

EEE3097S 2023

Milestone 2: First Progress Report

Group 2

15 September 2023



Aimee Simons (SMNAIM002)

Tilal Mukhtar (MKHTIL001)

Md Shaihan Islam (ISLMDS002)

1 TABLE OF CONTENTS

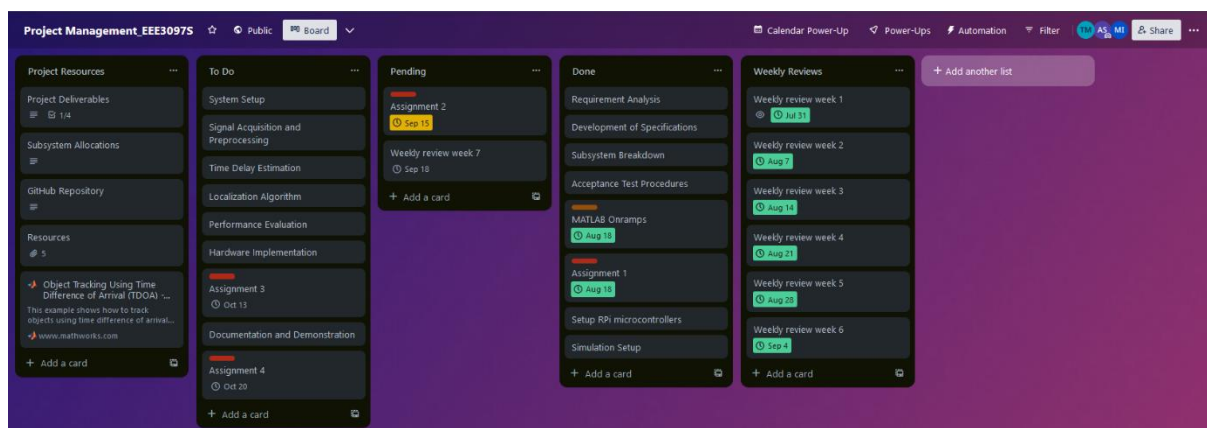
2	Admin Documents	3
2.1	Contributions.....	3
2.2	Project Management Page.....	3
2.3	GitHub Repository	3
2.4	Simulation Demonstration	3
2.5	Project Timeline.....	4
3	Simulation Setup.....	6
3.1	Simulation Environment	6
3.2	Simulation Approach	6
3.3	Simulation Simplifications or Assumptions	7
4	System Design and Implementation	8
4.1	Simulation Parameters.....	8
4.2	Signal Acquisition.....	8
4.3	Synchronization	9
4.4	Time Delay Estimation	9
4.5	Triangulation	10
4.6	Graphical User Interface	10
5	Simulation Results and Analysis.....	15
5.1	Overall System.....	15
5.1.1	Default Parameters	15
5.1.2	Varying SNR.....	16
5.1.3	Varying Cutoff Frequency of Lowpass Filter	17
5.1.4	Varying Latency	17
5.1.5	Varying Calibration Signal Position Error	18
5.1.6	Varying Microphone Position Error	18
5.1.7	Varying Source Signal Frequency	19
5.2	Signal Acquisition.....	20
5.3	Synchronization	23
5.4	Time Delay Estimation	24
5.5	Triangulation	25
6	Evaluation.....	27
6.1	ATPS.....	27
6.2	Next Steps in the Project (transitioning from simulation to physical implementation).....	30
7	Conclusion	31
8	Bibliography.....	32

2 ADMIN DOCUMENTS

2.1 CONTRIBUTIONS

Student Name	Student Number	Contributions
Md Shaihan Islam	ISLMDS002	Simulation Results and Analysis Evaluation (ATPs) Conclusion Simulation GUI code
Tilal Mukhtar	MKHTIL001	System Setup System Design and Implementation Simulation Results and Analysis Simulation code Simulation test code
Aimee Simons	SMNAIM002	System Setup System Design and Implementation Simulation Results and Analysis Simulation code Simulation GUI code

2.2 PROJECT MANAGEMENT PAGE



2.3 GITHUB REPOSITORY

Link to GitHub Repository:

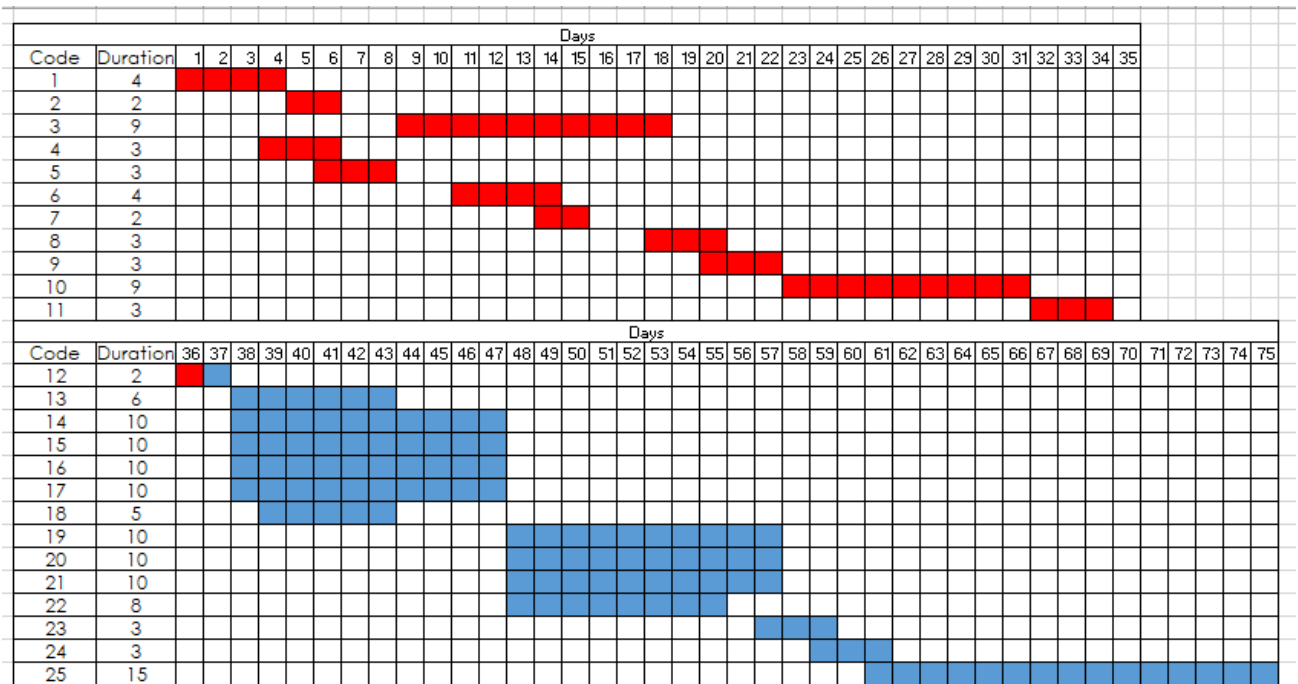
https://github.com/tshaihan/EEE3097S_Group2

2.4 SIMULATION DEMONSTRATION

Link to Simulation Directory:

https://github.com/tshaihan/EEE3097S_Group2/tree/main/Simulation

2.5 PROJECT TIMELINE



From the Gantt chart above, it can be seen that the project development is behind schedule by three days. This is because it was found that one of the Raspberry Pis, when configured for I2S communication with the microphones, was unable to record any sound. The suspicion of the group and the suspicion of the tutors is that the PCM Clock is faulty, but more debugging is being done to determine the fault. If thorough debugging has been done and the raspberry pi is still not working, a plan will be made to get it replaced.

The WBS for the remaining tasks is shown below:

Code	Activity	Duration	Dependency
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 synchronization	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

3 SIMULATION SETUP

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 Graphical User Interface (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.

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 Synchronization, Time Delay Estimation and Triangulation subsystems. The overall system consists of the two Raspberry Pi Zero W modules, four Adafruit I2S MEMS microphone breakout boards, (0.8x0.5 m) grid and the host device. The host device refers to the device used by the user to interact with system. The system also consists of two audio signal transmitters for transmitting the calibration signal and the signal of interest (source signal).

The following factors were considered when designing the simulation:

- The size of the grid
- The positions of the calibration signal and mics
- The error in the positions of the calibration signal and mics
- The speed of sound
- The sample rate of the mics
- The Signal to Noise Ratio of the signals produced by the mics
- The type of noise on the signals produced by the mics
- The maximum frequency of the calibration and source chirp signals
- The cutoff frequency of the lowpass filter applied to the signals produced by the mics
- the latency difference in communication to the two Raspberry Pi microcontrollers
- the type of synchronization used to synchronize 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.

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, desynchronization, and additive white Gaussian noise on the signals was considered.

The Synchronization subsystem was modelled as the post processing done on the signals to synchronize 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 synchronized 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.

This simulation approach was used because MATLAB provides a diverse set of toolkits, which make it possible to simulate such a system, which consists of time synchronisation, triangulation, cross-correlation and signal processing functions.

The creation of a GUI for the simulated system was also facilitated by tools provided by MATLAB. The use of a Graphical User Interface makes it easy for any user to interact with the system, without any prior knowledge of it.

Lastly, the simulation was tested for its performance using its default parameters as well as by varying individual parameters. The suitability of the default parameters and the overall performance of the simulation could thus be assessed.

