EEE3097S 2023

Paper Design

Group 9

18 August 2023

Table Of Contents

1. Requirement Analysis

1.1. Interpretation of the requirements

The project aims to design and implement a sound locating device by using acoustic triangulation to locate a sound source within a rectangular grid. The direction of the source will be indicated on an A1 size printed grid. An optional requirement for the system is to implement real-time acoustic source tracking involving continuously estimating the position of the sound source as it moves within the rectangular grid. The location of the sound source must be displayed through a user interface.

Key objectives of the project will include signal processing for sound acquisition, Raspberry Pi timing synchronisation, Reading data from the MEMS microphones, User Interface display and optionally real-time data processing.

The project is constrained to using two Raspberry Pi Zero single-board computers, four Adafruit I2S MEMS microphone breakout boards to locate sounds in the region of an A1 paper sized grid. Each Pi has 512 MB RAM and a 16 GB SD Card storage which will constrain our signal processing complexity and memory usage.

Relevant parameters that will be used in the project will be accuracy of the sound location, the signal-to-noise ratio our MEMS microphones provide, the sampling rate at which data is read from the microphones and the pre-processing techniques used on the signal.

1.2. Comparison of some possible implementation

Approaches for signal locating include Weighted Centroid Localization which assumes uniform sound propagation speed. It then determines the centroid (average position) from all the microphone pair intersections. Although it is simple to deploy, accuracy may degrade in noisy conditions or when sound transmission is uneven.

Another solution is using Multilateration is the process of locating the sound source based on the difference in the time of arrival between the microphones. The sound source is then found by solving a system of equations. Although it is more complicated than centroid localization, it can offer more accuracy and resilience under difficult circumstances.

Least Squares Estimation formulates the triangulation problem as an optimization task to minimize the sum of squared errors between the predicted and actual TDoA (time difference of arrival) values. It can handle measurement errors and noise, making it suitable for real-world scenarios.

Time Delay Estimation approaches include finding the highest cross-correlation between the audio signals from various microphones. Although it is quite straightforward and efficient, its accuracy may deteriorate in noisy or echo-filled environments.

Generalised Cross Correlation improves cross-correlation through phase-based techniques to increase the time delay estimates precision. It works well in noisy settings and may be used in conjunction with other signal processing methods to get superior outcomes. The generalized cross-correlation function is a more robust way to estimate the TDoA than the cross-correlation function.

1.3. Feasibility analysis

The Raspberry Pis should provide adequate processing power and MEMs microphone breakout boards will provide enough spectral data to be able to detect sound over a wide range of frequencies.

Given the capabilities of the Raspberry Pi and the available programming tools (such as Python and C), the development of algorithms for time delay estimates, translation, and graphical user interface is feasible, as well as conducting simulations to validate the signal processing techniques using MATLAB.

1.4. Bottlenecks

The Raspberry Pi Zero's limited computational power may make it difficult to process the amounts of audio data in real time. To provide prompt and precise sound source localization, the system must find a compromise between accuracy and processing speed.

Background noise has a substantial impact on sound localisation accuracy in real-world settings. Despite the use of noise reduction techniques, extremely noisy surroundings may still have an impact on the SNR and cause estimates to be off.

The proximity of microphones in the array may result in interference and crosstalk between the audio signals. This cross-interference might reduce the accuracy of time delay measurements and subsequently affect localization accuracy.

It's crucial to achieve precise synchronization between the Raspberry Pi Zero devices and microphones. The accuracy of the overall system can be greatly impacted by synchronization failure, which can lead to inaccurate time delay estimates.

The Raspberry Pi Zero's minimal memory and processing capacity may make it difficult to execute complex algorithms or cutting-edge signal processing methods.

2. Subsystem Design

2.1. Subsystem and Sub-subsystems Requirements

2.1.1. Raspberry Pi Zero Synchronization Requirements

The Pi synchronization subsystem must make sure that multiple Raspberry Pi Zero boards are precisely synchronized in time to facilitate accurate time-of-arrival measurements. This subsystem must:

- Reduce time measurement errors and guarantee reliable sound source localization.
- Avoid delays in time-critical operations, for example, time delay estimation and triangulation.
- Include a mechanism for adjusting the clocks of each Raspberry Pi Zero unit.
- Implement a synchronization protocol that describes how information about time synchronization is transmitted by each unit.
- Enable synchronization calculations and time delay estimations using timestamping of audio signals.
- Integrate with the time delay estimation subsystem.
- Include error handling mechanisms to detect and address synchronization errors.

