MILESTONE THREE: 15 October 2023

ASSIGNMENT 4:

FINAL REPORT

EEE3097S 2023

JNKKET001 LWSHOL001 PTTMAT005

Table of Contents

1. Document Introduction

This report details the process of implementing the acoustic signal triangulation system. Each subsystem, and the overarching system, is described in independent sections. This highlights the modifications and improvements made based on the previous simulation results and practical considerations. This provides an overview into the implementation of the system as a whole with the tools and methods used. It also aims to provide a comparison between how the project was simulated and how it was physically implemented. This allows for a benchmark on which to judge the real implementation and judge its success.

The report begins by outlining how the project was managed. This includes explaining how the project management tool Notion was utilised, how it was organised, how tasks were tracked and how the work was divided amongst team members.

Secondly, the report explores the experimental design of the system. The basics and goals for the project are clarified in the requirements and specifications section. Additionally, the connection between the specifications and ATP's was made, thereby illustrating how the criteria for the success of the project was created.

The report then goes on to describe the project's paper design. This includes explaining how all the subsystem and working parts of the project should operate and interact with each other. This section is very important for understanding the principle design ideas of the project without going into how and what this is done with.

Our system design is validated using simulations. These simulations, such as grid emulation and sound triangulation, were primarily done using matlab. This section outlines the process and implementation followed in order to get the simulation results which can later be compared to the actual implementation results.

The same process as the simulation validation was done for the actual physical implementation. The results were discussed and analysed. Following this, it was possible to compare the two sets of results and judge the effectiveness of the physical implementation. Furthermore, the ATP's were evaluated and a conclusion was drawn about the success of the project.

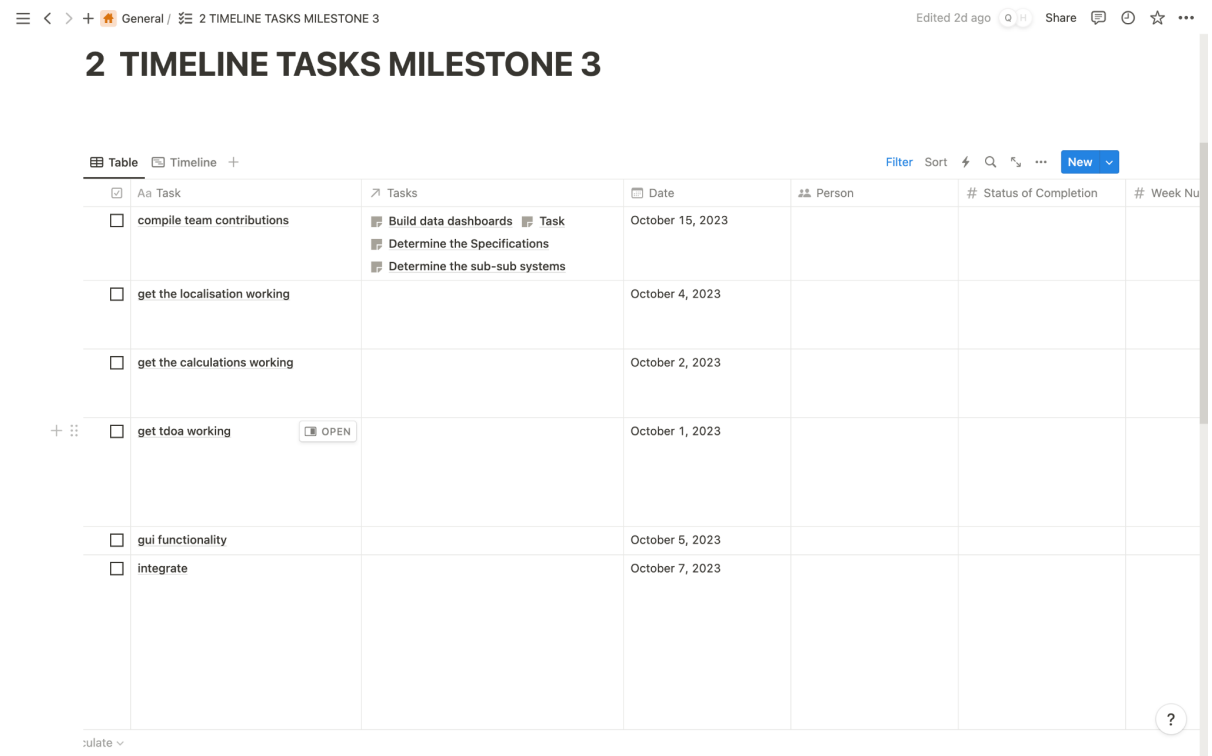
2. Admin Documents

2.1 Table of Contributions

JNKKET001	• Introduction • Consolidation of ATPs • Conclusion • Paper Design • Admin Documents • Validation using simulations
LWSHOL001	• Timeline • Data Collection and Analysis • Simulation Results and Analysis • Admin Documents • Validation using final implementation • Validation using simulations
PTTMAT005	• System Implementation minus GUI implementation • Overall Validation and Experimental Overview minus GUI • Future Plan • Validation using final implementation

Table 1: Table Containing the Contributions of Each Team Member

2.2 Project Management Tool



General / 2 TIMELINE TASKS MILESTONE 3

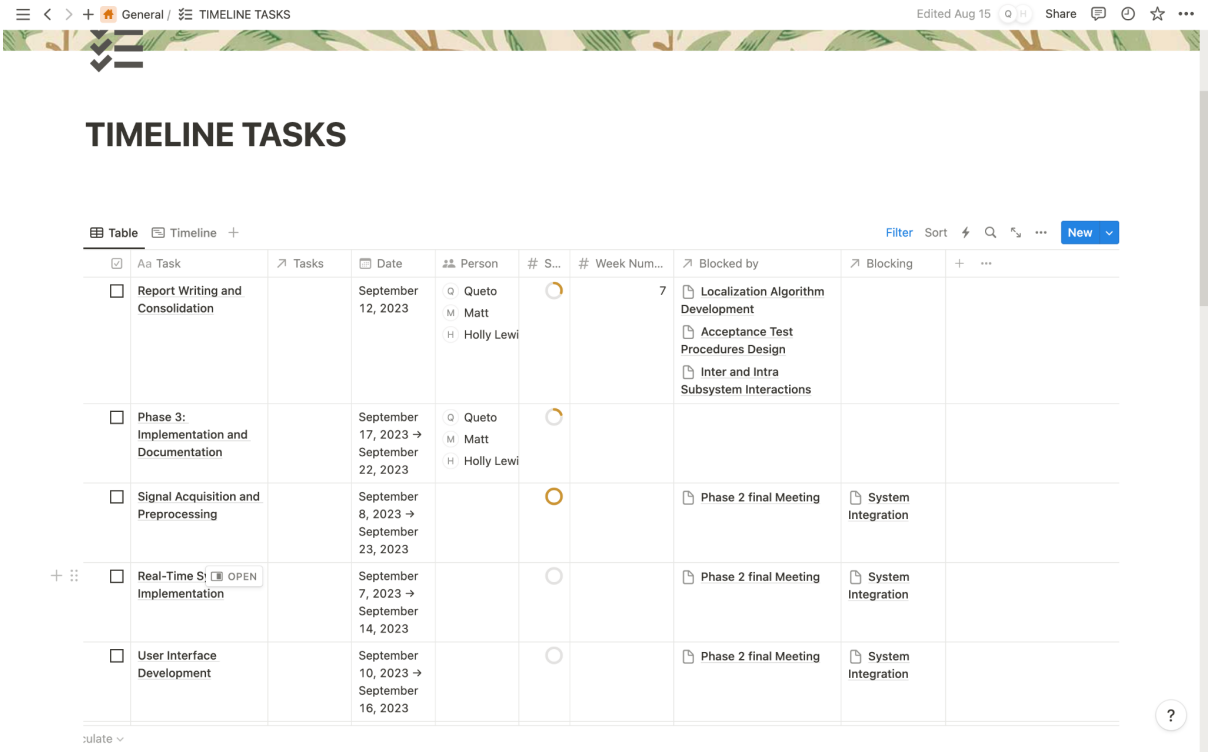
Edited 2d ago Share

2 TIMELINE TASKS MILESTONE 3

Table Timeline

Filter Sort New

Task	Tasks	Date	Person	Status of Completion	Week Nu
compile team contributions	Build data dashboards Task Determine the Specifications Determine the sub-sub systems	October 15, 2023			
get the localisation working		October 4, 2023			
get the calculations working		October 2, 2023			
get tdoa working OPEN		October 1, 2023			
gui functionality		October 5, 2023			
integrate		October 7, 2023			



General / TIMELINE TASKS

Edited Aug 15 Share

TIMELINE TASKS

Table Timeline

Filter Sort New

Task	Tasks	Date	Person	S...	Week Num...	Blocked by	Blocking
Report Writing and Consolidation		September 12, 2023	Queto Matt Holly Lewi		7	Localization Algorithm Development Acceptance Test Procedures Design Inter and Intra Subsystem Interactions	
Phase 3: Implementation and Documentation		September 17, 2023 → September 22, 2023	Queto Matt Holly Lewi				
Signal Acquisition and Preprocessing		September 8, 2023 → September 23, 2023				Phase 2 final Meeting	System Integration
Real-Time S OPEN Implementation		September 7, 2023 → September 14, 2023				Phase 2 final Meeting	System Integration
User Interface Development		September 10, 2023 → September 16, 2023				Phase 2 final Meeting	System Integration

