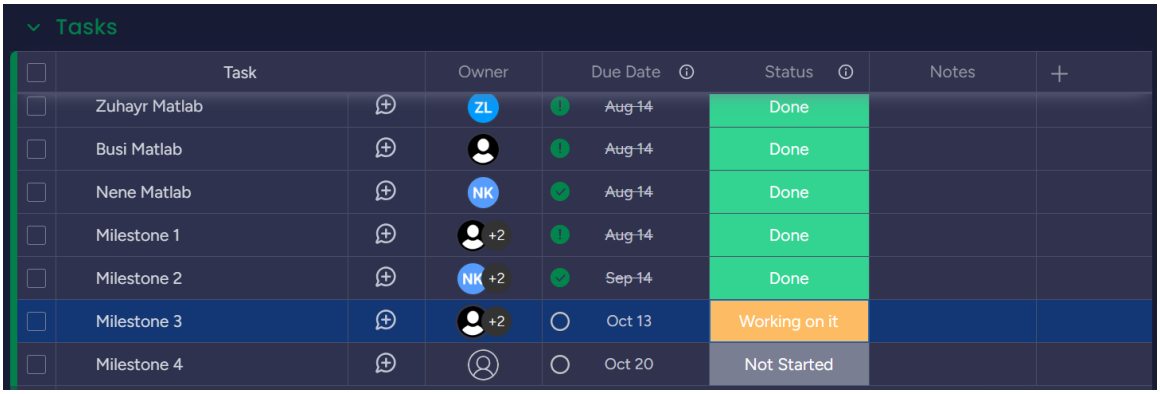
# Admin Documents

## Table of Contributions

| | |
|---|---|
| **Sibusisiwe Hlabangana** | Report writing<br>Triangulation code and grid code |
| **Nene Karingi** | Report writing<br>Time delay estimation code |
| **Zuhayr Loonat** | Report writing<br>Calibration signal and correcting error code |

## Project Management Tool



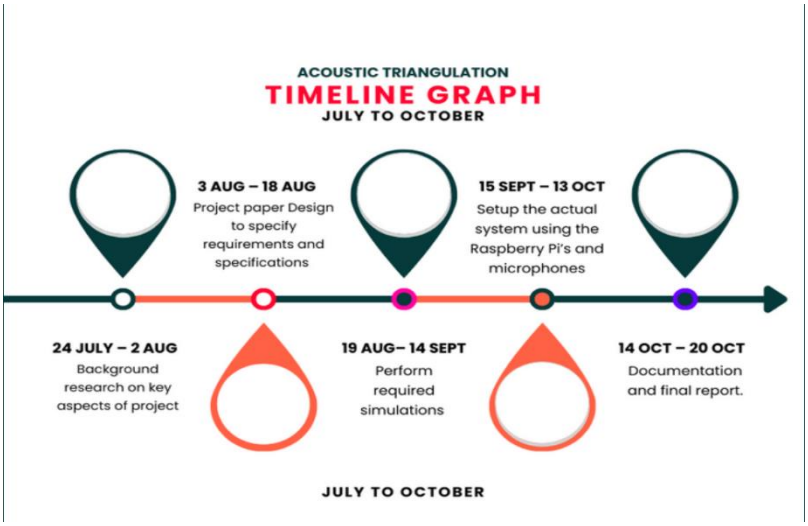

## Running the simulation

Find the simulation under the simulation folder where it can be run on MATLAB. One can change the source location (variables source_x and source_y) or you can have the points randomly generated.
You can also change the SNR to test different SNRs or have them randomly generated.

## Link to Gitlab

https://gitlab.com/zuhayrl/eee3097s_proj

## Project Timeline



Progress is on time. By the 14th of September, the group was able to write the project paper design and we have performed the required simulations.

# Simulation Setup:

## Simulation Environment, Approach and Tools

The principal objective of this simulation is to confirm the accuracy of sound source localization through the application of Time Difference of Arrival (TDoA) measurements obtained from four MEMS microphones. To achieve this, we have opted to use GCC-PHAT for time delay estimation in conjunction with weighted least square estimation algorithm to estimate the coordinates of the sound source.

### Grid structure

For the project, we have been provided with 4 receivers to triangulate the source signal. One only requires a minimum of 3 receivers to triangulate a point in 2D. However, having 4 means there is less reliance on each receiver and thus algorithm will still work even when one receiver is faulty.

Each receiver has been placed at the corner of a 0,5 x 0,5 m square grid and the simulation works by triangulating a point within that specified grid. Mic 1 is placed at (0,0), Mic 2 is at (0, 0.5), Mic 3 at (0.5,0) and Mic 4 is at (0.5, 0.5)

### Signal selection

The signal selected for the simulation is a chirp signal that is automatically generated by MATLAB. This signal was chosen as the ideal signal for autocorrelation as it has an unambiguous peak and distorts less when subject to noise due to its non-repetitive nature. The source signal and the calibration signal are a chirp signal which range from 0Hz to 1000HZ in 0.1 seconds.

The receiver signals have added noise and had a DC offset in order for the received signals to mimic real world conditions, so as to test the algorithms and techniques used for synchronization, noise reduction, time delay estimation and triangulation. In order to achieve this, the signals representing the received signals by the microphones had additive Gaussian noise, DC offsets due to the sensor errors in real life and they were delayed in order for the time delay and therefore triangulation algorithms to be performed.

### Simulation design

The simulation, as mentioned above, consisted of inputting points on the grid. From that, the time to each of the four receivers was calculated and the expected TDoAs were found. Using the TDoAs found in the previous step, four signals with additive Gaussian noise and random DC offsets, as well as the previously calculated delays were produced to simulate what would happen in the physical system. The signals were then passed through a noise reduction filter. Using the filtered signal, the TDoA was calculated for all of the four signals, and the values obtained were used in the triangulation signal locate the point, and finally the expected point and the calculated point were compared.

The entire simulation was carried out in MATLAB version R2023a using a range of pre-defined functions found in MATLAB found in the Communications Toolbox, DSP System Toolbox, Phased Array System Toolbox and the Signal Processing Toolbox, as well as other user defined functions to produce the simulation. All the required diagrams as well were plotted on MATLAB.

