Asynchronous Acoustic Localization Using Time Difference of Arrival

Prepared by:
David Gavriel Da Costa

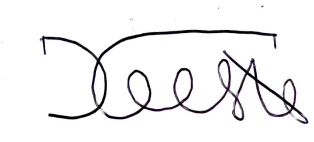
Prepared for:
Dr Stephen Paine
Department of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town
in partial fulfilment of the academic requirements for a
Bachelor of Science degree in Electrical and Computer Engineering

November 5, 2022

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.



November 5, 2022

David Gavriel Da Costa

Date

Acknowledgements

This project marks the end of a four-year degree and a larger academic journey. I feel immense gratitude to those who played a role in the completion of these milestones.

Concerning this project, there are a select few without whom it could not be completed.

To my supervisor, Dr Stephen Paine: You provided direction throughout the project, and guidance progress was slow. You were incredibly generous with your time but also showed restraint when necessary. I am grateful for the assistance you have provided throughout this project.

To Grant Norrie: This project could not have been completed without you. Your patience and willingness to give up your time to assist were genuinely selfless. Thank you for proofreading this report and, at times, reminding me how English works. In addition, your knowledge and wisdom were truly invaluable to the project's progression. I cannot thank you enough for the assistance you have provided.

To Dayalan Nair: Thank you for your willingness to answer my myriad questions, even when the relevance was obscure.

To Theodore, Nic, Mike, Skye, Jess and Zac, in no particular order: I am genuinely grateful to have experienced these past four years beside you. Thank you for the long phone calls I have with all of you, usually to confirm that we are in the same boat, whether floating or drowning.

Regarding the larger academic journey, I want to extend my gratitude to those who paved the way.

To my incredible parents, Alan and Andy: Mom and Dad, you have provided me with all I have, for which I am eternally grateful. Mom, you have truly been my compass, providing me with the unconditional love and support to persevere in my goals. Dad, thank you for the countless sacrifices you have made to give me this education and for paving the pathway for the journey I have taken.

To my second family, Linda and Motti: I am so fortunate to have you in my life. You have been my support throughout my time in Cape Town. Thank you for opening your home to me; your generosity cannot be understated.

Abstract

Acoustic localisation is the process of determining the coordinates of an audible signal's source by processing the signal measured by multiple receivers. This project aims to implement acoustic localisation using a method called Time Difference of Arrival (TDoA) triangulation. TDoA triangulation determines the coordinates of a signal source by using the Time of Arrival (ToA) of the signal at one receiver relative to another. This report documents the design process and results of a system that implements TDoA acoustic triangulation using a wireless network of distributed receivers.

A physical and simulated network consisting of four receivers was implemented and used to triangulate a known signal within a predefined space. The implementation of TDoA triangulation comprises three main aspects: synchronisation, TDoA estimation, and a triangulation algorithm. The synchronisation between receivers was achieved using a calibration signal to time align the recordings, TDoA estimation was performed using cross-correlation, and the triangulation algorithm used a Look Up Table (LUT) to determine the signal source's coordinates. The physical system implemented could triangulate the signal source in a $3.68\,\mathrm{m} \times 6.31\,\mathrm{m}$ grid with a mean accuracy of $0.29\,\mathrm{m}$, which is consistent with expected results obtained when simulating the hardware's limitations.

Abbreviations

ADC Analogue to Digital Converter

ALSA Advanced Linux Sound Architecture

ASIC Application Specific Integrated Circuit

CPU Central Processing Unit

ECM Electret Condenser Microphone

FERS Flexible Extensible Radar Simulator

GCC-PHAT Generalized Cross-Correlation Phase Transform

GPIO General Purpose Input Output

GPS Global Positioning System

I2S Inter-IC Sound

IoT Internet of Things

JFET junction-gate field-effect transistor

LUT Look Up Table

MEMS Micro-Electro-Mechanical System

NTP Network Time Protocol

OS Operating System

PCM Pulse-code modulation

PDM Pulse Density Modulation

PPM Parts Per Million

PPS Pulse Per Second

PWM Pulse-width modulation

SAR Successive Approximation Register

SCP Secure File Copy

SNR Signal to Noise Ratio

SoI Signal of Interest

SPL Sound Pressure Level

SSH Secure Shell

TDoA Time Difference of Arrival

ToA Time of Arrival

XML Extensible Markup Language

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The study of psychoacoustics provides an understanding of how the brain perceives and interprets sound. Humans interpret the direction from where a sound came by the time it arrives at each ear [3]. For example, a sound that occurs from the midpoint of the ears is heard by each ear simultaneously. However, as the sound shifts towards either side, there is a slight difference between the time the sound arrives at each ear. This is called interaural Time Difference of Arrival (TDoA), which is the time sound arrives at one ear relative to the other. Based on this slight delay or TDoA, the brain can infer the direction of the sound source [3].

Similarly, various applications in distributed sensor networks utilise TDoA for source localisation of received signals [4]. In a distributed sensor network, the TDoA is the time a signal reaches one sensor relative to another. TDoA acoustic triangulation[1] is a method which utilises the geometric relationships between the sound source and the microphones as well as the propagation velocity of the speed of sound to calculate the coordinates of the signal source.

Acoustic localisation was first used in WWI to triangulate gunshots in a battle field [5]. Today, acoustic localisation is still used in military applications such as earpieces worn by soldiers, which localise gunshots in a battlefield [6]. However, the spectrum of applications has broadened due to the development of smaller and low-power devices capable of implementing acoustic localisation. Today, acoustic localisation is used in applications ranging from household use cases such as Amazon's Alexa Echo, whose screen turns to face its' user when called upon [7], to assisted living applications that improve the caregiver's response times to abnormal sounds [8].

The implementation of acoustic triangulation entails the creation of an ecosystem conjoining hardware, digital signal processing and software, making it an interesting problem which stretches across different fields. This project aims to design and implement TDoA acoustic triangulation of a known signal utilising a network of spatially distributed microphones.

[1]The terms Acoustic localisation and Acoustic triangulation appear in the report. Acoustic localisation refers to the general practice of sound source localisation. In contrast, acoustic triangulation refers specifically to acoustic TDoA localisation

1.2 Objectives

The broad objective of this project is to implement a system of distributed sensors which triangulate a known audible signal[2] using TDoA.

1. To design a system which triangulates a known signal using TDoA.
2. To simulate a network of distributed sensors that captures and further triangulates a known audible signal.
3. To design and implement a physical network of distributed sensors that captures and further triangulates a known audible signal.

1.3 System Requirements

Based on the objectives mentioned above, the following system requirements are defined.

1. The system implemented consists of a network of spatially distributed sensors which capture signals in the audible range.
2. A known signal is triangulated using TDoA triangulation methodology.
3. The physical network of receivers communicates wirelessly.

1.4 Project Scope

This project entails the research, design and implementation of a TDoA acoustic triangulation. The scope of the project includes the following:

- A review which outlines similar applications and their methodologies.
- Well justified reasoning for all design choices made.
- An explanation of the theory applied when making design and other choices to the relevant extent.
- A stage of validation following each design choice.
- The design and implementation of a simulation of a network of receivers.
- The selection and implementation of suitable hardware used in the distributed sensor network.
- The design and implementation of various processes and algorithms involved in acoustic triangulation.
- Experiments and discussions which evaluate the performance of the simulated and physical systems implemented.

[2]The audible spectrum consists of all frequencies between 20 Hz and 20 kHz

1.4.1 Project Limitations

Like most engineering problems, the limitations of the project can be condensed to cost, time and power in no particular order. The project was limited to a budget of R 2000. Therefore, the system utilised low-cost hardware, which hinders the system's capabilities. The project was also to be completed in approximately fourteen weeks. Given the time and cost constraints, the system implemented was confined to a minimal working product. The system, therefore, is limited to non-real-time triangulation of a known signal in two dimensions.

As with all problems, multiple approaches can achieve the objectives of this project. Given the project's time constraints, the extent to which alternative methods are discussed is limited. However, all design choices implemented are well-reasoned and explained in detail.

The devices implemented in the system require a power source; this requirement confined the field tests to an indoor environment with access to a power supply. In addition, at the time of implementing the system, severe load-shedding interrupted the field tests on multiple occasions. The power requirements, along with time constraints, limited the extent to which the physical system was assessed.

1.5 Report Outline

The report begins with a review of previous and current implementations of acoustic localisation in chapter 2. The report then continues with an extensive explanation of acoustic localisation theory in chapter 3, providing the framework for implementing an acoustic triangulation. Chapter 4 then describes the design process and choices in producing the acoustic triangulation system. Following the system design is chapter 5 consisting of experiments and results in which the system is tested and evaluated. Finally, chapter 6 concludes the system implemented, followed by recommendations on how this project can be expanded.

Chapter 2

Literature Review

The following chapter presents a background to acoustic triangulation and provides a framework from which this project is developed. Signal source localisation is a well-documented practice with many alternative implementations. This literature review focuses on providing a qualitative analysis on the different methods of acoustic triangulation using TDoA. The following literature review begins by providing a high-level explanation of acoustic triangulation. The review then explores the history of source localisation followed by a brief discussion of the current use cases of acoustic localisation. The chapter then proceeds with an analysis of other papers which implement acoustic triangulation. The review then concludes with a summary outlining critical information found in the literature.

2.1 Triangulation Using Time Difference of Arrival

As the name suggests, TDoA triangulation is the process of determining the coordinates of a signal source based on the time the signal is detected by one receiver relative to another. TDoA triangulation is a localisation technique used when there is no information regarding the time the signal is transmitted, but the Time of Arrival (ToA) is known. Given the known propagation velocity of the signal, the coordinates of the receivers and the TDoA between the receivers, one can plot a curve that passes through all possible sound source coordinates, which will result in the same TDoAs. For example, if the signal source is in the centre between receivers, the function will be a straight line passing through all points equidistant to the receivers. However, when one receiver detects the signal earlier than the other, this will result in a curve which is biased towards the closer receiver. By adding a third receiver, additional curves can be plotted. The intersection of these curves will correspond to the coordinates of the signal source [9]. The figure below illustrates this further.

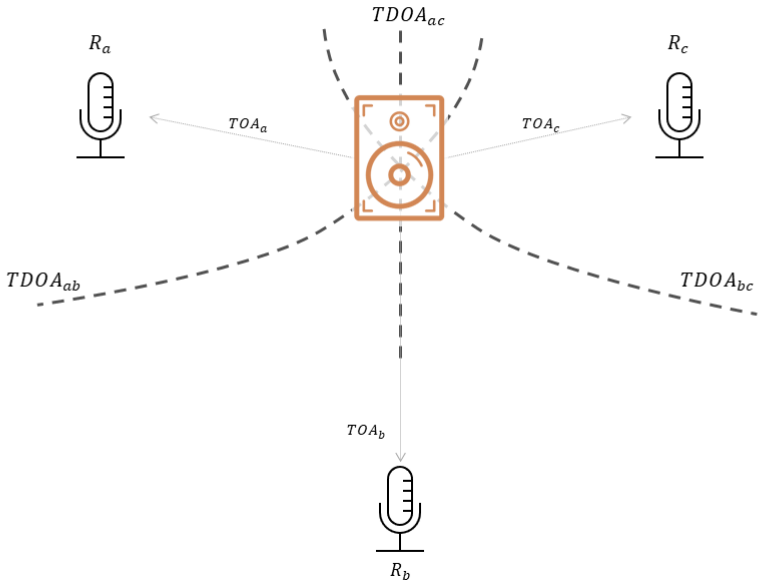


Figure 2.1: Curves drawn from TDOAs between receivers that intersect at the signal source

Figure 2.1 above illustrates the curves drawn based on TDoAs, which all intersect at the signal's source. The source is centred between $\boldsymbol{R_a}$ and $\boldsymbol{R_c}$. Therefore $\boldsymbol{TOA_a = TOA_b}$ and the $\boldsymbol{TDOA_{ac}}$ is a line where all points are equidistant to the receivers. In contrast, $\boldsymbol{R_a}$ is closer to the source than $\boldsymbol{R_b}$ therefore $\boldsymbol{TDOA_{ab}}$ plots a curve which is biased towards $\boldsymbol{R_a}$. The image above depicts the high-level theory behind TDoA, which this paper aims to implement.

2.2 History of Acoustic Localisation

Acoustic source localisation is a well-established practice, with the earliest implementation dating back to 1915. During World War I, Lawrence Bragg developed a system which triangulates the position of gunshots, which played a significant role in the British Victory in Cambrai in 1917 [5]. Bragg implemented triangulation using six microphones, six galvanometers and two observation posts. When someone in the observation post heard a gunshot or saw a muzzle flash, they would push a key which activates the film transport of the microphone's recordings to the galvanometer through a well-insulated long wire. The galvanometer then receives a current depicting the ToA of the sound, as shown in Figure 2.2 below.

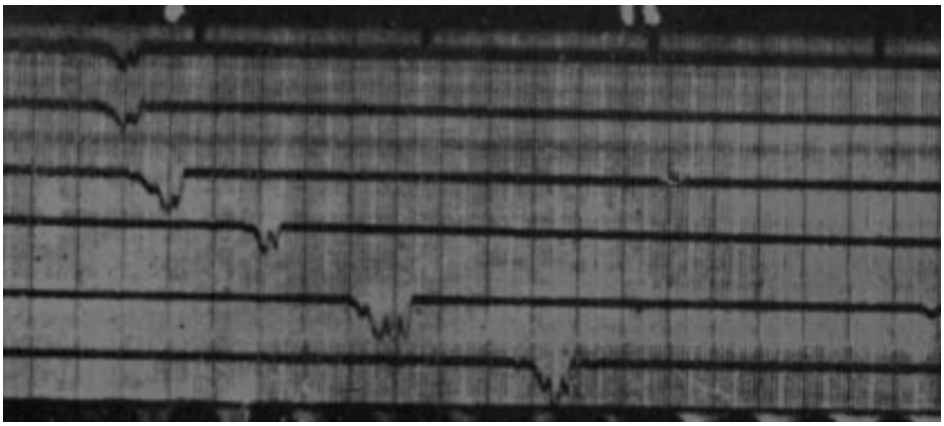


Figure 2.2: Original image of the 6 galvanometer readings after hearing a German gunshot

In Figure 2.2 above, one can see the six receivers picking up the gunshot's ToA at different microphones. Based on the position of the signal on the film, the TDoA can be read, and the coordinates of the sound source determined [10].

Another historical approach to acoustic triangulation was implemented by the U.S. Coast and Geodetic Survey in 1932 [11]. Acoustic triangulation was performed using a two-pen chronograph[1] on a ship and multiple hydrophones on the shoreline connected to a radio transmitter. One of the chronograph's pens marked the seconds the other pen marked when a specific signal was received from the radio station. When an explosion occurs, the shoreline hydrophones pick up the sound, and when a threshold is met, it activates the radio transmitter, which then outputs a signal. The ship receives the signal and marks the ToA on the chronograph. With the knowledge of the various propagation delays and the microphone's and ship's position, the coordinates of the explosion can be determined. The results showed a probable error of 400 m, which was considered a positive outcome, gave the equipment available and the novelty of localisation practices [11].

Although acoustic triangulation has expanded since these applications, many modern implementations are still based on the same use cases. This is because of the impulsive characteristics of gunshots and explosions, allowing for easier identification of the ToA of the event [12].

2.3 Current Use Cases

Shotspotter is an American-based company that offers gunshot localisation systems. Shotspotter utilises a network of acoustic sensors capable of recording and locating the coordinates of gunshots and other loud sounds. When three or more sensors detect a gunshot, the receivers transmits the recording to a central system which then identifies the ToA of the gunshot within the recording. The TDoA of the sound is then calculated, and the sound source is then triangulated [13].

Although Shotspotter utilises TDoA to triangulate the event, they combine this with other localisation methods to increase their results' accuracy. Shotspotter utilises a microphone array at each node, allowing for not only TDoA between nodes but also the direction of arrival for each array. The direction of arrival estimation can be done using various methods, such as fractional delays or beam-forming.

[1] A chronograph is a specific type of watch that is used as a stopwatch combined with a display watch.

Although these methods are beyond this project's scope, it is necessary to consider that most commercial implementations of acoustic localisation utilise multiple sensors and implement many different combined methods to achieve optimal results [13].

Beyond military and defence applications, acoustic localisation is also being researched for more complex applications. For example, Stephan Goetze et al. have found a use case for acoustic triangulation in assisted living scenarios. This source localisation application can locate and monitor patients based on sounds of movement, coughs, screams and falls [14].

Although most use cases are not dissimilar to historical applications, advancements in mathematics and technology have allowed modern implementations to utilise improved methods to triangulate a signal.

2.4 Acoustic Triangulation Implementations

The process of acoustic triangulation can be split up into two stages, the first includes ToA estimation in order to determine TDoA, and the second includes the triangulation algorithm, which utilises the TDoAs to localise the signal source. The following section discusses how others have implemented systems similar to this project. In addition, this section will provide a high-level discussion of triangulation methodologies and other outcomes and results that may assist in the design of this project.

2.4.1 Gunshot Triangulation using Calibration Signal for Synchronisation

The first implementation of acoustic triangulation discussed is done by Tobias Samuelsson [15]. In this instance, acoustic triangulation is performed using three microphones with the aim of triangulating a gunshot. The gunshot ToA estimation is performed by taking the rate of change of the signal received and then averaging the resulting waveform in windows. If the average of a window exceeds a determined threshold, then the first sample which exceeded the threshold within that window is returned as the estimated ToA.

In order to implement TDoA triangulation, one of the system requirements is that all devices are time synchronised. This implementation attempts to calibrate the devices by synchronising all sensors' clocks to a central unit. This is done through the use of a calibration signal which synchronises the clocks on all devices. Section discusses this further. This implementation also attempts to model the system's clock drift using a Kalman filter and then adjust for this drift in post-processing.

Samuelsson implemented triangulation using TDoA by using a triangulation algorithm created by R. Butcher and D. Mishra [16]. At a high level, this algorithm uses various simplifications of equation 3.5[2], to generate a closed-form solution for the intersection of the TDoA curves.

The paper assesses the triangulation performance when the devices are placed in various formations. In the first test, the receivers are placed in an array spaced 10 m between each device and triangulate a source **20 m** away from the centre. This test resulted in a **35.67 m** mean error over ten tests. Other tests included assessing the system's accuracy when receivers' are placed in a triangular formation, which resulted in poorer accuracy than the array structure. The second test obtained a mean error

[2]This equation is the function which forms the TDoA curves. The equation and is discussed in detail in section3.1.2

of **20.03 m** when triangulating a sound source **75m** away, and the third resulted in a mean error of **266 m** for a target **300m** away. The summary of the triangulation results is shown in the table below:

Formation	Distance to Target	Error
Array	20 m	35.67 m
Triangle	75 m	20.03 m
Triangle	300 m	266 m

Table 2.1: Triangulation Results obtained by Samuelsson [15]

When weighting the results in Table 2.1 above as a percentage of the target's distance, the system obtained a mean error of **97.9%** relative to the target's distance. The discussion of these results explained that the inaccuracy is due to clock drift [15].

Overall, this implementation showed the complexity of synchronising a network of devices using a calibration signal. Besides this paper, other literature was found that implements synchronisation using a calibration signal. This paper highlights the importance of accurate timing and a correct ToA estimation.

2.4.2 Triangulating a Receiver using 4 Transmitters

The second implementation of acoustic TDoA localisation assessed is done by Peng Wu et al. [17]. This paper assessed the accuracy of a specific algorithm when tracking an object within a grid. Peng Wu et al. took an alternative approach to triangulation by inverting the triangulation target and receiver system. The approach uses four transmitters that triangulate the receiver rather than four receivers that triangulate a transmitter. The theory behind this implementation is the same as the inverse system.

The transmitters are set up to repeatedly produce a repeating 0 Hz to 2.5 kHz chirp, which varies in length depending on the transmitter. One beneficial outcome of inverting the system is that since there is only one receiver and multiple transmitters, the system is inherently synchronised as one device records the chirps from all transmitters.

This paper aims to assess a triangulation algorithm called 'Hybrid of Weighted Least Squares and Firefly Algorithm' [17]. From a high-level, this algorithm is a combination of two different methods of formulating an error function which the algorithm attempts to minimise, the particulars of which are out of the scope. The system's performance is assessed using a 10 m***times***10 m grid with the transmitters on the corners and the receiver confined to the grid space. The grid allowed for more consistent testing and a limitation to possible receiver positions. The tests resulted in a mean error of **0.67 m**.

It should be noted that within this implementation, all results with an error greater than 2.5 m were considered 'bad results' and were discarded. In order to obtain the 0.67 m accuracy, 90 results were discarded, roughly 10% of the total results [17].

Key points worth highlighting in this implementation are the system's accuracy when correct time

synchronisation is achieved and that the grid structure allowed for more consistent assessments and a more straightforward definition of limitations of possible positions for the target.

2.4.3 Explosive Source Localisation in Indoor Environments

This paper is written by A.R Mohanty et al. [18]. The paper implements an acoustic triangulation system using a varying number of microphones. The paper assesses the performance of acoustic triangulation based on the number of microphones used and the formation of the microphones.

The triangulation system consisted of either 4,5 or 6 microphones. The microphones are synchronised by connecting them to a single data acquisition machine sampled at 1 MHz. The system implements cross-correlation[3] to detect the ToA of a 'low explosive bomb'. This ToA estimation method is an unconventional choice in this context as other applications utilise the rate of change thresholds when identifying an impulsive or explosive event, such as Samuelsson [15].

The system implemented various methods to estimate the events' coordinates. The system found that iterative estimation algorithms work best for acoustic triangulation compared to closed-form solutions such as the one implemented by Samuelson [15]. An example of an iterative algorithm is Matlab's 'f solve' algorithm. The algorithm works by repeatedly attempting to improve an initial guess for the coordinates until the system converges to a minimum, i.e., the intersection of the set of TDoA curves. An issue mentioned regarding this algorithm is that a poor initial guess may cause the algorithm to fail.

The experiment is set up to triangulate a recording of an explosive in an indoor environment. The table below shows the results obtained by A.R Mohanty et al. when using 4,5 and 6 microphones and the iterative algorithm.

No. Microphones	Distance to Target	Error
4	0.728 m	**0.01 m**
5	0.728 m	**0.001 m**
6	0.728 m	**0.005 m**

Table 2.2: Triangulation Results obtained by A.R Mohanty et al. using iterative algorithm [18]

In Table 2.2 above, the results showed that the triangulation accuracy increased from four to five microphones by a factor of 10 but decreased by 0.5c̀m when the 6th microphone was added. A.R Mohanty et al. explained that this minor variation was likely due to uncontrollable environment changes between the recordings. The results showed that as the number of receivers increases, so too does the triangulation accuracy, but the benefits are diminishing [18]. The results also show that a system with fine time synchronisation and a high sampling of 1 MHz rate can achieve a triangulation accuracy in the order of $\mathbf{10^{-2}}$ m.

This paper highlights that high-accuracy triangulation can be obtained when fine synchronisation and higher sampling rates are used. The paper also shows that the use of additional microphones increases

[3]Cross-Correlation is a means to determine the time lag at which two signals are most similar, this is further discussed in chapter 3

the accuracy of system but there are diminishing benefits when adding receivers.

2.4.4 TOA Estimation Using GCC-Phat and Cross-Correlation

The final paper discussed is written by V. Zetterberg et al. [19]. This paper does not implement triangulation but rather assesses the performance of different methods used for ToA estimations of acoustic signals. The paper assesses the performance of normalised cross-correlation called least mean squared estimation and Generalized Cross-Correlation Phase Transform (GCC-PHAT) when estimating the ToA of a signal[4] Both cross-correlation and GCC-PHAT are standard methods used for ToA estimation. From a high level, GCC-PHAT and cross-correlation output the similarity between the recording received and a reference signal at all possible time lags. The time lag corresponding to the maximum similarity is the time the signal likely occurred within the recording. These methods are further discussed in section 3.3.

The experiment was set up using a set of eight hydrophones and a moving sound source with a known position and velocity that passes all the receivers. The signals' ToA for both GCC-PHAT and cross-correlation are then compared to the ideal ToA based on the the position of the signal source at each time instant. The table below shows the averaged results for the normalised cross-correlation and GCC-PHAT when determine the ToA of a signal at varying distances.

Transmitter Distance	GCC-PHAT TOA Error	Cross-Correlation TOA Error
0.5 m	0.112 ms	0.113 ms
9 m	0.265 ms	0.333 ms
30 m	2.28 ms	4.76 ms

Table 2.3: Performance of GCC-Phat and Normalised Cross-Correlation for TOA estimation at varying distances to signal source

Table 2.3 above, shows two relevant outcomes from this experiment. Firstly the experiment shows that as the distance to the source increases, the more GCC-PHAT outperforms cross-correlation for ToA estimations. The other valuable outcome of this experiment shows that the greater the distance to the signal source, the greater the error in ToA estimation.

This paper provides valuable information regarding the performance GCC-PHAT and cross-correlation for ToA estimation. The experiments in this paper also provides insight into the ToA estimation accuracy as the distance to the source increases [19].

2.5 Summary of Implementations and Findings

The first implementation discussed was performed by Samuelsson [15]. There were various interesting aspects to the system's results. The first relevant outcome of this implementation is the importance of timing accuracy. Samuelsson's system obtained poor accuracy, and the explanation for this is clock

[4]The least mean squared algorithm utilises the outputs of the cross-correlation as a means to implement gradient descent for ToA estimation, the particulars of which are beyond the scope.

drift. This paper highlights that accurate timing and device synchronisation are paramount to acoustic triangulation using TDoA.

The second implementation reviewed is done by Peng Wu et al. [17]. This system had an alternative approach to localisation where the receiver is triangulated and not the transmitter. The sound selected to be played by the transmitters is a repetitive chirp. The experiments were performed in a 10 m$\boldsymbol{times}$10 m grid and obtained high-accuracy triangulation results.

The following paper discussed was implemented by A.R Mohanty et al. [18]. This implementation attempts to triangulate an impulsive event 0.7 m away from the microphone on the origin. This system utilised an iterative estimator function to triangulate the source. The experiments show that the system can accurately estimate the sound source's position when high-accuracy synchronisation is achieved between devices. The paper also utilised a triangulation algorithm different from Peng Wu et al. but still obtained high-accuracy results. This highlights the point that many different algorithms can triangulation a source, but the algorithm's performance is subject to accurate ToA estimation.

