derraz, "ArduinoMFCC," GitHub, April 24, 2023.[Online] Available: https://github.com/FouedDrz/arduinoMFCC.

Appendix A
Implementation Power Consumption



Measured on Teensy		Measured on ESP32	
Baseline(Idle State of System)		Audio Recording	
Current	0.16	Current	0.19
Voltage	5.11	Voltage	5.07
Power	0.8176	Power	0.9633
MFCC		MFCC MQTT Transmission	
Current	0.17	Current	0.19
Voltage	5.11	Voltage	5.11
Power	0.8687	Power	0.9709
LAeq		LAeq MQTT Transmission	
Current	0.17	Current	0.17
Voltage	5.11	Voltage	5.09
Power	0.8687	Power	0.8653
All Tasks		MFCC & LAeq w/ MQTT Transmission	
Current	0.17	Current	0.17
Voltage	5.11	Voltage	5.1
Power	0.8687	Power	0.867
		All Tasks	
		Current	0.2
		Voltage	5
		Power	1

Fig. 8. Full Power Consumption Testing of Implementation

Appendix B
MFCC Comparison of Teensy Implementation vs. MATLAB

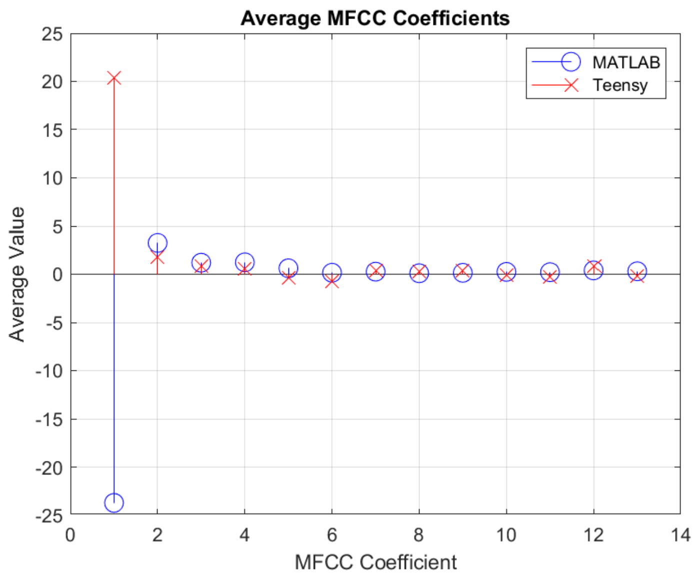


Fig. 9. Stem Plot Comparison of Average MFCCs with 60Hz Sine Input