2.1.2. Signal Acquistion Requirements

The signal acquisition subsystem is required to capture acoustic waves in the region around the A1-sized rectangular grid. Other requirements include:

- Using an array of four microphones and two single-board computers to process the audio data.
- A broad bandwidth to detect a variety of sounds in the audible frequency range.
- Integrating two single-board computers with the microphones to create a compact signal acquisition system.
- Acquiring signals using the TDoA technique for time delay estimations.
- Enabling directional sensing and source localization using optimum spatial arrangement of four microphones.
- Transmitting captured audio data efficiently to the computers while reducing delays and losses.
- Connecting the signal acquisition subsystem in a manner that achieves low power consumption for extended periods of operation during real-time acoustic tracking applications.
- Synchronization of microphones to ensure accurate measurements of time differences.
- A fast response time in real-time applications for capturing sounds in a dynamic environment.
- Moderate sensitivity and dynamic range to detect sounds with medium to high intensity such as clicks, gunshots, and loud booms.
- Implementing noise reduction techniques to achieve the best SNR and accuracy of acoustic localization in a noisy environment.

2.1.3. Time Delay Estimation Requirements

The main function of the time delay estimation subsystem is to implement the TDoA technique in accurately measuring the time differences between the arrival of sound signals at four different microphones connected to two single-board computers. To fulfil this function reliably, the following requirements must be met:

- The subsystem must accurately compute time differences between the arrival of audio signals at different microphones by employing the most efficient signal processing algorithm
- The subsystem must enable data processing from four microphones to implement the TDoA technique for acoustic localization.
- Real-time processing should be achieved by the subsystem to ensure rapid computations of sound locations.
- It must be calibrated to suitable sampling rates and allow for a variety of audio signal in the audible frequency range.
- It must employ algorithms that are computationally efficient for two single-board computers.
- The subsystem must account for reflections, multipath propagation and other effects that influence error rate in time delay estimations.
- It should interface time delay measurements to the localization algorithm for computing the sound source.

2.1.4. Triangulation Requirements

The triangulation subsystem needs to ensure accuracy and precision in the calculations for determining the location of the sound in the grid based on the TDoA technique. Other systems requirements include:

- Using four microphones and a minimum of three microphones in the subsystem to perform triangulation.
- The subsystem must use the best geometric configuration of the four microphones to achieve triangulation using the TDoA technique.
- The layout of the triangulation system must fit an A1-sized grid.
- The subsystem must be able to support real-time triangulation calculations for rapid localization and tracking of sound sources.
- The subsystem needs to efficiently map time delay measurement delays in the TDoA technique to a spatial coordinate system on the A1 grid paper.
- It must consider a range of environmental conditions that affect the speed of sound.
- It must have adjustable and testable parameters for implementation of different scenarios.
- Triangulation error analysis is required to minimize inaccuracies and resource usage.
- The triangulation subsystem must be able to account for background noise and other factors that affect the distinguishability of sounds.
- The triangulation subsystem must interface with two Raspberry Pi Zero single-board computers while using computational resources efficiently

2.1.5. Sound Tracking Requirements

Source triangulation needs to ensure real-time tracking of sound sources, continuously updating the position as the sound source moves. It needs to:

- Integrate with the triangulation subsystem to receive accurate sound source positions.
- Present sound source positions on a graphical interface.
- Provide data on sound source position in a standard format that can be interpreted by other parts of the overall system.
- Minimize the amount of time between position display and source detection,
- Implement filtering to reduce effect of noise on source tracking.
- Handle errors for cases where sounds cannot be reliably tracked.
- Minimize computational resource usage.
- Integrate with the user interface subsystem to display sound source positions with other system information.

2.1.6. User Interface Requirements

The user interface subsystem must be able to display information obtained from the other subsystems in a user-friendly format. The user interface should meet the following requirements:

- The user interface must be operable by people with varying levels of expertise.
- The interface should provide real-time updates of sounds captured by the signal acquisition subsystem.
- It should display the distance and location of the sound source.
- The subsystem should enable users to quickly figure out the spatial coordinates of sounds.
- Information about the state of the microphone array should be displayed by the user interface.
- It must enable configuration of parameters for adjustments in varying environmental conditions.
- The user interface should give notifications and alerts about significant sounds in the environment.