The final paper discussed assesses the performance of cross-correlation and GCC-PHAT for ToA estimations. This paper showed that GCC-PHAT outperforms normalised cross-correlation for ToA estimation. However, the paper also shows that the ToA estimation accuracy will lessen as the distance to the target increases.

When contrasting the methods discussed and comparing their results, the primary outcome is that the system's performance is subject to all aspects involving timing accuracy. The defining feature of the successful implementations is the accuracy of the ToA estimations and the inter-device synchronisation. Many different algorithms can produce the coordinates of the signal source given TDoA values. However, the system's performance depends on the accuracy of the TDoA values used. Consequently, to produce an accurate TDoA system, the additional focus should be placed on the accuracy of device synchronisation and ToA estimations rather than the triangulation algorithm itself.

Chapter 3

Theoretical Background

> There is geometry in the humming of the strings, there is music in the spacing of the spheres
>
> —*Pythagoras*

3.1 Acoustic Localisation

Localisation is an umbrella term for different methods of determining a signal source's location. In general, localisation is performed by processing the time the signal arrives at receivers in different locations [20]. Two common forms of acoustic localisation are ToA localisation and TDoA triangulation. Both implementations utilise the time the signal is detected to localise the signal.

3.1.1 Time of Arrival Localisation

ToA localisation requires knowledge of the time the signal is transmitted and received. The signal source's distance from the receiver can be determined using the signal's propagation velocity and transmission time. This implementation of localisation is well suited for active sensing applications such as radar and sonar, where the signal is transmitted at a known time, and the signal's reflection is received.

Given the scenario where a signal is transmitted at a known time, t_t and the ToA of the signal at a receiver is t_r. The distance from the receiver to the signal, d_{tr} can be calculated as follows:

$$d_{tr} = (t_r - t_t) \times c \tag{3.1}$$

In the context of this project, 'c' is the speed of sound, which can be calculated as follows:[1]

$$c = 331.3\sqrt{1 + \frac{\tau}{273.15}}\frac{m}{s} \tag{3.2}$$

Where τ is the temperature in *Celsius*[21].

Equation 3.1 forms a circle with radius $\boldsymbol{d_{tr}}$ centred on the receiver. By adding two more receivers to this system, the signal's position can be determined as three overlapping circles intersect at a single point. The intersection will correspond to the transmitter's location. Figure 3.1 below illustrates this further.

[1] The speed of sound is often estimated to $343\,m.s^{-1}$

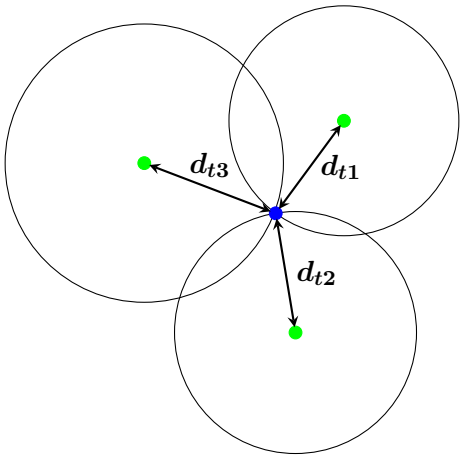


Figure 3.1: A figure of circles drawn from ToA values which intersect at the signal source

Where d_{ti} is the distance between the **transmitter** and **receiver** i. Figure 3.1 above depicts the intersection of the three circles drawn using equation 3.1. As shown, the intersection of these circles is shown to correspond to the signal source.

This project expands on this basic concepts to localise signals without knowing the transmission time. ToA localisation uses the absolute ToA of the signal, whereas TDoA localisation uses the relative ToA of the signal between two receivers.

3.1.2 Time Difference of Arrival Triangulation

Triangulation, in this context, utilises the difference between the ToA of a signal recorded by multiple receivers to identify the coordinates of the signal source. The signal travels a different distance to reach each receiver; therefore, the TDoA is a function of the distance between each receiver and the propagation velocity of the signal. The larger the TDoA between two receivers, the closer the signal source is to one receiver relative to another [22]. The Figure below and the description that follows illustrate this further.

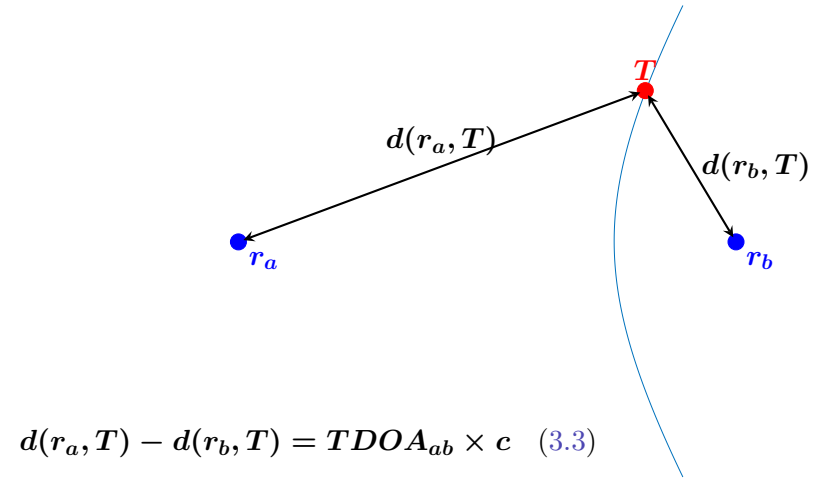


Figure 3.2: Figure of TDoA Curve passing through all possible coordinates for transmitter T

Figure 3.2 above was formulated as follows:

In Figure 3.2, there are two receivers at points $\boldsymbol{r_a}$ and $\boldsymbol{r_b}$. A signal is transmitted from some position $\boldsymbol{T}$ at time $\boldsymbol{t = 0}$. Receivers $\boldsymbol{r_a}$ and $\boldsymbol{r_b}$ then receive the signal at times $\boldsymbol{t_a}$ and $\boldsymbol{t_b}$ respectively. The TDoA between receivers $\boldsymbol{r_a}$ and $\boldsymbol{r_b}$, $\boldsymbol{TDOA_{ab}}$ can therefore be defined as $\boldsymbol{t_a}$-$\boldsymbol{t_b}$. Using Equation 3.1 $\boldsymbol{TDOA_{ab}}$ can be written as follows:

$$\boldsymbol{TDOA_{ab}} = \frac{\boldsymbol{d(r_a, T) - d(r_b, T)}}{\boldsymbol{c}} \tag{3.3}$$

Where $\boldsymbol{d(r_a, T)}$ and $\boldsymbol{d(r_b, T)}$ is the euclidean distance between receivers $\boldsymbol{r_a}$, $\boldsymbol{r_b}$ and the transmitter $\boldsymbol{T}$. Multiplying both sides of Equation 3.3 by the propagation velocity of the signal, the resulting equation is as follows:

$$\boldsymbol{TDOA_{ab} \times c = d(r_a, T) - d(r_b, T)} \tag{3.4}$$

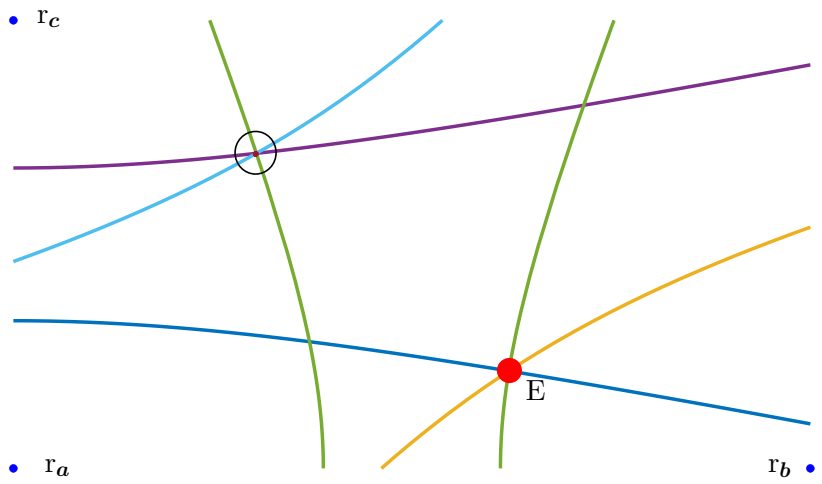
Equation 3.4 forms a curve representing all points on a plane where the difference between the distances $\boldsymbol{d(r_a, T)}$ and $\boldsymbol{d(r_b, T)}$ is fixed. The resulting equation forms a hyperbola with foci at points $\boldsymbol{r_a}$ and $\boldsymbol{r_b}$.

Figure 3.2 depicts this hyperbola based on a single TDoA value, where all points on the curve represent a possible transmitter position. However, Equation 3.4, and the graphic above, simplifies the hyperbolic equation, as a hyperbola is a function with a mirrored curve about an asymptote, which is omitted in Figure 3.2. The correct form of the function represents the distance between two points as a euclidean distance. When substituted into Equation 3.4 the following function arises:

$$\boldsymbol{TDOA_{ab} \times c} = \sqrt{\boldsymbol{(x_a - x_t)^2 + (y_a - y_t)^2}} - \sqrt{\boldsymbol{(x_b - x_t)^2 + (y_b - y_t)^2}} \tag{3.5}$$

Where $\boldsymbol{x_a}, \boldsymbol{x_b}, \boldsymbol{x_t}$ and $\boldsymbol{y_a}, \boldsymbol{y_b}, \boldsymbol{y_t}$ are the $\boldsymbol{x}$ and $\boldsymbol{y}$ co-ordinates of the receiver $\boldsymbol{r_a}, \boldsymbol{r_b}$ and transmitter $\boldsymbol{T}$ respectively.

Equation 3.5 shows that using the euclidean distance introduces squared terms into the function. This squared term produces an additional curve reflected about the hyperbola's asymptote. When a third receiver is added to the system, this will form an additional two TDoA values as there are three ToAs to compare. For each additional TDoA, an additional hyperbola is produced. All curves will cross the signal source. Therefore their intersections will correspond to the transmitter's coordinates. The following figure depicts the hyperbolas obtained using Equation 3.5 after adding a third receiver.



;

Figure 3.3: Figure of TDOA hyperbolas drawn from TOA values from receivers $\boldsymbol{r_a}$, $\boldsymbol{r_b}$ and $\boldsymbol{r_c}$ which intersect at the signal source T

Figure 3.3 above shows that the TDoAs between receivers $\boldsymbol{r_a}, \boldsymbol{r_b}$ and $\boldsymbol{r_c}$ form a set of hyperbolas that all intersect at the coordinates of the transmitter, $\boldsymbol{T}$. However, the hyperbola's reflection also produces a point of intersection which is circled in Figure 3.3. This undesired intersection introduces ambiguity in the solution to TDoA triangulation. This reflected curve results in the intersection of multiple hyperbolas having no closed-form analytical solution. Various algorithms have been implemented which introduce simplifications to Equation 3.5 to determine the correct coordinates of the source. These simplifications include using various assumptions regarding possible source positions, the particulars of which are beyond the project's scope.

The triangulation process can be split into two steps, the first is TDoA estimation, and the second is implementing a triangulation algorithm that determines the transmitter's co-ordinates [23]. To triangulate a transmitter using TDoA values, the ToA values need to be relative to the same time reference, i.e., the recordings start at the same time. Therefore, the system requires time synchronisation between receivers.

3.2 Synchronisation

Time synchronisation is an essential part of audio localisation. If receivers are not synchronised, it is impossible to obtain TDoA values [24]. There are currently many well-studied methods to implement time synchronisation between receiver in distributed networks. These include wired methods consisting of hard-wired master and slave clocks, wireless methods such as Global Positioning System (GPS) and Network Time Protocol (NTP) servers, as well as transmitting signals which calibrate the receiver's time [25] [26] [27].

3.2.1 Global Positioning System

In 1973 the U.S department of defence launched its' first GPS satellite. Currently, there are 24 GPS satellites orbiting the globe [28]. These satellites are equipped with atomic clocks, which synchronise receivers in the GPS network. Once a GPS receiver is turned on, it starts generating a 1-Pulse Per Second (PPS) signal with minimal drift and jitter [26], the concepts of drift and jitter are further discussed in subsection 3.2.4. Utilising GPS across devices provides accurate time synchronisation with an error in the order of nano-seconds [29]. However, the use of GPS for timing is an expensive solution to the synchronisation problem.

3.2.2 Network Time Protocol

NTP is another time synchronisation method commonly used for distributed systems. NTP works by exchanging messages containing timestamps between a host and peers within a server. NTP servers usually offer a time synchronisation between nodes in the order of milli-seconds [30].

3.2.3 Calibration Signal

Another form of synchronisation that may be implemented uses a calibration signal. This synchronisation is appropriate for low-cost applications where systems must be time aligned without access to common clocks or a GPS. Synchronisation using a calibration signal can be achieved in post-processing by time aligning the ToA of a signal from a known position. The process of synchronisation using a calibration signal's ToA is illustrated and described below.

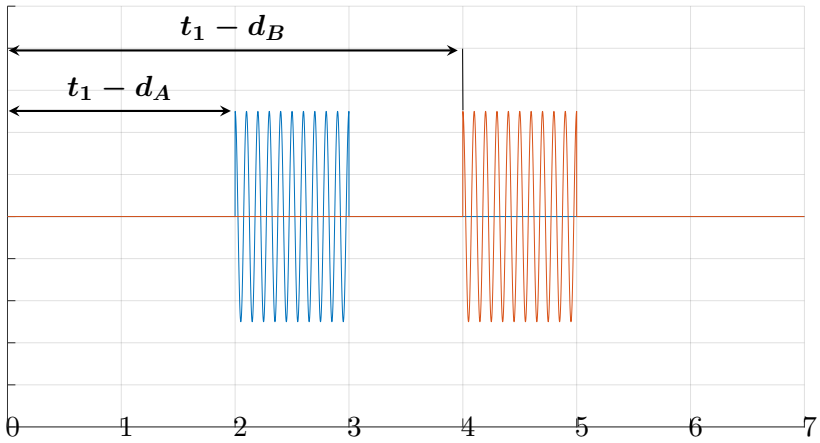


Figure 3.4: A figure of Calibration signal received by two unsynchronised receivers

Figure 3.4 shows two superimposed recordings received from receiver $\boldsymbol{A}$ in blue and receiver $\boldsymbol{B}$ in red. The following timing diagram illustrates how Figure 3.4 was formulated.

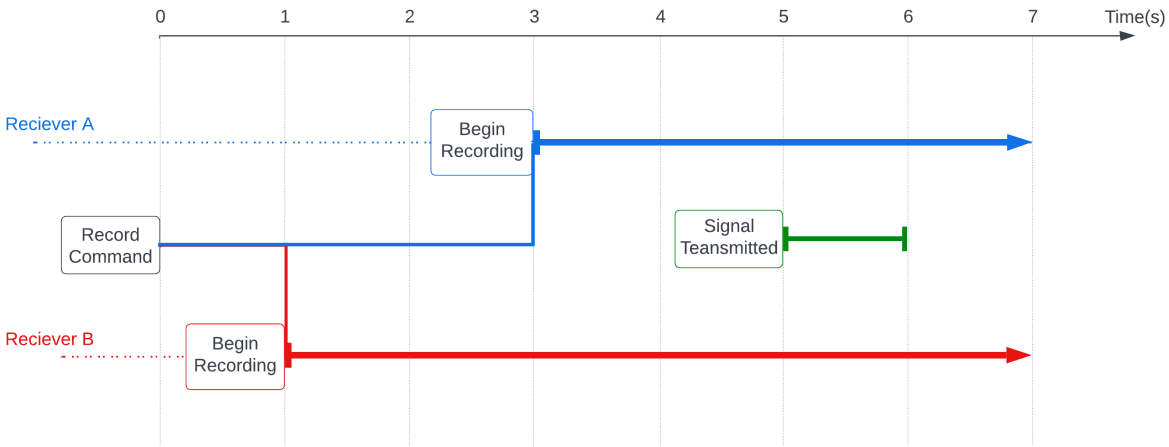


Figure 3.5: A figure depicting the timing diagram of the process that resulted in Figure 3.4

As shown in Figure 3.5, a command is sent to receivers $\boldsymbol{A}$ and $\boldsymbol{B}$ at time $\boldsymbol{t = 0}$ to begin recording. Microphone $\boldsymbol{A}$ receives the command after some delay at time $\boldsymbol{d_A = 3s}$ and Microphone $\boldsymbol{B}$ receives the command at time $\boldsymbol{d_B = 1s}$. These delays result in receiver $\boldsymbol{B}$ starting to record $\boldsymbol{2s}$ **before** receiver $\boldsymbol{A}$.

Once recording, a signal is played from from a distance of $\boldsymbol{0}$ m to each receiver[2] at time $\boldsymbol{t_1 = 5s}$. As shown in Figure 3.4, due to the delays, according to receiver $\boldsymbol{A}$ the signal occurred $\boldsymbol{t_1 - d_A = 2s}$ and to receiver $\boldsymbol{B}$ the signal occurred at time $\boldsymbol{t_1 - d_B = 4s}$. Given the prior knowledge of the position of the signal's source, the signal's expected ToA at the receivers can be calculated. The recordings are then shifted using this expected ToA to ensure that the calibration signal occurs at the expected time

[2] 0 m is used to exclude delay due to propagation time of the signal- to simplify the explanation

for both receivers. The ideal time of arrival, $\boldsymbol{t_i}$ of the calibration signal transmitted from point $\boldsymbol{T}$ is determined as follows.

$$\boldsymbol{t_{ix}} = \frac{\boldsymbol{d(T, X)}}{\boldsymbol{c}} \tag{3.6}$$

Where $\boldsymbol{X}$ denotes a receiver, $\boldsymbol{t_{ix}}$ is the ideal ToA of the signal at receiver $\boldsymbol{X}$, and $\boldsymbol{c}$ is the propagation velocity of the signal. In this context, $\boldsymbol{c}$ is the speed of sound or $\mathbf{343}\,\mathrm{m}.\boldsymbol{s^{-1}}$ Once the signals are shifted to the ideal ToA $\boldsymbol{t_{ix}}$, the correct ToA is redefined relative to $\boldsymbol{t_i}$ rather than the start of the recording. This results in all ToAs for each device relative to the same instant and therefore synchronised.

Applying this to the above mentioned scenario shown in Figures 3.4 and 3.5, since the signal was transmitted from a 0 m distance to the receivers, the expected ToA is 0 s. Therefore, receiver A's recording will be shifted back by 2 seconds and receiver B's recording will be shifted by 3 seconds, thereby synchronising the recordings.

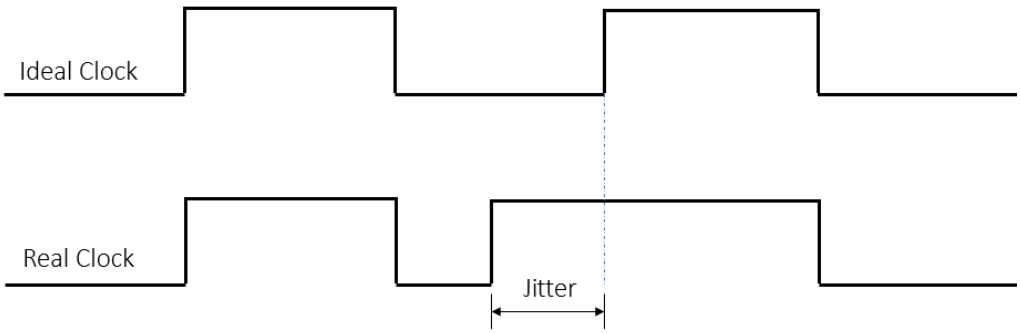
A concern regarding synchronisation using a calibration signal is that this method is rarely used in acoustic triangulation using TDoA. As mentioned in chapter 2, one paper discussed attempted this method and was unsuccessful in utilising calibration synchronisation for acoustic triangulation. No other papers that implement this method were found. Herefore, it is assumed that this method has yet to be successfully implemented in the context of acoustic triangulation using TDoA. This method is rarely used in similar applications as various clock non-idealities will compromise the timing accuracy, causing the devices to fall out of sync as time progresses.

3.2.4 Clock Non-Idealities

In general, the two clock limitations commonly assessed in audio applications are clock jitter and frequency drift. As with most hardware concepts, these two issues are low-level and require in-depth analysis for their behaviour and effects to be well understood, however, in the context of this paper, a surface-level understanding of these effects is adequate.

Clock Jitter

Jitter can be loosely defined as deviation of a clock edge from its' ideal position. Figure 3.6 below illustrates this concept in the time domain.



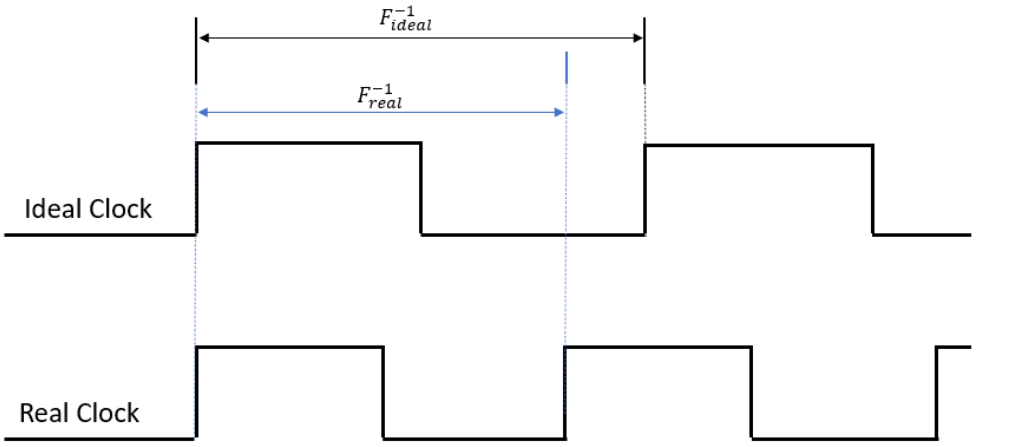
;

Figure 3.6: Figure depicting Clock Jitter

As shown in Figure 3.6 above, jitter occurs when the clock's periods are inconsistent, and the clock's edges deviate from the ideal position. Jitter's effect on a clock can be analogised to jerky-like movements in a projection film with an inconsistent frame rate.

Frequency Drift

Frequency drift, on the other hand, is when a clock's frequency is offset from the ideal clock, i.e., the oscillator is faster or slower than the specified oscillation rate. The following figure depicts the waveform of a clock with frequency drift.



;

Figure 3.7: Figure depicting Clock Drift

As shown in Figure 3.7 above, clock drift occurs when the clock's frequency deviates from the expected frequency of the system, causing short or long clock cycles. The effects of clock drift can be related to the film analogy. When a film's frame rate experiences drift, the resulting outcome is a shorter or longer film.

These clock complexities may affect the synchronisation process's accuracy, thereby impacting the correct TDoA estimations. Only once the receivers are correctly synchronised can the ToA values be used for triangulation.

3.3 Time of Arrival Estimation

The process of obtaining the ToA can be done through a coherence measure. A coherence measure will indicate the degree of similarity between a signal received and a reference signal at various time delays [31]. The time delay at which the similarity measure is a maximum will correspond to the ToA of the signal. The coherence algorithms commonly used for ToA estimation are Cross-Correlation and GCC-PHAT.

3.3.1 Cross Correlation

Cross-Correlation is a standard method used to estimate the similarity between two signals at variable time delays. The resulting output of the discrete cross-correlation is an array of values representing the correlation coefficient of the two signals at each every time delay.

The output of Equation 3.7 results in a series twice the length of the original signals, as both negative and positive delays are possible. A negative delay represents that the signal received in the recording leads the reference, whereas a positive delay shows the signal received lags the reference signal [32]. The ToA of the signal will correspond to the time delay at which the correlation coefficient is a maximum.

The normalised form of cross-correlation is commonly used as it is less sensitive to any linear changes in amplitude of the signals being correlated [33]. The normalised cross-correlation $\boldsymbol{CC_{12}}$ at some delay $\boldsymbol{d}$ between signals $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ is defined as follows:

$$\boldsymbol{CC_{12}(d)} = \frac{\sum_i (\boldsymbol{y_1(i)} - \boldsymbol{m(y_1)} \times (\boldsymbol{y_2(i-d)} - \boldsymbol{m(y_2)}}{\sqrt{\sum_i (\boldsymbol{y_1(i)} - \boldsymbol{m(y_1)})^2} \times \sqrt{\sum_i (\boldsymbol{y_2(i)} - \boldsymbol{m(y_2)})^2}} \tag{3.7}$$

Where $\boldsymbol{m(y)}$ is the mean of signal $\boldsymbol{y}$.