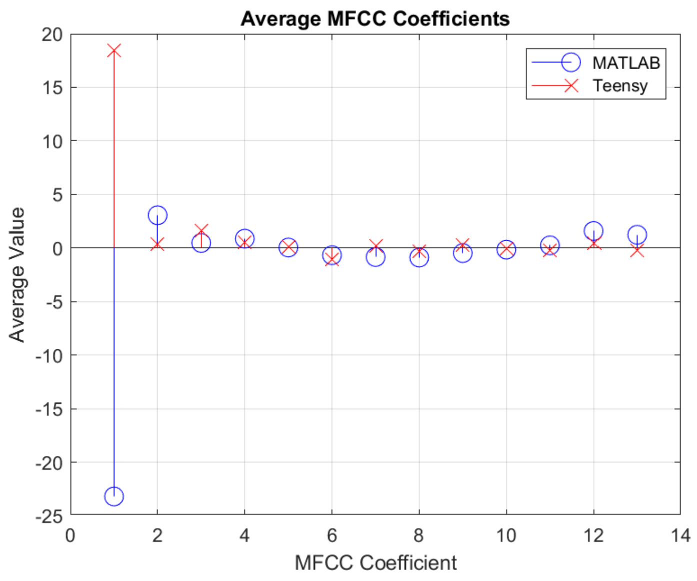


Fig. 10. Stem Plot Comparison of Average MFCCs with 600Hz Sine Input

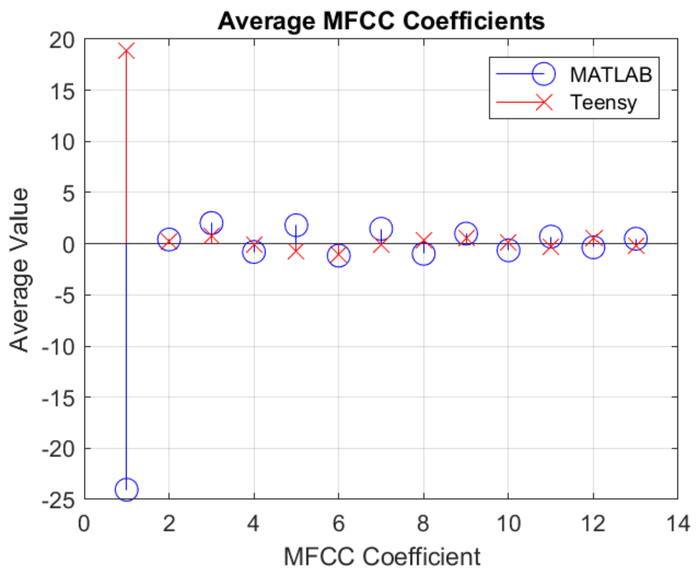


Fig. 11. Stem Plot Comparison of Average MFCCs with 6kHz Sine Input

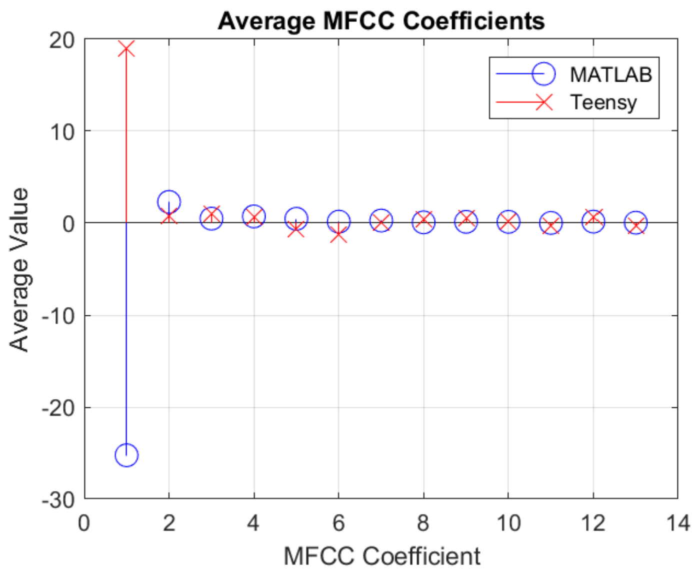


