FLINDERS UNIVERSITY

HONOURS THESIS

GPS Spoofing using a Software Defined Radio

Author:
Alastair WIEGELMANN

Supervisor:
Dr. Saeed REHMAN

A thesis submitted in fulfilment of the requirements
for the degree of Bachelor of Engineering(Electrical and Electronic)
(Honours)

College of Science and Engineering

May 28, 2021

Declaration of Authorship

I, Alastair WIEGELMANN, declare that this thesis titled, "GPS Spoofing using a Software Defined Radio" and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a degree of Bachelor of Engineering(Electrical and Electronic) (Honours).
- This document is in accordance with the plagiarism policy of Flinders University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

Abstract

Alastair WIEGELMANN

GPS Spoofing using a Software Defined Radio

As GNSS technology becomes more and more prevalent in society more needs to be done to ensure that it remains a robust system. There is a distinct issue with satellite technology. That is that the design life time leaves them open to attack as technology propels forward. Especially over the past 15 years the rate of expansion of technology has been high. While the GNSS system as a whole may have been technologically advanced compared to civilian technology when launched, that technology, and others, trickles down in to the hands of attackers. This thesis pointed out the ease in which a spoofing attack was able to be performed. The results show that with relatively low amounts of money and processing power, as well as technical understanding, successful spoofing attacks were performed. Using free and available software coordinates were chosen either a stand alone coordinate for a static spoof, or connected to form a path for a dynamic spoof. The coordinates were saved as text files that were passed into another software program, GPS-SDR-sim that converted the text file into a interlaced I/Q signal data file. This was loaded into the SDR via the GNURadio program and ethernet interface. The victim receivers were successfully spoofed, however the accuracy of the calculated location was less than what was expected for static attacks, with error values consistently over 20m. Dynamic spoofed scenarios seemed to fair much better with less error. While it is also true that with the setup used for testing that there were clear signs that the signal was spoofed there are products that do not have the processing power to perform software based anti spoofing checks. These include embedded systems, as well as existing systems. Many industries are reluctant to change equipment for many years, even in important industries like energy. This leaves this infrastructure open to attacks from this kind of system especially given how portable it is [46].

Acknowledgements

I wish to thank all those that have supported me through not only my thesis, but my entire degree. In particular Saeed, who guided me through the process of performing an Honours project. My parents have always been supportive of my decisions and I could not have made it this far without them. Most importantly I'd like to thank Jacqueline Miller for her endless patience for me through the early years of my degree and her endless compliments as well as grammatical help. To everyone who has had to listen to my endless ramblings about whatever it was I was working on at the time, thank you.

Contents

List of Figures

List of Tables

List of Abbreviations

ASIC	**A**pplication **S**pecific **I**tegrated **C**ircuit
BER	**B**it **E**rror **R**ate
BOC	**B**inary **O**ffset **C**arrier
BPSK	**B**inary **P**hase-**S**hift **K**eying
CDMA	**C**ode **D**ivision **M**ultiple **A**ccess
COTS	**C**ommerical **O**ff **T**he **S**helf
DGPS	**D**ifferential **GPS**
DSP	**D**igital **S**ignal **P**rocessing
ECEF	**E**arth **C**entered **E**arth **F**ixed
EGA	**E**uropean **G**NSS **A**gency
EGNOS	**E**uropean **G**eostationary **N**avigation **O**verlay **S**ervice
EM	**E**lectro**M**agnetic
ESA	**E**uropean **S**pace **A**gency
EW	**E**lectronic **W**arfare
FDMA	**F**requency **D**ivision **M**ultiple **A**ccess
GBAS	**G**round **B**ased **A**ugmentation **S**ystem
GLONASS	**GLO**bal **NA**vigation **S**atellite **S**ystem
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
HDOP	**H**orizontal **D**ilution **O**f **P**recision
IGSO	**I**nclined **G**eo **S**ynchronous **O**rbit
IRNSS	**I**ndian **R**eginal **N**avigation **S**atellite **S**ystem
LEO	**L**ow **E**arth **O**rbit
MEO	**M**edium **E**arth **O**rbit
NavIC	**Nav**igation (with) **I**ndian **C**onstellation
NMEA	**N**ational **M**arine **E**lectronics **A**ssociation
OSNMA	**O**pen **S**ervice **N**avigation **M**essage **A**uthentication
PNT	**P**osition, **N**avigation and **T**iming
PRN	**P**suedo **R**andom **N**umber
PVT	**P**osition, **V**elocity and **T**iming
QZSS	**Q**uasi-**Z**enith **S**atellite **S**ystem
RF	**R**adio **F**requency

RHCP	**R**ight **H**and **C**ircular **P**olarisation
SBAS	**S**atellite **B**ased **A**ugmentation **S**ystem
SDR	**S**oftware **D**efined **R**adio
SNR	**S**ignal (to) **N**oise **R**atio
SPS	**S**amples **P**er **S**econd
SVID	**S**pace **V**ehicle **ID**entification
UAV	**U**nmanned **A**erial **V**ehicle
V2V	**V**ehicle (to) **V**ehivle (communication)
V2X	**V**ehicle (to) (everything) (communication)
XML	e**X**tensible **M**arkup **L**anguage

Physical Constants

Constant Name	$Symbol = ConstantValue$ with units
Speed of Light	$c_0 = 2.997\,924\,58 \times 10^8\,\mathrm{m\,s^{-1}}$ (exact)
Standard Gravitational Parameter of Earth	$\mu_{earth} = 3.986\,004\,418 \times 10^{14}\,\mathrm{m^3\,s^{-2}}$
Circumference of Earth	$C_{earth} = 6.371 \times 10^6\,\mathrm{m}$ (average)

List of Symbols

Symbol	Name	Unit
a	distance	m
P	power	W ($\mathrm{J\,s^{-1}}$)
f	frequency	Hz
t	time	s
ω	angular frequency	rad
μ	standard gravitational parameter	$\mathrm{m^3 s^{-2}}$
ρ	distance (pseudorange)	m

Chapter 1

Introduction

1.1 Motivation

As society moves through the age of technology there is an exponential reliance on reliable access to position and time data. The main source of this information over the recent decades has been through GNSS constellations. GNSS services are now tightly integrated with many facets of life from personal navigation, public transport and management of energy infrastructure. This has made these services the target of attacks. In order to provide adequate defence knowledge of how the attacks are performed is required. There has been a lot more research put into the defence of spoofing attacks, and not so into the method of performing a successful spoofing attack or circumventing existing anti-spoofing methods.

As it currently stands there is no active research or efforts within GNSS attacks or EW (Electronic Warfare) in general from within Flinders University. Therefore, to ensure that Flinders Universityis able to keep up with moving trends and ever advancing technology an EW chair was devised with this project being the first completed in conjunction with. GNSS technology provides critical information for civilian purposes as well as in defence and military. There is a constant need to be aware of the latest research in such ares. This thesis will document background information on both GNSS operating principles as well as background research in spoofing. A workflow was generated and will be used as a starting point for continual research. This will set the tone for further research into EW.

1.2 Aims and Benefits

The aims of this thesis were to investigate and perform GPS spoofing attacks in order to gain a better understanding of the operation of the GPS infrastructure and attack methodology and to create hardware capable of performing a spoofing attack. This will be used to create a baseline for SDR capabilities research from within Flinders Universitywhere future research can build upon this initial documentation. The depth of research into the most cutting edge SDR and GPS Spoofing technologies will therefore not be covered in this paper but will be the topic of future research within the College of Science and Engineering at Flinders University. There was emphasis placed on the technical knowledge required to perform such attacks and the cost of performing them now as opposed to the past. This will allow for further work in counter-measures of GPS spoofing. This thesis will purely concentrate on the research of and implementation of a spoofing device. There will be no implementation of anti-spoofing methodology.

1.3 Research Questions

1. What hardware is required to perform a spoofing attack on commercial GPS receiver?

2. What technical expertise is required for a spoofing attack on a commercial GPS receiver?

3. How can a prototype GPS spoofer be implemented in a controlled environment?

4. How could a prototype be implemented in a real-time static or dynamic environment?

1.4 Limitations

Due to hardware and software limitations, all testing was performed on the Navstar GPS system only. There was no multi constellation attacks. However, the theory and basic structure of the attacks are relevant to all constellations since the same trilateration technique is common to them all.

Since GPS and more specifically the L1 frequency band of GPS is so ubiquitous in society there is open source software available that made this project

achievable. While it would be possible to extend the project to encompass other constellations and frequency bands, this would require much more time that is allowed for.

1.5 Thesis Structure

The rest of this thesis will have a structure as follows, Chapter 2 will introduce the history and concepts in GNSS technology as well as defining and introducing GNSS spoofing. Chapter 3 provides a summary of relevant research in the area of GNSS spoofing attacks and defences. Spoof defence will include detection and mitigation. Chapter 4 outlines the method used in the spoofing attack. Chapter 5 will show the results gathered from the experimentation. Chapter 6 will provide discussion regarding the results of the experiments as well as issues and solutions encountered while performing the experiments as well as recommendations for future work. Finally chapter 7 will provide a conclusion to the project.

