ENG470 Engineering Honours Thesis

UWB Radiolocation Technology: Applications in Relative Positioning Algorithms for Autonomous Aerial Vehicles

Submitted to the School of Engineering and Information Technology, Murdoch University in partial fulfilment of the requirements for the degree of

Bachelor of Engineering Honours

in the discipline of

Instrumentation and Control Engineering, Industrial Computer Systems Engineering

Author
Mr Luke Beumont Barrett

Supervisor
Dr Gareth Lee PhD, BSc Hons

2nd July 2018
© Murdoch University & Luke Beumont Barrett 2018
Author’s Declaration

I, Luke Beumont Barrett declare that I am the sole author of this thesis, which to the best of my knowledge contains no material previously published, unless appropriate acknowledgement has been made. This thesis contains no material previously submitted for any degree at any tertiary education institution.

_______________________________

Abstract

This report presents methods and technologies used to investigate the use of commercially available Ultra-Wide Band (UWB) radiolocation technology in the positioning of autonomous vehicles. UWB radio has unique transmission characteristics which make pico-second resolution timing possible, a requirement for centimetre accuracy in positioning applications.

Firstly, an investigation into the performance capabilities of UWB positioning infrastructure was undertaken, followed by the development of relative positioning algorithms utilising a relative and decentralised control approach. These algorithms are based on feedback provided by radiolocation data in combination with sensor fusion elements, including inertial (accelerometers), optical (optical flow sensors) and absolute ranging sensors such as Light Detection and Ranging (LIDAR).

Performance of the UWB infrastructure was determined utilising static and dynamic measurement techniques to verify overall positioning accuracy and statistical characteristics, with respect to two different ranging algorithms, Time Difference of Arrival (TDOA) and Two Way Ranging (TWR).

Data collected from an array of static collection points has shown, in the recommended infrastructure configuration, that commercially available UWB technology has a positional accuracy with an average offset from the real coordinate system of 7.30-15.28%. Logic-based tests using statistical inputs indicate the TDOA algorithm succeeds in more accurate positioning in 79.63%, 58.33% and 85.19% of test cases across twenty-seven points in three vertical planes respectively.

Data collected from autonomous flight-path algorithms has provided practical insight into applications of the technology and its dynamic capabilities. Results demonstrate that in a high radio interference environment, performance of the UWB positioning system and drone achieve a 0.11% and 1.23% error on average from the programmed flight path gradient for TDOA and TWR respectively.

The results of this investigation suggest that the technology is ready and capable of being implemented on real-world systems, with the recommendation that the technology be coupled with sensor fusion elements in relative positioning applications.
Acknowledgments

I would like to express my gratitude to my supervisor and lecturer Dr Gareth Lee for his continued support and guidance throughout the undertaking of this project and past studies.

I would like to acknowledge and thank Professor Parisa Arabzadeh Bahri, Associate Professor Graeme Cole, Dr Linh Vu and Dr David Parlevliet for your dedication. The years of experience and knowledge you have imparted to myself and other students has a value of which cannot be quantified.

To Mr Iafeta “Jeff” Laava, Mr Graham Malzer and Mr Mark Burt thank you for your technical guidance and support throughout my studies.

Finally, to my family and friends, without you the past four years would have been far less enjoyable and your support during my studies was no small factor in getting me to this point in my academic career.
Table of Contents

Author’s Declaration ........................................................................................................................... ii
Abstract ............................................................................................................................................... iv
Acknowledgments .............................................................................................................................. v
Table of Contents ................................................................................................................................. vi
Table of Figures ....................................................................................................................................... ix
Table of Equations ................................................................................................................................. x
Glossary of Terms .................................................................................................................................... xi
1.0 Introduction ....................................................................................................................................... 1
2.0 Literature Review ............................................................................................................................. 3
  2.1 Mainstream Radiolocation Systems .............................................................................................. 3
  2.2 Indoor Radiolocation ..................................................................................................................... 3
  2.3 Indoor Positioning Systems ........................................................................................................... 4
  2.4 Technical Challenges .................................................................................................................... 4
  2.5 Radiolocation Techniques ............................................................................................................ 4
      2.5.1 Angle of Arrival .................................................................................................................... 4
      2.5.2 Received Signal Strength .................................................................................................... 5
      2.5.3 Time of Arrival ..................................................................................................................... 5
      2.5.4 Time Difference of Arrival ................................................................................................. 6
      2.5.5 Two Way Ranging ................................................................................................................. 6
  2.6 Wireless Technologies for Ranging .............................................................................................. 7
      2.6.1 WIFI ..................................................................................................................................... 7
      2.6.2 Bluetooth Low Energy ......................................................................................................... 7
      2.6.3 UWB ..................................................................................................................................... 8
  2.7 Recent Developments in UWB Technology ................................................................................ 9
  2.8 UWB Transceivers and Signal Processing Techniques ............................................................... 9
  2.9 Application Studies ..................................................................................................................... 10
  2.10 Remaining Avenues for Investigation ....................................................................................... 11
3.0 Background and Tutorial ............................................................................................................... 12
  3.1 Hardware ....................................................................................................................................... 12
      3.1.1 Crazyflie Drone .................................................................................................................... 12
      3.1.2 LPS Framework ................................................................................................................... 13
      3.1.3 Crazyradio Dongle ................................................................................................................. 14
3.1.4 Z-Ranger Deck

3.1.5 Optical Flow & LIDAR Deck

3.2 Software Development Environment

3.2.1 Virtual Machine

3.2.2 Eclipse Compiler

3.2.3 GitHub Project Link

3.2.4 Software & Hardware Relationship

4.0 Objectives

4.1 Evaluation of Radio Positioning Technology

4.2 Modified configurations Novel Approach

5.0 Firmware Development Approach

5.1 Mapping of Critical Functions

5.1.1 Writing Setpoints

5.1.2 StabiliserTask

5.1.3 SystemWaitStart

5.2 Setpoint Structures

5.2.1 crtp_commander_rpyt.c

5.3 Execution Structure

5.3.1 Task

5.3.2 Timer

5.4 Debugging Methods

5.4.1 LED Indicators

5.4.2 Positioning Decks

5.4.3 Development Environment

5.4.4 Visual Feedback - Flight Characteristics

5.4.5 Developer Support

6.0 Autonomous Program I Development

6.1 Lidar Program Functional Structure

7.0 Autonomous Program II Development - Centralised LPS

7.1 LPS Flight Path Program Functional Structure

8.0 Decentralised RPS

8.1 Non-trivial position finding utilising sensor fusion approach

8.1.1 Relative position to another agent
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.2</td>
<td>Relative coordinate space</td>
<td>37</td>
</tr>
<tr>
<td>8.2</td>
<td>Decentralised Relative Positioning Algorithms</td>
<td>38</td>
</tr>
<tr>
<td>9.0</td>
<td>Autonomous Program III Development - Decentralised RPS</td>
<td>41</td>
</tr>
<tr>
<td>10.0</td>
<td>Autonomous Program IV - Decentralised RPS Wand Control</td>
<td>44</td>
</tr>
<tr>
<td>11.0</td>
<td>Experimental Configuration</td>
<td>47</td>
</tr>
<tr>
<td>11.1</td>
<td>Test Environment</td>
<td>47</td>
</tr>
<tr>
<td>11.2</td>
<td>Statistical Measures and Relevance in Performance Assessment</td>
<td>48</td>
</tr>
<tr>
<td>11.3</td>
<td>Methods of data collection and validation of the approach</td>
<td>48</td>
</tr>
<tr>
<td>11.3.1</td>
<td>LPS Framework</td>
<td>48</td>
</tr>
<tr>
<td>11.3.2</td>
<td>Apparatus and Data collection</td>
<td>50</td>
</tr>
<tr>
<td>11.3.3</td>
<td>Collecting Static Data Points</td>
<td>51</td>
</tr>
<tr>
<td>11.4</td>
<td>Methods of Data Processing</td>
<td>52</td>
</tr>
<tr>
<td>12.0</td>
<td>Results</td>
<td>54</td>
</tr>
<tr>
<td>12.1</td>
<td>Performance analysis of TWR and TDOA</td>
<td>54</td>
</tr>
<tr>
<td>12.2</td>
<td>Positioning Offset from Real World Coordinate System</td>
<td>54</td>
</tr>
<tr>
<td>12.3</td>
<td>Normal Distribution Plots and Histograms</td>
<td>55</td>
</tr>
<tr>
<td>13.0</td>
<td>Results: Dynamic Flight Path Data Analysis</td>
<td>58</td>
</tr>
<tr>
<td>13.1</td>
<td>Quantitative Analysis</td>
<td>58</td>
</tr>
<tr>
<td>14.0</td>
<td>Results: Industrial and Cooperative Robotic Applications</td>
<td>62</td>
</tr>
<tr>
<td>15.0</td>
<td>Future Work</td>
<td>63</td>
</tr>
<tr>
<td>15.1</td>
<td>Accuracy of UWB Positioning Coupled with Flight Dynamics of a Larger Drone</td>
<td>63</td>
</tr>
<tr>
<td>15.2</td>
<td>Implementation of a Multi-Agent Swarm Utilising 3D Feedback Approach</td>
<td>63</td>
</tr>
<tr>
<td>15.3</td>
<td>Overlay 3D Maps onto LPS - Dynamic Mapping of Obstacles &amp; Flight Path Optimisation</td>
<td>63</td>
</tr>
<tr>
<td>15.4</td>
<td>Geometric Swarm Formations Using a Decentralised Anonymous Approach</td>
<td>63</td>
</tr>
<tr>
<td>16.0</td>
<td>Conclusion</td>
<td>65</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>Appendix A</td>
<td></td>
<td>72</td>
</tr>
<tr>
<td>Appendix B</td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>Appendix C</td>
<td></td>
<td>86</td>
</tr>
<tr>
<td>Appendix D</td>
<td></td>
<td>92</td>
</tr>
<tr>
<td>Appendix E</td>
<td></td>
<td>97</td>
</tr>
<tr>
<td>Appendix F</td>
<td></td>
<td>101</td>
</tr>
</tbody>
</table>
Table of Figures

Figure 1: Crazyflie Drone with the LPS Tag Configuration .................................................. 12
Figure 2: LPS Node ............................................................................................................. 13
Figure 3: LPS Tag .............................................................................................................. 14
Figure 4: Radio Dongle and 3D Printed Enclosure ............................................................. 15
Figure 5: Z-Ranger Deck .................................................................................................. 15
Figure 6: Optical Flow Deck ............................................................................................. 16
Figure 7: Ubuntu Virtual Machine Running on Oracle VirtualBox ..................................... 16
Figure 8: Hardware Interaction ......................................................................................... 18
Figure 9: Setpoint Type Definition Code Extract ............................................................... 24
Figure 10: Writing to Setpoints Within "crtp_commander_rpyt.c" ........................................ 24
Figure 11: Source Code Execution Map ............................................................................ 26
Figure 12: Auto-reload vs One Shot Timers ...................................................................... 28
Figure 13: Collapsed View of the Z-axis Flight Routine (Autonomous Program I) .......... 33
Figure 14: 3D Flight Path Routine (Autonomous Program II) ............................................ 35
Figure 15: 3D Representation of Radio Distance Data ....................................................... 40
Figure 16: 2D Representation of Radio Distance Data ....................................................... 40
Figure 17: Autonomous Program III Feedback Approach ............................................... 42
Figure 18: Relative Positioning Search Algorithm (Autonomous Program III) .................. 43
Figure 19: Trigonometric Positioning Approach ................................................................. 44
Figure 20: Radio Wand Control (Autonomous Program IV) ............................................... 46
Figure 21: Murdoch University Pilot Plant ......................................................................... 47
Figure 22: LPS Node Height Offset .................................................................................. 49
Figure 23: LPS Node Configuration (Nodes 3, 5 & 2 are at the front) ................................ 49
Figure 24: Inverted Triangle Geometry x y Plane (Coordinates in Metres) ......................... 50
Figure 25: x y Plane Data Collection Points ...................................................................... 51
Figure 26: y z Plane Data Collection Points ...................................................................... 51
Figure 27: PVC Data Collection Stand Centred in the LPS Framework .............................. 52
Figure 28: Aluminium Extrusion Construction for Mounting LPS Nodes ......................... 53
Figure 29: Histogram and Normal Distribution Plot for Positioning Error (Top Plane) .... 56
Figure 30: Histogram and Normal Distribution Plot for Positioning Error (Middle Plane) .. 56
Figure 31: Histogram and Normal Distribution Plot for Positioning Error (Ground Plane) .. 56
Figure 32: Hexagonal Flight Path ..................................................................................... 59
Figure 33: TDOA & TWR x, y Scatter Plot Hexagonal Flight Path (blue) with programmed flightpath (red) .......................................................... 60
Figure 34: Program I (part a) ............................................................................................ 72
Figure 35: Program I (part b) ............................................................................................ 73
Figure 36: Program II (Part a) ........................................................................................... 74
Figure 37: Program II (Part b) ........................................................................................... 75
Figure 38: Program II (Part c) ........................................................................................... 76
Figure 39: Program III (Part a) .......................................................................................... 77
Figure 40: Program III (Part b) .......................................................................................... 78
Figure 41: Program III (Part c) .......................................................................................... 79
Figure 42: Program III (Part d).................................................................80
Figure 43: Program IV (Part a) ........................................................................81
Figure 44: Program IV (Part b) ........................................................................82
Figure 45: Program IV (Part c) ........................................................................83
Figure 46: MATLAB Data Processing Code (Part a) ...........................................85
Figure 47: MATLAB Data Processing Code (Part b) ...........................................85
Figure 48: Field Notes, LPS Configuration and Measurement Point Selection ....101

Table of Equations

Equation 1: Heisenberg’s Uncertainty Principle Derivation ..........................8
Equation 2: Trilateration Equation Set ..........................................................39
Equation 3: Sine Rule ..................................................................................44
Equation 4: x Orbital Velocity ....................................................................45
Equation 5: y Orbital Velocity ....................................................................45
Glossary of Terms

AoA  Angle of Arrival

CMOS  Complementary Metal Oxide Semiconductors

FreeRTOS  A free Real Time Operating System

GPS  Global Positioning System

ISM  Industrial Scientific & Medical is a reserved unlicensed radio frequency band

LOS  Line of Sight

LPS  Local positioning system

LPS Node  A radio transceiver with fixed position and assigned coordinates

LPS Tag  A radio transceiver typically attached to a moving object for tracking

MICs  Microwave Integrated Circuits

MMICs  Monolithic Microwave Integrated Circuits

NLOS  Non-Line of Sight

Pitch  Rotation about the transverse axis

Roll  Rotation about the longitudinal axis

RPS  Relative positioning system

RSS  Received Signal Strength

Swarm  A coherent formation of many entities (typically drones) which act independently to achieve a shared task

TDOA  Time Difference of Arrival

Thrust  Propulsive force from propeller

TOF  Time of Flight

Transceiver  Transmitter and receiver
1.0 Introduction

Within the aerospace and defence industry accurate positioning systems are a crucial component in the control of next-generation unmanned aerial vehicles. Drones cannot rely on Global Positioning Systems (GPS) indoors, leaving limited options for accurate positioning without requiring expensive and computationally complex vision systems, or inertial navigation. In outdoor environments, GPS accuracy often limits the capabilities of cooperative or swarm behaviours with multiple drones [1].

Radiolocation is the determination of distance or position using radio waves. It has become a key piece of infrastructure across many industries to provide accurate positioning information: location services, indoor positioning, navigation, tracking of personnel and assets which constantly require improved accuracy and versatility [2].

Ultra-Wide Band (UWB) technology provides an excellent means for wireless positioning. Currently there has been little to no investigation into the implementation and suitability of UWB technology in relative positioning systems for cooperative robotic applications. This area of research is of great interest as it does not rely on either centralised computation or a fixed wireless infrastructure. UWB infrastructure also provides a solution to indoor positioning systems where GPS services are unavailable, or do not provide an acceptable level of accuracy.

This report presents the methods and technologies used to investigate the use of commercially available UWB radiolocation technology in the positioning of autonomous aerial systems.

The primary goal of this project was to assess the capabilities of UWB technologies in positioning autonomous systems in two topologies, centralised radiolocation infrastructure, and decentralised relative positioning systems. These two topologies have been investigated through the development of autonomous control algorithms utilising positioning data obtained from UWB radio location hardware.

The second objective of this project has been to investigate the feasibility of relative positioning systems and control algorithms which suit applications in cooperative robotics, such as drone swarm structures.
Firstly, an investigation into the performance capabilities of the UWB infrastructure was undertaken. The performance of the UWB infrastructure was determined using static and dynamic measurement techniques to verify overall accuracy. Dynamic measurements were achieved by developing autonomous flight routines which were dependant on the UWB infrastructure or Local Positioning System (LPS).

This was followed by the development of relative positioning algorithms, utilising a relative and decentralised control approach. These algorithms have been based on feedback provided by radiolocation data in combination with sensor fusion elements, including inertial, optical and absolute ranging sensors.

The results of this investigation suggest that the technology is suitable for application and capable of being implemented on real-world systems, however it is recommended to couple UWB ranging technology with sensor fusion elements in relative positioning applications where only scalar radiolocation data is available.
2.0 Literature Review

This section presents a review of radio-based positioning systems, methods and technologies best suited to indoor positioning and specialised cooperative robotic applications.

Typically, radiolocation is analysed in either an indoor or outdoor context, with each having unique challenges. Current mainstream radiolocation infrastructures such as the Global Positioning System (GPS) are not capable of providing detailed indoor positioning information which is required for increasingly dependent commercial and consumer technologies.

Various radio transmission techniques, network architectures, and algorithms have evolved to suit either of these radiolocation applications. The focus of this section is to provide the reader with a fundamental understanding of radiolocation based indoor positioning, challenges faced, and the existing and emerging wireless technologies suited to its application. A focus is placed on ultra-wideband radiolocation as the dominant technology present in recent literature on radio positioning and the fundamental technology applied in this report.

2.1 Mainstream Radiolocation Systems

The Global Positioning System (GPS) is the most mainstream of radiolocation technologies. Although GPS is effective for most outdoor positioning applications it is not suited to indoor applications as stated in numerous studies [3-6]. GPS suffers from low received signal power and low visibility of satellites. This is due to the presence of obstacles in the line of sight path between the satellites and the receiver which cause signal interference [7]. Non-GPS technologies include Ultra-Wideband, Bluetooth, and WIFI which will be discussed in detail in the following sections.

2.2 Indoor Radiolocation

Indoor positioning poses unique challenges for localization by radio technologies. This section summarises primary issues surrounding radio transmission properties in line of sight (LOS) and non-line of sight (NLOS) environments.

NLOS environments are when the direct LOS path between two radio transceivers is blocked. Reflections of the signal carrier reach the receiving node and can misrepresent the true direct path. More specifically the delay of the first arriving signal will not represent the true time of
flight (TOF). Since the signal has travelled along a different path, a positive bias known as NLOS error is present in the temporal measurement [8]. In the absence of any information about NLOS errors, accurate location estimation is impossible. Such errors are known as NLOS propagation errors.

Carrier bands can also be subject to multipath propagation. This is when multiple replicas of the signal are transmitted and are superimposed after different propagation delays, causing a discrepancy between the desired signal and the transmission template [8].