3.3 SIMULATION SIMPLIFICATIONS OR ASSUMPTIONS

The following simplifications were made:

- A constant of 343 m/s was used for the speed of sound.
- The source signal was generated as a 5 second 0 to 100 Hz chirp signal with an amplitude of 1.
- The calibration signal was generated as a 5 second 0 to 1000 Hz chirp signal with an amplitude of 1.
- A sampling rate of 48 kHz was assumed for the mics.
- Errors in the clocks of the Raspberry Pi microcontrollers were not considered.
- It was assumed the Raspberry Pi microcontrollers would achieve synchronization within 10 ms using Network Time Protocol (NTP) synchronization [1].
- The Signal to Noise ratio (SNR) of the signals was kept at 65 dBW by default as rated by the mics [9].
- Additive white Gaussian noise was the only form of signal noise considered.
- The sound source was assumed to be stationary when emitting the signal.
- It was assumed that the host device is a computer running the Microsoft Windows operating system.
- It was assumed that all signal processing will be performed on the host device.
- The mics were assumed to be located on the corners of the grid.
- It was assumed that mics 1 and 2 would be connected the same Raspberry Pi module with mics 3 and 4 connected to the other.

4 SYSTEM DESIGN AND IMPLEMENTATION

4.1 SIMULATION PARAMETERS

The simulation has the following required parameters:

- x – the actual x-coordinate of the sound source (m)
- y – the actual y-coordinate of the sound source (m)

The simulation has the following optional parameters with default values:

- SNR – the Signal to Noise ratio of signals received from the mics
- Sample Rate – the sampling frequency of the signals received from the mics (Hz)
- Cutoff Frequency – the cutoff frequency of the lowpass filter applied to the signals received from the mics (Hz)
- Latency – the time delay difference in communication to the two Raspberry Pi microcontrollers (s)
- Calibration Position Error – the error factor of the position of calibration signal (m)
- Mic Position Error - the error factor of the positions of the mics (m)
- c – the speed of sound (m/s)
- Grid Size – the dimensions of the grid (m)
- Calibration Position – the position of the calibration signal (m)
- Mic Positions - the positions of the mics (m)

4.2 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 Raspberry Pi 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 synchronization of Network Time

Protocol (NTP) synchronization [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 Gaussian noise. This represents the expected noise on the mic signals. Only white Gaussian noise was considered due to technical limitations. The default SNR of signals is 65dBW 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 dBW. 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.

4.3 SYNCHRONIZATION

Signal synchronization 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 synchronized 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 synchronized signals was taken to display the results of this subsystem.

Additionally, the synchronization errors of the signals were also recorded for all the tests performed on the overall system. In one of the tests, the performance of the system was recorded for varying latency differences between the Raspberry Pi modules.

4.4 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 synchronized and the calibration signal has been removed.

For the simulation of this subsystem, the synchronized signals from the synchronization 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 error between the estimated TDOA values and the actuals TDOA values were also recorded for all the tests performed on the overall system.

4.5 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 it 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.

4.6 GRAPHICAL USER INTERFACE

To bring all these subsystems together, a GUI was used to perform a 'visual' simulation:

The GUI consisted of an X-Y axis, which represented the A1 grid provided for the project, a legend explaining the representation of different points on the axis, three buttons, labels to display the predicted position and respective errors and various numerical edit fields.

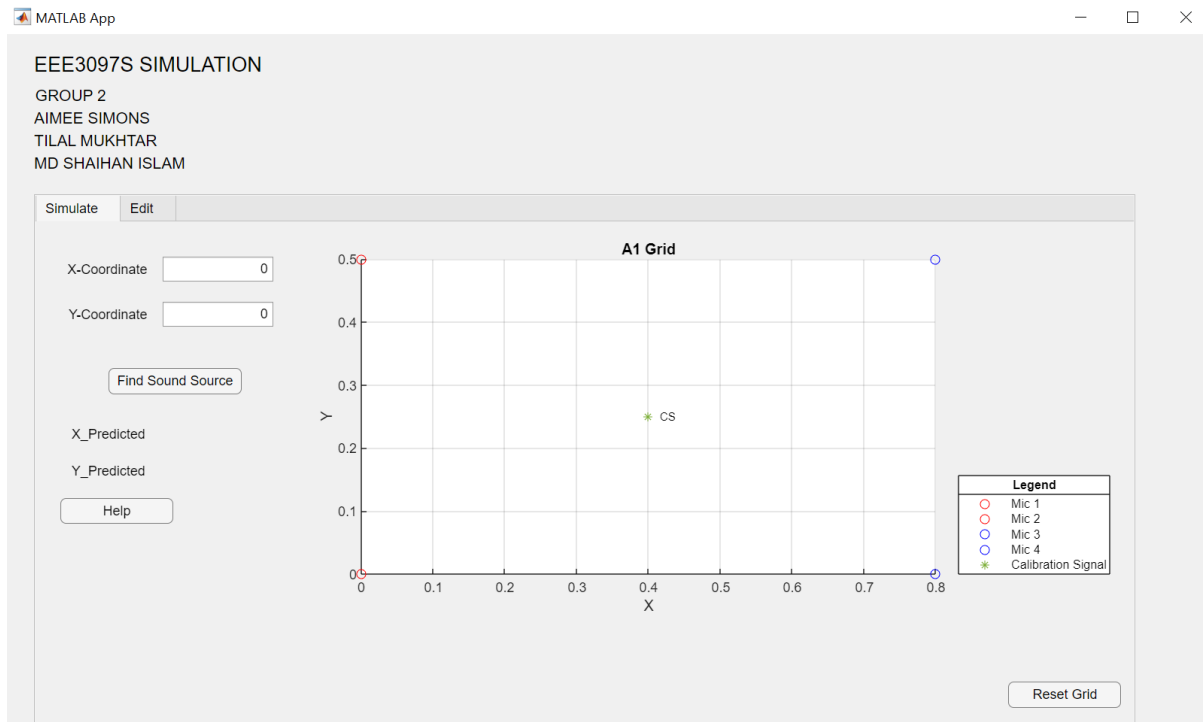


Figure 1: Simulation GUI

The positions of the microphones were created as constants and plotted on the set of axes in the startupFn. Microphones of the same colour are connected to the same Raspberry Pi.

```

properties (Access = private)
    m1_x = 0.0
    m1_y = 0.0

    m2_x = 0.0
    m2_y = 0.5

    m3_x = 0.8
    m3_y = 0.0

    m4_x = 0.8
    m4_y = 0.5
end

% Code that executes after component creation
function startupFcn(app)
    Initialise_Mics(app);
    plot(app.UIAxes,0.5,0.5,"*")
    hold(app.UIAxes,"on")
    text(app.UIAxes,0.52,0.5,'CS','fontsize' , 10)
    hold(app.UIAxes,"on");
    lgd = legend(app.UIAxes,{'Mic 1', 'Mic 2', 'Mic 3', 'Mic 4', 'Calibration Signal'}, 'Location', 'southeastoutside');
    title(lgd, "Legend")
end

methods (Access = private)

function Initialise_Mics(app)

    plot(app.UIAxes,app.m1_x*1.25,app.m1_y*2, 'o', 'Color','r')
    hold(app.UIAxes, 'on');

    plot(app.UIAxes,app.m2_x*1.25,app.m2_y*2, 'o', 'Color','r')
    hold(app.UIAxes, 'on');

    plot(app.UIAxes,app.m3_x*1.25,app.m3_y*2, 'o', 'Color','b')
    hold(app.UIAxes, 'on');

    plot(app.UIAxes,app.m4_x*1.25,app.m4_y*2, 'o', 'Color','b')
    hold(app.UIAxes, 'on');

end
end

```

Figure 2: Initialising mics' and Calibration Signal positions

The Calibration Signal is also plotted at runtime, denoted as 'CS'. The user then inputs an x-coordinate and a y-coordinate, which represents the represents the actual position of the acoustic sound source, and presses the button labelled "Find Sound Source".

x coordinate

y coordinate

Find Sound Source

Figure 3: Set Sound Source Actual Position

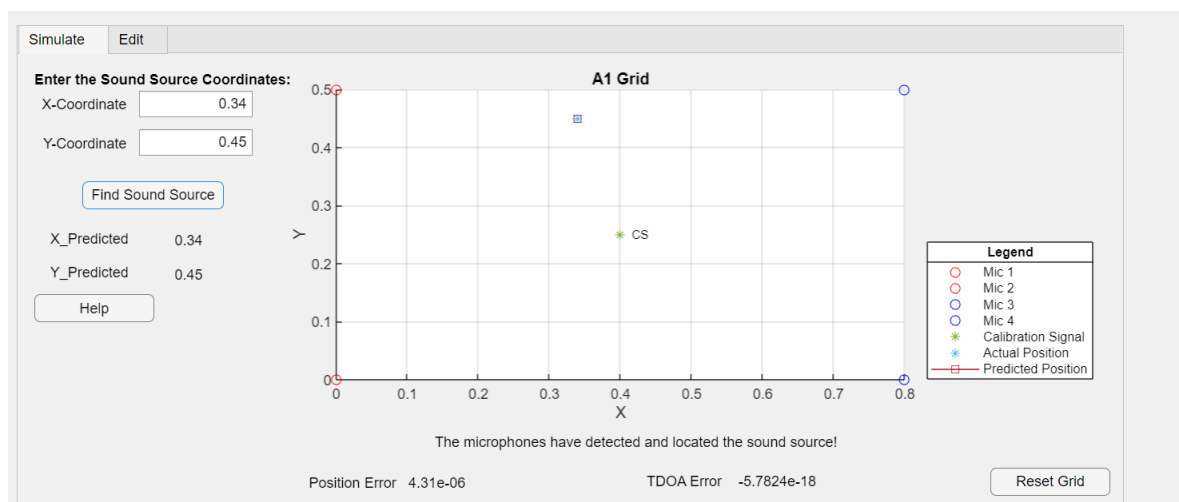


Figure 4: Showing the Actual and Predicted position on the grid.