## Rationale for Chosen Approach

The selection of MATLAB as the preferred software tool was motivated by its widespread familiarity and user proficiency, as well as its flexibility and numerous signal processing functions present. Another reason for choosing MATLAB was the ease of integrating the different subsystems to one another, and its ability to produce diagrams with which the results can be compared to one another.

The simulated approach was chosen due to its ability to exhaustively test the accuracy and effectiveness of each subsystem separately as well as the entire system as a whole in terms of locating the point where the acoustic signal was being produced. By comparing the value obtained by each subsystem to its expected value, it was possible to locate errors in the various subsystems present in the simulation and implement other methods without affecting the other subsystems. This increased the modularization of the entire system which was time saving in the entire simulation process. Additionally, it made it easier to test the requirements of each sub-subsystem, subsystem and the entire system as a whole.

### 1.1 Simulation inputs

The input of the simulation is the location of the source signal which will be used to calculate the TDoA for each mic and use that data to prove the validity of the simulation and the algorithms used.

### 1.2 Simulation output

The output of the system is a grid that shows the position of the expected location of the source signal and the computed location of the source signal. This allows for a visual representation of whether the algorithm works.

## Simplifications and Assumptions

The following simplifications and assumptions were made during the simulation process:

- The speed of sound is 343m/s regardless of the room conditions.
- The sampling rate of the microphones is infinitely high.
- No attenuation happened during transmission from the source to the receivers.
- No reflections of the audio signal occur.
- The error in each mic is a random value that remains consistent throughout the entire simulation.
- The Pis are synchronized physically using a wire to connect the Pis so there is no need for Pi synchronization using code.
- The simulation does not factor in ADC drifts and jitters which are one of the non-idealities in a physical system.
- The system only has the source signal played within the 0,5m x 0,5m grid.
- The GUI is a very simplistic grid that shows the expected value and actual value which is not how the actual program GUI will be.
- The SNR of the signals after transmission is between 10 and 100.

# System Design and Implementation

## Signal Generation

The signal of choice from the source is the default MATLAB chirp signal. The signal is a sinusoid with varying frequency with time. The signal was chosen due to its robustness to noise over a pure sinusoid. Both the time and frequency domain representation of the chirp signal are shown in the figure below.

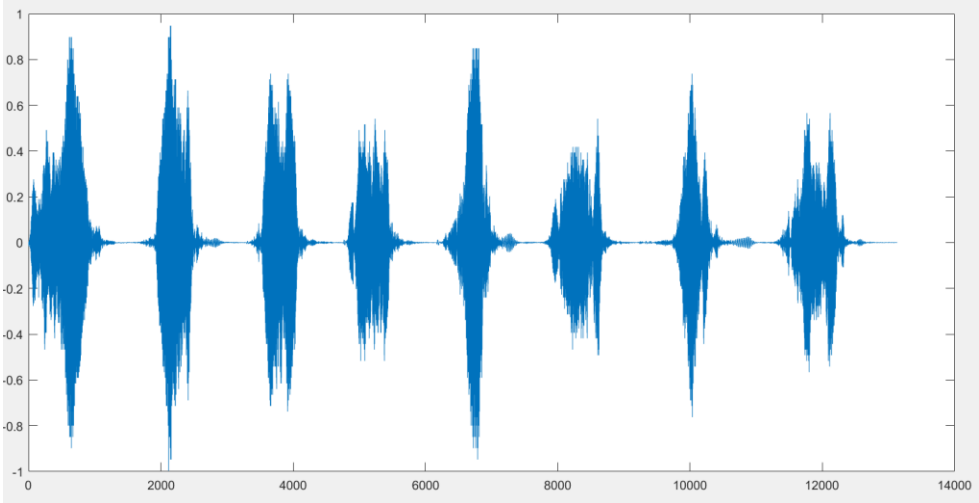



**Figure 1 Time domain representation of chirp signal**

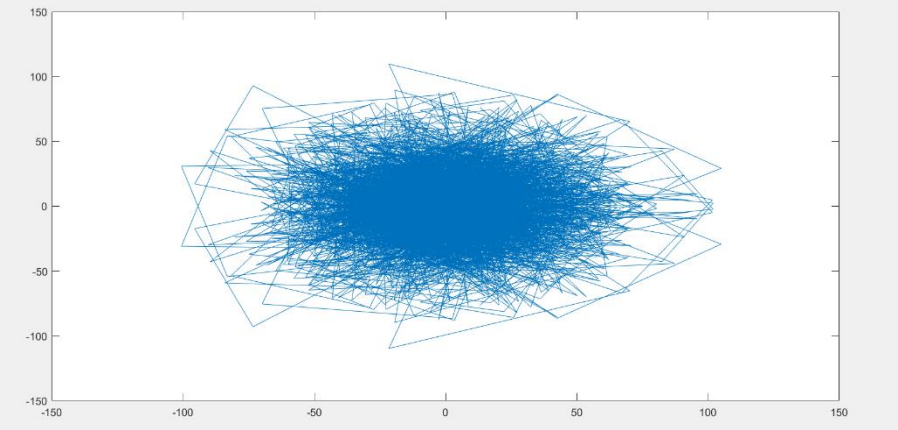



**Figure 2 Frequency domain representation of chirp signal**

For the simulation, the chirp signal was delayed for the TDoA calculation. An example of the delayed signals is shown below.

