EEE3097S 2023
Milestone 4: Final Project Report

Group 2

18 October 2023

Aimee Simons (SMNAIM002)

Tilal Mukhtar (MKHTIL001)

Md Shaihan Islam (ISLMDS002)

1 TABLE OF CONTENTS

2 ABBREVIATIONS

GCC-PHAT: Generalized Cross-Correlation Phase Transform

GPIO: General Purpose Input Output

I2S: Inter-IC Sound

LSE: Least Squares Estimator

MEMS: Micro-Electro-Mechanical System

Mic: Microphone

NTP: Network Time Protocol

OS: Operating System

RPi: Raspberry Pi

SCP: Secure File Copy

SNR: Signal to Noise Ratio

SSH: Secure Shell

TDoA: Time Difference of Arrival

ToA: Time of Arrival

USB: Universal Serial Bus

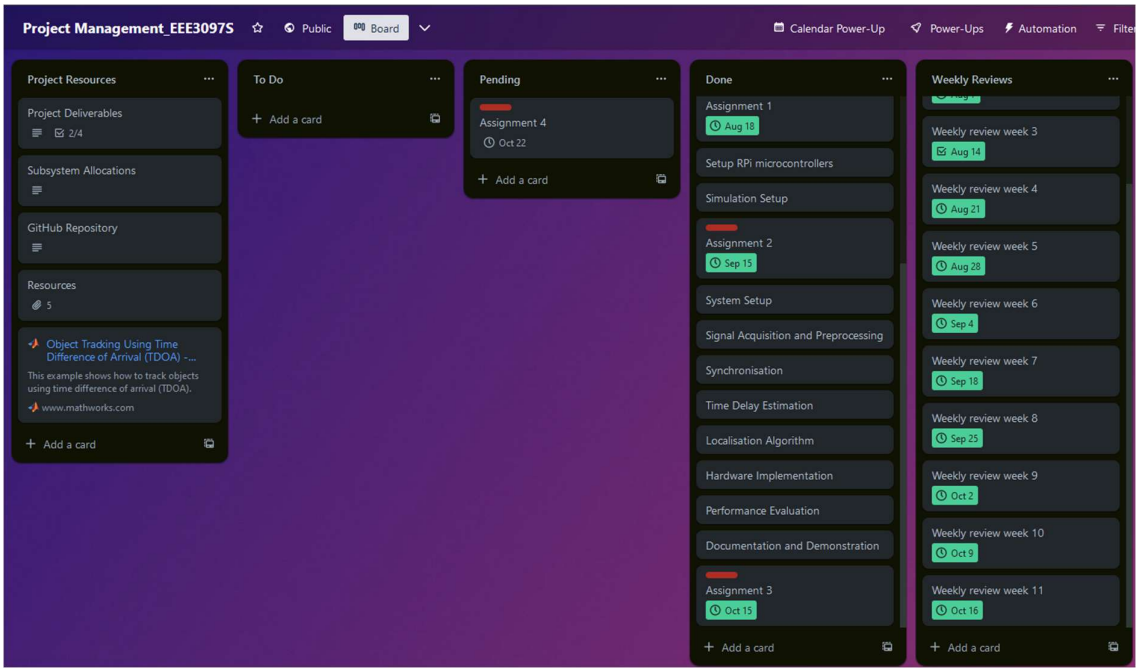
WBS: Work Breakdown Structure

3 ADMIN DOCUMENTS

3.1 CONTRIBUTIONS

Student Name	Student Number	Contributions
Md Shaihan Islam	ISLMDS002	Requirement Analysis (ATPs) Paper Design (ATPs and Timeline) Consolidation of ATPs and Future Plan Conclusion
Tilal Mukhtar	MKHTIL001	Introduction Validation using simulations. Validation using final implementation. Conclusion
Aimee Simons	SMNAIM002	Requirement Analysis Paper Design Conclusion

3.2 PROJECT MANAGEMENT PAGE



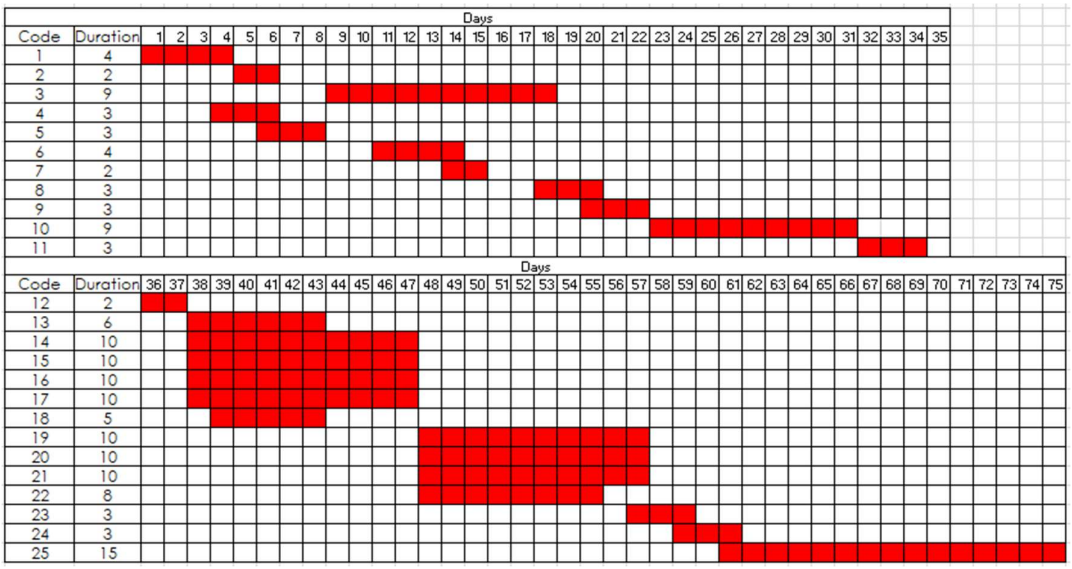
3.3 GITHUB REPOSITORY

Link to GitHub Repository:

https://github.com/tshaihan/EEE3097S_Group2

3.4 PROJECT TIMELINE

The cells shaded in red show the progress completed so far, whereas the cells in blue mean that the particular section is not complete and is still in progress.



From the Gantt chart above, it can be seen that all events were completed on time, without any major delays. Since all progress for the entire project is complete, the WBS no longer contains any remaining tasks.

4 INTRODUCTION

4.1 BACKGROUND

Acoustic localization is an important field with applications ranging from robotics to audio signal processing. Acoustic triangulation is a technique used to determine the position of a sound source by measuring the time difference of arrival (TDoA) of sound signals at multiple distributed microphones. This technique has produced great results in multiple fields in the industry with a wide range of projects currently under development like gunshot detection in the city and UAV surveillance. This project focuses on implementing an acoustic triangulation system using TDoA to accurately determine the position of an acoustic transmitter within a defined grid.

4.2 PROBLEM STATEMENT

The objective of this project is to design and implement an acoustic triangulation system using TDoA to accurately locate the position of a sound source within a rectangular grid. The system will utilize four microphone breakout boards and two Raspberry Pi Microcontrollers to capture and process audio signals, and then calculate the two-dimensional coordinates of the sound source within the grid and display the predicted results through a graphical user interface.

5 REQUIREMENT ANALYSIS

5.1 SYSTEM REQUIREMENTS DISCUSSION

Based on the problem statement brought forth, a list of requirements and, therefore, specifications should be drawn up to adequately ensure the success of the overall system. A basic summary of what the system should entail is as follows: The system should be able to efficiently record the audio from the calibration source and the sound source. Once recorded, the audio should be processed to eliminate any noise or imperfections, without causing any distortion. From these processed recordings, the system should be able to synchronise the audio received from separate RPI modules. Once synchronised, the ToA for each microphone should be determined. Once these ToAs are determined, the system will thereafter determine the TDoA between all the microphones. These TDoA values will be utilised within a triangulation algorithm and, as a result, provide two-dimensional coordinates, which estimate the position of the sound source. These coordinates will be displayed to the user through a Graphical User Interface.

5.2 SYSTEM REQUIREMENTS ANALYSIS

Based on the requirements discussion presented above, the following requirements analysis can be conducted.

The overall system is largely based on [1], and as such retains similar requirements. For the implementation of the system presented in [1], 2 types of microhomes were used: the MEMS microphone and the Electret Condenser Microphone (ECM). The difference between the MEMS microphone and the ECM is the fact that the MEMS microphone already converts analogue signals (sound signals) into digital signals, with no need for an Analogue to Digital Converter (ADC). As such, the MEMS would be the preferred microphone, as it utilizes less circuitry. For this project, 4 MEMS microphones have been provided.

From [9], it is determined that the rated SNR for the microphones is 65dB. The overall system should therefore account for variations in SNR due to background noise. This will be implemented using a bandpass filter to filter out both high frequency noise and low frequency noise.

3 Methods of synchronisation was discussed in [1]. These are: Global Positioning System (GPS), Network Trime Protocol (NTP) and a Calibration Signal. The main implementation that was focused on was the use of a Calibration Signal. In this implementation, an initial signal is made (ideally) at the centre of the designated grid, and then another signal is made at another location on the same designated grid. By “aligning” the initial calibration signal, it emulates the situation where the audio is completely synchronised. Figure 3.4 in [1] illustrates how this method is implemented. The synchronisation delays were determined using the GCC-PHAT method discussed subsequently. These sync delays are also utilised in the TDoA calculations.

In terms of Time Delay Estimation, many papers, including [1] and [5] utilized the same method to determine the ToA for each microphone, which involves the use of a cross-correlation function - more specifically the GCC-PHAT method. This involves finding the normalised cross-correlation function, initially introduced in [1], and thereafter multiplying it with a weighting function. This weighting function, seen as (3.8) in [1], utilises the Fourier transform of the original signal and the reference signal. The TDoA is thereafter determined by subtracting the ToA found at Mic 1 with any of the other ToAs (e.g. Mic i, where i=2,3,4…), as well as subtracting the sync errors for each microphone. This is discussed further in the “Time Delay Estimation” Subsystem section. In order to

prevent any inaccuracies in the delay estimation, [1] recommends that a frequency chirp signal be used, instead of a single frequency signal.

Various Triangulation algorithms were researched, in order to determine the most efficient and accurate method. From [5], the concept of multilateration is introduced. This is a method in which the TDoA for all sensors/microphones are utilized to plot hyperboloid functions, whereby the intersection of these hyperboloids is the estimated position of the sound source. Another implementation researched, is using the LSE algorithm. In Appendix B of [7], the equations for setting up the distance matrix is introduced. This calculation of the estimated coordinates is known as a Linear LSE, which means it treats each of the equations as a linear equation. The equations, however, are non-linear and so the concept of "Non-linear Least-Squares refinement" is introduced in [6]. This approach, explained in [6], allows the "maximum-likelihood estimation" to be obtained. This proves to be an iterative approach, shown as (12) in [6].

For designing a GUI, many methods were brought forward. Using the *Tkinter* library in Python was seen to be the most popular and most-used method of creating GUIs for User interaction. It was, however, discovered that PySimpleGUI (documentation seen in [10]), a library that utilises the Tkinter tools, provides a more user-friendly approach to designing GUIs, and as such should be easier to implement.

5.3 System Requirements List

The objective of this project is to design and implement an acoustic triangulation system using Time Difference of Arrival to accurately locate the position of a sound source within a rectangular grid.

The following system requirements have been identified:

1. The system shall be capable of determining the position of a stationary sound source within a rectangular grid.
2. The system shall provide two-dimensional coordinates relative to the rectangular grid.
3. The system shall utilize two RPi Microcontrollers.
4. The system shall utilize four microphones.
5. The system shall be capable of capturing audio signals from all microphones.
6. The system shall incorporate noise reduction techniques to improve the signal-to-noise ratio (SNR) of the captured audio signals.
7. The system shall calculate the TDoA of the audio signals from the microphones.
8. The system shall employ an appropriate triangulation algorithm to convert the TDoA data into two-dimensional coordinates.
9. The system shall provide a graphical user interface (GUI) for displaying the predicted location of the sound source.
10. The GUI should be intuitive and user-friendly.

5.4 System Specifications List

The following specifications can be derived from the system requirements:

Overall

1. The system will use two RPi Zero W modules and the microcontrollers will operate in parallel.
2. The system will use four Adafruit I2S MEMS Microphone breakout boards.

3. The system will use a 0.8 x 0.5 m grid printed on an A1 (0.841 x 0.594 m) sheet of paper.
4. The source and calibration audio signals will be generated using a pair of Bluetooth speakers.
5. The Bluetooth speakers will be each be connected to a mobile device.
6. A laptop running the Microsoft Windows OS will be used as a host device.
7. The RPi Microcontrollers will be powered via their Micro-USB ports.
8. The RPi Microcontrollers will receive a nominal input voltage of 5 V DC at an input current of 2.5 A.
9. The RPi Microcontrollers will be connected to the local network of the host device via Wi-Fi.
10. The RPi Microcontrollers will communicate with the host via the SSH and SCP protocols.
11. The microphone breakout boards will be powered via power connections to the RPi Microcontrollers.
12. The microphone breakout boards will receive a nominal input voltage of 3.3V DC.
13. The microphone breakout boards will communicate with the RPi Microcontrollers via the I2S serial communication protocol.
14. The system will be programmed using the Python programming language.
15. A Generalized Cross-Correlation Phase Transform algorithm will be applied to pairs of audio recordings, to acquire the TDoA data.
16. The system should ensure that the time synchronisation error between the RPi microcontrollers and calculated TDoA values are accurate within 10 microseconds.
17. The system should be capable of capturing audio signals within the audible spectrum.
18. The system should ensure that the sample rate of the microphones is 25kHz.
19. The system should ensure that SNR of the captured audio signals are greater than 50dB.
20. A Least Squares Estimation algorithm will be applied to the TDoA values acquired to determine the estimated coordinates of the sound source.
21. The system should provide the location of the sound source within a 1cm accuracy.
22. The GUI will be programmed using the PySimpleGUI library for Python.

5.5 System Acceptance Test Procedures

With the requirements and specifications defined clearly above, it is imperative that the appropriate system acceptance procedures be taken, to ensure the requirements are met to an acceptable degree. Acceptance tests would further assist in identification of malfunctions, as well as play an important role in the debugging process. After each acceptance test has been undertaken, it will be given either a “pass” (signifying that the subsystem has passed the Acceptance Test Procedure (ATP) or a ”fail” (signifying that the subsystem has failed the ATP).

The table below lists, in detail, the acceptance tests that have been performed, observed and critiqued for each of the requirements:

Specification	Figures of Merit	Acceptance Test	Acceptable Performance
1. The system will use two RPi Zero W modules and the microcontrollers will operate in parallel.	Two fully working Raspberry Pi Zero W modules with Raspbian OS installed onto the on-board micro-SD card storage	To test this specification, both the boards will be checked to see whether Raspbian OS has been installed and is working properly. The SD cards will also be checked to ensure it has not been corrupted.	For an acceptable performance, the Raspberry Pi modules will need to have microSD cards that are not corrupt. The microSD cards should also not contain any other data besides the Raspbian OS (32-bit).

		For the test for parallel operation, the time in which both the Pi's logged in using SSH (Secure-Shell Host) will be recorded.	Furthermore, to ensure parallel operation of the Pi's, the time in which both of the modules logged in has to be the same, this would signify a very small, negligible delay.
2. The system will use four Adafruit I2S MEMS Microphone breakout boards.	Four fully working Adafruit I2S MEMS microphone breakout boards	To test if these microphone breakout boards work appropriately, each will be connected to a Pi module, and will be tested to record audio for a specific amount of time.	An acceptable performance for this specification would mean that each of the microphones is able to record a sound and store it as an audio file. This audio file must then be audible and produce similar sound to the test audio.
3. The system will use a 0.8 x 0.5 m grid printed on an A1 (0.841 x 0.594 m) sheet of paper.	An A1 sheet of paper with a grid having a maximum x value of 0.8 and a maximum y value of 0.5	Although this sheet of paper is provided, the grid will be checked for any abnormalities. It will also be checked to see if it is in good condition to ensure it can be used effectively.	For an acceptable performance, the grid must have a maximum x value of 0.8 and a maximum y value of 0.5. The grid blocks must also be of equal size. The grid must be in a clean condition where every block can be clearly seen.
4. The source and calibration audio signals will be generated using a pair of Bluetooth speakers.	Two working Bluetooth speakers, preferably identical speakers	This acceptance test will ensure that the two Bluetooth speakers that will be used for the calibration signal and the source signal, are both working properly and can output the necessary chirp signals at different frequencies.	An acceptable performance for this test is that they must both have a common volume, preferably a common maximum volume. They should also be smaller than one grid block in size, to decrease location error. They should also be able to generate chirp signals at the required frequencies.
5. The Bluetooth speakers will be each be connected to a mobile device.	Two working Bluetooth speakers, preferably identical speakers	This acceptance test will involve checking the ability for the speakers to connect to a mobile device using Bluetooth.	For an acceptable performance, the speakers must be able to effectively connect to the two mobile devices that will be feeding the speakers with the chirp signals. The speakers must also be responsive when connected to a mobile device, it should not have a notable delay in response (e.g. more than one second)
6. A laptop running the Microsoft Windows OS will be used as a host device.	A working laptop with WiFi capabilities	This acceptance test will be used to determine whether the host device, a laptop running the Windows	Since most of the information will be extracted by the host device from the Pi's, it is essential that the host device

		operating system, will qualify to be the appropriate host device for the Pi modules.	is working adequately, has the ability to connect via WiFi as well as having the ability to successfully open more than one console at a time.
7. The RPi Microcontrollers will be powered via their Micro-USB ports	The light on the RPi that indicates that it is receiving power	To test this, a micro-USB cable will be connected from the USB port of a laptop to the micro-USB port of both the Pi modules.	For an acceptable performance, the green/yellow light on the Pi should turn on, indicating that it is able to receive power from the laptop.
8. The RPi Microcontrollers will receive a nominal input voltage of 5 V DC at an input current of 2.5 A.	Multi-meter voltage and current readings	For this test, a multi-meter will be used to check the voltage and current running through the Pi.	For an acceptable performance, the voltage at the power input of the Pi should be close to 5V and a maximum current of around 2.5A should be running through the circuit.
9. The RPi Microcontrollers will be connected to the local network of the host device via Wi-Fi.	Successful WiFi connection between the Pi's and the host device	The host network will try to connect the RPi's to its local network using WiFi. A PuTTY console was set up to connect to the IP address of the RPi's and connect them to the host network from there.	For an acceptable performance, both microcontrollers should be able to communicate to the host using the host network.
10. The RPi Microcontrollers will communicate with the host via the SSH and SCP protocols.	Successful ping messages on both ends of the RPi's.	To test this specification, both the Raspberry Pi Microcontrollers will be connected to each other, and then a 'ping' command will be sent from the console of one of the Pi's to the IP address of the other Pi and vice versa.	For acceptable performance, there must not be any loss of packets from either of the Pi's to each other. This ensures complete communication between the Pi's, hence confirming the utilisation of the system using two RPi Microcontrollers.
11. The microphone breakout boards will be powered via power connections to the RPi Microcontrollers. **12. The microphone breakout boards will receive a nominal input voltage of 3.3V DC.**	Multi-meter voltage and current readings	For these specifications, a multi-meter was used to ensure that the microphone was receiving the appropriate 3.3V power from the respective Pi's.	For an acceptable performance, the power port of the microphone breakout boards must record a 3.3V voltage reading on the multimeter.
13. The microphone breakout boards will communicate with the RPi Microcontrollers via the I2S serial	Successful response message	To test whether or not the microphones have successfully connected to the Pi's, the command "arecord -l" was entered into the Pi command line,	For an acceptable performance, the response message to the command "arecord -l" needs to be "SUCCESS", which indicates a successful connection of the Pi

communication protocol.		which determines if the I2S card is recognised.	to the microphone, and hence the two modules can communicate.
14. The system will be programmed using the Python programming language.	Efficient coding environment and access to necessary libraries	The Python programming language must be tested to ensure it is the correct language to undergo all the aspects of the project, such as signal acquisition, signal processing, etc. The appropriate libraries will then be tested to test this.	For an acceptable performance, the following standard Python libraries must be installed and working properly: -matplotlib -numpy -scipy -subprocess -gcc_phat.py
15. A Generalized Cross-Correlation Phase Transform algorithm will be applied to pairs of audio recordings, to acquire the TDoA data.	TDoA plots	To ensure that the GCC-PHAT algorithm works as expected, the TDoA delays for each of the microphones were plotted, to visually depict the performance of the cross-correlation algorithm.	For acceptable performance, the algorithm must be able to output values that can be plotted. Therefore, the time delay plots must show an acceptable cross-correlation of the signals. It cannot be so little that signals are not picked up, but it also cannot be so much that the signals are cut off or affected by noise.

16. The system should ensure that the time synchronisation error between the RPi microcontrollers and calculated TDoA values are accurate within 10 microseconds.	TDoA values of the audio signals from each microphone Synchronisation and TDoA errors	For this acceptance test, the TDoA values from each microphone will be received and analysed. The ideal expected (calculated) TDoA will first be determined using the appropriate TDoA eqaution. The "measured" TDoA calculated from the parameters provided by the microphones will then be compared to the ideal expected TDoA for TDoA error. In addition to that, since two Pi's are used, a perfect synchronisation between the microphones is not possible. The aim is, however, to ensure as small a synchronisation delay as possible. The synchronisation delays will be timed and analysed for this ATP.	For an acceptable performance, the TDoA algorithm will first have to be able to display four TDoA values, one for each microphone. These values must all be real, floating-point values. After that, the TDoA value will have to be accurate enough so that the triangulation algorithm can detect the sound source to as close to the original as possible. The average TDoA error should hence not exceed 10 microseconds. The average synchronisation error should also not exceed 10 microseconds.
17. The system should be capable of capturing audio signals within the audible spectrum.	TDoA values of the audio signals and TDoA plots	The highest possible frequency of sound that humans can hear is 20 kHz. The system will thus be tested using chirp signals with less than 20 kHz frequency, and the time delay plots will be checked, and the estimated TDoA values will be compared to the calculated TDoA values.	For an acceptable performance, the system should output accurate TDoA values, within 10 microseconds of the calculated TDoA values.
18. The system should ensure that the sample rate of the microphones is 25kHz.	TDoA values of the audio signals and TDoA plots	To test this, the TDoA plots can be checked for aliasing due to a low sample rate. The "raspberrypi.sh" file could also be checked, as that is where the sampling rate of the microphone is set	For an acceptable performance, the TDoA plot should be free from aliasing, and the "raspberrypi.sh" script should have "25000" after "-r"
18. The system should ensure that SNR of the captured audio signals are greater than 50dB.	SNR values	For this acceptance test, the performance of the filter used to refine the audio signals will be used. Both the signal power and	For acceptable performance, the output SNR value should be higher than 50 dB. The filter will underperform if the SNR is below this minimum

		noise power of the input unfiltered signal, as well as the output filtered signal will be measured to calculate the input SNR and output SNR, which will then be compared for performance.	threshold. The microphones have a maximum SNR of 65 dB, so the closer the SNR is to this maximum value without distortion, the better the filter performance.
19. A Least Squares Estimation algorithm will be applied to the TDoA values acquired to determine the estimated coordinates of the sound source. **20. The system should provide the location of the sound source within a 1cm accuracy.**	Positional Coordinates	In this acceptance test, a pair of coordinates will be expected from the triangulation algorithm that will use the TDoA information from the previous algorithm to output a position relative to the rectangular grid used. The coordinates provided by the algorithm will then be compared to the actual position of the sound source to calculate position error.	For an acceptable performance, the triangulation algorithm will first have to be able to output the coordinates, and these coordinates have to be real, floating-point numbers. After that, it should be checked if the provided coordinates are within the bounds of the rectangular grid. Finally, the position error should be analysed. A position error of 1 centimetre, or 0.1 units, should not be exceeded.
21. The GUI will be programmed using the PySimpleGUI library for Python.	User feedback	The GUI to be used will be coded using the PySimpleGUI library, and the effectiveness of this choice will be tested by asking other engineering students on their opinion of the GUI.	An acceptable performance would be an overall positive response from the volunteers of the survey, with few visible errors in the GUI to report.

6 PAPER DESIGN

The Interpretation of the Requirements can be viewed under the Requirements Analysis Section of this report.

6.1 POSSIBLE IMPLEMENTATIONS

6.1.1 Signal Acquisition

Many implementations of this systems were brought forth, which are discussed below:

6.1.1.1 Unknown Signals

This implementation is under the assumption that the overall structure of the impeding signal is unknown. This prevents the user from knowing the total duration of the recording, the duration of the sound signal, the frequency of the sound signal, and many more unknown variables. A way of implementing this is restricting or 'clipping' the recording to a desired length, as well as restricting the bandwidth of the processing algorithms. Although this is a generalised approach, and would most likely work in all cases, it could cause inaccuracies when the original signal is processed to the point where a ToA cannot be determined .i.e. clipped too soon or filtered to a point of distortion.

6.1.1.2 Known Signals

Complementary to the previous implementation, this implementation is under the assumption that the overall structure of the impeding signal is known. The recording length is kept constant, the calibration signal duration and frequency are also known, along with the source signal duration and frequency. This allows certain parameters to be previously determined. For example, the recording could be cut in half with no chance of cutting off any signals prematurely. Another example is setting the cutoff frequencies of the filter to predetermined values. Knowing the structure of the impeding signal is very specialised and might not work in general situations, but for this implementation it is sufficient.

6.1.2 Synchronisation

Synchronisation is the subsystem which ensures that the RPi modules' audio recordings are adequately synchronised, in order to prevent time delay inaccuracies.

6.1.2.1 NTP

NTP is also known as Network Time Protocol and allows for clock synchronisation between the 2 RPi modules. This is established by synchronising the clocks of each Pi with the network server which they are connected to. This, in theory, is the simplest implementation, and therefore requires no other specific hardware or software implementations. It is known, however that the RPi modules' clocks are unreliable and, as a result, the NTP could to be unreliable and yield inaccurate results.

6.1.2.2 Hardware

This simple synchronisation implementation involves utilizing a physical connection to each of the RPi modules. This involves using a serial peripheral such as UART, to communicate a command to both modules at the same time. Although easier to implement, with minimal software requirements, this implementation could be time consuming, in that the execution time could be longer than the other implementations, and hardware intensive.

6.1.2.3 Calibration Signal

Another synchronisation implementation involves using a calibration signal. This is implemented by having an initial signal located at the centre of the grid, and thereafter, playing the source signal at a certain location within the grid. Ideally, the ToAs for each mic should be the same, as the calibration signal is equidistant from each microphone. As such, the calibration signals within each of the recordings, need to be "in line". This is achieved by finding the delays between the calibration signals, using a cross-correlation function, using the first microphone recording as a reference signal. As a result of these delays, the audio recordings are 'cut' at the end of the calibration signals located in each recording, to essentially create the impression that the recordings 'started' at the same time. 2 implementations of utilizing a calibration signal is discussed below:

6.1.2.3.1 Time Slicing

Time Slicing would utilize splitting the audio signals in half to separate the calibration signal and source signal, in order to distinguish between the two sounds.

6.1.2.3.2 Frequency Slicing

Another implementation is using Frequency Slicing instead of Time Slicing. This involves filtering out the calibration signal instead of cutting it out completely from the audio recordings.

6.1.3 Time Delay Estimation

This subsystem determines the ToA of the sound signal for each of the microphones. This is implemented utilising the theory of cross-correlation, which measures the similarity between two time series functions.

6.1.3.1 Cross-Correlation

As expressed in [1], Cross-Correlation is the standard method to determine the time delays between 2 signals. The resulting function is an array twice the size of the original signal is known as the normalised cross-correlation function. This method is based solely in the time domain, and therefore does not differentiate between signals which have varying SNR.

6.1.3.2 GCC PHAT

GCC PHAT is also known as a "Generalized Cross-Correlation Phase Transform". In [1], it is discussed that this implementation is an extension to the above cross-corelation method. In this method, a weighting function is determined from the Fourier Transforms of each of the signals and multiplied by the normalised cross-correlation function discussed previously. This function generates single, narrow peaks at a delay time, whereas the standard cross-correlation function, which generates multiple peaks at different time delays. Because this method may provide a more definitive result, as well as accurate, this would be the preferred implementation.

6.1.4 Triangulation

The triangulation subsystem estimates the two-dimensional coordinates of the sound source, by utilizing the TDoAs from the previous subsystem.

6.1.4.1 Multilateration

This implementation is based off [5]. In this report, it is stated that this implementation uses TDoA to estimate the position of the sound source. The TDoA is determined by finding the difference between 2 microphones' ToAs. A hyperboloid is therefore plotted, where each point on the hyperboloid represents all locations with the same TDoA already determined. These are implemented for all microphones. The estimated position of the sound source is located at the intersection of all the hyperboloids. Figure 1.1 in [5] illustrates this implementation.

6.1.4.2 Linear LSE

Another implementation is using matrix calculations to solve for the TDoA equations. The equations are in the form:

$$\tau 1n * c = \sqrt{(y - y1)^2 + (x - x1)^2} - \sqrt{(y - yn)^2 + (x - xn)^2}$$

$where\ n\ is\ the\ microphone\ index\ and\ \tau 1n\ is\ the\ TDoA\ between\ microphone\ 1\ and\ microphone\ n.$

The following equation is also referenced in Appendix B of [7]. These equations are placed within a matrix, in order to solve for (x,y), the estimated position of the sound source. The matrix equation can also be found in Appendix B of [7]. This is established through a matrix calculation. Although this provides a very close estimate, the variables being worked with is in the form of a nonlinear equation. This implementation will be used but must be optimised to eliminate the errors due to the nonlinearity of the unknown variables.

6.1.4.3 Nonlinear LSE

This implementation follows on from the Linear LSE algorithm and uses an iterative process to go from the initial estimate, determined from the Linear LSE algorithm to the final estimate, by minimising the square of the errors. In this case, the Python function 'curve_fit' from the scipy.optimize library will be used. As such the Linear and Non-Linear LSE should be utilized in conjunction with one another to estimate the most accurate position of the sound source. This concept is brought up in [6].

6.2 Feasibility Analysis

3 Points of Feasibility will be discussed:

6.2.1 Technical Feasibility

Insofar as technical resources available, two RPi Zero W modules were provided, along with 4 microphones, 2 SD cards and an A1 grid. It can therefore be said that this design project is technically feasible as the majority of the equipment/materials needed were, in fact, provided. The members of the project are also equipped with the skills needed in order to solve the problem.

6.2.2 Financial Feasibility:

As mentioned previously, two RPi Zero W modules, 4 microphones, 2 SD cards and an A1 grid were all provided and, as such, did not require any financial input from the members of the design team. The software available that will be used for the algorithms are free to use and readily available. The only expense is the connectors need for the RPi Microcontrollers, in order to provide power to the boards as well as display the operating system on a monitor. As such, the project is financially feasible as not a lot of funds are needed to execute the solution.

6.2.3 Scheduling Feasibility

With reference to the fact that there are no required class test or exams scheduled for the design course, it will allow the members to focus solely on the design and implementation, with minimal distractions. It will, therefore, ensure that scheduling the milestones are more conducive to everyone's schedule. The project is, therefore, feasible with regards to scheduling and time constraints.

6.3 Possible Bottlenecks

6.3.1 Hardware Constraints

- The system is constrained to make use of two RPi Zero W Microcontrollers which limits the options available to synchronise the Microcontrollers.
- The system is constrained to make use of four Adafruit I2S MEMS Microphone breakout boards. This limits the possible accuracy of the TDoA and Triangulation algorithms.
- The system is constrained to an A1 grid size which limits the accuracy of the system.

6.3.2 Software Constraints

- The system will make use of the Python programming language which has worse performance than other suitable programming languages such as C++.

6.3.3 Time Constraints

- The project must be completed over the course of a semester.

6.3.4 Information Constraints

- As this is a relatively new topic to the some of the members, it might be initially difficult to grasp the concept and implement the solution correctly.

6.4 SUBSYSTEM DESIGN

6.4.1 Subsystem Breakdown

The following subsystems were identified:

1. Signal Acquisition
2. Synchronisation
3. Time Delay Estimation
4. Triangulation
5. User Interface

Additionally, all the subsystems will be contained within a host device which refers to a laptop used to host the local network for connection to the RPi microcontrollers. Moreover, the host device will be used to provide the GUI for the user to interact with the system.

6.4.2 Subsystem Requirements

6.4.2.1 Signal Acquisition

1. The Signal Acquisition subsystem shall capture audio signals from the 4 microphones.
2. The system shall incorporate noise reduction techniques to improve the SNR of the captured audio signals.
3. The subsystem shall provide communication between the RPi microcontrollers, and the host device used to provide an interface to the user.
4. The subsystem shall provide the processed audio signals used by the other subsystems.

6.4.2.2 Synchronisation

1. The subsystem should reduce the latency in data transmission and reception between the RPi microcontrollers and host device.
2. The subsystem shall establish precise time synchronisation between the two RPi microcontrollers.
3. The subsystem should include mechanisms to detect synchronisation failures.

6.4.2.3 Time Delay Estimation

1. The time delay estimation subsystem shall implement Time-Difference-of-Arrival (TDoA) algorithms to calculate the time differences between audio signal captures from different microphones.
2. The subsystem should be computationally efficient.

6.4.2.4 Triangulation

1. The triangulation subsystem shall employ an appropriate triangulation algorithm to convert the TDoA data into two-dimensional coordinates.
2. The subsystem shall be capable of determining the position of a stationary sound source within a rectangular grid.

6.4.2.5 User Interface

1. The User Interface Functionality sub-subsystem should provide real-time updates to the coordinate representation of the sound source’s location.
2. The User Interface shall be graphical.
3. The subsystem shall display a coordinate representation of the sound source's location.
4. The subsystem should be intuitive and user-friendly.
5. The subsystem should be designed to facilitate user interaction.

6.4.3 Subsystem Specifications

6.4.3.1 Signal Acquisition

1. The GUI will be programmed using a combination of the Python programming language and bash scripts.
2. The RPi Microcontrollers will be connected to the local network of a host device via Wi-Fi.
3. The host device will use the SSH protocol to transmit Unix commands simultaneously to the RPi Microcontrollers
4. The host device will use the SCP protocol to retrieve data from the RPi Microcontrollers.
5. The audio signals captured by the RPi Microcontrollers will be time synchronised using a calibration signal.
6. The calibration audio signal will be positioned at the centre of the grid.
7. The audio signals will be captured by the four Adafruit I2S MEMS Microphone breakout boards.
8. The microphones are omnidirectional.
9. The microphones have a frequency range of 50Hz - 15KHz.
10. The microphones will be set a sampling rate of 25kHz.
11. The microphones will be placed on the corners of the rectangular grid.
12. The microphones breakout boards will be connected to the RPi Microcontrollers via the I2S serial communication protocol.
13. Two microphones breakout boards will be connected to each RPi Microcontroller.
14. The captured audio signals will be stored on MicroSD cards connected to the RPi Microcontrollers.
15. The microphone breakout board has a rated SNR of 65dB.
16. The captured audio signals will be passed through a high pass filter, to filter out any frequencies lower than 1kHz and higher than 10kHz.

6.4.3.2 Synchronisation

1. The subsystem will be programmed using the Python programming language.
2. The audio signal data will be timestamped using the start of the audio signal captured from Mic 1 as a reference point.
3. A Generalized Cross-Correlation Phase Transform (GCC-PHAT) algorithm will be applied to the calibration signals acquired from each Mic with the calibration signal of Mic 1 as the reference.
4. The subsystem should provide a synchronisation accuracy within 10 microseconds.

6.4.3.3 Time Delay Estimation

1. The subsystem will be programmed using the Python programming language.
2. A Generalized Cross-Correlation Phase Transform (GCC-PHAT) algorithm will be applied to the source signals acquired from each Mic with the source signal of Mic 1 as the reference.
3. The TDoA values with respect to Mic 1 will be synchronised using the synchronisation delays provided the synchronisation subsystem.
4. The subsystem should provide TDoA values with an accuracy of at least 10 microseconds.