This displays the Sound Source location on the grid as an asterisk and the predicted location as a square. The provided labels display the actual numerical values, the position error, and the TDOA error obtained when calling the MATLAB script.

The simulation also accounts for invalid coordinates provided, e.g, the coordinates are outside the bounds of the grid.

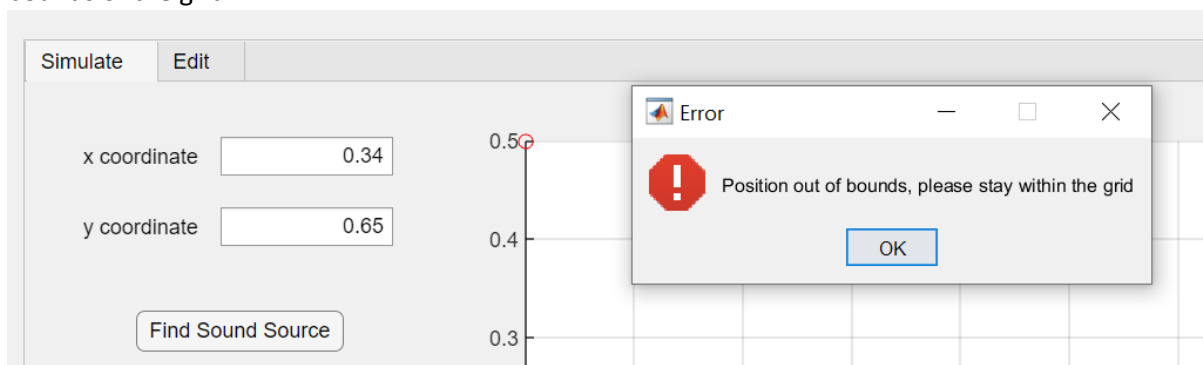
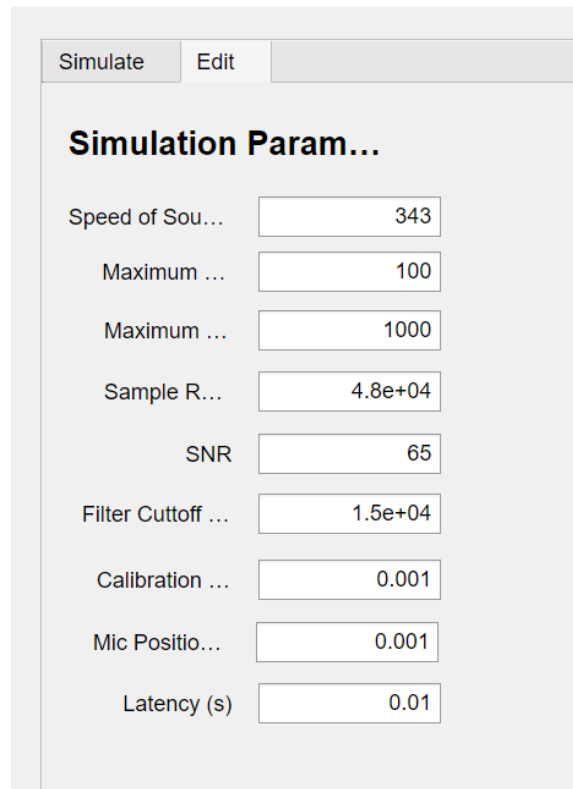


Figure 5: Error message when the coordinates are out of bounds.

Up until now, the simulation has been running using the default parameters chosen by group, as it was shown to produce the best outcomes. If different parameters wished to be used, the 'Edit' tab may be selected and the user may input their desired parameters.



Simulation Param...	
Speed of Sou...	343
Maximum ...	100
Maximum ...	1000
Sample R...	4.8e+04
SNR	65
Filter Cutoff ...	1.5e+04
Calibration ...	0.001
Mic Positio...	0.001
Latency (s)	0.01

Figure 6: Change to Desired Input Parameters

Using this 'Edit interface', experiments were run on the simulation varying certain parameters. The experiments run was, 'Varying the SNR', Varying the cutoff frequency of the lowpass filter', Varying the latency', 'Varying Calibration Signal Position Error', 'Varying Microphone Error' and 'Varying the Source Signal Frequency'.

Lastly, if faced with uncertainty with how to simulate, the 'Help' button may be pressed for more assistance. There is also a 'Reset Grid' Button, located on the far bottom, right-hand corner of the GUI, to clear the grid of all the predictions.

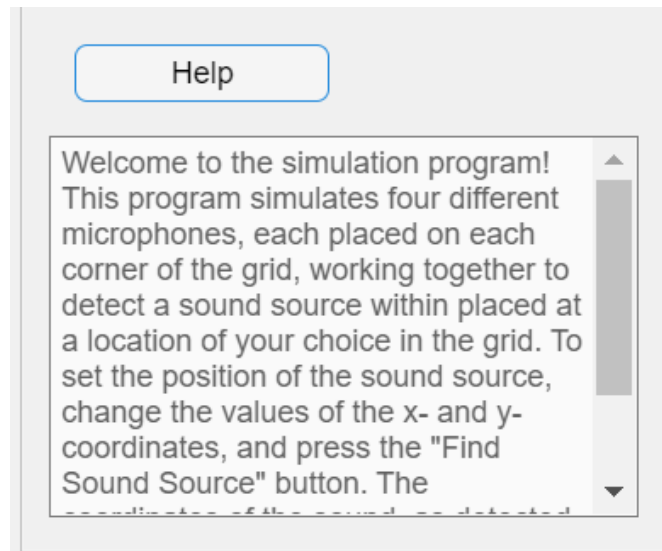


Figure 7: Help Button, to provide more assistance.

4.7 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.

5 SIMULATION RESULTS AND ANALYSIS

5.1 OVERALL SYSTEM

The performance of the system was tested using the *test.m* MATLAB function. The results were recorded in the *results.csv* file.

5.1.1 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:

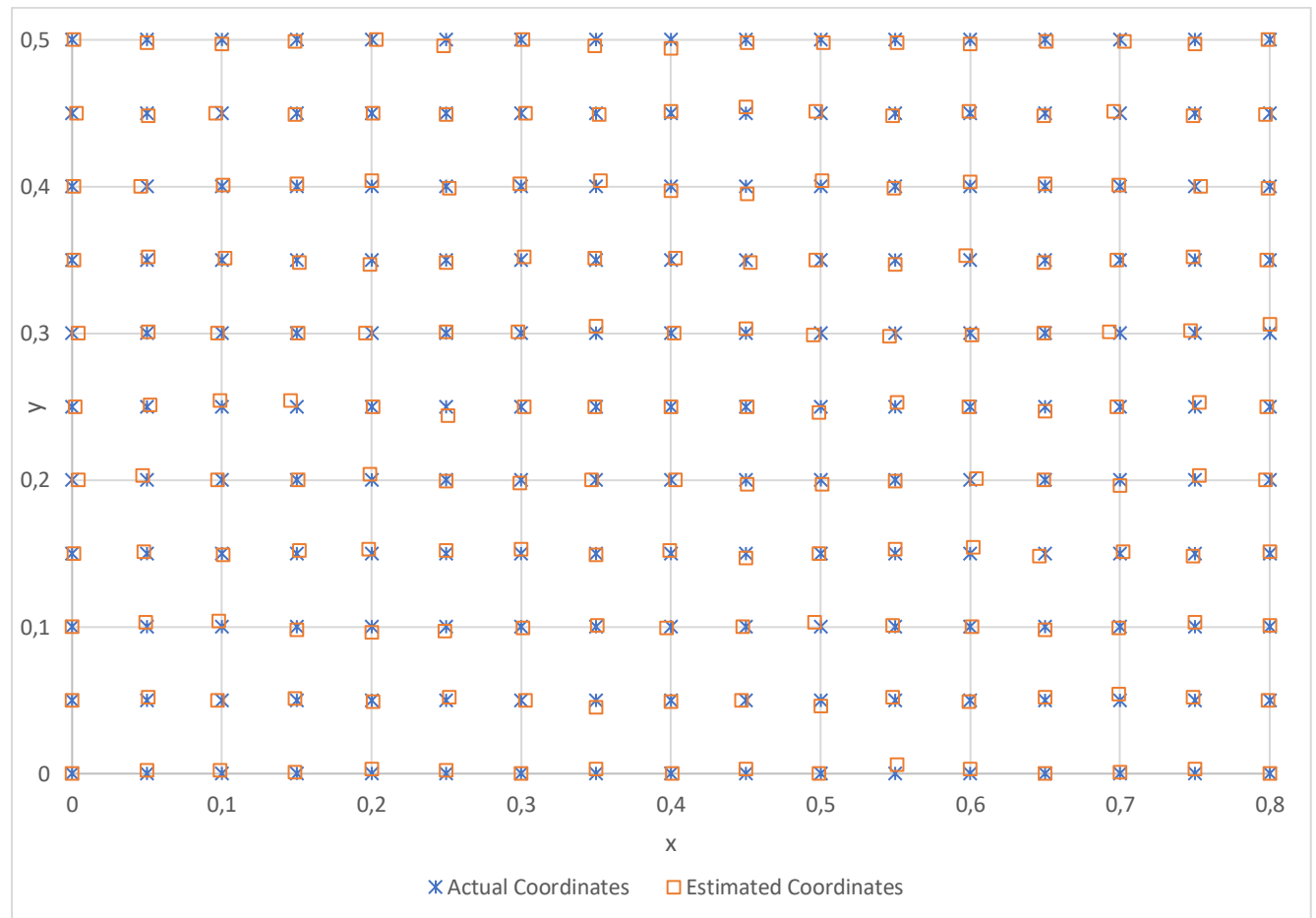


Figure 8: Graph of actual coordinates versus estimated coordinates.

