University of Cape Town

EEE3097S

ECE DESIGN

Paper Design

Author:	*Student Number:*
Murray Inglis	INGMUR002
Tinashe Timba	TMBTIN004
Dylan Trowsdale	TRWDYL001

August 18, 2023

Contents

1 Requirements Analysis

1.1 Objectives:

- Triangulate a sound source by calculating its 2D coordinates on a grid. Involves using Time Difference of Arrival (TDoA) algorithms to determine the sound source's position based on the time delay between microphones.
- Display the triangulation result: Show the calculated position on a graphical user interface (GUI).
- Record data from each microphone simultaneously. The system should capture audio from all four microphones simultaneously.
- Implement noise reduction: Filter and process the captured audio to reduce noise interference.

1.2 Constraints:

- Must use TDoA to triangulate the sound source: TDoA methods require accurate time synchronization between the microphones.
- Must use a GUI to display the result: A user interface is necessary to visualize the calculated sound source position.
- Should be capable of recording and processing data in real-time (optional): If real-time processing is desired, it's important to consider the computational capabilities of the Raspberry Pi units.
- Must use 2 Raspberry Pi's and 4 microphones: These hardware components are specified for the project.
- Configurable microphone setup: The system should allow adjusting the arrangement of microphones.
- Accurate location estimation within a specified tolerance: The calculated position should be within a certain range of the true sound source position.
- High signal-to-noise ratio to ensure accurate output: Minimizing noise interference is essential for accurate triangulation.

1.3 Performance Expectations:

- Should be capable of recording and processing data in real-time (optional).
- Configurable microphone setup.

1.4 Parameters:

- The system should provide accurate location estimation within a specified tolerance.
- High signal-to-noise ratio to ensure accurate output.

1.5 Feasibility Analysis:

1.5.1 Strengths

- Microphone Integration: The integration of 4 microphone breakout boards with the Raspberry Pi units is feasible and common. The Pis have the necessary hardware interfaces and GPIO pins to connect to the microphones.
- Signal Processing: TDoA calculations, time delay estimation, and triangulation algorithms are common techniques. There are many available libraries and code examples.

1.5.2 Weaknesses

- Timestamping: The raspberry pi's are running on separate clocks. An error of a millisecond will be an error of about 30cm which is a large portion of the grid size. Accurate timestamping may be hard to achieve with the software available.
- Coordination: Since the readings are being taken on 2 separate devices, it may prove difficult to transfer data reliably between them.

1.5.3 Opportunities

- Iterative Process: Iterative testing and refinement will be needed to ensure each subsystem works correctly and integrates seamlessly with others. This will be an important chance to verify our design and catch any errors preemptively.

1.5.4 Threats

- Time: Developing, testing, and integrating the various subsystems might take considerable time, especially being new to certain aspects of the project, like GUI development or synchronization mechanisms.
- Setbacks: These systems and calculations are unfamiliar. Unforeseen setbacks may occur.

1.6 Possible Bottlenecks:

- CPU performance (clock speed, sample rate): The clock speed of the Raspberry Pi and the sample rate for audio processing could impact the system's efficiency.
- Insufficient memory (RAM and external): Both RAM and external storage may become limiting factors, particularly for real-time data processing if this is done in real-time.
- Microphone quality (noise): The quality of the microphones affects the accuracy of captured audio data.
- Time management will be crucial to the project but will prove difficult with other assignments and projects.

2 Subsystem Design

2.1 Pi Synchronization:

2.1.1 Requirements:

- The Pis must be able to connect to the same network for data transmission and synchronization.
- Audio data captured by microphones should be synchronized between the two Pi devices.
- The synchronization mechanism should minimize data transmission delay.
- Wifi connectivity for internet access.

2.1.2 Specifications:

- Use NTP server synchronization for timestamping.
- Timestamps for audio recording must be synchronized with an error of less than 0.5ms between the devices
- One Raspberry Pi is designated as a master to do all calculations
- Both Raspberry Pis will record audio.
- Connect the raspberry pi over Eduroam or home wifi networks.