3.3.2 Generalised Cross-Correlation Phase Transform

GCC-PHAT was proposed as an extension of cross-correlation by Knapp and Carter [34]. GCC-PHAT aims to narrow the peak and minimise the side-lobes of the cross-correlation with the specific goal of determining the delay of one signal relative to another. However, GCC-PHAT does not not show the degree of similarity between the two signals [35]. This is done using a weighting function $\boldsymbol{w(k)}$ [34]. Given two signals $\boldsymbol{y_1(n)}$ and $\boldsymbol{y_2(n)}$ with Fourier transform's of $\boldsymbol{Y_1(K)}$ and $\boldsymbol{Y_2(k)}$ the weighting function $\boldsymbol{w(k)}$ is defined as follows:

$$\boldsymbol{w(k)} = \frac{\boldsymbol{1}}{|\boldsymbol{Y_1(k)Y_2(k)^*}|} \tag{3.8}$$

The frequency domain representation of Equation 3.7 is then multiplied by this weighting function as defined in 3.8 in order to obtain the GCC-PHAT function $\boldsymbol{R_{12}}$, which is described as follows:

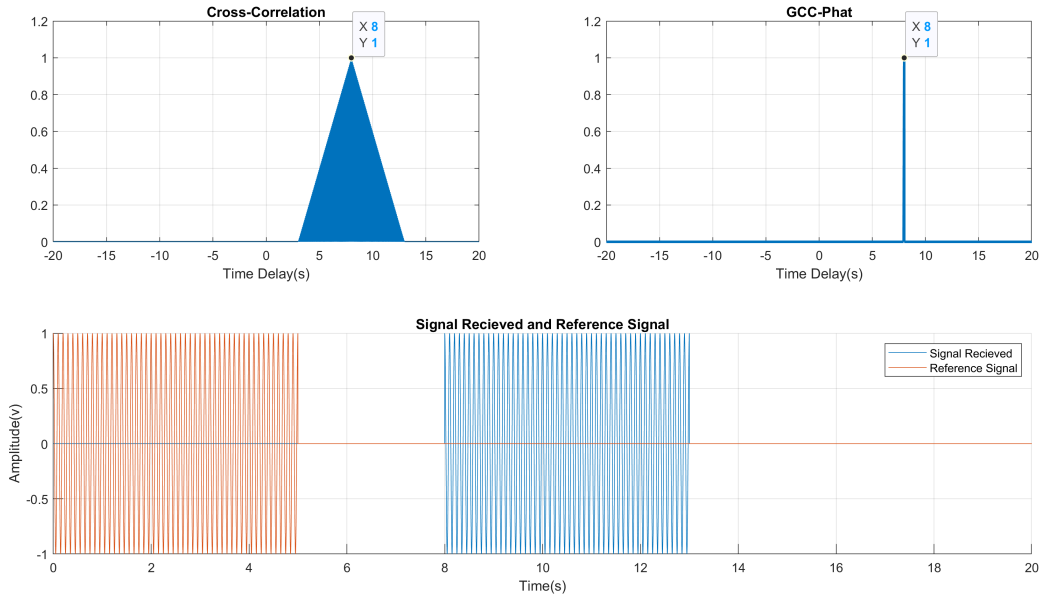
$$\boldsymbol{R_{12}} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{k=0}}^{\boldsymbol{N-1}} \frac{\boldsymbol{Y_1(k)Y_2(k)^*}}{|\boldsymbol{Y_1(k)Y_2(k)^*}|} \tag{3.9}$$

Where $\boldsymbol{N}$ is the length of the signals being compared.

By applying this weighting function, the input signals are 'whitened' [36], meaning the frequency components with a higher Signal to Noise Ratio (SNR) are enhanced while others are suppressed [23]. GCC-PHAT has proved to be independent of reverberation but is limited by low SNR environments which will downgrade its' performance [35]. Due to this weighting function, the GCC-PHAT output at the delay where the two signals are most similar will be the same as the cross-correlation. In contrast, all other points will be distorted and provide little insight into the correlation between the two signals at that point. For example, given the scenario when a recording contains a signal repeated twice, the first occurrence of the repeated signal has a higher SNR than the other. The output of GCC-PHAT Will show a clear peak at the time of the first signal, but the time at which the second signal occurs will provide little insight into the correlation between that signal and the reference signal. Section 4.5.1 Will analyse these effects further.

3.3.3 Comparing Cross-Correlation and GCC-Phat

The following figure displays the outputs of cross-correlation and GCC-PHAT when correlating a signal received at t=8 s with its reference signal. The signal used is a 10 Hz 5 s wave.



;

Figure 3.8: Cross Correlation and GCC-Phat of a signal delayed by 8 seconds

In Figure 3.8 above, both the cross-correlation and GCC-PHAT have a maximum when the time delay is 8 seconds, corresponding to the time that the signal is received. The figure also shows that GCC-PHAT is a single narrow peak due to the weighting function, whereas the cross-correlation has correlation coefficients which increase until the maximum. The cross-correlation has high side lobes due to the repetitive and periodic nature of the signals used. For this reason, when selecting a signal for ToA estimation, ideally, the signal should be non-repetitive, such as a frequency chirp.

3.3.4 Frequency Chirp

Frequency chirps are commonly used in radar applications. A chirp allows for an increased range resolution and SNR. Broadly, it can be thought of as a compromise between the high power and bandwidth of an impulse and the low power and low bandwidth of a pure tone, thereby allowing for higher-power signals to be transmitted within a limited bandwidth. A linear frequency chirp is an oscillating signal with a frequency that increases linearly from a start frequency to an end frequency within a defined period.

$$y(t) = cos\left(2\pi\left(\frac{f_1 - f_0}{2\tau}t^2 + f_0 t\right)\right) \tag{3.10}$$

Where f_0 and f_1 denotes the start and end frequency respectively and τ is the length of the chirp time.

Pulse Compression Gain

The time-bandwidth product B_τ is the chirp length multiplied by its frequency range. The time-bandwidth product determines the amount of SNR gain at the output of the correlation function [37].

Given a signal with an initial SNR, SNR_i, the output of the correlation function will result in an output SNR of SNR_o, which is related to the time-bandwidth product $B\tau$ as follows:

$$SNR_o = SNR_i \times 2B\tau \tag{3.11}$$

The time-bandwidth product defines the SNR gain. This gain allows for easier detection of the signal in noisy environments. In addition, this property of a linear chirp makes it an appropriate signal type used for ToA estimation as the time and bandwidth can be adjusted based on the application's requirements [37].

3.4 Microphones and Interfaces

The receiver network used in acoustic triangulation consists of multiple spatially separated microphones. When implementing a system consisting of various devices, one needs to consider the behaviour and interfaces for each component. This section discusses the microphones commonly used in similar applications and their respective communication standards.

3.4.1 MEMS Microphone and ECM

A microphone is a transducer which converts sound waves into electrical signals. Due to their small size and low power consumption, two microphone types are most popularly used, the Micro-Electro-Mechanical System (MEMS) microphone and the Electret Condenser Microphone (ECM) [2].

An image depicting a generic MEMS microphone is illustrated below.

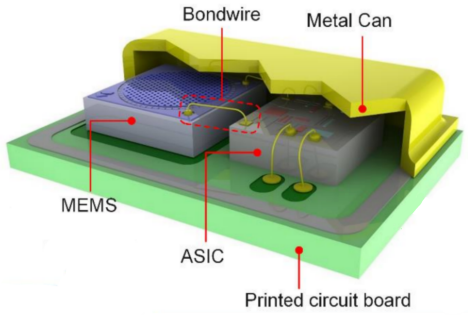


Figure 3.9: Diagram of MEMS microphone [1]

Figure 3.9 above shows the components used in a MEMS microphone. The microphone utilises a small hole and a mechanical diaphragm which forms a capacitor. The diaphragm moves when airwaves pass through the hole, thereby generating a voltage. As shown in Figure 3.9, the microphone is paired with an Application Specific Integrated Circuit (ASIC), which is a preamplifier for the signal. MEMS microphones can be found with either an analogue or digital output [2]. The digital output is either Inter-IC Sound (I2S) or Pulse Density Modulation (PDM), which are further discussed in the following section.
The following image depicts the structure of an ECM.

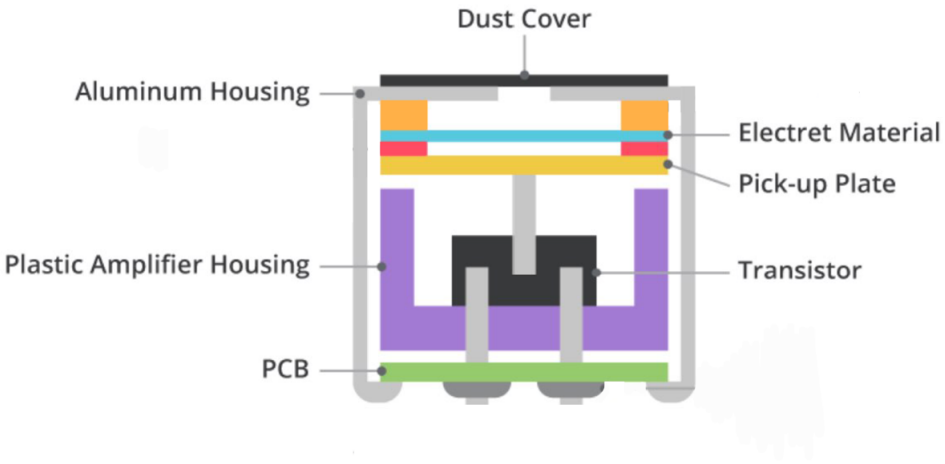


Figure 3.10: Diagram of ECM microphone [2]

Shown in Figure 3.10 above, an ECM utilises a hole, an electret material as a diaphragm, along with a conductive plate to form a capacitor. Like the MEMS microphone, the capacitor produces an electric signal as the diaphragm moves. These electric signals are then amplified using a junction-gate field-effect transistor (JFET). The ECM only has an analogue output, requiring additional circuitry consisting of an Analogue to Digital Converter (ADC) if a digital output is desired [2].

3.4.2 Sound Pressure Level

Both ECM and MEMS are similar in utilising a diaphragm and a conductive plate to form a capacitor. This capacitor outputs a voltage as airwaves pass through a hole. Due to the use of the diaphragm

and conductive plate, the microphones have a limited amount of air pressure that will output a voltage before the diaphragm either touches the conductive plate or bends to a maximum. This maximum pressure is defined as the microphone's maximum Sound Pressure Level (SPL). This value determines how loud a sound can be before distortion in the recording occurs [38].

3.4.3 Communication Standards

All microphones receive an analogue audio signal at the input, which, if required, is then sampled using an ADC to produce a digital signal. This digital signal is then converted to a specified communication standard format for interfacing. The two most common digital communication standards used for audio are I2S and PDM.

Pulse Density Modulation

PDM is a communication standard representing a sampled signal as a stream of bits. This method is commonly used for audio due to its simplicity and ability to represent higher-quality audio. PDM communication is implemented by outputting a stream of constant width pulses from a single pin. The information within the pulse stream is embedded in the changing period of the pulses. A low pass filter can be applied to the output signal which will integrate the bit stream and produce the original analogue signal [39].

Inter-IC Sound

I2S is a communication standard designed specifically for the communication of digital audio signals. I2S utilises three lines to communicate between a transmitter and receiver. These lines consisting of a clock signal, a word select and a data line. The word select is used to specify the left or right channel used for stereo devices. The data line transmits a Pulse-code modulation (PCM) words containing the audio information [40]. A PCM word is a set of binary values representing the rounded instantaneous voltage of the sample of the signal received [41].

3.5 Summary of Theory

The theory over-viewed in this chapter began by discussing the basic principles of ToA localisation; this concept was then expanded to TDoA triangulation. TDoA triangulation aims to determine the intersection of the TDoA curves which corresponds to the transmitter's coordinates. After this, the basic theory behind different forms of synchronisation was discussed, consisting of GPS, NTP servers and calibration signals. Within this section, various clock non-idealities were discussed, providing a better understanding of why synchronisation is subject to the clock's performance. Following the background to synchronisation was a brief theory section on ToA estimation where cross-correlation and GCC-PHAT were discussed and contrasted. This section also illustrated that GCC-PHAT minimises side-lobes of a cross-correlation by including a weighting function. Within the theory of correlation functions was a brief background into frequency chirps and why they are commonly used for signal detection. Following the chirp's theory is an overview of the different microphones widely used in similar applications and a brief background on various forms of communication between these microphones. Overall, the theory discussed in this chapter provides the necessary background to

implement acoustic triangulation. The subsequent chapter applies the theory to produce a network of devices that implements acoustic triangulation using TDoA.

Chapter 4

System Design

4.1 Design Process

The following section describes the design process and approach taken to implementing acoustic triangulation using the theory discussed in chapter 3. Implementing acoustic triangulation required multiple design iterations before the final system was selected. An agile approach is taken, where goals are split into smaller iterative tasks, which all combine to form one more extensive system. This design approach allowed for an adaptive and flexible design environment better suited for projects that are likely to be adjusted and redesigned as more information comes to light. As a consequence, the approach required that the tasks were split up with fine granularity to avoid a bottleneck in the design process.

4.1.1 System Design Pipeline

The general order of implementation for this project began with hardware selection as the project was time-constrained, and the process of acquiring the components will delay the project's progression. Following this, a general system structure was designed, which outlines the physical setup, general behaviour and expected inputs and output of the system's modules. Once complete, a simulation was then implemented and utilised for post-processing design. The post-processing consists of 3 stages, synchronisation, TDoA estimation and triangulation. Following the completion of each stage, the module was validated and either redesigned or added to the larger system. Once complete, the physical system was then designed and validated.

The flowchart below depicts the design pipeline.

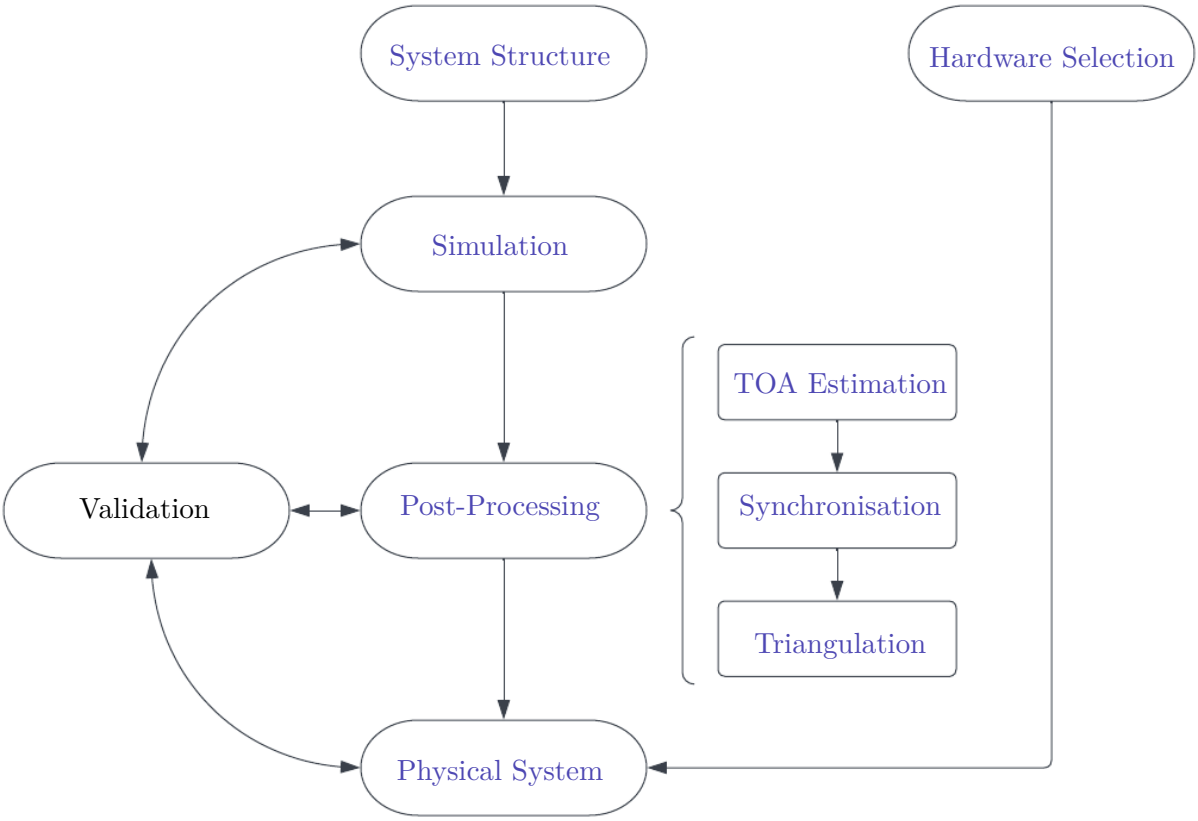


Figure 4.1: Flowchart showing the design pipeline of the acoustic triangulation system

4.2 System Structure

The design of the system structure aimed to provide the system framework. This section determined the number of devices used, the grid structure and the signal used for triangulation. Together, these design choices provided the information required to begin designing simulation.

4.2.1 Grid Structure

As explained in section 3.1.2, in order to triangulate a signal source, a minimum of 3 receivers is required. A system with four receivers was used as this allowed for less dependency on each receiver. Thus, the system could still triangulate the signal if one receiver had an issue.

Each receiver was placed on a corner of a rectangular grid; the system was designed only to triangulate a signal within this rectangle. A rectangle was used as it allowed for more straightforward calculations when implementing the triangulation algorithm described in triangulation design. The figure below illustrates the grid setup.

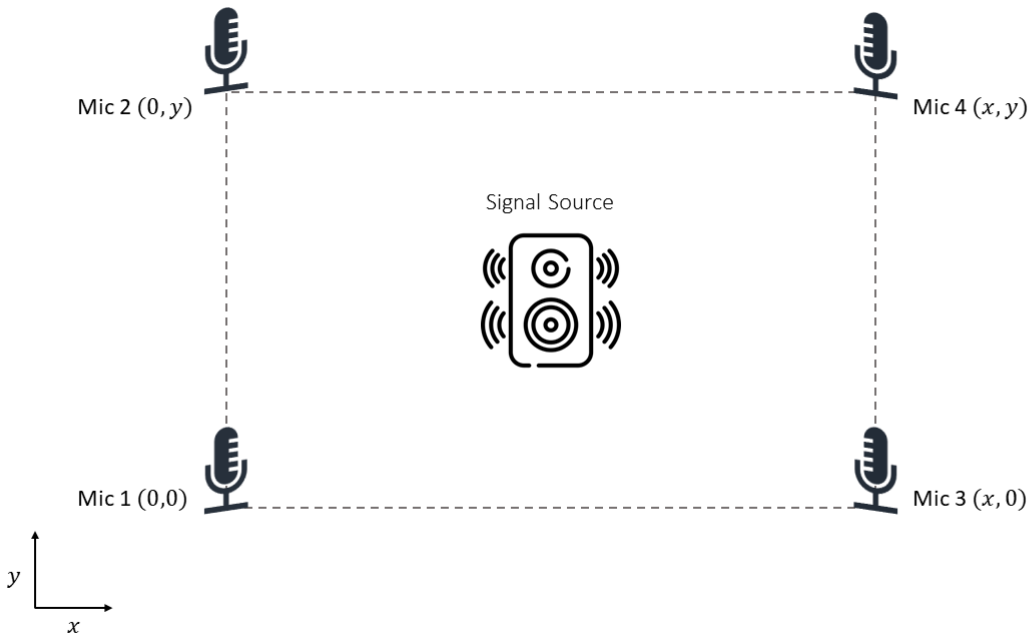


Figure 4.2: Diagram of receiver Grid used for triangulating a signal source

In Figure 4.2 above, the microphone on the bottom left corner of the grid, denoted as 'Mic 1' defines the origin point with co-ordinates of $(0, 0)$. 'Mic 2' defines the Y axis with co-ordinates $(0, y)$, 'Mic 3' is at point $(x, 0)$, defining the X axis and 'Mic 4' is defines the far corner of the grid at position (x, y).

4.2.2 Signal Selection

As emphasised in chapter 2, the triangulation performance is subject to the accuracy of the ToA estimations. As explained in chapter 3, the standard approach to ToA estimation is through a correlation function, determining the lag at which one signal is most similar to another. A factor affecting the accuracy of the ToA estimation is the properties of the signal which is correlated. In order to maximise the likelihood of accurate ToA estimation the signal used should have certain desirable auto-correlation properties[1]. The signal's auto-correlation should ideally have an unambiguous peak, a small peak width and low side lobes. This will allow for easier and more accurate identification of the ToA of the signal. A signal which satisfies these requirements is a frequency chirp due to its non-repetitive nature as described in chapter 3. Hence, the selected calibration signal and Signal of Interest (SoI)[2] for this project was a frequency chirp which ranges from $0Hz$ to $15000Hz$ over 5 seconds. The image below depicts the chirp, its spectrogram, and auto-correlation.

[1]An auto-correlation is the process of correlating a function with itself, which provides insight into any repeated patterns within the signal such as periodicity

[2]The SoI is the signal whose source is to be triangulated

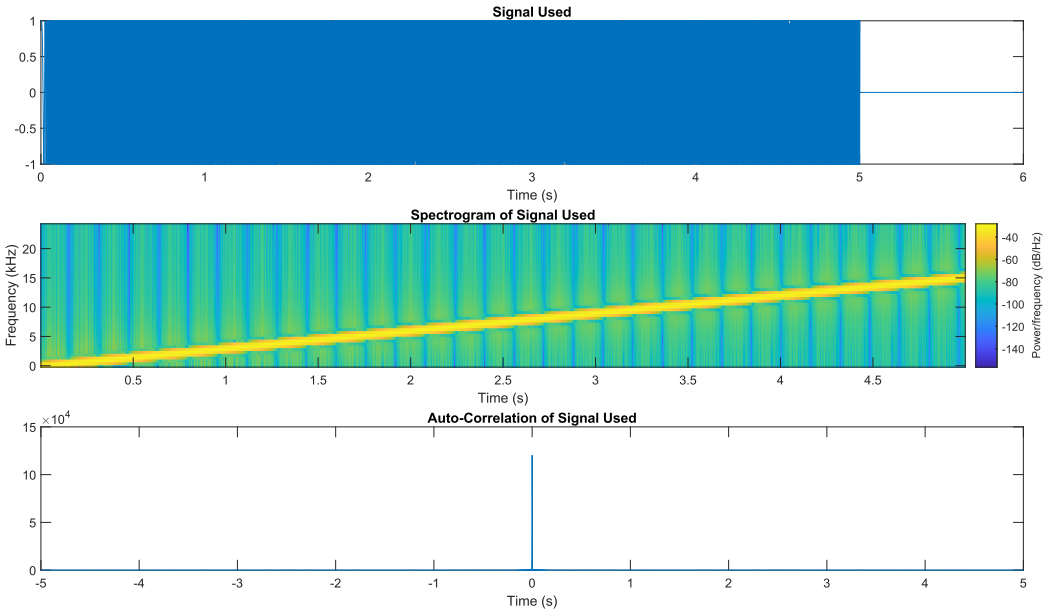


Figure 4.3: A figure of the signal used its' spectrogram and auto-correlation function.

Figure 4.3 above depicts a spectrogram of the signal, showing a $0Hz - 15000Hz$ frequency chirp over 5 seconds as well as an auto-correlation which has a sharp unambiguous peak as desired. The time-bandwidth product, $B\tau$ of this signal is 75000. Using Equation 3.11 the resulting compression gain is 150000 or $51.76dB$. This value provides insight into a theoretical minimum SNR required for accurate signal detection.

Following the design of the system's framework, a simulation was then implemented using these design choices.

4.3 Simulation Design

The software utilised for simulating the system is Flexible Extensible Radar Simulator (FERS). FERS is a software designed to simulate the performance and output of radar and sonar systems which can be extended to acoustic applications. FERS allows for the setup of multiple transmitters and receivers. The software can also simulate system hardware parameters, including ADC sample rates and ADC non-idealities, which can incorporate drift and jitter. These factors make this simulation software an ideal application for this project, capable of emulating physical systems and their outputs.

A goal for both the simulation and practical implementations was to maintain consistency and similarity between the two methods. Ideally, the simulation should be designed to take in the same inputs and generate the same outputs as the physical implementation; furthermore, when the physical system is complete, it can easily replace the simulation as the interfaces are the same.

4.3.1 Simulation Inputs and Outputs

The inputs to the system are the signals produced by the transmitter, and the outputs are the signals received by the receivers. The input read by the transmitter is in the form of an H5 file. This file contains the signal to be triangulated. The output of the simulation is an H5 file produced by each reliever.

4.3.2 H5 File

An H5 file contains data in a hierarchical data format which is designed to support large complex data sets. The H5 file is created using MATLAB's 'hdf5write' function. This function takes in three parameters, the H5 file name, the imaginary and real values of the signal to be transmitted, which in this context is the above mentioned 0 Hz to 15000 Hz, 5-second chirp.

4.3.3 XML File Design

The simulation is generated using a Extensible Markup Language (XML) file, which describes the scene of the simulation. An XML file allows a text-based format describing the data that forms the simulation environment. The FERS XML file begins by reading in H5 files containing the signals to be transmitted. It then follows by setting the general parameters of the overall simulation. The script then defines the parameters of the transmitters, receivers and clocks.

4.3.4 Simulation Parameters

The simulation parameters include the definition of the simulation's start time and end time, as well as the sample rate of the simulation. Another parameter which is adjusted for acoustic applications is the propagation speed, c, as the default is the speed of light, whereas this application's propagation velocity is the speed of sound, which is $\pm 343 m.s^{-1}$.

4.3.5 Clocks

FERS allows for a different clock to be defined for each transmitter and receiver. When creating a transmitter or receiver, the clock is then referenced. In addition, the frequency drift and clock jitter parameters can be set when defining a clock. When inputting a value for jitter and drift, FERS selects a random value between 0 and the inputted value when implementing the simulation. The ability to create different clocks with drift and jitter allows the simulation to emulate some non-idealities of the physical system.

4.3.6 Transmitters and Receivers

Within this simulation the transmitter represents a speaker and the receiver a microphone. The parameters inputted when defining a transmitter and receiver are mostly the same. The process of creating a transmitter or receiver begins with defining its' co-ordinates followed by the start time for recording and transmission. One also defines the clock that each device references, the signal transmitted and the antenna type. The antenna used in this application is isotropic as the signal should be transmitted to and received from all directions. An example of a FERS simulation is shown in section 7.6 in the appendix.

4.4 Simulation Validation

Once the simulation is designed, it is then validated. The validation aims to assess that the simulation and grid are set up correctly and behave as expected. The validation is based on the assumption that the software works as the FERS wiki suggest, validating the FERS software is beyond the scope of the project.

4.4.1 Simulation Validation Methodology

The simulation uses four receivers in the corners of a 200 m×200 m grid and a transmitter at the centre. The transmitter then plays a single 0 Hz-15 Hz 5-second chirp. The reason for using this chirp and not the 0 Hz-15 kHz chirp that will be used for triangulation is to allow for easier visualisation of the chirp's change in frequency, allowing for an easier graphical analysis of any signal distortions. The system parameters and clocks are all set to ideal values.

The expected outcomes are as follows:

1. The simulation will output four h5 files which contain signals received by each receiver.
2. The signals received will be an undistorted replica of the original chirp.
3. The ToA of the chirp will be **0.4123** s for all receivers.[3]

4.4.2 Simulation Validation Results

The following images depicts each receiver's outputted files.

