University of Southern Queensland

Faculty of Health, Engineering & Sciences

Precision RTK GNSS for low-cost robotic systems

A dissertation submitted by

Simon Castles

In fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Mechatronics)

Submitted: October, 2021

Abstract

The agriculture industry has been under increasing pressure to do more with less; increase efficiency and productivity while minimising cost and environmental impacts. Advanced technology has become an essential part of farming and with improvements in GNSS, robotics and geospatial software, so has the potential for improving accuracy and control of farming practices.

Real-time kinematic (RTK) GNSS offers centimetre level accuracy using a base station situated at a known location providing correction data to one or more rover stations within a 10 km range. In agronomy, RTK GNSS enables more sustainable and profitable management practices, such as tractor guidance, and more recently precision application of herbicides. However, the capital investment in establishing an RTK network to gain precise localisation benefits often erode the profitability of the system when compared to a less accurate, cheaper system with reduced benefits. The aim of this project is to design a low cost RTK GNSS sub-system for an autonomous robotic system and evaluate the accuracy of the system in an agricultural setting.

A system was designed and built using ArduSimple RTK receiver boards (based on the Ublox ZED-F9P receiver module), Xbee radio modules and an STM32 microcontroller to provide both position and heading data. To verify the accuracy of the designed system, a testing procedure based on ISO 12188-1 was developed, which involved the design and production of a drive system on a non-metallic track so that the actual path traversed could be accurately recorded. The data from the RTK receiver was then analysed to calculate cross-track error and therefore the accuracy of the system, finding that centimetre level accuracy is achievable with low-cost receivers, with the system estimated to have a relative cross-track accuracy of 12.13 mm and a heading accuracy of 0.45°.

Keywords: Precision agriculture, robotics, localisation, GNSS, RTK, accuracy.

University of Southern Queensland
Faculty of Health, Engineering & Sciences

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean
Faculty of Health, Engineering & Sciences

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

S. Castles

[redacted]

Acknowledgments

The author would like to sincerely thank the following people for their help and support during the project:

I want to thank my supervisor, Dr Craig Lobsey, for his support to take this project in this direction, continued guidance and the original idea of the topic.

I would also like to thank and acknowledge my wife for her constant love and support, solving problems together and allowing me to spread electronics and acrylic all over the house.

S. Castles

Contents

List of Figures

List of Tables

Nomenclature and Acronyms

BLDC	Brushless DC
CEP	Circular error probability
CTPCO	Critical Technologies Policy Coordination Office
DGPS	Differential GPS
GNSS	Global navigation satellite system
GPIO	General purpose input/output
GPS	Global Positioning System
MCU	Microcontroller
PVT	Position,velocity and time
RMS	Root mean square
ROS2	Robotic operating system 2
RTC	Real time clock
RTK	Real time kinematic
RTOS	Real time operating system
SBC	Single board computer
SSM	Site specific management
UART	Universal asynchronous receiver-transmitter

Chapter 1

Introduction

1.1 Background

The agriculture industry has been under increasing pressure to do more with less; increase efficiency and productivity while minimising cost and environmental impacts. Advanced technology has become an essential part of farming and with improvements in Global navigation satellite systems (GNSS), robotics and geospatial software, so has the potential for improving accuracy and control of existing farming practices and developing new methods of farming through the use of robotics and automation.

Until recently, precision agriculture has been out of reach for many small scale producers due to cost. Innovation in the Australian agriculture industry is low (Department of the Prime Minister and Cabinet 2021), with some farmers hesitant to introduce new technology due to capital cost outweighing benefits. However, the affordability and availability of precise GNSS has the potential to make sustainable practices more widely implemented. The Critical Technologies Policy Coordination Office (CTPCO) has identified GNSS augmentation and autonomous vehicles as two of the eight critical technologies that are essential for securing the nation's economic prosperity (Department of the Prime Minister and Cabinet 2021). This project evaluates a low-cost GNSS receiver for use as a localisation module for an autonomous agricultural robotic system.

Real-time kinematic (RTK) GNSS offers centimetre level accuracy using a base station situated at a known location providing correction data to one or more rover stations within a 10 km range. In agronomy, RTK GNSS enables more sustainable and profitable management practices, such as tractor guidance, and more recently precision application of herbicides (Rybacki et al. 2021). For more than two decades, precision agriculture has been on the verge of transforming production agriculture, with the potential for increasing yields while maintaining or reducing inputs (Thompson et al. 2019).

However, the capital investment in establishing an RTK network to gain precise localisation benefits often erode the profitability of the system when compared to a less accurate, cheaper system with reduced benefits (Knight & Malcolm 2009). Several recent studies have found adoption rates for precision agriculture struggles to exceed fifty percent of farmers/producers or planted acers (Thompson et al. 2019). These studies tend to focus on large scale producers (greater than 1000 crop acres), which have higher adoption rates of precision agriculture. Moreover, cost-savings was considered more beneficial than yield improvements or convenience (Thompson et al. 2019). This suggests developing a low-cost system could make precision agriculture more accessible to small scale producers.

This report will outline the project's aims, objectives and research methodology for evaluating a low-cost GNSS board for integration into an agricultural robotic system. It will also outline the the resources required, project planning and risks involved. Finally, this report will highlight areas for further development and future research.

1.2 Project Aim and objectives

The aim of this project is to design a low cost RTK GNSS sub-system for an autonomous robotic system and evaluate the accuracy of the system in an agricultural setting. In order to achieve the aim of the project, the following objectives were to be achieved:

- Design and build a RTK GNSS receiver base station which transmits differential correction data to one or more rover stations, and
- Design and build a RTK GNSS rover station capable of providing accurate position, heading and speed to a mobile robotic system.
- Evaluate the built system in line with relevant standards.

A major consideration of the design is that it will be a module of a larger, more complex robotic system with the potential for development/integration into a commercial product. Therefore, the output of the the rover station must be able to be integrated with the larger system, which will be achieved through implementing robotic operating system 2 (ROS2) on the software stack of the design. In addition to designing the RTK GNSS system, a test rig has been designed and built; and test procedures developed in line with the requirements of relevant standards to ensure the functionality of the RTK GNSS system.

Further detail on the aim and objectives of the project are outlined in the Project Specification (Appendix A).

1.3 Implications and Ethics

1.3.1 Benefits to agriculture

By designing a low cost RTK-GNSS localisation system, it is believed that numerous smaller, multi-purpose autonomous vehicles can be developed to replace large agriculture vehicles. Having lightweight vehicles combined with centimetre level precision from an RTK system, one major benefit to an agriculture business is reduced soil compaction due to the reduced loads and repeatability of vehicle movement. Additionally, improvements in efficiency, cost-savings and yield could have positive impacts on the wellbeing of farmers.

1.3.2 Ethical considerations

Low cost electronics can lead to the belief that the product is for short term use and lead to a build up of e-waste and unsustainable use of finite resources. Reliability for long term use of the RTK GNSS system should be incorporated into the design, as well as ease of use so that the system is not discarded due to complexity in integration to a farming system. Open-source programming can also allow for wider technical support to the operator if maintenance is required rather than the system requiring proprietary tools for servicing.

1.4 Overview of the dissertation

A summary of the chapters within this dissertation is provided below:

Chapter 2 - Outlines the literature review that occurred to support the development of the methodology implemented as well has the design choices made in developing system hardware and software to achieve the project's aims and objectives.

Chapter 3 - Outlines the development of the evaluation methodology from the standards and the system concept design to be evaluated.

Chapter 4 - Outlines the hardware selected for the testing rig and the system under evaluation.

Chapter 5 - Outlines the software development that occurred in order to capture GNSS data for evaluation.

Chapter 6 - Outlines the results obtained during testing of the designed system.

Chapter 7 - Discusses the interpretation of the results and

Chapter 8 - Concludes the dissertation and proposes further work to be conducted to build upon this project in evaluating the accuracy of low-cost RTK GNSS receivers and their use.

Chapter 2

Relevant Literature

2.1 Chapter Overview

This chapter examines the trends in precision agriculture and the relationship to localisation systems and the growing need for ever increasing accuracy in systems. It will also give an overview of a low cost satellite based localisation development board, the ArduSimple SimpleRTK2b, which is based around Ublox's ZED-F9P GNSS receiver.

2.2 Precision Agriculture

Precision agriculture has introduced technologies that greatly enhances productivity. Through the use of data collection from all types of sensors from the atmosphere, the plants and even the soil, farmers can see much more efficient production and a reduced impact on the environment (Wolfert et al. 2017, Tzounis et al. 2017). Often though, to get the most out of the data collected, precise GNSS receivers are required to achieve the precise dosage of nutrients, fertilizers and pesticides (if still required) to the precise location it is needed (Perez-Ruiz & Upadhyaya 2012).

Site specific management (SSM) is an area of precision agriculture which largely focuses on crop nutrient levels and delivering variable fertilizer doses based on localised soil sample data. Over the last century, research on the soil sampling density has consistently suggested reducing crop management zone resolution, with advances in modern sensor technologies recognising variability under a meter (Zhang 2016). This has required ever increasing accuracy of localisation systems, down to centimetre level, which can be achieved through laser and satellite based systems (Williams et al. 2020). But this increased accuracy also poses new challenges, such as increased cost in man power and money to collect and analyse ever increasing data. One possible solution to this new challenge is the deployment of fleets of small autonomous vehicles to collect

data on crop nutrients, yield measurement as well as pest and weed management (Shamshiri et al. 2018). This reinforces the requirement for low cost, precise localisation systems if small unmanned vehicles are going to be operating in fleets and probably along side humans.

2.3 Satellite Based Localisation

Satellite based GNSS is the main form of localisation for humans and machines in outdoor conditions in our modern era. Many countries have deployed their own GNSS constellations such as GPS by the USA, GLONASS by Russia, Galileo by the European Union and BeiDou by China. GNSS receivers made today are often multi-band receivers, capable of using multiple GNSS constellations to calculate position, velocity and heading (PVT) (Pini et al. 2020). Table 2.1 outlines the frequencies used by each constellation for the bands that the receiver module being evaluated operates on.

Table 2.1: GNSS frequencies based on Ublox (2020*a*)

Constellation	**Band**	**Freq (MHz)**
GPS	L1C/A	1575.42
	L2C	1227.6
GLONASS	L1OF	1598.0625-1609.3125
	L2OF	1242.9375-1251.6875
Galileo	E1-B/C	1575.42
	E5b	1207.14
BeiDou	B1l	1561.098
	B2l	1207.14

The most common means of a GNSS receiver to transmit PVT data in a usable form is through the use of the National Marine Electronics Association (NMEA) 0183 interface standard (NMEA-0183). The NMEA-0183 protocol uses simple ASCII sentence strings and can be used on many types of serial interfaces such as I2C, SPI or UART. All NMEA sentence frames begin with a $ and will be followed by a five character address field, a data field, checksum and end sequence (Ublox 2020*b*). The recommended minimum NMEA sentence is RMC, with an example presented below:

$$\$GPRMC, 123456, A, 1234.567, S, 12345.789, W, 123.4, 123.4, 123456, 123.4, E * 6A$$

Where $GPRMC represents a GPS RMC sentence address, with each element in the payload, separated by a comma, represent the UTC, status, latitude, north/south, longitude, east/west, speed, heading, date, magnetic variation and east/west; then ending in the checksum *6A (Trimble 2021*a*).

2.3.1 Real-time kinematic GNSS

Whilst a hand-held GNSS receiver common on mobile phones and other personal devices have accuracies in the ten's of meters, there are methods of improving the accuracy of a GNSS receiver. Differential GPS (DGPS) and RTK are real-time methods to achieve sub-meter and centimetre level accuracies respectably by having base stations transmit correction messages (RTCM) over radio to the GNSS receiver as depicted in figure 2.1 (Perez-Ruiz & Upadhyaya 2012, Zhang et al. 2002).

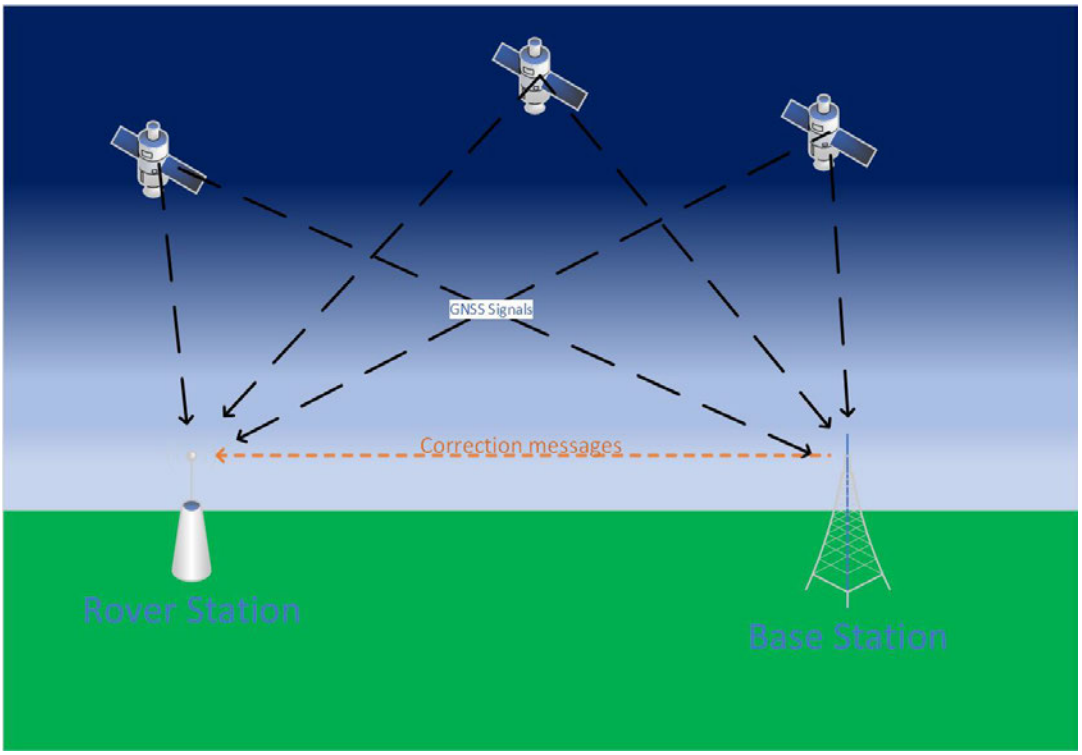


Figure 2.1: Improve GNSS accuracies through differential services

Whilst DGPS public radio transmitters provide reliable coverage over large areas, often for free, the increased accuracies in RTK systems is gained through the requirement of having a base station located within 10 km of the receiver. This generally requires a capital investment by private entities to establish their own RTK network, increasing the cost of a RTK system significantly (Perez-Ruiz & Upadhyaya 2012). With commercial RTK systems costing over $50,000, Knight & Malcolm (2009) found that the benefits to an agricultural business from using such a system was not as profitable as using a less accurate system with reduced benefits. Similarly, Thompson et al. (2019) found cost-saving was the key benefit of precision agriculture as highlighted by producers compared to yield and convenience.

2.3.2 ArduSimple ZED-F9P GNSS receiver

The ArduSimple SimpleRKT2b evaluation boards shown in figure 2.2 is representative of low-cost,multi-band RTK GNSS receivers that are available on the market today. A single receiver is available for purchase in Australia for less than $300 (Mouser 2021*a*) and a long range radio kit complete with two receivers, antennas and UHF radios for around $1000 (Mouser 2021*b*), allowing for a RTK based system to be deployed at a fraction of the cost of commercially developed systems.

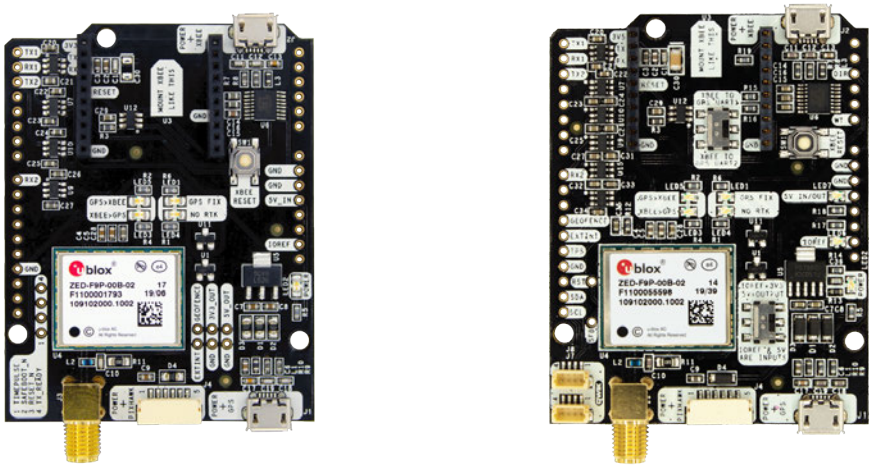


Figure 2.2: SimpleRTK2b V1 (left) and V3 (right) evaluation boards. Source: ArduSimple (2021)

At the heart of the SimpleRTK2b boards is the Ublox ZED-F9P GNSS receiver module. The ZED-F9P module is a multi-band receiver (using GPS, GLONASS, Galileo and BeiDou), capable of 1 cm positional accuracy, heading accuracy as small as 0.3° and a RTK navigational update rate as high as 20 Hz (Ublox 2020*a*). In additional to outputting navigational data in standard NMEA0183 messages (version 4.10), the ZED-F9P module is fully configurable through the use of Ublox's UBX protocol and achieves RTK correction messages through the use of RTCM 3.x protocol (Ublox 2020*b*).

A characteristic of the SimpleRKT2b boards, which simplifies deployment, is the inclusion of Arduino and Xbee form-factor header rails. This allows easy integration with many micro-controller development boards, including STM32, as well as Xbee's range of wireless boards. Often bundled with the ArduSimple kits to allow RTK communication between base station and rover station are the Xbee Pro SX long range radio, capable of transmitting up to 105 km (line of site) at data rates up to 250 kb/s (Digi 2020).

2.3.3 Evaluating agriculture localisation systems

The ZED-F9P receiver module has previously been evaluated in studies for positional accuracy. Hamza et al. (2020) evaluated positional quality and identified the range of displacements that could be detected. Whilst their study was looking at comparing a low-cost RTK system with commercial receivers, it was focused on geodetic use, and as such, focused on static tests of the ZED-F9P, where the only dynamic testing was carried out over a 15 cm test distance. Translation of the accuracy data for dynamic farming practices is limited, as the testing did not simulate the real-time dynamic requirements of an agricultural vehicle.

Prior to 2010, there was a lack of standards for evaluating dynamic systems that relied upon GNSS for localisation (ISO 2010), with many GNSS systems evaluated under static conditions which did not reflect how the systems were being used (Cole et al. 2004). This led to the International Standards Organisation (ISO) setting out a framework for evaluating and reporting dynamic systems in ISO 12188.

Part 1 of ISO 12188 sets out two types of tests to be conducted: a horizontal positioning test and a dynamic signal reacquisition test. Both test require the GNSS system under evaluation to be run a travel course, where the precise location of the tested system can be measured at one order of magnitude more accurate than the system itself. The travel course is to include two straight segments of 90 meters and a U-turn segment with a radius of between 5-10 meters (ISO 2010). In developing the standard, Stombaugh et al. (2008) used a motorised cart system on a I-beam track to evaluated a sub-meter class GNSS receiver and a low-cost receiver with 2-5 meter accuracy simultaneously. However, since this publication, there has been included in the standard, that there shall be no metallic objects within 50 meters of the course due to multipath interference (ISO 2010), which may preclude the use of an I-beam track to evaluate the ZED-F9P in this project.

The horizontal position test involves conducting 24 test runs of one hour duration over a 25 hour period and at speeds ranging from 0.1 to 5.0 m/s. During each test run, the recorded positions from the GNSS system on the straight segments are converted to localised Cartesian coordinates and compared to the recorded travel course positioning in order to report positioning accuracies (ISO 2010).

The dynamic signal reacquisition test involves blocking the satellite signal to the receiver during the U-turn segment of the travel course and recording the elapsed time until valid position data is transmitted once the satellite signal is unblocked. Three one-hour dynamic signal reacquisition tests are to be conducted, with a three hour pause between tests, within a total of 13 hours (ISO 2010).

2.3.4 The UBX message protocol

Ublox use their own messaging protocol, called UBX, which is a variable length payload frame protocol protected by a 8 bit Fletcher checksum. Any UBX packet (illustrated in figure 2.3) begins with a two byte preamble of 0xb5, 0x62 (ISO 8859-1 for µb) followed by four bytes identifying the message class, ID and payload length. The payload will consist of a number of 8-32 bit signed and unsigned integers or flag bit fields which represent data fields and are decoded based on their placement in the payload. Depicted in figure 2.3 are the fields of a NAV(class)-PVT(ID) message, which has a payload length of 92 bytes, where the first four bytes of the payload represent a 32 bit unsigned integer for the GPS time of week. Latitude and logitude are represented by 32 bit signed integers and there are other bytes within the payload where individual bits represent flags within the message (Ublox 2020*b*).

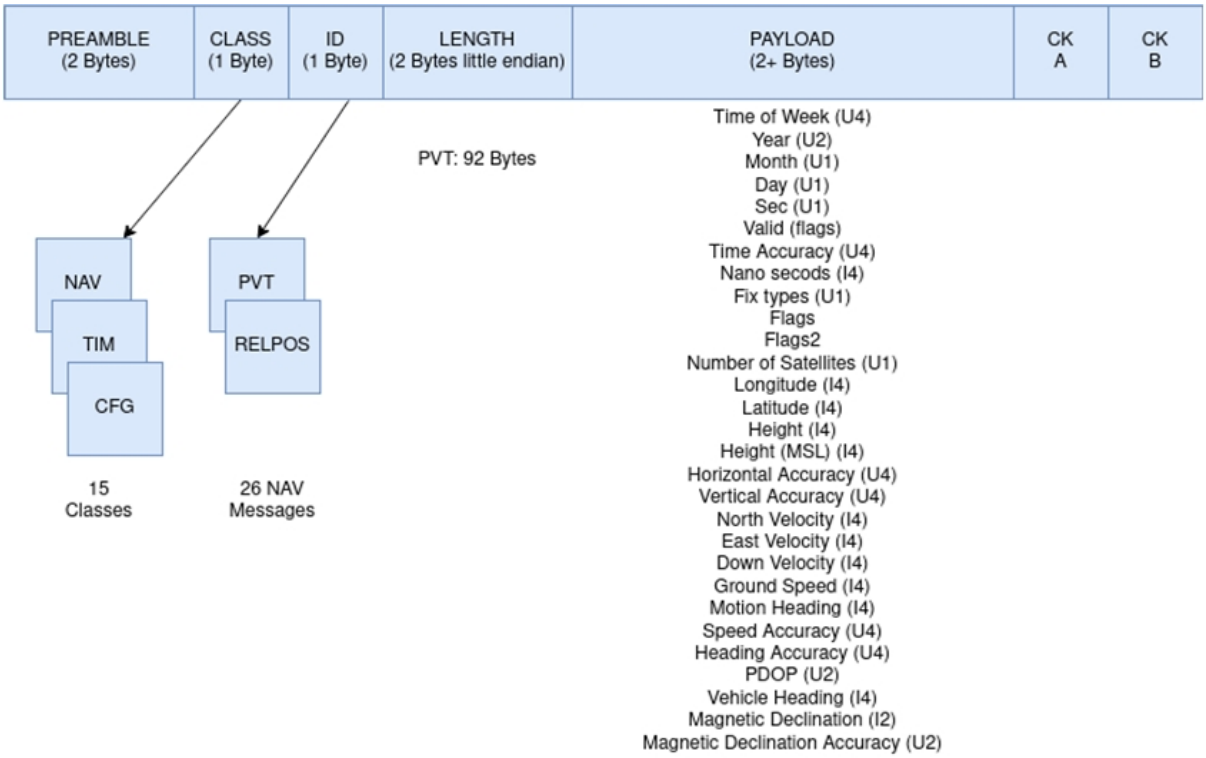


Figure 2.3: The UBX message frame (NAV PVT fields shown as example)

For this project, only a limited number of message types were used. Within the navigation class (NAV), the position velocity time (PVT) message provided all critical GNSS data for later analysis, whilst heading to the moving base was provided by the relative position (RELPOS) message. In addition. the TIMUTC message was used to get the UTC time solution in order to update the real time clock (RTC) and some config (CFG) messages were used to program the GNSS receiver during operation.

