EEE3097S 2023 Assignment 1: Paper Design

1st Travimadox Webb
Department of Electrical Engineering
University of Cape Town
Cape Town, South Africa
WBBTRA001

2nd Rumbidzai Mashumba
Department of Electrical Engineering
University of Cape Town
Cape Town, South Africa
MSHRUM006

3rd Best Nkhumeleni
Department of Electrical Engineering
University of Cape Town
Cape Town, South Africa
NKHBES001

***Abstract*—This paper presents an optimized design for the "Acotriangulator," a technologically advanced system designed for the precise identification and localization of sound sources within a predefined A1 rectangular grid. The core design utilizes the Time Difference of Arrival (TDoA) methodology, enhanced by a strategic arrangement of four microphones, and employs a shared computational model involving two Raspberry Pi Zero devices and a more powerful main processing system. This unique configuration capitalizes on the Raspberry Pi Zero devices for data acquisition and preliminary preprocessing, while the main processing system undertakes complex computational tasks. Furthermore, the system design includes several distinct yet interlinked subsystems, each carefully tailored for specific operations. The paper elaborates on each subsystem's requirements, specifications, and testing procedures to provide an in-depth understanding of the system's efficiency, scalability, and reliability. It also reflects on an extensive feasibility study, a critical step in the project development that informed key design decisions.**

***Index Terms*—(TDoA),Acotriangulator**

I. Introduction

This paper offers a detailed overview of the improved design of the "Acotriangulator," a meticulously crafted system aimed at locating sound sources with precision within a specific A1 rectangular grid. The system employs a sophisticated application of the Time Difference of Arrival (TDoA) technique, a thoughtfully designed array of four microphones, and a distributed computational model consisting of two Raspberry Pi Zero devices and a main processing system.

The initial section of the document focuses on a comprehensive feasibility study that informed crucial design decisions. Detailed investigations evaluated potential technological solutions, leading to an optimized model where Raspberry Pi Zero devices handle data acquisition and preliminary preprocessing tasks. In contrast, a robust main processing system manages more complex computational tasks, thereby enhancing overall system efficiency and performance.

The document further delves into the nuanced system design, which comprises several specialized subsystems, each handling specific operations such as sound emission control, hardware setup management, data acquisition, data preprocessing, and complex computational procedures. Each subsystem is examined, with a thorough discussion of refined requirements, precise specifications, and robust testing protocols.

The document culminates with an overview of the User Interface and testing subsystem, a vital component that facilitates real-time visual tracking of the sound source's location on a 2D grid. This interface is designed to ensure ease of use and effective visualization for users, thus completing a comprehensive discussion of this innovative sound localization system.

II. Acoutriangulator Feasibility

This comprehensive feasibility study examines the possible configurations for implementing the "Acoutriangulator" system. The system uses time difference of arrival (TDoA) methodology, four microphones, two Raspberry Pi Zero devices, and a rectangular A1 grid for precise acoustic triangulation. The primary objective is to identify a configuration that can accurately determine the position of a sound source within the grid.

A. Analysis of Raspberry Pi Zero Configuration Scenarios

The feasibility study includes exploring various system configurations that utilize two Raspberry Pi Zero devices in different roles. These scenarios were evaluated for their advantages and disadvantages.

1) Raspberry Pi-Only Processing: Initially, a model where all computational tasks were performed solely by the Raspberry Pi Zero devices was considered. However, due to the devices' limited processing power and efficiency, this model was found to be less than optimal.

2) Shared Processing Models: An alternative shared processing model was examined where the Raspberry Pi Zero devices handle only preprocessing tasks, sending preprocessed data to a more powerful main processing system, such as a PC, for the actual processing. This model optimizes the use of the Raspberry Pi Zero devices' capabilities and increases the overall system performance. The shared processing models considered include:

1) **Distributed Sensor Processing:** Each Raspberry Pi Zero is connected to two microphones. They preprocess the audio data from these sensors and then send it to

the main processing system for further processing and triangulation calculations.

2) **Sensor and UI Separation:** One Raspberry Pi Zero captures and preprocesses audio data from all four microphones, while the other manages the user interface. The preprocessed data is then sent to the main processing system for triangulation calculations.
3) **Redundancy:** Both Raspberry Pi Zeros preprocess data from all four microphones, creating a mirrored setup. This redundancy allows for continuous operation if one device fails.
4) **Scaling:** Initially, one Raspberry Pi Zero handles all tasks. A second device is added only when needed for more demanding preprocessing tasks.

TABLE I
RASPBERRY PI ZERO CONFIGURATION SCENARIOS

Model	Advantages	Disadvantages
Distributed Sensor Processing	Optimal use of RPi Zero's processing power. Allows for distributed workload	Slightly complex implementation
Sensor and UI Separation	Clean separation of responsibilities. Lessens processing load per device	Depends on the reliability of a single RPi for input
Redundancy	Ensures system reliability and availability	Requires additional resources, adds complexity
Scaling	Scalable as per demands	Not efficient for smaller operations

B. Microphone Arrangement Scenarios

Various potential arrangements for the four microphones on the A1 rectangular grid were considered:

1) **Corners Arrangement:** Microphones are placed at each corner of the rectangular grid.
2) **Centre Arrangement:** All four microphones are centrally located within the grid.
3) **Edge Arrangement:** Microphones are placed at the midpoints of each edge of the rectangular grid.
4) **Random Arrangement:** Microphones are randomly placed on the grid.