It can be observed that Figure 8 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 1: Statistics of the position error

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.

5.1.2 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 dBW.

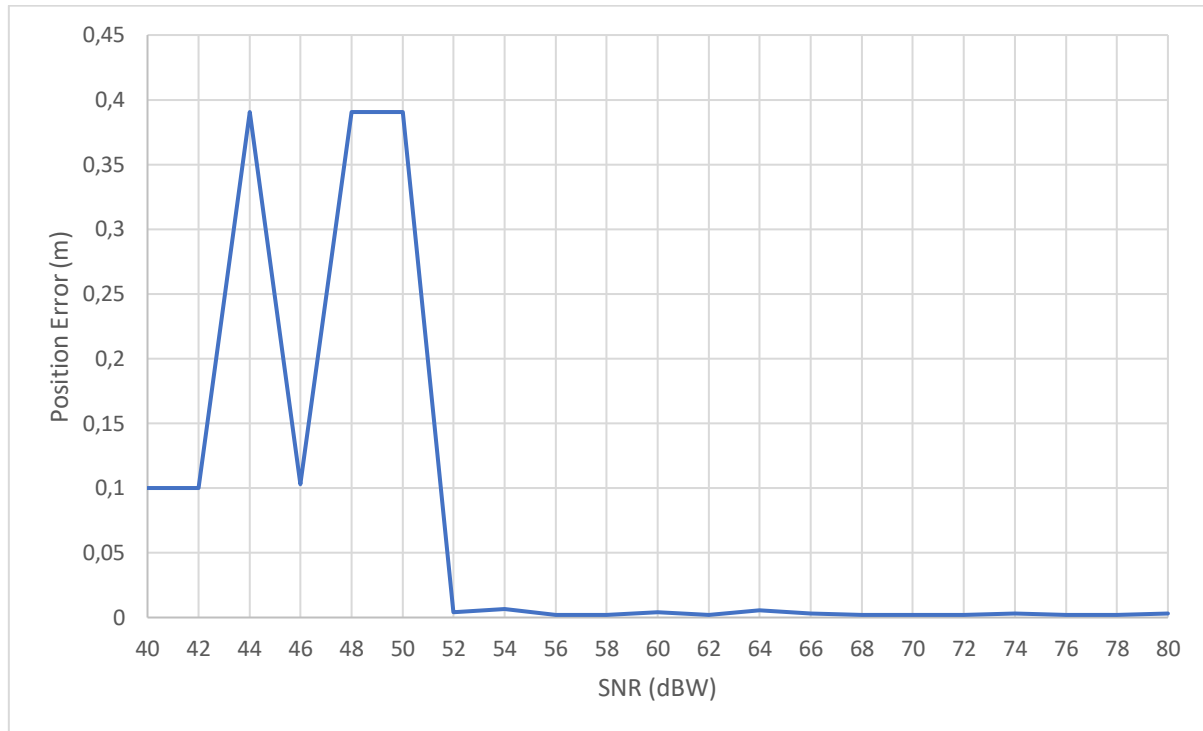


Figure 9: Graph of position error as a function of input SNR.

Figure 9 shows that the simulated system performs optimally for input SNR values greater than 52dBW. Moreover, the performance greatly deteriorates and becomes inconsistent for input SNR values less than 52dBW. Thus, for optimal performance environmental conditions should not decrease the input SNR below 52dB. This provides insight into the limited application of the system to different environments.

5.1.3 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 too 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.

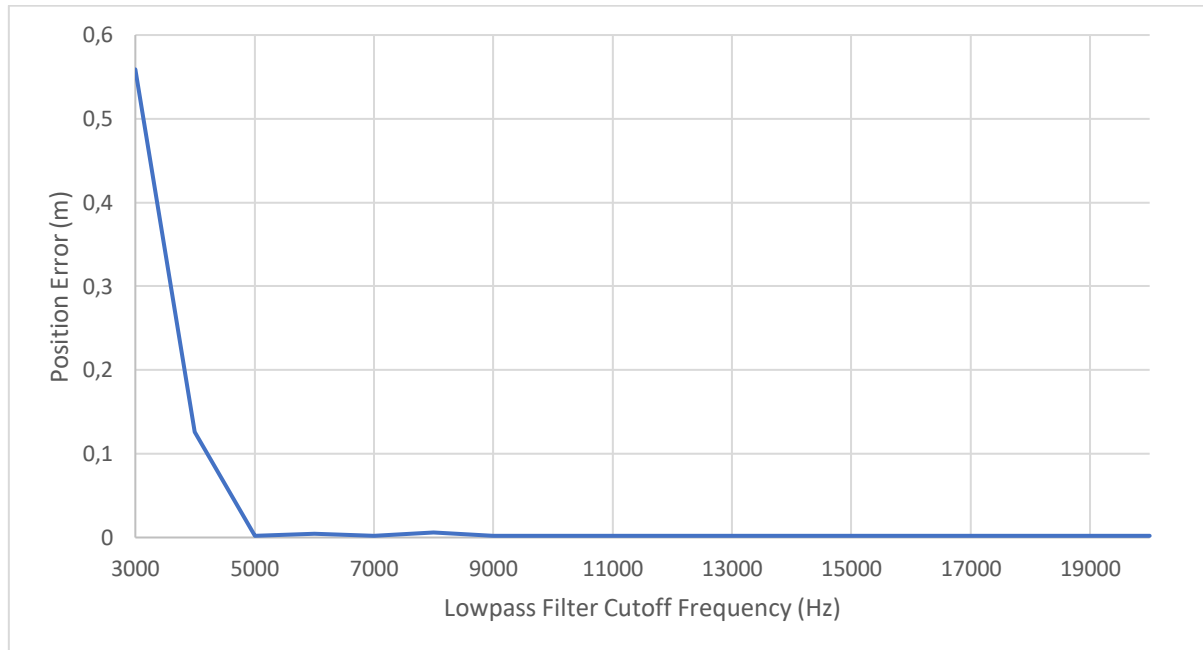


Figure 10: Graph of position error as a function of the lowpass filter cutoff frequency.

It can be observed that Figure 10 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 dBW. The tests will need to be repeated for a lower input SNR to further determine the ideal cutoff frequency of the lowpass filter.

5.1.4 Varying Latency

The latency difference is a measure of the desynchronization between the two Raspberry Pi modules. The synchronization error was defined as the average effect of desynchronization on the TDOA values. The system performance was tested for latency differences between the Raspberry Pi modules from 0.001 to 1.024 seconds. The results are tabulated below:

Latency Difference (s)	Synchronization 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 2: Synchronization Error

Table 2 shows that the synchronization error of the system is in the order of microseconds. This is ideal as an 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 desynchronization on the system is minimal with the other parameters kept default.

5.1.5 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:

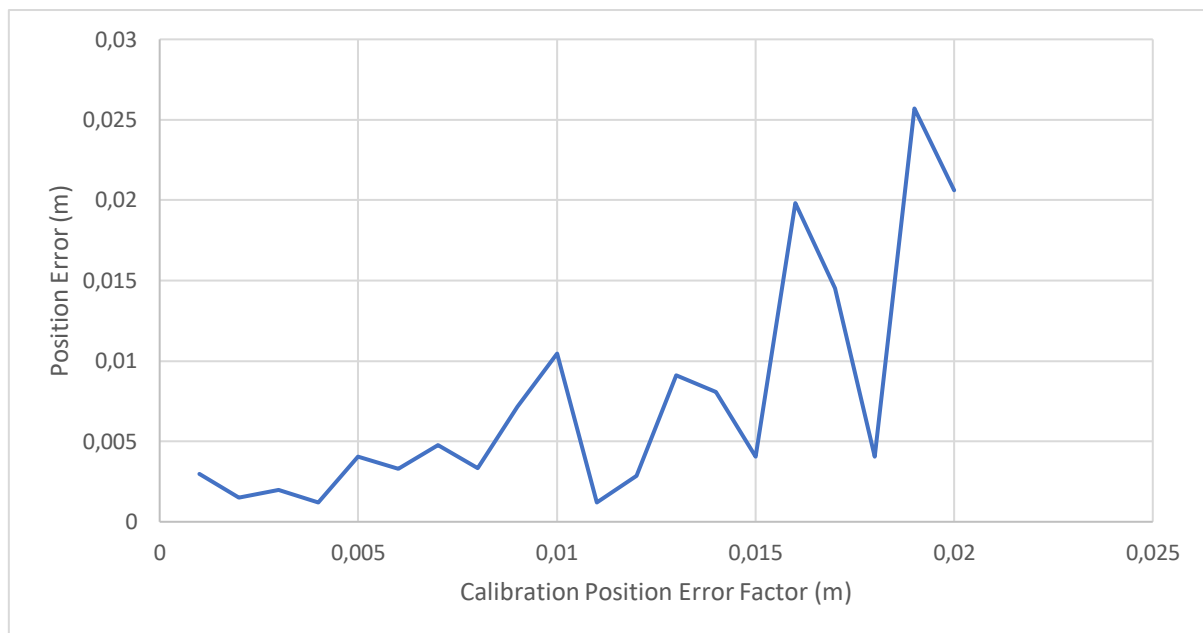


Figure 11: Graph of position error as a function of the calibration signal position error factor.