Chapter 2

GNSS Systems Overview

The format of this chapter is as follows, there is a section on the details behind GNSS systems and the history behind GNSS and how it came to be in its current state. After that an overview of the current and future GNSS constellations including operating principles was covered. Then real world uses of GNSS are listed. Following that the errors that account for the inaccuracies of calculated GNSS position are covered, including the definition of spoofing types.

2.1 History of GNSS Systems

In 2021 there are many different GNSS systems in place that service the globe. It is a technology that has become deeply ingrained in the everyday lives of most people around the world. The first digital based navigation system similar to what we know today was the use of terrestrial based radio transmitters in the form of hyperbolic navigation systems [26]. The concept was similar to that of the current satellite based systems, namely the use of pseudoranges, although the accuracy was far lower than is acceptable today. The USA developed and commissioned the first satellite based navigation system.

As of 2021 there are six operational GNSS systems in use. These are GPS (USA), GLONASS (Russia), Galileo (EU), BeiDou (China), IRNSS/NavIC (India) and QZSS (Japan). The QZSS system currently acts to complement the coverage of GPS in the East Asian and Oceanic regions with 4 operational satellites. There are plans to increase this number of satellites to allow for stand alone use as a GNSS provider [3]. The Indian IRNSS similarly provides regional coverage around India with 8 satellites in geostationary orbit [44] . There are three categories of orbit, LEO, MEO and GEO. These are categorised by how far from the earths surface the satellite orbits. Current GNSS systems utilise MEO orbits.

GNSS, and more specifically GPS and GLONASS, date back to 1957 during the space race between the USA and the then USSR [2]. When Russian satellite Sputnik 1 was launched scientists at the John Hopkins University were able to track its position and velocity by measuring the Doppler shift of the craft. This tracking from Doppler shift measurements continued with the launch of the subsequent satellites Sputnik 2 and Explorer 1. After this it was thought that the process could be reversed such that a satellite with a known position could be used to resolve an unknown position on earth. In the 1960's the TRANSIT satellite navigation system was made operational for US Navy use, mainly for position and navigation of nuclear ballistic missile submarines. TRANSIT received strong support due to the accuracy of up to 80ft (24m) which was a significant improvement over existing VLF (Very Low Frequency) hyperbolic navigation systems. After 32 years of operation the system was retired in 1996 after proving that space crafts could be reliable [6].

In 1973 the US combined two existing programs in TIMATION and 'Project 621B' to form the 'NAVSTAR Global Positioning System' which would later become known more commonly as GPS. The initial intention for the use of GPS was for military only. However, in 1983 there was an incident that saw a Korean Airlines flight shot down by a Soviet fighter mistaking it for a US aircraft when it wandered off course. After this incident President Reagan announced that GPS would be made available for civilian use. The military insisted that the accuracy of the system be purposely degraded for civilian use though selective availability such that GPS could not be used by adversaries. In 1993 the system was declared operational. To combat the intentional accuracy reduction of the GPS augmentation systems started to appear. Soon after there were Government funded versions of DGPS (Differential GPS) systems [2]. Augmentation systems significantly increases the accuracy by using a receiver with a known location and having that receiver calculate its position from the satellite signals and compare with the known position. These corrections are then broadcast.

The Soviet Union launched their first GLONASS satellite in 1982, and in the following three years 10 more were launched. There are some technical differences between the GLONASS and GPS constellations. The orbital planes are different and GLONASS uses an FDMA (Frequency Division Multiple Access) scheme as opposed to the GPS CDMA (Code Division Multiple Access). In 1993 president Yeltsin declared that the GLONASS constellation was fully operational, however this was not the case.

The TRANSIT satellite system orbit was in a LEO (Low Earth Orbit) polar orbit, whereas all modern GNSS systems utilise a MEO (medium earth orbit) ranging from 20,000km to 23,000km above the earths surface with multiple orbital planes. The number of planes differs between the different constellations as discussed below. The lower the orbit, the higher the velocity required to maintain the orbit as shown by the orbital mechanics equation 2.1. Equally a higher orbit requires a lower velocity which has impact on the Doppler shift at the receivers. Another factor is aerodynamic drag, which is higher the closer to the earths surface due to the atmosphere.

$$v \approx \frac{2\pi a}{T} \approx \sqrt{\frac{\mu_{earth}}{a}} \tag{2.1}$$

The history of the two other main GNSS systems is much shorter, with the Galileo constellation coming as a form of sovereignty for the European Union. The fist satellite was launched in 2011, and the service became operational in 2016. There is also a network of GPS augmentation sites that provide improved accuracy for those in the EU. This system is named EGNOS. This system provided an average of 1.5m accuracy over the EU territory. This was provided through the use of ground stations and a geostationary satellite broadcasting the timing corrections.

2.2 GNSS Systems

2.2.1 Introduction to GNSS Constellations

GNSS is a term used to describe any satellite system that provides position and timing information. All GNSS systems in use today have three separate segments that encompass the term GNSS. That is the ground segment, space segment and user segment. These are used in conjunction to allow the user to calculate with accuracy up to $\pm 10cm$. In order to calculate an unknown position on Earth, the exact position of the orbiting satellite must be known. The satellite will send it's exact position as well as it's current time down to Earth's surface. It's location, or ephemeris, can be tracked due to Kepler's laws and celestial mechanics. The time on board satellites is kept via the use of caesium based atomic clocks. The GPS clocks are accurate to approximately 10 nanoseconds , however, receivers will lose timing due to the reference interpretation of signals and typically provide accuracy of 100 nanoseconds. The value of the clocks are constantly monitored by the ground segment and

updated when required. Part of the receivers (the user segment) will pick up this signal. The components of the message sent are quite simple and consist mostly of the current time (of the satellite) and where the satellite is in the WGS-84 coordinate system[1]. This coordinate system has its origin at the centre of mass of the earth with the Z axis pointing at the north pole, the X axis pointing towards the prime meridian and the y axis perpendicular to both other axes. Due to the relative motion of the satellite and the receiver the transmitted signal will be shifted up or down the frequency range due to the Doppler effect. The combination of this Doppler shift, time taken to arrive and the satellite current location can be used to solve the position in three dimensional space. In order to solve for all dimensions (including time) four satellites are required since there are essentially 4 unknown terms to solve for as shown in equation 2.2 [1]. The ρ_i term indicates the distance from the receiver at the time of reception to that particular satellite at time of transmission and is known as a pseudorange. x_i, y_i and z_i are the coordinates of the i_{th} satellite in three dimensional space as dictated by the ECEF (Earth Centered Earth Fixed) coordinate system. This coordinate system is a version of a Cartesian coordinate systems where the origin is at the center of mass of the earth. x_u, y_u and z_u are the unknown coordinates of the receiver. The coordinates of all satellites are well known due to the distribution of the ephemeris. By getting 3 satellites, the unknown values can be solved using simultaneous equation techniques. The Δt is the error in the clock of the sender and receiver. Any drift in time will be multiplied by the speed of light and can cause very large errors with only a small drift in time. Therefore to get a proper three dimensional fix with high accuracy, a minimum of 4 satellites are required.

This method of position calculation is known as trilateration. The model shown below is a simplified model that ignores the effects of relativity.

$$\rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + c\Delta t \tag{2.2}$$

As can be seen from the above equation 2.2 the accurate synchronisation of time between the receiver and the satellites is required in order to calculate an accurate position. Each of the operational constellations use the same fundamentals to provide accurate position data, but each differ in some key aspects as discussed below.

2.2.2 Trilateration

The process of trilateration uses objects of known locations to calculate the unknown location of other objects. This is done through the use of overlapping circles based on how far the objects are apart. This is known as a pseudorange. To use satellites to determine the location of a receiver the exact location of the satellites must be known, as well as relative distance. Each satellite sends its precise orbital location (ephemeris), as well as the coarse orbital location (almanac) of the rest of the constellation as part of the navigational data. This coupled with the pseudorange of the satellite allows for calculating the position of the receiver. Pseudoranges rely on the knowledge that electromagnetic radiation propagates at the speed of light, as shown in equation 2.3. By measuring the time taken for the signal to transition from the each of the satellites to the receiver, a sphere of radius d centred on the radius indicates all possible locations that the receiver could be. By adding a second satellite, and second sphere based on the pseudorange, the possible locations of the receiver are reduced based on where the spheres intersect. For 2D position only 3 satellites are needed to have only a single possible solution, whereas 3 dimensional position requires 4 satellites as shown in Figure 2.1.