Each arrangement's effectiveness was evaluated based on coverage, accuracy of results, and adaptability to various conditions as shown in Table II.

C. Time Synchronisation

Time synchronisation among multiple Raspberry Pi devices is crucial for achieving accurate Time Difference of Arrival (TDoA) calculations. A slight desynchronization between devices can introduce significant errors into the sound source localization system. Hence, after evaluating multiple techniques based on precision, ease of implementation, and reliability, the Network Time Protocol (NTP) was chosen for this purpose.

TABLE II
MICROPHONE ARRANGEMENT SCENARIOS

Arrangement	Advantages	Disadvantages
Corners Arrangement	Maximised coverage. High potential for accurate triangulation	Sound sources near the grid's centre may require more precise localization.
Centre Arrangement	Good for pinpointing sound originating from the centre. Easier to set up	Limited in detecting sound from further edges
Edge Arrangement	Balanced coverage, relatively easy to set up	Potentially less effective for centre sounds
Random Arrangement	Allows for flexible placement and potentially good coverage	Might lead to gaps in coverage, less predictable

1) Choosing the Network Time Protocol (NTP): NTP is one of the oldest and most widely used protocols designed to synchronise the clocks of computers over a network. It operates in a hierarchical manner, with a stratified system of time sources that provide a way to distribute accurate time.

Reasons for selection:

1) **Precision:** NTP can synchronise clocks with millisecond accuracy over the public internet and even better in local networks. This level of accuracy ensures that the time delay measurements between Raspberry Pi devices remain consistent and reliable.
2) **Scalability:** Given that our system comprises multiple Raspberry Pi devices, NTP's architecture, which can support large-scale networks, makes it an ideal choice.
3) **Reliability and Robustness:** NTP has built-in mechanisms to handle lost packets, jitter, and transient failures. It constantly adjusts and compensates for any drift, ensuring consistent synchronisation over extended periods.
4) **Ease of Implementation:** The Raspberry Pi OS, Raspbian, has in-built support for NTP, making its integration straightforward. Moreover, there's a vast amount of documentation, community support, and pre-built packages available for NTP, facilitating a smoother setup process.
5) **Network Resilience:** Even if a Raspberry Pi momentarily loses its connection to the network, NTP can still maintain accurate timekeeping by estimating drifts based on previously received data.

In conclusion, given the critical importance of time synchronisation for TDoA calculations, adopting NTP offers a balance of accuracy, reliability, and ease of integration, making it an apt choice for the Acotriangulator System.

D. *Recommended Configuration*

1) Raspberry Pi Zero Configuration: Following the comprehensive evaluation, the Distributed Sensor Processing model is recommended. In this setup, each Raspberry Pi Zero is dedicated to tasks within their capacity, such as preprocessing audio data and sending it to the main processing system for triangulation calculations. This configuration ensures scalability, resilience under various operational conditions, and efficient troubleshooting, permitting continuous operation even in the event of device failures.

2) Microphone Arrangement: The Corners Arrangement has been selected for the microphone setup on the rectangular grid, as it provides the most comprehensive coverage, thereby maximising the likelihood of accurately locating a sound source.

In conclusion, the Acoutriangulator system is proposed to use a Distributed Sensor Processing setup for the Raspberry Pi Zero devices and a Corners Arrangement for the microphones. These decisions aim to optimise system performance and efficiency and offer scalability and resilience under various operational conditions. The Raspberry Pi Zero devices will handle preprocessing tasks, sending preprocessed data to a more powerful main processing system for actual processing. The final decision regarding the optimal time synchronisation method is Network Time Protocol(NTP). This feasibility study provides a robust foundation for developing an efficient, resilient, and robust sound localization system within the given timeframe.

III. ACTUAL SYSTEM DESIGN

The primary goal of the Acoutriangulator system is to provide an accurate and reliable sound localization solution. This system integrates sound emission, hardware setup, data capture, processing, transmission, and localization into a unified structure. The following sections give a clear overview of each subsystem, its functions, and how they work together. The System Overview and Hardware Block Diagrams are shown in Figure 1 and 2 respectively below:

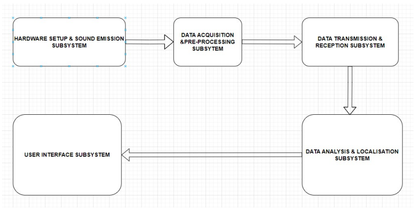
HARDWARE SETUP & SOUND EMISSION SUBSYSTEM
DATA ACQUISITION &PRE-PROCESSING SUBSYTEM
DATA TRANSMISSION & RECEPTION SUBSYSTEM
DATA ANALYSIS & LOCALISATION SUBSYSTEM
USER INTERFACE SUBSYSTEM

Fig. 1. System Overview Block Diagram

IV. HARDWARE SETUP AND SOUND EMISSION SUBSYSTEM

The Acoutriangulator's foundation is built on careful hardware setup combined with controlled sound emission. This combination ensures both reliability and precise management of auditory signals.

