# Indoor localization using Local Positioning Systems.

*MSc.*

By

SHIVAN RAMDHANIE




Department of Engineering Mathematics
UNIVERSITY OF BRISTOL & UNIVERSITY OF THE WEST OF ENGLAND

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of MASTER OF SCIENCE in the Faculty of Engineering.

SEPTEMBER 2020

Word count: 11,452

## Abstract

The use of UltraWide Band technology in hard environments for localisation has been addressed in various formats. A household kitchen falls under the case of hard environments with the presence of dynamic obstacle randomly providing No Line of Sight between the receivers and transmitters. It was found that while a receiver is in motion and NLOS occurred, erratic readings were obtained from the sensor making it not suitable for localisation when used in isolation. The overall goal is to achieve viable position estimates that an indoor drone or robotic system can use while operating in this environment. It is shown that the sensor readings can be integrated successfully in a Flight Control Unit and applying a simple sensor fusion algorithm with dead reckoning data is able to decrease the overall positional error of the system.

# Dedication and acknowledgements

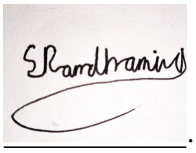
I would like to acknowledge and thank my mother and sister whose continued support over this year has been instrumental in completing this programme and research.

A special thanks to my supervisors, Dr. Wright and Prof. Caleb-Solly whose expertise, profound guidance and support made this achievement possible.

Finally, thanks must go to my previous co-workers and friends at VirtanaTT who inspired me to further my passion for robotics and whose experience equipped me with the necessary tools needed to effectively complete projects like this.

## AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED:.....  ..... DATE: .....**10/09/2020**.....

# NOMENCLATURE

- UAV - Unmanned Aerial Vehicle.
- GPS - Global Positioning System.
- LPS - Local Positioning System.
- FCU - Flight Controller Unit.
- UWB - Ultra-Wide Band.
- TOF - Time of Flight.
- PSoC - Programmable System on Chip.
- NLOS - No Line of Sight.
- TWR - Two Way Ranging.
- I2C- Inter-Integrated Circuit Bus.
- SPI - Serial Peripheral Interface.
- OSH - Open-Source Hardware.
- EKF - Extended Kalman Filter.
- NED - North, East, Down.

# Table of Contents


**Page**

**List of Tables** **xi**

**List of Figures** **xiii**

**1 Introduction** **1**

1.1 Aims & Objectives . . . . . 4

1.2 Background Research . . . . . 6

**2 Literature Review** **7**

2.1 Indoor Localisation Systems . . . . . 7

2.1.1 Passive Systems . . . . . 7

2.1.2 Active Systems . . . . . 8

2.2 Sensor Fusion . . . . . 9

2.3 Pozyx - Behind the Scenes . . . . . 10

2.3.1 Ultra-WideBand (UWB) . . . . . 10

2.3.2 Pozyx Localisation . . . . . 11

2.3.3 Pozyx - Arduino Implementation . . . . . 13

2.3.4 Summary . . . . . 13

**3 Research Methodology** **15**

3.1 Anchor Configurations . . . . . 15

3.2 Operational Parameters . . . . . 24

3.2.1 Loss of Anchor . . . . . 24

3.2.2 No Line of Sight . . . . . 25

3.3 Technical Design . . . . . 27

3.3.1 Flight Controller Unit . . . . . 28

3.3.2 Flight Firmware . . . . . 29

3.3.3 Communication Interface (I2C vs Serial) . . . . . 29

3.3.4 Sensor Fusion: Pozyx Readings and EKF on Ardupilot . . . . . 31

3.4 Summary . . . . . 35

**4 Design of Experiments** **37**
4.1 Benchmarking and Calibration . . . . . 38
4.2 Stationary Tags . . . . . 39
4.3 Localisation of a moving target . . . . . 40
4.4 Improving the Estimates . . . . . 42
4.5 Summary . . . . . 44

**5 Results and Analysis** **47**

**6 Discussion** **49**

**7 Conclusion** **51**
7.1 Conclusions . . . . . 51
7.2 Limitations, Mitigation and Future Work . . . . . 51

**A Appendix A** **53**

**B Appendix B** **55**
B.1 Pozyx I2C Library . . . . . 55
B.1.1 AP_Beacon_PozyxI2C.cpp . . . . . 55
B.1.2 AP_Beacon_PozyxI2C.h . . . . . 61
B.1.3 AP_Pozyx_test.cpp . . . . . 63

**C Appendix C** **65**
C.1 Python EKF implementation . . . . . 65

**D Appendix D** **69**

**E Appendix E** **71**

**F Appendix F** **73**
F.1 Ethical Review Checklist . . . . . 74
F.2 Ethical Review Checklist (cont'd) . . . . . 75

**Bibliography** **77**

# List of Tables


**Table** **Page**

3.1 Anchor locations for each Configuration . . . . . 18
3.2 Statistics of the data recorded for each configuration. . . . . 19
3.3 Comparisons of various FCU's that are commercially available. Source: Ebeid et al. (2018) Page: 2. . . . . 28

4.1 Trajectories used in the tests for data collection. . . . . 38

5.1 Metrics for the Distance from each path for different tests and positions. . . . . 48

# List of Figures

**Figure** **Page**


1.1 The typical setup for an autonomous UAV. . . . . 2

2.1 VISON setup for position of a UAV. (https://aerial-robotics-iitk.gitbook.io/wiki/estimation/setup-with-vicon) . . . . . 8
2.2 Setup used to compare UWB and BLE performance in a museum. Jiménez & Seco (2017) Page: 3 . . . . . 9
2.3 Simplified Block Diagram of the Pozyx tag. . . . . 10
2.4 The experimental setups for both a dog and a student. DeStefano et al. (2019) Page: 1. 12
2.5 Data collected from a student walking in a circle. DeStefano et al. (2019) Page: 2. . . 12

3.1 Bird's eye view of the test environment. . . . . 16
3.2 Sample plots of several anchor configurations. . . . . 20
3.3 Physical layout of the kitchen/Test Area . . . . . 22
3.4 Results obtained when an Anchor was lost. . . . . 25
3.5 The test scenario for No Line of Sight experiment . . . . . 26
3.6 Results obtained when a person provides No Line of Sight. . . . . 27
3.7 Radiolink Pixhawk used for the project . . . . . 29
3.8 Non-GPS loiter solution provided by Ardupilot. Source:https://ardupilot.org/copter/docs/common-pozyx.html . . . . . 30
3.9 Proposed wiring of the I2C interface being developed . . . . . 31
3.10 System being unit tested in the environment. . . . . 32
3.11 Wheeled mobile robot equipped with the Pozyx Tag. . . . . 36

4.1 Path taken by the person to randomly occlude anchors and tag. . . . . 38
4.2 Results for estimation with Line of Sight . . . . . 39
4.3 Results obtained with No Line of Sight and a Stationary Tag. . . . . 40
4.4 Results obtained with No Line of Sight while the tag is moved by a person. . . . . 41
4.5 Results obtained with No Line of Sight while the tag was mounted on the mobile robot. 42
4.6 Line of Sight estimates from EKF using dead reckoning and Pozyx readings. . . . . 43

4.7 Results obtained with No Line of Sight and EKF estimates using both dead reckoning and Pozyx readings. . . . . 44

5.1 Illustration of how a distance metric was developed for analysis. . . . . 48

A.1 Packet transfer in TWR. . . . . 53
A.2 4 Anchor and 1 Tag trilateration in 2D. . . . . 54

CHAPTER 1

# INTRODUCTION

In recent years Unmanned Aerial Vehicle (UAV) usage has grown exponentially becoming common in industry and households Custers (2016). A major part of UAV applications is their ability to localise themselves in the given environment with acceptable precision and accuracy. This is a common requirement in any robotic system but UAV's are often limited by strict payload requirements and therefore have to rely on sensors that are both lightweight and robust. (Mendoza-Mendoza et al. 2020) gives a good summary of physical components that are used in various vehicles but a UAV system, specifically, a quadrotor system can be summarised as follows:

- Rotor build - This section contains parts that should be researched based on the size and physical requirements of the UAV (drone). These include brushless motors, electronic speed controllers and frame size.

- Flight Controller Unit (FCU) - This acts as the motherboard and brain of the quadrotor system. It collates data from various sensors, sends commands to the motors and, if there is a companion computer attached, it collects and sends data to the FCU. Commercial FCU's contain the various control systems and laws required for stable flight and movement. Most have an array of sensors built-in.

- Sensors - These vary from inertial, positioning, barometric and camera. Aside from inertial and barometric sensors that are present in most FCU's, sensors are chosen based on the environment and use of the system.

- Companion computer - In some cases, higher-level processing is required by the system to execute autonomy and a secondary computer is used to do this.

- Transmitter and Receiver - This is used to implement manual control over the drone by a user.

Further research into sensors, we can classify UAV's based on their operating environment, indoors or outdoors. These give rise to two forms of localisation and navigation systems:

- Global Positioning Systems (GPS) - As the name suggests this setup uses GPS as well as other sensors.
- GPS-denied - These systems do not have access to GPS due to their operating environment.

In outdoor applications, GPS provides a reliable and fairly accurate way to localise with the use of several other sensors. However, indoor applications are denied the benefits of GPS and often must use other sensors for the task of localisation. Utilizing a similar concept of triangulation used by GPS's a Local Positioning System (LPS) can be used for indoor environments. (Pozyx Labs 2018*b*) has developed a commercial system that utilizes Ultra-WideBand technology (UWB) with a bandwidth of $\approx 500MHz$.

With indoor environments, users have more control of the environment so an LPS can create a feasible solution for indoor localisation for UAV/robots operating there. The bulk of this research was to integrate a commercial LPS directly into an existing FCU to produce accurate position estimates that can be used for autonomy. The measurements from the LPS was transformed into observations of the state of the UAV and fused with observations from other sensors. This fused pose estimate was then fed into the companion computer for off-board processing.

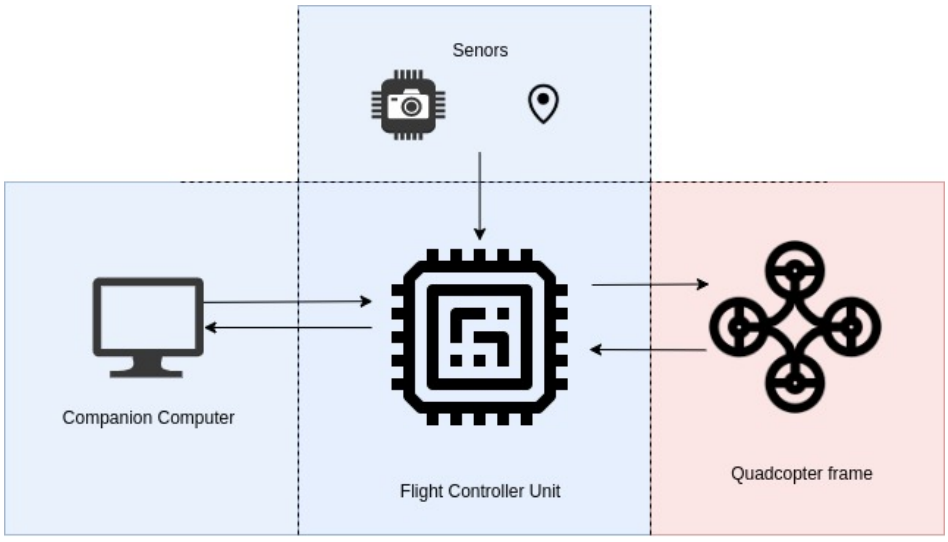



Figure 1.1: The typical setup for an autonomous UAV.

Figure: 1.1 shows a typical setup for UAV. The parts of the system highlighted in blue represent systems that was worked on during the course of this research. The idea is that the system being designed should provide localisation data which should be independent of the rotor

build. These were further scoped in the upcoming sections but it involved doing a quality exercise of the LPS to determine measurement uncertainty and limitations, writing additions or modifying the firmware of the FCU to integrate the LPS and setting up the pipelines for a companion computer to receive the pose estimates and use them. Additionally, the overall aim was to get better pose estimates that would form the basis of any autonomous control and navigation.

## 1.1 Aims & Objectives

Section: 1 briefly touched on what was addressed over the course of this research. Expanding on that, the research would entail the use of the commercial version of a UWB sensor for positioning from Pozyx Labs (2018*b*). At a high level the project was split into three modules that was researched, unit tested and finally integrated. Figure 1.1 highlights the major systems within the project and they are listed:

- The Pozyx LPS providing measurements that will be used in localisation.
- A flight controller collating fusing various observations from sensors to provide a pose estimate.
- A companion computer to visualise and utilise the pose information in a meaningful manner.

From these systems, with the overall aim of indoor localisation, the following objectives were created:

- Evaluation and qualitative analysis of the LPS, documented limitations from previously done work and current physical setups.
- Based on the qualitative analysis and experiments determine the best configuration in a household to place the anchors for the system.
- Use the incoming data from the sensors to produce a suitable measurement/observation model for the pose of the system.
- Relay the data to a flight controller unit via a suitable hardware interface.
- Delve into the firmware of the flight controller and integrate the sensor readings into the codebase.
- Apply sensor fusion algorithms to provide pose estimates either onboard the flight controller or through some other medium.
- Evaluate pose estimates in various scenarios to assess their viability.

These objectives can all be completed without flying the UAV autonomously. Given the current situation and time-frame, it was determined that setting up the pipelines to visualise the localisation in realtime from the companion computer was adequate for the last objective. Furthermore, with the autonomous flight being out of the scope of this project much of the work fell into software engineering to achieve the overall aim. This means investigating into the software libraries and interfaces for the Pozyx sensor, modifying and making additions to the Ardupilot flight stack to integrate the Pozyx sensor with the FCU, and researching the MAVLINK protocol and libraries to use the pose estimates on a Raspberry Pi 3 Model B+ PSoC. To achieve

these objectives a solid software engineering approach was applied with familiarity of Python and C++ programming languages. Additionally, pose estimates should be evaluated under various tests which are not ideal for the LPS system to see how robust the estimator was. A further limitation was due to the lack of accessible hardware and the scope of this research, if pose estimates are unavailable from the flight controller a similar setup should be designed. This was done to closely emulate the results expected from the flight controller unit.

## 1.2 Background Research

Indoor localisation has become a core part of many systems in recent years. These range from robotics, multimedia, logistics and sporting systems. Modern localisation systems can be split into active or passive systems. Active systems require the system being localised to have electronics to either process or send information that will be used to determine location. In passive systems the position is determined based on a variance of a measured signal or image data. As noted by Deak et al. (2012), some of these techniques include, Received Signal Strength Indicator (RSSI), Time of Arrival (TOA), Time Difference of Arrival (TDOA) and Angle of Arrival (AOA). The Pozyx commercial system uses UWB signals with a TOF technique in order to determine the position of a receiver (tag) in a network of transmitters (anchors). Since processing is done on-board the tag, it falls under the active localisation category. Active systems are ideal for indoor localisation systems for UAV's as the positional data can be fed directly to FCU's or companion computers in order to correct pose estimates.

As noted by the producers of Pozyx, the core of the system uses a communication bandwidth of $\approx 500MHz$ this results in pulses of $0.16ns$ wide. Assuming that speed of light is $299792458ms^{-}1$ we get pulses of length $0.04797m$ which is very small and hence robust to noise from reflections. The major factors affecting the performance of the system would be materials that would slow down the signals before they reach the tag. No Line of Sight (NLOS), conductors and changing mediums of travel are noted to affect the performance the most.

With the increasing complexity of FCU's it is possible to do relatively dense calculations in a real-time scenario without delegating them to a separate processing system. This is beneficial to indoor drone systems since they need to be small and maneuverable. A standard FCU comes equipped with several standard communication interfaces (I2C, Serial, SPI) so integrating external sensors is possible. Furthermore, multiple autopilot firmware provides a Hardware Abstraction Layer (HAL) making any sensor integration developed on one unit easily ported to another system. Additionally, onboard libraries contain sensor fusion implementations (Extended Kalman Filter (EKF)) that can combine the Pozyx data and on-board sensor data to provide fairly accurate positional data while in motion. Given EKF's use in localisation, an alternative can be developed that can provide position estimates similar to what can be obtained from the FCU.

CHAPTER 2

# LITERATURE REVIEW

## 2.1 Indoor Localisation Systems

### 2.1.1 Passive Systems

In summary, passive systems do not require the object being tracked to have some of the electronics on them to do positioning. Some examples of passive systems are (Deak et al. 2012):

- Computer Vision and Imaging systems.
- Tactile and contact sensors.
- Attenuation of signals.
- Differential air pressure.

A common example of computer vision-based localisation is to use a setup consisting of multiple cameras in a space trying to detecting a single object. Using the intrinsic and extrinsic properties of each camera it is possible to determine the transform to the object in a given frame with relatively high accuracy. A prime example of this is the commercial VICON motion capture systems. Aerial Robotics IITK (n.d.) shows a drone application in which the UAV is positioned using a VICON motion capture system. Figure 2.1 shows the simplified setup. It should be noted that the UAV must be equipped with a specialised marker that is used to identify it. Furthermore, the positions are fed to a companion computer connected to the autopilot system. The VICON setup provides highly accurate positions and is often used to gather ground truth positional data to compare to other positioning systems. The additional requirements of the VICON systems, however, do not make it feasible for indoor applications.