Chapter 3

Design Methodology

3.1 Chapter Overview

The evaluation method developed during this project is adapted from ISO 12188-1:2010, which outlines what tests should be conducted and how to generally calculate accuracy of satellite based localisation systems. This chapter will outline how the testing methodology was developed from the standards and the development of the concept system design to be assessed.

3.2 Project Methodology

This project used an simulated research design to test an RTK GNSS system under controlled real-world conditions to predict accuracy. To achieve the stated aims and objectives, this required the development of:

- A RTK GNSS system to provide localisation data to a robotics system.
- A testing procedure to ensure validity of the produced system.
- A method of analysing the data and report on the accuracy of the system.

3.2.1 Proposed testing methodology

To verify the accuracy of the designed system, a testing procedure based on the framework outlined in ISO 12188-1 but within the limitations outlined has been developed. A simple test track consisting of two straight segments 6.4 meters long and two u-turns with a nominal 5

meters diameter as shown in figure 3.1 was be used to assess the PVT accuracy of the ZED-F9P receiver module.

Figure 3.1: Test track layout

The track was made out of acrylic so as not to induce GNSS signal interference and incorporate a gear rack profile in order to provide a means of motorisation and accurate reference position data. Recording of the actual path traversed by the rover receiver was to be achieved by recording the distance through the use of a rotary encoder (CUI AMT-102) on a pinion which recorded the traverse distance along the outer track's rack. A photo-interrupter switch was originally designed to record the passing of the reference points A-D n figure 3.1 to indicate the start and end of each straight run of the travel course, but this was not able to be implemented on the day of testing.

The data from the GNSS rover reciever was converted to localised linear data points using the conversion factors calculated from equations 3.1 and 3.2, where a and b were the WGS84 semi-major and semi-minor axis of the ellipsoid, ϕ was the test site latitude and h was the average height above the ellipsoid (ISO 2010).

$$F_{lon} = \frac{\pi}{180^\circ}\left(\frac{a^2}{\sqrt{a^2 cos^2\phi + b^2 sin^2\phi}} + h\right) cos\phi \tag{3.1}$$

$$F_{lat} == \frac{\pi}{180^\circ}\left(\frac{a^2 b^2}{(a^2 cos^2\phi + b^2 sin^2\phi)^{\frac{3}{2}}} + h\right) \tag{3.2}$$

This converted the GNSS data into local Cartesian coordinates measured in meters, for which errors and ultimately accuracy could be estimated. Without a surveyed reference track, relative cross track error was calculated by comparing the data recorded for each pass of the GNSS receiver through the straight segment, calculating a theoretical reference line and measuring

the variation in position data for each pass similar to the methods used by Rounsaville et al. (2016).

3.2.2 Method

The scheduling for each phase is detailed in section B.1.

3.2.3 Limitations

Resources in the form of manufacturing and time were constrained to implement the full scale test procedures outlined in ISO 12188. In developing the standards, both Cole et al. (2004) and Stombaugh et al. (2008) were evaluating GNSS receivers with accuracies of several meters or sub-meter capability, whilst this project aims to evaluate a centimetre level receiver. Therefore, it is considered reasonable that the travel course can be reduced in size to allow for the limitations yet still produce significant data. However, the reduction of the track straight segment to 6.4 meters was well below an originally planned 20 meters, which limited the amount of data available on the straight segments in order to calculate cross track error. Manufacturing times combined with poor weather on the day of testing also prevented multiple test runs at different speeds as originally planned. In the end, only a single test run was able to be conducted where data was recorded, at a speed of 0.63 m/s. Further tests will be required at longer track lengths, varying speeds and longer test times in order to gain more beneficial data to be analysed in the future.

The four reference points marked A-D were originally planned to be surveyed, to provide the reference start and end mark for each straight segment (A-B and C-D) over which the horizontal position errors were to be calculated in combination with the encoder data using the outside track as a reference track. This would have allowed calculating an absolute accuracy of the system, rather than a relative cross track error.

Additional tests were also planned to be conducted with increased baseline distance between the base station and rover station to evaluate the effectiveness of the Xbee radio to maintain centimetre accuracy of the rover station of ranges up to 10 km.

3.3 System concept design

In order to evaluate the ability of the SimpleRTK2b evaluation board to provide accurate positional data for the requirements of an agricultural robotic system, the system to be designed will consist of two major sub-systems: a static base station and a rover station.

3.3.1 Static base station

The static base station's primary purpose is to provide differential GNSS service messages (RTCM 3.2) to the rover station via an Xbee Pro SX long range radio (using the industrial, scientific and medical (ISM) frequency band, 918-926 MHz, (*Australian Radiofrequency Spectrum Plan 2017 (Cwlth)* 2017)).

The base station will be able to operate in two modes of operation, survey-in mode and statically defined mode. In survey-in mode, the base station can be placed in any location prior to switching on, then when it is turn on, the base station will survey itself in with a minimum specified time and accuracy. In the statically defined mode, the base station will be positioned over a pre surveyed marker, with the positional data programmed into the ZED-F9P receiver.

The base station is to be powered by either battery or external DC source in order to allow flexibility in use in either a fixed position or in a field location. An LCD will display the status of the base station for quick reference to ensure it is operational during the evaluation and should display the status of the ZED-F9P receiver, battery and RTCM output. A microcontroller shall provide configuration of the ZED-F9P receiver, monitor power input and drive the LCD display.

A prototype of the base station is presented in figures 3.2 and 3.3, consisting of a SimpleRKT2b ver1 board, Xbee Pro SX radio, Arduino Uno microcontroller, LCD screen with I2C driver and powered by a 5 V power bank.

3.3.2 Rover station

The rover station's primary purpose is to provide positional data to an integrated agricultural robotic system.

The rover station will make use of two ZED-F9P modules so as to provide both PVT data, but also accurate heading. It will receive RTCM messages from the base station in order to provide accurate, real time positional and heading data.

3.4 System design requirements

The design requirements of the system have been broken down into two tables, one for the base station subsystem (table 3.1) and one for the rover station subsystem (table 3.2). Each requirement has been described as either essential or desirable for the purpose of the project, where desirable requirements are requirements which support future development of an agricultural robotics system and essential requirements are ones that must be met to evaluate the SimpleRKT2b board. Requirements highlighted in bold form the major criteria for evaluation

Figure 3.2: Base station prototype

Figure 3.3: Base station deployed

of the system as a low-cost RTK GNSS system.

Table 3.1: Static base station design requirements

Serial	Requirement	E/D
General requirements		
BS-G-01	The system shall use a ZED-F9P configured as a RTK base station.	E
BS-G-02	The system will transmit RTCM messages to one or more rover stations using Xbee Prox SX radio in an agricultural setting to a range of 20 km (to be evaluated).	E
BS-G-03	The system transmits RTCM 1004, 1005 messages, and relative MSM4 or MSM7 messages for GPS, GLONASS and Galileo constellations.	E
BS-G-04	The system is able to be configured to conduct a self-survey-in when a pre-surveyed position is not available.	D
Electrical requirements		
BS-E-01	The system is powered by either an external DC voltage or battery source.	E
BS-E-02	Battery system shall be designed to provide 25 hours of continuous operation. To ensure continuous RTCM correction data during testing (ISO12188). Some short intervals allow for battery change.	D
BS-E-03	The system shall have 5 V regulation for powering subsystems.	E
Hardware requirements		
BS-H-01	The system hardware is protected by a water resistant enclosure with external power and communications ports.	E
BS-H-02	The antenna system is mountable on a suitable tripod system for field use or has the ability to be placed in a fixed position.	E
Software requirements		
BS-S-01	The ZED-F9P is able to be flashed with configuration file in the field. Bare minimum to achieve requirements	E
BS-S-02	User can configure ZED-F9P with the MCU using UBX messages over a serial connection. Can be accomplished with either integrated button system or external connection.	D
BS-S-03	MCU monitors battery status and provides indication to user. Either through LCD or LED indicators or external communication protocol.	E
BS-S-04	MCU monitors ZED-F9P to ensure base station functionality and provide indication to user. Either through LCD or LED indicators or external communication protocol.	E

3.5 System preliminary design

Figure 3.4 below outlines the major subsystems of both the static base station and the rover station. It outlines the communications connections, protocols and speeds that will be used and the major message types and their direction in order to achieve centimetre level position accuracy. Wiring diagrams for the base station and rover station design are detailed in Appendix C.

Table 3.2: Rover station design requirements

Serial	Requirement	E/D
General requirements		
RS-G-01	The system shall use a ZED-F9P receiver configured as a moving base station to provide positional data.	E
RS-G-02	The system shall use a second ZED-F9P receiver configured as a heading module to provide relative position data to the moving base station.	E
RS-G-03	The system will receive RTCM messages from the static base station Xbee Pro SX radio in an agricultural setting at a range of 20 km.	E
RS-G-04	The system will provide 2 cm accuracy in position data (to be evaluated).	D
RS-G-05	The system will provide 0.4° accuracy in heading data (to be evaluated).	D
RS-G-06	The system has organic data logging capability.	E
Hardware requirements		
RS-E-01	The system is powered by an external 5 VDC regulated source.	E
RS-E-02	The heading ZED-F9P receiver and associated antenna and connectivity physically separated from remainder of the system with wireless connection to moving base for relative position and RTCM messages.	D
Hardware requirements		
RS-H-01	The system hardware is protected by a water-resistant enclosure with Eexternal power and communications ports.	E
RS-H-02	The GNSS antenna system will be designed to have a minimum of one meter separation between moving base and heading antennas, in either an in-line configuration or perpendicular to travel configuration.	E
RS-H-03	Each GNSS antenna shall have a ground plane.	E
Software requirements		
RS-S-01	The ZED-F9P is able to be flashed with configuration file in the field. Bare minimum to achieve requirements.	E
RS-S-02	User can configure ZED-F9P with the MCU using UBX messages over a serial connection from an external MPU.	D
RS-S-03	NMEA messages are parsed to provide date, time, position, elevation, geoidal separation, course over ground, number of visible satellites, correction status and satellite constellation configuration.	E
RS-S-04	Positional data from requirement RS-S-03 is stored on SD card through internal data logger.	E
RS-S-05	The MCU is designed as a ROS2 node providing positional data as a service.	D
RS-S-06	Requirement RS-S-03 is achieved with UBX messages in addition to NMEA messages.	D
RS-S-07	User can select mode of GNSS heading antenna configuration (in-line or perpendicular).	E
RS-S-08	User can configure position reference point for the overall robotics system.	D
RS-S-09	The system is able to provide positional data update at a rate of up to 10 Hz.	D

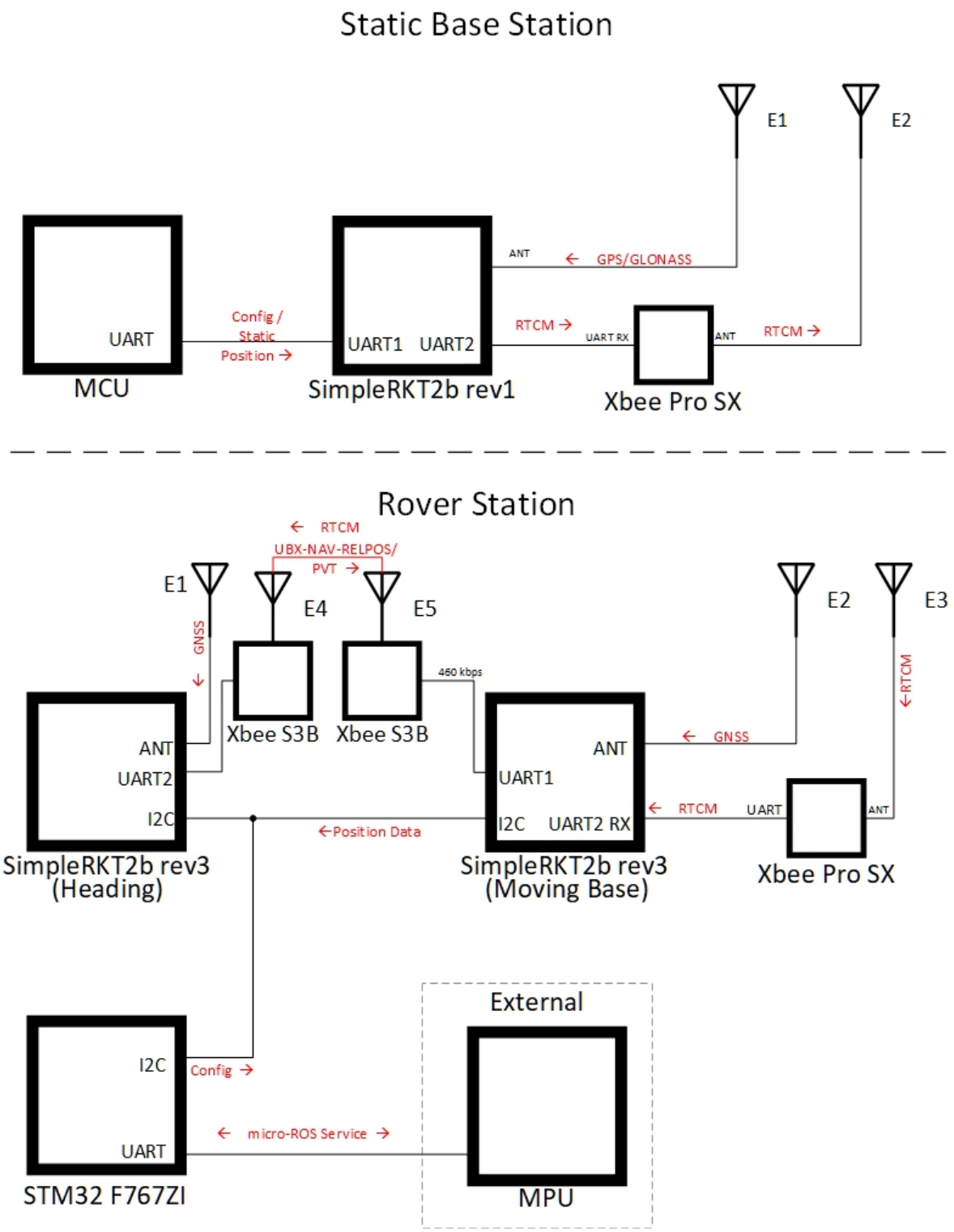


Figure 3.4: High level system design

Chapter 4

Hardware Implementation

4.1 Chapter Overview

This chapter will outline the hardware development that occurred during the project to develop both a testing rig to conduct an evaluation and the system to be evaluated.

4.2 The Test Rig

In order to evaluate the RTK GNSS rover system's accuracy, a test rig was designed to provide a reference track and drive system, so continuous repeatable test runs could be conducted. The system illustrated in figure 4.1 was designed where an acrylic track (in order to reduce RF interference) with a gear profile provides an accurate means of measuring the systems distance around the track.

The track itself was designed to be a modular sandwich construction, with the 6 mm thick rack pinned between two 3 mm thick guide rails, providing a grove for the v-slot wheels of the rolling elements of the drive mechanism. An exploded view of a straight section of track is shown in figure 4.2. For modularity, section lengths were determined to allow cutting of the gear tooth profile on either end to be symmetrical, resulting in straight sections being 402 mm long each, an outer track pitch diameter of 5.4 m and an inner track pitch diameter of 4.68 m (this allowed each curved section to span 10° of the curved segment). The completed track layout for the duration of the test is shown in figures 4.3 - 4.4.

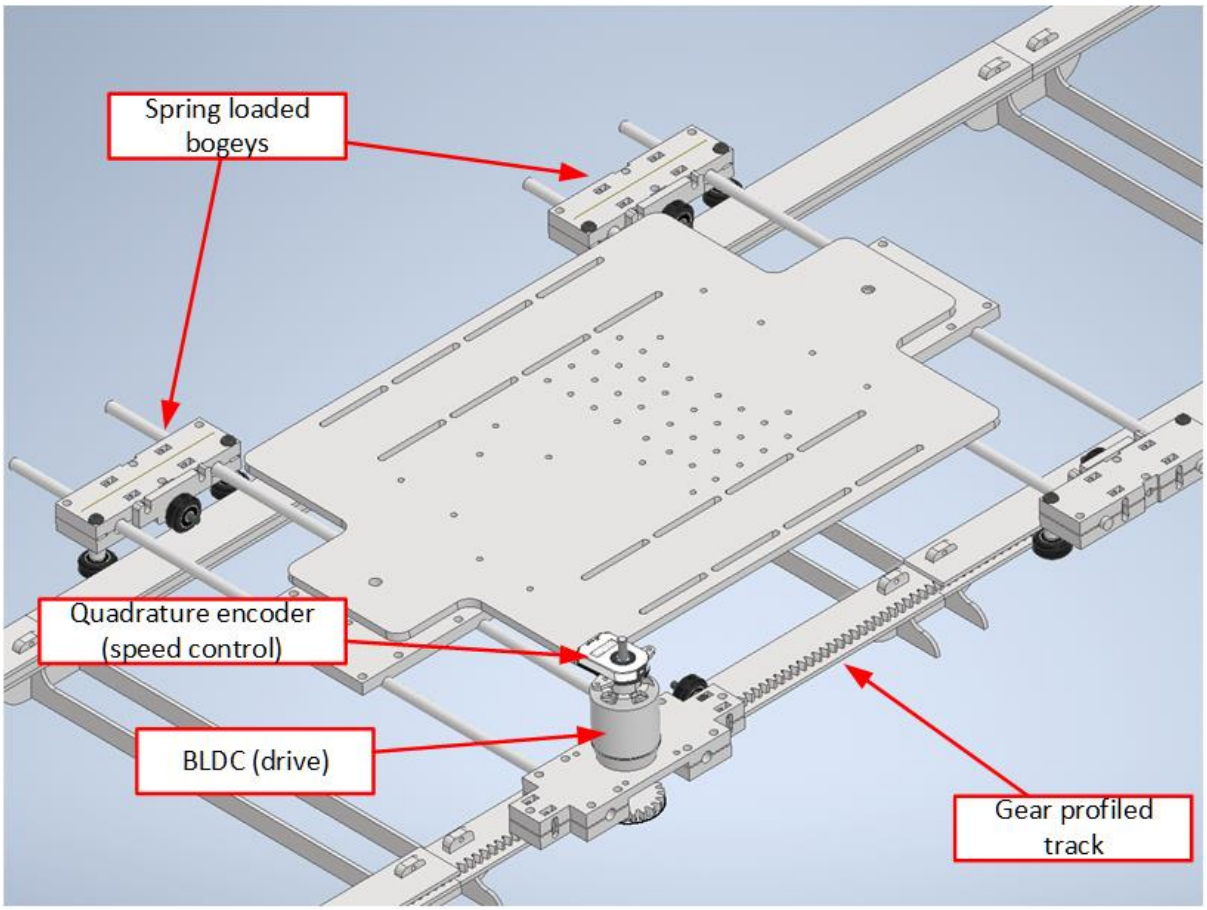


Figure 4.1: The test rig concept design.

4.2.1 The Drive Mechanism

In order to provide continuous controlled movement around the track, a brushless DC (BLDC) motor was selected to provide direct drive of a pinion, using the inbuilt gear profile of the track to provide locomotion. An open source motor controller board, the ODrive 3.6, was selected for speed control of the BLDC motor, which provided multiple motor control modes, quadrature encoder integration for closed-loop control and multiple communication interfaces (ODrive 2021).

To maintain traction, spring loaded 3D printed bogies with v-slot wheels clamped the drive mechanism to the track, with the springs mounted on the inner track side in order to keep the outer, reference track side at a fixed distance from the track. The final test rig drive system is shown in figure 4.6.

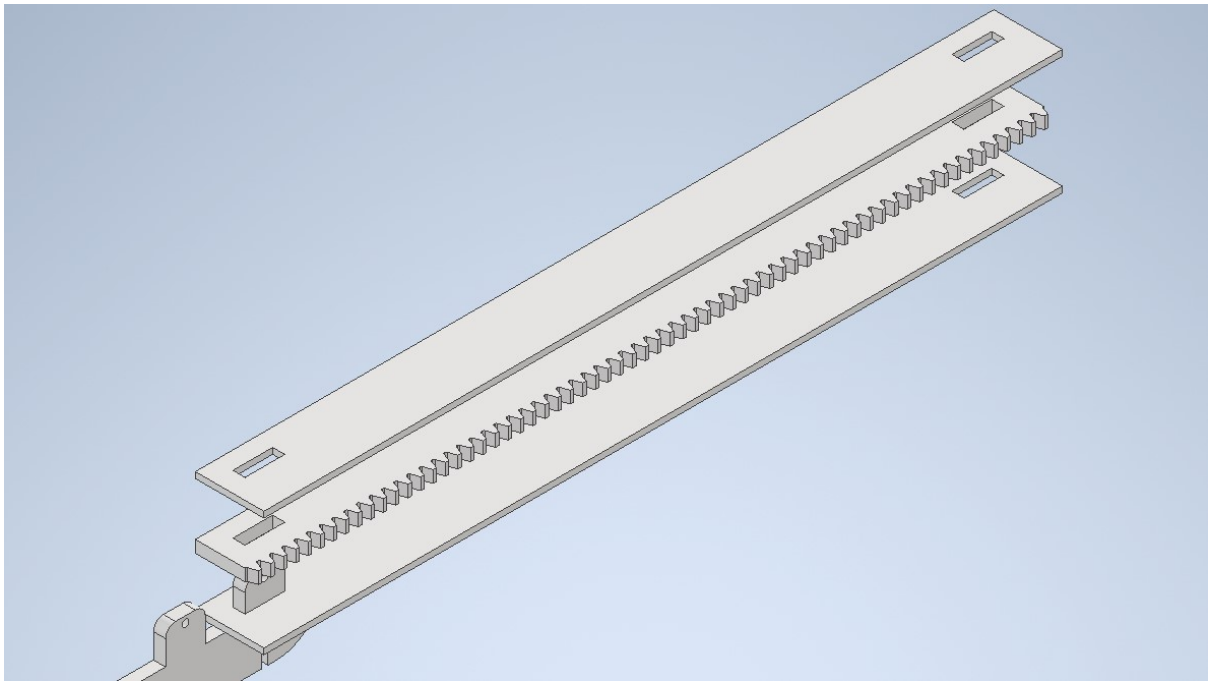


Figure 4.2: An exploded view of a straight track section.