2.3 Indoor Positioning Systems
Indoor positioning systems are typically composed of wireless technologies categorized as Personal Area Networks. These system architectures are composed of either fixed or ad-hoc wireless infrastructures with a point to point or mesh-based functionality. Personal Area Network applications are primarily concerned with mobile devices, internet of things technologies and local peer-to-peer communications.

2.4 Technical Challenges
Indoor navigation requires high accuracy positioning for any functional application. This means ranging resolutions must have sub-meter resolution, imposing a technical challenge of precision timing. Timing of picosecond flight times is required for centimetre accuracy in ranging applications. Until recently this was not feasible from a technical standpoint in mobile devices.

2.5 Radiolocation Techniques
Radiolocation is achieved through the interpretation of Radio Frequency signals. The methods for distance calculations from these signals are summarized in this section.

2.5.1 Angle of Arrival
The Angle of Arrival (AoA) method determines distance though calculating the angle at which the signal arrives from the device to the reference node. AoA requires a minimum of two reference nodes, with two angles calculated, the region where the device can exist is determined.

As explored in Survey of Wireless Indoor Positioning Techniques and Systems [9] AoA systems are sensitive to angular multipath, a major effect in NLOS environments. Carrier bandwidth
and the size of the antenna array also significantly affects the resulting accuracy [8]. Typically, AoA is used only for localization in applications requiring low accuracy or use in combination with other measurements. AoA techniques are preferred in LOS, short-range environments due to low accuracy and multipath effects.

2.5.2 Received Signal Strength

Received Signal Strength (RSS) is a radiolocation method based on modelling the signal strength of wireless network infrastructures [9]. Signal Strength-based positioning is commonly implemented by either modelling the network signal strength "fingerprinting" or to calculate distance through the lognormal path loss model [8].

RSS fingerprinting is implemented by collecting RSS measurements of an environment which are mapped to a geolocation. The device then cross-references RSS reading to a database of corresponding positions. In this case, ambiguity is a risk and can result in positioning errors [10]. Mohammad Zahidul H. Bhuiyan [11] has demonstrated the accuracy of RSS methods can be improved by implementing them as part of a sensor fusion network.

Lognormal path loss or path loss shadowing models are mathematical models that predict signal strength loss over a distance in high occupancy environments. When distances are calculated trilateration is used to determine position [3].

2.5.3 Time of Arrival

The Time of Arrival or TOA approach makes distance calculations using temporal measurements of the propagation delay between the signal transmitter and receiver. In the context of a positioning system, this is between the device and reference points or nodes. In 3D positioning, the measurement places the device on the surface of a sphere with radius \( r \) (Where \( r \) is the measured distance). If 2D placement is assumed, the device is placed on a circle with the same radius.

To determine the absolute position of this device in a 2D plane, a minimum of three reference nodes are required. In a 3D case, the minimum number of nodes increases to four. The position of the device is then estimated to be at the point of intersection of the circles, or in the 3D case, spheres. ToA has its limitations in the requirement for infrastructure, three or four reference nodes for 2D or 3D positioning respectively [3].
2.5.4 Time Difference of Arrival

Time Difference of Arrival (TDOA) analyses differences in signal arrival times between multiple reference nodes [12]. Measurements are taken from numerous device-to-node pairs with known locations. The device acts as the signal transmitter and measurements are made at the nodes. Arrival times are broadcast to the array of nodes for distributed processing, calculating the difference of arrival times [13]. This compiled information is then transmitted to the device for positioning. From the measurement between each transmitter and reference node, hyperboloids where the device could lie are determined. The device’s position is then estimated at the intersection of the hyperbolic curves. This methodology is known as multilateration [14].

TDOA requires the fixed reference nodes to have synchronized time sources to determine the time difference from the same transmitter signal. The Transmitter may have an independent time source [3]. Clock jitter is a source of error in systems relying on synchronization as publicized in [15], thus high levels of accuracy must be achieved with the carrier wave.

2.5.5 Two Way Ranging

Two Way Ranging or TWR, similarly to TDOA uses time differences to calculate distance. Where TDOA uses a single transmission to take temporal measurements, TWR makes two measurements: One from the transmitter to the reference node where a time stamp is recorded, then a second from the receiver back to the transmitter. Recorded pairs of timestamps are used to compute the range between devices [13]. Information is distributed in the system before the ranging computation can be accomplished, the data is brought to a central computation node.

The difference between the counter start and stop times in the original transmitting device represents the total elapsed time from the departure of the first message to the arrival of the acknowledgement. The difference between the counter start and stop values in the responder represents the total elapsed time from arrival of the data message to the departure of the acknowledgement. After these values are all brought together at a common compute node, they are subtracted, the difference is divided by two, and the time of flight is known [13].
2.6 Wireless Technologies for Ranging

The primary technologies used in radio positioning belong to a family of wireless protocols used in Personal Area Networks. These include familiar technologies such as WIFI and Bluetooth. These technologies are described with reference suitability and common implementation techniques in positioning systems.

2.6.1 WIFI

Wireless Local Area Network (WLAN) is a midrange wireless network standard (IEEE 802.11) which predominantly operates in the 2.4GHz band. WLAN has a bit rate of 11-108 Mbps and a transmission range of 50 to 100 meters [16].

Indoor localization through use of WLAN-based infrastructure is problematic due to high variations in radio frequency signals indoors. Although most WLAN positioning systems use signal strength due to indoor signal characteristics [17], WIFI-based TDOA location solutions do exist [9].

Techniques based on radio frequency fingerprinting require extensive data collection of an environment’s RSS profile [18]. Path loss models and Fingerprinting based method are the fundamental methods for WLAN RSS measurement [19].

The path loss model approach, in comparison to RSS, is more easily implemented. However, due to the noise of wireless signals and the interference of indoor obstructions accurately measuring the distance based on signal attenuation is still difficult [5].

WLAN proves an appealing standard for use in indoor location due to the availability of pre-existing wireless infrastructure. The accuracy of WLAN positioning systems using RSS is approximately 3 to 30 m, with an update rate in the range of few seconds which makes it suitable for low accuracy applications.

2.6.2 Bluetooth Low Energy

Bluetooth is a digital radio standard using frequency hopping spread spectrum in the 2.4GHz Industrial Scientific and medical (ISM) band. Three power bands exist at 1mW, 2.5mW, and 100mW achieving a transmission range between 10 and 100m [16]. Devices with Bluetooth capabilities are prevalent with most mobile devices supporting the technology. As the infrastructure exists utilising it for the development of a positioning system is desirable.
Bluetooth is another technology which is best suited to RSS fingerprinting [18]. The Bluetooth Low Energy (BLE) standard enables transmission of a very short message which can be used to detect proximity to a specific location based on the RSS to the transmitting node.

BLE shares the same indoor propagation characteristics as 2.4 GHz WIFI transceivers. However, an advantage of BLE over WIFI is the flexible deployment of node infrastructure [20]. Unlike WIFI access points which require large power sources, BLE beacons are typically battery powered and can be placed anywhere. Due to this, planned deployment can result in a predictable RSS pattern.

2.6.3 UWB

Ultra-Wide Band Wireless or UWB is an impulse radio or base-band wireless communication system. It utilises short duration pulses, or chirps occupying a wide bandwidth [21]. UWB defined by IEEE 302.15.4a occupies the 3.1-10.6GHz frequency range. Transmission range varies between 50 and 500m in NLOS and LOS environments respectively [22].

The accuracy of time-based distance measurement techniques can be improved by increasing the Signal-Noise-Ratio or the effective signal bandwidth. As UWB signals have a very large bandwidth, this property allows extremely accurate location estimates using time-based techniques via UWB radios [8]. This can be described by a derivation of Heisenberg’s Uncertainty Principle as seen in Equation 1, which describes an inversely proportional relationship between the delta-t or in our case time measurement, and the transmission bandwidth delta-f [23]. Use of a temporal distance measuring algorithm is desirable as increased accuracy can be obtained. The range and signal characteristics make UWB an ideal carrier technology for ranging applications.

\[
\Delta f \Delta t \geq \frac{1}{4\pi}
\]

Equation 1: Heisenberg’s Uncertainty Principle Derivation

UWB wireless is currently the most capable technology for indoor positioning as supported by multiple authors [3, 22, 24]. UWB is an effective transmission technology due to low power consumption [25], high ranging accuracy, an intrinsic multipath invulnerability [8], simultaneous data transmission [26] and transmission range [22]. Additionally, due to low
power transmission even across a wide bandwidth, UWB causes almost no interference with other services [27].

2.7 Recent Developments in UWB Technology

This section summarises technical developments of fundamental UWB technologies. Although UWB carrier technology was discovered in the 1960s, transmitter, and signal processing technology has only recently reached the level of sophistication required to utilise UWB wireless.

2.8 UWB Transceivers and Signal Processing Techniques

Applied research into the field of UWB radiolocation has been heavily influenced by the evolution of microwave electronics. As discussed by Robert J. Fontana [24] UWB development started with the invention of time-domain sampling oscilloscopes, avalanche transistors and tunnel diodes in the 1960s and 70s. These were required for the implementation of an impulse radio system. Over 30 years the development of short-range radar and impulse radio sources [28] has resulted in generation techniques for impulse waveforms such as Step Recovery Diodes in Microwave Integrated Circuits (MICs) - achieving 250 pico-second pulse width. Recent digital techniques utilising time-gated oscillators in Monolithic MICs (MMIC), used in modern UWB transmitter technologies, have been shown to achieve UWB pulse width of 65 pico-seconds [29].

The evolution of microwave technologies has moved away from MICs and MMICs towards the use of smaller Radio Frequency Complementary Metal Oxide Semiconductors (RF CMOS), continuing to reduce power consumption, size, and performance [25]. An application advantage of CMOS technology is the ability to integrate digital signal processing on the same chip, which is not possible with MIC or MMICs [30].

Commercial products are now being developed by companies such as DecaWave [31] which utilise CMOS technology to implement UWB, for positioning applications.

Complete CMOS UWB transceivers such as that tested by Yuanjin Zheng [32] can achieve similar resolutions to the MMICs of 200 ps, equivalent of 3cm ranging accuracy in a positioning system. Recent developments in digital signal processing have enabled accurate measurements of sub-nanosecond time frames required for such systems. This is achieved using Field Programmable Gate Array implemented Time to Digital Converters like that
developed by MinZhang [33]. Analog sampling methods such as those explored by Cemin Zhang [30] achieve similar results.

2.9 Application Studies

Applications of UWB radio technology in high accuracy indoor positioning has been a focus of recent studies [1, 22, 27, 34]. The literature on derivative UWB hardware such as sub-nanosecond pulse generators and high-speed digital signal processors cover broad applications and technical approaches. However, research concerning fundamentally different packaged UWB technology and its use in radiolocation is limited. This is primarily due to no recent or significant technical developments in commercially available technology. The primary body of knowledge regarding indoor localization with UWB technologies is concerned with algorithm implementation and system architecture.

3D positioning system infrastructure based on an optimised approach has been described by Zheng, achieving a theoretical 10cm accuracy through TOA averaging on a fixed geometry node-tag infrastructure [34]. In UWB Multicell Indoor Localization Experiment System with Adaptive TDOA Combination [35] a new TDOA algorithm is implemented on a similar setup, in this case, applied to robot positioning. Results demonstrated less than 15cm accuracy in a 2D plane. Novel cooperative TWR ranging algorithms implemented by Marcin Kolakowski [12] have shown to improve positioning accuracy with results indicating a standard deviated error of 1.5cm.

Furthermore, high precision positioning for medical applications has been investigated. Mohamed R. Mahfouz [36] compare non-coherent and coherent implementations achieving results of 5.24–6.37-mm dynamic and Static 3-D accuracy of 0.73 mm respectively for each method. Other authors have discussed the integration of UWB into existing GPS in infrastructures for increased availability and indoor usage [1].
2.10 Remaining Avenues for Investigation

Ultra-Wide Band technology provides an excellent means for wireless positioning. Past applied research has followed the implementation of different data processing algorithms and varied infrastructure architecture. From this, achievable baseline accuracy of UWB positioning systems has been established, its broader limitations and capabilities identified.

Research involving UWB radiolocation follow common themes of the implementation of datacentric algorithms, focusing on the best use of radiolocation data, geometric implementation of wireless infrastructure in centralised or decentralised control strategies. These are currently the fundamental differentiators in research. Presently there has been little to no investigation into the implementation and suitability of UWB technology in relative positioning systems for cooperative agent applications. This area of research is of great interest as it does not rely on either centralised computation or a fixed wireless infrastructure.
3.0 Background and Tutorial

This section provides a brief overview of the commercially packaged technology that has been used during this project. It is to be noted at this stage for this project, the hardware has not been reconfigured to provide any additional functionality.

3.1 Hardware

3.1.1 Crazyflie Drone

The Crazyflie 2.0 Drone referred to throughout this report as the Crazyflie or Drone is an open-source quad-rotor platform for developers and researchers [37].

Crazyflie 2.0 is a quadrotor drone based on the ARM Cortex M4 microcontroller [38]. It has been used by numerous notable organisations for education, research, and development [39]. Measuring 9x9cm [37] it has a payload capacity of 15g [38] and a flight time of 7 minutes [38]. The drone is equipped with a radio communication module which can be used to send flight instructions from either a Bluetooth enabled mobile device or a computer with the Crazyradio dongle [40]. This is the chassis on which all programs were developed and tested on. The Crazyflie as shown below in Figure 1 may be configured with the Loco positioning tag which will be explained in the next section.

![Figure 1: Crazyflie Drone with the LPS Tag Configuration](image_url)
3.1.2 LPS Framework

The Loco Positioning System is a local positioning system (LPS) accessory for the Crazyflie [41]. It consists of radio transceiver nodes and tags (See Figure 2 and Figure 3). Nodes are statically positioned with a defined position or coordinate [42]. Tags are located on an accessory deck and fitted to the drone [43] (See Figure 1). The radio transceivers are based on the Decawave DWM1000 module [44].

The LPS system uses Two Way Ranging or Time Difference of Arrival (TDOA) algorithms to calculate the position of a drone with an attached Loco tag. The onboard microcontroller calculates this position using temporal data between the reference nodes with defined positions, and the Loco radio Tag. The transceiver deck communicates with the LOCO nodes, acquiring positioning data. The modules have a baud rate of 6.8 Mb/s, support large numbers of transceiver decks, have an integrated antenna, power management system and clock signal source [42].

The Radio positioning hardware is based upon the DMW1000 module which can be configured in one of three operating modes – sniffer, node and tag. The module itself is a UWB transceiver which is capable of timing time of flight of radio transmissions in picosecond resolution [44].
In all demonstrations, both radio "node" and radio "tag" modes have been used in a pair. In theory it is possible to use the TDOA mode to calculate the distance to other transmitting nodes rather than to a tag. Alterations to firmware would have been required for this and due to payload limitations of the Crazyflie, a node-node system could not be flown. This does not compromise the underlying concept of the data obtained over communications channels, and node-tag pairs perfectly imitate characteristics of any node-node or hybrid node-tag cycling hardware. See Appendix E for email correspondence regarding this potential development.

A guide on the configuration of the Loco positioning system, including recommended geometry for the placement of the nodes, is available through the Bitcraze getting started guide [45].

### 3.1.3 Crazyradio Dongle

The Crazyradio Dongle is a USB radio transceiver which connects to the host computer [40]. This is typically the point of interface between the Crazyflie and the Python Client which is used to send commands, display, and log flight data. Refer to Figure 4.
3.1.4 Z-Ranger Deck

The Z-ranger deck is a vertical ranging accessory which utilises a laser ranging sensor or LIDAR. By measuring the time of flight of the laser light to a surface the distance can be calculated [46].

3.1.5 Optical Flow & LIDAR Deck

The Flow deck, as seen in Figure 6 is an optical sensor accessory which can determine the horizontal velocity ($x$, and $y$) [47]. This is done by detecting movements relative to a surface underneath the optical sensor. Similar technology is used in optical computer mice. The Flow deck also incorporates the same LIDAR sensor as used on the Z-ranging deck (Figure 5).
3.2 Software Development Environment

3.2.1 Virtual Machine

A Virtual Machine is provided by Bitcraze as a complete development environment [48]. It comes in the form of a virtualised Linux distribution (Ubuntu) [49]. The distribution comes pre-configured with a Python [50] drone control client, complete code project files, and an open source compiler Eclipse [51].
3.2.2 Eclipse Compiler

Eclipse is the compiler and software development program used to create and edit the Crazyflie firmware [51]. The Crazyflie, Loco modules, and Crazyradio firmware can be customised and further developed through eclipse or similar development environments. Once compiled new or adapted firmware can be flashed to the peripheral device using the radio dongle.

3.2.3 GitHub Project Link

GitHub is an online code repository and management system [52]. The firmware developed by Bitcraze and wider developer community is available through GitHub. A link to update the project files directly from GitHub is available on the Linux desktop. For this project the firmware was updated to the most recent release prior to commencement in August 2017. A second update was undertaken during final testing and implementation in April 2018.

3.2.4 Software & Hardware Relationship

The overall interaction between the above described hardware can be simply described as a series radio communication steps. A simple hypothetical LPS framework as depicted in Figure 8. Here we can see the LPS frame work consists of three LPS Nodes. These will have defined coordinates. LPS nodes communicate to each other using their own UWB transceiver, the DMW1000 module. This line of communication is to synchronise their clocks when the TDOA algorithm is used. Secondly each LPS node can communicate with the LPS Tag on the Crazyflie drone. This is separate and isolated from the radio communications over 2.4GHz radio which is used to send setpoints from a computer, through the Crazy Radio Dongle.

It is important to note that the firmware on the Crazyflie has been developed to be controlled by a centralised computing station which broadcasts commands. Autonomous functionality has not been incorporated. Additionally, the firmware has not been designed for use with the LPS system and sensor decks simultaneously for position control. Methods are described later in this report to address and overcome these limitations.
Figure 8: Hardware Interaction
4.0 Objectives

The primary goal of this project was to assess the capabilities of UWB technologies in positioning autonomous systems in two topologies, centralised radiolocation infrastructure (LPS), and decentralised Relative Positioning Systems (RPS). These two topologies were to be investigated through the development of autonomous control algorithms utilising positioning data obtained from UWB radio location hardware. Secondly to investigate the feasibility of the implementation of such positioning systems and control algorithms to applications in cooperative robotics such as drone swarm structures.

4.1 Evaluation of Radio Positioning Technology

The first phase of this project was aligned with gaining an understanding of the UWB radiolocation technologies selected for use. This included research into peripheral technologies utilised in commercial drones such as sensor fusion suites. In parallel, an in-depth reading of the open source code was undertaken to identify key functions, structure, and points of hardware interaction which would be required for further code development.

Phase two was aligned with the development of autonomous programs on the Crazyflie. These were used to demonstrate autonomous capabilities, and to evaluate the dynamic performance of UWB radiolocation infrastructure. This was achieved through the collection of positioning data both during flight (dynamic accuracy) and at static collection points. As the UWB technology has two ranging algorithms TDOA and TWR, an assessment of both was undertaken.