$$d = c \times t \tag{2.3}$$

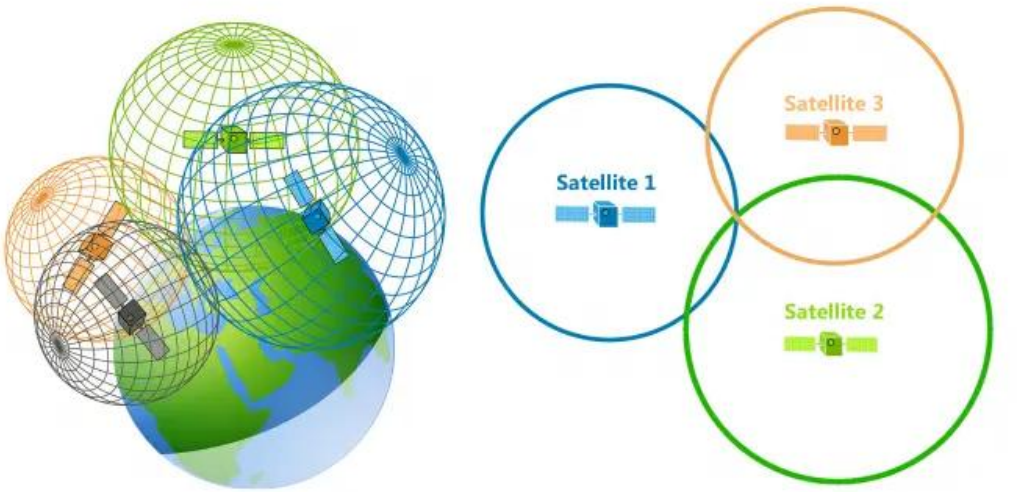

FIGURE 2.1: Illustration of trilateration used in GNSS positioning

2.2.3 Kepler's Laws of Planetary Motion

In the 1600's German astronomer Johannes Kepler described the orbits of the solar systems planets around the sun using three laws. These laws famously moved our understanding of orbits away from circles and described them as ellipses. The position of modern day satellites need to be accurately predicated and as such these laws have become important to the way we live our lives. Kepler's laws are as follows; First law states that the orbit of every planet is an ellipse with the sun at the one of the two foci. Second law states that a line joining the planet to the sun will sweep out equal area during equal intervals of time. The third law states that the ratio of the orbital period with the cube of the semi-major axis is the same for all objects orbiting the same primary. These laws can be adapted to objects that orbit other objects other than the sun. Examples are moons orbiting planets and artificial satellites orbiting the earth.

2.2.4 GPS

The GPS constellation consists of a nominal 24 satellites in 6 orbital planes (nominally 4 satellites per plane) to ensure that there is coverage globally at all times. There are also some spare satellites that are in orbit in the event that a operational satellite malfunctions. The constellation orbits at an altitude of 20,200km which equates to an orbital period of 12hours. There are multiple frequencies of differing properties that each satellite will transmit at. For GPS these are named the L1, L2 and L5 bands and have centre frequencies of 1575.42MHz, 1227.60MHz and 1176.45MHz respectively [30]. This is shown graphically in Figure 2.2. The L1 band is the most commonly used and was the first one to be made operational. In the past the L1 band was used for civilian purposes and the L2 band was a military band. Nowadays both have active civilian and encrypted signals co-exiting within their respective frequency bands. GPS predominately uses BPSK (Binary Phase Shift Keying) but also utilises (Binary Offset Keying) for signal modulation across all of the bands.

All satellites within the constellation will broadcast using the same frequency band (L1, L2 or L5). They are able to do this without interfering through the use of CDMA (Code Division Multiple Access) scheme. Each satellite has a unique chipping code that is used to spread the signal in the frequency domain. This code is a PRN (Pseudo Random Number). This same access scheme is used for GPS, Galileo and BeiDou.

TABLE 2.1: Summary of current and former GPS satellites

Block	Launched	In Service
Block I	11	0
Block II	9	0
Block IIA	19	0
Block IIR	13	8
Block IIR-M	8	7
Block IIF	12	12
Block III	4	4
Total	**76**	**31**

Initially the civilian C/A code was purposely degraded to ensure that adversaries of the USA could not use the system to its maximum effect. In May 2000 President Bill Clinton directed the removal of selective availability from the GPS service, and in 2007 it was announced that the newest version of GPS satellites would be manufactured without the selective availability capability [20] [36].

GPS hardware has had a number of revisions over the life of the system in the form of Blocks. Each block is a major revision, current generation satellites are Block III satellites. Block I satellites were only capable of transmitting on the L1 and L2 bands and only had a 5 year design life. Block II maintained the same transmitting frequency capabilities as before but included three axis stabilisation provided by reaction wheels. The solar arrays were increased from a rated 410W to over 700W. Later iterations of Block II included the L2C band for more robust civil signals and improvement to the military signal. There are currently 4 block III satellites in service with 6 more planned for launch. There are 12 block IIF satellites in operation which were launched from 2010 to 2016. Block IIR has 8 currently operational from 13 launches, block IIR-M has 7 operational from 8 launches and there are no operational satellites from Block I, II or IIA. This totals 31 operational satellites, 24 in operation and 7 spares as summarised in Table 2.1

Accuracy of the GPS system has increased over the years, and since selective availability was legislated against, to approximately 3m for public access and 3-4cm for encrypted.

2.2.5 Galileo

The Galileo constellation is a GNSS created by the ESA (European Space Agency) and operated by the EGA (European GNSS Agency) as a way of providing a modern navigation system that has global coverage. The Galileo constellation has 30 nominal satellites across 3 orbital planes. The first Galileo satellite was launched in 2011. The orbit height of the Galileo constellation is higher than the GPS orbit at 23,222 km.

Galileo operates on 3 different frequency bands, E1, E5 and E6. E1 shares the same centre frequency as the GPS L1 band, 1575.42 MHz. The E6 band does not share a frequency with the GPS system and operates at 1278.75 MHz. The E5 band is split into E5a and E5b sub-bands with E5a sharing the same frequency as L5, 1176.45 MHz, and E5b at 1207.14 MHz. The arrangement of the frequency bands can be seen in Figure 2.2 [28].

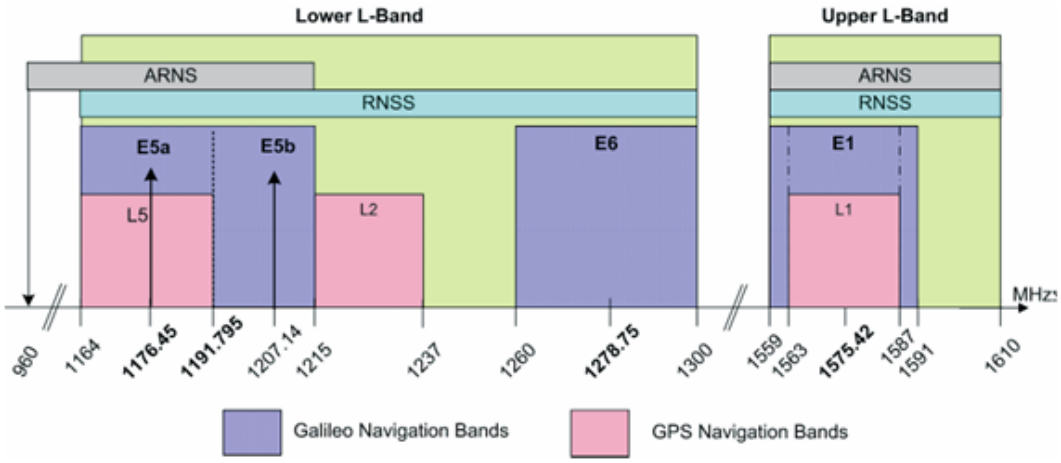

FIGURE 2.2: Galileo and GPS signal frequency arrangement [28]

To maintain interoperability between receivers on the L1 and E1 bands the modulation schemes differ. The E1 band utilises Composite BOC modulation, E6 utilises BPSK, and the E5 bands utilise Alternative BOC modulation. Galileo offers superior accuracy than the other constellations with 1m for the public access and 1cm for encrypted service.

There are special satellites that are part of the Galileo constellation whose job it is to ensure validity. These are known as GIOVE (Galileo In Orbit Validation Element) satellites. There is GIOVE-A and GIOVE-B launched in 2005 and 2008 respectively. Initially these satellites transmitted GNSS test signals on the E1 and E6 frequency bands to allow for monitoring and testing while the rest of the Galileo constellation was being commissioned [13]. Over this

time they have provided great information on the short and long term reliability and accuracy of the on board atomic clocks [39].

2.2.6 GLONASS

The Russian GLONASS constellation has it's history tightly intertwined with that of GPS and the space race, as mentioned above in 2.1. GLONASS consists of a nominal 24 satellites in 3 orbital planes. Since GLONASS uses FDMA, each satellite needs to transmit on a unique frequency [17]. This is in contrast to all other GNSS systems that use CDMA. Extra effort must therefore be applied to the reception of the signals since Doppler effects must be carefully considered. The use of FDMA does make GLONASS more resilient against jamming and spoofing attacks. In a CDMA system, like GPS, all satellites share the same carrier frequency (ignoring the Doppler Effect), thus if an attack is performed all satellites are susceptible. Where as the GLONASS satellites all occupy separate frequency bands making a system wide attack much harder and more expensive to perform. This was the motivation of using FDMA over CDMA initially. The use of FDMA over CDMA also causes and interoperability issue when considering multi constellation support. The receiver must become much more complex in order to support not only multiple carrier frequencies of different constellations but also due to the different access schemes. This increases cost and complexity where Galileo is much more compatible for this use case. There are three frequency bands that GLONASS operates in, L1, L2 and L3. These are separate to the L1, L2 of GPS [29].

GLONASS L1 band is from 1598.0625 MHz to 1605.375 MHz split into 14 channels. Each carrier is 562.5 kHz apart and is characterised by equation 2.4. k represents the channel number of the band and ranges from -7 to +6.

The L2 band ranges from 1242.9375 MHz to 1248.625 MHz with each carrier separated by 437.5 kHz.