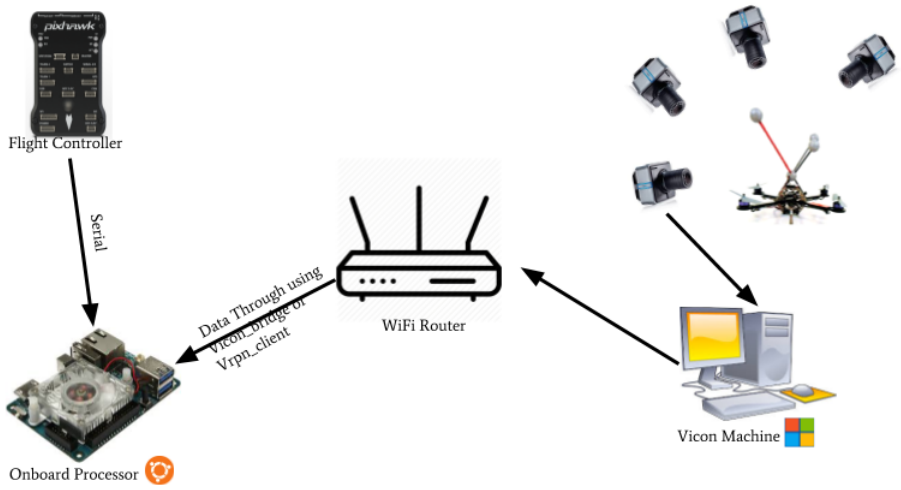



Figure 2.1: VISON setup for position of a UAV. (`https://aerial-robotics-iitk.gitbook.io/wiki/estimation/setup-with-vicon`)

### 2.1.2 Active Systems

In contrast to passive systems, active systems have the object being positioned equipped with electronics. Many indoor localisation techniques use this and some examples are (Deak et al. 2012)

- Radio-frequency identification
- UWB
- Wireless Local Area Network
- Bluetooth Low Energy (BLE)

Many of these setups use an anchor and tag configuration. The tag receives signals from multiple anchors and triangulates the tag.

An approach using and comparing UWB and BLE was developed by Jiménez & Seco (2017) to do localisation in a museum. Both methods are combined with a dead reckoning system to improve accuracy. Six paintings are equipped with both a BLE and UWB tag. The test setup first did a calibration where both sensors were placed at fixed points in the museum with a clear line of sight to each tag. From initial ranging performance, the UWB setup was shown to perform better with a distance variance of $\pm 0.4m$ while the BLE setup had errors of over $10m$. It is noted by the authors that a ranging approach with BLE is challenging since it uses an RSSI method

but can match the accuracy of the UWB setup when combined with a dead reckoning system after some initial steps by the user.

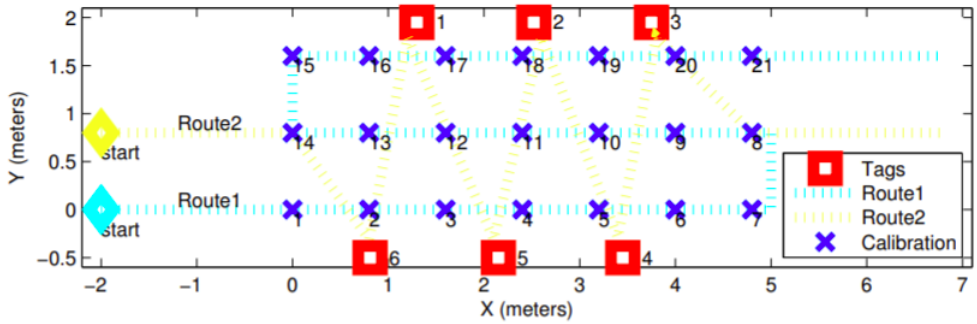



Figure 2.2: Setup used to compare UWB and BLE performance in a museum. Jiménez & Seco (2017) Page: 3

## 2.2 Sensor Fusion

Appendix: A describes the basic operation of the Pozyx system which is based on Time of Flight (TOF) calculations. With a tag it is possible determine a rough estimate of the position in a given reference frame. Additionally, the tag has an Inertial Measurement Unit (IMU), consisting of an Accelerometer, Gyroscope, Magnetometer and an Altimeter. These are used in one of the operational modes of the Pozyx system in order to improve accuracy. Sensor fusion combines multiple sources of data in order to get a fairly accurate estimate of the pose of the system, The measurements from the tag can be combined with the sensors onboard an FCU in order to achieve this. The de-facto sensor fusion technique is called the Extended Kalman Filter (EKF). Chadaporn et al. (2014) documented a useful description and example of how the EKF works. Algorithmically, the EKF is a recursive process using predictions based on the dynamics of the vehicles and updating the estimate based on these predictions and measurements from various sources. The major requirement for the EKF is that the process model and the measurement model are differentiable. The steps for the EKF are as follows:

1. Provide and initial estimate for the state, $\hat{x}_k^+$, and the prediction error, $P_k^+$.
2. Compute the Kalman gain, $K_k = P_k^+ H_k^T (H_k P_k^+ H_k^T + R)^{-1}$
3. Update the estimate with measurement $z_k$, $\hat{x}_k = \hat{x}_k^+ + K_k(z_k - h(\hat{x}_k^+))$
4. Update the prediction error, $P_k = (I - K_k H_k) P_k^+$
5. Project the state ahead, $\hat{x}_{k+1}^+ = f(\hat{x}_k, u_k, w)$
6. Project the Prediction error ahead, $P_{k+1}^+ = A_k P_k A_k^T + Q_k$

7. Start from Step 2 for the next time step.

Where K represents the Kalman gain, R is the Measurement Noise Covariance Matrix, Q is the Process Noise Covariance matrix. Tsai (1998) document work for localising a mobile robot using ultrasonic measurements. Although, the system was a wheeled mobile robot instead of a UAV, the calculations using the TOF measurements and dead reckoning data as well as the use of an EKF to improve the pose estimates are similar to what is trying to be achieved in this current project. A notable contribution of the work is the use of a voting scheme to obtain the best 'observation' for the robot's orientation before being fed into the EKF algorithm.

## 2.3 Pozyx - Behind the Scenes

This research project used a commercially available, active UWB sensor to aid in localisation of an indoor UAV system.

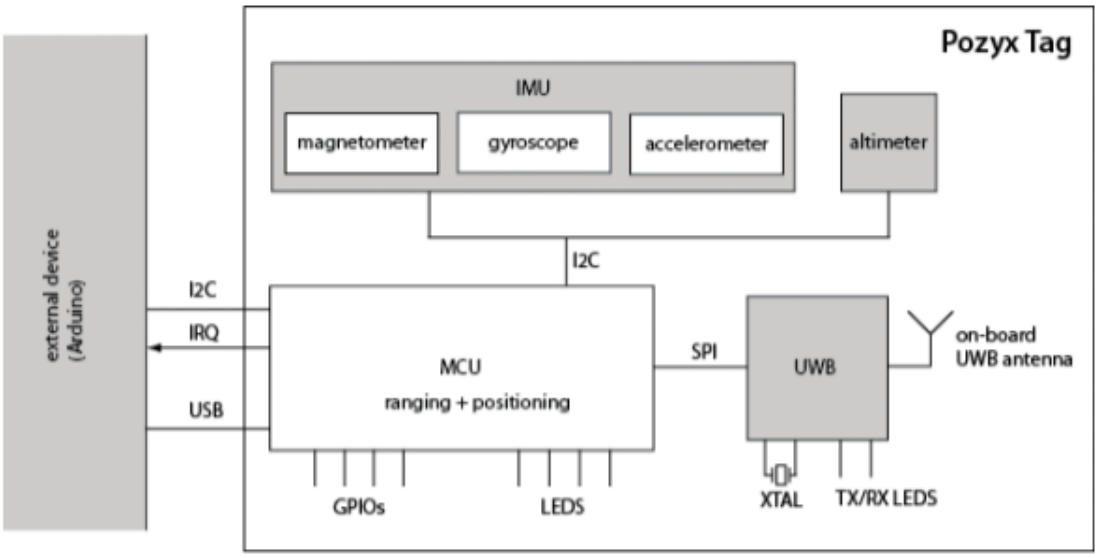



Figure 2.3: Simplified Block Diagram of the Pozyx tag.

### 2.3.1 Ultra-WideBand (UWB)

The core of the Pozyx system operates using a UWB approach. UWB is a short-range, low energy, high bandwidth communication radio technology. Radio waves travel at the speed of light ($c = 299792458ms^{-1}$) so using a TOF approach the range between a tag and an anchor can be obtained simply by:

$$d = c * TOF$$

Knowing the position of each anchor in a given reference frame, Mimoune et al. (2019) discussed a method to use raw range readings in order to determine the position of the tag. The positions can be described by the following system of equations:

$$\begin{bmatrix} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2=d_1^2 \\ \vdots \\ (x-x_n)^2+(y-y_n)^2+(z-z_n)^2=d_n^2 \end{bmatrix} \tag{2.1}$$

Where $n$ represents an index of an anchor and $(x,y,z)$ represents the position of the tag. This was converted into matrix form of **A.x = B**:

$$\begin{bmatrix} 1-2x_1-2y_1-2z_1 \\ \vdots \\ 1-2x_n-2y_n-2z_n \end{bmatrix} * \begin{bmatrix} x^2+y^2+z^2 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1^2-x_1^2-y_1^2-z_1^2 \\ \vdots \\ d_n^2-x_n^2-y_n^2-z_n^2 \end{bmatrix} \tag{2.2}$$

The position of the tag can be calculated as:

$$\hat{x} = (A^T A)^{-1} A^T B$$

From the algorithms and work presented we can confirm that the Pozyx system can achieve an accuracy of $\pm 10cm$ in standard environments with LOS. The work confirms that LOS of the anchors to tag is a major factor in accuracy and this will be taken into account when anchors are being placed in the research. Furthermore, the work mentioned in this section proposes an algorithm with raw range readings in order to do localisation, my research will focus on the integration of the Pozyx tag with a standard FCU, so the pose data coming directly from the tag can be used. The pose can be obtained via two modes: 1). A pure Two Way Ranging (TWR) Approach or 2.) A tracking approach using a Kalman prediction filter in addition to the TWR pose.

Additional work done by Di Pietra et al. (2019) also performs an evaluation of UWB systems for indoor applications. The work compares several commercial UWB systems available to consumers and their viability in ranging and pose estimation. The sensors are treated as black-box systems and the proprietary algorithms for each system were used. The results documented in this paper show useful steps for a primary evaluation of multiple anchor configurations in a given space.

### 2.3.2 Pozyx Localisation

In contrast to the raw readings obtained in the previous work, the Pozyx system has been used in several localisation systems, both indoors and outdoors. Experiments done by DeStefano et al. (2019) use the Pozyx system in multiple scenarios for the purpose of education. The setup is simple and the two-dimensional positions, x and y, form the basis of graphs that can be used to

determine average velocities from linear interpolation of a position versus time graph. Figures 2.4 and 2.5 show both how the physical setup looked and some of the data collected. Although the UWB positioning system has advantages over several other systems the researchers note that using the Pozyx system in this scenario is not ideal for several reasons. The accuracy of the system ($\pm 10cm$) leads to large variance in instantaneous velocity when calculations are done on a point to point basis.

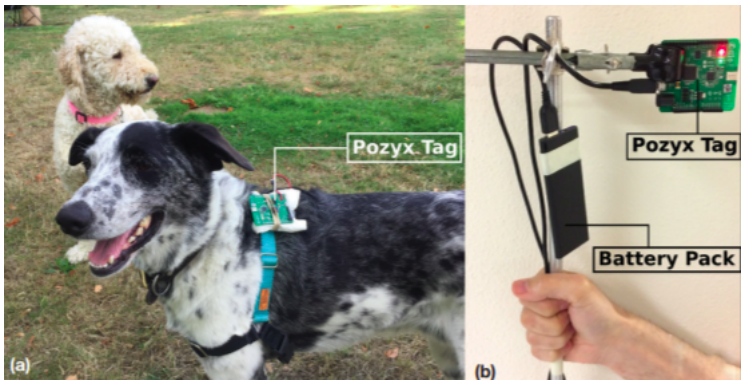



Figure 2.4: The experimental setups for both a dog and a student. DeStefano et al. (2019) Page: 1.

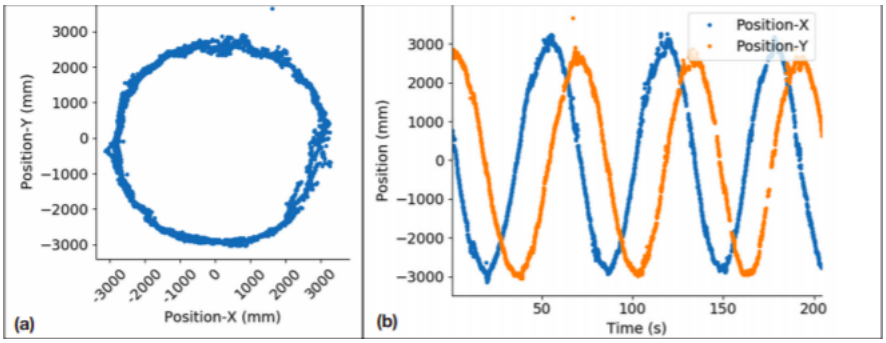



Figure 2.5: Data collected from a student walking in a circle. DeStefano et al. (2019) Page: 2.

Conceição et al. (2017) present a method for using the Pozyx in an outdoor environment. The systems operated in two ways: 1.) Using raw range values from the Pozyx sensor and using an EKF externally for pose estimates. 2.) The pose estimates coming from the Pozyx sensor itself. A modified version of the EKF is described in the first scenario. An additional check is done for outliers. Outliers can be described as measurements that do not fit the pattern of the previous measurements. If an outlier is detected it is simply not considered in that time step and the previously calculated values are used instead. The test environment was an outdoor scenario with a range error characterisation test which allowed for optimal anchor placement and a suitable testbed for implementing multiple positioning algorithms. A similar methodology would be adapted for this work with a focus on indoor limitations and the presence of dynamic obstacles (humans).

### 2.3.3 Pozyx - Arduino Implementation

Before continuing it should be noted that Ardupilot (2019) has addressed the idea of combining the Pozyx system with a Flight controller using the Ardupliot firmware. The implementation uses the Pozyx tag's compatibility with the Arduino UNO R3 or R2 pin layout. The Pozyx Arduino library is used to gather the relevant information from the Pozyx tag and then send it via serial to the FCU. This research aims to bridge this gap in the hardware and remove the need for the Arduino UNO for indoor navigation. The major driving force of this is to minimize the amount of extra hardware that should be mounted on an indoor UAV.

### 2.3.4 Summary

From an overview of work done with UWB technology and the Pozyx system, we can see it is a well-researched area in terms of evaluating the performance in static scenarios with few obstacles. However, there seems to be a gap with these systems being tested and localised in households where dynamic obstacles can be present in the area. Furthermore, a major objective of this research would be to integrate the Pozyx system with an FCU and utilize the sensor fusion systems onboard and evaluate the accuracy of the pose estimates. To that end, a similar approach taken by Di Pietra et al. (2019) would be used to first find an optimal anchor configuration and then evaluate it similar to experiments carried out by Conceição et al. (2017).

CHAPTER 3

# RESEARCH METHODOLOGY

From the literature there is a clear lack of results for localisation systems using the commercial Pozyx sensor network with drones in a household environment. To address the aims and objectives stated in 1.1, it is proposed that a lightweight onboard localisation be developed to check the feasibility of minimal hardware solution for an indoor drone. This would entail connecting a Pozyx tag directly to an FCU and use the pose localisation system existing already. To close the loop the pose information should be available in some format that can be used for autonomous control. Furthermore, the system should be tested in a practical environment hence the Pozyx anchor network was set up in a kitchen. A kitchen represents one of the highest traffic areas in a household and it contains various materials that will make raw readings from a UWB system noisy and inaccurate. As a kitchen will contain both dynamic and static, obstacles multiple anchor configurations must be tested in order to find optimal anchor positions in this given environment.

## 3.1 Anchor Configurations

Noting work from Di Pietra et al. (2019) and the setup procedures from Pozyx Labs (2018*b*) it is important to have at least 4 anchors setup in non-planar orientations for the best positioning results. To determine a suitable anchor configuration in the given space, a tag was placed at a fixed known point in a given reference frame and the mounted locations of the 4 anchors were varied on each wall of the kitchen in order to prevent planar configurations and ambiguity when the tag calculates its position. Figure 3.1 shows a simplified layout and toybox configuration of the tag and anchors. Grey areas represent areas in the space that is impossible to traverse, red are known obstacles that interfere and attenuate the UWB signal, blue represent anchors, green

the tag, cream is partially obstructed and the rest of the area is fully traversable. Before each run and data collection, the tag was manually configured with the location of each of the anchors in the reference frame of the kitchen and recorded for at least 5 seconds. The tag was triggered to calculate the position repeatedly within the timeframe. The function call takes $\approx 70ms$ so data was recorded at a frequency of $\approx 14.28Hz$. With a tag position of (1480,2330,980)mm, the following statistics were calculated. In addition to average error and standard deviation, kurtosis and skew metrics were added to show the number of outliers and symmetry of the recorded data. The error metrics are important to give an idea of how noisy the overall data is while kurtosis and skewness give an indication if the noise follows a uniform distribution which would be suitable for state estimators. From table 3.2 we can see that configuration 12 gives acceptable error metrics and reasonable skew and kurtosis making it the best out of the configurations tested. To further support this Figure 3.2 shows several results obtained with several configurations and it can be seen that configuration 12 produced the best results with minimal noise and measurement outliers as well as it was centered around the ground truth position of the tag.

