Localization of Bryde's whales using time difference of arrival principles

by

Joshua Cormick

Thesis presented in fulfilment of the requirements for the degree of

Master of Engineering (Electronic)

in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof D.J.J. Versfeld

March 2021

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
J. Cormick

Date: March 2021

Abstract

The thesis presented gives the detailed practical implementation for a passive audio detection system that makes use of time difference of arrival theory to locate the source of audio signals emitted from Bryde's whales. The system consists of three individual floating sensors deployed in the water that communicate with the user-unit on the research vessel. The sensors are comprised of Raspberry Pis and various other specific hardware. The transfer of the sensor readings and the calculations done with said data allows for the location of the audio signal to be calculated through a time difference of arrival algorithm.

The overall audio signal source localization system is made possible by integrating various sub-systems. These being; detection, sensor tracking, communication, time synchronization, and signal location calculation. The theoretical principles behind each of these subsystems are discussed as well as their practical implementation, simulations and testing.

In order to locate the source of an audio signal, one must first be able to identify whether an input signal to the system is the desired signal. A dynamic time warping algorithm is made use of in order to detect the desired audio signals. The validity of this algorithm on Bryde's whales has already been proven in the past, however further developments had to be made when implementing this algorithm in real-time.

The locations of the sensors are determined by fitting each sensor with a GPS module. The data that the GPS module feeds to the system is known as NMEA data, this is used to track the sensors. The sensor positions as well as detection instances are communicated to the user-unit for processing via LoRa. LoRa is a useful communication technology as the data can be transmitted directly over long distances.

A time difference of arrival algorithm is run on the user-unit. This algorithm makes use of the data received from the sensors to mathematically locate the source of the audio signal. The location of the sensors and any potential locations for the signal source is shown graphically via a Python interface on the screen attached to the user unit.

After testing it was found that the tracking and communication systems function adequately. The DTW based detector proved to run sufficiently, though there were instances of false-positive detections. Additionally, it was found that the TDOA system tested to be accurate to within several hundred meters when at a scale of kilometers when tested using a microcontroller in the laboratory.

Through testing the various sub-systems of the overall system appears as it would function sufficiently, though the system was only laboratory tested and was never deployed in the ocean due to time constraints.

Opsomming

Die tesis wat aangebied word, gee ‘n indiepte praktiese implementering vir ’n passiewe akoestiese deteksietelsel wat gebruik maak van verskil in aankomstyd-teorie om die posisie van klankseine van Bryde se walvisse te lokaliseer. Die stelsel bestaan uit drie individuele sensors wat in die water dryf, wat elk met die gebruikerseenheid op die navorsingsvaartuig kommunikeer. Die sensors bestaan uit Raspberry Pis en verskeie doelgerigte komponente.Die oordrag van die sensorlesings en die berekeninge wat met die data gedoen word, maak dit moontlik om die ligging van die klanksein te bepaal deur middel van ’n verskil in aankomstydalgoritme.

Die algehele akoestiese lokaliseringstelsel word moontlik gemaak deur verskillende substelsels te integreer, naamlik; opsporing, sensorvolging, kommunikasie, tydsinchronisasie, en seinlokalisering. Die teoretiese beginsels onderliggend aan elke substelsel word bespreek, asook die praktiese implementering, simulasie en toetsing van elke substelsel.

Om die oorsprong van ’n klanksein te lokaliseer, moet eers bepaal word of die insetsein na die stelsel die gewenste sein is. ’n Dinamiese tydsverskuiwingsalgoritme is gebruik as detektor vir die verlangde klankseine. Die geldigheid van hierdie algoritme op Bryde se walvisse is reeds in die verlede bewys, maar verdere aanpassings is gemaak om hierdie algoritme intyds te implementeer.

Die ligging van die sensors word bepaal deur elke sensor met ‘n GPS toe te rus. Die data wat die GPS-module na die stelsel voer, staan bekend as NMEA-data. Dit word gebruik om die sensors op te spoor. Die sensorposisies sowel as deteksies word deur LORA gekommunikeer na die gebruikerseenheid vir verwerking. LoRa is ’n nuttige kommunikasietegnologie wat dit moontlik maak om data direk oor lang afstande oor te dra.

’n Verskil in aankomstydalgoritme word op die gebruikerseenheid gebruik. Hierdie algoritme maak gebruik van die data wat van die sensors ontvang word om die oorsprong van die klanksein wiskundig te lokaliseer. Die ligging van die sensors sowel as moontlike liggings vir die seinbron word grafies vertoon op die skerm van die gebruikerseenheid deur van ’n Python-koppelvlak gebruik te maak.

Na toetsing is bevind dat die opsporing- en kommunikasiestelsels voldoende funksioneer. Die dinamiese tydsverskuiwings-gebaseerde detektor het voldoende gewerk, alhoewel daar gevalle was waar vals-positiewe opsporings gemerk is. Daarbenewens is bevind deur laboratoriumtoetse met behulp van ‘n mikroverwerker dat die verskil in aankomstydstelsel akkuraat is tot binne ’n paar honderd meter, getoets op ’n skaal van kilometers.

Deur toetsing van die verskillende substelsels van die algehele stelsel, blyk dit dat die stelsel voldoende funksioneer, alhoewel die stelsel slegs in die laboratorium getoets is en nooit in die oseaan ontplooi is nie as gevolg van tydsbeperkings

Acknowledgments

- My Mother, for her unconditional love and support, for inspiring me to be better and always putting the ones that she loves before herself. She is truly the most amazing person I have ever known.
- Professor Jaco Versfeld, my supervisor, for allowing me to pursue this academic endeavor and aiding me along the way.
- The NRF, for providing financial support to my research.
- Mr van Eeden for his help in milling the PCBs required for the system hardware.

Nomenclature

CSS	Chirp spread spectrum
DOAE	Direction of arrival estimation
DTW	Dynamic time warping
GNSS	Global navigation Satellite System
GPS	Global Positioning System
LBP	Local binary pattern
LoRa	Long range radio
LPWAN	Low power wide area network
NMEA	National Marine Electronics Association
NTP	Network time protocol
PPS	Pulse per second
RMS	Root mean square
SPI	Serial peripheral interface
TDOA	Time difference of arrival
TOA	Time of arrival
UART	Universal asynchronous receiver-transmitter

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

It would appear to be general knowledge to the public that the most common way to determine the location of whales is visually when they surface and inhale air. However, this is not a reliable form of locating animals that would otherwise spend the majority of their lives under the surface. This is especially true for Bryde's whales which are known to have erratic behaviour [1].

As such technological systems have been put in place in order to locate and study aquatic animals. One such method of locating the source of signals, such as aquatic animal vocalizations, is time difference of arrival.

Time difference of arrival based systems allow for localization of audio signals created by animals through passive means. Though there should also be a detection aspect implemented into these systems to confirm that the correct vocalizing animal, which is the signal source, is being located.

Through the utilization of multiple sensors deployed in an environment with known locations, the differences in arrival times of the signal at each sensor can be calculated. As such the location of the source of this signal can be determined through mathematical means implemented in the system.

The determining of the difference in arrival times of the signal at each sensor requires the sensors to have some method of detecting that the signal has arrived as well as the sensors having some degree of time synchronization between themselves.

1.2 Problem statement

In this thesis, we set out to develop and analyze a system that would be capable of locating the position of Bryde's whales in real-time using a time difference of arrival algorithm. In order to perform this task various sufficient tracking, communication, timing, and detection subsystems are required to work individually and in tandem with one another to produce meaningful results.

1.3 Objectives of this thesis

A primary aim of the thesis presented is to be able to detect Bryde's whale vocalizations from the input audio signals. This is essential for the localization of the vocalizing whale. The next primary aim of this thesis would be to use the detections and other information to locate vocalizing Bryde's whales and present the findings graphically in two-dimensions.

Three sensors free-floating in the ocean each have detection algorithms running on there local hardware. The sensors have the ability to process the audio data through the use of hydrophones. The sensors are to communicate with the user-unit using radio waves, specifically long range (LoRa) technology. The information that is sent by each sensor is the global positioning system (GPS) coordinates of the sensor as well as the timestamps for instances of signal detection.

An important aspect of the system that may not be immediately apparent is that it, and the researchers present, should interfere with the surrounding sea life as little as possible. For this reason, a passive free-floating system is being implemented rather than something resembling active sound navigation and ranging (SONAR), where the transmitted signals may interfere with marine life.

As previously stated the system should be running in real-time. This posed possibly one of the largest challenges faced when implementing the theory behind all the internal systems practically as there are many calculations done to be done in small time frames.

For the development of the time difference of arrival (TDOA) system, certain assumptions have been made from the start of development. These assumptions include the term real-time referring to the calculation presentation of information within a few seconds of receiving the required data. The term real-time is used here as a contrast to a post-processing approach. Another such assumption is that there will be a degree of error present in the calculations of the system within the extent of multiple meters. This error is present as the hardware used is only accurate to within a given degree of error.

1.4 Overview of this thesis

Chapter 2 contains research on various systems and techniques implemented by researchers in the past.

In Chapter 3 the theoretical principles chosen to be implemented in the different internal subsystems are described.

Chapter 4 allows for the descriptions of the implementation of the previously discussed theoretical principles as well as the hardware and software used.

The contents of Chapter 5 are the simulations and results of tests for the various internal systems as well as the testing of various implementation methods to make the system the most efficient and accurate.

Chapter 6 describes the conclusions and recommendations for the future development of the system.

Chapter 2

Literature survey

2.1 Introduction

There are many facets to the system that are required for it to function properly. The main functions of the system that must be implemented in the various subsystems are the ability to: detect whether a signal is a Bryde's whale short pulse call, track the sensors' GPS coordinates, transmit data from the sensors, and use transmitted sensor data to locate the audio signal source point of origin.

This chapter serves the purpose of noting and comparing the different methods of implementation in similar systems from the past, rather than discussing and illustrating in great detail the concepts of which they are composed. The concepts discussed are to be explained to an acceptable extent of understanding. However, more in-depth discussions and illustrations of the methods chosen to be implemented for the various functions of the system are reserved for Chapter 3.

2.2 Locating the audio signal source

2.2.1 Localization - Time difference of arrival (TDOA)

When performing TDOA the time of arrival values for a signal reaching a number of sensors at known locations is converted to differences in distances. These differences in distances when combined with the known sensor locations set up hyperbolic equations which at the points of intersection can be used to locate the origin of the signal.

Muanke et al. [2] devised a TDOA system to be used to locate Manatees. The researchers make use of the Hilbert transform in conjunction with cross-correlation. The implementation of the TDOA algorithm typically resembles Equation (2.1),

$$d_{i,j} = d_i - d_j = \sqrt{(x_i - x)^2 + (y_i - y)^2} - \sqrt{(x_j - x)^2 + (y_j - y)^2} \tag{2.1.a}$$

or alternatively,

$$d_{i,j} = d_i - d_j = \sqrt{(x - x_i)^2 + (y - y_i)^2} - \sqrt{(x - x_j)^2 + (y - y_j)^2}. \tag{2.1.b}$$

This equation denotes the difference in distances as mentioned previously. The variables d_i and d_j denote the distance from the i-th and j-th sensor to the signal source respectively. The number of sensors can range from 1 to N. It is assumed that the difference in these distance values, denoted by $d_{i,j}$, is known by multiplying the difference in time measurements by the speed at which the signal is known to propagate through the medium.

In addition to this, because the coordinates of the sensors, denoted by the i or j subscript, in Equation (2.1) are also known, one can begin to calculate the possible x and y values that represent the location of the signal source. These possible x and y values that satisfy the equation produce a hyperbolic curve. Using multiple sensors at different locations in tandem with one another generates multiple hyperbolic equations, at the point these curves intersect is the calculated origin of the signal.

Muanke et al. [2] go on to bring up a point of extreme importance when implementing the TDOA algorithm. If all the curves do not intersect at a signal point due to non-idealities in the system this creates a problem when trying to locate the correct position. They address this problem by designating the source location desired as the point of minimum distance between the different hyperbolas. Finding this minimum distance is made possible by implementing a least squares algorithm.

Figure 2.1 below is adapted from a paper by Rosić et al. [3] where the researchers illustrate the problem caused by the timing inaccuracy and methods implemented to find the best location for the signal source.

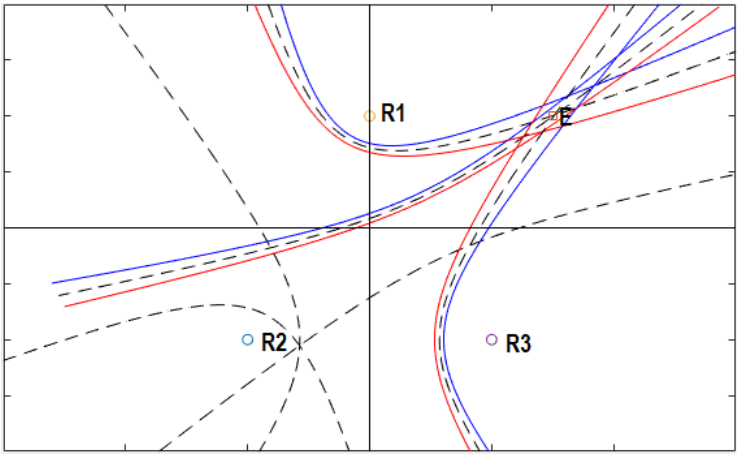

Figure 2.1: Margins of error in a TDOA system [3]

The information that can be gathered from the image is that the dotted lines are the accurate and ideally calculated hyperbolas and the solid lines are the hyperbolas calculated with errors.

In Equation (2.2) the arrival times at each sensor are denoted by the variable t. In this instance, the sensors are regarded as sensor i and sensor j. The variable c represents the speed at which the signal moves through the given medium. When multiplied together these values yield a value for the variable d which represents the difference in distance

between each sensor and the signal source. The variable n denotes an error that has been presumed to occur in the system for an arbitrary reason. The equation is written as,

$$r_{i,j} = c|t_i - t_j| = d_{i,j} + n_{i,j} \qquad where\ i \neq j. \tag{2.2}$$

The equations in question for each given set of sensors being examined can be rewritten in matrix notation if it is desired, $\mathbf{x} = [x, y]^T$ can be used to designate the signal source location and $\mathbf{x_j} = [x_j, y_j]^T$ can be used to designate the jth sensor location.

2.2.1.1 Non-linear least squares

As noted in the paper by Rosić et al. [3], non-linear least squares minimizes the objective function denoted by the indicator $J_{NLS}(\tilde{x})$. This is defined as the sum of squared residuals between the estimated and the measured TDOA values. Its corresponding mathematical equation as noted in Equation (2.3) with $\tilde{x}$ denoting the optimization variable is written as,

$$J_{NLS}(\tilde{x}) = min \sum_{i=1}^{N} R_{es,i}^2(\tilde{x}). \tag{2.3}$$

The value for $R_{es,i}(\tilde{x})$ as seen in Equation (2.3) is determined by the equation,

$$R_{es,i}(\tilde{x}) = \tilde{r}_{i,j} - r_{i,j}. \tag{2.4}$$

The value of $\tilde{r}_{i,j}$, as seen previously, is the measured distance and $\hat{x}$ denotes the optimal solution. This is expressed as,

$$\hat{x} = arg\ min\ J_{NLS}(\tilde{x}). \tag{2.5}$$

2.2.1.2 Weighted least squares

As noted in the paper by Rosić et al. [3], the non linear hyperbolic equations seen in Equation (2.1) and Equation (2.2) can be transformed into linear equations by squaring the combination of these equations to eliminate the square root signs. The combination of the equations is shown as,

$$r_{i,j} = \sqrt{(x - x_i)^2 + (y - y_i)^2} - \sqrt{(x - x_j)^2 + (y - y_j)^2} + n_{i,j} \qquad where\ i \neq j, \tag{2.6}$$

with the resultant squared and algebraically manipulated equation illustrated as the following,

$$(x_i - x_j)(x - x_j) + (y_i - y_j)(y - y_j) + r_{i,j}R_f = 0.5((x_i - x_j)^2 + (y_i - y_j)^2 - r_{i,j}^2) + m_{i,j} \tag{2.7}$$

where $i \neq j$.

The values of R_f and $m_{i,j}$, as seen in Equation (2.7), are determined by Equation (2.8) and Equation (2.9) as denoted as,

$$R_f = d_j = \sqrt{(x - x_j)^2 + (y - y_j)^2} \tag{2.8}$$

and

$$m_{i,j} = d_i n_{i,j}. \tag{2.9}$$

Additionally it is worth noting that $n_{i,j}^2$ is neglected when performing the algebraic manipulation to form Equation (2.7) for this approach.

The system is now considered linear and can be converted to the matrix form written as,

$$\mathbf{A}\boldsymbol{\theta} = \mathbf{b} + \mathbf{m}. \tag{2.10}$$

The $\mathbf{A}, \boldsymbol{\theta}, \mathbf{b}$ and $\mathbf{m}$ values are denoted as the following:

$$\mathbf{A} = \begin{bmatrix} (x_i - x_j) & (y_i - y_j) & r_{i,j} \\ (x_{i+1} - x_j) & (y_{i+1} - y_j) & r_{i+1,j} \\ (x_{i+2} - x_j) & (y_{i+2} - y_j) & r_{i+2,j} \\ \vdots & \vdots & \vdots \\ (x_N - x_j) & (y_N - y_j) & r_{N,j} \end{bmatrix} \qquad (for\ i \in \{1, 2, 3, ..., N\}\ and\ i \neq j) \tag{2.11}$$

$$\boldsymbol{\theta} = [(x - x_j) \quad (y - y_j) \quad R_f]^T \tag{2.12}$$

$$\mathbf{b} = 0.5 \begin{bmatrix} (x_i - x_j)^2 & (y_i - y_j)^2 & r_{i,j}^2 \\ (x_{i+1} - x_j)^2 & (y_{i+1} - y_j)^2 & r_{i+1,j}^2 \\ (x_{i+2} - x_j)^2 & (y_{i+2} - y_j)^2 & r_{i+2,j}^2 \\ \vdots & \vdots & \vdots \\ (x_N - x_j)^2 & (y_N - y_j)^2 & r_{N,j}^2 \end{bmatrix} \qquad (for\ i \in \{1, 2, 3, ..., N\}\ and\ i \neq j) \tag{2.13}$$

$$\mathbf{m} = [m_{i,j} \quad m_{i+1,j} \quad m_{i+2,j} \quad ... \quad m_{N,j}]^T \qquad (for\ i \in \{1, 2, 3, ..., N\}\ and\ i \neq j) \tag{2.14}$$

According to Rosić et al. [3], these equations allow for the weighted least squares (WLS) objective function to be produced which is illustrated as,

$$J_{WLS}(\boldsymbol{\theta}) = (\mathbf{A}\boldsymbol{\theta} - \mathbf{b})^T \mathbf{W} (\mathbf{A}\boldsymbol{\theta} - \mathbf{b}). \tag{2.15}$$

The value for $\mathbf{W}$ as seen in the previous equation is noted to be the weighting matrix. Its value is denoted by, $\mathbf{W} = (E\{\mathbf{m}\mathbf{m}^T\})^{-1}$. The researchers then go on to establish that the unconstrained optimization problem can thus be formulated as

$$min\ J_{WLS}(\boldsymbol{\theta}). \tag{2.16}$$

Furthermore, Equation (2.17) shows the WSL algebraic closed-form solution denoted by $\hat{x}_{WSL}$ which minimizes the objective function. This is written as

$$\hat{x}_{WLS}(\boldsymbol{\theta}) = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{b}. \tag{2.17}$$

2.2.2 Direction of arrival estimation (DOAE)

As can be assumed from the name 'Direction of arrival estimation', the process of DOAE is not to locate the audio signal source but rather to simply locate the direction from which the audio signal propagates. Kunin et al. [4] have described the processes of TDOA and DOAE in an article based on acoustic sensor arrays. Figure 2.2 below is adapted from the figure which they make use of to illustrate their points on the DOAE process.

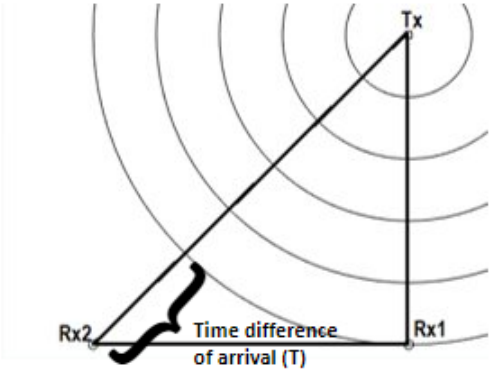

Figure 2.2: A standard DOAE setup illustration [4]

Tx has been designated as the transmitter which would be transmitting a signal through a given medium. Rx1 and Rx2 are the sensors used to detect this audio signal. It is seen in Figure 2.2 above that while DOAE serves a different purpose to TDOA localization, the DOAE process still makes use of a time difference of arrival based principle. The difference in times for the signal to reach the sensor is denoted by the letter T.

There needs to be an approximation made before the process can continue, according to Kunin et al. [4] the estimation of for direction of the source can be obtained through the use of the far-field model. This model assumes that the signal source is far away enough that the propagating waves do not appear to be circular, as they are in Figure 2.2, instead, they approximate a linear or planar form due to the assumed extremely large radius of the circular propagation. They illustrate this in a similar figure to the one shown in Figure 2.3.

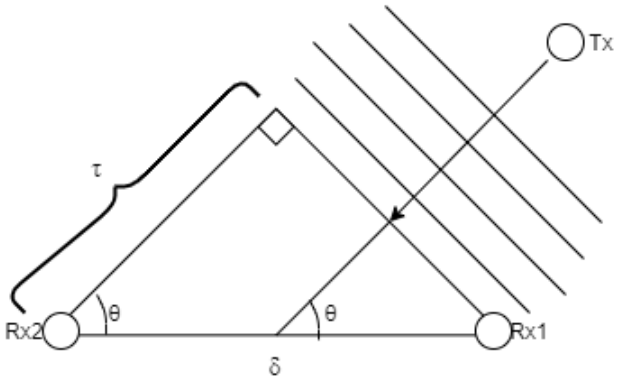

Figure 2.3: Far-field DOAE illustration
[4]

The difference in time value (T) can be multiplied by the speed at which the signal propagates through the designated medium to result in a distance value (τ). This value is then used in the equation,

$$\tau = \delta cos(\theta), \tag{2.18}$$

along with an assumed known value for δ to result in a calculated value for θ. This angle results in the direction of arrival estimation.

In the master's thesis by Vitaliy Kunin [5], the researcher has noted a near field module which does not make use of these previously mentioned assumptions and approximations. Figure 2.4 shown below is adapted from the researcher's illustration of the near field module being elaborated on.

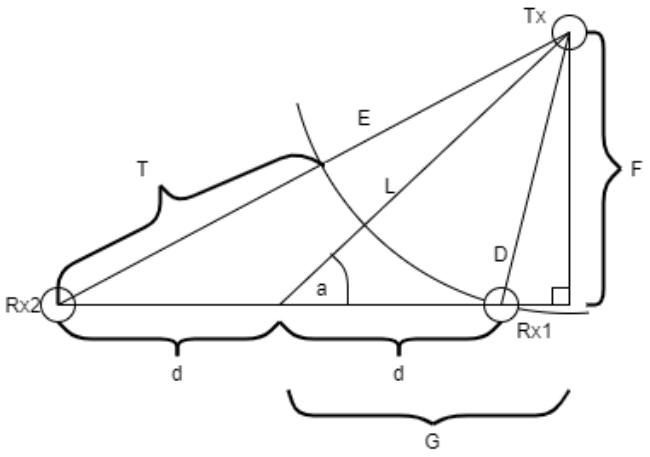

Figure 2.4: Near-field DOAE illustration
[5]

The sensors and the transmitter are designated by the same labels as seen in the previous figures. The process of using this near field model is more complex than the far-field model. The value of d is given to be half the distance between the sensors. L, E, and D are designated as the distances from, the midpoint between the sensors, sensor 2, and

sensor 1 to the signal source respectively. The value for T is denoted by the time difference of arrival that corresponds to what has been previously seen from the far-field model and a is the angle corresponding to the direction of arrival.

The distance value of T is determined as τ was when calculated with the far-field model previously. The process and equations that follow are performed according to the laws of trigonometry in conjunction with inspection of the figure. They are the steps noted in Vitaliy Kunin's thesis [5].

Initially the distances are defined in terms of one another,

$$T = E - D \tag{2.19}$$

and

$$L = \frac{(E + D)}{2}. \tag{2.20}$$

Additionally, the sin and cos functions of the angle a yields values for lengths F and G. It can be seen that,

$$sin(a) = F/L \tag{2.21}$$

yields a value for F when written as

$$F = sin(a)L \tag{2.22}$$

and

$$cos(a) = G/L \tag{2.23}$$

yields a value for G when written as

$$G = cos(a)L. \tag{2.24}$$

Then having established these values one can make use of Pythagoras' Laws for right angle triangles to yield values for,

$$D = \sqrt{(Lcos(a) - d)^2 + (Lsin(a))^2} \tag{2.25}$$

and

$$E = \sqrt{(Lcos(a) + d)^2 + (Lsin(a))^2}. \tag{2.26}$$

As such by combining equations 2.26, 2.25 and 2.19 one can produce a value for T written as,

$$T = \sqrt{(Lcos(a) + d)^2 + (Lsin(a))^2} - \sqrt{(Lcos(a) - d)^2 + (Lsin(a))^2}. \tag{2.27}$$

It is noted that in this system the locations of the sensors are known and the T is given from a measured value. Equation (2.27) gives the relationship between T and the angle of arrival (a). An important note made by Vitaliy Kunin [5] is that two-dimensional localization can be performed with two receivers as seen in the DOAE illustrations when the distances between the source and the receivers are known and with three receivers when

only the TDOA information is known. This has been shown in the TDOA localization section.

The DOAE principles previously discussed are now taken further with the aid of another master's thesis by Carlos Fernández Scola and María Dolores Bolaños Ortega [6]. In this thesis, the same approach of a near field model is taken. However, in this illustration, the equations using the points of interest make use of a Cartesian coordinate system. If their approach were to be put in different terms such that Figure 2.4 can still be used as a visual reference, it would be written as,

$$E = \sqrt{(Rx2_x - x)^2 + (Rx2_y - y)^2} \tag{2.28}$$

and

$$D = \sqrt{(Rx1_x - x)^2 + (Rx1_y - y)^2}. \tag{2.29}$$

Additionally, it is noted that for calculation purposes the Tx x and y values are denoted simply as "x" and "y" while the Rx values are denoted by their suffix first.

Using Equation (2.19) in conjunction with equations 2.28 and 2.29 and eliminating square roots by squaring the values results in the value of T being determined as,

$$T^2 = E^2 + D^2 - 2ED. \tag{2.30}$$

If a Cartesian coordinate axis system for Figure 2.4 is set up such that the midpoint between the sensors is the origin point (0;0) of the axes, logically, $Rx2_x = -Rx1_x$ and $Rx2_y = Rx1_y = 0$. As such Equation (2.30) is now simplified, reordered and written in Cartesian coordinates. It is now written as

$$y = \pm\sqrt{\frac{T^2}{4} - Rx1_x^2 + x^2(\frac{4Rx1_x^2}{T^2} - 1)}. \tag{2.31}$$

As previously mentioned the locations of the sensors are presumed to be known and the T value was produced previously from a measured value. Thus Equation (2.31) can be used to plot an equation of possible x and y values. Though there is a constraint, this being that the value under the square root must be equal or greater than zero. This constraint is denoted as,

$$x \geq \sqrt{-\frac{T^2(T^2 - 4Rx1_x^2)}{4(4Rx1_x^2 - T^2)}}. \tag{2.32}$$

According to the aforementioned thesis [6], the slope of this resultant curve from Equation (2.31) can be determined. From this, a value for the angle a is calculated by determining the arc-tangent of this slope and as such a resultant indicator for the direction of arrival has been estimated. This equation for finding the value of a is written as

$$a = tan^{-1}(\frac{dy(x)}{dx}). \tag{2.33}$$

2.2.3 TDOA and DOAE comparison

The most prevalent advantage of TDOA over DOAE is the ability of the process to actually locate the point of origin for the audio signal rather than giving a bearing on which it can be found.