2.1.3 UML Diagram:

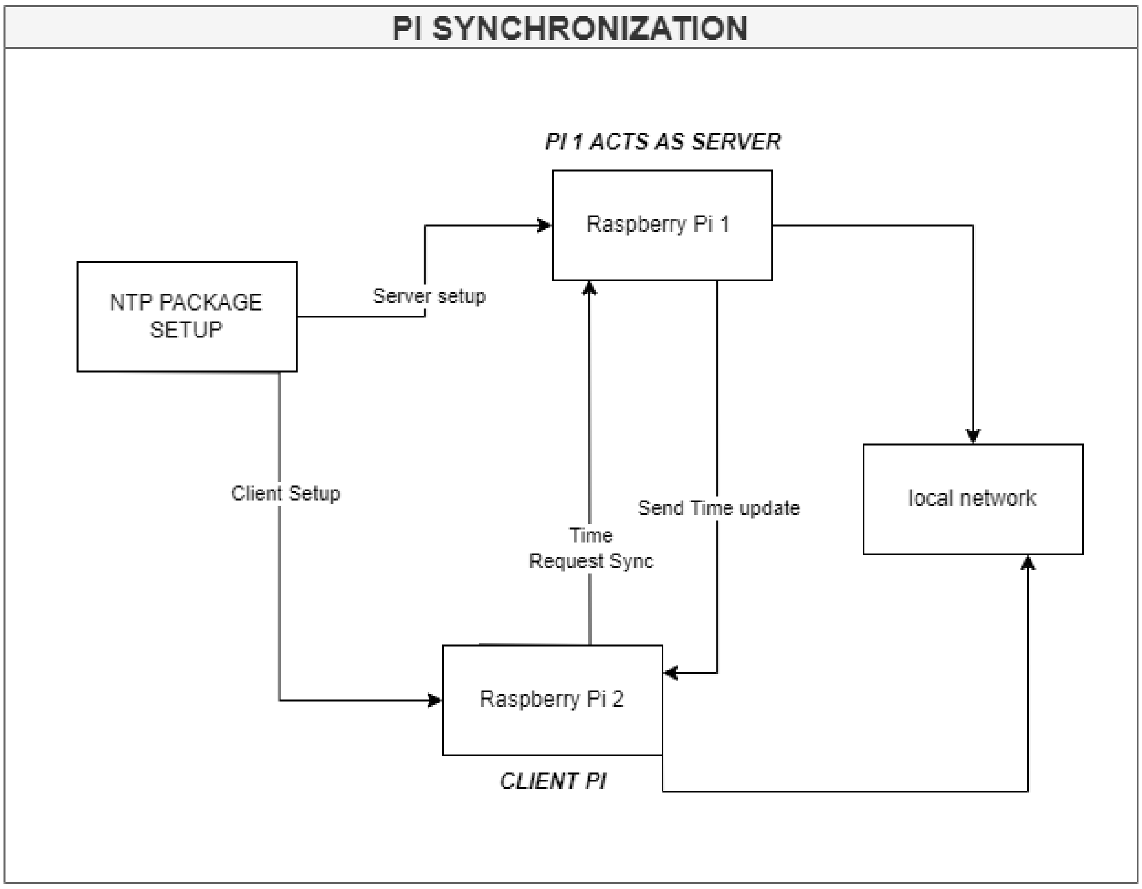


Figure 1: Pi Synchronization

2.2 Signal Acquisition:

2.2.1 Requirements:

- Use 4 microphone breakout boards and 2 raspberry pis to capture audio
- Signal acquisition needs to be autonomous without user involvement.
- Sample the audio above the required Nyquist rate.
- Bandlimit the input sound with filtering.
- The microphones should be physically small and efficient enough to be run on a Raspberry Pi.
- The microphones should draw a small supply of current.