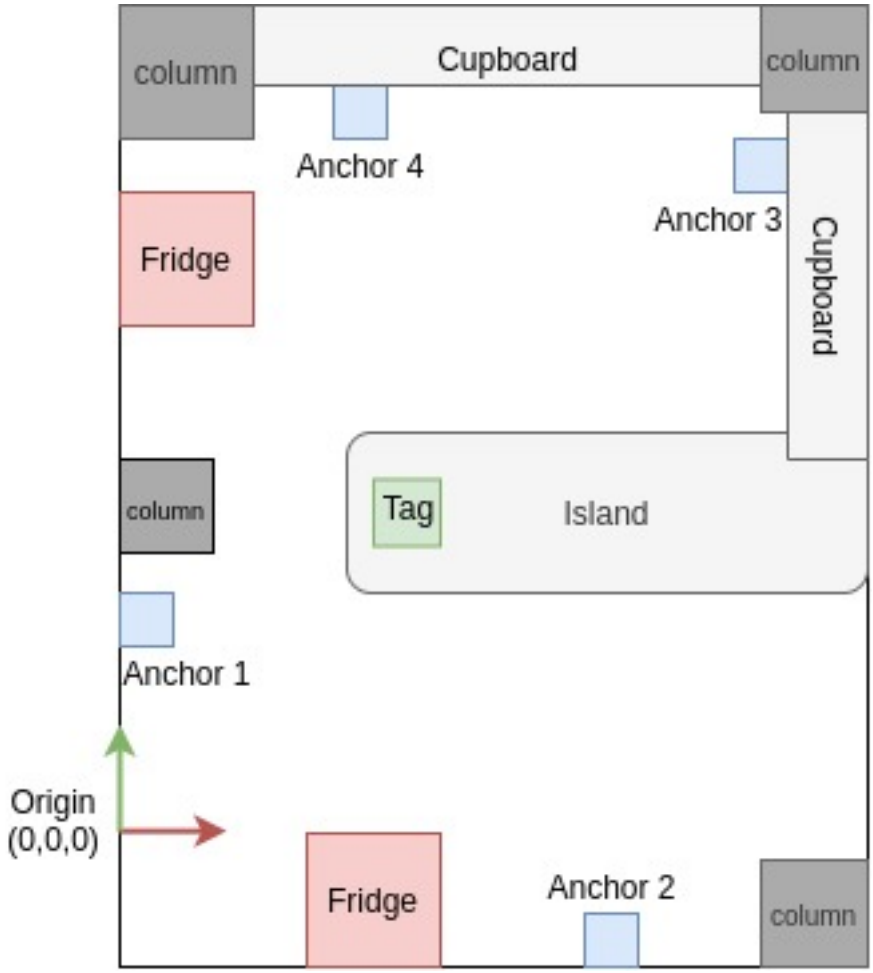



Figure 3.1: Bird's eye view of the test environment.

| Config | Anchor positions (mm) $\begin{pmatrix} Anchor1(x,y,z), \\ Anchor2(x,y,z), \\ Anchor3(x,y,z), \\ Anchor4(x,y,z) \end{pmatrix}$ |
|---|---|
| 1 | $\begin{pmatrix} (0,0,1115), \\ (3680,-405,1550), \\ (3655,4080,1906), \\ (270,4465,2090) \end{pmatrix}$ |
| 2 | $\begin{pmatrix} (0,665,1115), \\ (2995,-405,1889), \\ (3655,4080,1906), \\ (270,4465,2090) \end{pmatrix}$ |
| 3 | $\begin{pmatrix} (0,665,1115), \\ (2995,-405,1889), \\ (3655,4080,1906), \\ (1526,4559,837) \end{pmatrix}$ |
| 4 | $\begin{pmatrix} (0,665,1115), \\ (2995,-405,1889), \\ (3655,4080,1906), \\ (1070,5170,491) \end{pmatrix}$ |
| 5 | $\begin{pmatrix} (0,665,1115), \\ (2995,-405,1889), \\ (3960,3368,2304), \\ (1070,5170,491) \end{pmatrix}$ |
| 6 | $\begin{pmatrix} (0,665,1115), \\ (2737,-410,1913), \\ (3655,3598,1668), \\ (1187,4760,590) \end{pmatrix}$ |
| 7 | $\begin{pmatrix} (0,665,1115), \\ (2737,-410,1913), \\ (3655,4529,1777), \\ (1187,4760,590) \end{pmatrix}$ |
| 8 | $\begin{pmatrix} (0,645,1214), \\ (2737,-410,1913), \\ (3651,4120,1853), \\ (1066,4760,491) \end{pmatrix}$ |

| | |
|---|---|
| 9 | $\begin{pmatrix}(0,645,1214),\\(2737,-410,1913),\\(3970,3063,2320),\\(1066,4760,491)\end{pmatrix}$ |
| 10 | $\begin{pmatrix}(0,645,1214),\\(2737,-410,1913),\\(3651,3550,1810),\\(1066,4760,491)\end{pmatrix}$ |
| 11 | $\begin{pmatrix}(0,645,1214),\\(2737,-410,1913),\\(3651,3550,1810),\\(1591,4450,1775)\end{pmatrix}$ |
| 12 | $\begin{pmatrix}(0,645,1214),\\(2737,-410,1913),\\(3651,4120,1853),\\(1591,4450,1775)\end{pmatrix}$ |

Table 3.1: Anchor locations for each Configuration

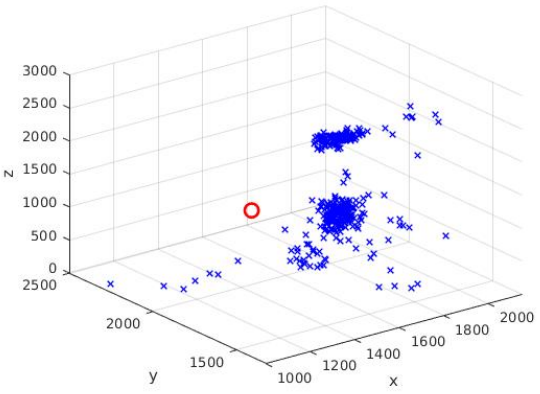
| Config | Avg. Error | Std. Deviation $\begin{pmatrix}x,\\y,\\z\end{pmatrix}$ | Kurtosis | Skewness |
|---|---|---|---|---|
| 1 | 342.5906 | $\begin{pmatrix}183.1707,\\205.3426,\\972.6239\end{pmatrix}$ | $\begin{pmatrix}2.349,\\1.5967,\\1.415\end{pmatrix}$ | $\begin{pmatrix}-0.4083,\\-0.3522,\\0.4551\end{pmatrix}$ |
| 2 | 173.5938 | $\begin{pmatrix}43.2345,\\45.6897,\\133.8616\end{pmatrix}$ | $\begin{pmatrix}21.8923,\\10.1765,\\119.1344\end{pmatrix}$ | $\begin{pmatrix}-2.8672,\\-0.3453,\\8.9025\end{pmatrix}$ |
| 3 | 203.8502 | $\begin{pmatrix}128.4693,\\118.3216,\\209.1637\end{pmatrix}$ | $\begin{pmatrix}14.2955,\\11.2045,\\2.9124\end{pmatrix}$ | $\begin{pmatrix}-2.579,\\-0.3920,\\0.4055\end{pmatrix}$ |
| 4 | 85.8562 | $\begin{pmatrix}40.2197,\\40.6081,\\66.2120\end{pmatrix}$ | $\begin{pmatrix}6.7558,\\7.5180,\\3.4722\end{pmatrix}$ | $\begin{pmatrix}-0.4260,\\-0.5344,\\-0.1242\end{pmatrix}$ |
| 5 | 64.8616 | $\begin{pmatrix}65.3883,\\53.8458,\\78.5751\end{pmatrix}$ | $\begin{pmatrix}13.3707,\\13.2178,\\11.0111\end{pmatrix}$ | $\begin{pmatrix}-0.6016,\\0.3745,\\0.0461\end{pmatrix}$ |

| | | | | |
|---|---|---|---|---|
| 6 | 189.933 | $\begin{pmatrix} 40.8435, \\ 32.6482, \\ 58.5618 \end{pmatrix}$ | $\begin{pmatrix} 11.8957, \\ 11.8924, \\ 11.092 \end{pmatrix}$ | $\begin{pmatrix} 0.7516, \\ -0.0100, \\ -0.1322 \end{pmatrix}$ |
| 7 | 100.6929 | $\begin{pmatrix} 75.8269, \\ 72.0940, \\ 41.9398 \end{pmatrix}$ | $\begin{pmatrix} 26.0285, \\ 17.4295, \\ 7.2328 \end{pmatrix}$ | $\begin{pmatrix} -3.1681, \\ -0.8384, \\ -0.7151 \end{pmatrix}$ |
| 8 | 138.1276 | $\begin{pmatrix} 26.2243, \\ 21.8750, \\ 70.6189 \end{pmatrix}$ | $\begin{pmatrix} 4.6048, \\ 52.2374, \\ 50.9467 \end{pmatrix}$ | $\begin{pmatrix} -0.4858, \\ 5.0752, \\ 5.1757 \end{pmatrix}$ |
| 9 | 258.296 | $\begin{pmatrix} 165.7105, \\ 95.081, \\ 475.0623 \end{pmatrix}$ | $\begin{pmatrix} 2.2833, \\ 1.7939, \\ 2.3406 \end{pmatrix}$ | $\begin{pmatrix} -0.6982, \\ 0.2746, \\ 0.7517 \end{pmatrix}$ |
| 10 | 235.348 | $\begin{pmatrix} 116.1306, \\ 89.9868, \\ 420.8322 \end{pmatrix}$ | $\begin{pmatrix} 3.6670, \\ 4.7058, \\ 3.8066 \end{pmatrix}$ | $\begin{pmatrix} 1.5079, \\ -1.7613, \\ -1.5615 \end{pmatrix}$ |
| 11 | 223.4468 | $\begin{pmatrix} 120.9674, \\ 79.8602, \\ 409.8192 \end{pmatrix}$ | $\begin{pmatrix} 3.389, \\ 4.6663, \\ 3.1988 \end{pmatrix}$ | $\begin{pmatrix} 1.1099, \\ -1.5397, \\ -1.2038 \end{pmatrix}$ |
| 12 | 111.8493 | $\begin{pmatrix} 27.568, \\ 17.7124, \\ 64.4278 \end{pmatrix}$ | $\begin{pmatrix} 4.8522, \\ 7.4335, \\ 6.2489 \end{pmatrix}$ | $\begin{pmatrix} -0.0239, \\ -0.8535, \\ 3.7132 \end{pmatrix}$ |

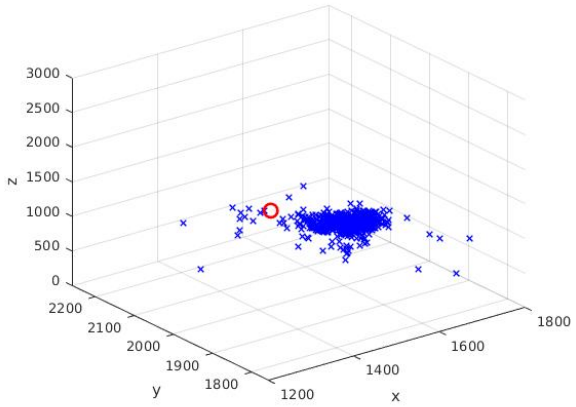
Table 3.2: Statistics of the data recorded for each configuration.

Figure: 3.3 shows the physical locations of the Pozyx anchors with respect to obstacles in the environment. Some things to note are:

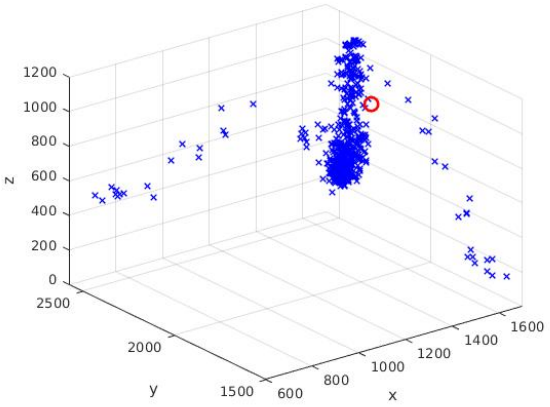
- The walls are made from concrete and drywall and there are no underlying materials that affected the performance of the anchors.
- The fridges have dimensions of $(600 * 600 * 1750)$mm.
- The corner of the island is located at $(1525, 2106, 918)$ and it has a dimension of $(2450 * 900 * 918)$mm.
- Above the island is considered traversable while beneath it will be considered to be impossible to traverse to simplify future experiments.



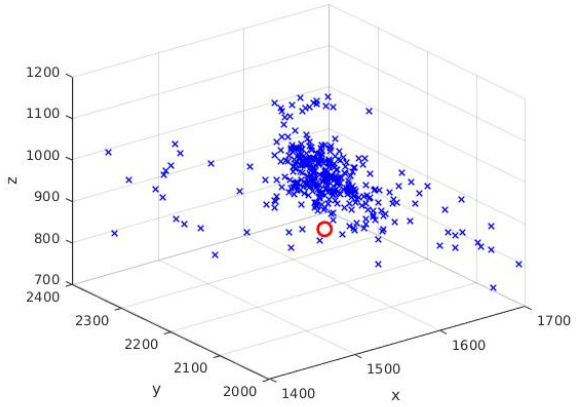

(a) Plot of Config 1



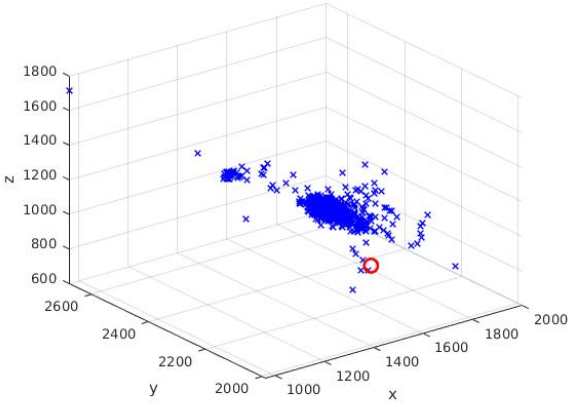

(b) Plot of Config 2



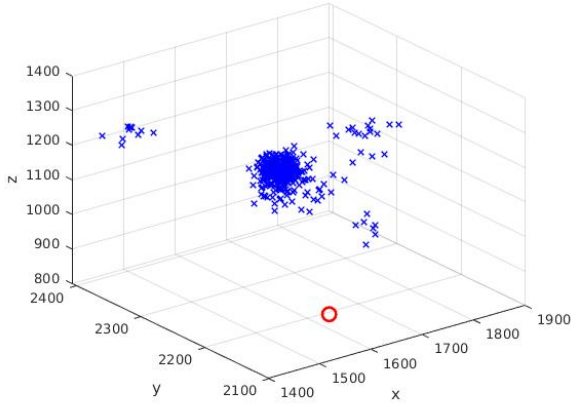

(c) Plot of Config 3




(d) Plot of Config 4

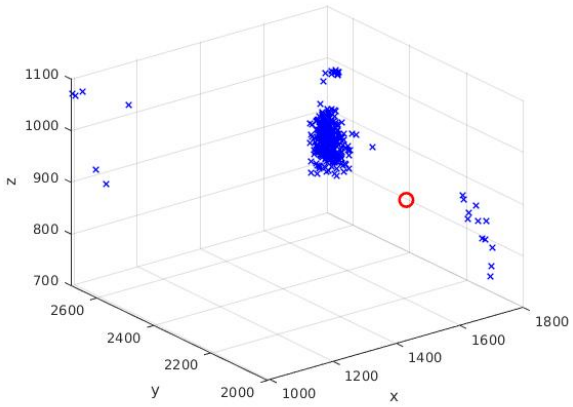



(e) Plot of Config 5

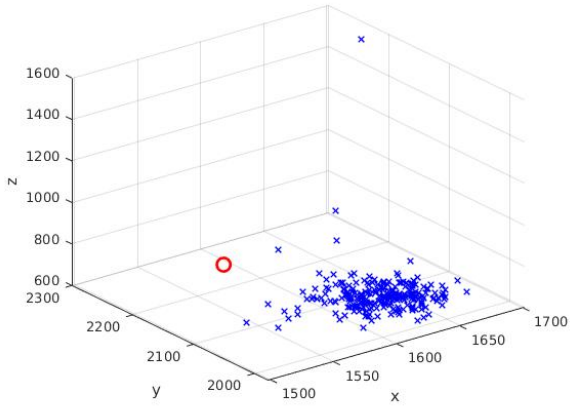



(f) Plot of Config 6

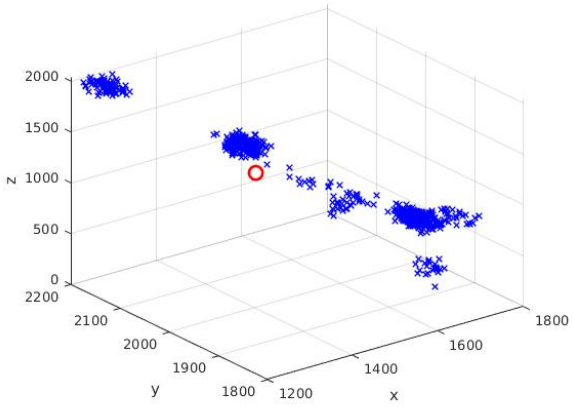
Figure 3.2: Sample plots of several anchor configurations.