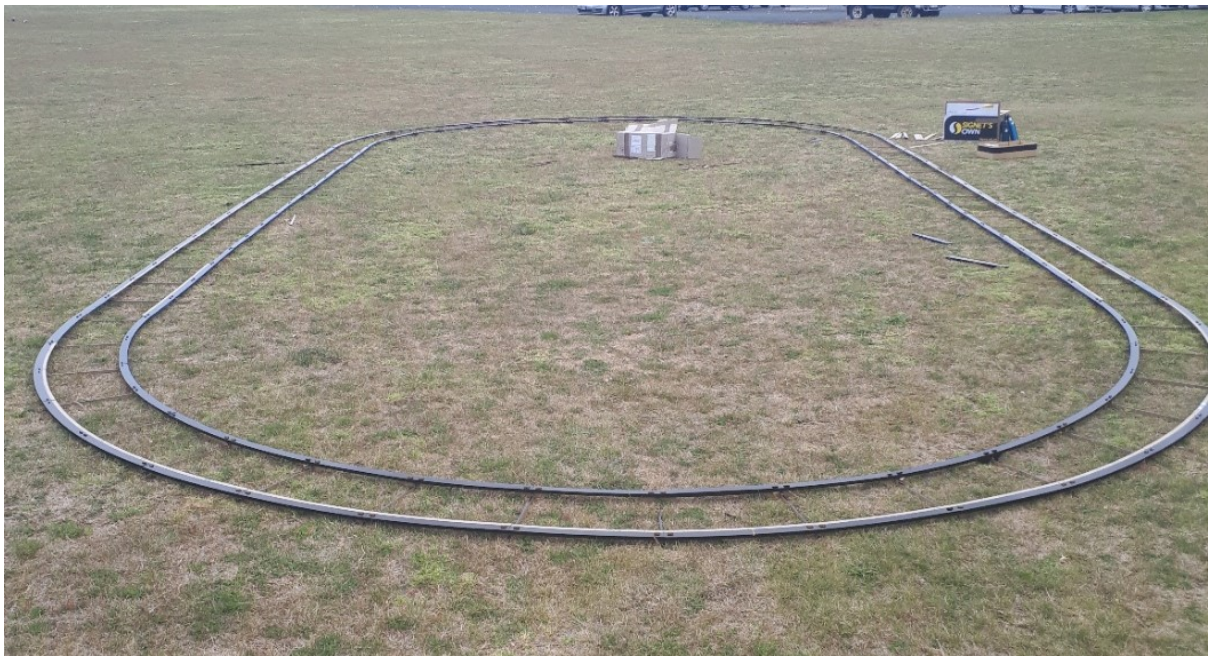


Figure 4.3: A view of the completed track.

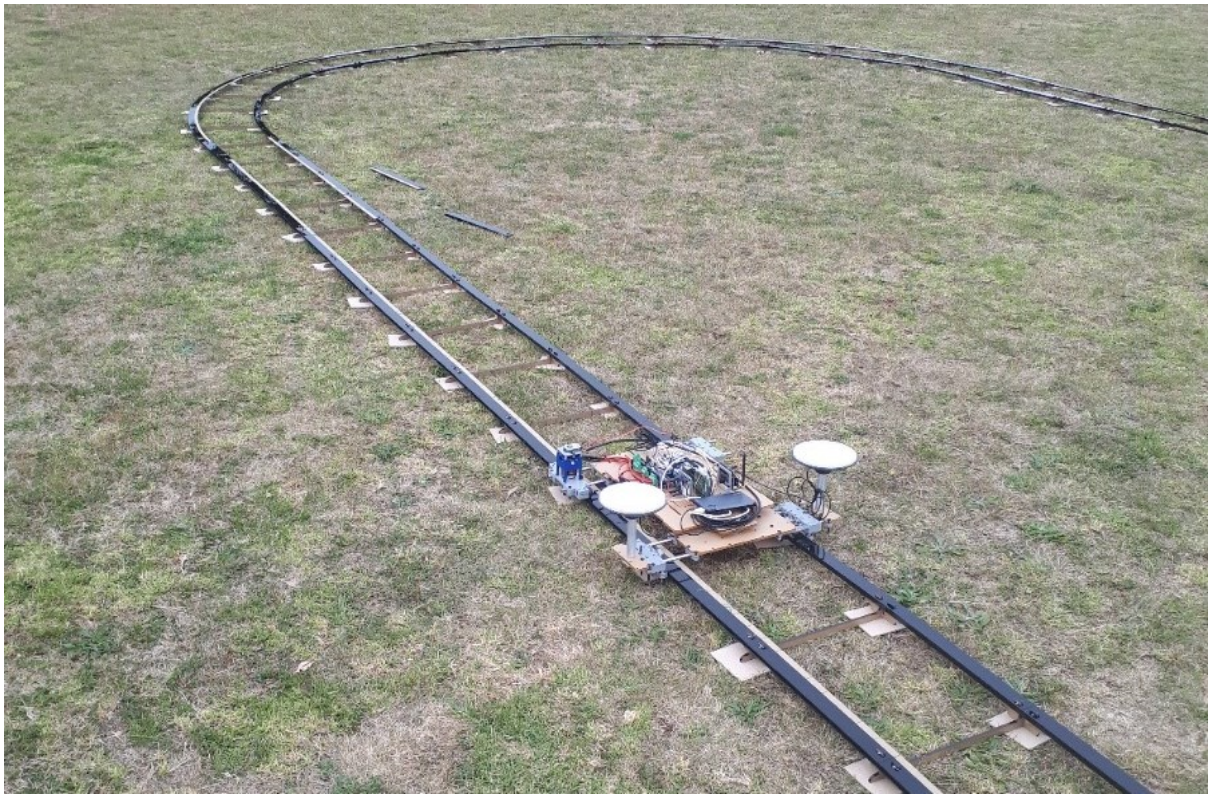


Figure 4.4: A view of the completed drive system on the track.

4.2.2 Power

Power to the BLDC motor was provided by a 3S LiPo battery (11.1-12.6 V, 3.3 Ah), whilst a separate 5 V, 10 Ah power bank was used to provide power to the remainder of the electronics.

4.3 The Microcontroller

The ST Nucleo-F767ZI microcontroller (MCU) development board was selected to provide a real-time system to log GNSS data and to log distance around the track for reference when analysing the data. The STM32F767ZI can operate up to 216 MHz frequency, has inbuilt hardware peripherals such as timers (with encoder mode support), RTC, hardware interrupts and universal asynchronous receiver-transmitter (UART) which has been used in this project (STMicroelectronics 2021). In addition, as a future integration of the RTK GNSS system for use as a localisation subsystem of an autonomous agricultural robot, the STM32F767ZI is also supported by Micro-ROS, a ROS2 implementation for microcontrollers.

4.3.1 Peripherals

Timers

A second quadrature encoder was mounted on the axel of the BLDC motor with the output connected to a 32-bit hardware timer on the MCU configured in encoder mode. This allowed the distance around the reference course to be tracked by the MCU without using processor resources, with the count value retrieved by the data logging software task when it was required.

UART

Two UART serial interfaces were used on the MCU, both configured for 115200 bps and using receive interrupts. UART2 was connected to the rover GNSS receiver for transmitting configuration messages and receiving GNSS data. UART3, which is converted to a USB virtual com port by the Nucleo development boards in-built ST-Link, was connected to the Raspberry Pi for the transmission of data to be logged and receiving user input.

4.3.2 GPIO interrupt

In addition to the dedicated hardware peripherals, a general purpose input/output (GPIO) interrupt was configured and connected to a photo interrupter, which detected when the test rig entered and exit straight segments of the reference course through the use of slots cut into the track (see figure 4.5). When the interrupt was triggered, the encoder timer count would be reset and the track segment variable incremented or reset after a complete loop through the reference course.

4.4 Raspberry Pi

A Raspberry Pi single board computer (SBC) was used to provide wireless control of the entire system and to log the data to file. Control and monitoring of the ODrive motor controller was achieved through the use of a python program specifically designed for the ODrive when connected by USB. As the MCU constantly transmitted data logging information when there was GNSS data available, a python script was executed on the Raspberry Pi to write all data received from the MCU to a log file when a test serial was occurring. Figure 4.6 shows the final test rig with all components connected ready for testing.

Figure 4.5: The cut slot at the end of straight track segment.

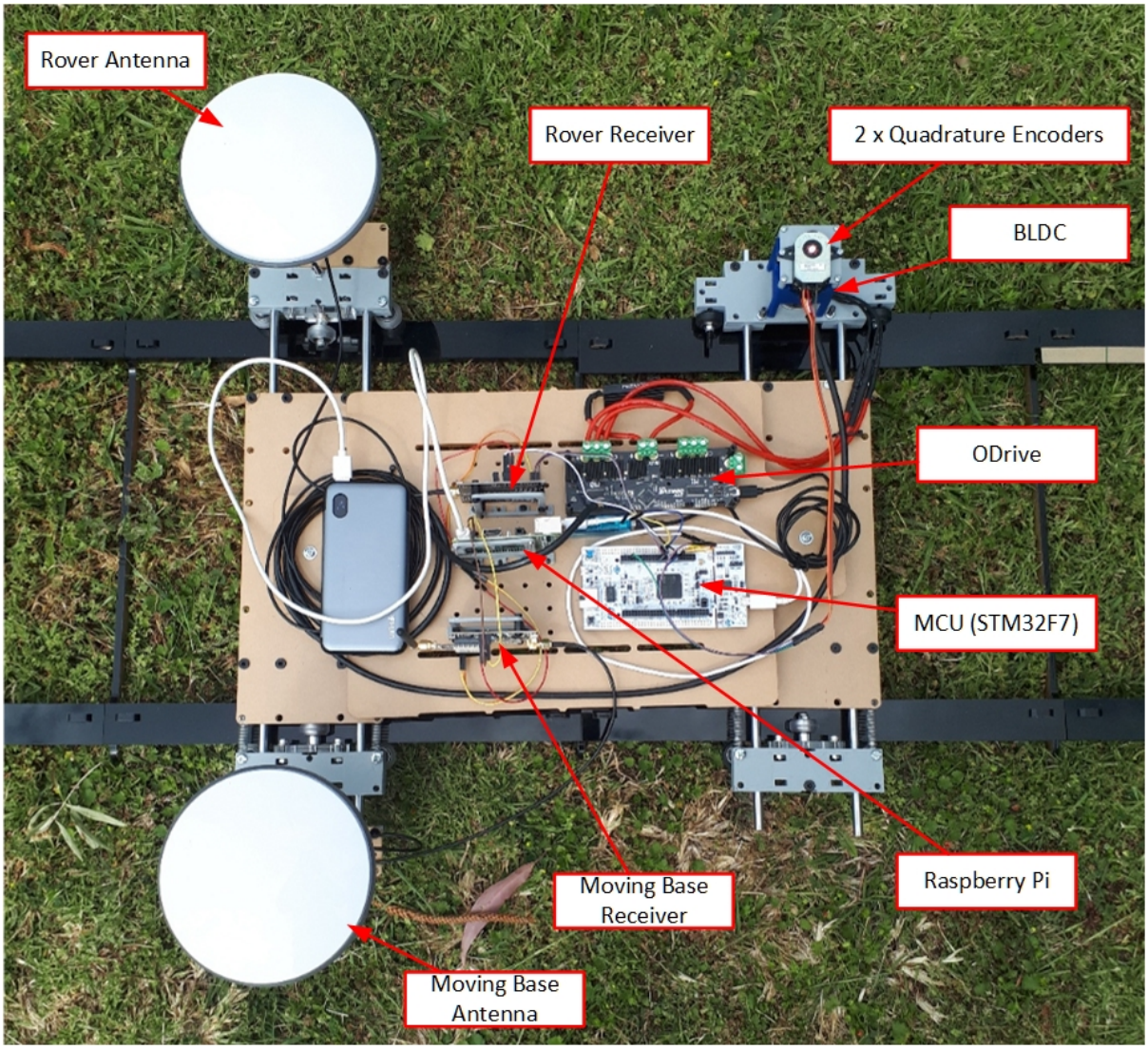


Figure 4.6: The final test rig.

Chapter 5

Software Development

5.1 Chapter Overview

This chapter will outline the design of the MCU, which used the FreeRTOS kernal to manage a number of tasks to accept user input, process the UBX messages, log all data, handle interrupts and update the RTC. Whilst this chapter will focus on the real time operating system (RTOS) tasks and key functions, the full source code listings used during data capture is provided in Appendix D.

5.2 RTOS Tasks

An open source RTOS called FreeRTOS was implemented on the MCU to execute independent tasks which handle user input, processing UBX messages and logging data in a timely and deterministic manner. FreeRTOS provides inbuilt functions such as queues and semaphores for intertask communication and to maintain the ability for multitasking to occur (Amazon Web Services n.d.). The use of FreeRTOS allows each task to be written as it's own program, with the primary tasks used in this project described below.

5.2.1 Processing UBX messages

Due to the variable length of UBX messages, each byte had to processed, as it was received, to determine message type and length. Whilst the MCU UART hardware handled the reception of bytes from the GNSS received, a FreeRTOS queue was implemented to store bytes for the UBX processing task. This task, depicted in figure 5.1, would only be scheduled if there was data in the queue to be processed, where each byte would be stored in a temporary buffer until the

message type and length could be determined and separate buffers created to store the message data. Once a complete message had been received and the checksum verified, if the message was a PVT message, a semaphore would be raised to enable the data logging task, which has a higher priority, to be immediately scheduled in order to capture the PVT data in a timely manner.

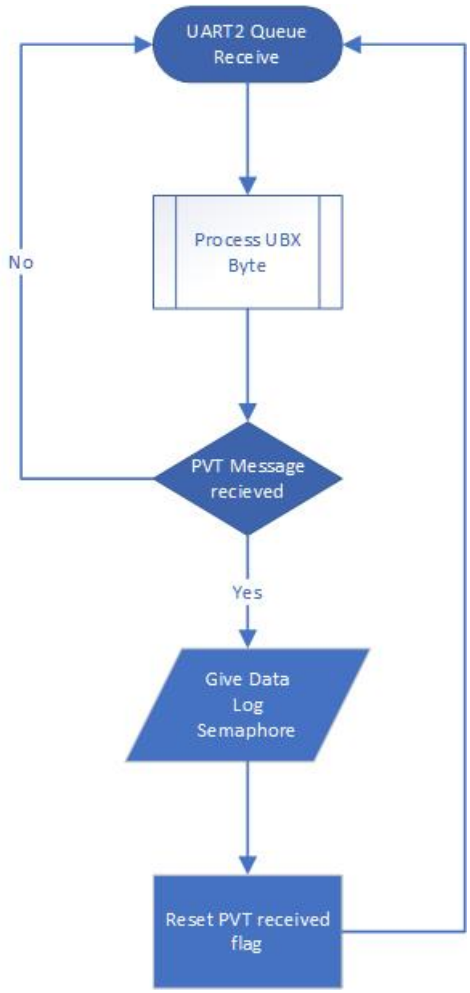


Figure 5.1: The UBX processing task flow chart

The use of the FreeRTOS queue allows the the UBX processing task to be suspended, to allow the data logging task to be executed, without dropping packets. This is because the hardware UART continues to store each subsequent byte within the queue and the UBX processing task is then able to continue once the data logging task has completed. This is illustrated in figure 5.2, where Segger Systemview was used to analyse the ability of the MCU to process the data logging task in time and for the queue system to maintain UBX processing.

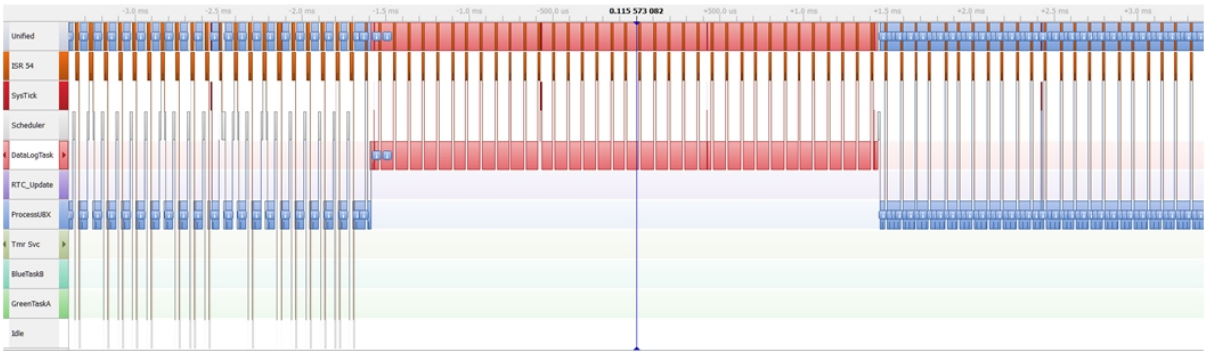


Figure 5.2: Segger Systemview of the data logging task with priority over the UBX processing task

5.2.2 Data Logging Task

In order to construct a data log line with all relevant information captured in a timely manner, the data log task was given high priority, but would only be scheduled when a semaphore was raised in the receiving of a PVT message. On execution of this task, the encoder timer count variable (which measures distance around the track), segment variable (recording which segment is currently being traversed) and the heading data were captured and combined the PVT data to create a message to be transmitted to the Raspberry Pi for actual logging (as shown in figure 5.3)

5.2.3 RTC update task

The RTC update (depeicted in figure 5.4) task executes on system startup or when a user sends a command to execute through dynamic creation and deletion. When this task is created, it will transmit a UBX message over UART to the GNSS receiver to send back a UTC time message in order to update the internal RTC of the MCU. This task will then wait for a semaphore to be raised to indicate that the UTC time message has been received, at which point, the MCU RTC will be updated with the current calendar and time data and then the task will delete itself to stop running.

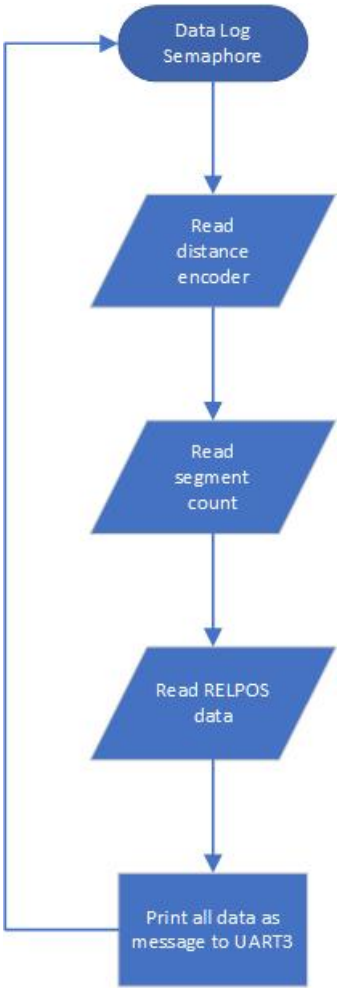


Figure 5.3: The data logging task flow chart

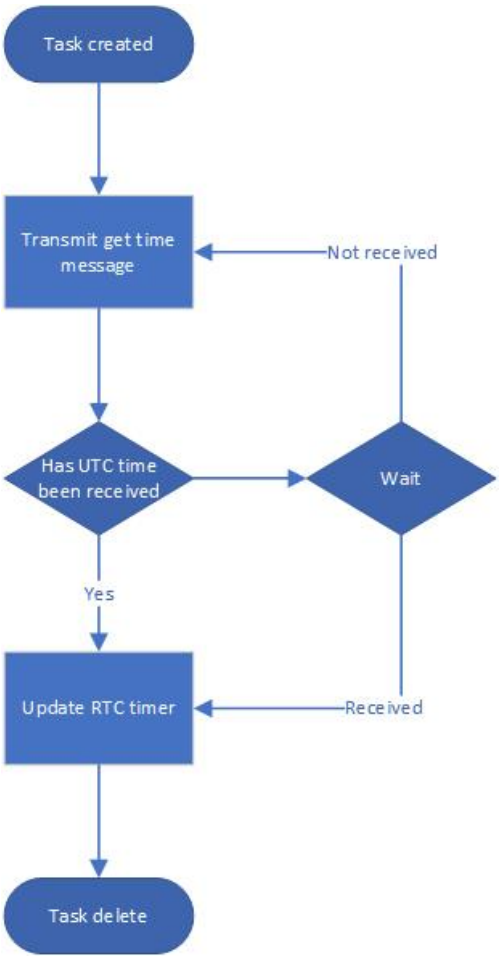


Figure 5.4: The RTC update task flow chart

Chapter 6

Results

6.1 Results

6.1.1 Collection of data

A total of 17.6 minutes of data was collected during the successful test run around the travel course, resulting in the capture of 5280 data points for the UTC time, distance encoder value, heading, latitude, longitude, height, number of satellites in view and the fix type. This data was processed in MATLAB, with the code listing provided in appendix E. Initially, the latitude and longitude data has been plotted (figure 6.1).

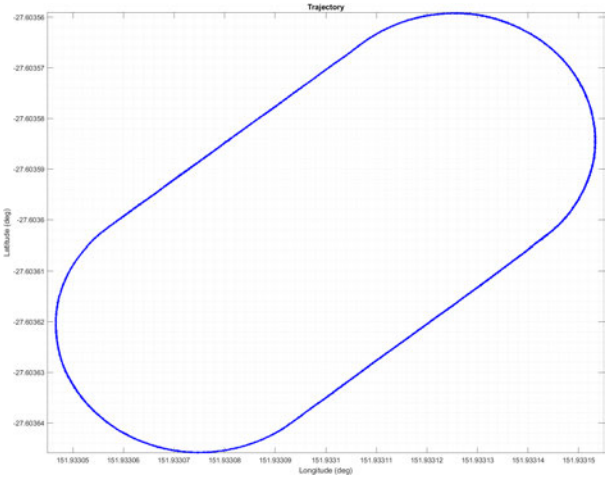


Figure 6.1: Trajectory data points.

The latitude and longitude coordinates were projected into a localised coordinate system so that error and therefore accuracy could be calculated using the method outlined in ISO 12188-1:2010 and section 3.2.1. This adjusted trajectory is shown in figure 6.2.

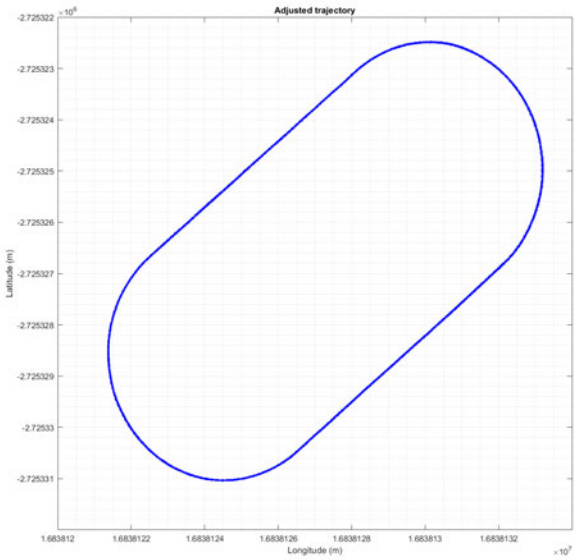
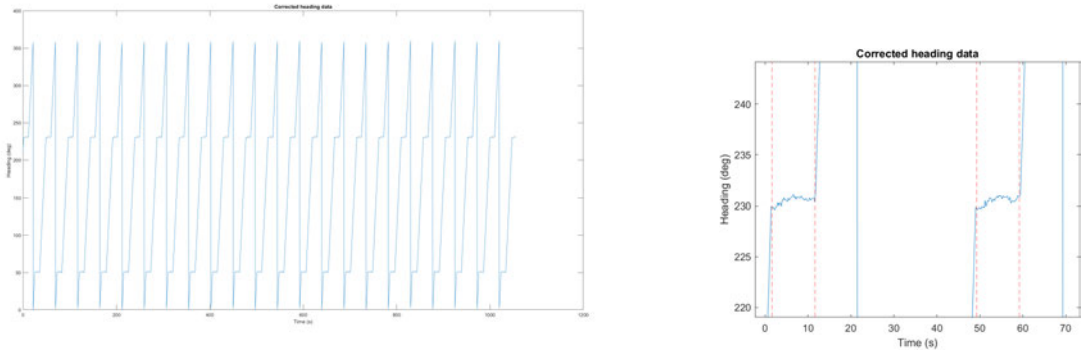


Figure 6.2: Adjusted trajectory.

In order to determine relative cross track error in a pass-to-pass manner, the heading data was plotted (figure 6.3a) to determine when the test trolly was on the straight segments. The start and end of the straight segments were determined from this plot by examining where the heading levelled out at 50° and 230° (figure 6.3b). This resulted in a total of 22 passes in each of the southwest and the northeast direction in which to analyse.



(a) Corrected heading

(b) Sample selection windows

Figure 6.3: Heading data plots

6.1.2 Relative Cross-track Accuracy

Adjusted trajectory points from the two straight segments were grouped together (1126 points in the southwest segment and 1123 points in the northeast segment). For each segment, a linear regression was conducted to determine a line of best fit which would be used as a reference line to calculated relative cross-track error (figures 6.4 - 6.6).