6.4.3.4 Triangulation

1. The subsystem will be programmed using the Python programming language.
2. A Linear Least Squares Estimation algorithm will be applied to the TDoA data using the coordinates of the microphones as reference points to produce an initial position estimate of the source signal.

3. A Non-Linear Least Squares Estimation algorithm will be applied to the TDoA data using the coordinates of the microphones as reference points to produce a final position estimate using the initial position estimate of the source signal.
4. The subsystem should provide a position estimate of the source signal with an accuracy of at least 1 cm.
5. The subsystem should provide a position estimate with a millimetre level precision.

6.4.3.5 User Interface

1. The GUI will be programmed using the Python programming language.
2. The GUI will be programmed using the PySimpleGui library for Python.
3. The GUI will allow the user to start and stop the acoustic triangulation process.
4. The GUI will display a coordinate grid with the predicted location of the sound source.
5. The GUI coordinate grid will reference the coordinates of the physical grid used.
6. The GUI will display the predicted two-dimensional Cartesian coordinates of the sound source.
7. The GUI will allow the user to view the audio signals captured by the microphone.

6.4.4 Inter-Subsystem Interactions

6.4.4.1 Synchronisation

1. The Pi Communication sub-subsystem (1.1) depends on the Signal Preprocessing (2.2) and User Interface Functionality (5.1) sub-subsystems.
2. Sub-subsystem 2.2. transmits the audio signal data of the audio signals captured and preprocessed by subsystem 2. This transfer is done via SCP.
3. Sub-subsystem 5.1 transmits the user input data captured by the User Interface. This transfer is done wirelessly via the SSH protocol over a local network.
4. The Pi Timing (1.2) sub-subsystem depends on sub-subsystem 1.1.
5. Sub-subsystem 1.1. transmits the time synchronisation data. This data is used to time synchronise the two RPi Microcontrollers.

6.4.4.2 Signal Acquisition

1. The Signal Capture subsystem depends on subsystem the 1.1 sub-subsystem.
2. Sub-subsystem 1.1 transmits the command to simultaneously capture the audio signals from all four microphones. The transmission to the microphones is done via the I2S protocol.
3. Sub-subsystem 2.2 depends on the Signal Capture sub-subsystem (2.1).
4. Sub-subsystem 2.1 transmits the captured audio signal data for preprocessing on the RPi Microcontrollers. This transmission is down via the I2S protocol.

6.4.4.3 Time Delay Estimation

1. The subsystem depends on sub-subsystem 5.1.
2. Sub-subsystem 5.1 transmits the audio signal data captured and preprocessed by subsystem 2. The audio signal data is used to calculate the TDoA of the captured audio signals at each microphone.

6.4.4.4 Triangulation

1. The sub-system depends on subsystem 3 and sub-subsystem 5.1.

2. Sub-system 3 transmits the TDoA data of the captured audio signals at each microphone. The TDoA data is used to calculate the two-dimensional Cartesian coordinates of the sound source.
3. Subsystem 5.1 transmits the user input data. The user input data is used to set the coordinates of the grid and microphones.

6.4.4.5 User Interface

1. The sub-subsystem depends on subsystem 4 and sub-subsystem 1.1.
2. Sub-system 4 transmits the localization data. This data provides the estimated position of the sound source.
3. Sub-subsystem 1.1 transmits the audio signal data captured and preprocessed by subsystem 2. This transfer is done wirelessly via the SCP protocol over a local network. The audio signal data is then retransmitted to subsystem 3 for further processing.
4. The subsystem depends on sub-subsystem 5.1.
5. Sub-subsystem 5.1 transmits the data to be displayed to the user by the GUI.

6.4.5 UML Diagram

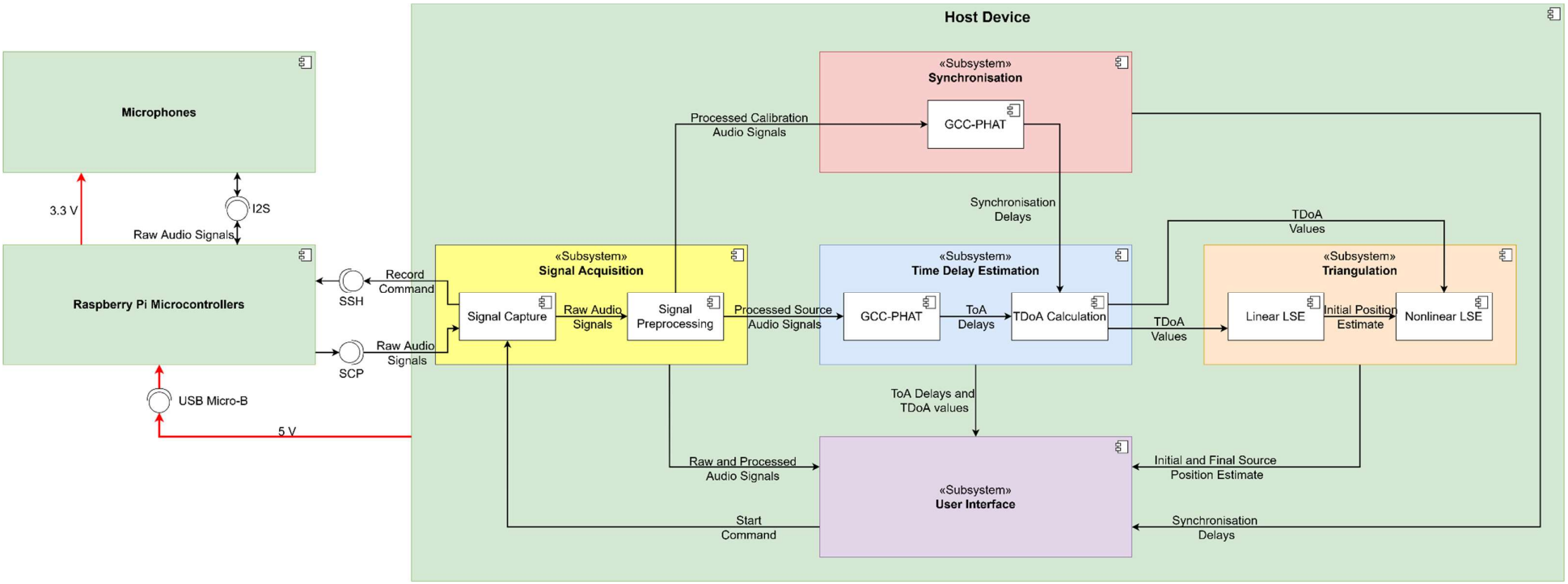


Figure 1: Component UML diagram of the subsystems

6.5 SUBSYSTEM ACCEPTANCE TEST PROCEDURE

Subsystem	Figures of Merit	Acceptance Test	Acceptable Performance
1. Pi Synchronisation			
1.1 Pi Communication	Successful ping message	To test this specification, both the Raspberry Pi Microcontrollers will be connected to each other, and then a 'ping' command will be sent from the console of one of the Pi's to the IP address of the other Pi and vice versa.	For acceptable performance, there must not be any loss of packets from either of the Pi's to each other. This ensures complete communication between the Pi's, hence confirming the utilisation of the system using two RPi Microcontrollers.
1.2 Pi Timing	SSH (Secure Shell) times for logging in of the RPi's	For this test, the time in which both the Pi's logged in using SSH (Secure-Shell Host) will be recorded.	For acceptable performance, the time in which both of the modules logged in has to be the same, this would signify a very small, negligible delay.
2. Signal acquisition			
2.1 Signal Capture	Signal acquisition plots	For this acceptance test, the signal acquisition plots will be analysed to look for expected signal plots.	For an acceptable performance, the signal acquisition plots will display a set of spikes for the calibration signal, then a after a moment of silence, shows a set of spikes for the source signal.
2.2 Signal Preprocessing	Signal to noise ratio (SNR)	For this acceptance test, the performance of the filter used to refine the audio signals will be used. Both the signal power and noise power of the input unfiltered signal, as well as the output filtered signal will be measured to calculate the input	For acceptable performance, the output SNR value should be higher than 50 dB. The filter will underperform if the SNR is below this minimum threshold. The microphones have a maximum SNR of 65 dB, so the closer the SNR is to this maximum

		SNR and output SNR, which will then be compared for performance.	value without distortion, the better the filter performance.
3. Time delay estimation	TDoA values	For this acceptance test, the TDoA values from each microphone will be obtained using the GCC-PHAT (Generalised Cross Correlation Phase Transform) algorithm and analysed. The ideal expected (calculated) TDoA will first be determined using the appropriate TDoA eqaution. The "measured" TDoA calculated from the parameters provided by the microphones will then be compared to the ideal expected TDoA for TDoA error.	For an acceptable performance, the TDoA algorithm will first have to be able to display four TDoA values, one for each microphone. These values must all be real, floating-point values. After that, the TDoA value will have to be accurate enough so that the triangulation algorithm can detect the sound source to as close to the original as possible. The average TDoA error should hence not exceed 10 microseconds.
4. Triangulation	Sound Source Coordinates outputted by the RPi microcontroller.	In this acceptance test, a pair of coordinates will be expected from the triangulation algorithm that will use the TDoA information from the previous algorithm to output a position relative to the rectangular grid used. The coordinates provided by the algorithm will then be compared to the actual position of the sound source to calculate position error.	For an acceptable performance, the triangulation algorithm will first have to be able to output the coordinates, and these coordinates have to be real, floating-point numbers. After that, it should be checked if the provided coordinates are within the bounds of the rectangular grid. Finally, the position error should be analysed. A position error of 1 centimetre, or 0.1 units, should not be exceeded.
5. User interface			

5.1 User Interface Functionality	Elements of the user interface, such as buttons, text boxes and display panels.	Each input and output element will be tested for functionality and correctness. The inputs will be executed numerous times, and the output of these inputs will also be recorded, to determine whether the input achieves the desired output/function.	Each element in the interface needs to behave as desired, and it must do so in a responsive, fast manner.
5.2 User Interface Design	Overall layout and outlook of the user interface, as well as ergonomics	The user interface will be tested in its ergonomic aspect and ease of use. A volunteer, with no prior knowledge of the project, will be asked to use the program and comment/rate on how easy or difficult the interface is to use and understand.	An average rating of 7 or above given by the volunteers will be deemed acceptable performance.

6.6 PROJECT TIMELINE

The Work Breakdown Structure (WBS) of the project is displayed below:

Code	Activity	Duration	Dependency
1	Proposition of project idea and commencement of planning	28/07/2023 - 31/07/2023	None
2	Begin background research on aspects of project	31/07/2023 - 01/08/2023	1
3	Complete MATLAB Onramp courses	05/08/2023 - 14/08/2023	1
4	Identification of project requirements	31/07/2023 - 02/08/2023	1
5	Analysis of project requirements	02/08/2023 - 04/08/2023	4
6	Development of specifications	07/08/20223 - 10/08/2023	4, 5
7	Subsystem Breakdown	10/08/2023 - 11/08/2023	4, 5, 6
8	Write up of inter- and intra-subsystem interactions	14/08/2023 - 16/08/2023	7

9	Define acceptance test procedures	16/08/2023 - 18/08/2023	8
10	Setup of simulation using MATLAB	21/08/2023 - 29/08/2023	3, 6, 8
11	Testing of MATLAB simulation	30/08/2023 - 01/09/2023	10
12	Setup of physical system	11/09/2023 - 12/09/2023	11
13	Pi synchronisation setup	13/09/2023 - 18/09/2023	12
14	Development of code for signal acquisition and preprocessing	13/09/2023 - 22/09/2023	11
15	Development of code for time delay estimation	13/09/2023 - 22/09/2023	11
16	Development of code for triangulation/localisation algorithm	13/09/2023 - 22/09/2023	11
17	Development of user interface	13/09/2023 - 22/09/2023	3
18	Testing and debugging of Pi synchronisation	14/09/2023- 18/09/2023	13
19	Testing and debugging of signal acquisition and preprocessing	22/09/2023 - 31/09/2023	14
20	Testing and debugging of time delay estimation	22/09/2023 - 31/09/2023	15
21	Testing and debugging of triangulation/localisation algorithm	22/09/2023 - 31/09/2023	16
22	Testing and debugging of user interface	22/09/2023 - 29/09/2023	17
23	Overall performance evaluation	31/09/2023 - 02/10/2023	13, 14, 15, 16, 17
24	Final Testing	02/10/2023 - 04/09/2023	23
25	Final Report Write-up	04/09/2023 - 18/09/2023	24

Table 1: The Work Breakdown Structure of the project

The Gantt chart for the development timeline is shown below:

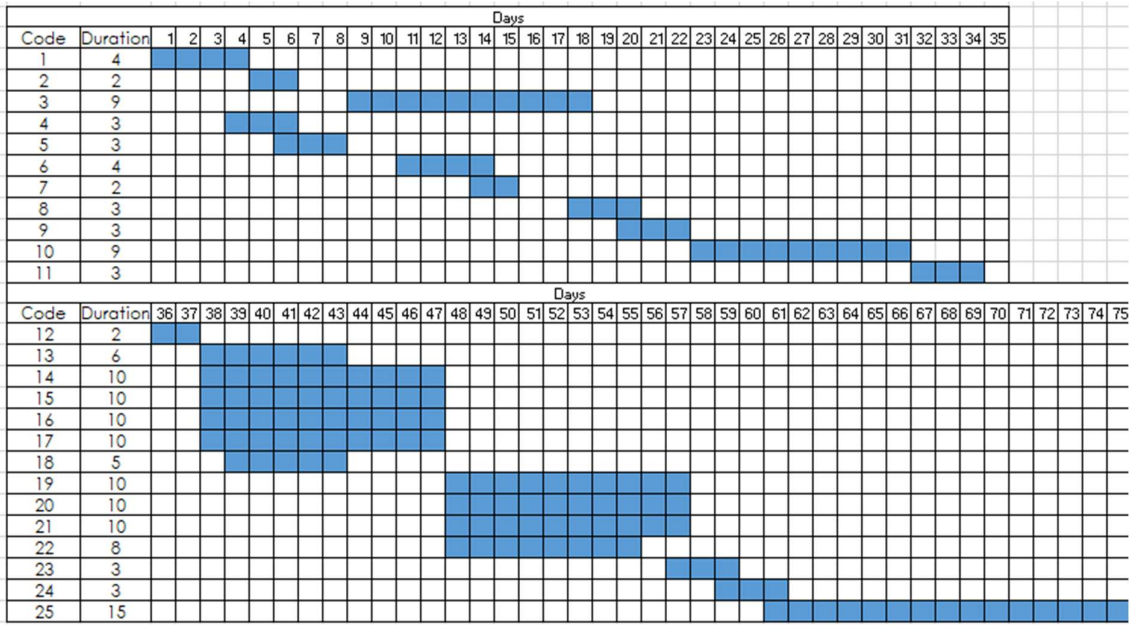


Figure 2: Gantt chart of the project development timeline

7 VALIDATION USING SIMULATIONS

7.1 MOTIVATION

The overall system and subsystems were implemented in a simulation before the final hardware-based implementation. This was done to validate the proposed system and subsystem implementations prior to the hardware-based implementation. The reason for this is that it is easier to find errors and adjust the simulation as opposed to the hardware-based implementation. Thus, the simulation provides an efficient means of improving the proposed system and subsystems prior to the final implementation.

7.2 PROCEDURE LIST

The simulation-based validation was done according to the following steps:

1. The simulation environment was defined.
2. The simulation approach was defined.
3. The simulation assumptions were defined.
4. The simulated system architecture was defined.
5. The simulated subsystems were developed according to their requirements and specifications.
6. The simulated subsystems were integrated into the overall system simulation according to its requirements and specifications.
7. The simulated system and subsystems were tested to evaluate their performances.
8. The results of the tests performed were analysed and discussed.

7.3 PROCEDURE DESCRIPTION

7.3.1 Simulation Environment

The simulation was designed and implemented using MATLAB in which the system simulation was implemented using a MATLAB function called **simulation.m**. Furthermore, a GUI, which provides a visual interface for running the simulation, and test function, which runs the simulation for a range of input parameters, were also implemented using MATLAB. These are called **GUI.m** and **test.m** respectively. The test function also records the results in a file called **results.csv**.

Additionally, the following MATLAB addons were used to implement the simulation function:

- Communication Toolbox,
- Optimization Toolbox,
- Phased Array System Toolbox,
- DSP System Toolbox,
- Signal Processing.

7.3.2 Simulation Approach

The Acoustic Triangulation using Time Difference of Arrival and a distributed sensor network system simulation was modelled similarly to the system presented in [1]. This is because the system being implemented is similar to the system presented in [1].

The main simulation function simulates the Signal Acquisition, Pi Synchronisation, Time Delay Estimation and Triangulation subsystems. The overall system consists of the two RPi Zero W modules, four Adafruit I2S MEMS Microphone breakout boards, (0.8 x 0.5 m) grid and the host device. As mentioned previously, the host device refers to the device used to connect to the RPi Microcontrollers and provide the GUI for the user to interact with the system. The system also consists of two audio signal transmitters for transmitting the calibration and source audio signals.

The following factors were considered when designing the simulation:

- The speed of sound.
- The dimensions of the grid.
- The positions of the calibration signal and Mics.
- The error in the positions of the calibration signal and Mics.
- The sample rate of the Mics.
- The white noise present on the audio signals produced by the Mics.
- The maximum frequencies of the calibration and source chirp signals.
- The cutoff frequency of the lowpass filter applied to the signals produced by the Mics.
- The desynchronisation delays in communication to the two RPi Microcontrollers.
- The type of synchronisation used to synchronise the signals produced by the Mics.
- The type of cross correlation function used for time difference estimation.
- The type of triangulation algorithm used by the system.

These were considered the most significant features to adequately model the system and subsystems defined by requirements and specifications.

The Signal Acquisition subsystem was modelled as the expected signals received from each Mic and the post processing of these signals. Furthermore, effects of position errors of the Mics and calibration signals, desynchronisation, and additive white noise on the signals was considered.

The Synchronisation subsystem was modelled as the post processing done on the signals to synchronise them to a common starting reference. This was done using a calibration signal with a known position and thus known ToA at each Mic.

The Time Delay Estimation subsystem was modelled as the estimation of the TDoA values of the synchronised signals using a cross correlation algorithm.

The Triangulation subsystem was modelled as the estimation of the source location using the estimated TDoA values and the known positions of the Mics.

The creation of a GUI for the simulated system was also facilitated by tools provided by MATLAB. Thus, the User interface subsystem could be modelled as the GUI used to interface with the simulated system.

This simulation approach was used because MATLAB provides a diverse set of toolkits, which make it possible to adequately simulated the system and subsystems. Furthermore, the simulation approach focuses on the requirement and specification of the systems and subsystems. Additionally, it provides as much accuracy as is reasonably achievable. Thus, the simulation approach allows for the system to provide useful results for the final implementation.

7.3.3 Simulation Simplifications

The following simplifications were made:

- A constant of 343 m/s was used for the speed of sound.
- The source audio signal was assumed to be stationary.
- The source audio signal was generated as a 5 second 0 to 100 Hz chirp signal with a constant amplitude of 1.
- The calibration audio signal was generated as a 5 second 0 to 1000 Hz chirp signal with a constant amplitude of 1.
- A sampling rate of 48 kHz was assumed for the Mics.
- The SNR of the audio signals was kept at 65 dB by default as rated by the Mics [9].
- Additive white noise was the only form of signal noise considered.
- It was assumed that the errors in the positions of the Mics and calibration signal were zero mean Gaussian distributed with a standard deviations of 1 mm.
- Errors in the clocks of the RPi Microcontrollers were not considered.
- It was assumed the RPi Microcontrollers would achieve synchronisation within 10 ms using NTP synchronisation [1].
- It was assumed that the host device is a computer running the Microsoft Windows OS
- It was assumed that all signal processing will be performed on the host device.

These simplifications greatly reduce the complexity of the implementation of the simulated system and subsystems. However, they also reduce the accuracy of the simulation. Thus, the assumptions made were kept as limited as possible. Some of the assumptions made can be applied to the final implantation as well which reduces their impact. The non-ideal conditions that were not considered will have a larger effect as they cannot be applied to the final implementation. Therefore, as many non-ideal effects as were technically feasible were implemented to limit the effect of the assumptions made.

7.3.4 Simulation System Design and Implementation

7.3.4.1 Simulation System Architecture

The architecture of system simulated is shown in the figure below:

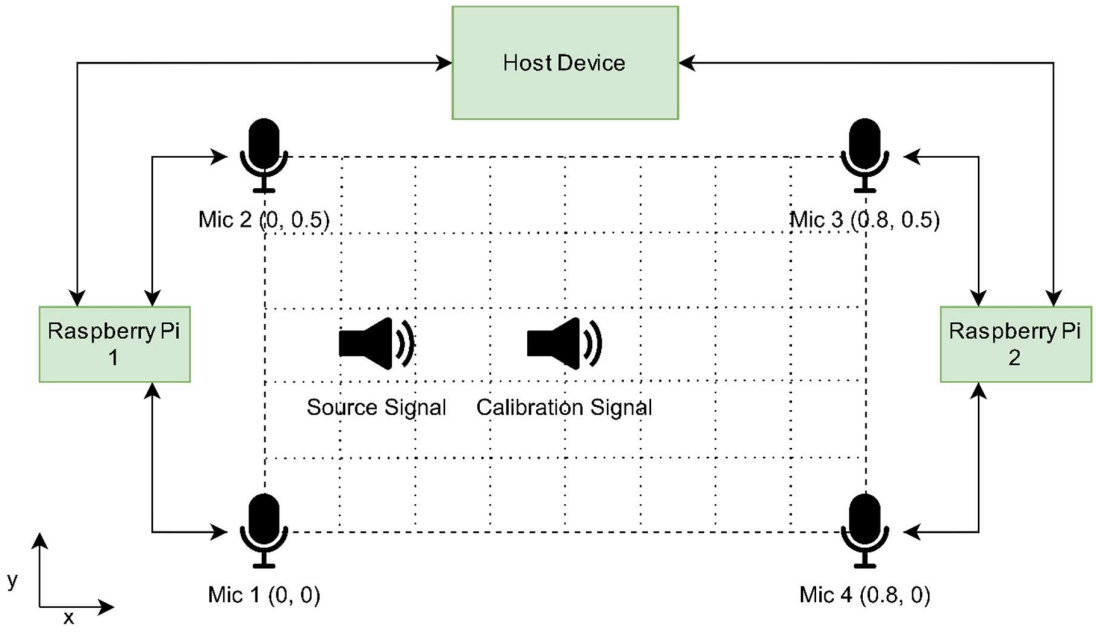


Figure 3: Architecture of the system that was simulated

The figure shows the system architecture modelled by the simulation. The distributed sensor array consists of the 4 Mics positioned at the corners of the grid. The Mics on the left and right edges of the grid are connected to their respective RPi Microcontrollers. The 2 RPi Microcontrollers are then connected to the host device via a local network. The host device is the laptop used to host the local network and interface the system with the user.

7.3.4.2 Simulation Parameters

The simulation has the following required parameters:

- **x (m)**: the actual x-coordinate of the sound source.
- **y (m)**: the actual y-coordinate of the sound source.

The simulation has the following optional parameters with default values:

- **SNR (dB)**: the Signal to Noise ratio of signals received from the Mics.
- **Sample Rate (Hz):** the sampling frequency of the signals received from the Mics.
- **Cutoff Frequency (Hz):** the cutoff frequency of the lowpass filter applied to the signals received from the Mics.
- **Latency (s)**: the desynchronisation delay applied to communication to the two RPi Microcontrollers.
- **Calibration Position Error (m):** the error factor of the position of calibration signal.
- **Mic Position Error (m)**: the error factor of the positions of the Mics.
- **c**: the speed of sound (m/s)
- **Grid Size (m):** the dimensions of the grid
- **Calibration Position (m):** the position of the calibration signal.
- **Mic Positions (m)**: the positions of the Mics.

7.3.4.3 Signal Acquisition

The signal acquisition subsystem was simulated in two steps. Firstly, the expected signals from each Mic were generated to represent the Mic recordings. Secondly, the generated signals were passed through a lowpass filter to reduce the high frequency noise on the signals. Low frequencies were not filtered because the calibration and source signals both contain low frequencies.

The signal generation makes use of a source signal and calibration signal. These are both 5 second chirp signals. Chirp signals were selected as they are optimal for computing time delays using cross correlation as mentioned in [1], although the algorithm will also work for regular sinusoidal signals.

The signal generation makes use of the actual positions of the calibration signal and Mics. These are the sum of the known positions and their respective errors. By default, the position errors are modelled as Gaussian distributed with a means of 0 m and standard deviations of 1 mm. The standard deviations are determined by the calibration position error and Mic position error parameters. Thus, the positions of the source and calibration signals and Mics are used to calculate the ToA of the signals at each Mic.

The signals are generated assuming a 2 second delay between the start of recording and the transmission of the calibration signal. Furthermore, a 2 second delay is assumed between the transmission of the calibration signal and the source signals. As the Mics are connected in pairs, a latency difference was applied between the signals from Mics 1 and 2 and the signals from Mics 3 and 4. This emulates the signals being in sync with their stereo counterparts but not being in sync with the microphones connected to the other RPi module. The latency difference was implemented by having the recordings from Mics 3 and 4 start later than the recordings from Mics 1 and 2. By default this is assumed to be 10ms as this is the expected synchronisation of NTP synchronisation [1]. The latency difference can be varied however it should be noted that it cannot exceed approximately 2s as this would result in the recording starting after the calibration signal has been transmitted.

Lastly, the signals are summed with additive white noise. This represents the expected noise on the Mic signals. Only additive white noise was considered due to technical limitations. The default SNR of signals is 65 dB as rated by the Adafruit I2S MEMS Microphone breakout boards [9].

The lowpass filter was implemented using the MATLAB lowpass function. The lowpass function uses a minimum-order filter with a stopband attenuation of 60 dB and compensates for the delay introduced by the filter. The default cutoff frequency was set to 10kHz which is a decade higher than the maximum frequency present in the signals. This was designed so that the filter caused minimal distortion on the signals while still significantly reducing high frequency noise.

To showcase the effectiveness of this subsystem, an experiment was run. In terms of various parameters, the input SNR was set to 30 dB. The maximum source and calibration frequencies were set to 1 Hz and 10 Hz respectively. The cutoff frequency of the lowpass filter was set to 100 Hz. The latency difference was set to 1s. Plots were taken at key stages to illustrate the execution of the subsystem.

Additionally, the performance of the subsystem was tested by varying individual signal and filter parameters and recording the effects on the overall system accuracy. The measurement of the SNR of the processed audio signals is the primary indicator of the performance of this subsystem. However, this was overlooked in the design of the subsystem and thus there were no results for this measurement. Consequently, the performance of the simulation of this subsystem could not be fully evaluated. This was addressed in the final implementation.

7.3.4.4 Synchronisation

Signal synchronisation was simulated via post-processing using a calibration signal. As mentioned before, the simulation does not take clock non-idealities into account. This includes clock jitter and clock drift. The delays are estimated by taking the GCC-PHAT cross correlation of the signals with the calibration signal and subtracting the known ToA values of the calibration signal from this. The signals are then shifted to remove the estimated delays. Subsequently, the calibration signal is removed from the signals by replacing it with zeros. The ends of the signals are padded with zeros to equalize their lengths. Thus, the resulting signals are thus synchronised to start at approximately the same reference point.

For simulating this subsystem, the same experimental parameters as the signal acquisition experiment were used, but with the SNR set to 65dB. Additionally, the position of the source was at the centre of the grid (0.4, 0.25). A plot of the synchronised signals was taken to display the results of this subsystem.

Additionally, the synchronisation errors of the signals were also recorded for all the tests performed on the overall system. This was used to evaluate the performance of the subsystem. The synchronisation error was defined as the difference between the estimated synchronisation delays and the actual synchronisation delays.

7.3.4.5 Time Delay Estimation

The time delay estimation is done using GCC-PHAT cross correlation of the signal from Mic 1 with the signals from the other Mics. This gives an estimate of the TDoA values between Mic 1 and the other Mics. This assumes that the signals have already been correctly synchronised and the calibration signal has been removed.

For the simulation of this subsystem, the synchronised signals from the synchronisation experiment ran previously were the input signals to the GCC-PHAT cross correlation function. A plot of the GCC-PHAT cross correlations of the signals was taken to display the results.

Additionally, the TDoA errors were also recorded for all the tests performed on the overall system. This was used to evaluate the performance of the subsystem. The TDoA error was defined as the difference between the estimated TDoA values and the actuals TDoA values.

7.3.4.6 Triangulation

Triangulation is done using a non-linear least squares estimation. The default trust-region-reflective algorithm of the MATLAB lsqcurvefit function is used. A function is defined that takes in the coordinates of the sound source and the positions of the Mics and outputs the differences in the distances from Mic 1 to the source and the other Mics to the source. The lsqcurvefit function then estimates the source position using the estimated TDoA values multiplied by c as the distances. Furthermore, this is done by iterating an initial position until if matches the expected distance values. The centre of the grid (0.4, 0.25) is used as the initial position as it is closest to all positions on the grid. Additionally, a lower and upper bound are placed on the solution. The lower bound is at the origin and the upper bound is at the maximum coordinates of the grid.

The default parameters were used in the experimentation of this subsystem, along with the position of the sound source being set to (0.8,0.5), which is at the far-right corner of the grid. The iterations of nonlinear least squares estimation algorithm were plotted to illustrate the steps taken to reach the final estimated source position.

Additionally, the position error which was used to measure the performance of the overall system is also the primary indicator of the performance of this subsystem. Thus, the results presented for the overall system also apply to the triangulation subsystem. Consequently, the results were not restated for this subsystem.

7.3.4.7 Graphical User Interface

The GUI was implemented to provide a visual interface for the simulated system. The components of the GUI include a (0.8 x 0.5 m) grid, a legend of the points plotted on the grid, buttons to interact with the simulation and the resultant of values of the simulation performed. The overall GUI is shown in the figure below:

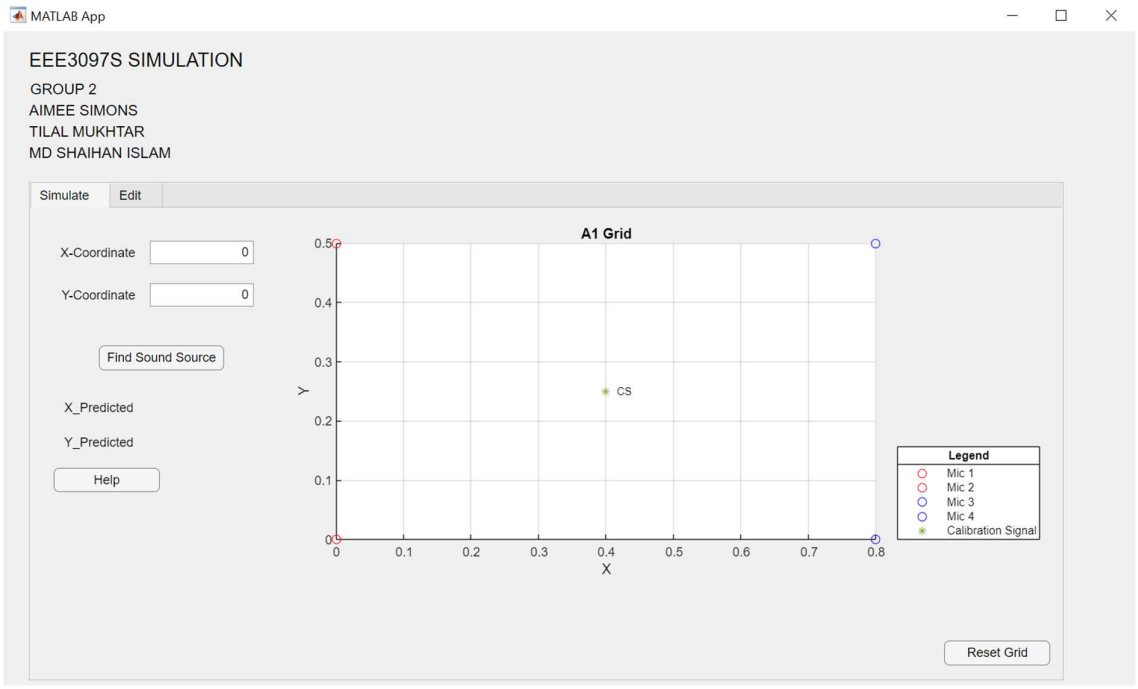


Figure 4: Simulation GUI

The user can input the coordinates which represent position of the source audio signal and press the button labelled “Find Sound Source” to initiate the simulation. The GUI will pass the given parameters to the simulation and display the results. This is shown in the figure below:

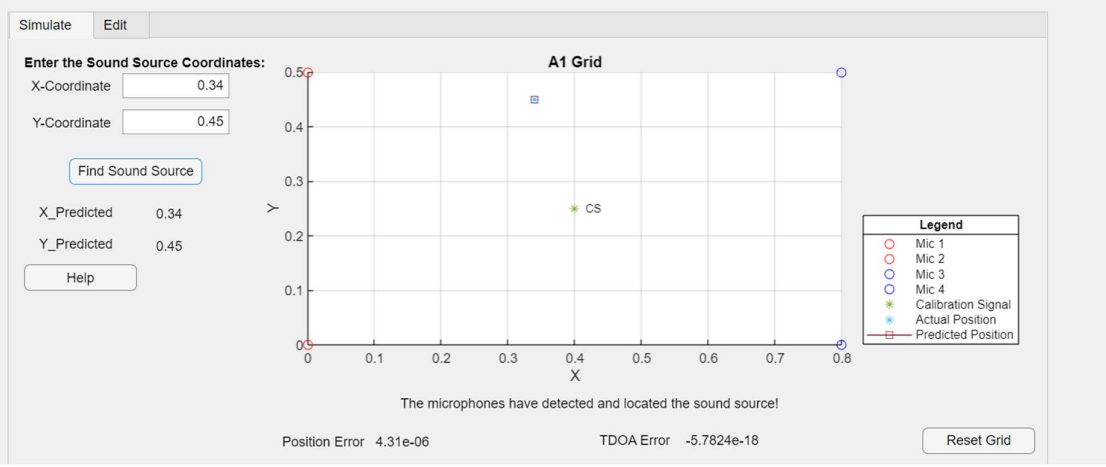


Figure 5: Showing the Actual and Predicted position on the simulation grid.

The GUI displays the actual source position and the estimated source position on the grid. The provided labels display the estimated coordinated, the position error, and the TDoA error obtained from the simulation. The grid can be reset by pressing the “Reset Grid” button. Additionally, the simulation also accounts for invalid coordinates provided, which represents the estimated position of the source signal being found outside the grid. The error message is shown in the figure below:

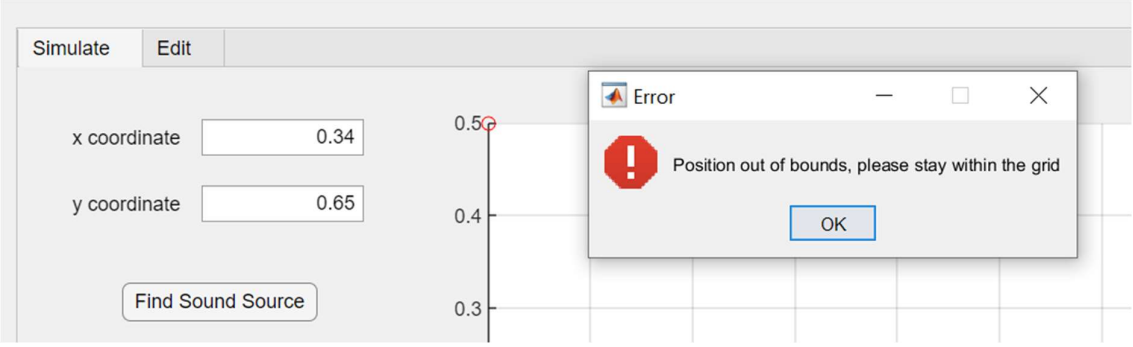


Figure 6: Error message when the simulation coordinates are out of bounds.