The L3 band ranges from 1197.9375 MHz to 1203.625 MHz with the same carrier separation as L2.

$$\begin{aligned} f_{k_{L1}} &= f_{0_{L1}} + k\Delta f_{L1} \\ f_{k_{L2}} &= f_{0_{L2}} + k\Delta f_{L2} \\ f_{k_{L3}} &= f_{0_{L3}} + k\Delta f_{L3} \end{aligned} \tag{2.4}$$

TABLE 2.2: Summary of BeiDou Signal Properties [31]

Signal	Carrier Frequency (MHz)	Modulation Type
B1-I	1561.098	BPSK
B1-Q	1561.098	BPSK
B1-C	1575.420	MBOC(6,1,1/11)
B1	1575.420	BOC(14,2)
B2-I	1207.140	BPSK
B2-Q	1207.140	BPSK
B2a	1176.460	AltBOC(15,10)
B2b	1207.140	AltBOC(15,10)
B3-I	1268.520	BPSK
B3-Q	1268.520	BPSK
B3-A	1268.520	BOC(15,2.5)
B3	1268.520	BPSK

2.2.7 BeiDou

From the BeiDou official website it states that "BeiDou has been constructed and operated by China with an eye on the needs of the country's national security, economic and social development", and was previously known as COMPASS. This is China's attempt at creating a sovereign navigation constellation.

The BeiDou constellation is made up of 35 satellites in total. This includes 5 geostationary satellites, 27 MEO satellites and 3 inclined geosynchronous satellites to give total earth coverage. The MEO satellites orbit at a height of 21,500km, and are distributed across 3 orbital planes. Through the use of the geostationary and geosynchronous satellites there is a region around Asia that has higher accuracy than what is present at other areas of the globe. This accuracy for open service is approximately 2.6m around the Asia Pacific region and 3.6m elsewhere [31]. The constellation contains 3 operating bands, B1, B2 and B3, which have frequencies as shown in Table 2.2. Each of the bands use the CDMA access scheme.

2.2.8 Augmentation Systems

As mentioned above there are two other GNSS constellations that act to improve availability and accuracy in regional areas. These are the Japanese QZSS and Indian IRNSS. Both of these extend the abilities of the USA GPS system. The 4 satellites that are operational for the QZSS constellation are able to transmit on the L1 and L5 band simultaneously. Doing so helps to resolve ionospheric errors by facilitating resolution of position from multiple frequencies [4].

The IRNSS constellation is different to the others mentioned above due to its orbit altitude of 36,000km making them geostationary. All of the satellites are positioned at this orbit. This allows for a very narrow and defined region of usability, which is over India [44].

Other forms of augmentation as mentioned in Section 2.1 were through ground based stations that use receivers to determine the timing error and broadcast these values in order to make the solution to the position calculation more accurate. This type of system allows for sub 1m accuracy. These systems are typically rolled out geographically locally and run by those local Governments.

2.2.9 GNSS Receivers

Traditional GNSS receivers are specialised hardware components that are able to receive the spread spectrum signal from the GNSS satellite and perform the required calculations to acquire the PVT data. This data is then provided serially to a main processor of the system in the form of NMEA (National Marine Electronics Association) sentences, see 2.2.9.1. There are many sentence types and not all receivers will output all sentence types.

2.2.9.1 NMEA Sentences

The NMEA (National Maritime Electronics Association) has a number of standards one of which is NMEA-0183. This standard outlines the combined electrical and data specifications for communication between marine electronic equipment. The data specification was adopted by GPS receiver manufacturers to ensure software could operate with any GPS hardware. The standard outlines the typical message structure. This structure lays the foundation for many different message types called sentences. These sentences carry different information depending on the sentence type, and it is often the case that multiple sentences will be required in order for the software to

TABLE 2.3: GNSS Identifier Mnemonics as per NMEA-0183

Talker Device	**Identifier**
Galileo Positioning System	GA
BDS (BeiDou System)	GB
NavIC (IRNSS)	GI
Glonass Receiver	GL
Global Navigation Satellite System	GN
Global Positioning System (GPS)	GP
QZSS	GQ

understand the full picture. One of the key inclusions within the data specification is the checksum. Each sentence has a checksum based on the contents of the message that can be easily checked to ensure no data was corrupted in the transmission process. The checksum method is XOR based where each character between the delimiting characters are XOR'd together and their integer value is used as the checksum.

Each of the main constellations has their own unique "Talker Identifier Mnemonics" as dictated by the NMEA-0183 standard Table 7 , which is summarised below in Table 2.3. This allows for easy differentiation of PVT sources without needing to know anything about the hardware itself.

insert citation

2.3 Applications of GNSS

GNSS applications are broad and deeply ingrained in almost everyone's everyday lives. These applications vary from providing position and navigation information for public transport or maps on a smartphone or sat-nav devices to providing accurate timing information for financial institutions or energy infrastructure [43] [27] as shown in Figure 2.3. This societal ubiquity has made GNSS as much a critical infrastructure as internet or energy provision, since these services rely on the services provided by GNSS.

2.4 Errors within GNSS

The main fight against GNSS positional accuracy is the introduction of different types of errors. These can be narrowed into three main groups; Satellite

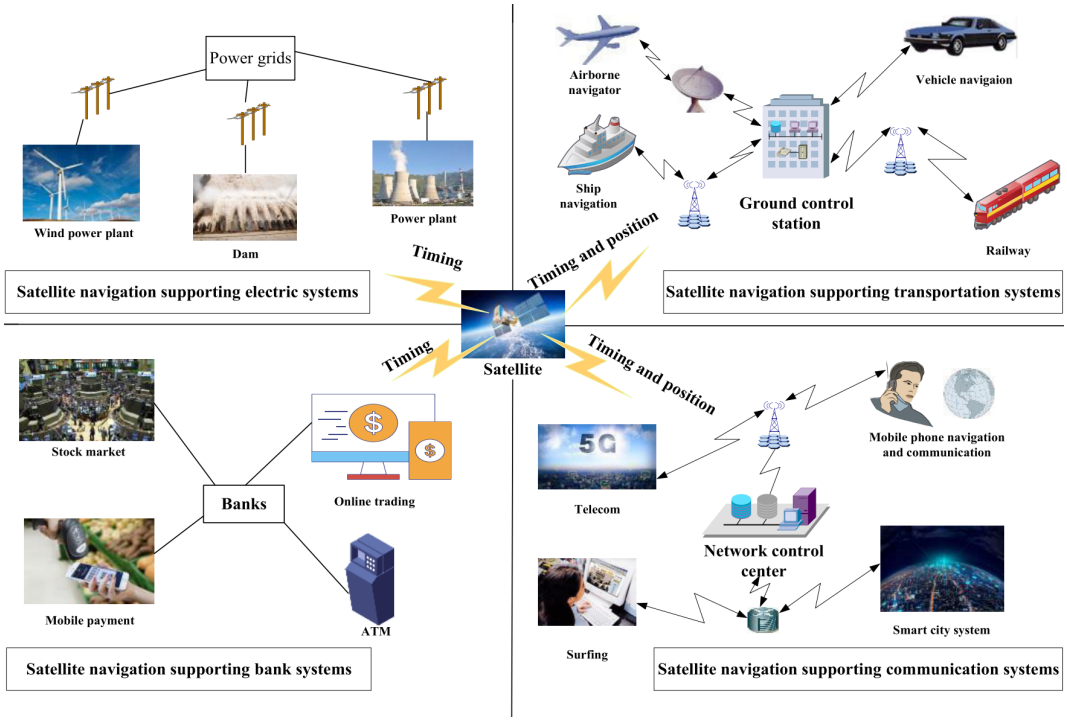

FIGURE 2.3: Applications of GNSS position and timing [43]

errors, Propagation errors and Receiver errors. Propagation errors are most common. When the signals pass through the ionosphere and the troposphere the signals refract, extending the time taken to get to the receiver, thus making the pseudorange not accurate. Other propagation errors include multipath and obstructions errors. Both of which are most notable in built up urban environments of in valleys where direct view of the sky can be difficult.

Satellite errors occur when the predicated orbit differs to the actual orbit. Other causes of errors from the satellites are due to drift of the onboard atomic clocks as well as low elevations where there are proportionally high dilution of precision.

Receiver errors occur due to the introduction of noise through the reception process. GNSS jamming and spoofing also constitute receiver error although these are intentional and malicious.

What all of these errors have in common is that create an incorrect value of the pseudorange which in turn incorrectly calculates the receiver position.

ref

TABLE 2.4: Summary of GPS Errors

Source of Error	**Error Range**
Satellite Clocks	$\pm 2m$
Orbital Errors	$\pm 2.5m$
Ionospheric Delays	$\pm 5m$
Tropospheric Delays	$\pm 0.5m$
Receiver Noise	$\pm 0.3m$
Multipath Errors	$\pm 1m$

2.5 Spoofing Overview

Attacks on the GNSS architecture can broadly be divided into two categories jamming and spoofing [43] [33]. Jamming can be described as causing intentional interference in the communication channel as to make the recovery of signal information impossible. A successful jamming attack will make it impossible to begin tracking satellites and thus be unable to calculate any of the PVT data. This differs from spoofing attacks where the main interest is to deceive any receiver that picks ups the signal into thinking it is elsewhere or else-when. This is done by different methods and stems from which property of the signal is being modified for the attack. A successful spoofing attack will have the receiver 'tracking' the signal produced by the spoofer and the receiver will not be aware that an attack is taking place. Common spoofing attacks are explained below.