4.2 Modified configurations Novel Approach

The final phase of the project was the investigation and development of relative positioning algorithms for decentralised relative positioning systems, with applications to cooperative robotic control. The requirements of cooperative robotic control, in applications such as swarm control algorithms, are based on simple objectives such as distance control between agents or robots in a formation. In a single dimension, this can be described as a “follow the leader” type function, consisting of a two-agent swarm. A base objective in this phase is to develop a control algorithm which can execute this type of functionality in two or more dimensions using relative positioning data. Hypothetical extensions of the algorithm are to be discussed with points of merit noted.
5.0 Firmware Development Approach

The firmware in this project was developed in three stages, each investigated in parallel.

1. Analyse the existing source code and identify core functions

Analysis of the code was done by setting an objective function and incrementally developing a program which successfully compiled and executed. In this instance the first step of development was attempting to write a thrust setpoint, this would simply prove that the program was able to write a setpoint to the drone’s onboard controller. Functions from the source code were identified by looking for similar setpoint writing operations and mapping execution (refer to Figure 11).

2. Develop an execution structure in which functions can be executed in real-time

Task and timed loop structures were investigated to provide a suitable environment for the code to be repeatedly executed. During early code development, existing periodic execution structures within the source code were used to test basic functions. Through reverse engineering existing functions, with reference to the FreeRTOS manual [53], a suitable execution structure was constructed.

3. Develop algorithms defined as functions and iteratively develop and build on functionality to achieve objectives

This approach was used in the development of all programs used for autonomous flight of the Crazyflie. The approach taken to the development of autonomous firmware was to build on functionality incrementally. Algorithms were iteratively developed and debugged until operational. From this point, algorithms would be re-written as functions and integrated into the main code files.

The firmware developed has primarily been constructed by reverse engineering existing source code. The functionality required to write setpoint values was understood by examining the structure of functions within the existing firmware, and the subsequent functions that are called as a result. The main execution path of source code discovered relating to the writing of setpoints is described in the flowchart in Figure 11, it is examined with mention to relevant findings in this section.
5.1 Mapping of Critical Functions

5.1.1 Writing Setpoints

The steps involved in writing a setpoint to the controller of the Crazyflie is described with reference to Figure 11 with extracts referred to using bold alphabetical notation. Note not all code is visible in the figure.

1. The Stabiliser.c file contains a Task StabiliserTask this repeatedly executes code labelled A
   a. The function “CommanderGetSetpoint” is called

2. “CommanderGetSetpoint” executes code labelled B
   a. The function “CommanderGetSetpoint” uses a queue variable “setpointQueue”

3. By searching for instance of “setpointQueue” the function “CommanderSetSetpoint” is found in C
   a. “CommanderSetSetpoint” takes a variable of type setpoint_t and integer. One holds the structure of setpoint values and the other a priority for the setpoint origin i.e. radio broadcast from the python client or Bluetooth mobile phone application.

4. Searching for instance of “CommanderSetSetpoint”, the function in D is responsible for receiving a radio communication packet from a specified port, decoding it and then writing it to the “setpointQueue” variable utilising CommanderSetSetpoint.
   a. This function is executed in a task setup and initiated in E and F.

From this example it can be seen the process to write a setpoint is as follows:

1. Define a setpoint structure within a “setpoint_t” variable type.
2. Write the “setpoint_t” variable to the “setpointQueue” using “CommanderSetSetpoint” and Write a corresponding setpoint priority value.
3. This will then be read by the “CommanderGetSetpoint” function within the StabiliserTask

This is the structure used to implement the core functionality within the autonomous programs to be described in more detail in later chapters.
5.1.2 StabiliserTask
GetExtPos first called within A has been used to enable and disable the Crazyflie’s external navigation functionality. The Crazyflie defines sensory inputs from hardware such as a sensor deck as “internal positioning” and other systems such as the Loco Positioning System as “external”. The Crazyflie has not been developed to use both internal and external positioning systems simultaneously and will toggle between using one or the other depending on the connection status of either system. To investigate the use of sensor fusion, radio location, and sensor deck feedback, it was necessary to find a workaround. Simply commenting out this function disabled the Loco positioning system data in the control algorithms allowing for use of both internal and external positioning.

5.1.3 SystemWaitStart
SystemWaitStart is a simple function which is used within almost all Tasks in the source code. It is a logical check to ensure before executing any hardware dependent functions, the system has undergone full initialisation. This is defined in system.c where the system is initialised.

5.2 Setpoint Structures
Having gained an understanding of how setpoints were written, the correct setpoint structure needed to be identified. Different Setpoint Structures are utilised depending on the hardware configuration of the drone. This is a function of what positioning feedback the drone has access to, inertial or absolute.

The Crazyflie without modification operates on an inertial sensor suite which includes gyroscopes and accelerometers. Thus, without modification, all setpoints are in velocity form. Absolute setpoint structures require additional hardware to be integrated into the Crazyflie. This includes either the flow, LIDAR deck, or the LPS radio tag. Each of these devices provides one or more dimensions of positional data. Refer to Table 1 for sensors and associated setpoint form.
### Table 1: External Positioning Deck Setpoint Form Summary

<table>
<thead>
<tr>
<th>Axis Feedback and Setpoint Form</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>External Deck</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flow Deck</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Z-ranger deck</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LPS Tag</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Setpoint Variable Structure: Within the firmware, a variable structure type "setpoint_t" is used to hold, read and write setpoints for the internal controller. This was the initial point of an investigation into how setpoints can be written and the various modes of operation. The structure consists of the definition of \( x, y, z \), Roll, Pitch, and Yaw variables. This definition provides the mode (absolute or velocity/rate).

Within "stabilizer_types.c" the type definition of the setpoint structure is defined. From this, we can see the structure contains the elements as outlined in Table 2.

### Table 2: Setpoint "setpoint_t" Structure

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint32_t</td>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td>attitude_t:</td>
<td>attitude, attitudeRate</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>thrust</td>
<td></td>
</tr>
<tr>
<td>point_t</td>
<td>position</td>
<td>vec3_s ((x, y, z))</td>
</tr>
<tr>
<td>velocity_t</td>
<td>velocity</td>
<td>vec3_s ((x, y, z))</td>
</tr>
<tr>
<td>bool</td>
<td>velocity_body</td>
<td></td>
</tr>
<tr>
<td>stab_mode_t</td>
<td>(x, y, z), roll, pitch, yaw</td>
<td>modeDisable, modeAbs, modeVelocity</td>
</tr>
</tbody>
</table>
The setpoint structure was primarily reverse engineered by comparing setpoint structures used throughout the source code. The file “crtp_commander_rpyt.c” contained a large series of setpoint cases which are implemented depending on the intended flight mode. These modes include, “Altitude Hold”, “Position Hold”, “Position Set”. For the intent of this project the Position Set mode was further analysed.
The setpoint structure is defined in two stages, firstly the mode of operation is defined and then parameters are assigned the setpoint value.

When the mode is switched to “position hold” the setpoint modes for $x$, $y$ and $z$ are set to \texttt{modeAbs}. Roll and pitch are disabled, and yaw is set to \texttt{modeAbs}. This indicates the data for the setpoint is to be in absolute form for the reference environment. Following this, the setpoint position parameter is used to define the values of the $x$, $y$, $z$, roll, pitch, yaw and thrust setpoints.
5.3 Execution Structure

5.3.1 Task

From mapping the code as discussed in section 10.1, it was concluded that the use of the function `SetSetpoint` was periodically executed within a defined task. This was approached as the execution structure for the autonomous code development.

A Task is an execution structure which behaves like a separate program, once entered it will typically run continuously. Tasks are used as a method to have programs executed in a multithread. This means that if there are three defined tasks, all three will be executed in parallel. This functionality is a significant advantage in a complex system such as a drone, where many programs must run simultaneously to manage power, control flight, and maintain radio communication with external controllers.

By reverse engineering the structure of Tasks defined within the code, and by following definition examples from the FreeRTOS manual [53] a Task was created in a separate `.c` file and successfully compiled.

The task was used to develop the first successful flightpath utilising the Z-ranger deck. This involved the writing of x and y velocities (0ms⁻¹), and absolute z-axis setpoints (an array of values ascending to 1m).

Ultimately a timer was used for the execution of setpoints. This was due to issues encountered when running programs within the LPS framework. Symptomatic flight behaviours such as multiple failures at the beginning of the flight path structure, and then “jumping” into the sequence of execution indicated the program was not being executed as programmed. Following correspondence with the developers at Bitcraze, who confirmed the task structure used was syntactically correct, the bugs in the code were likely related to collisions with other tasks within the firmware. This presumption can also be argued on the basis when the Task was running radio communications with the PC client were non-functional, which was another task managed function.
Following this conclusion, from a recommendation by Bitcraze, a timer was selected as the execution structure for the autonomous code. This execution method had been used by Bitcraze in a previous demonstration of some autonomous capabilities.

5.3.2 Timer

A timer is a software timed call-back function which executes from start to finish followed by exiting the sequence of execution. All timers that are defined are executed through a task which is defined when FreeRTOS scheduler is started. There are two main types of timer, one shot, and auto-reload timers. One-shot timers execute once and exit. Auto reload timers execute and restart on completion to achieve periodic execution [54]. Figure 12 represents the execution differences between auto-reload and one-shot timers.

![Figure 12: Auto-reload vs One Shot Timers](image)

The timer structure in comparison to a task is far more simple to implement. However, the drawbacks of using a timer are that variables are only temporarily stored as it is periodically executed not continually running like in a task. Also, no functions can be called which result in the function entering a blocked state, such as the vTaskDelay function.

Implementation examples of timers can be found in the FreeRTOS manual [54]

The firmware developed in this project utilised an auto-reload timer setup within a separate .c file as in the previous task-based implementation. With the timer-based execution of the autonomous programs developed, the main program is defined within the call-back function. This houses all main functions and is periodically executed at a definable rate.

To overcome the “memory loss” caused by periodic execution static variables are created to store values between loops. This also requires initialisation from the system.c file on start-up of the drone. This ensures a known starting condition which is typically setting all static
variables to zero. This is particularly crucial for index variables used to keep track of the loop execution count.

5.4 Debugging Methods

In-flight the Crazyflie has no existing method of sending error messages for debugging. The drone is capable of such functionality but would require significant development work to send messages over radio. As this was the case a combination of visual feedback, controlled experiments, and hardware simplifications were used to debug and develop code. This section briefly summarises some of the techniques and examples of implementation.

5.4.1 LED Indicators

By programming LEDs to indicate the stage of loop execution, a base understanding that the program was functioning in sequence could be established. This was helpful in bench testing execution structure without including flight setpoints. As a part of the Crazyflie's built-in functionality a small number of error messages are sent as a red LED sequences [55], this was for the most part only used to see that the battery was low or that the firmware had not been properly flashed to the drone.

5.4.2 Positioning Decks

During development work the lidar [46] and flow deck [47] were used to trial setpoint structures. This eliminated variability introduced by the radio positioning system. The use of these decks also aided in rapid development as it was easier to test without full infrastructure setup and configuration.

5.4.3 Development Environment

Using a consistent development environment was one way to eliminate potential code changes during development. As the firmware constantly undergoes updates from the wider community, it was not until the entire program was successfully tested that a copy of the Virtual Machine was created, and the source code library updated. This was to ensure if significant changes to the code had taken place the functionality implemented would still be able to be demonstrated. On testing, the code seamlessly integrated into the updated version. At this stage the new version included a more developed TDOA functionality (no longer experimental version) and allowed for the positioning nodes to be toggled between
TDOA and TWR firmware. This significantly reduced time in transitioning tests between TDOA and TWR setups in the data collection phase.

5.4.4 Visual Feedback - Flight Characteristics
During the testing of autonomous flight path programs, visual feedback was used to make inferences on program execution and then debug the code.

LPS Initialisation and Stability
During early stage testing of flight paths within the LPS (using TWR) the drone would start-up and take off poorly, before executing setpoints. By manually holding the drone and making it abort start-up until placed down, it was realised longer durations before start-up allowed the Crazyflie to take off in a more stable manner. This lead to the theory that the stability was a function of the onboard Kalman estimator [56] requiring a certain period of initialisation. This symptom appeared to be less severe with the TWR algorithm as the same routine run with the TDOA algorithm would crash. This could be explained by the difference in position feedback variance between the two algorithms. After implementing a delay before initialising the setpoint routine this flight characteristic was rectified in both TWR and TDOA.

Task Collision
In the previous section, an issue with task collision was mentioned. Visual feedback was the debugging method used in which this was diagnosed, followed by verification from Bitcraze. This resulted in the implementation of a timing structure instead of task-based program execution.

Yaw Rotation
During flightpath testing, when placing the drone down to start it, the drone would take off but begin to rotate before becoming unstable and crash. It was believed at the time there may have been an issue with how the yaw setpoint was being written. After multiple trials, it was found that when placed down before initialisation, if not facing forward in the positive x-direction the drone would spin. When verifying this with Bitcraze it was found this start-up position was required for proper initialisation of the onboard positioning system.
5.4.5 Developer Support

During the project through email correspondence, the developers were contacted to verify that the scope of the project was feasible based on the existing capabilities of the Crazyflie drone and Loco Positioning System. Following this, during the project the developers were contacted for support when attempts to diagnose issues with firmware failed. Email correspondence to Bitcraze can be found in Appendix E.
6.0 Autonomous Program I Development

The first fully developed autonomous program was designed to utilise the Z-ranger deck. This was to ascertain that the execution structure and method developed for writing consecutive setpoints was valid. By using the Z-ranger deck, potential variability that could be introduced with the LPS was eliminated. The Z-ranger deck takes direct measurements where-as the LPS takes relative measurements then compares to a defined reference space from which there can exist offset.

By limiting the program to only write absolute positioning setpoints in the z-axis, and set x, y setpoints to velocity at 0ms\(^{-1}\) simple functionality could be proven. To minimise variability in experimental flight tests, a nylon wire was threaded through the Printed Circuit Board mounting holes in the Crazyflie frame which acted as a non-restrictive tether in the Z axis. This kept the Crazyflie centred on the x, y plane, an acceptable control as accelerometers are subject to drift and this was not an x, y positioning evaluation.

Summarised program development decisions can be found in Table 3.

<table>
<thead>
<tr>
<th>Function</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight Path System</td>
<td>All values in a single array</td>
<td>Major setpoints, compute intermediary points</td>
<td>Generated by ruleset on initialisation</td>
</tr>
<tr>
<td>Code Container</td>
<td>Existing .c file in modules library</td>
<td>Dedicated .c file</td>
<td></td>
</tr>
<tr>
<td>Execution Structure</td>
<td>Timer</td>
<td>New dedicated Task</td>
<td>Loop in existing Task</td>
</tr>
<tr>
<td>Rate of execution</td>
<td>100ms</td>
<td>Realtime controller loop speed</td>
<td>Same as radio communication protocol</td>
</tr>
<tr>
<td>Code Structure/Philosophy</td>
<td>Function based structure</td>
<td>single loop, large executable script</td>
<td>separate code files</td>
</tr>
</tbody>
</table>

Table 3: Program Philosophy Development Matrix (Grey boxes indicate option selection)
6.1 Lidar Program Functional Structure

1. Firstly, the setpointbuilder function is called
   a. This takes the current flight path index value and uses it to extract the flight path array value for this time step
   b. The modes for x, y, z are set as abs, velocity, and velocity respectively
   c. The values for x, y are set to zero
   d. The value of z is set based on the currently indexed flight path value
   e. The temporary setpoint values are written to the external setpoint variable

2. The CommanderSetSetpoint function is called and the setpoints for this loop iteration are written to the commander

The controller uses the direct LIDAR measurements for positioning the Crazyflie within the vertical axis. The primary function of this program was to accurately execute a series of pre-programmed setpoints in the z-axis. The program was developed as a practical demonstration of the setpoint writing functionality before attempted implementation in the LPS.

Collapsed function view of this program can be seen in Figure 13. Refer to Appendix A Figure 34 & Figure 35 for the full source code.

```c
// INCLUDES GO HERE ****************************
#include "FreeRTOS.h"
#include "commander.h"
#include "system.h"
#include "autotest.h"
#include "task.h"
#include "led.h"

// VARIABLES AND CONSTANTS  *******************
int fpathindex;

// TASK DEFINITION
#define void autotestTask(){
#define void setpointbuilder(setpoint_t *setpoint, int *index){
#define void autotestInit(){
  xTaskCreate(autotestTask, AUTO TEST TASK NAME, AUTO TEST TASK STACKSIZE, NULL, AUTO TEST TASK PRI, NULL);
}  

Figure 13: Collapsed View of the Z-axis Flight Routine (Autonomous Program I)
7.0 Autonomous Program II Development - Centralised LPS

This program was developed to work within the fixed positioning infrastructure, demonstrating autonomous capabilities within a radio positioning infrastructure. The primary function of this program was to accurately execute a series of pre-programmed setpoints on start-up with no additional hardware such as the flow or Z-ranger deck. The program was also required to work with both TDOA and TWR algorithms. This would demonstrate the capability of an autonomous system within a fixed positioning infrastructure and the potential for its use in supporting multiple drones for swarming applications.

The flight path program was developed as a practical demonstration of the capabilities of the Crazyflie when operating autonomously while supported by the fixed location positioning system. The Crazyflie starts at a central point within the LPS infrastructure, where it is initialised. Following initialisation, the drone begins to iterate through three setpoint arrays, one for each axial setpoint. Each setpoint array has three stages of execution. The first stage is responsible for take-off and initial planar positioning. The second stage takes the Crazyflie from the first planar point and starts the geometric setpoint sequence, in this case a hexagon. This geometric pattern was ideal as it involved varied $x$, $y$ positioning and required smooth and accurate transition between points. The Final stage of the program is simply a conditional check which identifies the pattern is nearing completion and the index variable should be initialised at the point of origin. This allows the flight path to be repeated continuously until the battery fails.

7.1 LPS Flight Path Program Functional Structure

The program structure is based on a simple series of executable functions running inside a loop periodically executed through use of an auto-reload timer.

1. **Firstly, the setpointbuilder function** is called, this in turn using the Coordinate index “Coordindex”, will generate the $x$, $y$ and $z$ setpoints. Every 100th loop iteration increments the coordinate which the drone must fly towards.

2. This coordinate value is approached by taking 100 increments between the current position and the coordinate. This correlates to a one hundred 100ms increments.

3. These intermediate steps become the setpoint that is written to the onboard control system. This is done using an existing function “CommanderSetSetpoint” which sends the setpoint to the stabiliser routine.
Several conditional checks are made to determine if the setpoint must increment or decrement $x$, $y$ and $z$ values.

All repetitive tasks were developed in function form such that the same task could be executed for $x$, $y$ and $z$ setpoints with ease.

Collapsed function view of this program can be seen in Figure 14. Refer to Appendix A Figure 36, Figure 37 & Figure 38 for the full source code.