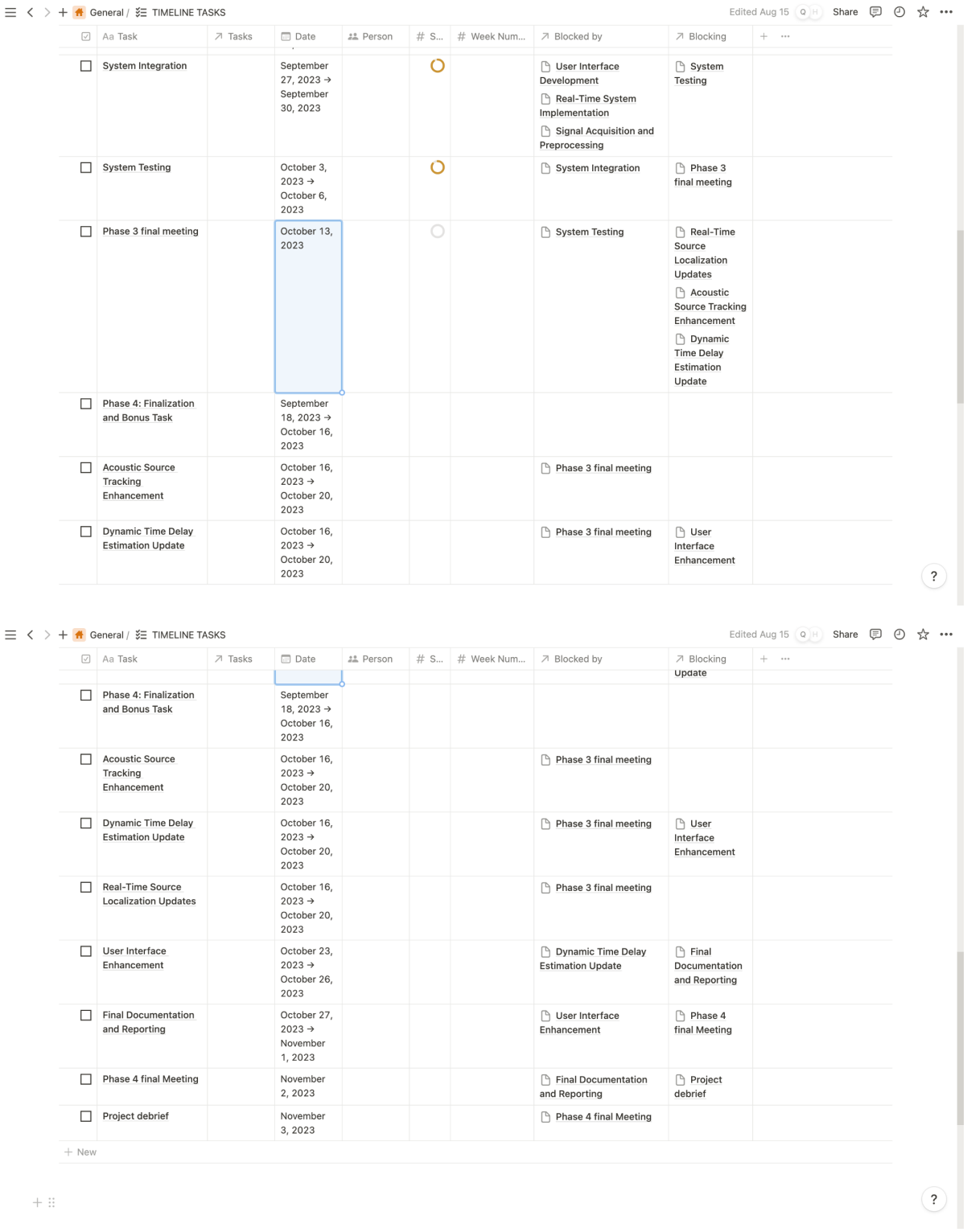


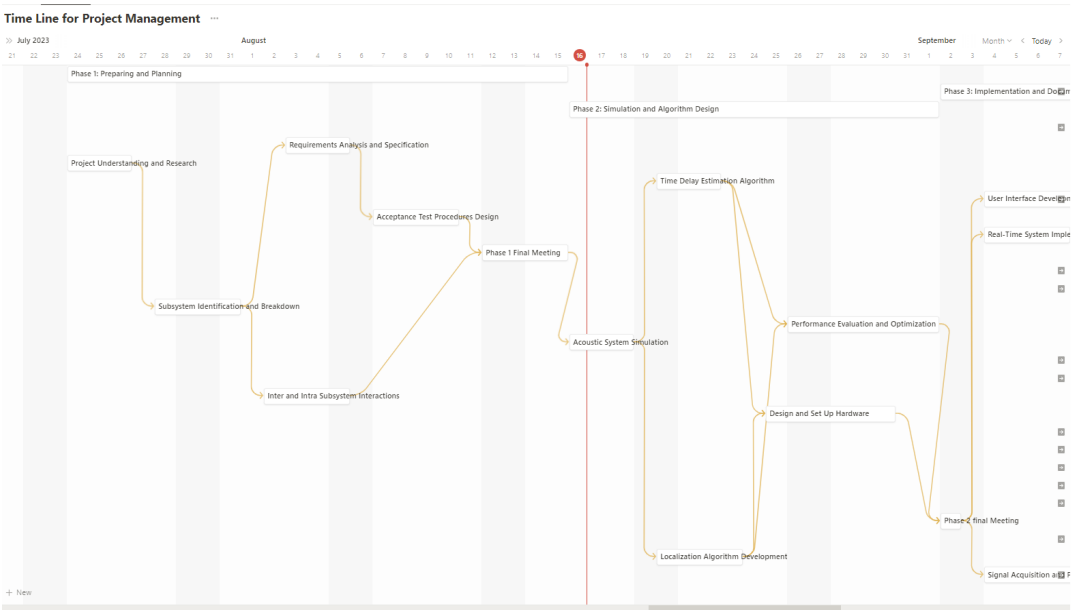
Figure 1: Snapshots from Project Management Tool

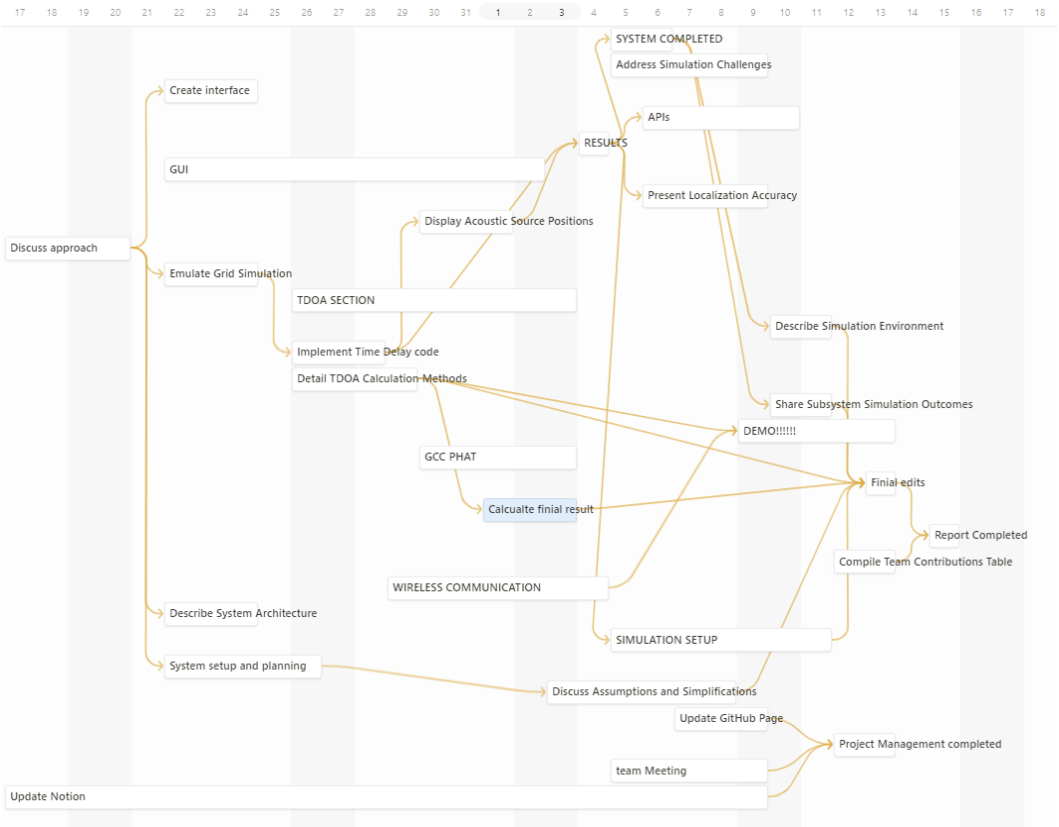
2.3 Link to GitHub Repository

An in depth account of all the project documentation, software and simulation results is available on our github for further perusal.