The optional parameters passed to the simulation by the GUI can be change from their default values using the “Edit” tab. This is shown in the figure below:

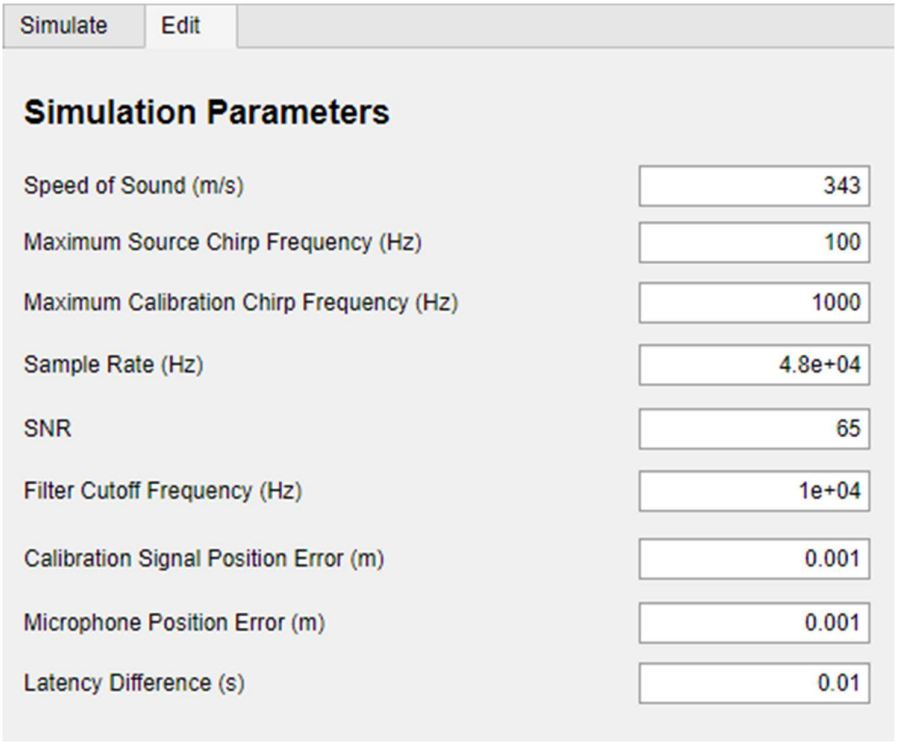


Figure 7: Change to optional simulation parameters

Lastly, the “Help” button was added to provide assistance to the user thereby making the GUI more user friendly.

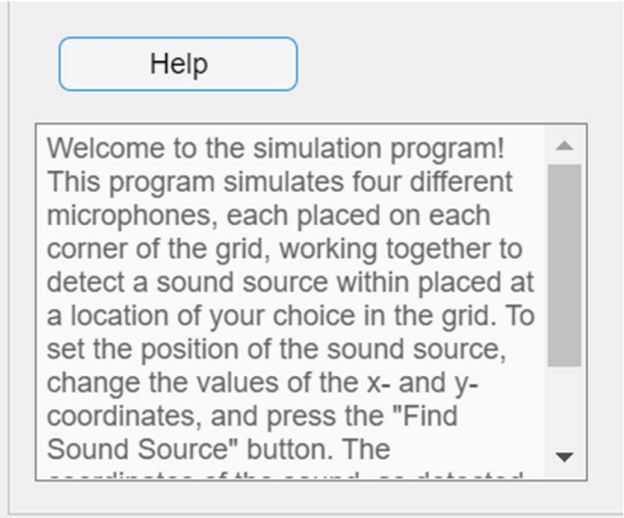


Figure 8: Help button of the simulation GUI, to aid the user.

7.3.5 Overall System

The overall system simulation consists of the combination of all the subsystems discussed. Its primary input parameter are the coordinates of the sound source. Its primary output parameters are the estimated coordinates of the sound source. As mentioned before, the **test.m** function was used to test the performance of the overall system and subsystems. The test function first records the performance of the simulated system using default parameters. It varies the position of the sound source over the entire grid in steps of 0.0005 m or 0.5 mm. The estimated positions and corresponding system errors are recorded in the *results.csv* file. Thereafter tests are performed by varying individual simulation parameters. All other parameters are kept at default and the source position is kept at the centre of the grid. The results of these tests are also recorded in the **results.csv** file.

7.4 RESULTS

7.4.1 Overall System

The position error of the estimated position was used as the primary measure of the performance of the overall system under various conditions.

7.4.1.1 Default Parameters

The simulation was run with default parameters for varying sound source positions across the entire grid. The results are shown in the figure below:

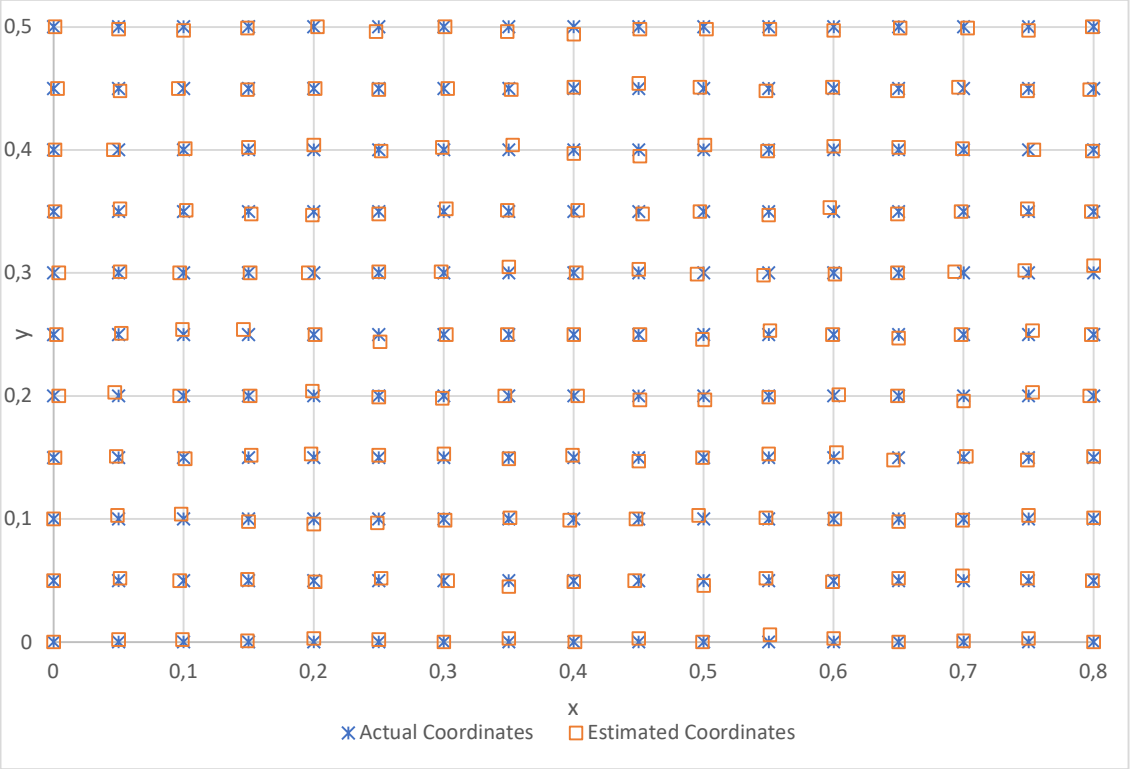


Figure 9: Graph of actual coordinates versus estimated coordinates produced by the simulation.

It can be observed that Figure 9 illustrates the simulated accuracy of the system. The position error was defined as the distance between the actual position and the estimated position. The average position error was 0.002567 m or 2.567 mm.

Mean Position Error (mm)	2.567
Median Position Error (mm)	2.236
Minimum Position Error (mm)	0
Maximum Position Error (mm)	7.071

Table 2: Statistics of the position error for the simulation

The precision of the estimated coordinates was limited to 3 decimal places in the x and y coordinates as this provides millimetre level precision. Moreover, it is unlikely that a higher precision would be achievable for the physical system.

7.4.1.2 Varying SNR

The default input SNR used was 65dB as rated by the Adafruit I2S MEMS Microphone breakout boards [9]. However, this does not take external environmental factors into account. Thus, the simulated performance of the system was tested for varying SNR values from 40 to 80 dB.

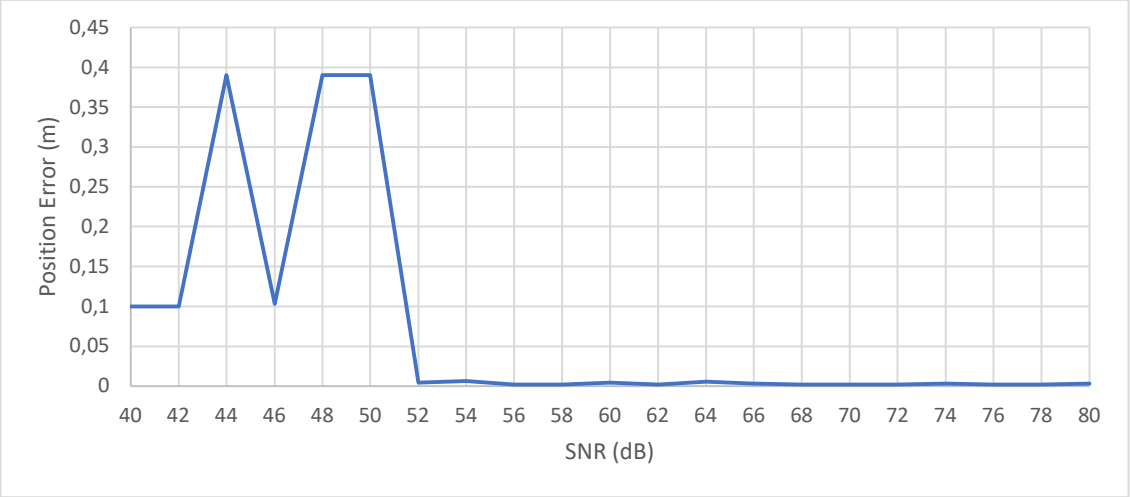


Figure 10: Graph of position error as a function of input SNR for the simulation.

Figure 10 shows that the simulated system performs optimally for input SNR values greater than 52 dB. Moreover, the performance greatly deteriorates and becomes inconsistent for input SNR values less than 52 dB. Thus, for optimal performance environmental conditions should not decrease the input SNR below 52 dB. This provides insight into the limited application of the system to different environments.

7.4.1.3 Varying Cutoff Frequency of Lowpass Filter

The cutoff frequency of the lowpass filter determines the attenuation of high frequency noise. A lower cutoff frequency has the potential for increased attenuation of high frequency noise. However, if the cutoff is placed to low then it may distort the calibration or source signal. The simulated performance of the system was tested for varying cutoff frequencies from 3 to 20 kHz.

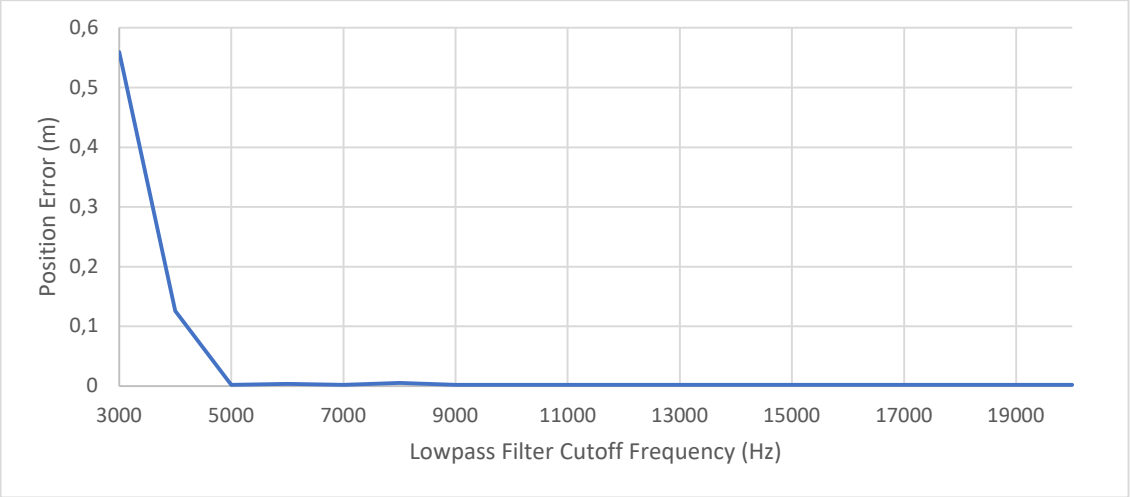


Figure 11: Graph of position error as a function of the lowpass filter cutoff frequency for the simulation.

It can be observed that Figure 11 illustrates that the position error greatly increases for cutoff frequencies below 5 kHz. This can be explained by the fact that the highest signal frequency is 1 kHz

which is too close to frequencies below 5 kHz. The lack of increasing position error for larger cutoff frequencies may be due to the relatively high input SNR of 65 dB. The tests will need to be repeated for a lower input SNR to further determine the ideal cutoff frequency of the lowpass filter.

7.4.1.4 Varying Latency

The latency difference is a measure of the desynchronisation between the two RPi modules. The synchronisation error was defined as the average effect of desynchronisation on the TDoA values. The system performance was tested for latency differences between the RPi modules from 0.001 to 1.024 seconds. The results are tabulated below:

Latency Difference (s)	Position Error (m)
0,001	0,000008
0,002	-0,000027
0,004	0,000001
0,008	0,000008
0,016	0,000008
0,032	0,000008
0,064	0,000008
0,128	0,000008
0,256	0,000008
0,512	0,000008
1,024	0,000008

Table 3: Position error as a function of the latency between the RPi Microcontrollers for the simulation

Table 3 shows that the position error of the system remains optimal regardless of the values of latency difference or synchronisation delay. This means that the performance of the system will not degrade as the desynchronisation between the RPi Microcontrollers increases. Consequently, additional synchronisation methods are not necessary. However, it should be noted that if the latency becomes larger than 1 second it may cause the audio signals to be clipped at their beginnings or ends. This is recording may start too early or too late to capture the entire signal. This can be minimised by increasing the size of the recording relative to the duration of the audio signal. Additionally, the implementation of NTP synchronisation should prevent the latency from becoming too large. Thus, the effect of signal desynchronisation on the overall system is minimal with the other parameters kept default.

7.4.1.5 Varying Calibration Signal Position Error

The calibration signal position error factor represents the standard deviation of the error in the position of the calibration signal. The default standard deviation is kept at 1mm by default. The effect of varying the standard deviation from 0.001 to 0.02 m was tested. The results of the tests are shown below:

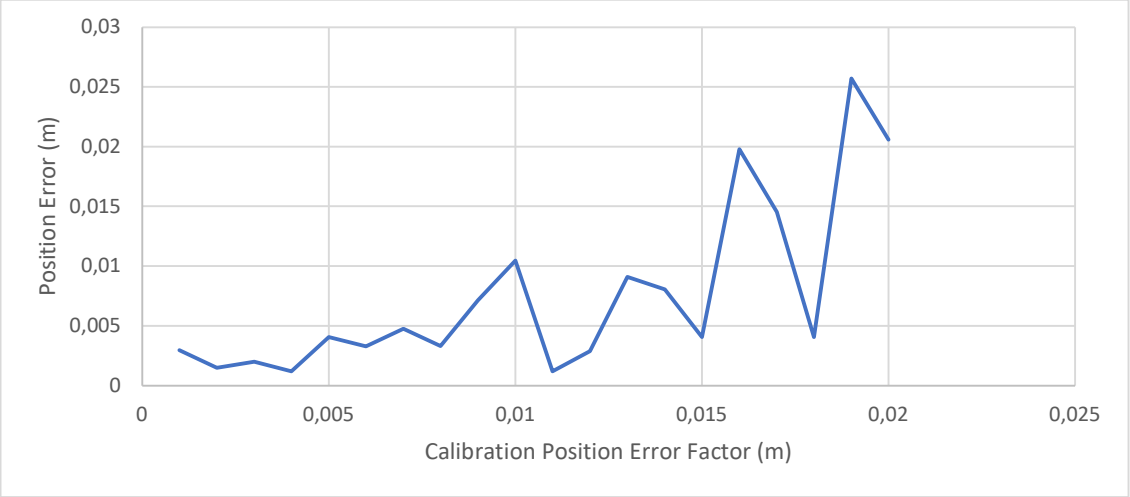


Figure 12: Graph of position error as a function of the calibration signal position error factor for the simulation.

Figure 12 shows a general trend of the mean position error increasing with an increased standard deviation. Furthermore, the standard deviation of the position error also appears to increase. This is expected as an error in the position of the calibration signal directly affects the signal synchronisation. An increase in the synchronisation error will result in an increase in the position error. Thus, the effect of the calibration signal position error on the system performance is significant.

7.4.1.6 Varying Microphone Position Error

The microphones also have a position error associated with them. This is because the microphones might not be located exactly at the corners of the grid. The Mic position error factor represents the standard deviation of the error in the positions of the Mics. The default standard deviation is kept at 1mm by default. The effect of varying the standard deviation from 0.001 to 0.02 m was tested. The results of the tests are shown below:

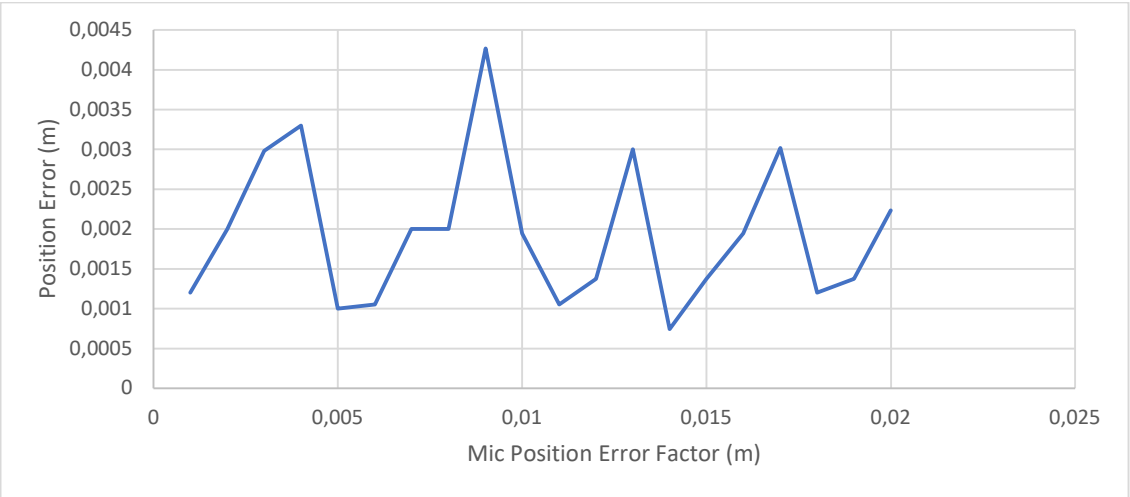


Figure 13: Graph of position error as a function of the Mic position error factor for the simulation.

From the graph above, it can be seen that varying the position error factor of the microphones, causes erratic and unpredictable position errors. There is no discernible correlation between the Mic position error and the source position error over the range of values tested. This may be due to the fact that the Mic position error is applied independently to each Mic. Thus, it is unlikely that these

errors always add together constructively. This makes the effects of the Mic position errors difficult to predict. Therefore, it would be prudent to minimise the Mic position errors in the actual system.

7.4.1.7 Varying Source Signal Frequency

The frequency of the source signal was selected to go from 0 to 100 Hz. The system was tested in order to determine whether this selection was optimal and its effect on the position error of the source. For this test, the maximum frequency of the source signal was varied from 10 Hz to 10240 Hz.

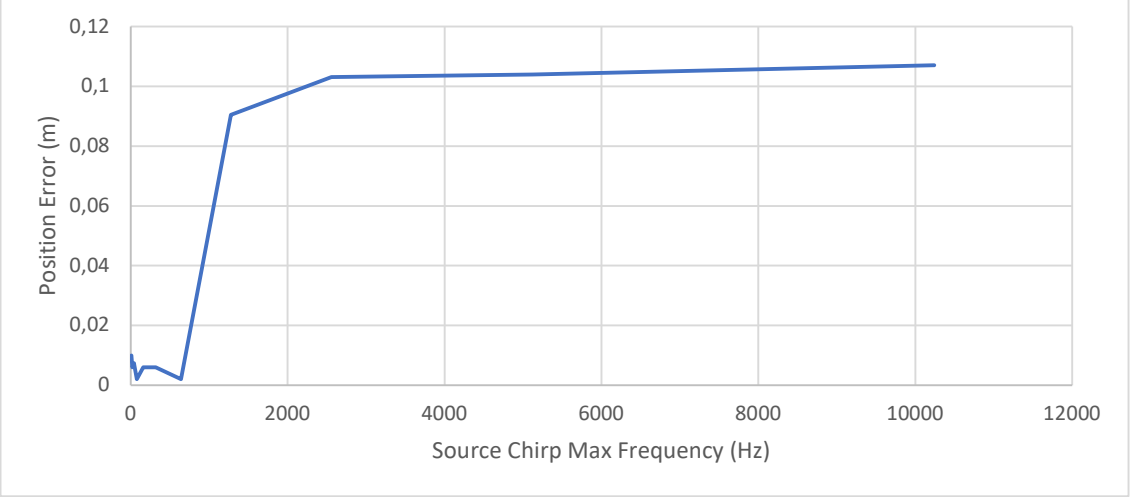


Figure 14: Graph of the position error as a function of the maximum frequency of the source chirp signal for the simulation.

Graph 14 shows that the optimal maximum source frequencies occur below approximately 1 kHz. This is while the cutoff frequency of the lowpass filter is 10 kHz and the maximum calibration frequency is 1kHz. Thus, these factors might be causing the deterioration of the performance at higher frequencies.

7.4.2 Signal Acquisition

The performance of the signal acquisition subsystem simulation can be illustrated by taking plots at the different stages of its execution. In order to ease visibility, the default simulation parameters were altered. The input SNR was reduced to 30 dB to make noise more visible. The maximum source and calibration frequencies were reduced to 1 Hz and 10 Hz respectively. Consequently, the cutoff frequency of the lowpass filter was reduced to 100 Hz. Lastly, the latency difference was increased to 1s to make the signal desynchronisation more visible.

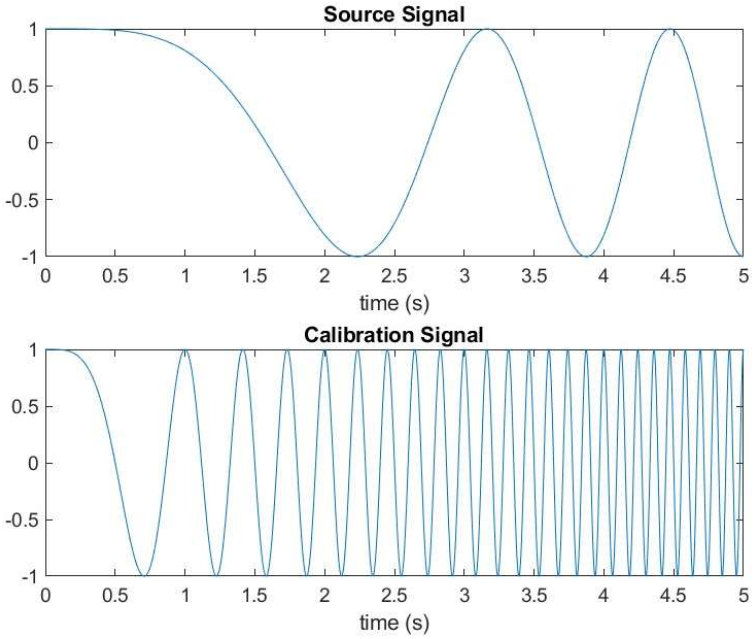


Figure 15: Plots of the source and calibration signals for the simulation.

Figure 15 shows the source and calibration signals used for the simulation. It can be observed that they are both 5 second chirp signals. Furthermore, the calibration signal has a maximum frequency ten times greater than that of the source signal. Their amplitudes are the same as the simulation does not take any differences in amplitude into account.

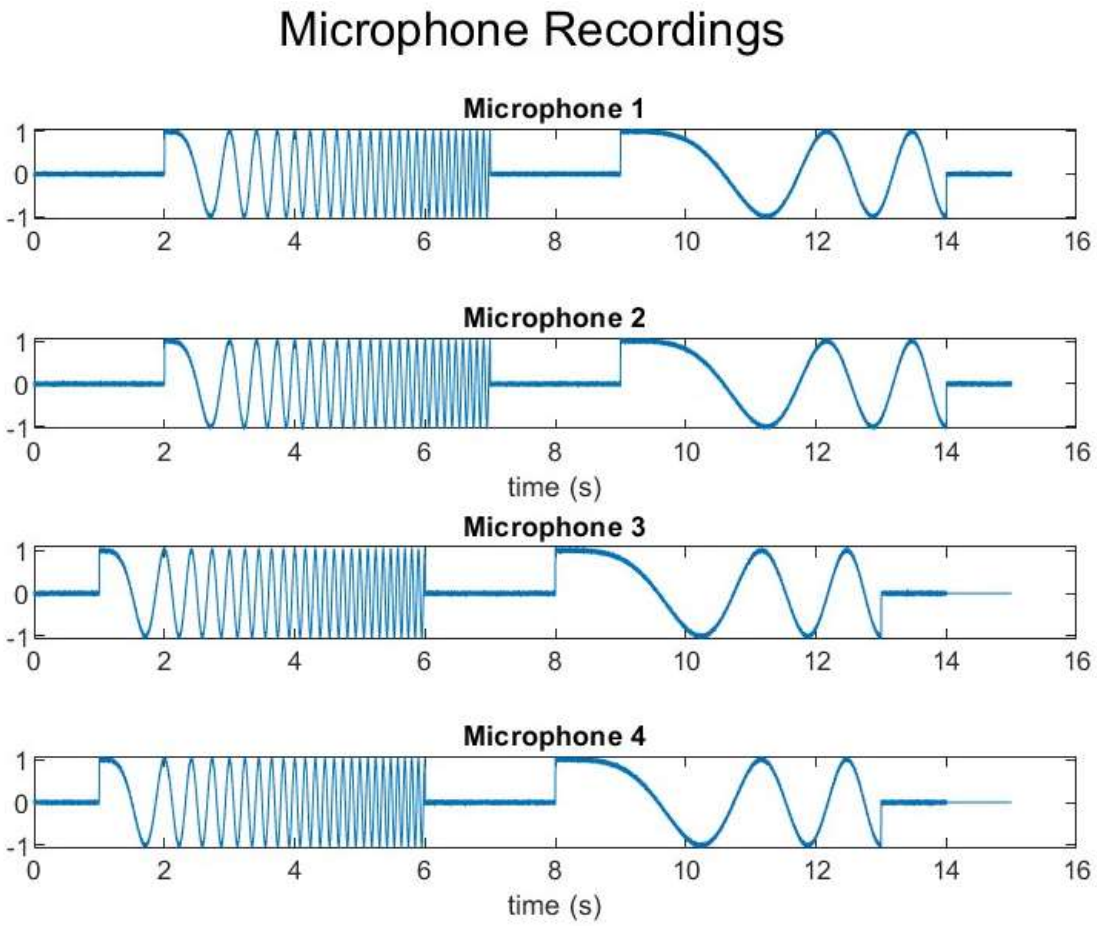


Figure 16: Plots of the Microphone recording signals generated for the simulation.

Figure 16 shows the expected signals generated for each Mic. It can be observed that the signals contain the calibration signal followed by the source signal. Moreover, the 1 second desynchronisation between the recordings from Mics 1 and 2 and Mics 3 and 4 is visible. The recordings from Mics 3 and 4 start 1 second later. The additive white Gaussian noise on the signals can also be clearly seen. The ToA values of the calibration and source signals are too small to be observed at the scale of the grid.

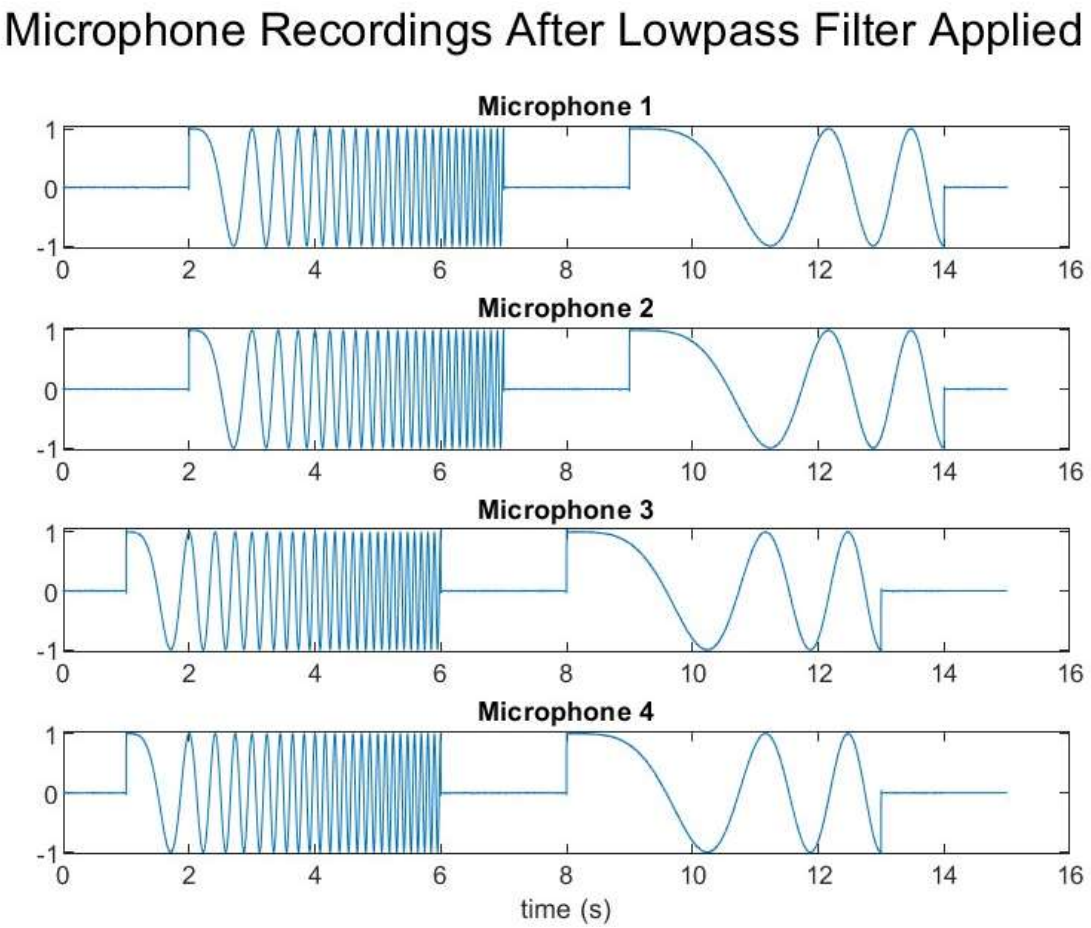


Figure 17: Plots of the Microphone recording signals after the lowpass filter has been applied for the simulation.

Figure 17 shows that the lowpass filter has substantially reduced the noise on the microphone signals. Additionally, the distortion of the signals is minimal and there does not appear to be any delays introduced by the filter.

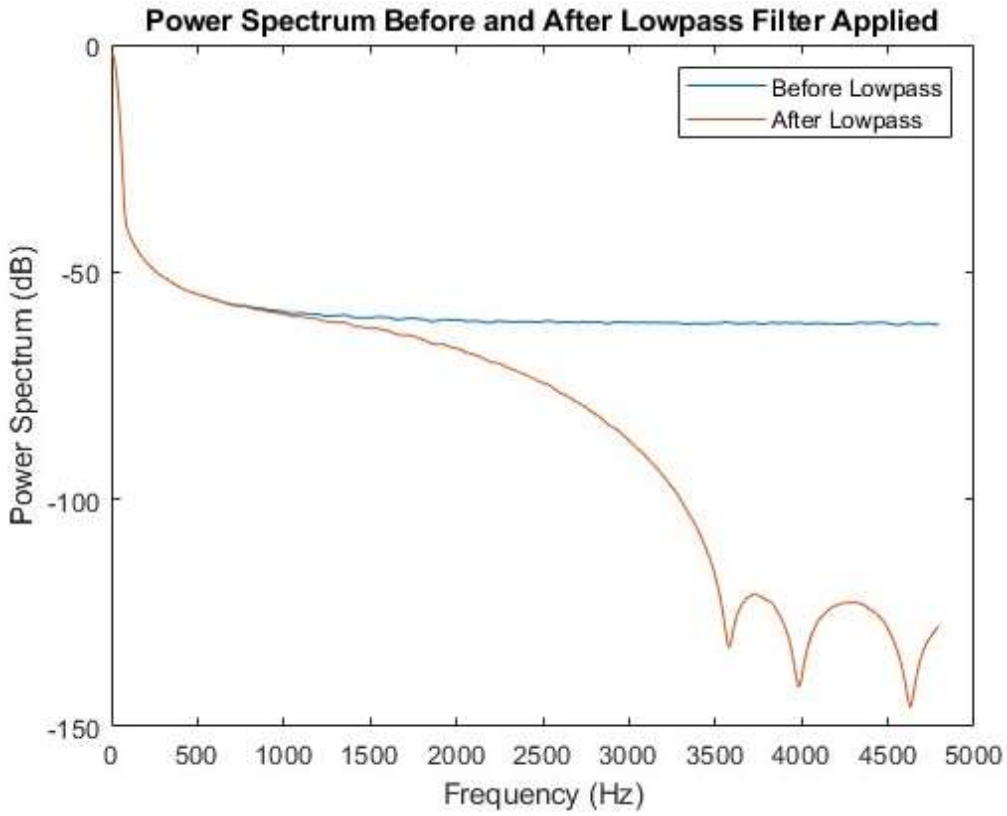


Figure 18: Plot of the power spectrum of a Microphone signal before and after it has been passed through a lowpass filter for the simulation.

Figure 18 further illustrates the reduction in high frequency noise by the lowpass filter with minimal distortion at low frequencies. The power spectrum for frequencies less than 1kHz is unaffected. The power at high frequencies greater than about 3.5kHz is significantly attenuated to below 100dB.

7.4.3 Synchronisation

The performance of the synchronisation subsystem simulation can be illustrated by taking plots at the different stages of its execution. In order to ease visibility, the default simulation parameters were altered. The maximum source and calibration frequencies were reduced to 1 Hz and 10 Hz respectively. Consequently, the cutoff frequency of the lowpass filter was reduced to 100 Hz. Lastly, the latency difference was increased to 1s to make the signal desynchronisation more visible. The input SNR was kept at its default 65 dB. The position of the source was at the centre of the grid (0.4, 0.25).