Though a clear advantage of DOAE is that the process is capable of functioning using just two sensors, as seen performed by Scola et al. [6]. It has been seen in multiple resources that while performing TDOA is possible with just three sensors, there are ambiguities present. These ambiguities can be eliminated by adding a fourth sensor. As such, this would double the amount of hardware that is required for TDOA compared to DOAE resulting in higher costs.

However, it is theorized that by using a graphically represented TDOA algorithm as well as utilizing prior information from the sensors, one can construct a TDOA localization algorithm with acceptable accuracy using three sensors. In addition, for the purposes of this thesis, the DOAE has an inherent problem, being that the sensors are free-floating in the ocean meaning that the axis upon which the calculations are based would be ever-changing. This would result in the angle of arrival being placed on a varying axis, which when out at sea on a research vessel is difficult to observe given that the vessel requires to be constantly sailed.

As such TDOA would be more effective for the purposes of this thesis, given that the results that it yields allow for locating of Bryde's whales with proximity to the research vessel.

2.3 Audio signal detection

2.3.1 A brief overview of whale vocalization signal processing

When analyzing the signals emitted during whale vocalizations in the past, researchers have used many different techniques for different species of whales. This section serves as a brief overview of which methods have been used in the past in order to process whale vocalizations. Two specific methods considered for the purposes of this thesis are discussed in the subsequent subsections.

In a paper by Ogundile et al. [7], two algorithms are used in order to detect Bryde's whale vocalizations. These algorithms are dynamic time warping and linear predictive coding. Brown et al. [8] also make use of a DTW algorithm in order to classify killer whale vocalizations. The DTW algorithm will be elaborated on in the following subsection.

Research into the characteristics of vocalizations from sperm whales has been done by Lohrasbipeydeh et al. [9]. In their research they discuss the characteristics of the "click" audio signals made by these whales. The motivation of their research is to design a "click energy based" detector.

In a paper by Esfahanian et al. [10], the authors compare two detection methods for the north Atlantic right whale. This whale species makes what the researchers refer to

as "upcalls". The researchers extract time-frequency features from spectrograms, this is compared to the other method which involves a local binary pattern (LBP) operator. The researchers concluded that using LBP features was a more accurate method of detection.

Additionally another paper has been published pertaining to north Atlantic right whale acoustic signal processing, This being authored by Dugan et al. [11]. In this paper the researchers compare different machine learning recognition algorithms. They present two new approaches, one based on artificial neural networks and the other on classification and regression tree classifiers. These presented algorithms are compared with with previous research done by Urazghildiiev et al. [12] which they have referenced in their paper. This research discusses the use a of multi-stage feature vector testing algorithm.

A paper authored by David Mellinger and Christopher Clark [13] describes the use of spectrogram correlation to recognize bowhead whale sounds. This is compared to three other methods, these methods being matched filters, neural networks, and hidden Markov models. The researchers found their method to have a high success rate and would have the potential to be particularly useful for detecting a vocalization when few instances of said vocalization are known. They note "few instances" to mean between 5 and 200 instances. In their paper David Mellinger and Christopher Clark [13] cited a particular source of interest being Stafford et al. [14] for their work done on detecting blue whale vocalizations. Stafford et al. [14] used the aforementioned matched filters to detect vocalizations.

In the subsequent subsections two algorithms are discussed and compared for the purposes of this thesis. It has been shown that there are multiple methods that can be used though for the purposes of this thesis DTW and spectrogram correlation are chosen to be investigated further. DTW has already proven to be effective on the calls being analyzed, as shown by Ogundile et al. [7] and spectrogram correlation may prove to be useful for the relatively few Bryde's whale short pulse call signals available at this time.

2.3.2 Dynamic time warping

This technique is excellent for determining the similarity between signals by comparing the shape of the signal curves which may vary in length. DTW was implemented in the early days of speech recognition development. Noticeably it has been used by Brown et al. [8] and more recently by Ogundile et al. [7] as mentioned in the previous section.

There are different methods of implementing the DTW algorithm however it is stated here that the four that will be discussed all start with the same step of determining the two-dimensional difference matrix, designated as D. The difference matrix is populated by aligning one of the chosen sequences, referred to as A, being compared along the vertical of the matrix and the other sequence, referred to as B, along the horizontal of the matrix and populating the values of the matrix with

$$D[i, j] = |A[i] - B[j]|. \tag{2.34}$$

From the point of having established a difference matrix, a cost matrix designated as M can be calculated. The calculation of this matrix varies depending on the method being implemented. However its function remains the same, it contains a continuously

running tally of differences between the sequences which accumulate towards the end of the two-dimensional cost matrix.

Once the cost matrix has been calculated using one of the above methods, the minimum path through the matrix from the final value to the initial value can be traced by declaring the final value as the current value then designating the immediate minimum previous value as the new current value and iterating through this process until the initial value is the current value. The path between these values made up of the selected values is the "minimum path" [8] [15].

The final value in the cost matrix, normalized by dividing by the length of the query, is considered to be the final "dissimilarity". The shorter sequence is dubbed the query by Brown et al. [8] and has been designated as A in Equation (2.34).

The names used for referring to the different methods of implementing the DTW algorithm appeared in the paper published by Brown et al. [8] mentioned previously. Upon inspection, it would appear that they have named these methods after the researchers who they accredit each method to and as such the same names have been used in this document. In addition to this note, the papers referenced by Brown et al. [8] in their article are referenced in each method subsection.

2.3.2.1 Ellis method

This method is considered to be the simplest of methods that are discussed. It involves populating the cost matrix by summing the difference matrix value in the given position with the minimum value of the three previously calculated cost matrix values, the previous horizontal value, the previous vertical value, and the previous diagonal value. The equation for this process as noted by Brown et al. [8] and Ellis [16] is shown as

$$M[i,j] = min \begin{pmatrix} M[i-1,j-1] \\ M[i-1,j] \\ M[i,j-1] \end{pmatrix} + D[i,j] \tag{2.35}$$

and a variation on this equation used by Ogundile et al. [7] is given as

$$D[i,j] = min \begin{pmatrix} D[i-1,j-1] \\ D[i-1,j] \\ D[i,j-1] \end{pmatrix} + |S1_i - S2_j|. \tag{2.36}$$

The difference in these equations is that Ogundile et al. have factored in the calculation of the difference matrix into the calculations for the cost matrix at any given point and additionally have labeled the cost matrix previously known as M, as D. It is presumed that this is because there has been no difference matrix referenced and thus the designation D is still available.

2.3.2.2 Sakoe-Chiba method

Hiroaki Sakoe and Seibi Chiba published a paper in 1978 which detailed dynamic programming algorithm optimization for spoken word recognition [17]. This method deviates from the more simple Ellis method by summing multiple previous values of the difference matrix that range further from the matrix position in question. These values are then weighted differently and included in the minimum function [8]. This is written mathematically as

$$M[i,j] = min \begin{pmatrix} M[i-1,j-1] + 2D[i,j] \\ M[i-2,j-1] + 2D[i-1,j] + D[i,j] \\ M[i-1,j-2] + 2D[i,j-1] + D[i,j] \end{pmatrix}. \tag{2.37}$$

2.3.2.3 Itakura method

This method used by Fumitada Itakura in 1975 in a paper that is based on the minimum prediction residual principle applied to speech recognition [18] was later noted and described by Brown et al. [8]. The variation from the Ellis method, apart from the numerical differences in the equation as seen below, is that there is a constraint put in place that must be observed. This constraint states that two elements cannot be selected sequentially from the same row in the matrix.

Brown et al. [8] present the example that if M[i, j - 1] were to be the minimum value and thus chosen then according to the constraint this would not be an option for the next value of the cost matrix in that given row. This equation is written as

$$M[i,j] = min \begin{pmatrix} M[i-2,j-1] \\ M[i-1,j-1] \\ M[i,j-1] \end{pmatrix} + D[i,j]. \tag{2.38}$$

2.3.2.4 Chai-Vercoe method

In 2003 Wei Chai and Barry Vercoe implemented this method when performing structural analysis of musical signals [19]. This was then later interpreted by Brown et al. [8] to result in Equation (2.39) shown below. In their paper Brown et al. [8] state that this method proved to be "extremely successful" when classifying killer whale calls in Marineland. Equation (2.39) is written as

$$M[i,j] = min \begin{pmatrix} M[i-1,j] + a \\ M[i,j-1] + b \\ M[i-1,j-1] + D[i,j] \end{pmatrix}. \tag{2.39}$$

They note the variable a to be the cost of an insertion and the variable b to be a cost of deletion. The value of b is chosen to be zero, as they state that a deletion results in a difference of length which should not be penalized. The value of a remains an adjustable parameter [8].

2.3.3 Spectrogram correlation

The method outlined in the paper by David Mellinger and Christopher Clark [13] involves converting the recorded signal into a spectrogram after which the spectrogram is cross-correlated with what they refer to as the "kernel". The kernel represents the signal of interest. The result of this cross-correlation process is a series of recognition values in the time domain. This series represents the "closeness" of the match between the recorded signal and the kernel at each time increment of the spectrogram. If a value in this resultant time series is larger than others it represents a closer match to the desired signal.

A good visual representation of this process described has been shown in a separate paper published by David Mellinger [20] in 2004. Figure 2.5 below is an adapted and simplified version of the one used in his paper.

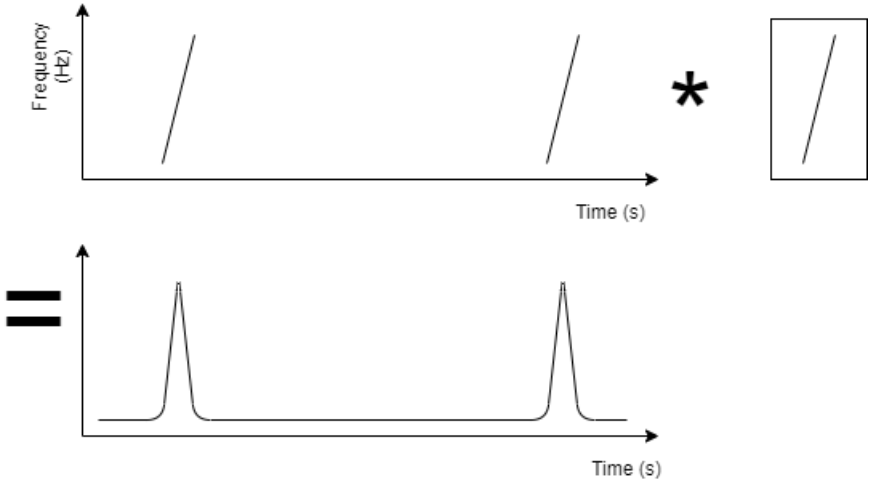

Figure 2.5: Spectrogram cross-correlation [20]

Figure 2.5 shows the recorded signal, that has been normalized previously to the researcher's standard. The desired signal represented by the kernel can be seen to be a linear increase in signal frequency over time. The resultant series shows multiple spikes and as such would imply multiple successful detections given that these spikes are over the predetermined detection threshold.

To construct a kernel, Mellinger et al. [13] implement a process of measuring the time and frequency characteristics for each sample vocalization's sections. They describe setting the start time of the vocalization to zero after which, the times and frequencies of each the vocalizations section's endpoints are measured. These measurements are then averaged between all the samples to obtain what would be the characteristics for an "average call". This is then used to create the kernel.

2.3.4 Dynamic time warping and spectrogram correlation comparison

For the proposed system in question, either detection approach would seem to be viable. However, upon investigation, it was found that while the time domain of the short pulse call has a distinct waveform, the spectrogram of the desired short pulse call does not have very distinct features. The statement that the spectrogram seen below does not show distinct features is affirmed when comparing the short pulse call spectrogram in Figure 2.6 with the spectrogram published by David Mellinger, shown in Figure 2.5.

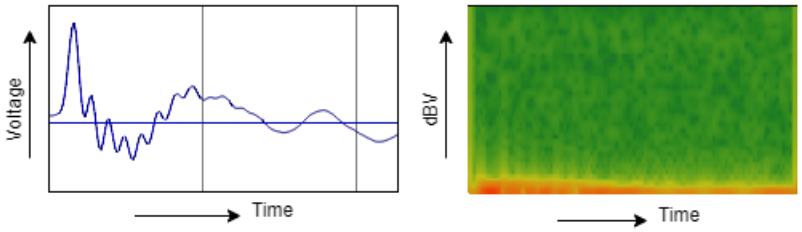

Figure 2.6: Time domain and spectrogram of a short pulse call signal

This creates a situation where utilizing the DTW method seems more applicable to this signal. In addition to this, as previously stated, the use of DTW algorithms to detect Bryde's whale short pulse calls has already been implemented and showed to be effective by Ogundile et al. As such the DTW approach has been chosen to be implemented as opposed to using spectrogram correlation.

2.4 Finding differences in arrival times

2.4.1 Cross-correlation

Muanke et al. [2] discuss a method of detecting signals and determining differences in arrival times. A Hilbert envelope technique is implemented by the researchers in tandem with a cross-correlation approach. The cross-correlation of the audio signals with one another yielded oscillating resultant functions. As such a Hilbert envelope is applied to the resultant functions. The Hilbert envelope of the function is defined as the magnitude of what Muanke et al. refer to as the "analytic signal of the cross-correlation function" [2]. The analytic signal is given the designation of $S(t)$ and is defined by the equation,

$$S(t) = s(t) + j\check{s}(t). \tag{2.40}$$

Where $s(t)$ is the cross-correlation function in question and $\check{s}(t)$ is the Hilbert transform of said equation. The equation for performing a Hilbert transform is given as

$$\check{s}(t) = H\{s(t)\} = \frac{1}{\pi}\int_{-\infty}^{+\infty}\frac{s(\rho)}{\rho - t}d\rho = h(t) \star s(t) \tag{2.41}$$

where the Hilbert kernel is denoted by $h(t) = \frac{-1}{\pi t}$.

The Hilbert envelope function is given the designation $E(t)$ and as previously stated, is equal to the magnitude of the analytic signal. This is written as

$$E(t) = \sqrt{(s(t))^2 + (\check{s}(t))^2}. \tag{2.42}$$

The result is shown visually by Muanke et al. [2]. An adaptation of the result can be seen in Figure 2.7. This figure shows a fabricated signal that would represent the cross-correlation between the signal from sensor 1 and the signals from the other three sensors along with the resultant Hilbert envelopes. The researchers note that the correct time delay can be found from the peak of the envelope, as circled in the figure, regardless of if the peak of the envelope corresponds to the peak of the cross-correlation function.

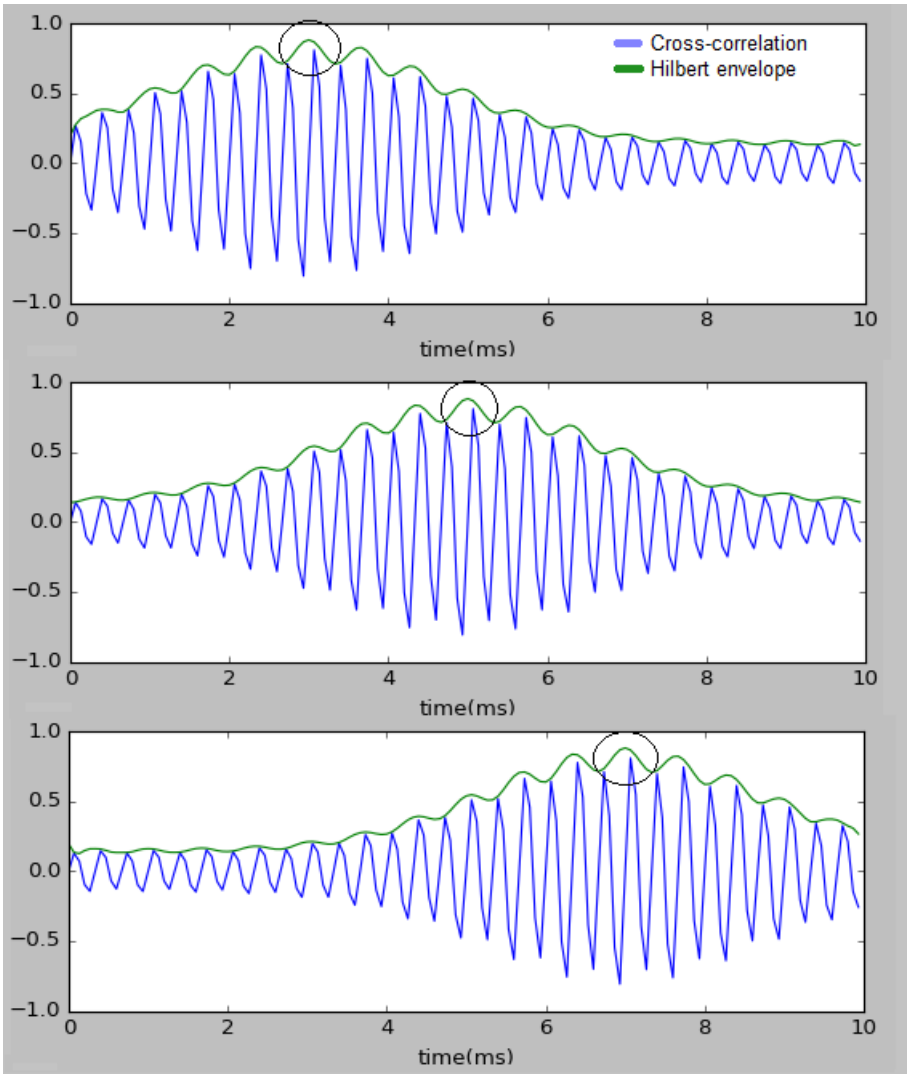

Figure 2.7: Hilbert envelope for the cross-correlation of sensor pairs (2;1), (3;1) and (4;1) [2]

Though it has already been stated that the cross-correlation signal used in Figure 2.7 is fabricated, for the sake of rigor it is noted that the figure presented by the researchers extends into the negative time-domain however the purpose of Figure 2.7 is to illustrate

the process implemented rather than to portray meaningful data and as such, there is no detriment to only representing data in the positive time domain.

2.4.2 GPS Time synchronisation

This process of determining the differences in arrival times involves synchronising all the sensors' clocks to a common time such that the timestamps are also synchronized. This process is not incredibly difficult when working in the timing ranges greater than sub-seconds. However timing the clocks to sub-second times is incredibly important for this system, as it is known that a 1 second time discrepancy of sound in water causes an approximate time discrepancy of 1.5 km.

As such the sensors need to be synchronized to within milliseconds for acceptably accurate results. In a paper published by Li et al. [21], the authors give a general overview of the challenges that arise in using narrow-band signals for localization. One such challenge was identified to be the synchronization. They note that using the internet domain (Network Time Protocol) can provide accuracy only in the millisecond range. These researchers utilize GPS signals for synchronization. They claim that this has "high potential to satisfy the needs of TDOA".

The researchers attach their units with GPS receivers that give off a satellite's pulse per second signal. The pulse per second signal typically has a rising edge aligned with the GPS second and can be used to synchronize the unit clocks. The pulse per second signal is used as an "external reference clock" and proves long term stability. In the findings by Li et al. [21] it is stated that if the external GPS-based clock is implemented there is no long-term clock drift between devices.

2.4.3 Cross-correlation and GPS time synchronisation comparison

Though both methods of finding differences in arrival times have their advantages, there appears to be an inherent problem with implementing cross-correlations between the signals in the system. This being that the sensors are not designed to have the ability to continuously transmit the signal information to a common machine where the cross-correlation of the signals can be performed.

One may argue situations in which signal information at each sensor can be transmitted continuously. As such the user unit on the research vessel would collect live signal data and perform the calculations locally. However, the problem with this argument is that the signals from each sensor would presumably interfere with one another if transmitted simultaneously. To prevent this interference it would make more sense to collect data and stagger the transition times of each sensor.

This dilemma can be overcome by allowing the sensors to independently detect signals locally and transmit accurate timestamps. Through the use of GPS time synchronization, the independent clocks can be synchronized to within milliseconds. The difference in these timestamps would intuitively, with an ideal detection system, represent the differences in arrival times. The detection system is of course not ideal, though there are many non-idealities that need to be overcome already present in the system according to its nature.

2.5 Conclusion

In conclusion, it was determined that moving forward with the development of the system that TDOA would be implemented over DOAE. Additionally, DTW and GPS time synchronization would both be implemented over their counterparts. The principles behind these chosen topics are discussed in depth in the following chapter.

When operating at sea the advantages of TDOA over DOAE become apparent. A DOAE system would allow for a reference line at which the whale would be located. However this line is orientated at the constantly moving sensors and when operating on a constantly moving sea vessel the ambiguity of simply having a direction, that is not referenced to the research vessel location, cannot compare to having a point of reference for the location of the whale.

DTW has been shown to be effective in the detection of Bryde's whale short pulse calls, as discussed previously. In addition to this, a GPS time synchronized system would prove to be effective in that the sensors already make use of GPS devices for tracking purposes. The time synchronization method also allows for the sensors to do local processing thus there would be no need for the live streaming for sensor data to be processed at the user-unit. The user-unit would only require the transmission of data at designated time periods.

Chapter 3

System overview and theoretical background

3.1 Introduction

This chapter serves the purpose of discussing in depth the principles of the methods that have been chosen to be implemented, as well as disusing the design of the system conceptually.

As stated in the previous chapter TDOA is implemented as the localization system. DTW is implemented in the audio signal detector. GPS time synchronization is used to synchronize the system clocks of the units and as such the units will yield synchronized timestamps for instances of detection.

In addition to these methods described in the previous chapter, GPS modules are used for the tracking of the units and LoRa is implemented as a communication system between the sensors and the user-unit.

3.2 System design overview

The system for calculating the Bryde's whale position as a whole will comprise of many subsystems. Some of these systems work in unison while others are separate, however they all yield sets of information or perform functions which are required in order for the overall system to function.

The overall system design resembles the diagrams seen in Figure 3.1 and Figure 3.2. These diagrams show the many subsystems implemented and how these subsystems interact with one another. The overall system has been separated into two diagrams, this is because the system consists of multiple sensors as seen in Figure 3.1 and a single user unit as seen in Figure 3.2. They have separate functions and thus warrant separate diagrams. It is noted that there are three sensors implemented into the system design in order for the time difference of arrival equation variables to be satisfied however Figure 3.1 only shows one of these sensors as they are designed to be identical.

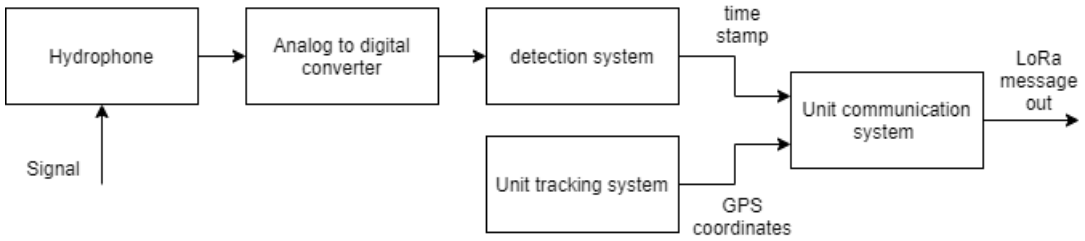

Figure 3.1: Diagram of the sensor system

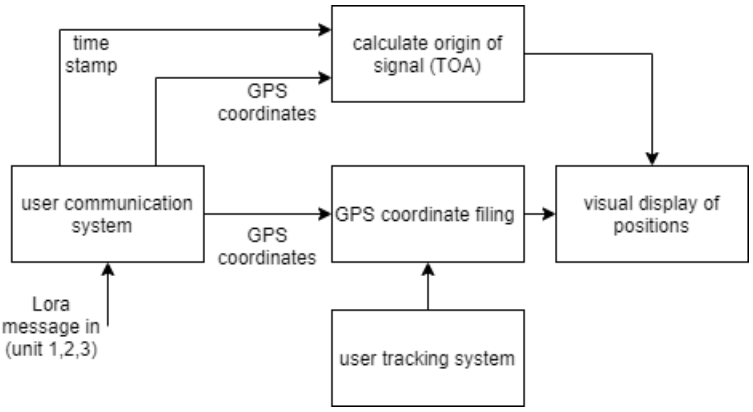

Figure 3.2: Diagram of the user-unit system

3.3 The time difference of arrival localization process

The majority of the information covered in this section and the subsequent subsections comes from a handout given to the students of New Mexico Tech for a module in 2008 [22]. This document proved to be extremely valuable for the process of understanding TDOA as it covers many of the theoretical aspects as well as MATLAB exercises for understanding the implementation of the mathematical concepts.

3.3.1 Equations of motion

A system that measures the time of arrival of a signal at different positions in space can be used to accurately calculate the signal's location, or the location of origin of the signal, in two or three-dimensional space. A system such as this makes use of time of arrival location principles, known as a TOA system or alternatively a TDOA system depending on what approach has been taken.

The core principle utilized in TOA systems and by extension, TDOA systems, is fairly simple in essence. It can be seen in Equation (3.1). This is known to be a simple equation of motion that most first see in secondary schooling,

$$d = v.t. \tag{3.1}$$

However one must consider the equation at multiple instances. Thus it would be more accurate to express Equation (3.1) as Equation (3.2) as seen below. Where t_1 and d_1 are the time and position at one instance and t_2 and d_2 are the time and position at another instance, respectively. Equation (3.2) is written as,

$$(d_1 - d_2) = v.(t_1 - t_2). \tag{3.2}$$

If one were to calculate the distance between the positions at which the target was observed at instance 1 and 2, one would find that the distance can be calculated by finding the Cartesian points in space of said target at both instances and then applying the Pythagorean theorem. This further manipulates the equation to form

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} = v.(t_1 - t_2) \tag{3.3}$$

for three-dimensional space and

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} = v.(t_1 - t_2) \tag{3.4}$$

for two-dimensional space.

It is by using Equation (3.2) that has further been expressed Equation (3.3) and Equation (3.4) that allows one to approach the process of constructing a TDOA system.

It is noted for the purposes of this thesis that the target examined emits an audio signal. The velocity of sound is known for different mediums. In addition to this known information, the x and y coordinates of the sensor, which is declared as the second instance of observation for the sound wave are also known. This in turn means that when applying Equation (3.4) that there are three unknowns. These being the x and y coordinates of the origin of the audio signal as well as the time taken for the signal to reach the second instance of observation.

In order to solve these three unknowns, three different equations are needed, and therefore in order to determine the origin of the audio signal three measuring stations/sensors are needed. One can make use of graphical representations in order to show the possible positions for the origin of the audio signal given the known and unknown values present. According to the setting up of mathematical formulas which coincide with already established equations that are explored in the following section, knowing the differences in the arrival times or more specifically the difference in distances between an unknown point and two measuring points of known location constrain the unknown point to lie on a hyperbolic curve constructed using the known values. It is because of this that in order to develop the TDOA (time difference of arrival) system one must first have a working understanding of hyperbola theory.

3.3.2 Hyperbola theory

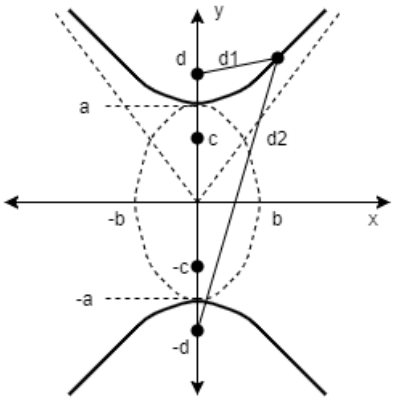

Figure 3.3: Layout and points of importance for a hyperbola [22]

Hyperbolas have the property that the difference between the distances from each focus point to any point upon the hyperbola is constant. This property can be written as a mathematical function,

$$|d_1 - d_2| = 2a \tag{3.5}$$

and can be related back to the points seen in Figure 3.3 above.

One implements the use of an absolute value because the difference in the distances may be negative and the absolute value would rectify this. It can be seen in Figure 3.3 that for two focus points there are also two hyperbolas that satisfy the hyperbolic equation. This second valid hyperbola appears according to the hyperbolic equation, as the squared y variable results in a positive and negative set of y solutions. The hyperbolic equation is written as

$$y^2/a^2 - x^2/b^2 = 1. \tag{3.6}$$

The focus points of the hyperbola can be seen in Figure 3.3 to be at the points [0;d] and [0;-d]. Having established this, it should also be noted the sum of the inverses of the x and y coefficients squared is equal to half the distance between the focus points squared, which is represented by d squared. Thus if one knows the location of the focus points, one would be able to work towards the a and b values. This becomes an important step when applying hyperbola theory to TDOA systems. This is written mathematically as,

$$a^2 + b^2 = d^2. \tag{3.7}$$

3.3.3 Implementing hyperbola theory into a TDOA system

When implementing the hyperbola equations into the TDOA algorithm it is assumed that the time at which the signal reaches the sensor and the sensor's position are both known. When practically implementing the hyperbola theory, the hyperbola focus points represent the sensors' positions. The different time values such as t_1 and t_2 represent the time at which sensor 1 and sensor 2 record detecting the signal.