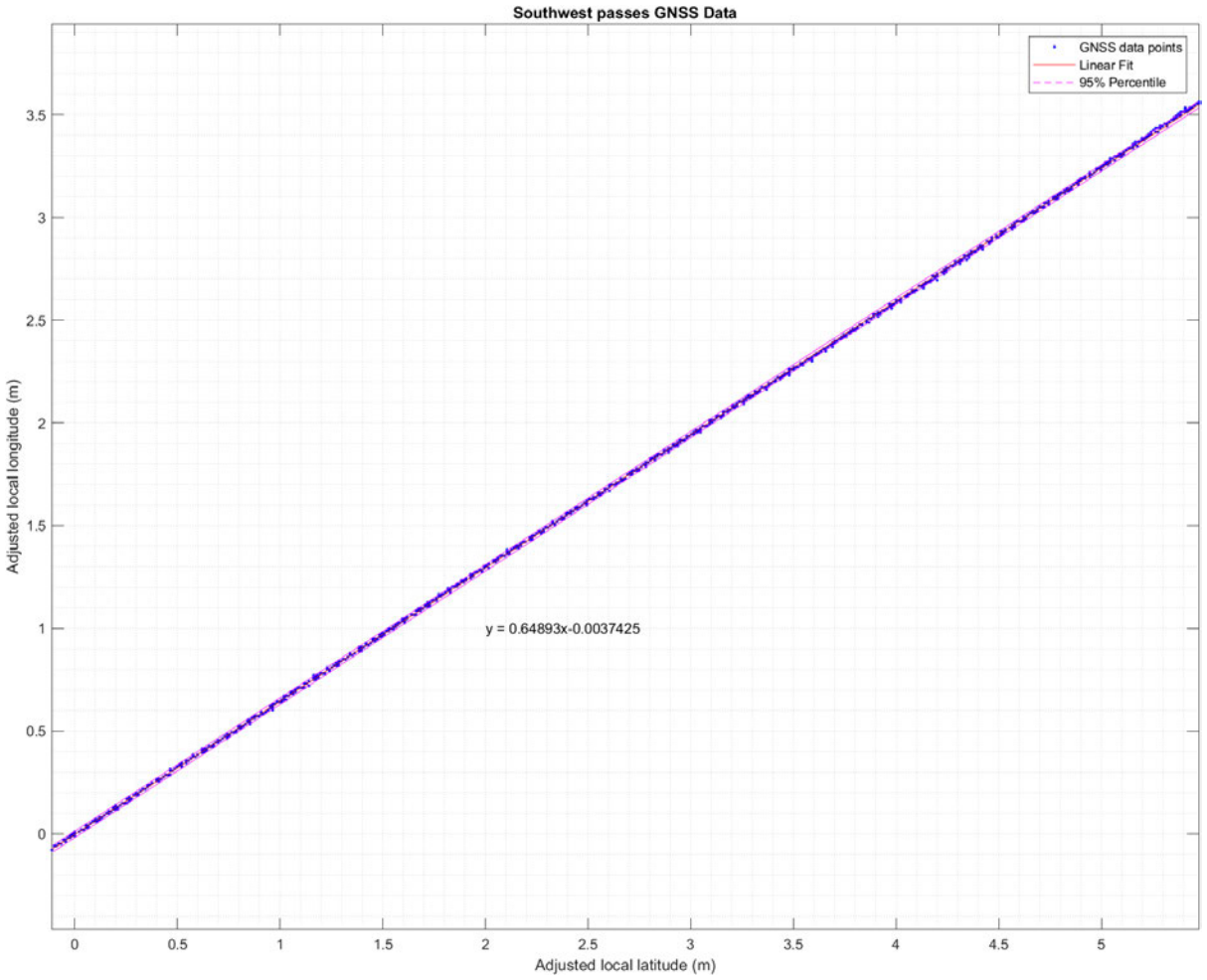


Figure 6.4: Adjusted localised GNSS data of southwest segment

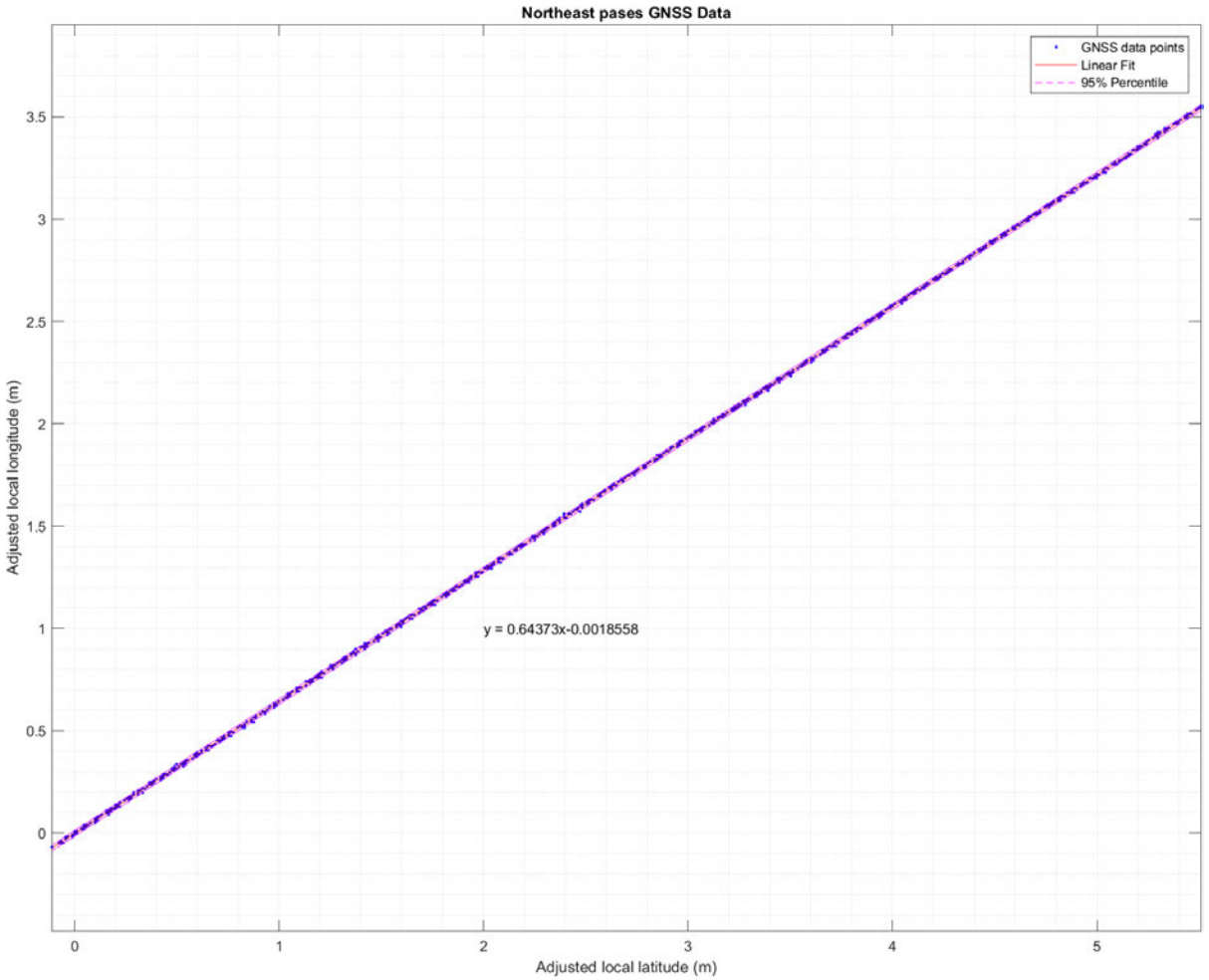


Figure 6.5: Adjusted localised GNSS data of northeast segment

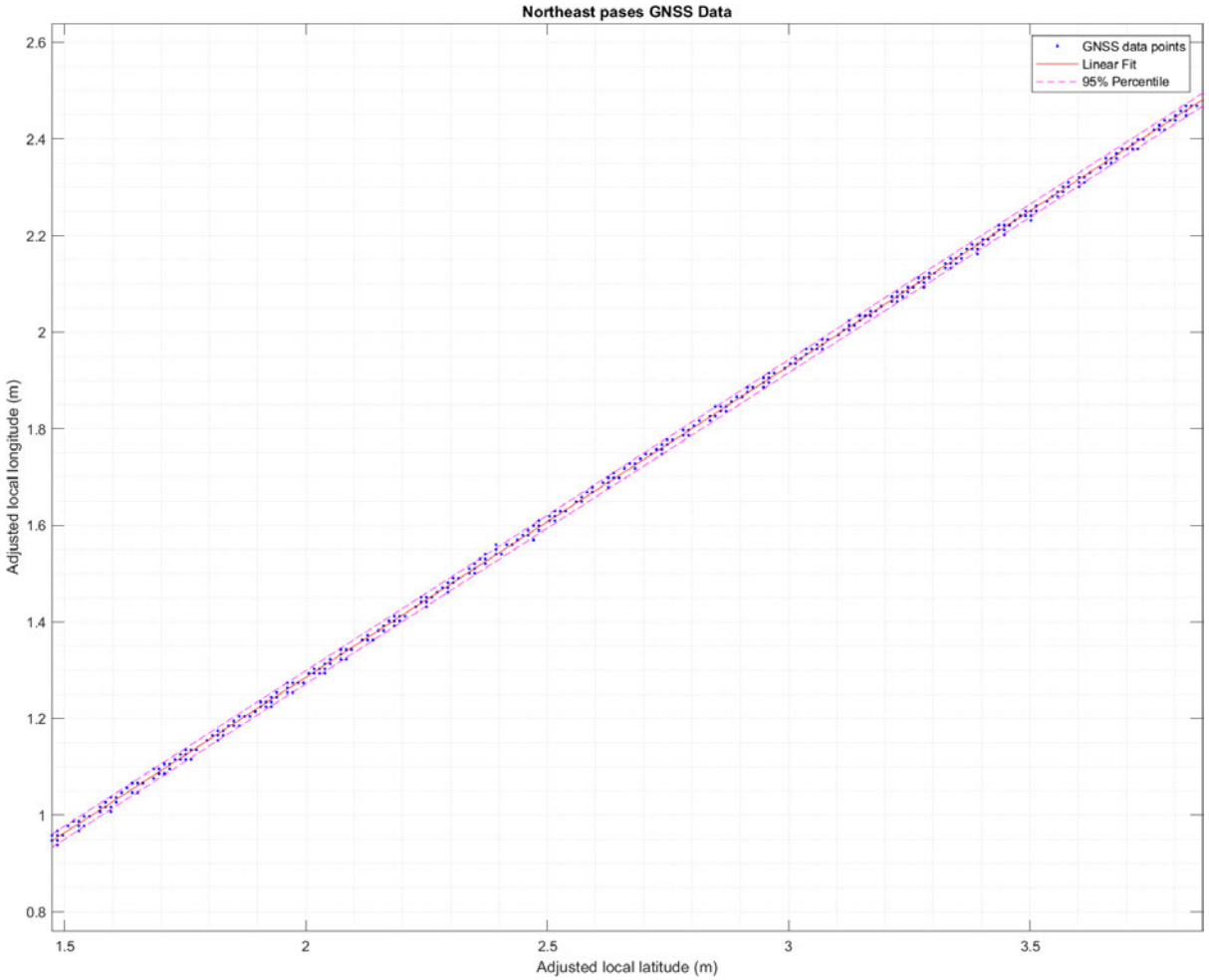


Figure 6.6: Adjusted localised GNSS data of northeast segment (close up view)

Using the equation for the line of best fit ($y = mx + c$) for each segment, the distance (absolute error) of each trajectory point (p_x, p_y) from the line of best fit was calculated algebraically using equation 6.1. The mean and standard deviation of unsigned errors from this reference line was then used to calculate relative cross track accuracy using equation 6.2 (ISO 2010). The results of these calculations are shown in figure 6.7, with an overall relative cross-track accuracy estimated to be 12.13 mm.

$$d = \left| \frac{mp_x - p_y + c}{\sqrt{1 + m^2}} \right| \tag{6.1}$$

$$accuracy = \sqrt{2}(\bar{x} + S_x) \tag{6.2}$$

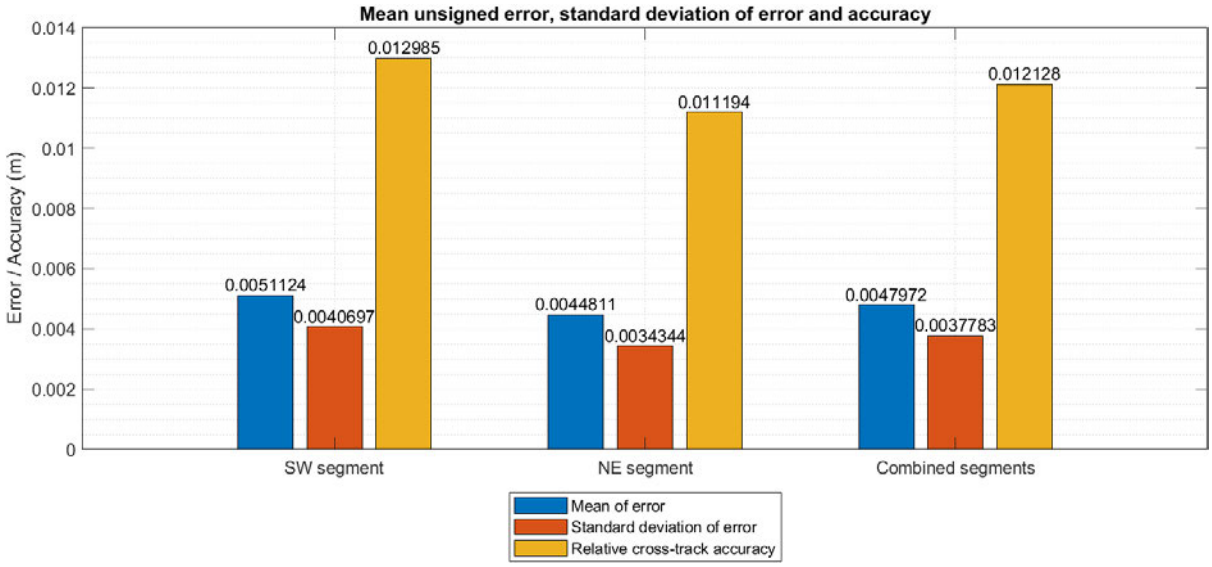


Figure 6.7: Mean and standard deviation of unsigned errors for each segment and overall.

6.1.3 Heading Accuracy

The heading data collected in the two straight segments were also collated into two groups, where table 6.1 shows the mean and standard deviation for each segment. Using the mean heading along each straight segment, the error from the mean heading for each data point was used to calculate heading accuracy ($\bar{x} + S_x$) (ISO 2010), with the results presented in figure 6.8, showing an overall estimated heading accuracy of 0.45°.

Table 6.1: Heading data

	$\bar{x}$ **(deg)**	S_x **(deg)**
Southwest Segment	230.4913	0.3743
Northeast Segment	50.8427	0.2479

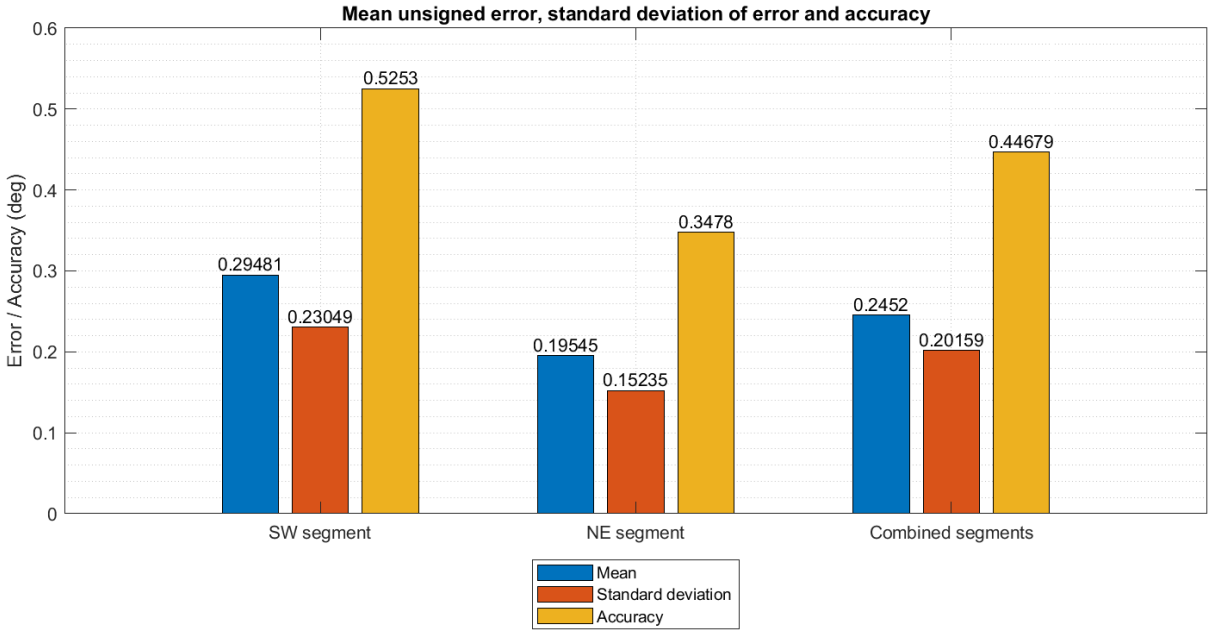


Figure 6.8: Mean and standard deviation of unsigned heading errors for each segment and overall.

6.1.4 Errors and Bias of Results

In order to check for errors and bias within the data obtained, the signed relative cross track errors were plotted for each segment and for each pass in figure 6.9, showing a significant pattern of distribution in the southwest travel when compared to the northeast passes.

Heading

The difference between mean heading and recorded heading were also plotted to check for patterns that may indicate bias in figures 6.10 (all data sequentially) and 6.11 (overlayed pass by pass). Again, a clear pattern emerged in the southwest segment, with 22 distinct passes showing in figure 6.10 indicating that the travel through this segment was not entirely straight.

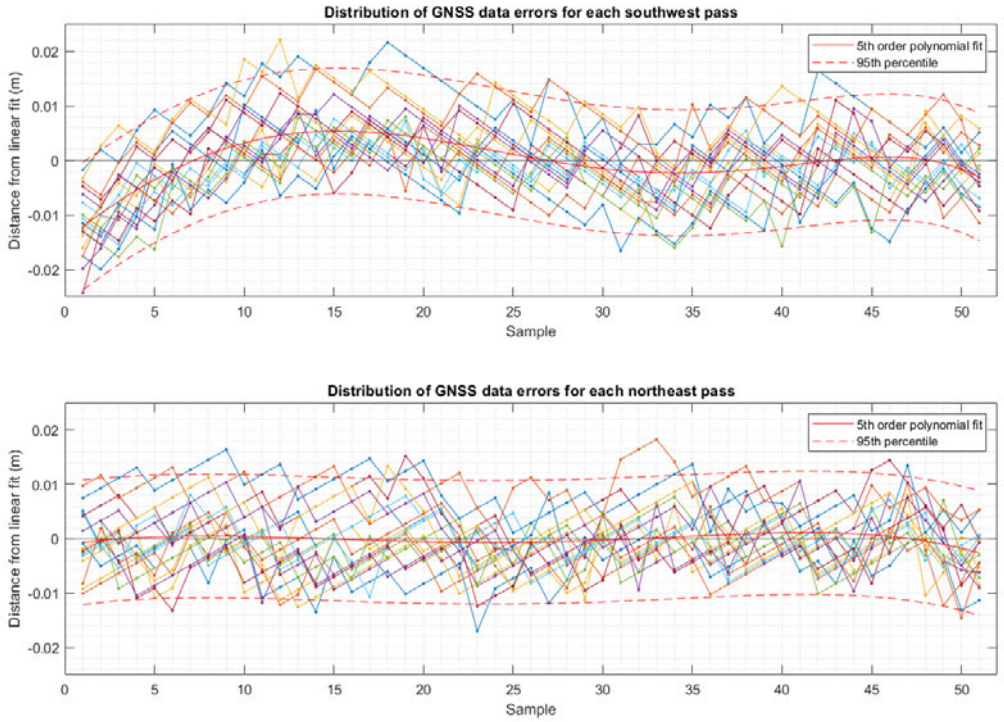


Figure 6.9: Signed heading errors for each straight segment.

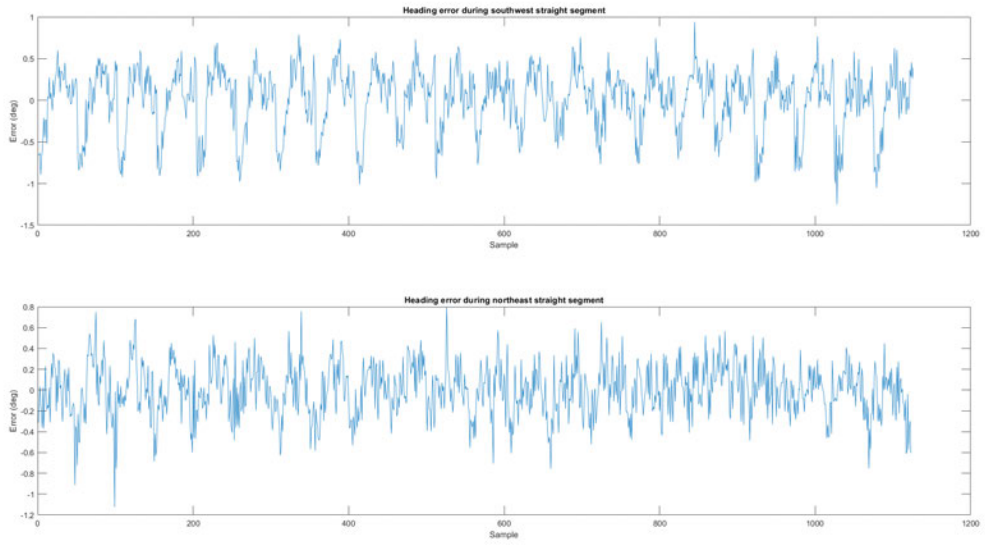


Figure 6.10: Signed heading errors for each straight segment.

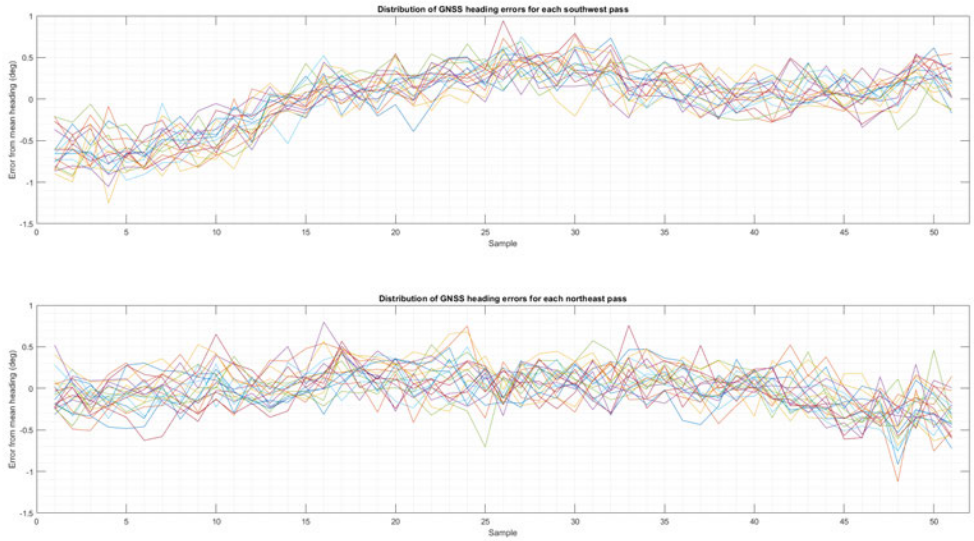


Figure 6.11: Signed heading errors for each straight segment.

