EEE3097S 2023 MILESTONE 3: SECOND PROGRESS REPORT

ACOUSTIC TRIANGULATION USING MULTILATERATION AND GENERAL CROSS-CORRELATION WITH PHASE TRANSFORM

16 October 2023

Simon Carthew - CRTSIM008
Robert Dugmore - DGMROB001
Si Teng Wu - WXXSIT001

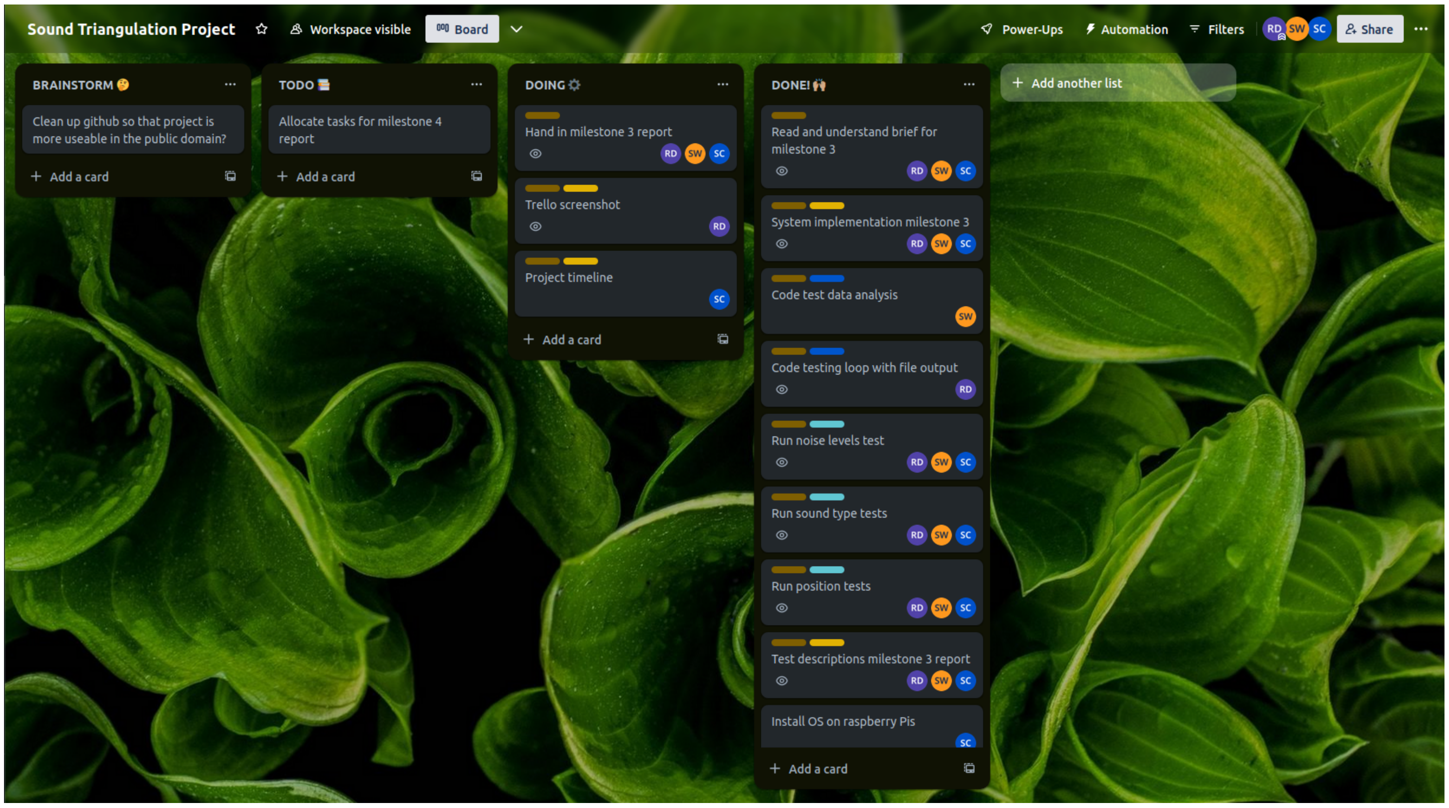
ADMIN DOCUMENTS

TABLE OF INDIVIDUAL CONTRIBUTIONS

The tasks listed below were only the initial tasks allocated to each person. Ultimately, the group met multiple times and gave extensive input to each other on all tasks listed.

Name	Sections
Simon Carthew	Admin Documentation, System Implementation [Signal Acquisition/PI Synchronisation], Experimental Setup and Data Collection [Signal Acquisition/PI Synchronisation], Results and Analysis [Signal Acquisition/PI Synchronisation], ATPs
Robert Dugmore	System Implementation [Hardware/GUI], Experimental Setup and Data Collection [TDOA/Overall/GUI], Results and Analysis [TDOA/Overall/GUI]], Conclusions
Si Teng Wu	System Implementation [TDOA/Triangulation], Experimental Setup and Data Collection [TDOA/Triangulation/Overall], Results and Analysis [TDOA/Triangulation/Overall], Improvements for Live Tracking

SCREENSHOT OF TRELLO PAGE



BRAINSTORM 🤔
- Clean up github so that project is more useable in the public domain?
- + Add a card

TODO 📚
- Allocate tasks for milestone 4 report
- + Add a card

DOING ⚙️
- Hand in milestone 3 report — RD SW SC
- Trello screenshot — RD
- Project timeline — SC
- + Add a card

DONE! 🙌
- Read and understand brief for milestone 3 — RD SW SC
- System implementation milestone 3 — RD SW SC
- Code test data analysis — SW
- Code testing loop with file output — RD
- Run noise levels test — RD SW SC
- Run sound type tests — RD SW SC
- Run position tests — RD SW SC
- Test descriptions milestone 3 report — RD SW SC
- Install OS on raspberry Pis — SC
- + Add a card

LINK TO GITHUB

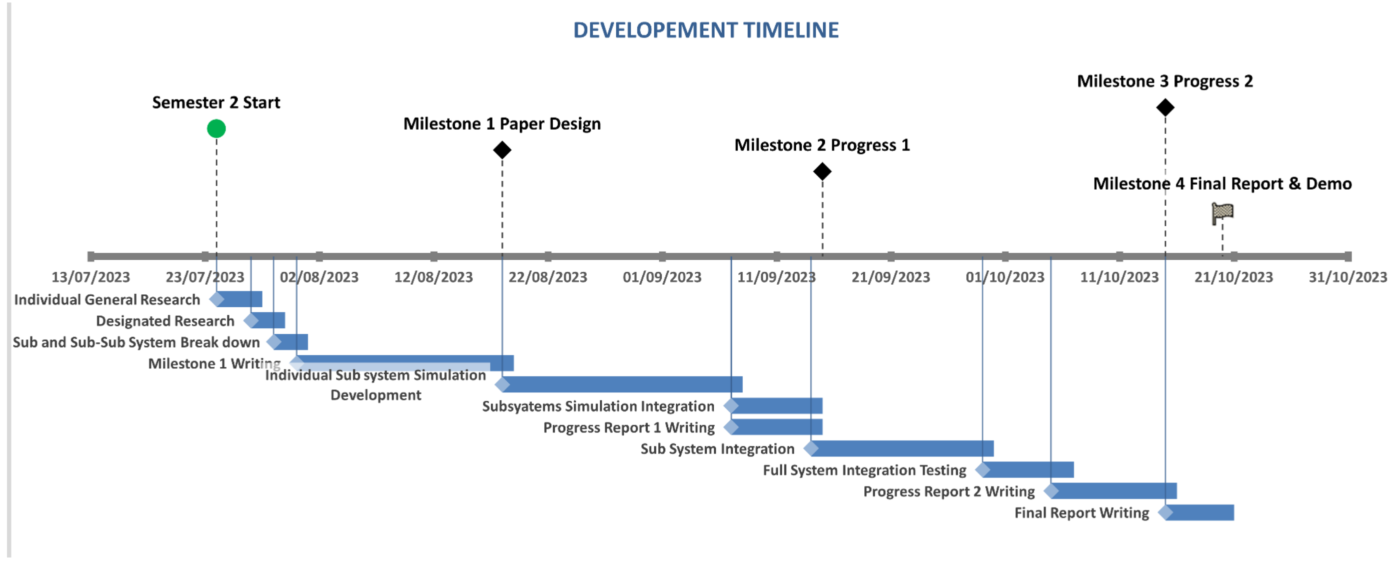
https://github.com/rothdu/EEE3097S

To start polling on the PI's, one must run the "python3 poll.py main rpi1 stereo" on one of the PI's and "python3 poll.py sec rpi12 stereo" on the second PI. The script "poll.py" is found in the "pi_side" directory within the "Main" directory.

Once polling on the PI's is set up, the script "gui.py", must be run on the local computer. Once the GUI is running, one can start real time tracking by pressing start, and stop by pressing pause. Plotting hyperbolas, changing to single-shot and changing microphone positions can all be done from the configuration window.

TIMELINE

Below is the updated project timeline:



The team has managed to keep up with all previously set goals. Some changes to the previous timeline are that the Milestone 3 deadline was shifted to the 16/10/2023 and the tasks completed before that were adjusted based on what was required for the demonstration. The Milestone 3 extension allowed for finer adjustments to be made to the tests done and the final report.

SYSTEM IMPLEMENTATION AND DATA COLLECTION

<u>SIGNAL ACQUISITION</u>

<u>DESCRIPTION OF IMPLEMENTATION</u>

Adafruit MEMS microphone

- The "arecord -l" command with extended parameters from the "i2smic.py" script was used to start stereo recording.
- Each PI had two microphones connected to it in stereo where each microphone was responsible for recording to one channel in a stereo wav file.

Recording

- During localisation, each PI is running a script called "poll.py".
- The "poll.py" script controls how and when each PI initiates a recording and sends a stereo wav file to the local computer.
- The PI's are set recording by calling the "arecord" command from the python script using a subprocess.
- The "arecord" function, in combination with a set of parameters, sets the microphones recording in stereo for a set amount of 9820 samples sampled at 44100 Hz.
- Each stereo wav file is saved to a file "<PI Name>_next_byte.wav".
- Once the recording is finished the file, the "poll.py" script calls an scp command using a subprocess, sending the wav file to the local computer.
- After sending the wav file to the local computer, each pi sends an empty text file, "<PI Name>_finished.tct", to the local computer, informing the local computer that a new sound byte has finished transferring to the computer.
- When exactly the PI's start recording is discussed further in the PI synchronisation subsystem implementation.

Signal Processing:

- Once the local computer has received a new byte of sound in the form of the two wav files from each PI, it will then begin processing the bytes.
- All four channels are extracted from the wav files from each PI into 4 individual numpy arrays.
- The left channels of both wav files are the reference microphones recordings and the right channels are the x and y axis microphones recordings from pi 1 and pi 2 respectively.
- These channels are extracted as explained above by the "readSignal()" function in the "localize.py" script.
- Once extracted, all the signals are passed through a band pass filter with cut offs at 200 and 20000 Hz.
- The low cut off frequency removes the low frequencies from the consistent popping noise at the start and throughout the recordings produced by the microphones.
- The high cut off frequency removes any high frequency noise components.
- After the bandpass, the first 1000 samples are removed from all arrays to get rid of the impulse at the start of the signal produced by the bandpass filter
- At this point all the signals are ready to be passed to the TDOA subsystem.

MODIFICATIONS AND IMPROVEMENTS

Network Transfer

- Transferring the data over the usb peripheral port on the PI's is no longer supported with the recent versions of raspberry pi os.
- This means that to transfer the data over the USB, one must connect the UART pins to a UART to USB chip and from that to a USB cable.
- After testing this method it proved to be much slower than expected.
- Transferring over the network using "scp" was investigated as an alternative method of transfer. It was found to be much faster, reliable and required much less overhead.
- Thus, the new method of sound byte transfer was decided to be SCPíng over the network.

CHALLENGES AND HOW THEY WERE ADDRESSED

Set Recording Time

- Getting the microphones to record continuously and extract sound bytes from the continuous recording was challenging due to the lack of resources.
- The adafruit website provided extensive documentation and steps on how to get the mics to record for set periods of time.
- The sound bytes were thus recorded for a set amount of samples rather than continuous and were sent as wav files instead of arrays of values.
- This method turned out to be better for the newly chosen method of network transfer.

SCP

- It became evident that the "scp" command was designed to be called from the local computer but it was necessary to call the command from the remote PI's as the computer has no way of telling when the last recording has finished.
- This issue was resolved by using port forwarding to forward the scp command back to the local machine that originally ssh'd into the PI instead of to a specific IP address.
- SCP'ing files require passwords to be entered every time which would slow the system down dramatically. Public keys set up.

CONSIDERATION FOR EXTENDED SIGNAL PROCESSING (LIVE TRACKING)

The system is designed to wait for an instruction to record a new sound byte as soon as the previous byte has been transmitted. This allows the PC to instruct the Pis to record new sound bytes at a specific sample frequency, or as fast as the system is able to achieve. Once the scripts used for signal acquisition are running on the Pis, the Pis will continue to wait for instructions to record data until the scripts are manually terminated, or the Pis are powered off.

Experimentation showed that the largest delay in the live tracking system came from transferring the signals to the PC. This time can be effectively reduced by decreasing the size of the sound bytes. Thus, the bytes were kept as short as possible, but within a range that produced acceptable accuracy.

Tests were also performed with transferring the files via the Pi's UART ports on the GPIO header, which connected to the PC via a USB-UART converter. This proved to be substantially slower than network transfer, and hence the wired setup was discarded.