In this implementation of the t_1 and t_2 values, the individual time values themselves are of less importance for constructing the hyperbolas than the difference between them. This invariably means that the time at which the signal is transmitted versus when it arrives at each station/sensor is of less importance than the difference in time of detection between t_1 and t_2 for the process of constructing the hyperbolas. Using Equation (3.2), which makes use of the previously discussed difference in times of arrival of t_1 and t_2, one is able to determine the difference between d_1 and d_2 by multiplying the t_1 and t_2 values by the velocity of the signal. In this case, the velocity of the signal would be the speed of sound in water which is approximately 1498 m/s.

The fact that d_1 and d_2 are known then means that Equation (3.5) allows for the solving of the a variable. In addition to this, the d variable needs not to be solved as previously stated the position of the stations is known and thus the distance between them is also known. Therefore Equation (3.7) would then be able to yield a value for b and thus the required hyperbolic equation is able to be plotted as all the required variables are known.

Figure 3.4 shows hyperbolas that have been plotted using this information and the process described above as well as fictitious required values. The source of the signal in Figure 3.4 is known to potentially lie at any point on these hyperbolas that have been created.

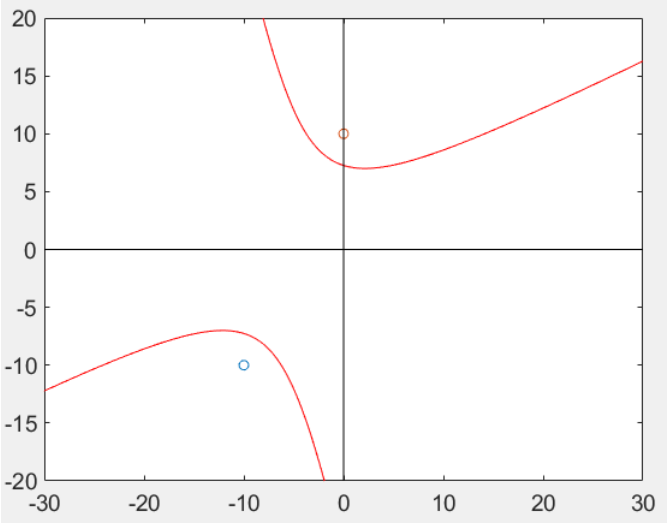

Figure 3.4: Potential positions of signal origin laying on 2 hyperbolas [22]

While having already vastly narrowed down the number of potential positions for the origin of the signal, this process is expanded upon by involving another sensor. This

creates new focus point combinations. The combinations are S1S2, S2S3, and S1S3, where the number following the S denotes the sensor's number.

It is the points where these hyperbolas intersect that result in the calculated position of the signal source. These previously mentioned combinations result in 6 total hyperbolas, as per Figure 3.3 each set of focus points result in 2 hyperbolas. These additional hyperbolas may also intersect, creating another calculated position of the signal, at this point additional information can be implemented to eliminate or limit ambiguity. By using the detection times themselves one can determine the proximity of the signal source to each sensor. Meaning that if sensor A has an earlier timestamp than sensor B, the source of the signal must in turn be closer to sensor A. This information can be used to eliminate position ambiguity and select the correct hyperbola from each set.

An example of the previously explained situations can be seen in the figures that follow. In Figure 3.5 the location of the signal source is at the coordinate (8,8) and in Figure 3.6 the location of the signal source is at the coordinate (-5,-5). Red lines denote the hyperbolas that are produced using positive y values and blue lines denote hyperbolas that are produced using negative y values, as according to the hyperbola equation y is squared it is known that $y^2 = (y)^2$ or $(-y)^2$.

It can be seen in both figures that there are originally 2 possible positions calculated. Though when implementing the difference in timestamps to determine the proximity of the sound source to the sensors and eliminate hyperbolas it can be seen that the ambiguity has been eliminated in both instances.

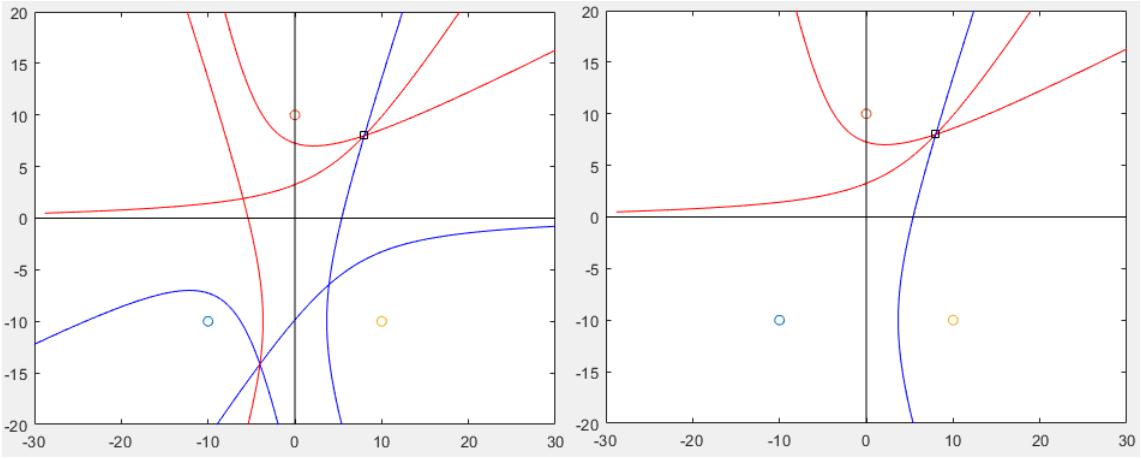

Figure 3.5: Basic TDOA localization vs TDOA localization with eliminated ambiguity for a signal source located at (8;8)

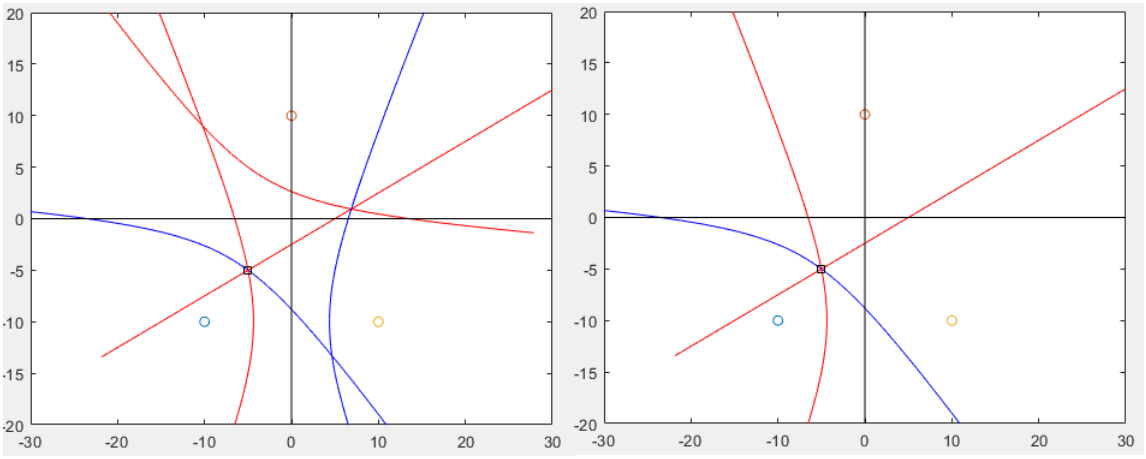

Figure 3.6: Basic TDOA localization vs TDOA localization with eliminated ambiguity for a signal source located at (-5;-5)

3.3.4 TDOA error analysis

In section 3.3.3 the calculations for the location of origin for the signal are in an ideal system. However, when implementing TDOA practically, small errors in various variables interfere with the precision of the location calculations. GPS devices are only guaranteed to give accurate positions within multiple meters. It is a culmination of possible errors be it; inaccurate GPS coordinates, variance in system processing times, or variance in data collection speeds that lead to a culminated error in arrival time or more specifically, detection time. This error in arrival time possibly present in each of the sensors mathematically leads to an error in location calculation. In this section, the effects of arrival time errors are analyzed by intentionally introducing normally distributed errors to the arrival time variables.

The probability distribution for these normally distributed time errors is portrayed as

$$f(x) = \frac{h}{\sqrt{\pi}} e^{-h^2(x-m)^2}. \tag{3.8}$$

This equation can be expanded upon by introducing

$$\bar{x} = \int_{-\infty}^{\infty} x f(x) dx \tag{3.9}$$

as the mean of a normal distribution function, with

$$\sigma^2 = \int_{-\infty}^{\infty} (x - \bar{x})^2 f(x) dx \tag{3.10}$$

and

$$\sigma = \frac{1}{\sqrt{2}h} \tag{3.11}$$

as the variance of the function. By implementing these equations, Equation (3.8) is now presented as

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\bar{x})^2/2\sigma^2}. \tag{3.12}$$

The graph for the Gaussian probability distribution that has been described mathematically in Equation (3.12) is shown as Figure 3.7. For this figure a mean of 5 and a variance of 1 is chosen.

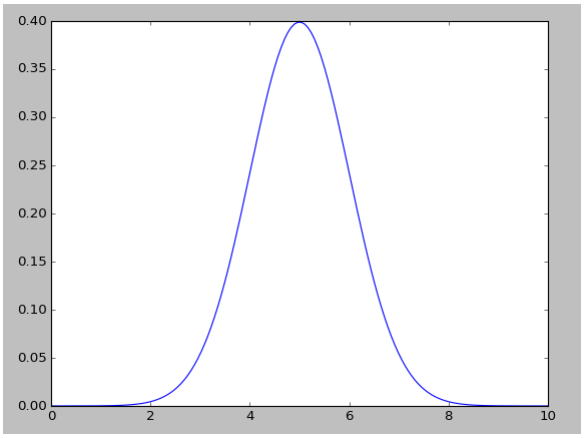

Figure 3.7: Gaussian probability distribution [22]

It is by applying these normally distributed errors into the calculations that one can start to observe the warping of the previously ideal hyperbolas. Figure 3.8 shows the same scenario as Figure 3.5. However there have been various errors introduced through multiple iterations. These errors are presented by multiple different variances introduced to the normal distribution equation. The variances range from 0.01 to 0.2 in steps of 0.02.

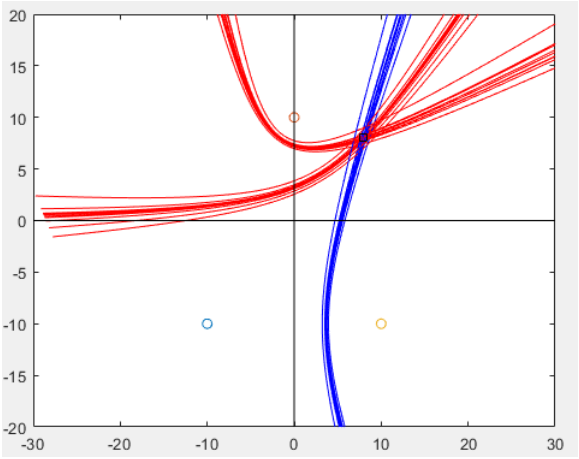

Figure 3.8: TDOA localization for a signal source located at (8;8) with introduced errors [22]

It can be seen from Figure 3.8 that the system has lost a great deal of accuracy however the overall general position managed to stay true. After this analysis has been performed one may be more inclined to find desired locations graphically rather than mathematically as a person can still see where the curves intersect to some accuracy however the lines may not actually intersect at the same location because of the margin of error.

Error analysis is performed further in Chapter 5 with genuine data when testing the system. Though the values and results in this section are still fictitious, they serve the purpose of introducing the concept of timing errors introduced to the system and how they may be approached.

3.4 The Dynamic time warping detection algorithm

The dynamic time warping algorithm has been referenced in many research papers since its inception, ranging from audio analysis to medical research. As mentioned in Chapter 2, it has noticeably been implemented by Ogundile et al. in a paper published in 2020 [7]. These researchers made use of the algorithm to detect Bryde's whale short pulse calls.

The DTW algorithm is implemented to find the best alignment of two independent sequences and thus is used in determining the similarity between the sequences. When DTW distance is compared to something such as Euclidean distance the usefulness of the algorithm becomes apparent [23][15].

In Chapter 2 it was noted that there are multiple methods for implementing the DTW and calculating the cost matrix. It is noted here that the Ellis method is followed for this thesis. In addition to this, it is noted that the DTW equation discussed further follows the format as it was written in the Ogundile et al. paper [7]. This is seen as Equation (2.36) and Equation (3.13) as opposed to Equation (2.35). Another choice made for this thesis is that the final "dissimilarity" is not the final value in the cost matrix normalized, it is just the final value without the normalization [8]. When developing the algorithm which is discussed in Chapter 4 it is noted that the DTW is performed on 1000 sample frames and thus all DTW matrix lengths are the same resulting in no need for normalization. As such it has been omitted from the theory that follows.

3.4.1 The Dynamic time warping process

When comparing the two time sequences using DTW the first step of the process is to receive all the points of information that will be used as input data for the algorithm. The number of observations in two time sequences need not be the same and the number of points is only limited by computational power. The two time sequences, dubbed time sequence 1 and time sequence 2, are used to populate a 2-dimensional matrix, this matrix being of size (x,y) where x is the number of observations in time sequence 1 and y is the number of observations in time sequence 2.

The value at each point in the 2-dimensional matrix is given by

$$D[i,j] = min \begin{pmatrix} D[i-1,j-1] \\ D[i-1,j] \\ D[i,j-1]. \end{pmatrix} + |S1_i - S2_j|. \tag{3.13}$$

Where D represents the value at a specified position in the matrix. S1 and S2 represent the sequence 1 and sequence 2 observation values at specified points, respectively [15][23].

3.4.2 Dynamic time warping illustration

3.4.2.1 Simple Illustration of performing the DTW algorithm on same signal shifted in the time domain

For the sake of simplicity, a non-complex triangular waveform with an amplitude of three units has been selected for this example. The example sequence can be seen below in Figure 3.9, it represents the sampled values of the triangular waveform.

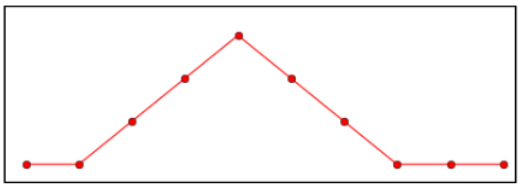

Figure 3.9: Example sequence

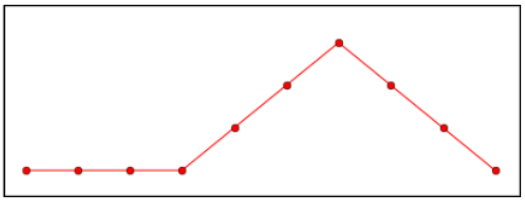

Figure 3.10: Example sequence, delayed by 2 samples

This sequence seen in Figure 3.9 can be delayed by 2 sample points, as seen in Figure 3.10, and then can be analyzed to determine the similarity between the two signals using DTW. If one were to examine these two sequences, using common sense one may be able to determine that they are the same triangular waveform that has been shifted. However if one applied the use of standard Euclidean distances to determine the similarity of the signals, the result would be that they are not similar at all, let alone the same signal that has been delayed in one instance. This can be seen in Figure 3.11.

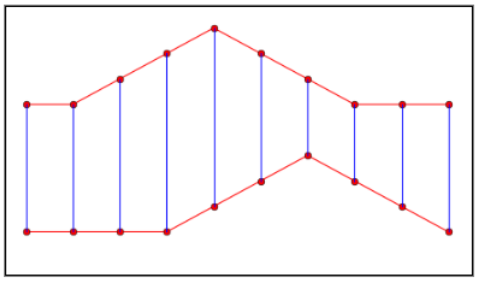

Figure 3.11: Euclidean matched sequences

One can find the differences between the sequences' values over all the points that have been matched and sum these differences to determine the similarity of the sequences. If one were to do this using Euclidean distances they would conclude that the sequences have a total difference of 10 units. This implies that the sequences being compared are not considered to be similar at all, however, one can see visually that this is not true. If the same process of subtracting the matched values from one another and then adding the differences is applied to the DTW matching of the sequences, it can be seen that because the total difference is 0 that the sequences are not only similar but represent the same signal that has been shifted in the time domain.

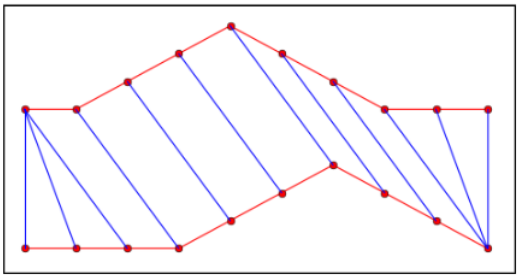

Figure 3.12: DTW matched sequences

To determine which observations of each sequence are matched in the DTW algorithm, one needs to make use of a 2-dimensional matrix. This matrix is populated by implementing Equation (3.13), as seen previously in section 3.4.1. The matrix generated by this process for these example sequences can be seen in Figure 3.13 below.

9	9	9	9	7	8	9	7	3	0
9	9	9	9	6	6	7	5	2	0
9	9	9	9	5	4	5	3	1	0
9	9	9	9	4	2	3	1	0	1
8	8	8	8	4	1	1	0	1	3
6	6	6	6	3	1	0	1	3	5
3	3	3	3	1	0	1	1	2	4
1	1	1	1	0	1	3	4	4	5
0	0	0	0	1	3	6	6	7	7
0	0	0	0	1	3	6	8	9	9

Figure 3.13: 2 dimensional DTW matrix populated using equation 1

From this matrix, the matched observations can then be determined by finding a path from D(10,10) to D(0,0). This path is determined by starting at the final instance in the matrix then proceeding to find the minimum of the values for the instances directly previous to the current instance and then setting this minimum value as the new current instance. This process is repeated until the initial instance is reached. A path for this matrix shown in Figure 3.13 can be seen in Figure 3.14 [15].

9	9	9	9	7	8	9	7	3	0
9	9	9	9	6	6	7	5	2	0
9	9	9	9	5	4	5	3	1	0
9	9	9	9	4	2	3	1	0	1
8	8	8	8	4	1	1	0	1	3
6	6	6	6	3	1	0	1	3	5
3	3	3	3	1	0	1	1	2	4
1	1	1	1	0	1	3	4	4	5
0	0	0	0	1	3	6	6	7	7
0	0	0	0	1	3	6	8	9	9

Figure 3.14: Path through the 2 dimensional DTW matrix

The selected path is now known and can be used to visually match observations in the sequences as seen in Figure 3.12. However it should be known that this visual matching as seen in Figure 3.12 is only used as a visual representation of the data stored in the matrix and while the matrix remains incredibly important, for this thesis the visual representation is shown only to illustrate the DTW process. While also true that the matrix as a whole is important for many applications of the DTW algorithm, for the purposes of this thesis only the final value, which for this example is D(10,10), is taken into account when determining if signals are similar. This final value shows the overall similarity as seen illustrated previously.

3.4.2.2 More complex Illustration comparing the DTW algorithm output of multiple sequences to determine comparative similarity

In the previous illustration, it was seen that if a signal is shifted in the time domain and does not lose any of its information then the DTW of that signal and its time-shifted version would show a difference of 0, meaning that they are identical signals. However, this alone is not of much use for identifying signals by analyzing them with the DTW algorithm. To do this, the system needs to be calibrated with template signals.

The logic that follows is that if two signals can be seen to be different but are still known to be from the same source, then if a signal from an unknown source is more similar to those signals from the known source than they are to each other, the unknown signal must too be from the same source. This principle can be illustrated below using signals of varying amplitude.

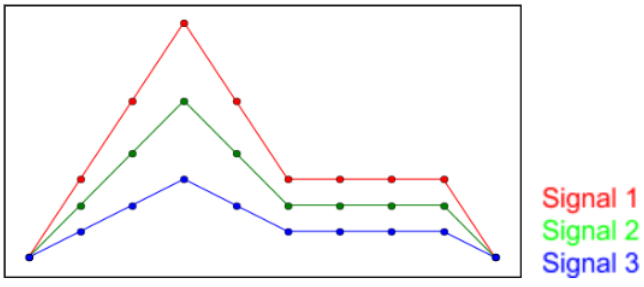

Figure 3.15: Sequences representing sampled signals of varying amplitudes

It can be seen above that the sequences that represent the sampled signals 1,2 and 3 all have the same waveform with the only difference being a variation in amplitude between the sampled signals. The sequences are all aligned with one another as it has already been shown previously that signals that have been shifted in time are still identical to their previous version, under the condition that no information has been lost.

To reemphasize what has previously been stated, if signal 1 and signal 3 are both known to be from the same source then using the DTW algorithm to determine the similarity between sequences it can be logically assumed that signal 2 is from the same source if it is more similar to one or more known source signals than the other known source signal is.

36	29	23	23	17	16	17	18	19	19
36	28	21	20	15	17	19	21	23	26
33	26	20	14	15	17	19	21	23	26
30	24	19	14	15	17	19	21	23	26
27	22	18	14	15	17	19	21	23	26
24	20	17	14	15	17	19	21	23	26
18	15	13	11	12	14	16	18	20	23
9	7	6	5	6	8	10	12	14	17
3	2	2	2	3	5	7	9	11	14
0	1	3	6	8	9	10	11	12	12

Figure 3.16: DTW matrix of signal 1 and signal 3 amplitudes

If one were to perform the DTW algorithm on signal 1 and signal 3 to determine their similarity, one can see from Figure 3.16 representing the 2 dimensional DTW matrix above that they have a total DTW difference of 19. This value can now be compared to the DTW matrices of signal 1 and signal 2, and signal 2 and signal 3, to see if the similarity of these signals is less or greater than 19.

12	8	12	14	16	12	12	12	12	10
12	6	8	12	14	10	10	10	10	10
11	5	7	11	13	9	9	9	9	9
10	4	6	10	12	8	8	8	8	8
9	3	5	9	9	7	7	7	7	7
8	2	4	6	7	6	6	6	6	8
6	2	2	5	6	7	8	9	10	12
3	1	3	7	9	9	9	9	9	11
1	1	4	9	12	13	14	15	16	17
0	2	6	12	16	18	20	22	24	24

Figure 3.17: DTW matrix of signal 3 and signal 2 amplitudes

36	22	18	20	13	11	11	11	11	10
36	20	14	16	9	9	9	9	10	13
33	19	13	14	8	8	8	9	10	13
30	18	12	11	7	7	8	9	10	13
17	17	11	8	6	7	8	9	10	13
24	16	10	5	7	11	15	19	23	32
18	12	8	5	7	11	15	19	23	32
9	5	3	2	4	8	12	16	20	26
3	1	2	5	6	7	8	9	10	13
0	2	6	12	16	18	20	22	24	24

Figure 3.18: DTW matrix of signal 1 and signal 2 amplitudes

It can be seen from the matrices above that the similarity of signals 1 and 2, and signals 2 and 3 is 10 in both cases. This would imply that signal 1 is more similar to signal 2 than it is to signal 3 and signal 3 is also more similar to signal 2 than it is to signal 1. From this information, it can be assumed the signal 2 is also from the same source as the other signals. This approach to identifying signals based on calibration with known signals forms the basis for the detection algorithm constructed and implemented in this thesis. More on the implementation of this detector can be found in Chapter 4.

3.5 LoRa technology

LoRa (Long Range) is known as a low-power wide-area network (LPWAN) technology. This technology allows for wide-area connectivity up to tens of kilometers for low data rates and low power consumption. LoRa usually uses a band of 125 kHz or more to broadcast a signal. This wideband is beneficial as it allows for more resistance to channel noise, Doppler effects, and fading [24].

LoRa technology is based on chirp spread spectrum modulation (CSS). This is a spread spectrum form of modulation that uses frequency modulated chirp pulses to encode the information being transmitted. The spread spectrum modulation technique involves varying a carrier signal in the frequency domain. A chirp signal is a signal that has a frequency increase, known as an up-chirp, or a frequency decrease, known as a down-chirp [25]. In Figure 3.19 one can see a visual representation of an up-chirp signal.

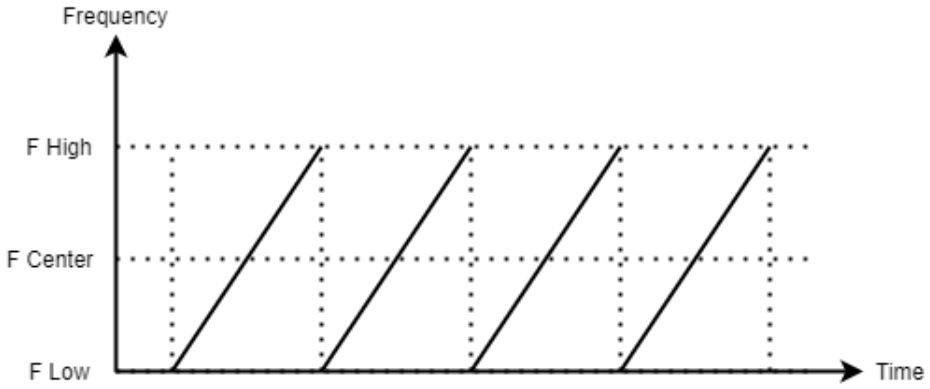

Figure 3.19: Up-chirp signal in the frequency domain [25]

It is seen that the frequency of the signal increases from the lowest point to the highest point and then returns to the lowest point repeatedly. This is reversed for down-chirp signals. The chirp signals are used as the carrier signals for the message being transmitted. This creates a transmitted signal that is resistant to multipath fading and Doppler shifts [24].

3.5.1 The code rate of LoRa messages

LoRa technology makes use of forward error correction for increased reliability in the receiver, the amount of forward error correction is determined by the code rate employed. The code rate can vary between 0 and 4. Where a code rate of 0 means that forward error correction has not been utilized. It is logically determined that if an encoder generates n output bits and a value of k bits are known to represent useful information then (n-k) bits are determined as "redundant bits". However these "redundant bits" allow for the receiver to find and correct errors in the sent message [24]. While beneficial in this aspect these bits also make the message data rates less effective. Thus as the code rate and therefore reliability increases the effectiveness/rapidity of the message data rate decreases.

3.5.2 The spreading factor of LoRa messages

The spreading factor (SF) used in LoRa messages can range between values of 4 and 7. Data rate and spreading factors are considered to be inversely proportional. The reason for using a higher spreading factor, thus lowering the data rate, is that higher spreading factors allow for an increase in the range for transmission [24].

When sending a symbol, the spreading factor represents the number of bits in the symbol that are encoded. A simple example is to say that the number 95 can be written in binary as '1011111'. This data is the called symbol. The spreading factor is the number of bits required to write the binary of the symbol. Thus in this example, the spreading factor would be 7. It is known for binary logic that an unknown symbol represented by SF bits

could represent one of 2^{SF} possible values. Thus the symbol in this example could have 128 possible values, ranging from 0 to 127 if it is comprised of 7 bits [25].

To transmit the symbol its value is encoded onto an up-chirp signal. The up-chirp signal is divided into 2^{SF} 'chips'. It is now established for this simplified explanation that the symbol has a value of 95, there are 7 bits and 128 chips. The way that these chips are ordered represents the symbol value [25]. The chip orientation for the symbol value, as well as other possible symbol values, can be seen in Figure 3.20. It is important to note that while the reality is far more complex, this simplified example is put in place to illustrate the spreading factor concept.