2.2. Subsystem and Sub-subsystems Specifications

2.2.1. Pi Synchronisation Specifications

Hardware Components: Two Raspberry Pi Zero single-board computers for synchronization purposes; RTC.

Software Components: Python-based synchronization script to ensure accurate timing between the two Raspberry Pi devices.

Functionalities:

- Establish a common time-based between the two Raspberry Pi devices for coordinated operation.
- Enable synchronized execution of signal acquisition and time delay estimation.
- Each Pi is connected to two microphones out of the 4 available microphones and combines the results synchronously.

Technical Parameters:

- Utilize the system clock provided by the Raspberry Pi hardware.
- Implement synchronization using network-based time synchronization protocols (NTP or PTP).

Communication Protocol: Exchange timing information between the two Raspberry Pi devices to ensure synchronization.

Error Handling: Implement error detection and correction mechanisms to account for variations in clock frequencies.

2.2.2. Signal Acquistion Specifications

Hardware Components: Raspberry Pi Zero for data processing and communication, and four Adafruit I2S MEMS microphone breakout boards for audio signal capture.

Software Components: Python script to interface with the microphones, read audio data from the I2S communication, and manage data synchronization.

Functionalities:

- Capture synchronized audio signals from all four microphones.
- Apply hardware-based synchronization mechanisms for accurate data acquisition.
- Preprocess audio signals to remove noise and enhance the signal quality.
- Detect sounds that emanate from source sources located up to ~30 m away.
- Stores time of arrival of sound at each microphone.

Technical Parameters:

- Set the sampling rate to 44.1 kHz to ensure accurate representation of audible frequencies in the range between 50 Hz and 17 kHz and a noise level of ~40 dB.

- Employ a buffer size of 1024 samples for efficient data handling.
- Microphone array in square configuration (0.1 m apart).
- Array located at centre of A1 paper.
- A1 grid placed at an angle of 0°, with respect to flat surface.