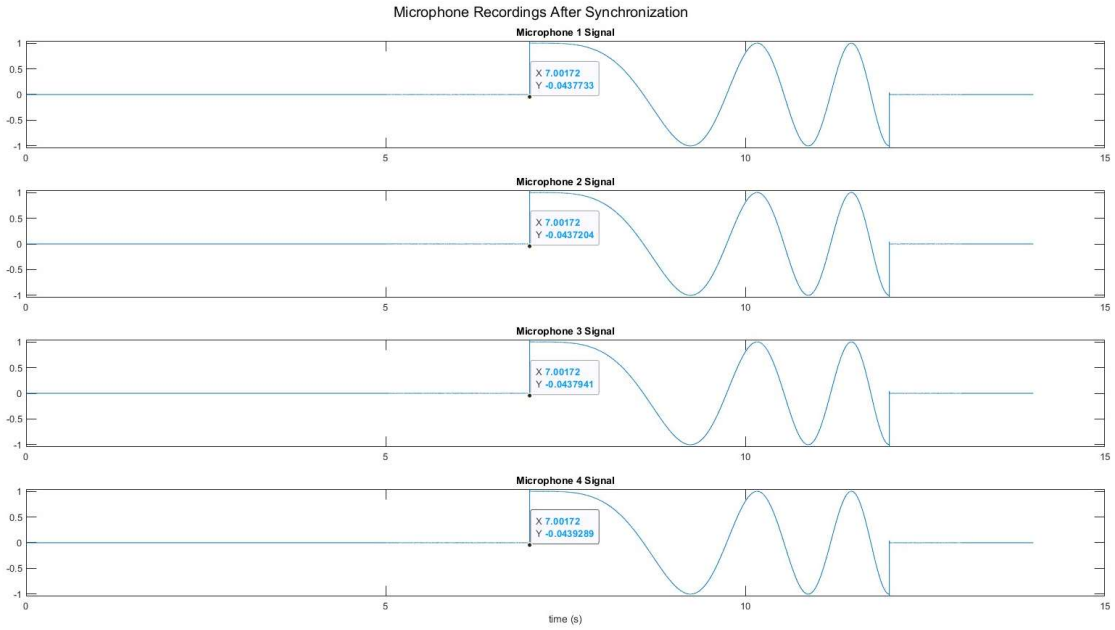


Figure 19: Plots of the Microphone recording signals after synchronisation for the simulation.

Since the source is located at the centre, there should be no difference in the ToA values of the recording signals at each Mic. Figure 19 shows that the correct synchronisation has been applied as the signals have the same ToA values. Moreover, the calibration signal has been removed as it is no longer required and may affect the time delay estimation.

7.4.3.1 Varying SNR

The simulated performance of the synchronisation subsystem was tested for varying SNR values from 40 to 80 dB. The results are shown in the figure below:

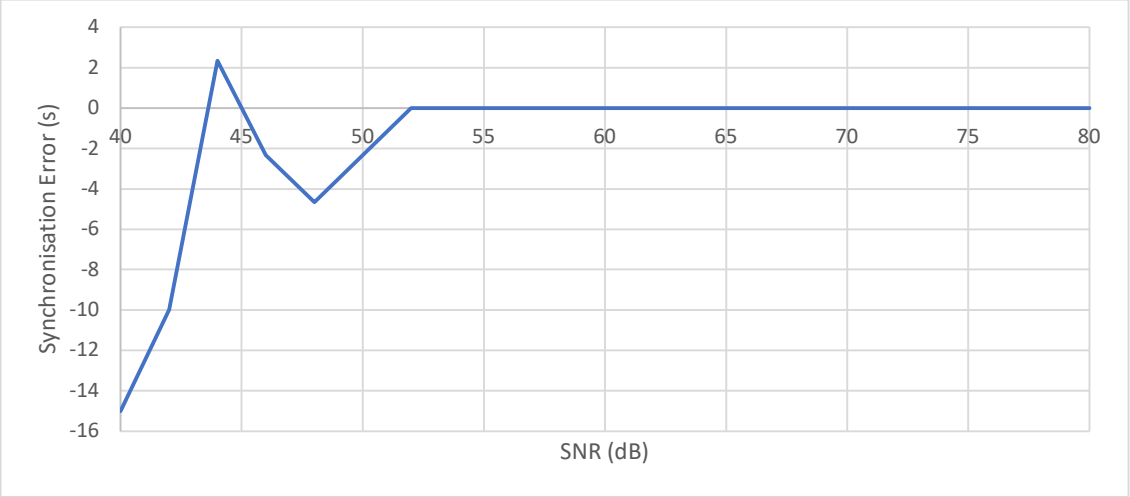


Figure 20: Graph of synchronisation error as a function of input SNR for the simulation.

Figure 20 shows that the simulated subsystem performs optimally for input SNR values greater than 52 dB. This matches the expected behaviour from the performance of the overall system for varying input SNR values. Moreover, the performance greatly deteriorates and becomes inconsistent for input SNR values less than 52 dB. Thus, for optimal performance environmental conditions should not decrease the input SNR below 52 dB. This provides insight into the limited application of the subsystem to different environments.

7.4.3.2 Varying Cutoff Frequency of Lowpass Filter

The cutoff frequency of the lowpass filter determines the attenuation of high frequency noise. A lower cutoff frequency has the potential for increased attenuation of high frequency noise. However, if the cutoff is placed to low then it may distort the calibration or source signal. The simulated performance of the synchronisation subsystem was tested for varying cutoff frequencies from 3 to 20 kHz. The results are shown in the figure below:

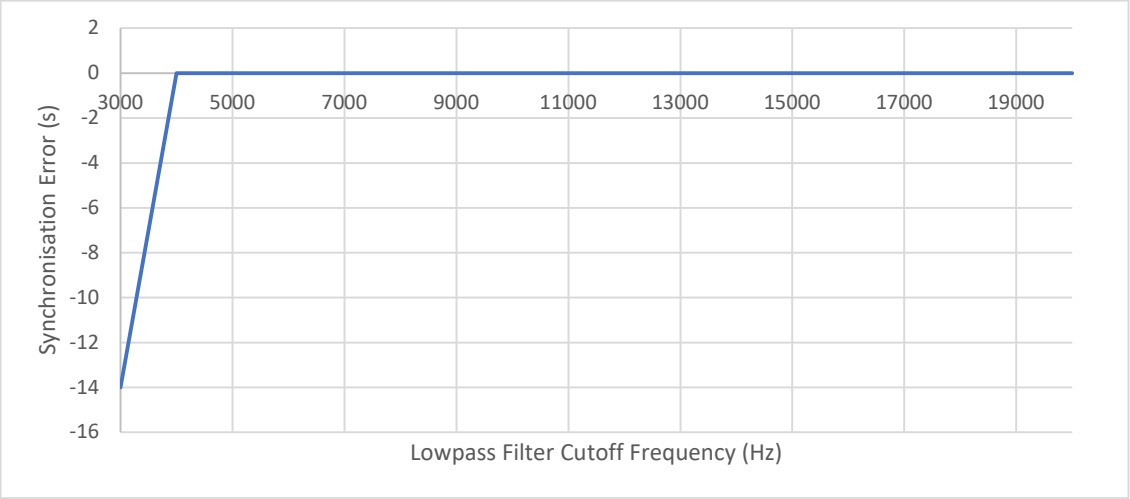


Figure 21: Graph of synchronisation error as a function of the lowpass filter cutoff frequency for the simulation.

It can be observed that Figure 21 illustrates that the synchronisation error greatly increases for cutoff frequencies below 5kHz. This can be explained by the fact that the highest signal frequency is 1kHz which is too close to frequencies below 5kHz. This matches the behaviour expected from the performance of the overall system for varying cutoff frequencies.

7.4.3.3 Varying Latency

The latency difference is a measure of the desynchronisation between the two RPi modules. The subsystem performance was tested for latency differences between the RPi modules from 0.001 to 1.024 seconds. The results are tabulated below:

Latency Difference (s)	Synchronisation Error (s)
0.001	0.000007
0.002	0
0.004	0
0.008	0
0.016	0
0.032	-0,000007
0.064	0
0.128	-0,000007
0.256	0
0.512	0
1.024	0

Table 4: Synchronisation error as a function of the latency between the RPi Microcontrollers for the simulation

Table 4 shows that the synchronisation error of the subsystem is in the order of microseconds. This is ideal as a time error of 1 microsecond corresponds to a distance error of 0.343 mm with the speed of sound at 343m/s. Thus, the effect of signal desynchronisation on the subsystem is minimal with the other parameters kept default.

7.4.3.4 Varying Calibration Signal Position Error

The performance of the synchronisation subsystem was tested for varying the calibration signal position error factor from 0.001 to 0.02 m was tested. The results of the tests are shown below:

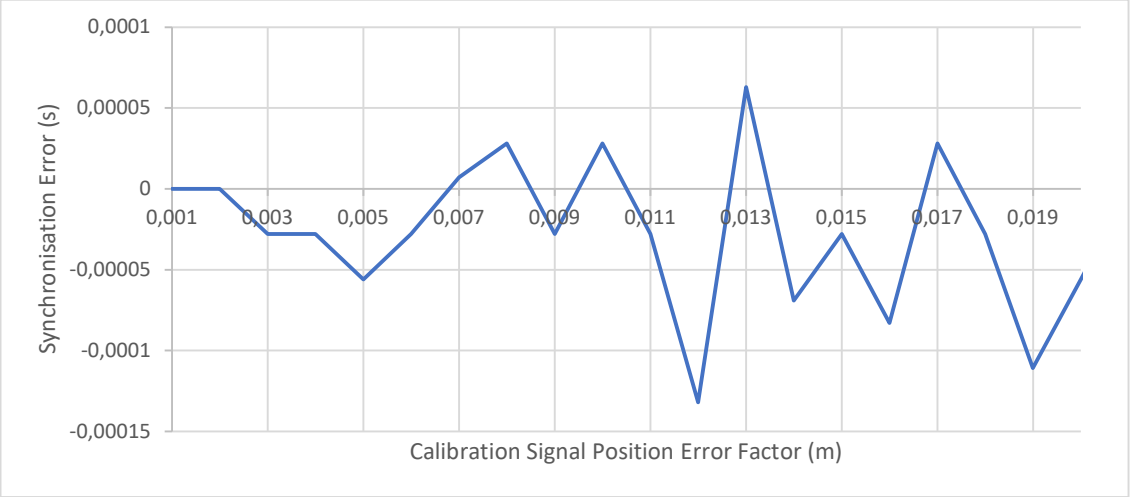


Figure 22: Graph of synchronisation error as a function of the calibration signal position error factor for the simulation.

Figure 22 shows a general trend of the magnitude of the mean synchronisation error increasing with an increased calibration position error. Furthermore, the standard deviation of the synchronisation error also appears to increase. This is expected as an error in the position of the calibration signal directly affects the signal synchronisation. This matches the expected behaviour from the performance of the overall system. Thus, the effect of the calibration signal position error on the synchronisation subsystem is significant.

7.4.3.5 Varying Microphone Position Error

The performance of the synchronisation subsystem was tested for varying the Mic position error factor from 0.001 to 0.02 m was tested. The results of the tests are shown below:

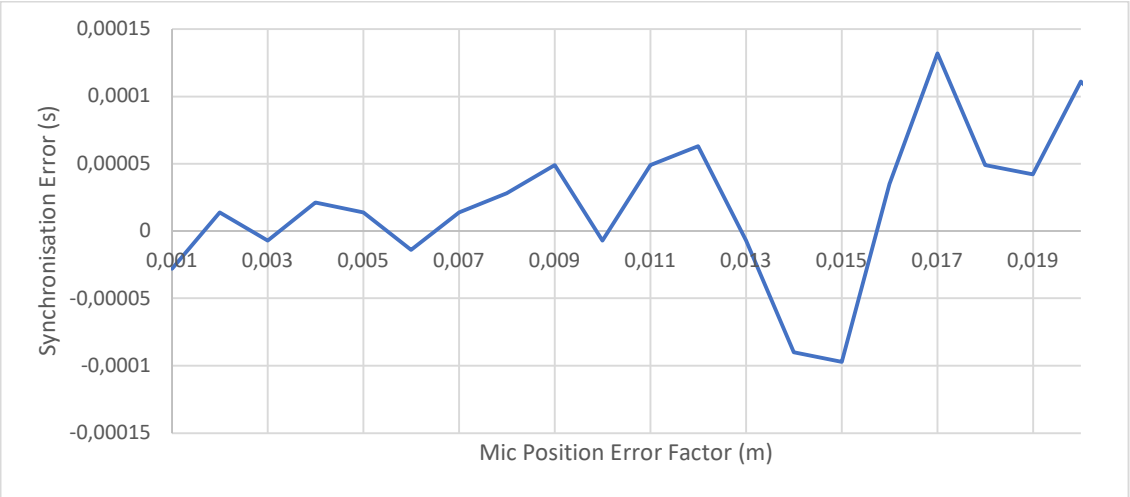


Figure 23: Graph of synchronisation error as a function of the Mic position error factor for the simulation.

From the graph above, it can be seen that varying the position error factor of the Mics, causes an increase in the standard deviation of the synchronisation error. This is different to the overall system where there was no discernible correlation between the Mic position error and the source position error over the range of values tested. This may be due to the fact that the synchronisation errors for

each Mic are independent. Thus, the net effect of the synchronisation errors on each Mic may partially cancel each other to produce a smaller effect on the position error. While the system is not as sensitive to the Mic position errors as the calibration position error, its effect should still be considered. Consequently, the Mic position error should be minimised for the final implementation.

7.4.3.6 Varying Source Signal Frequency

The frequency of the source signal was selected to go from 0 to 100 Hz. The synchronisation subsystem was tested in order to determine whether this selection was optimal and its effect on the synchronisation error. For this test, the maximum frequency of the source signal was varied from 10 to 10240 Hz.

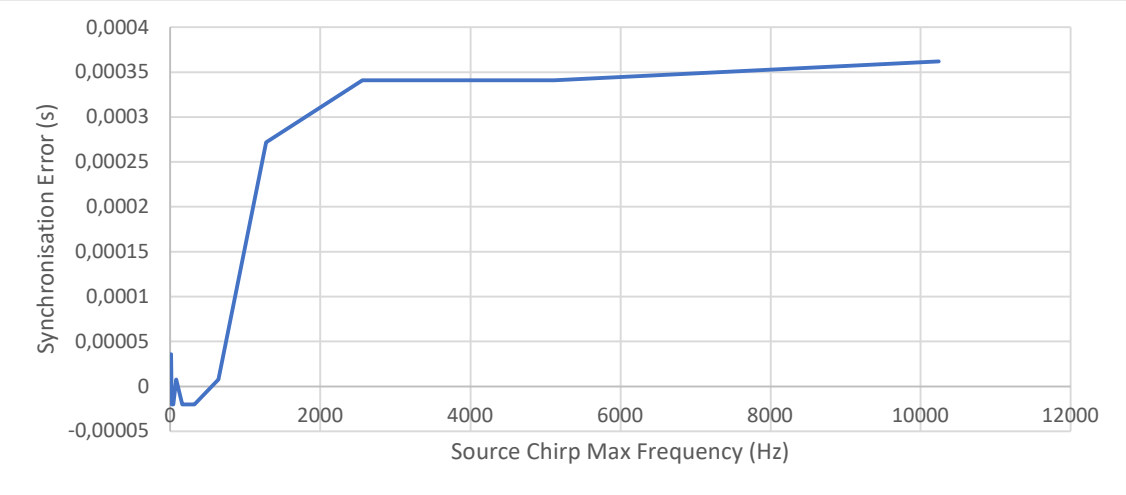


Figure 24: Graph of the synchronisation error as a function of the maximum frequency of the source chirp signal for the simulation.

Graph 24 shows that the optimal maximum source frequencies occur below approximately 1 kHz. This is while the cutoff frequency of the lowpass filter is 10 kHz and the maximum calibration frequency is 1kHz. Thus, these factors might be causing the deterioration of the performance at higher frequencies. This matches the expected behaviour of the overall system.

7.4.4 Time Delay Estimation

The performance of the time delay estimation subsystem simulation can be illustrated by taking plots at the different stages of its execution. The default simulation parameters were used. Additionally, the position of the source was at the centre of the grid (0.4, 0.25).

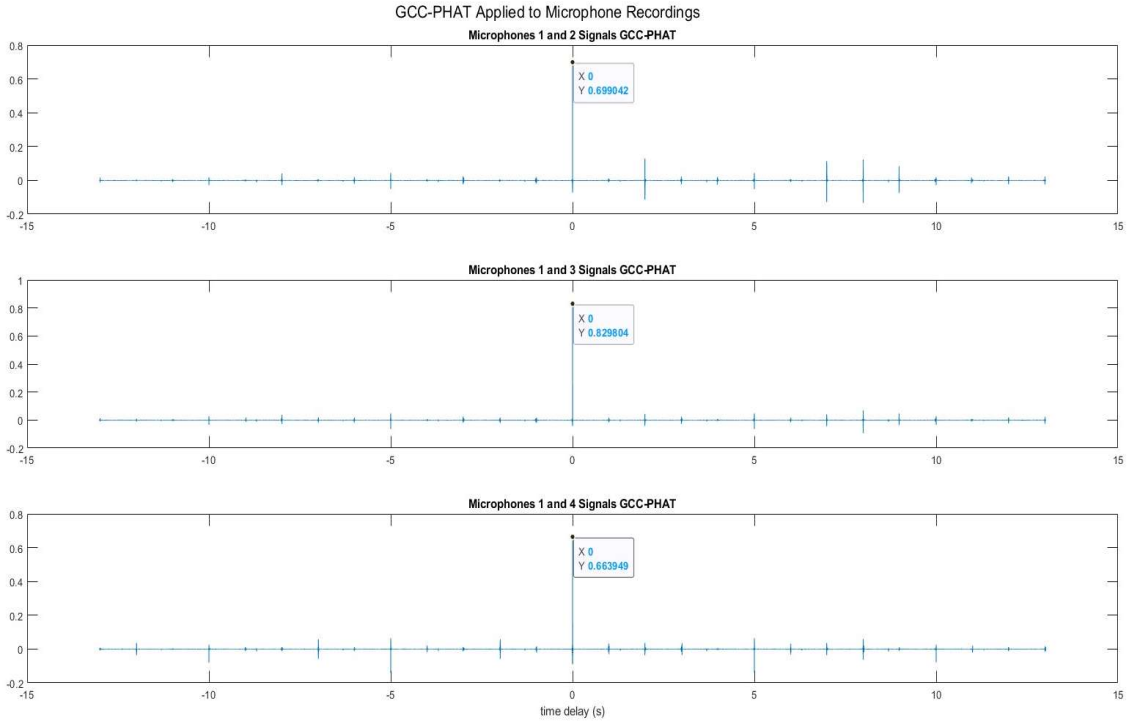


Figure 25: Plots of the GCC-PHAT cross correlation of the Microphone 1 signal with each of the signals from the other Microphones for the simulation.

Since the source is located at the centre of the grid, the TDoA values should all be zero. Figure 19 shows that the estimated TDoA values are zero as expected. Moreover, the peak at the correct time delay is dominant with no other significant peaks. This illustrates the optimal performance of the GCC-PHAT cross correlation for estimating the TDoA values.

7.4.4.1 Varying SNR

The simulated performance of the time delay estimation subsystem was tested for varying SNR values from 40 to 80 dB. The results are shown in the figure below:

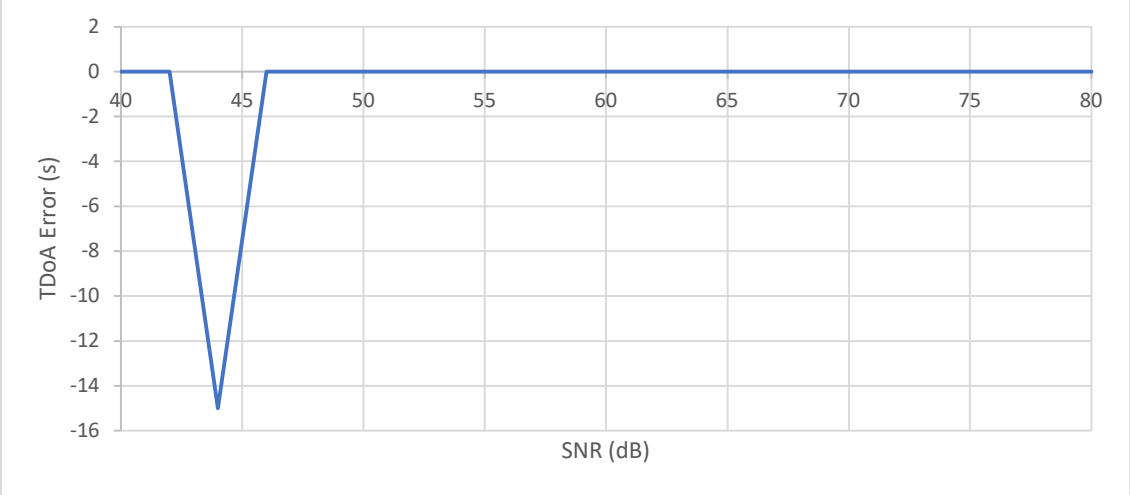


Figure 26: Graph of TDoA error as a function of input SNR for the simulation.

Figure 26 shows that the simulated subsystem performs poorly for input SNR values between 44dB and 46dB. This does not match the expected behaviour from the performance of the overall system

for and synchronisation subsystem for varying input SNR values. A possible explanation for this is that the synchronisation algorithm used can result in the calibration signal not being completely removed from the signals after synchronisation due to the synchronisation error. This would then bias the TDoA estimation towards the centre as this is the position of the calibration signal. Consequently, because the testing was done at the centre point the actual effect of the input SNR on the TDoA error and hence position error may be reduced compared to an arbitrary point. This shows that an alternative method of separating the calibration and source signals is needed to ensure that the estimated synchronisation and TDoA values are consistent. Additionally, future testing should be done at a point other than the centre of the grid.

7.4.4.2 Varying Cutoff Frequency of Lowpass Filter

The simulated performance of the time delay estimation subsystem was tested for varying lowpass filter cutoff frequencies from 3 to 20 kHz. The results are shown in the figure below:

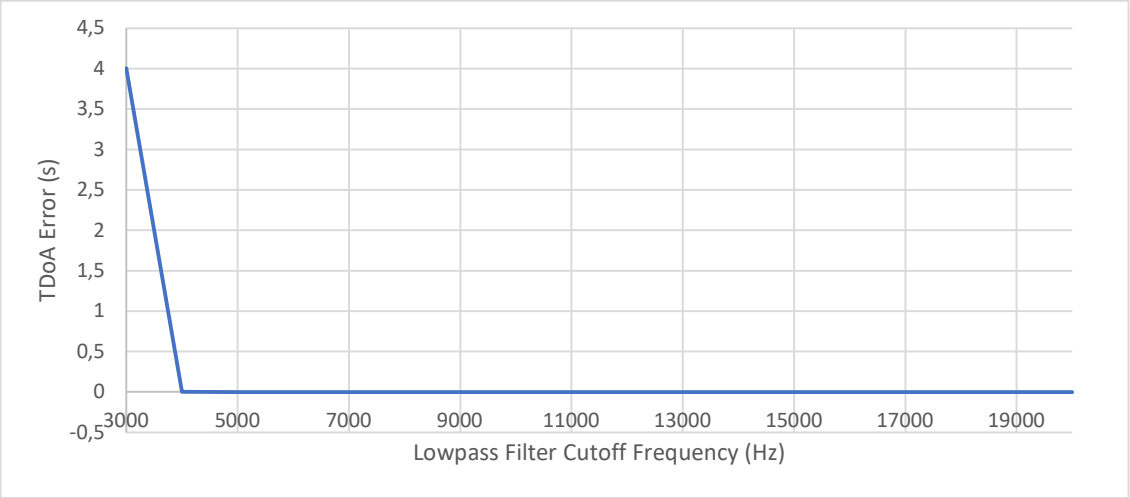


Figure 27: Graph of TDoA error as a function of the lowpass filter cutoff frequency for the simulation.

It can be observed that Figure 27 illustrates that the TDoA error greatly increases for cutoff frequencies below 5kHz. This can be explained by the fact that the highest signal frequency is 1kHz which is too close to frequencies below 5kHz. This matches the behaviour expected from the performance of the overall system for varying cutoff frequencies.

7.4.4.3 Varying Latency

The latency difference is a measure of the desynchronisation between the two RPi modules. The subsystem performance was tested for latency differences between the RPi modules from 0.001 to 1.024 seconds. The results are tabulated below:

Latency Difference (s)	TDoA Error (s)
0,001	0,000008
0,002	-0,000027
0,004	0,000001
0,008	0,000008
0,016	0,000008
0,032	0,000008
0,064	0,000008
0,128	0,000008
0,256	0,000008

0,512	0,000008
1,024	0,000008

Table 5: TDoA error as a function of the latency between the RPi Microcontrollers for the simulation

Table 5 shows that the TDoA error of the subsystem is in the order of microseconds. This is ideal as a TDoA error of 1 microsecond corresponds to a distance error of 0.343 mm with the speed of sound at 343m/s. Thus, the effect of signal desynchronisation on the subsystem is minimal with the other parameters kept default.

7.4.4.4 Varying Calibration Signal Position Error

The performance of the synchronisation subsystem was tested for varying the calibration signal position error factor from 0.001 to 0.02 m was tested. The results of the tests are shown below:

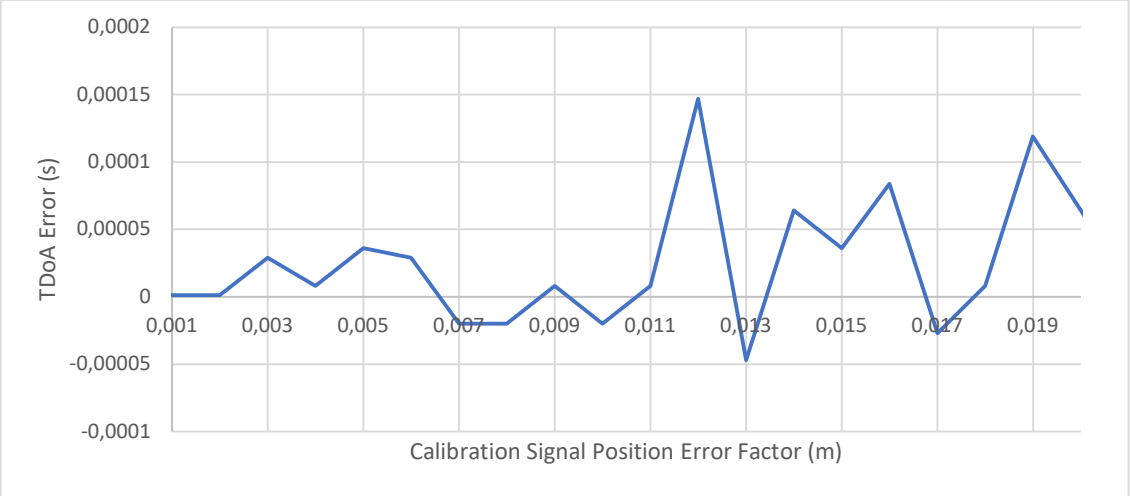


Figure 28: Graph of TDoA error as a function of the calibration signal position error factor for the simulation.

Figure 28 shows a general trend of the magnitude of the mean TDoA error increasing with an increased calibration position error. Furthermore, the standard deviation of the TDoA error also appears to increase. This matches the expected behaviour from the performance of the overall system and synchronisation subsystem. Thus, the effect of the calibration signal position error on the time delay subsystem is also significant.

7.4.4.5 Varying Microphone Position Error

The performance of the synchronisation subsystem was tested for varying the Mic position error factor from 0.001 to 0.02 m was tested. The results of the tests are shown below:

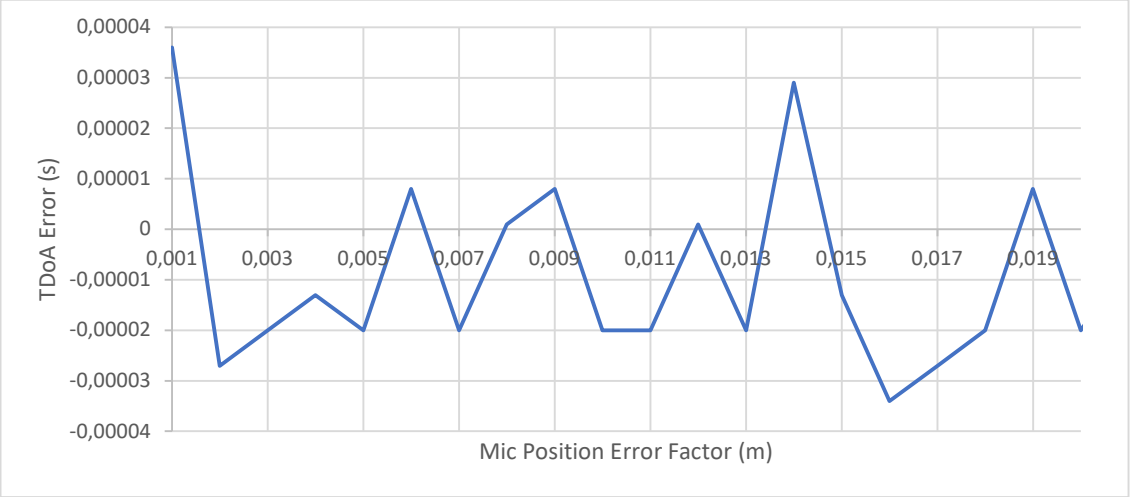


Figure 29: Graph of TDoA error as a function of the Mic position error factor for the simulation.

From the graph above, it can be seen that varying the position error factor of the Mics, causes erratic and unpredictable TDoA errors. There is no discernible correlation between the Mic position error and the source position error over the range of values tested. This matches the behaviour of the overall system. Furthermore, it supports the idea that the Mic position errors in the actual system should be minimised as far as possible.

7.4.4.6 Varying Source Signal Frequency

The frequency of the source signal was selected to go from 0 to 100 Hz. The time delay estimation subsystem was tested in order to determine whether this selection was optimal and its effect on the TDoA error. For this test, the maximum frequency of the source signal was varied from 10 to 10240 Hz.

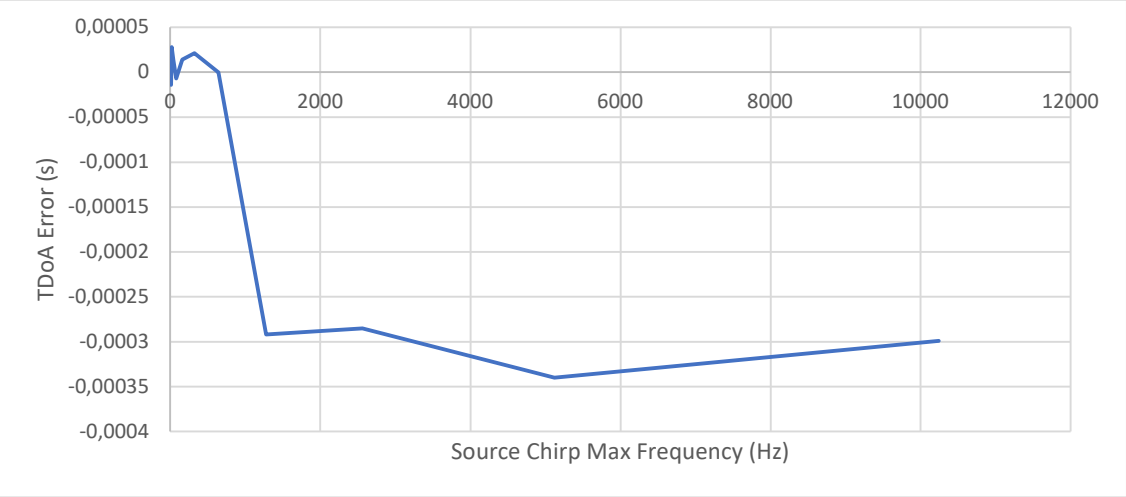


Figure 30: Graph of the TDoA error as a function of the maximum frequency of the source chirp signal for the simulation.

Graph 30 shows that the optimal maximum source frequencies occur below approximately 1 kHz. This is while the cutoff frequency of the lowpass filter is 10 kHz and the maximum calibration frequency is 1kHz. Thus, these factors might be causing the deterioration of the performance at higher frequencies. This matches the expected behaviour of the overall system.

7.4.5 Triangulation

The primary indicator of the performance of the triangulation subsystem is the position error of the estimated positions produced. However, these results were also used to measure the performance of the overall system and thus will not be repeated. Thus, the analysis to the overall system also applies to the triangulation subsystem.

The performance of the triangulation subsystem simulation can be illustrated by displaying the iteration of the nonlinear least squares estimation algorithm as it locates the estimated position of the source. The default simulation parameters were used with the source located at (0.8, 0.5). Since the initial estimated position of the source is at the centre of the grid, the corners will be furthest away from the initial estimate.

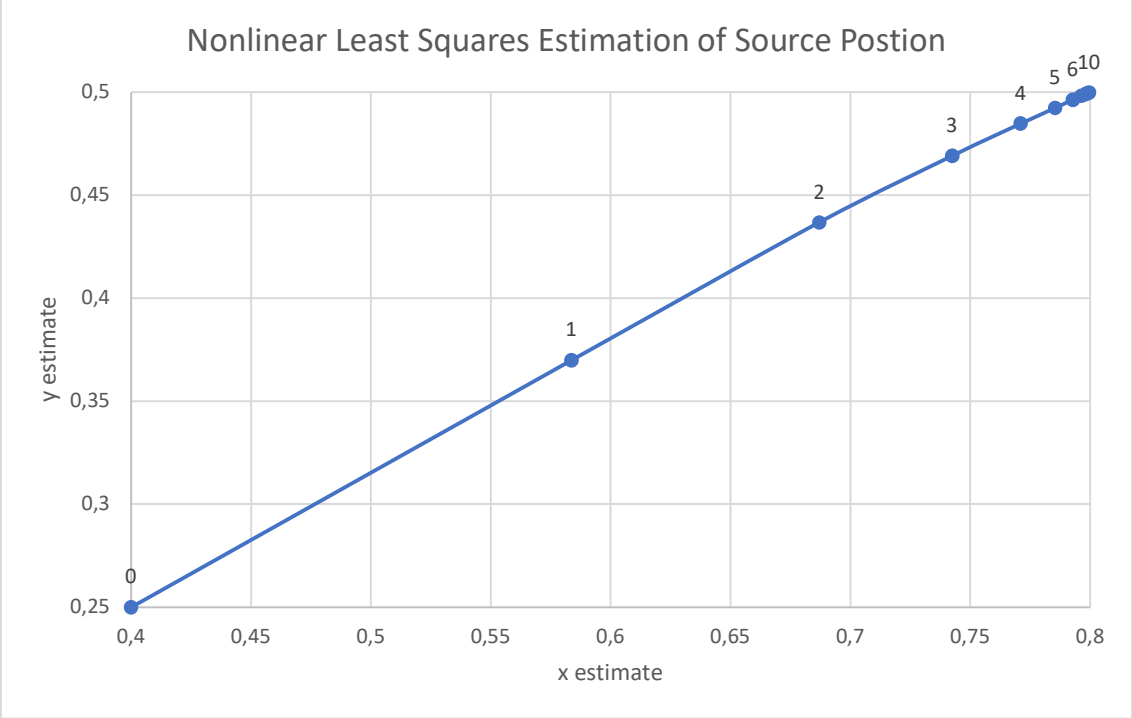


Figure 31: Graph of the coordinates estimated by the nonlinear least squares estimation algorithm at each iteration for the simulation.

Iteration	x estimate	y estimate
0	0.4	0.25
1	0.5836	0.3698
2	0.687	0.4367
3	0.7424	0.469
4	0.7709	0.4846
5	0.7854	0.4923
6	0.7927	0.4962
7	0.7963	0.4981
8	0.7981	0.499
9	0.799	0.4995
10	0.7995	0.4997

Table 6: Table of the coordinates estimated by the nonlinear least squares estimation algorithm at each iteration for the simulation.

Figure 31 and Table 6 illustrate the steps taken by the triangulation algorithm to estimate the coordinates of the source. The algorithm takes multiple iteration to arrive at its final estimation. The size of steps taken reduces the closer it gets to the final estimation. This significantly slows down the algorithm. This could be optimized by using a linear least squares estimation to find the initial point rather than using the centre of the grid. This can be applied to the final implementation to reduce the computational intensity of the triangulation subsystem.