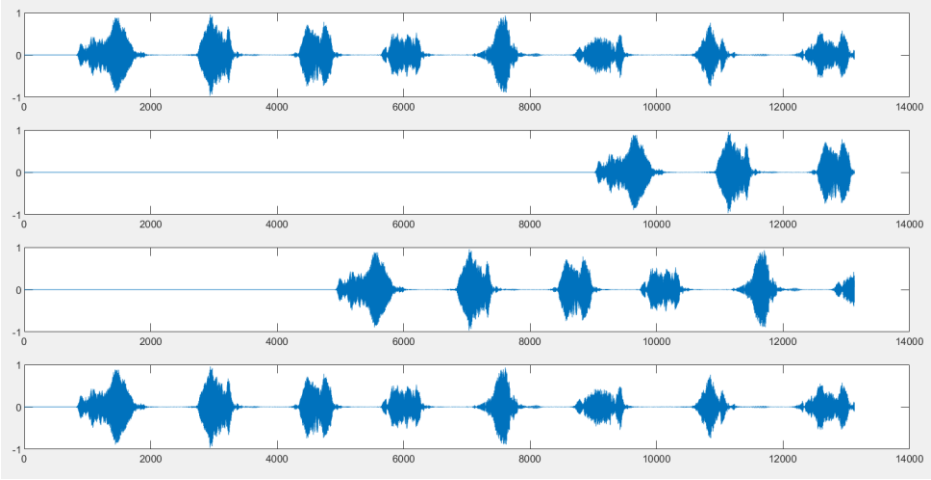



**Figure 3 Delayed chirp signals**

Gaussian noise and a DC offset were added to simulate real world conditions as well as the sensor errors. An example of the modified signals are shown below.

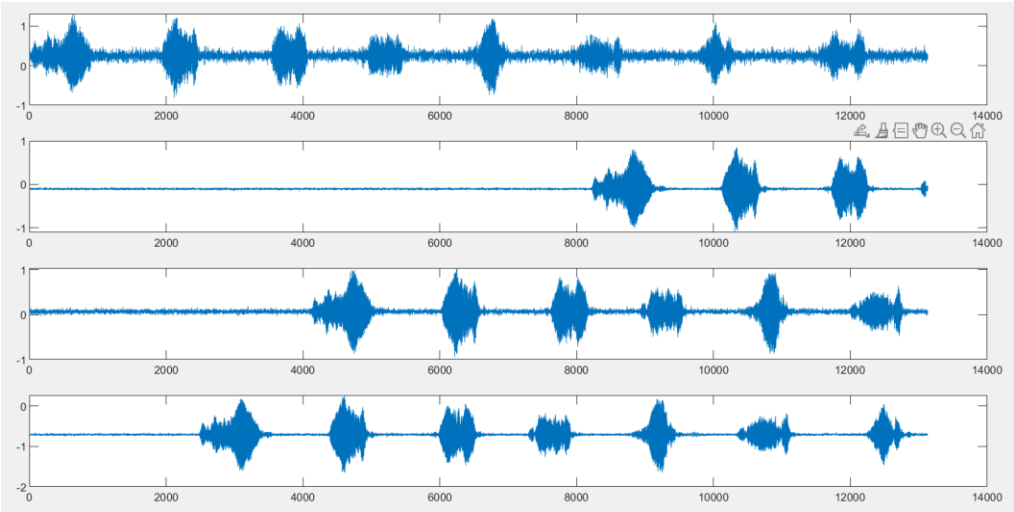


**Figure 4 Distorted chirp signals**

## Noise reduction

The noisy and delayed signals which are shown above are then passed through a Savitzky–Golay filter to smooth out the white noise that has been picked up from the environment during transmission. An example of the filtered signals is shown below, and is used to get the TDoA.

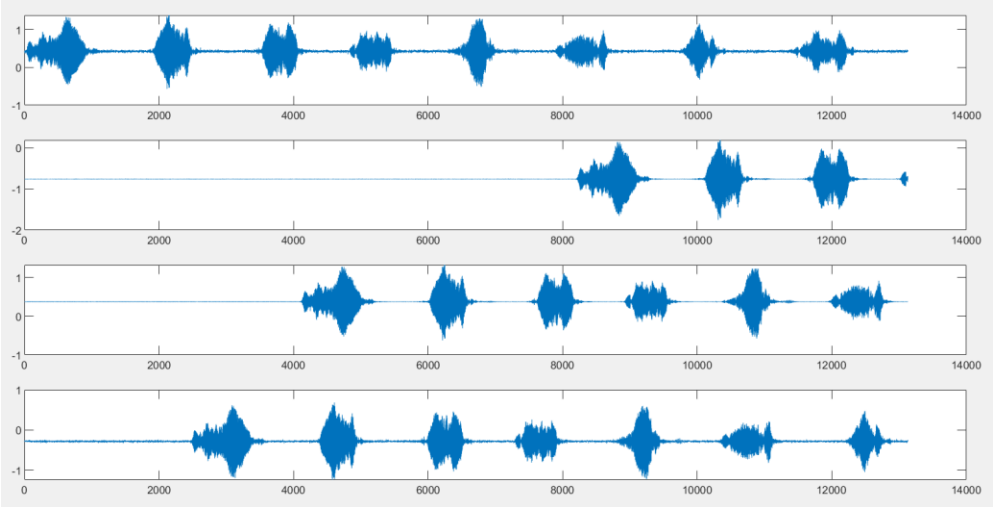


**Figure 4 Noise reduced chirp signals**

## Microphone Synchronization

A calibration acoustic signal is produced from the center of the grid. The expected time of arrival, $t_e$, to each microphone is calculated, and is meant to be equal for all four microphones. The actual time of arrival for each microphone, $t_a$, is obtained. The delay to each microphone is gotten as

$$delay = t_a - t_e$$

The microphones are then calibrated by subtracting the delay from each microphone from each TDoA.

## Time delay estimation

From the filtered signal, the TDoA is then calculated using the GCC-PHAT technique. The GCC-PHAT technique gives the time difference between two signals, using cross correlation and also provides phase information between the two signals. This is done using the gccphat method on MATLAB.

The GCC-PHAT is done with respect to mic 1 and an array of the TDoA values is created with respect to the reference signal (mic 1 is the reference mic).

The TDoA calculation uses the GCC-PHAT algorithm, an enhanced iteration of the GCC method, which constitutes a cross-correlation approach.

Given a source s(t) two signals are $x_1(t) and\ x_2(t)$ are received by the microphones where $x_1(t) = s(t) + n_1(t)$ and $x_2(t) = a.s(t-\tau) + n_2(t)$ where $n_1(t)\ and\ n_2(t)$ are additive Gaussian noise.