2.2.2 Specifications:

- 4 Adafruit I2S MEMS microphone breakout boards will be used.
- 2 Raspberry Pi zero w v1.1 will be used.
- The microphone breakout boards must be able to record 100 Hz to 10 kHz.
- The signal will be sampled at 100 kHz.
- I2S serial communication protocol will be used to communicate between the Raspberry pis and the microphones.
- The microphones must have an SNR of above 50dB
- The supply current for the microphones should be less than 3 mA
- The microphones should operate at a supply voltage of 3.3V
- There will be 2 microphones per Raspberry Pi

2.2.3 UML Diagram:

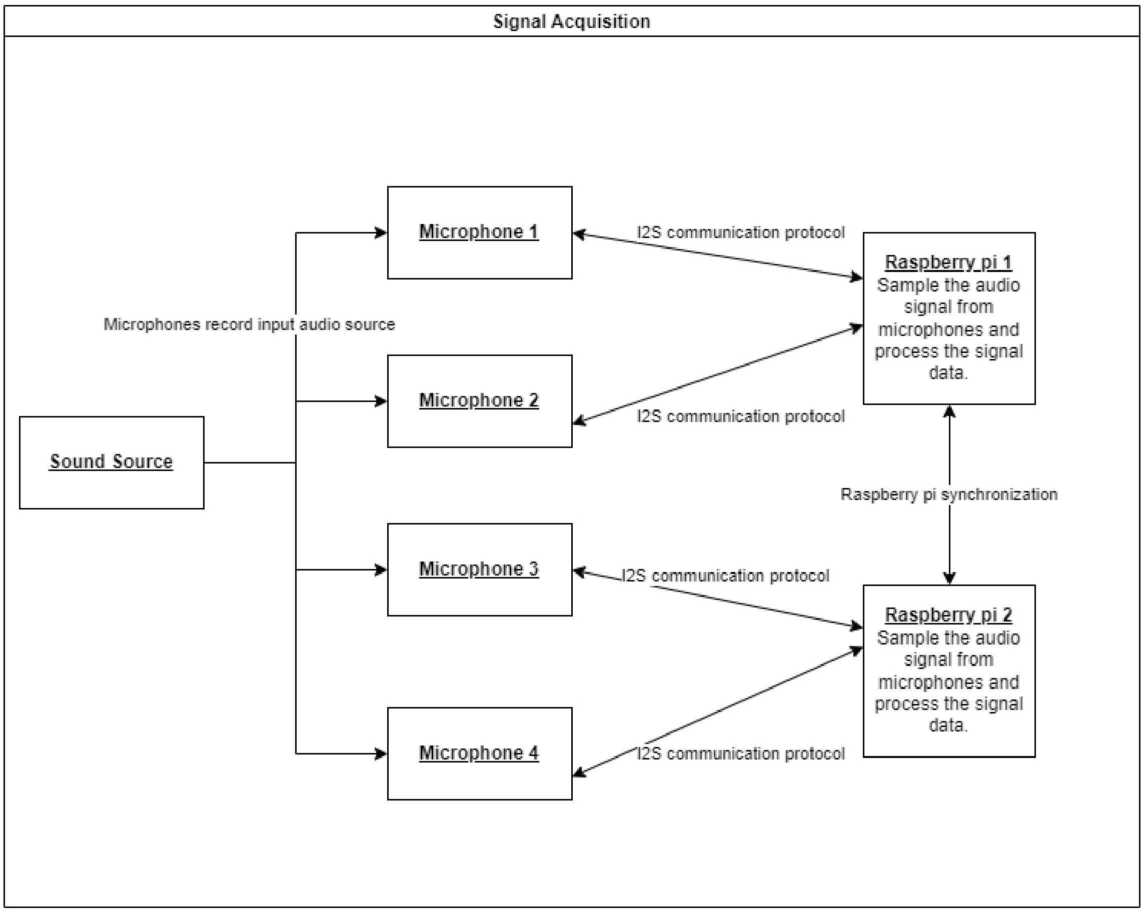

Signal Acquisition
Sound Source
Microphones record input audio source
Microphone 1
Microphone 2
Microphone 3
Microphone 4
I2S communication protocol
I2S communication protocol
I2S communication protocol
I2S communication protocol
Raspberry pi 1
Sample the audio signal from microphones and process the signal data.
Raspberry pi synchronization
Raspberry pi 2
Sample the audio signal from microphones and process the signal data.