2.5.1 Meaconing

Meaconing is a type of GNSS spoofing attack that involves recording a legitimate GNSS signal and re-broadcasting. This could be either instantly or at a different time or place. This requires little prior knowledge of the GNSS system that the attacker wishes to record other when used with existing open source software packages. Although there is a requirement of having the right equipment to be able to capture the raw analog signal.

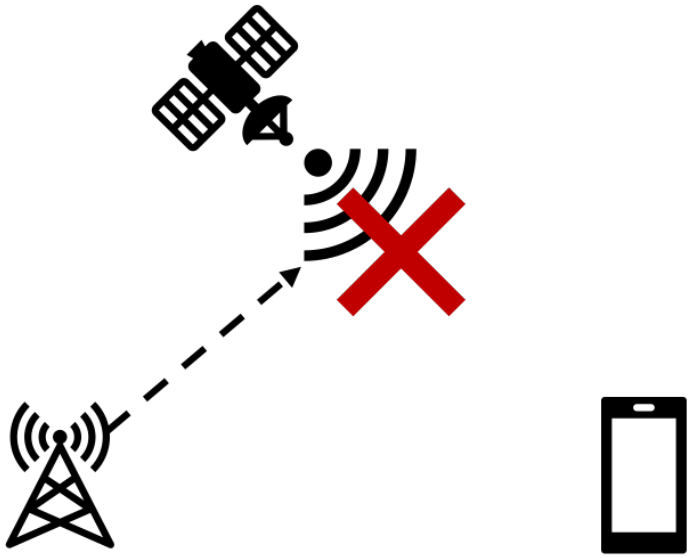

FIGURE 2.4: Representation of how a jammer is able to interfere with a GNSS signal

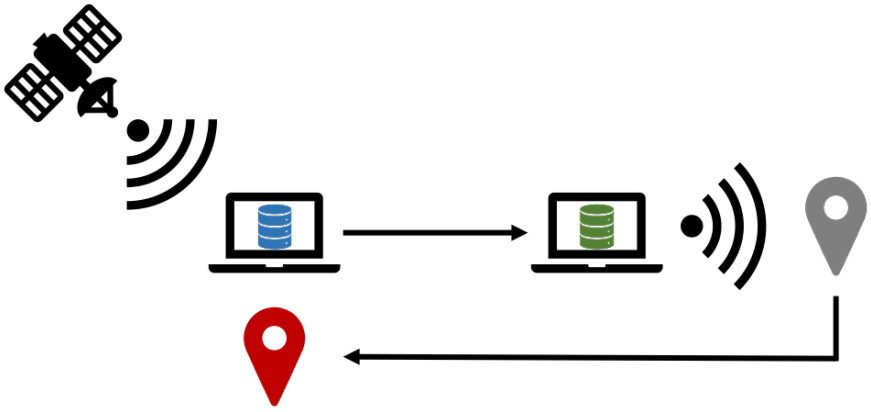

FIGURE 2.5: Representation of how a GNSS spoofer can perform a meaconing attack

2.5.2 Signal Generation

There are software packages and off the shelf hardware that are designed for testing GNSS receivers in a laboratory setting. These can be used to generate a that can be played in such a manner as to perform a spoofing attack. This differs from meaconing because knowledge of the satellite ephemeris and almanac is required. The almanac is a record of the rough location of all satellites in a constellation and the ephemeris is the exact location of each individual satellite. This knowledge is required as to produce the same results as recording the GPS signal from the desired location.

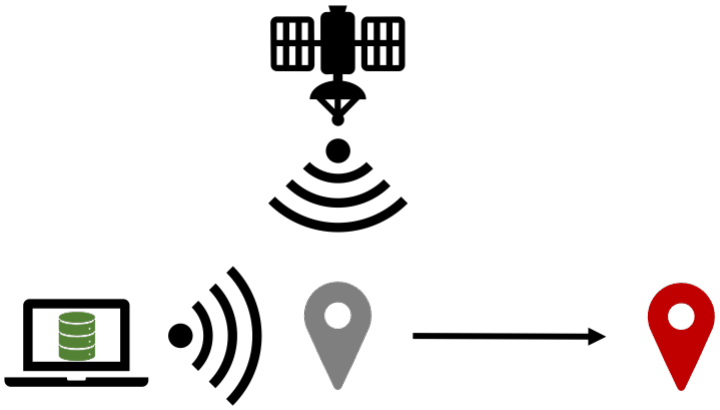

FIGURE 2.6: Representation of how a GNSS spoofing device uses a generated signal to force a receiver to incorrectly calculate its position

2.6 SDR Basics

Software defined radios are devices that enable the user to increase the flexibility of transmission and reception of RF signals by moving processing typically reserved for the analog domain into the digital domain. Thus SDRs move the signal processing from rigid and expensive hardware into flexible and in-expensive software. This allows for a wide ranges of frequencies from Hz up to GHz from the same device with minor or no hardware changes. This can also be done in real time using software. Typical tasks like modulation and demodulation are performed directly in analog hardware for normal radios. This makes them incredibly rigid and only suitable for 1 task. An SDR can implement multiple different modulation/demodulation techniques and switch between them based on selection criteria set by the user. This same is true for centre frequencies or carrier frequencies, they are able to be changed in real time from Hz to GHz depending on SDR model.

The USRP SDR lineup share the same basic block diagram design, as shown in Figure 2.7. This shows the way that the signals are transported from the analog front end of the antenna through to the digital ethernet interface and vice-versa depending on whether transmitting or receiving. Also shown in the image is the quadrature mixers which allow for complex sampling 2.5 as apposed to real sampling 2.6. This means that the in phase and quadrature phase are sampled separately, effectively doubling the sampling rate when compared to real only sampling. This means that when using complex sampling the Nyquist sampling theorem is conserved even when

the bandwidth is equal to the sampling rate.

SDRs need a device connected to them in order to off load processing of data and controlling radio parameters. This has typically been in the form of a PC or similar device, but with recent technological advances SBCs (single board computer) and smartphones are powerful enough to handle this task. There are examples in literature of using smartphones coupled with SDRs to perform many different radio based tasks.

$$x(t) = A_I(t)\cos(2\pi f_c t) - A_Q(t)\sin(2\pi f_c t) \tag{2.5}$$

$$x(t) = A(t)\cos(\omega t + \Phi(t)) \tag{2.6}$$

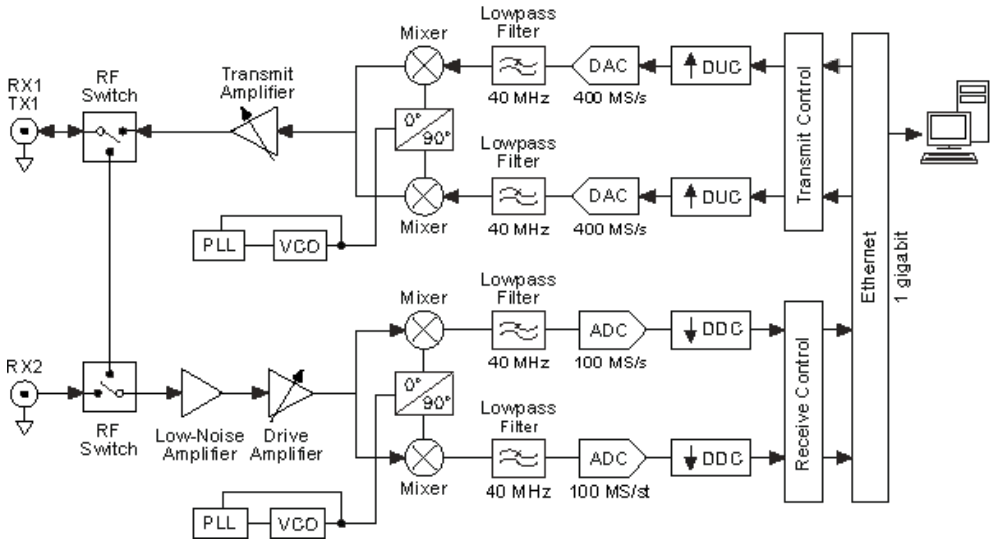

FIGURE 2.7: Simplified block diagram for the RF front end of the USRP SDRs

Chapter 3

Literature Review

3.1 Introduction

This section will go through the existing literature around the topic of GNSS spoofing and anti-spoofing. References were chosen based on their relevance in terms of content as well as age. There have been some significant performance increases in technologies over the past 10 years. This literature review will focus mainly on GNSS spoofing methods, with emphasis on the GPS constellation, and on potential anti-spoofing methods. While development of anti-spoofing algorithms is not an aim of this thesis, knowledge of such methods allow for understanding potential attack vectors.

To ensure that the review of current literature is accurate, some exclusion criteria was implemented.