The two signals are then converted into the frequency domain to give $X_1(\omega) = S(\omega) + N_1(\omega)$ and $X_2(\omega) = aS(\omega)e^{-j\omega\tau} + N_2(\omega)$. The cross correlation between the two signals is then computed via the equation $R_{x1,x2}(\omega) = X_1(\omega)X_2^*(\omega)$.

The Generalised cross correlation with Phase Transform (GCC-PHAT) at a delay d is given by $R_{x1,x2}^{GCC}(d) = \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{R_{x1,x2}(\omega)}{|R_{x1,x2}(\omega)|}e^{j\omega d}d\omega = \delta(d-\tau)$ which is a Dirac-delta function at a time d = $\tau$ which is the value of the TDOA between the two signals.

This is done in MATLAB by the function:

```
tau_est12 = gccphat(signal1,signal2,Fs);
```

where `tau_est12` is the GCC-PHAT of `signal1` with reference to `signal2` both having a sampling rate of `Fs`.

1. **Experiments run on subsystems.**

The subsystem was initially tested with no delays between two signals to ensure that the TDoA value would be 0 in order to simulate a signal produced from the center of the grid.

The subsystem was then tested with a reference signal and a signal delayed by a known amount in order to test the accuracy of the algorithm. The reference signal was then delayed with respect to the other signal by a known value. This was done in order to test whether the algorithm can detect whether the delay of a signal was positive or negative with respect to a reference signal. The subsystem was then tested with random delays in order to test whether the results were repeatable with numerous random values.

## Triangulation algorithm and testing

The triangulation algorithm used is least square estimation. The algorithm computes two matrices

The $\tau_{i1}$ is the value of the TDoA with mic 1 being the reference mic.

$$A = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & \tau_{12} \times c \\ x_3 - x_1 & y_3 - y_1 & \tau_{13} \times c \\ x_4 - x_1 & y_4 - y_1 & \tau_{14} \times c \end{bmatrix}$$

$$B = \begin{bmatrix} b2 \\ b3 \\ b4 \end{bmatrix}$$

Where $b_i = \frac{1}{2}\left(x_1^2 - x_i^2 + y_1^2 - y_i^2 + (\tau_{1i} \times c)^2\right)$

Thus, one gets the equation.

$$Ax = b$$

Thus, subsequently using the MATLAB lsq function, the equation is solved using least square estimation going through multiple iterations until it converges.

1. **Experiments run the algorithm.**

The algorithm was tested using ideal values of the TDoA with mic 1 being the reference mic. A source position is selected and the ToA for each mic is calculated based on the source signal location. The ToA calculated the distance formulae and dividing by the speed of sound (c – 343 m/s). One then computes each TDoA as:

$$\tau_{i1} = \tau_i - \tau_1$$

Thus, using the ideal TDoA for each mic and the positions of each mic, the data is placed in a matrix and the estimated source position is calculated. Continuous tests are performed to validate the accuracy of triangulation algorithm.

The random function is used to generate a x and y coordinate for the source signal thus allowing for an accurate representation of system operation.

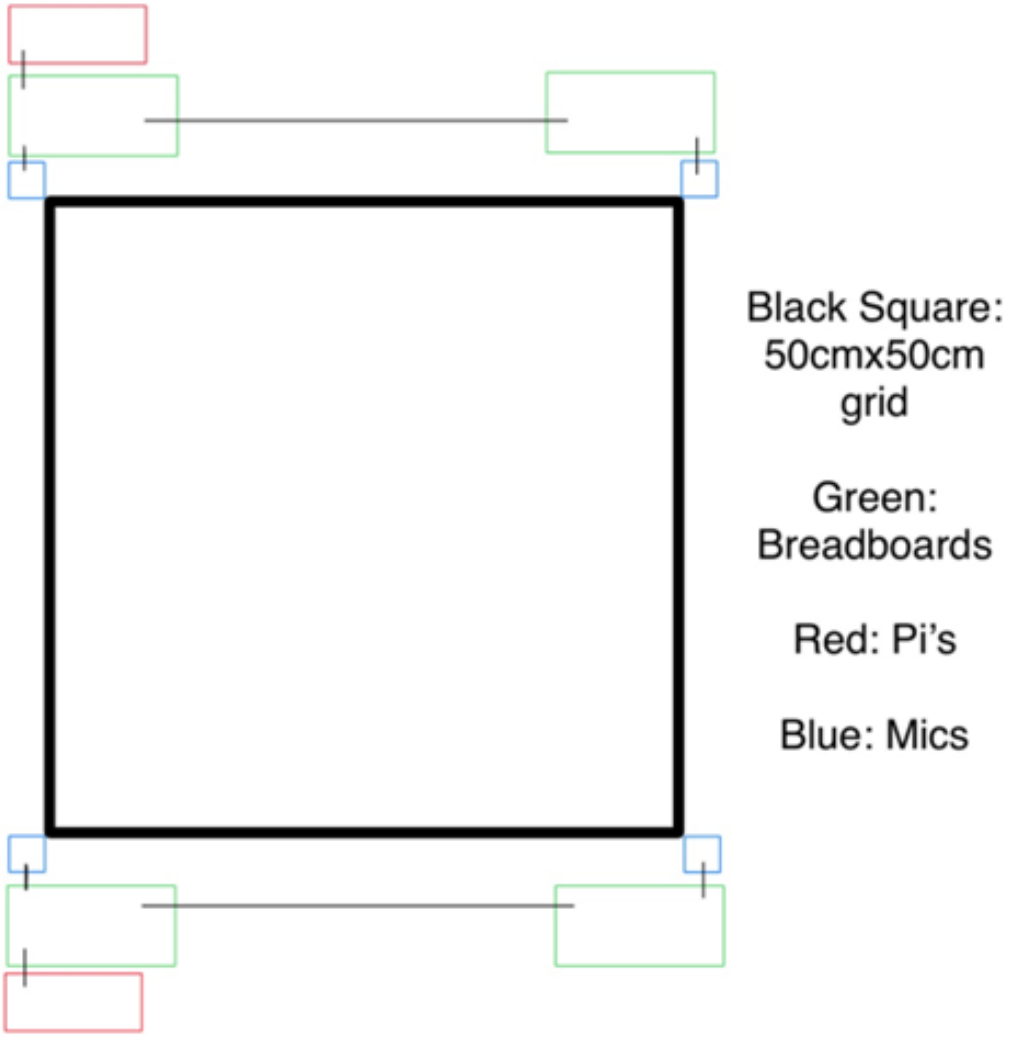
***Refer to file "triangulation_testing.m"

## Overall system architecture

To easily and accurately measure the sound emitted from the source, we decided to place the microphones in the corners of the 0.5m x 0.5m grid. Each microphone will measure sound simultaneously and the TDoA will be used to calculate the position of the source.