Figure 2: Signal Acquisition UML Diagram

2.3 Time Delay Estimation:

2.3.1 Requirements:

- Must be able to show the time delay on arrival for each microphone
- Must use cross-correlation to calculate the time delay.
- The sound source must be known by the system.

2.3.2 Specifications:

- The accuracy range must be $\pm 5\%$.

- The sound source is an 8 kHz sine wave.
- One microphone will be placed at (0,0) and will be used as a reference. The other microphones will have their inputs cross-correlated against the reference microphone.

2.3.3 UML Diagram:

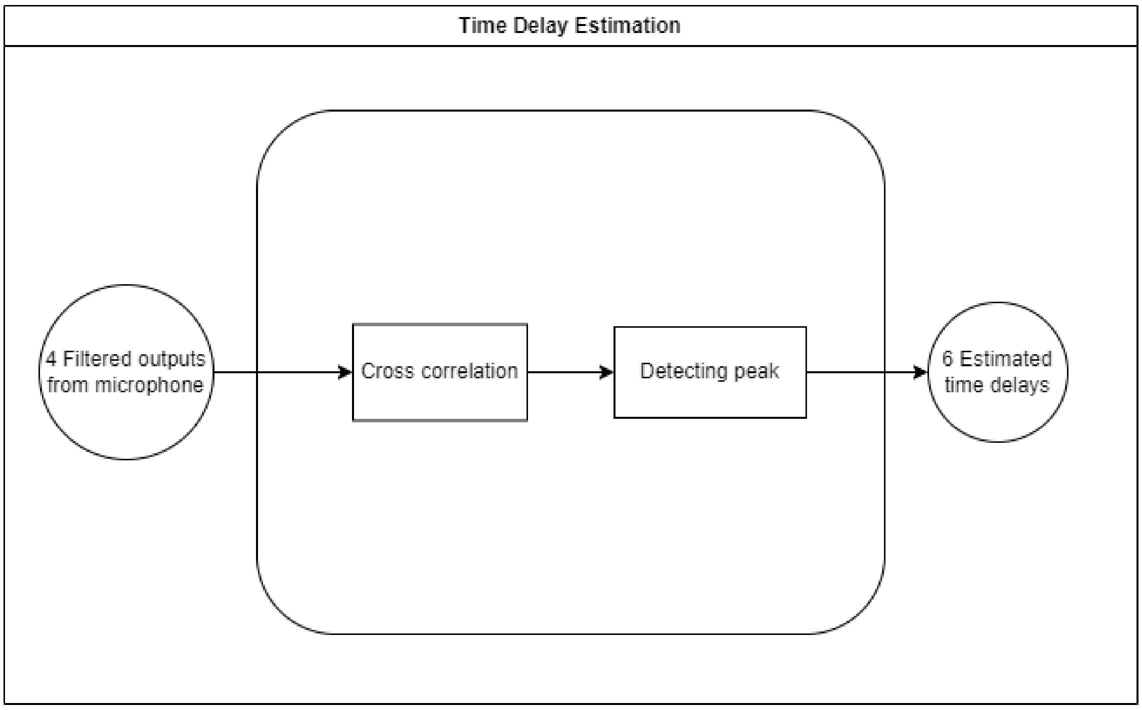


Figure 3: Time Delay Estimation UML Diagram

2.4 Triangulation:

2.4.1 Requirements:

- Must be able to calculate the position of the sound source within a given accuracy range.
- Must use 4 sensors, thus creating a system of 4 nonlinear equations.
- Must be able to calculate the distance from each microphone to the sound source.
- The position of the microphones must be known by the system and configurable.