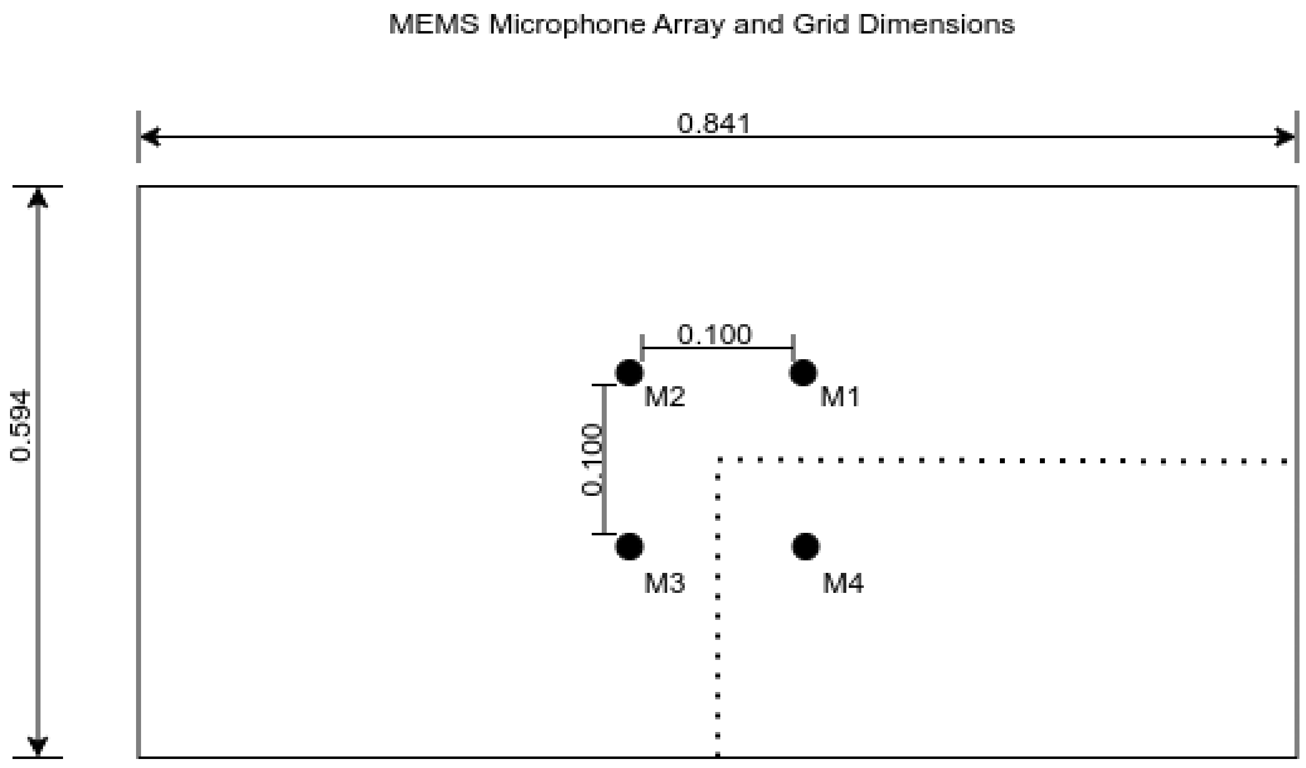


Figure 1 illustrating the MEMS microphone array spacing on the A1 sized grid. Measurements shown in meters.

Communication Protocol: Use I2S communication protocol for seamless data transfer between microphones and Raspberry Pi Zero.

Data Preprocessing:

- Apply bandpass filters to isolate frequencies in the range of 50 Hz to 15 kHz.
- Implement noise reduction techniques, such as spectral subtraction, to enhance signal-to-noise ratio.

2.2.3. Time Delay Estimation Specifications

Hardware Components: Raspberry Pi Zero for signal processing and time difference of arrival calculations.

Software Components: Python algorithms for time delay estimation

Functionalities:

- Takes in room temperature to calculate speed of sound. Equation for the speed of sound in air can be used to find speed of sound *v* in air as a function of absolute temperature:

$$v = \mathrm{sqrt}(\beta RT/M)$$

where *M* is the molar mass of air, *R* is the gas constant, β is the adiabatic index, and *T* is the absolute temperature in Kelvin [4].

- Calculate time differences of arrival between pairs of microphones on each Raspberry Pi.
- Determine time shift with the highest cross-correlation value to estimate time delay.
- Interface with the triangulation subsystem to provide accurate time delay measurements.

Technical Parameters:

- Implement FFT (Fast Fourier Transform) for efficient cross-correlation calculations.
- Employ windowing techniques to improve accuracy in cross-correlation.
- Standard speed of sound taken as 350 $m \bullet s^{-1}$ at room temperature.
- Cross-correlation function R(t) between two discrete signals $x_1[n]$ and $x_2[n]$ is computed as:

$$R(t) = \Sigma\, (x_1[n]*x_2[n+t])$$

Communication Protocol: Exchange time delay information with the triangulation subsystem via inter-process communication.

2.2.4. Triangulation Specifications

Hardware Components: Raspberry Pi Zero for triangulation calculations.

Software Components: Python algorithms for centroid localization, multilateration, or least squares estimation.

Functionalities:

- Process time delay information from the Time Delay Estimation subsystem.
- Calculate sound source location based on chosen triangulation algorithm.
- Interface with the User Interface subsystem to display location results.

Technical Parameters: Triangulation algorithm

Communication Protocol: Exchange calculated sound source location with the User Interface subsystem for display.

2.2.5. Sound Tracking Specifications

Hardware Components: Raspberry Pi Zero; 4 MEMS microphones; 16 GB memory storage; RTC.

Software Components: Real-Time Operating System for real-time processing; particle filter tracking algorithms for updating sound source positions accurately.

Functionalities:

- Track sound sources with frequencies between 400
- Continuous tracking of a sound source with minimal latency between position updates.
- Implementation of filtering techniques to minimize impact of noise and reduce noise-induced jittering.
- Accurate positioning and a confidence interval of 2 meters.
- Data logging for storing historical source positions for updating current position.
- Storage memory constraint of 2 GB.
- Track sounds that emanate from source sources located up to ~30 m away.

Technical Parameters: Particle filter tracking algorithm.