```c
#include "FreeRTOS.h"
#include "commander.h"
#include "system.h"
#include "autotest.h"
#include "task.h"
#include "led.h"
#include "timers.h"
#include "log.h"
#include "param.h"

// INCLUDES GO HERE ******************************

static xTimerHandle timer;

# void QuadrantVelocity(int 0, float *xVel, float *yVel, float mapVel)[]
// Quadrant conditional check
# void quadrantCondition(bool xState, bool yState, int Fastindex, int *Quadrant)[]
extern lpsAlgoOptions_t algoOptions;
float radiusdata;
static bool xmemory;
static bool ymemory;

# void setpointbuilder(setpoint_t *setpoint, int indexFast, int *Qmemory, float *Gmemory)[]
extern int fpindex;
extern int Coordindex;
extern bool togglebit;
extern int StartindexxWke;
extern float Radiusmemory;
extern int QuadrantMemory;

# void autoTimerCallback(TimerHandle_t timer[]

# void autotestInit()[]
```

Figure 14: 3D Flight Path Routine (Autonomous Program II)
8.0 Decentralised RPS

Following the successful implementation of three-dimensional control of the Crazyflie within the LPS, and evaluation of LPS infrastructure, attention was shifted to adapting the infrastructure into a point to point relative positioning system. At first from correspondence with Bitcraze, it was anticipated point to point positioning data would need to be developed utilising the time synchronisation properties of the TDOA algorithms to calculate distance. Although this is recommended in the future works chapter, on investigation it was found there was sufficient data available within the existing node-tag TWR system to implement the relative point to point system.

<table>
<thead>
<tr>
<th>Function</th>
<th>Option 1</th>
<th>Option 2</th>
<th>Option 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ranging Data</td>
<td>TWR Node-Tag Pair</td>
<td>TDOA Node-Node Pair</td>
<td>NA</td>
</tr>
<tr>
<td>Positioning x</td>
<td>Flow deck</td>
<td>Z-Ranger Deck</td>
<td>Accelerometers</td>
</tr>
<tr>
<td>Positioning y</td>
<td>Flow deck</td>
<td>Z-Ranger Deck</td>
<td>Accelerometers</td>
</tr>
<tr>
<td>Positioning z</td>
<td>Flow deck (Using the LIDAR component)</td>
<td>Z-Ranger Deck</td>
<td>Accelerometers &amp; Barometer</td>
</tr>
<tr>
<td>Wayfinding</td>
<td>UWB Ranging Data</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

It should be noted the development of the relative positioning system was done so in a way that avoided overly complex hardware configurations, to both minimise the cost of development but additionally develop efficient and novel approaches to wayfinding in NLOS environments and point to point control algorithms. Table 4 contains hardware which supports certain dimensions of axial feedback.

To avoid the development of trivial solutions Table 5 describes the reference data required for dimensional positioning and any ambiguity surrounding positioning cases.

<table>
<thead>
<tr>
<th>Number of Radio Transceivers</th>
<th>Dimension of Position Determination from a Direct UWB Ranging Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1D: No, 2D: No, 3D: No</td>
</tr>
<tr>
<td>2</td>
<td>1D: Yes, 2D: No, 3D: Half Plane Ambiguity</td>
</tr>
<tr>
<td>3</td>
<td>1D: Yes, 2D: Yes, 3D: Yes</td>
</tr>
<tr>
<td>4</td>
<td>1D: Yes, 2D: Yes, 3D: Yes</td>
</tr>
</tbody>
</table>
As can be seen from Table 5 3D positioning is trivial given 4 nodes of known position, and 2D trivial given 3 Nodes of known position. A pair of nodes, in the context of this experiment a node-tag pair is the most non-trivial case, more so when there is no absolute reference between either point. From the perspective of either the node or tag, it is the origin and the measurement between the two lays on a spherical surface at a radius equal to the measurement Figure 15. In this case, it would, other than by taking numerous measurements, be impossible to directly determine the position of one point from the other in 3D space.

The following section will briefly address the approach taken in terms of algorithm development in the context of a node-tag pair with sensor fusion integration for 3D relative positioning.

8.1 Non-trivial position finding utilising sensor fusion approach.

8.1.1 Relative position to another agent

Taking the node-tag pair, an individual agent “a” is aware of a scalar distance to another agent “b”. Without taking more measurements this is the most that can be determined. As the quadrotors used in this project are lightweight and not capable of hosting multiple tags for differential measurement systems. It is the ideal proving case for similar industrial applications where many small agents must cooperate in a large swarm. In these cases, constraints are typically placed on payload, energy storage, and cost.

8.1.2 Relative coordinate space

For positioning within an undefined coordinate space, the agent assumes its initialisation point as the origin, and from here utilising a lidar sensor we can determine the agent’s absolute position on a vertical axis. From this point the agent requires a method for \( x, y \) axial position determination. It can be assumed that almost all commercially available drones have onboard accelerometers which can be used for inertial position tracking. However these sensors are subject to substantial drift and are not suitable for defining a relative coordinate space. Recently there has been an increase in use of optical flow sensors for stable position holding during hover in commercially available drones. These sensors use imaging to determine the change in surface gradient and can output reliable \( x, y \) velocity values. Using optical flow sensors, an agent can now, from the origin with reasonable accuracy, track its position through space in \( x, y \) and directly in \( z \).
Amongst the data transmitted between the Crazyflie and the nodes, the Kalman filtered ranging data between each node and radio tag is available. This was located through identifying it within a set of logged variables transmitted to the Crazyflie PC client. By changing the variable structure containing the ranging data from a static to a global variable, the data structure could essentially be copied using the extern function within the developed autonomous firmware. From this point, algorithms could be developed around this ranging value.

8.2 Decentralised Relative Positioning Algorithms
The ranging data described above is a scalar value representing the distance between a node and a radio tag. The value has no vector attributes. It can be visualised as a spherical surface at a radius equal to the ranging data, originating at a central point being the drone as depicted in Figure 15. This is the challenge surrounding the development of an algorithm which can accurately position an agent with reference to this point. There is no certainty that moving in an arbitrary direction will result in a reduced ranging distance.

To simplify this problem and place it more in the context of drone control, we can assume the altitude of the agent is known. This is a reasonable assumption as for general functionality each individual agent is likely to require this data for general flight control. Assuming this we can take the ranging value as any point on the surface of a circle with radius r from the drone, as seen in Figure 16.

At this point without additional data or measurements, the vector to the node is not solvable. This is a limitation of omnidirectional radio transmission, unlike phased array transmissions where the source can be calculated [57]. To now derive the location of the “target”, the drone must move relative from its origin in free space and measure the change in r for a known change in its relative position.

As in most radio location infrastructure, trilateration involves the triangulation of a target by taking three reference measurements whose positions can be defined in a coordinate space. In the context of this problem, we assume the coordinate space is defined by the drone’s initial position being the origin. Feedback from the optical flow sensor is capable of accurately mapping the movement from this point. In this approach, the drone, starting at the origin must take three measurements of the radius at different points in the coordinate space. Then
the point of overlap in the three circles (from the three radial measurements) will give the target position.

This method requires relatively complex computational steps as described in Equation 2. Such operations onboard an embedded microprocessor can be a drain on computational resources and often at no real gain. In this case, the assumption would be the optical flow sensor is subject to minimal drift and measurements of the ranging variable had minimal variance. This is not true and made this approach unsuitable to proximal applications. At larger distances the variations in measurements would be a smaller percent error overall.

\[
\begin{align*}
(x - x_1)^2 + (y - y_1)^2 &= r_{2,1} \\
(x - x_2)^2 + (y - y_2)^2 &= r_{2,2} \\
(x - x_3)^2 + (y - y_3)^2 &= r_{2,3}
\end{align*}
\]

In comparison, a feedback approach demonstrates a more suitable method for position determination. The feedback approach involved a systematic logic-based approach to roughly place the target in one of 4 spatial quadrants. To achieve this the drone takes two measurements then determines the quadrant. Following the quadrant determination, the drone tracks a trajectory of 45° degrees from the axis of the located plane. Once tracked for a specified distance based on the ranging data, a second test is undertaken to check if the target remains in the original plane. Again, the trajectory is set at 45° degrees in the target quadrant for a specified distance until the Setpoint distance is achieved or a further test step is reached. This method is depicted in Figure 17.
Figure 15: 3D Representation of Radio Distance Data

Figure 16: 2D Representation of Radio Distance Data
9.0 Autonomous Program III Development - Decentralised RPS

Still using the structure of the previously implemented autonomous program; the 3D position control program can locate an agent by utilising scalar data. The objective of this program is to demonstrate the ability of a drone using only a scalar value, to ascertain the relative position of another agent. This capability would be integral to the forming of a cohesive swarm of drones without collision. It cannot be assumed that the agent or its target have the same orientation which could potentially allow for minimising variability in the problem. To locate the target agent the following function steps are taken.

1. Firstly, the setpointbuilder function is called. This function contains the setpoint structure address to be written to, an iterative variable and two memory variables Quadrant and radius.
   a. The drone travels in the positive X and Y direction at a fixed velocity and duration, at the beginning and end of each of these movements distance measurements, are recorded and compared to the previous measurements to detect movement away or closer to the target. These are placed in memory.
   b. The function setpointbuilder is responsible for measuring the distance between the agent and target agent when index variables are at key steps in the program.

2. A function QuadrantCondition is called which matches the four potential quadrant cases with the data collected in the previous step.

3. Having found the quadrant in which the target exists, the function Quadrantvelocity is called. This will direct the drone in the correct quadrant heading at a specified velocity and fixed duration.

4. Once this has been executed the program loops and repeats the search until the drone has achieved the distance from target setpoint, i.e. the error is minimised to zero.

Note: To make the program more efficient it is possible to conduct the search in the same $x$, $y$ directions that the target was previously detected in, thus continuing towards the target

This program demonstrates that based on a feedback approach by searching for the approximate position in terms of quadrants, iteratively checking the change in target distance convergence on the target will be achieved. This approach does require the use of the flow deck which is a reasonable assumption as the characteristic information is available through inertial sensors such as accelerometers. However the drift when utilising the optical flow deck is substantially less than that of low cost accelerometers [58].
It is worth noting that from this type of approach, if a single target node has a coordinate assigned within a defined space, a drone using RPS algorithms and sensor fusion elements, could navigate around a space with substantially lower cost than implementing full LPS infrastructure. In this case, the drone would need to navigate to this known point through a method similar to that described previously, and then "initialise" its position as that of the target node coordinate. This would also require some awareness of the orientation of the drone with respect to the room’s coordinate system. This could be achieved by using the onboard compass to align the magnetic bearing and align the space’s defined coordinate space.

Collapsed function view of this program can be seen in Figure 18. Refer to Appendix A Figure 39, Figure 40, Figure 41 & Figure 42 for full source code.
// INCLUDES GO HERE ******************************************
#include "FreeRTOS.h"
#include "lcodeck.h"
#include "commander.h"
#include "system.h"
#include "autotest.h"
#include "task.h"
#include "led.h"
#include "timers.h"
#include "log.h"
#include "param.h"

//
static xTimerHandle timer;

void QuadrantVelocity(int quadrant, float *xVel, float *yVel, float magVel)
{
  // Quadrant conditional check
  void quadrantCondition(bool xState, bool yState, int Fastindex, int *Quadrant)

  extern iptsAlgoOptions_t elgoOptions;
  float radiusdata;
  static bool xmemory;
  static bool ymemory;

  void setpointbuilder(setpoint_t *setpoint, int indexFast, int *Qmemory, float *Gmemory)

  extern int pathindex;
  extern int Coordindex;
  extern bool togglebit;
  extern int StartindexLuke;
  extern float Radiusmemory;
  extern int QuadrantMemory;

  void autoTimerCallback(TimerHandle_t timer)

  void autotestInit();

Figure 18: Relative Positioning Search Algorithm (Autonomous Program III)
10.0 Autonomous Program IV - Decentralised RPS Wand Control
In many vision-based positioning systems, a wand is used to direct a robotic entity through space. To explore further relative positioning systems in a three-node system, a radio wand was developed and tested.

The concept of this was derived from simple trigonometry. Taking the sine rule, if three sides of a triangle are known, the angle between any three of these sides may be calculated.

\[
\cos(A) = \frac{b^2 + c^2 - a^2}{2bc}
\]

The wand was designed such that two nodes were attached at a fixed distance apart on a stick. This known distance is side ‘a’. The radius measured between the drone’s radio tag and either node ‘one’ or node ‘two’ are the remaining sides ‘b’ and ‘c’. Three sides are known and the angle A between the agent and two nodes can be calculated. Ambiguity exists in this case (2 potential positions in a 2D plane, or any position along the circumference of a circle in a 3D space) and the bearing of the two-node wand may differ to that of the agent (i.e. a Z angle and vector approach is not possible). As this is the case a feedback approach is taken.

*Figure 19: Trigonometric Positioning Approach*
If the angle $A$ is greater than zero, the drone is not in line with both nodes on the wand and the drone must start to orbit one of the two nodes at a radius equal to that of the tag-node distance. Assuming the drone can orbit fast enough relative to the movement of the wand, it will reach a point in the orbit where angle $A$ equals zero. To compensate for variable distances to the wand, the orbital $x$ and $y$ velocities are functions of sine and cosine respectively, and have a dynamic magnitude equal to that of the radius as seen in Equation 4 and Equation 5.

*Equation 4: $x$ Orbital Velocity*

$$x = r \cos A$$

*Equation 5: $y$ Orbital Velocity*

$$y = r \sin A$$

On testing this algorithm, it was found the variation in radio positioning reading made determining the angle between nodes difficult. As suggested with a trilateration-based approach, variability in data is not suitable for more direct positioning calculations. To combat instability in flight, the angle calculation was only conducted every five loops or 500ms, whilst the sinusoidal velocities were continually executed at 100ms, reducing instability to a degree.

This case proves that the overall concept is valid. However are a more refined approach to implementing the functionality should be reviewed and implemented. This case also makes the point that by adding one additional point of data, unless the bearing of both agents is known, it is still not possible to directly calculate a trajectory to the target. Although a relative bearing may be calculated this is meaningless as $x$, $y$ velocities to this angle could be in any direction.

Collapsed function view of this program can be seen in Figure 20. Refer to Appendix A Figure 43, Figure 44 & Figure 45 for full source code.
static xTimerHandle timer;
extern lpsAlgoOptions_t algoOptions;
float radiusdata1;
float radiusdata2;
static float Adeg;
static float b;
static float c;

void setpointbuilder(setpoint_t *setpoint, int indexFast, int indexSlow){}

extern int fpathindex;
extern int Coordindex;
extern bool togglebit;
extern int Startindexluke;

void autoTimerCallback(TimerHandle_t timer){}

void autotestInit(){
timer = xTimerCreate( "LukeSTimer", pdMS_TO_TICKS(100), pdTRUE, NULL, autoTimerCallback);
xTimerStart(timer, 100);
}
11.0 Experimental Configuration

11.1 Test Environment

An important consideration regarding the experimental data collected throughout the project was the environmental effects in terms of potential radio interference.

The Murdoch University Pilot Plant is an industrial environment, with large metallic surfaces, from roller shutter doors and process equipment such as stainless-steel storage tanks. In addition to this, there are a considerable number or radio interference sources such as variable speed drives, AC motors, a radar level transmitter and an industrial wireless modem.

The rationale behind collecting data in such an electromagnetically saturated environment was to practically assess the performance of the radio positioning technology in an environment most like a potential industrial application. In this respect, the measures of performance are likely a worst-case scenario in an industrial environment with a reasonably high concentration of radio signal sources and reflective surfaces.

The Murdoch University Pilot Plant can be seen in Figure 21.
11.2 Statistical Measures and Relevance in Performance Assessment

To identify potential differences between the two algorithms, the following statistical measure were used to analyse performance and conduct a comparison. Taking these statistical measures, we can analyse the radiolocation in terms of its “stability” and “accuracy”.

Primary Statistical Measures used:

- Mean
- Standard Deviation
- Variance
- Range

The stability of the radiolocation data refers to the variance and standard deviations of readings. If the standard deviation or variance is large this means that the feedback data is noisy, and this may have a negative effect on position control algorithms in terms of stability.

The accuracy of the radiolocation system refers to the mean of the data. By taking a measurement in the reference space and comparing that to the measurement from the LPS, a percentage offset between the true measured position relative to the LPS infrastructure can be obtained. This will indicate how accurate the positioning system is relative to its defined coordinate system. In addition, normal distribution plots and histograms were used to visualise the data for further analysis.

11.3 Methods of data collection and validation of the approach

To obtain a clear understanding of the positioning properties of the two algorithmic implementations (TWR and TDOA) an unbiased, repeatable collection methodology was developed.

11.3.1 LPS Framework

Key to a repeatable collection method, was the construction of the LPS framework in which all data collection activities would be undertaken. It was important this structure be geometrically accurate as any errors in its geometry would translate into radio positioning errors. Aluminium extrusion tubes were used to construct a cube (3.10m x 3.10m x 3.10m) where the radio positioning nodes could be mounted (refer to Figure 28). As per the
recommended configuration from Bitcraze, the nodes were placed in an inverted triangle geometry with all nodes 0.15m away from floors and walls [45]. As seen in Figure 23 and Figure 24, the nodes were addressed 0-5 in with corresponding coordinate system as seen in the above mentioned figures. The vertical offset can be seen in Figure 22.

Figure 22: LPS Node Height Offset

Figure 23: LPS Node Configuration (Nodes 3, 5 & 2 are at the front)
11.3.2 Apparatus and Data collection

The points were selected within the LPS framework to obtain a range of measurements which represent the entire space accurately. For this reason, 9 measurements were taken over three vertical and horizontal planes. To optimise the symmetry between points, the twenty-seven measurement points were selected, all at a radius equal to 0.516m apart (see Appendix F for field notes on configuration). Measurement points in the $x$, $y$ and $y$, $z$ plane can be seen in Figure 25 and Figure 26 respectively.

![Figure 24: Inverted Triangle Geometry $x$ $y$ Plane (Coordinates in Metres)](image-url)
11.3.3 Collecting Static Data Points

To capture the statistical accuracy of the LPS, firstly the nine $x$, $y$ coordinates of the twenty-seven measurement points were marked out. Using these markings as a centre point, circles with equal radius to a PVC pipe were drawn. Three lengths of PVC pipe were cut at the corresponding testing heights. These pipes were then placed on the testing points marked out, allowing for the drone to be placed accurately in the $x$, $y$ and $z$ axis (Figure 27). Here the drone could be left to collect data. The PVC was selected due to its non-conductive properties reducing potential radio interference.
Data collection for each static measurement point lasted a minimum of one minute at a 10Hz collection rate. Thus 600 data points were collected at each measurement point, 5400 per plane, with 16200 data points collected in total per algorithm. This static data was collected as during operation the flight dynamics of the drone would have interfered with the true accuracy of the LPS.

To collect data useful in providing an indication of the dynamic accuracy of the LPS, autonomous flight routines were used to move the drone through the LPS framework. This followed a geometric flight path in the central plane for consistent and repeatable results.

11.4 Methods of Data Processing
A MATLAB script was used to generate plots and compute statistical measures from a standardised Excel spreadsheet. This script can be re-run with any data collected in the future with minimal configuration. Additionally, Excel spreadsheets were used to compute logical comparisons between numerical results for performance analysis. This was important in removing some potential human error as there were 108 individual indicators of performance per plane. Human error elimination was crucial. The MATLAB Script can be found in Appendix B.
Figure 28: Aluminium Extrusion Construction for Mounting LPS Nodes
12.0 Results

Results are primarily discussed with reference to the average positioning offset or error, and to the comparative performance analysis which tests for performance across all statistical measures between pairs of $x$, $y$, and $z$ data at a specific location. Reference is made to histograms which are useful in visualising the complete data set for the algorithms.

12.1 Performance analysis of TWR and TDOA

Results analysed per plane (9 points at the ground middle or top plane) show that overall, the best performing algorithm for positioning was TDOA. For each plane comparisons between statistical measures for the 9 points were analysed. From this analysis, TDOA succeeded at a rate of 79.63%, 58.33%, and 85.19%, for the ground, middle and top plane respectively (refer to Table 6).

Results indicate a more balanced performance between TDOA and TWR in the middle plane (58.33% v 41.67%). This is likely caused by the equidistance between the measurement points and nodes which are less likely to favour either algorithm.

Points which are most equidistant such as the point (1.55 m, 1.55 m, 1.55 m) do show a moderately increased performance for both algorithms in terms of statistical measures. Refer to tables in Appendix C.

12.2 Positioning Offset from Real World Coordinate System

Results show that on average (combining offsets across twenty-seven points) TDOA outperformed TWR in terms of combined planar offset. TDOA shows an average offset of 8.77% compared to TWR with an average of 10.11% (refer to Table 7).

When looking at the planar data independently it can be seen TDOA outperforms TWR in both the ground and middle plane, with a success rate of 55.55% and 70.37 percent respectively. TWR outperforms TDOA in terms of combined $x$, $y$, $z$ offset for both the top and middle plane with an average offset of 7.82% and 7.29% respectively.

The ground plane showed the most contrast in terms of performance between the two algorithms. TDOA achieved an offset of 10.27% compared to TWR’s offset of 15.21%. This could be explained due to proximity to the concrete floor of the pilot plant reflecting radio
signals. As TWR uses a round-trip methodology, errors caused by reflections are compounded in the calculation unlike TDOA.

Although another explanation for this larger differential in offset may be due to experimental error causing outliers in the data, the other statistical measures do not appear dissimilar to the other datasets.

12.3 Normal Distribution Plots and Histograms

When analysing the data sets for both TWR and TDOA across the three planes, it is clear to see the average error is an obscured image of accuracy. More realistically, most of the error values are centred around 0% extending to ±10%. This can be seen in Figure 29, Figure 30, and Figure 31.

From observing the results through statistical analysis, it can be said that TDOA provides the best overall performance in any installation point as there are indications that TWR is affected by proximity to solid surfaces. Results confirm TWR caused more positional offset from the real-world coordinate system than TDOA.

Additional histograms can be generated by executing the MATLAB script in Appendix B. The histograms generated are based on $x$, $y$, $z$ characteristics for each measurement point. These would be useful in analysing the positioning characteristics of each axial measurement at each measurement point in greater detail should it be necessary.
Figure 29: Histogram and Normal Distribution Plot for Positioning Error (Top Plane)

Figure 30: Histogram and Normal Distribution Plot for Positioning Error (Middle Plane)

Figure 31: Histogram and Normal Distribution Plot for Positioning Error (Ground Plane)
### Overall Best Performing Algorithm

<table>
<thead>
<tr>
<th>Plane</th>
<th>Successful Algorithm</th>
<th>Rate of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>TDOA</td>
<td>79.63%</td>
</tr>
<tr>
<td>Middle</td>
<td>TDOA</td>
<td>58.33%</td>
</tr>
<tr>
<td>Top</td>
<td>TDOA</td>
<td>85.19%</td>
</tr>
</tbody>
</table>

### Table 7: Average Offset (%) by Plane and Algorithm

**% Average Offset (x, y, z Combined)**

<table>
<thead>
<tr>
<th>Top Plane</th>
<th>Middle Plane</th>
<th>Ground Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWR</td>
<td>TDOA</td>
<td>TWR</td>
</tr>
<tr>
<td>7.825983</td>
<td>8.335974157</td>
<td>7.299160317</td>
</tr>
</tbody>
</table>

### Table 8: Relative Accuracy (offset %) by Algorithm

**Least Offset**

<table>
<thead>
<tr>
<th>Plane</th>
<th>Successful Algorithm</th>
<th>Rate of Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>TDOA</td>
<td>55.56</td>
</tr>
<tr>
<td>Middle</td>
<td>TDOA</td>
<td>70.37</td>
</tr>
<tr>
<td>Top</td>
<td>TWR</td>
<td>59.26</td>
</tr>
</tbody>
</table>
13.0 Results: Dynamic Flight Path Data Analysis

The autonomous flightpath algorithm allowed for the collection of dynamic radio-positioning data. This data provides insight into the dynamic accuracy of the combined quad-rotor and positioning system dynamics. Theoretically in a static configuration onboard filters and or estimators can derive the position within a very small margin of error.

The analysis of this data is done as a practical assessment for implementation in real-world environments and its suitability for industrial applications. Some analytical measures are used as a guide to compare expected performance and the experimental results.

13.1 Quantitative Analysis

The most suitable method for assessing the flight path data is divided into both statistical measures for the Z axis position and linear regression for the flightpath tracking in the \( x, y \) plane.

Each algorithm was assessed by processing five orbitals of the hexagonal flight path (refer to Figure 32 and Figure 33). This should be a large enough data set to average out any deviations in flight dynamics due to environmental disturbances (air flow) and transient radio interference.

By taking one side of the hexagonal flight path, as described in section 7.0 Autonomous Program II Development, and computing a line of best fit the experimental gradient can be found. This can then be compared to the gradient of the flight path to measure the percentage error. Selecting only one side of the hexagon is a reasonable method for analysis as the \( x, y \) plots presented a consistent symmetry in terms of variation in experimental data. No one side appeared to perform significantly better than another.

Results show that TDOA outperforms TWR in tracking the flightpath, with an error of 4.2% compared to 8.49% (see Table 9 and Table 10). This supports the findings of static data analysis. Using the same statistical measures as in the static data analysis the Z axis it was found TDOA outperformed TWR in terms of the average positioning error (see Table 11 and Table 12).
Figure 32: Hexagonal Flight Path
Figure 33: TDOA & TWR x, y Scatter Plot Hexagonal Flight Path (blue) with programmed flightpath (red)
### Table 9: TDOA Flight Path Gradient Analysis

<table>
<thead>
<tr>
<th>Orbit</th>
<th>Line Equation</th>
<th>Gradient offset/error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orbit 1</td>
<td>$y = 0.5171x - 0.2927$</td>
<td>3.42</td>
</tr>
<tr>
<td>Orbit 2</td>
<td>$y = 0.4931x - 0.2319$</td>
<td>-1.38</td>
</tr>
<tr>
<td>Orbit 3</td>
<td>$y = 0.5222x - 0.2935$</td>
<td>4.44</td>
</tr>
<tr>
<td>Orbit 4</td>
<td>$y = 0.5577x - 0.3634$</td>
<td>11.54</td>
</tr>
<tr>
<td>Orbit 5</td>
<td>$y = 0.4999x - 0.2428$</td>
<td>-0.02</td>
</tr>
<tr>
<td></td>
<td><strong>Average (Abs)</strong></td>
<td><strong>4.16</strong></td>
</tr>
</tbody>
</table>

### Table 10: TWR Flight Path Gradient Analysis

<table>
<thead>
<tr>
<th>Orbit</th>
<th>Line Equation</th>
<th>Gradient Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orbit 1</td>
<td>$y = 0.4479x - 0.1051$</td>
<td>10.42</td>
</tr>
<tr>
<td>Orbit 2</td>
<td>$y = 0.4414x - 0.093$</td>
<td>11.72</td>
</tr>
<tr>
<td>Orbit 3</td>
<td>$y = 0.5805x - 0.3898$</td>
<td>-16.10</td>
</tr>
<tr>
<td>Orbit 4</td>
<td>$y = 0.5032x - 0.2307$</td>
<td>-0.64</td>
</tr>
<tr>
<td>Orbit 5</td>
<td>$y = 0.5178x - 0.2556$</td>
<td>-3.56</td>
</tr>
<tr>
<td></td>
<td><strong>Average (Absolute)</strong></td>
<td><strong>8.49</strong></td>
</tr>
</tbody>
</table>

### Table 11: Z Axis Performance Analysis (TDOA at 1m, TWR at 1.15m)

<table>
<thead>
<tr>
<th>Statistic</th>
<th>TDOA</th>
<th>TWR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.0011</td>
<td>1.1377</td>
</tr>
<tr>
<td>Std</td>
<td>0.0246</td>
<td>0.0564</td>
</tr>
<tr>
<td>Variance</td>
<td>0.0006</td>
<td>0.0032</td>
</tr>
<tr>
<td>Range</td>
<td>0.2519</td>
<td>0.3849</td>
</tr>
<tr>
<td>Minimum</td>
<td>0.8747</td>
<td>0.8688</td>
</tr>
<tr>
<td>Maximum</td>
<td>1.1266</td>
<td>1.2537</td>
</tr>
</tbody>
</table>

### Table 12: Z Axis Offset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDOA</td>
<td>-0.11</td>
</tr>
<tr>
<td>TWR</td>
<td>1.23</td>
</tr>
</tbody>
</table>
14.0 Results: Industrial and Cooperative Robotic Applications

In this section, the data is analysed in a practical and qualitative manner with comment made on the feasibility of implementing an LPS or RPS within an industrial or robotics setting.

The results as presented above indicate the positioning accuracy of the LPS to realistically be between ±10%. This is about the 3.10x3.10x3.10m LPS geometry.

It can be assumed error will be most significant within smaller test spaces, as the timing radio signals over longer distances will not reach or exceed technical limitations of the hardware and onboard algorithms. Thus, taking the radio interference sources and relatively small-scale test environment, it is fair to assume the range of accuracy as presented in this report be a likely worst-case scenario in an industrial application.

The radiolocation technology performing as seen in the experiments would likely make little difference practically for most applications. This is due to the likely margin of error that would be allowed in most applications. For example, the likelihood an autonomous aerial vehicle will be required to position itself less than ±18-30cm in a 3m cubic space is unlikely as flight dynamics and variable environmental conditions must be considered. Not to mention the vehicle size itself compared to the error radius, as most commercial drones likely exceed a 30cm vehicle diameter.

The results of this report indicate that the practical capabilities of an LPS are within a suitable margin of error for integration into industrial applications.

The autonomous capabilities developed with the use of an RPS, demonstrate characteristics required by a swarm control system. The sensor fusion elements with the aid of radio positioning data allowed the drone to effectively navigate through a space to locate a target node. It has been demonstrated in a point to point undefined coordinate space feedback navigation is possible. This can be expanded on and integrated into a multiagent cooperative robotic application where individual agents work together without a centralised control system to perform as a cohesive unit.
15.0 Future Work

15.1 Accuracy of UWB Positioning Coupled with Flight Dynamics of a Larger Drone
This project analysed the dynamic accuracy of UWB coupled with the flight dynamics of a lightweight low power quadrotor. It is recommended a study be undertaken utilising a larger more powerful quadrotor which may provide a better indication as to the true dynamic accuracy of the UWB infrastructure. As an extension it is possible to directly measure the true dynamic accuracy using robotic arms which have absolute positioning encoders. This is also a potential application of the UWB positioning technology to reduce the cost of such robotic systems.

15.2 Implementation of a Multi-Agent Swarm Utilising 3D Feedback Approach
In the final stage of this project a 3D relative positioning algorithm was developed, however the problem was simplified in developing program components for a single horizontal plane. It is simple to extend this by including vertical measurement steps in the algorithm to determine where a target node is in an octant rather than a quadrant. This again can be extended defining the objective function of the drone to hold a distance to more than one target. Note all drones in this application would be required to have the same navigation program, such that the drones converge towards each other.

15.3 Overlay 3D Maps onto LPS - Dynamic Mapping of Obstacles & Flight Path Optimisation
Using static UWB infrastructure it is possible to program in environmental hazards or obstacles that correlate to defined coordinates. This could be applied to mapping an industrial warehouse such that a drone may navigate to a target destination and avoid obstacles during flight. This could be done dynamically or by preplanning optimal flight paths to the destination. By using ranging sensors such as LIDAR, new obstacles could be mapped dynamically on-route and saved for future flights.

15.4 Geometric Swarm Formations Using a Decentralised Anonymous Approach.
Development of complex decentralised algorithms to form specified geometries is an area of interest. It demonstrates non-trivial cooperative robotic capabilities which have an array of application areas. Without centralised control, decisions must be made by individual robots without knowledge of the intentions of other cooperating agents. To remove abstraction of this concept the task of forming a 3D formation with drones represents a decentralised
control problem. Taking a cube as the desired formation, a drone must occupy one of four vertices. Each vertex is a set distance away from the other 3. This is the beginning of the objective function for control. It would be recommended a random number be broadcast from each drone with all numbers compared on each individual drone. The drone which has the largest randomly generated number defines the geometric formation at a predefined vertical plane. The second, third and finally fourth largest number determines a duration a drone could wait before joining the geometric configuration. By essentially eliminating occupied positions, ambiguity in terms of selecting a vertex is reduced and by following a geometric ruleset the structure could be formed.
16.0 Conclusion

A commercially available UWB positioning system and quadrotor have been used to verify the performance of a commercially available UWB radiolocation system. Firstly, performance was assessed in an as-intended centralised configuration through static data collection points. Following this a dynamic accuracy assessment utilising an autonomous flightpath routine was undertaken.

Experiments have been conducted in The Murdoch University Pilot Plant, an industrial environment with a reasonably high concentration of radio signal sources and reflective surfaces. This has led to performance results reflecting a similar margin of error as would be expected in a real-world industrial application.

Aluminium extrusion tubes were used to construct a test framework where the radio positioning nodes could be mounted as per the recommended configuration from Bitcraze. This was key to a repeatable and consistent data collection methodology. In total 16200 data points have been collected across twenty-seven measurement points which best represent the characteristics of the LPS framework. MATLAB and Excel have been used to process and analyse the data through the generation of statistical metrics, histograms, distribution plots, and logic-based performance spreadsheets.

When analysing the static data sets for both TWR and TDOA across all points, it is clear TDOA outperforms TWR in terms of positional offset and statistical measures. On average both TWR and TDOA have error values that are centred around 0% extending to ±10%.

When analysing the dynamic data sets for both TWR and TDOA across all points, it is clear TDOA outperforms TWR in terms of both gradient error and statistical measures.

Through the testing of autonomous programs, it has been found that using scalar radiolocation values in combination with a sensor fusion approach, it is possible to navigate through use of relative positioning algorithms. Applications such as swarm control algorithms of drones and cooperative robotic tasks where relative position is required was used as a guide into the functional requirements for algorithm development and practical investigation.
Radiolocation is unique in these applications due to non-line-of-sight capabilities and omnidirectional transmission characteristics, which would be advantageous if applied to cooperative robotic applications such as swarm formations.

The radiolocation technology performing with an error percentage as seen in the experiments would likely make little difference practically for most applications. This is due to general expectations of most practical systems, the error in comparison to the robotic vehicle size itself and the likelihood results presented are a worst-case scenario in an environment with numerous radio interference sources.

Further studies could be undertaken to improve the understanding of the dynamic accuracy without the coupled effects of drone flight dynamics. Extensions of various autonomous algorithms could be further developed to tackle more complex positioning problems, including flight path optimisation, obstacle avoidance and multiagent geometric formations.
Bibliography


Appendix A

The figures within Appendix A are the full code extracts of the autonomous programs developed during this project.