PI SYNCHRONISATION

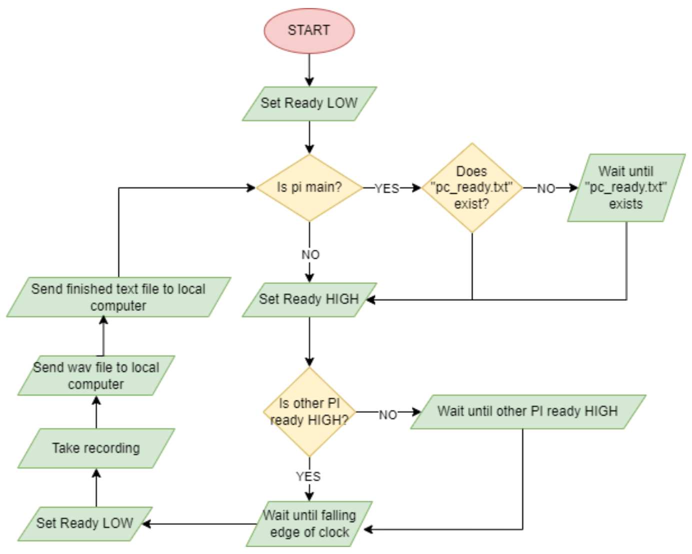
DESCRIPTION OF IMPLEMENTATION

To ensure that the PI's remain synchronised, they need to take recordings only when the local computer "asks" for a new sound byte, on the falling edge of a clock signal and when the other PI has finished recording and sent its last byte. Below are the separate parts that ensure that this happens:

Poll Script:

- During localization, both PI's run the python script "poll.py". They can either run the script as the main pi or the secondary pi which is determined by the parameter entered when the script is run.
- The "poll.py" script maintains synchronisation by using a GPIO pin as a "ready" signal that is set high when the PI is finished recording and transferring and set low when it is busy recording and transferring.
- Each PI takes in the other PI's ready pin as an input and sends its own ready pin as an output to the other PI.
- It also makes use of a 2 Hz PWM signal that is used as a general clock signal to dictate the exact time both PI's start recording when they are both ready.
- The clock signal is generated by one PI and connected to one of its own GPIO pins and one of the other PI's GPIO pins.
- Each PI will only start recording when both PI's read pins are set high and it is falling edge on the clock signal.
- In addition to all of these conditions, the PI that is configured as the main PI will be set waiting and "unready" until it receives an empty text file "pc_ready.txt" from the local computer that signals it to set its own ready pin high. This ensures that recordings will only happen when the PC "asks" for a new sample.
- Once the main PI receives the "pc_ready.txt" file, it will delete it after to ensure a new recording only happens after the local computer sends the file again to "ask" for a new sound byte.
- The function used to "ask" for a new byte is the "inform_ready()" function found in the "next_byte.py" script which sends the text file to the main PI.

Diagram of Synchronisation Flow


START
Set Ready LOW
Is pi main?
YES
Does "pc_ready.txt" exist?
NO
Wait until "pc_ready.txt" exists
NO
Set Ready HIGH
Is other PI ready HIGH?
NO
Wait until other PI ready HIGH
YES
Wait until falling edge of clock
Set Ready LOW
Take recording
Send wav file to local computer
Send finished text file to local computer

Possible PI States:

- Waiting for the other PI to be ready.
- Busy recording and transferring.
- Waiting for the “pc_ready.txt”

MODIFICATIONS AND IMPROVEMENTS

Local Computer Trigger

- The original plan was to have the PI’s send continuous numbered wav files. The issue with that is that the local computer would have to search through a directory for the most recent wav file which is unnecessary overhead. It would also make it slightly more difficult to know which is the most recent wav file that has finished transferring.
- It was decided to not continuously send files from the PI’s but rather have the local computer trigger the PI’s to start recording when it is ready to process a new sound byte.
- By doing this, the local computer will ot become overloaded with files that it will have to search through every time to process the next sound byte.
- In addition to this, the PI’s will send a “<PI Name>_finished.txt” indicating that they have finished sending the last wav file and that the local computer may begin processing it.

CHALLENGES AND HOW THEY WERE ADDRESS

Time Delay

- The PI’s were being triggered by the same falling edge but the time it took them to actually call the “arecord” command in a subprocess and start recording was unpredictable and inconsistent.
- The solution was to only compare recordings across the stereo mics connected to each PI with both PI’s having their own reference microphone. This results in drawing two intersecting hyperbolas instead of three.
- After testing, the time delay between PI’s was estimated to be in the millisecond range which meant that although the stereo recordings from each PI were slightly delayed from each other, the hyperbolas that they resulted in still corresponded to where the source was currently.

Transfer Time

- The time it takes for each PI to transfer the data to the local computer, whether using USB or network, was much longer and unpredictable than expected.
- The original design was to have the PI’s record and transfer the data within one clock cycle. Due to the limits of data transfer speeds, it was found that this is not possible.
- The chosen form of transferring, network, was also very unpredictable. Meaning that the PI’s take different amounts of time to send the data to the local computer.
- This meant that one Pi may finish transferring data before the other, resulting in it recording on the next falling edge of the clock while the other PI is still transferring.
- This issue was solved using the ready pins explained above. The use of the ready pins meant that both PI’s will only start recording on the next falling edge when both PI’s are ready.

TIME DIFFERENCE OF ARRIVAL (TDOA)

DESCRIPTION OF IMPLEMENTATION

Time Difference of Arrival (TDOA) is the difference in time of which the signals arrive at the microphones. In this implementation, the TDOA is measured as the time shift in the signals taken by microphones, relative to a reference signal; from a reference mic. A negative TDOA means the source is closer to the reference mic than the helper and vice versa.

The TDOA algorithm used is GCC-PhaT: Generalised Cross Correlation with Phase Transform. GCC-PhaT is a time delay estimation algorithm that extends the base cross correlation method by multiplying the Cross Power Spectral Density function by a weighting function, PhaT, before calculating the time delay. By doing so, the GCC-PhaT function will be left with only phase information, producing a delta function at the estimated delay while "pushing" other components outwards[1][2].

GCC-PhaT Algorithm

1. Given two signals $x_1(t)$ and $x_2(t)$ where $x_1(t)$ is the reference mic signal.
2. The Fourier Transform of each signal is taken, giving $X_1(\omega)$ and $X_2(\omega)$.
3. The conjugate of $X_2(\omega)$ is taken, giving $[X_2^*(\omega)]$
4. The Cross Power Spectral Density is calculated using:

$$G_{x1x2}(\omega) = X_2(\omega)[X_2^*(\omega)]$$

5. The weighting function is calculated and multiplied with the CPSD:

$$\psi_{PhaT}(\omega) = \frac{1}{\left|X_2(\omega)[X_2^*(\omega)]\right|}$$

$$G_{x1x2-PhaT}(\omega) = \frac{X_2(\omega)[X_2^*(\omega)]}{\left|X_2(\omega)[X_2^*(\omega)]\right|}$$

6. Inverse Fourier Transform is taken giving the GCC-PhaT function:

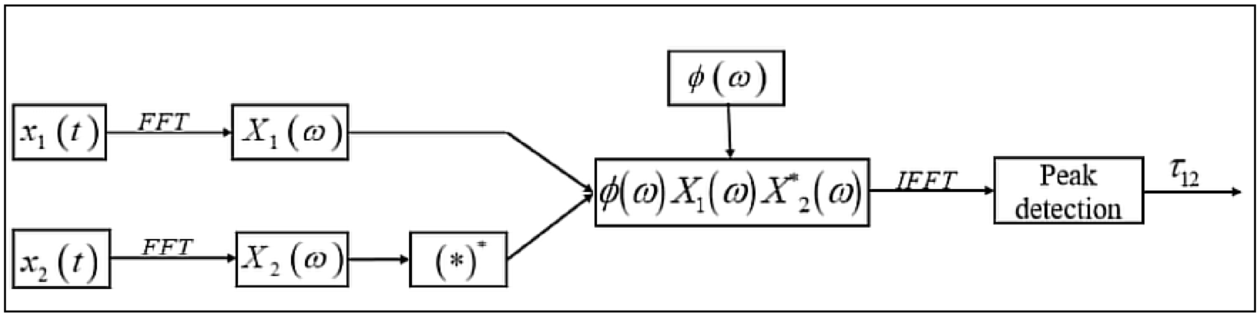
$$R_{x1x2-PhaT}(d) = IFFT\left\{\frac{X_2(\omega)[X_2^*(\omega)]}{\left|X_2(\omega)[X_2^*(\omega)]\right|}\right\}$$

7. The delay is the position of the highest peak in the GCC-PhaT functions:

$$d = argmax\left\{R_{x1x2-PHAT}(d)\right\}$$

Equations adapted from [1][2].

The following flow diagram demonstrates the algorithm. Taken from [3]



GCC-PhaT Code

A Python script, “gcc_phat.py”, written by Yihui Xiong was used [4]. The script contained a function called “gcc_phat” that takes in a [reference signal, signal, sample rate, max TDOA, interpolation value]. The function uses the same algorithm described above but also implements interpolation, scaling by sample rate and a max TDOA.

- Interpolation pads the same value x times, x = interpolation value. This lengthens the signal without changing its properties and increases the accuracy of the FFT, thereby increasing the accuracy of the TDOA.
- Scaling by sample rate is needed to convert the final value found using the argMax() into a time, since discrete values have a known time division unlike a continuous signal.
- Max TDOA limits the possible TDOAs the functions can return.

Execution In Code

The ‘gcc_phat’ function is used in the ‘localize.py’ script using the filtered signals provided by the Signal Acquisition subsystem.

1. A Max_TDOA of 10ms, interpolation value of 16 and a sample rate of 44 100 Hz was used.
2. Two TDOAs are calculated, one for each Pi.
3. TDOAs are filtered for 0 TDOAs since this is likely a failure.
4. TDOAs are used for triangulation and returned to GUI for further use.

MODIFICATIONS AND IMPROVEMENTS

Omitted Features

- The paper design included Kalman filtering to improve TDOAs for moving signals. This was not implemented since the source movement speed was limited to a very low speed and the update speed of the system was fast enough to be considered live. The plotted live points were also accurate to an acceptable degree, as shown in the Live Experiment.
- The paper design included a calibration mode. This was not implemented due to implementations of manual hardware calibration. Explanations can be found in the GUI subsystem.

Improved TDOAs

- In the previous design and in simulations, three TDOAs were calculated due to there being three pairs of microphones. The current implementation only calculates two due to the change in microphone configurations. This is further explained in the Triangulation subsystem.
- A longer sample time was recorded in order to help GCC-PhaT produce better results. The 0.1s or 4410 samples of recording used in simulation were found to produce constant errors but 9820 samples were able to produce a consistent result.

CHALLENGES AND HOW THEY WERE ADDRESSED

Large Delays using microphones in four corners

- A small but significant delay between the actual recording times of the mics from each Pi due to the calling of the microphones' function was discovered. This is explained in detail in the Pi Synchronisation subsystem.
- The delay would cause the TDOA to be incorrect because of the added delay between the two Pis.
- This delay was not found between stereo channels of the same Pi since the microphones from the same Pi would record simultaneously.
- The solution was to change the microphone configuration such that the right audio channel of each Pi was used as the reference mic (in the same position) and the respective left channels were used as the helper microphones. This produced two rather than three TDOAs

"Pop" at the start of the recording

- A consistent TDOA was found by the GCC-PhaT algorithm of approximately 1-6 µs between signals due to a "pop" or impulse artefact produced by the microphones at the start of every recording.
- This consistent TDOA represented the time delay between the stereo mics' starting times, which was evidently very small.
- The solution was to filter the signal by cutting out this artefact as well as using other techniques described in the Signal Acquisition subsystem. This allowed for GCC-PhaT to find the correct TDOAs

CONSIDERATIONS FOR DYNAMIC TIME DELAY ESTIMATION (LIVE TRACKING)

Each time a new sound snippet was recorded, the algorithm would analyse a new set of of TDOAs and use these to triangulate and plot a new point.

The TDOA algorithm as described is effective for slowly moving signals. No further additions were used for the live tracking system, as the system did not aim to track fast-moving signals.

TRIANGULATION

DESCRIPTION OF IMPLEMENTATION

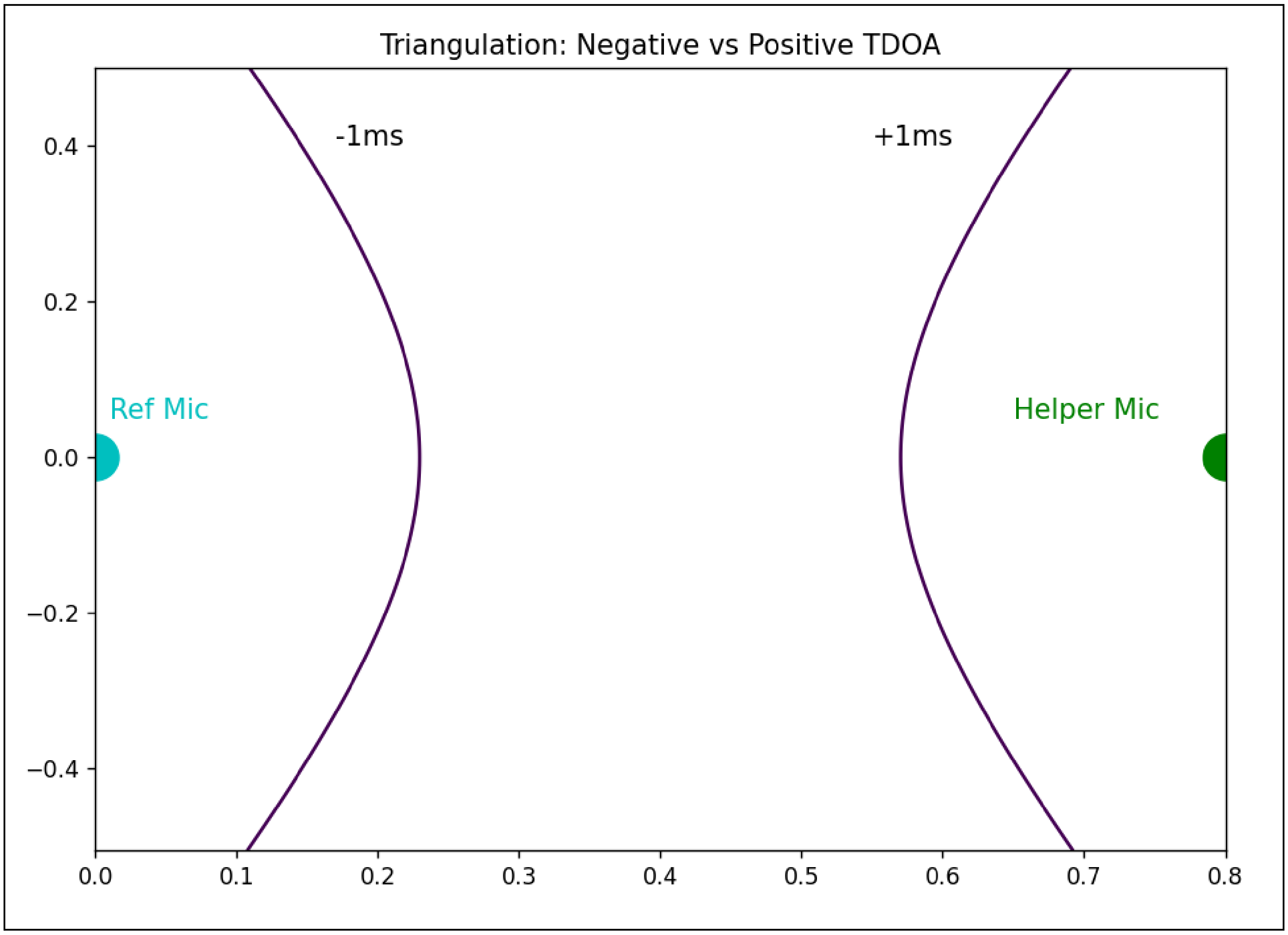
The triangulation formula uses the TDOA to produce a hyperbola along the axis of a pair of mics, one being a reference mic and the other a helper mic. The current implementation uses two mic pairs to produce two hyperbolas, of which an intersection between the two is found and used as the estimate of the source position.