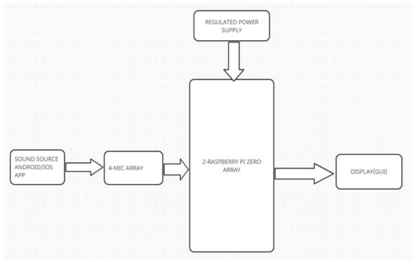

Fig. 2. Hardware Block Diagram

A. *Sound Emission*

The system's phone application serves as a control hub for sound emission, offering user-friendly access to crucial functionalities. A screenshot of the app in operation is shown in Figure 3 below:

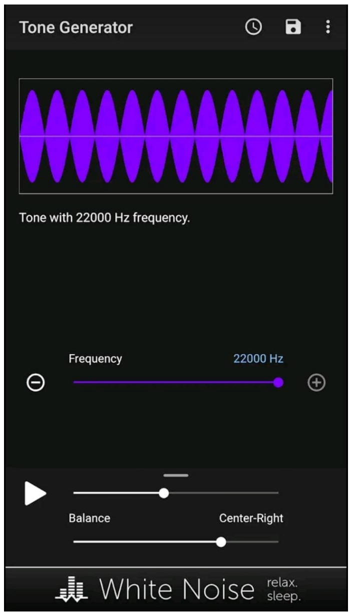

Fig. 3. Sound Emission

1) Requirements:

- Compatibility with contemporary smartphones
- Secure connection with the system

2) Specifications:

- Compatible with Android and iOS versions 7.1 or higher
- Utilises secure connection protocols

B. *Hardware Setup and Management*

This section delineates the key considerations, such as Raspberry Pi Zero device placement, microphone arrangement, and power considerations. The microphone layout is shown in Figure 4.

1) Requirements:

- Stable power source
- Optimal microphone placement

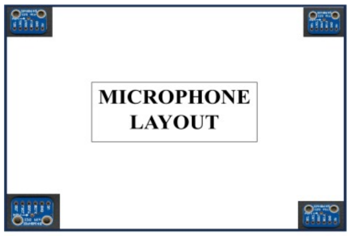

Fig. 4. Microphone Layout

2) Specifications:

- Utilises Raspberry Pi Zero models
- Specific power requirements detailed in Appendix XII-A

V. Data Acquisition and Pre-Processing Subsystem

The acquisition of synchronised, noise-filtered data is pivotal for accurate sound localization. This subsystem functions through three main phases: Time Synchronisation, Audio Acquisition, and Data Pre-Processing. Data Flow diagram of the subsystem is shown in Figure 5 below:

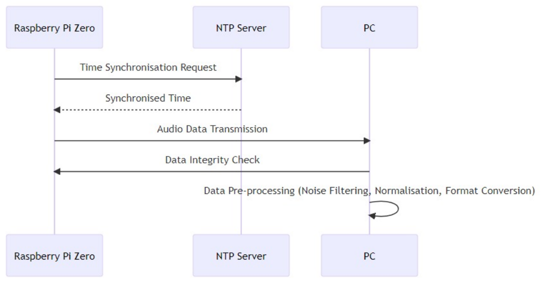

Fig. 5. Data Acquisition and Pre-Processing Data Flow Diagram

A. Time Synchronisation

In any Time Difference of Arrival (TDOA) sound localization system, accurate time synchronisation between participating devices is paramount. This synchronisation ensures that the timing measurements, on which the TDOA calculations are based, are consistent and reliable across all devices. The Network Time Protocol (NTP) has been selected for this purpose due to its ease of implementation, wide adoption, and satisfactory precision for our requirements.

1) Requirements:

- **Network Connectivity:** A stable network connection is crucial, as NTP relies on fetching time from external servers.
- **NTP Software:** The Linux operating system on the Raspberry Pi Zeros should have the NTP client software installed and properly configured.
- **Reliable NTP Servers:** Devices should be configured to synchronise to reliable NTP servers, preferably geographically closer for reduced latency.
- **Common Local Network:** To ensure minimal variance in synchronisation due to network latencies, devices should ideally be on the same local network.

2) Specifications:

- **Synchronisation Frequency:** The NTP clients on the devices will attempt synchronisation at regular intervals. This frequency can be adjusted based on the requirements.
- **Fallback Servers:** To ensure continued synchronisation even if one server becomes unreachable, a list of fallback NTP servers should be specified in the configuration.
- **Jitter and Drift Compensation:** NTP clients should be configured to automatically adjust for any detected clock drift and network jitter to maintain consistent synchronisation.
- **Logging and Monitoring:** For transparency and debugging purposes, NTP synchronisation activities should be logged. Monitoring tools can be set up to alert if synchronisation fails or if the time drift exceeds a specific threshold.

3) Implementation Details: **Setup:**

- All Raspberry Pi Zeros are connected to a singular local network.
- Each device is equipped with NTP client software, which is then configured accordingly.

Configuration:

- Firstly, the package repository is updated, followed by the installation of the NTP client:

```
sudo apt-get update
sudo apt-get install ntp
```

- To elevate synchronisation quality, all devices can be synchronised to an identical NTP server, or, if available, a local reference clock. This synchronisation can be accomplished by adjusting the `/etc/ntp.conf` file on each device to denote the desired NTP server addresses.
- Post configuration, the NTP service is restarted to embed the changes:

```
sudo service ntp restart
```

Through the judicious employment of NTP, we have crafted a robust framework for time synchronisation tailored for our TDOA sound localization system. This approach ensures broad-scale alignment, guaranteeing the reliability and precision of our localization results.