Fig. 12. Stem Plot Comparison of Average MFCCs with 15kHz Sine Input

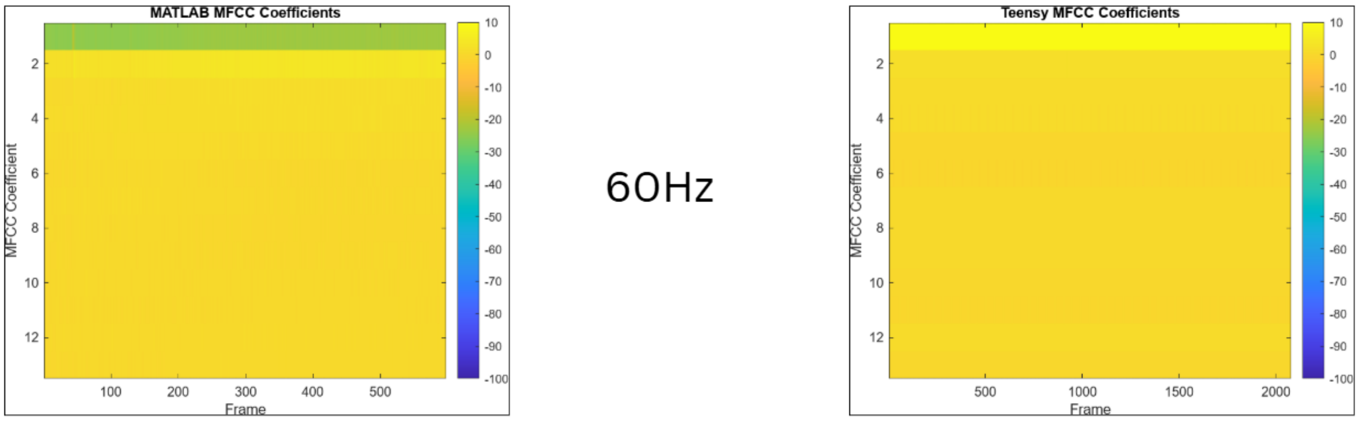


Fig. 13. MFCC Spectogram Comparison with 60Hz Sine Input

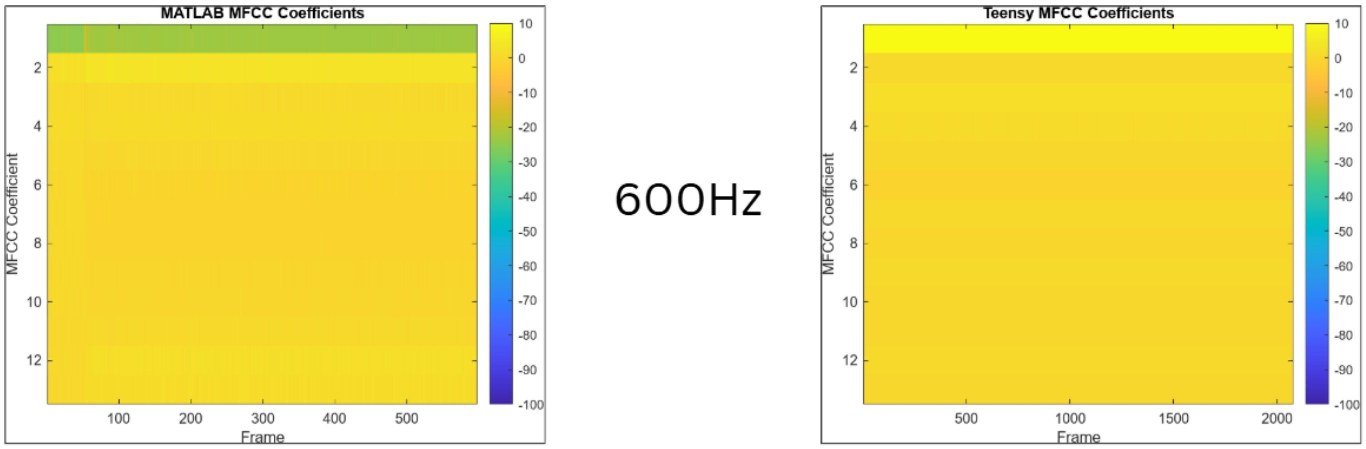


Fig. 14. MFCC Spectogram Comparison with 600Hz Sine Input

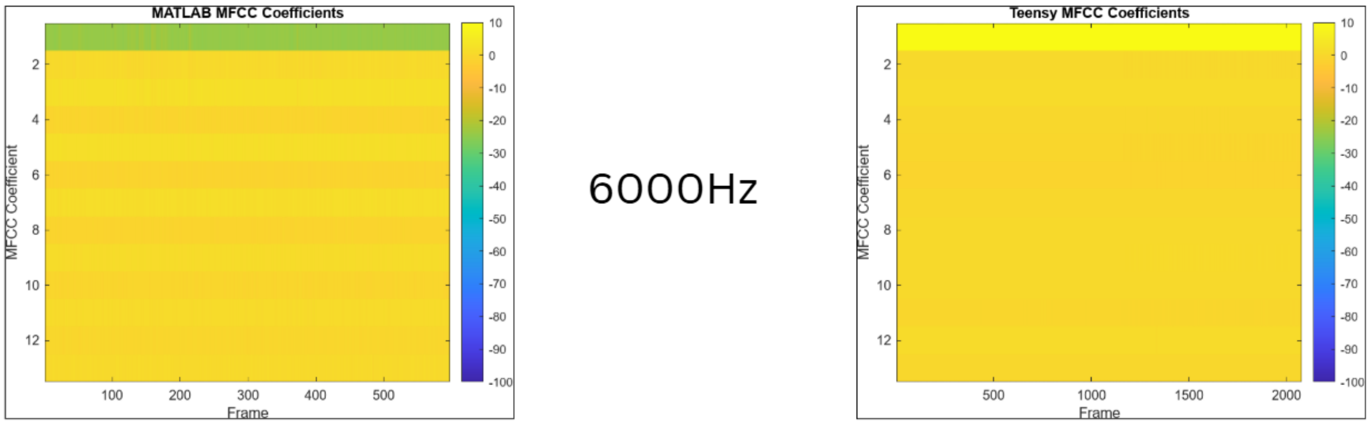