> https://github.com/quetojenkins/EEE3097S_EngineeringDesign_SoundTriangulationProject/tree/main

2.4 Adjusted Timeline





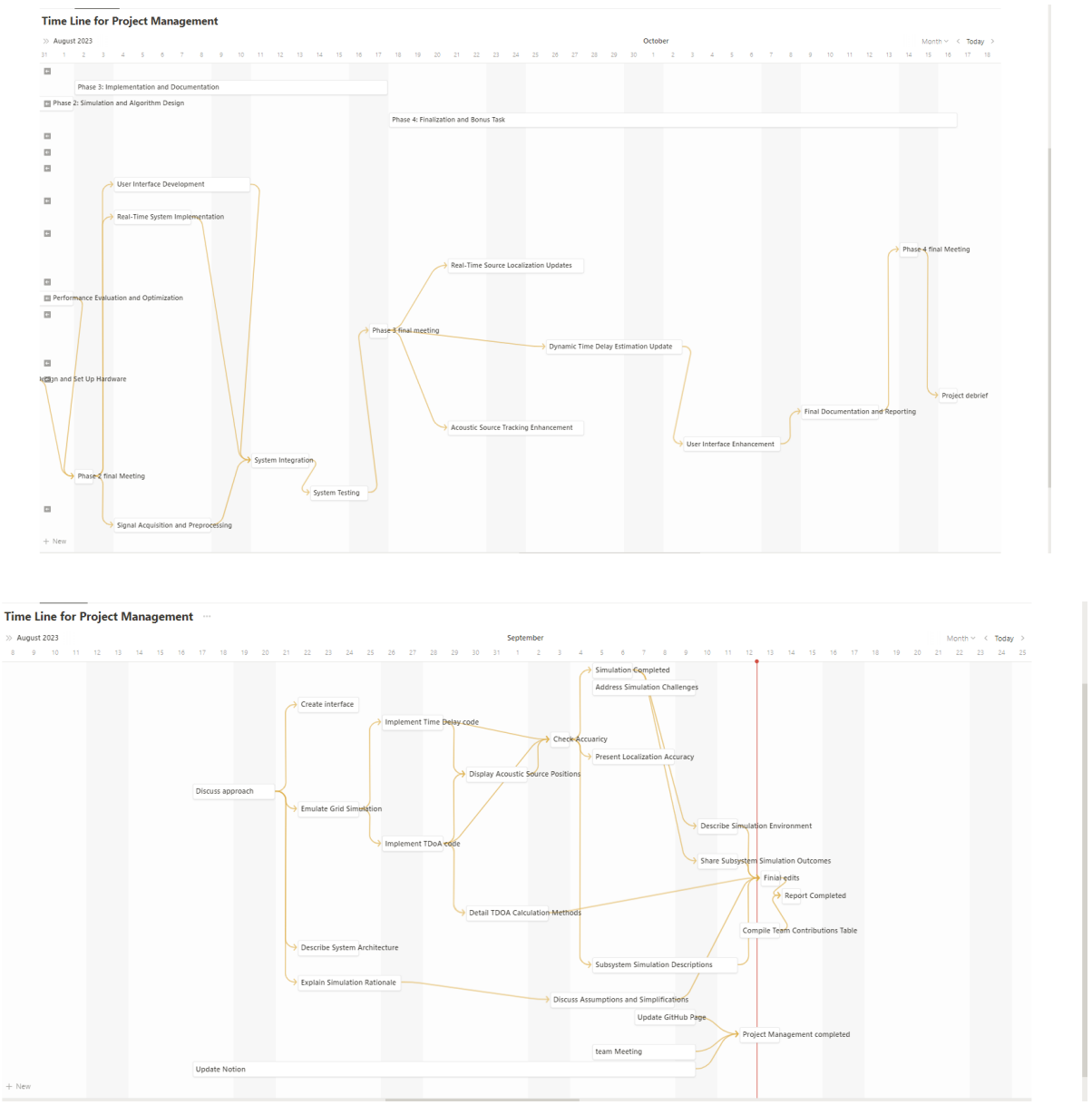


Figure 2: Timeline and Project Progress

We were successful in meeting our timeline goals, although small shifts were made, the final timeline of events is shown above.

3. Requirement Analysis

3.1 Requirement Discussion

The objective of the project is to design and implement an acoustic triangulation system using Time Difference of Arrival (TDoA) to accurately locate the position of a sound source within an A1 rectangular grid. The system must consist of an A1 rectangular grid, four microphone breakout boards and two Raspberry Pi microcontrollers. The system must feature audio signal capture and processing, calculation of two-dimensional coordinates of the sound source and a Graphical User Interface (GUI) for displaying results. The system should ideally be able to live track moving objects, with a realistic processing delay. It is important to note a potential limit on scalability is the computational power of the Raspberry Pi's, as taking more recordings per second to analyse could result in latency, however this will be accounted for when deciding on the frequency of recordings. The GUI may have potential delays or lags in updating due to data processing. The accuracy of the system will be limited by a number of factors, including processing power, noise and interference. These will need to be investigated during the development process and mitigated where possible, if mitigation is not possible then the effects will need to be minimised as much as possible.

3.2 Requirement Analysis

3.2.1 Microphone Selection

The selection is limited to *Adafruit i2s* microphones. Upon evaluation of the microphones it was determined that they are an appropriate choice for the project . This is based upon sensitivity, noise cancellation, and compatibility with Raspberry Pi. The *adafruit* microphone can record a range of 50Hz – 15kHz.

3.2.2 TDoA Calculation Algorithms

It was determined that the standard triangulation algorithm will not work due to the missing information about when the sound was initially played. Therefore it was decided that Time Difference of Arrival would be implemented due to its accuracy, computational complexity, and suitability for real-time processing. Using the time delay between a pair of microphones, a parabola representing possible locations can be derived. Using two pairs of microphones, with each pair orientated perpendicular to the other, two parabolas can be generated each representing a set of possible locations and orientated perpendicular to each. Finding the intersection of parabolas will allow for the approximate localization of the sound source.

3.2.3 Raspberry Pi Configuration

Investigations were conducted into various approaches to utilising the two Raspberry Pi microcontrollers effectively. This involved considering whether one Pi would manage signal capture while the other would handle TDoA calculation, or if both Pis would collaborate in performing all tasks or if it would be necessary for the Pi's to send the recording data to an external controller computer that could do the processing. After careful consideration the latter approach was chosen in order to make the workload of the Pis as even as possible which reduces latency and make sure any latency is even between the Pis.

3.2.4 GUI Frameworks

We explored two options for the user interface: utilising the Python module called pygame and creating a web interface. After careful evaluation, pygame was chosen because of its straightforward development process.

4. Subsystem Design Requirements and Specification

4.1. List of Requirements

4.1.1 Requirements for Audio Capture:

- High-quality microphones should be used so that sounds of a wide frequency range, with low noise, can be accurately captured for processing.
- The microphones must have synchronised clock signals and sampling rates to ensure the accurate measurement of time of arrival differences.
- The Raspberry Pi's need to provide high-precision time-stamping to account for very small input intervals.
- A high processing speed is required for real time tracking such that the sound can be quickly processed and be ready for the subsequent sound.
- The audio capture submodule must output to the time delay computation submodule an array representing the recorded sound.

4.1.2 Requirements for Time Delay Computation:

- The submodule must take in a digital representation of the recorded sound.
- The submodule must determine whether the sound is worth processing or if it is unwanted noise.
- The time delay between the two sound sources must be computed.
- The submodule must accurately measure the audio differences.
- The system must implement noise reduction techniques before the time-delay is computed.
- The subsystem must demonstrate real-time capability to ensure there are no processing delays.
- The submodule must be documented in a clear and concise manner so that it can be easily understood and maintained.

4.1.3 Requirements for Sound Localisation Submodule:

- The localisation submodule must be able to accurately estimate the 2D coordinates of a sound source within a rectangular grid.
- The submodule must be able to handle noise and other environmental factors.
- The submodule must be efficient in terms of processing time and memory usage.
- The localisation submodule should use a robust time delay estimation algorithm.
- The localisation submodule must be documented in a clear and concise manner so that it can be easily understood and maintained.

4.1.4 Requirements for Wireless Communication:

- The submodule must provide and interface for the other submodules to communicate between each other
- The submodule must have a protocol for easy communication
- The submodule must have allow for reliable communication
- The submodule must minimal latency for communication
- The submodule must allow for synchronisation between the Raspberry Pi's

4.1.5 Requirements for Frontend Submodule:

- The submodule must be user friendly.
- The submodule must display the location of the sound it detects in a human-understandable way.
- The GUI submodule must be documented in a clear and concise manner so that it can be easily understood and maintained.
- The GUI must display a digital grid, representing the A1 grid, to the user.
- The GUI must include control over basic settings.
- The GUI should be space and memory optimised.
- The GUI should be easily modifiable to account for imminent project updates.

4.2. List of Specifications

4.2.1 Specifications for Audio Capture:

- The submodule must output an array of recorded sound bits to the Time Delay Submodule.
- The Microphone must have a Signal to Noise Ratio (SNR) of at least 70dB to ensure accurate sound capture.
- The Microphone must have a frequency response of 50 Hz to 15 kHz to ensure that important sound details are captured.
- The microphones must be synchronised so that the TDoA calculations are accurate.
- The microphones must have a maximum synchronisation error of +/- 50 microseconds.
- Audio samples must be time stamped with a resolution of 0.1 seconds
- The audio-processing pipeline must have a maximum latency of 3 milliseconds from sound capture to position calculations.