A personal computer will be used to run the MATLAB calculations. The data will wirelessly be transferred to this PC.

The figure below shows the setup:




Rough diagram of the system

## Distributed sensor network structure

Two microphones are connected to each raspberry pi, each in the corner of the grid. Mic 1 is situated at point (0,0), Mic 2 at (0, 50), Mic 3 at (50,0) and Mic 4 at (50,50). These measurements are in cm. This method of configuration allows for high accuracy in the middle of the grid, however, very close to the edges there are slight chances of low accuracy.

One Pi is used for the top two mics and the other for the bottom two. For each Pi two bread boards will be used, one for the Pi and one mic, and the other for the other mic.

## Simulation results and analysis

### Simulation experiment outcomes for each subsystem

1. **Signal synchronization**

   To make sure that the Pis would give us accurate results even if they were not completely in sync, we implemented a method to counteract unsynchronized behaviors. This was done by using a reference signal from the midpoint of the grid. The midpoint was chosen as the time taken to each mic should be equal. Since the Pis could be out of sync, The TDoA for this were taken relative to Mic 1 and then used to find the error for each Pi, the error amount was calculated by subtracting the TDoA of the other mics from Mic 1. This was then set to be subtracted from the TDoA for the main calculations.

   This method works well under the assumption that the Pis clocks are out of sync and do not have any ADC non-idealities such as drift and jitters.

2. **Time delay estimation**

   The GCC-PHAT algorithm works extremely accurately to an accuracy of 0.5μs, from which it begins to give inaccuracies. The TDoA algorithm does not take into account un-synchronized microphones which an in turn make the triangulation algorithm produce erroneous results.

   In terms of noise, the time delay estimation algorithm works perfectly for any SNR greater than 2. If the SNR is less than 2, the subsystem begins to produce erratic results.

   In terms of the ATPs, the subsystem aligns with all the ATPs set out in terms of expected results, repeatability, accuracy and dealing with negative delays.

3. **Triangulation**

In ideal conditions where there is no error or any non-idealities, the algorithm converges to each value of the source signal. There is no difference between the input source signal location and the computed source signal location.

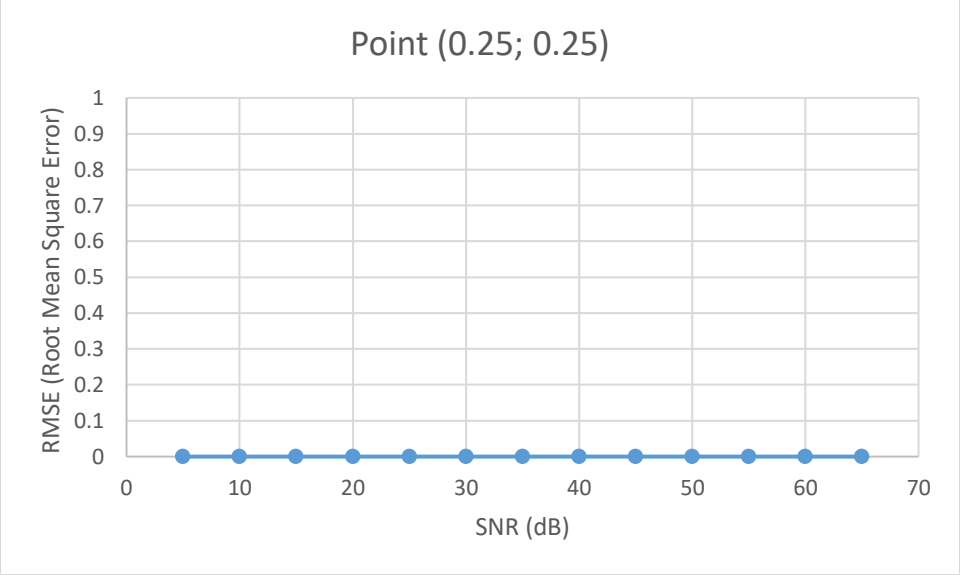
The algorithm works extremely accurately in ideal conditions however due to the non-idealities that exist, one needs to factor these in for the testing of the algorithm.

When a random error is placed on each the TDoA for the mic, the algorithm deviates from the expected source signal location.