```c
// INCLUDES GO HERE ******************************************
#include "FreeRTOS.h"
#include "commander.h"
#include "system.h"
#include "autotest.h"
#include "task.h"
#include "led.h"

// VARIABLES AND CONSTANTS *******************************
int fpathIndex;

// TASK DEFINITION
void autotestTask(){
    systemWaitStart();
    fpathIndex=0;
    while(1)
    {
        setpoint_t autosequencepoint;
        ledSet(CHG_LED, 1);
        vTaskDelay(pdMS_TO_TICKS(250));
        ledSet(CHG_LED, 0);
        vTaskDelay(pdMS_TO_TICKS(250));

        setpointBuilder(&autosequencepoint, &fpathIndex);
        commanderSetSetpoint(&autosequencepoint, AUTOTEST_TASK_PRI);

        fpathIndex++;
        if (fpathIndex>=21)
        {
            fpathIndex=21;
        }
    }
}

void automodesetup(setpoint_t *setpoint){
}
```

Figure 34: Program I (part a)
void setpointbuilder(setpoint_t *setpoint, int *index){
    float fpatharray[22] =
    {0,0,0,0,0,0,0,0,0,0,0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0};

    setpoint_t tempsetpoint;
    tempsetpoint.mode.x = modeVelocity;
    tempsetpoint.velocity.x = 0;

    tempsetpoint.mode.y = modeVelocity;
    tempsetpoint.velocity.y = 0;

    tempsetpoint.mode.z = modeAbs;
    //tempsetpoint.position.z = 0.20;
    tempsetpoint.position.z=fpatharray[*index];

    tempsetpoint.mode.roll = modeAbs;
    tempsetpoint.attitude.roll = 0;

    tempsetpoint.mode.pitch = modeAbs;
    tempsetpoint.attitude.pitch = 0;

    tempsetpoint.mode.yaw = modeVelocity;
    tempsetpoint.attitudeRote.yaw = 0;

    *setpoint=tempsetpoint;
    ledSet(LINK_LED, 1);
    vTaskDelay(pdMS_TO_TICKS(50));
    ledSet(LINK_LED, 0);
    vTaskDelay(pdMS_TO_TICKS(50));
}

void autotestInit(){
    xTaskCreate(autotestTask, AUTOTEST_TASK_NAME, AUTOTEST_TASK_STACKSIZE, NULL, AUTOTEST_TASK_PRI, NULL);
}

Figure 35: Program I (part b)
Figure 36: Program II (Part a)
setpoint_t tempsetpoint;

    // Flight Path Array
    float XCoord[13] =
    {1.55,1.55,1.55,1.55,1.55,2.50,2.50,1.50,0.50,0.50,1.55,2.50};
    float YCoord[13] =
    {1.55,1.55,1.55,1.55,1.55,2.00,1.00,0.50,1.00,2.00,2.50,2.00};
    float ZCoord[13] =
    {0.00,0.15,0.50,0.75,1.00,1.15,1.15,1.15,1.15,1.15,1.15,1.15};

    // Dynamic Set point
    float SPx = XCoord[indexSlow];
    float SPy = YCoord[indexSlow];
    float SPz = ZCoord[indexSlow];

    float SPx_prev;
    float SPy_prev;
    float SPz_prev;

    float TjxSP;
    float TjySP;
    float TjzSP;

    if (indexSlow==0){
        SPx_prev = 1.55;
        SPy_prev = 1.55;
        SPz_prev = 0;
    } else if (indexSlow>0){
        SPx_prev = XCoord[(indexSlow-1)];
        SPy_prev = YCoord[(indexSlow-1)];
        SPz_prev = ZCoord[(indexSlow-1)];
    }

    coordinateCondition(SPx, SPx_prev, indexFast, &TjxSP);
    coordinateCondition(SPy, SPy_prev, indexFast, &TjySP);
    coordinateCondition(SPz, SPz_prev, indexFast, &TjzSP);

    tempsetpoint.mode.x = modeAbs;
    tempsetpoint.mode.y = modeAbs;
    tempsetpoint.mode.z = modeAbs;

    tempsetpoint.mode.yaw = modeVelocity;

    tempsetpoint.position.x = TjxSP;
    tempsetpoint.position.y = TjySP;
    tempsetpoint.position.z = TjzSP;

Figure 37: Program II (Part b)
tempsetpoint.attitudeRate.yaw = 0;

//Write tempsetpoint to setpoint function call variable structure
*setpoint=tempsetpoint;
}

extern int fpindex;
extern int Coordindex;
extern bool togglebit;

void autoTimerCallback(TimerHandle_t timer)
{
    setpoint_t autosetpoint;
    setpointBuilder(&autosetpoint, fpindex, Coordindex);
    commanderSetSetpoint(&autosetpoint, AUTOTEST_TASK_PRI);
    fpindex++;
    if (fpindex>100)
    {
        fpindex=1;
        Coordindex++;
        togglebit = !togglebit;
        if(Coordindex>12)
        {
            Coordindex=7;
        }
    }

    ledSet(LINK_LED, togglebit);
}

void autotestInit()
{
    timer = xTimerCreate( "LukesTimer", pdMS_TO_TICKS(100), pdTRUE, NULL,
    autoTimerCallback);
    xTimerStart(timer, 100);
}

Figure 38: Program II (Part c)
// INCLUDES GO HERE ******************************

#include "FreeRTOS.h"
#include "locodeck.h"
#include "lpsTdma.h"
#include "lpsTdoaTag.h"
#include "lpsTwrTag.h"
#include "commander.h"
#include "system.h"
#include "autotest.h"
#include "task.h"
#include "led.h"
#include "crtp_localization_service.h"
#include "vl53l0x.h"
#include "position_estimator.h"
#include "timers.h"
#include "log.h"
#include "param.h"

//_____________________________________________

static xTimerHandle timer;

void QuadrantVelocity(int Q, float *xVel, float *yVel, float magVel){
  float xV;
  float yV;

  if (Q == 1){
    xV=magVel;
    yV=-magVel;}
  if (Q == 2){
    xV=magVel;
    yV=magVel;}
  if (Q == 3){
    xV=-magVel;
    yV=magVel;}
  if (Q == 4){
    xV=-magVel;
    yV=-magVel;}

  *xVel=xV;
  *yVel=yV;

Figure 39: Program III (Part a)
void quadrantCondition(bool xState, bool yState, int Fastindex, int *Quadrant) {
    int Q;
    if ((xState == yState) && (xState == 1)) {
        Q=2;
    }
    if ((xState == yState) && (xState == 0)) {
        Q=4;
    }
    if (xState < yState) {
        Q=1;
    }
    if (xState > yState) {
        Q=3;
    }
    *Quadrant = Q;
}

extern lpsAlgoOptions_t algoOptions;
float radiusdata;
static bool xmemory;
static bool ymemory;

void setpointIntbuilder(setpoint_t *setpoint, int indexFast, int *Qmemory, float *Gmemory) {
    setpoint_t tempsetpoint;
    // P2P Data
    radiusdata = algoOptions.distance[2];
    float xVelocity;
    float yVelocity;
    float V;
    int Qmem;
    float RadiusSP = 1.00;
    float error;
    error = radiusdata - RadiusSP;
    if (fpathindex==40) || (fpathindex==80) || (fpathindex==80)|| (fpathindex==110)) {
        *Gmemory = algoOptions.distance[2];
    }

Figure 40: Program III (Part b)
xVelocity=0;
yVelocity=0;

if ((indexFast>=40) && (indexFast<60)){
    xVelocity=0.20;
    xmemory = (radiusdata < *Qmemory);
}
if ((indexFast>=60) && (indexFast<80)){
    yVelocity=0.20;
    ymemory = (radiusdata < *Qmemory);
}
if (indexFast>=80){
    xVelocity=0;
    yVelocity=0;
    quadrantCondition(xmemory, ymemory, indexFast, &Qmem);
    *Qmemory=Qmem;
}
if (((indexFast>80) && (indexFast<100)){
    V=0.10;
    QuadrantVelocity(*Qmemory, &xVelocity, &yVelocity, V);
}
if (indexFast>100){
    xVelocity=0;
    yVelocity=0;
}

    tempsetpoint.mode.x = modeVelocity;
    tempsetpoint.mode.y = modeVelocity;
    tempsetpoint.mode.z = modeAbs;
    tempsetpoint.mode.yaw=modeVelocity;
    tempsetpoint.velocity.x = xVelocity;
    tempsetpoint.velocity.y = yVelocity;
    tempsetpoint.position.z = 0.60;
    tempsetpoint.attitudeRate.yaw = 0;

    //Write tempsetpoint to setpoint function call variable structure
    *setpoint=tempsetpoint;

extern int pathindex;
extern int Coordindex;
extern bool togglebit;

Figure 41: Program III (Part c)
extern int Startindexluke;
extern float RADIUSmemory;
extern int QuadrantMemory;

void autoTimerCallback(TimerHandle_t timer) {
    if (Startindexluke<=100){
        // DO NOTHING LET POS INITIALSEFor 10 seconds
        Startindexluke++;
    }
    if(Startindexluke>100)
    {
        setpoint_t autosetpoint;
        setpointbuilder(&autosetpoint, fpathindex, &QuadrantMemory, &Radiusmemory);
        commanderSetSetpoint(&autosetpoint, AUTOTEST_TASK_PRI);
        fpathindex++;
        if (fpathindex>10){
            // fpathindex=0;
        }
    }
}

void autotestInit(){
timer = xTimerCreate( "LukesTimer", pdMS_TO_TICKS(100), pdTRUE, NULL, autoTimerCallback);
xTimerStart(timer, 100);
}
// INCLUDES GO HERE **********************************

#include "FreeRTOS.h"
#include "locodeck.h"
#include "lpsTdma.h"
#include "lpsTdoaTag.h"
#include "lpsTwrTag.h"
#include "commander.h"
#include "system.h"
#include "autotest.h"
#include "task.h"
#include "led.h"

#include "crtplocalization_service.h"
#include "vl53l0x.h"
#include "position_estimator.h"

#include "timers.h"
#include "log.h"
#include "param.h"
#include "math.h"

// static xTimerHandle timer;

extern lpsAlgoOptions_t algoOptions;
float radiusdata1;
float radiusdata2;
static float Adeg;
static float b;
static float c;

void setpointbuilder(setpoint_t *setpoint, int indexFast, int indexSlow){
    setpoint_t tempsetpoint;

    // P2P Data
    radiusdata1 = algoOptions.distance[1];
    radiusdata2 = algoOptions.distance[2];

    float xVelocity;
    float yVelocity;

    Figure 43: Program IV (Part a)
float a=0.50;
float a2=a*a;

float pi = 3.141592;
float w=2*pi*0.1;

int T=indexFast;
float A;

if (Coordindex==0){
    b=radiusdata1;
    float b2=b*b;

    c=radiusdata2;
    float c2=c*c;

    A = acos((b2+c2-a2)/(2*b*c));
    Adeg = A*(180/pi);
}

    if (Adeg>10){
        xVelocity=b*cos(w*T);
        yVelocity=b*sin(w*T);
    }

    else if (Adeg<5){
        xVelocity=0;
        yVelocity=0;
    }

tempsetpoint.mode.x = modeVelocity;
tempsetpoint.mode.y = modeVelocity;
tempsetpoint.mode.z = modeAbs;

tempsetpoint.mode.yaw=modeVelocity;

tempsetpoint.velocity.x = xVelocity;
tempsetpoint.velocity.y = yVelocity;
tempsetpoint.position.z = 0.60;
tempsetpoint.attitudeRate.yaw = 0;

//Write tempsetpoint to setpoint function call variable structure
*setpoint=tempsetpoint;

Figure 44: Program IV (Part b)
extern int fpathindex;
extern int Coordinindex;
extern bool togglebit;
extern int Startindexlukex;

void autoTimerCallback(TimerHandle_t timer)
{
    if (Startindexlukex<=100)
    {
        //DO NOTHING LET POS INITIALISEfor 10 seconds
        Startindexlukex++;
    }
    if(Startindexlukex>100)
    {
        setpoint_t autosetpoint;
        setpointbuilder(&autosetpoint, fpathindex, Coordinindex);
        commanderSetSetpoint(&autosetpoint, AUTOTEST_TASK_PRI);
        fpathindex++;
        Coordinindex++;
        if (Coordinindex>100){
            Coordinindex=0;
        }
    }
}

void autotestInit(){
timer = xTimerCreate( "LukesTimer", pdMS_TO_TICKS(100), pdTRUE, NULL, autoTimerCallback);
xTimerStart(timer, 100);
}

Figure 45: Program IV (Part c)
Appendix B

The figures within Appendix B show the MATLAB code used to process the positioning data. The program extracts data from excel spreadsheets and generates statistical measures in table form.

```matlab
% Thesis Data Process MATLAB File

% 1. | Reads excel spreadsheets for a single plane
% 2. | Generates Histograms of X,Y,Z data for a 1:9 coordinate sets
% 3. | Generates statistical measures (values table)

clear all

gX=zeros(595, 9);
gY=zeros(595, 9);
gZ=zeros(595, 9);

dataX=zeros(6, 9);
dataY=zeros(6, 9);
dataZ=zeros(6, 9);

% substitute the correct file name or execution
% 'Top Plane Trimmed TWR.xlsx'
% 'Middle Plane Trimmed TWR.xlsx'
% 'Ground Plane Trimmed TWR.xlsx'

for i=1:1:9
    gX(:,i)=xlsread('Top Plane Trimmed TWR.xlsx',i,'A1:A600');
    gY(:,i)=xlsread('Top Plane Trimmed TWR.xlsx',i,'B1:B600');
    gZ(:,i)=xlsread('Top Plane Trimmed TWR.xlsx',i,'C1:C600');
end

for i=1:1:9
    figure(i)
    hold on
    subplot(3,1,1);
    histfit(gX(:,i),10);
    xlabel('X Axis (TWR Top Plane)')
    ylabel('Frequency')

    subplot(3,1,2);
    histfit(gY(:,i),10);
    xlabel('Y Axis (TWR Top Plane)')
    ylabel('Frequency')

    subplot(3,1,3);
    histfit(gZ(:,i),10);
    xlabel('Z Axis (TWR Top Plane)')
    ylabel('Frequency')
    hold off