Chapter 7

Discussion

7.1 Discussion

This project set out to design and test a low-cost RTK GNSS system, including a base station and rover station, capable of providing accurate position, heading, and speed data for a mobile robotic system. Given the low uptake of precision agriculture in small scale farming due to cost (Thompson et al. 2019), developing a low-cost system has the potential to make precision agriculture more accessible to small scale producers and go some way to reducing impacts of farming on the environment.

The results indicate that the low-cost RTK GNSS system designed is precise in horizontal position and heading, with a relative cross track accuracy of 12.13 mm and heading accurate to 0.45°. This is in line with the manufacturer's datasheet reporting 0.01 m + 1 ppm CEP horizontal accuracy and 0.56 degree accuracy (for a rover/moving base baseline of 0.5 m). In comparison, a high-cost RTK GNSS receiver, such as the Trimble R10, reports a dynamic horizontal RTK accuracy of 2 cm RMS (Trimble 2021*b*).

Examining the signed errors and bias within the data (figures 6.9 - 6.11), it can clearly be seen in the southwest passes that a bias exists, effecting the results. This bias can be explained by two factors: the window for data point selection for straight segment analysis and the true straightness of the track itself. By conducting a polynomial regression on the signed errors for each pass, it appears that the data window selected included trajectory points when the rover antenna had not yet completed a turn onto the straight segment, as indicated in the first sample points of each pass in figure 6.9. In addition, local maximum errors around sample 15 and 33 could also indicate that there deviations in the lay of the track around those points to consistently induce an error when compared to a line of best fit. Therefore, this could suggest that the system actually has increased relative cross-track and heading accuracy, with the northeast segment displaying a relative cross-track accuracy of 11.19 mm and heading accuracy

0.35°.

However, it should be noted, due to the limited testing conducted and without surveying the reference course, that the results are only relative accuracies, not absolute. Further accuracy errors will occur due to the accuracy of the base station which is sending correction data. Although, if the base station configuration remains the same, the error in absolute position data will also remain the same. Therefore, the precision of the rover station will remain the same, it will continue to report the same position data for a given position within the relative accuracy for a given position in space, it will just have the same absolute error from a global datum reference.

For use in agricultural robotics and automation, this absolute error can be negated where precision with accuracy in relation to a global datum reference is not required. By establishing a permanent base station for a RTK network, there will be a local datum reference, offset from the global reference by the base stations position error. Any rover station on this RTK network will have the same error in magnitude and direction, so in an example where there multiple autonomous robots working as a swarm, either in cooperation or independently, they will still maintain precise localisation data in relation to each other. Also, in the case of navigating crop rows, where the cross-track accuracy test is designed for, a vehicle, whether autonomous or using an auto steering system, is able to make repeatable and consistent travel along any course, thus reducing the variability of wheel ruts and therefore reducing the spread of soil compaction.

The overall cost of the RTK GNSS system designed in this project was less than $2,000 using development boards, which could be further reduced by designing custom PCBs around the ZED-F9P module for the required purpose. At this price point, it is feasible to achieve centimetre level localisation systems in multiple small autonomous vehicles.

The test track and driven trolley system developed during this project proved successful in providing a stable system in which to assess GNSS systems. However, further refinement and development of the system is required. As indicated in the bias in signed errors, data window selection for straight segments could have been automated and made more precise by implementing the originally planed photo-interrupter. This would have also allowed the use of the quadrature encoder count values to measure the actual distance along the straight segment, where overall position accuracy of the system could have been assessed, rather than just cross-track accuracy.

Chapter 8

Conclusions and Further Work

8.1 Conclusions

The key objectives of designing a low-cost RTK GNSS system and evaluating its position and heading accuracy was met, with limited testing indicating that the system designed was able to provide precise PVT data in a timely manner. Though many issues encountered limited the full implementation of the evaluation system, with further work, more data on the low-cost RTK GNSS system can be acquired to further validate the system for use in robotic localisation.

The ZED-F9P GNSS receiver module displayed promising results to provide a reliable, accurate and cost effective means for use in small autonomous vehicles, which can be deployed individually or in fleets to achieve a growing number of precision agriculture tasks such as acquiring site specific sensor data, weed and pest management.

8.2 Recommendations and Further Work

There were key design features of the testing rig which would have enhanced the quality of data collected which were not implemented on the day of testing due to time constraints. By increasing the length of the straight segments, implementing the photo-interrupter circuit, surveying the reference course and increasing the number and duration of test runs would have greatly enhanced the results obtained to truly evaluate a low-cost RTK GNSS system.

Additional straight sections were manufactured, which could have provided straight segment lengths of 20 meters. This would have yielded a higher ratio of data points along the straight segments, as during the testing that did occur, over half the data points were located on the curves of the reference track.

By implementing the photo-interrupter, data collection can be more streamlined, but also a reference position along the track can be determined, allowing for the calculating more than just cross-track and heading accuracy. In addition, the testing that did occur indirectly highlighted the need to move the photo-interrupter slot away from the transition from straight segment to curved segment so to insure rover receiver antenna is actually conducting straight travel when data is collected. This is just a matter of section placement when assembling the track.

Surveying the reference course, as well as the RTK base station, is essential for determining the absolute accuracy of the system under evaluation. It would provide a ground truth referenced to a global datum of the path travelled by the GNSS receiver for which to compare the output and determine errors. Thus providing a more rigorous evaluation of the system.

In addition to the physical changes, it is also recommended to carry out a more thorough testing methodology by conducting longer tests, at different speeds and over the period of a whole 24 hour period to capture the differences that receiver velocity and atmospheric conditions have on the receivers accuracy.

References

Amazon Web Services (n.d.), 'FreeRTOS', viewed 12 April 2021, https://www.freertos.org/index.html.

ArduSimple (2021), 'SimpleRTK2B OEM RTK receivers', viewed 20 April 2021, https://www.ardusimple.com/simplertk2b-receivers/.

Australian Radiofrequency Spectrum Plan 2017 (Cwlth) (2017). viewed 20 April 2021, https://www.legislation.gov.au/Details/F2016L02001.

Cole, J. T., Stombaugh, T. S. & Shearer, S. A. (2004), Development of a test track for the evaluation of gps receiver dynamic performance, *in* '2004 ASAE/CSAE Annual International Meeting, 1-4 August 2004, Ottawa, Ontario, Canada'.

Department of the Prime Minister and Cabinet (2021), 'Critical technologies discussion paper: Agriculture', Critical Technologies Policy Coordination Office, viewed 25 May 2021, https://pmc.gov.au/sites/default/files/publications/ctpco-discussion-paper-agriculture.pdf.

Digi (2020), *XBee/XBee-PRO SX radio frequency module – user guide*, Digi International. viewed 12 Aprl 2021, https://www.digi.com/resources/documentation/digidocs/pdfs/90001477.pdf.

Hamza, V., Stopar, B., Ambrozic, T., Turk, G. & Sterle, O. (2020), 'Testing multi-frequency low-cost gnss receivers for geodetic monitoring purposes', *Sensors,* **Vol. 20**(4375).

ISO (2010), *ISO 12188-1:2010 - Standard Tractors and machinery for agriculture and forestry – Test procedures for positioning and guidance systems in agriculture. Part 1: Dynamic testing of satellite-based positioning devices*, ISO.

Knight, B. & Malcolm, B. (2009), 'A whole-farm investment analysis of some precision agriculture technologies', *Australasian Farm Business Management Journal,* **Vol. 6**(1), p. 14. viewed 21 May 2021, https://cdn.csu.edu.au/__data/assets/pdf_file/0010/109459/AFBM_Journal_v06_n01_04_Knight_and_Malcolm.pdf.

Mouser (2021*a*), 'AS-RTK2b-F9P-L1L2-NH-02 ArduSimple | Mouser', viewed 29 May 2021, https://au.mouser.com/ProductDetail/780-F9PL1L2NH02.

Mouser (2021*b*), 'AS-STARTKIT-LR-L1L2-NANH-00 ArduSimple | Mouser', viewed 29 May 2021, https://au.mouser.com/ProductDetail/780-TKITLRL1L2NANH00.

ODrive (2021), 'ODrive robotics', viewed 12 April 2021, https://odriverobotics.com/.

Perez-Ruiz, M. & Upadhyaya, S. K. (2012), *GNSS in Precision Agricultural Operations*, IntechOpen. Publication Title: New Approach of Indoor and Outdoor Localization Systems, viewed 30 May 2021, https://www.intechopen.com/books/new-approach-of-indoor-and-outdoor-localization-systems/gnss-in-precision-agricultural-operations.

Pini, M., Marucco, G., Falco, G., Nicola, M. & De Wilde, W. (2020), 'Experimental testbed and methodology for the assessment of RTK GNSS receivers used in precision agriculture', *IEEE Access* **Vol. 8**, pp. 14690–14703. viewed 12 April 2021, https://ieeexplore-ieee-org.ezproxy.usq.edu.au/document/8955794.

Rounsaville, J. D., Dvorak, J. S. & Stombaugh, T. S. (2016), 'Methods for Calculating Relative Cross-Track Error for ASABE/ISO Standard 12188-2 from Discrete Measurements', *Transactions of the ASABE* **59**(6), 1609–1616. viewed 10 October 2021, http://elibrary.asabe.org/abstract.asp?aid=47588&t=3&dabs=Y&redir=&redirType=.

Rybacki, P., Przygodziński, P., Osuch, A., Blecharczyk, A., Walkowiak, R., Osuch, E. & Kowalik, I. (2021), 'The Technology of Precise Application of Herbicides in Onion Field Cultivation', *Agriculture* **11**(7), 577. viewed 24 August 2021, https://www.mdpi.com/2077-0472/11/7/577.

Shamshiri, R. R., Hameed, I. A., Balasundram, S. K., Ahmad, D., Weltzien, C. & Yamin, M. (2018), *Fundamental Research on Unmanned Aerial Vehicles to Support Precision Agriculture in Oil Palm Plantations*, IntechOpen. Publication Title: Agricultural Robots - Fundamentals and Applications, viewed 13 October 2021, https://www.intechopen.com/chapters/63775.

STMicroelectronics (2021), 'High-performance and dsp with fpu, arm cortex-m7 mcu with 2 mbytes of flash memory, 216 mhz cpu, art accelerator, l1 cache, sdram, tft, jpeg codec, dfsdm', viewed 12 April 2021, https://www.st.com/en/microcontrollers-microprocessors/stm32f767zi.html#overview&secondary=st_description_sec-nav-tab.

Stombaugh, T. S., Sama, M. P., Zandonadi, R. S., Shearer, S. A. & Koostra, B. K. (2008), Standardized evaluation of dynamic gps performance, *in* '2008 ASABE Annual International Meeting, 29 June - 2 July 2008, Providence, Rhode Island, USA'.

Thompson, N. M., Bir, C., Widmar, D. A. & Mintert, J. R. (2019), 'Farmer Perceptions of Precision Agriculture Technology Benifits', *Journal of Agricultural and Applied Economics*

51(1), 142–163. viewed 10 October 2021, https://www.cambridge.org/core/product/identifier/S1074070818000275/type/journal_article.

Trimble (2021*a*), 'NMEA-0183 message: RMC', viewed 20 May 2021, https://www.trimble.com/OEM_ReceiverHelp/V4.44/en/NMEA-0183messages_RMC.html.

Trimble (2021*b*), 'Trimble r10 model 2 GNSS system datasheet', viewed 12 October 2021, https://geospatial.trimble.com/sites/geospatial.trimble.com/files/2021-07/022516-332B_TrimbleR10-2_DS_USL_0721_LR.pdf.

Tzounis, A., Katsoulas, N., Bartzanas, T. & Kittas, C. (2017), 'Internet of things in agriculture, recent advances and future challenges', *Biosystems Engineering,* **Vol. 164**, pp. 31–48.

Ublox (2020*a*), *ZED-F9P Datasheet.* viewed 12 April 2021, https://www.u-blox.com/sites/default/files/ZED-F9P_DataSheet_%28UBX-17051259%29.pdf.

Ublox (2020*b*), *ZED-F9P interface description.* viewed 12 April 2021, https://www.u-blox.com/sites/default/files/ZED-F9P_InterfaceDescription_%28UBX-18010854%29.pdf.

Williams, R. D., Lamy, M.-L., Maniatis, G. & Stott, E. (2020), 'Three-dimensional reconstruction of fluvial surface sedimentology and topography using personal mobile laser scanning', *Earth Surface Processes and Landforms* **45**(1), 251–261. viewed 13 October 2021, http://onlinelibrary.wiley.com/doi/abs/10.1002/esp.4747.

Wolfert, S., Ge, L., Verdouw, C. & Bogaardt, M, J. (2017), 'Big data in mmart farming: a review', *Agricultural Systems,* **Vol. 153**, pp. 69–80.

Zhang, N., Wang, M. & Wang, N. (2002), 'Precision agriculture—a worldwide overview', *Computers and Electronics in Agriculture* **Vol. 36**(2), pp. 113–132. viewed 28 May 2021, https://linkinghub.elsevier.com/retrieve/pii/S0168169902000960.

Zhang, Q. (2016), *Precision agriculture technology for crop farming*, Taylor and Francis, Boca Raton. viewed 13 October 2021, https://usq.primo.exlibrisgroup.com/view/action/uresolver.do?operation=resolveService&package_service_id=3390667710004691&institutionId=4691&customerId=4690.

Appendix A

Project specification

ENG4111/4112 Research Project

Project Specification

For:	Simon Castles
Title:	Precision RTK GNSS for low-cost robotic system.
Major:	Mechatronics engineering
Supervisor:	Dr. Craig Lobsey
Enrolment:	ENG4111 – ONL S1, 2021
	ENG4112 – ONL S2, 2021
Project aim:	To design and evaluate a low-cost RTK-GNSS systems that can be used to make precision agriculture more widely available, and support the development of low-cost agricultural robotic systems.
Programe:	**Version 1, 17 March 2021**

1. Research background information on precision agriculture and the use of RTK-GNSS.
2. Review methods and standards of evaluating accuracy of localisation systems.
3. Review accuracy requirements for precision agriculture (e.g. guidance) and agricultural robotics.
4. Design a low-cost RTK-GNSS base station and rover station for an agricultural robotic system.
5. Design evaluation rigs for rover position and heading accuracy.
6. Evaluate the RTK-GNSS base station for position accuracy using "survey-in" mode vs survey mark.

7. Evaluate rover station benchmarks of position accuracy, heading accuracy and radio range.

If time and resources permit:

8. Conduct a field trial of the RTK-GNSS system mounted on agricultural vehicle.
9. Evaluate heading accuracy of a rover station consisting of two GNSS receivers vs a rover station consisting of GNSS + INS + compass.
10. Incorporate and evaluate pairing RTK-GNSS system data with an underground soil sensor receiver.

Appendix B

Project Management

B.1 Project Schedule

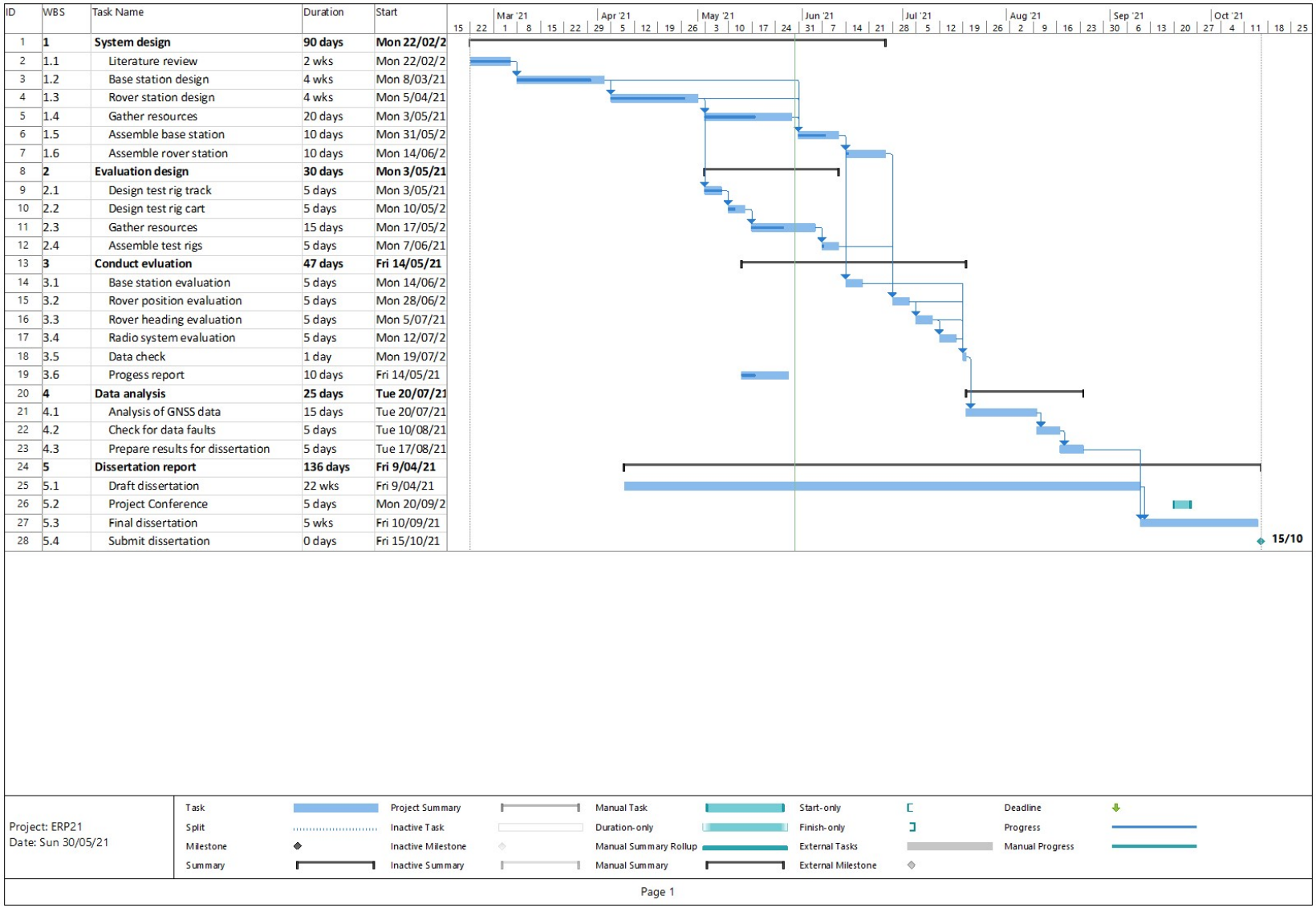
The was is planned to be carried out in 5 phases:

- Phase 1: Design and build an RTK GNSS system.
- Phase 2: Design and build position and heading evaluation test rig.
- Phase 3: Conduct evaluation and collect position and heading data of an RTK-GNSS system.
- Phase 4: Analysis of data.
- Phase 5: Dissertation report.

Each phase has been broken into tasks that will be undertaken to accomplish the project aim. Table B.1 outlines each task within the phases, which has been further broken down into the project schedule presented in figure B.1.

Table B.1: Phase breakdown

Phase 1 - Design and build an RTK-GNSS system	
1.1	Conduct literature review of precision agriculture, localisation and existing RTK-GNSS systems.
1.2	Design an RTK-GNSS base station with an open-source operating system allowing a user to input position or use "survey-in" mode.
1.3	Design an RTK-GNSS rover station with an open-source operating system with the ability to log GNSS data for export and later analysis.
1.4	Acquire additional resources for system assembly.
1.5	Assemble base station.
1.6	Assemble rover station.
Phase 2 – Design and build evaluation test rig	
2.1	Design a test rig for evaluation of rover position accuracy.
2.2	Design a test rig for evaluation of rover heading accuracy.
2.3	Acquire additional resources for test rig assembly.
2.4	Assemble test rigs.
Phase 3 - Conduct evaluation of position and heading accuracy of the system	
3.1	Conduct evaluation of the base station for position accuracy using "survey-in" mode vs survey mark.
3.2	Conduct evaluation of rover station position accuracy.
3.3	Conduct evaluation of rover station heading accuracy.
3.4	Conduct evaluation of system radio range.
3.5	Check to ensure results have been recorded for tasks 3.1, 3.2, 3.3 and 3.4.
3.6	Progress report.
Phase 4 – Analysis of data	
4.1	Prepare analysis of position data points, compare with calculated.
4.2	Check data for collection faults.
4.3	Prepare results for discussion in dissertation.
Phase 5 – Dissertation report	
5.1	Continually write up draft dissertation and submit to supervisor for feedback.
5.2	Present project at ENG4903 conference.
5.3	Finalise dissertation and submit for marking.



ID	WBS	Task Name	Duration	Start
1	**1**	**System design**	**90 days**	**Mon 22/02/2**
2	1.1	Literature review	2 wks	Mon 22/02/2
3	1.2	Base station design	4 wks	Mon 8/03/21
4	1.3	Rover station design	4 wks	Mon 5/04/21
5	1.4	Gather resources	20 days	Mon 3/05/21
6	1.5	Assemble base station	10 days	Mon 31/05/2
7	1.6	Assemble rover station	10 days	Mon 14/06/2
8	**2**	**Evaluation design**	**30 days**	**Mon 3/05/21**
9	2.1	Design test rig track	5 days	Mon 3/05/21
10	2.2	Design test rig cart	5 days	Mon 10/05/2
11	2.3	Gather resources	15 days	Mon 17/05/2
12	2.4	Assemble test rigs	5 days	Mon 7/06/21
13	**3**	**Conduct evluation**	**47 days**	**Fri 14/05/21**
14	3.1	Base station evaluation	5 days	Mon 14/06/2
15	3.2	Rover position evaluation	5 days	Mon 28/06/2
16	3.3	Rover heading evaluation	5 days	Mon 5/07/21
17	3.4	Radio system evaluation	5 days	Mon 12/07/2
18	3.5	Data check	1 day	Mon 19/07/2
19	3.6	Progess report	10 days	Fri 14/05/21
20	**4**	**Data analysis**	**25 days**	**Tue 20/07/21**
21	4.1	Analysis of GNSS data	15 days	Tue 20/07/21
22	4.2	Check for data faults	5 days	Tue 10/08/21
23	4.3	Prepare results for dissertation	5 days	Tue 17/08/21
24	**5**	**Dissertation report**	**136 days**	**Fri 9/04/21**
25	5.1	Draft dissertation	22 wks	Fri 9/04/21
26	5.2	Project Conference	5 days	Mon 20/09/2
27	5.3	Final dissertation	5 wks	Fri 10/09/21
28	5.4	Submit dissertation	0 days	Fri 15/10/21

Figure B.1: Project schedule

B.2 Resource Requirements

The resources required for this project have been broken up to hardware requirements, software requirements and additional time/space resources.

B.2.1 Hardware requirements

As the basis of this project, three SimpleRKT2b boards will be required to form the centre of the RTK GNSS base station and rover station (a second receiver being used to produce heading data without addional sensors). In order to achieve wireless transmission of RTCM messages between the base station and rover station, two Xbee Pro SX radios will be used. So that the GNSS PVT data can be accessed, readable and recorded, either a microcontroller or system on a chip (SoC) will also be required to implement either basic data acquisition or ROS2. Table B.2 outlines the essential hardware requirements for the design system.

Table B.2: Project system resource requirements

Resource item	Qty	Source	Remarks
simpleRTK2B-F9P RTK receiver	3	USQ	
Survey GNSS antenna	3	USQ	
Xbee Pro SX radio	2	USQ	For RTCM messages
Xbee Pro S3B radio	2	Student	For heading messages
STM32 Nucleo F767ZI	1	Student	FreeRTOS and MicroROS integration

For the travel course cart, propulsion will be achieved through a brushless DC motor (BLDC) controlled be the ODrive motor controller and STM32 microcontroller which will also function to record the actual path traversed. Table B.3 outlines the major hardware requirements for the travel course cart.