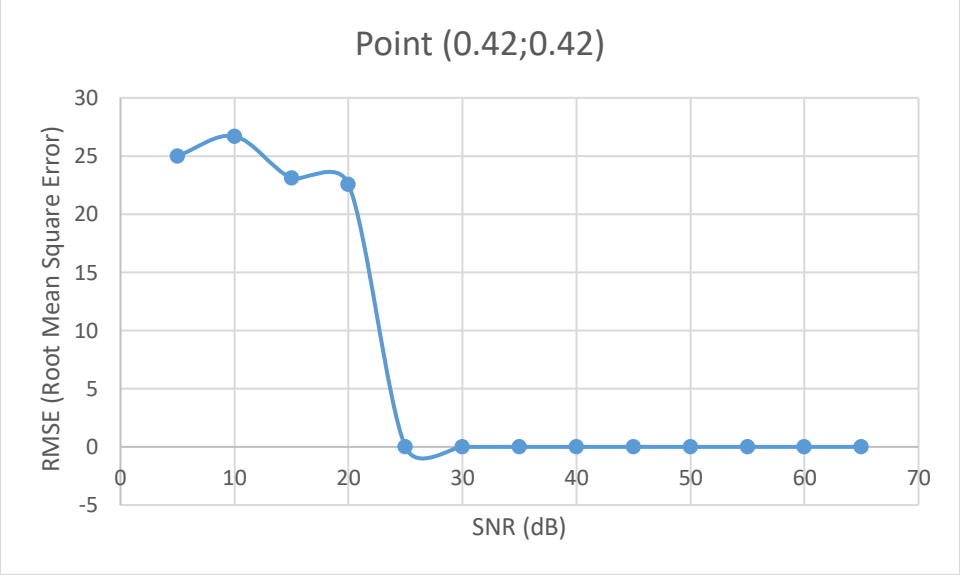
Adding random errors reduces the accuracy of the algorithm. When the error is in magnitude of 0.1ms, the system deviates from the expected source signal location by <5cm which aligns with the ATPs. However, when the error is in the magnitude of 1ms, the system deviates from the expected source signal location by 5 cm < x < 10cm. The system error occasionally is higher than the ATPs allowed error which is indictive of the sensitivity of the system to inaccuracies in the TDoA for each mic. In every 10 runs of the system, there system computes a value that is way too far from the expected source signal location.

**Simulation results for system as a whole**

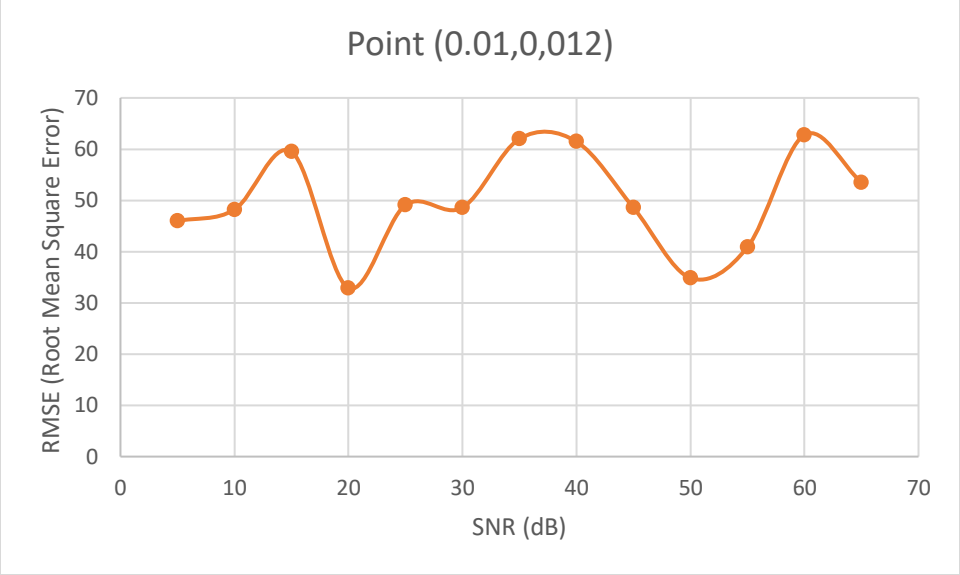
When testing the system for the same point and varying the SNR, the below graph summarizes the findings.




When the source signal is played at the center of the grid, even at low SNR, the algorithm accurate computes the location of the source signal.




As the source signals moves away from the centre of the grid, a higher SNR is needed to compute the values in the grid. As the signal to noise ration increases, the algorithm becomes more accurate at computing the value of the source signal. For SNR lower than 25dB, the accuracy of the system is very low however this increases as SNR, the system becomes more accurate as noise has a lower effect on the system.




For values 7cm near the edge of the grid, the values computed are wildly inaccurate. The values are only accurate in ideal situations where the SNR is infinite and there is no noise in the system.

**Accuracy and precision of system as a whole**

As discussed above with reference to the above graphs, for values near the center of the grid square, the SNR can be low, and the system can accurately locate the acoustic signal. The system works extremely accurately for these values with barely any RMSE detected.

The accuracy of the tracking when one moves away from the center declines for low SNR. Low SNR values have high RMSE. Thus, one requires at least 25dB SNR to get accurate tracking for any values that fall within $0.07m < x < 0.43m$ and $0.07m < y < 0.43m$

For values that fall within ±7 cm near the grid end, the tracking of the acoustic is highly inaccurate no matter what the SNR is. The system requires no noise to be able to accurately track the acoustic signal in this region of space.

The accuracy of the system decreases as the SNR decreases and RSME becomes $62m^2$.

The system has a precision of 1cm as if the system will output a similar estimated grid point. This is due to the fact simplifications made to simulation and furthermore, the system will produce the same TDoA each time as the assumption is that there is a constant delay on the mics due to not factoring in ADC jitters and drift.

As seen from a research paper published by UCT by David Da Costa on “Asynchronous Acoustic Localisation Using Time Difference of Arrival”, if the clock delays were randomised, the precision of the simulation would result in less precision as the system would have to compensate for a different clock delay for each recording of the calibration signal and the source signal each time and thus different estimated source signal would be computed.

**Challenges and limitations**

Signal synchronization

The main challenge that was faced concerning the synchronization of the Pis for this simulation was coming up with a method to counter act unsynchronized behaviors. We decided to use a reference signal to find the TDoAs relative to Mic 1 and then find the difference between those TDoAs to find the error due to unsynchronized behavior.

One of the main limitations we faced was that we had no way to implement and take into account latency from processing. These latency values are due to processor speeds, and we were not able to find a method to implement them in this simulation.

Noise Reduction

The biggest challenge was implementing a noise reduction filter on MATLAB that did not alter the phase of the signal while reducing the noise. The bandpass filter which was initially to be implemented did not maintain the phase of the filtered signal and therefore altered the subsequent time delay estimation calculation and produced erratic results. Therefore, the Savitzky–Golay filter was implemented which instead of bandlimitting the noise, uses a smoothing algorithm in order to reduce the noise of the signal.