Triangulation Formula for One Mic Pair

The triangulation formula is based on the distance formula for two points.

$$\sqrt{(x-x_1)^2 + (y-y_1)^2} - \sqrt{(x-x_2)^2 + (y-y_2)^2} = c(\Delta t)$$

where x, y is the estimated source position, x_1, y_1 is the reference mic position, x_2, y_2 is the helper mic position , c is the speed of signal propagation and Δt is the TDOA [5]. A negative TDOA produces a hyperbola concave towards the reference mic and a positive TDOA produces hyperbolas concave towards the helper mic.



Execution in code

The triangulation formula is used in the localize.py script.

1. TDOAs are multiplied by the speed of sound producing the right side of the formula. The constant from SciPy: scipy.constants.speed_of_sound was used as the speed of sound.
2. Two functions were defined using x and y variables representing the triangulation formula, one function for each mic pair.
3. The "scipy.optimize.fsolve" function from SciPy was used to solve the intersection of the two functions. The "scipy.optimize.fsolve" requires a starting point to solve the functions, of which x = 0.4, y=0.25 was used since it is the centre of the grid.
4. Hyperbolas are generated using a meshgrid and returned if requested by GUI
5. The intersection of the hyperbolas is returned as the estimated location of the source

Filtering of Answers

Certain answers or exceptions/warnings are filtered out and caught to ensure "good" estimations are being returned to the GUI.

- The triangulation algorithm isn't executed if GCC-PhaT fails
- Estimations outside the grid are discarded
- Warnings produced when "fsolve" cannot find an intersection are caught. Such warnings are produced due to a bad TDOA.

MODIFICATIONS AND IMPROVEMENTS

Omitted Features

- The paper design included calculations of uncertainties to show a heat map in GUI and display an error bubble around the estimated position. Due to time constraints and the feature not being necessary for the core system function, this feature was not included in the current implementation.

Change in Microphone Configuration - three to two hyperbolas

- In the simulations, three hyperbolas were constructed and solved in three pairs, resulting in up to three unique intersections. That would then require using the midpoint theorem to solve for the centre of the intersection to produce a best estimation of the source location. This was using the four corners configuration.
- Explained in Pi Sync and TDOA, the mics were changed such that there were effectively three mics, instead of four therefore only two hyperbola can be constructed. These hyperbolas' axes are orthogonal to each other and given "decent" TDOAs, only one unique solution can occur within the grid.
- The triangulation process was thinned down significantly since only 1 unique solution was possible and no midpoint theorem is needed.

New and Faster Non-Linear Solver

- The previous non-linear solver, “sympy.solve” from SymPy, was replaced with the current non-linear solver “scipy.optimize.fsolve” from SciPy.
- “Sympy.solve” solved for all possible solutions, produced complex results and took approximately 5 to 10 seconds to find the intersections.
- “Scipy.optimize.fsolve” produces a single solution, no complex solutions for real functions and found the correct intersection in 1ms.
- An increase in speed of 10000 times.

Simplified Filtering of Answers

- Filtering of complex solutions and multiple solutions no longer needed.
- Simply filtering for solutions inside the grid only.
- Catching a warning resulting from the solver being unable to find a solution.
- Not running code if an error is found, saving some time.

CHALLENGES AND HOW THEY WERE ADDRESSED

Approximation Starting Point and Warnings

- “Scipy.optimize.fsolve” requires a parameter specifying the starting point of the approximation. The algorithm solves the intersection by moving along the functions until the distance between the points gets smaller.
- After a certain threshold, if the distance becomes small enough, a solution will be returned, otherwise if no progress is made, a warning will be thrown and wherever the algorithm was will be returned in the solution.
- In early tests, the point x=0, y=0 was used as the starting point but later, this was found to be inconsistent as the algorithm would go the wrong way. It was found that the best point was x=0.4, y=0.25, which is the centre of our grid.
- In Python, warnings do not exit the program but do get printed and are a clear sign that something went wrong. Try-except does not work since it is not an exception. The solution was to convert the warning to an exception and catch that.

HARDWARE

DESCRIPTION OF IMPLEMENTATION

Microphones were placed on 3 corners of the grid, with 2 “reference” microphones placed on the corner between the other two microphones.

Microphones were connected to the I2S ports of the Pis using generic copper signal cable, with each Pi connected to 2 microphones in a stereo configuration.

3 GPIO lines were connected between the Pis, which are used for the Pis to share a common clock signal and to indicate ready status to each other, enabling Pi Synchronisation.

Each Pi was powered via a USB cable.

MODIFICATIONS AND IMPROVEMENTS

Microphone positions:

The paper design specified that the four microphones should be placed at the four corners of the grid. In simulation, this configuration was tested, along with an alternate configuration with all four microphones in a line on a single axis.

For the physical implementation, an alternate setup was used, as described in the previous section. Due to the constraints mentioned with Pi Synchronisation, only two TDOAs were calculated in the final system. It was thus possible to place two microphones together on one corner of the grid, which allowed for the synchronisation delay to be estimated by finding a TDOA between these two microphones at the same location.

Microphone jig:

The paper design specified the creation of a jig which held each of the microphones. For the final implementation, it proved simpler to use tape to stick the microphones to the grid.

Wiring:

The paper design specified the use of UTP wiring to connect the microphones to the Pis. This was replaced with simple signal wiring, which was more readily accessible and cost effective.

CHALLENGES AND HOW THEY WERE ADDRESSED

Poor connection to microphones:

The sound output from the microphones initially included a number of unwanted artefacts. These artefacts were reduced by carefully soldering the connecting wires, to ensure stable hardware connections between the Pis and the microphones.

GRAPHICAL USER INTERFACE (GUI)

DESCRIPTION OF IMPLEMENTATION

The GUI is built in python using the GUI library *PySimpleGUI* and the plotting library *MATPLOTLIB*.

Main page:

The main page is where the core operation of the program exists, including the primary control of the system and the graphical display of the plotted points.

The system operates either in continuous sampling mode, or single-shot mode as determined by the configuration settings. In continuous sampling mode, the system will continuously update as a specified frequency, or as fast as possible if the specified update frequency cannot be achieved. In single-shot mode, the user must prompt the system each time they want to take a new reading.

The main page includes:

- A “canvas” on which a *MATPLOTLIB* plot is shown.
- An Update Time text field
    - Displays the time taken to capture and analyse the most recent sound byte
- A Synchronisation Time text field
    - Optionally displays the synchronisation delay between the two Pis, found by assessing the TDOAs of the two reference mics
- A Start/Stop/Resume button
    - Acts as a start / pause button in continuous sampling mode. Acts as the start button for a single test in single-shot mode.
- A Config button
    - Takes the user to the configuration page
- A Tests button
    - Takes the user to the tests page
- An output message
    - Delivers informative feedback about what is happening on the GUI. Messages indicate when:
        - A new point has been plotted
        - An error has occurred with plotting a new point (the error message is displayed)
        - Systems settings have been updated
        - System has been started/paused/resumed

Configuration page:

The configuration page contains various options which control how the system behaves.

- Selection between continuous sampling and single-shot modes
    - Changes the mode of operation of the core system, as described above.
- Selection to plot hyperbolas

    - If selected, the main system plots the hyperbolas used for triangulation as well as the final estimated location of the sound source.
- Selection to show synchronisation delay
    - If selected, the system will display the estimated synchronisation delay between the two Pis, estimated by finding a TDOA between the two reference microphones (which are at the same position).
- Selection for dummy mode
    - If selected, the system will plot random points, rather than reading real data.
- Option to update sampling frequency
    - Used to change the update frequency in continuous sampling mode. Can be entered as a frequency in Hz or a period in ms.
- Option to enter new microphone positions
    - Used to enter the positions of the microphones, if the hardware configuration is changed.

Tests page:

The tests page includes a number of tests which are used to demonstrate the operation of the individual subsystems. This page is designed to give a user a graphical demonstration of the subsystems working individually, and was not directly used for the formal test procedures described in the sections below.

- Pi synchronisation test
    - User can prompt the Pis to simultaneously capture the current time, system displays the difference between the two times
- Signal acquisition test
    - User can prompt the system to capture and stores a new sound recording
    - Shows plots of the recorded data from each of the 4 mics, before and after filtering
- TDOA test
    - User inputs an x and y position and places a sound source at that position
    - System calculates the theoretical TDOAs from the inputted position, measures the actual TDOAs from the sound source in the physical system
    - System outputs the percentage difference between the theoretical and actual values
- Triangulation test
    - User inputs an x and y position
    - System calculates theoretical TDOAs based on the position and runs them through the triangulation algorithm
    - System outputs the percentage difference between the input and output positions
    - System can optionally display a plot showing the output position and the hyperbolas used for triangulation

Use of multithreading:

The system uses multithreading to allow the GUI interface and the system capturing and analysing data to run concurrently on separate threads. Since most modern PCs have multiple cores and thus can take advantage of multithreading, this allows two key advantages:

- Improved performance for each of the two subsystems (GUI and capture/analysis), since they will face less competition for shared computational resources
- No unnecessary delays on one system while the other is operating - GUI can continue to check for user input while capturing and analysing data in the background

MODIFICATIONS AND IMPROVEMENTS

Heatmap:

The paper design specified the inclusion of a heatmap as an alternative method for viewing the location of the sound source. The heatmap would show the likelihood of the sound source being in specified areas based on the degree of uncertainty in the measurements.

This feature was omitted, as the final physical implementation did not include extensive calculations of uncertainties, beyond those described in the testing section of this report.

Soft-configurable grid size:

The paper specified an option for a soft-configurable grid size.

This feature was omitted, as the physical system used only one grid size, so soft-configuration was not necessary

Calibration procedure:

The paper design specified the inclusion of prompts for a calibration procedure.

This feature was converted to the “microphone position” option as described above. The physical system used inputted microphone positions and relied on hardware calibration (i.e., ensuring that the microphones were correctly placed), rather than including software calibration.

Output to file:

The paper design specified the inclusion of an option to export data to a file.

It was decided that data output to files (i.e., for testing and validation purposes) could be more appropriately implemented in its own dedicated script, which could allow the programmer advanced control over the choice of and formatting of relevant data.

Subsystem tests:

The individual subsystem tests, as described in the previous section, were not included in the paper design. These features were added such that the system could show a graphical representation of the subsystems working individually.

Dummy mode:

"Dummy mode" was not included in the paper design. This feature was added as a means to test and validate the operation of the GUI that was not reliant on the physical system.

CHALLENGES AND HOW THEY WERE ADDRESSED

GUI freezing while capturing and analysing data:

The GUI implementation requires that the script continuously check for user input. If the script is halted or delayed while waiting for data capture, transfer, or analysis, the GUI "freezes" and stops responding to user input.

This was solved by implementing the GUI in a separate thread to the data capture/analysis. Short tasks (such as plotting the output points) are implemented in the GUI's thread, while longer tasks (such as data capture/analysis) are outsourced to a separate thread.

Confusing and cluttered interface from excessive items on GUI:

As more functionality was added to the GUI, it became confusing and cluttered.

To resolve this, the separate GUI pages were created for the configuration and subsystem tests.

CONSIDERATIONS FOR SOURCE LOCALIZATION UPDATES (LIVE TRACKING)

As described previously, the system includes a soft-configurable update frequency. The system attempts to capture and analyse a new sound recording at the interval specified by this update frequency. If the system is unable to match the update frequency, it instead updates as fast as possible.

The regular update frequency is implemented using timers from the Python time library, which are able to run functions after specific periods of time.

CONSIDERATIONS FOR VISUAL REPRESENTATION (LIVE TRACKING)

In continuous update mode, the graphical representation is updated every time a new sound recording is captured and analysed. The graphical representation is as previously described - a point on a plot, with the option to also plot the hyperbolas used for triangulation. Additionally, the GUI outputs the time taken to capture and analyse the latest sound recording in a dedicated text field.

EXPERIMENTAL SETUP AND DATA COLLECTION

PI SYNCHRONISATION - TIMESTAMP

This test tests the time delay between the two PI's being triggered to record using timestamps.

EXPLANATION OF SETUP FOR EXPERIMENTS

Both PI's are set running a python script "test_sync.py" as either the main or secondary PI. This script operates exactly as the "poll.py" script does but it transfers a text file with the current time instead of taking a recording. The local computer runs a script called "sync_test_time.py" which repeatedly calls "inform_ready()" from the "next_byte.py" script to get 10 timestamps from each PI. The absolute difference between these timestamps is saved to a text file and the average difference is printed to the end of the text file.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

- test_sync.py - this script is run on both PI's during testing
- sync_test_time.py - this script is run on the local computer to initiate and execute the testing

RATIONALE BEHIND EXPERIMENT DESIGN

This experiment was run to try and get an idea of the time delay between the two PI's execution of recording. This metric is important to test whether the PI synchronisation is working and that the PI's are at least starting recording on the same falling edge of the clock signal.

STEPS TAKEN TO ENSURE ACCURACY

To ensure that the time delay calculated is not influenced by any other factors, a separate script, "test_sync.py" was written to be run on the PI's instead of integrating the sending of the timestamp into the original "poll.py" script. Integrating the "poll.py" script would require conditions that may cause other unexpected delays.

PI SYNCHRONISATION - REFERENCE MIC TDOA

This test tests the time delay between the two PI's being triggered to record by finding the TDOA between two reference microphones. The reason that the TDOA between the two reference microphones should give the synchronisation time delay is because they are located at roughly the same position and thus any time delay would only be caused by a synchronisation error.