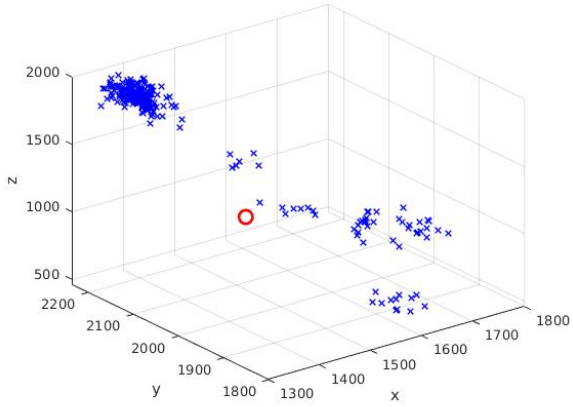

(g) Plot of Config 7



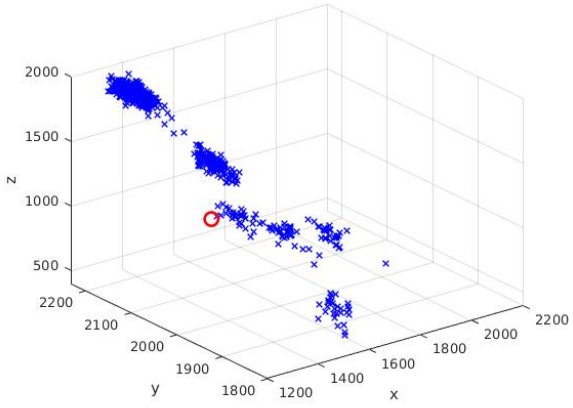

(h) Plot of Config 8



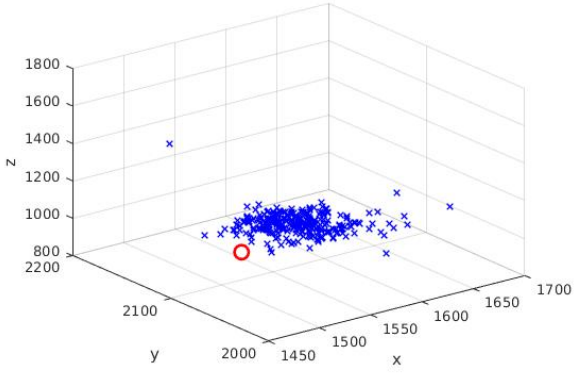

(i) Plot of Config 9




(j) Plot of Config 10

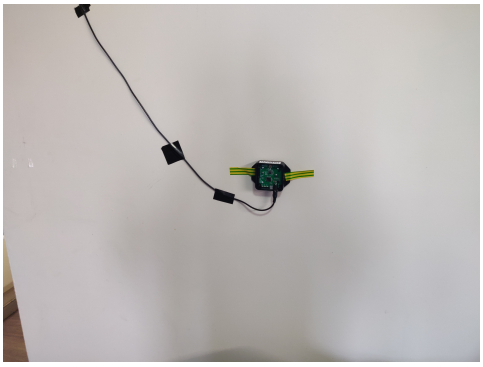



(k) Plot of Config 11

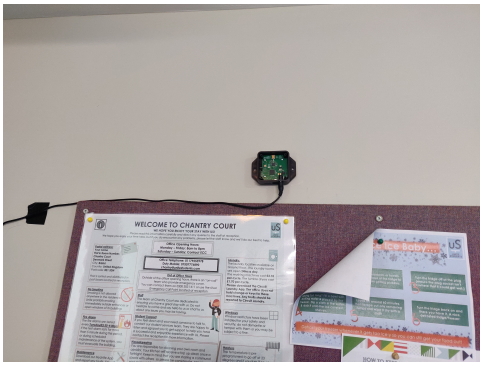



(l) Plot of Config 12

Figure 3.2: Sample plots of several anchor configurations. (cont'd)



(a) Anchor 1




(b) Anchor 2

(c) Anchor 3

(d) Anchor 4

(e) Actual Kitchen Layout

Figure 3.3: Physical layout of the kitchen/Test Area

## Summary

The initial results allowed for a suitable configuration to be obtained but these were recorded in ideal scenarios. Although these provide a good baseline for what to expect under good operational

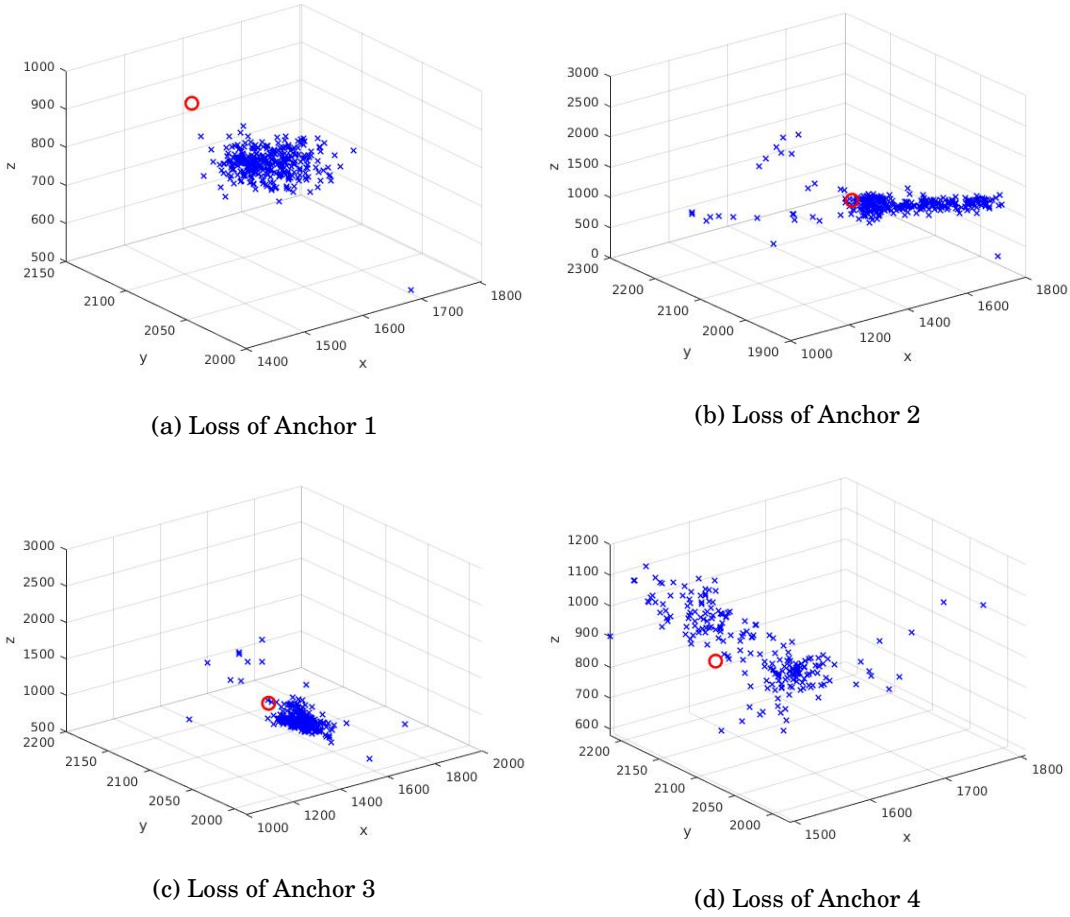
conditions, the parameters and limitations of the system would need to be tested before any form of optimisation and improvement to position estimates can be done. In the next Section: 3.2 we investigate this.

## 3.2 Operational Parameters

As mentioned in the Section: 3.1 the best anchor configurations for the given environment was determined but the results were collected in an ideal scenario. From work done by Mimoune et al. (2019) it was seen that the major factor affecting the positional accuracy seems to be No Line Of Sight (NLOS) between the anchors and a tag at any given time. Furthermore, the work discusses the use of using an optimal triplet of anchors in order to improve accuracy. This is an indication that a loss of a single anchor in the optimal configuration determined in the previous section should still yield reasonable results. To confirm the operational parameters the following experiments were carried out.

### 3.2.1 Loss of Anchor

In theory, a minimum of three anchors are all that is needed for determining the two-dimensional position, $(x, y)$, of the tag via trilateration. Losing a single anchor would, therefore, allow for the $(x, y)$ position to be determined with a fair amount of accuracy with discrepancies showing up in the z position readings. To test this data was recorded, starting with all four anchors and then systematically one anchor was turned of in each sample. Figure: 3.4 shows the plots obtained under this test. It can be seen that the $(x, y)$ position remains relatively unchanged in all with only a notable shift the z position with the loss of Anchor 1 and more noise introduced with the loss of Anchor 4.




(a) Loss of Anchor 1

(b) Loss of Anchor 2

(c) Loss of Anchor 3

(d) Loss of Anchor 4

Figure 3.4: Results obtained when an Anchor was lost.

### 3.2.2 No Line of Sight

The positioning calculations are based on a Two way ranging and Time of flight calculation scheme (see Appendix: A). The major crux of the calculation is that the wave is always moving at the speed of light this means that introducing an obstacle between any anchor and the tag that attenuates the signal would introduce discrepancies in the results obtained. Figure: 3.5 shows the setup used to test this scenario. The system was allowed to record data with no obstacles then a person walked along the trajectory indicated by the dotted arrows in the Figure and came to a rest as seen. It was ensured that the path taken by the person did not obstruct any of the anchors during motion. This was carried out 3 times:

1. Position 1: Provides NLOS between Anchor 2 and the tag.

2. Position 2: Does not provide NLOS between any anchors and the tag.

3. Position 3: Provides NLOS between Anchor 4 and the tag.

Position 2 was done to confirm that no reflections or attenuation occur with a person just being in the proximity of the tag. From Figure: 3.6 we can see that a person being introduced at position 1 obstructs the line of sight between an anchor and the tag and has adverse effects. It is clear that the 2 distinct blobs seen in the plots for position 1 and 3 show that the tag's 'position' has changed although it is stationary.

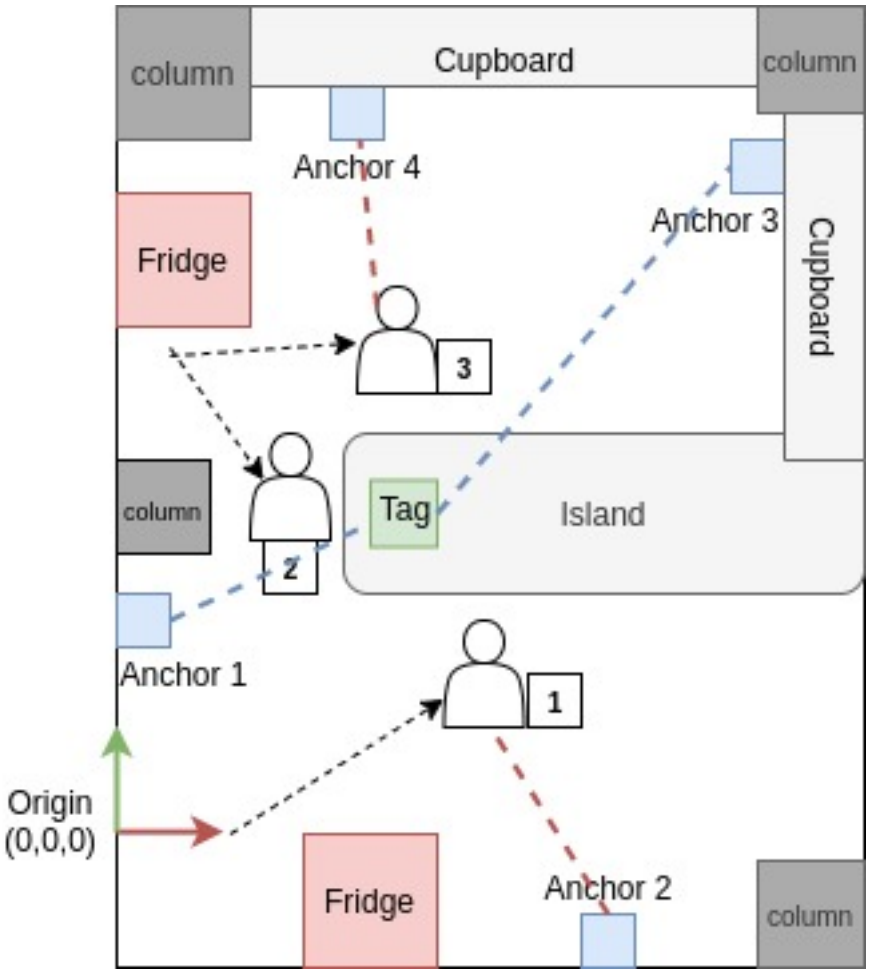
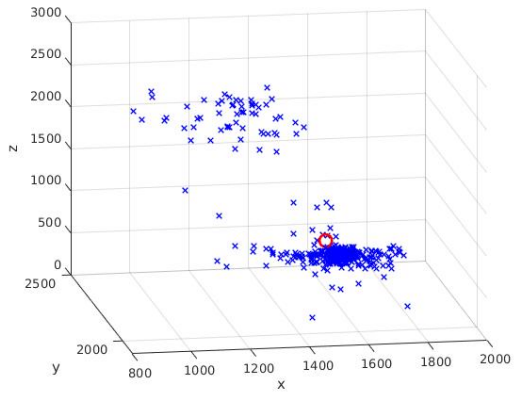
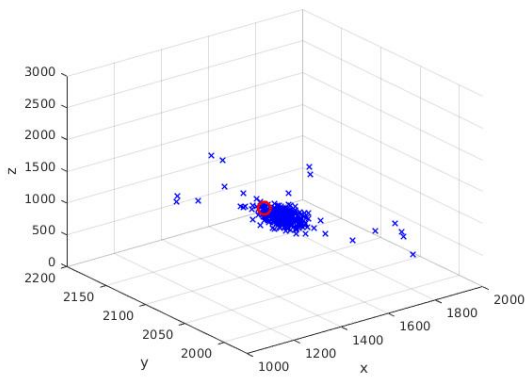



Figure 3.5: The test scenario for No Line of Sight experiment

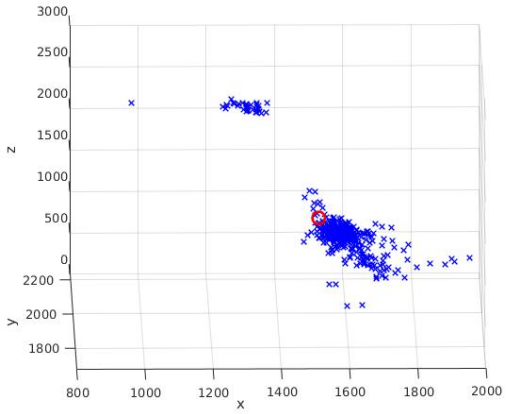
As it stands, NLOS is the major factor affecting the operation of the system and the accuracy of the raw readings. As such it was proposed that additional layers be added to process this position data and try to improve the positional accuracy of the system and make it robust.




(a) Person at Position 1




(b) Person at Position 2




(c) Person at Position 3

Figure 3.6: Results obtained when a person provides No Line of Sight.

## 3.3 Technical Design

At the end of Section: 3.2 it was shown that using the raw position readings coming from the sensor is not feasible in cases where NLOS is present. Furthermore, the overall goal is for the position of an indoor UAV to be recovered with a fair amount of accuracy since this forms the basis of any autonomous behaviour and navigation. As such it is desirable to use readings from multiple sensors but at the same time reducing the overall physical footprint of the final system to keep it lightweight. With that in mind, it is proposed to directly integrate the POZYX sensor system with a motherboard unit that is capable of doing sensor fusion, UAV control, and communicating with external systems. In recent years FCU's have become versatile and robust and these requirements are easily met so in this section the technical parameters of the research will be discussed.

### 3.3.1 Flight Controller Unit

In order to prevent consuming too much time on determining a suitable flight system the Open-Source Hardware (OSH) community was consulted in order to find a suitable FCU for the research. Survey work done by Ebeid et al. (2018) presents a qualitative analysis of several commercial hardware solutions. Table: 3.3 shows some of the major hardware considerations. All of the units have the standard UART, PWM and I2C interfaces in addition to other onboard sensors and interfaces. At the time of writing the Pixhawk series of FCU's are the most common and oldest systems. They have all the standard interfaces as well as several sensors allowing making them a keen choice for developers and researchers in a plug and play platform. Many in the series share the same interfaces with some of the smaller boards excluding some of the less popular interfaces. As such, to allow for flexibility in the technical design the Pixhawk line was chosen for this research. After deliberation and consulting the objectives and scope of this research it was determined that a basic Pixhawk would be satiable. Figure: 3.7 shows the board chosen, it was shipped quickly and it contains everything necessary to meet the objectives of the project allowing for quick prototyping and proof of concepts.

| Platform | MCU | Sensors | Licenses | Interfaces |
|---|---|---|---|---|
| Pixhawk | STM32F427 | b, m | BSD | c, s, a, pp, sb, ds |
| Pixhawk2 | STM32F427 | b, m | CC-BYSA-3.0 | c, s, a, pp, sb, ds |
| PixRacer | STM32F427 | b' m | CC-BY 4.0 | c, pp, sb, ds |
| Pixhawk 3 Pro | STM32F427 | b' m | CC-BY 4.0 | c, s, pp, sb, ds |
| PX4 FMUv5 and v6 | STM32F427 | b' m | CC-BY 4.0 | c, s, a, pp, sb, ds |
| CC3D | STM32F103 | None | GPLv3 | pp, ds, sb |
| APM 2.8 | ATmega2560 | b | GPLv3 | pp, a |
| Erle-Brain 3 | Raspberry Pi | b, m | CC BY-NC-SA | a |
| PXFmini | Raspberry Pi | b, m | CC BY-NC-SA | a |

Table 3.3: Comparisons of various FCU's that are commercially available. Source: Ebeid et al. (2018) Page: 2.
b: barometer, m: magnetometer, p: pitot tube sensor, c: CAN, s: SPI, a: ADC, pp: PPM, sb: S.BUS, ds: DSM, da: DAC, x: XBEE, au: AUX, [d]: discontinued.

Furthermore, since the chosen board is OSH it has several options of firmware that can be used which makes code and software developed on this system extendable to other boards given they are able to run the firmware.


Figure 3.7: Radiolink Pixhawk used for the project

### 3.3.2 Flight Firmware

Now that a suitable FCU was chosen the next major step was determining a flight stack to run on the board. The major firmware options for the Pixhawk are either Ardupilot or PX4 stack. Both are well documented and have their advantages. Both support a large array of vehicles but the major differences come from the licenses they are under. Without delving into the technicalities of the licenses it is often summarised that PX4 is attractive to business owners who want to protect their property whilst Ardupilot pushes for changes to be put back into the main codebase. Additionally, from a quick brief and use of each of the firmware, Ardupilot is slightly more documented due to its general public license and a bit more user friendly in terms of compilation and flashing firmware thanks to its pythonic based wrapper for compilation and uploading. Since both firmware cover all the generic UAV types and there was no need for any niche systems Ardupilot was chosen as the primary codebase. It is worth noting that there is a subsection of the Ardupilot codebase dedicated to Beacon based positioning systems which a previous Pozyx implementation is coded (Ardupilot 2019).

### 3.3.3 Communication Interface (I2C vs Serial)

As mentioned in Chapter: 2 there is an implementation of using a Pozyx system in a GPS denied scenario Ardupilot (2019). This shows that it is possible to get the positioning data into the Ardupilot codebase and have it interact with the onboard sensor fusion systems. A critique of this approach, however, is that it uses an additional Arduino Uno to pipe information from the Pozyx tag to the Pixhawk. Given the availability of the standard interfaces onboard the Pixhawk and the scope and objectives of this research project, it is feasible that the Uno can be stubbed out of this experiment and data can be integrated directly onto the Pixhawk. This means the physical integration of the system can be summarised into the following steps:

- Choosing a suitable interface.

- Placing the code within a suitable section of the codebase.
- Utilise the parent class of the section of the Ardupilot to integrate the sensor.
- After testing the base functionality, integrate the new library within the background scheduler so it can run and feed measurements in the background to the copter main code.

An Arduino (pozyxLabs 2020) and python libraries were originally designed by the Pozyx developers so it provided an algorithmic starting point for the functionality of the library being designed. The Arduino library is based around the I2C protocol whilst the python module uses the serial interface. As noted by the developers the python module is a port of the Arduino library and there are no plans to maintain the code and address bug fixes, as such the Arduino library was studied fo to build the new AP_Bacon extension. Furthermore, the I2C interface was designed onboard an embedded system so much of the flow can be ported effectively to the Pixhawk. Lastly, the previous implementation uses the UART/serial interface so that avenue has already been explored so in the interest of quick prototyping the I2C interface was chosen. A positive note is that the existence of the prebuilt system allows data and experiments to be collected in lieu of the development of this library since the information being provided is the same data generated by the Pozyx tag. The design of this library proves that the sensor data can be integrated directly whilst the pre-built systems, which are stable, can be used for data collection and experiments.