B. Audio Acquisition on Raspberry Pi Zero

Audio acquisition is a foundational phase in the TDOA sound localization system. This procedure ensures that sound is captured accurately and efficiently for further processing. The Raspberry Pi Zero, coupled with MEMS microphones, facilitates this phase. Given its compact nature, adaptability, and the available interface for MEMS microphones, the Raspberry Pi Zero emerges as an ideal candidate for this role.

1) System Components:

- **Audio Capture:** The system utilises MEMS microphones which are recognized for their sensitivity, reliability, and compactness. Their integration with the Raspberry Pi Zero ensures minimal signal loss and optimal sound quality.
- **Command Control:** The built-in command "arecord" on the Raspberry Pi Zero's Linux distribution is employed for audio capture. Specifically, the command `arecord -l` is utilised to list all the sound cards and digital audio devices, ensuring that our MEMS microphones are detected and functional.
- **Stereo Setup:** Given that two microphones are connected to each Raspberry Pi Zero, a stereo setup is implemented. This setup not only amplifies the sound capturing capability but also facilitates sound source localization by capturing sound from two distinct points.

2) Requirements/Specifications:

- **I2S Interface:** The Raspberry Pi Zero's I2S (Inter-IC Sound) interface should be enabled to facilitate the connection with MEMS microphones. In instances where I2S is disabled, a step-by-step guide can be followed here.
- **Sound Capture Properties:**
  - **Device Setting:** `-D plughw:0` indicates the primary hardware device for sound capture.
  - **Channels:** `-c2` represents a stereo recording due to the use of two microphones.
  - **Sample Rate:** `-r 48000` configures the sample rate at 48kHz, a standard for high-quality audio capture.
  - **Audio Format:** `-f S32_LE` sets the audio format to signed 32-bit little-endian.
  - **File Type:** `-t wav` ensures the captured audio is stored in WAV format.
  - **Volume Control:** `-V stereo` and `-v` command the system to display the volume levels for the stereo setup during the recording process.

3) Implementation: Once the prerequisites are met, audio can be captured from the MEMS microphones using the following command:*arecord -D plughw:0 -c2 -r 48000 -f S32 LE -t wav -V stereo -v filestereo.wav*.This command initiates the recording process, capturing audio in stereo from the two microphones and storing the resultant data in a WAV file named "file_stereo.wav".

The combination of Raspberry Pi Zero with MEMS microphones offers a compact yet efficient system for audio acquisition in our TDOA setup. Through the meticulous configuration and execution of the "arecord" command, the system captures high-quality stereo audio, laying a solid foundation for the subsequent phases of sound localization.

C. Data Pre-Processing in TDOA Sound Localization System

After acquiring audio data, the immediate step is its pre-processing. This phase is pivotal for ensuring that the acquired data is of the highest possible quality before being passed onto subsequent phases like localization calculations. The pre-processing phase undertakes tasks such as noise filtering, normalization, and format conversion to ensure data consistency and quality.

1) Components of Data Pre-Processing:

- **Noise Filtering:** Ambient environments usually introduce background noise which can impede the precise localization of sound sources. By employing specific algorithms, this undesired noise is systematically reduced or eliminated, ensuring that only the target audio signal is processed further.
- **Normalisation:** Given the variables in audio capture, such as distance from the source or interference, there might be discrepancies in the amplitude of the recorded signals. Normalization ensures that all audio signals are brought to a consistent amplitude level, making the data more homogenous and suitable for analysis.
- **Format Conversion:** Given the diversity of audio file formats and their respective properties, it is often necessary to have a standard format for further analysis. Should the acquired data not be in this predetermined format, it is converted to this standard to facilitate subsequent processing stages.

2) Requirements/Specifications:

- **Filtering Algorithm:** A software component or tool that can implement commonly used noise filtering techniques, like Spectral Subtraction or Wiener Filtering.
- **Normalisation Tools:** Software that can analyze the amplitude of the audio signals and adjust them to a consistent level without distorting the signal's inherent characteristics.
- **Format Conversion Software:** In case the audio data is not in the desired format, tools like FFmpeg or similar should be at hand to facilitate the conversion to the predetermined standard format.

3) Software & Libraries:

- `SoX`: A lightweight command-line utility suitable for audio manipulation on the Raspberry Pi Zero.
- `libsox-fmt-all`: This library extends the capabilities of sox, allowing it to handle a wide array of audio formats.

4) Implementation:

- **Noise Filtering:**
  - **Simple Band-pass Filtering:** Given that most ambient noises lie in lower or higher frequency ranges, a basic band-pass filter can be applied to retain only the frequencies of interest. This can be easily achieved using the Sox tool, a lightweight command-line utility suitable for the Pi Zero.
  - **Command:** `sox input.wav output_filtered.wav sinc 300-3.5k` This command retains frequencies between 300Hz and 3.5kHz, which is the general range of human speech. Adjustments can be made depending on the target audio's frequency range.

- **Normalisation:**
  - **SoX Normalisation:** sox can also be utilized for normalization, ensuring that audio clips have consistent amplitude.
  - **Command:** `sox input_filtered.wav output_normalized.wav gain -n` This command normalises the audio volume to 0dB.
- **Format Conversion:**
  - **SoX for Format Conversion:** Given its lightweight nature, sox is again recommended for format conversion. It can convert audio files into a plethora of formats.
  - **Example Command (to convert to MP3):** `sox output_normalized.wav output_final.mp3`

By harnessing the capabilities of simple, lightweight tools such as sox, the pre-processing of audio data on the Raspberry Pi Zero becomes efficient and straightforward. This approach ensures that the audio is of high quality and standardization without putting undue strain on the device's resources.