Table B.3: Project test rig resource requirements

Resource item	Qty	Source	Remarks
STM32 Nucleo F401RE	1	Student	For system control and DAQ
ODrive 3.6 56v motor controller	1	Student	For motor control
CUI AMT-102 rotary encoder	2	Student	For speed control and distance DAQ
4250 410 Kv BLDC	1	Student	For track movement
T-Slot Photo Interrupter	1	USQ	For reference point DAQ

B.2.2 Software requirements

For integration of this project into a larger robotics system, ROS2 will be implemented on the RTK GNSS rover station sub-system. If a microcontroller is used in this sub-system, a slimmed

down version of ROS2 for microcontrollers called micro-ROS will be implemented, which will also require FreeRTOS to be implemented on the STM32 F746ZG identified above.

B.2.3 Addition time/space resources

For rapid prototyping of the test rig track and cart components, a Trotec Speedy 300 CO^2 laser cutter will be used at the State Library of Queensland (SLQ). For final production of the test track, USQ's water-jet cutter will be trialled for suitability as an alternative to SLQ's resources.

To ensure accurate evaluation of the system, the RTK GNSS base station locations and the four reference points on the travel course will require to be professionally surveyed by GIS staff or students from USQ. A large space for the travel course to be located that provides 10° from horizon clearance to the antenna and is from from metallic objects at a distance of 50 m is also required for the duration of the evaluation phase of the project.

B.3 Risk management

The safety risk management plan for this project is presented below.

University of Southern Queensland **Offline Version**

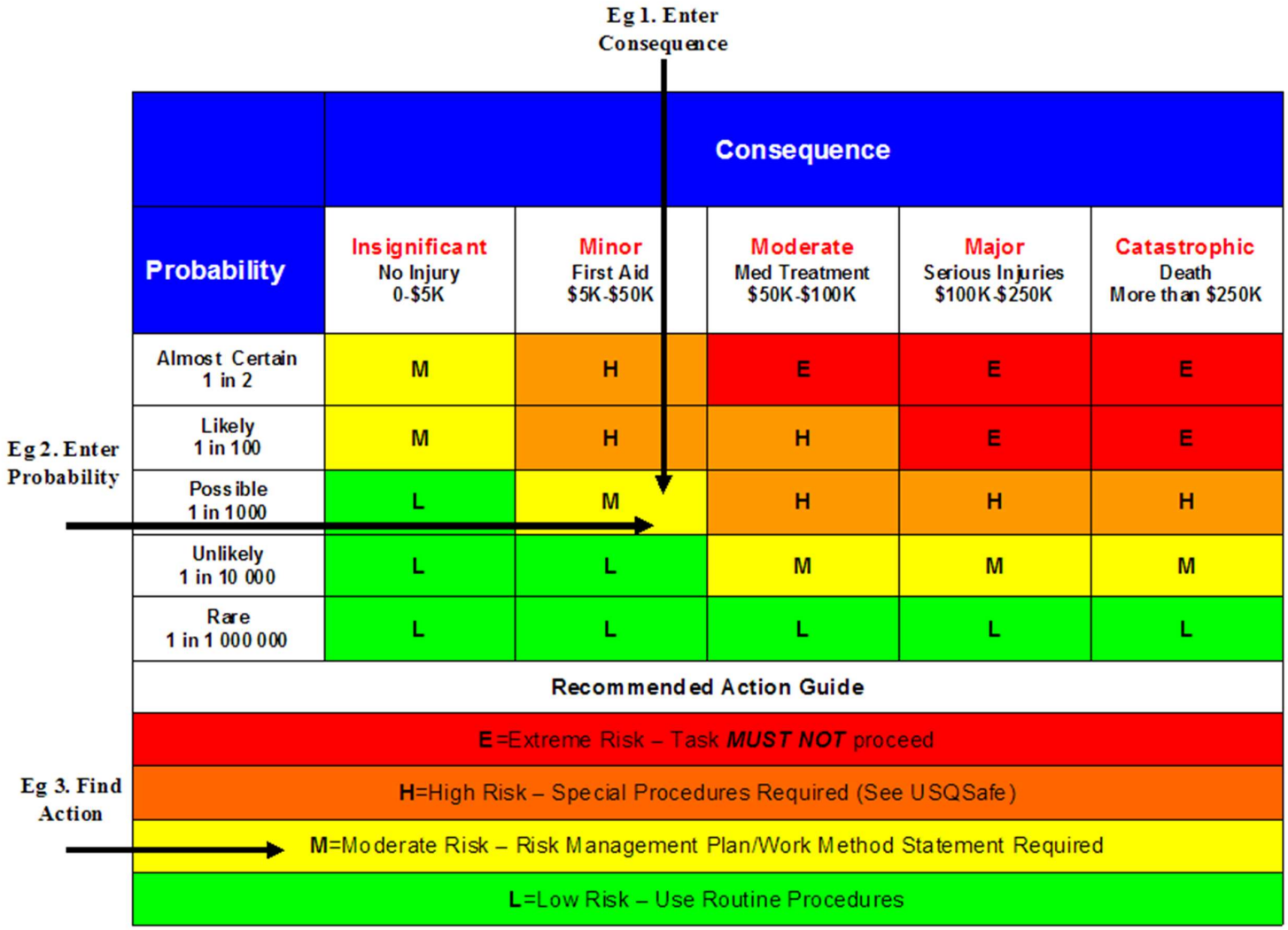
USQ Safety Risk Management System

Note: This is the offline version of the Safety Risk Management System (SRMS) Risk Management Plan (RMP) and is only to be used for planning and drafting sessions, and when working in remote areas or on field activities. It must be transferred to the online SRMS at the first opportunity.

Safety Risk Management Plan – Offline Version			
Assessment Title:	Precision RTK GNSS for low-cost robotic system	Assessment Date:	28/06/2021
Workplace (Division/Faculty/Section):	Faculty of Health, Engineering and Science	Review Date:(5 Years Max)	21/06/2021

Context			
Description:			
What is the task/event/purchase/project/procedure?	Construction and field tril of electronic equipment		
Why is it being conducted?	In completion of honours year project.		
Where is it being conducted?	USQ Toowoomba Campus		
Course code (if applicable)	ENG4112	Chemical name (if applicable)	N/A
What other nominal conditions?			
Personnel involved	1 x Undergraduate Student (Simon Castles)		
Equipment	Micro controllers, evaluation boards, low voltage DC powered motors.		
Environment	Open field under non incliment weather conditions.		
Other			
Briefly explain the procedure/process	A small lightweight robotic system will move on a 20 x 5 m track at speeds up to 5 m/s (18 km/h) continuously for 24 one hour blocks over a 25 hour period whilst recording localisation data.		

Assessment Team - who is conducting the assessment?	
Assessor(s)	Dr Craig Lobsey
Others consulted:	


Eg 1. Enter Consequence
Eg 2. Enter Probability
Eg 3. Find Action
Consequence
Probability
Insignificant
No Injury
0-$5K
Minor
First Aid
$5K-$50K
Moderate
Med Treatment
$50K-$100K
Major
Serious Injuries
$100K-$250K
Catastrophic
Death
More than $250K
Almost Certain
1 in 2
M H E E E
Likely
1 in 100
M H H E E
Possible
1 in 1000
L M H H H
Unlikely
1 in 10 000
L L M M M
Rare
1 in 1 000 000
L L L L L
Recommended Action Guide
E=Extreme Risk – Task MUST NOT proceed
H=High Risk – Special Procedures Required (See USQSafe)
M=Moderate Risk – Risk Management Plan/Work Method Statement Required
L=Low Risk – Use Routine Procedures

Step 1 (cont)	Step 2	Step 2a	Step 2b	Step 3			Step 4				
Hazards: From step 1 or more if identified	***The Risk:*** What can happen if exposed to the hazard without existing controls in place?	***Consequence:*** What is the harm that can be caused by the hazard without existing controls in place?	***Existing Controls:*** What are the existing controls that are already in place?	***Risk Assessment:*** Consequence x Probability = Risk Level			***Additional controls:*** Enter additional controls if required to reduce the risk level	***Risk assessment with additional controls:***			
				Probability	**Risk Level**	**ALARP? Yes/no**		**Consequence**	**Probability**	**Risk Level**	**ALARP? Yes/no**
Example											
Working in temperatures over 35° C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	catastrophic	Regular breaks, chilled water available, loose clothing, fatigue management policy.	possible	high	No	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
Working with LiPo batteries	Battery catches fire during recharging leads to serious injury or damage	Major	Use of LiPo safe pouch when charging, use appropriate LiPo battery charger, supervise charging	Unlikely	Moderate	No	Substitute NiCad/NiMH if possible	Minor	Rare	Low	Yes
Use of soldering iron	causes injury (burn to eyes/skin)	Moderate	Use of protective eyeware when soldering.	Unlikely	Moderate	No	Wear long sleeve, natural fibre clothing	Minor	Unlikely	Low	Yes
Use of soldering iron	Exposure to toxic fumes leeding to long term health effects	Moderate	Use of lead free solder	Possible	High	No	Use of ventilation fan	Moderate	Rare	Low	Yes
Extended period in fieldwork	Dehydration, heat injury leads to serious injury	Major	Regular breaks, ensure water availability, loose clothing, fatigue management policy	Unlikely	Moderate	No	Set up an admin area at travel course with food, water and shelter.	Moderate	Rare	Low	Yes
Fieldwork involving vehicles or robotic platforms	Collision leads to serious damage	Major	Complete USQ induction / training prior to use of vehicle or platform	Unlikely	Moderate	No	Conduct site induction to any personnel within the travel course area	Major	Rare	Low	Yes
		Select a consequence		Select a probability	Select a Risk Level	Yes or No		Select a consequence	Select a probability	Select a Risk Level	Yes or No
		Select a consequence		Select a probability	Select a Risk Level	Yes or No		Select a consequence	Select a probability	Select a Risk Level	Yes or No
		Select a consequence		Select a probability	Select a Risk Level	Yes or No		Select a consequence	Select a probability	Select a Risk Level	Yes or No
		Select a consequence		Select a probability	Select a Risk Level	Yes or No		Select a consequence	Select a probability	Select a Risk Level	Yes or No
		Select a consequence		Select a probability	Select a Risk Level	Yes or No		Select a consequence	Select a probability	Select a Risk Level	Yes or No

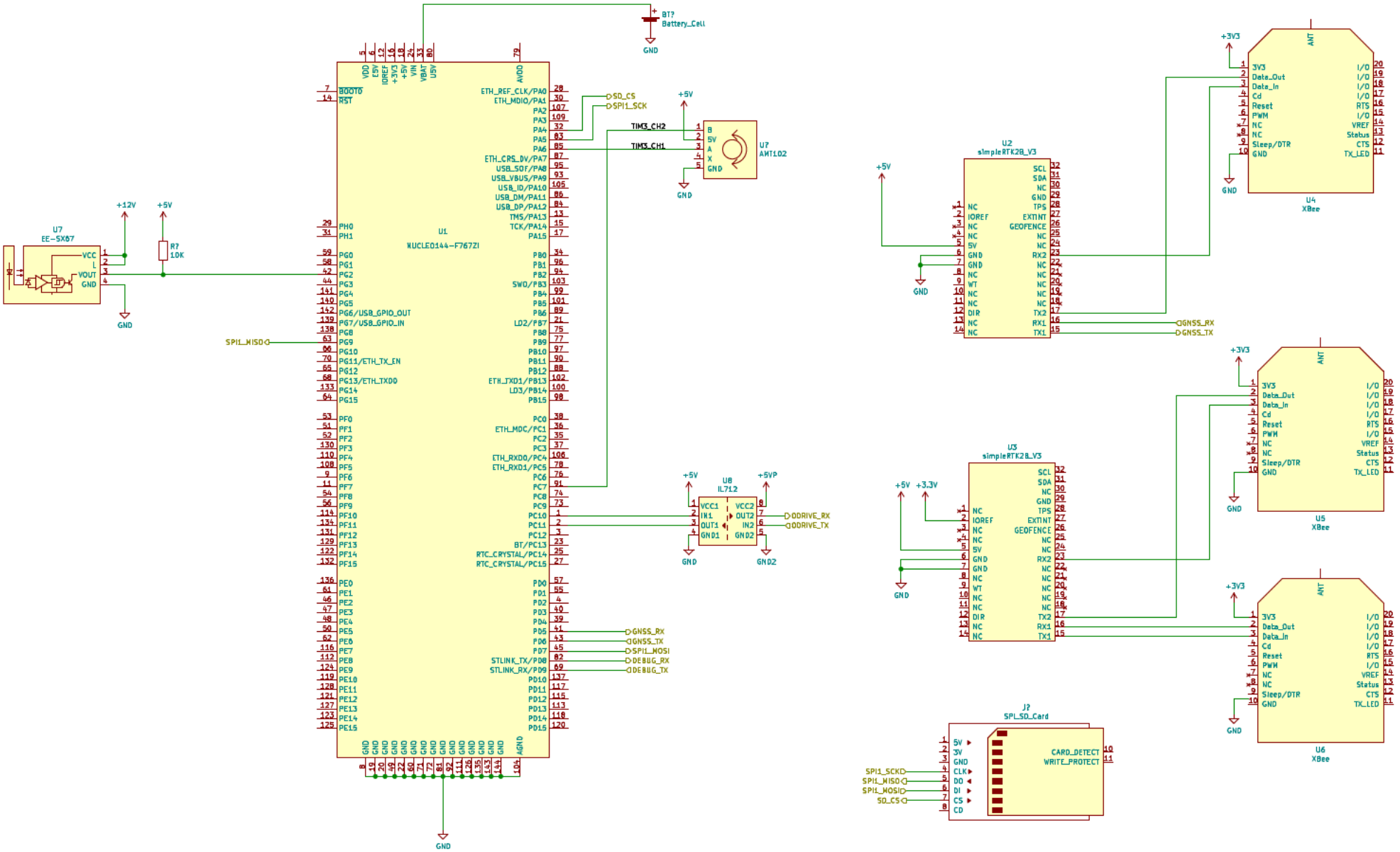
Step 5 - Action Plan (for controls not already in place)			
Additional controls:	*Resources:*	*Persons responsible:*	*Proposed implementation date:*
Substitute LiPo batteries with NiCad/NiMH if possible	NiCad/NiMH batteries	Simon Castles	21/06/2021
Wear long sleeve, natural fibre clothing in properly ventilated space when carrying out solderings	Ventilated soldering station	Simon Castles	1/06/2021
Set up an admin area at travel course with food, water and shelter	Temporary shelter, food, water	Simon Castles	28/06/2021
Site induction brief given to any personnel entering travel course area	Prepared brief	Simon Castles	28/06/2021
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.
			Click here to enter a date.

Step 6 - Approval					
Drafter's name:	Simon Castles			Draft date:	27/05/2021
Drafter's comments:					
Approver's name:	Dr. Craig Lobsey		Approver's title/position:	Senior Lecturer	
Approver's comments:					
I am satisfied that the risks are as low as reasonably practicable and that the resources required will be provided.					

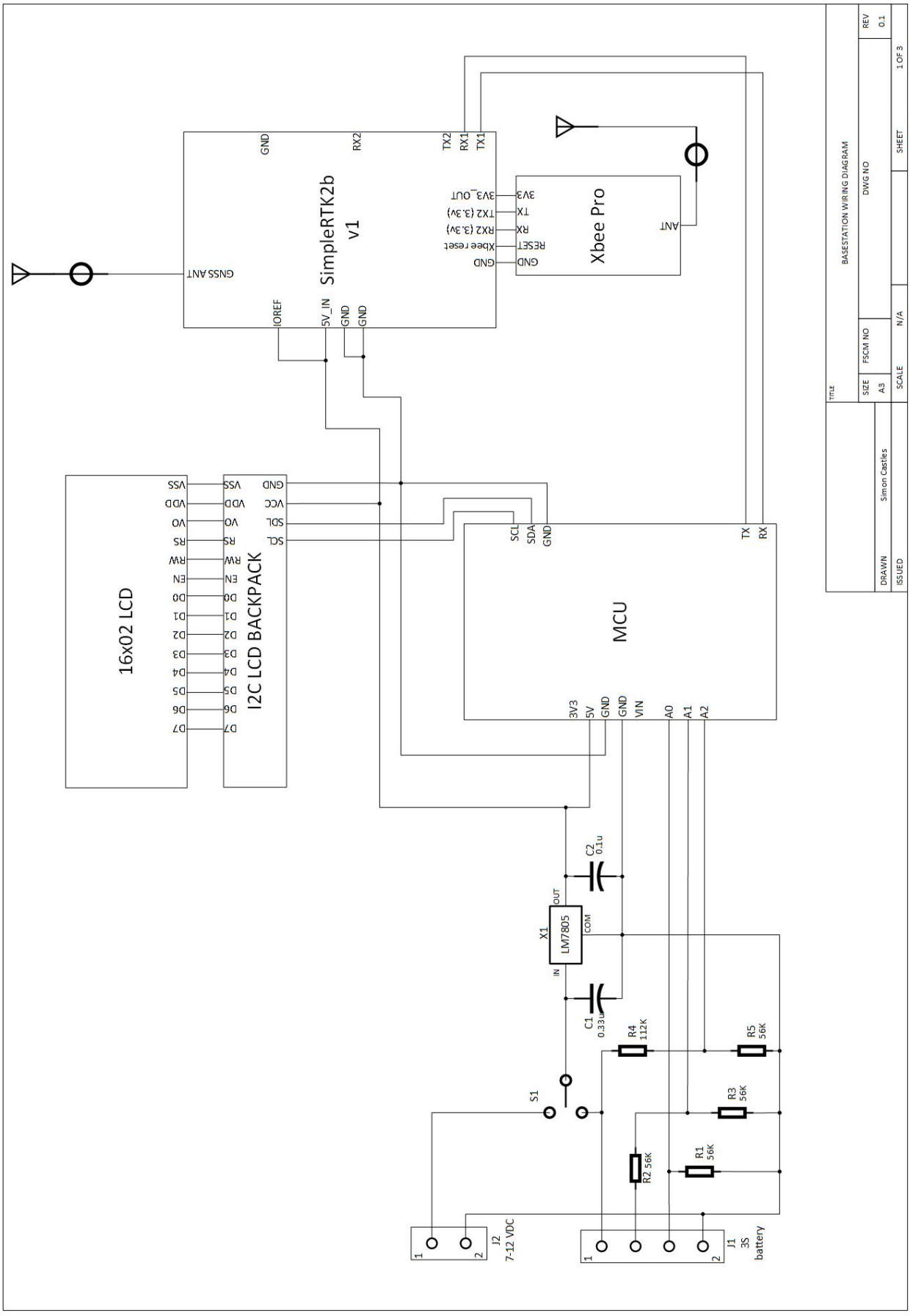
Approver's signature:		Approval date:	Click here to enter a date.

Appendix C

Wiring diagrams



Sheet: /	
File: TrollyDrive.sch	
Title:	
Size: A3	Date:
KiCad E.D.A. kicad (5.1.10)-1	Id: 1/1
	Rev:


BASESTATION WIRING DIAGRAM
TITLE
SIZE
A3
FSCM NO
DWG NO
REV
0.1
SCALE
N/A
SHEET
1 OF 3
DRAWN
Simon Castles
ISSUED
SimpleRTK2b
v1
GND
RX2
TX2
RX1
TX1
GNSS ANT
IOREF
5V_IN
GND
GND
3V3_OUT
TX2 (3.3v)
RX2 (3.3v)
Xbee reset
GND
Xbee Pro
3V3
TX
RX
RESET
GND
ANT
16x02 LCD
I2C LCD BACKPACK
VSS
VDD
VO
RS
RW
EN
D0
D1
D2
D3
D4
D5
D6
D7
GND
VCC
SDL
SCL
MCU
SCL
SDA
GND
TX
RX
3V3
5V
GND
GND
VIN
A0
A1
A2
X1
LM7805
IN
COM
OUT
C2
0.1u
C1
0.33u
R4
112K
R5
56K
R3
56K
R1
56K
R2 56K
S1
J2
7-12 VDC
J1
3S
battery

Appendix D

Microcontroller Source Code - C

D.1 Main C Program

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attentioe exn
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "fatfs.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
```

```c
#include <FreeRTOS.h>
#include <task.h>
#include <semphr.h>
#include <queue.h>
//#include <SEGGER_SYSVIEW.h>
//#include <Nucleo_F767ZI_GPIO.h>
//#include <Nucleo_F767ZI_Init.h>
#include <stm32f7xx_hal.h>

#include "UBX.h"

#include <stdio.h>
#include <string.h>
#include <stdarg.h> //for va_list var arg functions
#include <stdbool.h>

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define STACK_SIZE 128

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

RTC_HandleTypeDef hrtc;

SPI_HandleTypeDef hspi1;
```

```
TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;

UART_HandleTypeDef huart2;
UART_HandleTypeDef huart3;
UART_HandleTypeDef huart6;

/* USER CODE BEGIN PV */
char buffer[100];  //Buffer to hold SD card variable
uint8_t UART2_rxBuffer[1] = {0};  //where 2 is the UART number being used for GN

SemaphoreHandle_t semPtr = NULL; //create storage for a pointer to a semaphore
SemaphoreHandle_t DLsemPtr = NULL; //create storage for a pointer to a semaphore
SemaphoreHandle_t TIMEsemPtr = NULL; //create storage for a pointer to a getTIME

static QueueHandle_t uart2_BytesReceived = NULL;
static QueueHandle_t uart3_BytesReceived = NULL;

static bool rxInProgress = false;

uint8_t segment;
uint32_t encoder_cnt;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_USB_OTG_FS_USB_Init(void);
static void MX_RTC_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM2_Init(void);
static void MX_TIM3_Init(void);
static void MX_USART6_UART_Init(void);
/* USER CODE BEGIN PFP */
void USR_GPIO_Init(void);
void GreenTaskA( void * argument);
void BlueTaskB( void* argumet );
```

```
void DataLogTask( void* argumet );
void myprintf(const char *fmt, ...);
void ProcessUBXTask( void* NotUsed);
void RTC_UpdateTask( void* NotUsed);
void UserInputTask( void* argument );
/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick.
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
```

```
MX_USART3_UART_Init();
MX_USART2_UART_Init();
MX_USB_OTG_FS_USB_Init();
MX_RTC_Init();
MX_SPI1_Init();
MX_TIM2_Init();
MX_TIM3_Init();
MX_USART6_UART_Init();
MX_FATFS_Init();
/* USER CODE BEGIN 2 */
USR_GPIO_Init();
UBX_Init();
//Start timers
HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_ALL);
//SEGGER_SYSVIEW_Conf();

NVIC_SetPriorityGrouping(0);  //ensure proper priority grouping for freeRTOS

//create a semaphore using the FreeRTOS Heap
semPtr = xSemaphoreCreateBinary();
assert_param(semPtr != NULL);
DLsemPtr = xSemaphoreCreateBinary();
assert_param(DLsemPtr != NULL);
TIMEsemPtr = xSemaphoreCreateBinary();
assert_param(DLsemPtr != NULL);

if( xTaskCreate(GreenTaskA, "GreenTaskA", STACK_SIZE, NULL, tskIDLE_PRIORITY +
  NULL) != pdPASS){while(1);}


if( xTaskCreate(BlueTaskB, "BlueTaskB", STACK_SIZE, NULL, tskIDLE_PRIORITY + 1
  NULL) != pdPASS){while(1);}

//create task to carry out data logging
assert_param(xTaskCreate(DataLogTask, "DataLogTask", 256, NULL, tskIDLE_PRIORIT
  NULL) == pdPASS);

//setup tasks, making sure they have been properly created before moving on
uart2_BytesReceived = xQueueCreate(100, sizeof(char));
assert_param(uart2_BytesReceived != NULL);
```