EXPLANATION OF SETUP FOR EXPERIMENTS

Both PI's are set running a python script "poll.py" as either the main or secondary PI and a constant volume sound is played directly into the reference microphones. The local computer runs a script called "sync_test_tdoa.py" which repeatedly calls "inform_ready()" from the "next_byte.py" script to get 10 sound bytes from each PI. At each iteration of getting a new sound byte, the script extracts only the first channel (the reference microphone recordings) from both wav files, converts them to numpy arrays and filters them. It then finds the TDOA between these two arrays and saves them to a text file. After all 10 iterations, it finds the average TDOA and prints it to the end of the text file.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

- sync_test_tdoa.py - this script is run on the local computer to initiate and execute the testing
- Test signal - Finger Snap Sound Effect FX - YouTube [6]

RATIONALE BEHIND EXPERIMENT DESIGN

The experiment was run to try and get an idea of the time delay between the two PI's execution of recording. This metric is important to test whether the PI synchronisation is working and that the PI's are at least starting recording on the same falling edge of the clock signal. It was run in addition to the "timestamp" PI synchronisation test as it alone may not be a good isolated test of the PI synchronisation test as it relies heavily on the TDOA subsystem working correctly.

STEPS TAKEN TO ENSURE ACCURACY

To ensure that the test was accurate, a continuously loud sound was played into both reference microphones and was played from the other side of the grid to ensure that any difference in their position was negligible.

SIGNAL ACQUISITION - FILTERING

This test was done to compare the actual recorded signals to the signals after the filtering. This comparison was done to ensure that the popping produced at the start of the microphone had been removed by the filtering and that there was no persistent popping throughout the recordings.

EXPLANATION OF SETUP FOR EXPERIMENTS

Both PI's are set running a python script "poll.py" as either the main or secondary PI and a sound is played from a random point in the grid. The local computer runs the "gui.py" script and executes a single shot capture. After a new sound byte has been captured, the "signal_aqu_test.py" script is run. This script plots all four original unfiltered signals and all four filtered signals and saves them as two respective figures.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

- "Signal_aqu_test.py" - this script is run on the local computer after the sound byte is captured to plot the relevant signals

RATIONALE BEHIND EXPERIMENT DESIGN

This experiment was run to ensure that the microphones were recording the sounds correctly without any persistent popping sounds and that the popping at the beginning of the recording was removed by the filtering. It often occurred that when resetting up the system, some wire connections could have been weakened or loosened causing the microphones to pop. This test allowed for quick and easy validation that the microphones were not popping.

STEPS TAKEN TO ENSURE ACCURACY

To ensure that the signals being recorded were accurate, the sound used was played as loud as possible from the middle of the grid to ensure that all the microphones received a similar signal.

SIGNAL ACQUISITION - RECORDED VS ACTUAL

This test was done to compare known signals to the signals retrieved after filtering. This comparison was done to ensure that the signals being recorded correlated with the source signal.

EXPLANATION OF SETUP FOR EXPERIMENTS

Both PI's are set running a python script "poll.py" as either the main or secondary PI. The "poll.py" script is configured to record for 15 seconds. The local computer runs the "gui.py" script and executes a single shot capture. During this 15 second capture period a 10 second known signal is played from the centre of the grid. Once this 15 second sound byte has been captured, the "signal_aqu_test.py" script is run. This script plots all four original unfiltered signals and all four filtered signals, both with the original source signal plotted below and saves them as two respective figures.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

- "Signal_aqu_test.py" - this script is run on the local computer after the sound byte is captured to plot the relevant signals

RATIONALE BEHIND EXPERIMENT DESIGN

This test was run to ensure that the sound that the source signal matched that of the signal being recorded and that there were minimal discrepancies between the recorded sound and the actual sound.

STEPS TAKEN TO ENSURE ACCURACY

To ensure that the full test sound signal was captured by the sound byte, the "poll.py" script was configured to capture a 15 second sample. Using a longer source signal and sound capture time than that of the regular system execution is essential to be able to see the unique acoustic properties found in the test source signal. This allows one to identify if the original source signal has been captured correctly.

TRIANGULATION - SIMULATED TDOA

This experiment tests the triangulation formula using ideal TDOAs.

EXPLANATION OF SETUP FOR EXPERIMENTS

1. A list of source positions scattered throughout the grid is used to find the theoretical TDOAs between the sources and the mics pairs.
2. The theoretical TDOAs are used in the triangulation formula and the original source location for each position is estimated.
3. The results are tabulated and the distance from the estimated point to the actual position is compared against the max distance in the grid and taken as a percentage.
4. This percentage error must be less than **1%** to count as a Pass.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

“triTest.py” is used to run this experiment.

RATIONALE BEHIND EXPERIMENT DESIGN

The aim of this experiment is to isolate the triangulation formula and check its effectiveness and accuracy, given theoretically perfect TDOAs. Our implementation uses two algorithms that introduce uncertainty, GCC-PhaT and the triangulation formula. By ensuring TDOAs and therefore GCC-PhaT is ideal, we can determine to what extent failures in the overall system can be attributed to inaccuracies or failures in the triangulation formula.

STEPS TAKEN TO ENSURE ACCURACY

- All constants are calculated in code and carried over into calculations.
- No manual values are entered.
- “scipy.constants.speed_of_sound” is used for the propagation velocity of sound through air.
- “Exact scipy.optimize.fsolve” parameter settings to the current implementation were used.

GUI - ALL BUTTONS BEHAVING AS EXPECTED

EXPLANATION OF SETUP FOR EXPERIMENTS

A revised list of desired GUI features was created, as described in the GUI system implementation subsection.

For each feature, the buttons were clicked and tested to check that behaviour was as expected.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

N/A

RATIONALE BEHIND EXPERIMENT DESIGN

The core aim of the GUI is to provide appropriate features to users in a way that is accessible and easy to understand. This test ensures that the appropriate features are present. Since the GUI does not create any quantitative data, a simple qualitative assessment is all that is required to ensure that the features are present and behaving as desired.

STEPS TAKEN TO ENSURE ACCURACY

In addition to the specific test of the GUI, the continued use of the GUI program for demonstration purposes allowed feedback on the performance of the GUI in its intended scenario for use. This was used as a form of ongoing testing which allowed bugs in the program to be eliminated.

While production-level applications might include more extensive bug testing, this level of bug testing was deemed sufficient for the level of the application made.

GUI - QUALITATIVE USER FEEDBACK

EXPLANATION OF SETUP FOR EXPERIMENTS

Each group member provided qualitative feedback on the GUI design. The GUI was demonstrated to the EEE3097S tutors in the system demonstration, providing additional qualitative feedback.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

N/A

RATIONALE BEHIND EXPERIMENT DESIGN

The core aim of the GUI is to provide appropriate features to users in a way that is accessible and easy to understand. This test ensures that the application is accessible and easy to understand. Since the GUI does not create any quantitative data, a simple qualitative assessment is all that is required to ensure that the features are present and behaving as desired.

STEPS TAKEN TO ENSURE ACCURACY

In addition to the specific test of the GUI, the continued use of the GUI program for demonstration purposes allowed feedback on the usability of the GUI in its intended scenario for use. Feedback from group members using the GUI showed what additional features were missing from the application, so that these features could later be added.

TDOA - SIGNAL TYPES

EXPLANATION OF SETUP FOR EXPERIMENTS

The microphones were placed at positions (0.8, 0.0), (0.0, 0.0) and (0.8, 0.5), with the first position being the location of the two reference microphones.

The sound source was placed at position (0.5, 0.15). This was chosen because it is not directly between either of the microphone pairs so should have non-negligible TDOAs, and is well inside the grid.

10 sounds were captured and analysed by the system for each of the following signal types:

- 300 Hz Sine Wave
    - 300 Hz selected as it is above the cut-off frequency used to filter the signal, but still low enough for correlation to theoretically generate only one peak at the specified sound position
- 300 Hz Square Wave
    - 300 Hz selected as it is above the cut-off frequency used to filter the signal, but still low enough for correlation to theoretically generate only one peak at the specified sound position
- 100 Hz to 400 Hz Modulating Sine Wave
- Drumbeat [7] The same boring drumbeat for 1 hour straight
    - Repeating drumbeat is consistent enough for testing, but has a wider and more varied frequency spectrum than the sine/square waves
- Nyan cat song [8] Nyan Cat "Nyan"s for 1 Hour (Super Extended Version!) [HD]
    - Repeating drumbeat is consistent enough for testing, but has a wider and more varied frequency spectrum than the sine/square waves
    - Additionally, has more consistent volume levels than the drumbeat

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

N/A

RATIONALE BEHIND EXPERIMENT DESIGN

Testing in the simulation phase revealed that our implementation of the GCC-PHAT algorithm behaved unexpectedly for perfect sine waves, but was more consistent for recordings with a wider frequency spectrum.

This test varies the test signal, while keeping other factors (sound position and background noise levels) as constant as possible.

STEPS TAKEN TO ENSURE ACCURACY

Sound source position was kept constant. System was tested in a home environment with as little background noise as possible. 10 sounds were captured for each sound type to find average system performance.

TDOA - SOUND SOURCE POSITIONS

EXPLANATION OF SETUP FOR EXPERIMENTS

The microphones were placed at positions (0.8, 0.0), (0.0, 0.0) and (0.8, 0.5), with the first position being the location of the two reference microphones.

The sound source was placed at 19 unique positions:

- The centre of the grid - (0.4, 0.25)
- The corners of the grid - (0.0, 0.0), (0.0, 0.5), (0.8, 0.0), (0.8, 0.5)
- The edges of the grid - (0.4, 0.0), (0.4, 0.5), (0.0, 0.25), (0.8. 0.25)
- Other locations in the grid - (0.1, 0.4), (0.3, 0.4), (0.5, 0.4), (0.7, 0.4), (0.1, 0.1), (0.3, 0.1), (0.5, 0.1), (0.7, 0.1), (0.15, 0.3), (0.65, 0.3)

For each position, 10 recordings were captured and analysed. The drumbeat was used as the signal.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

N/A

RATIONALE BEHIND EXPERIMENT DESIGN

Since the system has microphones at only 3 corners, it is useful to analyse the performance at various locations in the grid. Additionally, it is useful to analyse “boundary conditions”, such as the sound source being at the centre, corners, or edges of the grid.

STEPS TAKEN TO ENSURE ACCURACY

System was tested in a home environment with as little background noise as possible. 10 sounds were captured for each sound type to find average system performance.

TDOA - BACKGROUND NOISE LEVELS

EXPLANATION OF SETUP FOR EXPERIMENTS

The microphones were placed at positions (0.8, 0.0), (0.0, 0.0) and (0.8, 0.5), with the first position being the location of the two reference microphones.

The sound source was placed at position (0.5, 0.15). This was chosen because it is not directly between either of the microphone pairs so should have non-negligible TDOAs, and is well inside the grid.

A speaker was mounted on a tripod above the centre of the grid, from which sound was played to emulate background noise. White noise was used as the background noise. [9] White Noise 1 Hour Long

The best performing signal type (i.e., the Nyan Cat Song) was used as the signal.

A smartphone was placed in the centre of the grid. The sound level from the sound source was measured. The same smartphone was used to measure the sound level from the "background noise" speaker. A Signal-to-Noise Ratio (SNR) was then determined by comparing the sound level of the signal and the emulated background noise.

The SNR was varied in a range from -10 dB to 10 dB by varying the level of the background noise. At each chosen SNR, 10 recordings were captured and analysed.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

Speaker mounted on a tripod above the grid, from which noise of varying levels was played.

Smartphone to measure sound levels.

RATIONALE BEHIND EXPERIMENT DESIGN

The speaker was mounted above the centre of the grid to emulate typical background noise from distant sound sources, which would arrive at the speakers at roughly the same time.

The method used to measure SNR aims to roughly measure the ratio of signal sound level to background sound level in the grid. This may not be a completely accurate representation of the SNR recorded by each microphone, since this would differ for each microphone, and is dependent on other factors such as:

- Distance from sound source
- Internal microphone noise levels
- Noise levels from circuitry

This test aims to provide the best possible emulation of varying background noise with the equipment available.

White noise was chosen as the background noise, as it theoretically contains equal power across the frequency spectrum.

This test does not emulate the presence of other signals in the vicinity of the system, such as people talking. These types of signals are too varied to comprehensively test. Furthermore, in general, the system should read these as “alternate signals” rather than “noise”, and should generally only be picked up if they are substantially louder than the actual signal. Furthermore, these signals would typically occur outside of the grid, which the system would identify as an error.

This test varies the background noise level while keeping other factors (sound position and signal type) as constant as possible.

STEPS TAKEN TO ENSURE ACCURACY

Sound source position was kept constant. System was tested in a home environment with as little background noise as possible. 10 sounds were captured for each sound type to find average system performance.

OVERALL SYSTEM - SIGNAL TYPES

EXPLANATION OF SETUP FOR EXPERIMENTS

The overall system test was run concurrently with the TDOA test described previously. For this test, the data was run through the entire system instead of just the TDOA subsystem. For each test, the output points were analysed against the known position of the sound source.

All other details are as described for the TDOA test.

OVERALL SYSTEM - SOUND SOURCE POSITIONS

EXPLANATION OF SETUP FOR EXPERIMENTS

The overall system test was run concurrently with the TDOA test described previously. For this test, the data was run through the entire system instead of just the TDOA subsystem. For each test, the output points were analysed against the known position of the sound source.

All other details are as described for the TDOA test.

OVERALL SYSTEM - BACKGROUND NOISE LEVELS

EXPLANATION OF SETUP FOR EXPERIMENTS

The overall system test was run concurrently with the TDOA test described previously. For this test, the data was run through the entire system instead of just the TDOA subsystem. For each test, the output points were analysed against the known position of the sound source.

All other details are as described for the TDOA test.