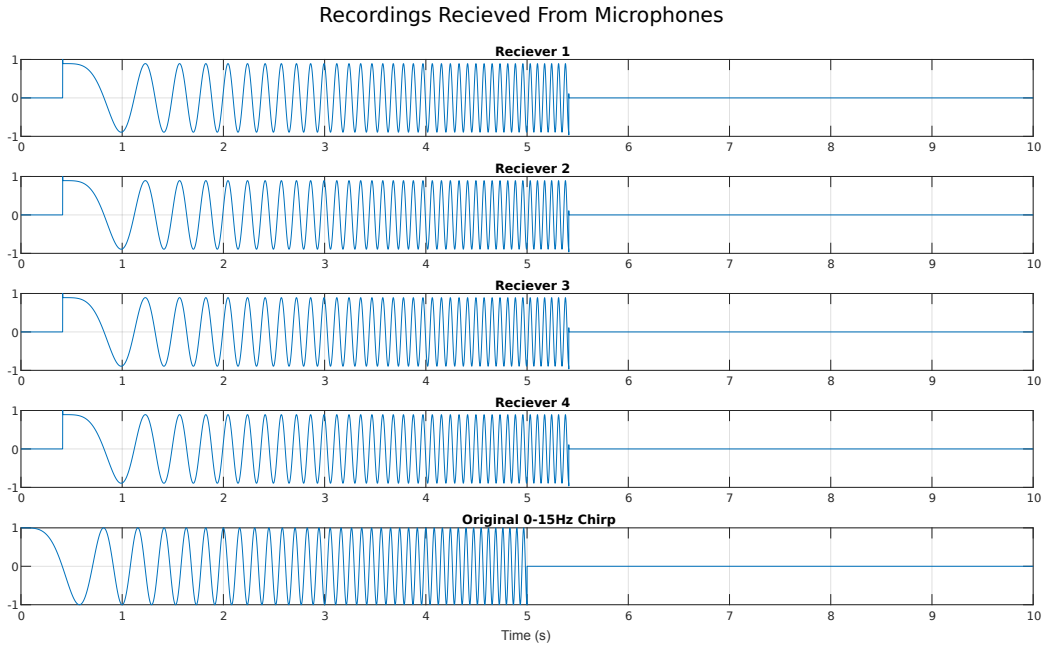


Figure 4.4: Recordings of signal Received by 4 receivers

[3]All receivers are $\mathbf{141.42}\boldsymbol{m}$ from the transmitter and the propagation velocity is $\mathbf{343}\boldsymbol{m.s^{-1}}$ therefore the ToA $= \mathbf{141.42 \div 343 = 0.4123}$

Figure 4.4 above depicts the four recordings received as well as the signal transmitted at the bottom. The figure shows that all four receivers received an undistorted version of the original signal. The following figure displays the ToA of the received signals using methods described in section 4.5.

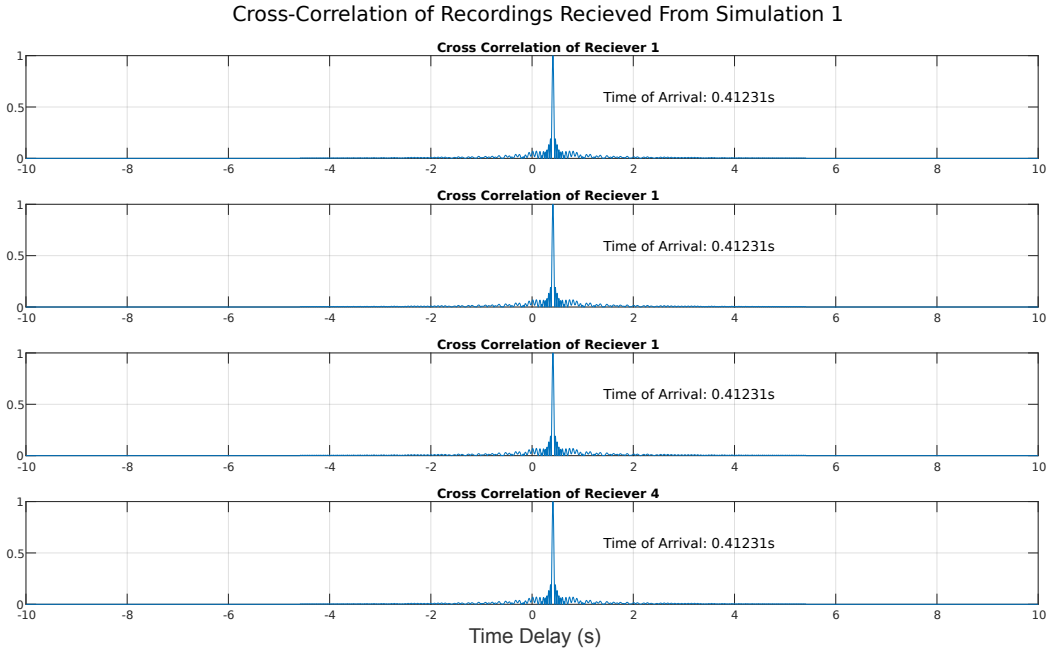


Figure 4.5: Cross-Correlation of Recordings Received by 4 receivers

Figure 4.5 above depicts the time of arrivals for all receivers being **0.4123** s. Together Figure 4.4 and Figure 4.5 validate expected outcome 1, as four H5 files containing the transmitted signals are outputted, outcome 2 as all receivers depict an undistorted version of the original signal and expected outcome 3 as all ToAs are at **0.4123** s. Following the validation of the simulation setup, the simulation is then utilised to assist in the post-processing design.

4.5 Post-Processing Design

The system's post-processing is split into three parts. The first is ToA estimation design, where the method is discussed and implemented. Once the system is capable of accurately estimating the ToA then synchronisation can be implemented using the theory discussed in section 3.2. The final design stage consists of the triangulation algorithm, which utilises the TDoA values to triangulate the source.

4.5.1 Time of Arrival Estimation Design

As explained in section 3.3, the ToA of the signals is determined through the means of a coherency function, which outputs the similarity between a signal and a reference signal at different times. The method used for ToA estimation is cross-correlation. This function is implemented using Matlab's 'xcorr' function. The function takes in the signal received and the reference signal. It outputs two arrays, one containing the correlation of the two signals at each time delay and the other array containing

the time delays. The ToA estimation is defined as the time delay corresponding to the maximum correlation value.

The function implementation is shown below:

```
[Correlation_Array , Delay_Array] = xcorr(Signal_Received, Reference_Signal);

[ ~ , TOA_Index] = max(Correlation_Array);     % Determine index of maximum correlation

TOA = Delay_Array(TOA_Index);                 % Obtain TOA from delay array using index
```

It was only later in the design process that it was found that GCC-PHAT is the preferred option for ToA estimation. However, an attempt to substitute GCC-PHAT for cross-correlation was unsuccessful. For reasons to be explained in subsection 4.5.2, the receiver will record the same reference signal twice within the recording, and the ToA of both occurrences need to be determined. The repeated signal gives rise to an issue with GCC-PHAT.

As explained in section 3.3, GCC-PHAT is the same as cross-correlation with an added weighting function. The weight function aims to increase the performance in a reverberant environment which contains echos. The added weighting results in an output with its' maximum occurring at the same delay as the cross-correlation maximum, but all other correlation values are minimised or distorted. For this reason, GCC-PHAT does not perform well when identifying more than one occurrence of a reference signal within a recording, as it is distorted in the same manner as an echo. The following figures depict the result of GCC-PHAT and cross-correlation when determining the ToA of two reference signals occurring at $\boldsymbol{t = 2s}$ and $\boldsymbol{t = 7s}$. The first chirp has an amplitude of $\mathbf{1}\boldsymbol{v}$, whereas the second has an amplitude of $\mathbf{0.5}\boldsymbol{v}$.

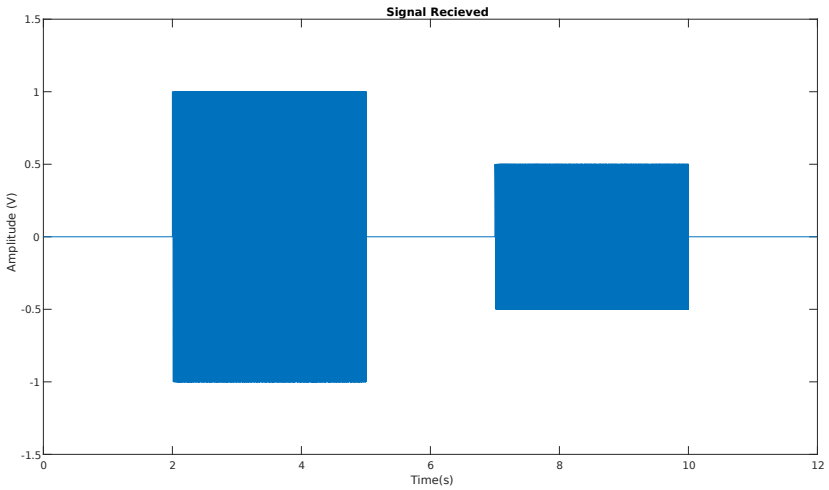


Figure 4.6: Repeated reference signal with different amplitudes

Figure 4.6 above depicts a signal containing the same chirp which occurs twice with different amplitudes.

The amplitudes of the two signals are different to emphasise the issue with GCC-PHAT in this context. The following figure depicts the output of both the GCC-PHAT and cross-correlation of the signal received and the reference signal.

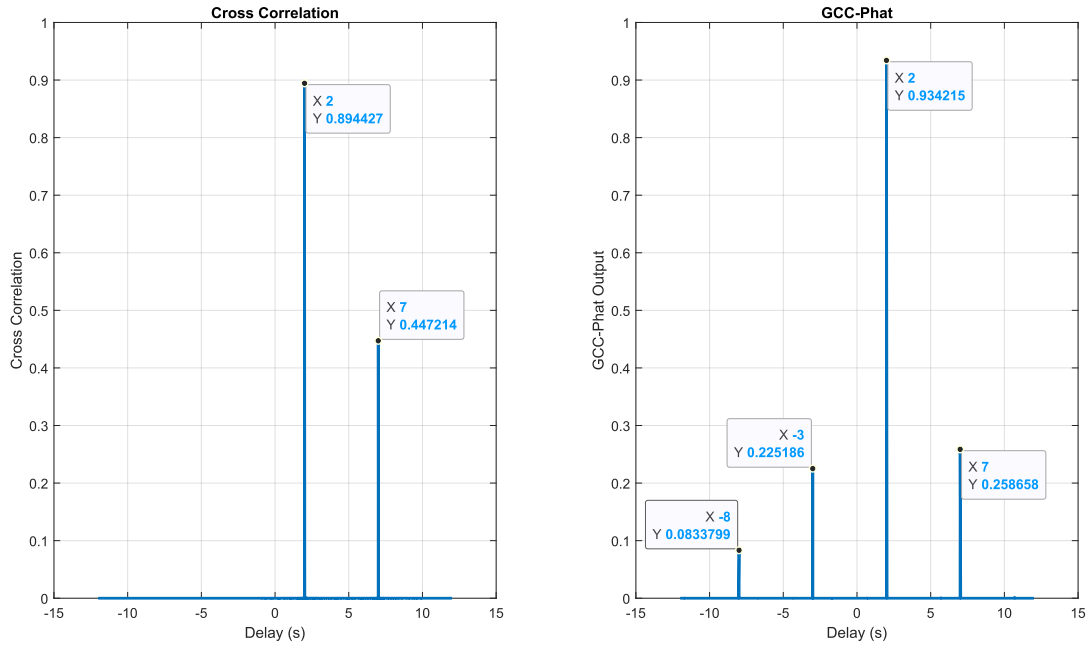


Figure 4.7: Cross Correlation and GCC-PHAT of the signal received and reference signal

In Figure 4.7 above, both GCC-PHAT and cross-correlation's maximum occurs at values occur at $\boldsymbol{t = 2s}$ and the second largest value occurs at $\boldsymbol{t = 7s}$. Although this is the correct outcome for both coherency functions, the second signal introduces undesirable peaks when using GCC-PHAT. This brings light to an issue that can arise in an uncontrolled environment. When implemented in a physical system, the incorrect peak is occasionally selected as noise increases the height of the undesired peaks, leading to ambiguous ToAs. A solution to this issue is to utilise two different signals instead of the same reference signal for both calibration and triangulation and to perform GCC-PHAT for both signals separately. However, this was not implemented due to time constraints and the satisfactory performance of the cross-correlation.

4.5.2 Synchronisation Design

Ideally, the ToA estimations for each receiver will be relative to the same recording start time, i.e., all devices record in a synchronised manner. Synchronisation is an issue which arises when implementing hardware, as the simulation is designed for all devices to begin recording at the same time. However, following unsuccessful attempts at other synchronisation approaches explained in section 4.8, the decided synchronisation method would be through the use of a calibration signal.

A concern regarding this synchronisation method is that it is rarely used in TDoA acoustic triangulation. As mentioned in the chapter 2, Samuelson attempted to implement this method of synchronisation but was unable to triangulate a sound source [15]. Furthermore, this approach is yet to be successfully

implemented in the context of acoustic triangulation using TDoA. However, this method was the most viable option due to the project's hardware and cost constraints. The approach is based on the theory explained in section 3.2 and is implemented as follows.

The recording received from a single device is expected to have two recorded signals, the calibration signal and the SoI. A function called 'find_TOAs' is used to extract the ToAs of these two signals. The function's arguments are the recording and the reference signal to which the recording will be correlated. The correlation of the reference signal and the recording should result in two peaks, one for each signal. The function then returns the two peaks' ToAs.

The following figures provide a graphical depiction of the synchronisation process described in section 3.2 when applied to a physical recording.

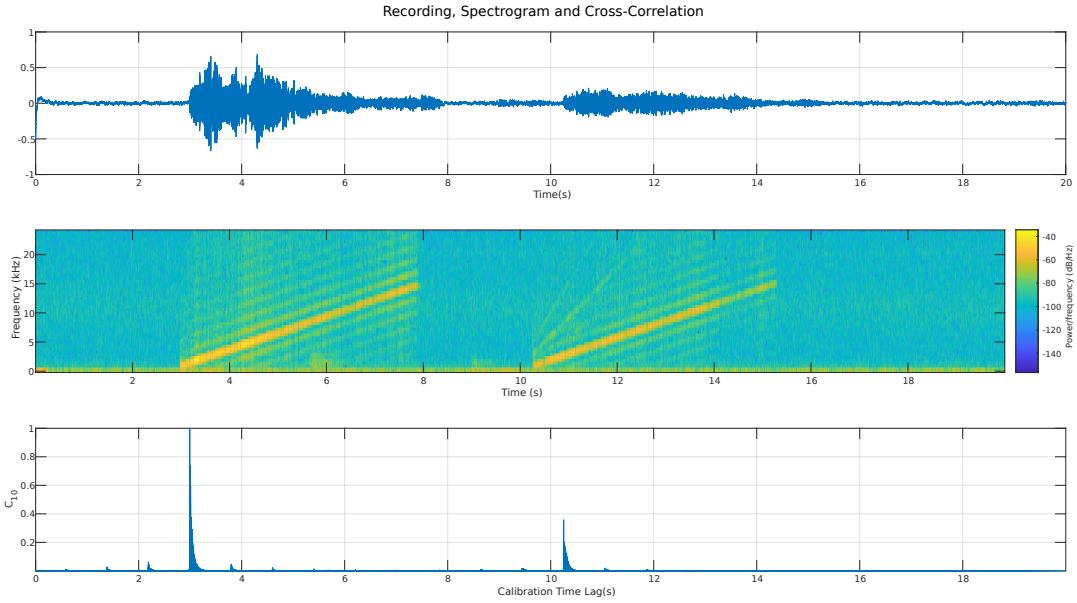


Figure 4.8: Figure of Received Recording's Amplitude, Spectrogram and Cross-Correlation Output

The first plot in Figure 4.8 depicts a recording of the calibration signal and SoI. The spectrogram below shows the two 0 Hz - 15 kHz chirps. The bottom plot shows the cross-correlation of the recording and the reference signal. The correlation peaks are aligned with the start of the two signals in the recording and spectrogram. The following figure is an annotated graphic of the four devices' recordings' correlated to the reference signal.

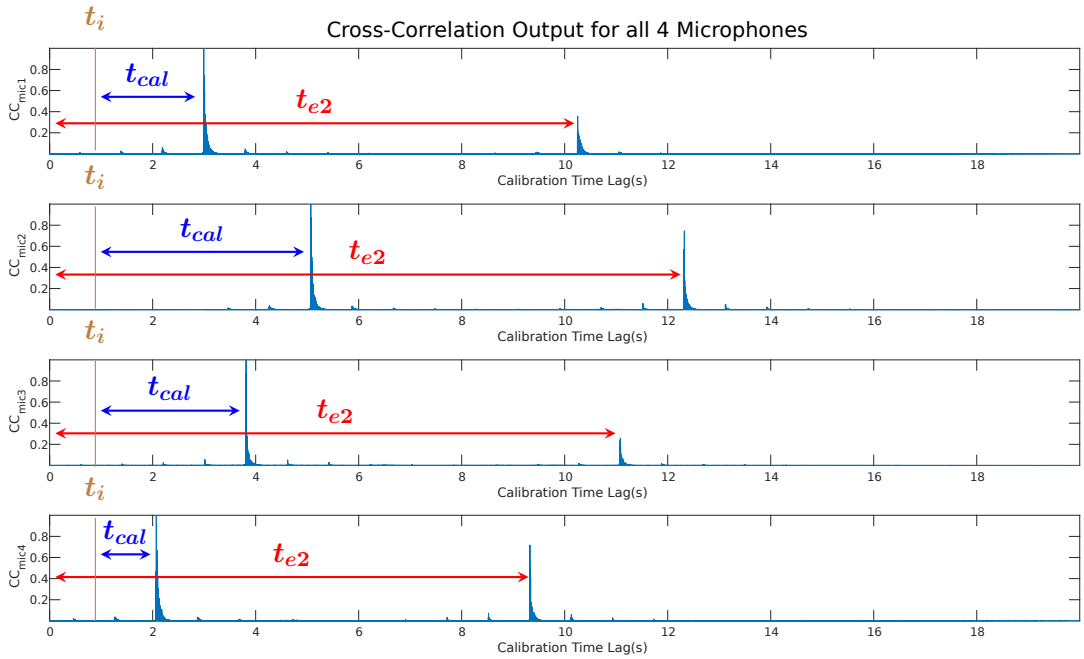


Figure 4.9: Annotated Cross Correlation of four receiver's recordings

In Figure 4.9 above, four cross-correlations are shown, each containing two peaks. Within the graphic, the value $\boldsymbol{t_{cal}}$ is defined as the ToA of the calibration signal relative to the ideal ToA of the signal defined as $\boldsymbol{t_i}$. In this instance, the calibration signal source is equidistant from all devices. Therefore, all recordings will have the same $\boldsymbol{t_i}$. Also depicted is $\boldsymbol{t_{e2}}$ which is the ToA of the SoI relative to the start of the recording. The synchronised ToA of the SoI is defined as $\boldsymbol{t_{e2} - t_{cal}}$. The following image depicts this process, but instead of subtraction, a circular shift is used for a graphical explanation.

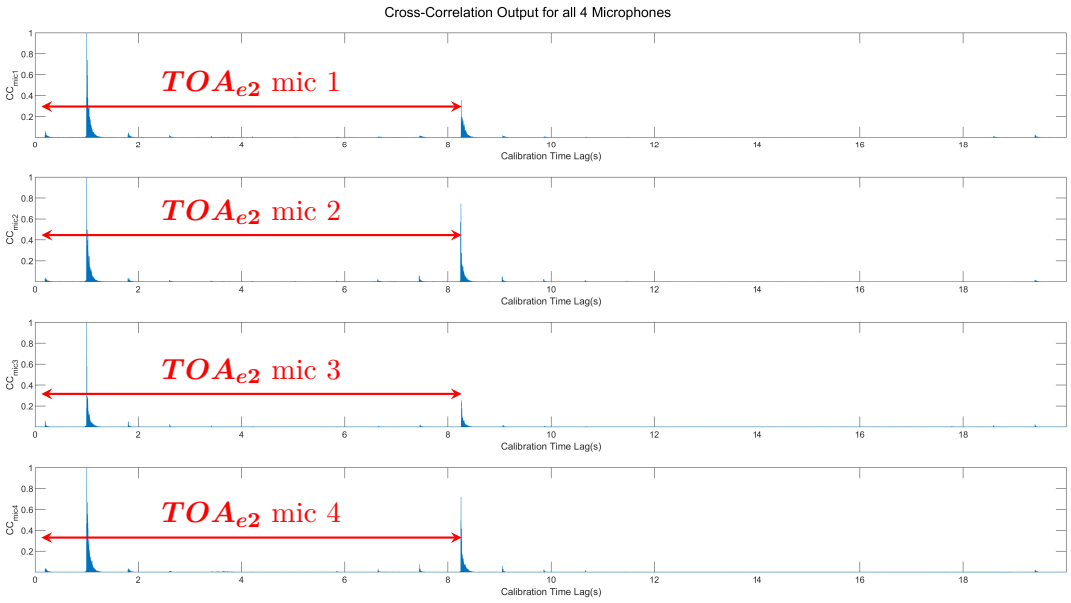


Figure 4.10: Annotated time aligned Cross Correlation of four receiver's recordings

In Figure 4.10 above, the first signal occurs simultaneously relative to the start of each device's recording, thereby showing that all four devices' recordings are synchronised.

The synchronised ToA values are then used to obtain the TDoA between each receiver. With access to four different ToAs, six different TDoAs can be produced. One can then utilise these six values as arguments to the triangulation algorithm.

4.5.3 Calibration Signal Position

In Figure 4.9, the ideal ToA of the calibration signal, $\boldsymbol{t_i}$ is the same for all devices as the calibration source is in the centre of the grid, equidistant to all receivers. However, the calibration source is not limited to this position as $\boldsymbol{t_i}$ can be determined for all grid points. Therefore in the idealised scenario, the choice of the calibration position is arbitrary.

This statement is confirmed using two FERS simulations. The simulations assess the synchronisation accuracy of the system when calibrated from different points on the grid. The two calibration points selected are the grid centre and a grid corner. The simulations are set up to play the calibration signal from the given position followed by the SoI from the same position. If the synchronised ToA of the SoI is the same as the ideal time $\boldsymbol{t_i}$ then the system is correctly synchronised. The simulation results showed that the choice of calibration position is arbitrary as all SoI's ToAs were as expected. The full results of both simulations are shown in appendices section 7.1.

Although this result shows that the choice of the calibration signal position is arbitrary, this is in idealised conditions. All microphones have a maximum rated SPL. This value determines the maximum sound pressure the microphone can handle before the signal is distorted. When a sound source is placed next to the microphone, the SPL may be exceeded, leading to signal distortion. To avoid possible

issues relating to the SPL, the calibration source is placed in the centre of the grid when implementing the system. Nonetheless, a system which calibrates from the same position as a receiver should be considered in future work as it is easier to set up because it does not require measuring the centre of the grid.

4.5.4 Triangulation

The triangulation algorithm utilises the TDoAs between the receivers and attempts to identify the SoI's coordinates. As explained in subsection 3.1.2, the triangulation algorithm identifies the intersection of the hyperbole generated from the TDoA values. Although chapter 2 mentions different methods for solving this intersection, they are complex; accordingly, a more straightforward approach is taken to triangulate the SoI. The approach taken makes use of a Look Up Table (LUT). The LUT is structured in the following manner:

$\boldsymbol{TDOA_{41}}$	$\boldsymbol{TDOA_{31}}$	$\boldsymbol{TDOA_{21}}$	$\boldsymbol{TDOA_{42}}$	$\boldsymbol{TDOA_{32}}$	$\boldsymbol{TDOA_{43}}$	$\boldsymbol{X}$	$\boldsymbol{Y}$

Table 4.1: Column Headers for LUT

The first six columns of the LUT shown in Table 4.1 are the TDoAs for each receiver combination. The final two columns show the corresponding $\boldsymbol{X, Y}$ coordinates of of the signal that will produce those TDoA values. The LUT is essentially a table containing all combinations of TDoAs and the intersection of their corresponding hyperbolas. The LUT is generated by determining the ideal ToA of a signal at every point in a grid with a specified resolution. The LUT is constructed by looping through all grid positions. Within each loop, the propagation time of a sound from that position to each receiver is appended to the table. The algorithm which generates the LUT can be found in the appendices section 7.2.

The LUT works by returning the X and Y value of the row, which most closely resemble all the TDoA values extracted from the recordings. As shown in Figure 3.3, one TDoA corresponds to all points on a curve, but the intersection of the curves corresponds to a point. Therefore the row selected should resemble **all** the TDoA values received rather than resembling some TDoAs very closely and others not as much. By selecting a row in which all elements most closely resemble the TDoAs received, then the point of intersection is prioritised over the correct curve. This is done by finding the minimum sum of the squared absolute differences between the TDoAs received and the TDoAs in the LUT. By squaring the difference, a more significant difference between two TDoA values has a more significant effect on the position than a minor difference. The following example illustrates this concept using TDoAs between three receivers:

$\boldsymbol{TDOA_{31} = 3}$	$\boldsymbol{TDOA_{21} = 2}$

Table 4.2: Example of TDOA values of a signal

Table 4.2 above depicts the TDoAs received from three receivers. The following table depicts the LUT used for triangulation.

$\boldsymbol{TDOA_{31} = 2}$	$\boldsymbol{TDOA_{21} = 3}$	$\boldsymbol{X = 3}$	$\boldsymbol{Y = 3}$
$\boldsymbol{TDOA_{31} = 5}$	$\boldsymbol{TDOA_{21} = 2}$	$\boldsymbol{X = 2}$	$\boldsymbol{Y = 1}$

Table 4.3: Example of Look Up Table used for triangulation

In this example, the sum of the differences between the TDoAs received in Table 4.2 and the two rows in the LUT shown in Table 4.3 are the same.
Row 1 = $\boldsymbol{|3-2| + |2-3| = 2}$
Row 2 = $\boldsymbol{|3-5| + |2-2| = 2}$

whereas the squared distances result in the following:
Row 1 = $\boldsymbol{|3-2|^2 + |2-3|^2 = 2}$
Row 2 = $\boldsymbol{|3-5|^2 + |2-2|^2 = 4}$

This algorithm will hence select Row 1's co-ordinates, $\boldsymbol{X = 3}$, $\boldsymbol{Y = 3}$ as the received TDoAs as it is element by element more similar to the received values.