Time delay estimation

The biggest challenges that occurred were regarding the algorithm of use and the accuracy of GCC-PHAT. Initially, the algorithm to be implemented was cross correlation, but the challenge was that it did not provide any information on whether it was a positive or negative delay between the signals which was necessary for the triangulation, thus GCC-PHAT was used in order to compensate for this. In terms of the accuracy, when dealing with small delays, less than $10^{-3}$ second, GCC-PHAT was producing erroneous results. In order to overcome this, for the time delay estimation calculation was performed in milliseconds.

Triangulation

The biggest challenge was the time constraint. Due to the limited time, the ability to implement a more sophisticated triangulation algorithm was not realizable.

Additionally, implementing the math equations for triangulation using MATLAB was difficult as there are many resources available on MATLAB and although that is a huge advantage, it is also a huge drawback. The large selection of resources means sometimes it can be overwhelming and difficult to navigate. The mathematics behind triangulation is already complex and thus sometimes it can be difficult figuring out which methods and functions to use. This challenge was addressed by consulting Google and using the online resources to gather more information.

The biggest challenge was trying to ensure that the other subsystems worked well as triangulation depends heavily on accurate TDoA or else there is huge deviations from the actual value. A work around was using a 0,4m x 0,4m grid as that region had more accurate tracking than using the entire 0,5m x 0,5m grid.

## Evaluation of ATPs

### Pi synchronization

a) **Test procedures**
    a. **Method 1 – Run NTP check Code**
       Run the NTP check code
    b. **Mthod 2 – Use reference signal to check time difference between pis**
       Set a reference signal from the middle and use it to find time difference between the Pis

**b) Expected results.**

a. **Method 1**
Command runs and shows connection and time out of sync for each Pi, delay should be close to zero or zero.

b. **Method 2**
Delay should be zero

c.

Pis should be synced to the same time.

For this section the ATP evaluation could not be done as it falls under the physical construction of the system. However, the ATPs have been included as shown below.

**c) Record of actual results**

| | Checklist |
|---|---|
| NTP connection between both Pis | |
| Out of sync time = 0 secs | |
| Reference signal shows TDoA = 0 secs for both Pis | |

**Signal acquisition**

**a) Test procedures**

**i.) Microphone Array**

A simple indicator of whether the microphone is working is by adding a visual indicator such as an LED and if the LED is on then the microphone is working. Thus, we know the microphones are capturing sound.

1. Play a sound.

**ii.) Noise Reduction**

2. Calculate SNR before and after signal pre-processing.
3. Check phase before and after noise reduction.
4. Compare signal before and after noise reduction.

**b) Expected results.**

**i.) Microphone Array**

**1.** All microphones should detect the sound.

**ii.) Noise Reduction**

2. SNR should be higher after the pre-processing of the signal.
3. Same phase after noise reduction
4. Smoother function after noise reduction

**c) Record of actual results**

| | **Checklist** |
|---|---|
| Simultaneous Recording | Cannot test yet |
| Improved SNR | YES |

| | |
|---|---|
| No phase shift | YES |
| Smoother function | YES |

## Data Acquisition

### Procedure

- Check if data is transferred through I2S
- Check if data is sent to parent

### Expected Results

- Pis receive data through I2S
- Parent PC receives data over SSH

### Checklist

Since this section deals only with the simulation this ATP cannot be evaluated at this time.

| | Checklist |
|---|---|
| Pis Receive data | |
| Parent PC receives data | |

## Sound Sync

### Procedure

- Place sound in the middle of the grid

### Expected Results

- Sound should reach each mic at the same time

### Checklist

| | Checklist |
|---|---|
| Synchronised Sound detection | |

**Time delay estimation**

a) **Test procedures**

1. Using a point with known TDoA values, compare the calculated value to the expected result
2. Place microphone in the exact centre of the grid
3. Check if numerical values are produced by MATLAB
4. Check if the subsystem can represent negative delays
5. Test the same signal, but delayed an with noise
6. Check if signal with DC offset receives the same TDoA as without

b) **Expected results.**

1. The calculated TDoA should be within ± 5 μs of the expected value
2. All the TDoA values should be 0
3. MATLAB should produce a numerical value for the TDoAs
4. Negative delays should be represented by the subsystem

5. Noisy and delayed TDoA value should be within ± 5 μs of the expected value
6. Signals with and without DC offsets should have the same TDoA value

**c) Record of actual results**

| | **Expected value** | **Actual value** | **Error** |
|---|---|---|---|
| Accuracy of TDoA | **5μs** | **0.5μs** | **None** |
| Center point of TDoA | **0 delay** | **0 delay** | **None** |
| Numerical Value | **Produces** | **Produces** | **None** |
| Negative Delay | **Same value as Positive** | **Same value as positive** | **None** |
| Noisy signal | **5 μs accuracy** | **< 2μs accuracy** | **None** |
| Offset Value | **Identical** | **Identical** | **None** |

**Triangulation**

**a) Test procedures**

To start testing the triangulation, place a sound signal within the grid block. Begin with a calibration signal placed in the middle of the grid that will be used to check if the triangulation algorithm is producing the correct location. The signal will then be moved to different locations within the 0,5m x 0,5m grid block.

1. To check algorithm outputs a x and y coordinate for the calculated grid point
2. To check the output of the algorithm gives a grid value ±5cm.

**b) Expected results.**

1. X and Y coordinates
2. Grid value computed is ±5cm (Mean Absolute Error (MAE) ) within actual grid point.

**c) Record of actual results**

| | **Expected value** | **Actual value** | **Error** |
|---|---|---|---|
| Computes an x and y coordinate | Outputs x and y coordinates | **Outputs x and y coordinates** | **None** |
| Mean Absolute Error (MAE) | ±5cm | ±10cm | **±5cm** |
| | | | |

**User interface**

The GUI purpose is to give a visual representation of the physical system. It needs to be able to show what the triangulation algorithm has calculated as the output grid point on the

**a) Test procedures**

1. GUI displays a grid.
2. GUI displays the coordinates of sound source.
3. GUI displays a dot on the grid where the sound source is locates.