dataX(1,i)=mean(gX(:,i));
dataX(2,i)=std(gX(:,i));
```

84
Figure 46: MATLAB Data Processing Code (Part a)

dataX(:,1)=var(gX(:,i));
dataX(:,2)=range(gX(:,i));
dataX(:,3)=nanmin(gX(:,i));
dataX(:,4)=narmas(gX(:,i));
dataY(:,1)=mean(gY(:,i));
dataY(:,2)=std(gY(:,i));
dataY(:,3)=var(gY(:,i));
dataY(:,4)=range(gY(:,i));
dataY(:,5)=nanmin(gY(:,i));
dataY(:,6)=narmas(gY(:,i));
dataZ(:,1)=mean(gZ(:,i));
dataZ(:,2)=std(gZ(:,i));
dataZ(:,3)=var(gZ(:,i));
dataZ(:,4)=range(gZ(:,i));
dataZ(:,5)=nanmin(gZ(:,i));
dataZ(:,6)=narmas(gZ(:,i));
end

Figure 47: MATLAB Data Processing Code (Part b)
Appendix C

The Tables within Appendix C are the Logic based test results used to compare the \( x \), \( y \) and \( z \) values across the twenty-seven measuring points for both the TDOA and TWR Algorithm. Each table contains the \( x \), \( y \), \( z \) algorithm data pairs for one of twenty-seven data collection points.

The logic tests compared whether the Mean, Standard Deviation, Variance, and Range was better for either TWR or TDOA with the successor generated as the result. These Successors as either “TDOA” or “TWR” were counted and used to calculate a percentage of tests succeeded by the respective algorithms.

### Table 13: Logic Test Result Table Set (Ground Plane)

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5434</td>
<td>0.5417</td>
<td>TDOA</td>
<td>0.5723</td>
<td>0.4609</td>
<td>TDOA</td>
<td>0.5373</td>
<td>0.6294</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0196</td>
<td>0.0185</td>
<td>TDOA</td>
<td>0.0136</td>
<td>0.0245</td>
<td>TWR</td>
<td>0.0243</td>
<td>0.0166</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0004</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0002</td>
<td>0.0006</td>
<td>TWR</td>
<td>0.0006</td>
<td>0.0003</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1221</td>
<td>0.1512</td>
<td>TWR</td>
<td>0.0964</td>
<td>0.1347</td>
<td>TWR</td>
<td>0.1312</td>
<td>0.0940</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.4734</td>
<td>1.5483</td>
<td>TDOA</td>
<td>0.7195</td>
<td>0.4170</td>
<td>TDOA</td>
<td>0.6954</td>
<td>0.5418</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0263</td>
<td>0.0156</td>
<td>TDOA</td>
<td>0.0189</td>
<td>0.0206</td>
<td>TWR</td>
<td>0.0226</td>
<td>0.0157</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0007</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0004</td>
<td>0.0004</td>
<td>TWR</td>
<td>0.0005</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1714</td>
<td>0.0938</td>
<td>TDOA</td>
<td>0.1167</td>
<td>0.1434</td>
<td>TWR</td>
<td>0.1452</td>
<td>0.0826</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.4901</td>
<td>2.8030</td>
<td>TWR</td>
<td>0.6654</td>
<td>0.1917</td>
<td>TWR</td>
<td>0.7707</td>
<td>0.4877</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0245</td>
<td>0.0358</td>
<td>TWR</td>
<td>0.0277</td>
<td>0.0398</td>
<td>TWR</td>
<td>0.0223</td>
<td>0.0286</td>
<td>TWR</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
<td>0.0013</td>
<td>TWR</td>
<td>0.0008</td>
<td>0.0016</td>
<td>TWR</td>
<td>0.0005</td>
<td>0.0008</td>
<td>TWR</td>
</tr>
<tr>
<td>Range</td>
<td>0.1521</td>
<td>0.2088</td>
<td>TWR</td>
<td>0.1607</td>
<td>0.2073</td>
<td>TWR</td>
<td>0.1196</td>
<td>0.1669</td>
<td>TWR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5890</td>
<td>0.5233</td>
<td>TDOA</td>
<td>1.5933</td>
<td>1.5251</td>
<td>TDOA</td>
<td>0.6424</td>
<td>0.6479</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0227</td>
<td>0.0135</td>
<td>TDOA</td>
<td>0.0314</td>
<td>0.0098</td>
<td>TDOA</td>
<td>0.0255</td>
<td>0.0122</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0005</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0010</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0007</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1221</td>
<td>0.0740</td>
<td>TDOA</td>
<td>0.1587</td>
<td>0.0731</td>
<td>TDOA</td>
<td>0.1462</td>
<td>0.0948</td>
<td>TDOA</td>
</tr>
</tbody>
</table>
### Table 1: 1.55 | 1.55 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.4509</td>
<td>1.4796</td>
<td>TDOA</td>
<td>1.5801</td>
<td>1.5544</td>
<td>TDOA</td>
<td>0.6566</td>
<td>0.6083</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0342</td>
<td>0.0137</td>
<td>TDOA</td>
<td>0.0223</td>
<td>0.0136</td>
<td>TDOA</td>
<td>0.0365</td>
<td>0.0132</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0012</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0005</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0013</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1737</td>
<td>0.0795</td>
<td>TDOA</td>
<td>0.1505</td>
<td>0.0951</td>
<td>TDOA</td>
<td>0.2086</td>
<td>0.0803</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

### Table 2: 2.58 | 1.55 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.4186</td>
<td>2.5592</td>
<td>TDOA</td>
<td>1.5713</td>
<td>1.5582</td>
<td>TDOA</td>
<td>0.5885</td>
<td>0.5292</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0296</td>
<td>0.0123</td>
<td>TDOA</td>
<td>0.0259</td>
<td>0.0160</td>
<td>TDOA</td>
<td>0.0319</td>
<td>0.0136</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0009</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0007</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0010</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1657</td>
<td>0.0750</td>
<td>TDOA</td>
<td>0.1525</td>
<td>0.0872</td>
<td>TDOA</td>
<td>0.1940</td>
<td>0.0833</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

### Table 3: 0.516 | 2.58 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.4640</td>
<td>0.4533</td>
<td>TWR</td>
<td>2.5404</td>
<td>2.6140</td>
<td>TDOA</td>
<td>0.6033</td>
<td>0.6662</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0195</td>
<td>0.0171</td>
<td>TDOA</td>
<td>0.0187</td>
<td>0.0143</td>
<td>TDOA</td>
<td>0.0287</td>
<td>0.0146</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0004</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0003</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0008</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1078</td>
<td>0.0936</td>
<td>TDOA</td>
<td>0.1220</td>
<td>0.0888</td>
<td>TDOA</td>
<td>0.1765</td>
<td>0.0906</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

### Table 4: 1.55 | 2.58 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.3494</td>
<td>1.4901</td>
<td>TDOA</td>
<td>2.4218</td>
<td>2.6760</td>
<td>TDOA</td>
<td>0.6937</td>
<td>0.6219</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0237</td>
<td>0.0150</td>
<td>TDOA</td>
<td>0.0235</td>
<td>0.0180</td>
<td>TDOA</td>
<td>0.0256</td>
<td>0.0151</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0006</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0007</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1269</td>
<td>0.0870</td>
<td>TDOA</td>
<td>0.1542</td>
<td>0.0936</td>
<td>TDOA</td>
<td>0.1398</td>
<td>0.0818</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

### Table 5: 2.58 | 2.58 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.3462</td>
<td>2.5258</td>
<td>TDOA</td>
<td>2.3598</td>
<td>2.6643</td>
<td>TDOA</td>
<td>0.6773</td>
<td>0.5525</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0249</td>
<td>0.0181</td>
<td>TDOA</td>
<td>0.0260</td>
<td>0.0205</td>
<td>TDOA</td>
<td>0.0239</td>
<td>0.0151</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0007</td>
<td>0.0004</td>
<td>TDOA</td>
<td>0.0006</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1423</td>
<td>0.1048</td>
<td>TDOA</td>
<td>0.1700</td>
<td>0.1105</td>
<td>TDOA</td>
<td>0.1239</td>
<td>0.0970</td>
<td>TDOA</td>
</tr>
</tbody>
</table>
Table 14: Table 10: Logic Test Result Table Set (Middle Plane)

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.6539</td>
<td>0.4963</td>
<td>TDOA</td>
<td>0.5802</td>
<td>0.3965</td>
<td>TWR</td>
<td>1.6307</td>
<td>1.6620</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0244</td>
<td>0.0162</td>
<td>TDOA</td>
<td>0.0195</td>
<td>0.0159</td>
<td>TDOA</td>
<td>0.0233</td>
<td>0.0137</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0004</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0005</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1593</td>
<td>0.1051</td>
<td>TDOA</td>
<td>0.1020</td>
<td>0.0939</td>
<td>TDOA</td>
<td>0.1434</td>
<td>0.0738</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.5841</td>
<td>1.5228</td>
<td>TDOA</td>
<td>0.6671</td>
<td>0.5096</td>
<td>TDOA</td>
<td>1.6800</td>
<td>1.6174</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0190</td>
<td>0.0131</td>
<td>TDOA</td>
<td>0.0136</td>
<td>0.0141</td>
<td>TWR</td>
<td>0.0155</td>
<td>0.0102</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0004</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0002</td>
<td>0.0002</td>
<td>TWR</td>
<td>0.0002</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1217</td>
<td>0.0784</td>
<td>TDOA</td>
<td>0.0877</td>
<td>0.1104</td>
<td>TWR</td>
<td>0.0891</td>
<td>0.0658</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.4850</td>
<td>2.5245</td>
<td>TDOA</td>
<td>0.6292</td>
<td>0.4624</td>
<td>TDOA</td>
<td>1.8422</td>
<td>1.8098</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0170</td>
<td>0.0149</td>
<td>TDOA</td>
<td>0.0150</td>
<td>0.0196</td>
<td>TWR</td>
<td>0.0203</td>
<td>0.0111</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0003</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0002</td>
<td>0.0004</td>
<td>TWR</td>
<td>0.0004</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1001</td>
<td>0.0832</td>
<td>TDOA</td>
<td>0.0935</td>
<td>0.1191</td>
<td>TWR</td>
<td>0.1167</td>
<td>0.0633</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5927</td>
<td>0.2488</td>
<td>TWR</td>
<td>1.5844</td>
<td>1.4900</td>
<td>TWR</td>
<td>1.7215</td>
<td>1.7006</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0262</td>
<td>0.0522</td>
<td>TWR</td>
<td>0.0297</td>
<td>0.0492</td>
<td>TWR</td>
<td>0.0322</td>
<td>0.0309</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0007</td>
<td>0.0027</td>
<td>TWR</td>
<td>0.0009</td>
<td>0.0024</td>
<td>TWR</td>
<td>0.0010</td>
<td>0.0010</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1460</td>
<td>0.2805</td>
<td>TWR</td>
<td>0.1590</td>
<td>0.2864</td>
<td>TWR</td>
<td>0.1858</td>
<td>0.1938</td>
<td>TWR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.5140</td>
<td>1.4663</td>
<td>TWR</td>
<td>1.5624</td>
<td>1.5573</td>
<td>TDOA</td>
<td>1.7224</td>
<td>1.6921</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0237</td>
<td>0.0127</td>
<td>TDOA</td>
<td>0.0140</td>
<td>0.0152</td>
<td>TWR</td>
<td>0.0225</td>
<td>0.0106</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0002</td>
<td>0.0002</td>
<td>TWR</td>
<td>0.0005</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1291</td>
<td>0.0735</td>
<td>TDOA</td>
<td>0.0772</td>
<td>0.0956</td>
<td>TWR</td>
<td>0.1272</td>
<td>0.0684</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.4321</td>
<td>2.5275</td>
<td>TDOA</td>
<td>1.5310</td>
<td>1.4975</td>
<td>TWR</td>
<td>1.7672</td>
<td>1.7760</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0176</td>
<td>0.0158</td>
<td>TDOA</td>
<td>0.0276</td>
<td>0.0168</td>
<td>TDOA</td>
<td>0.0243</td>
<td>0.0114</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0003</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0008</td>
<td>0.0003</td>
<td>TDOA</td>
<td>0.0006</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0994</td>
<td>0.0903</td>
<td>TDOA</td>
<td>0.1456</td>
<td>0.1019</td>
<td>TDOA</td>
<td>0.1362</td>
<td>0.0594</td>
<td>TDOA</td>
</tr>
<tr>
<td>Metric</td>
<td>XTWR</td>
<td>XTDOA</td>
<td>Successor</td>
<td>YTWR</td>
<td>YTDOA</td>
<td>Successor</td>
<td>ZTWR</td>
<td>ZTDOA</td>
<td>Successor</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------</td>
<td>-----------</td>
<td>------</td>
<td>-------</td>
<td>-----------</td>
<td>------</td>
<td>-------</td>
<td>-----------</td>
</tr>
<tr>
<td>Mean</td>
<td>0.5251</td>
<td>0.2749</td>
<td>TWR</td>
<td>2.4936</td>
<td>2.5940</td>
<td>TDOA</td>
<td>1.6937</td>
<td>1.6914</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0225</td>
<td>0.0370</td>
<td>TWR</td>
<td>0.0178</td>
<td>0.0354</td>
<td>TWR</td>
<td>0.0209</td>
<td>0.0228</td>
<td>TWR</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0005</td>
<td>0.0014</td>
<td>TWR</td>
<td>0.0003</td>
<td>0.0013</td>
<td>TWR</td>
<td>0.0004</td>
<td>0.0005</td>
<td>TWR</td>
</tr>
<tr>
<td>Range</td>
<td>0.1565</td>
<td>0.3342</td>
<td>TWR</td>
<td>0.1019</td>
<td>0.2221</td>
<td>TWR</td>
<td>0.1225</td>
<td>0.1519</td>
<td>TWR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.5714</td>
<td>1.5489</td>
<td>TDOA</td>
<td>2.4643</td>
<td>2.7459</td>
<td>TWR</td>
<td>1.7900</td>
<td>1.7127</td>
<td>Successor</td>
</tr>
<tr>
<td>STD</td>
<td>0.0213</td>
<td>0.0291</td>
<td>TWR</td>
<td>0.0162</td>
<td>0.0370</td>
<td>TWR</td>
<td>0.0285</td>
<td>0.0276</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0005</td>
<td>0.0008</td>
<td>TWR</td>
<td>0.0003</td>
<td>0.0014</td>
<td>TWR</td>
<td>0.0008</td>
<td>0.0008</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.1390</td>
<td>0.1826</td>
<td>TWR</td>
<td>0.0938</td>
<td>0.3344</td>
<td>TWR</td>
<td>0.1512</td>
<td>0.1726</td>
<td>TWR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.3518</td>
<td>2.4745</td>
<td>TDOA</td>
<td>2.4940</td>
<td>2.5622</td>
<td>TDOA</td>
<td>1.7968</td>
<td>1.7797</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0135</td>
<td>0.0131</td>
<td>TDOA</td>
<td>0.0138</td>
<td>0.0358</td>
<td>TWR</td>
<td>0.0163</td>
<td>0.0107</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0002</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0002</td>
<td>0.0002</td>
<td>TWR</td>
<td>0.0003</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0760</td>
<td>0.0718</td>
<td>TDOA</td>
<td>0.0806</td>
<td>0.0840</td>
<td>TWR</td>
<td>0.1006</td>
<td>0.0601</td>
<td>TDOA</td>
</tr>
</tbody>
</table>
### Table 15: Logic Test Result Table Set (Top Plane)

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.6597</td>
<td>0.4361</td>
<td>TDOA</td>
<td>0.6979</td>
<td>0.4164</td>
<td>TDOA</td>
<td>2.4473</td>
<td>2.6716</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0049</td>
<td>0.0108</td>
<td>TDOA</td>
<td>0.0062</td>
<td>0.0150</td>
<td>TDOA</td>
<td>0.0038</td>
<td>0.0147</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0002</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0268</td>
<td>0.0700</td>
<td>TDOA</td>
<td>0.0440</td>
<td>0.0874</td>
<td>TDOA</td>
<td>0.0248</td>
<td>0.0954</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.5809</td>
<td>1.4701</td>
<td>TWR</td>
<td>0.6850</td>
<td>0.4435</td>
<td>TDOA</td>
<td>2.4913</td>
<td>2.7637</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0054</td>
<td>0.0111</td>
<td>TDOA</td>
<td>0.0052</td>
<td>0.0119</td>
<td>TDOA</td>
<td>0.0042</td>
<td>0.0105</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0354</td>
<td>0.0715</td>
<td>TDOA</td>
<td>0.0304</td>
<td>0.0667</td>
<td>TDOA</td>
<td>0.0253</td>
<td>0.0642</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.5271</td>
<td>2.5235</td>
<td>TWR</td>
<td>0.5933</td>
<td>0.5232</td>
<td>TDOA</td>
<td>2.6457</td>
<td>2.8085</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0052</td>
<td>0.0091</td>
<td>TDOA</td>
<td>0.0056</td>
<td>0.0092</td>
<td>TDOA</td>
<td>0.0037</td>
<td>0.0089</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0316</td>
<td>0.0513</td>
<td>TDOA</td>
<td>0.0350</td>
<td>0.0550</td>
<td>TDOA</td>
<td>0.0228</td>
<td>0.0597</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5390</td>
<td>0.3741</td>
<td>TWR</td>
<td>1.5139</td>
<td>1.5155</td>
<td>TDOA</td>
<td>2.5919</td>
<td>2.8046</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0054</td>
<td>0.0102</td>
<td>TDOA</td>
<td>0.0047</td>
<td>0.0095</td>
<td>TDOA</td>
<td>0.0040</td>
<td>0.0092</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0347</td>
<td>0.0586</td>
<td>TDOA</td>
<td>0.0373</td>
<td>0.0595</td>
<td>TDOA</td>
<td>0.0220</td>
<td>0.0532</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.4568</td>
<td>1.4195</td>
<td>TWR</td>
<td>1.5069</td>
<td>1.5483</td>
<td>TDOA</td>
<td>2.4562</td>
<td>2.7291</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0050</td>
<td>0.0084</td>
<td>TDOA</td>
<td>0.0072</td>
<td>0.0084</td>
<td>TDOA</td>
<td>0.0031</td>
<td>0.0080</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0001</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0368</td>
<td>0.0467</td>
<td>TDOA</td>
<td>0.0412</td>
<td>0.0461</td>
<td>TDOA</td>
<td>0.0166</td>
<td>0.0427</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.4569</td>
<td>2.5168</td>
<td>TDOA</td>
<td>1.5055</td>
<td>1.5227</td>
<td>TDOA</td>
<td>2.5968</td>
<td>2.7867</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0054</td>
<td>0.0097</td>
<td>TDOA</td>
<td>0.0055</td>
<td>0.0098</td>
<td>TDOA</td>
<td>0.0062</td>
<td>0.0106</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0361</td>
<td>0.0659</td>
<td>TDOA</td>
<td>0.0339</td>
<td>0.0587</td>
<td>TDOA</td>
<td>0.0343</td>
<td>0.0620</td>
<td>TDOA</td>
</tr>
</tbody>
</table>
### 0.516 | 2.58 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.6366</td>
<td>0.3553</td>
<td>TWR</td>
<td>2.2885</td>
<td>2.6501</td>
<td>TDOA</td>
<td>2.5891</td>
<td>2.8632</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0045</td>
<td>0.0197</td>
<td>TDOA</td>
<td>0.0059</td>
<td>0.0231</td>
<td>TDOA</td>
<td>0.0031</td>
<td>0.0208</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0004</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0005</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0004</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0268</td>
<td>0.1176</td>
<td>TDOA</td>
<td>0.0336</td>
<td>0.1300</td>
<td>TDOA</td>
<td>0.0164</td>
<td>0.1217</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

### 1.55 | 2.58 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.5781</td>
<td>1.4576</td>
<td>TDOA</td>
<td>2.4434</td>
<td>2.7455</td>
<td>TWR</td>
<td>2.4679</td>
<td>2.6900</td>
<td>TDOA</td>
</tr>
<tr>
<td>STD</td>
<td>0.0079</td>
<td>0.0130</td>
<td>TDOA</td>
<td>0.0127</td>
<td>0.0367</td>
<td>TDOA</td>
<td>0.0093</td>
<td>0.0164</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0001</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0002</td>
<td>0.0013</td>
<td>TDOA</td>