Figure 11 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 synchronization. An increase in the synchronization 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.

5.1.6 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:

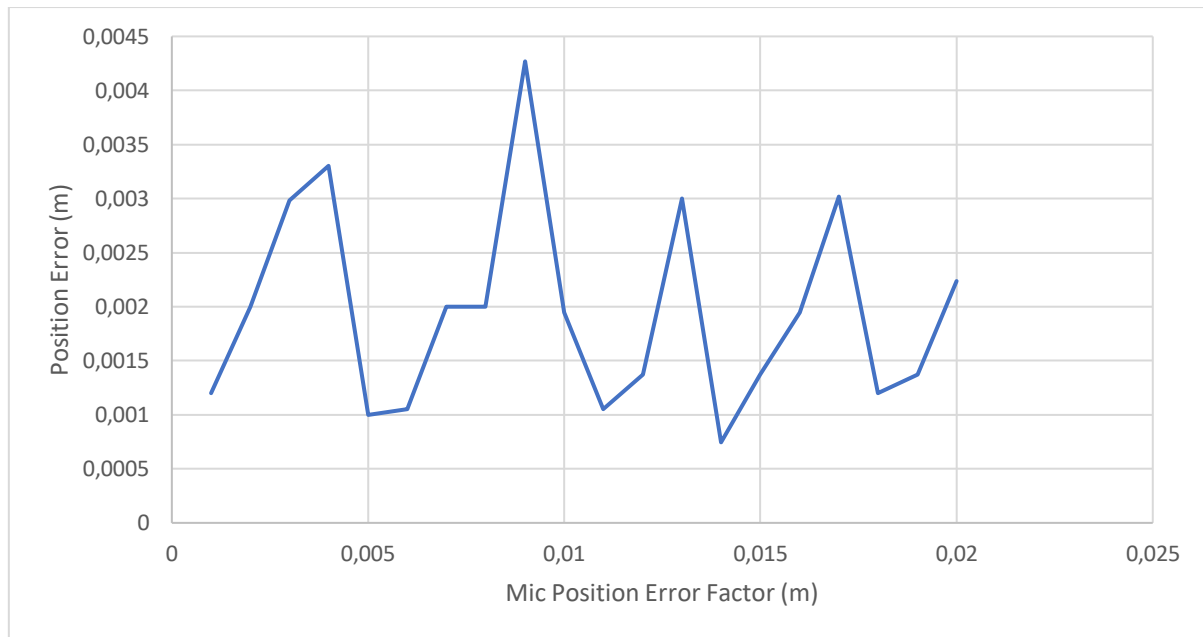


Figure 12: Graph of position error as a function of the mic position error factor.

From the graph above, it can be seen that varying the position error factor of the microphones, causes erratic and unpredictable position error. 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.

5.1.7 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.

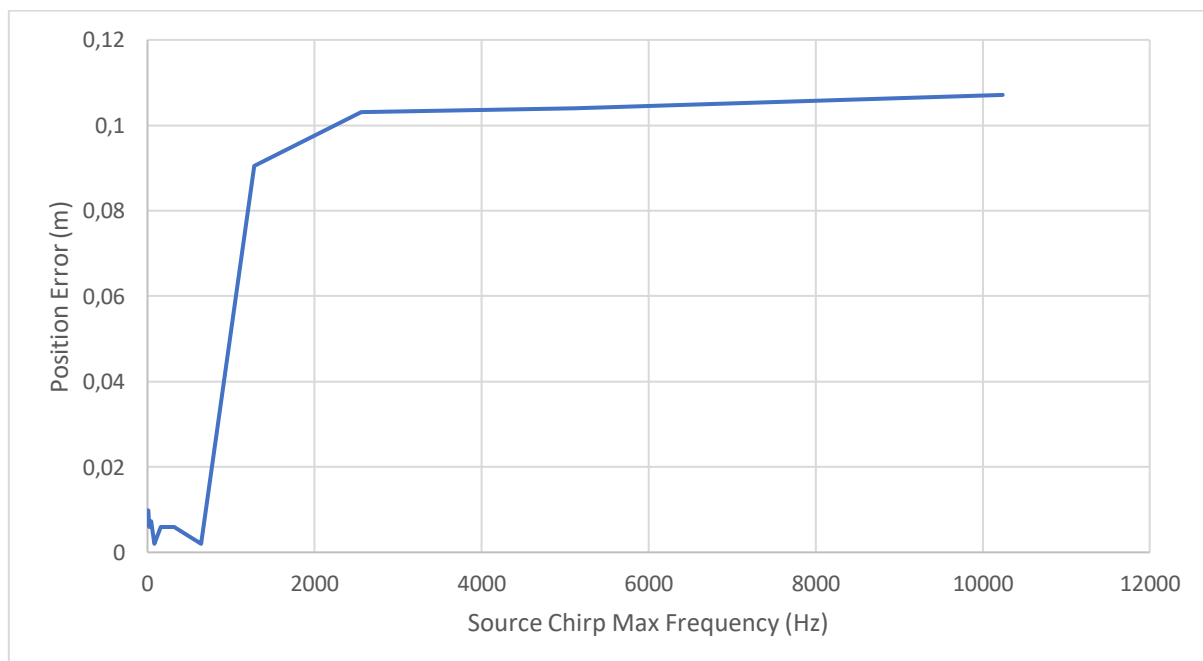


Figure 13: Graph of the position error as a function of the maximum frequency of the source chirp signal.

Graph 13 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.

5.2 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 dBW 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 desynchronization more visible.

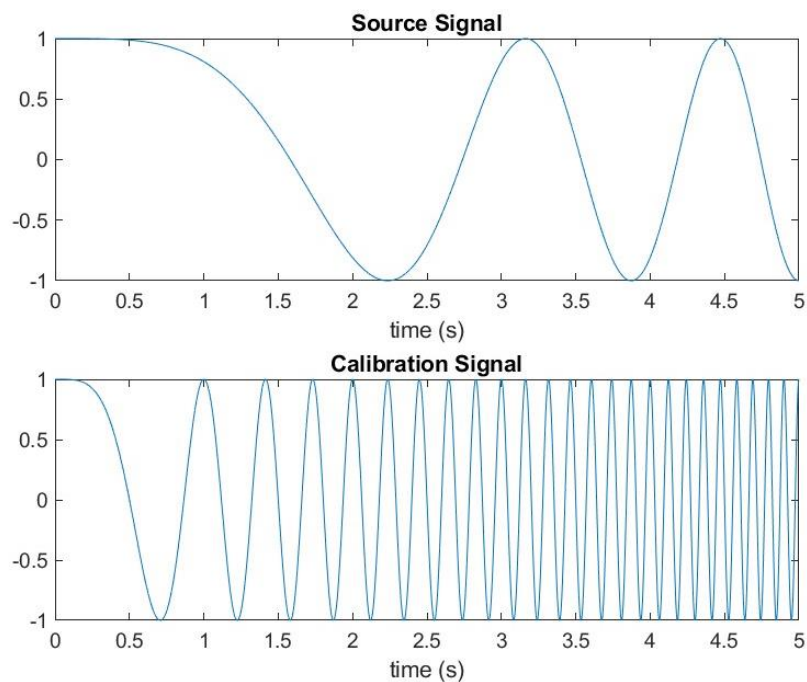


Figure 14: Plots of the source and calibration signals.

Figure 14 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.