- Exclusion criteria
  To help narrow the scope of the literature review certain criteria was selected to be omitted
  - Age of source: > 2010 for SDR specific | > 2008 for all others
    When it comes to the implementation of spoofers using SDR platforms, the most up to date information is required since the software typically used in these cases are open source and prone to frequent changes.
    This is less of an issue when detailing the operation of the GPS navigation system, or, to a lesser extent, effective spoofing attacks and defences. This is because most off the shelf receivers have not got any spoof mitigation built into their designs [27].
- Inclusion criteria
  - Application specifics
    While not all sources will have an application, whether to attack

or defend, it should be noted that there was an importance placed on getting sources that applied the theory described.

  - SDR implementation
    Since this project relied on the use of an SDR platform for the application of a spoofer it was important that a large number of sources also use a similar platform. This helped in forming a strategy to complete the project. It was also important so that time was not wasted reinventing the wheel.

- Keywords
  - GPS
  - GPS Spoofing
  - GNSS Spoofing defence
  - GPS UAV Attack
  - GPS Spoof mitigation
  - GPS Spoofing autonomous vehicles
  - meaconing
  - GNSS threat analysis
  - GNSS threat survey

There was an exception to the age criteria [41]. This was commonly referenced by many other sources so warranted inclusion.

3.2 Literature Review

How GPS works has been covered in depth over the past few decades. The operating principles for GPS and other GNSS systems is well known and was covered in the previous chapter 2.

From literature the consensus is that the same thing that has made GPS ubiquitous with navigation and positioning has also made it a target for exploitation and manipulation. That is the workings of the infrastructure are well known and public and are transparent and predictable [11] [47]. This is problematic since this infrastructure is seen as a critical service by many industries including utility management, healthcare/ emergency services and security [27] [33]. Wang, Chen, and Pan [40] noted that the signals presented to the receivers are usually trusted without any authentication or

other checking. The authors show that this trust can be exploited without physical access to or altering the software of the target device. One of the signal properties that makes spoofing simple is the received signal power from space which is less than -130dBm making overpower the legitimate signal a simple task. Having such a system so prone to threats is not ideal.

3.2.1 Spoofing Attack Literature

The further development of SDR platforms has driven down the cost of launching GNSS based spoofing attacks. Devices such as the Hack-RF, USRP, BladeRF and others have been documented for this use [47] [45]. These devices are all examples of Software defined radios that are capable of duplex operation. This combined with open source software, as described in [9] [14] [47]. The former is a software package, GPS-SDR, that converts a compatible SDR into a GPS receiver without any knowledge of GPS or signal processing required. With some understanding this could be modified to capture the raw signal for use in a Meaconing spoofing attack. The later refers to the setup and use of the GPS-SDR-SIM program. This program provides a method for producing a modulated GPS signal for transmission. It requires the RINEX file for the date and time of the spoof attack as well as a set of coordinates to reproduce. Since it is open source it can be easily modified for any use case.

Wang, Chen, and Pan [40] created a spoofing device based on the HackRF SDR platform in conjunction with the open source program GPS-SIM-SDR. Hack-RF was chosen as the SDR platform because of its open source nature, it also had the required specifications to perform the spoof attack. It was shown in this paper that the use of an SDR and open source software was able to fool client devices such as apple smartphones into thinking they were else where. This included specific apps that require location data such as Uber and DiDi. The authors provided a list of recommendations to minimise the risk of being spoofed. These recommendations are non trivial and some required alterations to how the receiver processes the signal data.

Tippenhauer et al. [38] investigated the exact requirements for successfully spoofing a receiver. From the experiments it was found that there are 4 parameters that have required values in order to successfully spoof a GPS signal. That is the relative signal power must be $\geq +2dB$, the constant time offset must be $\leq 75ns$, location offset must be $\leq 500m$ and the relative time offset must be $\leq 80ns$. These values correspond to situations where the receiver already has a lock on satellites. It was found that when values outside

of those mentioned above, the lock would be lost and the spoofing would be unsuccessful. The more victims that are trying to be spoofed the more restrictive the location for performing the spoof becomes.

Software based GPS simulation tools similar to that of GPS-SDR-SIM have also been created using a combination of C and MATLAB [8]. In this example the author used C to ensure processing efficiency of the modulated signal, and used MATLAB to simulate multi-path errors and uncertainty. The resultant file was then saved to the hard drive of the host computer. This could then be loaded into an SDR for transmission, although this was not specifically touched on in the paper.

Over recent times there has been an increase in the industries that rely on the timing and positional data provided by GNSS systems. One such industry is autonomous vehicles, in particular drones. Drones can be used for hobbies, professional photography/ videography or for surveillance purposes. They are small and can be controlled from great distances. All commercially available drones have some form of GNSS/GPS location service built in, and as such they become a target for spoofing attacks. Some drone manufactures have built in auto landing features in the event that the drone enters into restricted airspace's. This was to combat civilians who were flying within airports, causing safety and security issues. This is one such way that spoofing attacks could be utilised now and in the future, by transmitting a signal that would make the drone perceive itself to be in a restricted airspace and land. This kind of attack has been successfully conducted as shown in [47]. Kerns et al. [21] developed a system to perform a series of spoof attacks on UAVs based on an SDR [16]. The goal of this was to be able to control the flight path of the UAV without raising any alarm's from the victim. This paper establishes the required conditions in which a UAV will be susceptible to being captured from a spoofing attack, as well as the range of post capture control that the spoofer will have over the victim. From testing it was found that spoof attacks were successful from up to 50m and up to a velocity of 10m/s. Simulations were produced for analysis of post capture control of the UAV. When testing, both covert and overt spoofing methods were used, distinguished by whether or not the spoofer made an attempt to avoid detection. For the most part there was no practical difference between covert and overt methods since the commercial GPS units did not trigger any anti spoof mechanism when being subjected to an overt attack. Field testing showed that the spoof attempt caused unrecoverable navigation errors which resulted in the UAV crashing. Future work should increase the sophistication of the attacks.

This may decrease chances of the drone crashing due to unsuccessful attacks.

There has been research into ways that GPS spoofing attacks could be used in road navigation scenarios. Zeng et al. were able to implement an algorithm for road network modelling and navigation spoofing using GPS. This algorithm coupled with a HackRF SDR meant that the authors were able to create a lunchbox sized spoofing device. Although in research it was found that the victim devices were able to register a difference in location from network based sources and GNSS based sources, the victims prioritised the location resolved from GNSS sources over network or cellular. While this attack strategy may be successful against people not familiar with their surrounding area, someone who is familiar or is paying close attention should be able to tell they are being led to an incorrect area. Where this is less likely to be the case is with driverless vehicles. In the paper written by De La Torre, Rad, and Choo [7] regarding driverless vehicle safety, it was noted that there is a significant threat to these types of systems that rely heavily on reliable GPS signals. Although there have been proposed solutions to this problem through the use of other sensor information available locally to each vehicle and in the form of an ad-hoc network known as V2V (vehicle to vehicle) and more broadly V2X (vehicle to everything) [5], this is still in its infancy and will require joint work from all vehicle manufacturers.

Psiaki and Humphreys commented on the effect of GNSS spoofing of a cooperative victim. That is when someone is willing to aid the attacker in performing an attack. This may be implemented to circumvent position based restrictions or if being GPS tracked during certain activities. Psiaki and Humphreys used the example of a fisherman wanting their GNSS receiver to falsely report the boat had stayed out of protected areas [27].

In 2012 Jafarnia-Jahromi et al. [18] investigated the different spoofing and antispoofing techniques available. Jafarnia-Jahromi et al. noted that spoofing attacks can be divided into 3 main categories: GPS signal simulator, Receiver-Based spoofers and Sophisticated Receiver based spoofers. These attack strategies come about because of vulnerabilities in the GPS system. These vulnerabilities can be described in the three operational layers of GPS, signal processing, data bit, and position/navigation solutions. Testing spoofing techniques is difficult to achieve since there are regulations around the emission of EM radiation at certain frequencies and power levels. There were three methods used to test the spoofing/antispoofing techniques. These were Indoor re-transmissions, spoofing using recorded data (No RF transmission), and using RF combiners to combine authentication and spoofed signals. The

results that were acquired showed that the commercial GPS receivers were vulnerable to a number of spoofing techniques. Previous research into the effects of spoofing can be found in [16]. This predates more modern methods for launching a successful spoofing attack and uses a specialised DSP chip within a specialised hardware configuration. The findings of this paper was used to recommend anti-spoofing methods for unsophisticated spoofing attacks. Humphreys et al. was able to implement complex spoofing setups with multiple phase locked radios that were able to overcome some anti-spoofing techniques.