OVERALL SYSTEM - LIVE TEST

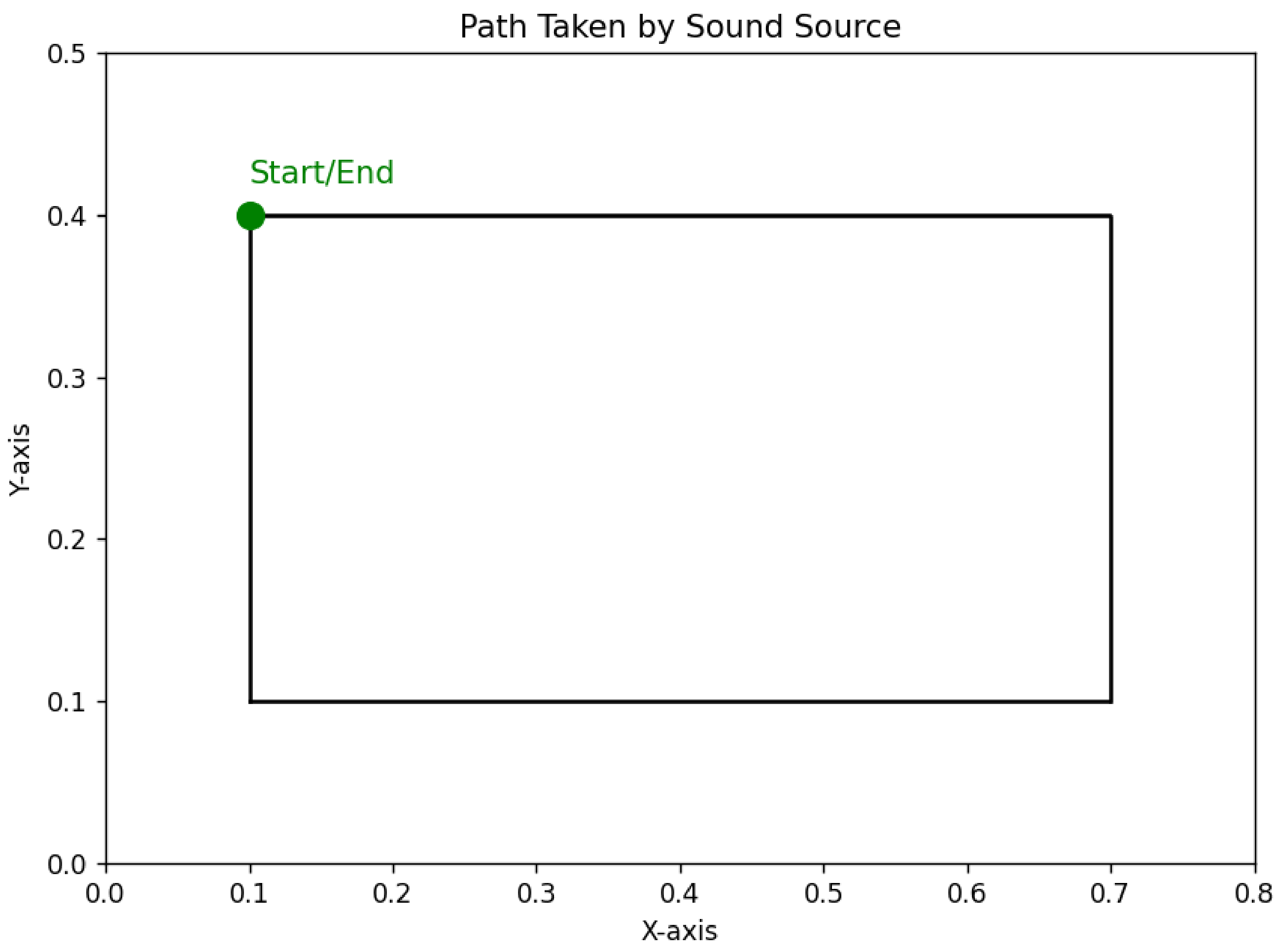
This experiment tests the "Live" RTS capabilities of our system. The aim of this experiment is to recreate a path taken by the sound source using estimated positions.

EXPLANATION OF SETUP FOR EXPERIMENTS

The microphones were placed at positions (0.8, 0.0), (0.0, 0.0) and (0.8, 0.5), with the first position being the location of the two reference microphones.

The sound source was placed at position (0.1,0.4) and the Nyan cat song was used.

The source was moved from along the following points: (0.1,0.4), (0.7,0.4), (0.7,0.1), (0.1,0.1), (0.1,0.4) forming a path of a rectangle at a speed of 0.03 m/s.



The system was allowed to capture data as fast as possible and write it to a csv file:

The data was then extracted, plotted and the average latency between data captures was outputted.

ADDITIONAL EQUIPMENT NEEDED FOR SETUP

N/A

RATIONALE BEHIND EXPERIMENT DESIGN

This experiment is kept very simple in order to demonstrate that the system is able to track a live moving object without manual interference. Once the system has started capturing data, the only manual interaction is the moving of the source which is external to the system. This will also show the performance of the combined subsystems.

STEPS TAKEN TO ENSURE ACCURACY

- All individuals present were to keep absolutely quiet.
- External noise was kept to a minimum

RESULTS AND ANALYSIS

PI SYNCHRONISATION - TIMESTAMP

TABLE OF RESULTS

Test No.	**Absolute Time Difference (Seconds)**
1	0.00258
2	0.00254
3	0.00255
4	0.00252
5	0.00252
6	0.00248
7	0.00247
8	0.00242
9	0.00243
10	0.00239
Ave	0.00249

COMPARISON TO SIMULATION

This test was not simulated as there was no “capturing” of the sound signal in the simulation as the signals for each microphones were generated from one wav file with specific delays added to each. The only delays being added to each microphone were those caused by the time taken for the source sound to arrive at the microphone rather than any delay caused by synchronisation errors.

ANALYSIS

From the above results we can see that the time difference of triggering is extremely consistent between tests. This shows that there is some consistent hardware delay between both PI’s. The clock is running at 2 Hz with a period of 0.5 seconds, so we can deduce that both PI’s are triggering on the same falling edge of the clock signal because the measured delay is below that of the clock signal's period. This consistent delay can be attributed to the time it takes both PI’s to get from being triggered by the falling edge to starting recording.

PI SYNCHRONISATION - REFERENCE MICROPHONE TDOA

TABLE OF RESULTS

Test No.	**Absolute Time Difference (Seconds)**
1	0.00036
2	0.01871
3	0.01496
4	0.01668
5	0.01856
6	0.01754
7	0.02134
8	0.02137
9	0.02093
10	0.07976
Ave	0.02302

COMPARISON TO SIMULATION

This test was not simulated for the same reason that the “signal acquisition timestamp” test was not simulated.

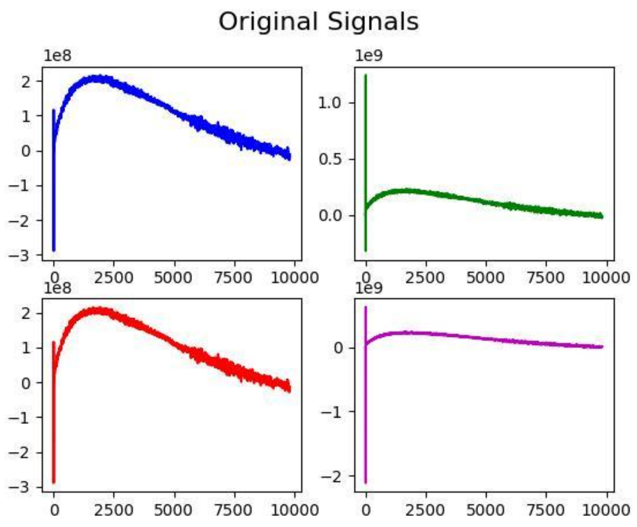
ANALYSIS

From the above results we can see that the time difference of triggering is relatively consistent between tests. This shows that there is some consistent hardware delay between both PI’s. As seen in the “signal acquisition timestamp” test, both PI’s are triggering on the same falling edge of the clock signal because the measured average delay is below that of the clock signal's period. This consistent delay can be attributed to the time it takes both PI’s to get from being triggered by the falling edge to starting recording. The time delay is slightly longer and less consistent than that of the “timestamp” test because the time it takes for the PI’s to call a subprocess is less consistent than that of taking a timestamp. The inconsistencies may have also arisen from failures related to the TDOA calculations. For this reason, the “timestamps” test can be considered a more accurate test of the synchronisation time delay.

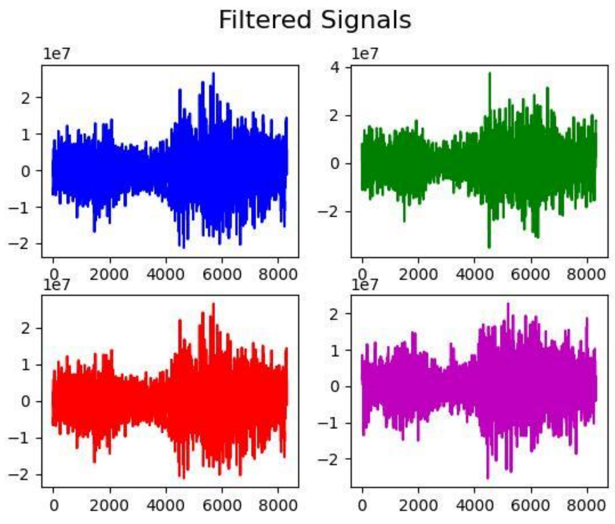
SIGNAL ACQUISITION - FILTERING

RESULTANT FIGURES

Below is the plotted sound byte signals without before filtering and cutoffs:



Below is the plotted sound byte signals without after filtering and cutoffs:



COMPARISON TO SIMULATION

This test was not simulated due to the fact that the signals that were used in the simulation were not recorded from a microphone and thus did not contain any unwanted frequency components added during the recording of the source signal.

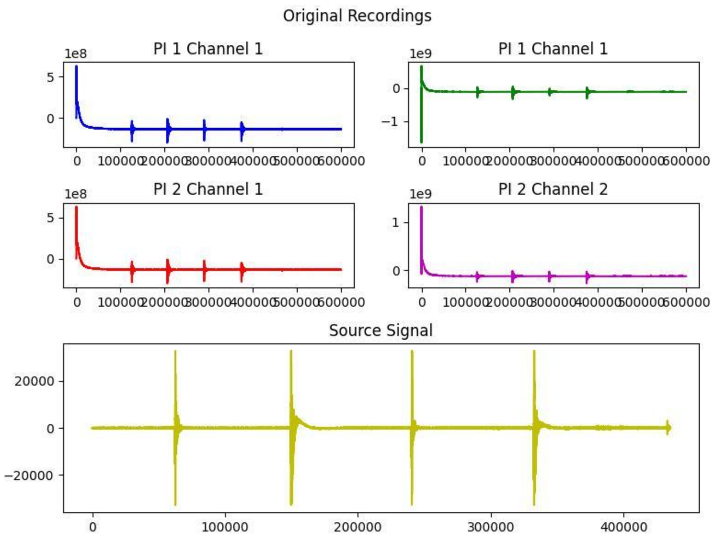
ANALYSIS

The figures from the test show that the initial pop is completely removed by the bandpass filter and the impulse at the start of the signal resulting from the bandpass filter is removed by the cut off. There is a clear correlation between the signal with little distortion which shows that the methods of filtering are successful.

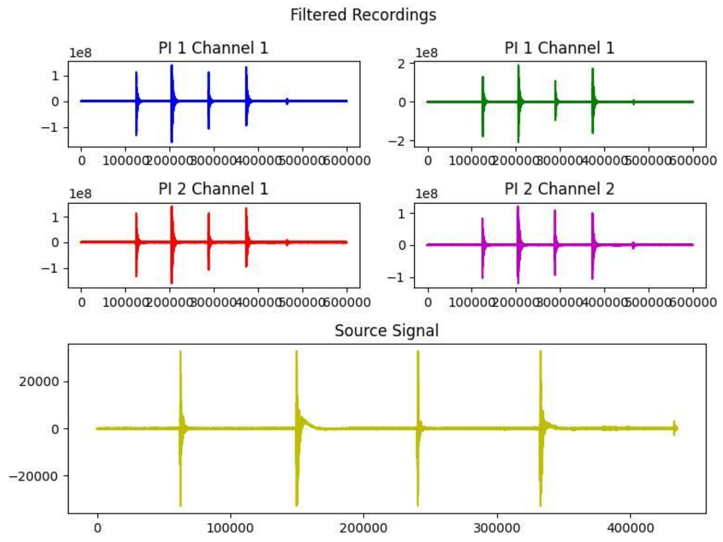
SIGNAL ACQUISITION - RECORDED VS ACTUAL

RESULTANT FIGURES

Below is the plotted unfiltered recorded signals and actual signal:



Below is the plotted filtered recorded signals and actual signal:



COMPARISON TO SIMULATION

This test was not simulated due to the fact that the signals that were used in the simulation were not recorded from a microphone. Thus the signal that was “captured” by each microphone was

ANALYSIS

From the above resultant figures it is clear to see that the original source signal was recorded successfully by all four microphones with little to no visible distortion. The source signal is both clear in the unfiltered and filtered signals but is far more visible in the filtered signals.

TRIANGULATION - SIMULATED TDOA

TABLE OF RESULTS

Test Pos	**Estimated Pos**	**Percentage Error**	**Pass/Fail**
0.1,0.1	0.10000000000000155,0.09999999999999437	6.19E-13	Pass
0.3,0.1	0.29999999999894017,0.10000000000282826	3.2E-10	Pass
0.5,0.1	0.5,0.09999999999999994	7.36E-15	Pass
0.7,0.1	0.7000000000019673,0.09999999999908986	2.3E-10	Pass
0.1,0.2	0.10000000000000003,0.19999999999999998	4.16E-15	Pass
0.3,0.2	0.29999999999999793,0.2000000000000033	4.1E-13	Pass
0.5,0.2	0.5000000000057143,0.19999999999224918	1.02E-09	Pass
0.7,0.2	0.7000000000000001,0.19999999999999996	1.32E-14	Pass
0.1,0.3	0.099999999999999,0.3000000000000009	1.42E-13	Pass
0.3,0.3	0.3000000000000006,0.299999999999999	1.24E-13	Pass
0.5,0.3	0.5000000000000008,0.29999999999999866	1.63E-13	Pass
0.7,0.3	0.6999999999999429,0.3000000000000657	9.23E-12	Pass
0.1,0.4	0.09999999999113385,0.4000000000109413	1.49E-09	Pass
0.3,0.4	0.3000000000002207,0.39999999999651253	3.7E-10	Pass
0.5,0.4	0.5000000000000012,0.39999999999999875	1.87E-13	Pass
0.7,0.4	0.700000000000709,0.4000000000020868	2.34E-10	Pass

Pass/Fail criteria: within 1% error of test position.

COMPARISON TO SIMULATION

The triangulation formula was not tested in isolation during simulations since the formula was mathematically sound and the previous non-linear solver, “sympy.solve” produced perfect results given perfect TDOAs.

The simulations for the whole system passed all tests with low error rates. However, the non-linear solver has been changed to “scipy.optimize.fsolve” and the produced results are not exact, so it was decided that triangulation may introduce some error to the final result.

ANALYSIS

The above table shows the original Test position, the estimated x,y positions, the percentage error and whether that specific position passed the test, i.e. less than 1% error. The estimated positions were accurate to the tenth significant figure at the very least. Percentage error was well below 1% by a minimum of an order of magnitude of -9. Lastly, it can be noted that all tests passed.