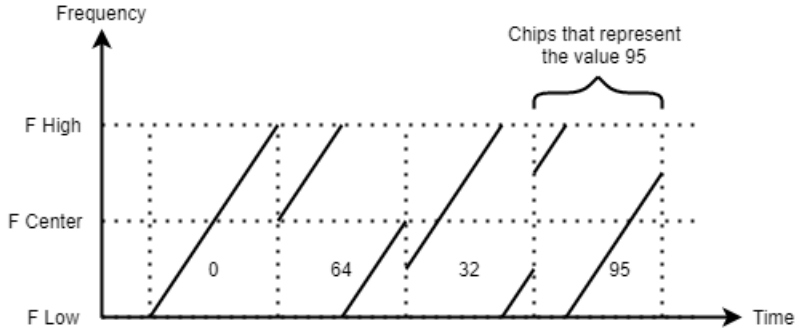

Figure 3.20: Example chip orientation representing a value [25]

3.5.3 The packet structure of a LoRa message

The packet structure can be seen in Figure 3.21 below. It can be seen from Figure 3.21 that there are 4 main fields that the packet is separated into. These are; the preamble field, the header field, the payload field, and the cyclic redundancy check field which is optional [24].

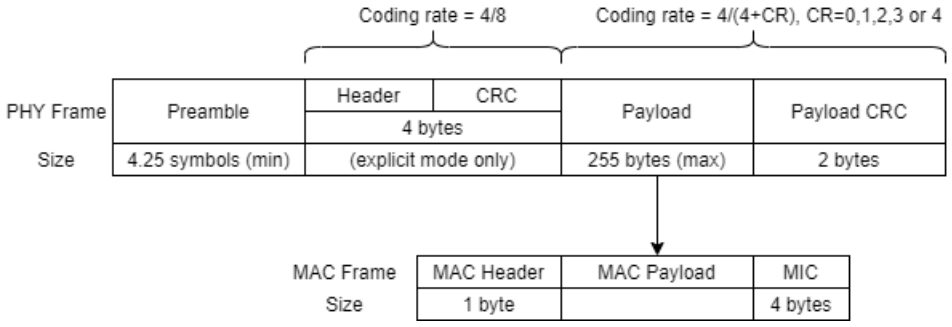

Figure 3.21: LoRa packet structure [24]

According to Noreen et al. [24] the preamble field is used to synchronize the receiver with the incoming data flow. The header field's purpose is determined by one of two available

operation modes. In the first mode, the number of bytes in the header field shows the forward error correction code rate, the payload length, and also whether there is a cyclic redundancy check present. In the second mode, it is specified that both code rate and payload are fixed. This mode excludes the header field and thus is transmitted faster.

Additionally, Noreen et al. [24] state that the payload field can be up to 255 bytes in length and is broken up into 3 further fields. The media access control (MAC) header field defines the frame's type, protocol version, and direction. The MAC payload field contains the data being sent. The message integrity code (MIC) field contains the digital signature of the payload. The cyclic redundancy check (CRC) field, which is noted to be optional, is used to protect the payload from errors. The header field also contains its own cyclic redundancy check (CRC) section, this allows the receiver to discard a packet with an invalid header field.

3.6 GPS functional principles

GPS (Global Positioning System) technology falls under the category of what is known as GNSS (Global Navigation Satellite System) technology. It was developed by the Department of Defense in the United States of America and is now readily available for civilian use. GPS technology is designed and implemented such that at any point in the world, at any given moment a GPS module has a direct line of sight with a minimum of four GPS satellites in orbit around the Earth [26]. The combination of the data from these multiple satellites is what allows one to locate their position. Many principles govern the functionality of GPS technology, these are to be separately discussed in the subsections that follow.

3.6.1 Trilateration

To calculate unknown positions of objects one uses the distance between objects. For the sake of simplicity, the concepts of trilateration can be explained in two dimensions and then expanded further into 3 dimensions. When measuring range, or distance from an object to another potential object in two-dimensional space one would work with circular distances, while one of the objects resides in the center of the circle [26]. An example to illustrate the meaning of this statement is if object A and object B are two units of distance apart, and the location of object A is known but the location of object B is unknown. It can be seen in Figure 3.22 that object B can potentially be at any point on the circle with object A at the center and a radius of two units. Note: for GPS "objects" of known location would be GPS satellites and object B would be a GPS receiver.

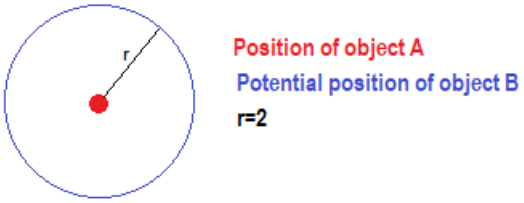

Figure 3.22: The potential position of object B in relation to object A

If this is then expanded upon, while remaining in the two-dimensional plane if one were to know the location of a third object (object C) as well as its distance from object B which is three units of distance for this example, one could produce another circle which object B would lay on. Thus as illustrated Figure 3.23, object B now has potential positions at the instances where the circles intersect one another and thus the possible locations of the object have decreased in number. This process can then be iterated upon with known positions of objects and distances between objects until only one potential position remains and thus the position of the object is found.

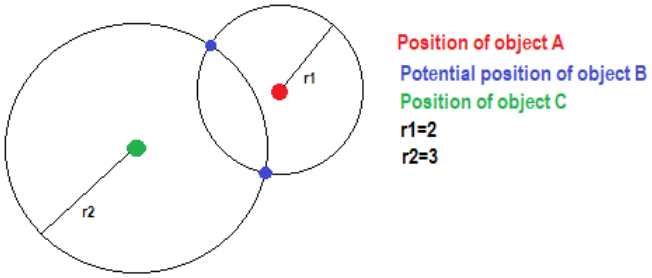

Figure 3.23: The potential position of object B in relation to object A and C

This process of locating an unknown position of an object can then be elevated to three-dimensional space. This is accomplished by making use of spheres rather than circles and the points that once represented possible locations of the object are now three-dimensional regions. These regions, like the two-dimensional scenario, become more accurate with more known object locations and distances.

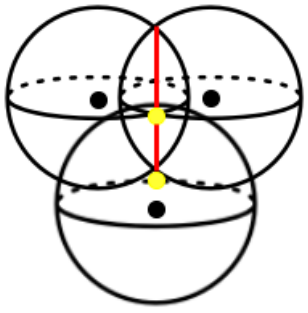

Figure 3.24: Example of GPS position calculation in 3-dimensional space [26]

Figure 3.24 is a simplified version of a figure from a paper published by Saffet Vatansever and Ismail Butun [26]. It shows the explanation for locating an object that is being made for three-dimensional spaces. It is shown in Figure 3.24 that with the use of three objects of known location, which are the GPS satellites, as denoted by the black points, one can determine two possible locations for the object of unknown location which is the GPS receiver, denoted by the yellow points. In this instance, it is noted that the closest of these possible points to the earth's surface is determined to be the position of the object, though utilizing a fourth satellite at the least is required for determining a more accurate location of the object.

3.6.2 Implementing cross-correlation to identify GPS satellites

The process of implementing cross-correlation to identify GPS satellites is discussed by Vatansever et al. [26]. It is important to discuss that satellites used for GPS each create a unique pseudo-random code which is 1023 bits in length. An identical code is created by GPS receiver units for each GPS satellite. The generated codes are pseudo-random to ensure that the cross-correlation between two different codes is extremely low.

As previously stated, a GPS receiver unit has the codes of all GPS satellites. Thus by implementing cross-correlation between the known codes and the codes being received the receiver can determine which satellites it is currently receiving information from, as the matching codes will yield a peak cross-correlation coefficient.

3.6.3 Calculating the position of the receiver

As explained in section 3.6.1 the location of the receiver is determined by using many variables from multiple satellites. It was seen from the process undertaken that to locate the object of unknown position one would need to know the satellites' positions as well as their distances from the receiver.

These two variables for each satellite, being the distance to the receiver and satellite position, are determined by information transmitted from the satellite to the receiver. The satellite transmits ranging code and navigation data. The ranging code allows for the satellite to be identified by the receiver. Additionally, it allows the receiver to measure the difference in the send-time and arrival-time of the signal [26]. This allows the receiver to calculate its range from the satellite. This range can be determined by applying the equation,

$$distance = (velocity)(time)$$
$$d = (299,79106m/s)(t_1 - t_2). \tag{3.14}$$

Where the distance is determined by multiplying the speed of light by the difference in time of when the message was sent and when it was received.

It is noted that this is an extremely simplified explanation for position calculation via GPS. Many advanced processes are implemented to accurately calculate the positions of

GPS receivers. Though for this thesis and the sake of brevity, in-depth explanations into these processes have been omitted, as their concepts are not necessary for the development of the system.

3.7 Conclusion

The principles of the methods implemented in the system have been discussed. As described in this Chapter, LoRa and GPS technology have been implemented to track and communicate from the senors. This technology requires specific hardware which must be integrated with the physical system. This hardware, as well as the software required to make use of it, is discussed in Chapter 4.

Additionally, as described in this chapter TDOA is implemented to locate the signal source and DTW is implemented in the detector. The DTW algorithm implemented in the signal detector relies on a variation of the principles explained previously, this variation is elaborated on in Chapter 4 when discussing the functionality of the detector algorithm.

As seen when discussing the TDOA algorithm, graphical representations of the TDOA data give much context and meaning to the results of the TDOA data. As such a screen is implemented in the user-unit of the system, to present the data in a meaningful way that can be understood easily.

Chapter 4

Practical implementation of theoretical principles

4.1 Introduction

Having discussed the theoretical background of the methods that are implemented in the system, the processes of implementing these theoretical principles are now discussed in this chapter.

There is a substantial difference between understanding theoretical principles and implementing the principles into methods such that it function adequately. There are many physical limitations presented with regards to processing speeds as well as data management that are dealt with.

This chapter discusses the hardware and software implemented into the system and how they interact with one another. Additionally, there are discussions of how the code has been written as to create the most efficient and accurate outcomes while still dealing with non-idealities and limitations, specifically with regards to the detection algorithm.

Discussed in this chapter is the implementation of the TDOA localization algorithm and how it differs from the theoretical implementation shown in Chapter 3. Described is the process of how the data is presented to and displayed by the program. In addition to this the processes of implementing LoRa communication, GPS tracking, and time synchronization are described in their relevant subsection.

4.2 System hardware

4.2.1 Raspberry Pi

In this thesis, various models of Raspberry Pi units have been used. This is not for any particular purpose or function but rather because of the availability of models. For this section which discusses the Raspberry Pi and for the sake of brevity specifically the Raspberry Pi model 3 B is referred to, while the actual models implemented do vary from this model.

The Raspberry Pi is a commonly known single-board computer that functions similarly to a desktop computer. Its small size and various features make it ideal for projects that

require internet connections, the simple combination of various hardware components, or running complex calculations.

According to the Raspberry Pi's documentation [27], the Raspberry Pi contains a Quad-Core Processor that runs at a speed of 1.2GHz, 1 GB of RAM, an onboard WiFi chip, four USB ports, a Micro SD port which usually contains an SD card with the operating system loaded, a full-sized HDMI port, a Micro USB power connector, a DSI display port, 40 GPIO pins and much more hardware that has not been used for the implementation and developing of this thesis project.

The GPIO pins of the Raspberry Pi are of great importance for the implementation of this project. Through these pins, the rest of the hardware can communicate with the Raspberry Pi, except the Zoom U22 which communicates through the USB port. The Raspberry Pi allows for communication via SPI and UART technologies which are used for the LoRa and GPS units respectively. A diagram detailing the layout of the GPIO pins can be seen in Figure 4.1.

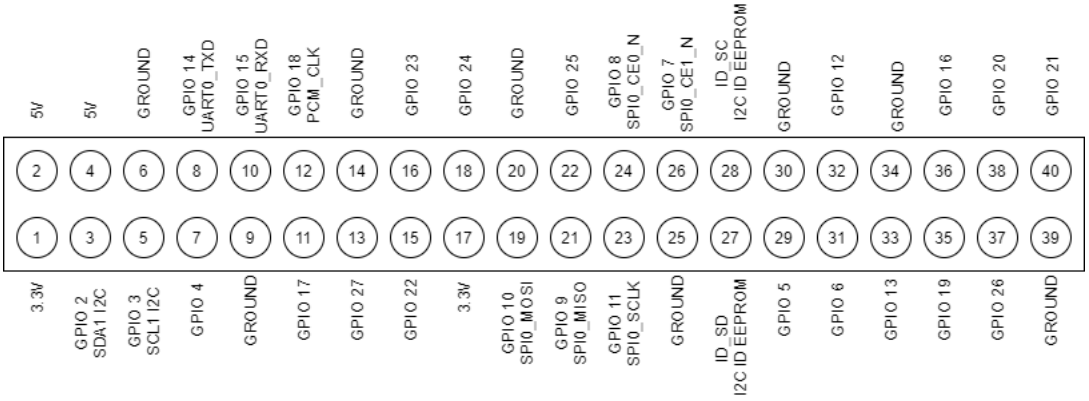

Figure 4.1: Raspberry Pi GPIO pin layout

4.2.2 Zoom U22

According to the Zoomcorp website [28], the Zoom U22 is stated to be is a high-quality mobile audio interface that is capable of recording at resolutions up to 24-bits at a sampling rate of 96 kHz. The device is powered through the Raspberry Pi USB port and functions as an analog to digital converter for the system. The device allows for 48V of phantom power which is utilized by the hydrophone.

It is apparent from the documentation that the Zoom U22 has been designed to function mainly as an audio interface which allows musicians to record music played to their computers. However, the device also happens to function as an adequate ADC for the system that functions at a suitable sampling rate as well as supplying the hydrophone with phantom power.

A drawback to using the Zoom U22 as an audio interface is the size of the device. The device is the largest single component used in the system. Its size had to be accounted for when choosing dimensions for the sensor outer casing. However, because of the device's

clear benefits such as sampling rate, USB capabilities, bit resolution, and the ability to supply phantom power, the size of the device can be justified.

4.2.3 Aquarian audio H2a hydrophone

The H2a hydrophone user's guide [29] states that it has been designed to be a cheaper and more user-friendly alternative to military and lab-grade hydrophones. It has been stated by the manufacturer that the hydrophone is capable of picking up audio signals which range between frequencies of below 20Hz and over 100kHz. Further specifications are listed in Figure 4.2.

Sensitivity	-180dB re: 1V/µPa
Useful range	10 Hz to 100kHz
Polar Response	Omnidirectional
Operating depth	<80 meters
Output impedance	2 kΩ
Power	0.3 mA

Figure 4.2: H2a hydrophone specifications [29]

The output impedance listed above is the typical output impedance. The output impedance is stated to be set in part by the bias current supplied by the Zoom U22 [29]. In addition to this output impedance, there is a cable impedance present which varies depending on the length of the cable. These impedances limit the high-frequency performance of the hydrophone.

4.2.4 SX1278 LoRa-02 unit

The LoRa hardware that is implemented for the communication system is a model SX1278 LoRa-02 unit. The pin descriptions of the PCB board connected to the SX1278 component is shown in Table 4.1. The component is noted to operate at a supply voltage of 3.3V, with an operating frequency of 433 MHz, and communicates with the Raspberry Pi via SPI. It is claimed that the module can operate up to a distance of 10 km [30].

pin number	pin name	pin description
1	ground	ground voltage
2	ground	ground voltage
3	3.3V	Supply voltage
4	Reset	Resets the unit
5	DIO 0	Digital in/out 0
6	DIO 1	Digital in/out 1
7	DIO 2	Digital in/out 2
8	DIO 3	Digital in/out 3
9	ground	ground voltage
10	DIO 4	Digital in/out 4
11	DIO 5	Digital in/out 5
12	SCK	Clock Input (SPI)
13	MISO	Data Out (SPI)
14	MOSI	Data In (SPI)
15	NSS	Chip Select (SPI)
16	ground	ground voltage

Table 4.1: Pin descriptions for the PCB of the SX1278 LoRa module [30]

The supplier of the package does not make a detailed schematic of the PCB and additional components implemented available. However, they do supply a diagram for the LoRa PCB pin layout. An adaptation of the PCB unit pin layout, showing the physical positions of the pins described in Table 4.1 is shown in Figure 4.3.

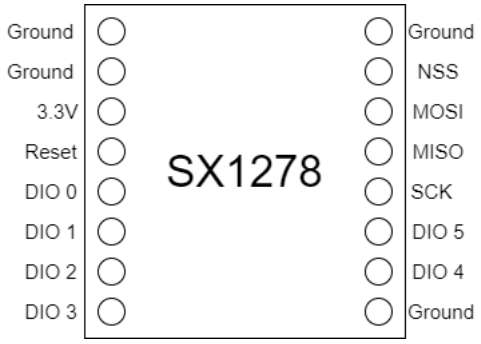

Figure 4.3: LoRa PCB pin layout [30]

4.2.5 ATGM336H GPS unit

The ATGM336H unit is noted to actually be a BDS (Beidou satellite navigation system) unit as opposed to a GPS unit. It is referred to as a GPS unit throughout this thesis. While technically incorrect, referring to the unit as a GPS unit rather than a BDS unit is more commonly understood as GPS has become a common term seen in modern everyday

life. It has been determined that technologies are incredibly similar for this thesis and as such referring to the module as a GPS module should have little to no detriment.

The device is noted to be low power and for continuous operation requires less than 25mA of current at 3.3V. The manufacturer states that the device is accurate to within 2.5m [31]. This inaccuracy of course leads to errors in the TDOA algorithm which has already previously been addressed.

The unit communicates with the Raspberry Pi through the UART pins (TX and RX). A diagram illustrating the pin layout of the PCB that the unit itself is connected to is given in Figure 4.4. Similarly to the LoRa unit, the ATGM336H has been placed on a PCB and the schematic of this PCB is not known. The PCB gives connection ports of the RX, TX, supply voltage, ground, and pulse per second (PPS).

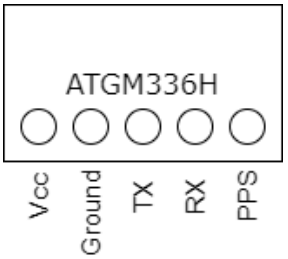

Figure 4.4: ATGM336H PCB pin layout
[31]

4.2.6 Hardware configuration

The GPIO pin configuration for both the sensors and the receiver Raspberry Pi units are the same. They both allow for connections to the GPS and LoRa units. The diagram for the connections between the Raspberry Pi and these units is shown in Figure 4.5.

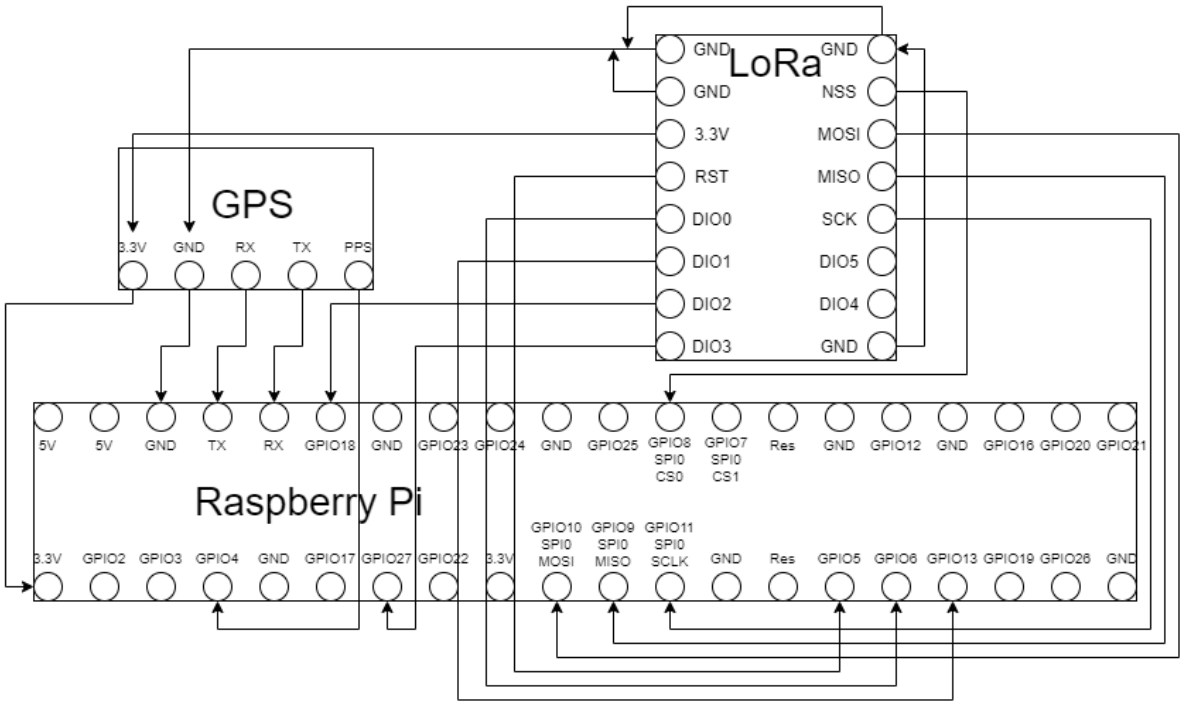

Figure 4.5: GPIO configuration connections

PCBs were designed and manufactured for the connections shown in Figure 4.5. These PCBs allow for the GPS, LoRa, and Raspberry Pi components to be mounted to one another in a modular fashion. As such, they can be replaced if need be. The board layout of the designed PCB is shown in Figure 4.6

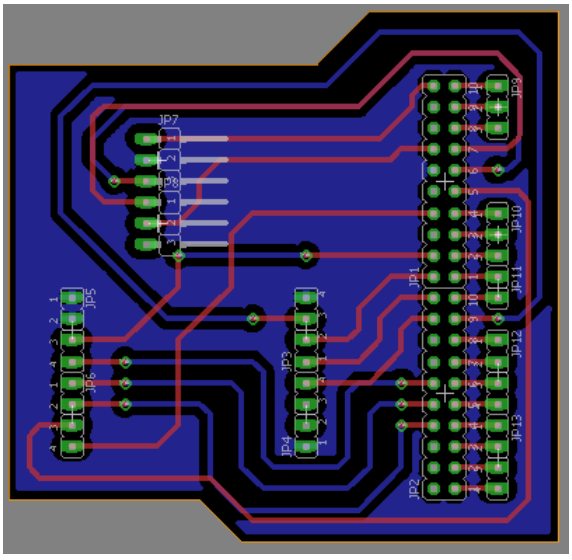

Figure 4.6: Designed mountable PCB layout

In addition to the hardware illustrated in the previous figures, each sensor consists of the Zoom U22 and hydrophone connected through a USB port, and the user-unit consists of

a small touch screen connected to the DSI display connector. These connections are not illustrated as they are very straightforward.

The PPS pin of the GPS unit is connected to the GPIO4 pin of the Raspberry Pi. This is not required for tracking the unit as the collection of GPS data is performed through the UART. Rather it serves the purpose of supplying the Raspberry Pi with a constant precise input that can be used to increase system clock accuracy.

4.3 System Software

4.3.1 Linux

Linux is an open source operating system. The Linux kernel is the core of the operating system. This kernel handles the communication between the user and the hardware of the machine. Based on the fact that Linux is open source it can be modified and customized for different platforms, in a process called porting [32]. There are several distributions of Linux ported to the Raspberry Pi chipset. One of the distributions is called Raspbian followed by an indicator of the Raspbian version.

Raspbian is an operating system that is based on Debian which is a Linux distribution. It has been optimized by the developers for the Raspberry Pi hardware. It comes with pre-compiled software and over thirty-five thousand packages [32]. The development of the Raspbian operating system is ongoing, striving for the improvement of stability and performance. Raspbian is a good starting point for inexperienced people as it comes with a graphical user interface similar to mainstream operating systems.

4.3.2 Python

Python is a programming language that was created in the 1980s. It is widely used across the world in many different fields. The language is a powerful straightforward and generally user-friendly programming language with many libraries that allow for easily implemented complex functions. Python has the capability to run across multiple platforms such as Linux and Windows making the development of code across separate machines easier [32].

Python allows the user to access information from the GPIO pins and USB ports of the Raspberry Pi with relative ease which makes it ideal for this system's development. Also, the libraries for LoRa communication, accessing serial data, DTW development, and threading were of great importance to the development of the system.

However, the Python programming language does not execute functions at the speed at which another language such as C can. This fact led to the development of a C-based DTW detector which is accessed by Python code to execute at satisfactory speeds. Though the Python DTW library proved to be far too slow in execution times, it still proved of importance to verify the accuracy of the designed C code.

4.4 Implementation of the time difference of arrival principles

As previously seen in Chapter 3, which noted in-depth how the time difference of arrival principles function mathematically, a basic TDOA system was previously designed to create the figures that were used to illustrate determining the source of a signal as well as the error analysis. This system was created in MATLAB and has already served its purpose of being a tool to illustrate the principles being explained and outcomes described. It was created to be a system that functioned based on differences in distance rather than differences in time, as this made the implementation easier. This choice also allowed for better testing circumstances as one could input the desired signal source location and test to see if the correct response was presented by the algorithm.

This functional code developed in MATLAB for the testing and illustration of TDOA principles can be adjusted to function in Python such that it can continuously monitor for input timestamps as well as continuously changing GPS coordinates and perform TDOA calculations based on these constantly changing values.

4.4.1 Presenting the TDOA algorithm with input data

To function correctly the TDOA algorithm requires a constantly updating input of GPS coordinates for each sensor, as well as the timestamp representing the time at which each sensor detected the target signal. The GPS coordinates of each sensor, as well as the timestamps of each sensor, are stored in separate text files. In addition to these six text files, there is an additional text file which contains the GPS coordinates of the user unit on the research vessel. This is stated to be 'additional' because it serves no purpose in performing the TDOA algorithm, rather it is used to locate the sensors when they are to be collected.

The format of the stored data in these text files must be kept constant for the correct data to be fetched and used in calculations. The format of storing data in the GPS documents is [unit/sensor number; sensor latitude; sensor longitude]. The sensor number is used when receiving the sensors' data from LoRa to sort the coordinates into the correct files and the latitude and longitude are used as the y and x coordinates of the sensor in the TDOA calculations. This information stored in these files is not overwritten or deleted until done manually by the user such that the sensors' path can be examined if need be. The software is designed to search the files for the most recent GPS coordinates and then accordingly updates the system.

The format of the timestamp files is slightly different in that the format simply holds an identifier as well as the minutes and decimal seconds. The format is as such, [1/0 identifier; timestamp minutes; timestamp seconds]. The identifier's importance is that it allows for the software to determine what timestamps have already been used in calculations denoted by a 0 or what timestamps are new and have not yet been collected by the software denoted by a 1. When calculating the difference in distances, the minutes are converted into seconds such that one is working with only decimal seconds. This difference in times, in seconds, is then multiplied by the speed of sound in water to result in a difference in distances. Like the GPS files, the timestamp files hold data that needs to be manually

deleted. Having a log of detecting times for each sensor in chronological order may prove useful when examining and testing the efficiency of the system and as such the data will not be overwritten, this also though in turn sets up the need for the 1/0 identifier.

It is noted here that a shortfall of this implementation is that separate instances of a Bryde's whale creating a short pulse call cannot be distinguished from one another. As such a timestamp from instance A may be given to the TDOA algorithm with another timestamp from instance B, resulting in an incorrect localization result. Implementation to resolve this issue would be a key part of future development.

4.4.2 Functional versus theoretical TDOA implementation

In this instance, the term 'theoretical TDOA implementation' refers to the basic implementation of the TDOA algorithm previously referenced that was used to create figures, perform analyses, and test for different source locations. Functional TDOA implementation refers to the more advanced implementation of TDOA that reads in actual sensor data and is designed to be the final functional version of the TDOA algorithm used.

It has already been previously stated, though it is reiterated here, that a difference in these implementations is that the functional algorithm works with timestamps and converting these differences in timing to differences in distance, while the theoretical algorithm works with known chosen distances between points. This approach for the theoretical implementation was chosen to easily change the coordinates for the origin of the fictitious signal for the testing of the algorithm without having to convert to fictitious timing values.

When converting timing to distances there is an inherent problem in that degrees of longitude and latitude are not equal in distance. While the theoretical algorithm works in normal decimals of equal magnitude on the X and Y axes, one cannot simply take this approach in a practical implementation without first converting the X (longitude) and Y (latitude) axes to the same equal units of distance. To solve this issue, the X and Y axes are converted from degrees into meters before the calculations begin, and then after the calculations are finished the coordinates are converted back to degrees before the data is displayed.