Fig. 15. MFCC Spectogram Comparison with 6kHz Sine Input

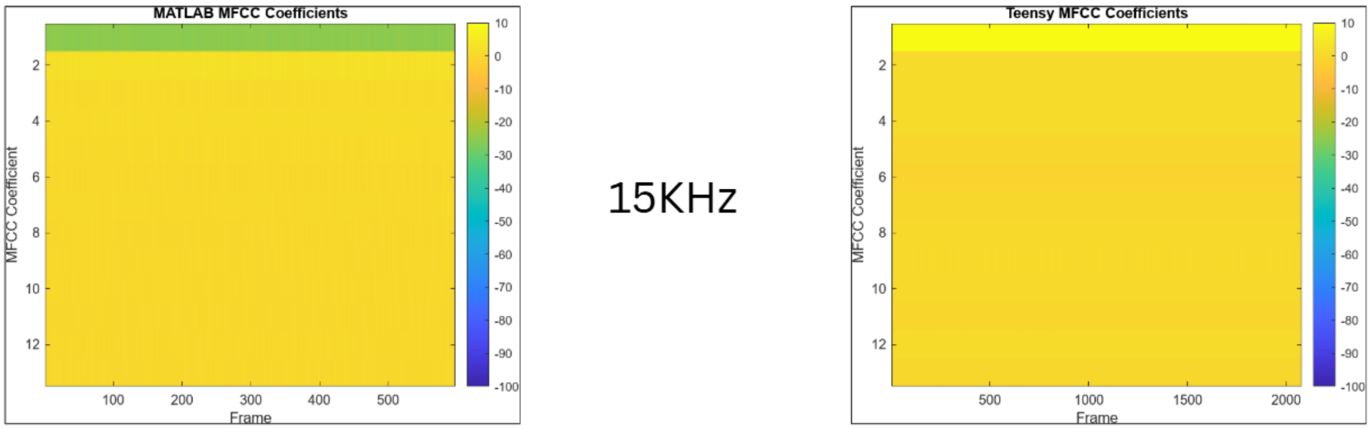


Fig. 16. MFCC Spectogram Comparison with 15kHz Sine Input

Appendix C
Additional Deployment Result Plots

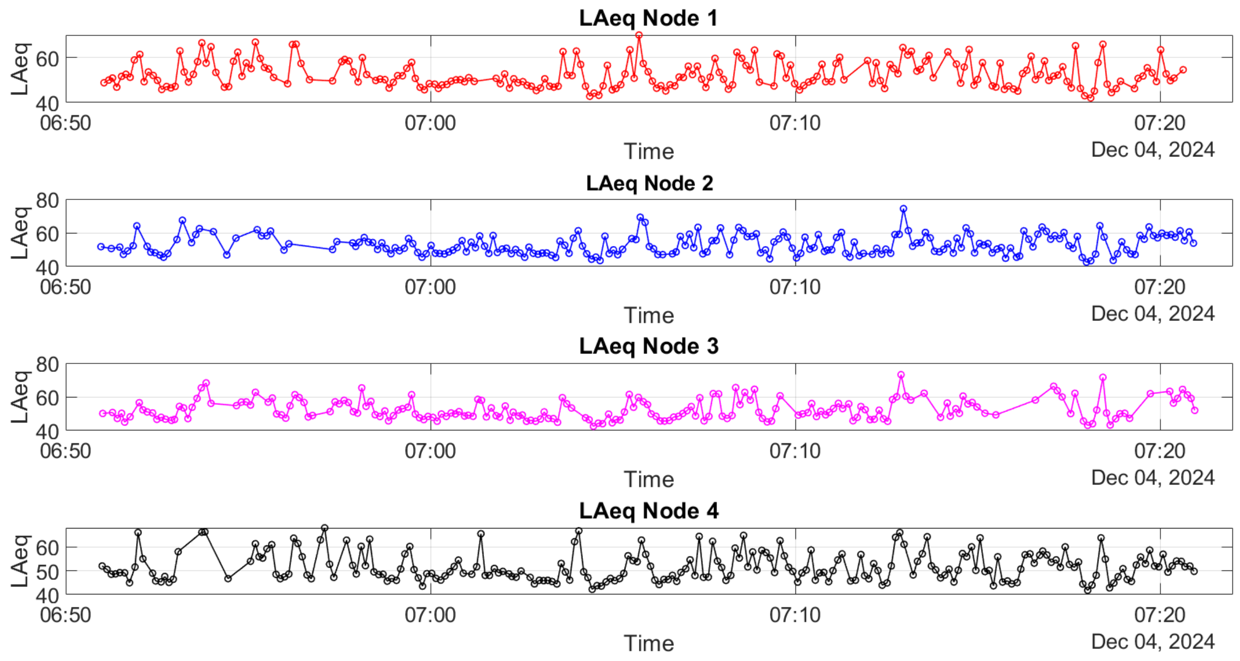


Fig. 17. L_{Aeq} Readings in E. Delos Santos Street