4.2.2 Specifications for Time Delay Computation:

- The submodule must accept an array of recorded sound as an input with a minimum sample rate of 40kHz and a 16-bit resolution. This is to standardise the data to maintain its quality.

- The sound module must include an algorithm to analyse the input sound array and determine if the sound is significant.
- The submodule must implement a cross correlation algorithm to calculate the time delay between the sound sources.
- The submodule must achieve a time-delay resolution of at least 10 microseconds. The high resolution ensures precise localisation later on.
- The submodule must include a bandpass filter with a configurable frequency range to remove unwanted noise before cross-correlation.
- The submodule's time delay computation process must not introduce processing delays exceeding 20 milliseconds.
- The submodule must provide time delay outputs in milliseconds with a floating point decimal of 5 decimal places.

4.2.3 Specifications for Sound Localisation Submodule:

- The localisation submodule must be able to accurately estimate the 2D coordinates of a sound within $\pm$ 50 millimetres inside a A1 rectangular grid.
- The submodule must be able to ignore noise and other environmental factors with a Signal-to-Noise Ratio (SNR) of at least 70 dB.
- The submodel must have a processing time less than 50 milliseconds.
- The submodule must calculate parabolic equations from each pair of microphones using TDOA.
- The submodule must use Simultaneous equations to calculate grid coordinate from Parabolic equations.
- The submodule must output the grid coordinate to the Frontend Submodule.

4.2.4 Specifications for Wireless Communication:

- The Wireless Communication Submodule must be written in Python
- The submodule must communicate communicate over WiFi
- The submodule must communicate with a latency less than 20 milliseconds
- The submodule must use a plaintext custom written protocol that implements sending commands and results between the clients and server

4.2.5 Specifications for Frontend Submodule:

- The GUI submodule must use the "pygame" Python module to display the graphical location on an animated map of the A1 grid.
- The GUI submodule must be able to display the predicted results of the localisation submodule in a graphical user interface every 0.1 second.
- The GUI submodule must be able to stop and start the program.
- The GUI submodule Complete program must be less than 10 megabytes

5. Acceptance/Performance Test Procedure Creation based on Specifications

5.1 Specifications for Audio Capture ATP:

5.1.1 Microphone SNR, Frequency Response and Synchronisation Test

Specification: The microphone should capture sound with an SNR of at least 70 dB and capture frequencies of 50 Hz to 15 kHz. The pairs of microphones should have synchronised audio capture

Criteria: The captured audio should exhibit an SNR of 70 dB or higher when compared to a reference signal. The captured audio should exhibit consistent sensitivity across the specified frequency range. The time difference between the synchronised microphones should be within +/- 1 microseconds.

5.1.2 Audio Sample Timestamping and Latency Test

Specification: The audio samples should be time stamped with a resolution of at least 1 microsecond.

Criteria: The time difference between consecutive timestamps should be no less than 1 microsecond. The time to record and save the sample should not exceed 1.5 times the length of the overall recording time.

5.2 Specifications for Time Delay Computation ATP:

5.2.1. Input Sound Quality and Time Delay Calculation Accuracy Test

Specification: The subsystem should accurately assess the significance of input sound arrays, identify meaningful sound sources, and accurately calculate time delays.

Criteria: The subsystem should correctly classify at least 90% of significant sound sources while minimising false positives. The calculated time delays should have a mean absolute error of less than 1 microsecond when compared to a known reference.

5.2.2 Real-Time Processing Test

Specification: The time delay computation process should not introduce processing delays exceeding 2 seconds.

Criteria: The processing time for time delay computation should be consistently below 2 seconds under normal operating conditions.

5.3 Specifications for Sound Localisation Submodule ATP:

5.3.1. Localisation Accuracy and Noise Rejection Test

Specification: The localization subsystem should accurately estimate the 2D coordinates of a sound source within 50 millimetres of the actual position on the A1 rectangular grid.

Criteria: The calculated coordinates should have a mean absolute error of less than 50 millimetres when compared to the actual sound source position.

5.3.2. Processing Time Test

Specification: The processing time for sound localisation calculations should be less than 2 seconds to ensure real-time performance.

Criteria: The average time taken to calculate the sound's position should be consistently below 2 seconds across different scenarios.

5.4 Specifications for Wireless Communication:

5.4.1 Handling of Multiple Connections Simultaneously

Specification: The system should be able to handle multiple connections simultaneously in order to to avoid dropping connections from the Pi's

Criteria: The system should at least be able to handle 2 simultaneous connections in order to be effective for our needs.

5.4.2 Transmission Time

Specification: The system should be able to transmit the recordings in a reasonable amount of time that doesn't affect the overall performance of the system.

Criteria: The system should transmit the sound file in approximately half the length of time that the recording itself is.

5.5 Specifications for Frontend Submodule:

5.5.1. GUI Performance Test:

Specification: The GUI Subsystem using "pygame" should render the graphical location on the animated map smoothly, with no noticeable lag or stuttering during timely updates.

Criteria: The frame rate of the GUI animation should be consistently above 10 frames per second (FPS) under normal operating conditions.

5.5.2. Real-Time Display Test

Specification: The GUI should accurately display predicted results from the localisation subsystem every 0.1 seconds, maintaining synchronisation with the backend processing.

Criteria: The displayed results should update at intervals of 0.1 seconds with minimal deviation, ensuring real-time visualisation, and the GUI should respond to user interactions (start/stop) without crashes.

5.5.3. Program Size Test

Specification: The complete program, including all required modules and assets, should have a file size of less than 1 GB.

Criteria: The total size of the program's files should not exceed 1 GB to ensure efficient distribution and storage.

6. Paper Design

6.1. Feasibility Analysis

Upon inspection of the design's feasibility three main sections were analysed – technical feasibility, resource feasibility, and scalability. After evaluating the technical feasibility of implementing the acoustic triangulation system, the availability of required components and software libraries and tools. Components were chosen to effectively handle real-time audio processing and TDoA calculations. Little resource analysis was conducted as most resources were provided to and therefore met the availability and budget criteria. Furthermore, since the components, specifically the Raspberry Pi's, are widely used, there is significant online information. The scalability of this project is high; one can increase the grid size and add more microphones for a more accurate calculation. The group will plan for potential expansion and ensure the aforementioned approach can accommodate it.

6.2 Inter-Subsystem and Inter-Sub-subsystems Interactions

6.2.1 Audio Capture and Conversion

The Audio Capture and Conversion Sub-module is responsible for the capture and conversion of the audio signals, thus it takes in real analogue sound as the input and will output arrays representing those sound recordings to the Time Delay Calculation Sub-module. The Audio Capture and Conversion Sub-module has three subsystems inside it, which are responsible for:

- Audio Capture
- Microphone Synchronisation - this will be accomplished through the use of a stereo recording using both Microphones, then split into individual channels
- Audio Conversion

These subsystems work with each other to achieve the final goal of the overall Audio Capture and Conversion Sub-module.

6.2.2 Time Delay Calculation

The Time Delay Calculation Sub-module is responsible for calculating the actual time delays between the sound recorded at each microphone. It will take in the arrays from the Audio Capture and Conversion Sub-module representing the recorded sound and output a number representing the time difference in seconds between the sound arriving at the 2 microphones to the Wireless Communication Sub-module. It consists of the following subsystems:

- Sound Analysis – determine in which sound recording segment the sound of interest is in
- Time Delay Computation

6.2.3 Wireless Communication

The Wireless Communication Sub-module is tasked with implementing communication between the different Pi's and therefore communication between the different Sub-modules. It will communicate with various other Sub-modules and pass data between them as necessary. Its main functions will be to retrieve the time difference in seconds from the Time Delay Calculation Sub-module and pass that to the Sound Localization Sub-module. Furthermore it will also be responsible for sending any necessary commands to modules regarding configuration settings that need to be changed on the fly, like frequency of sound captures for example. These configuration settings will most likely be changed on the Front end Sub-module and then communicated over a custom protocol to the Audio Capture and Conversion Sub-module. It requires the following subsystems:

- Wireless Connection
- Wireless Communication Protocol

6.2.4 Sound Localisation

The Sound Localisation Sub-module will be in charge of the actual calculation of the position of the sound. It will take in 2 time differences in seconds from the Wireless Communication Sub-module, each representing a pair of microphones time difference of sound arrival. From those 2 values it will calculate the actual grid coordinates of the sound and pass that on to the Front End Sub-module. This Sub-module is broken down into the following subsystems:

- TDOA Equation – determine parabolic possibilities from time difference for each microphone pair.
- Position Localisation – Use the parabolic equations to find an intersection of the most likely position the sound originated from.
- Pi Synchronisation – Ensure that the Pi's are both outputting times that are representing the same sound and at the same time.

6.2.5 Front end

The Front end Sub-module is responsible for displaying the grid coordinates from the Sound Localization Sub-module in a clear manner, most likely on a visual grid. It will also have some configuration setting tools in order to change the operation of the overall system, these settings commands will be sent through the Wireless Communication Sub-module to the appropriate Sub-module. It will be made up of the following subsystems:

- Location Graphical Display
- Settings Interface

6.3 Possible Bottlenecks

6.31 Computational Load

It is important to note a potential limit on scalability is the computational power of the Raspberry Pi's, as taking more recordings per second to analyse could result in latency, however this will be accounted for when deciding on the frequency of recordings. Furthermore, the Python code itself could result in latency due to the high level nature of Python. To combat this, efficient code will be produced and the possibility of rewriting in the lower level compiled language will be investigated if necessary. Using parallelisation would also ensure that when the project is possibly scaled to include real time tracking the system will be able to respond quickly and accurately to changes in sound source positions as that will require much more processing.