It can be concluded from these test results that the triangulation algorithm, if given correct TDOAs will effectively find the original source location with near perfect accuracy and consistency. The triangulation algorithm also has no effect on the estimated value and effectively all error in the final estimated value is entirely caused by the TDOAs produced by GCC-PhaT.

GUI - ALL BUTTONS BEHAVING AS EXPECTED

TABLE OF RESULTS

Functionality	**Present in GUI**	**Behaves as expected**
Plotting of points	Yes	Yes
Start/Pause/Resume Button	Yes	Yes
Continuous and Single-Shot Modes	Yes	Yes
Plotting of Hyperbolas	Yes	Yes
Display of Synchronisation delay	Yes	Yes
Soft-configurable sampling frequency	Yes	Yes
Dummy mode (displays random points)	Yes	Yes
Setting microphone positions	Yes	Yes
Pi Synchronisation Test	Yes	Yes
Signal Acquisition Test	Yes	Yes
TDOA Test	Yes	Yes
Triangulation Test	Yes	Yes

COMPARISON TO SIMULATION

The simulation used only a simplified GUI that produced various plots relating to the simulated results. Functionality for the final system is much more extensive.

ANALYSIS

Although the implemented functionality differs from the paper design, as described in the system implementation section, all necessary functionality for the final system is present and behaves as expected.

GUI - QUALITATIVE USER FEEDBACK

RESULTS

All group members approved of the final implementation of the GUI. EEE3097S course tutors and teaching assistants gave positive feedback during the system demonstration.

ANALYSIS

User feedback indicates that the system is adequate for the given implementation. If the system were converted to a production level application, ongoing feedback from users would allow for further refining of system functionality.

TDOA - SIGNAL TYPES

TABLE OF RESULTS

Test No.	**Percentage error with specified signal**									
	Drumbeat		**300Hz Sine**		**300Hz Square**		**100Hz to 400Hz Modulating Sine**		**Nyan Cat**	
	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2
1	304.73	-1.65	-19.98	-13.36	-19.78	-1.55	-19.68	-13.26	17.2	22.7
2	-0.8	-1.55	-1.93	-13.31	-19.68	-1.65	-5.2	-13.31	-0.7	-1.19
3	-0.7	-1.49	-19.68	-13.46	-40.85	-1.6	-19.78	-13.31	-0.7	-1.14
4	-24.74	-1.55	-19.73	-13.31	-19.78	-1.55	-19.68	-13.26	-0.9	-1.19
5	-0.85	-1.6	-37.63	-13.36	24.42	-1.7	-19.78	-13.31	-0.85	-1.24
6	-0.8	-1.6	164.42	-13.36	-0.7	-1.7	-0.85	-1.55	-1.06	-1.14
7	-20.65	-1.55	-19.83	-13.26	-18.91	-1.49	-0.9	-13.31	-0.96	-1.19
8	-30.68	-1.49	6.0	-13.21	1.4	-13.26	-19.78	-13.31	-0.44	-1.34
9	-0.8	-1.55	-21.62	-13.31	7.13	-1.6	-19.78	-13.31	-0.85	-0.93
10	-1.06	-1.8	-19.83	-13.46	-8.02	-1.6	-19.58	-13.26	-1.11	-1.34
P/F	Fail	Pass	Fail	Fail	Fail	Pass	Fail	Fail	Pass	Pass

COMPARISON TO SIMULATION

The final version of the simulation used only one type of sound source, thus the results are not directly comparable.

However, qualitative analysis during the design of the simulation showed inconsistent use of the GCC-PHAT algorithm with repeating signals such as sine waves; using a sound recording with a wider frequency spectrum yielded more consistent results. This is mimicked by the tests of the physical system, which fail for both the sine and square waves.

ANALYSIS

The results show that GCC-PHAT worked most effectively with continuous non periodic signals that have varying frequencies such as the Nyan Cat song, which is the only test that was able to pass the test criteria.

The drum beat is the second best signal with a 100% pass rate on TDOA2 and a 60% pass rate on TDOA1. The drum beats failure may be due to the fact that the drum beat is not consistent and due to the short sample time, an "empty" recording could have been made where there are not enough usable samples for GCC-PhaT to correlate.

The other 3 signals were all periodic by nature and did not perform consistently. This confirms the conclusion that GCC-PhaT does not work well with periodic signals or discontinuous signals.

The 100 Hz to 400 Hz modulating sine wave is negatively impacted by the filtering that the system performs during signal acquisition. The 200 Hz High-pass filter cuts out significant amounts of signal as it modulates below 200 Hz, which provides some explanation for this signal's poor performance in this test.

For many of the failed tests, only one of the two TDOAs failed to meet the specified criteria. Unfortunately, however, successful triangulation is reliant on both TDOAs being correct, so one of the TDOAs failing is still not ideal performance for the system overall.

TDOA - POSITIONS

TABLE OF RESULTS

Corners:

Test No.	**Percentage error at position**							
	0 0		**0 0.5**		**0.8 0.5**		**0.8 0.0**	
	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2
1	9.35	-1.39	1.47	0.83	-2.24	52.95	-14.52	-9.98
2	9.5	-1.29	47.92	1.03	-2.24	6.55	-13.8	-9.47
3	9.35	-1.44	1.42	0.88	-2.14	6.76	-13.49	-9.47
4	81.22	-14.33	1.42	0.78	-2.19	7.27	-9.04	-9.26
5	9.25	-1.49	1.53	0.88	-2.24	7.07	-13.54	-9.37
6	9.4	-1.54	-224.16	0.88	-2.24	7.01	-85.67	-53.05
7	9.35	-1.39	1.42	0.88	-164.85	7.12	-8.84	-7.58
8	9.3	-1.44	1.42	0.83	-2.29	7.07	-7.76	-7.42
9	9.25	11.55	1.47	0.88	-2.24	6.96	-7.76	-7.42
10	-102.93	-16.99	1.47	1.03	-2.29	6.86	-83.98	-7.32
P/F	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Pass

Pass/Fail criteria: 6+ /10 tests within 10% of theoretical TDOA.

Edges:

Test No.	**Percentage error at position**							
	0.0 0.25		**0.4 0.5**		**0.8 0.25**		**0.4 0.0**	
	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2
1	3.21	-0.72	0.0	1.12	-3.72	0.05	0.77	-1.69
2	3.21	-0.31	-222.72	1.07	-3.72	0.15	0.31	-1.84
3	3.16	-0.72	-0.05	1.07	-3.77	0.2	0.61	-1.69

4	3.11	-0.56	1.69	1.12	-63.98	0.2	0.77	-1.74
5	3.16	-0.15	-33.1	1.18	-3.77	0.51	0.0	-1.74
6	3.11	-0.2	0.1	1.18	-3.77	0.2	0.2	-1.74
7	3.21	-0.51	0.0	1.12	-3.77	0.15	0.77	-1.69
8	72.32	-0.46	-84.76	1.12	-3.72	0.15	4.19	-1.79
9	3.11	-0.56	0.0	1.07	-70.68	0.2	0.77	-1.69
10	12.16	-0.61	0.0	1.12	-3.77	0.31	0.15	-1.74
P/F	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass

Pass/Fail criteria: 6+ /10 tests within 10% of theoretical TDOA.

<u>Inside grid:</u>

<u>Test No.</u>	**<u>Percentage error at position</u>**									
	<u>0.1 0.4</u>		**<u>0.3 0.4</u>**		**<u>0.5 0.4</u>**		**<u>0.7 0.4</u>**		**<u>0.1 0.1</u>**	
	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2
1	57.56	0.12	0.04	0.52	-0.81	0.86	-2.37	2.37	3.64	-0.89
2	73.06	0.23	0.04	0.47	-0.96	0.86	-58.38	2.37	43.39	-0.89
3	1.55	0.17	-0.06	0.52	-0.76	0.86	-2.37	2.32	59.09	-0.99
4	1.55	0.23	-0.06	0.58	-18.04	0.86	-2.47	2.32	3.59	-0.94
5	1.45	11.43	16.46	0.58	-0.91	0.86	-34.29	2.32	106.26	-0.74
6	1.6	0.23	-1.34	0.47	-0.96	0.86	-2.42	2.32	3.64	-0.84
7	1.6	0.07	-0.06	0.58	21.34	0.86	-2.32	2.32	3.54	-0.94
8	-17.79	0.07	0.04	0.47	-0.96	0.86	-50.91	2.37	3.59	-0.84
9	1.55	0.02	-0.01	0.47	-0.96	0.81	-2.37	2.32	60.88	-10.51
10	1.55	0.23	0.04	0.47	-0.86	0.86	-40.94	2.32	3.59	-0.79
P/F	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass

Pass/Fail criteria: 6+ /10 tests within 10% of theoretical TDOA.

Test No.	**Percentage error at position**											
	0.3 0.1		**0.5 0.1**		**0.7 0.1**		**0.15 0.3**		**0.65 0.2**		**0.4 0.25**	
	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2
1	1.09	-0.68	-0.89	-1.73	-3.75	-1.86	1.86	-0.3	68.89	-0.05	0.0	-0.1
2	18.02	-0.63	-1.3	-1.78	-3.7	-1.86	1.76	-0.09	-2.31	-0.2	0.0	-0.72
3	20.53	-0.58	-0.89	-1.73	-3.7	-1.91	28.67	-0.4	-2.26	-0.1	-0.36	-0.31
4	1.14	-0.58	-0.84	-1.73	-3.7	-1.91	1.81	-0.4	-2.36	-0.1	-0.31	-0.51
5	1.09	-0.68	-46.21	-1.78	-42.57	-1.91	-220	-0.14	-38.94	-0.15	-6.85	-0.15
6	1.09	-0.58	-0.89	-1.73	-3.75	-1.91	1.86	-0.45	-2.31	-0.05	-0.41	-0.1
7	1.14	-0.63	-0.84	-1.73	-3.75	-1.91	1.81	-0.35	-2.36	-0.26	-0.1	-0.15
8	1.04	-0.58	-24.67	-1.73	-69.78	-1.91	39.51	-0.04	-2.26	-0.1	0.0	-0.61
9	17.1	-0.63	-0.84	-1.73	-3.75	-1.91	37.72	-0.24	-2.36	-0.26	-0.05	-0.46
10	0.99	-0.63	-0.89	-1.78	-3.7	-1.91	1.81	-0.35	-2.31	-0.26	-0.15	-0.1
P/F	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass

Pass/Fail criteria: 6+ /10 tests within 10% of theoretical TDOA.

COMPARISON TO SIMULATION

In the simulation, every test run on the system passed with an error of below 5%. The system performed less consistently for the physical tests. For the physical system, inspection of the results shows that a number of individual tests failed, which did not occur in simulation. Nevertheless, under the modified pass/fail criteria (6+ /10 tests passed), the performance of the physical system is still acceptable, even though it does not match the performance in simulation.

ANALYSIS

18/19 positions tested are able to correctly find the position of the sound source within 10% accuracy 60% of the time.

The only position for which the system fails is (0.8, 0.0). For the testing setup used, this was directly above the two reference microphones. This is thus a boundary case which should be noted for users of the system. It is also useful to note that the typical accuracy of the system for passed tests is lower when the microphone is at the corners of the grid, which are either directly on top of the microphones, or very far away from the microphones.

For the failed test, only one of the two TDOAs failed to meet the specified criteria. Unfortunately, however, successful triangulation is reliant on both TDOAs being correct, so one of the TDOAs failing is still not ideal performance for the system overall.

TDOA - BACKGROUND NOISE LEVELS

TABLE OF RESULTS

Test No.	**Percentage error at SNR**									
	+8 dB		**+4 dB**		**0 dB**		**-5 dB**		**-10 dB**	
	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2	TDOA 1	TDOA 2
1	-0.7	-1.03	-0.7	-1.03	-0.8	-1.19	-0.8	-1.03	-0.75	-1.19
2	-0.7	-1.03	-29.81	-1.24	-0.75	-1.24	-13.03	-1.19	-0.8	-13.31
3	-0.8	-1.24	-0.7	-1.03	-0.75	-1.14	-0.85	-13.31	-16.51	-1.19
4	-0.85	-1.24	-0.75	-1.14	-0.75	-1.14	-17.27	-1.14	-18.81	-13.21
5	-0.8	-1.24	-18.71	-1.24	-0.85	-1.19	-21.52	-13.31	-148.38	-13.31
6	-0.8	-1.24	-0.85	-1.19	-0.85	-1.19	-0.8	-13.31	-0.85	-13.31
7	-104.85	-1.24	-0.75	-1.19	-0.85	-1.08	-0.85	-1.14	-0.85	-13.26
8	-0.8	-1.14	-0.75	-1.19	177.31	-1.24	-0.8	-1.24	-19.73	-1.24
9	-0.85	-1.03	-0.96	-1.19	-0.7	-1.19	-0.8	-0.88	-19.73	-1.08
10	-0.7	-1.19	-0.9	-1.19	-0.8	-1.24	-1.01	-1.19	-0.75	-13.31
P/F	Pass	Pass	Pass	Pass	Pass	Pass	Fail	Fail	Fail	Fail

Pass/Fail criteria: 8+ /10 tests within 10% of theoretical TDOA.

COMPARISON TO SIMULATION

Tests with noise in simulation showed no difference between results with varying noise levels. This differs from the physical system test, which clearly shows that the system fails more frequently with higher levels of background noise.

This can largely be explained by the fact that simulation used random, uncorrelated noise at each of the microphones. Uncorrelated signals should have no impact on the GCC-PHAT algorithm, explaining the lack of changes to results in simulation.

Conversely, the emulated background noise in the physical test comes from the same source and thus may be correlated, hence its ability to affect the performance of the GCC-PHAT algorithm.

In a real application, some elements of the noise may be correlated, while others may be uncorrelated. It is useful to know that uncorrelated noise elements on each microphone should not affect the results, while correlated signals (such as a constant sound source in the general area of the microphones) may adversely affect the results.

ANALYSIS

The system shows strong performance with noise levels at or below the sound level of the signal. The system is able to correctly assess the TDOA more than 80% of the time, within 10% accuracy.

At higher noise levels, the system is no longer able to correctly assess the TDOA more than 80% of the time, and assessment of the system’s accuracy is thus less meaningful.

Noise levels do not appear to substantially affect the accuracy of the system, rather affecting the proportion of the time that the system fails to find the correct TDOA.