8 VALIDATION USING FINAL IMPLEMENTATION

8.1 MOTIVATION

The final implementation of the system was done to provide a hardware-based validation of the proposed system. This was done to account for the real-world conditions that were not accounted for in the simulation-based validation. Additionally, the insights gained from the simulation-based validation could be used to improve the final implementation. Thus, the hardware-based validation assesses the performance of the final design achieved. Moreover, it provides a more accurate validation of the implementation. Therefore, the hardware-based implementation was needed to ultimately conclude the performance achieved by the final implementation.

8.2 PROCEDURE LIST

The simulation-based validation was done according to the following steps:

1. The final subsystems were developed according to their requirements, specifications and the insights gained from the simulation-based validation.
2. The final subsystems were integrated into the final overall system simulation according to its requirements, specifications and the insights gained from the simulation-based validation.
3. The experiments used to test the performance of the final system and subsystems were defined.
4. Data was collected from the experiments performed on the final system and subsystems.
5. The results of the tests performed were analysed and discussed.

8.3 PROCEDURE DESCRIPTION

8.3.1 System Design and Implementation

8.3.1.1 Overall System Implementation

Hardware:

The overall system implementation consists of the following hardware components:

- 2 Raspberry Pi Zero W modules
- 4 Adafruit I2S MEMS Microphone breakout boards
- An A1 (0.8x0.5 m) paper grid
- A laptop running the Microsoft Windows OS (Host Device)
- 2 Mobile Devices
- 2 Bluetooth Speakers
- 3 Micro-B USB cables
- A USB Hub
- 2 Breadboards
- Jumper cables
- Electrical Tape

The 2 RPi modules were each connected to a pair of the Adafruit I2S MEMS Microphone breakout boards via jumper cables connected through the 2 breadboards. The connections between the RPi modules and the Mic breakout boards utilized the I2S interface. The Mics were each placed on a separate corner of the A1 paper grid. The coordinates of Mics 1 to 4 were (0, 0), (0, 0.5), (0.8, 0.5) and (0.8, 0) respectively. Electrical tape was used to secure the Mics and jumper cables in place. The RPi modules were connected to the host device wirelessly via a local network. Additionally, the RPi modules were connected to a USB hub via Micro-B USB cables. The USB hub was connected to the host device via a Micro-B USB cable. The USB hub was used to power both RPi modules from the host device. Lastly, the 2 Bluetooth speakers were used to produce the required audio signals on the grid. They were each wirelessly connected to a mobile device via Bluetooth. The mobile devices were used to select the audio signals.

With regards to challenges, one of the RPi modules provided were faulty and had to be replaced, which pushed the project behind schedule for a short period of time. Once replaced, the microphones showed signs of not recording, even though both RPi modules were configured correctly. This was suspected to be lack of secure wiring and connections. This was addressed by soldering each of the microphones onto Veroboard as well as soldering Female Header Sockets for easy and secure connections.

Software:

The overall system consists of the 6 Python programs given below.

- **gui.py**: a graphical interface for the user to interact with the system.
- **signal_acquisition.py**: extracts and processes the audio signals from the recording files.
- **gcc_phat.py**: a library implementation of the GCC-PHAT algorithm.
- **time_delay_estimation.py**: estimates the time delays between a set of signals and a reference signal.
- **triangulation.py**: triangulates the 2D position of a signal based on the receiver positions and TDoA values.

Additionally, the system makes use of the following Python libraries:

- NumPy
- SciPy
- Matplotlib
- PySimpleGUI
- subprocess

Lastly, the system also makes use of the following collection of bash scripts:

- main.sh
- pi_one.sh
- pi_two.sh
- raspberrypi.sh

With regards to challenges, it was initially difficult to synchronise the start of the recordings on both microphones. To address this, thorough research on parallel scripts was conducted. This provided a good basis for understanding how scripts run and also how to execute them in the background. Further information on the execution of these scripts is discussed in the next section.

8.3.1.2 Signal Acquisition

The Signal Acquisition subsystem consists of two stages. The first stage is the capturing of the raw audio signals recorded from each microphone. The second stage is the preprocessing of the raw audio signals into processed audio signals that can be used by the other subsystems.

Signal Capture:

The Signal Capture stage was implemented using a local network connecting the host device with the Raspberry Pi Microcontrollers. A collection of bash scripts was created to initiate the recording of the audio signals and the transmission of these from the Raspberry Pi Microcontrollers to the host device. The bash scripts would transmit a record instruction to both of the Raspberry Pi Microcontrollers via a Secure Shell (SSH) connection. Once the recordings were completed, they were retrieved via the Secure Copy Protocol (SCP). Initially, a parallel-SSH was to be used to start the recording on both RPi Microcontrollers simultaneously. It was found, however, that parallel-ssh was not compatible with the task and, therefore, an '&' character was used between the 2 Raspberry Pi scripts. The syntax and logic of these bash scripts will be discussed further in the Data Collection and Analysis section.

Signal Preprocessing:

The microphones and algorithms used were also found to be incredibly sensitive to noise. What was not accounted for in the previous milestone, was the presence of a large amount of low frequency noise within the system. As such, a bandpass filter was implemented in place of a lowpass filter. The processing of each signal consists of the following stages:

1. The raw audio signal is split in two halves.
2. The first half is designated as the calibration signal.
3. The second half is designated as the source signal.
4. A Hann window is applied to each half.
5. Each half is filtered by a Butterworth bandpass filter with its corresponding frequency band.
6. Each half is normalised to have a maximum magnitude of 1.

With regards to challenges, it was found that the frequency bands of the source and calibration signals chosen as well as the order of the bandpass filter severely impacted the results obtained, as it could either let too much noise pass through, or it could distort the signal, leading to inaccurate results. This was addressed by testing the performance of the algorithms these quantities were varied, to determine their optimal values. This is discussed further in the Results section.

8.3.1.3 Synchronisation

The Synchronisation subsystem was implemented using a calibration signal as previously stated. Furthermore, the Network Time Protocol (NTP) was also implemented to determine if it would sufficiently synchronise the recordings. The frequency range of the calibration signal was moved to 6-10 kHz to accommodate the change in the source signal frequency to 1-5kHz. A dead band of 1kHz was kept between the signals to reduce the chance of frequency aliasing. A challenge with synchronisation was separating the calibration and source signals from the recordings. Thus, to solve the issue, time-slicing was implemented. The 16 second recording was divided into two halves. The first half was allocated to the calibration signal and the second half was allocated to the source signal. The synchronisation delays are estimated using the GCC-PHAT algorithm. The calibration signal acquired from Mic 1 was used as the reference signal for the cross correlation with the calibration signals from each Mic. This includes the correlation of the calibration signal of Mic 1 with itself to ensure that a delay of zero is obtained. The synchronisation delays calculated were then used to compensate for the desynchronisation between the signals from different Mics.

8.3.1.4 Time Delay Estimation

The Time Delay Estimation Subsystem was implemented using the GCC-PHAT algorithm. The time delay estimation was done using GCC-PHAT cross correlation of the source signal from Mic 1 with the source signals from each of the other Mics, including Mic 1. Mic 1 was included to detect any obvious errors in the time delay estimation. This process produces the estimated ToA delays of the audio signals at each Mic with respect to Mic 1. However, these ToA delays include the synchronisation delays, and this must be removed to produce accurate TDoA values. The TDoA values are calculated according to the following equation:

$$TDoA_{1i} = (ToA_1 - ToA_i) - (Sync\ Delay_1 - Sync\ Delay_i)$$

This equation is used to produce synchronised TDoA values for Mics 2, 3 and 4 with respect to Mic 1. The only significant change made to the physical implementation compared to the simulated implementation is that the synchronisation delays were compensated for by calculation rather than shifting the signals as this proved to be much simpler.

With regards to challenges, if the synchronisation delays were extremely inaccurate, then the TDoAs will also be extremely inaccurate. Proper synchronisation between the audio signals was, therefore, ensured to improve the accuracy of the TDoA.

8.3.1.5 Triangulation

Triangulation is done using a non-linear least squares estimation. The nonlinear LSE algorithm used is the trust-region-reflective algorithm that was used in the simulation implementation. This is implemented using the Python SciPy library curve_fit function which is equivalent to the MATLAB lsqcurvefit function used in the simulation implementation. As such, no real challenges emerged when implementing this subsystem. A function was defined that takes in the coordinates of the sound source and the positions of the Mics and outputs the differences in the distances from Mic 1 to the source and the other Mics to the source. The curve_fit function then estimates the source position using the estimated TDoA values multiplied by the speed of sound as the distances.

Furthermore, this is done by iterating an initial position until if matches the expected distance values. The significant change from the simulation implementation was that a linear LSE was added to produce the initial position from the TDoA values and Mic positions. This is done instead of using the centre of the grid (0.4, 0.25) as the initial point as was done in the simulation interpretation. Additionally, a lower and upper bound are placed on the solution. The lower bound and upper bound coordinates were set to 0.001m greater than the grid size, so that estimates outside the grid could be detected and correctly rejected.

8.3.1.6 Graphical User Interface

Since the code was no longer in MATLAB but now in Python, a Python GUI had to be implemented to work with the other programs such as the triangulation, time delay estimation and signal acquisition programs. The "PySimpleGUI" library was used to create the GUI. Though previously the "Tkinter" library was considered initially, "PySimpleGUI" was found to be much easier to implement and contains much fewer lines of code for lower latency. The GUI consists of six buttons, a grid and 5 lines of text, consisting of the estimated synchronisation delays, estimated TDoA delays and other estimated values. The initial startup GUI setup can be seen below:

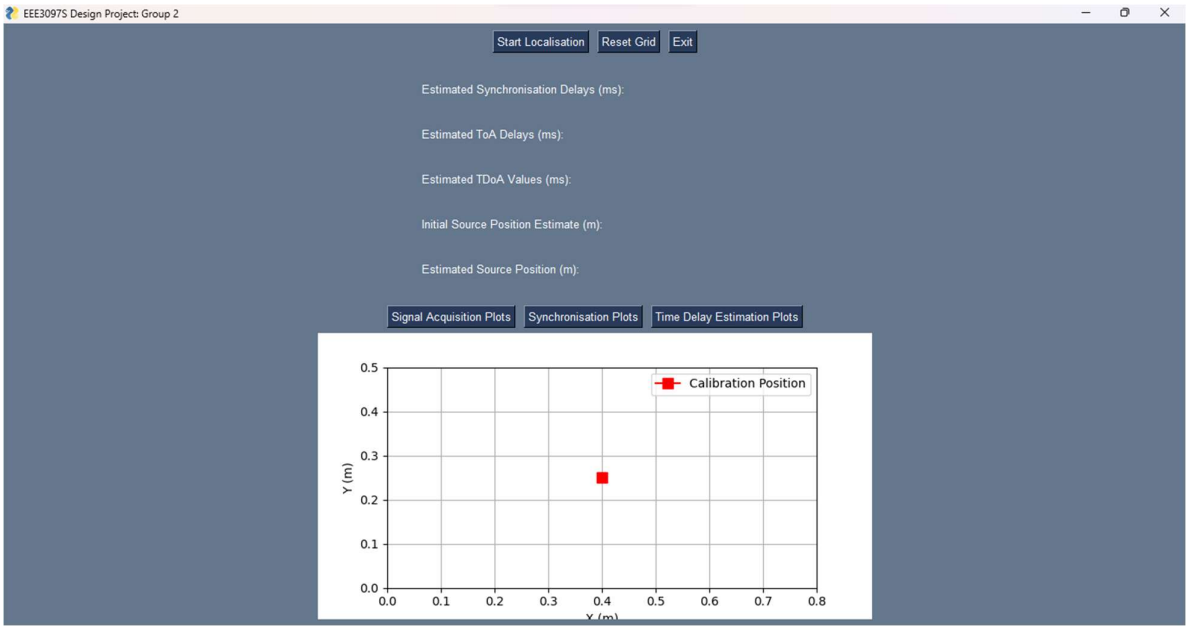


Figure 32: Project GUI

The "Start Localisation" button calls an SSH command to tell the Raspberry RPis to start recording the sound. Thereafter, the data obtained from the RPis is analysed by the respective TDoA and triangulation programs and displayed on the GUI. The estimated coordinates are also plotted on the grid. The screenshot below shows the displayed information for the approximate coordinates (0.55, 0.45):

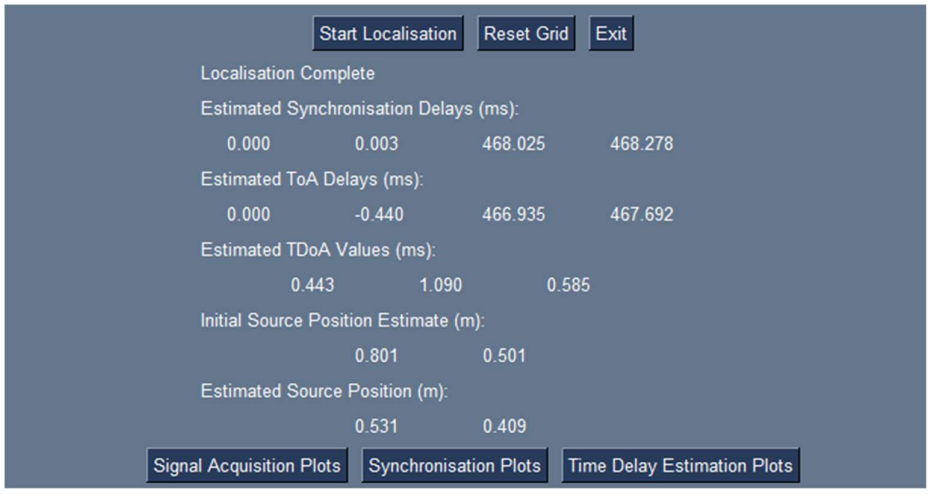


Figure 33: GUI TODA calculation outputs

The image below shows the grid, which contains the source signal positioned at (0.55, 0.45):

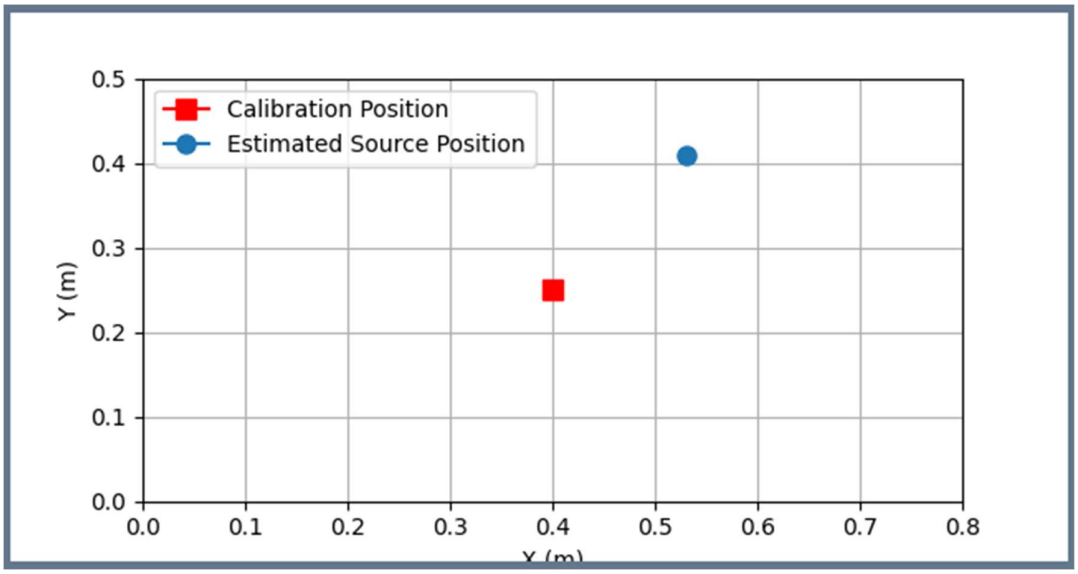


Figure 34: Displaying the source position with an input source position of (0.55,0.45)

The GUI also has buttons to display the signal acquisition plots, the synchronisation plots and the time delay estimation plots. For the example above the image below shows the signal acquisition plots:

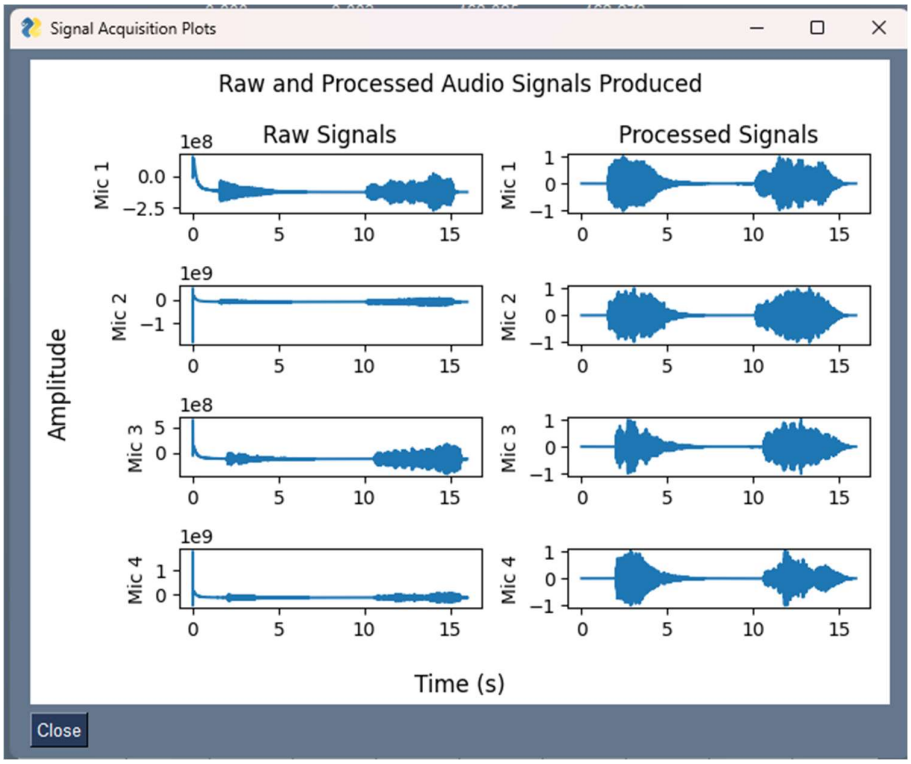


Figure 35: Signal Acquisition plots associated with the input source signal at position (0.55,0.45)

The image below shows the synchronisation plots for the same example:

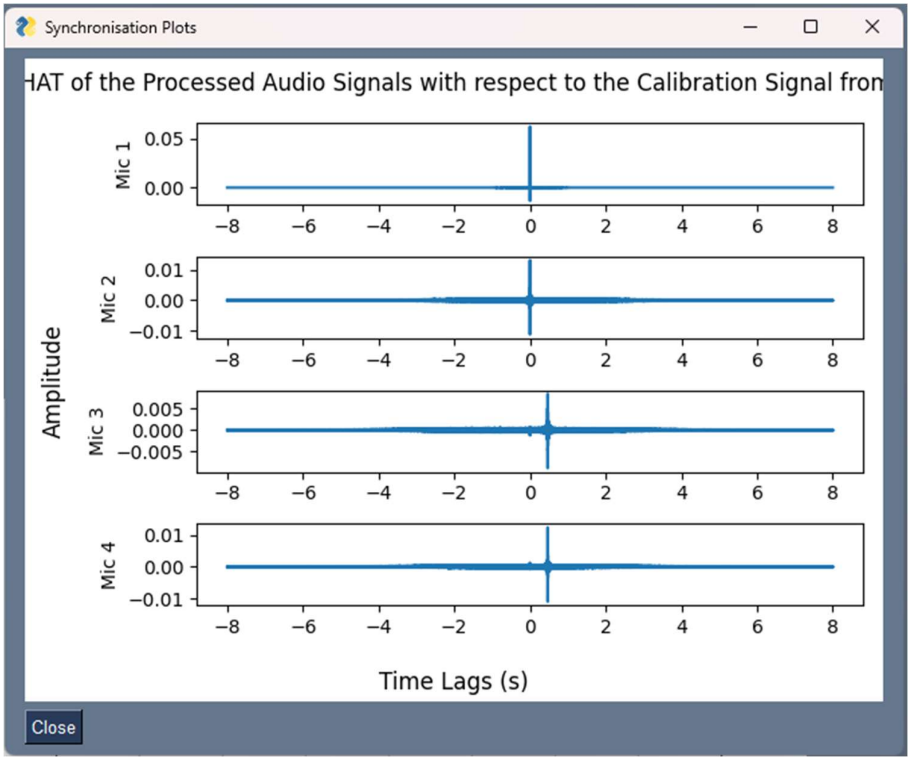


Figure 36: Synchronisation plots for an input source signal at position (0.55,0.45)

Finally, the image below shows the time delay estimation plots:

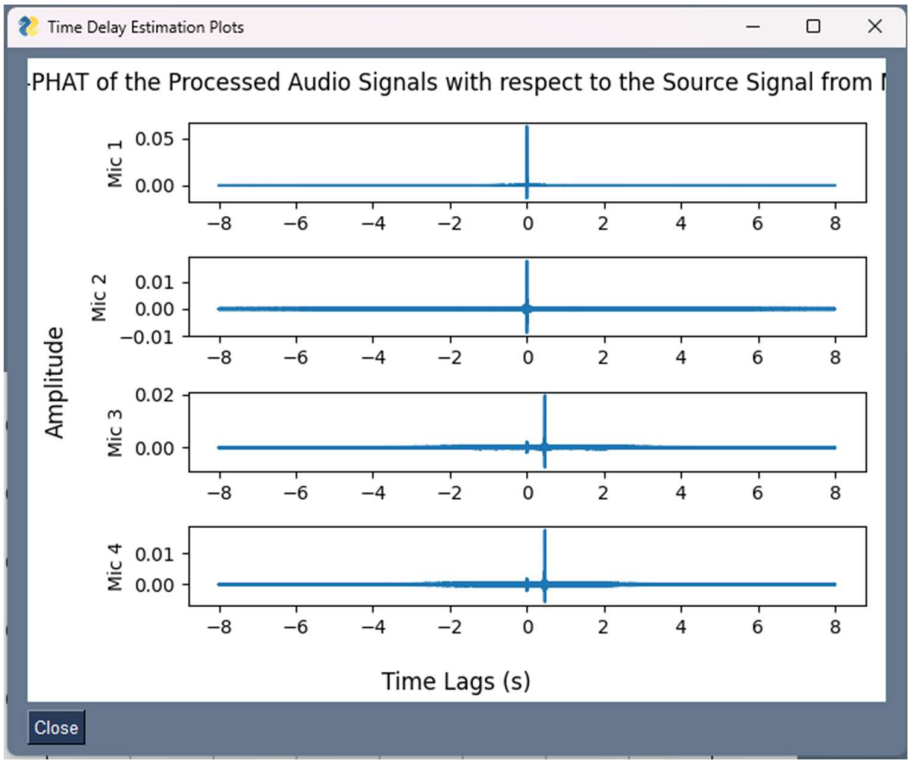


Figure 37: Time Delay Estimation plots for an input source signal at position (0.55,0.45)

The GUI contains a ‘’Reset” button, which clears the grid, allowing for the location of multiple different locations without having to restart the program.

There were multiple challenges faced in this section, with the most noteworthy one being the difficulty in positioning of GUI elements as compared to the MATLAB App Designer. Since the app is entirely coded, to position an item to one’s preference would require coordinate inputs for each element. Furthermore, the app experienced cross-compatibility issues, where the app would work very well on a certain laptop but not as well on another. There were also challenges faced during the creation of the ‘Reset’ button, as it would sometimes resize the graph or delete the gridlines.

8.3.2 Experimental Setup

8.3.2.1 Overall System

The following apparatus was used for this physical implementation:

- 1 x A1 grid
- 2 x Raspberry Pi Zero W modules
- 2 x Red Mushroom Bluetooth Speakers – Pictured in Figure
- Electrical Tape
- 4 x the Adafruit I2S MEMS Microphone breakout boards
- 3 x Micro-USBs
- 1 x mini-HDMI adapter
- 2 x Breadboards
- Jumper cables
- Veroboard
- Female Headers
- 1 x USB Hub
- 2 x IPads
- 1 x Mobile phone
- 1 x Asus Vivobook Laptop (Local Host)

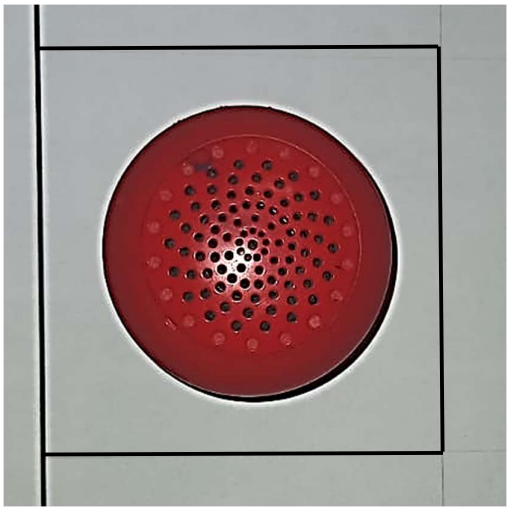


Figure 38: Image of the Mushroom Bluetooth speaker with a grid block as a reference.

In Figure 7, the size of the Bluetooth Speaker can be seen, with reference to a grid block. This provides insight to range of positions, which could be outputted by the triangulation algorithm.

The microphones were soldered onto Veroboard, along with Female Headers, in order to ensure a reliable connection.

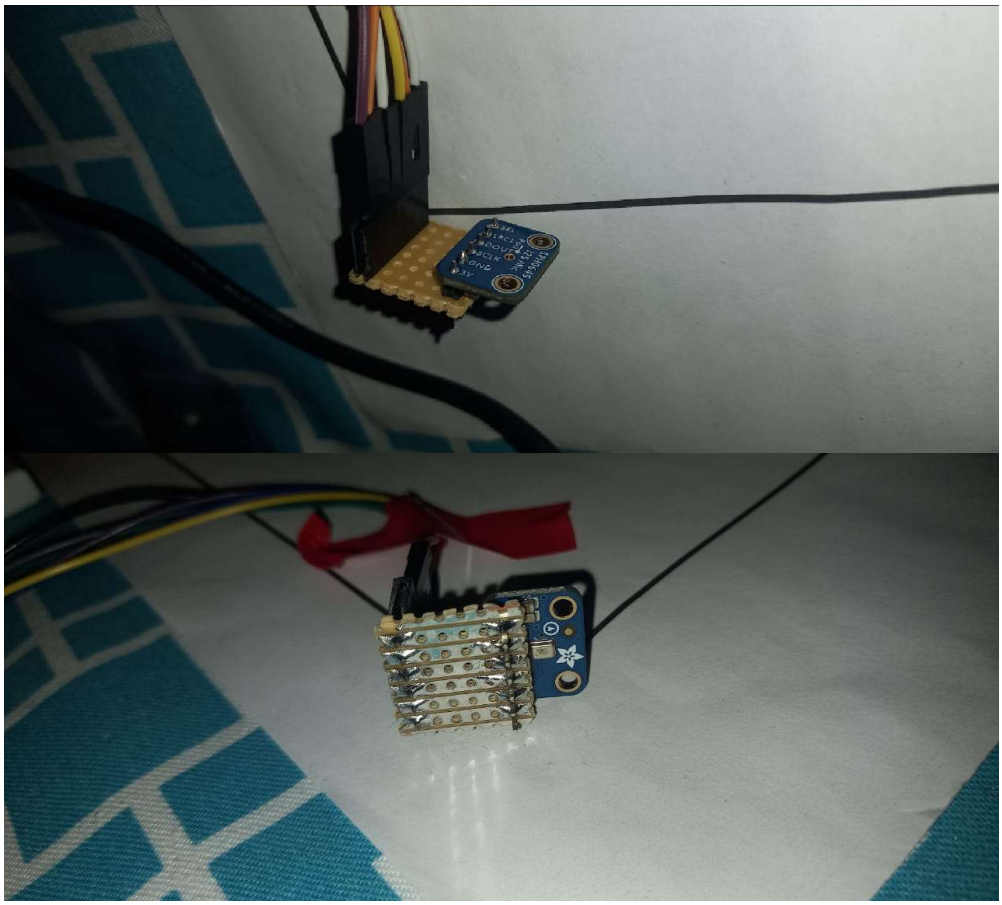


Figure 39: Connections associated the I2S Mics.

2 Microphones were connected to 1 Pi each, using a breadboard. Jumper cables were used to ensure these connections.

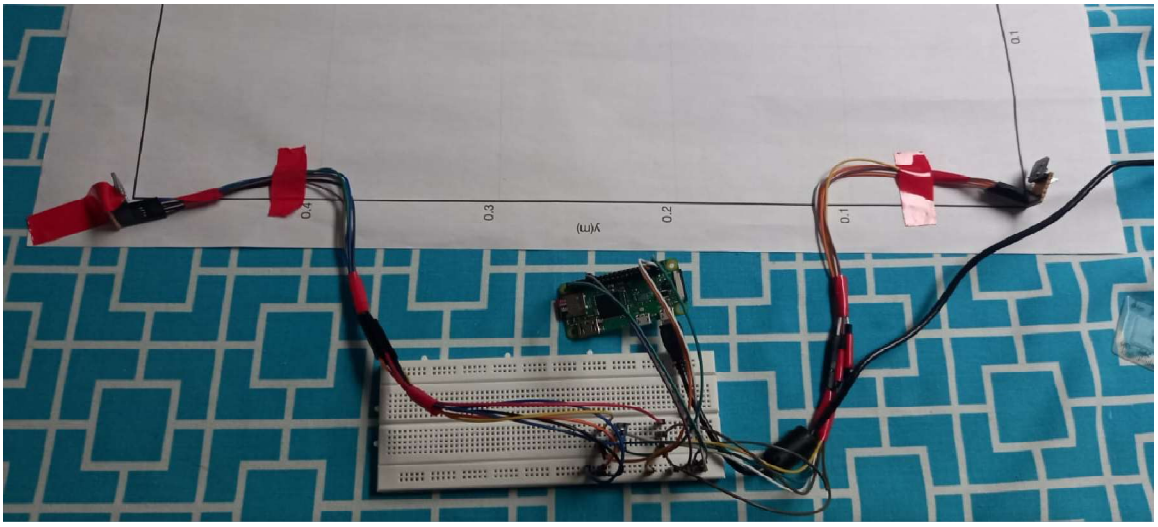


Figure 40: Connections between the RPI modules and the I2S Mics.

The microphones were placed at each corner and made to point towards the centre of the grid. They were held steady using some electrical tape. Please note that Figure 9 is for demonstration purposes only, as the microphone on the right-hand side is not exactly correctly orientated.

The complete experimental setup is shown below:

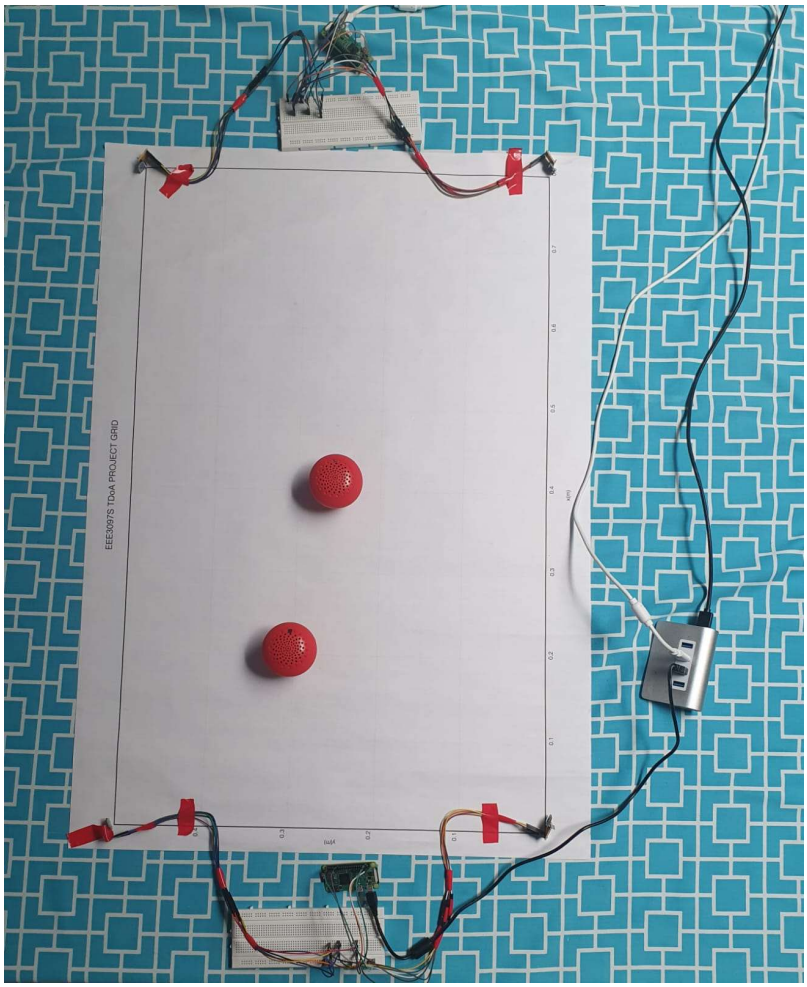


Figure 41: Overall System Experimental Setup.

The Bluetooth Speaker in the centre is responsible for producing the calibration signal, while the other Bluetooth speaker is responsible for producing the source signal.

8.3.2.2 Signal Acquisition

Bluetooth Speakers were used instead of mobile devices because it was found that the mobile devices' speakers were not sufficient to produce a loud enough sound required to achieve the level accuracy that was needed. The Bluetooth speakers could also handle low frequencies, which could not be heard through the mobile devices' speakers.

The signal acquisition subsystem was tested in two steps. Firstly, the signals recorded from each Mic were obtained. Secondly, the received signals were passed through a bandpass filter to reduce the high and low frequency noise in the signals.

The signal recordings comprised of a source signal and calibration signal. These are both 5 second chirp signals. Chirp signals were selected as they are optimal for computing time delays using cross correlation as mentioned in [1], although the algorithm will also work for regular sinusoidal signals.

As the Mics are connected in pairs, a latency difference is evident between the signals from Mics 1 and 2 and the signals from Mics 3 and 4.

The high pass filter was implemented using the Python 'butter' function. The high pass function uses a 6^{th} order filter. The filter was designed to cause minimal distortion on the signals while still significantly reducing high frequency and low frequency noise.

To showcase the effectiveness of this subsystem, an experiment was run. The performance of the subsystem was tested by varying individual signal and filter parameters and recording the effects on the overall system accuracy.

8.3.2.3 Synchronisation

The frequency range of the calibration signal was moved to 6-10 kHz to accommodate the change in the source signal frequency to 1-5kHz. A dead band of 1kHz was kept between the signals to reduce the chance of frequency aliasing. As discussed in the Subsystem Implementation section, time-slicing was implemented. The audio recordings were each divided into two halves. The first half was considered the calibration signal, and the second half was considered the source signal. The synchronisation delays are estimated using the GCC-PHAT algorithm. The calibration signal acquired from Mic 1 is used as the reference signal for the cross function. The calibration signal of Mic 1 was cross correlated with itself to ensure that a delay of zero is obtained.

Additionally, to test further this subsystem, the position of the source was varied. A plot of the synchronised signals was displayed to ensure that the signals were correctly synchronised.

8.3.2.4 Time Delay Estimation

The time delay estimation is done using GCC-PHAT cross correlation of the signal from Mic 1 with the signals from the other Mics. This gives an estimate of the TDoA values between Mic 1 and the other Mics. This assumes that the signals have already been correctly synchronised, and the calibration signal has been removed. No additional equipment or components were used for this subsystem, as it was entirely software based.

For the testing of this subsystem, the synchronised signals from the synchronisation experiment ran previously were the input signals to the GCC-PHAT cross correlation function. A plot of the GCC-PHAT cross correlations of the signals was taken to display the results.

Additionally, the TDoA values were also recorded for all the tests performed on the overall system and displayed on the GUI.