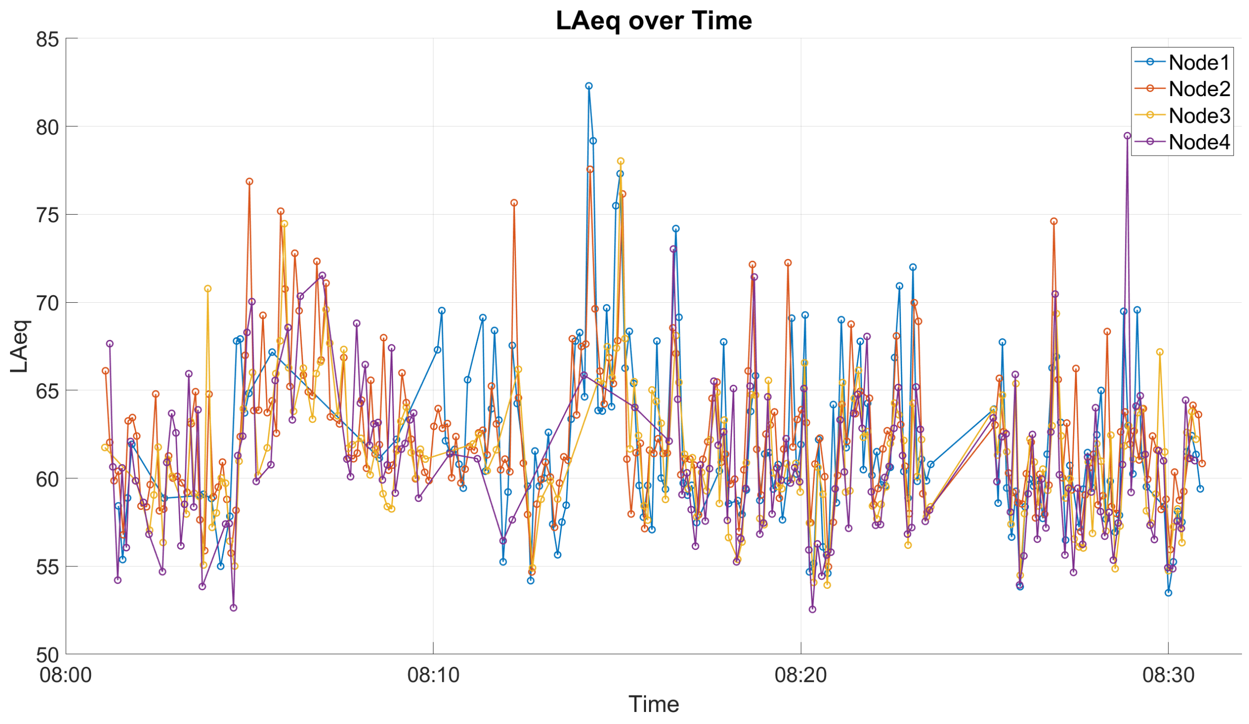


Fig. 18. Superimposed L_{Aeq} Readings in Roces-Osmeña Intersection

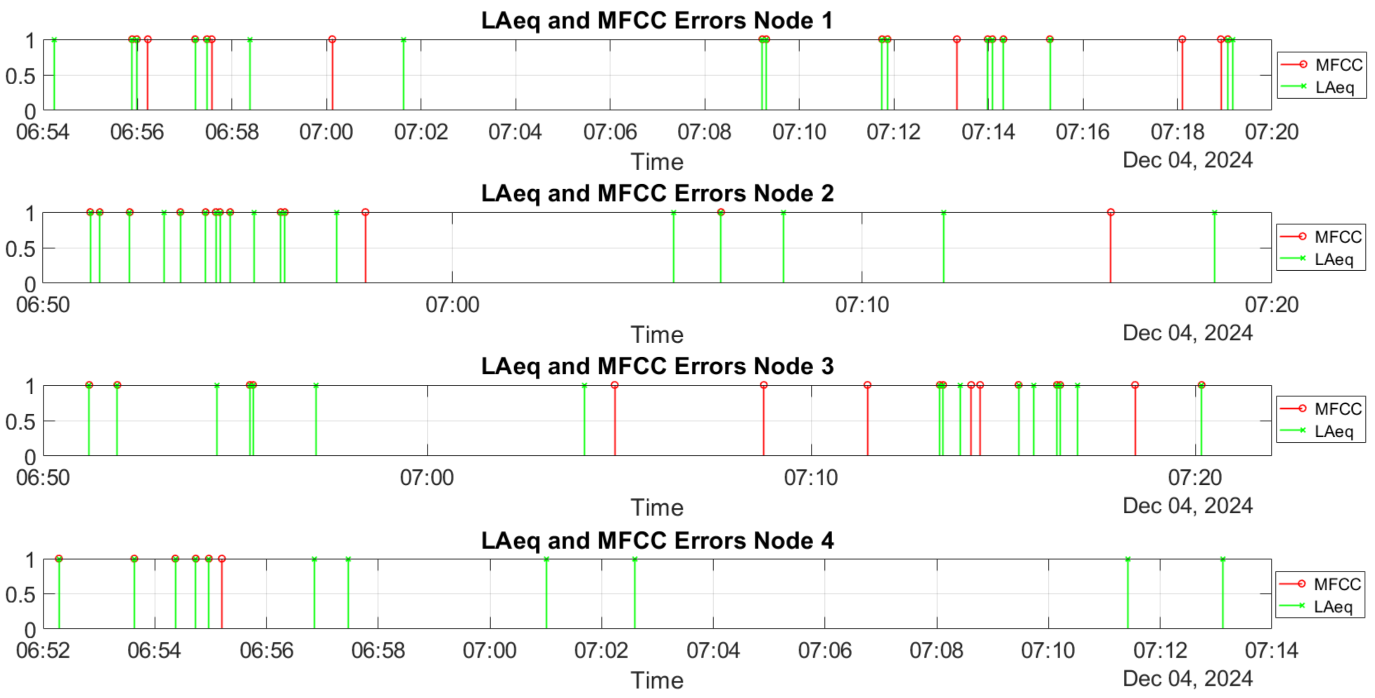


Fig. 19. L_{Aeq} and MFCC-error time series plot in E. Delos Santos Street

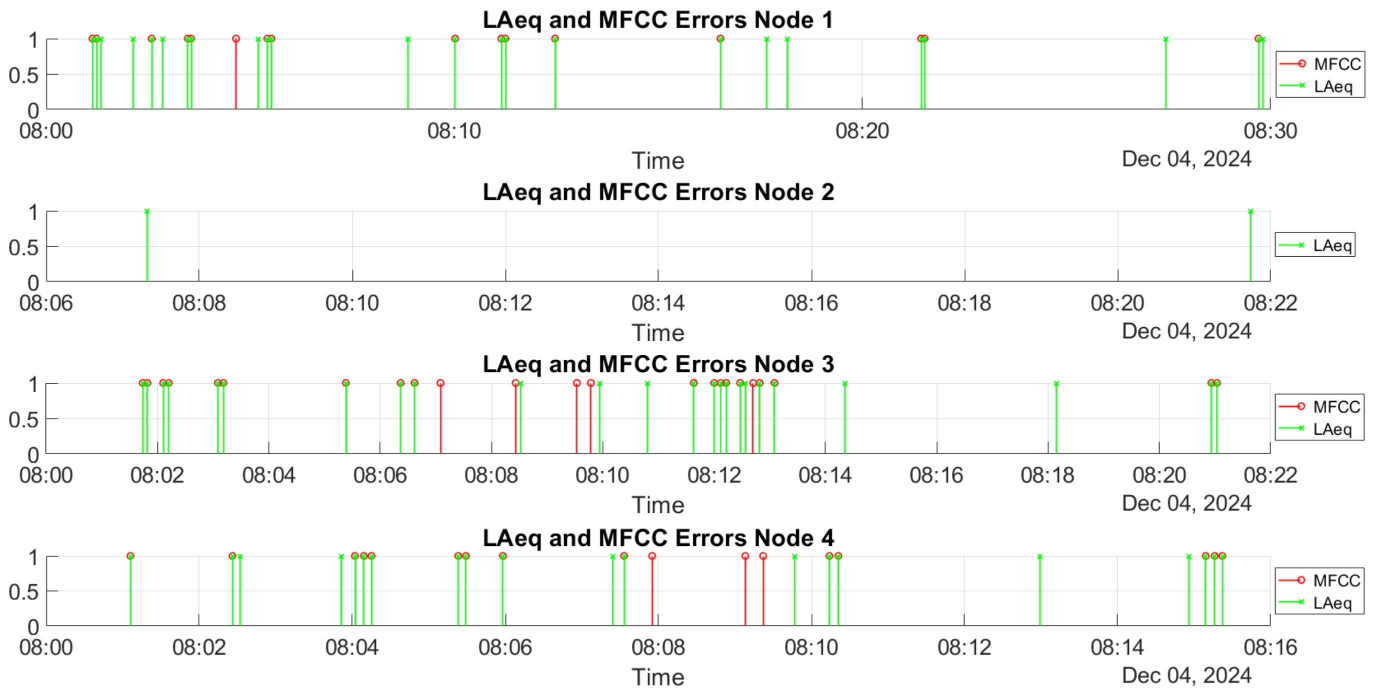


Fig. 20. L_{Aeq} and MFCC-error time series plot in Roces-Osmeña Intersection