Microphone Recordings

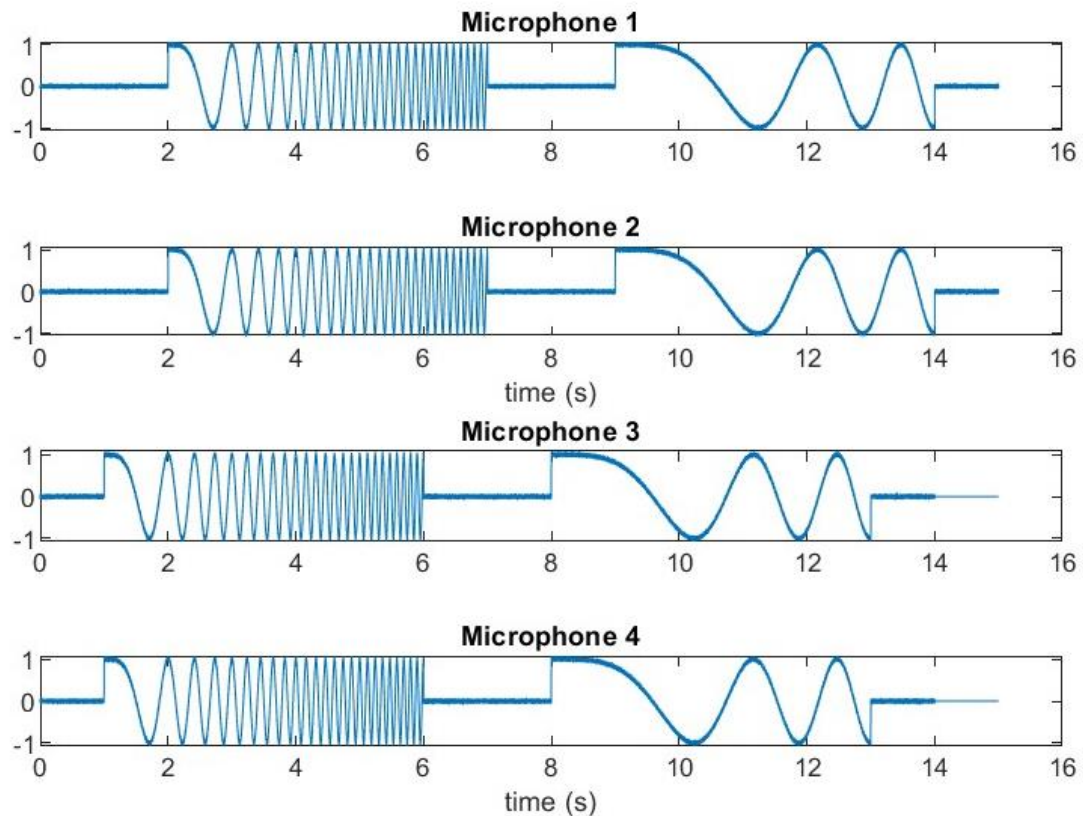


Figure 15: Plots of the microphone recording signals generated.

Figure 15 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 desynchronization 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.

Microphone Recordings After Lowpass Filter Applied

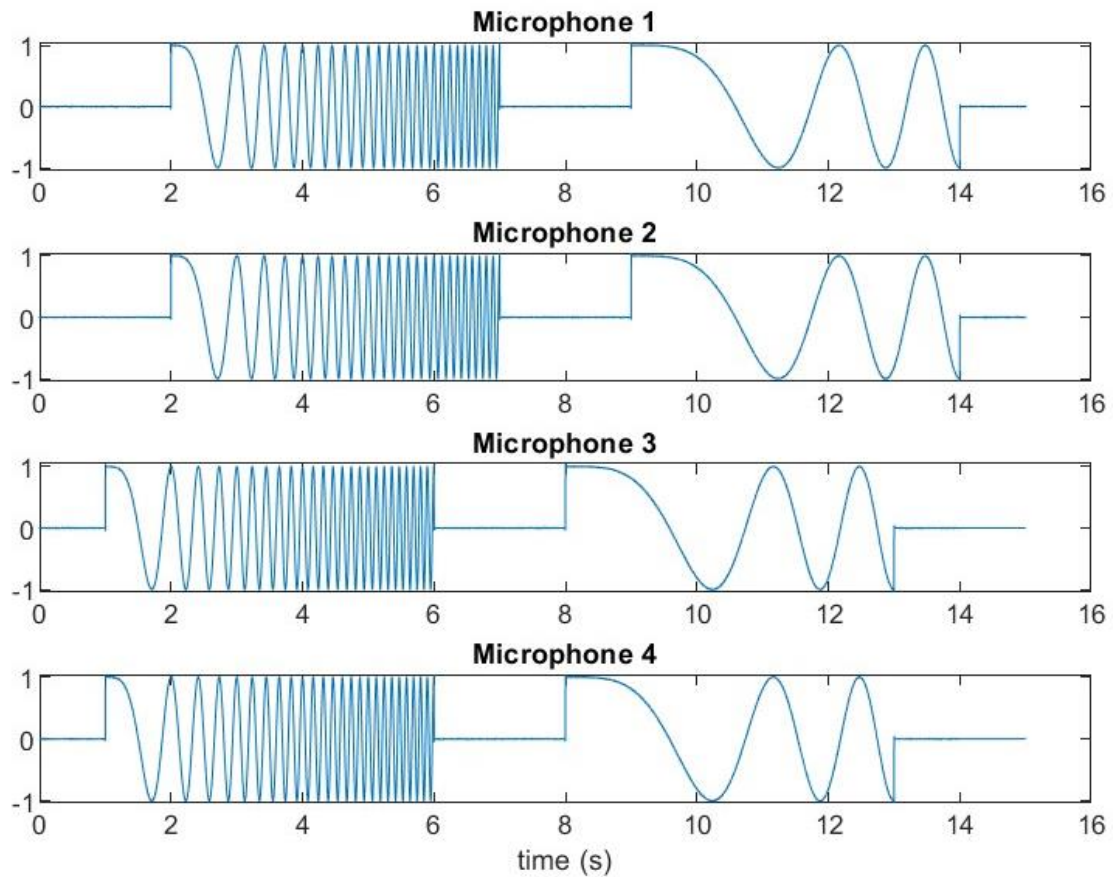


Figure 16: Plots of the microphone recording signals after the lowpass filter has been applied.

Figure 16 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.

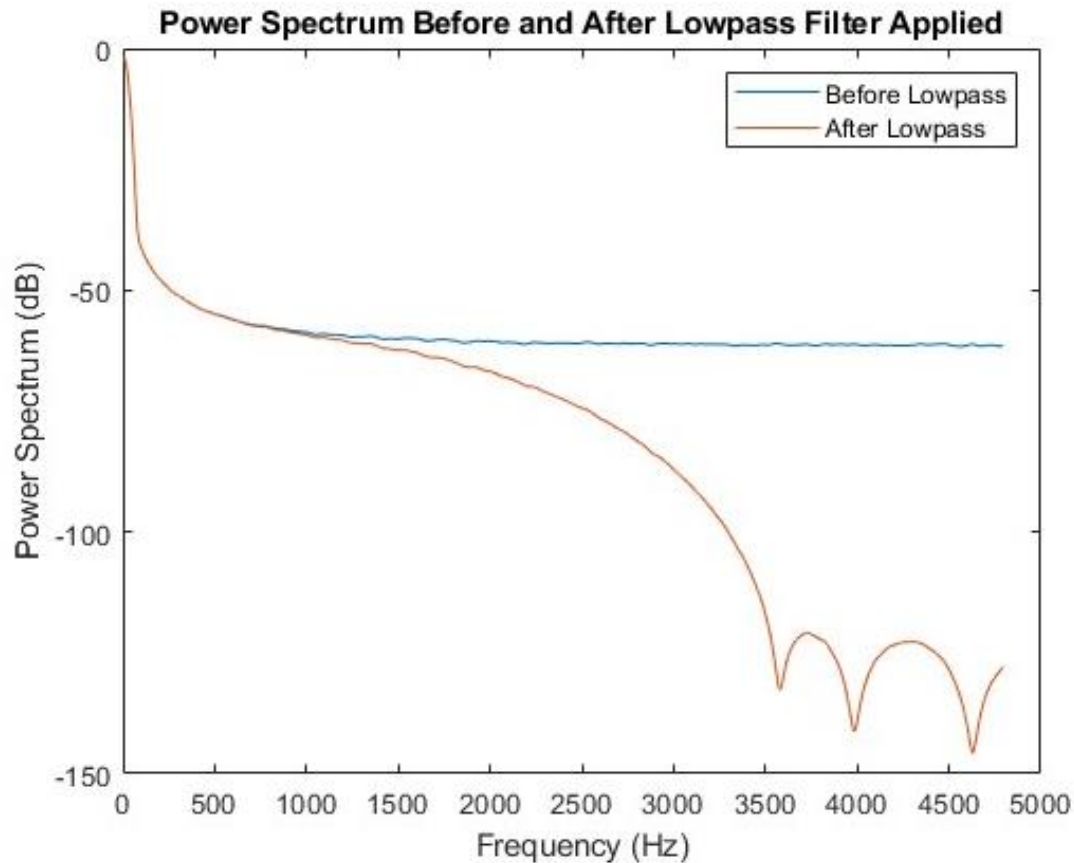


Figure 17: Plot of the power spectrum of a microphone signal before and after it has been passed through a lowpass filter.

Figure 17 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 100dBW.

5.3 SYNCHRONIZATION

The performance of the synchronization 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 desynchronization more visible. The input SNR was kept at its default 65 dBW. The position of the source was at the centre of the grid (0.4, 0.25).