VI. Data Transmission and Reception Subsystem

The Data Transmission and Reception Subsystem serves as the conduit for transmitting acquired audio data from the Raspberry Pi Zero to a PC for further analysis. With the primary goal of ensuring the data's integrity and seamless transition, this subsystem employs secure communication protocols and verification techniques. Data Flow diagram for the subsystem is shown in Figure 6 below:

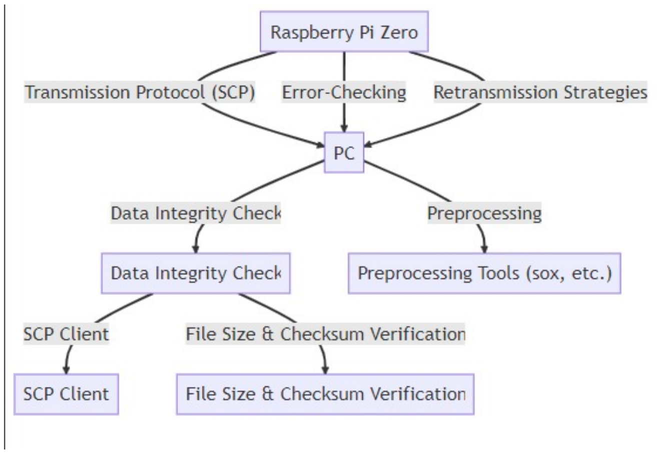
Raspberry Pi Zero
Transmission Protocol (SCP)
Error-Checking
Retransmission Strategies
PC
Data Integrity Check
Preprocessing
Data Integrity Check
Preprocessing Tools (sox, etc.)
SCP Client
File Size & Checksum Verification
SCP Client
File Size & Checksum Verification

Fig. 6. Data Transmission and Reception Subsystem Data Flow Diagram

A. Data Transmission

1) Components of Data Transmission:

- **Transmission Protocol:** A set of rules and conventions directing the data transfer from the Raspberry Pi Zero to the PC.
- **Error-Checking:** Mechanisms that confirm the data sent matches the data received.
- **Retransmission Strategies:** Provisions for situations when transmission encounters failures.

2) Requirements for Data Transmission::

- **Network Configuration:** Both the Raspberry Pi Zero and the PC should be on the same local network or should be reachable over a secured external network.

3) Software & Tools::

- **SCP (Secure Copy Protocol):** Part of the SSH package, this tool enables secure file transfers between devices.

4) Specifications for Data Transmission::

- **Transmission Protocol:** Secure Copy Protocol (SCP) is chosen due to its inherent security and ease of use with Raspberry Pi Zero and PC environments.
- **Error-Checking:** SCP, by nature, ensures data integrity during transmission. It will raise an error if the file transfer is not successful or if there's any data mismatch.
- **Retransmission Strategies:** In the unlikely event of a transmission failure, manual retransmission using the SCP command can be initiated.

B. Data Reception and Further Preprocessing

1) Components of Data Reception::

- **Data Integrity Check:** Ensuring the received data on the PC is complete and uncorrupted.
- **Preprocessing:** Modifying the received data into the required format for subsequent stages.

2) Requirements for Data Reception::

- **Reception Software & Tools:** SCP client on the PC, usually part of an SSH package.
- **Preprocessing Tools:** Depending on the desired format, tools like sox or other PC-compatible tools might be used.

3) Specifications for Data Reception::

- **Data Integrity Check:** After data reception on the PC, the file size and (if possible) a checksum can be cross-verified. SCP inherently ensures file integrity, but secondary checks provide added assurance.
- **Preprocessing:** Depending on the subsequent analysis stages and requirements on the PC, preprocessing might involve tasks like data normalisation, conversion, or filtering.

The Data Transmission and Reception Subsystem guarantees a secure and efficient flow of audio data from the Raspberry Pi Zero to a PC. By utilising secure protocols like SCP and implementing secondary integrity checks, this subsystem ensures the data remains untainted and primed for further analysis stages in the TDOA sound localization system.

VII. Data Analysis and Localization Subsystem

The Data Analysis and Localization Subsystem stands as the analytical core of the TDOA sound localization system. The main objective of this subsystem is to process the transmitted audio data, determining the Time Difference of Arrival (TDoA) and subsequently calculating the source position of the sound. The extracted data is transformed into actionable insights, shedding light on the sound source's precise location. Block diagram for the subsystem is shown in Figure 7 below:

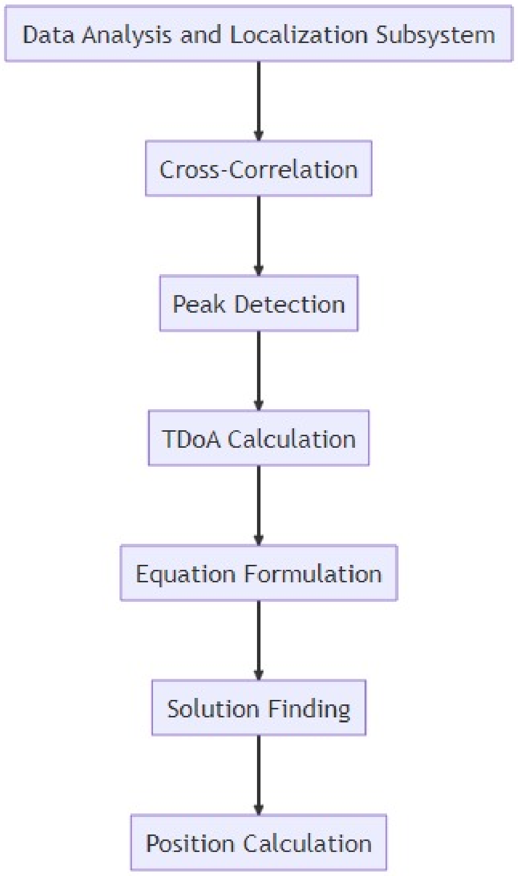

Fig. 7. Data Analysis and Localization Subsystem Block Diagram

A. TDoA Calculation

1) Components of TDoA Calculation:

- **Cross-Correlation:**A process that helps determine the delay between two signals by evaluating the similarity of the signals as they shift over each other.
- **Peak Detection:**fter cross-correlation, the delay corresponds to the location of the peak in the cross-correlation function.

2) Requirements for TDoA Calculation: Software & Tools:

- Libraries that support signal processing, such as SciPy or NumPy, for the chosen programming environment.
- A GCC-PHAT algorithm implementation or library compatible with Raspberry Pi Zero

3) Specifications for TDoA Calculation: Cross-Correlation: Utilises the GCC-PHAT method, a robust technique for computing the similarity between two signals irrespective of their amplitudes, which facilitates accurate delay estimation. Peak Detection: Once the cross-correlation function is computed, peak detection algorithms identify the maximum value, which represents the time difference of arrival.

B. Position Calculation

1) Components of Position Calculation:

- **Equation Formulation:**Transforms the obtained TDoA values into a set of mathematical equations representative of the sound source's potential positions.
- **Solution Finding:** Determines the exact position by solving the formulated equations.

2) Requirements for Position Calculation:

- **Mathematical Tools:**Libraries that support advanced mathematical operations, such as linear algebra packages in SciPy or other equivalent tools suitable for the analysis.

3) Specifications for Position Calculation:

- **Equation Formulation:**Utilising the TDoA values, equations are formulated based on the physical setup of the microphones and the mathematical representation of sound wave propagation.
- **Solution Finding:**The Least Squares method is adopted to provide an optimal solution to the equations, which narrows down the exact location of the sound source. This involves numerical methods which find the best-fitting solution to the system of equations that might not have an exact solution due to noise or other interferences.

Given a setup with 4 microphones in a square grid layout, finding the sound source's 2D position becomes a geometric problem. Let the four microphones be placed at corners of a square, $M_1(0,0)$, $M_2(a,0)$, $M_3(0,a)$, and $M_4(a,a)$, where a is the side length of the square.

4) TDoA Calculation: For a sound source $S(x,y)$ in the plane of the grid and the speed of sound c, the time taken for the sound to travel to M_i is:

$$t_i = \frac{\sqrt{(x-M_{ix})^2+(y-M_{iy})^2}}{c} \tag{1}$$

The Time Difference of Arrival (TDoA) between any two microphones, say M_i and M_j, is then:

$$\Delta t_{ij} = t_i - t_j \tag{2}$$

which can be expanded as:

$$\Delta t_{ij} = \frac{\sqrt{(x-M_{ix})^2+(y-M_{iy})^2} - \sqrt{(x-M_{jx})^2+(y-M_{jy})^2}}{c} \tag{3}$$

This gives us the time difference in terms of the sound source's coordinates.

5) Position Calculation Using Least Squares:: We can form hyperbolic equations for the position of S using the measured TDoAs. For every Δt_{ij}, an equation is created. We seek a position S that satisfies all these equations, and the Least Squares method aims to minimise the error for all equations collectively.

Given our microphone layout and using the TDoA equations derived above, we can set up a system of nonlinear equations:

$$\text{For } M_1 \text{ and } M_2 : \Delta t_{12} = \frac{\sqrt{x^2+y^2} - \sqrt{(x-a)^2+y^2}}{c} \tag{4}$$

$$\text{For } M_1 \text{ and } M_3: \Delta t_{13} = \frac{\sqrt{x^2+y^2} - \sqrt{x^2+(y-a)^2}}{c} \tag{5}$$

And so on for other microphone pairs...

The position (x, y) that best fits all these equations (in a Least Squares sense) gives the sound source's location.

6) Implementation: All this will be implemented in python language

The Data Analysis and Localization Subsystem stands as the decisive component in the TDOA sound localization system. By leveraging advanced signal processing techniques and mathematical modelling, this subsystem ensures that raw audio data is transformed into actionable insights, pinpointing the sound source's location with high precision.

VIII. User Interface Subsystem

The User Interface Subsystem acts as the point of interaction between the system and its users, streamlining the experience and ensuring that the data produced by the system is easy to comprehend.

A. Display Interface

The primary visual element of the GUI is a representation of a 2D grid, mirroring the microphone layout. The system will mark the calculated position of the sound source on this grid, providing an intuitive representation of the source's location relative to the microphones.