6.4. UML Diagram

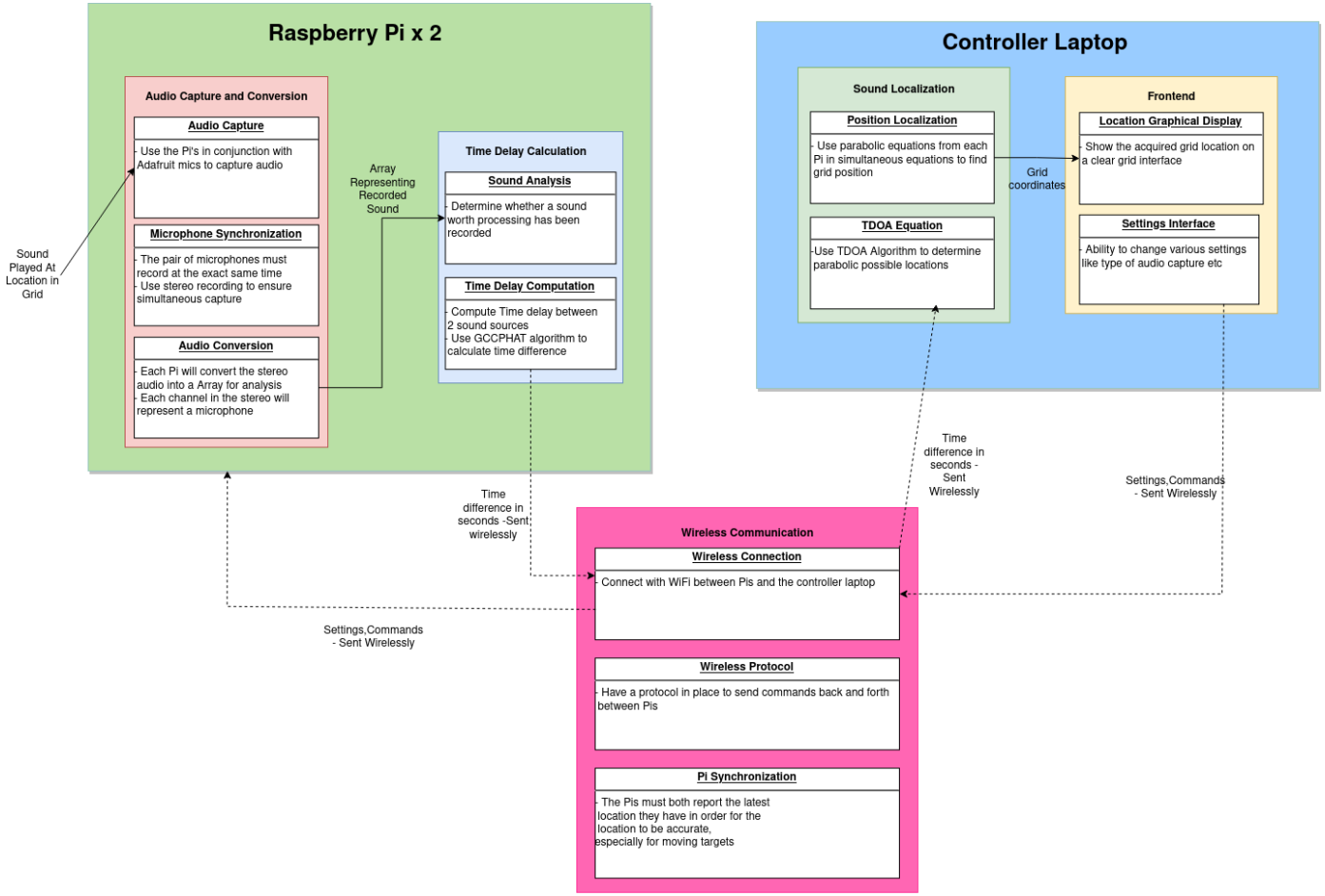


Figure 3: Figure Showing the UML diagram and inter subsystem Relations of the project.

7. Validation Using Simulations

7.1 Need for simulation based validation

We believed that it was incredibly important to validate our design using a simulation. It was necessary for a multitude of reasons, the first being that it would ensure that our method did indeed work under ideal circumstances. Furthermore it is useful as it provides an ideal benchmark that can be used later on for comparison with the real world implementation in order to gauge the performance.

MATLAB was chosen to simulate the grid (in which sound was triangulated). MATLAB was further used to analyse the data and implement the algorithms that computed the position of the triangulated acoustic sound. Additionally MATLAB was used to generate all the graphics and plots used to verify the simulations veracity.

7.2. Discussion of Simulation Steps

7.2.1 Grid Emulation Decision

MATLAB was chosen to emulate and display the A4 grid in which the acoustic triangulation will occur. MATLAB as a simulation environment for the Grid Emulation was chosen due to its ease of implementation. The software provides a very user friendly environment to define and manipulate grids. It also makes it easy to place objects in the grid. This is because of the rich set of built in functions and libraries that MATLAB has offered. Lastly, MATLAB is notorious for its emphasis on visualisation, making it very easy to plot the grid exactly as wanted.

7.2.2 Sound Propagation Model Decision

MATLAB was chosen to simulate the sound propagation in a grid. It was used because it is exceptionally well-suited for mathematical modelling and numerical simulations. This means that the semi-complex equations and arrays were very easy to manipulate due to MATLAB's excellent features.

The approach to simulating the sound propagation between source and receivers (microphones) was to use mathematical equations to calculate first the distance between source and receivers, then using those values the time delays of the sound from the source to the receiver were computed using the expected speed of sound (343 m/s).

7.2.3 Acoustic Sound Signal Generation Decision

MATLAB was used for the generation of the acoustic sound “played” in the grid. This was chosen because MATLAB has various sounds preloaded that are easy to use. Furthermore MATLAB can very easily “delay” the sounds using predefined functions, this was important in simulating the sound travel as mentioned above.

7.2.4 TDoA Equations

MATLAB is excellent for performing triangulation using Time-Difference-of-Arrival (TDoA) equations due to its excellent numerical computation capabilities and extensive mathematical libraries. It simplified the implementation of TDoA algorithms, allowing localisation of the sound and providing tools for visualisation and analysis of results.

The approach to simulating the triangulation algorithm was to calculate the Time Difference of Arrival between two microphones in a pair. (The microphones were paired on opposite sides of the grid). Using the time delay in simultaneous equations, the coordinates of where the sound was played could be calculated.

7.2.5 Simulation Outputs Decision

The following visualisation and data was produced from the simulations:

1. Visualisation of the grid with the actual and calculated positions of sound displayed.
2. Positions of Microphones on grid
3. Coordinates of actual sound printed to screen.
4. Coordinatesof calculated sound position printed to screen
5. Benchmark times for various part of program

7.3. Simulation Experimental Set-Up

7.3.1. Audio Capture and Conversion SubsystemSimulation

In order to simulate a sound being played and travelling MATLAB was used to calculate a random location for the sound somewhere in the grid space. A gong sound was loaded from a file and chosen for its impulse like frequency representation. Gaussian noise was then added to the signal. Then, having the positions of the four microphones defined in the MATLAB Code the distance between the sound location and each of the four microphones was calculated. Once completed, it was used to compute the time it would take for the sound to travel from the source location to the location of each microphone. This was used to add a time delay to the gong sound waveform for each of the microphones. The four audio signals were *captured* and passed in the form of arrays to the Time Delay Calculation Submodule, which would take place on the Raspberry Pi’s.

7.3.2. Time Delay Calculation Subsystem Simulation

The Time Delay Calculation Subsystem takes in four arrays that contain the sound recorded at each microphone. All the arrays were expected to contain roughly the same signal, but with varying time shifted values. Using the GCC_PHAT algorithm packaged for MATLAB, the submodule calculated the time delays between the microphones in each opposite pair of mics. This assumes that each opposite pair of microphones are synchronised thereby making the time delay meaningful. Please see figure 3 for more context. This module outputs two double precision values, namely tau1_2 and tau3_4, that represent the time difference of arrival between each microphone in a pair in seconds. These values are outputted to the Sound Localisation Subsystem.

7.3.3. Sound Localisation Subsystem Simulation

The Sound Localisation Subsystem takes in two time difference values from the aforementioned Time Delay Calculation Subsystem. Each time difference value is used to generate a parabola of possible locations of the sound source on the grid. By using simultaneous equations with two of these parabolas, with each parabola in perpendicular directions, the interception point is found, which represents the detected position. These positions are sent to the Frontend Subsystem to display the final results.

7.3.4. Frontend Subsystem Simulation

The Frontend Subsystem takes in the detected and calculated sound location, and for the case of the simulation also takes in the actual sound location for comparison. It then plots a grid with the values for comparison. The microphone positions are also plot

7.4 Overall System Architecture

7.4.1 Raspberry Pi’s

As shown in the implementation there is code that will be run on the Raspberry Pi’s. While not implemented in these simulations, the Raspberry Pi’s and using the microphones and various code will be responsible for converting real audio into timestamped arrays representing the sound each microphone picked up and noting when a sound worth processing had been detected. Furthermore Time Delay calculations will take place on the Raspberry Pi’s. Once those calculations are complete the results will be sent to a “Main Controller” for the Sound Localisation stage. This could possibly take place on an external machine that would also be in charge of the Frontend Display, alternatively this could take place on the Raspberry Pi’s. This will be decided in the final implementation and possibly have the option for both methods, i.e.

having the ability to operate the project without an external Frontend Display in a headless mode.