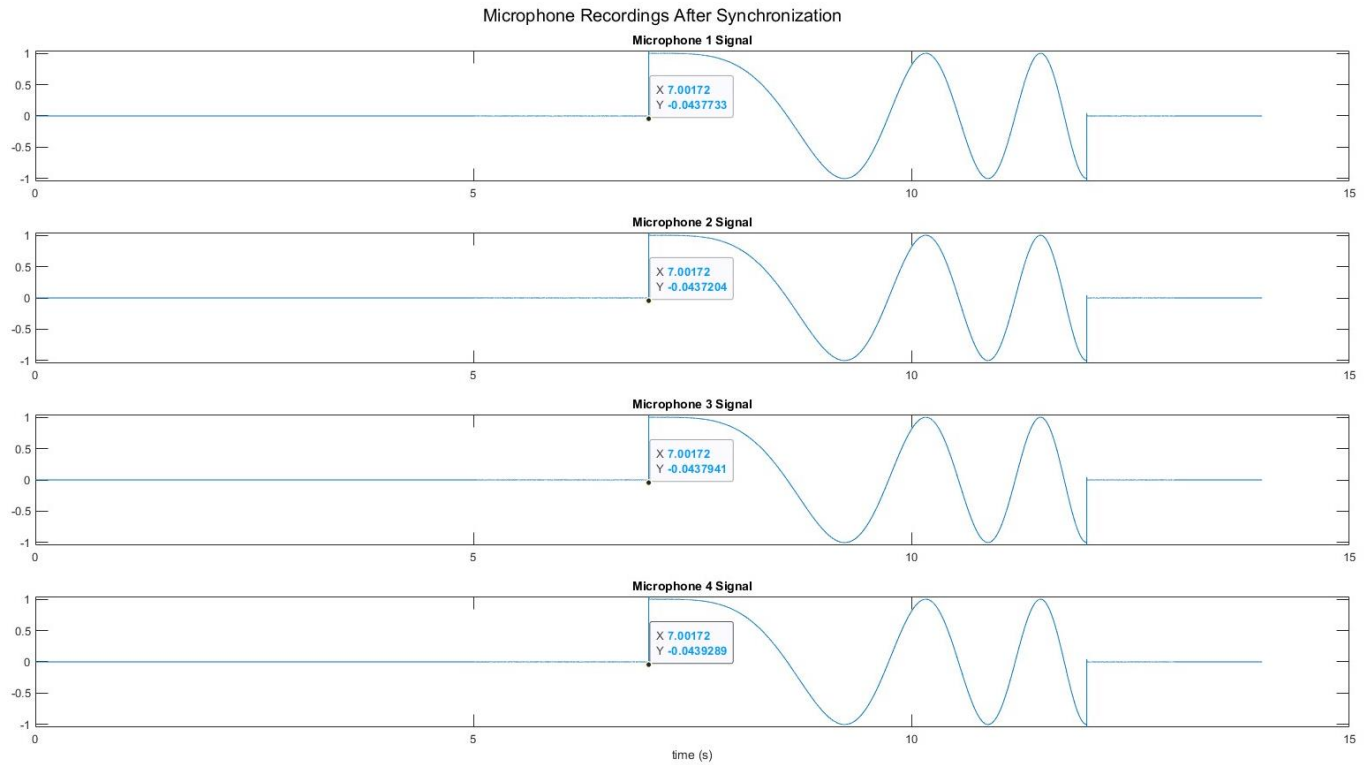


Figure 18: Plots of the microphone recording signals after synchronization.

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 18 shows that the correct synchronization 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.

5.4 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).

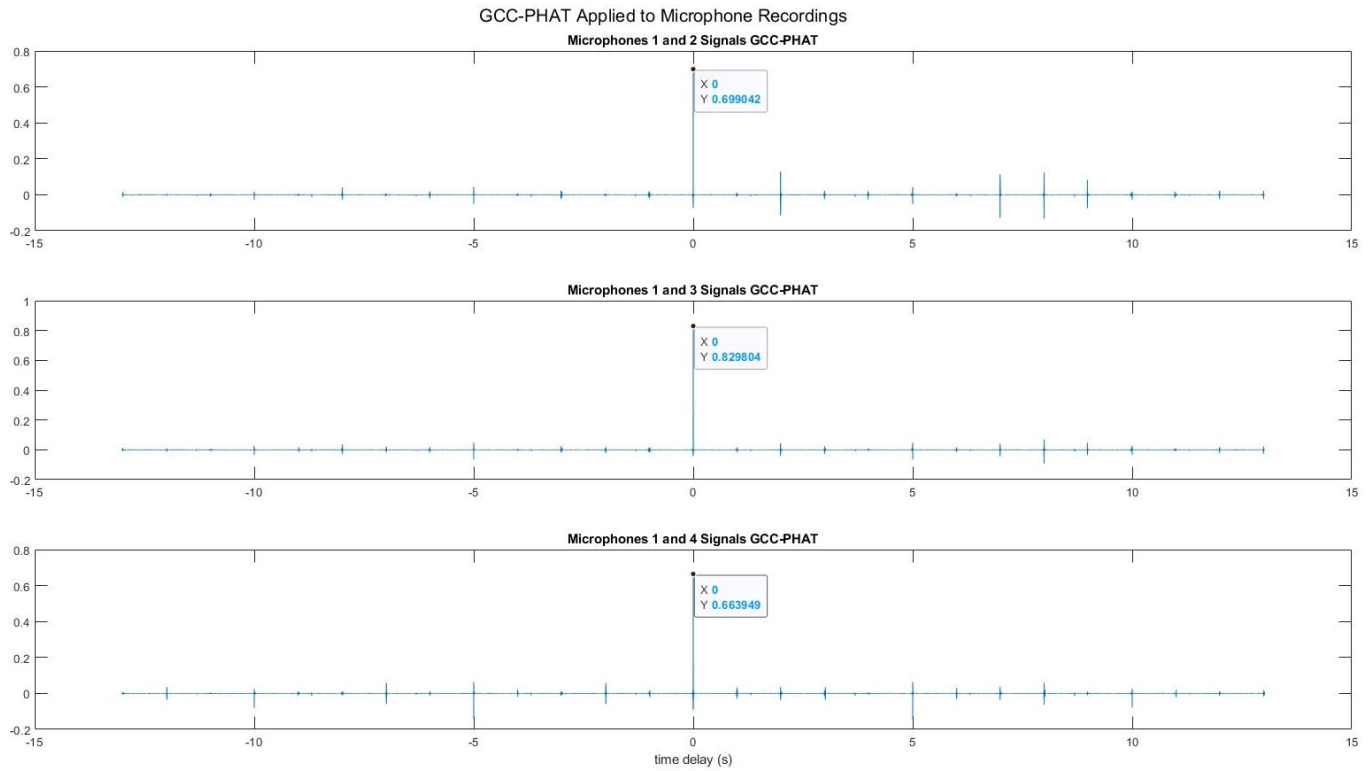


Figure 19: Plots of the GCC-PHAT cross correlation of the microphone 1 signal with each of the signals from the other microphones.

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.

5.5 TRIANGULATION

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.

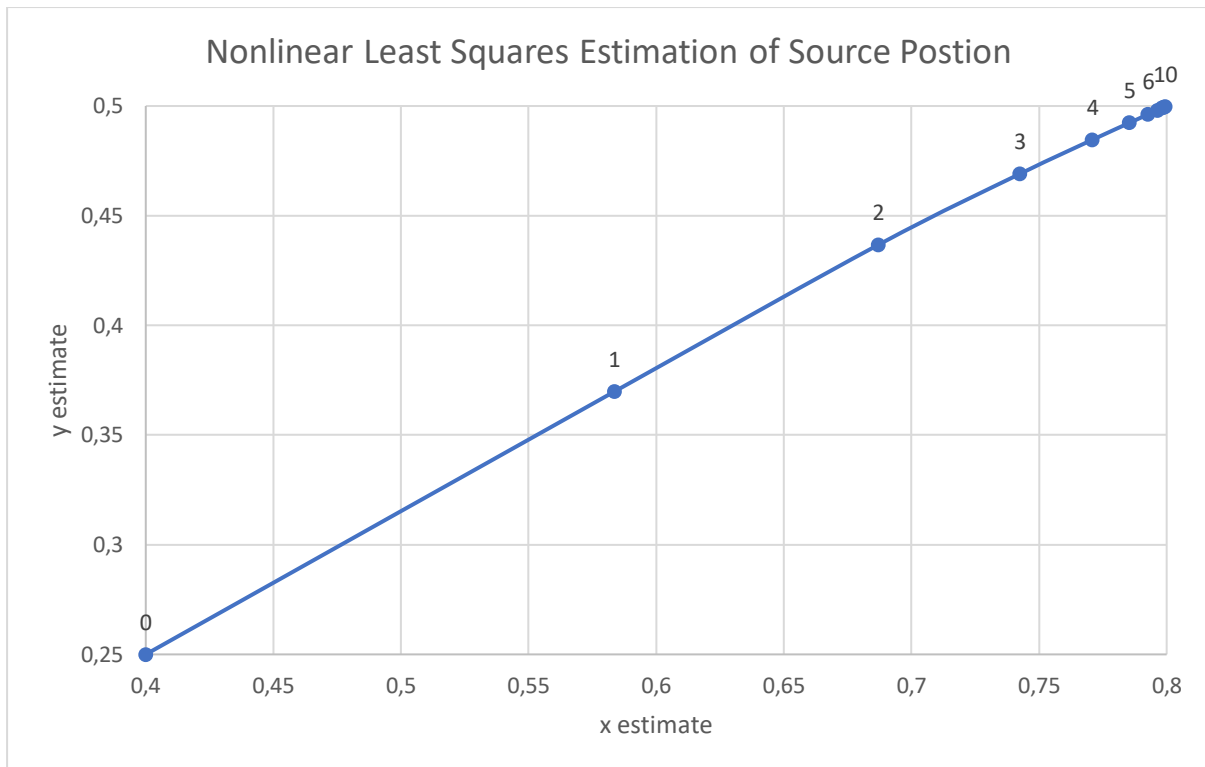


Figure 20: Graph of the coordinates estimated by the nonlinear least squares estimation algorithm at each iteration.