Detection	**Attack Techniques**												
Techniques	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
D1	X	X	X	X	X	X	X	X	X	X	X	X	X
D2	∼	✓	X	X	∼	X	X	X	X	X	X	X	X
D3	∼	∼	∼	∼	∼	X	X	∼	∼	∼	∼	X	X
D4	∼	✓	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼	∼
D5	✓	✓	✓	✓	✓	∼	∼	✓	✓	✓	✓	∼	∼
D6	X	✓	✓	X	X	✓	X	✓	✓	X	X	✓	X
D7	X	✓	✓	∼	X	✓	∼	✓	✓	∼	X	✓	∼
D8	X	✓	✓	∼	X	✓	∼	✓	✓	∼	X	✓	∼
D9	∼	✓	✓	✓	∼	✓	✓	✓	✓	✓	∼	✓	✓
D10	✓	✓	✓	✓	✓	✓	✓	✓	∼	∼	∼	∼	∼
D11	✓	✓	✓	✓	✓	✓	✓	X	∼	∼	∼	∼	∼
D12	X	✓	✓	∼	X	✓	∼	✓	✓	∼	X	✓	∼
D13	X	✓	✓	∼	X	✓	∼	✓	✓	∼	X	✓	∼

Detection probability matrix keys: ✓– high, ∼ – intermediate or case-dependent, X – low

	Detection Techniques Key		**Attack Techniques Key**
D1	Pseudorange-based RAIM	A1	Meaconing, single RX ant., single TX ant.
D2	Observables and RPM	A2	Open-loop signal simulator
D3	Correlation function distortion monitoring	A3	RX/SP, single TX ant., no SCER
D4	Drift monitoring (clock offset, IMU/position)	A4	RX/SP, single TX ant., SCER
D5	Observables, RPM, distortion, and drift monitoring	A5	Meaconing, multi. RX ants., single TX ant.
D6	NMA*	A6	Nulling RX/SP, single TX ant., no SCER
D7	NMA* and SCER detection	A7	Nulling RX/SP, single TX ant., SCER
D8	Delayed symmetric-key SSSC*	A8	RX/SP, single TX ant., sensing of victim ant. motion
D9	NMA*, SCER detection, RPM, and drift monitoring	A9	RX/SP, multi. TX ants., no SCER
D10	Multiple RX antennas	A10	RX/SP, multi. TX ants., SCER
D11	Moving RX antenna	A11	Meaconing, multi. RX ants., multi. TX ants.
D12	Dual-RX keyless correlation of unknown SSSC codes	A12	Nulling RX/SP, multi. TX ants., no SCER
D13	Symmetric-key SSSC* [e.g., P(Y) equiv.]	A13	Nulling RX/SP, multi. TX ants., SCER

* Detection techniques requiring changes to the Signal In Space (SIS); TX: Transmitter; RX: Receiver; RX/SP: Receiver-Spoofer

FIGURE 3.1: Summary of spoofing attack and defence methods and their effectiveness against each other as found in [27]

3.2.2 Anti-Spoofing Literature

Within the literature there has been a categorisation of anti-spoofing techniques into detection and mitigation. Detection techniques have been more researched since they are much easier to implement. Mitigation requires overcoming the spoofing attack and maintaining lock with the legitimate GPS signals. It is common within research to investigate methods of spoofing attack and to combine this with recommendations for anti-spoofing. This was the case with Jafarnia-Jahromi et al. [18], Schmidt et al. [33], Wu et al.

[43] and Humphreys et al. [16]. The later, as mentioned above in Section 3.2.1 a spoofing device was developed and tested against some anti-spoofing methods. From this anti-spoofing recommendations were developed. The recommendations are as per Jafarnia-Jahromi et al. Jafarnia-Jahromi et al. [18] investigated antispoofing techniques available. Antispoofing can be broken down into 2 groups spoof detection and spoof mitigation, with each of these being able to be further broken into subcategories.

In Schmidt et al. [33] a review and analysis of the GNSS spoofing threat as well as the countermeasures of the time was provided. This report aimed to fill the gap in 3 main areas of GNSS spoofing research, namely assessing the exact threat scenarios for commonly cited targets, investigate the practical impediments of performing a spoofing attack in the field and lastly to survey and asses the performance of proposed GNSS spoofing defences. When defining spoofing, Schmidt et al. determined that a power ratio (spoofed vs authentic) of just 1.1 was sufficient to force the receiver to move onto the spoofed signal. And with authentic GNSS signal strengths of approximately $-153dBW$ to $-160dBW$, this ratio is easy to achieve. It was noted that the only detectable differences between a legitimate and spoofed one is the timing difference, signal angle of arrival, signal strength, Doppler shift and SNR. Most receivers as of 2016 were not capable of registering these differences. The AGC (Automatic Gain Control) functionality common on GNSS receivers adjusts the signal gain to compensate for any changes in the received signal strength, and in doing so makes the receiver more susceptible to a spoofing attack. The two main spoofing attack types analysed were *Meaconing* and *SCER*. SCER is an evolution of the Meaconing attack where a single satellite signal is rebroadcast after a set delay. Meaconing is the simplest spoofing attack and is a capture and re-transmission attack. This attack is difficult to perform on the military P(Y) code, but simple to perform for the civilian C/A code. The most commonly referred to GNSS spoof attack targets came in the form of Power Disruption networks, shipping, aircraft, trains, criminal security tags and mobile phones. Each of these examples were analysed in detail for potential attack vectors. It was also noted that there are significant practical difficulties to performing a spoofing attack outside of an academic setting where variables can be tightly controlled. First and foremost the cost of the hardware, while declining quickly, is still out of reach of many of the general public costing upwards of $2300 at the time. Current pricing for the USRP N210 in Australia is approximately $4000. Those

without the technical know how could purchase an off the shelf GNSS simulator unit, however these start at $20,000 for a multi constellation generator. Practical difficulties continue with distance from the target, which, due to the inverse square law of radio propagation means to ensure a sufficiently high SNR that distance from attacker to target should be minimised. This is made difficult if the target is moving. Other difficulties were raised including need for direct line of sight and the need for multiple simulators for multi GNSS receivers. While it was found that current generation GNSS receivers were susceptible to spoof attacks, there were countermeasures that could be employed to either detect or mitigate the spoofing threat. Of the proposed GNSS spoofing countermeasures all can be summarised with four categories, Signal processing defences, cryptographic defences, Correlation with other timing sources, Radio spectrum and antenna defences. It was found that at least 1 method from each of the categories was highly effective at defending against an attack. Given the practical difficulties in performing a spoofing attack and the effective defences that have been developed (although not used in the wider community as yet) it was concluded that the threat posed by GNSS spoofing was low but with the potential to increase as the practical implementation became easier and cheaper and if companies did not implement countermeasures. Recommended future work was based around real time detection and location of spoofing attacks.

The review provided in [33] was expanded by a review in 2020 by Wu et al. [43]. Wu et al. provided a survey of the spoofing and anti-spoofing measures that have been researched recently. A detailed list of spoofing and anti-spoofing techniques were covered within the literature, but as mentioned above all can fit within the 2 categories of forwarding or generating based techniques. From the research it was found that most of the anti-spoofing techniques require some form of hardware alteration. There were some methods that did not require such modifications, namely those that relied upon antenna array detection and angle of arrival detection methods. It was found that there is no spoofing method that is immune to all anti-spoofing techniques, and equally there is not single anti-spoofing method that is able to detect and mitigate all spoofing methods. It was advised to continue researching and developing advanced anti-deception systems since the deception models will continue to advance.

The authors outlined challenges faced by GNSS systems which include: PVT accuracy, availability of GNSS systems, integrity of the system, and the ease in which interference can occur.

From [27] the defence techniques could be broken down into three categories; pseudo-range RAIM (receiver autonomous integrity monitoring), looking at the difference between a true and spoofed signal, and looking for interaction between true and spoofed signals. Pseudo-range RAIM is considered too weak against modern spoofing attacks and was not considered in this publication. The specific methods of defence tested in this paper were advanced signal processing techniques, encryption based, drift monitoring, signal geometry and multi pronged spoofing defence.

The effectiveness of each spoof detection technique was tabulated and compared. As was the spoof mitigation techniques. Each detection and mitigation method was given a complexity, effectiveness and spoofing scenario generality rating with notes made about the received capability requirements and are summarised in Table 3.2 and Table 3.3. It was shown that with modest, low complexity spoof detection and mitigation strategies some of the spoof attacks were able to be overcome.

Foruhandeh et al. [11] discussed a method of spoof detection by creating unique a fingerprint from each of the transmitting satellites. This fingerprint algorithm was able to determine if a transmission was authentic or spoofed. This fingerprinting algorithm implementation was found to be fast, simple, cost effective and robust when compared to previous hardening attempts.

Another form of spoof detection was discussed in [37]. Swaszek et al. [37] used multiple receivers in order to process the signals and determine if the signals were authentic or spoofed. These techniques were varied and ranged from simple methods like monitoring the power level of the received signal, to more complex like analysing the presence of vestigial peaks in the correlator output. When there is no spoofer present there will be unique signal characteristics at each of the receivers. The concept of detecting a spoofing event was to compare the position of each receiver against the known relative positions. This was chosen as it will not require any hardware modifications and the software processing can be done by an external processor, making it suitable for retrofitting to existing systems. The testing was performed under the assumption that the spoofing transmitter only had one antenna. In the first case there was approximately 1% chance of a false alarm while the probability of detection was found to be intractable. It was noted that to enable an initial statistical analysis of the solution, some simplifying assumptions were chosen. As such, future works should attempt to reduce the simplifications where possible to create results that are applicable to real world scenarios.

Other means of spoofing detection aim to use existing infrastructure or

existing sensors built into products. In [32] Sanders proposed a method of spoofing detection and localisation using V2V (Vehicle to Vehicle) communications. Using a network of cooperative vehicles the spoofer could be localised by analysing the respective Doppler shifts. After laboratory testing on stationary spoofers, it was found to be effective at providing an approximate location at up to 200m away. When the spoofer was moving, this was decreased to 50m and was movement direction agnostic. In another paper, Jansen et al. proposed a method for detecting spoofing attack on commercial airplanes by monitoring the broadcast information for erratic or unexplainable changes in position or velocity [19]. This was named Crowd-GPS-sec.