Initially, the algorithm implemented this process on all six TDoAs in the same array, which raises a possible issue. Suppose a single device has a sampling error or other issue. In that case, all TDoAs relative to that device will significantly differ from the ideal LUT row, and the correct coordinates will not be selected. Therefore to minimise the reliance on every device working as desired, the system will split the row of TDoAs into four subsets. Each subset excludes a different device, i.e. the subset excluding device four consists of $\boldsymbol{TDOA_{31}}$ and $\boldsymbol{TDOA_{21}}$. This process will result in four different possible estimated positions of the SoI.

Following this adjustment, further processing is required to select one of the four possible solutions. It was found that when a single device has some sampling error, all TDoA's, which include that device, pull the estimated position to the edge of the grid. Therefore, for this reason, the system limits possible positions to exclude the edges of the grid, and all estimates on the grid's edge are ignored. Given the situation when multiple solutions are inside the grid's bounds, the average of the estimated positions is selected. If the average position selected has an error greater than 1.5 m, then the most accurate estimation from the possible solutions is manually selected[4].

4.6 Post-Processing Validation

The post-processing validation aims to asses the outputs of the ToA estimation, the signal synchronisation methodology, and the use of the triangulation LUT.

[4]Ideally the system should not require manual inputs but based on the system objectives, the goal is to implement a system capable of acoustic triangulation. Optimising this process is beyond the scope.

4.6.1 Post-Processing Validation Methodology

The assessments was implemented using FERS simulations. The validation of the ToA estimation will utilise the same simulation and results as subsection 4.4.2 illustrated in Figure 4.4 as the validation of the simulation design encompassed the validation of ToA estimation. The simulation assessing synchronisation and triangulation was set up as follows.

- The four receivers was placed on the corners of a 200 m×200 m grid.
- Each receiver will reference the same clock, with no frequency or phase error[5].
- The calibration signal occurred from the centre of the grid at $\boldsymbol{X = 100}$ m, $\boldsymbol{Y = 100}$ m
- The SoI occurred from co-ordinates $\boldsymbol{X = 80}$ m, $\boldsymbol{Y = 130}$ m
- Both signals were a 0 Hz-15000 Hz 3 second chirp.
- All received recordings were circular shifted by some random value between 0 seconds and 5 seconds to assess the synchronisation process.
- centimetre resolution was used for the LUT.

The expected outcome for the post processing are as follows:

1. The ToA estimation using cross-correlation correctly estimates the time the signal is detected.
2. Once all recordings are synchronised, the ToA of the calibration signal will be the same as the ideal ToA, $\boldsymbol{t_i}$.
3. The LUT accurately predicts the co-ordinates of the SoI using the TDoA values.

4.6.2 Time of Arrival Validation Results

The time of arrival validation is done using the same simulation and results as subsection 4.4.2. As explained, the expected ToA of the signal transmitted from the centre of the **200** m×**200** m grid is **0.4123 seconds**. Figure 4.5 depicts the estimated ToA's at precisely the expected ToA, thereby validating point 1 as the cross-correlation accurately identifies the ToA of the signal.

4.6.3 Synchronisation and Triangulation Validation Results

The following image depicts the 4 recordings received from the simulation designed above:

[5]This experiment aims only to assess the correct implementation of post processing and not the systems capabilities under non ideal conditions.

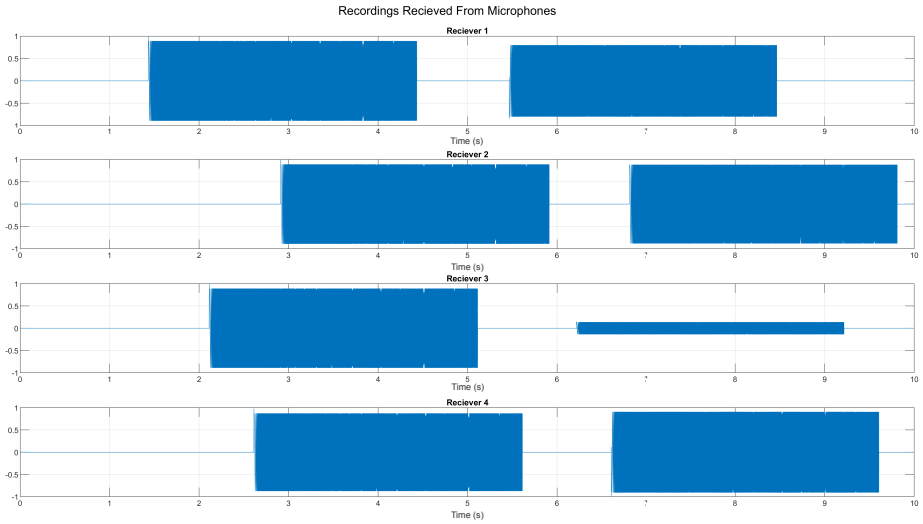


Figure 4.11: Recordings Received by 4 receivers

Figure 4.11 above depicts the four receiver's recordings. The figure shows all calibration signals occurring at a different time; thus, the recordings are not yet synchronised. Once the synchronisation process is implemented, the ideal ToA of the calibration signal, $\boldsymbol{t_i}$ is expected to be $\mathbf{0.41231\,s}$[6]. The following table shows the ToA of the calibration signal compared to $\boldsymbol{t_i}$ after the synchronisation process is implemented.

Receiver	$\boldsymbol{t_i}$**(ms)**	**TOA of SoI(ms)**	**Error**(ms)
1	412.307	412.307	**0**
2	412.307	412.307	**0**
3	412.307	412.307	**0**
4	412.307	412.307	**0**

Table 4.4: ToA of time shifted calibration signal from simulation

The resulting ToA of the calibration signal is the same as the expected outcome $\boldsymbol{t_i}$, thereby validating point 2 as the recordings have successfully been aligned based on the ToA of the calibration signal.

The following figure depicts the resulting estimated position based on the TDoAs of the SoI between each receiver.

[6]All receivers are equidistant to the calibration signal which occurs from $\mathbf{100, 100}$ m therefore $\boldsymbol{t_i} = \frac{\sqrt{100^2+100^2}}{343} = \mathbf{0.41231}$ s

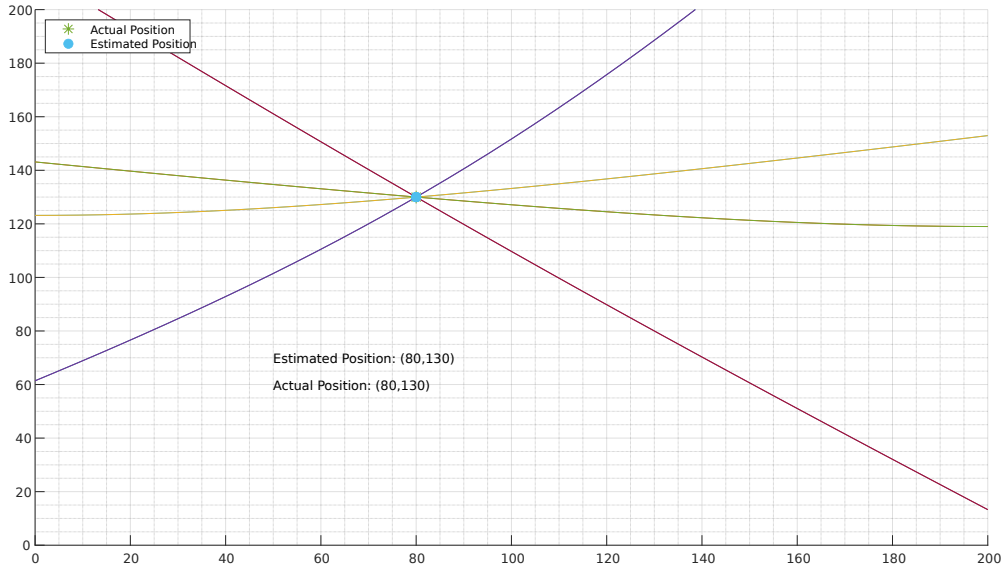


Figure 4.12: A figure of the estimated position of the signal source using look up table and TDOA hyperbolas intersecting the signal source

Figure 4.12 above depicts the estimated position resulting from the triangulation algorithm, which uses the LUT. The hyperbolas are drawn by using Equation 3.5 based on the TDoA values. As shown in the image, the estimated position lies on the intersection of the hyperbolas at point $\boldsymbol{X = 80, Y = 130}$ on the grid, which is the same as the SoI. This result validates expected outcome 3 as the triangulation algorithm using the LUT results in the correct position estimation.

4.7 Hardware Selection

The hardware components required for this project are a microphone and a microcontroller/computer. The hardware selection process began sequentially, where an attempt was made to find a board that suited the already available microphones. However, this process was unsuccessful due to interfacing issues and a lack of suitable board availability. It was then decided that the hardware selection process should progress in a parallel manner where the board and microphones are selected in conjunction rather than selecting one component followed by selecting another that suits its' interfaces and requirements. The selection process begins by analysing the various hardware requirements for this system.

4.7.1 Board Requirements

Based on system objectives, the hardware requires that the system should have some form of wireless communication.As explained in section 3.4, microphones are either digital or analogue. The digital microphones commonly interface using either PDM or I2S. If the selected microphone is digital, then the communication interface used should be supported by the board. If the microphone has an analogue output, the board must have a reliable ADC.

Another factor considered is the board's community. A board with a strong community will have many

resources, libraries and support available. Since this project is time-constrained, a strong community may speed up the development process as the issues faced in the project are likely already solved and explained on public forums. As explained, the project has a short development time frame. Thus the final requirement is that the board should be locally available.

A summary of these requirements is as follows:

1. Wireless Communication
2. I2S, PDM or ADC
3. Strong Community
4. Locally Available

4.7.2 Microphone Requirements

An explicit requirement for the hardware is that it should be capable of triangulating an **acoustic** signal. Audible signals are within the range of 20 Hz to 20 kHz, furthermore according to the Nyquist sampling theorem, the minimum required sampling rate to identify a signal in this range is 40 kHz. Accordingly, the microphone should be capable of sampling at a rate $>$ **40** kHz.

Another means to measure a microphone's capability is through the microphone's self-noise or SNR. According to [42] a 'good' SNR for a microphone used for audio applications is $\mathbf{74}\,\boldsymbol{dB}$. However, the use of microphones for this project is for signal detection rather than high-quality audio applications. Thus this level SNR can be seen as a reference standard for higher-quality recording, and the microphone's SNR do not need to uphold this standard. Based on the expected compression gain explained in section 4.2, a SNR slightly short of this value is sufficient. Similar to the board requirements, the microphone selected should have a communication format capable of being interfaced with the chosen board, a strong community and be locally available.

A summary of the microphones requirements are as follows:

1. $>40\boldsymbol{kHz}$ sample rate
2. I2S, PDM or analogue
3. Strong Community
4. Locally Available

4.7.3 Hardware Options

The three boards which were selected for comparison are the ARDUINO UNO WiFi REV2, the ESP32 and the Raspberry Pi 0 W. The Arduino and Raspberry Pi were selected for comparison as both boards were available for use. The ESP32 was also considered, as it is commonly used in similar applications with many resources discussing the implementation of similar systems on an ESP32. The table depicting the boards' specifications is shown below.

Board	I2S	ADC	WiFi	Core	Software	Availability
ESP-WROOM-32	YES	YES	YES	XtensaLX6	Arduino IDE	Locally Available
Raspberry Pi 0 W	YES	NO	YES	BCM2835	Rasbian OS	Available
Aurduino Uno Wifi	NO	YES	YES	ATmega4809	Arduino IDE	Available

Table 4.5: Table of Board Comparison [43][44][45]

The 4 microphones selected for comparison are the Adafruit PDM MEMS microphone, the Waveshare Sound-sensor V2, the INMP441 and the Re-speaker 2 mics pihat. The Adafruit microphone and the Sound Sensor were compared as they were already available for use when the project began. On the other hand, the Respeaker and INMP441 were selected for comparison as they are both well-documented microphones which have been used in similar applications.

Microphone	Sample Rate	ADC SNR	Type	Interfacing	Availability
INMP441	$\leq$ **48** kHz	61 dBA	MEMS,Digital	I2S	Unavailable
SoundSensor	N/A	N/A	ECM, Analogue	Analogue	Available
Adafruit	$\leq$ **48** kHz	61 dBA	MEMS,Digital	PDM	Available
Respeaker	$\leq$ **48** kHz	94 dB	MEMS,Digital	I2S	Locally Available

Table 4.6: Table of Microphone Comparison [46][47][48][49]

The table above shows that all the microphones have a SNR $\geq$ 61dB besides the Sound Sensor V2 as the SNR is dependant on the ADC used. For this project's application, this level SNR is high enough to implement the system. All microphones analysed also have a high enough sample rate to record all signals in the audible spectrum. The INMP441 is a good option for a microphone, but due to its unavailability and the project's time constraints for ordering, it is not a viable choice.

ESP-WROOM-32

The ESP32 is a low-cost, low-power micro-controller, commonly used for Internet of Things (IoT) applications. The ESP32 is a viable option as it accommodates I2S, a standard interface for digital microphones. The board also has an ADC, which can interface with analogue microphones. The downfall of the ESP32 is that the ADC used in the ESP32 is a 12-bit Successive Approximation Register (SAR) ADC. This ADC is considered a non-ideal ADC for audio purposes as it is susceptible to noise and requires additional circuitry components and signal 'cleaning' to minimise this issue which is beyond the scope of this project [50][51]. The ADC also only has 12-bit resolution whereas the industry standard is between 16bit-32bit resolution for audio use[52]. Given these factors, the ESP32 should only be paired with a digital microphone leaving the INMP441 and the Adafruit PDM MEMS microphone as possible options for this board. The Re-speaker 2 mics Pi-hat is incompatible as the pin structure, and board are designed specifically for a Raspberry Pi.

Raspberry Pi 0W

The Raspberry Pi 0W is a micro-computer rather than a microcontroller as it can run an Operating System (OS). The Raspberry Pi does have I2S interface, making it a viable option for a digital

microphone. However, it does not have an ADC; consequently, it cannot interface with an analogue microphone without an external ADC. The University of Cape Town also has the board available without the need to order for use. An issue with the Raspberry Pi is that it does have PDM, but according to Adafruit support, the Raspberry pi does not implement PDM well due to Central Processing Unit (CPU) time and switching issues. This is because it is a single core Linux Based machine which is a time-sliced OS, which makes it difficult to accurately sample [53]. The Raspberry pi 0W can however interface with the INMP441 and the Respeaker 2 mics Pihat using I2S. The Raspberry Pi also has a very strong community with well-documented resources for audio applications and forums that provide additional support.

Arduino-Uno

Like the ESP32, the Arduino-Uno is a microcontroller commonly used for IoT and sensor networks. The Arduino does not have I2S or PDM and cannot interface with digital microphones. However, the board does have 10-bit SAR ADC with a maximum sampling rate of $9.615\boldsymbol{kHz}$[54], which is insufficient for audio purposes and accordingly is not a viable option for this use case.

The remaining microphone-board pairs are as follows:

1. ESP32 - Adafruit PDM MEMS Microphone
2. Raspberry Pi 0W - Respeaker 2 mics Pi-hat

Between these options, the Raspberry Pi and the Respeaker is selected for the following reasons:

- Raspberry Pi is available through the University of Cape Town
- The Respeaker is designed specifically for the Raspberry Pi, the system is well documented and easily configurable.

Figure 4.13 below is an image of the Raspberry Pi 0w with the Respeaker 2 mics Pi-Hat mounted ontop.

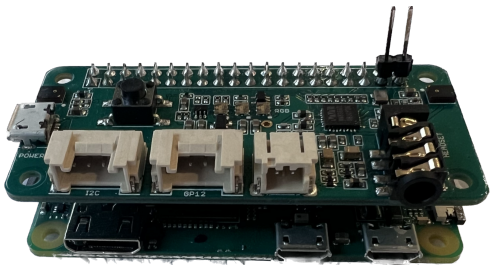


Figure 4.13: Image of Raspberry Pi and Respeaker 2 mics Pihat

4.7.4 Hardware Limitations

Although the hardware is carefully selected, various non-idealities that come with utilising any physical hardware should be considered. As emphasised, one of the critical aspects of TDoA is accurate timing. Accordingly, the hardware limitations focused on is the system's clock non-idealities.

The Re-speaker does not have an onboard clock; instead, it utilises an input clock from the Raspberry Pi. The Raspberry Pi's clock is driven by a 19.2MHz crystal oscillator which is subject to drift. Clock drift is generally defined in Parts Per Million (PPM) representing the clock's frequency deviation relative to 1 MHz.

The Raspberry Pi's crystal oscillator is rated to have a tolerance of $\pm\mathbf{20}$ PPM at $\mathbf{25\,deg}$ C and a maximum further drift of $\pm\mathbf{5}$ PPM in the first year [55]. Furthermore, the oscillator is physically placed next to the system's core which heats up due to use. According to [56] the Raspberry Pi 0s core reaches $\pm\mathbf{57\,deg}$ C at $\mathbf{35\%}$ load. Based on this, one can only assume the oscillator operates at temperatures beyond $25\,\mathbf{deg}$ C. Therefore, the minimum expected clock drift is $\pm\mathbf{25}\boldsymbol{PPM}$. Many experiments have been performed which assess the Raspberry Pi's clock drift. In one instance, an experiment was conducted to determine the clock's drift over 3 hours. The result of this experiment showed that the clock drifted **1.5** seconds [57] resulting in a $\mathbf{138}$ ppm[7]. Following these results, the Raspberry Pi's clock drift is expected to be in the range of **25** PPM and **138** PPM.

According to the Respeaker's data-sheet [58], the system uses a Sigma Delta ADC. The basic theory behind this ADC is that it obtains the average of the signal in a given period. This period is determined by the oversampling rate, i.e., if the master clock rate is 19.2MHz and the sample rate is 48kHz, the system will have an oversampling rate of:

$\frac{\mathbf{19.2}\boldsymbol{M}}{\mathbf{48}\boldsymbol{k}} = \mathbf{400}$[59].

When oversampling, the sampling frequency depends on the clock's frequency; consequently, any frequency error will propagate through the down-conversion, i.e., a 400Hz clock drift on a 19.2MHz clock will translate to a 1Hz drift at a 48kHz sample rate[8]. The system implemented utilises a sample rate of 48 kHz. Thus, the Raspberry Pi oscillator's minimum drift of 25ppm and maximum drift of 140ppm can be converted to a frequency error when sampled at 48kHz as follows.

$$\boldsymbol{F_{error}} = \frac{\mathbf{25{,}140}}{\mathbf{1000000}} \times \mathbf{48000}$$

$$\boldsymbol{F_{error}} = [\mathbf{1.2, 6.66}]\,\text{Hz}$$

Therefore, the frequency error for the Raspberry Pi will range between $\mathbf{1.2}$ Hz and $\mathbf{6.66}$ Hz. The impact of this frequency error is later assessed in section 5.1.

4.8 Physical System Design

The process of physically implementing this project included many iterations and redesigns. The design process was structured in a modular manner where individual pieces of the system are completed before integrating that piece into the larger system. The design process begins with setting up the Raspberry Pi and Respeaker. Following this is implementing a method to achieve the basic goal of performing a recording with the hardware and outputting a '.wav' file[9]. The next goal was to implement a system that communicates wireless with all four Raspberry Pi's to begin recording, followed by a collection of

[7] $\frac{\mathbf{1.5}}{\mathbf{60\times60\times3}} = \mathbf{138}$ PPM
[8] $\frac{\mathbf{400}}{\mathbf{19.2\times10^6}} \times \frac{\mathbf{1}}{\mathbf{48000}} = \mathbf{1}\,\boldsymbol{Hz}$
[9] A wav file is an audio format that stores uncompressed audio data using a bitstream [60]

the recordings. Once this is complete, the Raspberry Pis are then placed in a grid structure, and the triangulation process can be implemented using the post-processing designed in section 4.5.

4.8.1 Raspberry Pi 0 w and Re-Speaker 2-mics Pi-Hat Setup setup

Raspberry Pi Setup

Once all the Raspberry Pis are received, the first step is to install the OS. The selected OS is the Debian Bullseye 32bit Raspberry Pi OS as recommended by Raspberry Pi [61]. Once the OS is installed, a static IP address is set for each device, allowing a set manner to communicate with each Raspberry Pi without the need to reference a new IP address when it changes. Each Raspberry Pi is then connected to a hot-spot provided by a laptop which shares the local network connection. Thereby, the devices do not require network details when the network changes. Once the Raspberry Pi is set up, it is logged into via SSH to begin setting up the Respeaker 2 Mics-Pihat[62][10].

Re-speaker 2 Mics PiHat Setup

The Respeaker 2-mics Pihat and Raspberry Pi interface through all 40 of the Raspberry Pi's Pin-outs by mounting the Respeaker on the Raspberry Pi as shown in Figure 4.13. Once connected, the various libraries are cloned from Git-Hub and further installed [47]. Finally, this step is validated through a successful recording, as described in the following subsection.

4.8.2 Recording Process, Communication and Commands

Once the Respeaker is set up, the next step is to obtain a recording from the device. A decision was made to utilise the terminal's record command provided by Linux's Advanced Linux Sound Architecture (ALSA)[63]. The ALSA command used to record is as follows:

```
arecord  -f S16_LE -r 48000 -d 20 -c 1 -D plughw:1 Recording_Name.wav
```

The parameters to the 'arecord' command are as follows:

- `-f S16_Le`

  File format = signed 16 bit Little Endian

- `-r 48000`

  Sample Rate = 48 kHz

- `-d 20`

  Record duration = 20 seconds

- `-c 1`

  Number of receivers = 1

- `-D plughw:1`

[10] For more details on SSH, see [62].

Device Name = plughw:1 (Device number of Respeaker 2-mics Pihat)

This command was performed on all devices and validated as it outputted the expected '.wav' file containing a 20 second recording with 48 KHz sample rate.

Following the successful recording implementation, the next step is synchronising the device's recordings. Before using a calibration signal for synchronisation, the initial approach was to attempt to synchronise all the devices' recording commands so they begin recording simultaneously. This was attempted through three approaches: hard-wired interrupts, synchronised clocks using NTP servers and remote interrupts. Once these methods were attempted, the final choice was an SSH command that triggers all the Raspberry Pis.

NTP Server Synchronisation

The initial approach taken in an attempt to synchronise the recordings was by syncing all of the Raspberry Pi's clocks to a NTP server. Once synchronised, a local Python script on each device will timestamp the start of the recording, which, if all clocks are in sync, one can obtain an accurate ToA of the signal based on the timestamps received. As explained in chapter 3 NTP, timing accuracy depends on the stratum level of the connection. When setting up the NTP server on a Raspberry Pi the stratum level was stratum 15, corresponding to no synchronisation or connection; hence, accurate timing was not achievable. This lack of connection could have been due to many factors, such as poor or lack of NTP servers in the Western Cape South Africa or complexities in the set-up process. Following this, further research was done into NTP servers. It was discovered that even a stratum two server, the best possible synchronisation from an NTP server only offered millisecond synchronisation. Millisecond synchronisation is not accurate enough for the system at hand as a 10-millisecond offset accounts for as much as $\pm \mathbf{3.43}\boldsymbol{m}$ error based on the distance sound travels in that time. Consequently this approach was not a viable option.

Hard Wired Interrupts

The following synchronisation attempt was implemented using hard-wired interrupts. An interrupt works by triggering a process when a signal is received by General Purpose Input Output (GPIO) pin. The first step in this process was to solider two output pins to the top of the device, one for ground and one for a GPIO input. Figure 4.13 shows the two soldiered pins on the Pi. The next step is to create a Python script on the Raspberry Pi, which waits for a high input on the specified GPIO pin; once the high signal is received, the system begins recording. However, this implementation did not work for the following reasons. Firstly the system received false positive signals and triggered recordings at random moments. An attempt was made to implement software debouncing[11] to fix this problem. The result was that the system did not trigger randomly, but the debouncing introduced timing uncertainties, which caused the systems not to be synchronised. Another concern regarding hard-wired interrupts is the lack of portability of the system. For that reason, it was not a viable solution.

[11] Bouncing is the tendency of any two metal contacts in an electronic device to generate multiple signals; debouncing is ensures that only a single signal will be acted upon [64]

Remote Interrupts

Remote interrupts are similar to hard-wired interrupts, but instead of using a physical wire to transmit the signal. A software called GPIO Zero which uses the *PIGPIO* library allows for access to the Raspberry Pi's GPIO pins over a network[65][66]. The system was set up to substitute the hard wired interrupt with the remote interrupt in the same Python script.

When implementing this approach, an issue was encountered when attempting to record; the output of the recordings was empty. It was discovered that *PIGPIO* reads the hardware's PCM clock by default, which is also read by the Respeaker. Therefore, *PIGPIO* needs to reference the device's Pulse-width modulation (PWM) clock to ensure both recording and remote triggering can occur simultaneously. Like the hard-wired GPIO attempt, the Python script was used, which waits for a trigger over the network. This approach did not work as the trigger delay over the network was unpredictable and triggered the recording process the devices at different times.

Following this, it was decided to use the calibration signal described in section 4.5.2 to synchronise the device's recordings in post-processing rather than attempting to synchronise the start of each recording. Although, as mentioned, this method is yet to be successfully implemented for acoustic localisation using TDoA, it was the most suitable approach given the constraints of the project.

Although remote interrupts did not work to synchronise the recording's start time, it was still used to trigger the recordings once the synchronisation method using a calibration signal was implemented. Remote GPIO was later abrogated following a timing assessment shown in subsection 5.2. This method was changed as it introduced unnecessary computational complexity to the recording process resulting in sampling issues in the recording process.

High Priority SSH Command

Based on results obtained from the timing assessment in subsection 5.2, as well as research done into optimising the systems sampling rate and minimising CPU time for unnecessary tasks. The approach to solving this optimising the sampling was minimising the systems tasks and increasing the recording process's priority. As explained, the initial approach was to SSH into the device to run a Python script which then waits for an interrupt to begin recording. However, this process is computationally expensive, and since the goal is to minimise computational complexity, this approach is adjusted.

The first change was to utilise an SSH command rather than SSH'ing into the Raspberry Pi. This allowed for a single command to be sent to the Raspberry Pi to execute instead of logging into the device to execute the script, which may utilise unnecessary peripherals when interacting with the device's shell. The second step taken was to remove the remote interrupts. Remote interrupts are unnecessary as it was a computationally expensive approach to triggering the recording process, and there are simpler and more efficient ways to do so. Another change was not to utilise a Python script to execute the recording command, as it added unnecessary complexity to the recording process.