- The position of the microphones must be configurable and programmable.

2.4.2 Specifications:

- The accuracy range must be ±5%.
- The equations are solved using an iterative algorithm (Least squares).
- The time delay from each microphone to the reference can be used to create distance equations.

2.4.3 UML Diagram:

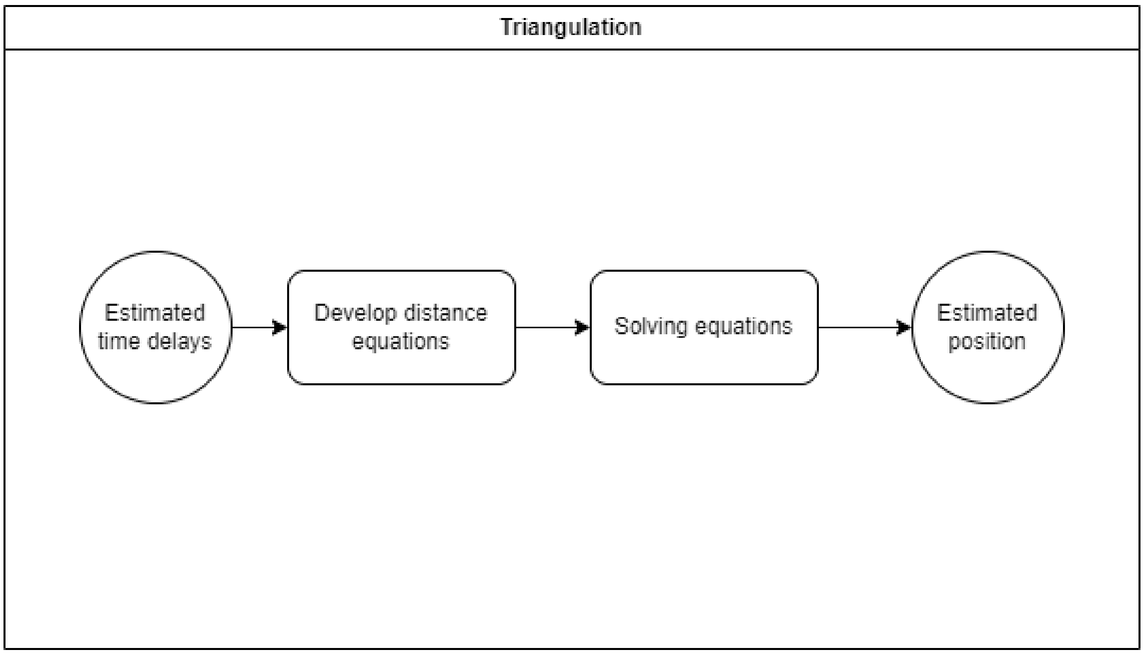


Figure 4: Triangulation UML Diagram

2.5 User Interface:

2.5.1 Requirements:

- Must use a Graphical user interface to display the coordinates of the predicted results.
- Be able to start and stop measuring.
- Enter configuration mode.

- Log the data related to each localization session for analysis of results.
- Notify a user when each localization session is finished or if errors have occurred.

2.5.2 Specifications:

- A graphical user interface is to be used for a user to start and stop the sound localization session, display the output results of the predicted source location on the grid to a monitor screen, display, or terminal, and access program settings for the sound localization session.
- Output a notification message to the user to notify them of the status of the localization session when it has ended.
- Alternatively, use the terminal to interact with the program.