OVERALL SYSTEM - SIGNAL TYPES

TABLE OF RESULTS

Test No.	**Percentage error of final point with specified signal**				
	Drumbeat	**300Hz Sine**	**300Hz Square**	**100Hz to 400Hz Modulating Sine**	**Nyan Cat**
1	516.36	15.11	10.62	14.91	19.59
2	1.26	10.6	10.56	10.78	0.97
3	1.21	15.05	21.84	14.99	0.93
4	13.28	14.97	10.62	14.91	0.99
5	1.3	23.73	14.95	14.99	1.02
6	1.3	445095.64	1.38	1.26	0.98
7	11.09	14.98	10.16	10.62	1.0
8	16.45	11.57	10.75	14.99	1.09
9	1.26	15.78	4.6	14.99	0.8
10	1.47	15.11	4.32	14.87	1.13
P/F	Fail	Fail	Fail	Fail	Pass
AVG	N/A	N/A	N/A	N/A	0.99

Pass/Fail criteria: 8+ /10 tests within 10% of actual speakers position.

AVG considers only passed tests.

COMPARISON TO SIMULATION

The final version of the simulation used only one type of sound source, thus the results are not directly comparable.

However, qualitative analysis during the design of the simulation showed inconsistent use of the GCC-PHAT algorithm with repeating signals such as sine waves; using a sound recording with a wider frequency spectrum yielded more consistent results. This is mimicked by the tests of the physical system, which fail for both the sine and square waves.

ANALYSIS

The only test that achieved an accuracy within 10% for 80% of tests was the “Nyan Cat” song.

As described previously, the system appears to behave poorly for repeating signals such as a sine or square wave; signals with a wider frequency spectrum tend to perform better.

The 100 Hz to 400 Hz modulating sine wave is negatively impacted by the filtering that the system performs during signal acquisition. The 200 Hz High-pass filter cuts out significant amounts of signal as it modulates below 200 Hz, which provides some explanation for this signal’s poor performance in this test.

The final factor to be considered is signal volume. Qualitative analysis shows that the Nyan Cat song has more consistent volume levels than the drumbeat, which only has high volumes in short bursts. Since the system only analyses ~20 ms of audio data at a time, signals with periods of low volume can suffer if the system happens to record its data during a period of time when the signal volume is low.

The results of the overall system are similar to those of the TDOA test. This, combined with the very successful results from the triangulation test, indicate that the TDOA subsystem is the largest factor influencing the accuracy of the overall system.

OVERALL SYSTEM - SOUND SOURCE POSITIONS

TABLE OF RESULTS

Corners and edges:

Test No.	**Percentage error of final point at position**							
	0 0	**0 0.5**	**0.8 0.5**	**0.8 0.0**	**0.0 0.25**	**0.4 0.5**	**0.8 0.25**	**0.4 0.0**
1	5.41	3.76	30.23	7.57	2.68	1.42	3.66	2.31
2	5.42	44.93	364.75	7.19	2.9	4364625	3.7	2.37
3	5.45	3.8	420.66	7.07	2.64	1.34	3.76	2.27
4	47.78	3.58	4.73	55.63	2.65	2.41	43.36	2.37
5	5.43	3.93	4.65	7.07	2.98	35.89	3.86	2.17
6	5.54	1314670	4.63	50.46	2.88	1.52	3.76	2.22
7	5.41	3.8	913449.02	45.52	2.76	1.42	3.74	2.31
8	5.42	3.69	4.68	40.36	48.32	99404822	3.7	3.78
9	331.19	3.87	4.6	40.36	2.65	1.35	47.31	2.31
10	1388.25	4.2	4.58	63.7	10.28	1.42	3.79	2.21
P/F	Pass	Pass	Pass	Fail	Pass	Pass	Pass	Pass
AVG	5.44	3.83	4.64	N/A	2.77	1.55	3.75	2.43

Pass/Fail criteria: 6+ /10 tests within 10% of actual speakers position.

AVG considers only passed tests.

Inside grid:

Test No.	**Percentage error of final point at position**										
	0.1 0.4	**0.3 0.4**	**0.5 0.4**	**0.7 0.4**	**0.1 0.1**	**0.3 0.1**	**0.5 0.1**	**0.7 0.1**	**0.15 0.3**	**0.65 0.2**	**0.4 0.25**
1	42.78	0.67	0.87	2.52	2.32	1.07	1.47	2.08	1.3	191.43	0.1
2	53.07	0.61	0.93	50.6	23.53	9.72	1.53	2.05	1.28	1.48	0.68

3	1.71	0.65	0.85	2.5	32.05	11.05	1.47	2.05	19.27	1.46	0.36
4	1.77	0.71	13.34	2.57	2.33	0.99	1.47	2.05	1.26	1.53	0.52
5	18.7	12.15	0.91	27.41	59.51	1.07	24.2	24.42	157922	23.02	4.05
6	1.82	0.99	0.93	2.53	2.28	0.97	1.47	2.08	1.3	1.51	0.26
7	1.64	0.71	15.53	2.46	2.31	1.04	1.47	2.08	1.26	1.5	0.16
8	23.45	0.61	0.93	42.61	2.26	0.95	12.91	39.65	26.05	1.46	0.58
9	1.54	0.59	0.9	2.5	36.04	9.23	1.47	2.08	24.93	1.5	0.44
10	1.77	0.61	0.89	33.18	2.22	0.98	1.52	2.05	1.26	1.47	0.13
P/F	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass
AVG	1.71	0.68	0.9	2.51	2.29	2.89	1.48	2.06	1.28	1.49	0.73

Pass/Fail criteria: 6+ /10 tests within 10% of actual speakers position.

AVG considers only passed tests.

<u>COMPARISON TO SIMULATION</u>

The simulation test procedure had various differences to the current test procedure:

- The tests in simulation were done using the four corners mic configuration whereas this test was done with three corners mic configuration. The current implementation constructs two hyperbolas which have a single intersection that has to be taken as the estimated position. This would mean we can expect less accurate results compared to the simulation as a third hyperbola is no longer available to compare intersections with.
- A different triangulation algorithm was used in simulation vs the one used in the current implementation. However, it is shown in the Triangulation - simulated TDOAs test that the error of an estimated position is independent of the triangulation formula and entirely dependant on the TDOA
- In simulation, percentage error was calculated for the x and y positions separately, whereas the current implementation uses the percentage error based on the distance between the estimated and actual position. It is therefore inappropriate to directly compare percentage errors, but general comparisons can be made.
- A point was tested only once in simulation since the same value would be produced repeatedly, whereas the current implementation tests each position 10 times and determines the pass/fail based on the number of correct estimations.

In the simulation, every test run on the system passed with an error of below 5%. The system performed less consistently for the physical tests. For the physical system, inspection of the results shows that a number of individual tests failed, which did not occur in simulation. Nevertheless, under the modified pass/fail criteria (6+

/10 tests passed), the performance of the physical system is still acceptable, even though it does not match the performance in simulation.

As with the simulation, the system tends to be less accurate when the sound source is very close to the microphones or the grid axes, as seen from the first eight tests, which include the only tests with average accuracy of over 3%.

ANALYSIS

18/19 positions tested are able to correctly find the position of the sound source within 10% accuracy 60% of the time.

Though these results are imperfect, they are sufficient. Since the system can successfully find the position of the sound source more than 50% of the time, the position can accurately be determined at almost any position by running multiple tests and using the median result.

As concluded from the live tracking test, later in this report, the system is capable of running an individual test in less than 3 seconds. Running 10 tests thus typically takes around 30 seconds, which is still reasonably short, and is sufficient to get a result that is very likely to be accurate, by taking the median value of the 10 test results.

The only position for which the system fails is (0.8, 0.0). For the testing setup used, this was directly above the two reference microphones. This is thus a boundary case which should be noted for users of the system. It is also useful to note that the typical accuracy of the system for passed tests is lower when the microphone is at the corners of the grid, which are either directly on top of the microphones, or very far away from the microphones.

The results of the overall system are similar to those of the TDOA test. This, combined with the very successful results from the triangulation test, indicate that the TDOA subsystem is the largest factor influencing the accuracy of the overall system.

OVERALL SYSTEM - BACKGROUND NOISE LEVELS

TABLE OF RESULTS

Test No.	**Percentage error at SNR**				
	+8 dB	**+4 dB**	**0 dB**	**-5 dB**	**-10 dB**
1	0.85	0.85	0.98	0.87	0.98
2	0.85	16.01	1.01	7.01	10.62
3	1.02	0.85	0.94	10.62	8.89
4	1.02	0.94	0.94	9.31	14.52
5	1.02	10.07	0.99	15.73	155876.28
6	1.02	0.99	0.99	10.62	10.62
7	80.15	0.98	0.91	0.95	10.58
8	0.94	0.98	378.44	1.02	10.62
9	0.87	1.0	0.97	0.76	10.64
10	0.97	0.99	1.02	1.01	10.62
P/F	Pass	Pass	Pass	Fail	Fail
AVG	0.95	0.95	0.97	N/A	N/A

Pass/Fail criteria: 8+ /10 tests within 10% of actual speakers position.

AVG considers only passed tests.

COMPARISON TO SIMULATION

Tests with noise in simulation showed no difference between results with varying noise levels. This differs from the physical system test, which clearly shows that the system fails more frequently with higher levels of background noise.

This can largely be explained by the fact that simulation used random, uncorrelated noise at each of the microphones. Uncorrelated signals should have no impact on the GCC-PHAT algorithm, explaining the lack of changes to results in simulation.

Conversely, the emulated background noise in the physical test comes from the same source and thus may be correlated, hence its ability to affect the performance of the GCC-PHAT algorithm.

In a real application, some elements of the noise may be correlated, while others are uncorrelated. It is useful to know that uncorrelated noise elements on each microphone will not affect the results, while correlated signals (such as a constant sound source in the general area of the microphones) may adversely affect the results.

ANALYSIS

The system shows strong performance with noise levels at or below the sound level of the signal. The system is able to correctly assess the position of the sound source more than 80% of the time, within 1% accuracy.

At higher noise levels, the system is no longer able to correctly assess the position of the sound source more than 80% of the time, and assessment of the system's accuracy is thus less meaningful.

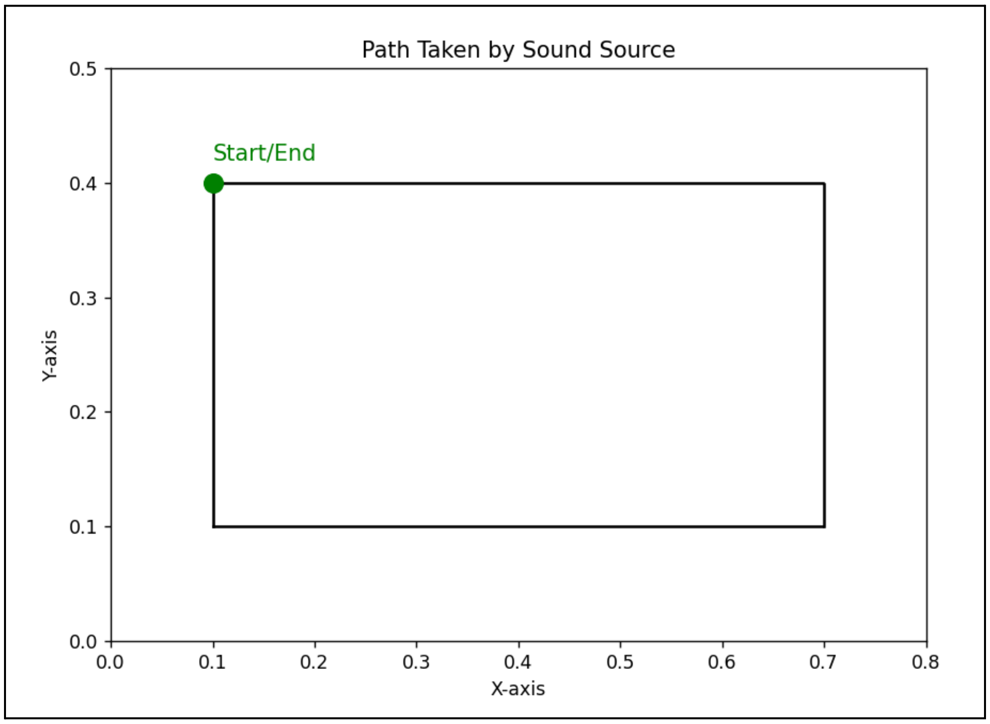
Noise levels do not appear to substantially affect the accuracy of the system, rather affecting the proportion of the time that the system fails to plot the correct point.

The results of the overall system are similar to those of the TDOA test. This, combined with the very successful results from the triangulation test, indicate that the TDOA subsystem is the largest factor influencing the accuracy of the overall system.

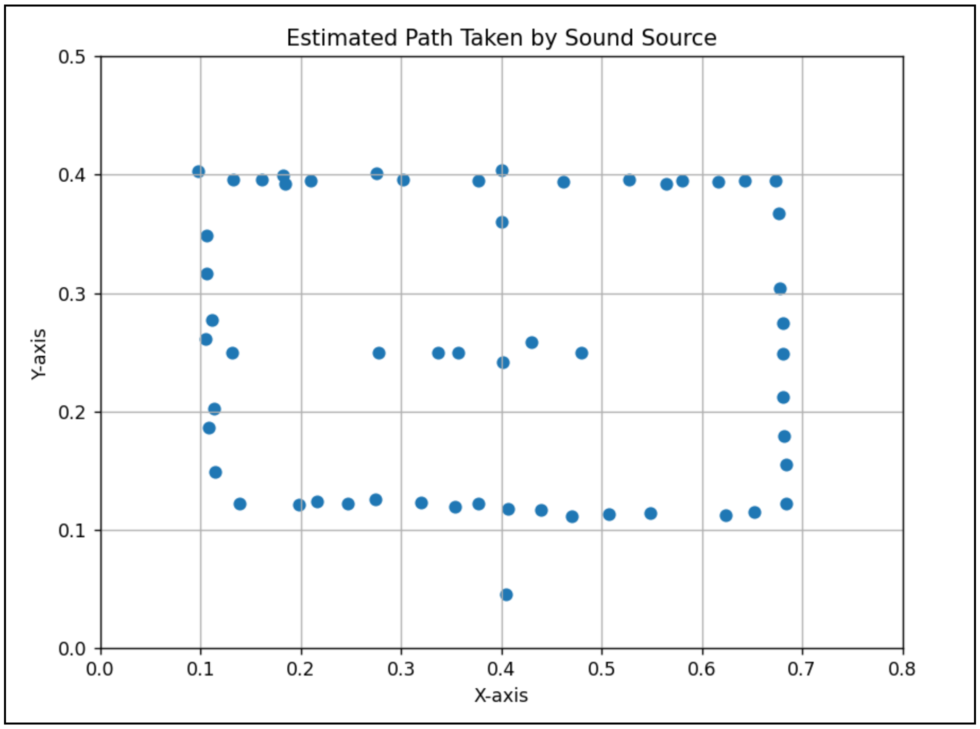
OVERALL SYSTEM - LIVE TEST

RESULTS

Below is a plot of the actual path taken by the source:



Below are the plotted points from the live tracking test:



COMPARISON TO SIMULATION

“Live” tests or moving source tests were not done in simulations. At the time, it was not certain whether live tracking would be achievable since we had yet to do any physical tests and implementations. Only stationary source tests were done in simulation.