1) Specifications:

- **Resolution:** The grid will have a suitable resolution to ensure the accurate portrayal of the sound source location.
- **Dynamic Update:** The grid will update in real-time, reflecting any change in the sound source position instantly.
- **Visualisation:** The exact position of the sound source will be highlighted, potentially using a colour gradient to indicate the strength or clarity of the signal.

B. User Interaction

To make the system adaptable to different scenarios and user needs, the GUI will offer controls to adjust various system settings.

1) Specifications::

- **Adjustable Grid Size:** Allows users to modify the grid dimensions according to the physical microphone layout.
- **Calibration Settings:** Provides options to calibrate the system if there are any shifts in microphone positions or other parameters.
- **Threshold Adjustments:** Lets users set a threshold below which signals will be disregarded, aiding in noise handling.

C. Web Hosting

The interface is structured as a locally hosted website, allowing easy access from different devices within the network. This structure ensures flexibility and can accommodate remote observations and interactions.

1) Specifications::

- **Local Server:** The system will employ lightweight server software suitable for Raspberry Pi, such as Flask for Python.
- **Cross-Platform:** The web-based interface ensures compatibility across various devices, from PCs to smartphones.
- **Real-time Updates:** Leveraging WebSockets or similar technologies to ensure real-time data streaming to the interface.

D. Error Handling

Given the complexities involved, the system is designed to be resilient against failures, ensuring that users are promptly informed of any issues.

1) Specifications:

- **Alert System:** Pop-up notifications or banners to inform users of any errors or issues.
- **Diagnostic Tools:** A log display or console window to provide detailed error messages and potentially troubleshooting steps.
- **Redundancies:** If minor errors occur, the system will attempt self-correction before notifying the user.

E. Implementation:

For creating such a GUI, the combination of Flask (a Python micro web framework) and JavaScript can be particularly effective. Flask allows for easy backend integration, while JavaScript, combined with libraries like D3.js or Chart.js, can handle real-time data visualisation. For error handling, Flask can catch exceptions and forward them to the GUI. Additional steps can be taken to ensure robustness and user-friendliness, such as integrating user feedback mechanisms and regular system health checks.

IX. Acceptance Test Procedure

The full ATP can be found in the Appendix XII-B

X. Individual Contributions

The Table III below shows the contribution of each author to the above designs

Name	Contribution	Percentage
Travimadox Webb	Writing	50%
	Research	50%
	compiling	50%
Rumbidzai Mashumba	Compiling	50%
	Writing	25%
	Research	25%
Best Nkhumeleni	Research	25%
	Writing	25%

TABLE III
INDIVIDUAL CONTRIBUTIONS TABLE

XI. Development timeline

The Project development timeline is shown in Figure 8 below.

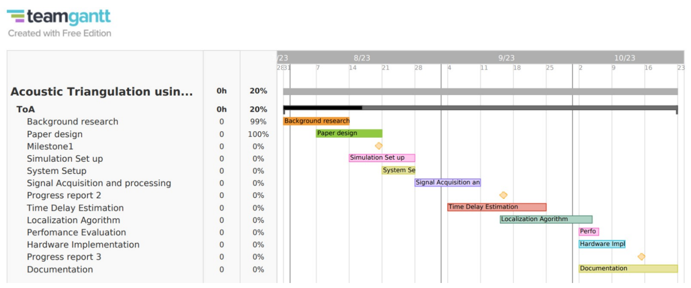

Fig. 8. Development Timeline

XII. Conclusion

In conclusion, this paper delineates our methodology pertaining to the utilization of Time Difference of Arrival (TDoA) for sound source triangulation. Our approach leverages an arrangement of Edge-based microphones and employs Distributed Sensor Processing. By introducing a sound with a predetermined frequency into our designated grid, we effectively employ TDoA principles to triangulate the sound's source. The fundamental premise lies in calculating the time discrepancies of the sound's arrival at each individual microphone. This positional information is subsequently exhibited through our web-based platform, which is hosted locally, to visually illustrate where our sound is being received from.

Bibliography

References

[1] Y. Yan, H. Wang, X. Shen, K. He, and X. Zhong, "Tdoa-based source collaborative localization via semidefinite relaxation in sensor networks," *International Journal of Distributed Sensor Networks*, vol. 11, no. 9, p. 248970, 2015.

[2] G. Simon, "Acoustic moving source localization using sparse time difference of arrival measurements," in *2022 IEEE 22nd International Symposium on Computational Intelligence and Informatics and 8th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (CINTI-MACRo)*, 2022, pp. 1–12.

[3] K. D. Frampton and J. L. Junkins, "Acoustic self-localization in a distributed sensor network," *IEEE Sensors Journal*, vol. 8, no. 2, pp. 165–172, 2008.

[4] Y. Meng, Y. Zhang, and C. Harrison, "Acustico: Tap detection and localization using wrist-based acoustic sensing," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI'21)*, 2021, pp. 1–13.

[5] D. G. Da Costa, "Asynchronous acoustic localization using time difference of arrival," B.Sc. Thesis, University of Cape Town, 2022.

[6] M. R. Bai, S.-S. Lan, and J.-Y. Huang, "Time difference of arrival (tdoa)-based acoustic source localization and signal extraction for intelligent audio classification," in *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 2766–2770.