These changes strip away unnecessary aspects of the process, which wasted CPU time. The last change made is to increase the priority of the recording process. UNIX systems utilise a 'nice' value which determines the CPU priority of a command. The higher the priority, the more CPU time given to

executing the command. Nice values range from **−20** to **19** where **−20** is highest priority and **19** the lowest.

It should be noted that some of the adjustments above may not have assisted in minimising the CPU time given to other processes but analysing the exact effects of these adjustments is beyond the project's scope. However, the overall goal of the changes is to minimise the recording process complexity and strip out unnecessary processes. This goal has been achieved through these adjustments.

The final recording command is as follows:

```
"ssh pi_name@IP_ADDR sudo nice -n -20 arecord  -f ...
S16_LE -r 48000 -d 20 -c 1 -D plughw:1 name.wav"
```

The above command executes the recording command mentioned earlier on the Raspberry Pi with the username 'pi_name' and IP address 'IP_ADDR'. The other addition to the command is the portion with 'sudo nice -n -20' this command uses root permissions on the Raspberry Pi to specify the 'niceness' of the command, which in this scenario is **−20**, maximum priority.

Recording Process

The recording process is implemented through a Python script on a laptop. The script begins by defining the usernames and IP addresses of the four Raspberry Pis. It then defines four parallel commands using the Pythons 'multiprocessing' library. The four commands are the aforementioned high priority ssh command which references the different Raspberry Pis. Once the commands are executed, the laptop waits for five seconds to ensure the recordings have begun. The laptop then plays the 0 Hz to 15 kHz five second chirp as the calibration signal as discussed in section 4.2. Once this signal has played, the SoI is played from a separate speaker. The entire recording process is **20** seconds. Once the recordings are complete, the Python script then executes a Secure File Copy (SCP) command, which collects the recordings from each Raspberry Pi [67]. The Python script can be found in appendices section 7.7.

The SCP command is as follows:

```
scp pi_name@IP_ADDR:recording.wav local_DIR_name
```

This command collects the file named 'recording.wav' from the Raspberry Pi with the name 'pi_name' and IP address 'IP_ADDR' and saves it to the directory 'local_DIR_name' on the laptop.

This process was implemented and successfully triggered the start of the recordings on each device followed by an automatic download of all recordings. This recording process, therefore, forms a network of devices that can record and further transmit recording wirelessly as set out in the project's objectives.

4.9 Summary of System Design

The system will utilise four microphones placed on the corners of a rectangular grid. Once the devices begin recording, the calibration signal is played from a known position. Following the calibration signal, the SoI occurs from an unknown position. Once all recordings are received, the ToA of the calibration signal and the SoI are determined using cross-correlation. The ToA of the calibration signal is then

used to synchronise all recordings. Once synchronised, the TDoAs of the signal are then determined and utilised for triangulation. The triangulation algorithm is implemented by using a LUT, which retrieves the coordinates of the row that most closely resembles the TDoAs. The flowchart below depicts this process.

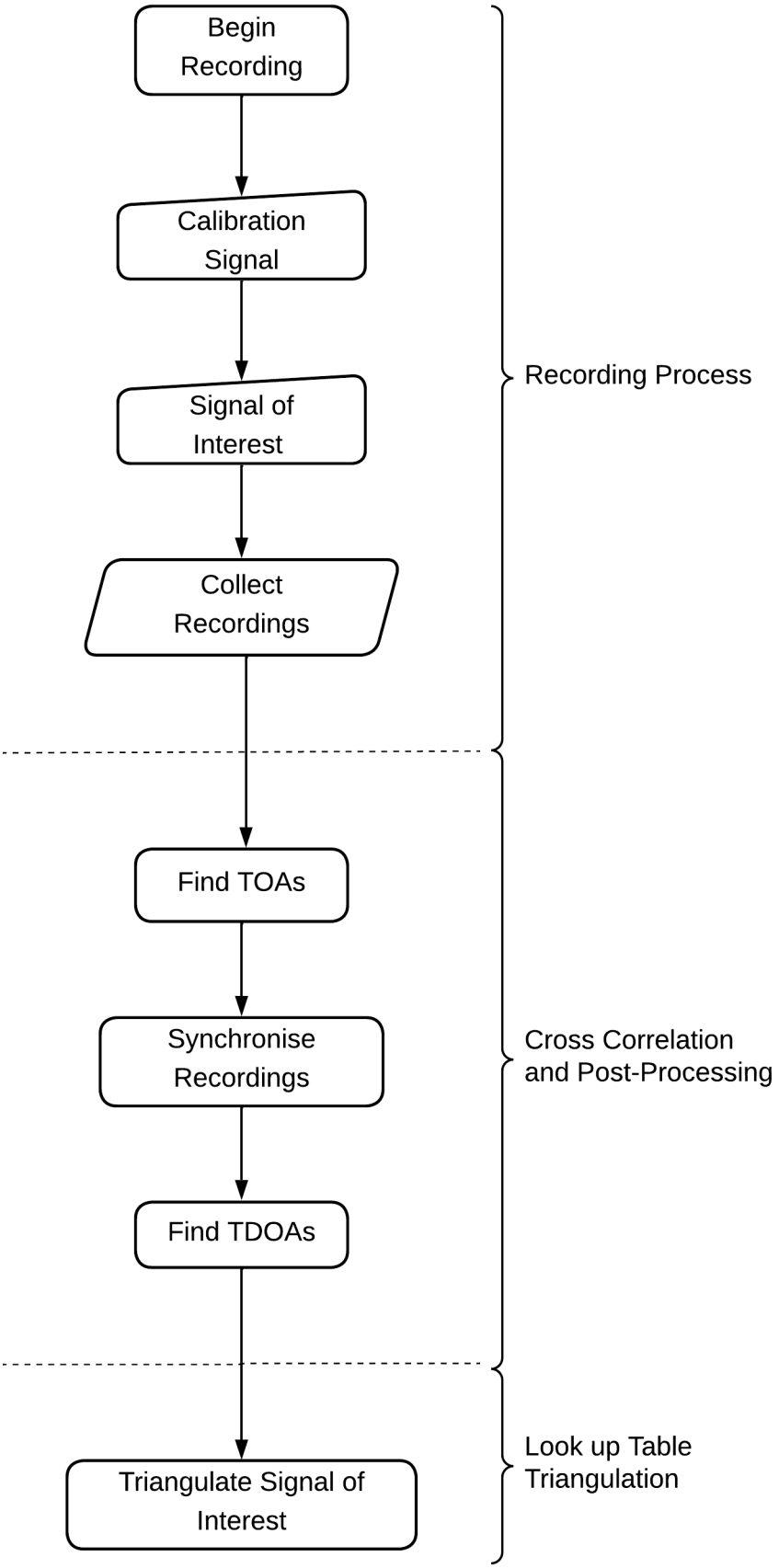


Figure 4.14: A Flowchart showing the steps taken to triangulate a signal source

Chapter 5

Experiments and Results

The experiments and testing process is split up into two parts. The first section consists of the simulation experiments and results, and the second covers the practical implementation assessments, field tests and results. The simulation experiments aim to understand the system's behaviour under conditions that cannot be controlled and provide insight into the expected performance of the physical system. The physical system experiments aim to assess if the hardware used is capable of acoustic triangulation and to evaluate the system's triangulation performance.

5.1 Simulation Experiments

The following experiments assess the system's behaviour in the presence of clock non-idealities. These experiments' aim is twofold: the first is to understand how the system would behave under conditions that cannot be controlled when assessing the physical system. The second is to provide insight into the physical system's expected performance based on a simulation which emulates the non-idealities of the hardware used in the project.

The following two assessments analyse the system performance under varying jitter and drift. Ten different jitter and drift values are used, ranging 0 Rad-2π Rad and 0 Hz-10 Hz, respectively. Four normally distributed random values centred on the error value with a standard deviation of 0.1 are generated for each jitter and drift error. i.e., when assessing a drift of 5 Hz, a random value centred on five will be used as the clock's drift parameter. A random value is used because if all clocks have the same error, it is as if they all reference the same clock. This will result in all clocks being in sync, which is contrary to the experiment's aim. The simulation is run one-hundred times, ten times for each error value. The outcomes are then averaged and used for the final result.

5.1.1 Phase Noise (Jitter)

The following image depicts the accuracy of the estimated position versus the phase error of the clock. The Figure's plot is drawn from the average of 100 simulations.

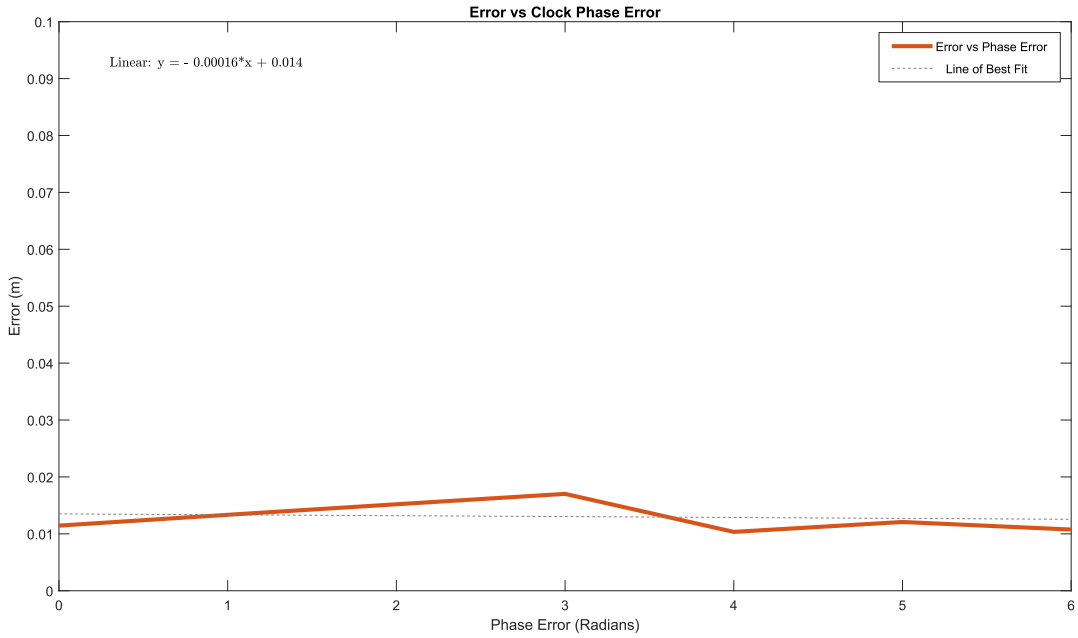


Figure 5.1: Figure showing the Triangulation Error due to Clock Jitter

5.1.2 Frequency Noise (Drift)

The following figure depicts the system's accuracy versus the clock's frequency error.

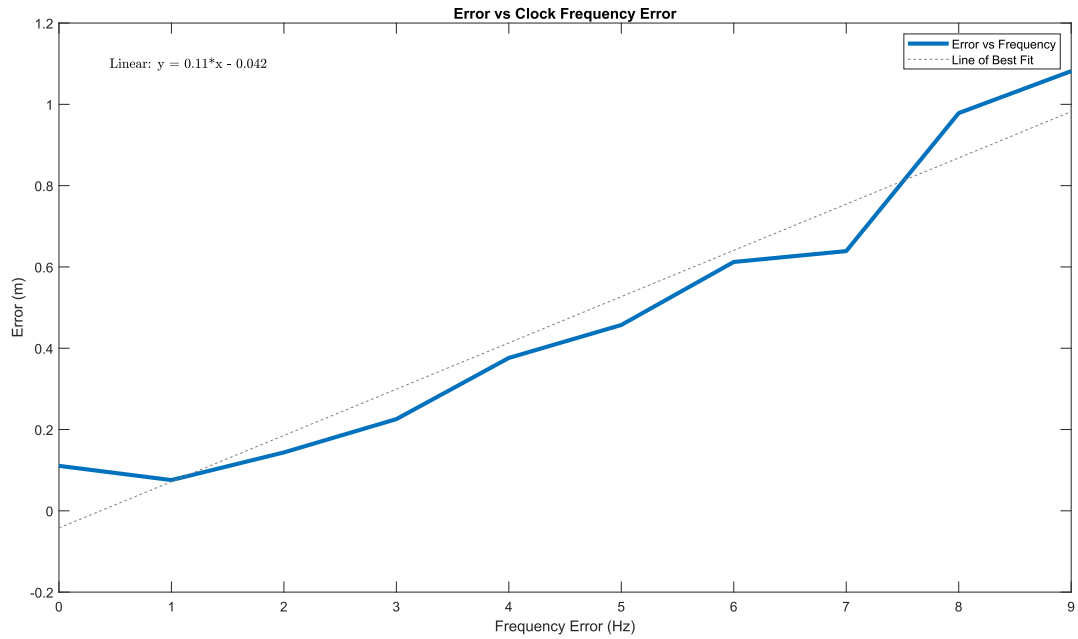


Figure 5.2: Figure showing the Triangulation Error due to Frequency Drift

5.1.3 Results Analysis

Figure 5.1 above shows that clock jitter has minimal impact on the system's accuracy. The error ranged from 0.012 m at 0 Rad to 0.019 m at π Rad with a weak gradient of 0.00016 $\frac{m}{Rad}$ on the line of best fit. The simulation shows that the maximum effect of jitter on the system's accuracy is an error of 0.007 m. One can see that the system's performance worsens until the jitter reaches πRad and then gradually betters as the clock's jitter approaches 2π. The effect of jitter error is minimal and can be considered insignificant when assessing the accuracy of the system when compared to other possible sources of error.

In contrast, Figure 5.2 illustrates that a clock frequency error significantly impacts the system's accuracy. The system's accuracy ranges from 0.1 m to 1.1 m with a frequency drift of 0 Hz to 10 Hz, respectively. The line of best fit has a steep gradient of 0.11 $\frac{m}{Hz}$. This simulation shows that clock frequency errors majorly impact the system's accuracy.

5.1.4 Simulation Result's Discussion

The results obtained from the simulations displayed valuable insight into the system's behaviour. The first interesting result is that a clock jitter plays little role in the systems accuracy. The second result obtained shows that minor frequency errors on the clock will result in major errors in the triangulation accuracy. This point emphasises the importance of an accurate clock when implementing acoustic triangulation. A possible reason for a frequency drift having a more significant impact than jitter is as follows.

Frequency drift is not limited to a specific range. It can result in a significant sampling rate deviation from the ideal frequency. Take, for example, a system which samples at the double the specified rate. Any signal will be perceived to of arrived at twice the expected ToA as double the number of samples had occurred before the signal was received. In contrast, jitter is limited to a sampling error between 0 Rad and 2π Rad. As shown in Figure 5.1 the maximum error occurs at a jitter of π Rad from the ideal sample time. At 48 kHz, a deviation of π Rad accounts for a timing error of 10.416 μs.[1] This time deviation is insignificant considering that sound can only travel 0.0035 m.

As explained in subsection 4.7.4, the Raspberry Pi is expected to have a clock drift between 1.2 Hz and 6.66 Hz. Relating these values to Figure 5.2, the hardware is expected to obtain a triangulation error between 0.1 m and 0.65 m. Therefore, these values provide the expected triangulation capabilities of the hardware used in this system.

5.2 System Timing Assessment

As emphasised throughout the report and shown in section 5.1, the systems' triangulation performance is subject to the timing accuracy of the devices used. If a sampling error occurs, this will directly affect the accuracy of the ToA estimation, thereby resulting in inaccurate triangulation. Accordingly, the

[1] $1 \div (2 \times 48000) = 10.416 \times 10^{-6} s$

following timing assessment was set up to determine if the system built can obtain the fine timing required to perform acoustic triangulation.

The timing assessment aims to determine if there are any sampling or timing errors in the recording process used. The assessment is set up as follows:

- The four devices are placed next to one another in a line.
- The signal source is placed in front of the array of microphones.
- A 0 Hz to 15 kHz, 5 second chirp is used as the signal.
- The signal is repeated precisely every 10 seconds for a total of 180 seconds.

The assessment determines the ToA of each signal. The expected result is that the signal is received every 10 seconds. Given that the signal occurs every 10 seconds, the plot of the signal's ToA on the Y axis versus time on the X axis should result in a perfectly straight line with an equation $Y = 10X + b$. The value b is not relevant in this context as it refers to the ToA of the first chirp. However, the gradient of the straight line corresponds to the time between each chirp. Therefore, any offset from the 10 second period between ToAs will correspond to a deviation in the gradient of this slope, thereby depicting a sampling error in the recording process. Suppose there is a formidable deviation from this 10 second period. In that case, the system's recording process needs to be more precise and must be changed to perform acoustic triangulation.

5.2.1 Remote GPIO Timing Assessment

As explained in subsection 4.8.2, the initial approach to implementing the recording process utilised a Python script on each Raspberry Pi. When the script is executed, it waits for a remote GPIO interrupt to trigger the recordings. The following experiment assesses this implementation's timing accuracy.

Figure 7.1 below depicts the results obtained from microphone 4's recording. The spectrogram of the recordings received, sound played and other microphone's results can be found in appendices section 7.3. The figure shows three different plots. The top plot is the above mentioned ToA versus time plot. The plot in the middle is the first derivative of this plot which will show the gradient of the above plot which in this instance should be 10 seconds per chirp. The final plot is the second derivative of the ToA versus time plot. This plot will display any ToA deviation from the ideal 10 seconds per chirp.

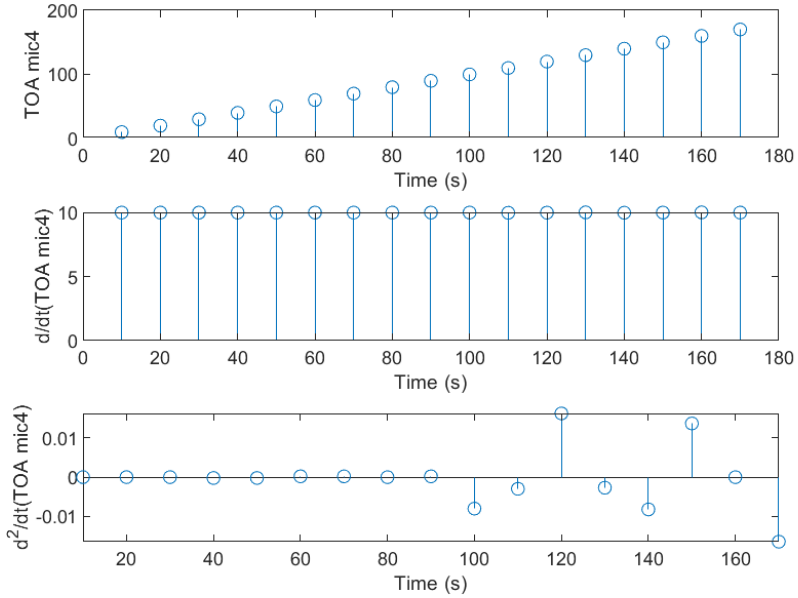


Figure 5.3: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 4's Recording

In Figure 5.3 above, when analysing the top two plots graphically, one can see that the results are as expected. The first plot shows a straight line graph, and the second plot shows the first derivative of the top plot is ± 10 (fracsecondchirp. However, the figure's bottom plot shows that the deviation in ToA is ± 0 seconds until the 80^{th} second. From that point on, one can see major deviations in the ToA. At worst, the ToA deviation is ± 0.01 seconds.

An error of this magnitude will result in triangulation accuracy of $\pm 3.43m$[2]. This result shows that the current system is not capable of acoustic triangulation as the sampling errors will cause significant inaccuracies when determining the ToA of the signal. Therefore, the recording process needed to be redesigned in an attempt to improve the system's sampling performance.

5.2.2 High-Priority SSH command Timing Assessment

As explained in subsection 4.8.2, following the results of the Remote GPIO timing assessment, the recording process was redesigned. The new recording method stripped unnecessary processes and implemented a high-priority Secure Shell (SSH) command. Following the implementation of the new design, a timing assessment was performed but with minor adjustments. The adjustments are as follows. First, the timing assessment was done over 300 seconds, allowing for the evaluation of sampling inaccuracies over a longer time-frame. The second change altered the repetition time from 10 seconds to 5 seconds. Accordingly, the expected straight line should be in the form $Y = 5X + C$. This change increases the resolution of the timing assessment, allowing the analysis to identify sampling errors over smaller time-frames.

It should be noted that although changes were made between the new system and the old system's

[2]A microphone which receives a signal 0.01 s late will result in the TDoA value offset by 0.01 s, which, based on the distance a sound travels in that time, corresponds to a $0.01 \times 343 = 3.43$ m error.

timing assessment, comparing the results is still equitable. This is because the changes made only adjust the assessment's resolution and length and do not compromise the assessment's integrity.

Figure 5.4 below depicts microphone 4's timing assessment results following after the system changes are made.

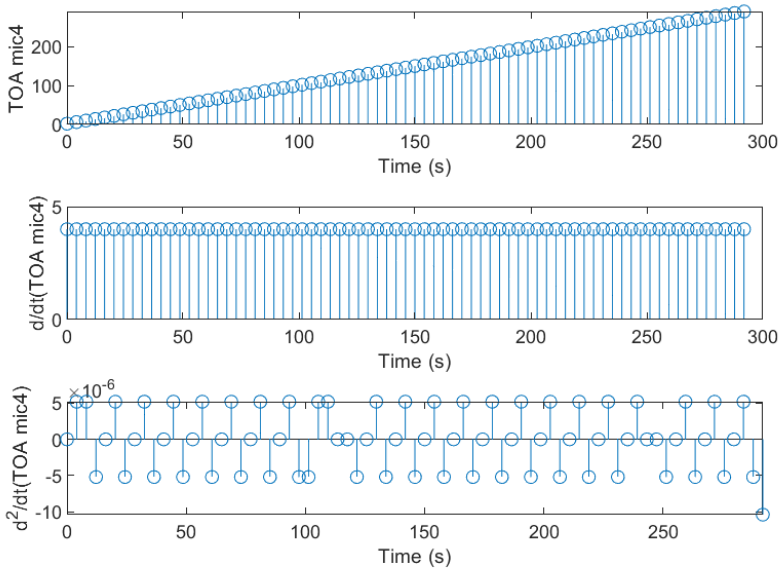


Figure 5.4: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 4's Recording after system changes

The first and second plots in Figure 5.4 above show that the ToA of the signals occurs every $\pm$ 5 seconds as expected. The bottom plot shows the new system obtained a deviation in ToA in the order of 1×10^{-6} seconds. This result significantly improved compared to the previous system's deviation, which was in the order of 1×10^{-2} seconds. Furthermore, this result shows that the design changes successfully minimised sampling errors.

5.2.3 Timing Assessment Discussion

The timing assessment was set up to assess the sampling accuracy of the recording process implemented. When the initial recording process using remote GPIO was evaluated, the sampling error was in the order of $10^{-}2$ seconds. Following a system redesign and the implementation of the high-priority SSH command, the timing assessment showed sampling inaccuracies in the order of $1 \times 10^{-}6$ seconds. Attempting to relate these results directly to the drift and jitter assessment performed in section 5.1 is complex as it requires a deeper analysis of clock errors which is beyond the project's scope. However, these results show that the system's redesign successfully optimises the recording process' sampling accuracy and produces a system with an accurate enough sampling to perform acoustic triangulation. Therefore, the new recording process is used for acoustic triangulation and the following field tests.

5.3 Field Test

The field test aims to assess the triangulation accuracy of the physical system. The tests were conducted inside a $8 \times 5 \times 4$m carpeted room with pillows around the microphones to minimise noise and multi-path propagation. The tests are conducted by assessing the triangulation performance at nine different test points, repeated ten times for each test point. The repeated test at each point will allow a more accurate depiction of the system's average performance. In addition, this allows for evaluating the triangulation's accuracy and precision. The field test is set up as follows.

- The four devices are placed on the corners of a 3.68 m×6.31 m grid.
- The signal for both the calibration signal and the SoI is a 0 Hz-15 kHz 5 second chirp.
- The calibration signal is played from the centre of the grid.
- The calibration signal is played from a 30 cm wide laptop with stereo speakers.
- The SoI is played from a smartphone with sterio speakers.
- The devices are set to sample at 48 kHz.
- The recordings are all 20 seconds long.
- The LUT uses centimetre resolution.
- All receivers and transmitters were elevated 1.5 m above the ground.[3].

5.3.1 Experiment Limitations

Throughout the experimentation process various non idealities may lead to error when triangulating the events. The first possible limitation is due to human error when measuring the grid. The grid was carefully measured by using a tape measure, but it is difficult to get exact and accurate measurements. Following multiple attempts to measure the grid, an acceptable error range of $\pm 5cm$ was decided upon.

Ideally, the transmitter should be a point source when triangulating a signal. Accordingly, a FERS simulation was set up to assess if the system's accuracy is affected by a non-point source. The simulation assessed this by using two-point sources centred on the transmitter positions to emulate the effects of a stereo speaker. Both point sources output the same signal at the same time. The simulation results can be found in Table 7.3 and Table 7.4 in Appendices section7.5. The simulation outcomes show that the stereo speakers do not affect the triangulation accuracy, given that the following conditions are met. Firstly, the stereo speakers are parallel to the X or Y axis of the grid. Secondly, the stereo speakers' centre is in the transmitter's desired position. Therefore, using stereo speakers should not affect the triangulation accuracy.

The experiments are performed in a room where there are possibly acoustical non-idealities that may lead to multi-path propagation. However, evaluating the effects of multi-path is beyond the project's scope. Along with these factors, hardware limitations discussed in subsection 4.7.4 will further impact the system's performance.

[3]This is to ensure the signal transmitted has a direct line of sight to the receivers

Throughout the experiments the speed of sound is assumed to be $343\,m.s^{-1}$, this is the speed of sound in dry air at $20\,degC$. The speed of sound in reality may vary slightly from this value depending on the environment. However, the environmental factors could not be measured therefore, the propagation velocity of the signal could not be precisely defined.

5.3.2 Field Test Results

Figure 5.5 below depicts a point map of the nine test points and the ten estimated positions per test point. The '⋆' represents the test point, the '•' illustrates the estimated position, and the circle displays the mean error radius.