```
  uart3_BytesReceived = xQueueCreate(10, sizeof(char));
  assert_param(uart3_BytesReceived != NULL);


  assert_param(xTaskCreate(UserInputTask, "UserInputTask", STACK_SIZE, NULL,
    tskIDLE_PRIORITY + 1, NULL) == pdPASS);

  assert_param(xTaskCreate(ProcessUBXTask, "ProcessUBX", STACK_SIZE, NULL,
    tskIDLE_PRIORITY + 2, NULL) == pdPASS);

  assert_param(xTaskCreate(RTC_UpdateTask, "RTC_Update", STACK_SIZE, NULL,
    tskIDLE_PRIORITY + 3, NULL) == pdPASS);
  //start the scheduler - shouldn't return unless there's a problem
  vTaskStartScheduler();

  //if you've wound up here, there is likely an issue with overrunning the freeR

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
```

```
/** Configure LSE Drive Capability
*/
HAL_PWR_EnableBkUpAccess();
__HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
/** Configure the main internal regulator output voltage
*/
__HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
/** Initializes the RCC Oscillators according to the specified parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE|RCC_OSCILLATORTYPE_L
RCC_OscInitStruct.HSEState = RCC_HSE_BYPASS;
RCC_OscInitStruct.LSEState = RCC_LSE_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLM = 4;
RCC_OscInitStruct.PLL.PLLN = 216;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 9;
RCC_OscInitStruct.PLL.PLLR = 2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
  Error_Handler();
}
/** Activate the Over-Drive mode
*/
if (HAL_PWREx_EnableOverDrive() != HAL_OK)
{
  Error_Handler();
}
/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
```

```
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_7) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_RTC|RCC_PERIPHCLK_USA
                                |RCC_PERIPHCLK_USART3|RCC_PERIPHCLK_USART6
                                |RCC_PERIPHCLK_CLK48;
  PeriphClkInitStruct.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
  PeriphClkInitStruct.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  PeriphClkInitStruct.Usart3ClockSelection = RCC_USART3CLKSOURCE_PCLK1;
  PeriphClkInitStruct.Usart6ClockSelection = RCC_USART6CLKSOURCE_PCLK2;
  PeriphClkInitStruct.Clk48ClockSelection = RCC_CLK48SOURCE_PLL;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief RTC Initialization Function
  * @param None
  * @retval None
  */
static void MX_RTC_Init(void)
{

  /* USER CODE BEGIN RTC_Init 0 */

  /* USER CODE END RTC_Init 0 */

  RTC_TimeTypeDef sTime = {0};
  RTC_DateTypeDef sDate = {0};

  /* USER CODE BEGIN RTC_Init 1 */

  /* USER CODE END RTC_Init 1 */
  /** Initialize RTC Only
  */
  hrtc.Instance = RTC;
```

```
  hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
  hrtc.Init.AsynchPrediv = 127;
  hrtc.Init.SynchPrediv = 255;
  hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
  hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
  hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }

  /* USER CODE BEGIN Check_RTC_BKUP */

  /* USER CODE END Check_RTC_BKUP */

  /** Initialize RTC and set the Time and Date
  */
  sTime.Hours = 0x0;
  sTime.Minutes = 0x0;
  sTime.Seconds = 0x0;
  sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
  sTime.StoreOperation = RTC_STOREOPERATION_RESET;
  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
  {
    Error_Handler();
  }
  sDate.WeekDay = RTC_WEEKDAY_MONDAY;
  sDate.Month = RTC_MONTH_JANUARY;
  sDate.Date = 0x1;
  sDate.Year = 0x0;

  if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */

  /* USER CODE END RTC_Init 2 */

}
```

```c
/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}
```

```
/**
  * @brief TIM2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM2_Init(void)
{

  /* USER CODE BEGIN TIM2_Init 0 */

  /* USER CODE END TIM2_Init 0 */

  TIM_Encoder_InitTypeDef sConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM2_Init 1 */

  /* USER CODE END TIM2_Init 1 */
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 0;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 4294967295;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
  sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC1Filter = 0;
  sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
  sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
  sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
  sConfig.IC2Filter = 0;
  if (HAL_TIM_Encoder_Init(&htim2, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
```

```
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM2_Init 2 */

  /* USER CODE END TIM2_Init 2 */

}

/**
  * @brief TIM3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM3_Init(void)
{

  /* USER CODE BEGIN TIM3_Init 0 */

  /* USER CODE END TIM3_Init 0 */

  TIM_SlaveConfigTypeDef sSlaveConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM3_Init 1 */

  /* USER CODE END TIM3_Init 1 */
  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 0;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 2;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }
  sSlaveConfig.SlaveMode = TIM_SLAVEMODE_EXTERNAL1;
  sSlaveConfig.InputTrigger = TIM_TS_ETRF;
```

```
  sSlaveConfig.TriggerPolarity = TIM_TRIGGERPOLARITY_NONINVERTED;
  sSlaveConfig.TriggerPrescaler = TIM_TRIGGERPRESCALER_DIV1;
  sSlaveConfig.TriggerFilter = 0;
  if (HAL_TIM_SlaveConfigSynchro(&htim3, &sSlaveConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
```

```
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * @brief USART3 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART3_UART_Init(void)
{

  /* USER CODE BEGIN USART3_Init 0 */

  /* USER CODE END USART3_Init 0 */

  /* USER CODE BEGIN USART3_Init 1 */

  /* USER CODE END USART3_Init 1 */
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart3) != HAL_OK)
```

```
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART3_Init 2 */

  /* USER CODE END USART3_Init 2 */

}

/**
  * @brief USART6 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART6_UART_Init(void)
{

  /* USER CODE BEGIN USART6_Init 0 */

  /* USER CODE END USART6_Init 0 */

  /* USER CODE BEGIN USART6_Init 1 */

  /* USER CODE END USART6_Init 1 */
  huart6.Instance = USART6;
  huart6.Init.BaudRate = 115200;
  huart6.Init.WordLength = UART_WORDLENGTH_8B;
  huart6.Init.StopBits = UART_STOPBITS_1;
  huart6.Init.Parity = UART_PARITY_NONE;
  huart6.Init.Mode = UART_MODE_TX_RX;
  huart6.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart6.Init.OverSampling = UART_OVERSAMPLING_16;
  huart6.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart6.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart6) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART6_Init 2 */
```

```
  /* USER CODE END USART6_Init 2 */

}

/**
  * @brief USB_OTG_FS Initialization Function
  * @param None
  * @retval None
  */
static void MX_USB_OTG_FS_USB_Init(void)
{

  /* USER CODE BEGIN USB_OTG_FS_Init 0 */

  /* USER CODE END USB_OTG_FS_Init 0 */

  /* USER CODE BEGIN USB_OTG_FS_Init 1 */

  /* USER CODE END USB_OTG_FS_Init 1 */
  /* USER CODE BEGIN USB_OTG_FS_Init 2 */

  /* USER CODE END USB_OTG_FS_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
```

```
  __HAL_RCC_GPIOG_CLK_ENABLE();

  /* Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(SD_CS_GPIO_Port, SD_CS_Pin, GPIO_PIN_RESET);

  /* Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LD1_Pin|LD3_Pin|LD2_Pin, GPIO_PIN_RESET);

  /* Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(USB_PowerSwitchOn_GPIO_Port, USB_PowerSwitchOn_Pin,
                    GPIO_PIN_RESET);

  /* Configure GPIO pin : USER_Btn_Pin */
  GPIO_InitStruct.Pin = USER_Btn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USER_Btn_GPIO_Port, &GPIO_InitStruct);

  /* Configure GPIO pin : SD_CS_Pin */
  GPIO_InitStruct.Pin = SD_CS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(SD_CS_GPIO_Port, &GPIO_InitStruct);

  /* Configure GPIO pins : LD1_Pin LD3_Pin LD2_Pin */
  GPIO_InitStruct.Pin = LD1_Pin|LD3_Pin|LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /* Configure GPIO pin : RMII_TXD1_Pin */
  GPIO_InitStruct.Pin = RMII_TXD1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
  HAL_GPIO_Init(RMII_TXD1_GPIO_Port, &GPIO_InitStruct);
```

```
  /*Configure GPIO pin : USB_PowerSwitchOn_Pin */
  GPIO_InitStruct.Pin = USB_PowerSwitchOn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(USB_PowerSwitchOn_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : USB_OverCurrent_Pin */
  GPIO_InitStruct.Pin = USB_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USB_OverCurrent_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : USB_SOF_Pin USB_ID_Pin USB_DM_Pin USB_DP_Pin */
  GPIO_InitStruct.Pin = USB_SOF_Pin|USB_ID_Pin|USB_DM_Pin|USB_DP_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : USB_VBUS_Pin */
  GPIO_InitStruct.Pin = USB_VBUS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USB_VBUS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : RMII_TX_EN_Pin RMII_TXD0_Pin */
  GPIO_InitStruct.Pin = RMII_TX_EN_Pin|RMII_TXD0_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 6, 6);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

}
```

```
/* USER CODE BEGIN 4 */
/**
 * Task A periodically 'gives' semaphorePtr
 * NOTES:
 * - This semaphore isn't "given" to any task specifically
 * - giving the semaphore doesn't prevent taskA from continuing to run.
 *   Notice the green LED continues to blink at all times
 */
void USR_GPIO_Init(void)
{
        GPIO_InitTypeDef GPIO_InitStruct = {0};

        /*Configure GPIO pins : PG2 PG3 */
          GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;
          GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
          GPIO_InitStruct.Pull = GPIO_PULLUP;
          HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

          /* EXTI interrupt init*/
            HAL_NVIC_SetPriority(EXTI2_IRQn, 0, 0);
            HAL_NVIC_EnableIRQ(EXTI2_IRQn);

            HAL_NVIC_SetPriority(EXTI3_IRQn, 0, 0);
            HAL_NVIC_EnableIRQ(EXTI3_IRQn);
}

void GreenTaskA( void* argument )
{
        uint_fast8_t count = 0;
        while(1)
        {
                //every 5 times through the loop, give the semaphore
                if(++count >= 5)
                {
                        count = 0;
                        //SEGGER_SYSVIEW_PrintfHost("Task A (green LED) gives se
                        xSemaphoreGive(semPtr);

                }
```

```
                HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_SET);
                vTaskDelay(100/portTICK_PERIOD_MS);
                HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
                vTaskDelay(100/portTICK_PERIOD_MS);
        }
}

/**
 * wait to receive semPtr and triple blink the Blue LED
 */
void BlueTaskB( void* argument )
{
        while(1)
        {
                //'take' the semaphore with a really long timeout
                //SEGGER_SYSVIEW_PrintfHost("Task B (Blue LED) attempts to take
                if(xSemaphoreTake(semPtr, portMAX_DELAY) == pdPASS)
                {
                        //SEGGER_SYSVIEW_PrintfHost("Task B (Blue LED) received
                        //triple blink the Blue LED
                        for(uint_fast8_t i = 0; i < 3; i++)
                        {
                                HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_P
                                vTaskDelay(50/portTICK_PERIOD_MS);
                                HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_P
                                vTaskDelay(50/portTICK_PERIOD_MS);
                        }
                }
//              else
//              {
//                      This is the code that will be executed if we time out wa
//                      the semaphore to be given.  In the case of a 1 mS tick r
//                      will only provide a delay of around 50 days.
//                      Unless "#define INCLUDE_vTaskSuspend 1" is configured in
//              }
        }
}

void DataLogTask( void* argumet )
{
```

```
        //RTC_DateTypeDef GetDate;  //Get date structure
        //RTC_TimeTypeDef GetTime;   //Get time structure
        char buf[100] = {0};
        int len;
        int distance;
        int seg;
        int32_t heading;
          while(1)
          {
                  if(xSemaphoreTake(DLsemPtr, portMAX_DELAY) == pdPASS)
                  {
                          HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET

                          //distance = encoder_cnt;
                          distance = TIM2->CNT;
                          seg = segment;
                          heading = RELPOS_data.relPosHeading;


                          //SEGGER_SYSVIEW_PrintfHost("PVT received");

                          len = sprintf(buf,"%i, %i, %li, %02i:%02i:%02i, %ld, %
        seg,distance,heading,PVT_Data.hour,PVT_Data.min,PVT_Data.sec, PVT_Data
        PVT_Data.lon,PVT_Data.hMSL,PVT_Data.numSV, PVT_Data.fixType);

                          HAL_UART_Transmit(&huart3, buf, len, 200);
                          //SEGGER_SYSVIEW_PrintfHost(buf);
                          //HAL_Delay(5);  //Simulate longer time
                          HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RES
                  }
          }

}
void myprintf(const char *fmt, ...) {
  va_list args;
  va_start(args, fmt);
  vsnprintf(buffer, sizeof(buffer), fmt, args);
  va_end(args);

  int len = strlen(buffer);
```

```
  HAL_UART_Transmit(&huart3, (uint8_t*)buffer, len, 1000);

}

void ProcessUBXTask( void* NotUsed)
{
        char nextByte;
        char buf[50] ={0};

        startReceiveInt();

        while(1)
        {
                xQueueReceive(uart2_BytesReceived, &nextByte, portMAX_DELAY);
                //SEGGER_SYSVIEW_PrintfHost("%c", nextByte);
                UBX_ProcessBuffer(nextByte, &packetBuf);
                if(PVT_Data.PVTreceived == 1){
                        //SEGGER_SYSVIEW_PrintfHost("PVT received");
                        xSemaphoreGive(DLsemPtr);
                        PVT_Data.PVTreceived = 0; //reset flag
                }
                if(TIMEUTC_data.TIMEUTC_received == 1){
                        //SEGGER_SYSVIEW_PrintfHost("TIMEUTC received");
                        xSemaphoreGive(TIMEsemPtr);
                        TIMEUTC_data.TIMEUTC_received = 0; //reset flag
                }
                if(RELPOS_data.RELPOS_received == 1){
                        //SEGGER_SYSVIEW_PrintfHost("RELPOS received");
                        RELPOS_data.RELPOS_received = 0;
                }


        }
}
void startReceiveInt( void )
{
        rxInProgress = true;
        USART2->CR3 |= USART_CR3_EIE;    //enable error interrupts
        USART2->CR1 |= (USART_CR1_UE | USART_CR1_RXNEIE);
        //all 4 bits are for preemption priority -
```

```
        NVIC_SetPriority(USART2_IRQn, 6);
        NVIC_EnableIRQ(USART2_IRQn);
}
void startReceiveInt3( void )
{
        rxInProgress = true;
        USART3->CR3 |= USART_CR3_EIE;    //enable error interrupts
        USART3->CR1 |= (USART_CR1_UE | USART_CR1_RXNEIE);
        //all 4 bits are for preemption priority -
        NVIC_SetPriority(USART3_IRQn, 6);
        NVIC_EnableIRQ(USART3_IRQn);
}

void USART2_IRQHandler( void )
{
        portBASE_TYPE xHigherPriorityTaskWoken = pdFALSE;
        //SEGGER_SYSVIEW_RecordEnterISR();

        //first check for errors
        if(     USART2->ISR & ( USART_ISR_ORE_Msk |
                                USART_ISR_NE_Msk |
                                USART_ISR_FE_Msk |
                                USART_ISR_PE_Msk ))
        {
                //clear error flags
                USART2->ICR |= (USART_ICR_FECF |
                                USART_ICR_PECF |
                                USART_ICR_NCF |
                                USART_ICR_ORECF);
        }

        if(     USART2->ISR & USART_ISR_RXNE_Msk)
        {
                //read the data register unconditionally to clear
                //the receive not empty interrupt if no reception is
                //in progress
                uint8_t tempVal = (uint8_t) USART2->RDR;

                if(rxInProgress)
                {
```

```
                        xQueueSendFromISR(uart2_BytesReceived, &tempVal, &xHighe
                }
        }
        //SEGGER_SYSVIEW_RecordExitISR();
        portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}

void USART3_IRQHandler( void )
{
        portBASE_TYPE xHigherPriorityTaskWoken = pdFALSE;
        //SEGGER_SYSVIEW_RecordEnterISR();

        //first check for errors
        if(     USART3->ISR & ( USART_ISR_ORE_Msk |
                                        USART_ISR_NE_Msk |
                                        USART_ISR_FE_Msk |
                                        USART_ISR_PE_Msk ))
        {
                //clear error flags
                USART3->ICR |= (USART_ICR_FECF |
                                        USART_ICR_PECF |
                                        USART_ICR_NCF |
                                        USART_ICR_ORECF);
        }

        if(     USART3->ISR & USART_ISR_RXNE_Msk)
        {
                //read the data register unconditionally to clear
                //the receive not empty interrupt if no reception is
                //in progress
                uint8_t tempVal = (uint8_t) USART3->RDR;

                if(rxInProgress)
                {
                        xQueueSendFromISR(uart3_BytesReceived, &tempVal, &xHighe
                }
        }
        //SEGGER_SYSVIEW_RecordExitISR();
        portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}
```

```
/* An attempt to use HAL instead of low level code.
void ProcessUBXTask( void* NotUsed)
{
        char nextByte;
        char buf[50] ={0};

        HAL_UART_Receive_IT (&huart2, nextByte, 1);

        while(1)
        {
                xQueueReceive(uart2_BytesReceived, &nextByte, portMAX_DELAY);
                SEGGER_SYSVIEW_PrintfHost("%c", nextByte);
                UBX_ProcessBuffer(nextByte, &packetBuf);
                if(PVT_Data.PVTreceived == 1){
                        SEGGER_SYSVIEW_PrintfHost("PVT received");
                        xSemaphoreGive(DLsemPtr);
                        PVT_Data.PVTreceived = 0; //reset flag
                }
                if(TIMEUTC_data.TIMEUTC_received == 1){
                        SEGGER_SYSVIEW_PrintfHost("TIMEUTC received");
                        xSemaphoreGive(TIMEsemPtr);
                        TIMEUTC_data.TIMEUTC_received = 0; //reset flag
                                }


        }
}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
        portBASE_TYPE xHigherPriorityTaskWoken = pdFALSE;
        SEGGER_SYSVIEW_RecordEnterISR();
    HAL_UART_Receive_IT(&huart2, nextByte, 1);
    xQueueSendFromISR(uart2_BytesReceived, &tempVal, &xHigherPriorityTaskWoken);
    SEGGER_SYSVIEW_RecordExitISR();
    portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
}
*/
void RTC_UpdateTask( void* argument )
{
```

```
RTC_DateTypeDef sdatestructure;
RTC_TimeTypeDef stimestructure;

uint8_t getTIMEUTC[]={0xB5,0x62,0x01,0x21,0x00,0x00,0x22,0x67};


while (1) {


        HAL_UART_Transmit(&huart2, getTIMEUTC, 8, 100);  //Transmit requ
        //SEGGER_SYSVIEW_PrintfHost("Transmitted getTIME");


        if(xSemaphoreTake(TIMEsemPtr, 2000/portTICK_PERIOD_MS) == pdPASS
        {
                //SEGGER_SYSVIEW_PrintfHost("TIME received");
                //##-1- Configure the Date
                sdatestructure.Year = (uint8_t) (TIMEUTC_data.year - 200
                sdatestructure.Month = TIMEUTC_data.month;
                sdatestructure.Date = TIMEUTC_data.day;

                if (HAL_RTC_SetDate(&hrtc, &sdatestructure, RTC_FORMAT_B
                {
                        Error_Handler();
                }

                //##-2- Configure the Time
                stimestructure.Hours =  TIMEUTC_data.hour;
                stimestructure.Minutes =  TIMEUTC_data.min;
                stimestructure.Seconds = TIMEUTC_data.sec;
                stimestructure.TimeFormat = RTC_HOURFORMAT_24;
                stimestructure.DayLightSaving = RTC_DAYLIGHTSAVING_NONE
                stimestructure.StoreOperation = RTC_STOREOPERATION_RESET

                if (HAL_RTC_SetTime(&hrtc, &stimestructure, RTC_FORMAT_B
                {
                        Error_Handler();
                }
                HAL_RTCEx_BKUPWrite(&hrtc,RTC_BKP_DR0,0x32F6);
```

```
                        HAL_RTC_GetTime(&hrtc, &stimestructure, RTC_FORMAT_BIN);
                        char buf[30]={0};
                        int len = sprintf(buf,"RTC Time: %02d:%02d:%02d\r\n",sti
        stimestructure.Minutes, stimestructure.Seconds);
                        HAL_UART_Transmit(&huart3, buf, len, 100);
                        //SEGGER_SYSVIEW_PrintfHost(buf);

                        //SEGGER_SYSVIEW_PrintfHost("TIME updated");
                        vTaskDelete(NULL);
                        }

                vTaskDelay(5000/portTICK_PERIOD_MS);
        }
}

// EXTI Line15 External Interrupt ISR Handler CallBackFun
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13) // If The INT Source Is EXTI Line15 (B13 Pin)
    {
        portBASE_TYPE xHigherPriorityTaskWoken = pdFALSE;
        //SEGGER_SYSVIEW_RecordEnterISR();

        ++segment;
        if (segment >3) segment = 0;
        //SEGGER_SYSVIEW_PrintfHost("Segment: %i",segment);
        TIM2->CNT = 0;
        encoder_cnt = 0;
        //SEGGER_SYSVIEW_RecordExitISR();
        portYIELD_FROM_ISR(xHigherPriorityTaskWoken);
    }
    else if(GPIO_Pin == GPIO_PIN_2)
    {
        //++encoder_cnt;
    }
    else if(GPIO_Pin == GPIO_PIN_3)
    {
        ++segment;
        if (segment >3) segment = 0;
        encoder_cnt = 0;
```

```
        TIM2->CNT = 0;
    }
}

void UserInputTask( void* argument )
{
        char nextByte;
        char buf[50] ={0};

        startReceiveInt3();

        while(1)
        {
                xQueueReceive(uart3_BytesReceived, &nextByte, portMAX_DELAY);
                //SEGGER_SYSVIEW_PrintfHost("%c", nextByte);
                if(nextByte == 't' || nextByte == 'T')
                {

                        int len = sprintf(buf,"Updating RTC Clock\r\n");
                        HAL_UART_Transmit(&huart3, buf, len, 100);
                        //SEGGER_SYSVIEW_PrintfHost("Starting RTC task");
                        assert_param(xTaskCreate(RTC_UpdateTask, "RTC_Update", S
        tskIDLE_PRIORITY + 3, NULL) == pdPASS);
                }
        }
}
/* USER CODE END 4 */

/**
  * @brief  Period elapsed callback in non blocking mode
  * @note   This function is called  when TIM6 interrupt took place, inside
  * HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment
  * a global variable "uwTick" used as application time base.
  * @param  htim : TIM handle
  * @retval None
  */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* USER CODE BEGIN Callback 0 */
```

```
  /* USER CODE END Callback 0 */
  if (htim->Instance == TIM6) {
    HAL_IncTick();
  }
  /* USER CODE BEGIN Callback 1 */