ANALYSIS

Of the 58 captured test points, two were off the grid entirely, 8 were noticeably not on the path and 48 points can be considered reasonably on the path of the source. This means that 10/58 or 17.2% of the points failed. Comparing the above figures, it can be seen that the general shape of the estimated path matches the real path. From the output of the code, the average latency between data captures was 2.556 seconds.

The final implementation of the system includes more robust measures to identify and subsequently not show points that are not considered valid. As such, these incorrect points appear less frequently than in the results from this test, where all points were shown for completeness.

It can be concluded from the above tests that the current system is able to achieve live tracking and can be considered a real-time system, given that the source is moving at 0.03m/s. The system updates fast enough that the occasional failure to find a point is inconsequential, as the user can simply wait a few seconds so that a new, valid point can be found.

EVALUATION (ATPS)

ATPs

	Figures of merit	Test Procedure	Acceptable performance	Discarded/Altered /Unchanged/New
1	Same number of sound bites for all four microphones	Generate an array of timestamp data from the two Raspberry Pis over a number of interrupt triggers and see if the master and slave Pi output the same data in the arrays. This will show if the data collection will yield the same results as they are starting at the same time.	Four sound bites, no less, no more.	Discarded - Signals sent as a wav file recorded for a set amount of samples. No longer need to validate sound byte length
2	When both PI's are ready, they sample in the next clock cycle	"PI synchronisation" tests using timestamps and reference microphone TDOAs.	The calculated time delay is less than the period of the clock signal	Altered - Signals sent as a wav file recorded for a set amount of samples. No longer need to validate sound byte length
3	Clock signal is 2 Hz	Connect the output of the clock signal to an oscilloscope and measure the frequency.	Clock signal is between 1.8 and 2.2 Hz	Altered - Speed of clock no longer important as samples aren't taken every clock cycle
4	Microphones are able to read a 10 second sound source without distortion	Signal acquisition test "recorded vs actual" used to determine whether the source signal is present in the microphone recordings without any distortion.	Sound source must be visible in all four signals recorded from all four microphones.	Altered - better to test whether the source signal is actually being recorded without distortion than to test whether the bit rate is correct.

5	A numpy discrete sound magnitude array is correctly collected by each mic	Play a 280 Hz signal to each mic and capture a few periods. Send data and plot in Python on the main computer to check for a valid sine wave. Test to be conducted at the maximum and minimum distance to the mics in order to test for saturation or insufficiency.	The output is a valid sine wave and the difference in amplitude at max and min distance is within 10%	Discarded - No need to check the magnitude of the recordings as the magnitude of the recordings is strongly determined by where the sound source is. A small signal magnitude may not mean that the mic is not working but rather that the signal is far away/
6	Estimated TDOA matches theoretical TDOA	Tested in the “overall system - positions” test where a variety of source points are each tested 10 times and the percentage error of the TDOA for each test is calculated for each source point.	At Least 6/10 of the tests pass with a TDOA percentage error of less than 10%	Altered - test procedure altered for physical system and acceptable performance more realistic for real systems
7	Estimated TDoA is still accurate under noisy conditions	Tested in the “overall system - background noise levels” test where a variety of source points are each tested 10 times and the percentage error of the TDOA for each test is calculated for each source point.	At Least 6/10 of the tests pass with a TDOA percentage error of less than 10%	Altered - test procedure altered for physical system and acceptable performance more realistic for real systems
8	Estimated position from hyperbolas when source is moving at maximum speed of 0.1m/s.	Simulate a scenario of a moving signal and check if the algorithm is correct. Then create various TDOA samples with known points on the grid and compare results to actual position. Finally, move the sound source on known paths in the grid space and compare the estimated paths to the actual paths.	Accuracy is within 2% tolerance for x and y.	Discarded - no need to test the actual positions recorded when doing live tracking, acceptable if the path formed by live tracking resembles path taken.

9	Estimated x and y position from hyperbolas when source is moving at maximum speed of 0.1m/s.	Simulate a scenario of a moving signal and check if the algorithm is correct. Then create various TDOA samples with known points on the grid and compare results to actual position. Finally, move the sound source on known paths in the grid space and compare the estimated paths to the actual paths.	Accuracy is within 5% tolerance for x and y.	Discarded - no need to test the actual positions recorded when doing live tracking, acceptable if the path formed by live tracking resembles path taken.
10	Graphics updating correctly.	Feed dummy data to the UI and test if it correctly displays everything to check performance and accuracy. The test is passed if the data is correct, and appears within expected time.	Updates can occur at less than 1 fps independently of the time it takes to retrieve the position data.	Unchanged
11	User interface receives qualitative approval on the basis of useability from different people.	Ask others to try to use the UI and obtain feedback and qualitative approval	At least 5 different approval ratings.	Unchanged
12	All buttons on the user interface perform expected behaviour when clicked.	Click all the buttons on the interface and make sure that all the buttons perform the task that is expected.	Every button is checked and passes, no ghost buttons.	Altered - Method of testing altered
13	Estimated position matches actual position.	Tested in the “overall system - positions” test where a variety of source points are each tested 10 times and the percentage error for each test is calculated for each source point.	At least 6/10 of the tests pass with a percentage error of less than 10%	New
14	Estimated position is still accurate under noisy conditions.	Tested in the “overall system - background noise levels” test where a variety of SNR’s are each tested at the same point 10 times and the percentage error for each test is calculated for each source point.	At least 6/10 of the tests pass with a percentage error of less than 10%.	New

15	Path plotted by live tracked sound source resembles the real path with limited failures.	Tested in the “overall system - live test” test where the sound source is moved along a known path and the recorded positions are saved and plotted to generate a path taken by the live tracked object.	Only 20% of the plotted points are failures and the plotted path matches that of the actual path taken.	New
16	All distortions are filtered out from the acquired signals.	Tested in the “signal acquisition-filtered” test where the actual recordings are compared to the filtered signals.	All obvious distortions removed from the original recorded signal.	New
17	Triangulation estimates accurate positions given perfect TDOAs.	Tested in the “triangulation -simulated TDOAs” test where the triangulation formula is tested in isolation.	Estimated positions match actual positions with a percentage error of less than 1%.	New

Evaluations

ATP	PASS/FAIL	REASON/DISCUSSION	Improvements
2	PASS	It was found from both PI synchronisation tests that the estimated synchronisation time delay was well below the period of the clock signal. Thus both PI’s initiate recording on the same falling edge of the clock signal.	One could improve on this design by researching alternative ways of starting recording that are more consistent and less delayed. Resulting in a more consistent time delay that could be filtered out using a calibration system.
3	PASS	The PWM signal was measured using an oscilloscope and the measured frequency was 2.1 Hz which is within the acceptable performance metric.	N/A
4	PASS	The original test source signal is clearly visible in both the filtered and unfiltered recorded signals.	N/A
6	18/19 PASS	18/19 of the points had at least 6/10 of the tests pass with a TDOA percentage error of less than 10%. The only point that failed was the reference microphone position.	One could possibly improve this by testing different reference microphone positions.

7	3/5 PASS	System passed up to SNR = 0 dB and failed at all higher SNR’s	One could improve the systems performance at higher levels of SNR by investigating other types of filters that may be able to filter out the background noise.
10	PASS	GUI is able to update positions at a rate lower than 1 fps if dummy positions are used which makes it independent of the time taken to get the position data.	N/A
11	PASS	Five different people approved the GUI as interactive, responsive and intuitively designed.	N/A
12	PASS	All buttons perform the tasks that they are expected to perform.	N/A
13	18/19 PASS	18/19 of the points had at least 6/10 of the tests pass with a position percentage error of less than 10%. The only point that failed was the reference microphone position	One could possibly improve this by testing different reference microphone positions.
14	3/5 PASS	System passed up to SNR = 0 dB and failed at all higher SNR’s	One could improve the systems performance at higher levels of SNR by investigating other types of filters that may be able to filter out the background noise.
15	PASS	Failure rate of 17.2% < 20% Estimated path noticeably resembles real path	The main limitation is the delay between data capture caused by the Pis collecting data and sending it to the computer. By using a Pi with wired ethernet access, or directly using microphones connected to the computer, the system will be able to update much faster and more consistently produce correct results.
16	PASS	Popping sound present at the start of all of the original signals is clearly filtered out in the filtered signal	N/A
17	PASS	Triangulation formula produces perfect results given a correct TDOAs	N/A

POTENTIAL IMPROVEMENTS FOR LIVE TRACKING:

Kalman Filtering [10]

A Kalman filter is a type of α−β−(γ) filter that predicts/estimates the current estimated value based on the prediction of the previous value and a measurement input. The key feature of the Kalman filter that is useful for live tracking is its ability to predict the position of an object based on its velocity. This can be used to make estimations on the source location and adjust the TDOAs accordingly, or it can improve the final estimated value produced by triangulation. The challenge here is ensuring our real time measurements are correct and that not only completely wrong measurements are filtered but even somewhat incorrect measurements are removed. This would require a revision on the GCC-PhaT algorithm and improvement in hardware to ensure predictions are given the best data. The Kalman filter gets more accurate and closer to the true value when given more samples, so it would be perfect for a continuous RTS like a live tracking system [10].

Interpolation in estimated value [11]

Interpolation is a technique used to find the estimated value between two known values. The most basic and common form of interpolation is linear interpolation. Given two points (X1, Y1), (X2, Y2) and asked to interpolate the middle value (X,Y) using linear interpolation [11]:

$$Y = Y1 + (X - X1)\frac{(Y2-Y1)}{(X2-X1)}$$

The main use for live tracking is using interpolation in tandem with the Kalman filter to interpolate values between the predicted value and the previous measured value. By doing so the system can be given the illusion of being twice as fast as it actually is. This will produce a smoother path when the GUI updates the points and allow for faster movement of source without the GUI skipping values.

Interpolation is already used in GCC-PhaT in the form of value padding to lengthen the signal similar to a technique called zero padding but linear interpolation can be used for GCC-phat to produce smoother signals which should produce better TDOAs.

Linear interpolation is the most basic form of interpolation but more advanced forms of interpolation may incorporate prediction algorithms to determine value and are more accurate and robust.

CONCLUSION

All subsystems were implemented in an acceptable form as per the requirements from the paper design. Some subsystems were modified, but ultimately still reflect a useful and functional system.

The system demonstrates reasonable accuracy. It is able to track the position of a sound source in the grid within 10% accuracy over 60% of the in almost all positions on the grid, with accuracy generally being lower and less reliable near the edges of the grid. The system is also able to track the position of a sound source at an optimal position in the grid, with 10% accuracy over 80% of the time of the grid with a signal to background noise ratio of 0 dB or higher.

With this degree of accuracy, the system can consistently produce reliable results by taking multiple readings at the same position and using the median value of all readings.

The type of signal used for the system is important. Signals with broad frequency ranges (non-repeating signals such as music) and consistent volume levels offer better performance for the system. This primarily comes from the fact that the sound recording which is analysed is extremely short.

The benefit of this short sound recording is that the system is able to implement live tracking with a minimum update period of around 2.5 seconds. Potential improvements for live tracking include incorporation of Kalman filtering and interpolation for position-velocity prediction.

The system passes all ATPs and is thus considered a success overall in terms of the requirements identified based on the project brief.

REFERENCES:

[1] Ritu and S. K. Dhull, “A comparison of Generalized Cross-Correlation methods for time delay estimation,” Social Science Research Network, Nov. 2017, Accessed: Oct. 15, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3072083

[2] C. Knapp and G. Carter, “The generalized correlation method for estimation of time delay,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 4, pp. 320–327, Aug. 1976, doi: 10.1109/tassp.1976.1162830

[3] X. Wen and J. Wang, “TDOA Location Accuracy Experiment,” Journal of Physics, vol. 1237, no. 3, p. 032031, Jun. 2019, doi: 10.1088/1742-6596/1237/3/032031.

[4] Y. Xiong, “GCC-PhaT Library,” GitHub, Mar. 24, 2017. Accessed: Oct. 15, 2023. https://github.com/xiongyihui/tdoa/blob/master/gcc_phat.py .

[5] “Object Tracking using Time Difference of Arrival (TDOA) - MATLAB & Simulink - MathWorks United Kingdom.” Accessed: Oct. 15, 2023. https://uk.mathworks.com/help/fusion/ug/object-tracking-using-time-difference-of-arrival.html

[6] SoundSling. (2021, April 2). Finger Snap Sound Effect FX [Video].YouTube. Accessed: Oct. 15, 2023. [Online]. Available: https://www.youtube.com/watch?v=C6qUZZFBsso

[7] EverybodyBurts. (2016, October 5). The same boring drumbeat for 1 hour straight [Video]. YouTube. Accessed: Oct. 15, 2023. [Online]. Available: https://www.youtube.com/watch?v=pL0WW0WckPw

[8] sboss. (2011, April 16). Nyan Cat “Nyan”s for 1 Hour (Super Extended Version!) [HD] [Video]. YouTube. Accessed: Oct. 15, 2023. [Online]. Available: https://www.youtube.com/watch?v=GE8M5QM1sf8

[9] White Noise - Topic. (2019, January 13). White noise 1 hour long [Video]. YouTube. Accessed: Oct. 15, 2023. [Online]. Available: https://www.youtube.com/watch?v=lzmSKX5TF3g

[10] Alex Becker (www.kalmanfilter.net). (n.d.). Online kalman filter tutorial. Accessed: Oct. 15, 2023. [Online]. Available: https://www.kalmanfilter.net/alphabeta.html

[11] Linear Interpolation Formula - derivation, formulas, examples. (n.d.). Cuemath. Accessed: Oct. 15, 2023. [Online]. Available: https://www.cuemath.com/linear-interpolation-formula/