8.3.2.5 Triangulation

Triangulation is done using a non-linear least squares estimation. The default trust-region-reflective algorithm of the MATLAB lsqcurvefit function is used. A function is defined that takes in the coordinates of the sound source and the positions of the Mics and outputs the differences in the distances from Mic 1 to the source and the other Mics to the source. The lsqcurvefit function then estimates the source position using the estimated TDoA values multiplied by c as the distances. Furthermore, this is done by iterating an initial position until if matches the expected distance values. The centre of the grid (0.4, 0.25) is used as the initial position as it is closest to all positions on the grid. Additionally, a lower and upper bound are placed on the solution. The lower bound is at the origin and the upper bound is at the maximum coordinates of the grid.

The default parameters were used in the experimentation of this subsystem, along with the position of the sound source being set to (0.8,0.5), which is at the far-right corner of the grid. The iterations of nonlinear least squares estimation algorithm were plotted to illustrate the steps taken to reach the final estimated source position.

8.3.2.6 Graphical User Interface

In order to visualize the process more clearly, a Graphical User Interface (GUI) was created to showcase the accuracy and precision of the overall system, as well as the effectiveness of each of the various subsystems involved in the overall system. From Figure 6 buttons can be seen.

- **Start Localisation**: Starts the recording process and, thereafter, calculates the estimated position.
- **Reset Grid**: Clears the grid of all estimated positions.
- **Exit**: Closes the GUI
- **Signal Acquisition Plots**: Displays the signals before being processed as well as after being processed.
- **Synchronisation Plots**: displays the signals before and after being synchronised.
- **Time Delay Estimation Plots:** displays the cross-correlation outputs obtained by the GCC-PHAT function.

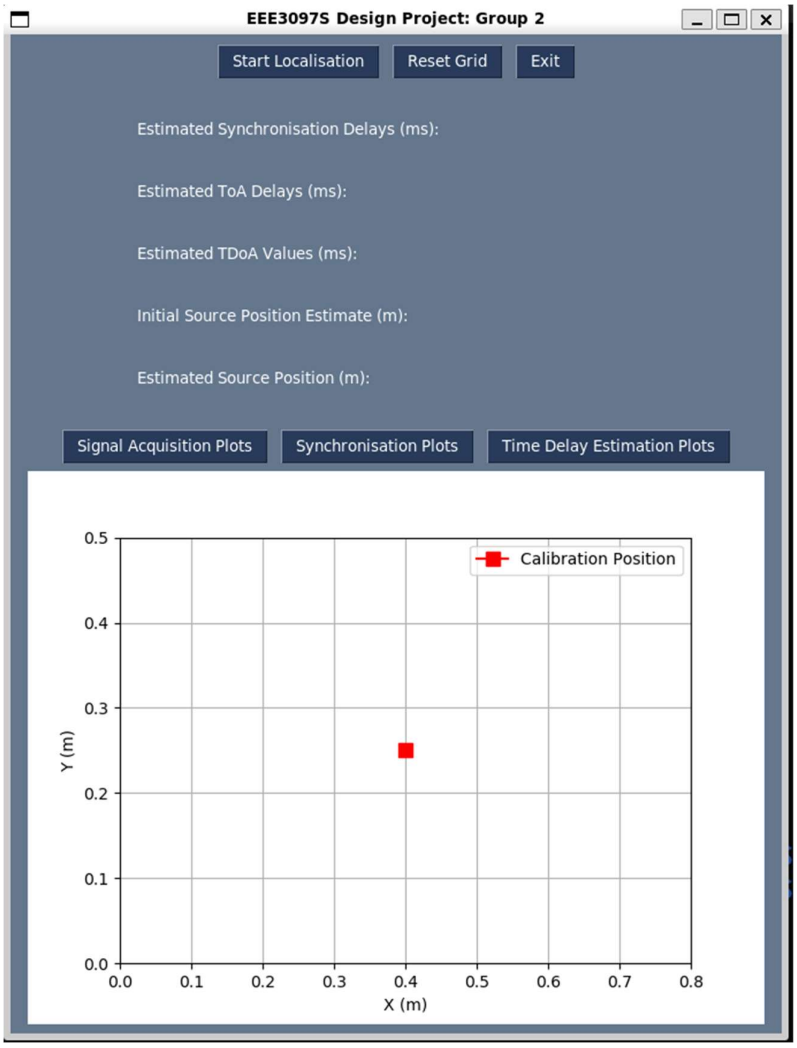


Figure 42: Experiment GUI

The GUI also comprises of a 0.8x0.5 grid, which emulates the A1 grid. The Calibration Signal is plotted at runtime, denoted as a red square. The user then presses the button labelled “Start Localisation”. This starts the recording of the microphones and, thereafter, calculates the estimated coordinates. In this case, the recorded sound was located at the point (0.4,0.4).

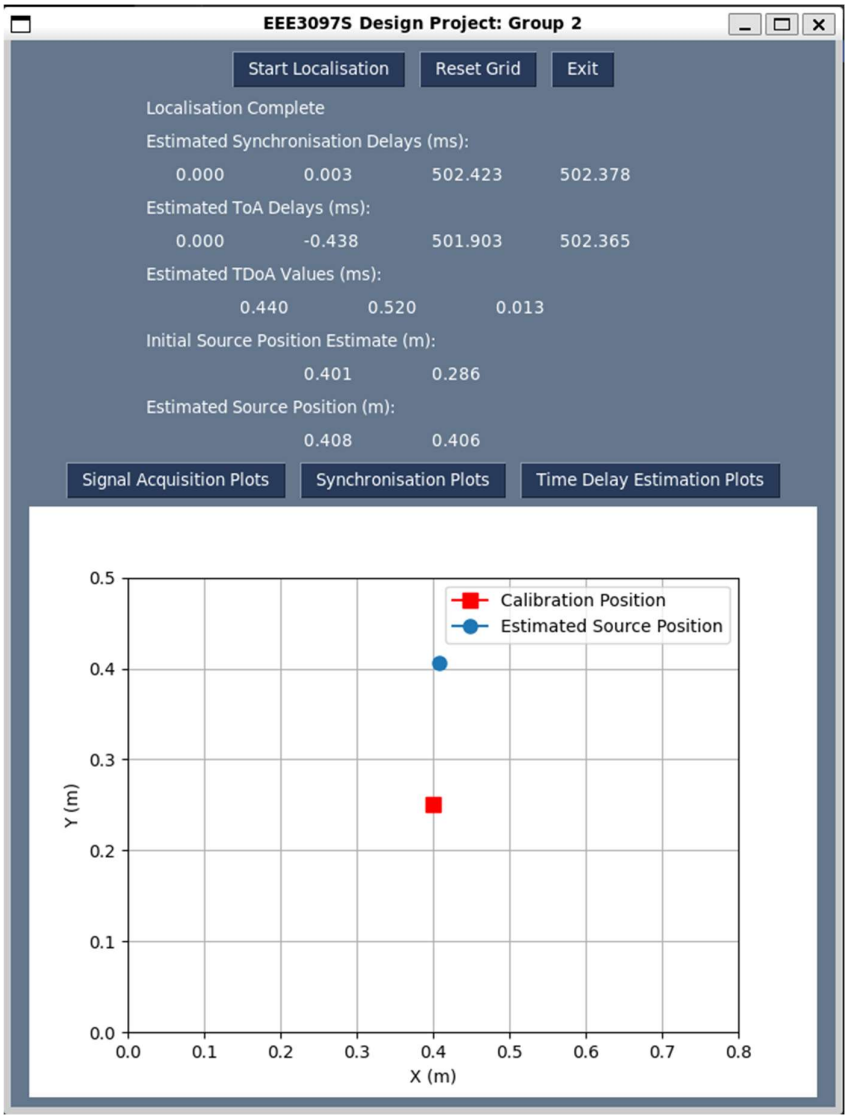


Figure 43: Showing the Actual and Predicted position on the grid.

The Sound Source location is thereafter displayed on the grid as a dot. The provided labels display the actual numerical values, the position error, and the TDoA estimation obtained when calling the respective Python functions.

The simulation also accounts for invalid coordinates provided, e.g, the coordinates are outside the bounds of the grid. The sound source was located at the point (0.4,0.6).

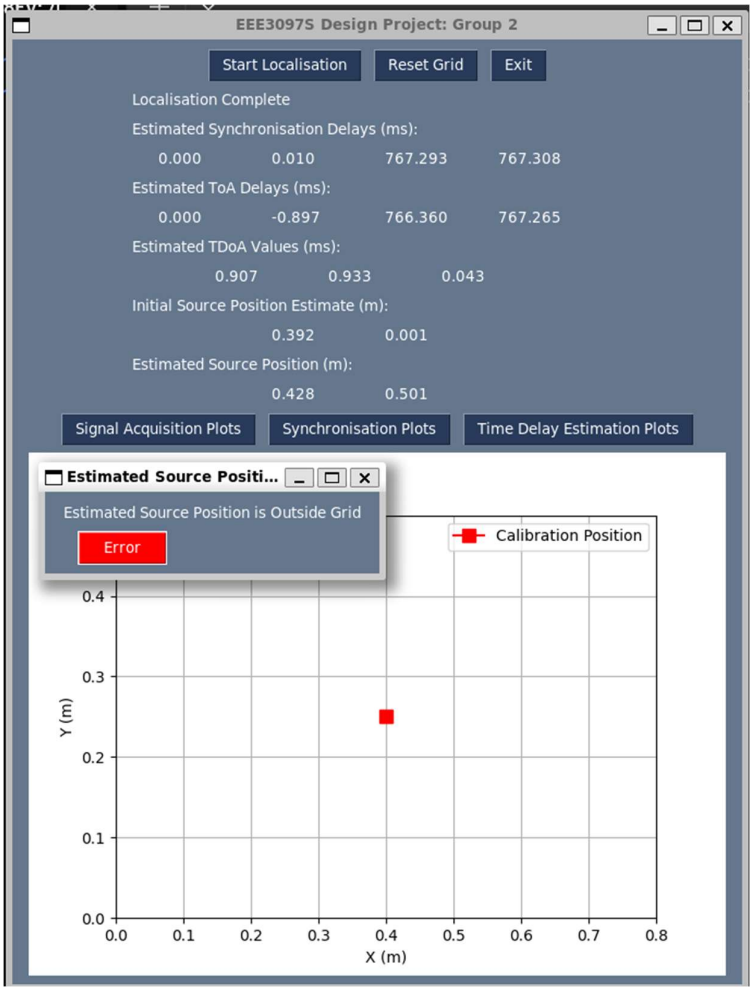


Figure 44: Error message when the coordinates are out of bounds.

The triangulation function will only generate a maximum x-coordinate of 0.801 and a maximum y-coordinate of 0.501.

8.3.3 Data Collection and Analysis

8.3.3.1 Process for Collecting Acoustic Signals

As expressed in the System Implementation section, a series of bash scripts were used to record and store the signals for analysis. Both Raspberry Pis contained the bash script shown in Figure.

```
#!/usr/bin/bash

# Define the output file name
OUTPUT_FILE="recording.wav"

# Use arecord with the specified options
arecord -D plughw:0 -c2 -r 25000 -f S32_LE -t wav -V stereo -v -d 16 "$OUTPUT_FILE"

# Check if the recording was successful
if [ $? -eq 0 ]; then
    echo "Recording completed successfully."
else
    echo "Recording encountered an error."
fi
```

Figure 45: raspberrypi.sh script.

This bash script handles the recording of the audio signals. The 'arecord' command is set up to record in stereo mode for 16s, at a sampling frequency of 25000Hz. The output file is amended based on which RPI module the script is on.e.g. on Raspberry Pi 1, the output file was renamed to 'recording_1.wav'. The message "Recording completed successfully" is displayed when the recordings are finished.

The bash script shown in Figure 15, is named pi_one.sh and is used to ssh into Raspberry Pi 1. Since the Pi is password protected, 'sshpass', along with the password 'group2' as an argument is used to speed up the process, to prevent the program from requesting a password. The only differences between pi_one.sh and pi_two.sh is that pi_one.sh has a 'sleep' command for 3 seconds, and they each contain their respective IP addresses.

```
#!/usr/bin/bash
sleep 3
sshpass -p 'group2' ssh group2@192.168.242.97 'bash -s < raspberrypi.sh'
```

Figure 46: pi_one.sh script.

```
#!/usr/bin/bash
sshpass -p 'group2' ssh group2@192.168.242.43  'bash -s < raspberrypi.sh'
```

Figure 47: pi_two.sh script.

In order to implement both of these scripts at the same time, on the local host, the bash script shown in Figure 17 is executed. This script is named main.sh. The line "./pi_one.sh & ./pi_two.sh" starts the execution of both those bash scripts at relatively the same time. There is, however, a small delay between the execution of pi_one.sh and pi_two.sh. The next two lines are executed once the recordings are completed. These 2 lines copy the recording files onto the local host via SCP and sshpass.

```
#! /usr/bin/bash

./pi_one.sh & ./pi_two.sh

sshpass -p 'group2' scp group2@192.168.242.97:recording_1.wav .
sshpass -p 'group2' scp group2@192.168.242.43:recording_2.wav .
```

Figure 48: main.sh script.

The main.sh script is ultimately executed in the Python program as shown in Figure .

```
subprocess.run("bash main.sh", shell=True)
```

Figure 49: Script implementation in Python program.

The corresponding TDoA measurements are obtained through the synchronisation and TDoA algorithms expressed in the Experimental Setup section, denoted as 'Synchronisation' and 'Time Delay Estimation'.

8.3.3.2 Preliminary Analysis or Insights gained from the Collected Data.

It was found that there was a slight 'pop' at the beginning of the microphone recordings. This proved to cause inaccuracies in the estimated positions.

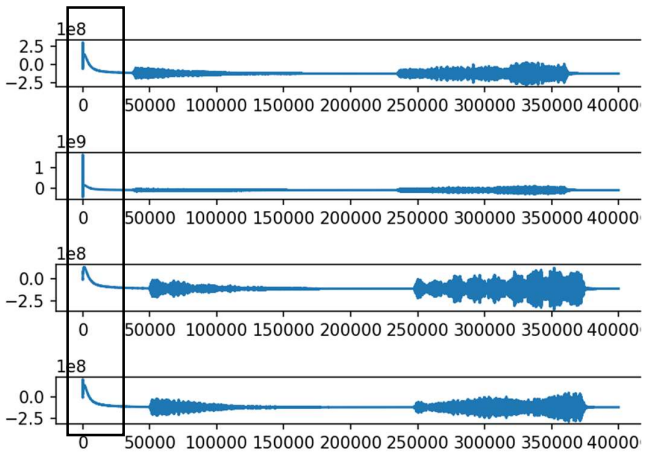


Figure 50: Original, raw audio recordings, illustrating the 'pop' at the beginning of the recording.

This is portrayed in Figure 19. In order to eliminate this 'pop', preprocessing procedures were conducted and were discussed in the System Implementation section, under the Signal Acquisition subheading.

It was also noted that the time between the calibration signal and the source signal should be at least 3s, in order to avoid clipping when the audio recordings are clipped in half.

It was also observed that an excess in noise caused the start of the source signal to become indistinguishable from that noise, which yielded inaccurate results. As such, in order to eliminate as much noise as possible, the volume of the audio signals would have to be increased to a maximum.

8.4 RESULTS

8.4.1 Overall System

The performance of the overall system was characterized by the position error the estimated source positions with respect to the actual source positions. The position error was defined as the distance - in metres - between the actual position and the estimated position.

8.4.1.1 Default Parameters

The result of the overall system performance at default parameters is given in the figure below:

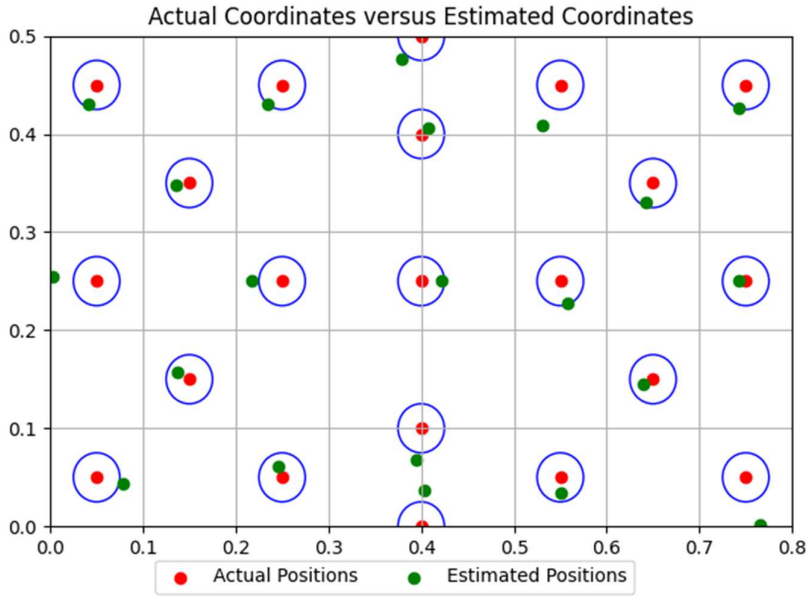


Figure 51: Graph of actual coordinates versus estimated coordinates

It can be observed that Figure 20 illustrates the accuracy of the system. Additionally, the blue circles depict the footprint of the speakers used. The size of the speakers represents an uncertainty in the actual position of the source signal. The use of smaller sized speakers would decrease this uncertainty, however there were no other speakers available for use. Figure 20 shows that 13 of the 21 positions estimated lie within the extent of the speakers. Moreover, most of the estimates that are not within this limit are visually close to it. This suggests that the system functions as intended despite the errors present. Furthermore, the position estimates do not appear to have a consistent error that could be accounted for by a constant offset. Lastly, the position estimates were rounded to a precision of 3 decimal places as this provides millimetre level precision which is the most that can be reliable measured physically. The precision of the estimated positions with respect to the repeated recordings at the same positions was not measured due to time constraints limiting the number of recordings that could be taken.

The position errors of estimated positions of the audio recordings taken are given in the figure below:

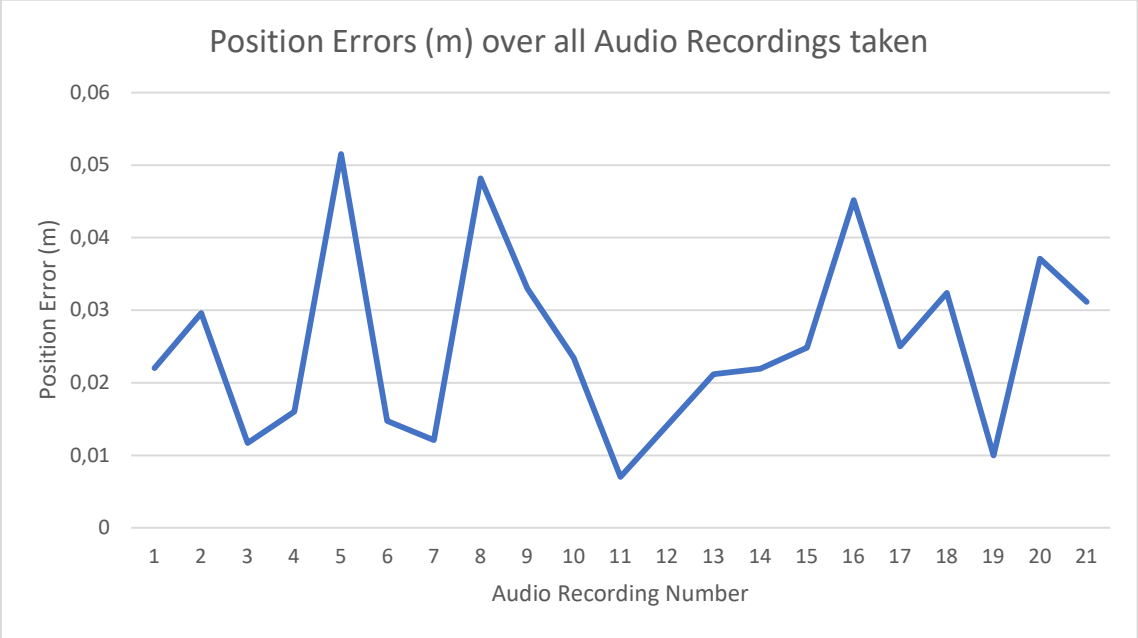


Figure 52: Graph of the position errors for the audio recordings taken.

Figure 21 shows that the position errors can fluctuate by around 5cm. There appear to be 3 outliers that are much greater than the rest of the values. These correspond to audio recordings 5, 8 and 16 which occur at positions (0.75, 0.05), (0.05, 0.25), (0.55, 0.45) respectively. These points are all close to the edge of the grid and which suggests the system may have degraded performance at the edges.

The statistics of the position error of the overall system are given in the table below:

Mean Position Error (m)	0.025
Median Position Error (m)	0.023
Minimum Position Error (m)	0.007
Maximum Position Error (m)	0.052
Standard Deviation of Position Error (m)	0.013

Table 7: Statistics of the Position Error of the Overall System

The statistics can be more intuitively illustrated by the Box and Whisker plot below:

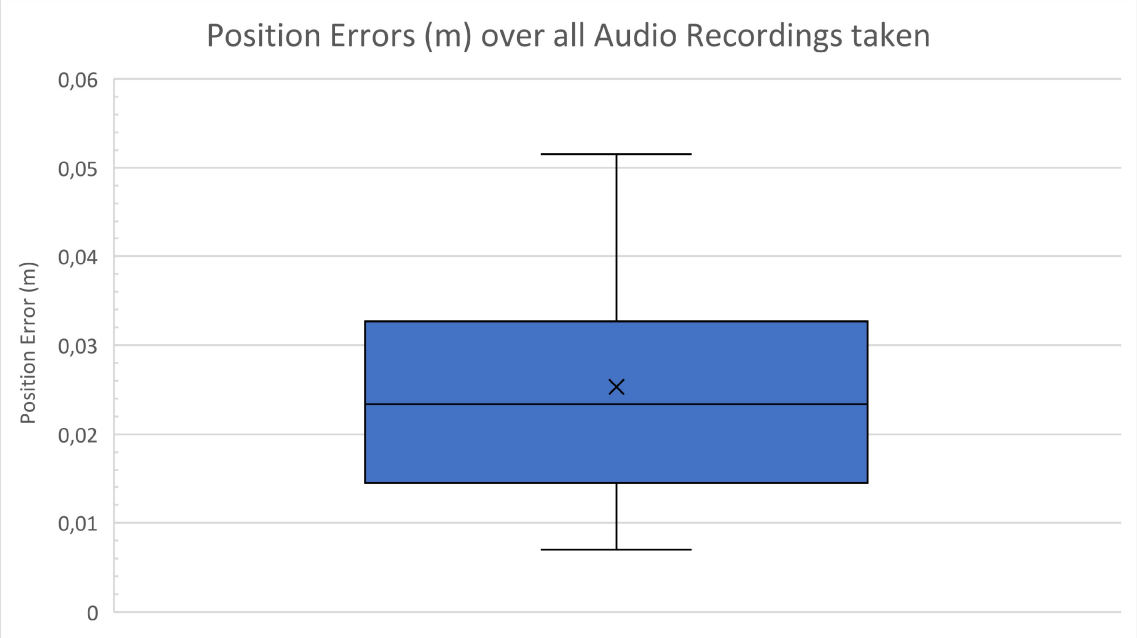


Figure 53: Box and Whisker Plot of the Position Errors of the Overall System

Figure 22 illustrates the distribution of the position errors over the recordings taken. Furthermore, it shows that 75% of the recordings produced a position error less than approximately 0.032m. This represents a significant majority of the recordings taken. And it is only 0.007m or 7mm greater than the extent of the speakers used. This further supports the observations made about figure.

The physical system can be compared to the simulated system to determine how closely its performance matches the performance predicted by the simulation. The performance of the simulated system is shown in the figure below:

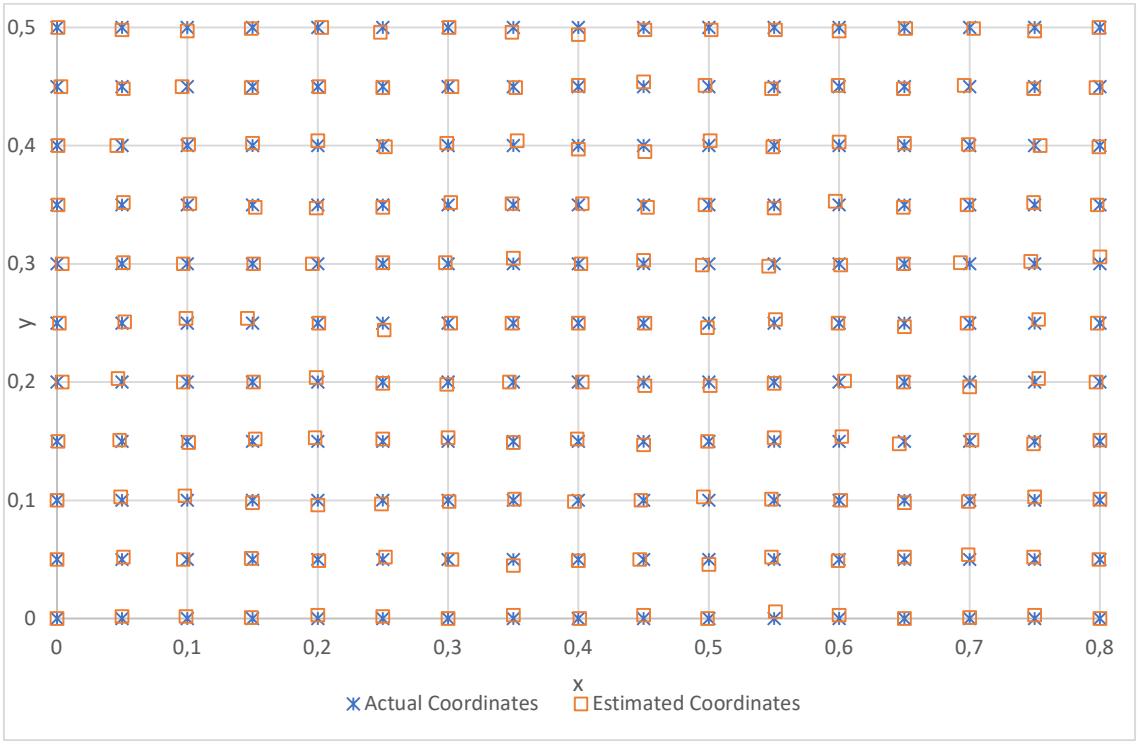


Figure 54: Simulation Results.

This shows that the results of the physical implementation are significantly worse than simulated. This likely due to the simplifications made in the simulation that do not take some of the physical non-ideal conditions into account. For example, the simulation was performed using signals with constant amplitude and only White noise was considered. Thus, this shows the significance of the non-ideal parameters. Improvements to the system implementation can be made by focusing on these areas.

As another point of comparison, the statics of the position error of the simulated system are given in the table below:

Mean Position Error (m)	0.003
Median Position Error (m)	0.002
Minimum Position Error (m)	0
Maximum Position Error (m)	0.007
Standard Deviation of Position Error (m)	0.001

Table 8: Statistics of the Simulation Position Error of the Overall System

These statistics suggest that the physical system performs approximately 10 times worse than expected from the simulations. This implies that there is significant room for improvement by optimizing the system to account for the non-ideal effects. This would likely involve more sophisticated signal processing techniques such as Kalman filtering which could not be applied due to time constraints.

8.4.1.2 Varying White Noise Levels

The results of the performance of the overall system with varying levels of background White Noise is shown below:

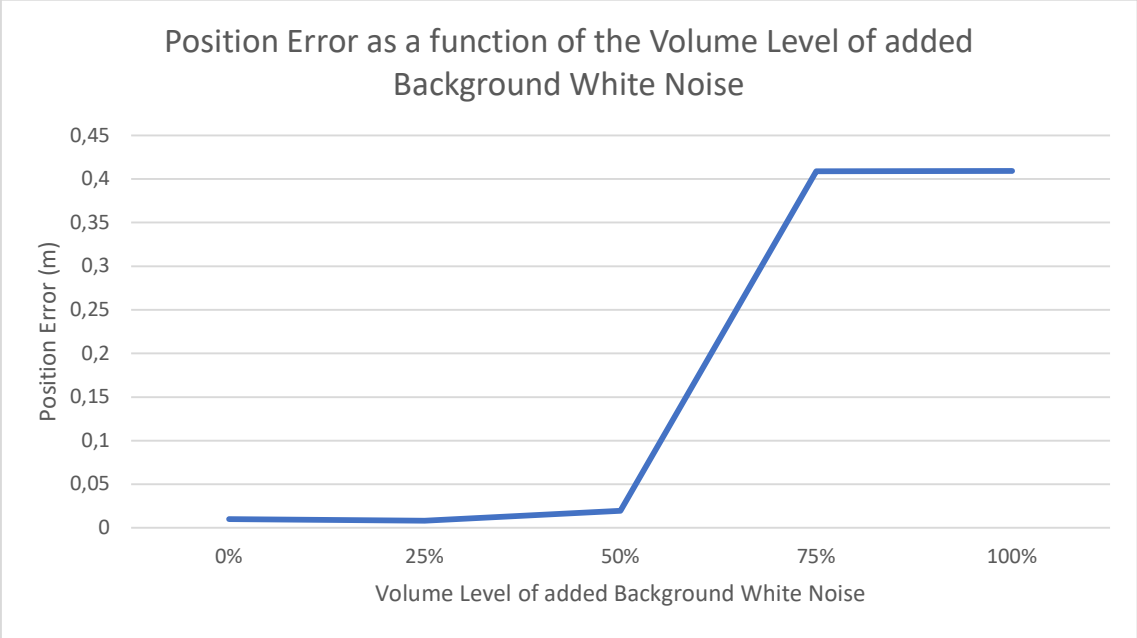


Figure 55: Graph depicting the position error as a function of the volume level of background white noise.

This shows that the system performs optimally for a certain level of noise and is totally inaccurate above a certain level of noise. The threshold lies somewhere between 50% and 75% of the max noise volume produced. Due to technical limitations the volume of noise cannot be meaningfully quantified due to technical limitations in the production of the noise. Thus, only a qualitative analysis is performed on this data. The implications are that the system is strongly sensitive to background noise above this unknown threshold. The threshold could be determined experimentally by measuring the noise signal prior to performing the source position estimation. However, this was not done due to time limitations. Thus, it is suggested to limit the application of the current system implementation to optimal environments with low levels of noise.

8.4.1.3 Varying Source and Calibration Signal Frequency Bands

The performance of the system for varying source and calibration signal frequency bands is illustrated by the table below:

Source Signal Frequency Range (kHz)	Calibration Signal Frequency Range (kHz)	Position Error (m)
1-2	3-4	0.413
1-3	4-6	0.032
1-4	5-8	0.018
1-5	6-10	0.01
1-2	8-10	0.413

Table 9: Position Error with reference to the Source Signal and Calibration Signal range varied.

The table shows that the position error is inversely proportional to the frequency bandwidths of the source and calibration signals. This suggests that a larger bandwidth maximises the performance of the system. Additionally, the comparison between the first and last values shows that the gap

between the source and calibration signal bands has no noticeable effect on the position error. This is likely because time slicing has been utilised to separate the source and calibration signals. However, a larger gap between their bands could allow for frequency slicing to be implemented which has the potential to be more versatile. This is because the signals would not have to be sent during fixed time intervals which proved to be a challenge to reliably achieve during the recording process.

As a point of comparison, the performance of the simulated system as a function of the maximum frequency of the source signal is shown below:

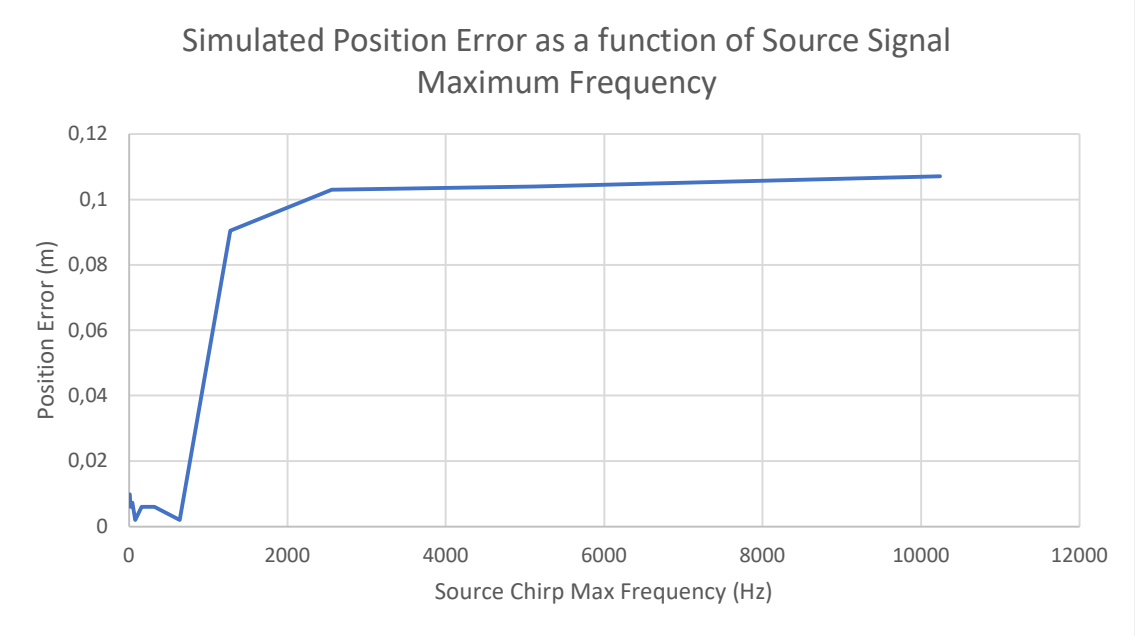


Figure 56: Graph depicting the Simulated Position Error as a function of the source signal maximum frequency.

Since the minimum frequency of the source signal was kept constant the Figure 56 shows the effect of increasing the source signal bandwidth. The simulation results appear to contradict the behaviour of the physical system as the position error increases with the increase of the source signal bandwidth. This is likely due to the differences in the implementation of the simulated system and the physical system. Namely, the frequency of the calibration signal and the separation of the source and calibration signals. The simulation relied on the source and calibration frequencies not overlapping and this condition was broken as the source signal's bandwidth was increased.

8.4.1.4 Varying Bandpass Filter Order

The performance of the system for variations in the order of the bandpass filter is shown below:

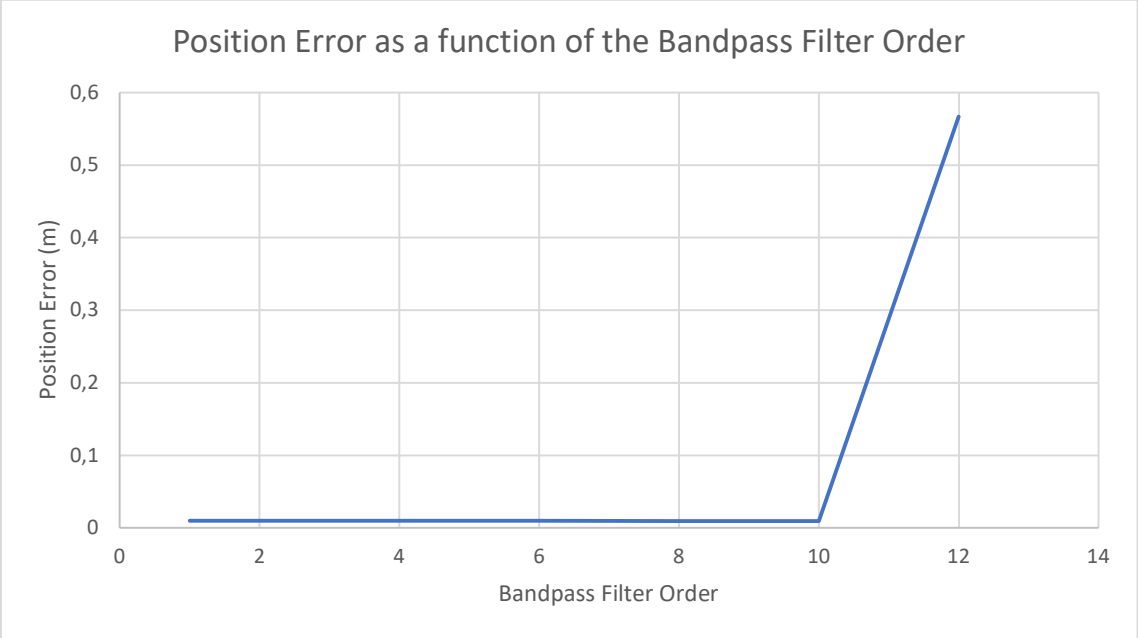


Figure 57: Position Error as a function of the Bandpass Filter Order.