  /* USER CODE END Callback 1 */
}

/**
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

```
/********************** (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Appendix E

MATLAB Analysis Code

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% %%%%%%%%%%%%%%%%%% Week 12 Sample Matlab Code %%%%%%%%%%%%%%%%%%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% File       Name : Analysis.m
%% Authors    Name : Simon Castles
%% Email Address : u1083859@umail.usq.edu.au
%% Environment    : Matlab R2021b
%% Log of Change :2021 S2, Initial Version
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; close all; % clear all variables from workspace & close all figures
clc; % clear command window
figcnt = 0; % count of figure



M=importdata('Test_File.txt');
encoder = str2double(M.textdata(:,2));
heading = str2double(M.textdata(:,3))./100000;
Latitude = M.data(:,1)./10000000;
Longitude = M.data(:,2)./10000000;
h_msl = M.data(:,3)./1000;
h_msl_bar = mean(h_msl);
h = h_msl_bar + 42.231;

%WGS84 Conversion factors
a = 6378137; %Semi-major axis
```

```
b = 6356752.3142; %Semi-minor axis
phi = mean(Latitude);

F_lat = pi/180 * ((a^2 / (sqrt(a^2 * (cosd(phi))^2 + b^2 * (sind(phi))^2 ))) + h
    * cosd(phi);
F_lon = pi/180 * (((a^2 * b^2) / ( ( a^2 * (cosd(phi))^2 + b^2 * (sind(phi))^2 )
    ) ) + h);

Lat_adj = Latitude.*F_lat;  %Adjusted latitude
Lon_adj = Longitude.*F_lon; %Adjusted logitude

figcnt=figcnt+1;
figure(figcnt);
fprintf('Please see Figure %d for trajectory\n',figcnt);
plot(Longitude,Latitude,'b.-'); grid minor; axis equal;
xlabel('Longitude (deg)'),ylabel('Latitude (deg)')
title('Trajectory')
print -deps Trajectory

figcnt=figcnt+1;
figure(figcnt);
fprintf('Please see Figure %d for adjusted trajectory\n',figcnt);
plot(Lon_adj,Lat_adj,'b.-'); grid minor; axis equal;
xlabel('Longitude (m)'),ylabel('Latitude (m)')
title('Adjusted trajectory')
print -deps Adjusted_trajectory

figcnt=figcnt+1;
figure(figcnt);
fprintf('Please see Figure %d for heading data\n',figcnt);
plot(heading);
title('Recorded heading')
xlabel('Sample'); ylabel('Heading (deg)')
print -deps Recorded_heading

hold on;

corrected_heading = zeros(length(heading),1);
for k=1:length(heading)
    if heading(k)> 270
```

```
        corrected_heading(k) = heading(k) - 270;
    else
        corrected_heading(k) = heading(k) + 90;
    end
end
plot(corrected_heading)

time = 0:0.2:(length(corrected_heading)*0.2);
figcnt=figcnt+1;
figure(figcnt);
plot(time(1:length(corrected_heading)),corrected_heading);
xlabel('Time (s)'); ylabel('Heading (deg)')
title('Corrected heading data')
print -deps Corrected_heading

xline(9/5-0.2,'--r');
xline(59/5-0.2,'--r');
xline(247/5-0.2,'--r');
xline(297/5-0.2,'--r');

%% Windows

SW1 = 9:59; SW2 = 247:297; SW3 = 485:535; SW4 = 723:773;
SW5 = 960:1010; SW6 = 1198:1248; SW7 = 1436:1486; SW8 = 1674:1724;
SW9 = 1911:1962; SW10 = 2150:2200; SW11 = 2387:2437; SW12 = 2625:2675;
SW13 = 2862:2912; SW14 = 3099:3150; SW15 = 3337:3388; SW16 = 3575:3625;
SW17 = 3813:3863; SW18 = 4050:4100; SW19 = 4287:4338; SW20 = 4525:4575;
SW21 = 4763:4813; SW22 = 5000:5050;

SW = [SW1,SW2,SW3,SW4,SW5,SW6,SW7,SW8,SW9,SW10,SW11,SW12,SW13,SW14,SW15,...
    SW16,SW17,SW18,SW19,SW20,SW21,SW22];

NE1 = 128:178; NE2 = 367:417; NE3 = 605:655; NE4 = 842:892;
NE5 = 1079:1129; NE6 = 1317:1367; NE7 = 1555:1605; NE8 = 1793:1843;
NE9 = 2030:2080; NE10 = 2268:2318; NE11 = 2506:2556; NE12 = 2744:2794;
NE13 = 2981:3031; NE14 = 3218:3269; NE15 = 3456:3506; NE16 = 3694:3744;
NE17 = 3932:3982; NE18 = 4169:4219; NE19 = 4406:4456; NE20 = 4644:4694;
NE21 = 4882:4932; NE22 = 5119:5169;

NE = [NE1,NE2,NE3,NE4,NE5,NE6,NE7,NE8,NE9,NE10,NE11,NE12,NE13,NE14,NE15,...
```

```
    NE16,NE17,NE18,NE19,NE20,NE21,NE22];

%% Southwest passes

%y = [Lat_adj(9:59);Lat_adj(247:297);Lat_adj(485:535)];
%x = [Lon_adj(9:59);Lon_adj(247:297);Lon_adj(485:535)];
y = Lat_adj(SW);
x = Lon_adj(SW);

[p,S] = polyfit(x,y,1);
[y_fit,delta] = polyval(p,x,S);
figcnt=figcnt+1;
figure(figcnt);
plot(x,y,'b.')
hold on
plot(x,y_fit,'r-')
plot(x,y_fit+2*delta,'m--',x,y_fit-2*delta,'m--')
title('Southwest passes GNSS Data')
legend('GNSS data points','Linear Fit','95% Percentile')
xlabel('Adjusted latitude (m)');ylabel('Adjusted longitude (m)');
grid minor; axis square;
print -deps SW1

m = p(1);
c = p(2);
txt = ['y = ' num2str(m) 'x' num2str(c)];
text(16838128,-2725330,txt);

d=(m.*x-y+c)/sqrt(1+m^2);

r_count = 0;
l_count = 0;
zero_count = 0;
for k=1:length(d)
    if d(k)<0
        r_count = r_count + 1;
    elseif d(k) > 0
        l_count = l_count + 1;
    elseif d(k) == 0
        zero_count = zero_count +1;
```

```
    end

end
l_count;
r_count;
zero_count;



d_abs_SW = abs((m.*x-y+c)/sqrt(1+m^2));

d_mean_SW = mean(d_abs_SW);
d_std_SW = std(d_abs_SW);
d_max_SW = max(d_abs_SW);
d_95_SW = prctile(d_abs_SW,95);


ACC_SW = sqrt(2)*(d_mean_SW + d_std_SW);
fprintf('SW pass left count: %u, right count: %u, zero count: %u\n',...
    l_count,r_count,zero_count)
fprintf('SW pass accuracy (mm): %f\n',ACC_SW*1000)

%% Northeast Passes
y = Lat_adj(NE);
x = Lon_adj(NE);

[p,S] = polyfit(x,y,1);
[y_fit,delta] = polyval(p,x,S);
figcnt=figcnt+1;
figure(figcnt);
plot(x,y,'b.')
hold on
plot(x,y_fit,'r-')
plot(x,y_fit+2*delta,'m--',x,y_fit-2*delta,'m--')
title('Northeast pases GNSS Data')
legend('GNSS data points','Linear Fit','95% Percentile')
xlabel('Adjusted latitude (m)');ylabel('Adjusted longitude (m)');
grid minor; axis square;
print -deps NE1
```

```
m = p(1);
c = p(2);
txt = ['y = ' num2str(m) 'x' num2str(c)];
text(16838125,-2725325.5,txt);

d=(m.*x-y+c)/sqrt(1+m^2);
r_count = 0;
l_count = 0;
zero_count = 0;
for k=1:length(d)
    if d(k)<0
        r_count = r_count + 1;
    elseif d(k) > 0
        l_count = l_count + 1;
    elseif d(k) == 0
        zero_count = zero_count +1;
    end

end
l_count;
r_count;
zero_count;

d_abs_NE = abs((m.*x-y+c)/sqrt(1+m^2));

d_mean_NE = mean(d_abs_NE);
d_std_NE = std(d_abs_NE);
d_max_NE = max(d_abs_NE);
d_95_NE = prctile(d_abs_NE,95);

ACC_NE = sqrt(2)*(d_mean_NE + d_std_NE);

fprintf('NE pass left count: %u, right count: %u, zero count: %u\n',...
    l_count,r_count,zero_count)
fprintf('NE pass accuracy (mm): %f\n',ACC_NE*1000)
%% Overall relative position accuracy

d = [d_abs_SW;d_abs_NE];
d_mean = mean(d);
d_std = std(d);
```

```
d_max = max(d);
d_95 = prctile(d,95);
ACC = sqrt(2) * (d_mean + d_std);
fprintf('Total relative cross-track accuracy (mm): %f\n',ACC*1000)

RMSE = sqrt(mean(d.^2));
%% Speed analysis

speed = zeros(length(encoder)-1,1);

for k=1:(length(encoder)-1)
    speed(k) = (encoder(k+1)-encoder(k)) / 0.2 * pi * 0.04 / 8192;
end
time = 0:0.2:(length(speed)*0.2);
figcnt=figcnt+1;
figure(figcnt);
t = time(1:length(speed));
plot(t,speed,'b.')
grid minor;
title('Speed')
xlabel('time (s)');ylabel('speed (m/s)'); axis([0 1055 0 1.2]);

[p, S] = polyfit(t,speed,1);
[y_fit,delta] = polyval(p,t,S);
hold on
plot(t,y_fit,'r-')
plot(t,y_fit+2*delta,'m--',t,y_fit-2*delta,'m--')
legend('calculated speed','linear fit','95th percentile')
print -deps Speed1

s_mean = mean(speed);
s_std = std(speed);
s_delta = abs(speed-s_mean);

% yline(s_mean,'r')
% yline(s_mean+2*s_std,'--r')
% yline(s_mean-2*s_std,'--r')

t2 = 1:5:length(encoder);
s2 = zeros(length(t2)-1,1);
```

```
for k=1:(length(t2)-1)
    s2(k) = (encoder(t2(k+1))-encoder(t2(k))) * pi * 0.04 / 8192;
end

figcnt=figcnt+1;
figure(figcnt);
subplot(2,1,1);
plot(0:length(s2)-1,s2,'b-')
title('Average speed')
xlabel('time (s)');ylabel('speed (m/s)'); axis([0 1055 0 1.2]);
yline(0.6283,'r-')
legend('Derived encoder speed','Commanded speed (0.628 m/s)')
sc = 0.6283;
s_error = sc - s2;
s_error_abs = abs(s_error);
subplot(2,1,2);
plot(0:length(s_error)-1,s_error)
title('Speed error')
xlabel('time (s)');ylabel('error (m/s)'); axis([0 1055 -0.1 0.1]);
s_error_mean = mean(s_error_abs);
s_error_std = std(s_error_abs);
print -deps Speed2
%%
time = 0:0.2:(length(Lat_adj)*0.2);
figcnt=figcnt+1;
figure(figcnt);
yyaxis left
plot(time(1:length(Lat_adj)),Lat_adj)
ylabel('Latitude (m)');
yyaxis right
plot(time(1:length(Lat_adj)),Lon_adj)
xlabel('time (s)')
ylabel('Longitude (m)')



%% Southwest pass by pass error
x=[Lon_adj(SW1),Lon_adj(SW2),Lon_adj(SW3),Lon_adj(SW4),Lon_adj(SW5),...
    Lon_adj(SW6),Lon_adj(SW7),Lon_adj(SW8),Lon_adj(SW10),Lon_adj(SW11),...
    Lon_adj(SW12),Lon_adj(SW13),Lon_adj(SW16),Lon_adj(SW17),Lon_adj(SW18)...
```

```
    ,Lon_adj(SW20),Lon_adj(SW21),Lon_adj(SW22)];
y=[Lat_adj(SW1),Lat_adj(SW2),Lat_adj(SW3),Lat_adj(SW4),Lat_adj(SW5),...
    Lat_adj(SW6),Lat_adj(SW7),Lat_adj(SW8),Lat_adj(SW10),Lat_adj(SW11),...
    Lat_adj(SW12),Lat_adj(SW13),Lat_adj(SW16),Lat_adj(SW17),Lat_adj(SW18)...
    ,Lat_adj(SW20),Lat_adj(SW21),Lat_adj(SW22)];
p = polyfit(x,y,1);
d2=(p(1).*x-y+p(2))/sqrt(1+p(1)^2);
d2_mean = mean(d2')';
d2_std = std(d2);
% [p, S] = polyfit(1:length(d2_mean),d2_mean,5);
% [f, D] = polyval(p,1:length(d2_mean),S);

d2_sample = [(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',...
    (1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:5
[p, S] = polyfit(d2_sample,d2,5);
[f, D] = polyval(p,1:length(d2_mean),S);

figcnt=figcnt+1;
figure(figcnt);
subplot(2,1,1);
plot(d2,'.-')
hold on
yline(0)

p1 = plot(f,'r');
p2 = plot(f+D*2,'r--');
plot(f-D*2,'r--')

title('Distribution of GNSS data errors for each southwest pass')
legend([p1 p2],'5th order polynomial fit', '95th percentile')
xlabel('Sample');ylabel('Distance from linear fit (m)');
grid minor; axis([0 52 -0.025 0.025]);

d3_abs_mean = mean(mean(abs(d2-d2_mean)));
d3_abs_std = mean(std(abs(d2-d2_mean)));
d3_acc = sqrt(2) * (d3_abs_mean + d3_abs_std) * 1000;

d4_mean = mean(mean(abs(d2-f')));
d4_std = mean(std(abs(d2-f')));
d4_acc = sqrt(2) * (d4_mean + d4_std) * 1000;
```

```
%% Northeast pass by pass error
x=[Lon_adj(NE1),Lon_adj(NE2),Lon_adj(NE3),Lon_adj(NE4),Lon_adj(NE5),...
    Lon_adj(NE6),Lon_adj(NE7),Lon_adj(NE8),Lon_adj(NE10),Lon_adj(NE11),...
    Lon_adj(NE12),Lon_adj(NE13),Lon_adj(NE16),Lon_adj(NE17),Lon_adj(NE18)...
    ,Lon_adj(NE20),Lon_adj(NE21),Lon_adj(NE22)];
y=[Lat_adj(NE1),Lat_adj(NE2),Lat_adj(NE3),Lat_adj(NE4),Lat_adj(NE5),...
    Lat_adj(NE6),Lat_adj(NE7),Lat_adj(NE8),Lat_adj(NE10),Lat_adj(NE11),...
    Lat_adj(NE12),Lat_adj(NE13),Lat_adj(NE16),Lat_adj(NE17),Lat_adj(NE18)...
    ,Lat_adj(NE20),Lat_adj(NE21),Lat_adj(NE22)];
p = polyfit(x,y,1);
d2=(p(1).*x-y+p(2))/sqrt(1+p(1)^2);
d2_mean = mean(d2')';
d2_std = std(d2);
% [p, S] = polyfit(1:length(d2_mean),d2_mean,5);
% [f, D] = polyval(p,1:length(d2_mean),S);

d2_sample = [(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',...
    (1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:51)',(1:5
[p, S] = polyfit(d2_sample,d2,5);
[f, D] = polyval(p,1:length(d2_mean),S);

%figcnt=figcnt+1;
%figure(figcnt);
subplot(2,1,2);
plot(d2,'.-')
hold on
yline(0)

p1 = plot(f,'r');
p2 = plot(f+D*2,'r--');
plot(f-D*2,'r--')

title('Distribution of GNSS data errors for each northeast pass')
legend([p1 p2],'5th order polynomial fit', '95th percentile')
xlabel('Sample');ylabel('Distance from linear fit (m)');
grid minor; axis([0 52 -0.025 0.025]);

print -deps Error
```

```
d3_abs_mean = mean(mean(abs(d2-d2_mean)));
d3_abs_std = mean(std(abs(d2-d2_mean)));
d3_acc = sqrt(2) * (d3_abs_mean + d3_abs_std) * 1000;

d4_mean = mean(mean(abs(d2-f')));
d4_std = mean(std(abs(d2-f')));
d4_acc = sqrt(2) * (d4_mean + d4_std) * 1000;

%% Southwest passes

%y = [Lat_adj(9:59);Lat_adj(247:297);Lat_adj(485:535)];
%x = [Lon_adj(9:59);Lon_adj(247:297);Lon_adj(485:535)];
y = Lat_adj(SW) - Lat_adj(SW(end));
x = Lon_adj(SW) - Lon_adj(SW(end));

[p,S] = polyfit(x,y,1);
[y_fit,delta] = polyval(p,x,S);
figcnt=figcnt+1;
figure(figcnt);
plot(x,y,'b.')
hold on
plot(x,y_fit,'r-')
plot(x,y_fit+2*delta,'m--',x,y_fit-2*delta,'m--')
title('Southwest passes GNSS Data')
legend('GNSS data points','Linear Fit','95% Percentile')
xlabel('Adjusted local latitude (m)');ylabel('Adjusted local longitude (m)');
grid minor; axis equal;
print -deps SW2

m = p(1);
c = p(2);
txt = ['y = ' num2str(m) 'x' num2str(c)];
text(2,1,txt);

d=(m.*x-y+c)/sqrt(1+m^2);

r_count = 0;
l_count = 0;
zero_count = 0;
for k=1:length(d)
```

```
    if d(k)<0
        r_count = r_count + 1;
    elseif d(k) > 0
        l_count = l_count + 1;
    elseif d(k) == 0
        zero_count = zero_count +1;
    end

end
l_count;
r_count;
zero_count;



d_abs_SW = abs((m.*x-y+c)/sqrt(1+m^2));

d_mean_SW = mean(d_abs_SW);
d_std_SW = std(d_abs_SW);
d_max_SW = max(d_abs_SW);
d_95_SW = prctile(d_abs_SW,95);


ACC_SW = sqrt(2)*(d_mean_SW + d_std_SW);
fprintf('SW pass left count: %u, right count: %u, zero count: %u\n',...
    l_count,r_count,zero_count)
fprintf('SW pass accuracy (mm): %f\n',ACC_SW*1000)

%% Northeast Passes
y = Lat_adj(NE) - Lat_adj(NE(1));
x = Lon_adj(NE) - Lon_adj(NE(1));

[p,S] = polyfit(x,y,1);
[y_fit,delta] = polyval(p,x,S);
figcnt=figcnt+1;
figure(figcnt);
plot(x,y,'b.')
hold on
plot(x,y_fit,'r-')
plot(x,y_fit+2*delta,'m--',x,y_fit-2*delta,'m--')
```

```
title('Northeast pases GNSS Data')
legend('GNSS data points','Linear Fit','95% Percentile')
xlabel('Adjusted local latitude (m)');ylabel('Adjusted local longitude (m)');
grid minor; axis equal;
print -deps NE2

m = p(1);
c = p(2);
txt = ['y = ' num2str(m) 'x' num2str(c)];
text(2,1,txt);

d=(m.*x-y+c)/sqrt(1+m^2);
r_count = 0;
l_count = 0;
zero_count = 0;
for k=1:length(d)
    if d(k)<0
        r_count = r_count + 1;
    elseif d(k) > 0
        l_count = l_count + 1;
    elseif d(k) == 0
        zero_count = zero_count +1;
    end

end
l_count;
r_count;
zero_count;

d_abs_NE = abs((m.*x-y+c)/sqrt(1+m^2));

d_mean_NE = mean(d_abs_NE);
d_std_NE = std(d_abs_NE);
d_max_NE = max(d_abs_NE);
d_95_NE = prctile(d_abs_NE,95);

ACC_NE = sqrt(2)*(d_mean_NE + d_std_NE);

fprintf('NE pass left count: %u, right count: %u, zero count: %u\n',...
    l_count,r_count,zero_count)
```

```
fprintf('NE pass accuracy (mm): %f\n',ACC_NE*1000)

%% Bar graphs
figcnt=figcnt+1;
figure(figcnt);
groups = categorical({'SW segment', 'NE segment', 'Combined segments'});
groups = reordercats(groups,{'SW segment', 'NE segment', 'Combined segments'});


data = [d_mean_SW d_std_SW ACC_SW; d_mean_NE d_std_NE ACC_NE; d_mean d_std ACC];
b = bar(groups,data);
grid minor;
ylabel('Error / Accuracy (m)')
title('Mean unsigned error, standard deviation of error and accuracy')
legend('Mean of error', 'Standard deviation of error', 'Relative cross-track acc
legend('Location','southoutside')

xtips1 = b(1).XEndPoints;
ytips1 = b(1).YEndPoints;
labels1 = string(b(1).YData);
text(xtips1,ytips1,labels1,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

xtips2 = b(2).XEndPoints;
ytips2 = b(2).YEndPoints;
labels2 = string(b(2).YData);
text(xtips2,ytips2,labels2,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

xtips3 = b(3).XEndPoints;
ytips3 = b(3).YEndPoints;
labels3 = string(b(3).YData);
text(xtips3,ytips3,labels3,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

%% heading accuracy
heading_sw = heading(SW);
heading_sw_m = mean(heading_sw);
heading_sw_std = std(heading_sw);
heading_sw_error = heading_sw - heading_sw_m;
```

```
heading_sw_error_m = mean(abs(heading_sw_error));
heading_sw_error_std = std(abs(heading_sw_error));
heading_sw_error_acc = heading_sw_error_m + heading_sw_error_std;

heading_ne = heading(NE);
heading_ne_m = mean(heading_ne);
heading_ne_std = std(heading_ne);
heading_ne_error = heading_ne - heading_ne_m;
heading_ne_error_m = mean(abs(heading_ne_error));
heading_ne_error_std = std(abs(heading_ne_error));
heading_ne_error_acc = heading_ne_error_m + heading_ne_error_std;

heading_error = [heading_sw_error;heading_ne_error];
heading_error_m = mean(abs(heading_error));
heading_error_std = std(abs(heading_error));
heading_error_acc = heading_error_m + heading_error_std;


figcnt=figcnt+1;
figure(figcnt);
subplot(2,1,1)
plot(heading_sw_error)
xlabel('Sample');ylabel('Error (deg)');title('Heading error during southwest str
subplot(2,1,2)
plot(heading_ne_error)
xlabel('Sample');ylabel('Error (deg)');title('Heading error during northeast str

figcnt=figcnt+1;
figure(figcnt);
groups = categorical({'SW segment', 'NE segment', 'Combined segments'});
groups = reordercats(groups,{'SW segment', 'NE segment', 'Combined segments'});


data = [heading_sw_error_m heading_sw_error_std heading_sw_error_acc; heading_ne
     heading_ne_error_std heading_ne_error_acc; heading_error_m heading_error_st
b = bar(groups,data);
grid minor;
ylabel('Error / Accuracy (deg)')
title('Mean unsigned error, standard deviation of error and accuracy')
legend('Mean', 'Standard deviation', 'Accuracy');
```

```
legend('Location','southoutside')
xtips1 = b(1).XEndPoints;
ytips1 = b(1).YEndPoints;
labels1 = string(b(1).YData);
text(xtips1,ytips1,labels1,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

xtips2 = b(2).XEndPoints;
ytips2 = b(2).YEndPoints;
labels2 = string(b(2).YData);
text(xtips2,ytips2,labels2,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

xtips3 = b(3).XEndPoints;
ytips3 = b(3).YEndPoints;
labels3 = string(b(3).YData);
text(xtips3,ytips3,labels3,'HorizontalAlignment','center',...
    'VerticalAlignment','bottom')

%% Heading errors
heading_sw = heading(SW);
heading_sw_m = mean(heading_sw);

heading_sw_p = [heading(SW1),heading(SW2),heading(SW3),heading(SW4),heading(SW5)
    heading(SW7),heading(SW8),heading(SW10),heading(SW11),heading(SW12),heading(
    heading(SW17),heading(SW18),heading(SW20),heading(SW21),heading(SW22)];

heading_sw_perror = heading_sw_p - heading_sw_m;
subplot(2,1,1)
plot(heading_sw_perror);
title('Distribution of GNSS heading errors for each southwest pass')
%legend([p1 p2],'5th order polynomial fit', '95th percentile')
xlabel('Sample');ylabel('Error from mean heading (deg)');
grid minor; axis([0 52 -1.5 1]);

heading_ne = heading(NE);
heading_ne_m = mean(heading_ne);

heading_ne_p = [heading(NE1),heading(NE2),heading(NE3),heading(NE4),heading(NE5)
    heading(NE7),heading(NE8),heading(NE9),heading(NE10),heading(NE11),heading(N
```

```
    heading(NE15),heading(NE16),heading(NE17),heading(NE18),heading(NE19),headin
    heading(NE22)];
heading_ne_perror = heading_ne_p - heading_ne_m;
subplot(2,1,2)
plot(heading_ne_perror);
title('Distribution of GNSS heading errors for each northeast pass')
%legend([p1 p2],'5th order polynomial fit', '95th percentile')
xlabel('Sample');ylabel('Error from mean heading (deg)');
grid minor; axis([0 52 -1.5 1]);
```