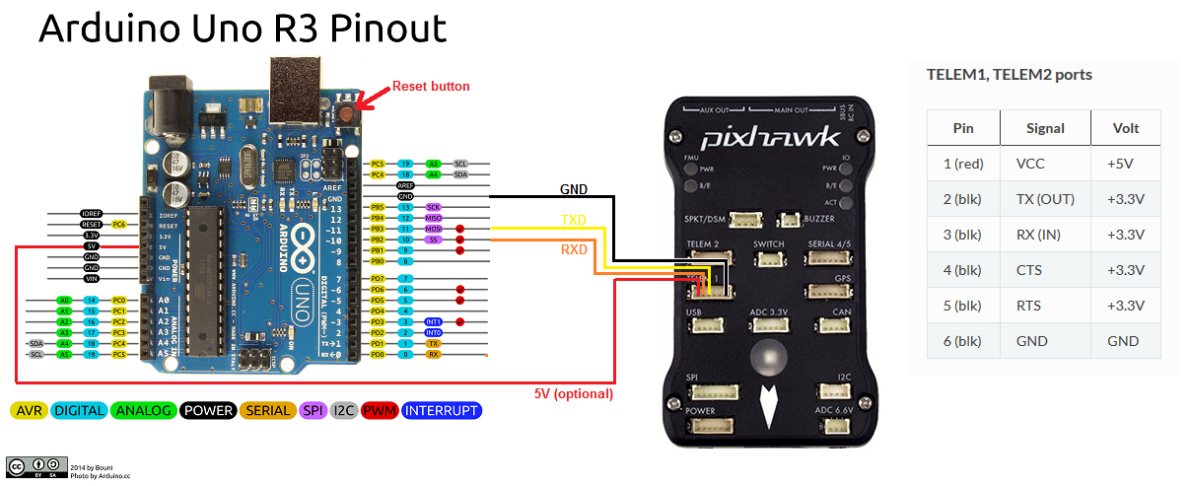



Figure 3.8: Non-GPS loiter solution provided by Ardupilot. Source:`https://ardupilot.org/copter/docs/common-pozyx.html`

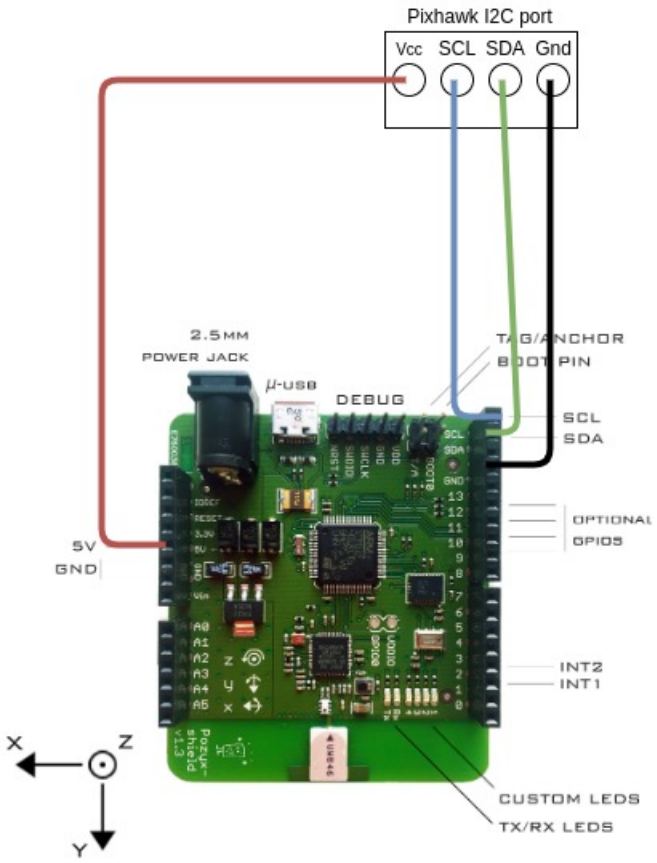



Figure 3.9: Proposed wiring of the I2C interface being developed

### 3.3.4 Sensor Fusion: Pozyx Readings and EKF on Ardupilot

As mentioned in Chapter: 2 sensor fusion forms the basis of estimating states of the system when there are multiple measurements of that state present. The de facto one used in navigation and localisation happens to be the Extended Kalman Filter (EKF). The Ardupilot codebase implements an EKF to estimate 24 states. The state being affected by the implementation will be the position estimates given in North, East, Down (NED) frame. The NED reference frame is similar to the standard left-handed notation used in several standard robotic systems with the major difference being the Down axis which is opposite in direction to the standard Z axis in robotic systems. The developers of the Pozyx system, Pozyx Labs (2018*b*), designed the firmware of the tag in such a way that obtaining information or triggering any functions are done with sequential reading and writing operations to various registers on the tag. Pozyx Labs (2018*a*) lists all the functionality of the tag via the registers. From the registers, it is possible to obtain both the distances of the tag to each anchor as well as the position of the tag within the workspace. It was confirmed by developers that the positioning algorithms operate in two modes.

1. A pure triangulation using the TOF readings from each anchor.

Figure 3.10: System being unit tested in the environment.

2. A tracking mode which combines IMU and TOF readings in a Kalman Filter to improve accuracy.

For the purpose of this research, the position readings were used since these can be directly integrated into the Ardupilot codebase immediately. Furthermore, the tag was configured to be used in the tracking mode to improve the accuracy of the measurements. Lastly, from section: 3.2 and notes from the Pozyx team and Mimoune et al. (2019) 4 anchors may not provide sufficient data for a full 3D position so the system was hardcoded to operate in a 2.5 dimension which means the height (z reading) was fixed on a surface so the tag calculated and provided (x,y, z reading) positioning when requested. The final step was to have this implemented on the Pixhawk with the Ardupilot firmware within the Beacon subsystem of the code. This meant that the following steps had to be implemented.

- Initialise the I2C bus on the Pixhawk and configure the tag.
- On an 'update()' function call, set the vehicle position based on the tag reading and set the distance between each anchor and the tag.
- A 'healthy()' function call, should return true if the Pixhawk was able to retrieve viable data from the tag.

Appendix: B shows the core code designed to do this but the overall integration with the Ardupilot codebase can be seen by the pseudo-code below.

```
1   Main vehicle code
2       loop()
3       {
4           beacon_reading = beacon->get_vehicle_position()
5           EKF_update()
6           {
7               ...
8               fusebcnreading()
9               ...
10          }
11          .
12          .
13          .
14  Background Scheduler
15      loop()
16      {
17          ...
18          beacon->update()
19          ...
20      }
21
22      }
```

Appendix: B.1.3 also shows a unit test to ensure the standard interface and data encapsulation is exactly what is needed and provided by the other beacon measurement systems in the Ardupilot codebase. Although it was verified that the code was integrated and the measurements were being captured by the main vehicle code and EKF measurement function operating indoors prevented the internal compass/magnetometer from working properly. When the compass is disabled the 'yawAlignComplete' check, which is required by the EKF to see if beacon measurements are viable to use, is set whilst the UAV is in flight. Due to only having access to an FCU at the time, the onboard EKF estimates could not be obtained and sent via MAVLink without a refactor to the core code which is outside the scope of this research project. However, to rectify this a simple two state EKF's were implemented in python and on a mobile robot to determine if viable position estimates could be generated that can be used in a UAV or robotic system.

### EKF Design

As discussed in Section: 2.2 and the preceding section an EKF was implemented to improve the position estimates from the Pozyx sensor. The major driving force for using an EKF over other filters and estimators was to mimic the Ardupilot setup as closely as possible as to give hints for future work and testing of the UAV whilst in flight. Once again the tag will be operated in 2.5 dimensions in tracking mode to determine if the system can withstand NLOS when dynamic obstacles are introduced into the environment. Similar to the setup in Section: 3.2 the tag was allowed to operate and record in various scenarios while static and moving. It should be noted

that since we are operating in 2.5 dimensions a two state filter will be implemented. Furthermore, since the tag outputs direct measurements of the state the matrices are simplified and easy to implement.

System equations:

$$f(\hat{x},\hat{u},t) = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$$

$$z(\hat{x}) = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$$

Linearised state transition model:

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Linearised observation model:

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

An added benefit of using the Pozyx sensor is that it is possible to obtain the measurement uncertainty for each measurement which allows it to be incorporated in the calculation of the Kalman Gain at each time step.

$$R = \begin{bmatrix} Var(x)(t) & Cov(x,y)(t) \\ Cov(x,y)(t) & Var(y)(t) \end{bmatrix}$$

Another adjustment from the standard EKF is that outlier measurements were not used. Outliers arose due to sudden jerks in motion causing the measurements coming out of the tag to be erratic and vary wildly in certain time steps.

The above EKF design represents a pure pythonic approach using only the readings from the sensor. Given that it only has once set of observations it can be considered a simple filter rather than a state estimator. To emulate the process that should run on the Pixhawk is is desirable to actually incorporate another stream of measurements that would improve the position estimates. To this end, an EKF was applied on a wheeled mobile robot running dead reckoning localisation. To easily fuse the dead reckoning readings, the robot sensors were simplified to outputting the $(x, y)$ position based on left and right wheel encoder readings. Additionally, the board is Arduino based so integration with the Pozyx system is straightforward. Figure: 3.11 shows how this system was connected and used.

This following system of equations shows the EKF on the wheeled mobile robot:

$$f(\hat{x},\hat{u},t) = \begin{bmatrix} x & 0 \\ 0 & y \end{bmatrix}$$

$$z(\hat{x}) = \begin{bmatrix} x_{tag} & 0 \\ x_{encoders} & 0 \\ 0 & y_{tag} \\ 0 & y_{encoders} \end{bmatrix}$$

Linearised state transition model:

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Linearised observation model:

$$H = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} Var(x)_{tag} & 0 & 0 & 0 \\ 0 & Var(x)_{encoders} & 0 & 0 \\ 0 & 0 & Var(y)_{tag} & 0 \\ 0 & 0 & 0 & Var(y)_{encoders} \end{bmatrix}$$

## 3.4 Summary

This section thoroughly discussed the methodology and design process used to achieve the objectives and scope of this research. Although Ardupilo'st EKF could not be used due to several limitations and scoping of this project, the feasibility of an EKF, in general, was explored as a method of improving the overall position estimates of the system which can then be extended for use in the future within an Ardupilot based FCU without the current limitations. In the next chapter the results of the EKF will be presented under various scenarios to show the effectiveness and limitations of the Pozyx sensor in a household environment.

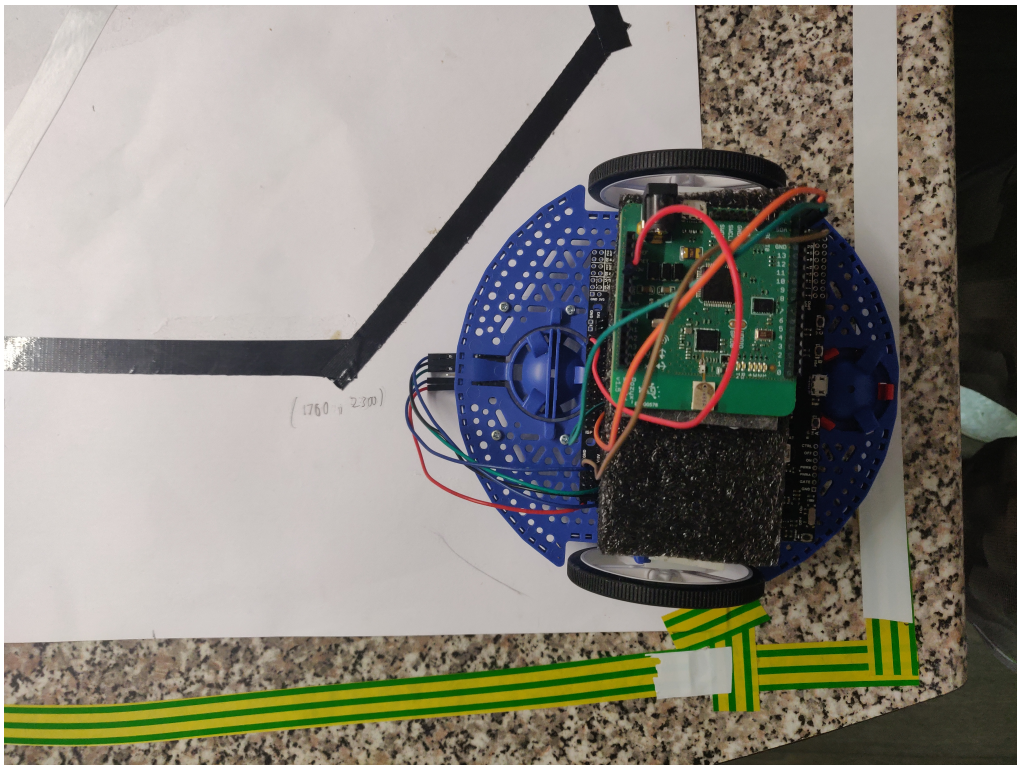


Figure 3.11: Wheeled mobile robot equipped with the Pozyx Tag.

CHAPTER 4

# DESIGN OF EXPERIMENTS

Chapter: 3 documents the design of the hardware and algorithmic implementations of the system that was used. Section: 3.2 shows some of the results obtained in a configuration step and determining the limitations of the unprocessed data. To evaluate the performance of the designs, several Test suites were planned which exploited the No Line of Sight (NLOS) issue seen in previous sections. To provide NLOS, rather than systematically occlude an anchor and tag, a person walked a fixed path (see Figure: 4.1) at random speeds whilst the data was being recorded. This was done as to emulate a more realistic environment that the system would be used in.

For tests that had the tag moving two trajectories were outlined. Table: 4.1 shows the waypoints of each of the trajectories. Trajectory one is a quadrilateral whilst trajectory two is a combination of straight-line segments. In these tests, the tag was moved along the chosen trajectory whilst NLOS occurred randomly due to a person walking.

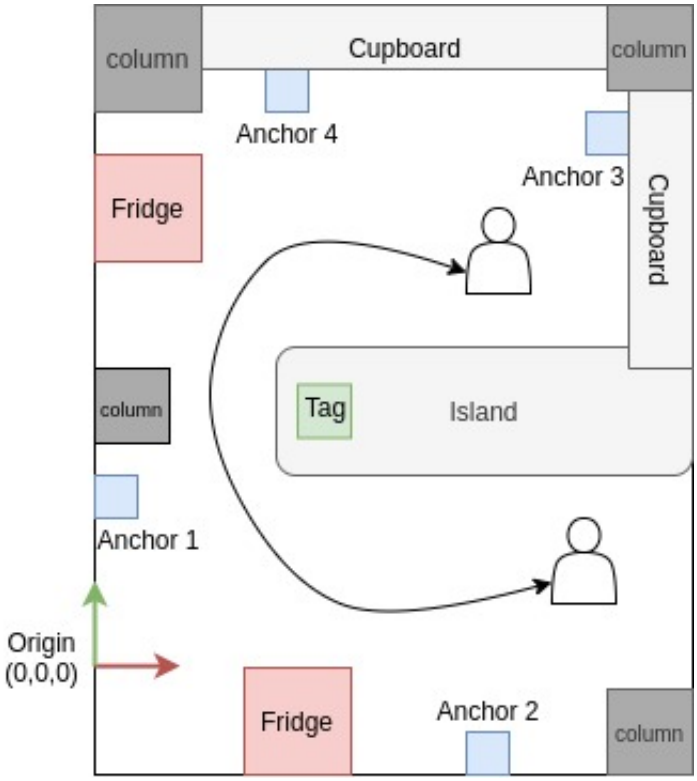



Figure 4.1: Path taken by the person to randomly occlude anchors and tag.

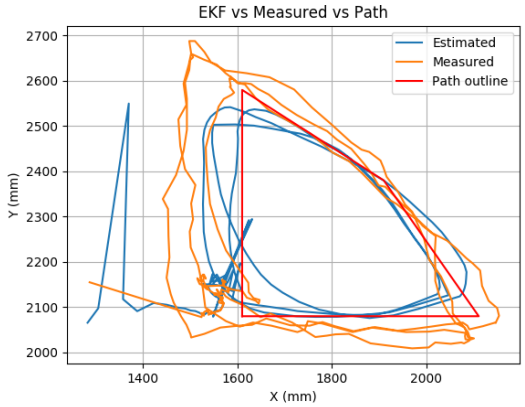
| | Waypoints $(x, y)$(mm) |
|---|---|
| Trajectory 1 | (1610, 2080)<br>(2111, 2080)<br>(1910, 2380)<br>(1610, 2580)<br>(1610, 2080) |
| Trajectory 2 | (1610, 2080)<br>(1710, 2200)<br>(1760, 2300)<br>(1770, 2500)<br>(1840, 2580)<br>(1934, 2625)<br>(1992, 2595) |

Table 4.1: Trajectories used in the tests for data collection.

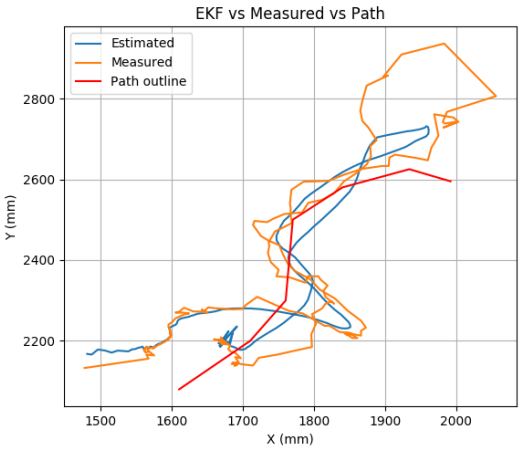
## 4.1 Benchmarking and Calibration

Before testing the system with limitations a benchmark and calibration test was run to record results in an ideal scenario. This test provided a base that can be used to compare the results of the system operating in non-ideal situations. Figure: 4.2 shows the results obtained for this benchmarking. It is seen although the measured position from the tag is slightly noisy, it is well

within the ±10cm accuracy quoted by the Pozyx system once the system has started moving and some time has passed.