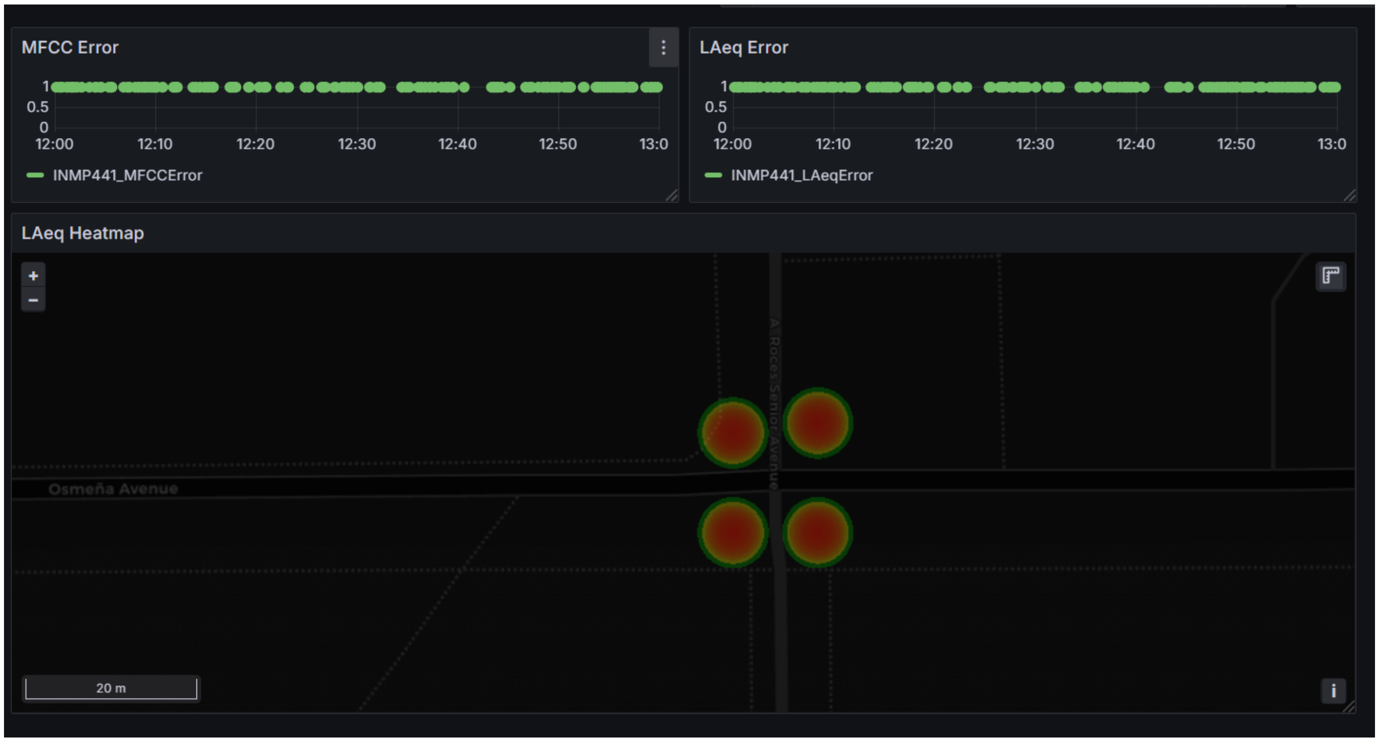


Fig. 21. Grafana Dashboard

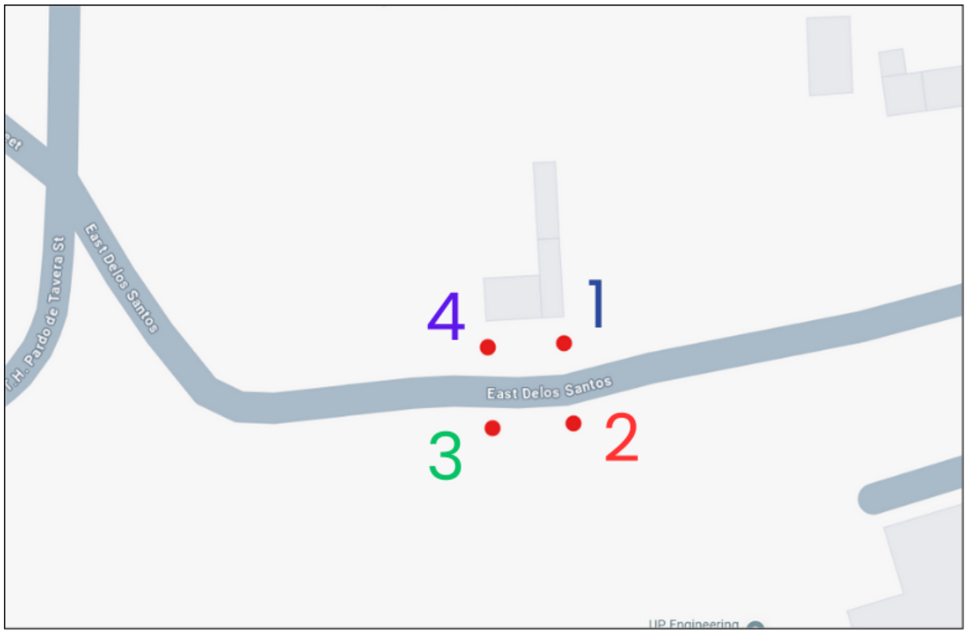


Fig. 22. Node Placements along E. Delos Santos Street

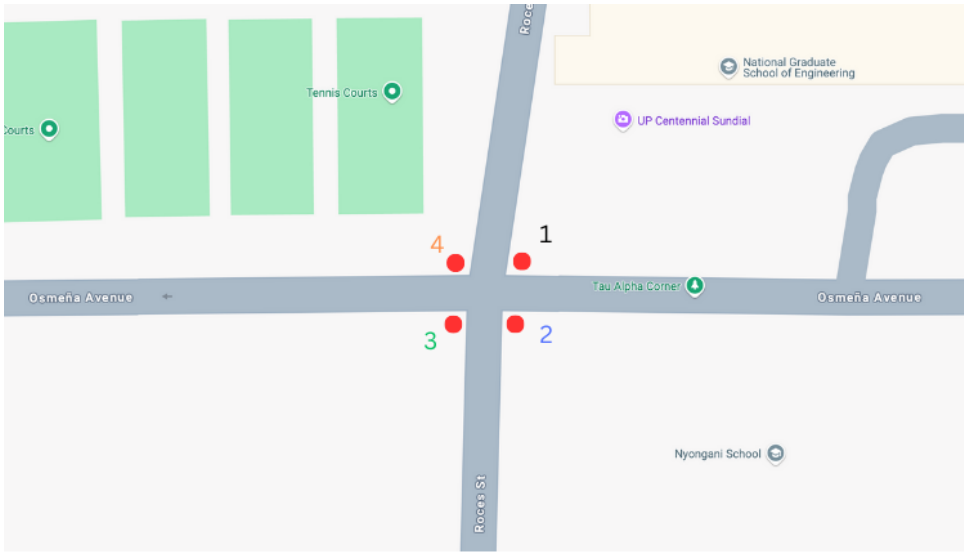


Fig. 23. Node Placements on the Roces-Osmeña Intersection