7.4.2 Microphones

As shown below the microphones will be placed at the centre of each edge of the grid. The microphones that are “paired together” are microphone 1 and 2, and microphone 3 and 4 denoted by the colours blue and magenta respectively. The microphones themselves have little responsibility besides the actual recording aspect.

8. Simulation Results and Validation

8.1 Results

8.1.1 Time Difference of arrival Results:

Table 2: Table containing the Time Difference of Arrival Results and relative accuracy to the actual values.

Actual Tau1-2 (s)	Calculated Tau1-2 (s)	Error(s)	Accuracy (%) Tau1-2	Actual Tau3-4 (s)	Calculated Tau3-4 (s)	Error(s)	Accuracy (%) Tau3-4	Average Accuracy (%)
-0.00125761	-0.00122070	3.69058E-05	97.065396	7.64E-04	7.32E-04	3.15132E-05	95.87489	96.47014349
-7.83E-04	-7.32E-04	5.06862E-05	93.52756147	-8.86E-04	-8.54E-04	3.19618E-05	96.39442	94.96098875
-9.24E-05	-9.04E-05	1.96055E-06	97.87744309	3.41E-04	3.66E-04	2.50153E-05	92.66834	95.27289377
0.001137036	0.001098633	3.84034E-05	96.62249634	7.90E-04	7.32E-04	5.78415E-05	92.68073	94.65161451
0.001190501	0.001220703	3.02023E-05	97.46305761	-5.69E-04	-6.10E-04	4.17534E-05	92.65678	95.05991648
AVERAGES:		**3.16317E-05**	**96.5111909**			**3.7617E-05**	**94.05503**	**95.2831114**
STANDARD DEVIATION		**1.81602E-05**	**1.548823555**			**1.14384E-05**	**1.705949995**	**0.626341262**

8.1.2 Sound Localisation calculation Results:

Table 3: Table containing accuracy of calculated sound grid position vs actual grid position.These results were generated using a random x and y position with the 100% accurate tau result.

Actual X Position (mm)	Calculated X Position (mm)	Error(mm)	X Accuracy (%)	Actual Y Position (mm)	Calculated Y Position (mm)	Error(mm)	Y Accuracy (%)	Average Accuracy (%)
651.7789	651.7789	0	100	452.896	452.896	0	100	100
5.51E+02	5.51E+02	0	100	59.1839123	59.1839123	0	100	100
416.0106	416.0106	0	100	308.6419	308.6419	0	100	100
292.955	292.955	0	100	169.56	169.56	0	100	100
180.7383	180.7383	0	100	115.3294	115.3294	0	100	100
AVERAGES		**0**	**100**			**0**	**100**	**100**
STANDARD DEVIATION		**0**	**0**			**0**	**0**	**0**

8.1.3. Integrated System Results:

Table 4: Table containing the combined results of the calculated sound positions vs the actual sound source positions and the relative accuracy.

Actual X Position (mm)	Calculated X Position (mm)	Error (mm)	X Accuracy (%)	Actual Y Position (mm)	Calculated Y Position (mm)	Error (mm)	Y Accuracy (%)	Average Accuracy (%)
651.7789	638.8475	12.9314	98.01598364	452.896	437.1978	15.6982	96.53382	97.27490078
5.51E+02	538.6672	11.87693404	97.84269175	59.18391229	72.4011	13.21718771	77.6676	87.75514614
416.0106	421.1942	5.1836	98.75397406	308.6419	313.0456	4.4037	98.5732	98.66358745
292.955	292.3895	0.5655	99.80696694	169.56	157.9354	11.6246	93.14426	96.47561133
180.7383	171.6514	9.0869	94.97234399	115.3294	101.5366	13.7928	88.04052	91.50643049
AVERAGES:		**7.928866809**	**97.87839208**			**11.74729754**	**90.79188**	**94.33513524**
STANDARD DEVIATION		**5.091887536**	**1.609432586**			**3.896094639**	**6.817492042**	**4.56226876**

Table 5: Table containing the final accuracy and standard deviation. The position was modelled as a vector from the origin to the sound position so that a single value, magnitude, could be used to compare the calculated and the actual results.

Actual magnitude (mm)	Calculated magnitude (mm)	Error (mm)	Accuracy (%)
793.6816246	774.1240499	19.55757478	97.5358413
553.7161538	543.5110594	10.20509447	98.1569809
518.0006194	524.7876731	6.78705361	98.68975956
338.4866698	332.3179355	6.168734246	98.17755475
214.3996352	199.4339095	14.96572573	93.0197056
AVERAGES:		**11.53683657**	**97.11596842**
STANDARD DEVIATION		**5.084093234**	**2.080536151**

8.2. Discussion of Results

8.2.1. Time Difference of Arrival Results:

- For Tau1_2 (Time Delay):
    - The calculated values are close to the actual values, resulting in accuracy of 96.5% and a standard deviation of 1.55% within this accuracy.
    - The precision of these values was high as the actual values had an average error of 1.82mm a standard deviation of 1.55 mm within the error meaning that the values were both very accurate and very precise.
- For Tau3_4 (Time Delay):
    - Similar to Tau1_2, the calculated values are quite close to the actual values, resulting in accuracy of 94.05% with a standard deviation of 1.71% within this accuracy.
    - The precision of these values was high as the actual values had an average error of 1.14mm a standard deviation of 1.71mm within this error meaning that the values were both very accurate and very precise.
- Total Accuracy:
    - The overall accuracy for this section averages at 95.28%. The inaccuracy this represents may be a result of the sampling rate and the additive noise, however this result is satisfactory.
    - The average precision of these values is exceptionally high, with an average standard deviation (of accuracy) of 0.62%. This means that not only are the results correct, but the correctness is replicated consistently.
    - These slight inaccuracies were caused by a finite sampling rate and a small amount of noise present

8.2.2. Sound Localisation Calculation Results:

- For X, Y, Final Positions:
    - Actual and calculated X/Y positions are the same, resulting in 100% accuracy as well as 0 standard deviation as the calculation is merely a formula and therefore when using correct values there will be no error.
    - This shows that our method of working out the position of the sound from the arrival time differences is not only 100% accurate but also 100% precise, this means that the sound localisation module will calculate the exact position every time when given the correct tau value.

8.2.3. Integrated System Results:

- For X Position:

    - There is a tiny discrepancy between actual and calculated X positions, resulting in an average of 97.87% accuracy and a standard deviation of 1.61%.
- For Y Position:
    - Compared to X Position, there is a slightly larger difference between actual and calculated Y positions, resulting in an accuracy of 90.79% and a standard deviation of 6.81%.
    - This was due to an outlier that had 77.67% accuracy. This was caused by its very small y coordinate value, so although the error amount was consistent with the rest of the tests, when compared to its small actual value it produced a large percentage error.
- Total Accuracy:
    - The overall accuracy for the entire project was 97.16% with a standard deviation of 2.08%.
    - The average error was 11.54mm with a standard deviation of 5.08mm.
    - This accuracy and precision far exceeds our needs as well as our specifications. Our system not only produces accurate results, but will produce accurate results every time which is shown by its low standard deviation values which prove it has high precision.
    - The slight inaccuracies are due to the errors in the tau values which were caused by a finite sampling rate and a small amount of noise present.

The system is very scalable, the high accuracy and precision proves that it is a good base for a larger system, by adding more microphones the system can be expanded to detect sound in a larger area or if more microphones were added it could improve the accuracy and precision for the existing grid.

9. Validation Using Final Implementation

9.1 Need For Final Implementation Validation

It was obviously very necessary to validate our design in the real world using real components once we had determined that the design worked in the simulation. Doing this also allowed us to determine under which circumstances our design performed adequately and which circumstances caused it to function sub optimally. Furthermore using our simulation results we could compare how our implementation in the real world performed compared to an ideal simulated case.

Our project was implemented using the Python programming language. Some of the code was run on the Pi's themselves in order to record and transmit the sounds to the Controller system. The controller then processed the sounds and did the necessary calculations to locate the sound.

9.2 Discussion of Steps for Physical Implementation Validation

9.2.1 Sound Capture Subsystem Steps

The capture of sound took place on the Raspberry Pi's themselves. Each Pi was responsible for two microphones. Using the *arecord* utility, each Pi used the mics in a stereo configuration in order to synchronise the recording. Sound chunks were recorded for 2 seconds at a sampling rate of 48 000 Hz. Each Pi would record its 2 second stereo chunk, read the file into memory, split up the 2 channels which each represent the recording from a microphone and then transmit each channel as an array to the controller. The Pi's both wait for the Communication Subsystem to alert them as to when they should record. This will be further discussed in the Communication Subsystem description. The code for this module can be found in *client1_2.py* and *client3_4.py*. They also contain some of the code needed for the Communication Subsystem.

This module communicates with the rest of the system through the Communication subsystem. Its input is essentially a signal telling each Pi when to record. Each Pi outputs two arrays; the left and right channel of the stereo file that it recorded. This gets sent to the controller laptop for additional processing and calculations.