(a) Person moving Tag on Trajectory 1.

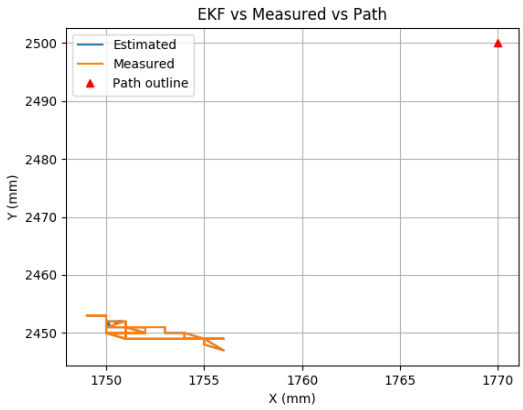



(b) Person moving Tag on Trajectory 2.

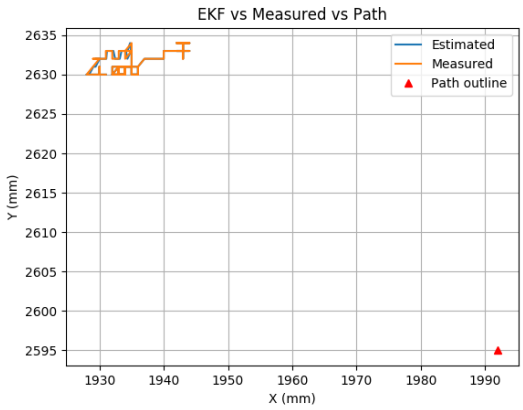
Figure 4.2: Results for estimation with Line of Sight

## 4.2 Stationary Tags

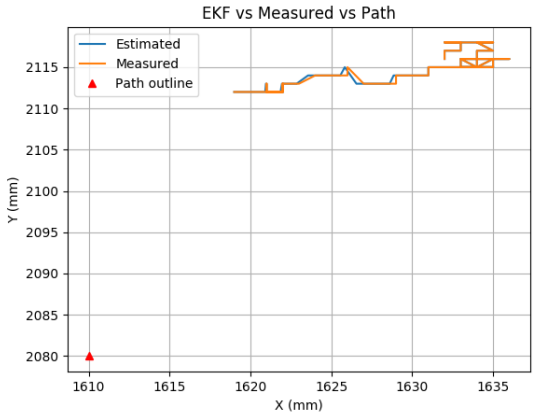
For these evaluations the designed system was tested with stationary tags, similar to what was done in Section: 3.2. These were carried out to cover the trivial cases where NLOS proved ot be an issue for even a stationary tag. For the stationary tag tests, the tag was placed at a known point in the workspace and it was introduced to the basic limitations discussed in previous chapters. The results for the loss of a single anchor was no different than the case recorded previously without the EKF . However, Figure: 4.3 shows the results obtained with the tags located at various positions whilst a person walks randomly in the environment. In contrast to the previous section, it can be seen the system with the current configuration and EKF is able to withstand NLOS between the anchors and the tag with minimal change in the perceived position. However, it is evident that the system is now susceptible to a steady-state error which is expected due to the Pozyx's integration with an IMU . The tag's position will not change drastically and this error will be present as long as the tag is stationary.




(a) Result with tag at (1770,2500)




(b) Result with tag at (1992,2595)



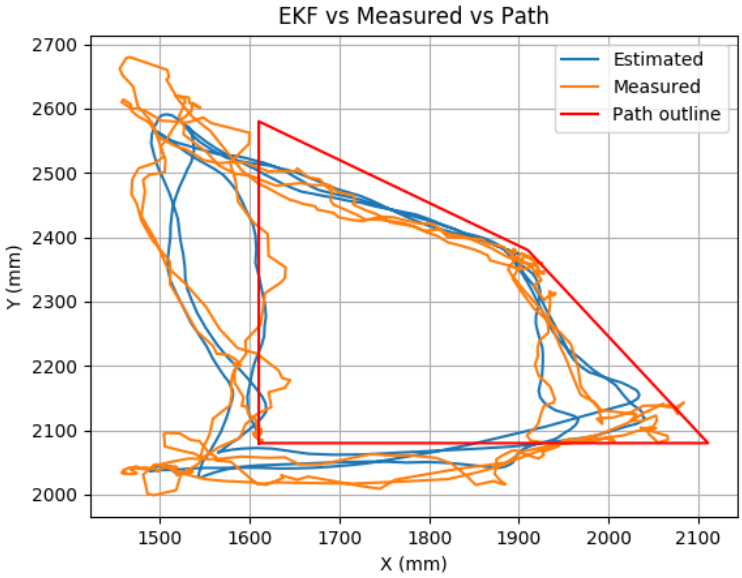

(c) Result with tag at (1610,2080)

Figure 4.3: Results obtained with No Line of Sight and a Stationary Tag.

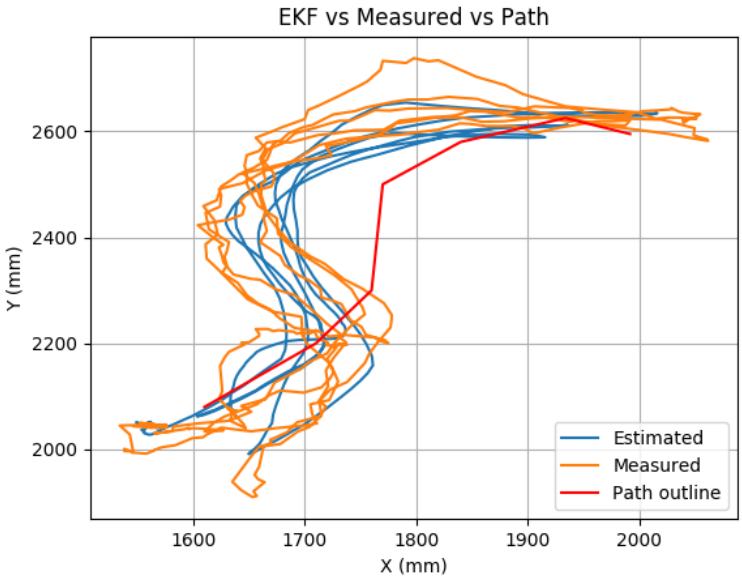
## 4.3 Localisation of a moving target

The following experiments would evaluate the designed system while the tag was moving. This test was done to determine if the processing on the Pozyx tag in addition to an EKF would be able to provide suitable position estimates without the need for any additional measurements or processing. Figure: 4.4 shows the paths recorded for each of these trajectories while the tag was moved by a single person. It can be seen that as expected the EKF smoothens the supposed trajectory providing a better estimate. Noteworthy is that using the IMU on the Pozyx tag causes the position to veer slightly on turns. Additionally, the Pozyx tag was mounted onto a mobile robot that was built to follow a line. This secondary setup was used to ensure that there was no interference from the person moving the tag and any errors seen would be due to the NLOS caused by the person walking in the environment as seen in Figure: 4.1. Furthermore, using the mobile robot would indicate if the position estimates would be viable in a fully autonomous scenario. Figure: 4.5 shows the result obtained while the robot followed Trajectory 2 over two

seperate runs.




(a) Movement along Trajectory 1




(b) Movement along Trajectory 2

Figure 4.4: Results obtained with No Line of Sight while the tag is moved by a person.

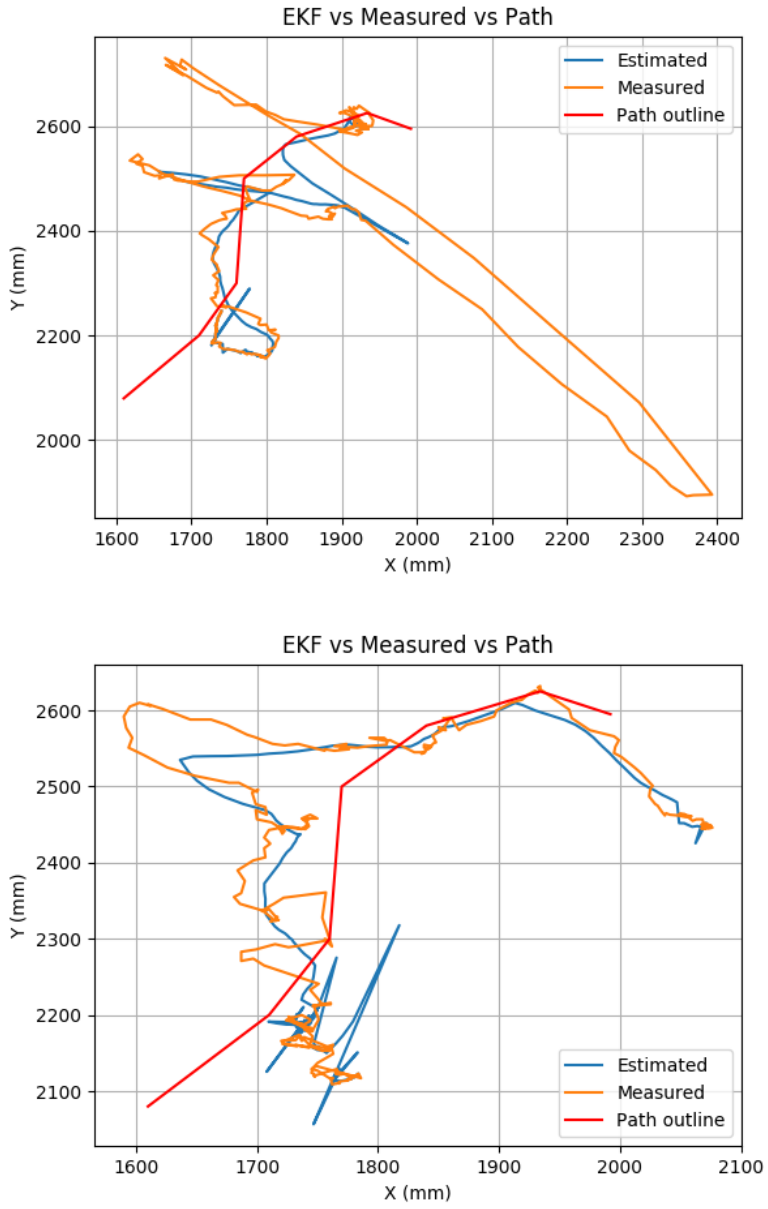



Figure 4.5: Results obtained with No Line of Sight while the tag was mounted on the mobile robot.

## 4.4 Improving the Estimates

In UAV's there are different sensors that provide a multiple measurements for the same state. As discussed in Section: 2.2 it is useful to combine multiple sensor readings in order to improve the overall estimate of the state. In Section: 3.3.4 two methods of using the EKF was discussed. For previous sections of this chapter, the input of the EKF consisted only of the measurements from the Pozyx tag. In order to compliment the Pozyx measurements, dead reckoning data from the

mobile robot was also used as an input into the EKF . Figure: 4.6 shows the results obtained when the system was run in an ideal situation (Line of Sight). This was done to once again set a base for comparing the No Line of Sight results. Figure: 4.7 shows the results obtained from three separate runs of the robot. The plots show, the actual path taken by the robot, the readings from the Pozyx tag, the readings from the dead reckoning system and the output of the EKF .

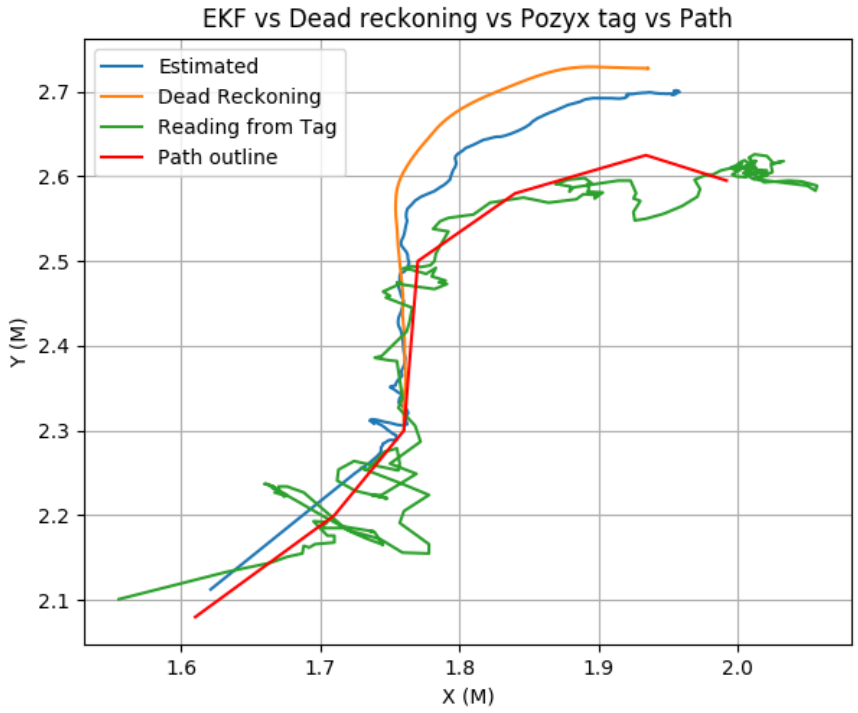



Figure 4.6: Line of Sight estimates from EKF using dead reckoning and Pozyx readings.

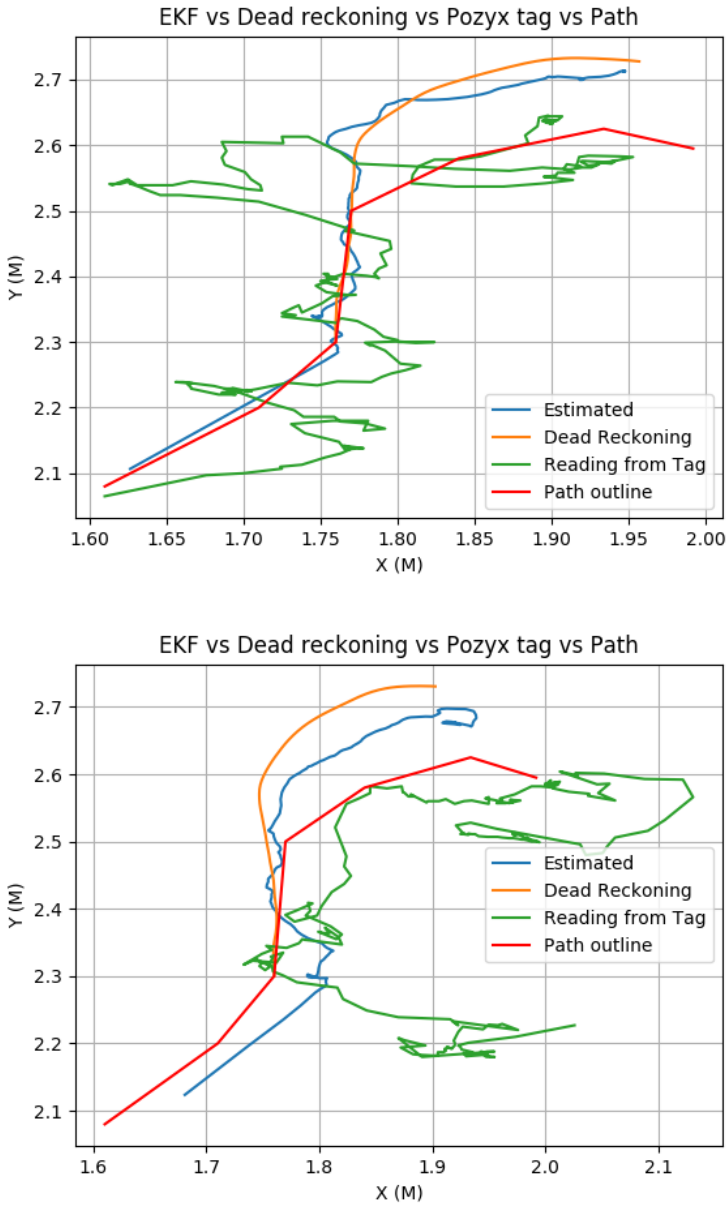



Figure 4.7: Results obtained with No Line of Sight and EKF estimates using both dead reckoning and Pozyx readings.

## 4.5 Summary

This chapter documented the experiments designed and ran in order to evaluate the performance of the Pozyx system for indoor localisation. These tests were necessary as they addressed the feasibility of using the Pozyx system in UAV's and other forms of autonomous systems. In the next chapter the results will be discussed providing a qualitative and quantitative analysis of the plots seen in this section.

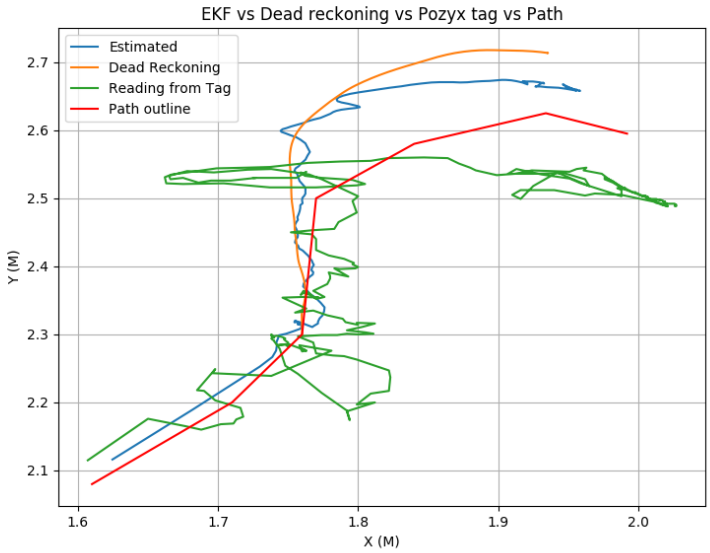



Figure 4.7: Results obtained with No Line of Sight and EKF estimates using both dead reckoning and Pozyx readings. (cont'd)

CHAPTER 5

# Results and Analysis

To document the effectiveness of the Pozyx setup results were gathered in several scenarios as seen in Chapter: 4. From a qualitative point, the embedded EKF fusing the dead reckoning data and tag data seems to outperform the other methods in terms of falling close to the actual path and being robust to the erratic behaviour introduced by No Line of Sight. The plots also highlight a major issue that arose when the tag was in motion and there was No Line of Sight between one of the anchors and the tag. Although it was easy to pinpoint the major issues from the plots, a quantitative measure was required to soldify the findings. A useful metric would be to see the distance from the supposed positions to the path being traveled. Since the robot moved autonomously and there was no perception of where it should be at a given time and a way of effectively measuring the distance to the path needed to be designed. Figure: 5.1 shows what is available while data is being recorded. The green arrows represent the minimum distance for a candidate point and the path. Appendix: D shows the general pseudo-code to obtain the green distance for each point. A benefit of using a distance metric is that it is easy to interpret and ideally should be zero if the pose estimates are completely accurate. This also means that standard statistics on this distance metric would give a clear indication of which method produced the best results. Table: 5.1 shows the final metrics obtained for the various results. The table is color coded to indicate what the inputs to the EKF are for each test. In addition to the EKF fused results, the statistics for the tag and dead reckoning positions were included to set a benchmark for comparison. As expected the estimates combining the dead reckoning and tag readings clearly outperforms the others. The EKF implemented on the WMR is quite simple so it is expected that similar or even better results would be obtained on the FCU whilst in flight.