2.5.3 UML Diagram:

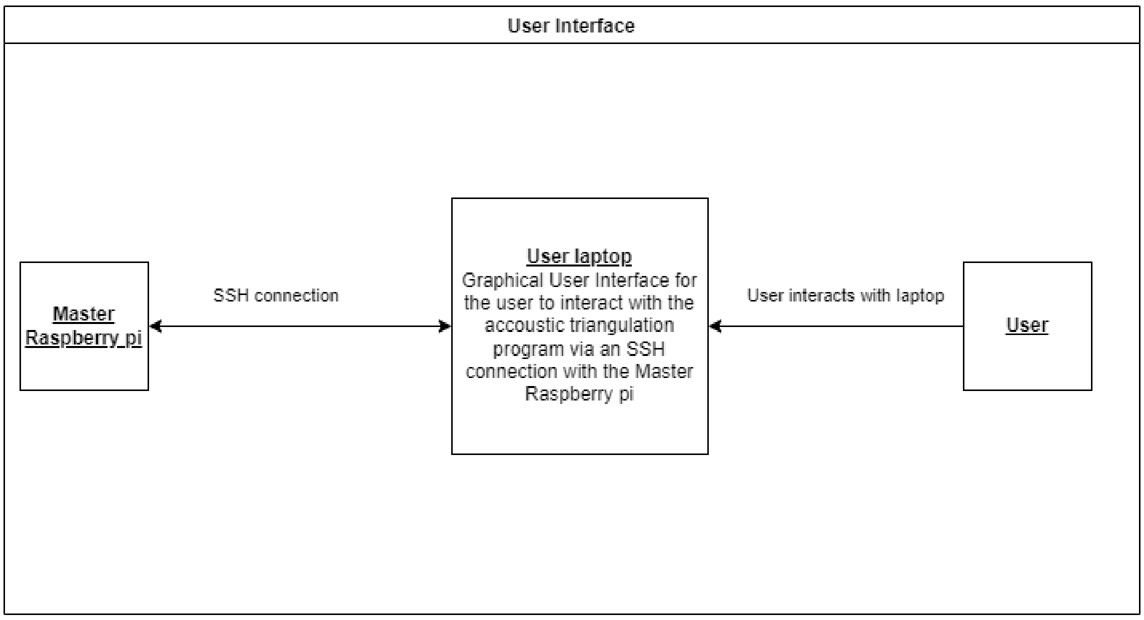

User Interface
Master Raspberry pi
SSH connection
User laptop
Graphical User Interface for the user to interact with the accoustic triangulation program via an SSH connection with the Master Raspberry pi
User interacts with laptop
User