The figure shows that the position error remains constant for orders below 10. Above an order of 10 the bandpass filter appears to introduce too much distortion to the signal. This suggests that filtering with a lower order filter is more resilient to distortions introduced by the bandpass filter. This must be balanced with the effect of the noise outside the frequency bands of the source and calibrations signals. A filter order of 6 was found to function acceptably according to the recordings taken at default parameters.

8.4.2 Signal Acquisition

The Signal Acquisition subsystem is responsible for capturing and preprocessing the audio signals. The performance of the subsystems preprocessing was used as a measurement for its performance. The SNR of the raw captured signals and processed signals were estimated to assess this performance. It should be noted that the SNR estimated were done by assuming that all the power within the frequency bands of the source and calibration signals are signal power and the rest is noise power. This assumption greatly simplifies the SNR estimation but also overstates the performance of the bandpass filter used to process signals.

8.4.3 Default Parameters

The performance of the Signal Acquisition subsystem is illustrated by the figure below:

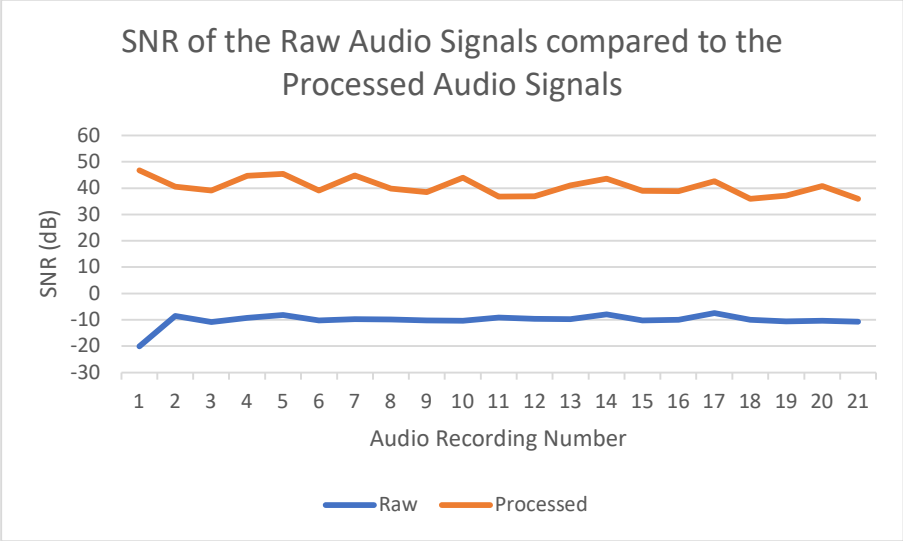


Figure 58: Graph of the SNR of the raw and processed audio signals recorded.

Figure 58 suggests that the Signal Acquisition subsystem performs well as it increases the SNR of the raw signals by approximately 50dB. This is represents a linear increase of 100,000 which is relatively large. However, as stated previously, the SNR measurement used overstated the performance of the bandpass filter. A more accurate measurement would thus be a comparison of the position error to the SNR of the processed signals.

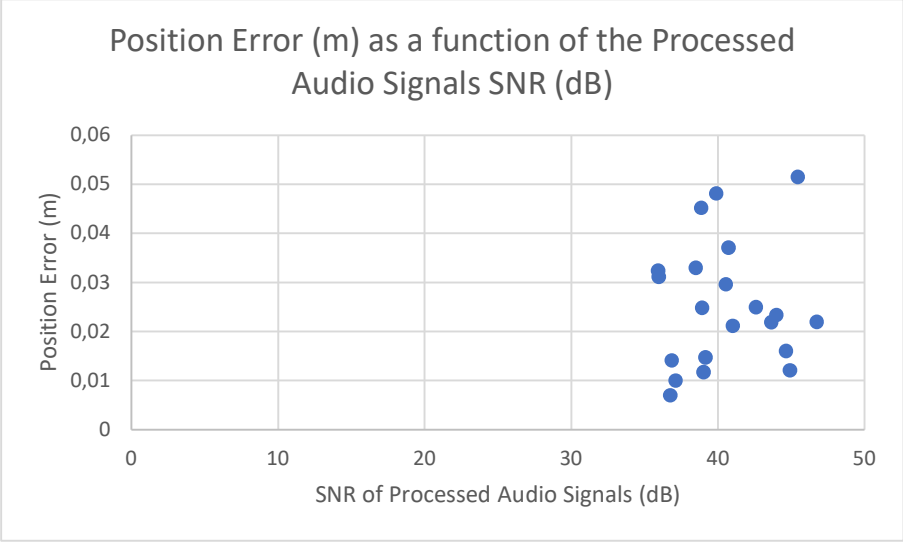


Figure 59: Graph depicting the position error as a function of the SNR of the processed audio signals.

Figure 59 shows that there is no notable correlation between the SNR of the processed signals and the position error. Although, this may be due to the limited range of SNR values observed. It is possible that that the system performance would degrade at SNR values below those observed. This will be examined using added background White noise which decreases the SNR below the levels observed ordinarily.

For comparison the performance of the simulated system with varying SNR of the raw audio signals is shown below:

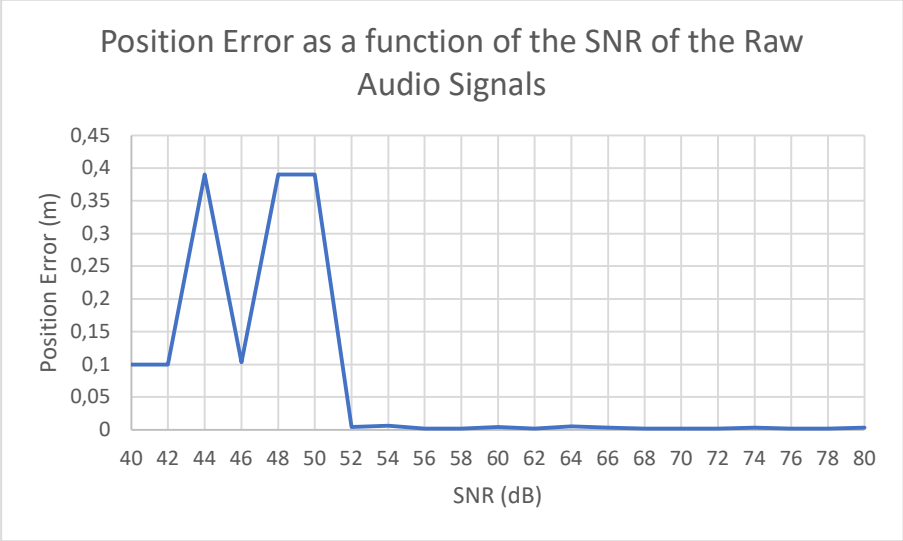


Figure 60: Graph depicting the position error as a function of the SNR of the raw signals for the simulation.

Figure 60 shows that the system performance deteriorates significantly for SNR values below 52dB on the raw audio signals. This does not match the behaviour of the observed system which functions acceptably at raw audio signal SNR values as low as -10dB and processed audio signal SNR values as low as 35dB. This is likely due to differences in the SNR measurement processes. Additionally, the majority of the raw signal noise for the physical system was outside the frequency bands of the source and calibration signals. This means that it could be easily filtered out to achieve a much larger increase in SNR for the processed signals. The SNR of the simulated system was evenly distributed and thus had a larger effect even after filtering.

8.4.3.1 Varying White Noise Levels

The performance of the Signal Acquisition subsystem for varying levels of background white noise is shown in the figure below:

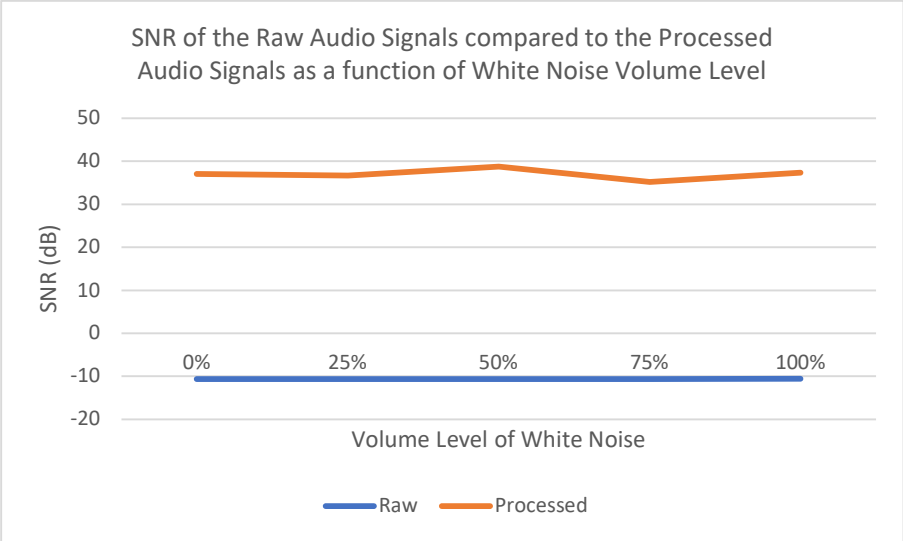


Figure 61: Graph of position error as a function of the lowpass filter cutoff frequency.

The figure shows that the added background white noise has no noticeable effect on the SNR values of the raw and processed signals. Considering the increase in the position error for volume levels above 50% this indicates that the current SNR measurements are inadequate to properly characterise the performance of this subsystem. Thus, a more sophisticated SNR estimation technique is required to assess the performance of the Signal Acquisition subsystem more accurately. This could not be implemented due to technical and time constraints.

8.4.3.2 Varying the Source and Calibration Frequency Bands

The performance of the Signal Acquisition subsystem for varying frequency bands of the source and calibration signals is shown in the table below:

Source Signal Frequency Range (kHz)	Calibration Signal Frequency Range (kHz)	SNR of Raw Audio Signals (dB)	SNR of Processed Audio Signals (dB)
1-2	3-4	-7.88	34.01
1-3	4-6	-6.78	40.68
1-4	5-8	-9.75	40.69
1-5	6-10	-10,69	37,1
1-2	8-10	-14.83	37.35

Table 10: Signal Frequency Variation and associated SNRs.

The table shows that the bandwidths of the source and calibration signals have no significant or consistent effects on the SNR of the processed audio signals. This seemingly contradicts the observations made about the effects of the bandwidths on the performance of the overall system. This further suggests that the SNR measurements are not sufficiently accurate.

8.4.3.3 Varying the Bandpass Filter Order

The performance of the Signal Acquisition subsystem for varying values of the bandpass filter order is shown in the figure below:

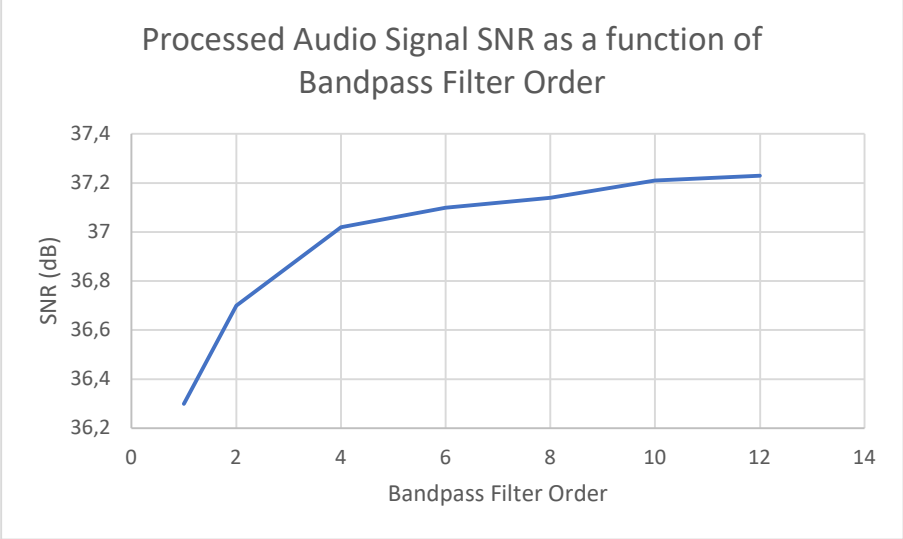


Figure 62: Graph Depicting SNR as a function of Bandpass Filter Order.

The figure suggests that increasing the filter order improves the SNR of the processed audio signals. This matches what is intuitively expected. However, the effect of this increase in SNR greatly diminishes as the filter order is further increased. Furthermore, it was observed earlier that increasing the filter order too much may significantly distort the signals. Thus, it can be seen from

Figure 62 that a filter order from 4 to 8 may be optimal. Thus, the chosen filter order of 6 falls within this optimal range.

8.4.4 Synchronisation

The performance of the synchronisation subsystem could not be quantitively measured as there was no way to measure the true synchronisation delay between the audio signals. Thus, the estimated synchronisation delays are compared to what the expected behaviour.

8.4.4.1 Default Parameters

The estimated synchronisation delays for the audio recordings take are shown in the figure below:

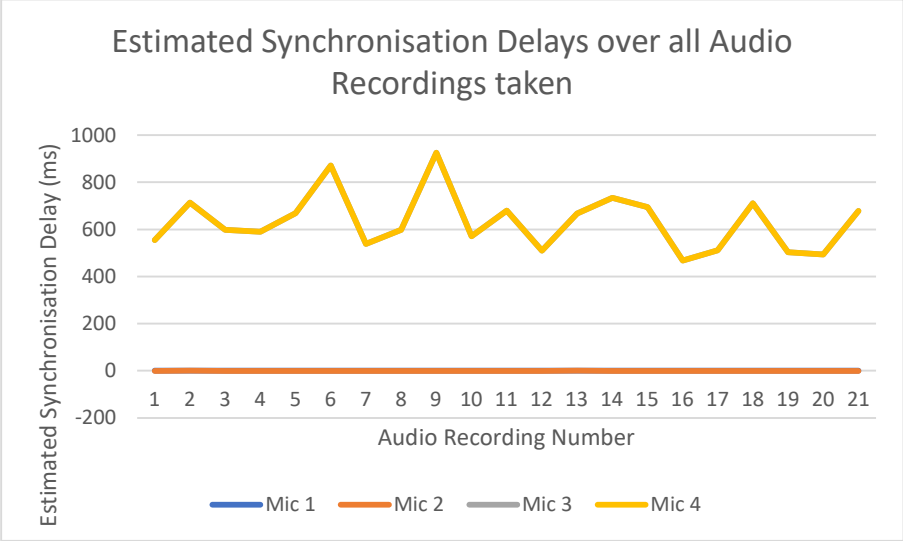


Figure 63: Estimated Delays for all audio recordings.

The figure shows that the estimated synchronisation delays are as expected. Since the calibration signal from Mic 1 is used as the reference the delays are expected to be around zero for Mics 1 and 2. Furthermore, the delays are expected to be some non-zero value that is approximately equal between the signals from Mics 3 and 4. Thus, the subsystem behaves as expected. The figure also illustrates that the synchronisation delays are much greater than 10ms despite utilising NTP. This suggests that NTP synchronisation performed significantly worse than expected. Consequently, the synchronisation could not be achieved solely through the use of NTP. Therefore, implementation of the calibration signal was necessary to achieve acceptable synchronisation between the RPi modules.

The statistics of the estimated synchronisation delays are shown in the table below:

Mean Synchronisation Delay (ms)	632.687
Median Synchronisation Delay (ms)	598.746
Minimum Synchronisation Delay (ms)	468.151
Maximum Synchronisation Delay (ms)	926.025
Standard Deviation of Synchronisation Delay (ms)	120.916

Table 11: Statistics of the Synchronisation Delays.

Additionally, the estimated synchronisation delay statistics are shown visually in the figure below:

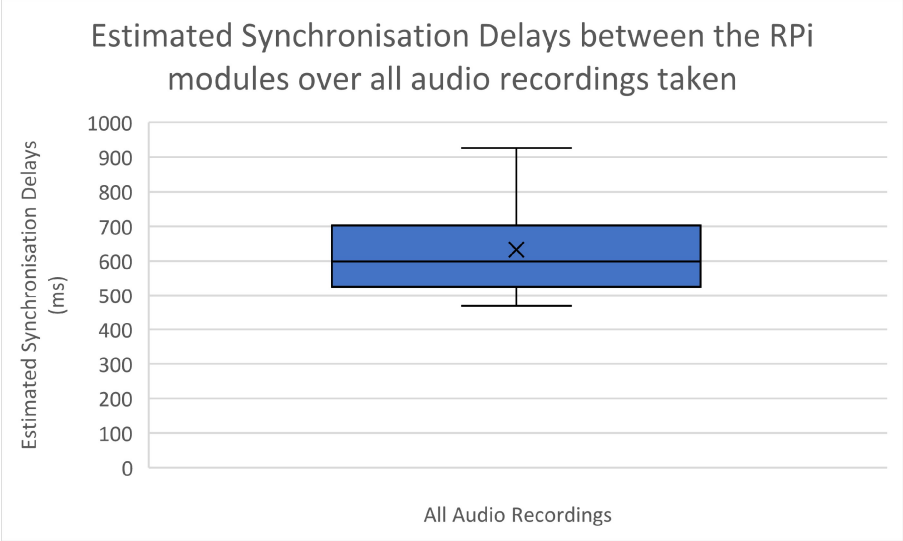


Figure 64: Box and Whisker Plot of the Estimated Synchronisation Delays for all audio recordings.

Both the table and the figure confirm that the estimated synchronisation delays are much greater than the expected 10ms expected from NTP synchronisation.

8.4.4.2 Varying White Noise Levels

The performance of the subsystem for varying levels of background white noise is shown in the figure below:

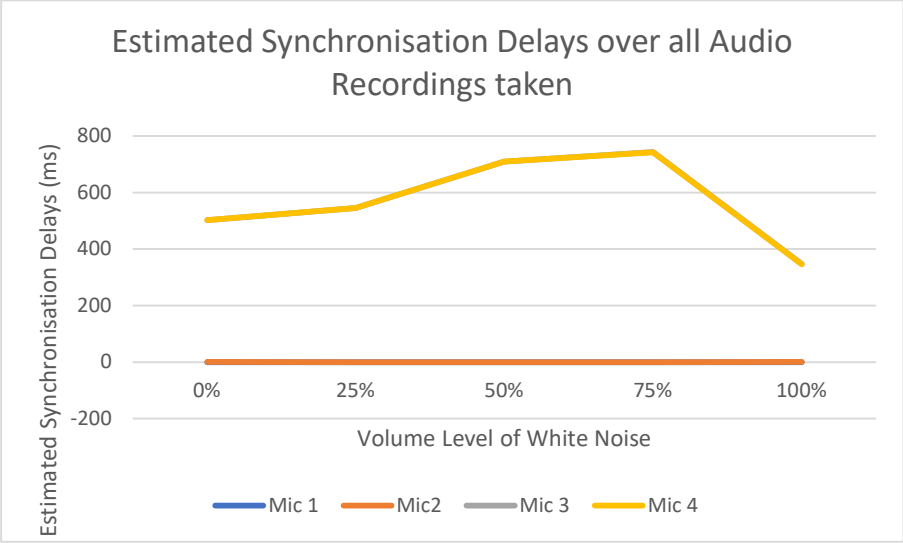


Figure 65: Estimated Synchronisation Delays under white noise conditions.

The figure shows that the subsystem performs equally well at all the levels of white noise tested. This is shown by the fact that there are no noticeable deviations from its expected behaviour.

8.4.4.3 Varying Source and Calibration Signal Frequency Bands

The performance of the Synchronisation subsystem for varying frequency bands of the source and calibration signals is shown in the table below:

Source Signal Frequency Range (kHz)	Calibration Signal Frequency Range (kHz)	Estimated Synchronisation Delays of Mic 1 (ms)	Estimated Synchronisation Delays of Mic 2 (ms)	Estimated Synchronisation Delays of Mic 3 (ms)	Estimated Synchronisation Delays of Mic 4 (ms)
1-2	3-4	0	0,038	801,072	801,088
1-3	4-6	0	0,115	419,61	419,535
1-4	5-8	0	0,023	472,768	472,95
1-5	6-10	0	0,002	502,422	502,378
1-2	8-10	0	0	407,315	415,07

Table 12: Estimated Delays with reference to Signal Frequency Variation.

The table shows that there are no noticeable deviations from the expected behaviour at all the tested frequency bands.

As a point of comparison, the performance of the simulated subsystem as a function of the maximum frequency of the source signal is shown below:

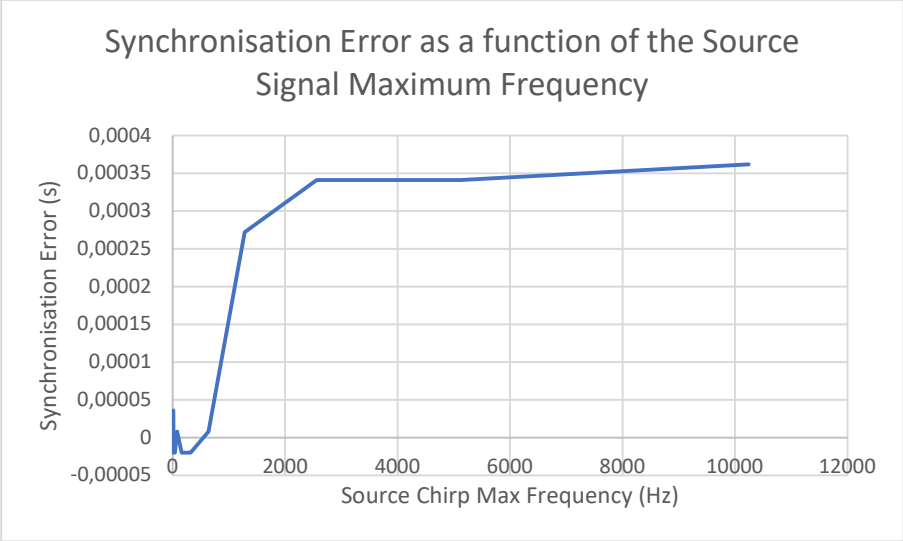


Figure 66: Graph depicting the Simulated Synchronisation Error as a function of the source signal maximum frequency.

Since the minimum frequency of the source signal was kept constant the Figure 66 shows the effect of increasing the source signal bandwidth. The simulation results appear to contradict the behaviour of the physical system as the synchronisation error increases with the increase of the source signal bandwidth. This is likely due to the differences in the implementation of the simulated system and the physical system. Namely, the frequency of the calibration signal and the separation of the source and calibration signals. The simulation relied on the source and calibration frequencies not overlapping and this condition was broken as the source signal's bandwidth was increased.

8.4.4.4 Varying Bandpass Filter Order

The performance of the Synchronisation subsystem for varying orders of the bandpass filter is shown in the table below:

Bandpass Filter Order	Estimated Synchronisation Delays of Mic 1 (ms)	Estimated Synchronisation Delays of Mic 2 (ms)	Estimated Synchronisation Delays of Mic 3 (ms)	Estimated Synchronisation Delays of Mic 4 (ms)

1	0	0,002	502,422	502,378
2	0	0,002	502,422	502,378
4	0	0,002	502,422	502,378
6	0	0,002	502,422	502,378
8	0	0,005	502,422	502,378
10	0	0,008	502,422	502,378
12	0	0,062	0,002	502,378

Table 13: Estimated Delays with reference to Bandpass Filter Order.

The table shows that the subsystem behaves as expected at all orders of the bandpass filter except the 12th order. This suggests that the increased position error observed at this filter order was at least partially due to incorrect synchronisation. This is because estimated synchronisation delay of Mic 3 is close to zero, but it is expected to be close to the value of the estimated synchronisation delay of Mic 4.

8.4.5 Time Delay Estimation

The performance of the Time Delay Estimation subsystem was measured using the error in the estimated TDoA values. This was defined as the difference between the estimated TDoA values and the ideal TDoA values. The ideal values were calculated according to the actual position of the source for each of the audio recordings taken.

8.4.5.1 Default Parameters

The performance of the Time Delay Estimation Subsystem is shown in the figure below:

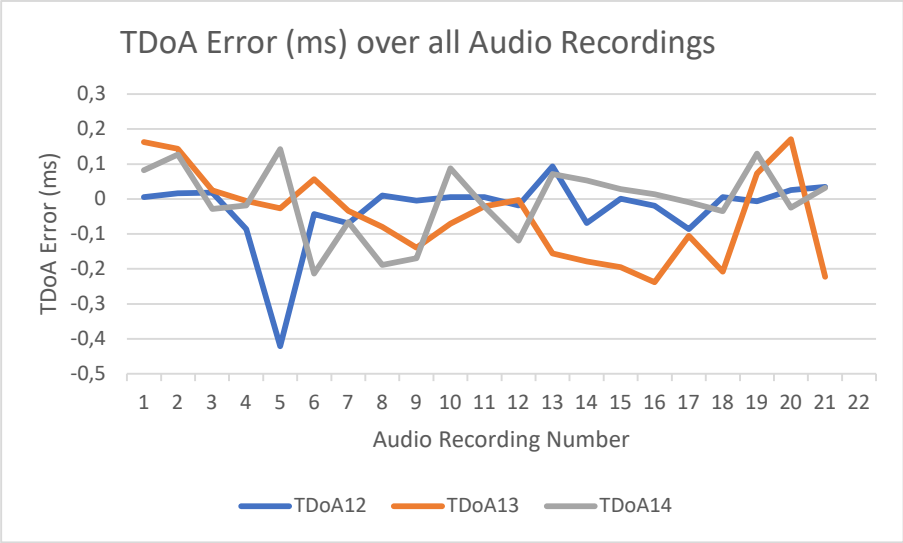


Figure 67: TDoA errors for all audio recordings.

The figure shows that the TDoA errors do not vary consistently or in similar ways. The TDoA errors appear to lie mostly between -0.2ms and 0.2ms with a bias towards the negative values. The negative values indicate that the estimated TDoA value was less than the ideal TDoA value. This suggest that the system is more likely to underestimate the TDoA values than it is to overestimate them. This asymmetry could potentially be reduced by applying a small constant offset to the TDoA values so that their mean and median values are closer to zero.

The statistics of the TDoA Errors are shown in the table below:

Mean TDoA Error (ms)	-0,029
Median TDoA Error (ms)	-0,010
Minimum TDoA Error (ms)	-0,421
Maximum TDoA Error (ms)	0,171
Standard Deviation of TDoA Error (ms)	0,110

Table 14: Statistics of the TDoA Errors.

Further, the statistics of the TDoA Errors are visually illustrated in the figure below:

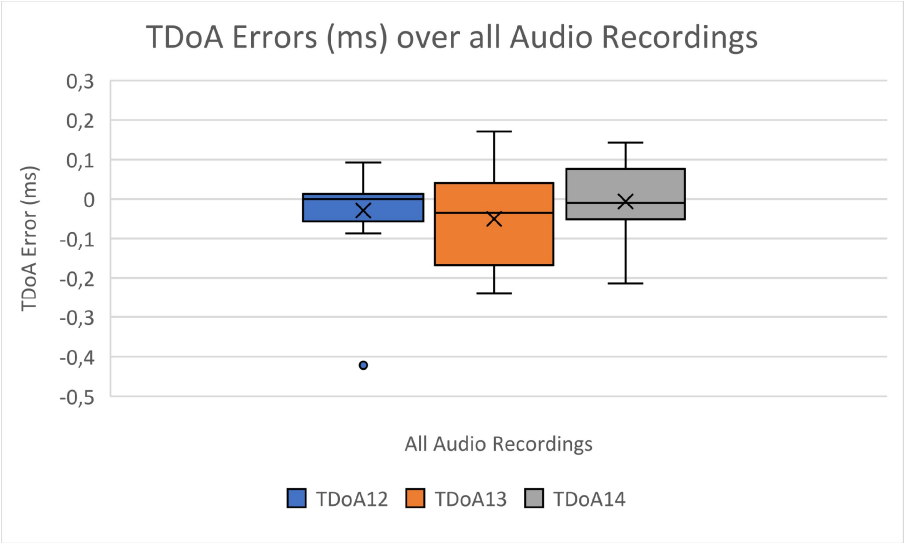


Figure 68: TDoA Errors for all audio recordings.

The table and figure confirm that the TDoA errors are biased towards the negative values. Additionally, the figure shows the differences in the distribution of the TDoA errors between Mic 1 and Mics 2, 3 and 4. The TDoA errors appear to correlate with the distances between Mic 1 and the other Mics. This suggests that Mics that are closer together produce more accurate TDoA values. The suggests that an alternative Mic arrangement could potentially be more optimal than the current implementation with the Mics at the corners of the grid. However, this was not tested due to time constraints. Additionally, TDoA errors one standard deviation away from the mean value would be from -0.139ms to 0.089ms. This accounts for approximately 68% of the values and would theoretically produce a maximum position error of 0.048m for 68% of the values. This is not what was observed in the performance of the overall system which indicated that approximately 75% of the position errors were below 0.032m. This is likely explained by the fact that the TDoA errors of each Mic with respect to Mic 1 do not appear to be strongly correlated. Thus, their combined effects result in a lower position error. Thus, the performance of the Time Delay Estimation subsystem can be considered acceptable.

8.4.5.2 Varying White Noise Levels

The performance of the Time Delay Estimation subsystem for varying levels of background white noise is shown in the figure below:

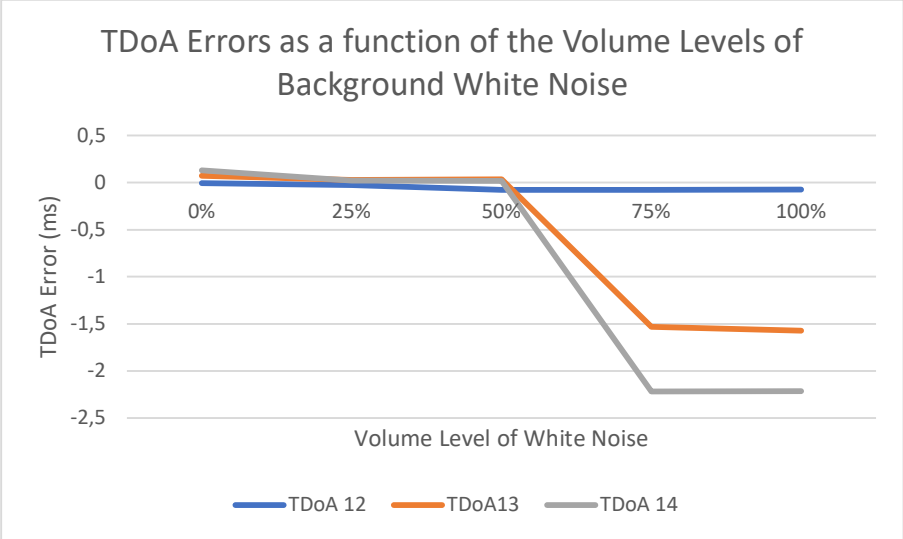


Figure 69: TDoA Errors as a function of Background White Noise.

The figure shows that the system performs optimally for noise levels up to 50% of the maximum white noise volume level. Above this level the TDoA errors increase significantly. This is likely due to errors in both the synchronisation delays and ToA delays which are combined to produce the estimated TDoA values. Thus, the increase in the TDoA errors is likely the cause of the observed increase in position errors at noise levels above the threshold. This suggests that a more robust time delay estimation technique may be required for the system to be more resilient to varying noise levels. This was not tested due to technical and time constraints. Consequently, this supports the suggestion of the limiting the use of the system in ideal environments where the noise levels are below the threshold.

8.4.5.3 Varying Source and Calibration Signal Frequency Bands

The performance of the Time Delay Estimation Subsystem for varying frequency bands of the source and calibration signals is shown in the table below:

Source Signal Frequency Range (kHz)	Calibration Signal Frequency Range (kHz)	TDoA 12 Errors (ms)	TDoA 13 Errors (ms)	TDoA 14 Errors (ms)
1-2	3-4	-0,28016	800,6248	801,088
1-3	4-6	0,090843	0,029843	-0,043
1-4	5-8	0,010843	-0,13516	0,032
1-5	6-10	-0,00716	0,072843	0,13
1-2	8-10	-0,31716	406,9098	415,07

Table 15: TDoA Errors with referemnce to Signal Frequency Variation.

The table shows that the TDoA errors are extremely large at bandwidths of 1kHz for the source and calibration signals. Additionally, it shows that the performance of the Time Delay Estimation subsystem remains acceptable for bandwidths from 2kHz to 4kHz. This supports the observations of the effect of the bandwidth on the position errors.

As a point of comparison, the performance of the simulated subsystem as a function of the maximum frequency of the source signal is shown below:

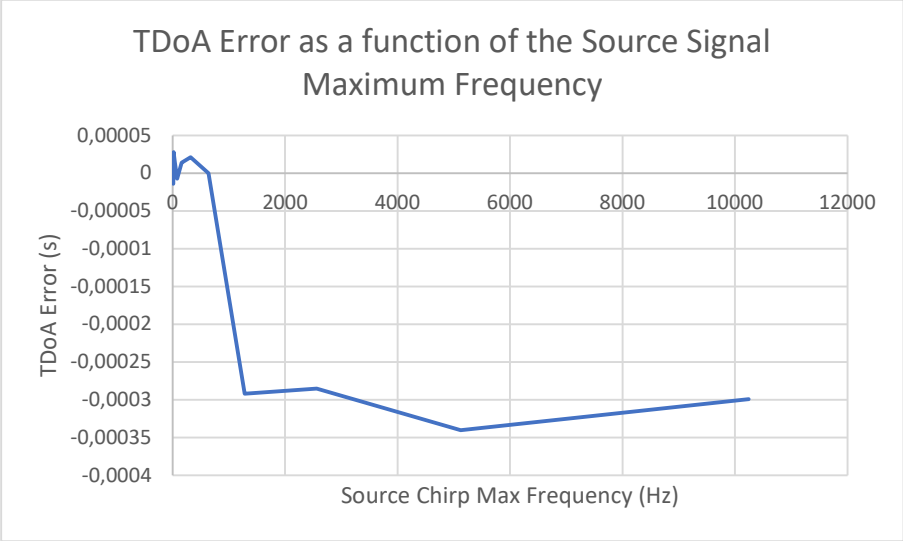


Figure 70: Graph depicting the Simulated TDoA Error as a function of the source signal maximum frequency.

Since the minimum frequency of the source signal was kept constant the Figure 70 shows the effect of increasing the source signal bandwidth. The simulation results appear to contradict the behaviour of the physical system as the TDoA error increases with the increase of the source signal bandwidth. This is likely due to the differences in the implementation of the simulated system and the physical system. Namely, the frequency of the calibration signal and the separation of the source and calibration signals. The simulation relied on the source and calibration frequencies not overlapping and this condition was broken as the source signal's bandwidth was increased.