**b) Expected results.**

1. GUI displays a simple 50x50 grid with labelled axis.
2. GUI displays the written (x,y) coordinates on laptop screen
3. GUI displays a dot that represents grid point where sound source is located.

**c) Record of actual results**

| | **Checklist** |
|---|---|
| GUI labelled grid display | ✓ |
| GUI coordinate display | ✓ |
| GUI dot on grid matching coordinates | ✓ (Accuracy depends on SNR and location) |

## Improvements and modifications

Although most of our ATPs were met, a potential improvement to the simulation would be to factor in more non-idealities. The system needs to be able to handle real world situations and

While doing this part of the project we realised that by using the standard NTPd suite we will probably get inaccurate measurements since the latency between the mic could be in the hundred milliseconds. Therefore, we decided to switch to the ChronyD suite, since the latency is a lot less.

Furthermore, to improve the accuracy and meet the ATPs, we have decided to have the system track a acoustic signal within a 0.4mx0.4m while keeping the mics 0.5m apart in order to get a more accurate value within the middle.

## Transition from simulation to physical implementation

The next step in the project is building the physical system. The triangulation code and the time delay code will be very useful in the next step of the assignment however the system inputs will be changing.

The code will need to be modified to take in sound files and process audio files instead of processing MATLAB generated signals. Some adjustments need to be made to the current code to be able to handle ADC jitters and drift when specifying the ADC sampling rate

Our system will have two mics connected on one Pi each and thus two mics will be referencing the same clock and have the same delay. The two mics will have the same clock drift as they reference the same clock. This will have to be tested to physically see how the system operates in real life and

The system also needs to be able to handle the latency issue of using NTP and this code needs to be included in the code for the physical system.

There might be a need to change the coding language depending on physical implementation the data cannot be processed using MATLAB on the Pis as they do not have the processing power.

The physical system will use physical noise filters combined with the Savitzky–Golay filter to reduce the effect from noise and ensure there is an adequate SNR

# Conclusion

## Key findings

Noise Reduction

- Bandpass and lowpass filters are not suitable due to phase shifts caused to the signal.
- Savitzky–Golay filter is a more suitable filter.

Time Delay Estimation

- GCC-PHAT is an extremely accurate and reliable algorithm to use
- For accurate TDoA values, an SNR of > 2 dB is required.
-

Triangulation

- A minimum SNR of 45dB is required to ensure accurate tracking of acoustic signal.
- Least square estimation is an adequate triangulation method given it has a high SNR and needs high synchronization.

**Lessons learnt and insights.**

The simulation opened the group's eyes to the many different factors that influence the system. Having to consider a multitude of non-idealities to consider.

This section has showed us the importance of considering all time errors that could occur, being from the physical system to the software. Due to the times being so low, even small changes can cause the answers to change drastically.

Implementing a simple triangulation is straightforward however, there are more accurate triangulation algorithms however they are more complex and difficult to implement. MATLAB has a lot of different functions that have made the simulation easier.

Savitzky–Golay filter is a better filter than bandpass or lowpass filter and thus will be used for the physical implementation. As through research one has found that testing the system in a room with a carpet and padding the walls around the speakers with pillows would reduce the effects of reflections on the system.

## References:


(1967) *Understanding GCC-phat as a Feature*, *Signal Processing Stack Exchange*. Available at: https://dsp.stackexchange.com/questions/74574/understanding-gcc-phat-as-a-feature (Accessed: 15 September 2023).

*awgn* (no date) *Add white Gaussian noise to signal - MATLAB*. Available at: https://www.mathworks.com/help/comm/ref/awgn.html (Accessed: 15 September 2023).

*lowpass* (no date) *Lowpass-filter signals - MATLAB*. Available at: https://www.mathworks.com/help/signal/ref/lowpass.html (Accessed: 15 September 2023).

*Plot* (no date) *MATLAB & Simulink*. Available at: https://www.mathworks.com/help/thingspeak/remove-dc-component-and-display-results.html (Accessed: 15 September 2023).

*smooth data* (no date) *Smooth noisy data - MATLAB*. Available at: https://www.mathworks.com/help/matlab/ref/smoothdata.html (Accessed: 15 September 2023).

*Random number generation* (no date) *Random Number Generation - MATLAB & Simulink*. Available at: https://www.mathworks.com/help/matlab/random-number-generation.html (Accessed: 15 September 2023).

*Adding constant offset to sensor* (no date) *MATLAB Answers - MATLAB Central*. Available at: https://www.mathworks.com/matlabcentral/answers/491677-adding-constant-offset-to-sensor (Accessed: 15 September 2023).

*noise filter* (no date) *MathWorks*. Available at: https://www.mathworks.com/help/images/noise-removal.html (Accessed: 15 September 2023).

*gccphat* (no date) *Generalized cross-correlation - MATLAB*. Available at: https://www.mathworks.com/help/phased/ref/gccphat.html (Accessed: 15 September 2023).

(No date) *Savitzky-Golay smoothing and differentiation filter - eigenvector*. Available at: https://eigenvector.com/wp-content/uploads/2020/01/SavitzkyGolay.pdf (Accessed: 15 September 2023).

*snr* (no date) *Signal-to-noise ratio - MATLAB*. Available at: https://www.mathworks.com/help/signal/ref/snr.html (Accessed: 15 September 2023).

*isqr* (no date) *Solve system of linear equations - least-squares method - MATLAB lsqr - MathWorks United Kingdom*. Available at: https://uk.mathworks.com/help/matlab/ref/lsqr.html (Accessed: 15 September 2023).

(No date a) *Total least squares method for robust source ... - sage journals*. Available at: https://journals.sagepub.com/doi/full/10.1155/2011/172902 (Accessed: 15 September 2023).

*Jeffrey T guido* (no date) *TDoA Page*. Available at: https://www.jeffreyguido.com/work/tdoa_page.html#:~:text=The%20difference%20in%20arrival%20time,ij%20r%20i%20j%20.&text=In%20addition%2C%20to%20simplify%20calculations,%2C%20(0%2C0)%20. (Accessed: 15 September 2023).