Figure 5: User Interface UML

2.6 Inter Sub-System Interaction

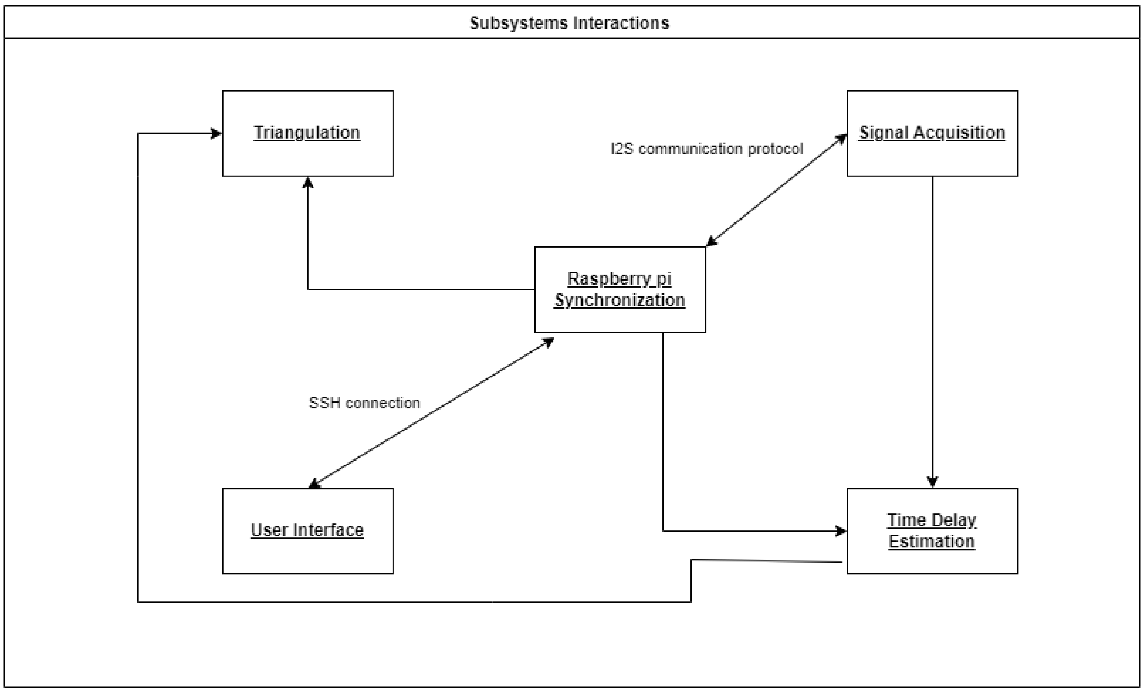


Figure 6: Subsystem Interaction

3 Acceptance Test Procedures

3.1 Figures of Merits

1. Position Accuracy via TDoA calculation: The deviation between the calculated position using the TDoA algorithm and the actual position of the sound source.
2. Sound Detection and Signal-to-Noise Ratio: The frequency range for which the microphones are able to pick up the sound clearly. The ratio of the signal generated to the surrounding noise.
3. Synchronization Stability and Accuracy: Testing the pi synchronization is consistent.
4. Usability Testing of GUI: The ease of use of the user interface.
5. Unit test for code: Debugging and testing algorithm.

3.2 Experiment Design

3.2.1 Position Accuracy Test

Setup:

- Place the microphones at a standard position on the grid.

Procedure:

1. Emit sound at specified grid points.
2. Calculate position using the TDOA algorithm.
3. Measure position deviation and processing time.
4. Repeat for multiple points.

Acceptable Performance:

- Mean position deviation within ±2cm.
- 80% of positions within ±5cm.

3.2.2 Sound Detection and SNR Test

Setup:

- Place the microphones randomly and ensure the sound source is available

Procedure:

1. Play a sound of known frequency along with real-world noise
2. Detect and process the sound
3. Calculate the SNR
4. Vary the frequency and gain of the sound source.

Acceptable Performance:

- The microphones are able to pick up sound from 0-20 kHz.
- The SNR is high.
- Minimal effect of noise on position accuracy.

3.2.3 Synchronization Stability and Accuracy

Experiment Design:

1. Continuously monitor the clock synchronization between the Raspberry Pi units over an extended period.
2. Measure synchronization accuracy over time.

Acceptable Performance:

- Synchronization error should remain within the specified range (e.g., within a few milliseconds) consistently.

3.2.4 Usability Testing of GUI

Experiment Design:

1. Engage users who are not familiar with the system.
2. Have them interact with the GUI to perform tasks like starting/stopping measurement and changing settings.

Acceptable Performance:

- Users should be able to easily navigate and perform tasks using the GUI without confusion.