9.2.2 Time Delay Calculations Subsystem Steps

The Sound Localisation Subsystem takes in two time difference values from the Time Delay Calculation Subsystem. Each time difference value is used to generate a parabola of possible locations of the sound source on the grid. By using simultaneous equations with two of these

parabolas, (each parabola is in perpendicular directions), the intersection point is found and this represents the detected position. The Sound Localisation Subsystem used Pythons' Sympy library to solve the symbolic simultaneous equations.

This subsystem takes in the Time Difference of Arrival values from the Time Delay Calculation Subsystem and converts them into an x and y position representing the position of the sound. This data is then given to the Communication Subsystem which communicates it to the Frontend Subsystem.

9.2.3 Communication Subsystem Steps

The synchronisation effort here is not designed to perfectly synchronise the Pi's as that is unnecessary in our Localisation Method, but rather to ensure that they both transmit their data at similar enough times to avoid unnecessary idle time. This overall process was implemented using the Python web server module named Flask. Flask listened for connections from the Pi's which made API calls using the Python requests library. Flask was used as it provides good performance and multithreading is built in. This was useful for handling the Pi's simultaneous connection.

The Communication Subsystem also made use of the Python Multiprocessing library, this was used to allow the Pi to run the Time Delay Calculation and Localisation Subsystems on a separate CPU core. Thus, allowing for a reduced delay in processing time. The code that implements the transmission from the Pi's can be found in *client1_2.py* and *client3_4.py*. The code that implements the receiving the Pi's data and calls the Time Delay Calculation and Localisation Subsystems can be found in *ControllerServerMultiCore.py.* Once the actual position has been determined, the Communication Subsystem passes this information to the Frontend Subsystem through the use of a text file called data.txt that the Frontend Subsystem is continuously reading from.

9.2.4 Front End Subsystem Steps

The main role of the GUI is to provide the user of the localisation system with the localisation data produced in a readable manner. This means the GUI should display the calculated coordinates of the position of sound emanation not in terms of calculations or numbers but in an infographic way. Additionally, it is responsible for displaying the current processing speeds and the connection statuses of the pi's. The GUI was designed such that it is the central point of control for the user over the system.

10. Final Experimental Setup

The acoustic signal collection process in our system was fully automated. If a sound was played within the grid while the system was live, it would continually take 2-second samples and print out the analysis while the next sample was being recorded. This was placed in a loop and occurred infinitely until the system was terminated.

To test the system efficiently, various sounds were tested. Initially, a speaker was used; however, it was determined that the speaker output a sound from too large of a surface area and therefore skewed the results. Next, music from a phone speaker was tested, which had a similar problem as phones have two speakers (top and bottom) . For preliminary testing, clapping was used. Still, this was further improved by clicking loudly and then further improved by using a clicking device used in dog training. The click that it produced is localised to a small area. The clicker can be placed accurately on the grid, thereby reducing the human error element created by slightly moving claps/clicks. Furthermore, this means that the acoustic sound does not need to be stored as it can be easily produced. There was experimentation with recording the sound; however, when analysing the waveforms, it was determined that this simply increased the amount of noise while also lowering the overall power of the signal. Therefore, it was determined that it would be better to manually create a new sound each time by using the clicker.

To collect data for the analysis 5 varying points were chosen. The points were chosen to give a good spread of x and y values and ensure that the system can accurately determine where the sound is anywhere in the grid, including the edge cases. Below shows the points that were chosen to test the system.

In order to isolate the sound localisation module, it was tested with correct tau values so that the only error would be within the module itself. To do this, the matlab simulation that was instituted for milestone 2 was used to generate correct tau values corresponding to the 5 test points. These tau values were then inputted to the sound localisation function and the positions it returned were printed to the screen and copied into excel. It was expected for these results to be 100% accurate as the same algorithm that was used that was extensively tested for milestone 2.

11. Final Results and Analysis

11.1. Time Difference of arrival final results:

Table 2: Table containing the Time Difference of Arrival Results and relative accuracy to the actual values.

Coordinates of sound	Actual Tau1-2 (s)	Calculated Tau1-2 (s)	Error(s)	Accuracy (%) Tau1-2	Actual Tau3-4 (s)	Calculated Tau3-4 (s)	Error(s)	Accuracy (%) Tau3-4	Average Accuracy (%)	Average Error
(100,100)	1.562E-03	1.607E-03	4.517E-05	97.11	-5.358E-04	-5.143E-04	2.146E-05	96.00	96.55	3.33E-05
(200, 4000	1.074E-03	9.883E-04	8.592E-05	92.00	6.519E-04	5.560E-04	9.592E-05	85.29	88.64	9.09E-05
(400, 250)	0	1.276E-04	2.760E-05	72.40*	0	5.688E-05	4.313E-05	56.88*	64.64*	3.54E-05
(600,200)	1.154E-03	1.229E-03	7.487E-05	93.51	-2.266E-04	-2.188E-04	7.820E-06	96.55	95.03	4.13E-05
(800,0)	-1.735E-03	-1.596E-03	1.389E-04	91.99	-6.776E-04	-7.161E-04	3.857E-05	94.31	93.15	8.88E-05
AVERAGES:			**7.450E-05**	**89.40**			**4.138E-05**	**85.80**	**87.60**	**5.79E-05**
STANDARD DEVIATION			**4.285E-05**	**8.71**			**3.003E-05**	**15.03**	**11.79**	**2.93E-05**
VARIANCE			**1.836E-09**	**94.74**			**1.127E-09**	**282.20**	**173.66**	**8.57E-10**

11.2. Sound Localisation calculation final results:

Table 3: Table containing accuracy of calculated sound grid position vs actual grid position.These results were generated using a random x and y position with the 100% accurate tau result.

Actual X Position (mm)	Calculated X Position (mm)	Error(mm)	X Accuracy (%)	Actual Y Position (mm)	Calculated Y Position (mm)	Error(mm)	Y Accuracy (%)	Average Accuracy (%)
100	100	0	100.00	100	100	0	100.00	100.00
200	200	0	100.00	400	400	0	100.00	100.00
400	400	0	100.00	250	250	0	100.00	100.00
600	600	0	100.00	200	200	0	100.00	100.00
800	800	0	100.00	0.01	0.01	00	100.00	100.00
AVERAGES		**0**	**100**			**0**	**100**	**100**
STANDARD DEVIATION		**0**	**0**			**0**	**0**	**0**
VARIANCE		**0**	**0**			**0**	**0**	**0**

11.3. Integrated System final results:

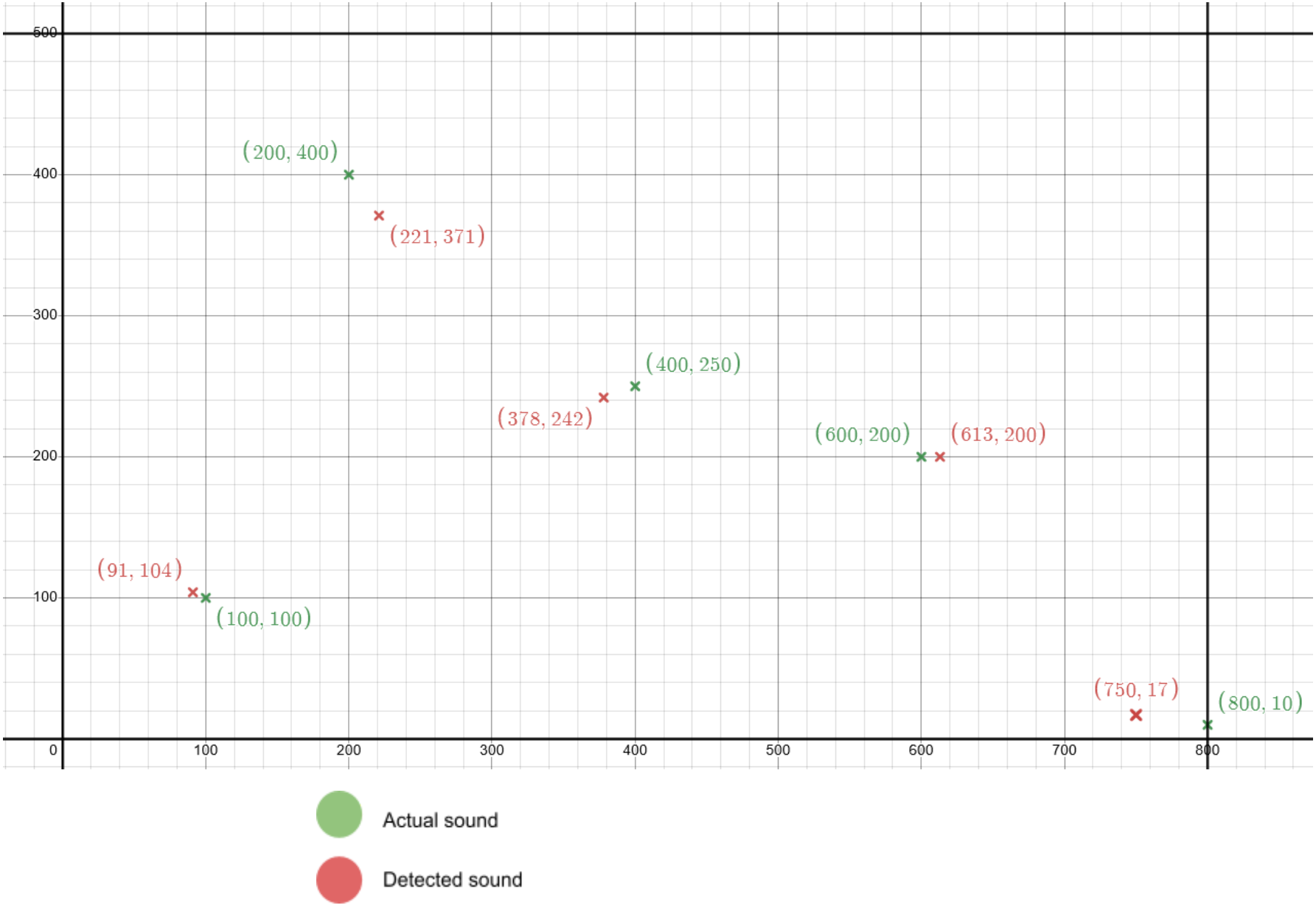
Table 4: Table containing the combined results of the calculated sound positions vs the actual sound source positions and the relative accuracy.