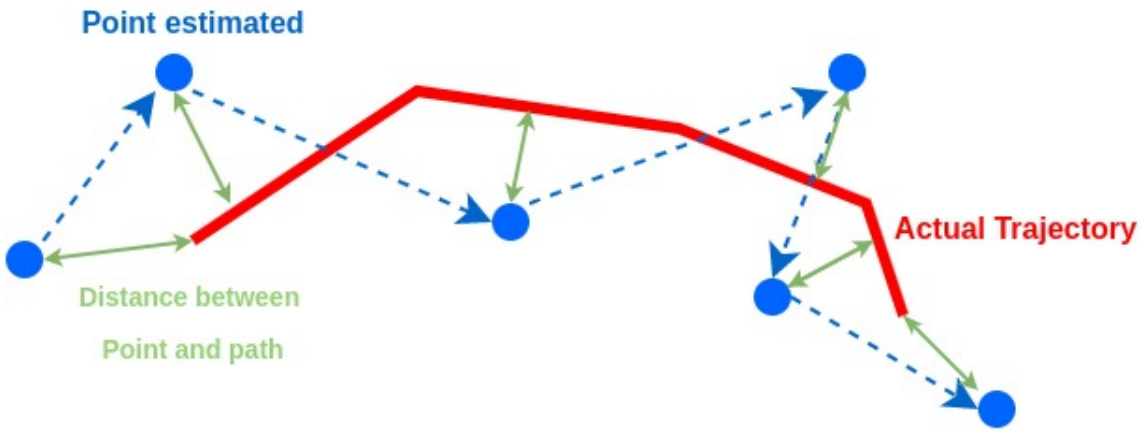



Figure 5.1: Illustration of how a distance metric was developed for analysis.

LOS = Line of Sight
NLOS = No Line of Sight
EKF = Extended Kalman Filter
WMR = Wheeled Mobile Robot

| | Max | Mean | Variance | Skew | Kurtosis |
|---|---|---|---|---|---|
| EKF input is only the Readings of the Pozyx Tag | | | | | |
| Trajectory 1 traced by human with LOS | 0.4031 | 0.115 | 0.007 | 1.219 | 0.5642 |
| Trajectory 2 traced by human with LOS | 0.1563 | 0.06 | 0.0016 | 0.1164 | -1.432 |
| Trajectory 2 by WMR with LOS | 0.0795 | 0.04 | 0.0009 | -0.0735 | -1.7372 |
| Trajectory 1 by human with NLOS | 0.331 | 0.114 | 0.006 | 0.7851 | -0.1624 |
| Trajectory 2 by human with NLOS | 0.137 | 0.0431 | 0.0009 | 1.0405 | 0.3421 |
| Trajectory 2 by robot with NLOS | 0.3297 | 0.05416 | 0.00189 | 1.0415 | 3.8087 |
| EKF input consists of Dead reckoning readings & Pozyx tag Readings | | | | | |
| Trajectory 2 by WMR with NLOS | 0.087 | 0.036 | 0.00047 | 0.1634 | -0.2665 |
| Raw Dead Reckoning readings | 0.1081 | 0.067 | 0.00136 | -0.9802 | -0.8165 |
| Raw Pozyx Tag Readings | 0.1124 | 0.069 | 0.00153 | -0.5188 | -1.352 |

Table 5.1: Metrics for the Distance from each path for different tests and positions.

CHAPTER 6

# DISCUSSION

In Chapters: 1 and 2 the use and efficacy of using a UWB based system for localisation was explored. Although many of the research dove into the localisation not many highlighted the use in real-life scenarios and environments. In the previous research, the systems researched the use in GPS denied environments but not much covered the cases of dynamic obstacles in the environment that provide NLOS and hence errors in TOF calculations. As mentioned previously, Mimoune et al. (2019) provide a comprehensive analysis and evaluation of localisation based on the Pozyx system but it addressed the issues of static obstacles with NLOS. The gap of research addressing dynamic obstacles in the environment using the UWB Pozyx sensor is where this research comes in. The aim of this research can be split into two broad tasks:

1. Show it is possible for an FCU to receive and use the Pozyx sensor readings.

2. Evaluate an approach to use the sensor readings to get viable position estimates.

From Section: 3.3 it is shown that it is possible to pipe the information from the tag to the Pixhawk unit through its I2C interface. Furthermore, the written test, see Appendix: B.1.3, confirms that the Pixhawk's AP_Beacon subsystem is indeed able to access the Pozyx tag's measurement through the designed interface. Through the right configuration of user parameters and recompilation of the codebase, it was also confirmed that the new AP_Beacon library runs successfully with the main code's scheduler to update the beacon's sensor reading periodically so that the other subsystems in the codebase can use. The core subsystem that uses the beacon's frontend readings would be the Ardupilot's EKF estimation modules. Researching that subsystem it was confirmed that the beacon frontend was being used to obtain vehicle positions but was not using it in the fusion step of the EKF to estimate position. At the end of Section: 3.3.4 it was discovered that the EKF required certain checks to be passed before fusing the beacon's vehicle

position and due to the lack of hardware this could not be accomplished. After reviewing the objectives and scope of this research project it was determined that the refactor required needed to get the estimates from the Pixhawk was not necessary. It was proven that the data can be used on the Pixhawk and the position estimates can be obtained via other methods in order to evaluate the use of the Pozyx system in indoor localisation with persons in the environment.

To evaluate the use of the sensor for indoor localisation mutliple scenarios were developed as seen in Chapter: 5. Since EKF's are used commonly in navigation and control systems, as seen in the case of Ardupilot, it was utilised in this research in order to get position estimates in the household environment. The EKF was implemented in two ways as seen in Section: 3.3.4 with one using the tag measurements alone and an embedded one combining dead reckoning data with the tag readings. From the tag's measurements while a person walked randomly in the environment it was shown that this caused major issues in the readings, particularly when NLOS occurred while the tag was being turned. This result was expected since the tag was configured to utilise its internal IMU in a filtering algorithm. The combination of increased TOF readings due to NLOS and the IMU readings whilst in motion gave substantial positioning errors of magnitude of hundreds of centimeters. These positioning errors can be seen prominently in Figure: 4.5. Unreliable readings like these are why fusing multiple measurements of state is necessary. Dead reckoning is a common way of determining pose estimates in vehicles and robotic systems. Ardupilot also uses dead reckoning in its EKF so employing one here addresses the capabilities and position estimates that can be possible when the system is finally used on a UAV. From the distance metrics seen in Table: 5.1 it is clear to see that a simple fusion of data is able to increase the position estimates greatly with the max distance from line (8.7cm) being better than the quoted accuracy of the Pozyx system ($\pm$10cm) even with NLOS present. In general, the closer any of the distance metrics are to zero the better the positioning result.

CHAPTER 7

# CONCLUSION

## 7.1 Conclusions

Throughout this research the idea of using a commercially available UWB sensor system for indoor localisation was thoroughly checked. In addressing the research question and objectives, a software engineering approach was employed to select various hardware options, choose suitable communication interfaces based on previous technical work and existing software solutions, integrate libraries with the existing code base so it can run seamlessly in the main code, and finally run unit and integration tests to ensure everything worked as needed. Although a viable design was developed and presented it could not be tested due to limitations. For the evaluation of the Pozyx system another embedded system was used. From the evaluation a major issue that was immediately identified from the measurements was the large positional error that arose when the system was in motion and there was No Line of Sight between Anchors and the tag. This showed that the Pozyx system alone cannot be used for localisation and navigation. However, as with any robotic or UAV system, it is not expected that only one sensor would be used to generate position estimates. As seen from the Table: 5.1 the use of an EKF with a simple model and dead reckoning measurements of position greatly improved the position estimates and it is expected that any system that employs this, such as the Ardupilot codebase, should be able to get better position estimates in order to carry out their autonomous tasks.

## 7.2 Limitations, Mitigation and Future Work

Due to the current pandemic situation, some of the major objectives of the research project had to be changed. Initially, it was planned that a UAV would be flown autonomously indoors utilising the poses from the FCU in a higher level planner but due to the closing of the labs this objective

was removed. There was no drone readily available to test the new software on and there was no way to easily determine the ground truth positions of the UAV while in flight. Additionally, having no UAV available hindered the testing on the FCU since the FCU had to be in flight in order for the EKF to use the Pozyx readings in the position estimates. To compensate for the lack of position estimates from the Pixhawk and Ardupilot, another robot was used that provided dead reckoning data. This was suitable for the scenario since a similar approach is employed in Ardupilot although how the dead reckoning estimates were generated is different due to it being a different vehicle/actuated system. Furthermore, the lack of an actual laboratory space constrained what tests could be done. Using a shared space meant that experiments had to be confined to a single counter-top and measurements were made as accurate as possible without the use of any external tools that would be available in the laboratory. Future work and extension of this research would be to actually test the developed AP_Beacon system on an UAV and compare the pose estimates while in flight to the ground truth estimates. Furthermore, the de-scoped objective of actually navigating the UAV would be another extension of this research and work towards a completely viable indoor navigation system. A final extension of this work would be to extensively test the EKF estimates to track and localise a person within the workspace without being confined to a specific area.

APPENDIX A

# APPENDIX A

## Two Way Ranging (TWR)

TWR is a ranging method that utilises TOF and delays during transmission of a packet in order to determine the range between a tag and anchor. Figure A.1 provides a simple illustration of how this works. The distance for an individual tag and an anchor can be obtained by:

$$d = c.\frac{(TT2 - TT1) - (TA2 - TA2)}{2}$$

This is repeated for each anchor and then the position of the tag can be determined via trilateration. Geometrically, the position of the tag can be described as the point intersection of all the circles with distance,d , from the tag. This can be seen in Figure A.2.

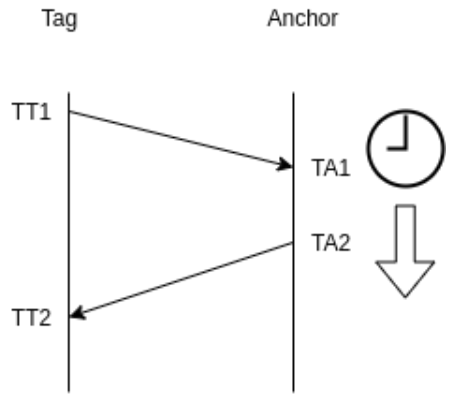



Figure A.1: Packet transfer in TWR.

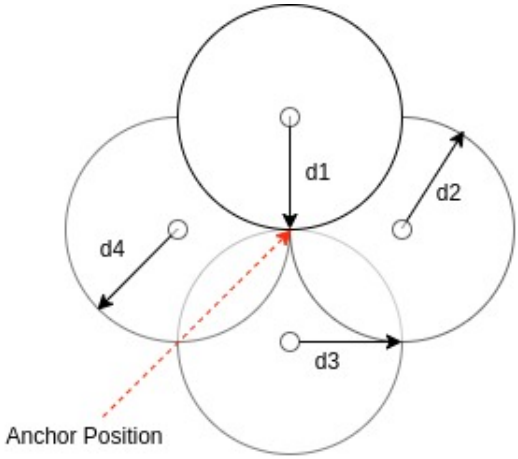



Figure A.2: 4 Anchor and 1 Tag trilateration in 2D.


# APPENDIX B

## B.1 Pozyx I2C Library

Full code and changes can be found at: https://github.com/shivrar/ardupilot

### B.1.1 AP_Beacon_PozyxI2C.cpp

```
//
// Created by shivan on 7/14/20.
//

#include "AP_Beacon_PozyxI2C.h"

// This is a directive declaration so we'll grab hal from the relevant definition file
extern const AP_HAL::HAL& hal;


void AP_Beacon_PozyxI2C::_init(int8_t bus)
{
    if (bus < 0) {
        // default to i2c external bus
        bus = 1;
    }
    _dev = std::move(hal.i2c_mgr->get_device(bus, POZYX_I2C_ADDRESS));
    if (!_dev) {
        return;
    }

    hal.console->printf("Starting POZYX device reading on I2C\n");
```

```
// lets initialise the actual tag

uint8_t whoami;
uint8_t regs[3];
regs[2] = 0x12;


hal.scheduler->delay(250); // This might not be the cleanest but I'll look for something cleaner
if(this->reg_read(POZYX_WHO_AM_I, regs, 3) == POZYX_FAILURE){
    hal.console->printf("Failure to read info on shield\n");
    return;
}
whoami = regs[0];
hal.console->printf("Who am I Pozyx:%x \n", whoami);

this->reg_function(POZYX_DEVICES_CLEAR, nullptr, 0, nullptr, 1); // Leaving this in for now
//Iterate and set the positions in the tag. Look at possibly not hardcoding this
for(unsigned int i = 0; i < sizeof(this->_anchors)/ sizeof(_anchors[0]); i++){
    uint8_t dev_params[15];

    Vector3i curr_anc_pos = _anchor_pos[i];

    int32_t x = curr_anc_pos.x;
    int32_t y = curr_anc_pos.y;
    int32_t z = curr_anc_pos.z;


//  Lord forgive me for I have memcpy'ed again
    memcpy(dev_params, _anchors+i, 2); //First 2 bytes refer to the Network id
    dev_params[2] = 1; //Third byte to represent an anchor
    memcpy(dev_params+3, &x, 4); // Next 4 bytes is the x position
    memcpy(dev_params+7, &y, 4); // Next 4 bytes is the y position
    memcpy(dev_params+11, &z, 4); // Next 4 bytes is the z position
    //TODO: can redo this with the DO_RANGING register to both add and get the distance -> could
    bool status = this->reg_function(POZYX_DEVICE_ADD, dev_params, 15, nullptr, 1);
    if(!status){
        hal.console->printf("Error Setting Anchor Positions\n");
    }
    set_beacon_position((uint8_t)i, Vector3f(x/1000.0f,y/1000.0f,z/1000.0f));
}


uint8_t algorithm = POZYX_POS_ALG_TRACKING;
uint8_t pos_dim = 1;
uint8_t pos_alg = algorithm | (pos_dim<<4);
this->reg_write(POZYX_POS_ALG, &pos_alg, 1);
```

```
    int32_t z_pos = 928;

    //Due to the limited anchors and configurations lets just work on a planar surface for now.
    this->reg_write(POZYX_POS_Z, (uint8_t*)&z_pos, sizeof(int32_t));

    this->_update_beacons();
}

AP_Beacon_PozyxI2C::AP_Beacon_PozyxI2C(AP_Beacon &frontend):
    AP_Beacon_Backend(frontend)
{
    this->_init(1); //Setup the pozyx device
}

bool AP_Beacon_PozyxI2C::healthy()
{
    // Should return true if we are constantly getting good data based on the POZYX_TIMEOUT
    return ((AP_HAL::millis() - last_update_ms) < AP_BEACON_TIMEOUT_MS);
}

// update
void AP_Beacon_PozyxI2C::update(void)
{

    //Should call the set_vehicle_position() from the parent class to pass it into whatever needs it -> loo
    //implementation with arduino
    // Update the beacon distance so they can be healthy
    this ->_update_tag();
    this->_update_beacons();
}

int AP_Beacon_PozyxI2C::reg_read(uint8_t reg_address, uint8_t *pData, uint32_t size)
{
//    This preliminary test is a good precursory check but there's some weird logic in the macro
//    if(!IS_REG_READABLE(reg_address))
//        return POZYX_FAILURE;
// THIS WORKS !!!!
    WITH_SEMAPHORE(_dev->get_semaphore());
    bool status = this->_dev->read_registers(reg_address, pData, size)? POZYX_SUCCESS : POZYX_FAILURE;
    return status;
}

int AP_Beacon_PozyxI2C::reg_write(uint8_t reg_address, uint8_t *params, int param_size)
{
    uint8_t status;

    uint8_t write_data[param_size+1];
```

```
    write_data[0] = reg_address;
    memcpy(write_data+1, params, param_size);

    WITH_SEMAPHORE(_dev->get_semaphore());
    status = this->write_read(write_data, param_size+1, nullptr, 0);
    return status? POZYX_SUCCESS:POZYX_FAILURE;
}

int AP_Beacon_PozyxI2C::reg_function(uint8_t reg_address, uint8_t *params, int param_size, uint8_t *pl
{
    // Call a register function using i2c with given parameters, the data from the function is stored
    uint8_t status;

    // First let's write the relevant info
    uint8_t write_data[param_size+1];
    write_data[0] = reg_address;
    // As the pozyx folks said this feels clunky but it's probably better than for looping it for the
    memcpy(write_data+1, params, param_size);
    uint8_t read_data[size+1];

    WITH_SEMAPHORE(_dev->get_semaphore());
    status = this->write_read(write_data, param_size+1, read_data, size+1);
    if(status == POZYX_FAILURE) {
        return status;
    }
    memcpy(pData, read_data+1, size);
    // the first byte that a function returns is always it's success indicator, so we simply pass this
    return read_data[0];
}

bool AP_Beacon_PozyxI2C::write_bytes(uint8_t *write_buf_u8, uint32_t len_u8)
{
    WITH_SEMAPHORE(_dev->get_semaphore());
    return _dev->transfer(write_buf_u8, len_u8, NULL, 0);
}


int AP_Beacon_PozyxI2C::write_read(uint8_t* write_data, int write_len, uint8_t* read_data, int read_le
{
//    Core function write and read functionality
    bool status;
    WITH_SEMAPHORE(_dev->get_semaphore());
    status = _dev->transfer(write_data, write_len, read_data, read_len)?  POZYX_SUCCESS : POZYX_FAILU
    return status;

}
```

```
void AP_Beacon_PozyxI2C::_update_beacons(void)
{
    //Iterate and set the distance of the beacons if the tag is healthy
    if(this->healthy()) {
        for ( unsigned int i = 0; i < sizeof(this->_anchors) / sizeof(_anchors[0]); i++) {
            //TODO: there is some weird things happening here for with the DORANGING nonsense

            float x = this->_anchor_pos[i].x / 1000.0f;
            float y = this->_anchor_pos[i].y / 1000.0f;
            float z = this->_anchor_pos[i].z / 1000.0f;

            float distance = safe_sqrt(Vector3f(x, y, z).distance_squared(this->_curr_position));
            set_beacon_distance((uint8_t) i, distance);
        }
    }
}


void AP_Beacon_PozyxI2C::_update_tag()
{
    if(!this->reg_function(POZYX_DO_POSITIONING, nullptr, 0, nullptr, 1)){
        hal.console->printf("Error triggering positioning\n");
        return;
    }
    this->last_tag_update_ms = AP_HAL::millis();
    uint8_t interrupt_status;
    bool error = false;
    uint8_t error_code;
    this->reg_read(POZYX_INT_STATUS, &interrupt_status, 1);

    if(CHECK_BIT(interrupt_status, 0) == 1){
        this->reg_read(POZYX_ERRORCODE, &error_code, 1);
        hal.console->printf("Error Code =%x\n", error_code);
        error = true;
    }

    if(error){
        hal.console->printf("Error occurred, unable to position\n");
        return;
    }
    else {
        int32_t coordinates[3];
        if (this->reg_read(POZYX_POS_X, (uint8_t * ) & coordinates, 3 * sizeof(int32_t)) == POZYX_SUCCESS)
            Vector3f veh_position(
                    Vector3f(coordinates[0] / 1000.0f, coordinates[1] / 1000.0f, coordinates[2] / 1000.0f))
            this->_curr_position = veh_position;
            set_vehicle_position(veh_position, 0.2f); // I can get the xy covarince error buttttttt I 'm ju
```

```
            this->last_update_ms = AP_HAL::millis();
//            hal.console->printf("Vehicle position set\n");
        }

    }


}
```