Communication Protocol: Exchange calculated sound source location with the User Interface subsystem for display; interact with triangulation subsystem for continuous location updates.

2.2.6. User Interface Specifications

Hardware Components: Utilize Raspberry Pi Zero for user interface control and display; A1 paper with grid; display interface

Software Components: Develop a Python-based graphical user interface (GUI) using libraries

Functionalities:

- Display real-time updates of captured audio signals and sound source location.
- Allow users to configure parameters, such as sampling rate and noise reduction settings.
- Use A1 paper as a form of graphical interface for mapping directions to system display.

Technical Parameters:

- Implement responsive design to ensure usability on different screen sizes.
- Ensure the GUI operates smoothly within the computational capabilities of Raspberry Pi Zero.

Communication Protocol: Receive sound source location information from the Triangulation subsystem for display.

2.3. Inter-Subsystem and Inter-Sub-subsystems Interactions

2.3.1. Subsystem Interactions

A. User Interface and Sound Tracking Subsystem

The user interface interacts with the sound tracking system for visualizing the real-time positions of track. Sound tracking updates sound source positions and transmits them to the user interface to display them on a graphical interface.

B. Pi Synchronization and Signal Acquisition Subsystem

This interaction ensures that each Pi provides a synchronized clock for precise time-arrival measurements by the signal acquisition subsystem.

C. Signal Acquisition Subsystem and Time Delay Estimation Subsystem

After capturing audio from the MEMS microphones, the time delay estimation subsystem employs the data to estimate the time differences between sound arrivals at different microphones in the array.

D. Time Delay Estimation Subsystem and Triangulation Subsystem

Time delay estimations are transferred to the triangulation subsystem which uses the estimations, as well as known sensor positions to calculate sound source positions. The triangulation subsystem requires accurate information from the time delay estimation subsystem.

2.3.2. Sub-Subsystem Interactions

a. Signal Acquisition Subsystem and Source Tracking Subsystem

Audio data acquired from the signal acquisition subsystem is provided to the sound tracking subsystem. Information about the audio signal can be matched to the sound source's audio signature.

b. Triangulation Subsystem and Source Tracking Subsystem

The triangulation subsystem provides calculated sound source positions which form a crucial part of tracking sound sources in real-time.

c. Source Tracking Subsystem and User Interface

The user interface displays positions provided by the source tracking subsystem to a graphical interface allowing users to monitor the sound source.

d. User Interface and Triangulation Subsystem

Requests can be made by the user interface to the triangulation subsystem to start or stop triangulation calculations so that a user can have control over the computational process.