It is known that all over the world a degree of latitude distance is equal to approximately 111 km with a few decimals of variance. However, because the longitude lines converge at the poles, depending on the latitude of one's location on earth the distance of a degree of longitude varies by a substantial amount. There are methods of calculating the distances of degrees in meters at various points on earth. The latitude degree value of False Bay is approximately $34^{o}S$. This latitude results in a degree of latitude being equal to approximately 110.9km and a degree of longitude being equal to approximately 92.4km [33].

The Python functional TDOA interface has been designed to be self-adjusting in scale such that the research vessel and all sensors remain on the interface at a given time. A hyperbola from each pair of hyperbolas that have been previously discussed has been chosen using sensor proximity to estimate the correct hyperbola in each set as per the timestamp values. The end result is shown in Figure 4.7 with the sensors denoted by green markers and the research vessel denoted by the blue marker.

The correct location for the origin of the signal in Figure 4.7 was chosen to be $33.6^oS; 18.9^oW$. It is noted here that the optimization algorithms discussed in Chapter 2 are incredibly useful. However, because the location is being shown to users graphically and is not found through mathematically calculating the best single point, the optimization algorithms are omitted from the practical implementation of the TDOA algorithm.

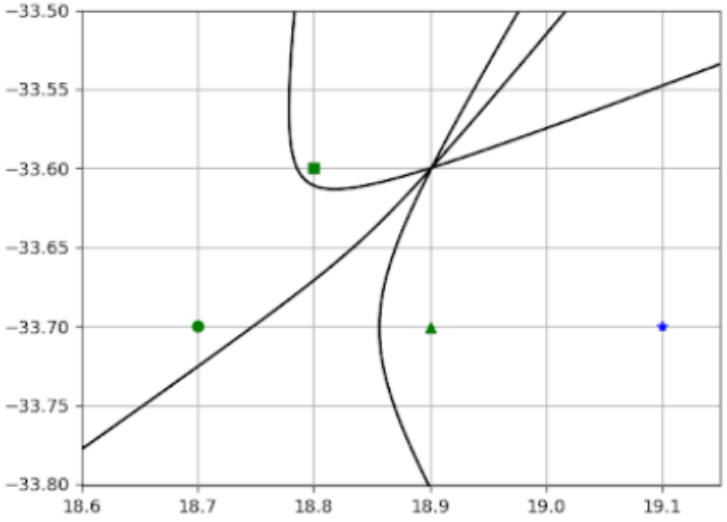

Figure 4.7: Python TDOA interface

4.5 Implementation of the DTW algorithm to detect signals

The operation of the DTW algorithm as well as key examples for this thesis have already been covered in Chapter 3. This established knowledge is required to understand the implementation of the DTW algorithm into a detector for short pulse Bryde's whale calls which will be discussed in this section. As is the case with all other software developed for the system, the detector algorithm was developed as Python code.

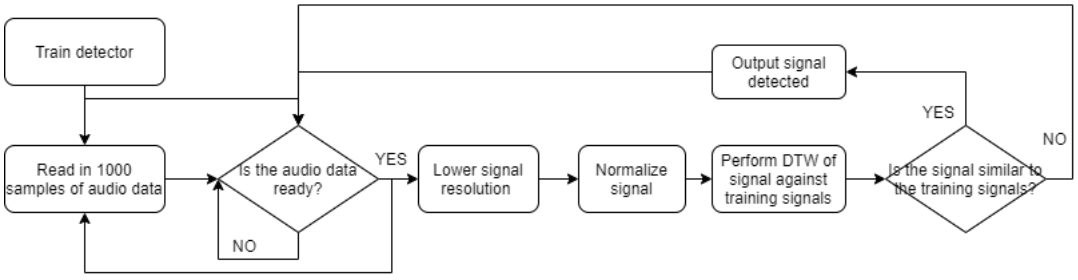

Figure 4.8: Flowchart of the detecting system

It can be seen in Figure 4.8 above that there is an instance of multitasking in this detector. The system must be able to read in audio data while it determines if the previous set of data is a short pulse Bryde's whale call. This process of implementing these two tasks simultaneously is elaborated on in section 4.5.3. It should also be noted that the "is the

audio data ready?" block comes with an implication that the detecting process should be faster than the time that it takes the system to read in 1000 samples of audio data. A window of 1000 samples was chosen because a window of this size allows for the important features of the signal to be contained with some, but not excessive, additional spacing.

The Zoom U22 integrated into the system samples audio signals at 96 kHz. This means that because Hz is the unit representing samples per second, the time taken to sample 1000 points of data can be seen as,

$$96000 = \frac{1000}{Time}.$$

This is rearranged as

$$Time = \frac{1000}{96000} \tag{4.1}$$

and as such

$$Time = 10.4ms.$$

Therefore the process of detecting whether the read-in signal is a short pulse Bryde's whale call, as seen after the affirmative response of the "is the audio data ready?" block condition, should run faster than 10.4 ms such that this aspect of the system is ready to receive the next set of audio data when it is ready and minimal data will be lost. More information on the process of how the detector was implemented to run under 10.4 ms can be found in section 4.5.4.

4.5.1 Calibration of the detector

4.5.1.1 Calibration signal background

It was seen in Chapter 3 that one can make assumptions on the source of a signal if it falls within the bounds of similarity of signals that are known to come from a certain source. Therefore if the detector is calibrated with signals that are known to be Bryde's whale short pulse calls then a signal of unknown origin can be determined to be from a Bryde's whale or not depending on how similar it is to other known Bryde's whale signals which are now referred to as "template signals".

The template signals used for this thesis were acquired in False Bay by Professor D.J.J. Versfeld on his research vessel using buoys with hydrophones and audio recorders attached. A map of False Bay is shown in Figure 4.9. The audio recordings are believed by Professor Versfeld to contain Bryde's whale short pulse calls as he claims that in each instance that the audio signal has appeared in the recordings, he had visually confirmed Bryde's whales in the area. In addition to this, the template signals have been taken from multiple recording sessions.

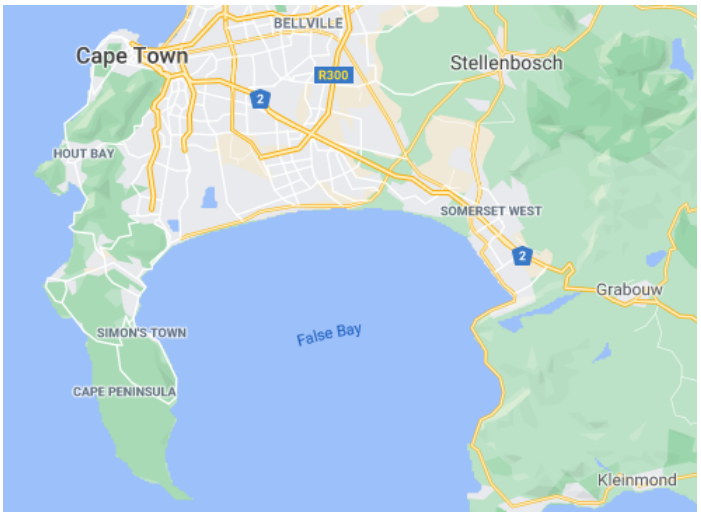

Figure 4.9: Google map of the False Bay ocean area

A Bryde's whale short pulse call can be seen in Figure 4.10 below. It was seen in the audio recordings during development that though they change in amplitude as well as having various degrees of variation and interference that the short pulse call waveform constantly shows as some variation of the signal seen in Figure 4.10. Variations can be seen superimposed on one another in Figure 4 of the paper published by Ogundile et al. [7], additionally separate calls are shown to vary in amplitude in the testing set illustrated by Figure 5.1.

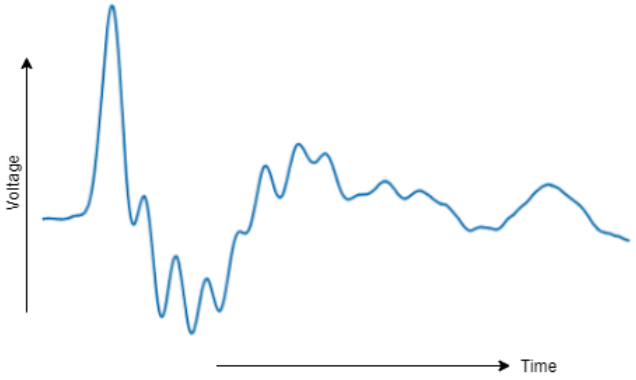

Figure 4.10: Short pulse Bryde's whale test signal

4.5.1.2 Process of calibrating the detector

Before discussing how the calibration of the detector is implemented, it should be stated that when the detector is establishing the 'threshold similarity' that a signal can have and still be classified as Bryde's whale call, it would logically follow that the more template signals the detector can compare the signal to, the more accuracy the detector will have, assuming that the features of the different signals vary from one another. However, the

fact that the system needs to run in what would appear to be real-time to a user is heavily influenced by the number of template signals, as the signal would have to be compared to each one of these template signals within the allotted amount of time.

Thus choosing the amount of test signal inputs to give to the detector is a process of balancing accuracy versus operation time. It was found that giving the detector algorithm 15 template signal inputs gives appropriate accuracy as well as keeping the operation time within the real-time limit. It is shown by Ogundile et al. [7] that when calibrating the DTW detector for this short pulse call, that there is not a considerable difference in detection accuracy when supplying the algorithm with between 12 and 18 template signals. As such 15 template signals were used because the Raspberry Pi has low complexity and as such an amount less than 18 was chosen but additionally an amount more than 12 signals chosen given for good measure.

Below in Table 4.2 are the steps taken to calibrate the detector, as these steps are listed briefly, the explanations of the steps taken are to follow below Table 4.2. In Table 4.2, N denotes the amount of chosen template signals, 15 in this case.

Step	Description
1	Load N known signals into N arrays (array1, array2, array3, ..., arrayN)
2	Cut array sizes down to 1000 values
3	Shift array values over 250 places
4	Load first 250 values with zeros
5	Lower signal resolution to lower array size
6	Normalize the signals
7	Perform the DTW of each array against the (N-1) others, including itself.
8	In each case determine the similarity by selecting the final amount in the DTW matrix as seen in Chapter 3
9	Store this value in a new 2D array where the row is the number of the array being compared to the others and the column is the array number that it is being compared to. (example: the DTW value of array2 and array3 goes in row 2 column 3)
10	Having completed the NxN array with a diagonal line of zeros across it, find the maximum value in each column and record it in a new array
11	This new array of size N is the bounds of similarity that a signal must fall between to be classified as a Bryde's whale signal

Table 4.2: Step-by-step process of calibrating the detector

In step 2 the audio signals loaded in are cut to 1000 samples each, this amount of samples was chosen because it allowed the signal to retain an adequate amount of its information without importing a signal which would not be so large that the important features would be nullified by the down-sampling process. The key signal being looked for is a sudden spike from zero that then reverberates back to zero as can be seen in Figure 4.10. Making

use of 1000 samples also allows the detection algorithm to have more time to function, if the time window is smaller the detection algorithm will have less time to execute.

Step 3 and 4 work in tandem with one another. The purpose of these steps is to shift the values of signal up in the array index, loading previous index values with zeros. This can be seen in Figure 4.11. The purpose of shifting the signal and placing zeros preceding it is to assure that the signals detected are pulses from zero or background noise. This process was implemented when it was found in testing that the detection algorithm kept detecting signals that were not short pulse calls. It is believed that this was occurring because of the position of the signal at the beginning of the 1000 sample window being analyzed. When the previous information on the signal was no longer visible in the sample window, the signal resembled a short pulse call waveform. However, upon investigation, it was found that the detections were false positives. In other words depending on where the signal is positioned in the window the lost preceding information may indicate that it is not the desired signal and as such using this shifting method ensures that the beginning of the signal is being searched for. It is noted here that results for the testing of shifted versus non-shifted signals used to calibrate the detector is shown in Section 5.2.2.1. Additionally it is stated for the sake of rigor that in the paper by Ogundile et al. [7], the authors show the effectiveness of a DTW based detector used on Bryde's whale short pulse calls through confusion analysis as shown in their Tables 2 to 7.

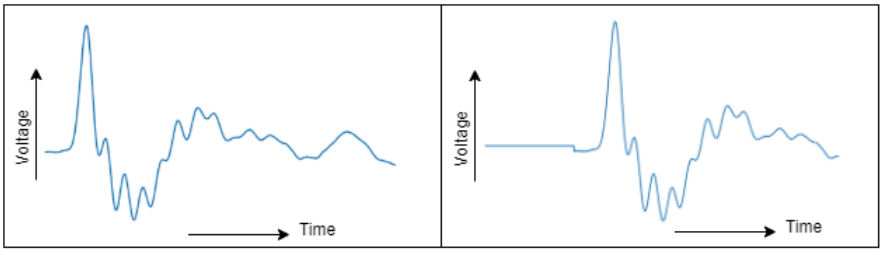

Figure 4.11: Original and shifted test signals

The resolution of the signal is lowered to increase the speed of the detection algorithm. This is discussed in greater detail in section 4.5.4.

Steps 8 to 11 are the core of calibrating the detector, whereas previous to these steps the data could be seen as just being prepared for steps 8 to 11. The DTW total difference of two signals is performed in all possible signal pair orientations to populate a 15x15 array. This array has a diagonal zero across its center because the total difference in distance (similarity) of a signal and itself is zero. In addition to this, the values are also mirrored across this diagonal according to how that array has been constructed. Figure 4.12 shows the format of how the array looks with variables substituted in for the values.

DTW of signal x and signal y is represented by D(x,y) in Figure 4.12 below.
$D(x,y) = D(y,x)$
$D(y,y) = D(x,x) = 0$

0	D(1,2)	D(1,3)	D(1,4)	D(1,5)	D(1,6)	D(1,7)	D(1,8)	D(1,9)	D(1,10)	D(1,11)	D(1,12)	D(1,13)	D(1,14)	D(1,15)
D(2,1)	0	D(2,3)	D(2,4)	D(2,5)	D(2,6)	D(2,7)	D(2,8)	D(2,9)	D(2,10)	D(2,11)	D(2,12)	D(2,13)	D(2,14)	D(2,15)
D(3,1)	D(3,2)	0	D(3,4)	D(3,5)	D(3,6)	D(3,7)	D(3,8)	D(3,9)	D(3,10)	D(3,11)	D(3,12)	D(3,13)	D(3,14)	D(3,15)
D(4,1)	D(4,2)	D(4,3)	0	D(4,5)	D(4,6)	D(4,7)	D(4,8)	D(4,9)	D(4,10)	D(4,11)	D(4,12)	D(4,13)	D(4,14)	D(4,15)
D(5,1)	D(5,2)	D(5,3)	D(5,4)	0	D(5,6)	D(5,7)	D(5,8)	D(5,9)	D(5,10)	D(5,11)	D(5,12)	D(5,13)	D(5,14)	D(5,15)
D(6,1)	D(6,2)	D(6,3)	D(6,4)	D(6,5)	0	D(6,7)	D(6,8)	D(6,9)	D(6,10)	D(6,11)	D(6,12)	D(6,13)	D(6,14)	D(6,15)
D(7,1)	D(7,2)	D(7,3)	D(7,4)	D(7,5)	D(7,6)	0	D(7,8)	D(7,9)	D(7,10)	D(7,11)	D(7,12)	D(7,13)	D(7,14)	D(7,15)
D(8,1)	D(8,2)	D(8,3)	D(8,4)	D(8,5)	D(8,6)	D(8,7)	0	D(8,9)	D(8,10)	D(8,11)	D(8,12)	D(8,13)	D(8,14)	D(8,15)
D(9,1)	D(9,2)	D(9,3)	D(9,4)	D(9,5)	D(9,6)	D(9,7)	D(9,8)	0	D(9,10)	D(9,11)	D(9,12)	D(9,13)	D(9,14)	D(9,15)
D(10,1)	D(10,2)	D(10,3)	D(10,4)	D(10,5)	D(10,6)	D(10,7)	D(10,8)	D(10,9)	0	D(10,11)	D(10,12)	D(10,13)	D(10,14)	D(10,15)
D(11,1)	D(11,2)	D(11,3)	D(11,4)	D(11,5)	D(11,6)	D(11,7)	D(11,8)	D(11,9)	D(11,10)	0	D(11,12)	D(11,13)	D(11,14)	D(11,15)
D(12,1)	D(12,2)	D(12,3)	D(12,4)	D(12,5)	D(12,6)	D(12,7)	D(12,8)	D(12,9)	D(12,10)	D(12,11)	0	D(12,13)	D(12,14)	D(12,15)
D(13,1)	D(13,2)	D(13,3)	D(13,4)	D(13,5)	D(13,6)	D(13,7)	D(13,8)	D(13,9)	D(13,10)	D(13,11)	D(13,12)	0	D(13,14)	D(13,15)
D(14,1)	D(14,2)	D(14,3)	D(14,4)	D(14,5)	D(14,6)	D(14,7)	D(14,8)	D(14,9)	D(14,10)	D(14,11)	D(14,12)	D(14,13)	0	D(14,15)
D(15,1)	D(15,2)	D(15,3)	D(15,4)	D(15,5)	D(15,6)	D(15,7)	D(15,8)	D(15,9)	D(15,10)	D(15,11)	D(15,12)	D(15,13)	D(15,14)	0

Figure 4.12: Array that stores the DTW of the template signals

Once the array as seen in Figure 4.12 has been populated, the maximum value is stored from each column. This is done to find the maximum difference that signals can have and still be known to be a Bryde's whale short pulse call. This final array is used to compare with the DTW of the read-in signals and the template signals to determine whether a signal is a short pulse call or not. A more in-depth illustration of this comparison process can be found in the section that follows.

4.5.2 Reading in audio information and detecting signals

To read in the audio signals, as it has been described previously, the Zoom U-22 which serves as the ADC is connected to the USB port of the Raspberry Pi. The data is read using a module named PyAudio. PyAudio is a Python interface to PortAudio. PortAudio is a C library that is famous for dealing with audio. The data is read in from the audio stream 1000 samples at a time. Once the 1000 samples have been stored, that data is moved to the detection algorithm and the system will immediately start to read in audio data again. As such the collection of audio data is continuous and as one frame ends another should begin. This method of running two processes at once is elaborated on in section 4.5.3 and is used to minimize information loss.

When determining if a signal is a Bryde's whale short pulse call or not, the process is similar to the calibrating of the detector. The signal once received is lowered in resolution, this is done to increase the speed of the DTW algorithm and it is explained in more detail in section 4.5.4. The signal is then normalized and the DTW algorithm can then start. The read-in signal, referred to in Figure 4.13 as S, is compared to all the 15 various template signals using the DTW algorithm. The similarities of the read-in signal and all the template signals are then stored in a 1-dimensional array comprising of 15 values, which would be the same shape and size as the maximum difference matrix as described in the last stanza of section 4.5.1.2.

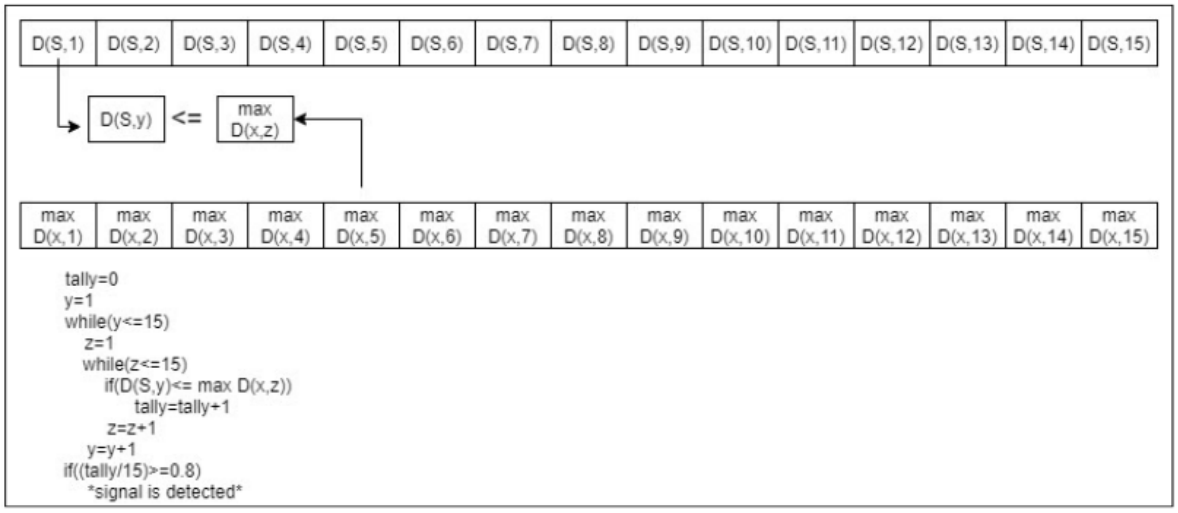

Figure 4.13: Visual illustration of the process of detection

It can be seen from the code at the bottom of Figure 4.13 that the DTW of the signal and each template signal is compared and then if the signal is more similar to each of the template signals, which are known to be short pulse calls, than another known template signal, it too would logically be identified as a Bryde's whale short pulse call. There is however an account for variation in the signals thus the signal need only be a 80% match to all the template signals as it was found in testing that perfect matches are not always given for the known signals that were being tested.

4.5.3 Threading implemented in the detector

It has been stated previously that for the loss of information in the audio readings to be at a minimum, threading has been implemented. It has been implemented such that the detection signals and the reading of the audio data can happen 'simultaneously'.

If functions are run in different threads it means that they have different flows of execution. It appears to the user that the two functions will be running simultaneously. However the different threads do not run simultaneously, they only appear to [34]. This would possibly result in the loss of some information of the read-in signal though through testing it was found that it was not noticeable.

It is believed that PyAudio already has a form of threading built into the code as through testing it was found that opening an audio stream allows for the user to read, display and manipulate data in real-time. Though this is just theorized and has not been investigated first hand. It is felt that this is not reliable enough for the detection of signals at such short time intervals and as such further steps were taken when implementing threading in the system.

The function for reading in audio data from the USB port and the function for executing the detection algorithm are both run in separate threads. This means that technically the functions may not be run simultaneously as stated above. However, the most important part of this implementation of the threading technique is that the functions do not run sequentially.

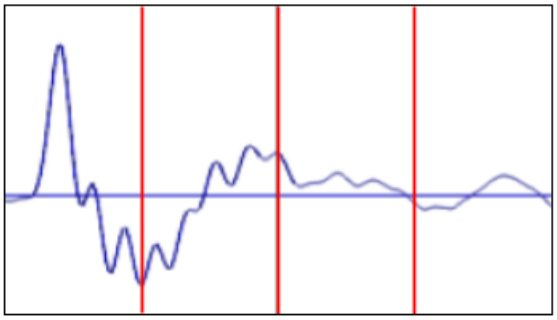

Figure 4.14: Signal read in with ideally timed threading implemented

For the sake of explanation, assume that the 4 segments in Figure 4.14 separated by red lines are all the same length of time (1 time unit). It takes 1 time unit for the data to be read into the array and thus it reads in 1 segment at a time to be loaded into the detection algorithm. For the sake of explanation, it is assumed that the time taken to pass data to the detection function is zero. The detection algorithm is also designed to be faster than 1 time unit such that it can receive that data when the data is ready to be passed. Assume it takes half a time unit to complete the detection process.

It is seen in Figure 4.14 that because the processes are running concurrently and reading in the audio data takes half a time period longer to complete than the detection algorithm, with other listed ideal assumptions, there has been no loss in data between the segments of the signal.

Now consider Figure 4.15 below where the code is run without the concept of threading. It can be seen because the data is read in 1 segment at a time and the detection takes half a time period to determine the outcome of the detection algorithm that there is a substantial loss in information between the segments. It can be seen that because the detection algorithm takes half a time unit that over the four segments a whole unit of time segment of the signal has been cumulatively lost.

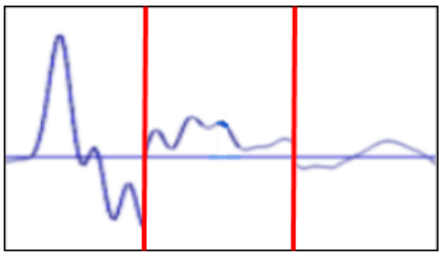

Figure 4.15: Signal read in with functions running linearly

To control the order of execution of the two functions in separate threads a flag has been implemented and acts as the indicator of when a specific function should execute or wait. The flowchart in Figure 4.16 gives a visual indication as to how the flag is implemented and coordinates the function.

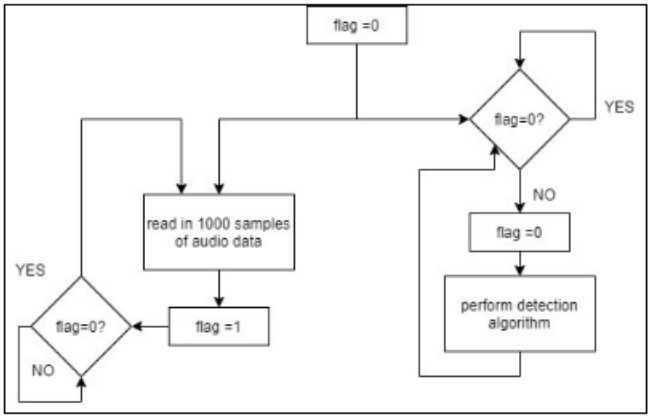

Figure 4.16: Simplified flowchart of coordinating 2 threads using a flag

Through various testing with sinusoidal signals, that are omitted from this thesis, it was found that there is an occasional delay between the loading of 1000 samples and the detection process. This may be caused by a fail-safe flag set in place that does not allow the passing of data if the detector happens to still be running. As such an additional step has been taken to minimize the loss of useful information, rather than reading in 1000 samples at a time and passing between the reading and detection processes 10 000 samples are read in at a time. These 10 000 samples are then split into 10 separate frames that are sequentially examined. As such the loss between audio frames read in is minimized when compared to the size of the signal being read in.

4.5.4 Increasing performance

A large obstacle to overcome when developing the detection algorithm is execution time. As previously stated, for as little information to be lost as possible when reading in audio data, the detector is required to run faster than the input of 1000 samples. As per Equation (4.1), this means that the entirety of the detection process must complete execution in less than 10.4 milliseconds.

When first developing the detection algorithm a DTW library was imported into Python to perform the necessary DTW computations. However, it was found after trying to calibrate the algorithm with 15 template signals, each of which consists of 1000 samples, that the Python library simply is not fast enough.

There are multiple factors to consider when discussing both the speed and accuracy of the detector created. These main factors are discussed in the following subsections and more information on speed and accuracy testing and results can be found in Chapter 5.

4.5.4.1 Implementing ctypes to the Python code

It is understood that interpreted languages, like Python, do not match the performances of compiled languages such as C. Though it is slower in execution, Python allows for the use of many existing libraries to manage functions such as LoRa communication and the reading of audio data for the USB port. As such a way to combine the speed of C with

the tools available to Python was desired. To achieve this ctypes were implemented into the Python code.

A C file for performing the DTW detector algorithm was created in its entirety. This was then tested by comparing its results with a Python DTW library available online. It proved to be reliable, calculating the same values as the Python library in a fraction of the time. These testing results are shown in Chapter 5.

The C DTW file is complied into a shared object file (.so) that can be accessed by Python code. The Python code accesses this file by making use of the .CDLL(".so file") function of the ctypes library that is imported [35].

In an article [35], the author performed speed tests for various forms of the same code. This code was written to perform the task of simply swapping values in an array of length N. Three versions of code were compared to one another, ctypes, normal Python code, and cython. He notes that cython improves performance compared to normal Python code by 35-40%. However, ctypes is 33 times faster than normal Python. Figure 4.17 is adapted from the author's findings.