### B.1.2 AP_Beacon_PozyxI2C.h

```
//
// Created by shivan on 7/14/20.
//
/*
    This work documented consists part of the MSc. Robotics Dissertation.
     Much of the ifc would be based on reading specific registers of the Pozyx tag since most of the proces
    already done onboard.
 */
#pragma once

#include "AP_Beacon_Backend.h"
#include "AP_Pozyx_definitions.h"
#include <AP_HAL/I2CDevice.h>


#define CHECK_BIT(var,pos) ((var) & (1<<(pos)))
//Simple macro for bit checking

class AP_Beacon_PozyxI2C : public AP_Beacon_Backend
{
public:
    // Constructor
    AP_Beacon_PozyxI2C(AP_Beacon &frontend);

    // return true if sensor is basically healthy (we are receiving data)
    bool healthy() override;

    // update
    void update(void) override;

    // This is essentially the main function call we'll use to get the relevant data from the pozyx
    //TODO: Check to see if this is already an implementation for this
    int reg_read(uint8_t reg_address, uint8_t *pData, uint32_t size);

    int reg_write(uint8_t reg_address, uint8_t *params, int param_size);

    int reg_function(uint8_t reg_address, uint8_t *params, int param_size, uint8_t *pData, uint32_t size);

    int write_read(uint8_t* write_data, int write_len, uint8_t* read_data, int read_len);

    bool write_bytes(uint8_t *write_buf_u8, uint32_t len_u8);

private:
//    uint16_t interval_ms = 100;

    // Lets have a member variable to store the hal i2c ifc
```

```
    AP_HAL::OwnPtr<AP_HAL::Device> _dev;

    uint32_t last_update_ms;
    uint32_t last_beacon_update_ms;
    uint32_t last_tag_update_ms;

    Vector3f _curr_position;

    //TODO: Look at coding these in a parameter, for the time being to avoid hard coding let's do the
//TODO: I have to leave this in for the time being for configuring the tag before use. Not sure why bu
    const uint16_t _anchors[4] = {0x6E2B, 0x676C, 0x670A, 0x6E22};      // the network id of the ancho
    const Vector3i _anchor_pos[4] = {Vector3i(0, 645, 1214), Vector3i(2737, -410, 1913),
     Vector3i(3651, 4120, 1853), Vector3i(1541, 4450, 1775)};

    // Setup the relevant stuffies for the Pozyx tag
    void _init(int8_t bus);

    void _update_beacons(void);

    void _update_tag(void);
};

//
```

### B.1.3 AP_Pozyx_test.cpp

```
/*
  simple test of I2C interfaces
 */

#include <AP_HAL/AP_HAL.h>
#include <AP_Beacon/AP_Beacon_PozyxI2C.h>
#include <AP_Beacon/AP_Beacon.h>
#include <AP_Common/Location.h>
#include <stdio.h>

void setup();
void loop();
void set_object_value_and_report(const void *object_pointer,
                                 const struct AP_Param::GroupInfo *group_info,
                                 const char *name, float value);

const AP_HAL::HAL& hal = AP_HAL::get_HAL();
static AP_SerialManager serial_manager;
AP_Beacon beacon{serial_manager};

// try to set the object value but provide diagnostic if it failed
void set_object_value_and_report(const void *object_pointer,
                                 const struct AP_Param::GroupInfo *group_info,
                                 const char *name, float value)
{
    if (!AP_Param::set_object_value(object_pointer, group_info, name, value)) {
        printf("WARNING: AP_Param::set object value \"%s::%s\" Failed.\n",
               group_info->name, name);
    }
}

void setup(void)
{
    set_object_value_and_report(&beacon, beacon.var_info, "_TYPE", 3.0f);
    set_object_value_and_report(&beacon, beacon.var_info, "_LATITUDE", 0);
    set_object_value_and_report(&beacon, beacon.var_info, "_LONGITUDE", 0);
    set_object_value_and_report(&beacon, beacon.var_info, "_ALT", 10);

    set_object_value_and_report(&serial_manager, serial_manager.var_info, "0_PROTOCOL", 13.0f);
    serial_manager.init();
    beacon.init();
}

void loop(void)
{
    beacon.update();
```

```
    Vector3f pos;
    Location origin;
    float accuracy = 0.2f;
    beacon.get_vehicle_position_ned(pos, accuracy);
    printf("Position: %f %f %f\n", static_cast<double>(pos.x),
     static_cast<double>(pos.y), static_cast<double>(pos.z));
    if (beacon.get_origin(origin)) {
        printf("Origin: %il %il %il\n", static_cast<double>(origin.lat),
         static_cast<double>(origin.lng), static_cast<double>(origin.alt));
    }
    else{
        printf("This is where it's dying\n");
    }

    for(int i = 0; i< 4; i++){

        AP_Beacon::BeaconState state;
        beacon.get_beacon_data(i, state);
        printf("Beacon %i, is %i with distance: %f\n", i, state.healthy,
        state.distance);

    }
    hal.scheduler->delay(100);
}

AP_HAL_MAIN();
```

APPENDIX C

# APPENDIX C

## C.1 Python EKF implementation

```
#!/usr/bin/env python
"""
This is a simple script file to read in pozyx tag position data and apply and ekf on it
since we're operating on a surface this will be a simple 2 state system.
"""

import pandas as pd
import numpy as np
import math
from filterpy.kalman import ExtendedKalmanFilter
import matplotlib.pyplot as plt

def fx(x, dt):
    return x

# Since we get X and Y the observation model is just the identity matrix
def H_of(x):
    return np.eye(2, 2)

def H_x(x):
    return x

def F_of(x):
    return np.eye(2, 2)
```

```
# We can update the measurement noise
def R_unc(x_e, y_e, xy_e):
    return np.array([[x_e, xy_e], [xy_e, y_e]])


if __name__ == "__main__":
    data = pd.read_csv("for_thesis/path_traingle_person_random.csv", skiprows=1)

    # time = data['time(s)'].to_numpy()
    x_measured = data['x'].to_numpy()
    y_measured = data['y'].to_numpy()
    z_measured = data['z'].to_numpy()

    x_err = data['x_err'].to_numpy()
    y_err = data['y_err'].to_numpy()
    xy_err = data['xy_err'].to_numpy()

    ekf = ExtendedKalmanFilter(dim_x=2, dim_z=2)
    ekf.x = np.array([x_measured[0], y_measured[0]])
    ekf.R = np.diag([0.1, 0.1])
    ekf.predict()
    estimates = []

    for i in range(1, x_measured.shape[0], 1):
        if math.sqrt((x_measured[i] - x_measured[i-1])**2 + (y_measured[i] - y_measured[i])**2) < 30:
            measurements = np.array([x_measured[i], y_measured[i]])
            ekf.update(z=measurements, HJacobian=H_of, Hx=H_x, R=R_unc(x_err[i], y_err[i], xy_err[i]))
            ekf.predict()
            estimates.append(ekf.x)

    ests = np.array(estimates)
    triangle_x = [1610, 2111, 1910, 1610, 1610]
    triangle_y = [2080, 2080, 2380, 2580, 2080]

    c_x = [1610, 1710, 1760, 1770, 1840, 1934, 1992]
    c_y = [2080, 2200, 2300, 2500, 2580, 2625, 2595]
    # print(ests.shape)
    # plt.plot(ests[:, 0], ests[:, 1], x_measured, y_measured, c_x, c_y, 'r')
    plt.plot(ests[:, 0], ests[:, 1], x_measured, y_measured, triangle_x, triangle_y, 'r')
    plt.xlabel('X (mm)')
    plt.ylabel('Y (mm)')
    plt.title('EKF vs Measured vs Path')
    plt.legend(['Estimated', 'Measured', 'Path outline'])
    plt.grid()
    plt.show()
```

```
    # print(x_measured[0], y_measured[0])
```

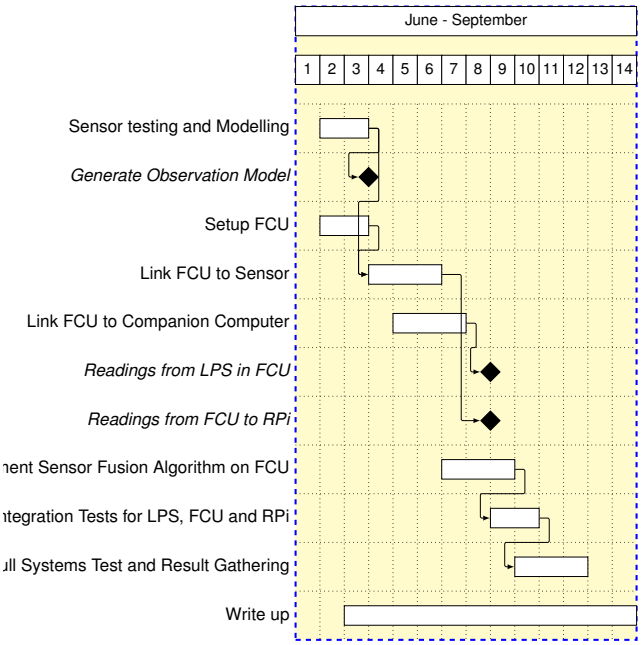

# APPENDIX D

Psuedo code for distance metrics

```
1   Generate line vector segments based on waypoints: line_vecs
2   Get a vector from the start of each line segment(line_vecs) vector to each point: line_to_recorded_vecs
3   dot_prod = dot(line_to_recorded_vecs * line_vecs / |line_vecs|)
4   projections = min(line_vecs / |line_vecs|, max(0, dot_prod))
5   norm_line_vecs = line_vecs / |line_vecs|
6   distances = length(line_to_recorded_vecs - projections*norm_line_vecs)
7   get_min_distance_for_each_point(distances)
```

APPENDIX E

# APPENDIX E




# APPENDIX F

# F.1 Ethical Review Checklist

University of the West of England
BRISTOL

**Faculty of Environment & Technology**
**Faculty Research Ethics Committee (FET FREC)**

**ETHICAL REVIEW CHECKLIST FOR UNDERGRADUATE AND POSTGRADUATE MODULES**
*Please provide project details and complete the checklist below.*

**Project Details:**

| | |
|---|---|
| **Module name** | **Dissertation** |
| **Module code** | **UFMED4-60-M** |
| **Module leader** | **Dr. Severin Lemaignan, Dr. Jonathan Winfield, Dr Maryam Atoofi** |
| **Project Supervisor** | **Professor Praminda Caleb-Solly** |
| **Proposed project title** | **Data-fusion for indoor localization for assistive robots.** |

**Applicant Details:**

| | |
|---|---|
| **Name of Student** | **Shivan Ramdhanie** |
| **Student Number** | **19044844** |
| **Student's email address** | **wj19041@bristol.ac.uk, Shivan2.Ramdhanie@live.uwe.ac.uk** |

| | CHECKLIST QUESTIONS | Y/N | Explanation |
|---|---|---|---|
| **1.** | Does the proposed project involve **human tissue, human participants, environmental damage, the NHS, or data gathered outside the UK**? | N | *If the answer to this is 'N' then no further checks in the list need to be considered.* |
| 2. | Will participants be clearly asked to give consent to take part in the research and informed about how data collected in the research will be used? | | |
| 3. | If they choose, can a participant withdraw at any time (prior to a point of "no return" in the use of their data)? Are they told this? | | |
| 4. | Are measures in place to provide confidentiality for participants and ensure secure management and disposal of data collected from them? | | |

# F.2 Ethical Review Checklist (cont'd)

| | CHECKLIST QUESTIONS | Y/N | Explanation |
|---|---|---|---|
| 5. | Does the study involve people who are particularly vulnerable or unable to give informed consent (eg, children or people with learning difficulties)? | | |
| 6. | Could your research cause stress, physical or psychological harm to anyone, or environmental damage? | | |
| 7. | Could any aspects of the research lead to unethical behaviour by participants or researchers (eg, invasion of privacy, deceit, coercion, fraud, abuse)? | | |
| 8. | Does the research involve the NHS or collection or storage of human tissue (includes anything containing human cells, such as saliva and urine)? | | |

Your explanations should indicate briefly for Qs 2-4 how these requirements will be met, and for Qs 5-8 what the pertinent concerns are.

- If Qs 2-4 are answered Yes (Y) and Qs 5-8 are answered No (N), no further reference to the Research Ethics Committee will be required, unless the research plan changes significantly.
- If any of Qs 5-8 are answered Yes (Y), then approval from the Faculty Research Ethics Committee is required *before* the project can start. Approval can take over a month. Please consult with your supervisor about the process.

**Your supervisor must check your responses above *before* you submit this form.**

**Submit this completed form via the *Assignments* area in Blackboard (or elsewhere if so directed by the module leader or your supervisor).**

After you have uploaded, your supervisor will confirm that the checklist above has been correctly completed by marking this form as Passed/100% via the *My Grades* link on the Blackboard Welcome tab.

**If your submitted answers indicate that further ethical approval is indeed required, then you must also send this completed form to ResearchEthics@uwe.ac.uk**

Guidance is available at http://www1.uwe.ac.uk/research/researchethics.
Further guidance can be obtained via the module leader, in the first instance, or the Department's Faculty Research Ethics Committee representatives, including your department's *AHoD for Research*.

# BIBLIOGRAPHY


Aerial Robotics IITK (n.d.), 'Visual inertial odometery', https://aerial-robotics-iitk.gitbook.io/wiki/estimation/setup-with-vicon.
Accessed : 2020-07-09.

Ardupilot (2019), 'Pozyx for non-gps navigation', https://ardupilot.org/copter/docs/common-pozyx.html.
Accessed: 2020-07-12.

Chadaporn, K., Baber, J. & Bakhtyar, M. (2014), Simple example of applying extended kalman filter.

Conceição, T., dos Santos, F. N., Costa, P. & Moreira, A. P. (2017), Robot localization system in a hard outdoor environment, *in* 'Iberian Robotics conference', Springer, pp. 215–227.

Custers, B. (2016), Drones here, there and everywhere introduction and overview, *in* 'The Future of Drone Use', Springer, pp. 3–20.

Deak, G., Curran, K. & Condell, J. (2012), 'A survey of active and passive indoor localisation systems', *Computer Communications* **35**(16), 1939–1954.

DeStefano, P. R., Siebert, C. & Widenhorn, R. (2019), 'Using a local positioning system to track 2d motion', *The Physics Teacher* **57**(7), 508–509.

Di Pietra, V., Dabove, P., Piras, M. & Lingua, A. (2019), Evaluation of positioning and ranging errors for uwb indoor applications., *in* 'IPIN (Short Papers/Work-in-Progress Papers)', pp. 227–234.

Ebeid, E., Skriver, M., Terkildsen, K. H., Jensen, K. & Schultz, U. P. (2018), 'A survey of open-source uav flight controllers and flight simulators', *Microprocessors and Microsystems* **61**, 11–20.

Jiménez, A. R. & Seco, F. (2017), Finding objects using uwb or ble localization technology: A museum-like use case, *in* '2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)', pp. 1–8.

Mendoza-Mendoza, J. A., Gonzalez-Villela, V., Sepulveda-Cervantes, G., Mendez-Martinez, M. & Sossa-Azuela, H. (2020), *Advanced Robotic Vehicles Programming*, Springer.

Mimoune, K., Ahriz, I. & Guillory, J. (2019), Evaluation and improvement of localization algorithms based on uwb pozyx system, *in* '2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)', pp. 1–5.

Pozyx Labs (2018*a*), 'Developer tag: Register overview', `https://www.pozyx.io/products-and-services/developer-tag/datasheet-register-overview`.
Last Accessed: 2020-08-12.

Pozyx Labs (2018*b*), 'Pozyx accurate positioning'.

pozyxLabs (2020), 'Pozyx-arduino-library', `https://github.com/pozyxLabs/Pozyx-Arduino-library`, note = Acessed: 2020-07-12.

Tsai, C.-C. (1998), 'A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements', *IEEE Transactions on Instrumentation and Measurement* **47**(5), 1399–1404.