8.4.5.4 Varying Bandpass Filter Order

The performance of the Time Delay Estimation subsystem for varying orders of the bandpass filter are shown in the table below:

Bandpass Filter Order	TDoA 12 Errors (ms)	TDoA 13 Errors (ms)	TDoA 14 Errors (ms)
1	-0,01216	0,072843	0,013
2	-0,01216	0,072843	0,013
4	-0,01216	0,072843	0,013
6	-0,00716	0,072843	0,13
8	-0,01916	0,072843	0,013
10	-0,01516	0,072843	0,013
12	0,050843	-502,347	0,013

Table 16: TDoA Errors with reference to Bandpass Filter Order.

The table shows that the TDoA values remain optimal for bandpass orders up to 10 but significantly deteriorate at a bandpass order of 12. This is likely caused by the errors in synchronisation observed at this filter order. Thus, this results in the observed position error at this filter order.

8.4.6 Triangulation

The performance of the triangulation subsystem is measured through the estimated source positions and the associated position errors. However, the same measurement was used to characterise the overall system. Thus, the results already presented in the overall system section will not be repeated but they apply to the Triangulation subsystem as well as the overall system. Instead, alternative measurements of the performance of the subsystem will be presented.

8.4.6.1 Performance of the Linear LSE

The linear LSE was added to the physical implementation of the system while it was absent from the simulated system. Since it was added to reduce number of iterations performed by the nonlinear LSE the performance of the linear LSE is determined by the initial position error. This is defined as the distance between the initial position given to the nonlinear LSE and the estimated position produced by the nonlinear LSE. Previously the centre point of the grid was used as the initial point as it the closest to all other points on the grid. Thus, the performance of the linear LSE can be evaluated by whether the initial errors it produces outperforms those produce by the centre point. The comparison of the initial position errors of the linear LSE and centre point are shown in the figure below:

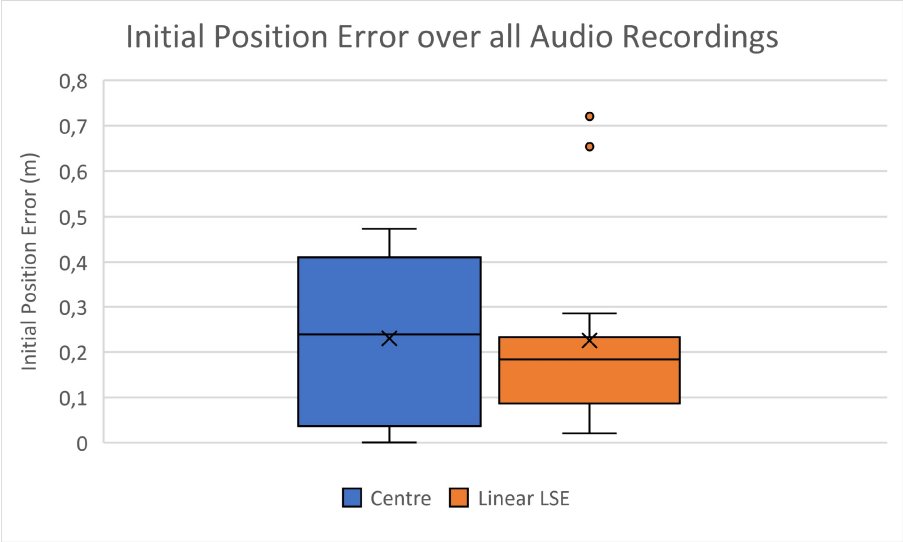


Figure 71: Box and Whisker Plot for Initial position for all audio recordings.

The figure shows that the linear LSE does outperform the centre point by a relatively significant margin. The range, mean and median of the initial position errors of the linear LSE are all less than those of the centre point. Additionally, 75% of the linear LSE initial position errors fall below approximately 0.22m while only approximately 50% of those of the centre point fall in the same region. Thus, the results show that the linear LSE sufficiently outperforms the centre point to justify its use with the added processing overhead. This is because the overhead of solving the linear LSE is relatively minor compared to that of additional iterations of the nonlinear LSE.

8.4.6.2 Performance of Non-Linear LSE

An alternative measurement of the performance of the Triangulation subsystem besides position error is the coordinate errors. This is defined as the errors in the estimated x and y coordinates compared to their ideal values. The performance of the Triangulation subsystem with respect to the coordinate errors are shown in the figure below:

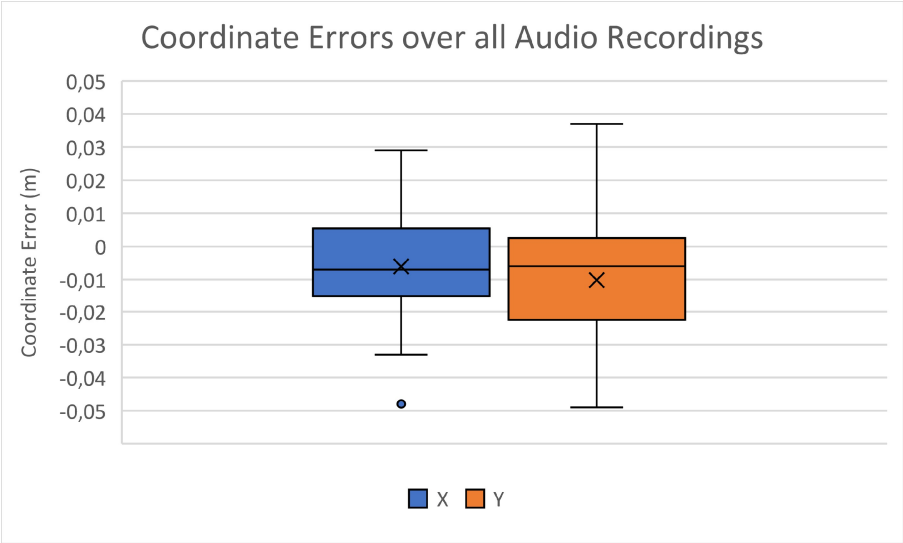


Figure 72: Box and Whisker Plot for Coordinate Errors for all audio recordings.

The figure shows that the errors in the x and y coordinates are not identical. This is expected as the x coordinates have a greater range that the y coordinates. However, the y coordinate errors appear to have a larger range than the x coordinate errors which is not explained by this. This could possibly be due to the fact that the positions and angles of the Mics result in more accurate x coordinate values. The complete cause of this cannot be determined without further investigation. This was not done due to time constraints.

The performance of the Triangulation subsystem can also be illustrated qualitatively analysing whether the estimated position is appropriate for the give TDoA values. This is because the position errors measured relative to the ideal position are influenced by the errors in the TDoA values as well as the performance of the Triangulation subsystem. Thus, analysing the estimated position without regarding the ideal position gives a more neutral measurement of performance. Five points were chosen on the grid to illustrate this performance. The points were (0.4, 0.25), (0.15, 0.15), (0.35, 0.15), (0.15, 0.65) and (0.35, 0.65). Plots illustrating the hyperbolas produced by the TDoA values acquired at these points and the estimated source position were generated. These plots are shown below:

Figure 73: Graph of the hyperbolas produced by the TDoA values acquired at a source position of (0.4, 0.25) and the associated estimated source position

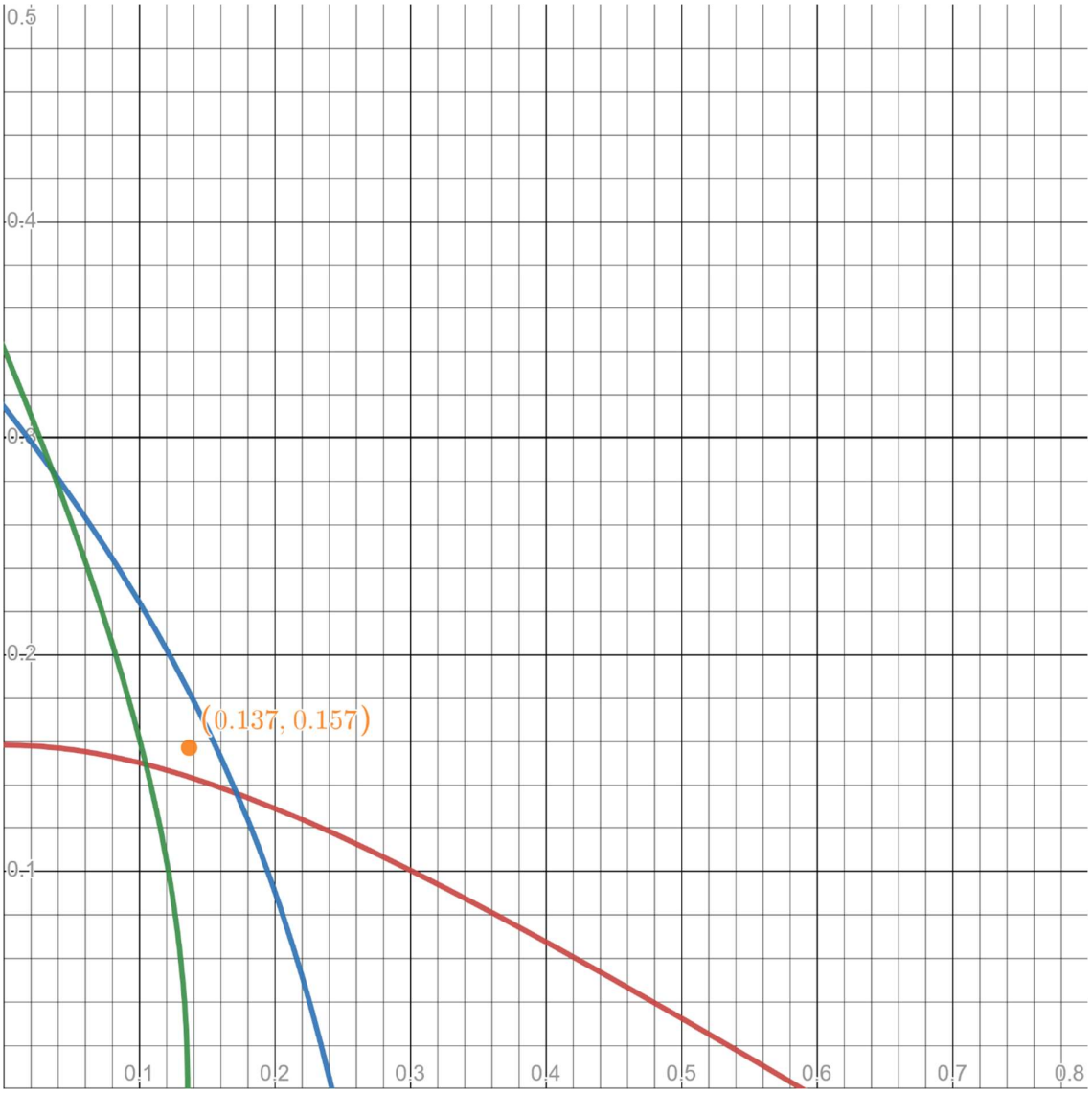


Figure 74: Graph of the hyperbolas produced by the TDoA values acquired at a source position of (0.15, 0.15) and the associated estimated source position

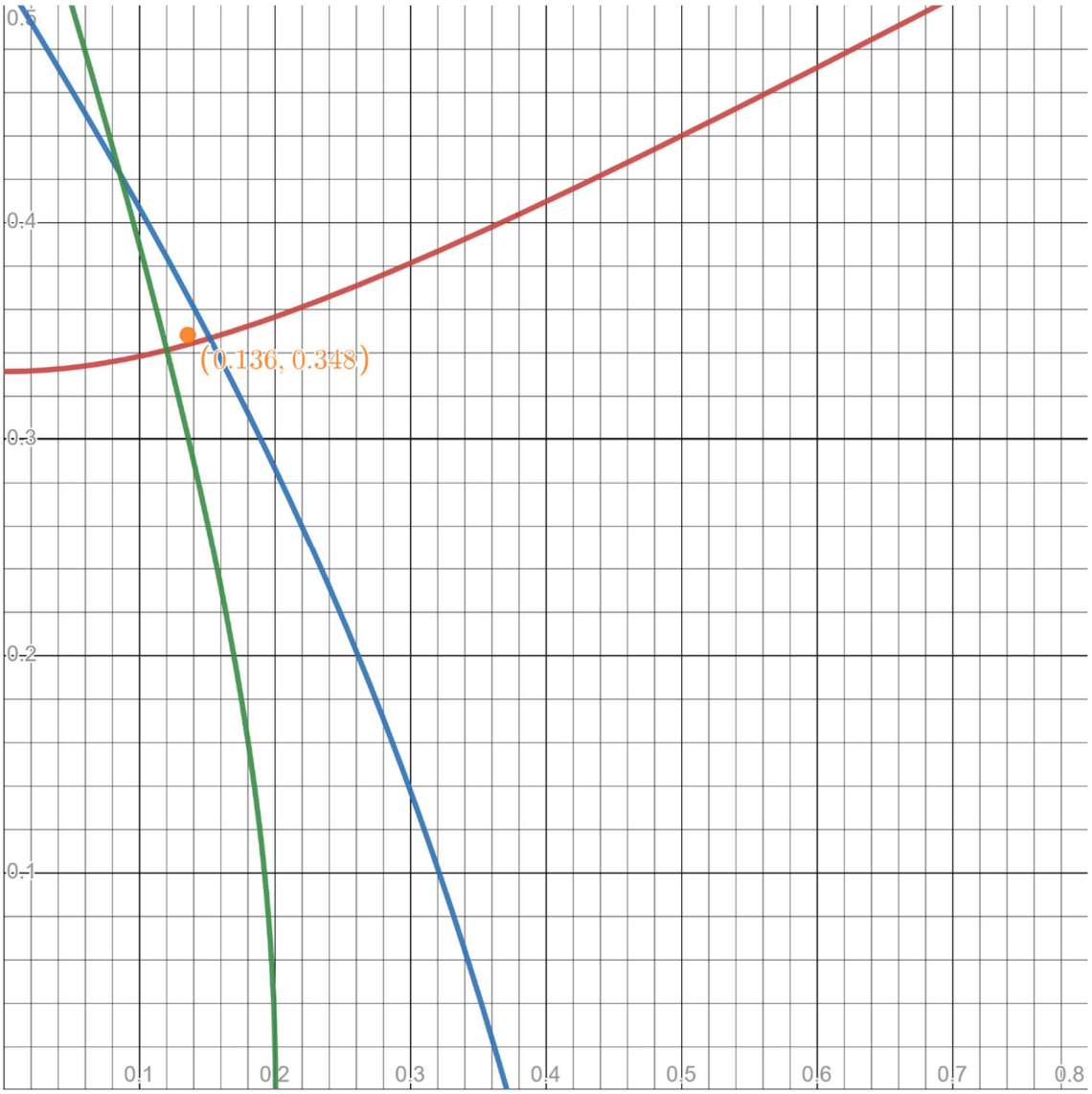


Figure 75: Graph of the hyperbolas produced by the TDoA values acquired at a source position of (0.15, 0.35) and the associated estimated source position

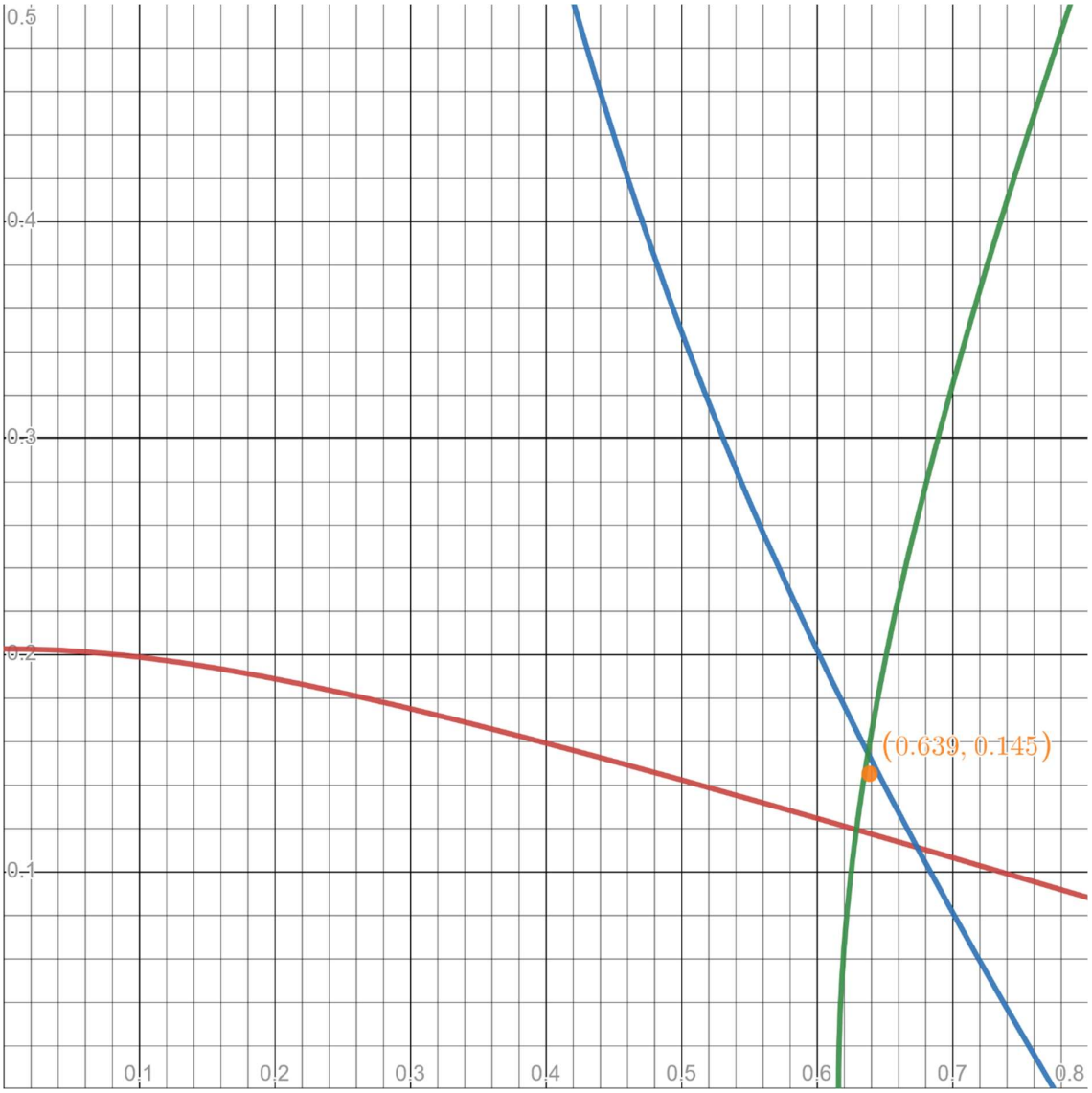


Figure 76: Graph of the hyperbolas produced by the TDoA values acquired at a source position of (0.65, 0.15) and the associated estimated source position

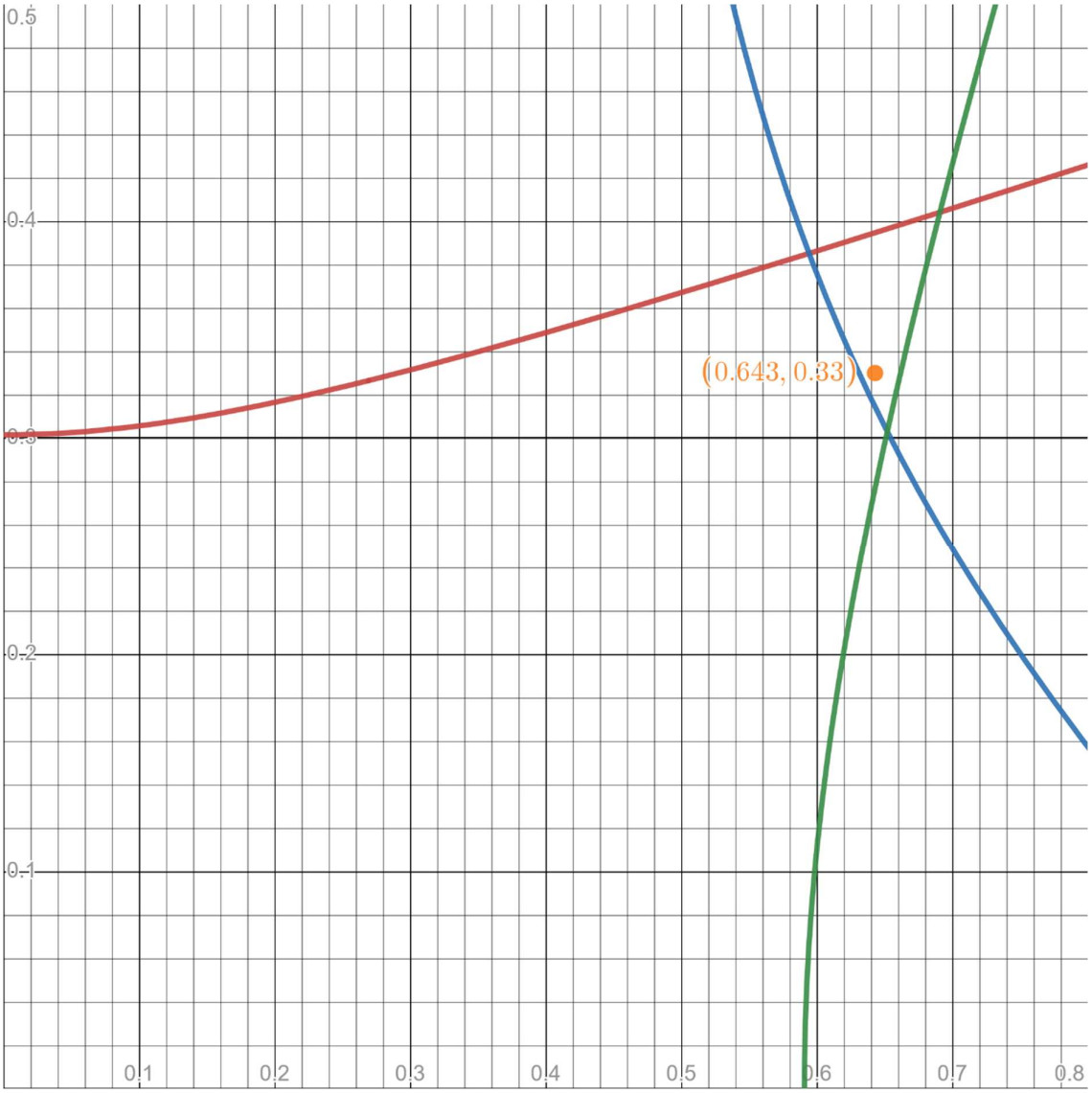


Figure 77: Graph of the hyperbolas produced by the TDoA values acquired at a source position of (0.65, 0.35) and the associated estimated source position

The plots show that for all of the positions chosen the estimated source position matches what is expected by the hyperbolas generated by the estimated TDoA values at the respective positions. Additionally, the plots show that the hyperbolas usually do not have a single intersection point and this explains the poor performance of the linear LSE initial position estimate compared to the nonlinear LSE position estimate. Thus, it can be concluded that the Triangulation subsystem performs acceptably.

9 CONSOLIDATION OF ATPS AND FUTURE PLAN

9.1 ATPS

Acceptance tests procedures (ATPs) of specifications

Specification for Acceptance Test	**Acceptance Test**	**Test has been met?**	**Changed specifications**	
			Old Specification	New Specification
1. **The system will use two RPi Zero W modules and the microcontrollers will operate in parallel.**	To test this specification, both the boards will be checked to see whether Raspbian OS has been installed and is working properly. The SD cards will also be checked to ensure it has not been corrupted. For the test for parallel operation, the time in which both the Pi's logged in using SSH (Secure-Shell Host) will be recorded.	**Yes**		
2. The system will use four Adafruit I2S MEMS Microphone breakout boards.	To test if these microphone breakout boards work appropriately, each will be connected to a Pi module, and will be tested to record audio for a specific amount of time.	**Yes**		
3. **The system will use a 0.8 x 0.5 m grid printed on an A1 (0.841 x 0.594 m) sheet of paper.**	Although this sheet of paper is provided, the grid will be checked for any abnormalities. It will also be checked to see if it is in good condition to ensure it can be used effectively.	**Yes**		
4. **The source and calibration audio signals will be generated**	This acceptance test will ensure that the two Bluetooth speakers that will be used for the calibration signal and	**Yes**	**The sound source will be generated using an**	**The source and calibration audio signals will be generated**

	using a pair of Bluetooth speakers.	the source signal, are both working properly and can output the necessary chirp signals at different frequencies.		**Android smartphone.**	**using a pair of Bluetooth speakers.**
5.	**The Bluetooth speakers will be each be connected to a mobile device.**	This acceptance test will involve checking the ability for the speakers to connect to a mobile device using Bluetooth.	**Yes**		
6. **7.**	**A laptop running the Microsoft Windows OS will be used as a host device.**	This acceptance test will be used to determine whether the host device, a laptop running the Windows operating system, will qualify to be the appropriate host device for the Pi modules.	**Yes**		
8.	**The RPi Microcontrollers will be powered via their Micro-USB ports**	To test this, a micro-USB cable will be connected from the USB port of a laptop to the micro-USB port of both the Pi modules.	**Yes**		
9.	**The RPi Microcontrollers will receive a nominal input voltage of 5 V DC at an input current of 2.5 A.**	For this test, a multi-meter will be used to check the voltage and current running through the Pi.	**Yes**		
9.	**The RPi Microcontrollers will be connected to the local network of the host device via Wi-Fi.**	The host network will try to connect the RPi's to its local network using WiFi. A PuTTY console was set up to connect to the IP address of the RPi's and connect them to the host network from there.	**Yes**		
10.	**The RPi Microcontroller**	To test this specification, both the	**Yes**		

s will communicate with the host via the SSH and SCP protocols.	Raspberry Pi Microcontrollers will be connected to each other, and then a 'ping' command will be sent from the console of one of the Pi's to the IP address of the other Pi and vice versa.			
11. **The microphone breakout boards will be powered via power connections to the RPi Microcontrollers.** 12. **The microphone breakout boards will receive a nominal input voltage of 3.3V DC.**	For these specifications, a multi-meter was used to ensure that the microphone was receiving the appropriate 3.3V power from the respective Pi's.	**Yes**		
13. **The microphone breakout boards will communicate with the RPi Microcontrollers via the I2S serial communication protocol.**	To test whether or not the microphones have successfully connected to the Pi's, the command "arecord -l" was entered into the Pi command line, which determines if the I2S card is recognised.	**Yes**		
14. **The system will be programmed using the Python programming language.**	The Python programming language must be tested to ensure it is the correct language to undergo all the aspects of the project, such as signal acquisition, signal processing, etc. The	**Yes**		

	appropriate libraries will then be tested to test this.			
15. **A Generalized Cross-Correlation Phase Transform algorithm will be applied to pairs of audio recordings, to acquire the TDoA data.**	To ensure that the GCC-PHAT algorithm works as expected, the TDoA delays for each of the microphones were plotted, to visually depict the performance of the cross-correlation algorithm.	**Yes**		
16. **The system should ensure that the time synchronisation error between the RPi microcontrollers and calculated TDoA values are accurate within 10 microseconds.**	For this acceptance test, the TDoA values from each microphone will be received and analysed. The ideal expected (calculated) TDoA will first be determined using the appropriate TDoA eqaution. The "measured" TDoA calculated from the parameters provided by the microphones will then be compared to the ideal expected TDoA for TDoA error. In addition to that, since two Pi's are used, a perfect synchronisation between the microphones is not possible. The aim is, however, to ensure as small a synchronisation delay as possible. The synchronisation delays will be timed and analysed for this ATP.	**Yes**		
17. **The system should be capable of**	The highest possible frequency of sound that humans can hear is 20	**Yes**		

capturing audio signals within the audible spectrum.	kHz. The system will thus be tested using chirp signals with less than 20 kHz frequency, and the time delay plots will be checked, and the estimated TDoA values will be compared to the calculated TDoA values.			
18. **The system should ensure that the sample rate of the microphones is 25kHz.**	To test this, the TDoA plots can be checked for aliasing due to a low sample rate. The "raspberrypi.sh" file could also be checked, as that is where the sampling rate of the microphone is set	**Yes**	**The system should ensure that the sample rate of the microphones is greater than 40kHz.**	**The system should ensure that the sample rate of the microphones is 25kHz.**
19. **The system should ensure that SNR of the captured audio signals are greater than 50dB.**	For this acceptance test, the performance of the filter used to refine the audio signals will be used. Both the signal power and noise power of the input unfiltered signal, as well as the output filtered signal will be measured to calculate the input SNR and output SNR, which will then be compared for performance.	**Yes**	**The system should ensure that SNR of the captured audio signals are greater than 60dB.**	**The system should ensure that SNR of the captured audio signals are greater than 50dB.**
20. A Least Squares Estimation algorithm will be applied to the TDoA values acquired to determine the estimated coordinates of the sound source. **21. The system should provide**	In this acceptance test, a pair of coordinates will be expected from the triangulation algorithm that will use the TDoA information from the previous algorithm to output a position relative to the rectangular grid used. The coordinates provided by the algorithm will then be	**Yes**		

the location of the sound source within a 1cm accuracy.	compared to the actual position of the sound source to calculate position error.			
22. The GUI will be programmed using the PySimpleGUI library for Python.	The GUI to be used will be coded using the PySimpleGUI library, and the effectiveness of this choice will be tested by asking other engineering students on their opinion of the GUI.	**Yes**	**The GUI will be programmed using the Tkinter library for Python.**	**The GUI will be programmed using the PySimpleGUI library for Python.**

Acceptance tests procedures of Subsystems

Subsystem	**Acceptance Test**	**Test has been met?**	**Changed specifications**	
			Old Specification	**New Specification**
1.1 Pi Communication	To test this specification, both the Raspberry Pi Microcontrollers will be connected to each other, and then a 'ping' command will be sent from the console of one of the Pi's to the IP address of the other Pi and vice versa.	**Yes**		
1.2 Pi Timing	For this test, the time in which both the Pi's logged in using SSH (Secure-Shell Host) will be recorded.	**Yes**		
2.1 Signal Capture	For this acceptance test, the signal acquisition plots will be analysed to look for expected signal plots.	**Yes**	**The system should ensure that the sample rate of the microphones is greater than 40kHz.**	**The system should ensure that the sample rate of the microphones is 25kHz. The source and calibration audio signals will be generated using a pair of Bluetooth speakers.**

			The sound source will be generated using an Android smartphone.	
2.2 Signal Preprocessing	For this acceptance test, the performance of the filter used to refine the audio signals will be used. Both the signal power and noise power of the input unfiltered signal, as well as the output filtered signal will be measured to calculate the input SNR and output SNR, which will then be compared for performance.	**Yes**	**The system should ensure that SNR of the captured audio signals are greater than 60dB.**	**The system should ensure that SNR of the captured audio signals are greater than 50dB.**
3. Time Delay Estimation	For this acceptance test, the TDoA values from each microphone will be obtained using the GCC-PHAT (Generalised Cross Correlation Phase Transform) algorithm and analysed. The ideal expected (calculated) TDoA will first be determined using the appropriate TDoA eqaution. The "measured" TDoA calculated from the parameters provided by the microphones will then be compared to the ideal expected TDoA for TDoA error.	**Yes**		
4. Triangulation	In this acceptance test, a pair of coordinates will be expected from the triangulation algorithm that will use the TDoA information from the previous algorithm to	**Yes**		

	output a position relative to the rectangular grid used. The coordinates provided by the algorithm will then be compared to the actual position of the sound source to calculate position error.			
5.1 User Interface Functionality	Each input and output element will be tested for functionality and correctness. The inputs will be executed numerous times, and the output of these inputs will also be recorded, to determine whether the input achieves the desired output/function.	**Yes**		
5.2 User Interface Design	The user interface will be tested in its ergonomic aspect and ease of use. A volunteer, with no prior knowledge of the project, will be asked to use the program and comment/rate on how easy or difficult the interface is to use and understand.	**Yes**	**The GUI will be programmed using the Tkinter library for Python.**	**The GUI will be programmed using the PySimpleGUI library for Python.**

9.2 FUTURE PLAN

Based on the overall results and discussion the following steps have been outlined to improve the system for a practical setting:

- Advanced noise reduction techniques that can minimise the noise on the signals without causing significant distortion. For example, a higher performance filter could be used to increase the practical usability of the system.
- GPS based synchronisation to achieve precise synchronisation without the need for a calibration signal.
- A more advanced time delay estimation technique that is robust to correlated noise and reverberations in the signals.
- A more advance triangulation algorithm that can detect biases in the received TDoA values and correct for them to provide a more accurate position estimate without requiring perfect positioning of the receivers or Mics.
- A more functional and user-friendly graphical user interface (GUI) could be developed for later implementations.

- The TDoA and triangulation algorithms could be modified to allow sound source detection across different types of terrain.

10 CONCLUSION

To summarize all the key findings:

- The overall system achieved a mean position error of 0.025 m.
- The physical implementation is sensitive to high levels of noise.
- The optimal Source Signal Frequency range, generating a chirp signal, was found to be 1 kHz to 5 kHz.
- The optimal Calibration Signal Frequency range, generating a chirp signal, was found to be 6 kHz to 10 kHz.
- These frequency ranges represent the maximum bandwidths of the source and calibration signals that were tested.
- The order of the Bandpass filter should be between 1 and 6 depending on the noise conditions.
- Whilst the errors in the TDoA and triangulation results may be a bit over the theoretical maximum limit, they were still able to produce an accurate localisation of the sound source.
- It is imperative that the speaker (sound source) and microphones synchronise the playback, otherwise it will lead to too much clipping of the sound source and may severely affect the triangulation and TDoA.

In conclusion, it can be said that the simulated implementation and physical implementation of the sound detection by triangulation was a success. During the development of the project, numerous lessons were learnt, and much insight was gained, as follows.

The lessons learned and insights gained:

- Communication between group members is vital to prevent issues such as clashes in git commits and possible overwriting of files, potentially nullifying changes made by other group members.
- Building on the point above, it is also very important to take note of commits made by group members so that one knows what changes have been made, and whether these changes are reflected after pulling the git repository.
- Testing should always be conducted in advance to allow enough time for debugging.
- Simulation is extremely useful in testing the effectiveness of algorithms with little cost involved.
- Simulation implementation will not directly map to the physical implementation.

11 BIBLIOGRAPHY

[1] Da Costa, D.G. (2022) Time Difference of Arrival Acoustic Triangulation Using a Distributed Sensor Network. Dissertation

[2] "RPi documentation," RPi. [Online]. Available: https://www.raspberrypi.com/documentation/

[3] "Adafruit I2S MEMS Microphone breakout," Adafruit Learning System, [Online]. Available: https://learn.adafruit.com/adafruit-i2s-mems-Microphone-breakout/overview

[4] B. Jin, X. Xu, and T. Zhang, "Robust time-difference-of-arrival (TDoA) localization using weighted least squares with cone tangent plane constraint," Sensors (Basel, Switzerland), vol. 18, 03 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/3/778

[5] D. Dalskov and S. K. Olesen, 'Locating acoustic sources with multilateration', Master's, 2014. [Online]. Available: https://vbn.aau.dk/ws/files/198526294/Measurements/Source%20signals

[6] M. Pollefeys and D. Nister, 'Direct computation of sound and Microphone locations from time-difference-of-arrival data', in 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, 2008, pp. 2445–2448. [Online]. Available: https://cvg-pub.inf.ethz.ch/WebBIB/papers/2008/001_PollefeysICASSP08.pdf

[7] T. Ho et al., "Acoustic Source Localization," Georgia Institute of Technology. [Online]. Available: https://eceseniordesign2022spring.ece.gatech.edu/sd22p11/finalwrittenreport.pdf

[8] N. R. Kumarasiri, Development of novel algorithms for localization in wireless sensor networks. The University of Toledo, 2014.

[9] "I2S Output Digital Microphone Datasheet", SPH0645LM4H-B, Rev. B, Knowles, 2015. [Online]. Available: https://cdn-shop.adafruit.com/product-files/3421/i2S+Datasheet.PDF

[10] "PySimpleGUI," www.pysimplegui.org. https://www.pysimplegui.org/en/latest/