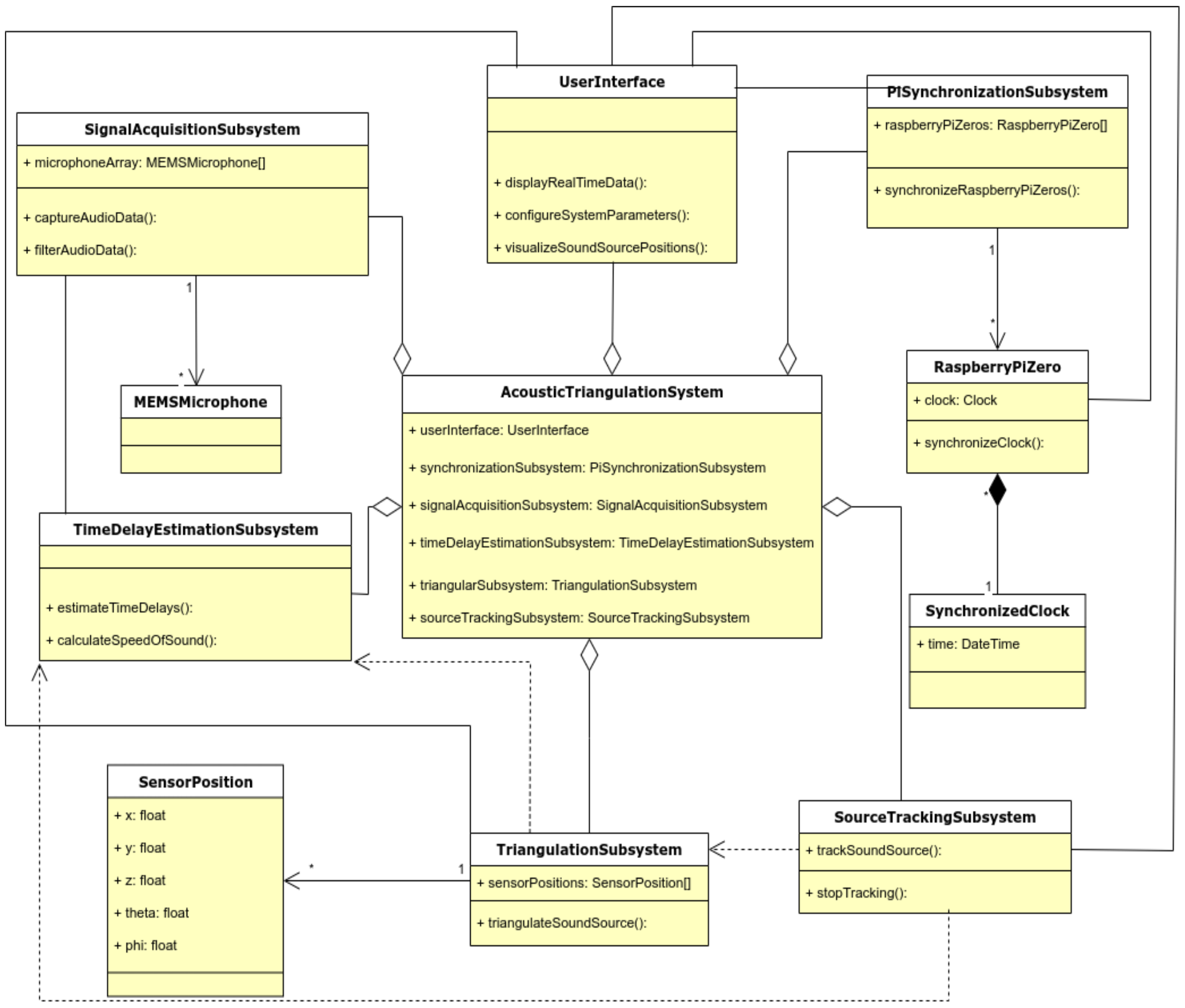
e. Source Tracking Subsystem and Display Results Sub-Subsystem

Updated positions are displayed by the display sub-subsystem, reflecting the real-time movement of sound sources and assist users in tracking them.

2.4. Acoustic Triangulation System UML Diagrams

2.4.1. UML Class Diagram

The AcousticTriangulationSystem contains a UserInterface, PiSynchronizationSubsystem, SignalAcquisitionSubsystem, TimeDelayEstimationSubsystem, TriangulationSubsystem, and SourceTrackingSubsystem. The attributes of these systems and their primary functions are described by the UML class diagram below.



UserInterface
+ displayRealTimeData():
+ configureSystemParameters():
+ visualizeSoundSourcePositions():
PiSynchronizationSubsystem
+ raspberryPiZeros: RaspberryPiZero[]
+ synchronizeRaspberryPiZeros():
SignalAcquisitionSubsystem
+ microphoneArray: MEMSMicrophone[]
+ captureAudioData():
+ filterAudioData():
MEMSMicrophone
AcousticTriangulationSystem
+ userInterface: UserInterface
+ synchronizationSubsystem: PiSynchronizationSubsystem
+ signalAcquisitionSubsystem: SignalAcquisitionSubsystem
+ timeDelayEstimationSubsystem: TimeDelayEstimationSubsystem
+ triangularSubsystem: TriangulationSubsystem
+ sourceTrackingSubsystem: SourceTrackingSubsystem
RaspberryPiZero
+ clock: Clock
+ synchronizeClock():
SynchronizedClock
+ time: DateTime
TimeDelayEstimationSubsystem
+ estimateTimeDelays():
+ calculateSpeedOfSound():
SensorPosition
+ x: float
+ y: float
+ z: float
+ theta: float
+ phi: float
TriangulationSubsystem
+ sensorPositions: SensorPosition[]
+ triangulateSoundSource():
SourceTrackingSubsystem
+ trackSoundSource():
+ stopTracking():
1
*

Figure 2 representing an object-oriented view of the acoustic triangulation system.