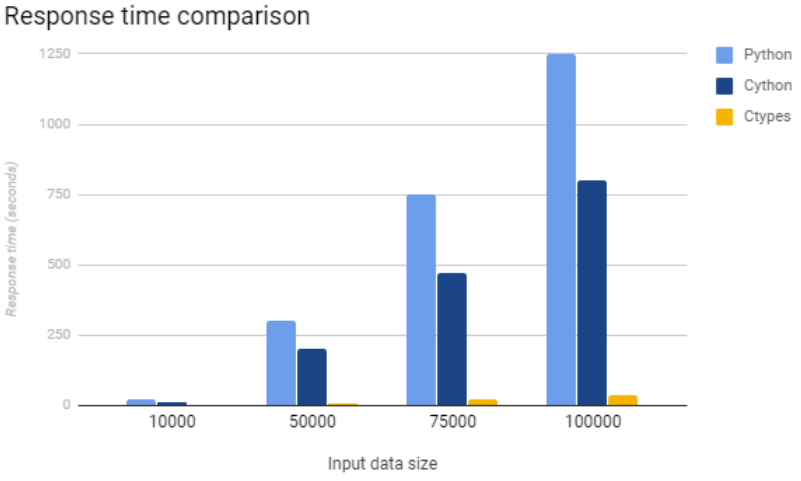

Figure 4.17: Speed comparisons for different implementations of code [35]

4.5.4.2 Decreasing the audio signal resolution

Decreasing the resolution of the signals is done to increase performance. It can be seen in the testing section that decreasing the resolution exponentially speeds up execution time. This is because of the 2-dimensional DTW arrays created when performing the DTW of 2 signals. The 2 dimensional arrays have dimensions based on the length of the signal arrays, thus the number of values in one of these arrays for the 1000 sample signals given to the detector is 1 000 000 values.

However if one were to half the signal sizes by only recording every second sample of the template and read-in signals then because each signal array is 500 samples in length

the DTW matrix only has 250 000 values that must be computed as opposed to 1 000 000. The function used to lower the signal resolution is shown below in Figure 4.18. The function reads in the signal array, designated as the variable x, and the length of the signal array. It then loops through the original array and only loads the values that are located at the downscale factor's multiples into the new array that is then returned. Thus if the factor chosen is 2, as seen below, the returned signal array will be half the size of the original.

```
def cutting(x,len):
    count=0
    count2=0
    fac=2
    newlen=round(len/fac)
    x1=np.zeros(newlen)
    while count<len-fac:
        x1[count2]=x[count]
        count=count+fac
        count2= count2+1
    return x1, newlen
```

Figure 4.18: Function implemented for lowering signal array size

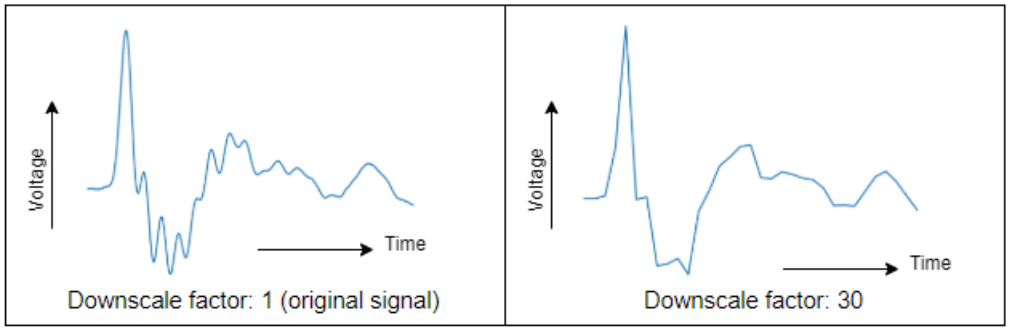

Figure 4.19: Signal received from the cutting function at various downscale factors

It is noted that another method of decreasing execution time would be to use fewer template signals as there would be less DTW algorithms to perform and less maximum values to be compared to when detecting the signals. Both of these methods logically lower execution time however they too would lower the accuracy of the detector. After testing the algorithm it was chosen to rather lower the signal resolutions than the number of template signals used.

4.5.5 Normalizing signals

It was found when developing the detection system that the variation in signal amplitude was causing the detector to be not adequately accurate. It was determined that there needed to be a method implemented to standardize the signal amplitudes to improve detection accuracy. The method implemented is commonly known as normalization, and

when normalizing audio signals there are 2 common approaches. These approaches are known as peak normalization and loudness normalization [36].

Both forms of normalization are discussed in this subsection, as well as illustrations for the effect that they have on signals commonly seen in the system. Through testing the system it was decided that loudness normalization would be implemented over peak normalization, as the process proved to be more fitting to the updating frame design of the detection system.

The reasons for normalizing signals may vary depending on the tasks at hand. For this system, normalization was implemented in response to the DTW detector. It was estimated that the detector was detecting noise as signals based on the variation in the template and read-in signal amplitudes. By normalizing the amplitudes of signals it is considered that the detector would become more reliant on waveform shape to determine detections.

4.5.5.1 Peak normalization

The process of implementing peak normalization is to find the highest sample value in the audio samples and then apply a gain such that the peak values are at a standard level. Figure 4.20 shows 2 signals that will be used to elaborate on the points made.

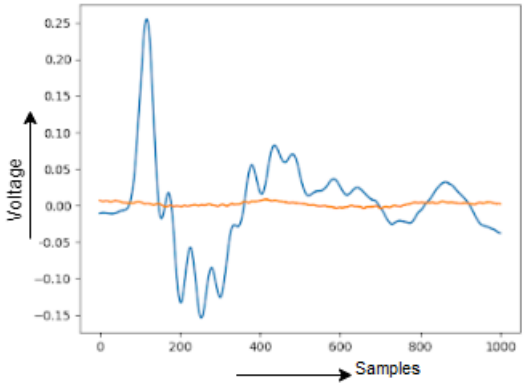

Figure 4.20: A Bryde's whale short pulse call signal (blue) and noise signal (orange)

If it is chosen the standard maximum value should be 1 for the signal seen in Figure 4.20 then the gain with which the samples are multiplied can be calculated as

$$(Peak\ value)\ \cdot gain = normalized\ peak = 1 \qquad [36]$$

thus,

$$gain = \frac{1}{(Peak\ value)}. \tag{4.2}$$

Having multiplied the sampled Bryde's whale short pulse call signal values through by the gain the normalized signal now resembles Figure 4.21. Ideally, the desired signals would clearly stand out from the noise however this is not the case. The system is designed to

normalize all signals regardless of the waveform and as such the noise that has previously been shown to be minimal when compared to the signal, is now shown in Figure 4.21 to be significant.

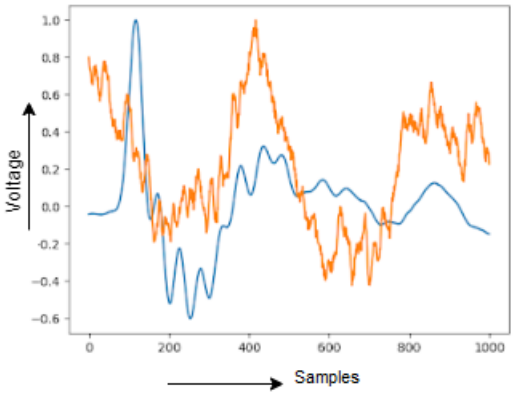

Figure 4.21: Peak normalized Bryde's whale short pulse call signal (blue) and noise signal (orange)

If peak normalization were to be applied for the peak value of each frame separately then a signal with a peak of x in frame A would have the same normalized value as a signal with a peak of y in frame B, where x is significantly smaller than y. Thus normalizing a signal in a given frame using the peak in that frame would yield false results. A similar result to this elaboration is shown in Figure 4.21 with the signal and noise.

This form of normalization was examined and implemented when developing the system however it was soon found that peak normalization in its purest form would not meet the requirements for the system. As previously discussed the system reads in frames of 1000 samples in size to be worked on. If the gains are calculated for each frame separately, this sets up a dilemma.

However, the peak normalization algorithm can be adapted to fit the implementation of the detector. By finding the largest peak experienced thus far, retaining this value, and normalizing it with respect to it, then replacing it when a new greater peak is found one can draw more standardized results. There are still problems with this approach though. If one were to experience low volume noise for a significant time at the start of the process, the loudest peak would still be low and thus the gain implemented would make this noise appear to be signals of value until an adequate peak is found.

4.5.5.2 Loudness normalization

This form of normalization is based on the overall loudness of the audio signals as opposed to just the peaks. To normalize in terms of loudness one would normalize to the RMS value of the signal [36][37] by applying

$$signal_{normalized} = \frac{signal}{RMS} \tag{4.3}$$

where

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} |signal_i^2|}{N}}. \tag{4.4}$$

When Equation (4.3) is translated into the digital domain to be implemented with the samples of the signals it is written as

$$s_n[k] = \frac{s[k]}{\sqrt{\frac{s[k] \cdot s[k]}{N}}}. \tag{4.5}$$

Figure 4.22 illustrates the loudness normalization of the separate signals seen in Figure 4.20 similarly to Figure 4.21 previously. Figure 4.22 shows that while the noise has still become more substantial, the amplitude is lesser than the peak normalization method.

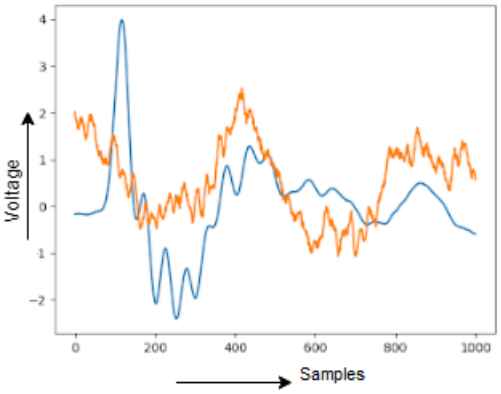

Figure 4.22: Loudness normalized Bryde's whale short pulse call signal (blue) and noise signal (orange)

This form of normalization has been implemented into the detection system as it can be calculated for each frame without the dilemmas that peak normalization incurs. As just stated there is no need for a 'cross-frame' normalization, the likes of which were mentioned for peak normalization. The reason for this is because the windows are normalized to their own average once read into the detector and when calibrating the detector the same process was undergone for the frames containing the template signals.

4.6 Implementation of GPS technology to track and monitor sensor positions

As previously discussed, the GPS unit is connected to the Raspberry Pi via the UART pins. To access the information in Python, the serial library is implemented which gives one access to the data being sent to the UART pins of the Raspberry Pi. It is important before continuing to reiterate and clarify the difference between GPS and Beidou. As previously stated GPS is a type of GNSS. GPS technology was created by the United States of America. Beidou serves the same purpose as GPS however it was created by China.

Though the term GPS is used to describe the tracking of the sensors moving forward it is noted here that the satellites being used at any point to track the sensors' positions may be Beidou satellites rather than GPS satellites. This statement is present to note the difference in technology, or lack thereof, being implemented. Though the term GPS is used throughout this document, rather than Beidou or the more general GNSS term, this definition has no real impact on the implementation of the tracking system.

4.6.1 NMEA Data structure

The data that is transmitted to the serial port is known as NMEA data. This stands for National Marine Electronics Association and it is a standard data format supported by all GPS manufacturers. The NMEA data structure is quite straightforward to interpret once one has an understanding of the structure. The table that follows contains information for understanding the NMEA messages.

Indicator	Description
$GNGGA	Time, position, and fix related data of the receiver.
$GNGLL	Position, time and fix status.
$GNGSA $GPGSA $BDGSA	Used to represent the IDs of satellites which are used for position fix. When both GPS and Beidou satellites are used in position solution, a $GNGSA sentence is used for GPS satellites and another $GNGSA sentence is used for Beidou satellites. When only GPS satellites are used for position fix, a single $GPGSA sentence is output. When only Beidou satellites are used, a single $BDGSA sentence is output.
$GPGSV $BDGSV	Satellite information about elevation, azimuth and CNR, $GPGSV is used for GPS satellites, while $BDGSV is used for Beidou satellites.
$GNRMC	Time, date, position, course and speed data.
$GNVTG	Course and speed relative to the ground.
$GNZDA	UTC, day, month and year and time zone.

Table 4.3: NMEA Indicators and their meanings
[38]

When the table above is applied to the example data in Figure 4.23 the process of understanding the NMEA data becomes more straightforward. This data in Figure 4.23 below is taken directly from the system while in development. The information of interest for the system's purpose is the information that follows the '$GNGGA' indicator as this contains the time as well as the position of the sensor. Additionally, it is shown in the table that the message layout is 'Time, position, and fix related data of the receiver'. Therefore when analyzing Figure 4.23 below it is seen that the time is "13:11:15" and the location coordinates are " $3355.73527S; 01851.97138E$" in the degrees, minutes, and decimal minutes format.

```
$GNGGA,131115.000,3355.73527,S,01851.97138,E,1,15,0.9,167.7,M,0.0,M,,*6E
$GNGLL,3355.73527,S,01851.97138,E,131115.000,A,A*51
$GPGSA,A,3,04,07,08,09,11,16,21,26,,,,,1.5,0.9,1.1*3D
$BDGSA,A,3,08,09,10,11,12,13,19,,,,,,1.5,0.9,1.1*27
$GPGSV,3,1,10,04,55,316,19,07,22,229,36,08,60,348,13,09,44,253,40*73
$GPGSV,3,2,10,11,22,004,22,16,53,148,39,21,24,064,25,26,21,117,38*70
$GPGSV,3,3,10,27,74,093,03,33,,,24*41
$BDGSV,2,1,08,06,15,139,,08,17,119,43,09,65,139,21,10,06,111,27*6B
$BDGSV,2,2,08,11,65,139,21,12,65,139,22,13,65,139,21,19,65,139,21*6A
$GNRMC,131115.000,A,3355.73527,S,01851.97138,E,0.07,0.00,200820,,,A*69
$GNVTG,0.00,T,,M,0.07,N,0.13,K,A*26
$GNZDA,131115.000,20,08,2020,00,00*44
$GPTXT,01,01,01,ANTENNA OK*35
```

Figure 4.23: Real NMEA data from the serial port

4.6.2 NMEA Message processing

As previously stated the information that follows the '$GNGGA' indicator is of importance to the system for tracking purposes. However, the rest of the messages can be disregarded as they are deemed unnecessary for the development of the tracking system and though the information is not inherently useless it is not required for the tracking system implementation.

To implement the process of disregarding non '$GNGGA' messages from the serial port, Python code was developed. This code disregards the messages received from the serial port that do not contain the '$GNGGA' indicator. After the message received from the serial port is confirmed to be correct, the message is passed to a separate function that then processes the message further. After the message has been passed on from the function that confirms the correct indicator, the process of analyzing the message and allocating the appropriate information to the correct variables begins. The process of sorting through the $GNGGA message is described and shown in Table 4.4 below.

Step	Description
1	Store data after $GNGGA indicator
2	Split the stored data at the commas and load this split data into an array
3	Allocate the data in the corresponding array positions to variables
4	Convert the coordinates from strings to floats
5	Convert the float values to degree decimal format

Table 4.4: Process of breaking up the '$GNGGA' message

4.6.3 Presenting the GPS coordinates in OpenCPN

In order to present the data in a user-friendly method available while at sea, the GPS coordinates of each sensor as well as that of the research boat are displayed graphically as AIS units on OpenCPN as well as on the Python TDOA interface. The OpenCPN method can be used as a back up to view and track the coordinates of the sensors graphically

in the circumstance that the Python time difference of arrival interface malfunctions or lacks a feature provided by the OpenCPN application. The Python files for interfacing with OpenCPN are adapted from the code posted by a GitHub user [39]. In order for the plotting of AIS points in OpenCPN to function, the coordinates of each sensor are written to separate text files, these text files are then called on by separate code that ports the most recent coordinates in the text files to the OpenCPN program.

In order to port the coordinates into the OpenCPN application they first need to be translated into a format that OpenCPN can read in. The process of doing so and then porting to the application is performed by Python code. This code is designed to fetch the coordinates of each sensor from the appropriate text files that have been used to store the coordinates received from LoRa communication.

There are 3 sensors in the water as well as the boat. Thus there are 4 GPS coordinates of hardware to display in total. The information required for plotting each sensor position in OpenCPN is the type of vessel, the Maritime Mobile Service Identity (MMSI), the status of the vessel, the speed, the longitude, the latitude, the course, the heading, and finally a timestamp.

As discussed previously, the information transmitted for each sensor position is the sensor number and the longitude/latitude of the sensor. Thus the rest of the information used to plot the coordinates in OpenCPN is not transmitted from the sensor itself but is rather set to arbitrary default values as they are not of importance at this time. This can be seen in the code extract seen below in Figure 4.24.

```
LineDict2['TYPE'] = '1';
LineDict2['MMSI'] = '0';
LineDict2['STATUS'] = '5';
LineDict2['SPEED'] = '0';
LineDict2['LON'] = str(redlon);  //longitude from sensor
LineDict2['LAT'] = str(redlat);    //latitude from sensor
LineDict2['COURSE'] = '0';
LineDict2['HEADING'] = '0';
LineDict2['TIMESTAMP'] = '2019-11-21T07:11:47';
```

Figure 4.24: Setting the appropriate values for an AIS target [39]

Before the data is sent to OpenCPN application the data needs to be converted into an NMEA format and thus all the data that was disregarded from the original NMEA messages can just be substituted with chosen fixed data because as previously stated for these purposes it is not required. In addition to this, sending the unnecessary data anyway would require longer transmission times from the LoRA units, as there would be much larger information to encode, send and decode.

The data for the NMEA message is gathered, constructed, and then converted into basic binary strings. A basic illustration of one of these conversions would be if the status = 5 then the converted status = [‘0’, ’1’, ’0’, ’1’]. The function designed for converting the values to binary values is designed to convert integer values and thus all float values,

such as the longitude and latitude are converted to an integer format before converting to binary form [39].

The large binary message, consisting of all the data required, is broken up into 6-bit segments. These segments are converted to integers which then have a correlating character for AIS encoding. The messages as a whole are then converted into an AIS string consisting of all the characters that have been encoded which can then be ported into OpenCPN [39]. The results of this conversion can be seen in the Figure 4.25.

{'TYPE': '1', 'MMSI': '3', 'STATUS': '5', 'SPEED': '0', 'LON': '18.8', 'LAT': '-33.7',
'COURSE': '0', 'HEADING': '0', 'TIMESTAMP': '2015-11-19T07:10:57'}

!AIVDM,1,1,,A,100000mP00QF3l1deo`0001j0000,O*79

Figure 4.25: AIS data and encoded AIS data for OpenCPN

When the message for each sensor resembles the one as seen above, the message is ported into the OpenCPN application using the socket library in Python. The end result of the visualization process can be seen in Figure 4.26.

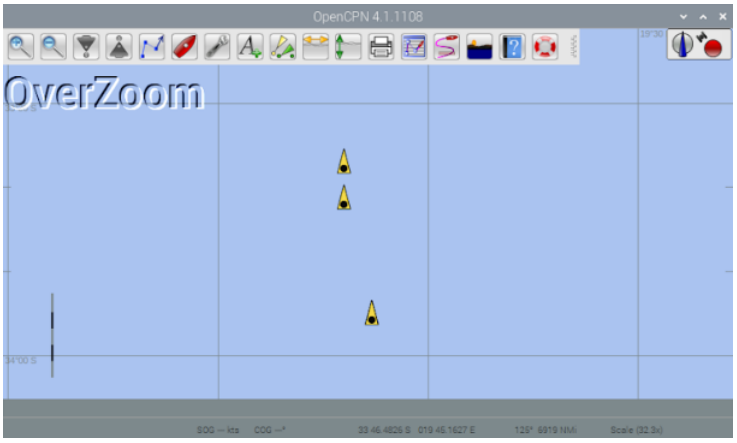

Figure 4.26: AIS targets plotted at the 3 sensor coordinates

4.7 Implementation of LoRa technology to transmit and receive sensor data

The LoRa technology has been implemented in Python code to transmit information from the sensors and receive said information at the user unit on the research vessel. Therefore intuitively two different versions of LoRa of code had to be developed to execute this send and receive functionality in devices that function independently from one another.

LoRa technology was intended for a 'node' to communicate with a 'gateway'. Nodes are designed to transmit, via LoRa, directly to in-range gateways. Gateways are designed to

relay messages between nodes and a central network server by making use of an internet connection [40]. It is because of this terminology that the LoRa communication system implemented for this thesis is believed not to be deemed a standard LoRa communication setup, as there is no gateway present. Though it is also possible for LoRa nodes to communicate with each other. One such instance of this is when information needs to be transmitted over long distances [41].

The LoRa configuration implemented in this system is described as peer to peer LoRa communication. In such a configuration the transmitter of data is referred to as a 'server' and the receiver of such information is referred to as a 'client' [41]. As such in this system, there would be what resembles a star topology present, as the sensors are all servers and the user unit would be the client. As to prevent confusion in the client receiving data from multiple servers, the transmission instances in the servers are staggered as to not allow the signals to interfere with one another.

4.7.1 LoRa unit setup

The SX127x Python library has been installed from a GitHub repository [42]. The LoRa package of this library has been implemented to set up LoRa communication. Additionally, the board configuration (board_config) package has been implemented to set the board and LoRa parameters.

This process involves allocating the correct pins on the LoRa module to the GPIO pins that they are physically connected to and allowing for the communication of information sent over SPI. Also, the module is set to use a low band frequency of between 137-175 Hz and 410-525 Hz [42].

The appropriate GPIO pins need to be designated as input pins or output pins according to their functionality. The only pin that requires to be allocated as an output pin in this setup is the connection to the reset of the LoRa module. One can also allocate the LED pin as an output to give visual indications of the system's functionality though it is not actually necessary for the functioning of the LoRa unit.

The SPI is set up using the spidev Python library and configured to function at a maximum speed of 5MHz. While it is known that the SPI can function at speeds which double the chosen amount, a lesser value has been implemented in order to err on the side of caution, as stated in the pySX127x documentation [42].

The LoRa.py file contains functions that pertain to the use of the LoRa unit for the purpose of transmitting and receiving data. This file contains important functions that allow the user to; set the mode of the device, receive various values and statuses of the LoRa module, and of course send and receive messages. It is through the LoRa.py file that one can control the unit that has been set up through the board_config.py file.

4.7.2 LoRa implementation in the user unit receiver

In order to function, a LoRa class is written in the receiver unit. This class requires 3 primary functions; the init function, the start function, and the on_rx_done function.

The init function is implemented to initialize the LoRa module with a 433MHz carrier frequency and a 125kHz bandwidth. The init function then puts the module into a "sleep mode". This is done to conserve power usage [41].

The start function's purpose is to configure the LoRa module as a continuous receiver (RXCONT) of LoRa signals. In addition to setting the module as a receiver the start function also continually obtains information required for operation such as the Receiving signal strength Indicator and the status of the module [41].

The on_rx_done function is triggered once a message has been received. The function stores the received information in a variable which is then decoded from ASCII code to normal characters and joined together to form the whole message. Once the message is in the correct format it can be scanned for the appropriate unit/sensor number and then be saved in the appropriate file that stores said sensor's data to be accessed by the code that displays the information, as previously discussed in section 4.4.1. The module is then put back into sleep mode until another message is received.

4.7.3 LoRa implementation in the sensor transmitter

The transmitter makes use of similar functions as the receiver unit described in the previous subsection. The class implemented makes use of its own init and start functions as well as an on_tx_done function. The fact that the Python libraries available make the use of the LoRa units easier, the functions that have been required for operating the units are very similar in execution and function. Thus the descriptions of these functions that follow are very similar to those previously.

The init function that is implemented in the transmitter similarly to the receiver initializes the LoRa module with a 433MHz carrier frequency and a 125kHz bandwidth and then puts the module into a "sleep mode". The only noticeable difference that has been implemented between the transmitter and receiver init functions is that the DIO mapping for the LoRa unit has been changed.

The start function configures the LoRa module as a transmitter of LoRa signals. The message that is to be transmitted is converted into ASCII characters to be transmitted and decoded by the receiver in this function.

The on_tx_done function is triggered once a message has been sent. This function then puts the transmitter back into a standby mode and flushes the system.

4.8 Unit time synchronization

The sensor units are known to transmit timestamps at the instance of detection, these timestamps are then transmitted and used in the TDOA algorithm. However to execute this process accurately the timing on the sensors must be synchronized such that the timestamps are all relative to one another. The synchronization is performed by setting the clock on each sensor unit to the GPS time read into each system independently. Additionally, the timing on the sensors is made more accurate through the use of the pulse per second signal available on the GPS units.

4.8.1 The process of synchronizing the unit clocks

The process of synchronizing the unit clocks implemented is adapted from the process detailed in a web-page posted by Steve Friedl [43].

The PPS data is obtained and utilized by implementing the pps-tools package. The Linux operating system has kernel support for PPS input. Linux associates an extremely accurate timestamp with the rising edge of the PPS signal and makes these timestamps available [43].

Information is added to the /boot/config.txt file which enables the PPS support and indicates which GPIO pin the signal will travel through. It has been shown by the diagram illustrated in Figure 4.5 that the PPS pin of the GPS module has been connected to GPIO4 of the Raspberry Pi. The PPS information was printed to the terminal and is shown in Figure 4.27.

```
pi@raspberrypi:~ $ sudo ppstest /dev/pps0
trying PPS source "/dev/pps0"
found PPS source "/dev/pps0"
ok, found 1 source(s), now start fetching data...
source 0 - assert 1593291964.591066707, sequence: 569 - clear  0.000000000, sequence: 0
source 0 - assert 1593291965.591074988, sequence: 570 - clear  0.000000000, sequence: 0
source 0 - assert 1593291966.591085665, sequence: 571 - clear  0.000000000, sequence: 0
source 0 - assert 1593291967.591096602, sequence: 572 - clear  0.000000000, sequence: 0
source 0 - assert 1593291968.591108424, sequence: 573 - clear  0.000000000, sequence: 0
source 0 - assert 1593291969.591119934, sequence: 574 - clear  0.000000000, sequence: 0
```

Figure 4.27: PPS information

The valid information for the GPS time as well as the PPS timing is fed into the network time protocol (NTP) which has been configured to receive this information and make an accurate time available to the units independently from one another. The NTP has the capabilities to corroborate the GPS time information with information fetched from internet servers however this feature of the NTP has been excluded as the sensor units have no valid internet connection, they rely solely on GPS for receiving timing information.

It is described by Steve Friedl [43] that the NTP receives the GPS time information, which has previously been loaded into a package called GPSD specifically for GPS data, via shared memory. In order to give the NTP access to this shared memory, aspects of the /etc/ntp.conf file needs to be modified. By placing fake "servers" for the NTP to access in this file one can provide the NTP with the information in shared memory. These "servers" have IP addresses in the 127.127.t.u range, where the value of t is the clock driver type and the value of u is the unit number within the said type. The IP address 127.127.28.u is the Shared Memory driver supported by GPSD that is required for this process.

```
pi@raspberrypi:~ $ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*SHM(2)          .PPS.           0 l   53   64    1    0.000   -0.342   0.001
+SHM(0)          .GPS.           0 l   49   64    3    0.000  -174.57  12.119
```

Figure 4.28: NTP information

The resultant NTP is shown in Figure 4.28. To elaborate on the information portrayed in this figure, the PPS and GPS information has been labeled and any external internet servers have been omitted. The meanings of the field indicators, shown in Figure 4.28, are elaborated on in Table 4.5. In addition to this table, there is a supplemental table, Table 4.6, which lists the peer status words present for each peer of the NTP and their meanings.

Indicator	Description
remote	The name of the time source
refid	The identifying tag
st	The quality of the server (lower numbers are better than higher numbers).
t	The type of clock, u represents unicast over the network, l represents local and b represents broadcast
when	The time (seconds/minutes/hours) since the last time packet was received
poll	The poll interval in seconds
reach	The bitwise notation of reachability of this peer
delay	The "round-trip" delay, in milliseconds, to reach this peer
offset	The difference, in milliseconds, between this peer and the local clock
jitter	The average deviation in time from the peer, in milliseconds

Table 4.5: NTP field indicators and their meanings
[43]

Word	Description
nothing	Discarded as not valid
x	Discarded by the intersection algorithm
-	Discarded by the cluster algorithm
+	Included by the combine algorithm
#	Backup time source
*	Indicates a System peer
o	Indicates a peer whose driver support is directly compiled into NTPD

Table 4.6: NTP status words and their meanings
[43]

4.8.2 Serial port conflict

Once the NTP is implemented it continuously pulls the GPS data from the GPSD package which in turn pulls the information from the serial port connected to the GPS module.

This creates a dilemma between the time synchronization process and the tracking subsystem of the sensors. The dilemma is rooted in the fact that both the GPSD package and the Python code that has been written to obtain the GPS coordinates of the sensors are trying to access information from the serial port. As such when the Python code is run an error is returned stating that the serial port is currently busy.

The solution to this problem that was implemented is for the Python code that manages the tracking of the sensors to pull the GPS information for the GPSD package that is running rather than from the serial port itself. As such the GPSD package has access to the serial port and the NTP and tracking code both share access to the GPSD, the conflict was resolved.