<td>0.0001</td>
<td>0.0003</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0469</td>
<td>0.0713</td>
<td>TDOA</td>
<td>0.0802</td>
<td>0.2069</td>
<td>TDOA</td>
<td>0.0497</td>
<td>0.0934</td>
<td>TDOA</td>
</tr>
</tbody>
</table>

### 2.58 | 2.58 | 0.516

<table>
<thead>
<tr>
<th>Metric</th>
<th>XTWR</th>
<th>XTDOA</th>
<th>Successor</th>
<th>YTWR</th>
<th>YTDOA</th>
<th>Successor</th>
<th>ZTWR</th>
<th>ZTDOA</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>2.3930</td>
<td>2.3882</td>
<td>TWR</td>
<td>2.5234</td>
<td>2.7453</td>
<td>TWR</td>
<td>2.5730</td>
<td>2.7789</td>
<td>TWR</td>
</tr>
<tr>
<td>STD</td>
<td>0.0044</td>
<td>0.0129</td>
<td>TDOA</td>
<td>0.0050</td>
<td>0.0230</td>
<td>TDOA</td>
<td>0.0039</td>
<td>0.0113</td>
<td>TDOA</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0000</td>
<td>0.0002</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0005</td>
<td>TDOA</td>
<td>0.0000</td>
<td>0.0001</td>
<td>TDOA</td>
</tr>
<tr>
<td>Range</td>
<td>0.0256</td>
<td>0.0847</td>
<td>TDOA</td>
<td>0.0281</td>
<td>0.1357</td>
<td>TDOA</td>
<td>0.0322</td>
<td>0.0741</td>
<td>TDOA</td>
</tr>
</tbody>
</table>
Appendix D

The tables in Appendix D are the raw statistical data sets as computed by MATLAB. Each table contains six calculated measures, across the nine points for a single coordinate on one of three measurement planes. Note that these nine points are not specific to any coordinate.

Table 16: Performance Metrics for 9 Points at Top Plane

<table>
<thead>
<tr>
<th>TOP PLANE</th>
<th>TDOA X</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Mean</td>
<td>0.4361</td>
<td>1.4701</td>
<td>2.5235</td>
<td>0.3741</td>
<td>1.4195</td>
<td>2.5168</td>
<td>0.3553</td>
<td>1.4576</td>
</tr>
<tr>
<td>STD</td>
<td>0.0108</td>
<td>0.0111</td>
<td>0.0091</td>
<td>0.0102</td>
<td>0.0084</td>
<td>0.0097</td>
<td>0.0197</td>
<td>0.0130</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0004</td>
</tr>
<tr>
<td>Range</td>
<td>0.0700</td>
<td>0.0715</td>
<td>0.0513</td>
<td>0.0586</td>
<td>0.0467</td>
<td>0.0659</td>
<td>0.1176</td>
<td>0.0713</td>
</tr>
<tr>
<td>Max</td>
<td>0.3994</td>
<td>1.4349</td>
<td>2.5010</td>
<td>0.3441</td>
<td>1.3964</td>
<td>2.4831</td>
<td>0.3006</td>
<td>1.4234</td>
</tr>
<tr>
<td>Min</td>
<td>0.4693</td>
<td>1.5065</td>
<td>2.5522</td>
<td>0.4026</td>
<td>1.4431</td>
<td>2.5491</td>
<td>0.4182</td>
<td>1.4947</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOP PLANE</th>
<th>TDOA Y</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Mean</td>
<td>0.4164</td>
<td>0.4435</td>
<td>0.5232</td>
<td>1.5155</td>
<td>1.5483</td>
<td>1.5227</td>
<td>2.6501</td>
<td>2.7455</td>
</tr>
<tr>
<td>STD</td>
<td>0.0150</td>
<td>0.0119</td>
<td>0.0092</td>
<td>0.0095</td>
<td>0.0084</td>
<td>0.0098</td>
<td>0.0231</td>
<td>0.0367</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0005</td>
<td>0.0013</td>
</tr>
<tr>
<td>Range</td>
<td>0.0874</td>
<td>0.0667</td>
<td>0.0550</td>
<td>0.0595</td>
<td>0.0461</td>
<td>0.0587</td>
<td>0.1300</td>
<td>0.2069</td>
</tr>
<tr>
<td>Max</td>
<td>0.3685</td>
<td>0.4123</td>
<td>0.4940</td>
<td>1.4829</td>
<td>1.5255</td>
<td>1.4924</td>
<td>2.5722</td>
<td>2.6519</td>
</tr>
<tr>
<td>Min</td>
<td>0.4559</td>
<td>0.4790</td>
<td>0.5490</td>
<td>1.5424</td>
<td>1.5716</td>
<td>1.5511</td>
<td>2.7022</td>
<td>2.8588</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOP PLANE</th>
<th>TDOA Z</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Mean</td>
<td>2.6716</td>
<td>2.7637</td>
<td>2.8085</td>
<td>2.8046</td>
<td>2.7291</td>
<td>2.7867</td>
<td>2.8632</td>
<td>2.6900</td>
</tr>
<tr>
<td>STD</td>
<td>0.0147</td>
<td>0.0105</td>
<td>0.0089</td>
<td>0.0092</td>
<td>0.0080</td>
<td>0.0106</td>
<td>0.0208</td>
<td>0.0164</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0004</td>
<td>0.0003</td>
</tr>
<tr>
<td>Range</td>
<td>0.0954</td>
<td>0.0642</td>
<td>0.0597</td>
<td>0.0532</td>
<td>0.0427</td>
<td>0.0620</td>
<td>0.1217</td>
<td>0.0934</td>
</tr>
<tr>
<td>Max</td>
<td>2.6234</td>
<td>2.7307</td>
<td>2.7766</td>
<td>2.7784</td>
<td>2.7085</td>
<td>2.7521</td>
<td>2.7965</td>
<td>2.6449</td>
</tr>
<tr>
<td>Min</td>
<td>2.7188</td>
<td>2.7949</td>
<td>2.8363</td>
<td>2.8316</td>
<td>2.7512</td>
<td>2.8141</td>
<td>2.9181</td>
<td>2.7384</td>
</tr>
<tr>
<td>TWR X</td>
<td>Mean</td>
<td>0.6597</td>
<td>2.5271</td>
<td>1.5809</td>
<td>0.5390</td>
<td>1.4568</td>
<td>2.4569</td>
<td>0.6366</td>
</tr>
<tr>
<td></td>
<td>STD</td>
<td>0.0049</td>
<td>0.0052</td>
<td>0.0054</td>
<td>0.0054</td>
<td>0.0050</td>
<td>0.0054</td>
<td>0.0045</td>
</tr>
<tr>
<td></td>
<td>VAR</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td>0.0268</td>
<td>0.0316</td>
<td>0.0354</td>
<td>0.0347</td>
<td>0.0368</td>
<td>0.0361</td>
<td>0.0268</td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>0.6474</td>
<td>2.5119</td>
<td>1.5629</td>
<td>0.5221</td>
<td>1.4390</td>
<td>2.4391</td>
<td>0.6212</td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>0.6742</td>
<td>2.5435</td>
<td>1.5983</td>
<td>0.5567</td>
<td>1.4757</td>
<td>2.4752</td>
<td>0.6480</td>
</tr>
</tbody>
</table>

| TWR Y | Mean  | 0.6979 | 0.5933 | 0.6850 | 1.5139 | 1.5069 | 1.5055 | 2.2885 | 2.4434 | 2.5234 |
|       | STD    | 0.0062 | 0.0056 | 0.0052 | 0.0047 | 0.0072 | 0.0055 | 0.0059 | 0.0127 | 0.0050 |
|       | VAR    | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0001 | 0.0000 | 0.0000 | 0.0002 | 0.0000 |
|       | Range  | 0.0440 | 0.0350 | 0.0304 | 0.0373 | 0.0412 | 0.0339 | 0.0336 | 0.0802 | 0.0281 |
|       | Max    | 0.6704 | 0.5753 | 0.6730 | 1.4981 | 1.4857 | 1.4877 | 2.2745 | 2.4019 | 2.5094 |
|       | Min    | 0.7144 | 0.6103 | 0.7034 | 1.5354 | 1.5269 | 1.5215 | 2.3081 | 2.4821 | 2.5374 |

| TWR Z | Mean  | 2.4473 | 2.6457 | 2.4913 | 2.5919 | 2.4562 | 2.5968 | 2.5891 | 2.4679 | 2.5730 |
|       | STD    | 0.0038 | 0.0037 | 0.0042 | 0.0040 | 0.0031 | 0.0062 | 0.0031 | 0.0093 | 0.0039 |
|       | VAR    | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0001 | 0.0000 |
|       | Range  | 0.0248 | 0.0228 | 0.0253 | 0.0220 | 0.0166 | 0.0343 | 0.0164 | 0.0497 | 0.0322 |
|       | Max    | 2.4355 | 2.6350 | 2.4798 | 2.5821 | 2.4473 | 2.5763 | 2.5810 | 2.4472 | 2.5622 |
|       | Min    | 2.4603 | 2.6578 | 2.5051 | 2.6041 | 2.4639 | 2.6106 | 2.5975 | 2.4969 | 2.5944 |

Table 17: Performance Metrics for 9 Points at Middle Plane

<table>
<thead>
<tr>
<th>Middle PLANE</th>
<th>TDOA X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Mean</td>
<td>0.4963</td>
</tr>
<tr>
<td>STD</td>
<td>0.0162</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0003</td>
</tr>
<tr>
<td></td>
<td>Range</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>0.1051</td>
</tr>
<tr>
<td></td>
<td>0.2805</td>
</tr>
<tr>
<td></td>
<td>0.3342</td>
</tr>
<tr>
<td>TDOA Y</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.3965</td>
</tr>
<tr>
<td>STD</td>
<td>0.0159</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0003</td>
</tr>
<tr>
<td>Range</td>
<td>0.0939</td>
</tr>
<tr>
<td>Max</td>
<td>0.3572</td>
</tr>
<tr>
<td>Min</td>
<td>0.4511</td>
</tr>
<tr>
<td>TDOA Z</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>1.6620</td>
</tr>
<tr>
<td>STD</td>
<td>0.0137</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0002</td>
</tr>
<tr>
<td>Range</td>
<td>0.0738</td>
</tr>
<tr>
<td>Max</td>
<td>1.6234</td>
</tr>
<tr>
<td>Min</td>
<td>1.6972</td>
</tr>
<tr>
<td>TWR X</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.6539</td>
</tr>
<tr>
<td>STD</td>
<td>0.0244</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
</tr>
<tr>
<td>Range</td>
<td>0.1593</td>
</tr>
<tr>
<td>Max</td>
<td>0.5696</td>
</tr>
<tr>
<td>Min</td>
<td>0.7289</td>
</tr>
<tr>
<td>TWR Y</td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.5802</td>
</tr>
<tr>
<td>STD</td>
<td>0.0195</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0004</td>
</tr>
<tr>
<td>Range</td>
<td>0.1020</td>
</tr>
<tr>
<td>Max</td>
<td>0.5325</td>
</tr>
</tbody>
</table>
Table 18: Performance Metrics for 9 Points at Ground Plane

<table>
<thead>
<tr>
<th></th>
<th>TWR Z</th>
<th>TDOA X</th>
<th>TDOA Y</th>
<th>TDOA Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.6345</td>
<td>0.7078</td>
<td>0.6742</td>
<td>1.6598</td>
</tr>
<tr>
<td>Mean</td>
<td>1.6307</td>
<td>1.6800</td>
<td>1.7215</td>
<td>1.7224</td>
</tr>
<tr>
<td>STD</td>
<td>0.0233</td>
<td>0.0155</td>
<td>0.0203</td>
<td>0.0322</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0005</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0005</td>
</tr>
<tr>
<td>Range</td>
<td>0.1434</td>
<td>0.0891</td>
<td>0.1167</td>
<td>0.1272</td>
</tr>
<tr>
<td>Max</td>
<td>1.5550</td>
<td>1.6398</td>
<td>1.7891</td>
<td>1.6965</td>
</tr>
<tr>
<td>Min</td>
<td>1.6983</td>
<td>1.7289</td>
<td>1.9058</td>
<td>1.7989</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TDOA Z</th>
<th>TDOA X</th>
<th>TDOA Y</th>
<th>TDOA Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1.6983</td>
<td>1.7289</td>
<td>1.9058</td>
</tr>
<tr>
<td>Mean</td>
<td>1.6307</td>
<td>1.6800</td>
<td>1.7215</td>
</tr>
<tr>
<td>STD</td>
<td>0.0233</td>
<td>0.0155</td>
<td>0.0203</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0005</td>
<td>0.0002</td>
<td>0.0004</td>
</tr>
<tr>
<td>Range</td>
<td>0.1434</td>
<td>0.0891</td>
<td>0.1167</td>
</tr>
<tr>
<td>Max</td>
<td>1.5550</td>
<td>1.6398</td>
<td>1.7891</td>
</tr>
<tr>
<td>Min</td>
<td>1.6983</td>
<td>1.7289</td>
<td>1.9058</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>TDOA X</th>
<th>TDOA Y</th>
<th>TDOA Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0.6345</td>
<td>0.7078</td>
<td>0.6742</td>
</tr>
<tr>
<td>Mean</td>
<td>1.6307</td>
<td>1.6800</td>
<td>1.7215</td>
</tr>
<tr>
<td>STD</td>
<td>0.0233</td>
<td>0.0155</td>
<td>0.0203</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0005</td>
<td>0.0002</td>
<td>0.0004</td>
</tr>
<tr>
<td>Range</td>
<td>0.1434</td>
<td>0.0891</td>
<td>0.1167</td>
</tr>
<tr>
<td>Max</td>
<td>1.5550</td>
<td>1.6398</td>
<td>1.7891</td>
</tr>
<tr>
<td>Min</td>
<td>1.6983</td>
<td>1.7289</td>
<td>1.9058</td>
</tr>
<tr>
<td></td>
<td>Mean</td>
<td>STD</td>
<td>VAR</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>Mean</td>
<td>0.6294</td>
<td>0.0166</td>
<td>0.0003</td>
</tr>
<tr>
<td>STD</td>
<td>0.0166</td>
<td>0.0157</td>
<td>0.0002</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0003</td>
<td>0.0002</td>
<td>0.0001</td>
</tr>
<tr>
<td>Range</td>
<td>0.0940</td>
<td>0.0826</td>
<td>0.1669</td>
</tr>
<tr>
<td>Max</td>
<td>0.5824</td>
<td>0.5069</td>
<td>0.4254</td>
</tr>
<tr>
<td>Min</td>
<td>0.6765</td>
<td>0.5894</td>
<td>0.5923</td>
</tr>
</tbody>
</table>

**TWR X**

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>STD</th>
<th>VAR</th>
<th>Range</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5434</td>
<td>0.0196</td>
<td>0.0004</td>
<td>0.1221</td>
<td>0.4791</td>
<td>0.6012</td>
</tr>
<tr>
<td>STD</td>
<td>0.0196</td>
<td>0.0263</td>
<td>0.0002</td>
<td>0.1714</td>
<td>1.3902</td>
<td>1.5617</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0004</td>
<td>0.0007</td>
<td>0.0006</td>
<td>0.1521</td>
<td>0.5191</td>
<td>0.7195</td>
</tr>
<tr>
<td>Range</td>
<td>0.1221</td>
<td>0.1714</td>
<td>0.1521</td>
<td>0.1221</td>
<td>1.3558</td>
<td>1.5657</td>
</tr>
<tr>
<td>Max</td>
<td>0.4791</td>
<td>1.3902</td>
<td>2.4302</td>
<td>1.6573</td>
<td>2.4302</td>
<td>2.5404</td>
</tr>
<tr>
<td>Min</td>
<td>0.6012</td>
<td>1.5617</td>
<td>2.5823</td>
<td>0.5145</td>
<td>1.4092</td>
<td>2.4465</td>
</tr>
</tbody>
</table>

**TWR Y**

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>STD</th>
<th>VAR</th>
<th>Range</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5723</td>
<td>0.0136</td>
<td>0.0002</td>
<td>0.0964</td>
<td>0.5251</td>
<td>0.6215</td>
</tr>
<tr>
<td>STD</td>
<td>0.0136</td>
<td>0.0189</td>
<td>0.0001</td>
<td>0.1167</td>
<td>0.6644</td>
<td>0.7811</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0002</td>
<td>0.0004</td>
<td>0.0008</td>
<td>0.1607</td>
<td>0.5691</td>
<td>0.7298</td>
</tr>
<tr>
<td>Range</td>
<td>0.0964</td>
<td>0.1167</td>
<td>0.1607</td>
<td>0.1587</td>
<td>1.5161</td>
<td>1.6748</td>
</tr>
<tr>
<td>Max</td>
<td>0.5251</td>
<td>0.6644</td>
<td>1.5161</td>
<td>0.5152</td>
<td>1.5048</td>
<td>1.6574</td>
</tr>
<tr>
<td>Min</td>
<td>0.6215</td>
<td>0.7811</td>
<td>0.7298</td>
<td>0.5145</td>
<td>0.5145</td>
<td>1.4092</td>
</tr>
</tbody>
</table>

**TWR Z**

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>STD</th>
<th>VAR</th>
<th>Range</th>
<th>Max</th>
<th>Min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.5373</td>
<td>0.0243</td>
<td>0.0006</td>
<td>0.1312</td>
<td>0.4703</td>
<td>0.6015</td>
</tr>
<tr>
<td>STD</td>
<td>0.0243</td>
<td>0.0226</td>
<td>0.0005</td>
<td>0.1452</td>
<td>0.6097</td>
<td>0.7548</td>
</tr>
<tr>
<td>VAR</td>
<td>0.0006</td>
<td>0.0005</td>
<td>0.0005</td>
<td>0.1462</td>
<td>0.6973</td>
<td>0.8169</td>
</tr>
<tr>
<td>Range</td>
<td>0.1312</td>
<td>0.1452</td>
<td>0.1462</td>
<td>0.2086</td>
<td>0.6973</td>
<td>0.7014</td>
</tr>
<tr>
<td>Max</td>
<td>0.4703</td>
<td>0.6097</td>
<td>0.6973</td>
<td>0.5387</td>
<td>0.4748</td>
<td>0.6688</td>
</tr>
<tr>
<td>Min</td>
<td>0.6015</td>
<td>0.7548</td>
<td>0.8169</td>
<td>0.5387</td>
<td>0.5387</td>
<td>0.6688</td>
</tr>
</tbody>
</table>
Appendix E

Email correspondence concerning the ranging capabilities of the LPS nodes, and possible firmware modifications.

Hi,

I am currently determining the feasibility of hardware for use in my engineering honours thesis.

Would you be able to clarify whether the Loco Positioning Node can be used as a "Tag" and "Anchor/Node" at the same time, or just one functionality at a time?

Also utilising the "Anchor/Node" functionality, does the positioning "node/anchor" measure distance from the other nodes or only from the tag?

Thanks,


Hi Luke,

In the currently implemented system the nodes can act as both a tag and anchors. In Two Way Ranging mode, which is the simplest mode and supports one tag, the tag will ping the anchors one after each other and report the distance to the anchors. Currently the mode are independent; a node will be either set as tag or anchor. The system is designed this way since we want the position information in the tag (i.e. in the robot).

So to answer your questions with the current implementation:
- The node can be used either as tag or anchor, there is currently no anchor/tag mode (though TDoA anchor can be considered as tag/anchor, see below).
- Currently the tag will measure the distance to the anchors, the results being the distance to all the anchors.

One idea, if you want and anchor/tag mode, would be to use the experimental TDoA mode: in TDoA mode we need to measure the time of flight between anchor in order to synchronize there clock, this means that the anchor are essentially constantly measuring distance between each-other. Right now the algorithm is designed in such a way that this information is sent back in the air so that a listener (for example a node in sniffer mode) can synchronize the anchor clock and measure its own position, the algorithm could be modified to send the information on the usb port which would allows you to get the distance to all other anchor from each anchor.

It is an open source and everything is based on the capability of the radio to time-stamp very accurately the departure and arrival of packet. This means that you will be able to adapt the system to your needs. We currently have a focus on programming modes to fly a swarm of Crazyflies using the nodes as fixed anchor, however the firmware is designed in such a way that it is easy to add new algorithm to it.

I hope these information helped.
Best regards,

Hi [Redacted]

Thank you very much for your detailed response, sounds like I will be using the TDoA method you described!

So if I understand correctly - utilising TDoA with USB essentially allows all anchors to measure the distance to the other anchors as no dedicated "sniffer" is required?

With outputting data over USB, is the algorithm added to the firmware of the Crazyflie or the anchor itself?

I'm in the planning phase of my thesis and will be acquiring hardware soon - trying to get an idea of what I will need to do in order to get things started!

Thanks again for your help.

Regards,

Hi Luke,

Yes, if the anchors outputs data on USB, it could output the distance (or at least enough info to calculate the distance). So anything connected to the USB port will get the distance from this anchor to all the others. The nodes also have a TTL 3.3V serial port if you need the data on another embedded system.

This would be a change to the anchor only; the Crazyflie in TDoA Tag mode is already in sniffer mode, this is how TDoA works (and this is why the nodes do not have a TDoA tag mode, the sniffer is the TDoA tag :).

Two points that might be important:
- TDoA is still considered experimental; it flies but it can be improved. Most importantly the Crazyflie needs to be in the convex hull formed by the anchor in order to locate itself properly (this is why we use 8 anchors in TDoA mode)
- TDoA schedules the anchor communication using a fixed TDMA scheme: each anchor have a fixed timeslot assigned according to their ID, the system is currently setup to work with 8 anchors. To add anchors the update rate of the system has to be decreased (more timeslot need to be added).

Best regards,
Email correspondence concerning Task-based execution structures and intermittent execution problems potentially caused by collisions or race conditions.

Hi,

I have been developing an autonomous positioning task on the Crazyflie, intended for use with the LOCO positioning system. The task writes a series of (x, y, z) set-points to the commander using CommanderSetSetpoint.

I have encountered a bug in my code which results in the set-point routine only being executed properly if I hold the Crazyflie for 3-5 seconds after powering on before putting it down to fly. However, the behavior is not always consistent.

From what I can guess this is likely due to either race conditions or priorities, I have reverse engineered similar instances of the "CommanderSetSetpoint" function but am not certain I am using it appropriately - similarly for my set-point structure.

Any advice would be greatly appreciated!

Thanks,


The general format of my task in pseudocode is as follows:

```c
void autotestTask()
{
    systemWaitStart();

    while (1)
    {
        // Delay and LED blink for debugging
        ledSet(CHG_LED, 1);
        vTaskDelay(pdMS_TO_TICKS(25));
        ledSet(CHG_LED, 0);
        vTaskDelay(pdMS_TO_TICKS(25));

        // Function which writes values into the set-point structure.
        setpointBuilder(&autosetpoint);

        // Take the the setpoint generated and write it to commander
        CommanderSetSetpoint(&autosetpoint, 2)
    }
}
```

I am attaching `autotestTask` which is executed on line 156 of `system.c` after `commInit` and `commanderInit`.

```c
void autotestInit()
{
    vTaskCreate(autotestTask, AUTOTEST_TASK_NAME, AUTOTEST_TASK_STACKSIZE, NULL, AUTOTEST_TASK_PRI, NULL);
}
```

What I am writing to the set-point is as follows:

- `tempsetpoint.mode.x = modeAbs;
- `tempsetpoint.mode.y = modeAbs;
- `tempsetpoint.mode.z = modeAbs;

- `tempsetpoint.position.x = 1.55; // Fixed to just test functionality tracking Z set point array
- `tempsetpoint.position.y = 1.55;
- `tempsetpoint.position.z = Value; // This variable is determined by an indexed array of Z positions

What I am writing to the set-point is as follows:
Hi Luke!

I'm not sure what this could be but I agree, it might be some sort of race condition.

We have actually done more or less what you are working on for the ICRA conference last year. You can find the code on github [https://github.com/loiterazzi/crazyflie-firmware-experiments/tree/icra-2017](https://github.com/loiterazzi/crazyflie-firmware-experiments/tree/icra-2017). Even if it is slightly outdated you can maybe get some inspiration.

We used a timer instead of a task in our implementation and you can find it in the arc/modules/arcr/treasure.c file. Also note that we init the module from the systemInit() function as opposed to the system task that you use.

In general I think timers are easier to use and have less risk of concurrency problems.

Please let me know this does not solve your problem!

Thanks.

I have managed to implement my code using the timer as you suggested.

Do you have any suggestions for trying to get the task functionality to work with the other tasks? It may be a point I wish to discuss in my thesis.

I have noticed within set-point structures throughout the source code the yaw is set to some "CAREFREE" mode. I find when I start my Crazyflie it works but only if facing forward in the world coordinate system. How should I approach setting the yaw i.e. velocity mode, or perhaps disabled?

Thank you for your help, I plan to post in the Bitcraze forum my methods for autonomous flight as I see many community members are trying to achieve this.

Regards,


Hi!

I'm happy that it works.

I think the task code that you supplied in the initial email looks OK, I'm not sure why it does not work. Maybe you could try to convert your currently working timer code to use a task instead?

The "crazyfree" stuff is only used if you fly the Crazyflie from a gamepad and should not really be related to what you are working on. It is mapping from user action to setpoints.

For the starting direction you are correct, you must start it pointing in the positive X direction (please see [https://www.bitcraze.io/getting-started-with-assisted-flight-position-hold/](https://www.bitcraze.io/getting-started-with-assisted-flight-position-hold/) for more info).

The position estimator (it actually estimates the full pose) assumes it is oriented in positive X at start up. It will converge to the correct orientation eventually if moved around, but there is very little directional information from the LPS system and it will take a long time. When using a positioning system that supports full pose (a mocap system for instance) the estimator will converge to the correct direction quickly.

>>> How should I approach setting the yaw i.e. velocity mode, or perhaps disabled?

The question is more complex than it might seem from the start. The API (the setpoint_i in stabilizer_types.h) is very flexible and it is possible to mix and match velocity/abs positions and so on in many ways, unfortunately the controllers do not support all combinations. The best way to approach this is to look at the CRTP protocol to see what is supported, since this is actually implemented in the controller. You can find the docs for the Generic Setpoint CRTP Port on the wiki [https://wiki.bitcraze.io/doc/crazyflie.cmdp-generic_setpoint](https://wiki.bitcraze.io/doc/crazyflie.cmdp-generic_setpoint). When you have found a suitable type, head over to the firmware and the cmdp_commander_generic.c file and take a look at how the setpoint struct is populated.
Appendix F
Field notes regarding the selection of equidistant measurement points.

Figure 48: Field Notes, LPS Configuration and Measurement Point Selection