[7] A. Patwari, S. Kumar, S. Kumar, and S. Kumar, "Real-time audio source localization using a raspberry pi and microphone array assembly," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 5, pp. 1972–1978, May 2020.

Appendix

A. Power Requirements for Raspberry Pi Zero W

The actual power requirement s of the raspberry pi to be used in the system are listed in Table IV below:

TABLE IV
Power Requirements for Raspberry Pi Zero W

Parameter	Value
Input Voltage (V)	5V
Recommended Current (A)	1.2A
Idle Power Consumption	120mA (0.6W)
Max Power Consumption	230mA (1.15W)
USB Ports	1 (micro USB)
Power Source	micro USB power

B. Acceptance Test Procedure

The ATP document is shown in the next page:

Acceptance Test Plan (ATP) for Acotriangulator System

1. Sound Emission Subsystem

Objective: To validate the audible signal generation capability across specified frequencies.

Test Procedure:

1. Use a calibrated sound frequency meter to validate the emitted sound frequency. Test at multiple points within the range 20 Hz to 20,000 Hz.
2. Measure the sound volume at a 1-metre distance using a decibel meter.

Test No.	Frequency(Hz)	Measured Volume(dB)	Pass/Fail
1			
2			

Expected Results:

1. The emitted sound is within the specified frequency range.
2. The volume at 1 metre is at least 70 dB.

2. Hardware Setup and Management Subsystem

Objective: To ensure correct installation and optimal layout configuration.

Test Procedure:

1. Perform a visual inspection to check the installation of Raspberry Pis and microphone breakout boards.
2. Confirm the layout formation of microphones.

Test No.	Inspection Checkpoint	Status(Pass/Fail)
1	Raspberry Pi installation	
2	Microphone breakout board installation	
3	Microphone layout (Square format)	

Expected Results:

1. All hardware components are installed and functioning correctly.
2. Microphones are arranged in a square formation.

3. Audio Acquisition and Processing Subsystem

Objective: To validate the system's ability to capture, digitise, and filter audio signals.

Test Procedure:

1. Record audio signals and analyse the digital output.
2. Introduce background noise and test the system's noise-filtering capability.

Test No.	Signal Type(Regular/Noise)	SNR(dB)	Status(Pass/Fail)
1			
2			

Expected Results:

1. Accurate digital representation of the captured audio.
2. An SNR of more than 60 dB is maintained even in noisy conditions.

4. Data Transmission and Reception Subsystem

Objective: To validate the efficient and error-free transfer of audio data from the Raspberry Pi to the PC using the `scp` command.

Test Procedure:

1. Use the `scp` command to transfer a sample audio file from the Raspberry Pi Zero to the PC.
2. Verify the reception and integrity of the transferred file.

Test No.	Test Action	Expected File Size	Received File Size	Status(Pass/Fail)
1	Transfer Sample Audio File			

Expected Results:

1. The transferred file is intact without any data loss or corruption.

5. Data Analysis and Localization Subsystem

A. **TDoA Calculation Subsystem**

Objective: To validate the accuracy of TDoA calculations.

Test Procedure:

1. Introduce test audio data and measure calculated TDoA.
2. Vary the position of sound sources and test TDoA calculations.

Test No.	Sound Source Position	Calculated TDoA(ms)	TDoA Error(ms)	Status(Pass/Fail)
1				
2				

Expected Results:

1. TDoA calculation error is less than 1ms.
2. Accurate TDoA calculations for varying sound source positions.

B. **Position Calculation Subsystem**

Objective: To test the accuracy of sound source position calculations.

Test Procedure:

1. Use test TDoA data to calculate sound source position.
2. Introduce sound sources outside the microphone grid and evaluate position estimates.

Test No.	True Position(x,y)	Calculated Position(x,y)	Position Error(m)	Status(Pass/Fail)
1				
2				

Expected Results:

1. Position calculation error is less than 0.5 meters.
2. The system provides reasonable position estimates even for out-of-grid sound sources.

6. GUI (Graphical User Interface) Subsystem

Objective: To ensure an intuitive, accurate, and responsive GUI display.

Test Procedure:

1. Feed the GUI with test position data and observe the display accuracy.

2. Simulate system failures and assess GUI error message displays.

Test No.	Test Input Data	GUI Display(x,y)	Display Lag(ms)	Error Message(if any)	Status(Pass/Fail)
1					
2					

Expected Results:

1. Sound source positions are displayed accurately in real-time.
2. System failures are accurately reported via GUI error messages.

**7. System

Testing and Validation Subsystem**

Objective: To ensure comprehensive testing and validation of the system.

Test Procedure:

1. Conduct end-to-end system tests to validate all components.
2. Use built-in diagnostic tools to simulate various failures.

Test No.	Subsystem Tested	Identified Problem(if any)	Status(Pass/Fail)
1	Hardware Setup and Sound Emission Subsystem		
2	Audio Acquisition and Processing Subsystem		
3	Data Transmission and Reception Subsystem		
4	Data Analysis and Localization Subsystem		
5	GUI (Graphical User Interface) Subsystem		

Expected Results:

1. All components pass their respective ATPs.
2. Diagnostic tools accurately identify and report simulated problems.

Review and Approval:

This Acceptance Test Plan has been prepared by Travimadox Webb with over 1 month of experience in Sound localisation using TDoA. The procedures listed are based on industry standards and best practices.

Approver's Name: ______________

Signature: ______________

Date: ______________