4.9 Conclusion

It has been shown throughout this chapter how the theoretical principles discussed in Chapter 3 have been physically implemented with the use of the combination of hardware and software.

It has been shown that the implementation of these principles has not been incredibly straightforward. Multiple steps had to be taken to ensure that the various aspects of the system functioned properly, such as the various steps taken to ensure detector speed and accuracy. Testing results of the various forms of detector implementation are shown in Chapter 5.

The system has shown to be functional throughout the process of practical implementation, though it needs to be tested to show how effective it is as a Bryde's whale localization system. The testing of the various aspects of the system and the results that are produced are shown in Chapter 5.

Chapter 5

Simulations, testing and results

5.1 Introduction

The implementation of the various aspects of the system have been described in the previous chapter and what follows is the testing of these aspects of the system.

The primary tests on the detector are done to measure its accuracy and speed and as such determine the best method of implementation for the detector.

The TDOA localization system is tested to determine its accuracy. This is done by quantifying its margin of error for the calculated location of signal sources versus the known location of signal sources.

Additionally, the time synchronization between units and the unit tracking and communication systems are also tested through various means.

5.2 Detector tests implemented during development

5.2.1 Testing detector speed

In order to test the speed as well as accuracy for the detection algorithm, the system has been given the signal seen in Figure 5.1 to work through 1000 samples at a time, as it would if it were simply reading in the data from the audio source. These calls present in the signal is believed to come from the same whale. It is understood that testing using the same whale as the testing data source is limiting, though it can be seen that there is variance in the calls present in the signal. This signal was used because it shows clear calls sequentially after each other in a relatively small time frame. Additionally it is noted that the Bryde's whale call data obtained first hand by the researchers is limited and currently the research team has no way of discriminating between whales when capturing data.

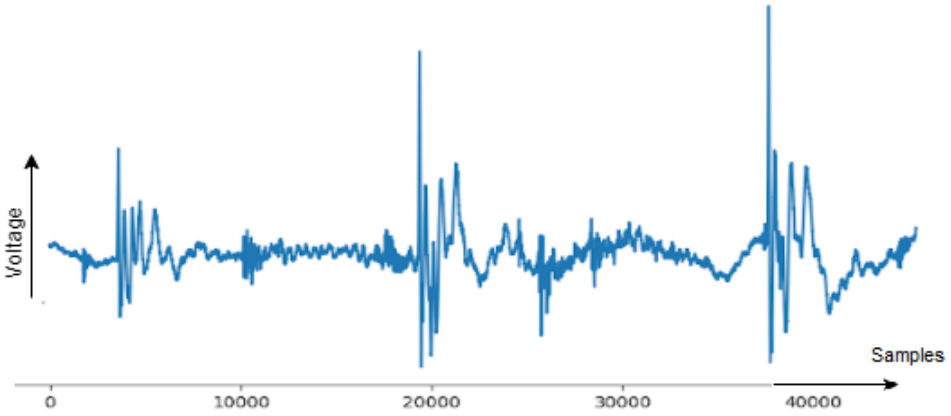

Figure 5.1: Testing signal

It can be seen in Figure 5.1 above that there are 3 main areas of interest, where the short pulse calls first start, these instances are at 3000, 19000, and 37000 samples into the audio signal as a whole. The specific 1000 sample windows where these signals are located are shown in Figure 5.2 below.

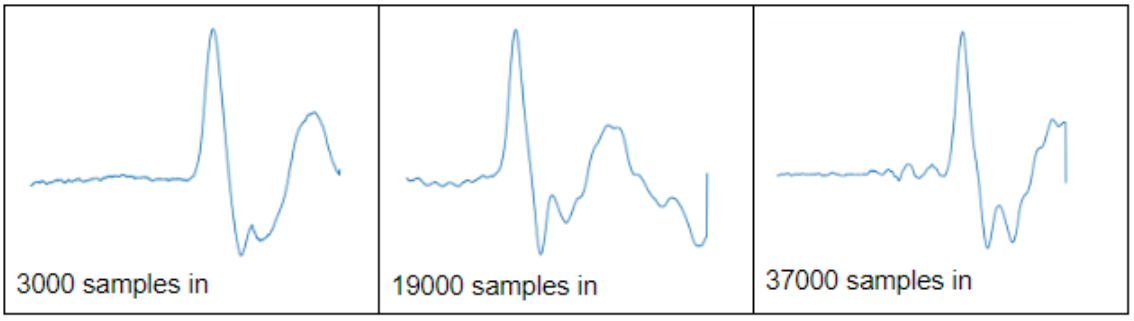

Figure 5.2: Each desired target window

For the majority of the tests done these signals are detected as well as other non-desired signals. One of the aims of this testing is to set up the detector to detect these 3 instances with as few 'false positives' as possible. However, before examining accuracy, the speed of the detector needs to be prioritized. As it has been explained previously that the whole detection process must run in under 10.4 milliseconds per 1000 sample window.

Two main facets make up the total detection algorithm, these being the preparation of the data and the detection process using said data. An aspect of Figure 4.8 is highlighted in Figure 5.3 which categorizes the parts of the detection algorithm that are for preparing the data and that are for detecting the desired signals in the data. These processes are speed tested separately because the preparation process requires substantially less time to execute compared to the detection process, though there are still changes in execution time that should be noted.

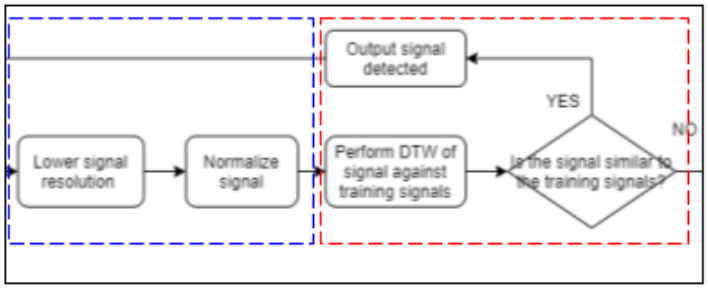

Figure 5.3: Signal preparation processes (blue) and signal detection processes (red)

The preparation of the signals is known to consist of normalizing the signal and lowering its resolution. Both of the processes have been discussed previously. Both processes require sorting through the signal array and thus, logically, the smaller the array is the less time it would take to execute these functions. This is why as the downscale factor gets larger the execution time gets smaller. The times can be seen to flatten out near zero and there is no noticeable change in time for downscale factors between 40 and 100, this is suspected to be because the hardware speed of the device itself is now the main contributor to the execution time rather than the number of executions.

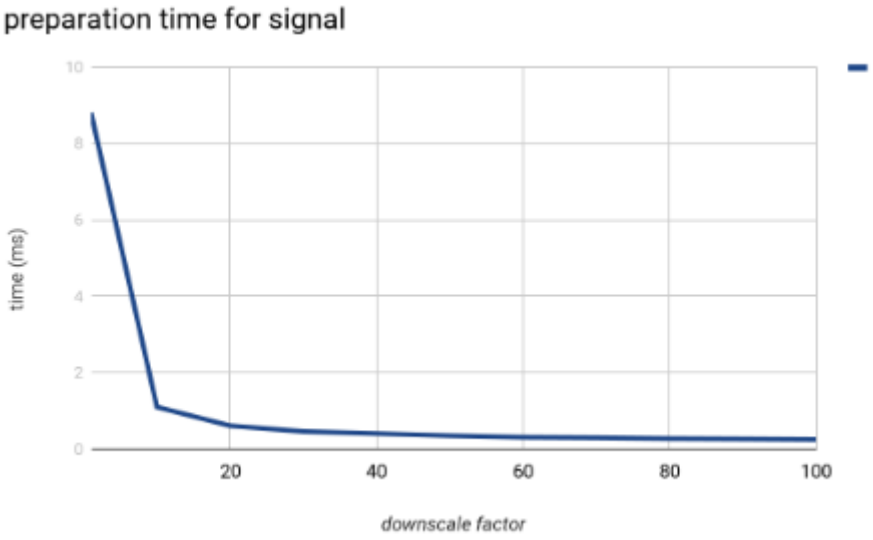

Figure 5.4: Preparation times for signals over different downscale factors

When analyzing the times of the detection aspect of the algorithm after the data has been prepared 2 speed tests have been run, one for the ctypes implementation of the DTW algorithm and one for the DTW Python library. The graphs in Figures 5.5 and 5.6 have been kept separate as they have a substantial difference in scale.

It can easily be seen that the ctypes implementation of the detector algorithm is remarkably faster than the pure Python implementation. It is noted here that the lowest time that the pure Python reads here is 0.0895 seconds at a downscale factor of 100. This means that with only 10 points of information, the pure Python implementation cannot

come close to the speed of the ctypes implementation, which crossed the 0.0895-second mark when having a downscale factor between 1 and 10.

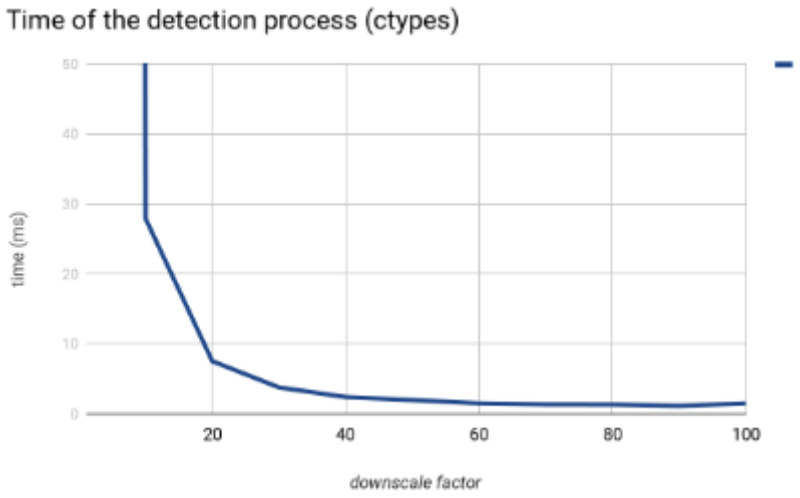

Figure 5.5: Detection times for signals over different downscale factors using ctypes

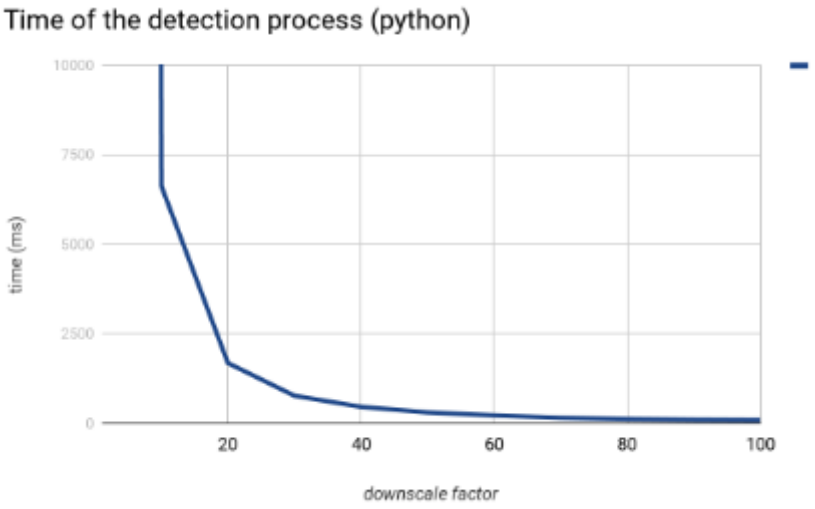

Figure 5.6: Detection times for signals over different downscale factors using only Python

Figure 5.7 shows the total time for the whole detection process to run when using ctypes. It can be seen from the graph that for the algorithm to execute within the allotted time the downscale factor would have to be approximately 20 or higher.

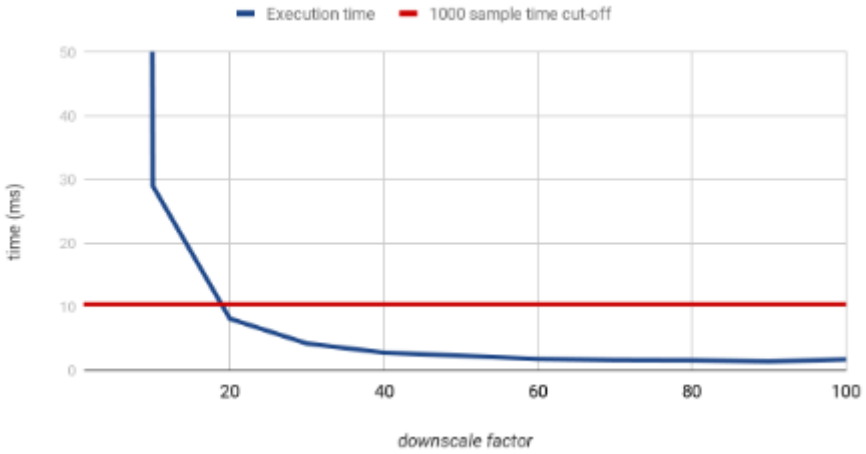

Figure 5.7: Execution time for the whole detection algorithm

5.2.2 Testing detector accuracy

5.2.2.1 Accuracy tests for shifted/non-shifted signals of different resolutions

It had been mentioned previously that the template signals if shifted in time were presumed to give more accurate results. This has been tested and the results follow. In figures 5.8 and 5.9 the graphs show the total amount of signals detected in the test signal, shown in Figure 5.1. As previously stated there should ideally only be 3 detections. All cases where a detection is found that is in addition to these 3 instances is deemed a 'false positive'.

In figures 5.8 and 5.9 the darker shade of blue represents the number of the desired signals detected, which ideally should be 3. The lighter shade of blue shows the number of 'false positives' detected in the test signal. Therefore the combination of dark and light blue represent the total number of detections.

It is noted before showing the results that in these accuracy tests loudness normalization was implemented in the detector.

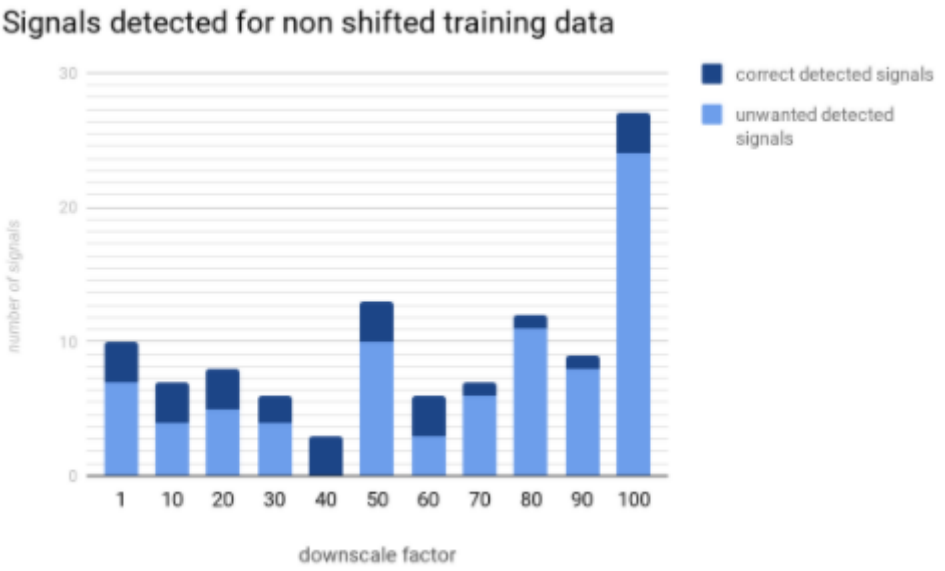

Figure 5.8: Number of detected signals using non shifted template signals

The graph in Figure 5.8 shows that when the template signals are not shifted there are many incorrect instances of detection. One can see that only at the factor of 40 does the detector manage to only detect the 3 instances and no others. When this is compared to Figure 5.9 one can see that with the data shifted the accurate detection rate is much higher, as the detector proved to be accurate for factors between 10 and 40. This is excluding factor 30, in which for an unknown anomaly the detector only detected one instance.

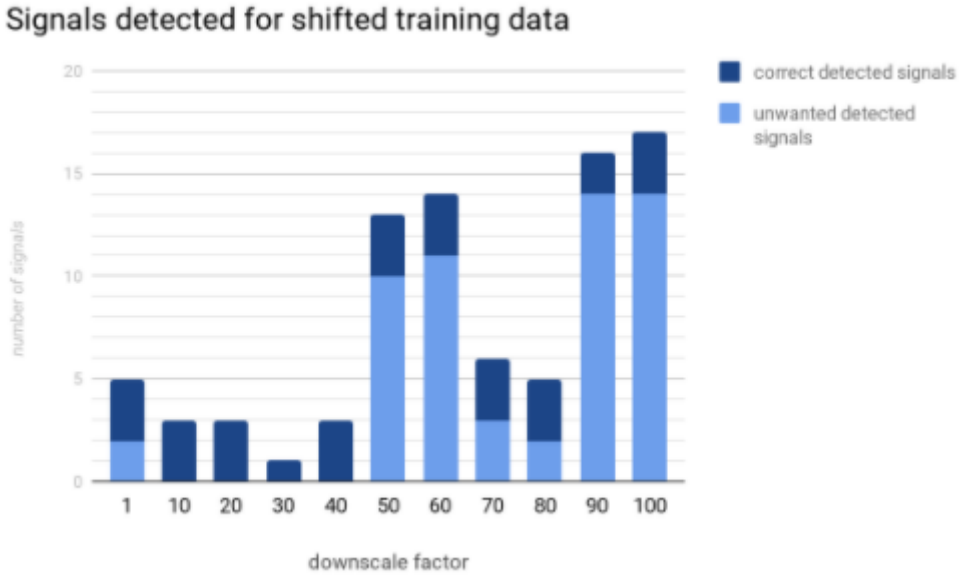

Figure 5.9: Number of detected signals using shifted template signals

As the shifted template signals have proved to be more accurate at detecting the desired Bryde's whale short pulse calls, the shifted versions of the template signals have been implemented and are present in the subsequent tests and simulations.

5.2.2.2 Accuracy tests for different types of normalization

The results for the loudness normalization are not shown in the following figures because it has already been shown as Figure 5.9 on the account that the shifted template signals were implemented moving forward and loudness normalization was implemented in the previous tests.

The results for the implementation of peak normalization, at different downscale factors, when reading through the same testing signal is shown in Figure 5.10. When this is compared to Figure 5.9 it can clearly be seen that loudness normalization results in a far more accurate detection algorithm.

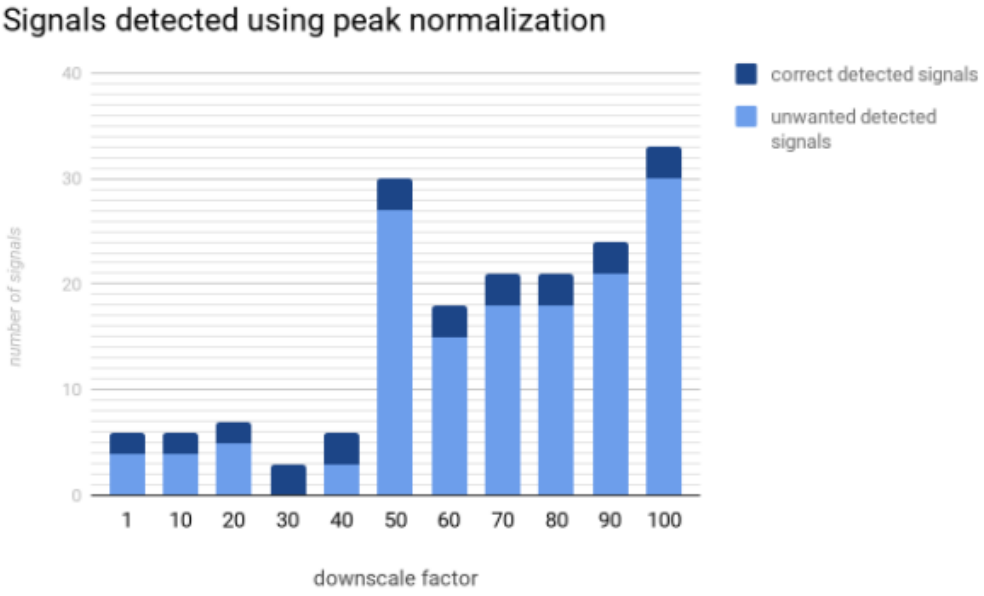

Figure 5.10: Number of detected signals for the detector implementing peak normalization

For the sake of rigor, though it has been shown that loudness normalization results in a more accurate detector than peak normalization, the results for a detector with no normalization implemented are shown in Figure 5.11. It can be seen that without normalization present, the detector is incredibly less accurate. In fact, it should be stated that for most of the downscale factors, the detector produced 45 instances of detection. This is the number of frames in the whole audio signal and as such the detector determined that all frames in the large signal were Bryde's whale short pulse calls. It goes without saying that this is incredibly inaccurate and the importance of the signal normalization process is apparent.

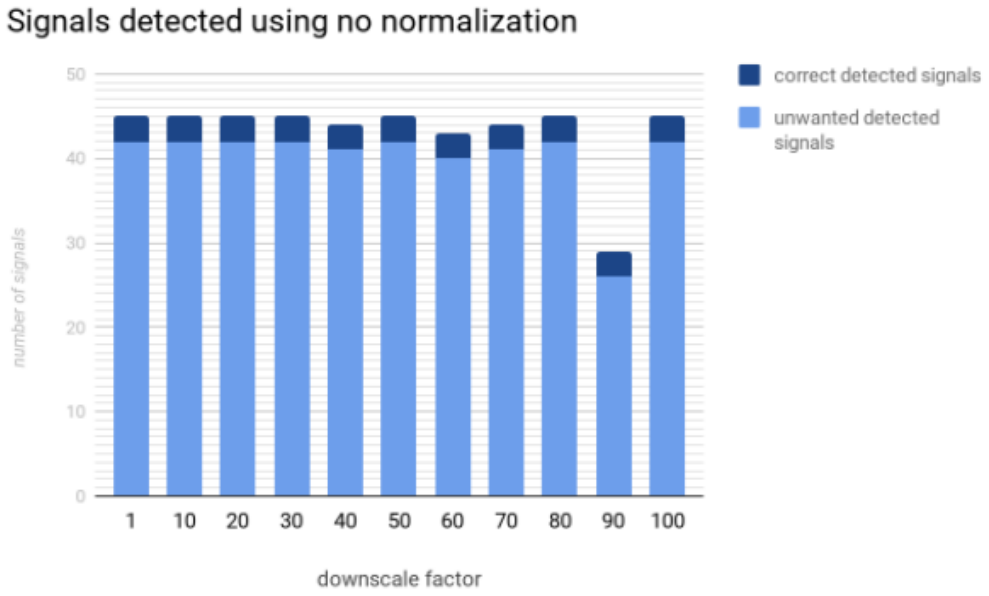

Figure 5.11: Number of detected signals for the detector implementing no normalization

5.3 Time synchronization testing performed during development

As discussed in Chapter 4 the Raspberry Pi system clocks of each sensor are synchronized making use of NTP and the PPS signal provided by the GPS units. The extent to which the units are synchronized has been tested and the results are presented in this section.

The process of testing the Raspberry Pi clock synchronization between units that was devised involves presenting a voltage to a designated GPIO pin on two units from an Arduino microcontroller and displaying the timestamp, accurate to microseconds, at which the voltage was detected at the pin. The difference in these timestamps is then calculated.

Performing this test involved stepping down the 5V logic of the Arduino to the 3.3V logic of the Raspberry Pi. This was performed by implementing a basic potential division circuit comprised of resistors.

The microcontroller was programmed to output a 1ms long pulse every two seconds. The Python code implemented on the Raspberry Pis was designed to wait for voltage input, display the time at which the input was first detected, then wait for the voltage to go low and high again before displaying the next timestamp.

In this test, five timestamps were recorded for each ten minute interval over the span of an hour from the point at which the NTPs were reset. The five timestamps allowed for average differences in timing to be calculated. This test was performed three separate times for the sake of rigorous testing. The timing results are shown in Figure 5.12.

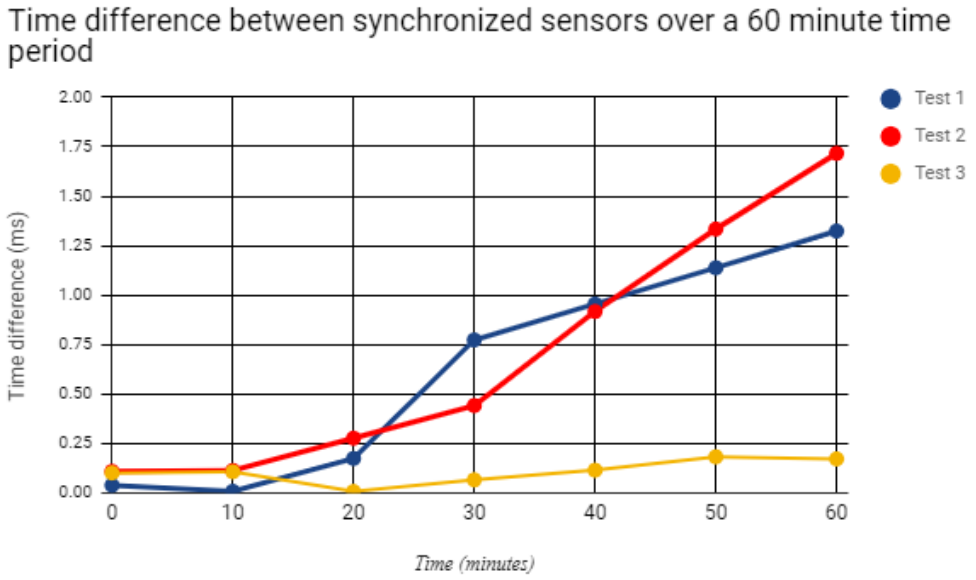

Figure 5.12: The time difference between synchronized sensors' clocks

It can be seen from the data collected and presented that approximately past the 10 minute mark the timing difference begins to increase in test 1 and test 2. However, test 3 contradicts this trend as the timing difference remains more constant in this test. As such this was investigated, it was found that the NTP had marked the PPS and GPS inputs as "false tickers" and were disregarded after approximately 10 minutes. The effects of this occurrence are shown to vary from the data presented in Figure 5.12.

The general trend was found to be that the clocks begin to desynchronize by a margin and with more time spent not synchronized this margin grows. The question that follows this finding would be, to what degree is this margin of error acceptable? It was known when first designing the system that there would be timing errors present from various sources.

In order to determine whether this problem warranted the time it would take to fix it, a TDOA error test was performed making use of the test 1 data as the timing errors present in the system. The resultant TDOA output is shown in Figure 5.13 that follows.

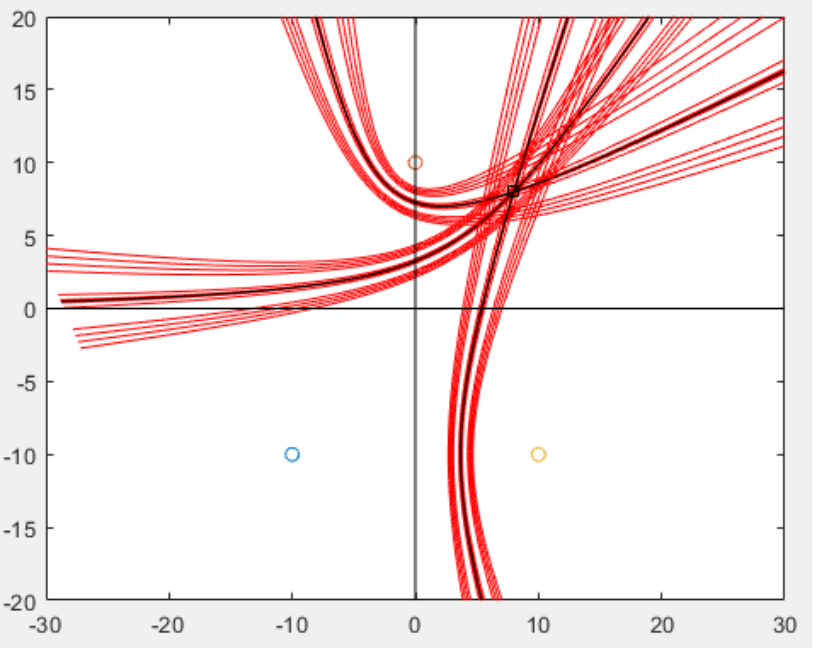

Figure 5.13: TDOA presented in meters with the Test 1 timing errors present

In Figure 5.13 the black lines show the ideal case with no timing errors and the red lines represent the outcome with each timing error incorporated into the system. The scale of this test is set in meters and it can be seen that at a maximum, the timing error introduced a distance error of a few meters. As such when working with a scale in the range of kilometers or coordinate degrees with GPS components that are too only accurate to within a few meters, this timing error could be deemed as acceptable and does not necessarily warrant correcting.