2.4.2. UML State Machine Diagram

The acoustic triangulation system is described by the State Machine diagram below which shows the transition from System Initialization to Idle after initialization. It transitions from Idle to Data Acquisition when triggered to start data collection. When data acquisition is completed, the system transitions from Data Acquisition to Time Delay Estimation. After accurate time delay estimation, the system moves to the Triangulation state. Provided the triangulation is successful, the system transition to Source Tracking for continuous tracking of sound sources. Upon sound source tracking, the system enters Display Results to visualize the source positions. Then the system transitions back to Idle. When the system is to be turned off, it transitions from Idle to End.

be

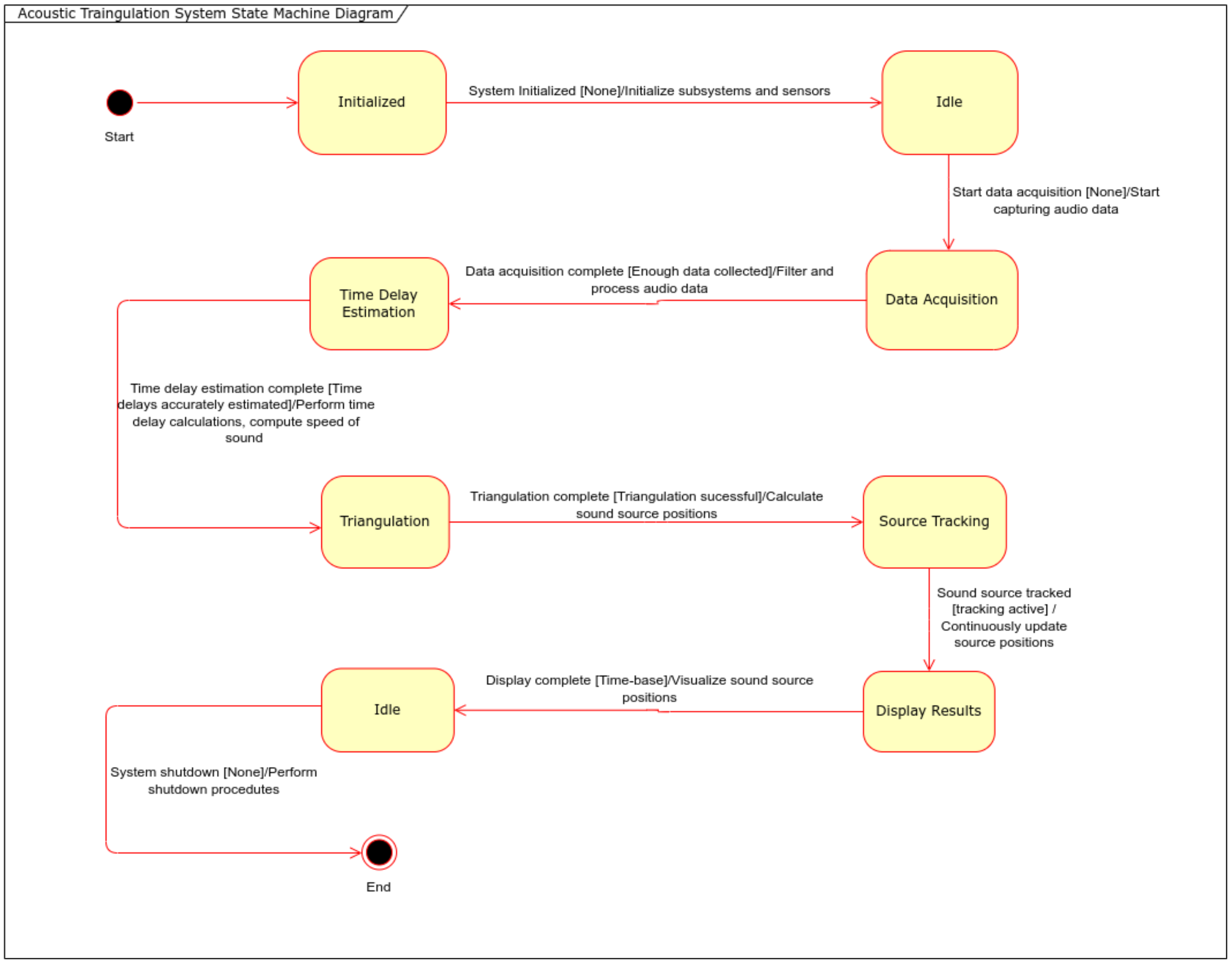

Acoustic Traingulation System State Machine Diagram
Start
Initialized
System Initialized [None]/Initialize subsystems and sensors
Idle
Start data acquisition [None]/Start capturing audio data
Data Acquisition
Data acquisition complete [Enough data collected]/Filter and process audio data
Time Delay Estimation
Time delay estimation complete [Time delays accurately estimated]/Perform time delay calculations, compute speed of sound
Triangulation
Triangulation complete [Triangulation sucessful]/Calculate sound source positions
Source Tracking
Sound source tracked [tracking active] / Continuously update source positions
Display Results
Display complete [Time-base]/Visualize sound source positions
Idle
System shutdown [None]/Perform shutdown procedutes
End

Figure 3 showing the operation of the acoustic triangulation system using two Raspberry Pi Zero boards and four MEMS microphones.

3. Acceptance Test Procedure

3.1. Figures of merits based on which you would validate your final design.

Localization Accuracy: Deviation between estimated and actual sound source positions. This will provide insight into the system's precision in determining sound source locations within the grid

Real-time Performance: Processing time for sound source localization. This assesses the system's efficiency in processing data in real-time scenarios.

Noise Robustness: Ability to accurately localize sound sources in noisy environments. The Signal-to-Noise Ratio (SNR) will be used to quantify the noise level at which accurate localization can still be achieved.

Computational Efficiency: Resource utilization and execution time of algorithms.

System Reliability: Consistency and stability of results across multiple tests

3.2. Experiment design to test these figures of merit.

Localization Accuracy Test:

- Place the sound source at known positions within the grid.
- Record the system's estimated positions and compare with actual positions.
- Calculate the Root Mean Square Error (RMSE) or Euclidean distance between estimated and actual positions.

Real-time Performance Test:

- Measure the time it takes for the system to calculate sound source positions.
- Vary the number of microphones and input signal lengths to assess scalability.

Noise Robustness Test:

- Introduce controlled levels of background noise to the input signals.
- Evaluate the system's ability to accurately locate the sound source amidst noise.

Computational Efficiency Test:

- Measure CPU and memory usage during signal processing.
- Compare the execution time of different algorithms (e.g., cross-correlation vs. GCC).

3.3. Acceptable performance definition

Localization Accuracy: RMSE below a certain threshold (e.g., 5% of the grid size).

Real-time Performance: Localization time less than a specific value (e.g., 100 ms).

Noise Robustness: Accurate localization maintained with up to 20 dB of background noise.

Computational Efficiency: CPU usage below a certain percentage, execution time within a limit.

4. Development timeline

IC-Work-Breakdown-Structure-Dictionary--8721

5. Table listing individual contributions.

Name	**Individual Contribution**
Bonga Njamela	Interpretation of requirements
	Signal acquisition requirements
	Time delay estimations requirements
	Signal acquisition specifications
	Time delay estimations specifications
	UML & OOP Diagrams
Charles Portman	User Interface requirements
	User Interface specifications
	Development timeline
	Subsystem Interactions
	Sound tracking requirements
	Sound tracking specifications
Morris Nkomo	Comparison of interpretations
	Feasibility analysis and Bottlenecks
	Pi synchronisation specifications
	Acceptance test procedure
	Triangulation specifications
	Triangulation Requirements

6. References

1. Coban, Atakan, and Niyazi Coban. “Using Arduino in Physics Experiments: Determining the Speed of Sound in Air.” Physics education 55.4 (2020): 43005–. https://uct.primo.exlibrisgroup.com/permalink/27UCT_INST/o2vtft/cdi_crossref_primary_10_1088_1361_6552_ab94d6, last visited: 07-08-2023.

2. Chen, Lin, et al. "Acoustic Source Localization Based on Generalized Cross-Correlation Time-Delay estimation." *Procedia engineering* 15 (2011): 4912-4919.

3. Fan, Jing, Qian Luo, and Ding Ma. "Localization Estimation of Sound Source by Microphones Array." *Procedia Engineering* 7 (2010): 312-317.

4. Worland, Rand S., and D. David Wilson. "The Speed of Sound in Air as a Function of Temperature." *The Physics Teacher* 37.1 (1999): 53-57.

7. Appendix