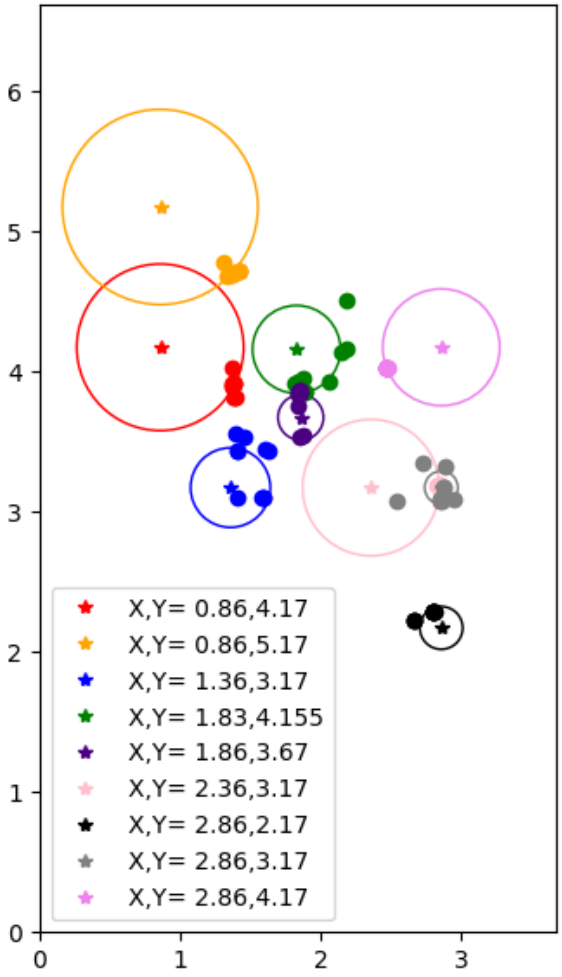


Figure 5.5: Point map of triangulation position estimates and average estimate from field test result

The table below depicts the average performance of the ten estimated positions for the nine test points. The mean error is defined as the average error between the test point and the estimated positions. In contrast, the precision error is defined as the average offset between the estimated position and the

calculated mean position. The calculations for the accuracy and precision use equations 7.1 and 7.3 shown in appendices section 7.4.

Test Point (X,Y)	Mean Estimated Position (X,Y)	Mean Error(m)	Precision Error(m)
(0.86 , 4.17)[⋆]	(1.33 , 3.92)	0.54	0.027
(0.86 , 5.17)[⋆]	(1.33 , 4.75)	**0.63**	0.024
(1.36 , 3.17)[⋆]	(1.5 , 3.31)	0.19	0.134
(1.83 , 4.16)[⋆]	(1.96 , 4.02)	0.19	**0.147**
(1.86 , 3.67)[⋆]	(1.85 , 3.77)	0.1	0.054
(2.36 , 3.17)[⋆]	(2.8 , 3.18)	0.44	0.004
(2.86 , 2.17)[⋆]	(2.77 , 2.26)	0.13	0.042
(2.86 , 3.17)[⋆]	(2.83 , 3.15)	**0.04**	0.078
(2.86 , 4.17)[⋆]	(2.51 , 4.04)	0.38	**0.002**

Table 5.1: Table showing triangulation mean accuracy and mean precision over 90 tests

5.3.3 Results Analysis

As shown in Figure 5.5 the physical system is capable of acoustic triangulation as the estimated positions are in close proximity to the transmitter being triangulated. The maximum triangulation error obtained is **0.63 m**, which relates to the test point **(X=0.86,Y=5.17)**, coloured in orange. Whereas, the most accurate test point at coordinates **(2.86,2.17)** has a mean error of **0.04 m** depicted in Figure 5.5 in grey. Figure 5.5 also shows that the estimated positions are tightly grouped for each point. As shown in Table 5.1 the precision ranges from **0.002 m** shown in Figure 5.5 in pink to a maximum precision error of **0.147 m** shown in green. The overall average accuracy for all 90 tests is **0.293 m** with a mean precision of **0.057 m**.

5.4 Triangulation Results Discussion

The results obtained in section 5.3 show the system is capable of triangulating the signal with an accuracy ranging from **0.04 m** to **0.64 m** and with a high precision between **0.002 m** and **0.147 m**. The system obtain a mean error of **0.29 m** and a mean precision of **0.057 m**. The high precision and accuracy shows that the system the system triangulates a source with repeatable and accurate estimations.

5.4.1 Sources of Error

The experiment shows that the system's accuracy deviates depending on the test point triangulated. Initially, it was thought that the accuracy differences between test points is a result of the randomness of clock errors in the system. However, following further analysis, it was determined that this assumption is only partially correct. If this explanation were true then the deviation in accuracy when changing the test point should be comparable to the deviation in accuracy over multiple tests at the same test point.

However the experiment results contradict this expected outcome. The experiment shows that there is a clear distinction between the system's precision and accuracy. Furthermore, certain test points obtained a much higher precision than others, ergo, there must be an attribute specific to each test point that impacts the triangulation performance such as acoustics or multi-path in specific regions in the experiment room. To reinforce this statement, all signal sources near a wall in the room obtained a poorer accuracy than the test-points in the centre. The walls in the room where in close proximity to the Orange, Red and Pink test points shown in Figure 5.1. Therefore, the system errors is expected to be due to a combination of clock errors and environmental acoustic effects.

5.4.2 Performance Discussion

The triangulation results in isolation do not provide insight into the holistic performance of the system designed. Therefore, the results are compared and paralleled to the results obtained from the simulations in section 5.1 and the performance of similar systems discussed in chapter 2. Following the comparison of the results is a brief discussion of the system's performance, considering the novel approach to synchronisation in this context.

Comparison to Simulation Results

When compared to the simulation, the physical system's results closely resembled the expected performance based on the results obtained in section 5.1. As explained, the Raspberry Pi is expected to have a frequency error ranging from 1.2 Hz to 6.66 Hz. Figure 5.2 shows that a system with this frequency error range should obtain a triangulation accuracy between 0.1 m to 0.65 m.

These results are similar to the practical implementation's results which range from 0.04 m to 0.64 m. Furthermore, the average simulation performance in the expected frequency error range is 0.35 m, whereas the mean error for the physical system is 0.293 m. This minor difference in performance shows that the system implemented successfully gets results which closely mimic the simulated theoretical performance of a system with the same clock errors. Á corollary of this result is that a simulation that emulates the physical system was successfully implemented as set out in the project's objectives.

Comparison to Similar Implementations

Peng Wu et al. [17] tested their system in a 10 m×10 m grid and obtained a mean accuracy of 0.67 m, which is 0.38 m less accurate than this system's 0.29 m mean accuracy. A.R Mohanty and Chinmayi Mahapatra, on the other hand, obtained a mean accuracy of 0.01 m, which is 0.28 m more accurate than the mean accuracy obtained in this project [18].

However, their system utilised a high-accuracy clock connected to all the recording devices. In contrast, this system utilised asynchronous clocks, which were subject to various clock non-idealities. Overall, the system designed in this project obtained a triangulation accuracy comparable with similar implementations of acoustic triangulation.

System Novelty

Although novelty was not an objective of the project, the system built implements a novel approach to using asynchronous devices in a TDoA acoustic triangulation system.

This system obtained notable results using a calibration signal to synchronise the recordings received. This method is rarely used in acoustic triangulation, and similar projects that have implemented synchronisation using a calibration signal, such as Samualsons' system, obtained inaccurate triangulation results [15]. When formulating the literature review, aside from Samualsons' implementation, no other literature discussing acoustic triangulation systems that synchronise devices using a calibration signal was found.

However, this project implemented a system which utilised synchronisation using a calibration signal and obtained results comparable to systems which utilise synchronised, high-accuracy clocks. Although this was not an initial objective of this project, the system implemented proves acoustic localisation can be achieved using devices with asynchronous clocks. This paper, therefore, successfully implements a novel approach to synchronising devices in an TDoA acoustic triangulation system.

Chapter 6

Conclusion & Future Recommendations

6.1 Report Summary

The objective of this project was to implement a network of distributed sensors capable of acoustic triangulation using TDoA. Acoustic triangulation is an interesting problem that stretches multiple engineering disciplines—solving this problem combines aspects of hardware, software and signal processing.

The report began in chapter 2 with a review of various literature relevant to acoustic triangulation. The review began with a high-level explanation of TDoA triangulation. It then continued with a brief discussion of its historical and current applications. The chapter then followed with a discussion of similar projects, their design choices and results. The review found that the TDoA algorithm used to triangulate the source requires less consideration than the process initially obtaining the correct TDoA values. Furthermore, the defining features of a successful TDoA triangulation system are the accuracy of the ToA estimations and the correct synchronisation between devices.

Chapter 3 then elaborated on the theory required to implement the acoustic triangulation. The chapter began by providing the mathematical basis of TDoA triangulation. It then continued with a background to the various synchronisation methodologies used to implement triangulation, with an additional focus on using a calibration signal to time-align all recordings. Following this, cross-correlation and GCC-PHAT were explored and compared as methods for ToA estimation. Next, the frequency chirp was introduced, and its desirable SNR gain properties were discussed. The chapter then concluded with a brief background into the ECM and MEMS microphones and their different communication standards, which were later considered when selecting the hardware for the system.

The report then continued with the body of the project in chapter 4 where the system was designed and implemented. The chapter began by introducing the design pipeline, which sets out the different modules in the system and the order in which they were designed. The pipeline started with the design of the general system structure, consisting of the placement of receivers in a rectangle formation and the choice of the signal used, which was the aforementioned frequency chirp. The chapter then continued with the design of a FERS simulation, followed by a validation assessment which confirmed the simulation was set up correctly. The simulation was then used to assist in the design of the various post-processing stages of the system.

The post-processing consisted of the ToA estimation, the synchronisation methodology and the triangulation algorithm. First, cross-correlation was selected as a means to perform ToA estimation. Following this, the synchronisation process using a calibration signal was designed and implemented, noting that other synchronisation methods were attempted and did not produce adequate results. Finally, the TDoA values were determined using the synchronised ToA values and used for triangulation. The triangulation algorithm designed makes use of a LUT. The LUT works by returning the X and Y coordinates in a row with TDoA values that most closely resemble the values obtained. Each of the three post-processing stages was validated and produced the expected results.

Section 4.7 discussed the selection and implementation of the hardware used in the system. The section began by outlining the various hardware requirements for the system, followed by a comparison of different microphones and boards. The Respeaker 2 mics-Pihat was selected as the microphone and the Raspberry Pi 0W as the board. This combination was chosen as it met the various hardware requirements. In addition, they were available locally, easily configurable and well-supported. The report then discussed the various limitations of the devices. It was discovered that the system is expected to have a clock drift between 1.2 Hz and 6.66 Hz. Next, 32bit Raspberry Pi OS was installed, and each device was assigned a static IP address. The Re-Speaker was then configured and mounted on the Raspberry Pi. Following this, the recording process was designed where initially, remote GPIO triggered the recordings but was later changed following a failed timing assessment. The changes successfully minimised unnecessary processes and increased the CPU time given to the recording process. This, in turn, optimised the recording process's sampling accuracy. The final recording process uses a high-priority SSH command and a SCP command to execute and retrieve the recordings. This design was then validated, concluding that a network of wireless receivers was successfully implemented.

The system implementation then concludes in chapter 5 with an evaluation and discussion of the simulated and physical systems designed. The first set of experiments assessed the effect of clock non-idealities on the simulated system. These results showed that clock jitter has a minimal impact on the system's accuracy as it is limited to a maximum sampling error of π Rad. In contrast, drift has an adverse impact on the system's triangulation performance as the sampling error due to drift can be more significant. Another outcome of this experiment is that the simulations with a frequency drift in the same range as the Raspberry Pi's expected drift results in a triangulation error between 0.1 m and 0.65 m.

The timing assessment followed the simulation experiments, which evaluated the hardware's sampling accuracy. The first timing assessment was performed on the system, which utilised remote GPIO interrupts to trigger the recording process. The timing assessment resulted in sampling errors in the order of 1×10^{-2} s and therefore, the design was not fit for acoustic triangulation. The system was then redesigned to use a high-priority SSH command to execute the recordings. The redesigned system obtained sampling errors in the order of 1×10^{-6} s; therefore, successfully optimising the system's sampling accuracy.

Finally, section 5.3 assessed the triangulation performance of the system. The experiment was conducted by evaluating the triangulation accuracy over ninety tests. The physical system was successfully capable of triangulating the signal source and obtained accuracy ranging from 0.04 m to 0.63 m and a mean accuracy of 0.293 m. This system also had a precision ranging between 0.002 m and 0.05 m, showing

that the system produces accurate and consistent results. Furthermore, the accuracy and precision results were interrogated, and it was concluded that the system's errors were likely due to a combination of environmental factors and clock non-idealities.

The performance was then assessed by comparing the results to simulations and other implementations. The comparison found that the system's accuracy deviated from the expected simulated accuracy by a maximum of 0.06 m, which shows that the system obtained results that closely resembled the expected accuracy. The system also obtained a triangulation accuracy within the range of similar applications, noting that the system utilised a calibration signal to synchronise the recordings. In contrast, the compared projects used synchronised, high-accuracy clocks to sample the signal.

6.2 Conclusion

Overall this project achieved all objectives initially set out. A wireless network of distributed acoustic sensors was designed and implemented in simulation and using the selected hardware. The physical system obtained accurate triangulation results that concur with the expected results obtained when simulating the hardware used. In conclusion, this project was a success, and the systems designed obtained remarkable triangulation results.

6.3 Future Recommendations

Like any other engineering solution, the system's grandeur is limited by cost and time. There is a multitude of adjustments and additions that can be made to the system to increase its' expansiveness.

6.3.1 Finer Timing and Synchronisation

As explained in section 5.4 the systems accuracy was hindered due to clock non-idealities. A recommendation for any system which expands on this project is to improve the timing accuracy of the hardware used. A suggested approach will be to use a GPS on each device. This will result in accurate timing and fine synchronisation between receivers in the network. This will likely improve the triangulation accuracy of the system.

Another benefit of using a GPS to synchronise the devices is that if one adds a transmitter to each receiver, then using ToA localisation mentioned at the beginning of chapter 3 one can auto configure the devices and remove any need to measure the position of the receivers in the grid. Pairing this with a battery to each device will make the system portable and automatically configurable.

However, if the future work on the system does not implement synchronisation using a GPS then a recommendation is to adjust the current position of the calibration signal from the centre to the corner of a grid. Currently the system calibrates from the centre and it is a strenuous process to determine the centre of the grid. A better approach will be to place the calibration signal transmitter above one of the receivers, this will drastically simplify the setup time.

6.3.2 Triangulating in Three Dimensions

A clear addition that can be made to the current system is to add an additional dimension to the triangulation. This project utilised four receivers, which in theory is capable of triangulating a signal in 3-dimensions. However, the fourth receiver was used as a fail-safe in the case a receiver malfunctioned. The system can therefore utilise the fourth receiver to triangulate in three dimensions or add a fifth receiver if the redundancy is beneficial.

6.3.3 Real-Time Triangulation

The system in its' current form is a prototype and cannot be applied to any use case as the triangulation process begins manually. A clear addition is to implement a system which is capable of live triangulation of the signal source. Therefore if the system is to be applied to any use case, real-time triangulation is necessary.

6.3.4 Triangulating an Unknown Signal

The ToA estimation used in this project compared a recording received with a reference signal. This approach limited the triangulation process to a known signal. Unless the system is applied to a use case which aims to triangulate a signal with well defined properties such as a gunshot, then the system should be expanded to triangulate unknown signals.

6.3.5 Signal Classification

Based on the above-mentioned recommendation, another addition that can be made if unknown signals are triangulated is the implementation of a signal classification model. The addition of a classification model to the system will expand the possible use cases for the system.

6.3.6 Sensor Fusion

As mentioned in chapter 2, when acoustic triangulation is applied in the commercial space, the system does not work in isolation. Instead a multitude of sensors all work in combination to produce a more robust system. Triangulation using TDoA can be combined with other localisation techniques. For example each receiver can can be converted to an array of receivers. This will allow for both TDoA triangulation, direction of arrival triangulation and beam-forming

Chapter 7

Appendix

7.1 Calibration Position Simulation

Receiver	t_i**(ms)**	**TOA of SoI(ms)**	**Error**(ms)
1	20.6153	0	0
2	20.6153	0	0
3	20.6153	0	0
4	20.6153	0	0

Table 7.1: Synchronisation results when calibrated from centre

Receiver	t_i**(ms)**	**TOA of SoI(ms)**	**Error**(ms)
1	0.0000	0	0
2	29.1545	0	0
3	29.1545	0	0
4	29.1545	0	0

Table 7.2: Synchronisation results when calibrated from corner of grid, $0, 0$

Both results above show that the system successfully calibrates from both the centre and the corner of the grid.

7.2 Look Up Table Code

The following snippet of code is the function used to generate the LUT.

```
generate_LUT(resolution,Max_X,Max_Y):
    c = 343 #Speed of Sound
    grid_size = Max_x*Max_Y
    LUT = [resolution*grid_size] #Empty array for lookup table

    for X in range(Max_X):
```

```
for Y in range(Max_Y):

    # Find Euclidean distance from each Mic to X,Y
    TOA_1 = dist([Mic1_X,Mic1_Y],[X,Y])/c
    TOA_2 = dist([Mic2_X,Mic2_Y],[X,Y])/c
    TOA_3 = dist([Mic3_X,Mic3_Y],[X,Y])/c
    TOA_4 = dist([Mic4_X,Mic4_Y],[X,Y])/c

    #Find all combinations for TDOA's
    TDOA_41 = TOA_4 - TOA_1
    TDOA_31 = TOA_3 - TOA_1
    TDOA_21 = TOA_2 - TOA_1
    TDOA_42 = TOA_4 - TOA_2
    TDOA_32 = TOA_3 - TOA_2
    TDOA_43 = TOA_4 - TOA_3

    row = [TOA_41 , TOA_31 , TOA_21 , TOA_42 , TOA_32 , TOA_43 , X , Y]
    LUT.append(row)
```

7.3 Timing Assessment additional Results

7.3.1 Additional results of Timing Assessment Before System Changes

The figure below depicts the recordings specrograms which depict the chirp signals received following the timing assessment when using remote GPIO, before system changes.

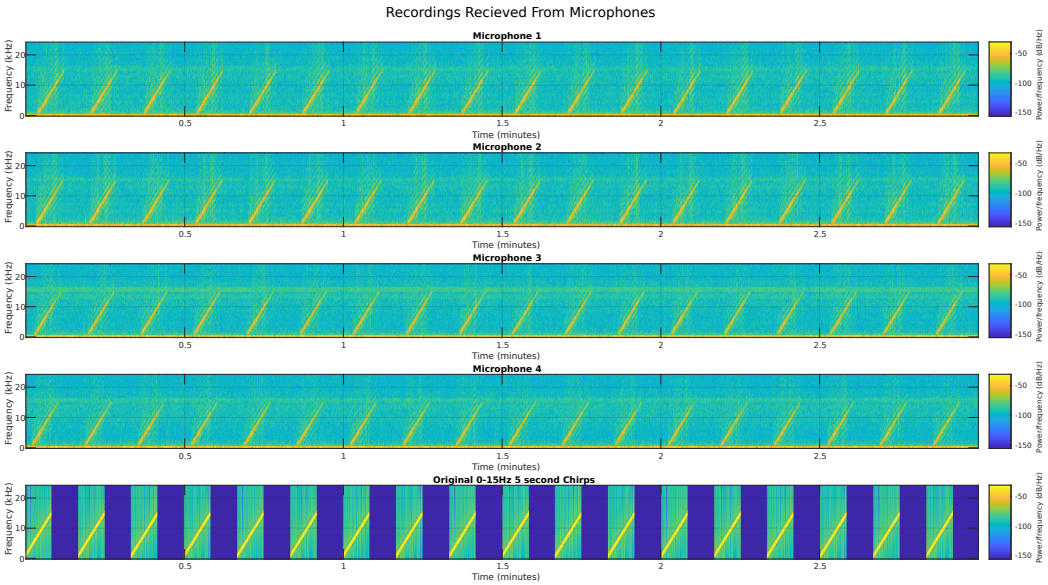


Figure 7.1: Spectrogram of Recordings Received by 4 receivers and Sound Played

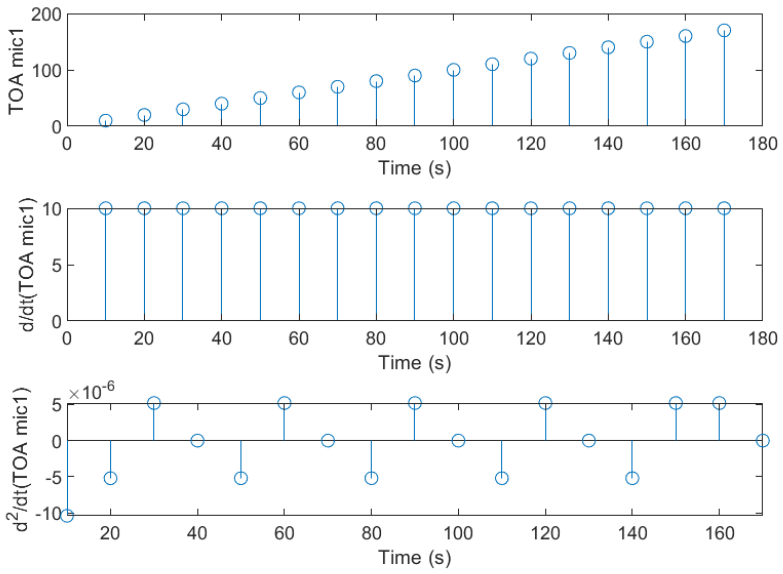


Figure 7.2: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 1's Recording

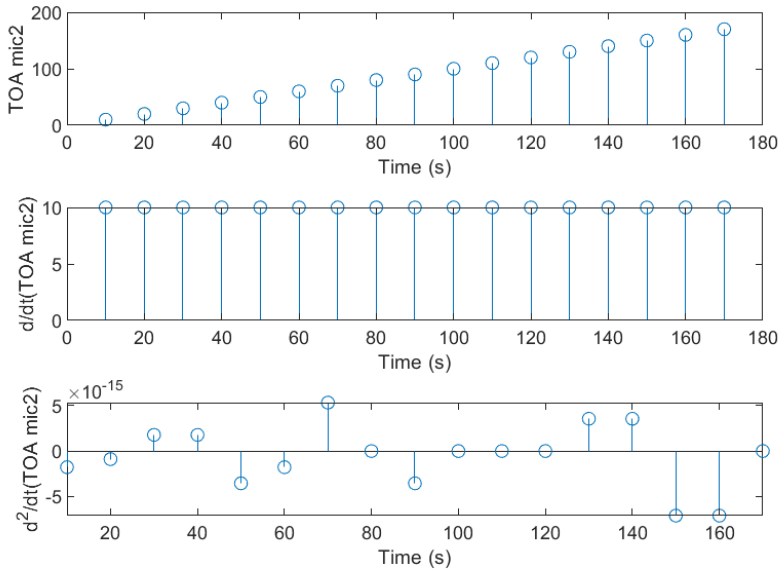


Figure 7.3: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 1's Recording

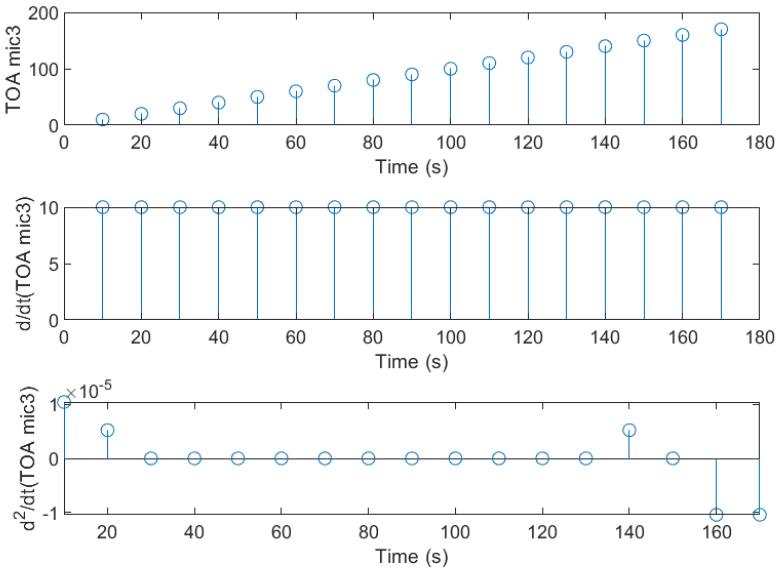


Figure 7.4: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 1's Recording

7.3.2 Additional results of Timing Assessment After System Changes

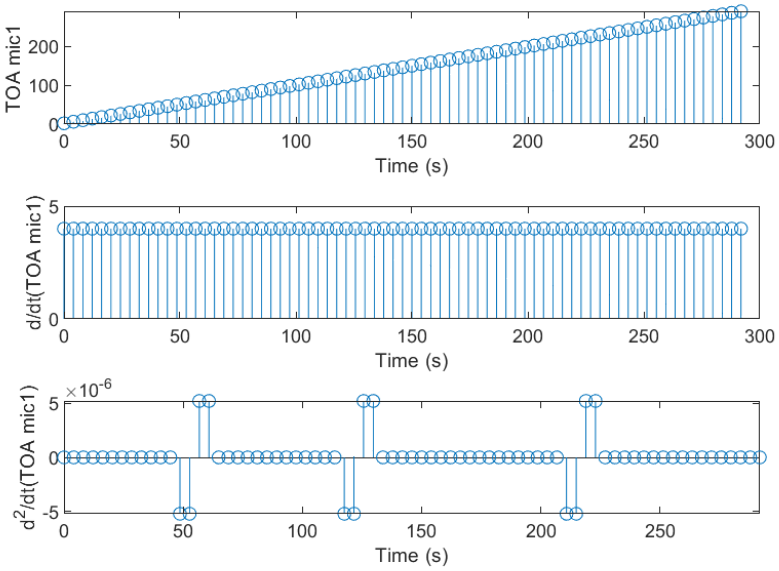


Figure 7.5: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 1's Recording

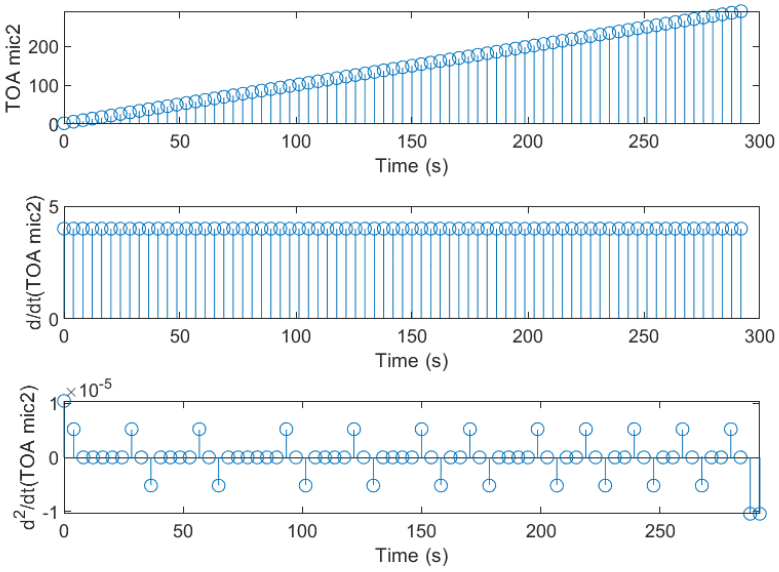


Figure 7.6: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 1's Recording

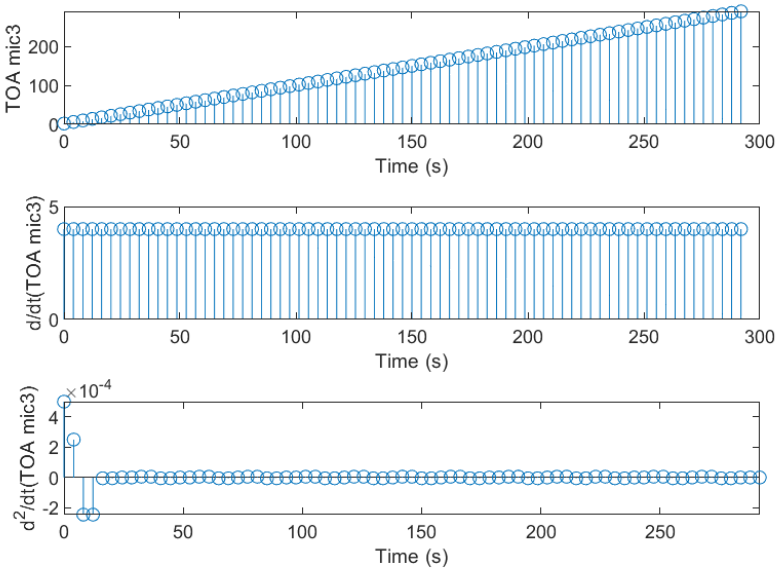


Figure 7.7: TOA vs time, $\frac{d}{dt}TOA$ vs time and $\frac{d^2}{dt}TOA$ vs time For Microphone 1's Recording