5.4 Testing the TDOA system through a physical simulation

In order to test the accuracy of the TDOA localization system, the system was tested with the use of delayed pulses from an Arduino that had been calculated to yield a previously decided location.

The detection algorithm was set to recognize pulses of 1 ms in width from the Arduino. The Zoom devices from each sensor were attached to pins of the Arduino via modified audio jacks. Once it was determined that the detector algorithm on each sensor could detect the pulses through the Zoom devices, the test was developed further.

The clocks of the Raspberry Pis are synchronized as discussed and tested previously. Additionally, the sensors are given fictitious GPS coordinates that correlate with the chosen time delay values. These time delays between signals are implemented by setting a designated Arduino pin to high voltage for the duration of 1ms, bringing the voltage low again, and then delaying the Arduino by a chosen calculated time value before the process of setting pin voltage to high for each sensor repeats itself.

The results are shown in Figures 5.14 to 5.17 that follow. A chosen point designated by a red square is chosen as the location of interest, the delays correlate to this point, and the sensor locations. The circles represent the calculated location for the signal source by the system. Different circle colours represent sets of data produced from different times that the test has been run, as it has been run three times for each point. The sensors are denoted by black triangles. Additionally, one of the sensors has been shifted an arbitrary distance to account for the asymmetry caused by having free-floating sensors.

The recording of the data is done manually and detections that would be deemed as false positives at each sensor are omitted. However it can still be seen that non-ideal results have not been omitted, these results create outliers that can be observed in the figures. The figures show the data at a scale of which it would be viewed practically, as to fit the sensors in the frame, on the left, and an enhanced scale at which the point clustering can be seen more clearly on the right.

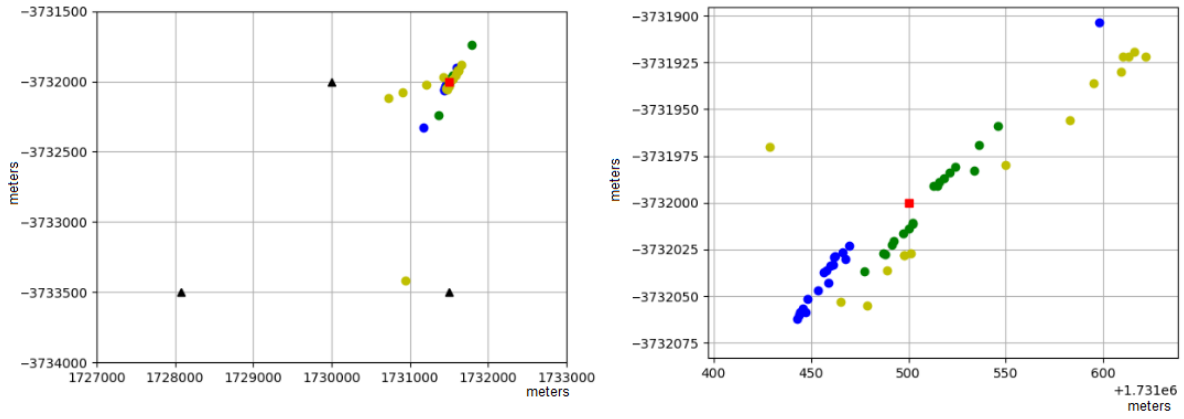

Figure 5.14: TDOA testing results for position 1

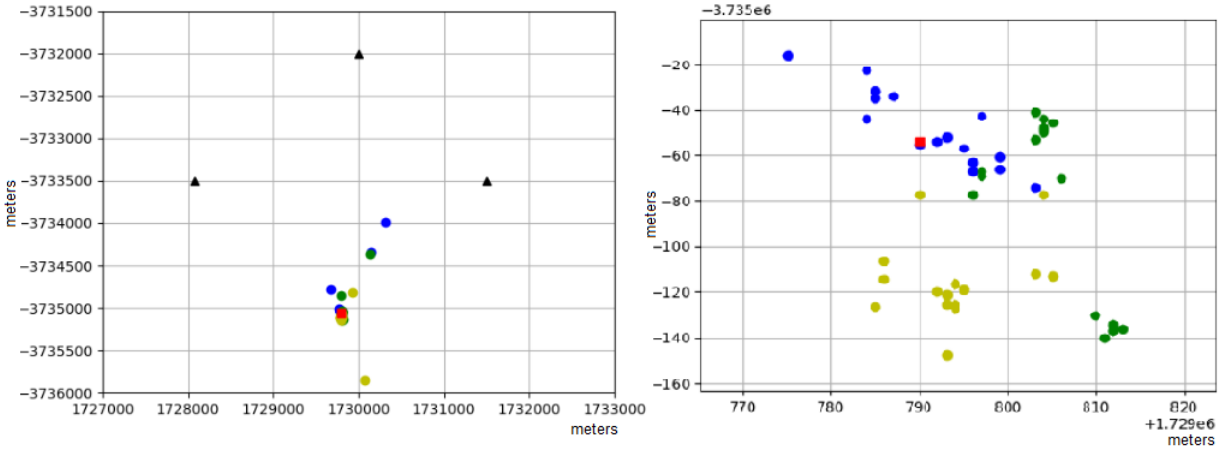

Figure 5.15: TDOA testing results for position 2

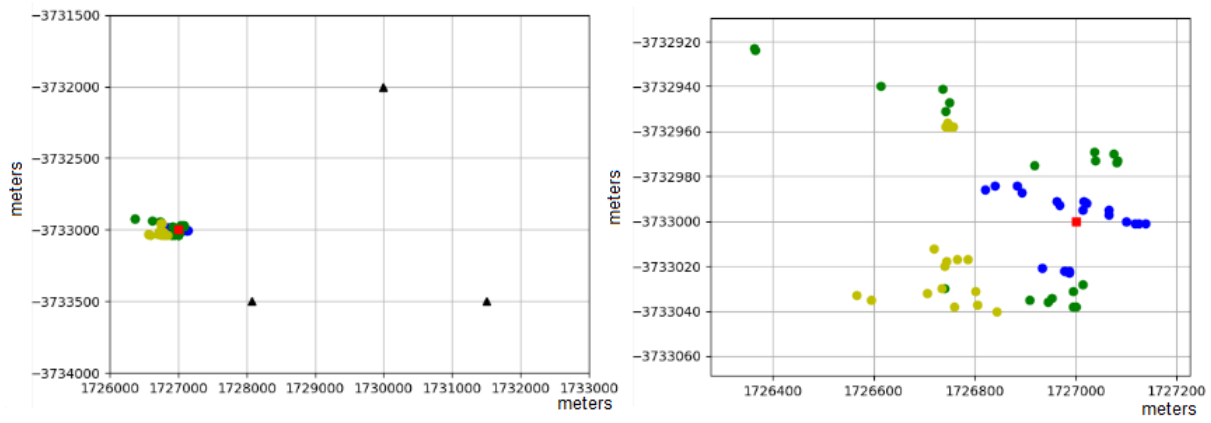

Figure 5.16: TDOA testing results for position 3

Each test produces approximately 20 points of data. As previously stated, these points are recorded manually. This is done to avoid the cluttering of images with multiple hyperbolas. Each point of data is chosen as an approximate point inside the area created by the overlapping hyperbolas with present errors. An example of this is shown in Figure 5.17. The intersection of the lines has been magnified to the extent that it can be seen that they do not intersect at a single point. It is noted that the extent to which an appropriate point is chosen is in terms of meters and does not extend to sub-meter accuracy.

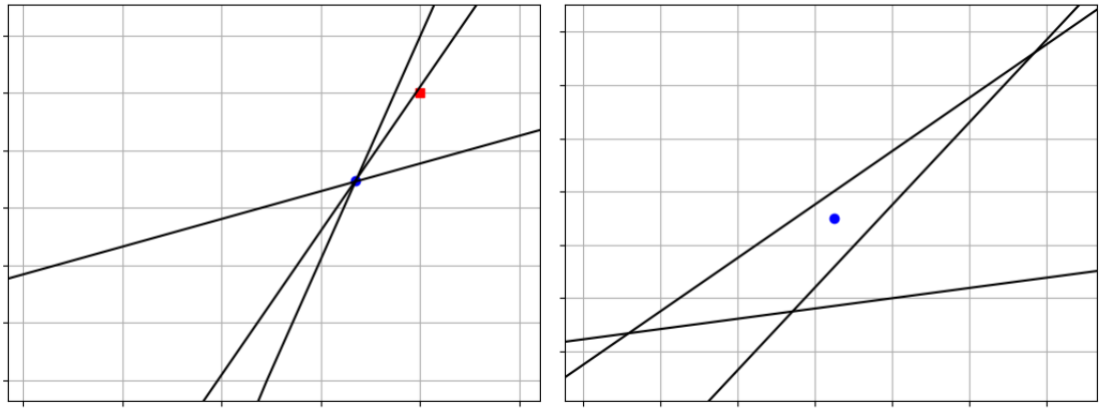

Figure 5.17: Replacing intersecting hyperbolas with an approximate point

Tables 5.1 to 5.3 show the tabulated results for the various TDOA tests. It can be noted that various aspects of the results have been calculated to populate the table. The table is broken up into the test results for each testing position. This entails each test individually as well as grouping all the data together for an overall result for each position. The mean and standard deviations are calculated and used to determine the outlier points. An outlier is defined here as a point that has a distance from the target location greater than three times the standard deviation added to the mean distance [44].

Test indicator	Mean distance from the target location (m)	Standard Deviation distance from the target location (m)	Distance between mean and target location (m)	Number of outliers
Position 1				
Test 1	148	284	3	1
Test 2	58	95	16	1
Test 3	233	350	92	1
Overall	146	276	28	2
Position 2				
Test 1	127	300	115	1
Test 2	87	163	35	1
Test 3	114	179	87	1
Overall	110	223	35	3
Position 3				
Test 1	74	51	5	0
Test 2	176	185	133	0
Test 3	262	62	259	0
Overall	169	140	130	2

Table 5.1: Tabulated results from TDOA testing

The tabulation of the results has been repeated with the outliers omitted. This is done to examine the significance of the effects of these points on the tabulated results. For the calculations the outlier locations are replaced with the previously calculated average point for each test. This is done such that the calculation code written for these tests does not have to be extensively modified.

Test indicator	Mean distance from the target location (m)	Standard Deviation distance from the target location (m)	Distance between mean and target location (m)
Position 1			
Test 1	82	92	69
Test 2	38	56	13
Test 3	161	189	47
Overall	101	141	30
Position 2			
Test 1	74	178	63
Test 2	51	47	24
Test 3	75	50	48
Overall	67	108	14
Position 3			
Test 1	74	51	5
Test 2	176	185	133
Test 3	262	62	259
Overall	152	109	113

Table 5.2: Tabulated results from TDOA testing given the omission of previously determined outliers

It can be determined by observing the results in the previous tables that the few anomalies that created the outliers present in the results greatly affect the results of the system. The overall average distances from the target position to each resultant point for position 1 and position 2 have decreased by approximately 40 meters. Additionally for these positions the standard deviations have decreased by more than 100 meters.

When analyzing the data to determine the cause of the outliers it was found that in each case a single sensor was notably delayed in its detection result. There are many possible causes for this, the system may have stalled in that sensor resulting in a delayed timestamp or the detection may also be a false positive within an ambiguous time difference to a missed detection.

The root mean square error values for all tests are shown in table 5.3 that follows. The RMSE values have been calculated in both scenarios that include and exclude outliers. The RMSE value has been calculated by implementing the equation,

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{5.1}$$

where $(\hat{y}_i - y_i)^2$ denotes the difference in distance between the observed and desired positions and n denotes the number of values included in the calculation [45].

Test indicator	Root mean square error with outliers included (m)	Root mean square error with outliers excluded (m)
Position 1		
Test 1	320	124
Test 2	79	48
Test 3	243	144
Overall	312	173
Position 2		
Test 1	327	192
Test 2	130	49
Test 3	121	51
Overall	251	129
Position 3		
Test 1	90	90
Test 2	180	180
Test 3	153	153
Overall	219	189

Table 5.3: Root mean square error of the localization results

5.5 Sensor tracking and communication systems test

In order to perform testing on the tracking and communication of the units before deploying the sensors, a land test was devised. This test involves setting up one of the sensors in the development laboratory and taking the user-unit in its entirety outside where it can be moved around and that movement can be tracked and seen on screen.

The unit plots its location represented by red circles. The sensor has a constant position represented by a blue circle, its location is transmitted constantly. Additionally, the beginning of the sensors' route traveled is denoted by a square and the end by a triangle. The sensor location allows for a stationary point of reference in this test. The location of the user-unit is determined every 30 seconds.

When plotting the location data for this test, the previous locations of the unit continue to be displayed and subsequent locations are joined to one another with lines to show the route taken by the unit. The plot of the testing route is shown in Figure 5.18. In this figure, the green line represents the route as marked by a commercial Garmin GPS device.

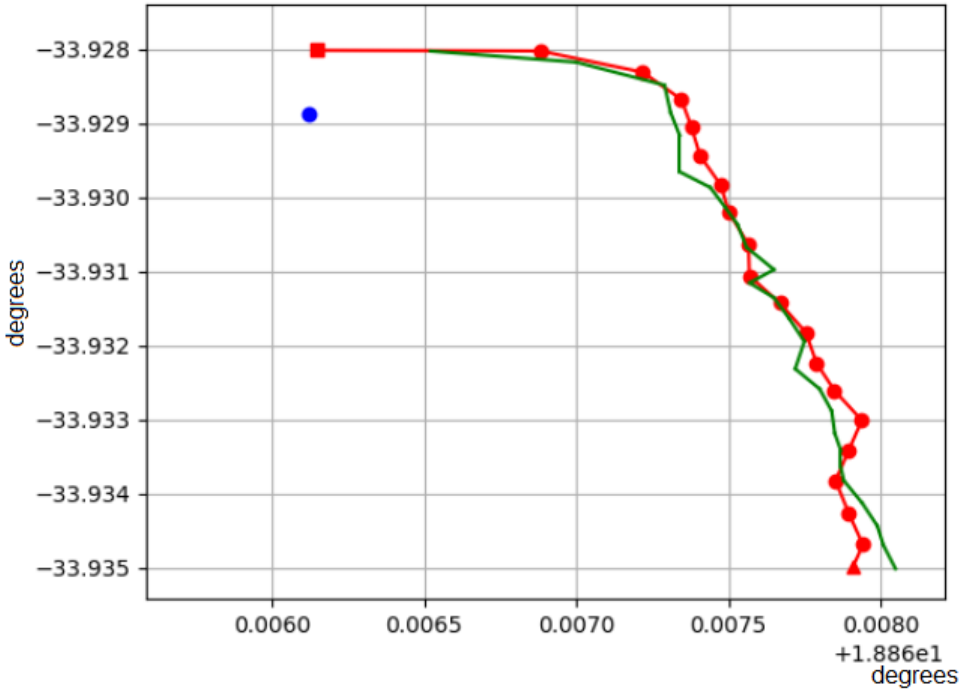

Figure 5.18: Tracking and communication system test result

Figure 5.18 shows the functionality of the tracking system as well as the communication system. The scale of the plot is represented in coordinate degrees. It is noted that the position of the sensor is constantly updating, though there is only one point shown. The location is overwritten each time that the position is updated. This is done because the sensor is stationary and the result would otherwise simply be a tight cluster of overlapping points. This is deemed as unnecessary for this test.

5.6 Conclusion

The testing in this section has shown that the DTW detector algorithm functions within the allotted time frame at an acceptable accuracy. Additionally, it was shown that shifted template signals, when combined with loudness normalized signals, result in the most accurate detector for the given test signal.

The TDOA system test shows the functionality of the detector, TDOA system, and time synchronization between sensors as all these systems needed to function in tandem to produce meaningful data for that test. The results show that the system is accurate to within less than 200 meters in all cases when outliers are omitted except for test 3 at position 3. When examining the results for this test it is seen that the mean distance from the target is large though the standard deviation is much smaller. This indicates data that is clustered closely together though it has been shifted in this situation due to an unknown factor which may be the fault of the microcontroller used to test the system.

Additionally, the functionality of the GPS tracking and communication system has been shown. The results of the designed GPS tracking system do differ from the Garmin device used. It was already known that the GPS sensors would only be accurate within a few meters and this inaccuracy should not have a considerable effect on the overall system, as it has been previously seen to be accurate to within less than 200 meters.

Chapter 6

Conclusion

6.1 Conclusions

The main purpose of this thesis is known to be the documentation of the development of a system capable of localizing vocalizing Bryde's whales through time difference of arrival principles. Though, as noted in the introduction for this thesis the objectives set out to be achieved related more to the functionality of the various subsystems and their integration to achieve the overall goal.

The detector, which makes use of DTW principles, has been shown to be accurate. Specifically, it has been shown to be the most accurate when the training signals have been shifted with zeros placed in the front of the signals, and when loudness normalization has been implemented in preparing the signals before they are transferred to the DTW algorithm. Additionally, the detector has shown to be at its highest level of accuracy, for the given testing signal as seen in Chapter 5, when the signal is down-scaled in resolution to a factor between 10 and 40. This factor would result in the signal frame being 100 to 25 samples in length.

The TDOA system has shown to be functional through controlled in-laboratory tests. This testing not only shows that the TDOA localization system is functional but also the detection system and unit clock synchronization, as these aspects of the system need to work in unison to create meaningful data for this test. In-laboratory testing proved to be beneficial as the desired locations were known and thus the results could be tabulated and compared to the ideal locations.

It was seen that there were anomalies/outliers present in the system, though these were not present in all tests. Additionally, the system proved to be accurate to within a few hundred meters of the desired location. This error in distance would not have come from the clock synchronization aspect of the system, as they were synchronized to within milliseconds and as such this would only account for a few meters worth of distance error. The fault is likely with the Python detector and the frames in the detector, this is elaborated on further in section 6.2.

An important objective for the system developed is that it should be able to function in real-time. This meaning that post-processing should not be necessary. Though the real-time functionality is somewhat unstable due to the combination of false-positive and missed detections, the system still manages to identify the desired signals and produce appropriate timestamps in real-time.

Aside from the errors in the detection of signals, another facet that limits the real-time capabilities of the system is the LoRa communication. Transmissions from the sensors need to be staggered as to not interfere with one another. As a result of this, the detection times are only communicated to the user-unit, that performs and displays the TDOA localization algorithm, at chosen time intervals. Though this logically does not affect the detection times, it does create a delay of several seconds to a minute for the presentation of the TDOA results.

The communication and tracking systems both function reliably. It was found during development that the LoRa communication system was hindered by something in the vicinity of the Stellenbosch engineering faculty but when tested outside, the communication system functioned more reliably. It was shown in Chapter 5, in the GPS tracking test that the GPS tracking system, when compared to a commercial GPS device, has a degree of error. This was known beforehand and was stated by the supplier of the hardware. Though the GPS system still proved to coincide with the commercial GPS tracker to within a margin of meters.

It is noted here that the non-apparent goal of interfering with sea life has been achieved. The system functions passively and as such, no sound signals are introduced to the ocean environment apart from the sounds made by the research vessel. Additionally, the free-floating design of the sensors means that ocean life is physically interfered with as little as possible.

6.2 Recommendations for Future Work

Though Python proved to be incredibly valuable for the building of the system, the transition to C may prove to be more beneficial in terms of speed. This speed improvement may result in more accurate timestamps and therefore more accurate TDOA position data.

Additionally, the system could be moved from Raspberry Pis to microprocessors with embedded code. This hardware could also be combined with an ADC embedded into the hardware of the system.

The cause of the presence of false-positive detections in the system is not known. However, future development of the system should include a process of eliminating these detections. Furthermore development done to distinguish between vocalization instances would increase the reliability of the system, though due to time constraints this had to be overlooked presently.

Further testing should be performed, in particular, sea trials would be incredibly beneficial. The use of in-laboratory testing was prioritized in this thesis for the sake of time as well as meaningful results that can be quantified. The system can be tested in water by emitting the required sound signals underwater at a known location and then using that location as a reference for the tabulation of data. However, this approach was avoided for the sake of this thesis because of the previously mentioned time constraints. In addition to this reasoning, the test described resides in an ambiguous ethical area, as the effects of the introduction of these sound signals into the marine animals' habitat are unknown.

Bibliography

[1] G. Cooke. (2020). "Whales of hermanus", [Online]. Available: `https://www.theroyalportfolio.com/whales-of-hermanus/`.

[2] P. B. Muanke and C. Niezrecki, "Manatee position estimation by passive acoustic localization", *The Journal of the Acoustical Society of America*, 2007.

[3] M. Rosic, M. Simic, and P. Pejovic, "Hybrid genetic optimization algorithm for target localization using TDOA measurements", *4th International Conference on Electrical, Electronic and Computing Engineering, IcETRAN*, 2017.

[4] V. Kunin, M. Turqueti, J. Saniie, and E. Oruklu, "Direction of arrival estimation and localization using acoustic sensor arrays", *Journal of Sensor Technology*, 2011.

[5] V. Kunin, "Sound and ultrasound source direction of arrival estimation", M.S. thesis, Chicago, Illinois, 2010.

[6] C. F. Scola and M. D. B. Ortega, "Direction of arrival estimation: A two microphones approach", M.S. thesis, Karlskrona, Sweden, 2010.

[7] O. Ogundile and J. Versfeld, "Analysis of template-based detection algorithms for inshore Bryde's whale short pulse calls", *IEEE Access*, 2020.

[8] J. C. Brown, "Automatic classification of killer whale vocalizations using dynamic time warping", *The Journal of the Acoustical Society of America*, 2007.

[9] H. Lohrasbipeydeh, T. Dakin, T. A. Gulliver, and A. Zielinski, "Characterization of sperm whale vocalization energy based on echolocation signals", in *2013 OCEANS - San Diego*, 2013, pp. 1–5.

[10] M. Esfahanian, H. Zhuang, N. Erdol, and E. Gerstein, "Comparison of two methods for detection of north atlantic right whale upcalls", Sep. 2015.

[11] P. J. Dugan, A. N. Rice, I. R. Urazghildiiev, and C. W. Clark, "North atlantic right whale acoustic signal processing: Part i. comparison of machine learning recognition algorithms", in *2010 IEEE Long Island Systems, Applications and Technology Conference*, 2010, pp. 1–6.

[12] I. R. Urazghildiiev, C. W. Clark, T. P. Krein, and S. E. Parks, "Detection and recognition of north atlantic right whale contact calls in the presence of ambient noise", *IEEE Journal of Oceanic Engineering*, vol. 34, no. 3, pp. 358–368, 2009.

[13] D. K. Mellinger and C. W. Clark, "Recognizing transient low-frequency whale sounds by spectrogram correlation", *The Journal of the Acoustical Society of America*, 2000.

[14] K. Stafford, C. Fox, and D. Clark, "Long-range acoustic detection and localization of blue whale calls in the northeast pacific ocean", *The Journal of the Acoustical Society of America*, vol. 104, pp. 3616–25, Jan. 1999.

[15] T. S. Körting. (2017). "How DTW (dynamic time warping) algorithm works", [Online]. Available: `https://www.youtube.com/watch?v=_K1OsqCicBY`.

[16] D. Ellis, *Dynamic time warp (DTW) in matlab*, Available at `https://www.ee.columbia.edu/~dpwe/resources/matlab/dtw/` (2012/09/05).

[17] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[18] F. Itakura, "Minimum prediction residual principle applied to speech recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 67–72, 1975.

[19] Wei Chai and B. Vercoe, "Structural analysis of musical signals for indexing and thumbnailing", in *2003 Joint Conference on Digital Libraries, 2003. Proceedings.*, 2003, pp. 27–34.

[20] D. K. Mellinger, "A comparison of methods for detecting right whale calls", *Canadian Acoustics*, vol. 32, no. 2, pp. 55–65, 2004.

[21] Z. Li, D. C. Dimitrova, and T. Braun, "TDOA-based localization system with narrow-band signals", *Conference on Networked Systems (NetSys)*, 2013.

[22] B. Rison, *Time of arrival notes*, EE 389 Mathematical Engineering course notes, 2008.

[23] A. Mueen and E. J. Keogh, *Extracting optimal performance from dynamic time warping*, KDD 2016: 2129-2130.

[24] U. Noreen, A. Bounceur, and L. Clavier, "A study of lora low power and wide area network technology", in *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2017, pp. 1–6.

[25] R. Lie. (2018). "Lora/lorawan tutorial", [Online]. Available: `https://www.mobilefish.com/developer/lorawan/lorawan_quickguide_tutorial.html`.

[26] S. Vatansever and I. Butun, "A broad overview of gps fundamentals: Now and future", in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, 2017, pp. 1–6.

[27] (2016). "Raspberry pi 3 model b datasheet", [Online]. Available: `https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/2020826.pdf`.

[28] (2020). "Zoom u-22", [Online]. Available: `https://zoomcorp.com/en/jp/audio-interface/audio-interfaces/u-22/`.

[29] (2015). "H2a hydrophone user's guide", [Online]. Available: `https://www.aquarianaudio.com/AqAudDocs/H2a_manual.pdf`.

[30] (2018). "Sx1278 lora rf module", [Online]. Available: `https://components101.com/wireless/sx1278-lora-rf-module-features-dimension-datasheet`.

[31] (2019). "Neo m8n mini gps module", [Online]. Available: `https://www.robotics.org.za/NEO-M8N-MOD?search=gps`.

[32] V.-M. Yli-Heikkilä, "Home surveillance with raspberry pi", Bachelor's thesis, 2015.

[33] (2011). "Length of a degree of latitude and longitude calculator", [Online]. Available: `http://www.csgnetwork.com/degreelenllavcalc.html`.

[34] J. Anderson. (2019). "An intro to threading in python", [Online]. Available: https://realpython.com/intro-to-python-threading/.

[35] A. Gorhe. (2017). "Performance gain by writing a c extension in python", [Online]. Available: https://medium.com/@abhijeetagorhe/performance-gain-by-writing-a-c-extension-in-python-12dda9aa8ee6.

[36] (2014). "What is the correct way of normalizing an audio signal?", [Online]. Available: https://www.researchgate.net/post/What_is_the_correct_way_of_normalizing_an_audio_signal.

[37] David. (2014). "How to normalize audio - why do it? everything you need to know", [Online]. Available: https://www.learndigitalaudio.com/normalize-audio.

[38] (2014). "Nmea output description", [Online]. Available: http://navspark.mybigcommerce.com/content/NMEA_Format_v0.1.pdf.

[39] transmitterdan, *Aisconverter*, https://github.com/transmitterdan/aisconverter, 2018.

[40] P. Ram. (2018). "Lpwan, lora, lorawan and the internet of things", [Online]. Available: https://medium.com/coinmonks/lpwan-lora-lorawan-and-the-internet-of-things-aed7d5975d5d#:~:text=LoRa%20and%20LoRaWAN%20Network%20Topology,mesh%20topology%20eg.&text=End%20Nodes%20transmit%20directly%20to,central%20network%20server%20using%20IP..

[41] A. Raj. (2019). "Lora with raspberry pi – peer to peer communication with arduino", [Online]. Available: https://circuitdigest.com/microcontroller-projects/raspberry-pi-with-lora-peer-to-peer-communication-with-arduino.

[42] R. Silva, *Pysx127x*, https://github.com/rpsreal/pySX127x, 2019.

[43] S. Friedl. (2018). "Building a gps time server with the raspberry pi 3", [Online]. Available: http://www.unixwiz.net/techtips/raspberry-pi3-gps-time.html.

[44] J. Brownlee. (2020). "How to remove outliers for machine learning", [Online]. Available: https://machinelearningmastery.com/how-to-use-statistics-to-identify-outliers-in-data/.

[45] J. Moody. (2019). "What does rmse really mean?", [Online]. Available: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e.