Coordinates of sound	Actual X Position (mm)	Calculated X Position (mm)	Error (mm)	X Accuracy (%)	Actual Y Position (mm)	Calculated Y Position (mm)	Error (mm)	Y Accuracy (%)	Average Accuracy (%)
(100,100)	100	91	9	98.88	100	104	4.00	96.00	97.44
(200, 4000	200	221	21	89.50	400	371	29.00	92.75	91.13
(400, 250)	400	378	22	94.50	250	242	8.00	96.80	95.65
(600,200)	600	613	13	97.83	200	200	0.00	100.00	98.92
(800,0)	800	750	50	93.75	10	17.00	7.00	30.00	61.88
AVERAGES:			**23**	**94.89**			**9.60**	**83.11**	**89.00**
STANDARD DEVIATION			**16.05**	**3.32**			**10.09**	**24.33**	**15.44**
VARIANCE			**257.50**	**13.76**			**127.30**	**888.11**	**238.51**

Table 5: Table containing the final accuracy and standard deviation. The position was modelled as a vector from the origin to the sound position so that a single value, magnitude, could be used to compare the calculated and the actual results.

Coordinates of sound	Actual magnitude (mm)	Calculated magnitude (mm)	Error (mm)	Accuracy (%)
(100,100)	141.42	138.19	3.23	97.72
(200, 4000	447.21	431.84	15.38	96.56
(400, 250)	471.70	448.83	22.87	95.15
(600,200)	632.46	644.80	12.35	98.05
(800,0)	800.06	750.19	49.87	93.77
AVERAGES:			**20.74**	**96.25**
STANDARD DEVIATION			**15.87**	**1.60**
VARIANCE			**314.73**	**3.21**

Figure 4: Illustration of the actual sound position vs the calculated sound position in testing



11.4 Integrated System calculation Results Analysis and Discussion

The errors in the final system integration properage from the time difference of arrival (TDoA) calculation. The errors, as previously discussed, are caused by noise and distortion. However, overall the system produced good results and can be analysed in 3 ways. Firstly, but looking at the standard accuracy. This however produces some misleading results and is affected by the magnitude of the position as previously discussed, however it does give a good overview and produces an accuracy of 89%. Secondly, it can be analysed by looking at the magnitude of the error, this is the most useful as it is not affected by the position magnitude and therefore gives the most raw result for accuracy.Using this method, there was an average error of 16.3 mm. Finally, you can look at the accuracy of the vector. This was calculated by turning the coordinates into a vector from the origin to give a % accuracy result that was not affected by the 0 coordinate as previously discussed. During the testing and results, a total accuracy of 96.25% and an error of 20.74mm was calculated. These results were surprisingly accurate and are only a representation of our system in almost perfect conditions. Furthermore, when taking the average of the produced results the errors evened out to give a more accurate location. Furthermore this is our system's accuracy when given a large amount of time. Even so, this shows that our system has the potential to be very accurate when placed in the correct environment. Moreover, the results are not only accurate but also very precise. When looking at table 4, the standard deviation of the accuracy was only 15.44%. Similarly when looking at table 4, the standard deviation of the accuracy was 1.60%

12. Consolidation of ATPS and Future Plans

12.1 ATPS

The ATP's for the project can be viewed in section 4.3.

The following Table illustrates whether the project specifications have been met. This is shown by testing the ATP's. As you can see, all the project specifications previously mentioned in section 4.2 have been met. This means that the previous specifications do not have to be adjusted to accommodate for the current state.

Subsystem	ATP	Pass or Fail	Reason
Frontend	GUI Performance Test	PASS	*Hardcoded*
	Real time Display Test	PASS	*Hardcoded*
	Program Size Test	PASS	*Memory Profiler Validated*
	GUI processing Speed Test	PASS	*Pass, updates every 0.1 seconds*
Sound Localisations	Localisation Accuracy and Noise Rejection Test	PASS	*94% accuracy and off by 16 mm*
	Processing Time Test	PASS	*0.8s*
Time Delay Computation	Input Sound Quality and Time Delay Calculation Accuracy	PASS	*90% accuracy*
	Real Time Processing Test	PASS	*1.2 seconds average*
Audio Capture Acceptance Tests	Microphone SNR, Frequency Response and Synchronisation Test	PASS	*Acceptable SNR and 50-15KHz audio capture*
	Audio Time stamping and Latency Test	PASS	*Sampling rate of 48 000 2.5 seconds time*
Communication Module	Handling of Multiple Connections Simultaneously	PASS	*Pi's were connected simultaneous and consistently*
	Transmission Time	PASS	*0.8 seconds*

12.2 Future Plans Overview

Our system performed very well and we believe that we completely met the requirements and specifications we set out to achieve, that being said through the implementation process the group had numerous great ideas that could be implemented given an extended timeline and additional funding/components. Some of the ideas that will be discussed are aimed at scaling the operation of this project to a size appropriate for real world application and others are aimed at just making the project itself better and more functional. In order to make the future plans easier to understand we will group them by which subsystem they would affect. This is also done due to the modular design of the project, making it possible for these plans to be implemented incrementally or on specific subsystems if that were desired.

12.3 Sound Capture Subsystem Future Plans

The group had various ideas for how it would be possible to improve and scale the Sound Capture Subsystem. The first idea the group had was changing the location of the microphones. This was examined during the actual implementation stage and it was determined that changing the microphones locations changed what areas of the grid the system would more accurately detect on, this would be something worth reexamining when scaling to the real world. The next obvious improvement needed for real world scaling would be significantly improved microphones. While the microphones we used were adequate for the scale we were working at, for projects aiming to work on areas that could possibly extend to more than $1\ km^2$ significantly more sensitive microphones would need to be used. The positioning of the microphones would also have to be considered as putting them too high might affect the TDOA calculations and putting them too low might result in sound not reaching them as well due to being blocked by houses and other buildings. Another incredibly useful feature that would be very necessary for this project to operate at large scale would be the implementation of noise classification, by this we are referring to the use of algorithms to determine what the sounds coming into the microphones are, for example being able to determine that the loud sound that was just observed was a gunshot. This could be implemented in various ways, for example the use of a machine learning model, the operators of the project could then specify which sounds the project should try to locate and which sounds shouldn't be located and only analyse sound samples in which a sound that is actually worth determining the location of is contained.

12.4 Time Delay Subsystem Future Plans

This part of the project was something we had few improvements for. It would possibly be worth investigating some alternative methods for determining the delay between the microphones but we were overall happy with the performance. Our chosen algorithm was

accurate enough that when scaled up it would provide time delays that are accurate enough to determine sound locations to great levels of accuracy.

12.5 Sound Localization Subsystem Future Plans

One of the possible improvements we foresee here could be the addition of more microphones. Using additional microphones would allow for greater accuracy and even the possibility of the addition of the 3rd dimension to the localization.

12.6 Communication Subsystem Future Plans

The main addition we thought would be extremely useful and that would take place in the communication system is the ability to dynamically add new microphones to the setup. When a new microphone is added the position of it would obviously be required as well but apart from this it would be very nice if that could automatically take care of the rest of the process. This would be especially useful when deploying the system in real world applications.

12.6 Front end Subsystem Future Plans

We were exceptionally happy with our frontend, any changes that would be required for it would mainly be due to additional features added at a later stage in other modules, that might require some input or output from or to the frontend display.

13. Conclusion

This report shows the path from concept to realisation in the implementation of the acoustic signal triangulation system. It detailed the project management, experimental design, and paper design in order to explain how the successful implementation of the system was done.

The simulation validation process was important to test our theoretical design ideas and prove that there was merit to implementing them physically. In the validation of actual implementation, the comparison of two sets of results (simulation and physical) allowed for analysis of how effective the actual implementation was. From this comparison, it is clear to see that the project's actual implementation was highly effective and performed exceptionally well. The success of the project was further validated by the passing of all the ATP’s which were designed to test whether the project met the initial specifications. In this way, our project met all our initial specifications.

Despite the success of the project its scale was limited. Thus, this report also discussed how and what must be done to prepare this project for real world, large scale implementation.

Overall, this report shows how this project was conducted from start to finish, how its success was validated and tested and what must be done further to prepare it for a larger scale implementation.