7.4 Accuracy and Precision Calculations

$$\overline{Acc} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(X_i - X_{tp})^2 + (Y_i - Y_{tp})^2} \tag{7.1}$$

$$\overline{X}, \overline{Y} = \left(\frac{1}{n}\sum_{i=1}^{n} X_i\right), \left(\frac{1}{n}\sum_{i=1}^{n} Y_i\right) \tag{7.2}$$

$$Precision = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(X_i - \overline{X})^2 + (Y_i - \overline{Y})^2} \tag{7.3}$$

7.5 Stereo Speaker Simulation Results

Ideal Position(X,Y)	Stereo Position(X,Y),(X,Y)	Estimated Position(X,Y)	Triangulation Error(m)
(50,50)	(45,50),(55,50)	(50,50)	0
(50,50)	(50,45),(50,55)	(50,50)	0

Table 7.3: Table showing triangulation error when stereo sources are centred on the ideal position and parallel to X or Y axis

Ideal Position(X,Y)	Stereo Position(X,Y),(X,Y)	Estimated Position(X,Y)	Triangulation Error(m)
(50,50)	(45,45),(55,55)	(45,55)	5

Table 7.4: Table showing triangulation error when stereo sources are centred on the ideal position and diagonal to X and Y axis

7.6 FERS Simulation Example

The following is an example of a FERS XML simulation. The simulation below is set up with the to play a calibration signal, followed by the SoI. Each receiver references a different clock which in this example has a jitter and drift error of 0.1 Radians and 0.1Hz.

```
<!--Simulation input file. Acoustic Triangulation 4 mics-->

    <!--Define General System Parameters-->

    <parameters>
      <starttime>0.0 </starttime>
      <endtime>20.0</endtime>
      <c>343</c>     <!--Speed of Sound-->
      <rate>48000</rate>    <!--Simulation Sample Rate-->
      <export csv="false" binary="true" csvbinary="true"/>
      <!--<adc_bits>24</adc_bits>-->
    </parameters>



    <!--Define Signals-->

    <pulse name="chirp" type="file" filename="../../../../chirp_1000_1500.h5">
```

```
 <power>10</power>
 <carrier>1250</carrier> <!-- VHF Band -->
</pulse>

<pulse name="noise1" type="file" filename="../../../../chirp_1000_1500.h5">
 <power>10</power>
 <carrier>1250</carrier> <!-- VHF Band -->
</pulse>

<!--Define Clocks-->


<timing name="clock">
<frequency>14400</frequency>

</timing>

<timing name="clock1">
<frequency>44100</frequency>
<random_freq_offset> 0.1</random_freq_offset>
<random_phase_offset>0.1</random_phase_offset>
</timing>

<timing name="clock2">
<frequency>44100</frequency>
<random_freq_offset> 0.1</random_freq_offset>
<random_phase_offset>0.1</random_phase_offset>
</timing>

<timing name="clock3">
<frequency>44100</frequency>
<random_freq_offset> 0.1</random_freq_offset>
<random_phase_offset>0.1</random_phase_offset>
</timing>

<timing name="clock4">
<frequency>44100</frequency>
<random_freq_offset> 0.1</random_freq_offset>
<random_phase_offset>0.1</random_phase_offset>
</timing>
```

```
<!--Define Transmitters and Receivers -->
<antenna name="isotropic" pattern="isotropic">
</antenna>

<platform name="Transmitter">
 <motionpath>
  <positionwaypoint>

   <x>0.7</x>
   <y>1.2</y>
   <altitude>0</altitude>
   <time>0</time>
  </positionwaypoint>
 </motionpath>
 <fixedrotation>
  <startazimuth>0.0</startazimuth>
  <startelevation>0</startelevation>
  <azimuthrate>0</azimuthrate>
  <elevationrate>0</elevationrate>
 </fixedrotation>
 <transmitter name="Tx_chirp" type="pulsed" antenna="isotropic" pulse="chirp" timing="clock">
  <prf>0.1</prf> <!-- make this 0.1 for a 10s simulation-->
 </transmitter>
</platform>

<platform name="Transmitter">
 <motionpath>
  <positionwaypoint>
   <x>1.14</x>
   <y>2.055</y>
   <altitude>0</altitude>
   <time>0</time>
  </positionwaypoint>
 </motionpath>
 <fixedrotation>
  <startazimuth>0.0</startazimuth>
  <startelevation>0</startelevation>
  <azimuthrate>0</azimuthrate>
  <elevationrate>0</elevationrate>
 </fixedrotation>
 <transmitter name="Tx_noise1" type="pulsed" antenna="isotropic" pulse="noise1" timing="clock">
  <prf>0.1</prf> <!-- make this 0.1 for a 10s simulation-->
 </transmitter>
```

```
</platform>



<platform name="Mic1_Rx">
 <motionpath>
  <positionwaypoint>
   <x>0</x>
   <y>0</y>
   <altitude>0.0</altitude>
   <time>0</time>
  </positionwaypoint>
 </motionpath>
 <fixedrotation>
  <startazimuth>0</startazimuth>
  <startelevation>0</startelevation>
  <azimuthrate>0</azimuthrate>
  <elevationrate>0</elevationrate>
 </fixedrotation>
 <receiver name="Mic1" type="pulsed" antenna="isotropic"  timing="clock1">
  <window_skip>0</window_skip>
  <window_length>10</window_length> <!-- make this 10 for a 10s simulation-->
  <prf>0.1</prf> <!-- make this 0.1 for a 10s simulation-->
  <noise_temp>0</noise_temp>
 </receiver>
</platform>

<platform name="Mic2_Rx">
 <motionpath>
  <positionwaypoint>
   <x>0</x>
   <y>3</y>
   <altitude>0.0</altitude>
   <time>0</time>
  </positionwaypoint>
 </motionpath>
 <fixedrotation>
  <startazimuth>0</startazimuth>
  <startelevation>0</startelevation>
  <azimuthrate>0</azimuthrate>
  <elevationrate>0</elevationrate>
 </fixedrotation>
 <receiver name="Mic2" type="pulsed" antenna="isotropic"  timing="clock2">
```

```
  <window_skip>0</window_skip>
  <window_length>10</window_length> <!-- make this 10 for a 10s simulation-->
  <prf>0.1</prf> <!-- make this 0.1 for a 10s simulation-->
  <noise_temp>0</noise_temp>
 </receiver>
</platform>

<platform name="Mic3_Rx">
 <motionpath>
  <positionwaypoint>
   <x>3</x>
   <y>0</y>
   <altitude>0.0</altitude>
   <time>0</time>
  </positionwaypoint>
 </motionpath>
 <fixedrotation>
  <startazimuth>0</startazimuth>
  <startelevation>0</startelevation>
  <azimuthrate>0</azimuthrate>
  <elevationrate>0</elevationrate>
 </fixedrotation>
 <receiver name="Mic3" type="pulsed" antenna="isotropic" timing="clock3">
  <window_skip>0</window_skip>
  <window_length>10</window_length> <!-- make this 10 for a 10s simulation-->
  <prf>0.1</prf> <!-- make this 0.1 for a 10s simulation-->
  <noise_temp>0</noise_temp>
 </receiver>
</platform>

<platform name="Mic4_Rx">
 <motionpath>
  <positionwaypoint>
   <x>3</x>
   <y>3</y>
   <altitude>0.0</altitude>
   <time>0</time>
  </positionwaypoint>
 </motionpath>
 <fixedrotation>
  <startazimuth>0</startazimuth>
  <startelevation>0</startelevation>
  <azimuthrate>0</azimuthrate>
```

```
      <elevationrate>0</elevationrate>
     </fixedrotation>
     <receiver name="Mic4" type="pulsed" antenna="isotropic" timing="clock4">
      <window_skip>0</window_skip>
      <window_length>10</window_length> <!-- make this 10 for a 10s simulation-->
      <prf>0.1</prf> <!-- make this 0.1 for a 10s simulation-->
      <noise_temp>0</noise_temp>
     </receiver>
    </platform>
    </simulation>
```

7.7 Record Python Script

```
import multiprocessing
from time import sleep
import os
import sys

import sounddevice as sd
import soundfile as sf



def pi1():
    os.system("ssh pi1@192.168.137.240 sudo nice
    -n -20 arecord  -f S16_LE -r 48000 -d 20 -c
    1 -D plughw:1 pi1.wav")

def pi2():
    os.system("ssh pi2@192.168.137.71 sudo nice
    -n -20 arecord  -f S16_LE -r 48000 -d 20 -c
    1 -D plughw:1 pi2.wav")

def pi3():
    os.system("ssh pi3@192.168.137.198 sudo nice
    -n -20 arecord  -f S16_LE -r 48000 -d 20 -c
    1 -D plughw:1 pi3.wav")

def pi4():
    os.system("ssh pi4@192.168.137.103 sudo nice
    -n -20 arecord  -f S16_LE -r 48000 -d 20 -c
    1 -D plughw:1 pi4.wav")
```

```
if __name__ == '__main__':
    argumentList = sys.argv[1:] # Get file name from input
    file = argumentList[0]

    os.system("mkdir "+str(file)) #Create File to put recordings

    #Define Parallel High-Priority SSH commands
    process1 = multiprocessing.Process(target=pi1)
    process2 = multiprocessing.Process(target=pi2)
    process3 = multiprocessing.Process(target=pi3)
    process4 = multiprocessing.Process(target=pi4)

    #Execute parallel commands
    process1.start()
    process2.start()
    process3.start()
    process4.start()
    sleep(5) #Sleep 5 seconds

    #Define calibration signal
    filename = 'chirp_0_15000_5s.wav'
    # Read and Extract data and sampling rate from file
    data, fs = sf.read(filename, dtype='float32')
    sd.play(data, fs) #Play signal
    status = sd.wait()  # Wait until file is done playing

    sleep(15) #Wait until recordings complete

    #Retrieve the recordings from each device
    os.system("scp pi1@192.168.137.240:pi1.wav "+file)
    os.system("scp pi2@192.168.137.71:pi2.wav "+file)
    os.system("scp pi3@192.168.137.198:pi3.wav "+file)
    os.system("scp pi4@192.168.137.103:pi4.wav "+file)

    exit
exit
```

Bibliography

[1] N. Peña García, L. Aguilera-Cortés, M. González-Palacios, J.-P. Raskin, and A. Herrera-May, "Design and modeling of a mems dual-backplate capacitive microphone with spring-supported diaphragm for mobile device applications," *Sensors*, vol. 18, p. 3545, 10 2018.

[2] "Comparing mems and electret condenser (ecm) microphones," Jan 2019. [Online]. Available: https://www.cuidevices.com/blog/comparing-mems-and-electret-condenser-microphones

[3] M. Konishi, "Study of sound localization by owls and its relevance to humans," *Comparative Biochemistry and Physiology Part A: Molecular Integrative Physiology*, vol. 126, no. 4, pp. 459–469, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1095643300002324

[4] B. Jin, X. Xu, and T. Zhang, "Robust time-difference-of-arrival (tdoa) localization using weighted least squares with cone tangent plane constraint," *Sensors (Basel, Switzerland)*, vol. 18, 03 2018.

[5] D. Voss, "This month in physics history - july 1915: William lawrence bragg works on sound ranging for artillery detection," *Aps.org*, vol. 28, no. 7, 07 2019. [Online]. Available: https://www.aps.org/publications/apsnews/201907/history.cfm

[6] T. Whipple, "Sound technology gives enemy snipers nowhere to hide," May 2019. [Online]. Available: https://www.thetimes.co.uk/article/sound-technology-gives-enemy-snipers-nowhere-to-hide-dr7x8jxl6

[7] L. Alexander, "Amazon echo show 10 buch das detailliertesteste handbuch fur echo show 10 - das neue echo show - inoffizieller ratgeber," 2020. [Online]. Available: \protect\relax$https://www.amazon.com/echo-show-10/dp/B07VHZ41L8/ref=sr_1_4?crid=2JHQP2PPARYE9&keywords=alexa%2Becho&qid=1666250548&qu=eyJxc2MiOiI0LjkyIiwicXNhIjoiNC4yMyIsInFzcCI6IjMuNjgifQ%3D%3D&sprefix=alexa%2Bech%2Caps%2C311&sr=8-4\protect\relax\protect\begingroup\immediate\write\@unused\def\MessageBreak˙\let\protect\edefYourcommandwasignored.\MessageBreakTypeI<command><return>toreplaceitwithanothercommand,\MessageBreakor<return>tocontinuewithoutit.\errhelp\let\def\MessageBreak˙\def\errmessageLaTeXError:Badmathenvironmentdelimiter.¨SeetheLaTeXmanualorLaTeXCompanionforexplanation.˙TypeH<return>forimmediatehelp\endgroup

[8] M. Cobos, J. Perez-Solano, L. Berger, S. Merilampi, and A. Sirkka, "Acoustic-based technologies for ambient assisted living," *Introduction to Smart eHealth and eCare Technologies*, pp. 159–180, 2016.

[9] Inpixon, "Time difference of arrival (tdoa) multilateration." [Online]. Available: https://www.inpixon.com/technology/standards/time-difference-of-arrival

[10] W. Kloot, "Lawrence bragg's role in the development of sound-ranging in world war i," *Notes and Records of The Royal Society - NOTES REC ROY SOC*, vol. 59, pp. 273–284, 09 2005.

[11] G. T. Rude, "Radio acoustic sound ranging," *The International Hydrographic Review*, no. 2, Jun. 2018. [Online]. Available: https://journals.lib.unb.ca/index.php/ihr/article/view/28428

[12] A. Chacon-Rodriguez, P. Julian, L. Castro, P. Alvarado, and N. Hernandez, "Evaluation of gunshot detection algorithms," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 2, pp. 363–373, 2011.

[13] B. Siuru, "Gunshot location systems," *Annotation*, 2007.

[14] S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *Journal of Computing Science and Engineering*, vol. 6, no. 1, pp. 40–50, 2012.

[15] T. Samuelsson, "Sound ranging using multilateration and kalman filter," 2020.

[16] R. Bucher and D. Misra, "A synthesizable vhdl model of the exact solution for three-dimensional hyperbolic positioning system," *Vlsi Design*, vol. 15, 1970.

[17] P. Wu, S. Su, Z. Zuo, X. Guo, B. Sun, and X. Wen, "Time difference of arrival (tdoa) localization combining weighted least squares and firefly algorithm," *Sensors*, vol. 19, no. 11, p. 2554, 2019.

[18] C. Mahapatra and A. R. Mohanty, "Explosive sound source localization in indoor and outdoor environments using modified Levenberg Marquardt algorithm," *Measurement*, vol. 187, p. 110362, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224121012562

[19] V. Zetterberg, M. Pettersson, and I. Claesson, "Comparison between whitened generalized cross correlation and adaptive filter for time delay estimation with scattered arrays for passive positioning of moving targets in baltic sea shallow waters," in *Proceedings of OCEANS 2005 MTS/IEEE*, 2005, pp. 2356–2361 Vol. 3.

[20] Y. LIU, J. S. Bolton, and P. Davies, "Acoustic Source Localization Techniques and Their Applications," Jun. 2021. [Online]. Available: https://nae.edu/255823/Acoustic-Source-Localization-Techniques-and-Their-Applications

[21] G. S. Wong, "Speed of sound in standard air," *The Journal of the Acoustical Society of America*, vol. 79, no. 5, pp. 1359–1366, 1986.

[22] T. A. Rhinehart, L. M. Chronister, T. Devlin, and J. Kitzes, "Acoustic localization of terrestrial wildlife: Current practices and future opportunities," *Ecology and Evolution*, vol. 10, no. 13, pp. 6794–6818, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ece3.6216

[23] S. Liu, C. Zhang, and Y. Huang, "Research on acoustic source localization using time difference of arrival measurements," in *Proceedings of 2012 International Conference on Measurement, Information and Control*, vol. 1, 2012, pp. 220–224.

[24] J. yong Yoon, J.-W. Kim, W.-Y. Lee, and D.-S. Eom, "A tdoa-based localization using precise time-synchronization," in *2012 14th International Conference on Advanced Communication Technology (ICACT)*, 2012, pp. 1266–1271.

[25] A. A. Syed and J. Heidemann, "Time synchronization for high latency acoustic networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications.* IEEE, 2006, pp. 1–12.

[26] L. Gasparini, O. Zadedyurina, G. Fontana, D. Macii, A. Boni, and Y. Ofek, "A digital circuit for jitter reduction of gps-disciplined 1-pps synchronization signals," in *2007 IEEE International Workshop on Advanced Methods for Uncertainty Estimation in Measurement*, 2007, pp. 84–88.

[27] H. Cho, J. Jung, B. Cho, Y. Jin, S.-W. Lee, and Y. Baek, "Precision time synchronization using ieee 1588 for wireless sensor networks," in *2009 International Conference on Computational Science and Engineering*, vol. 2, 2009, pp. 579–586.

[28] S. Kumar and K. B. Moore, "The evolution of global positioning system (gps) technology," *Journal of science Education and Technology*, vol. 11, no. 1, pp. 59–80, 2002.

[29] W. Lewandowski, G. Petit, and C. Thomas, "Precision and accuracy of gps time transfer," *IEEE Transactions on Instrumentation and Measurement*, vol. 42, no. 2, pp. 474–479, 1993.

[30] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[31] A. Brutti, M. Omologo, and P. Svaizer, "Comparison between different sound source localization techniques based on a real data collection," in *2008 Hands-Free Speech Communication and Microphone Arrays*, 2008, pp. 69–72.

[32] P. Bourke, "Cross correlation," *Cross Correlation", Auto Correlation—2D Pattern Identification*, 1996.

[33] J.-C. Yoo and T. H. Han, "Fast normalized cross-correlation," *Circuits, Systems, and Signal Processing*, vol. 28, no. 6, p. 819–843, 2009.

[34] J. M. Villadangos, J. Ureña, J. J. García-Domínguez, A. Jiménez-Martín, Hernández, and M. C. Pérez-Rubio, "Dynamic adjustment of weighted gcc-phat for position estimation in an ultrasonic local positioning system," *Sensors*, vol. 21, no. 21, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/21/7051

[35] Ritu and S. K. Dhull, "A comparison of generalized cross-correlation methods for time delay estimation," *IUP Journal of Telecommunications*, vol. 8, no. 4, pp. 31–46, 11 2016. [Online]. Available: https://www.proquest.com/scholarly-journals/comparison-generalized-cross-correlation-methods/docview/1855901212/se-2

[36] X. Wan and Z. Wu, "Sound source localization based on discrimination of cross-correlation functions," *Applied Acoustics*, vol. 74, no. 1, pp. 28–37, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X1200165X

[37] B. R. Mahafza, *Pulse Compression*. CHAPMAN amp; HALL/CRC.

[38] juanfregoso, "Understanding Mic Specifications: Max SPL and THD," Mar. 2022. [Online]. Available: https://mxlmics.com/understanding-mic-specifications-max-spl-and-thd/

[39] "Pulse width modulation." [Online]. Available: https://www.realdigital.org/doc/333049590c67cb553fc7f9880b2f79c3

[40] B. Miller, "[insights gt; benchtop]," Apr 2022. [Online]. Available: https://blogs.keysight.com/blogs/tech/bench.entry.html/2022/04/29/i2s__the__digital__audiointerfacethatrockstoday-6CGE.html

[41] V. K. Garg, *Pulse Code Modulation*. Morgan Kaufmann, 2007.

[42] Arthur, "What Is A Good Signal-To-Noise Ratio For A Microphone?" [Online]. Available: https://mynewmicrophone.com/what-is-a-good-signal-to-noise-ratio-for-a-microphone/

[43] 2022. [Online]. Available: https://store.arduino.cc/products/arduino-uno-wifi-rev2

[44] 2022. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[45] 2022. [Online]. Available: https://www.espressif.com/en/products/modules

[46] Invensens, "INMP441 Datasheet Omnidirectional Microphone with Bottom Port and I2 S Digital Output." [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf

[47] "Build Your Own Amazon Echo Using a RPI and ReSpeaker HAT." [Online]. Available: https://www.hackster.io/idreams/build-your-own-amazon-echo-using-a-rpi-and-respeaker-hat-7f44a0

[48] "Sound sensor." [Online]. Available: https://www.waveshare.com/sound-sensor.htm

[49] "Adafruit PDM Microphone Breakout." [Online]. Available: https://learn.adafruit.com/adafruit-pdm-microphone-breakout/overview

[50] "Analog to Digital Converter - ESP32 - — ESP-IDF Programming Guide v4.2 documentation." [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/v4.2/esp32/api-reference/peripherals/adc.html

[51] J. (https://electronics.stackexchange.com/users/199993/justme), "Esp32 adc not good enough for audio/music?" Electrical Engineering Stack Exchange, uRL:https://electronics.stackexchange.com/q/530969 (version: 2020-11-06). [Online]. Available: https://electronics.stackexchange.com/q/530969

[52] "Digital Audio Basics: Audio Sample Rate and Bit Depth." [Online]. Available: https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html

[53] Adafruit_Support_Mike, "Adafruit PDM Microphone with Raspberry Pi 4 - adafruit industries." [Online]. Available: https://forums.adafruit.com/viewtopic.php?f=8&t=186884

[54] "Arduino ADC: Everything you Must Know about the Built-In ADC." [Online]. Available: https://www.best-microcontroller-projects.com/arduino-adc.html

[55] "specification control drawing 19.2mhz crystal resonator2022," 2022. [Online]. Available: https://www.farnell.com/datasheets/2860337.pdf

[56] X. Zeng, "Does the raspberry pi zero 2 require a heatsink? — picockpit | monitor and control your raspberry pi: free for up to 5 pis!" 2022. [Online]. Available: https://picockpit.com/raspberry-pi/does-the-raspberry-pi-zero-2-require-a-heatsink/

[57] R. Meades, "Clock drift - raspberry pi forums," 2022. [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=211140

[58] W. MicroElectronics, "Wm8960," Oct 2011. [Online]. Available: http://www.sunnyqi.com/upLoad/product/month_1306/WM8960.pdf

[59] R. Andrews, "Ti training," 2017. [Online]. Available: https://training.ti.com/sites/default/files/docs/adcs-sar-delta-sigma-basic-operation-presentation.pdf

[60] "Need to open a wav file?" [Online]. Available: https://www.roxio.com/en/file-formats/wav-file/

[61] "Raspberry Pi Documentation - Raspberry Pi OS." [Online]. Available: https://www.raspberrypi.com/documentation/computers/os.html

[62] "What is SSH (Secure Shell) and How Does it Work? Definition from TechTarget." [Online]. Available: https://www.techtarget.com/searchsecurity/definition/Secure-Shell

[63] "arecord(1) - Linux man page." [Online]. Available: https://linux.die.net/man/1/arecord

[64] M. Murray, "Debouncing a Switch in Hardware or Software," Sep. 2019. [Online]. Available: https://www.thegeekpub.com/246471/debouncing-a-switch-in-hardware-or-software/

[65] "4. Configuring Remote GPIO — GPIO Zero 1.6.2 Documentation." [Online]. Available: https://gpiozero.readthedocs.io/en/stable/remote_gpio.html

[66] "pigpio library." [Online]. Available: https://abyz.me.uk/rpi/pigpio/python.html

[67] "Using SCP to copy files between computers, with examples." [Online]. Available: https://www.ssh.com/academy/ssh/scp