3.2.5 Unit tests for the raspberry pi code

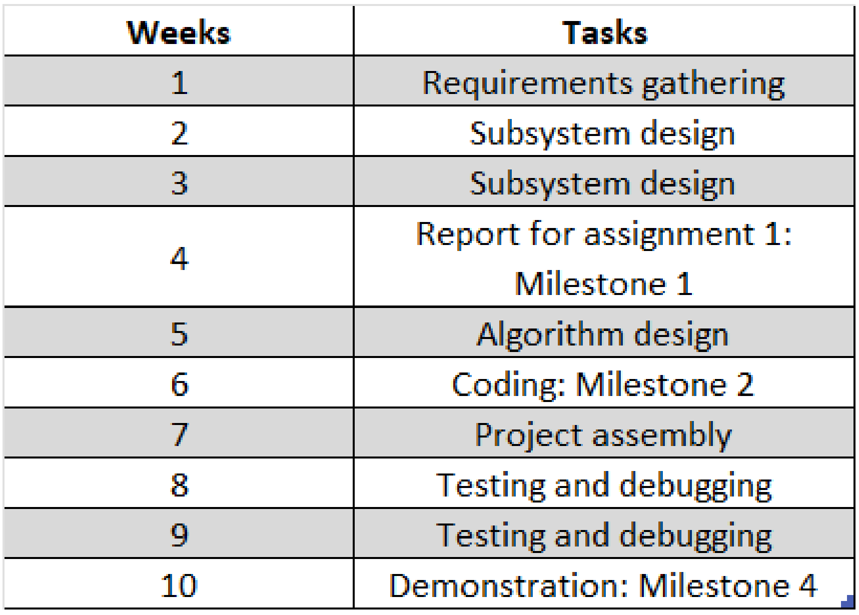
Experiment Design:

1. Test code section by section.
2. Debug the code using the Raspbian debugger.

Acceptable Performance:

- The code runs without errors.

4 Development Timeline, Trello Page, and Contributions



Weeks	Tasks
1	Requirements gathering
2	Subsystem design
3	Subsystem design
4	Report for assignment 1: Milestone 1
5	Algorithm design
6	Coding: Milestone 2
7	Project assembly
8	Testing and debugging
9	Testing and debugging
10	Demonstration: Milestone 4

Figure 7: Development timeline

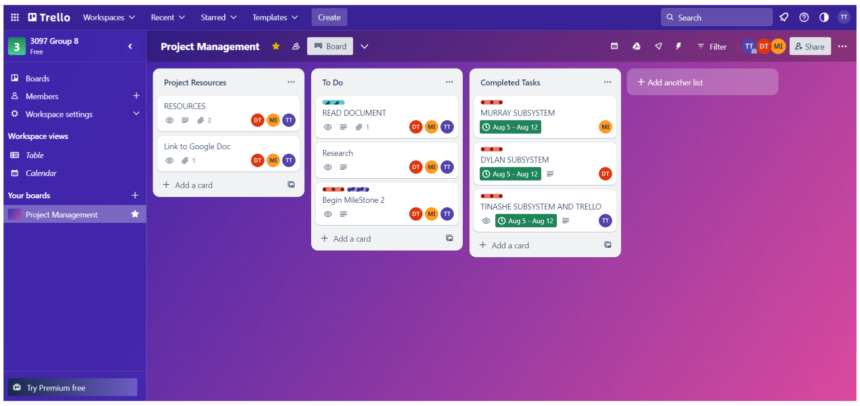


Figure 8: Project Management Board

Name	Student Number	Contributions
Dylan Trowsdale	TRWDYL001	Signal Acquisition Subsystem, User Interface Subsystem, Development Timeline, Subsystem Interactions
Murray Inglis	INGMUR002	Triangulation Subsystem, Time Delay Estimation, GitHub setup, Requirements Analysis
Tinashe Timba	TMBTIN004	Pi-Synchronization Subsystem, Trello Page setup, ATPS, Feasibility Analysis, Possible Bottlenecks

References

[1] i2s datasheet. https://cdn-shop.adafruit.com/product-files/3421/i2S+Datasheet.PDF. Accessed: 2023-08-17.

[2] Source localization using generalized cross correlation. https://www.mathworks.com/help/phased/ug/source-localization-using-generalized-cross-correlation.html. Accessed: 2023-08-18.

[3] Time synchronization on the raspberry pi. https://www.hackster.io/kamaluddinkhan/time-synchronization-on-the-raspberry-pi-d11ece. Accessed: 2023-08-14.