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

Figure 20 and Table 3 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. Furthermore, the tolerance of the algorithm can be reduced to reduce the number of iterations. However, a decreased tolerance may also reduce its accuracy. As the source is assumed to be stationary, the triangulation algorithm is not required to be real time. Thus, the primary indicator of its performance is the accuracy of its estimation of the source location. Consequently, optimizations for speed are a secondary concern.

6 EVALUATION

6.1 ATPS

For this simulation, the following ATPs were tested:

ATP 2.1: Signal Capture:

Acceptance test for signal capture: For signal capture, an acceptance test will be performed to determine whether the microphone was able to capture the sound being emitted from the source. Furthermore, a test can be conducted to ensure the microcontroller sends a response message once it has been able to capture the sound signal.

The response message to indicate that the microphones had detected a sound and located the sound source, was included in the GUI design, as can be seen in the image below:

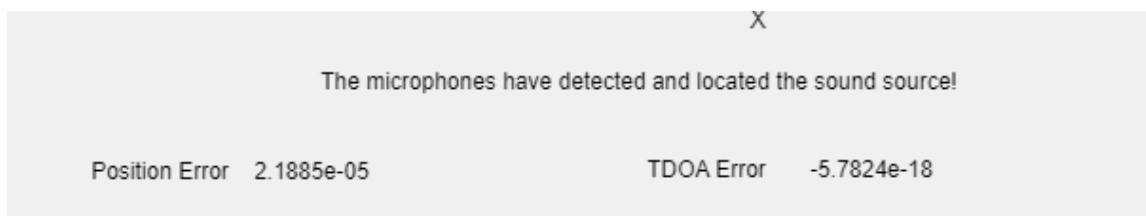


Figure 21: Message displayed after a successful sound source detection and location.

This message is received instantly after the sound source is located, hence passing the acceptance test above.

Acceptance Test: Passed

ATP 2.2: Signal Preprocessing

Acceptance test for Signal Preprocessing: This sub-subsystem requires the timestamps to be captured for the audio signals, and this will be tested using the Pi microcontrollers, which will be outputting these timestamps.

Tests will also be conducted to measure the signal-to-noise ratio (SNR). Information regarding the noise in the system will be taken from the filter that will be used to attenuate the noise from the signal, which can then be used to calculate the SNR.

From the tests above, it was seen that the system performs optimally for an input SNR of above 52 dB, which is an acceptable performance since it is below the microphone's threshold SNR rating of 65 dB. It is important to note that this simulation assumed ideal conditions, as environmental noise may cause the SNR to drop to below 52 dB. Given the above however, the simulation does behave as expected.

Acceptance Test: Passed

ATP 3: Time Delay Estimation

Acceptance test for Time Delay Estimation: Correctness of the time delay will be tested for. Information regarding the time delay will be computed using the host device, and the distance that was calculated using the Time-Difference-of-Arrival equations will be outputted.

The test script used to compute the results for the simulation had also computed data for the TDOA error for each point from which the sound source had to be detected. In order to produce an accurate TDOA estimate, the estimate must be accurate to 10 microseconds. This means the average TDOA error cannot exceed 10 microseconds. From all the points computed in section 5.1.1, the average TDOA error is:

6,41711E-08 seconds

This value is much less than the maximum threshold of 10 microseconds, hence this passes the acceptance test.

Acceptance Test: Passed

ATP 4: Triangulation

Acceptance test for Triangulation: The sound source will be placed at various positions within the grid, for example: close to a microphone, at the center of the grid, as well as outside the grid. The system will then be tested to see if it provides the appropriate coordinate information

In section 5.1.1, it can be seen that a number of test points were plotted on and estimated (detected) coordinates were compared with the input coordinates. The simulation grid has a length of 0.8 units and a width of 0.5 units. Scaling this with the dimensions of an actual A1 page, 0.1 units would represent 100mm, or 10 cm. The estimated coordinates were computed using a triangulation algorithm as explained in section 5.5. In order to test the accuracy of the triangulation algorithm, the test script also recorded the position error for each point detected. For the triangulation algorithm to accurately detect the location of the sound source, the point must be within 1 cm (0.01 units) of the input coordinates. From all the points computed in section 5.1.1, the average position error is:

0,002567 units

This value is much less than the maximum threshold of 0.01 units, hence this passes the acceptance test.

Acceptance Test: Passed

ATP 5.1: User Interface Functionality

Acceptance test for 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.

Each component in the graphical user interface (GUI) works as intended. While most aspects of the user interface are responsive, the 'Find Source Button' is considerably slow, taking between 5-7 seconds before the answer is displayed on the GUI. It is suspected that MATLAB processing time plays a factor in causing this delay, in addition to the signal acquisition and triangulation algorithms.

Acceptance Test: Passed

ATP 5.2: User Interface Design

Acceptance test for 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.

A fellow student in the Mechatronics stream, was asked to critique the interface design and these were her overall thoughts: The interface design is overall, really good, as it is easy to work with. She especially liked the display and that all the information is visible to the user. The only problem was, as she was someone who didn't know the scope of the project, she didn't know what to input into the x- and y- coordinate edit fields without pressing the help button. A label above the x- and y- coordinate edit fields was created to alleviate this confusion shown in Figure 21.

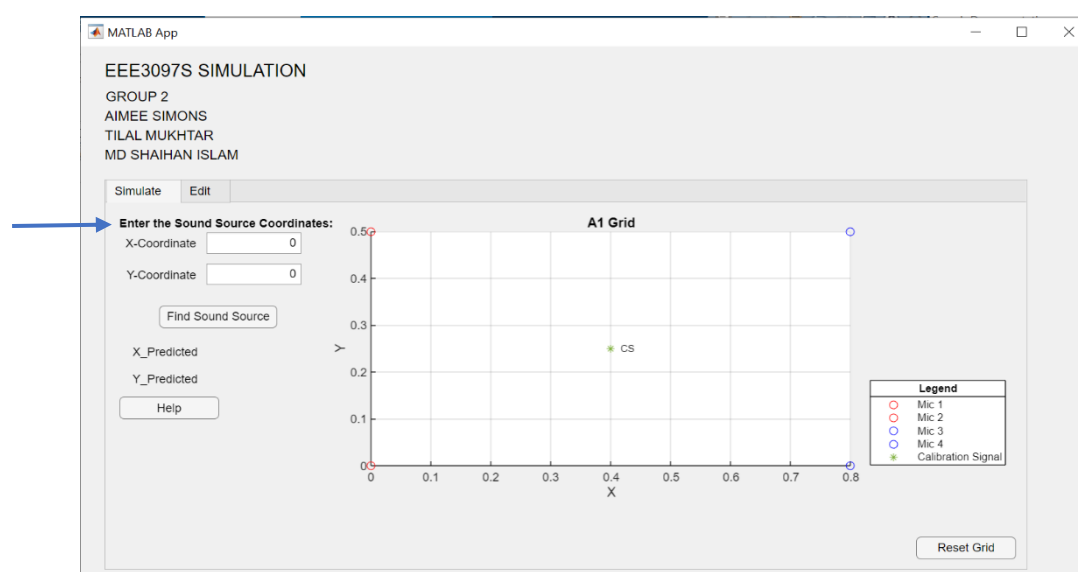


Figure 22: Improved GUI.

Acceptance Test: Passed

6.2 NEXT STEPS IN THE PROJECT (TRANSITIONING FROM SIMULATION TO PHYSICAL IMPLEMENTATION)

1. The hardware, meaning the 2 Raspberry Pis and the 4 microphones, will be connected to form the distributed sensor network described in this report.
2. Tests must be done to ensure that the Pis both record at the same time when provided with a script through parallel-ssh.
3. All code will be translated from MATLAB code into python code.
4. Tests will be done to ensure that the signals are pre-processed correctly, using the python code.
5. A GUI using Tkinter will be created to replace the simulation GUI.
6. All members will download the 'Tone Generator' App to generate the source signals required.

7 CONCLUSION

To summarize all the key findings:

- The Triangulation algorithm provided a very accurate estimation of the position of the sound source with a very low position error.
- The algorithm achieves the lowest position errors for SNRs greater than 52 dB.
- The cutoff frequency of the lowpass filter should be above 5kHz, as lower cutoff frequencies will cause higher position errors.
- Increasing in the synchronization error will result in an increase in the position error.
- The position error of the microphones should be kept to a minimum as it does not really affect the position error in a concise manner.
- Increasing the source frequency will cause an increase in position error.
- The position error increases with an increase in calibration signal position error.
- The low pass filter substantially reduces noise and distortion from the signal detected by the microphones.
- A chirp signal was used instead of other standard signals such as a sine wave because the chirp signal has a much better cross-correlation, which make it more ideal for time delay estimation.

In conclusion, it can be said that the simulation of the sound detection by triangulation was a success. The simulation took into account as many factors as possible that could play a role in the actual project, for example noise, microphone position error, etc. During the development of the simulation, numerous lessons were learnt and much insight was gained, as follow.

The lessons learned and insights gained:

- The concept of a chirp signal being introduced was new to some of the group members.
- Using MATLAB as a simulation platform proved to be very insightful, as the abundance of resources available was evident and made designing the simulation less complicated.
- 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.
- Information regarding the potential optimal conditions for the system performance was gathered through the simulation testing

8 BIBLIOGRAPHY

- [1] Da Costa, D.G. (2022) Time Difference of Arrival Acoustic Triangulation Using a Distributed Sensor Network. Dissertation
- [2] “Raspberry pi documentation,” Raspberry Pi. [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://ecesenioridesign2022spring.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>