Miralles et al. [23] and Oligeri et al. [25] discuss methods of spoofing detection on mobile devices. The Android operating system released an update in 2016 the Android API which included access to the Raw GNSS properties including navigation messages, pseudo-ranges, Doppler frequencies, constellation status and others. An overview of the changes made over recent versions can be found in [22]. From these properties a GPS spoof detection program was developed without the need to modify the hardware of the GPS receiver or the transmitting satellite. From the available information and sensors of the chosen phone, 4 sensor combinations were chosen 1) network location provider, 2) AGC (automation gain control) and C/No engine, 3) inertial sensor data and 4) pseudo-range residual metrics. It was noted that there exists an app that combines all of the mentioned methods together "GNSSAlarm" which was used for testing. From testing it was found that phones tend to give a higher priority to GPS based location than that of other methods (network based). From testing it can be seen that each of the metrics available can be used to combat certain types of spoofing attacks, and when used together as in GNSSAlarm, a robust solution is available [23].

The existing cellular network infrastructure can be used to determine if position and time information received from GNSS sources is accurate. Previously proposed methods for GPS signal detection have typically required special hardware and are difficult to deploy since they are based on the physical properties of the received GPS signal. This new proposal uses broadcasts from cellular towers to validate the position received by the GPS signal. Parameters from the cell towers used to determine the validity of the GPS signal include the number of in-range base stations, the distance to those stations and the received power of the signal. This method was tested against a real spoofing attack and it was found that the there was 0% false positive complete detection of the spoofing signals tested with negligible delay [25].

As mentioned above, there has also been research performed in the context of spoof mitigation. There is a cross over between detection and mitigation since the presence of a spoofer or spoofed signal must first be detected before it is acted upon.

The most up to date sources, [24] [12] [34], propose spoofing mitigation strategies that are reflective of the advancement of technology over the past decade. Morales-Ferre et al. [24] provided a detailed survey of the current GNSS interference landscape in relation to the aviation industry. The development of mitigation techniques was based around the mathematical modelling of different attack types. A detailed summary of mitigation strategies based on the attack/interference type was provided. By grouping all of the resources together this paper was able to produce a single repository where defences against all attack categories mentioned above. Jamming and spoofing signal models were presented, with a class baseband signal model and a list of unknown parameters. Jammers were classified into five different classes based on the complexity of the attack. Single/multi tone AM/FM type attacks made up class 1 jammers, single and multi chirp attacks made up class 2 and 3. Chirp signals with frequency bursts were class 4 jammers and class 5 were DME-like/pulse jammer. Spoofers were divided into simplistic spoofer, intermediate spoofer, sophisticated spoofer and meaconer. The unintentional interference models were also discussed and it was noted that adjacent channel interference is typically harmonics leaking over from other frequency bands. This may be from DVB or LTE sources and is modelled as AM or FM tones. Co-channel interference is most often caused by other GNSS services transmitting at the same frequency. This is modelled as increased AWGN (Additive White Gaussian Noise). A summary of the different detection methods was produced. Detection of a single interference type is simple and can be determined using PDFs (probability density functions). The scenario in which there is no interference will be Gaussian, whereas in the presence of interference the received signal may not be depending on the interference type. Front end detection uses the raw signal before going though the ADC, commonly AGC detectors are used. Since the AGC will attempt to maintain a steady input power it can be monitored to see if the gain requirement suddenly drops, indicating a high power signal and most likely a spoofer. The next type of detection method was the pre-correlator type, in which the signal is converted to a different domain to estimate characteristics of the signal or separate the GNSS and interference signals. The next type

of mitigation strategies were post-correlator types. These are typically reserved for spoofing mitigation as opposed to jamming mitigation. This anti-spoofing method is only effective if the authentic signal can be recognised and separated from the fraudulent one. If not then mitigation is not achievable. The final technique is performing mitigation at a navigation level. This removes the requirement for adding special correlator hardware and uses the typical output from normal receivers. This method works by discarding signals that are deemed to be counterfeit.

Gamba, Nicola, and Motella [12] provided an implementation of the Galileo OS-NMA for embedded devices running ARM based processors. The OS-NMA is an attempt to include spoofing countermeasures within the message structure from the satellite. The NMA cryptographic data is mostly unpredictable, which can be used to protect against replay attacks like meaconing attacks [10]. Software profiling was performed with performance comparisons undertaken between different platforms. There are OSNMA ready commercial hardware receivers, however the authors decided that a software implementation would be best suited as new features are easily added and the algorithm can be easily accessed and modified as needed. An overview of the OS-NMA was provided with parameters, values and descriptions. Testing was performed on a Dell PC with Xeon processor and an ODroid running ARM based exynos processor. It was shown that the ARM based device suffered with execution times ballooning. However the receiver was able to decode the OS-NMA code successfully. Future work will go into optimising the software for different platforms and utilising more powerful hardware to close to gap with more powerful PC machines.

Schmidt et al. [34] developed an SDR for spoofing mitigation purposes since typical COTS GNSS receivers do not have a way of being upgraded since their hardware is set and not able to change. Having a receiver with upgradeable hardware, such as that found in SDR platforms, would make for an ideal GPS receiver. As such SDRs are well suited as GNSS receivers, however, they are challenged by processing the GNSS data and anti spoofing method. Schmidt et al. performed experiments to see how effective SDR platforms were at implementing a mitigation strategy in real time. A reduced complexity MMSE (minimum mean squared error) algorithm was developed for use on the in house developed SDR platform. This algorithm was compared against MF (matched filter) techniques. It was found that at an ISR (interference to signal ratio) of 30dB that MF correlators were unable to receive any usable signal, with BER (bit error rate) of approximately 50%. Using the

proposed maximise correlator, at a 30dB ISR, showed a BER gain of approximately 10^5. However, in testing it was found that most COTS receivers were not able to perform these duties in real time due to the increase in complexity of the antispoofing algorithms.

A consistent theme in the literature is the inclusion of cryptography into GNSS constellations. The method employed by Wesson, Rothlisberger, and Humphreys included adding digital signatures within the GPS civil navigation message. This type of cryptographic defence will secure the receiver against replay spoof attacks. Authentication of each civil GPS signal is performed every 5 minutes. For the proposed authentication strategy a P_D of >0.97 was achieved over the range of 37-51$dB - Hz$. The implementation of such a system would require modifications to the GPS satellite system themselves, as well as a redesign of GPS receivers. The modifications required for the GPS receivers were documented in detail in [42]. For systems that can not easily or readily upgraded to take advantage of these improvements an external device called "The GPS Assimilator", as proposed in [15], could be used to clean the incoming signals. Although it was noted that receivers based on SDR platforms would be able to implement the authentication algorithm easily.

The summary of spoofing at the time of [41] is shown in Table 3.1. While there are more complicated strategies available now, many commercial receivers do not implement them, as such the basic strategies are still valid.

3.3 Literature Review Conclusion

Some of the early assessments of the spoofing threat included methods for performing said attacks using specialised hardware [16]. This increased the technical knowledge required to be able to perform an attack.

Common recommendations from the literature was the implementation of, at a minimum, basic anti-spoofing infrastructure. This is more common place in applications that have suitable software processing capabilities. But dedicated COTS receivers do not typically have any defence strategies in place. Cryptographic solutions are frequently suggested in literature, and have been implemented with success. However such implementations require alterations to the space segment of the GNSS constellation. This may be implemented for the next generation satellites but for now this is not feasible.

TABLE 3.1: Summary of Spoofing Mitigation Methods from [41]

Test Statistic	**Function**	**Limitation**
Absolute signal power	Limit the spoof signal power	Antenna attitude and environment related
Signal power changing rate	Detect stationary spoof station	Antenna attitude and environment related
Relative signal strengths on all carriers	Detect spoofing on single carrier	Affected by ionosphere refraction
Range rate	Bound the phase and code range rate	Relate to GPS receiver's moving direction
Doppler shift	Detect spoof that uses one transmitter to spoof all satellites	None
Correlation peaks	Correlate L1/L2 binary messages	Low performance on Y-code
GPS signal after removing all navigation data	Recover authentic data	Requires low spoof/authentic signal power ratio
Range differences: phase/-code, L1/L2	Identify signal source	Needs to be L1/L2 receiver
Ephemeris data	Verify ephemeris data including satellite position	None
Signal power and data	Jump detection	None

Some literate have proposed workarounds should the constellations be upgraded, such as [15] where an external device performs the cryptographic checks before passing the signal to the outdated receiver.

Areas of future research were also outlined and include but are not limited to interconnecting the satellites together, increasing accuracy of atomic time keeping as well as introducing advanced anti-spoofing techniques, more wide spread support of multi constellation receivers, and increasing anti interference capabilities of receivers through either hardware, software or both [43].

Anti-spoofing methods have previously been abandoned because of cost and complexity issues [11].

TABLE 3.2: Summary of Spoofing Detection Methods from [18]

Anti-Spoofing method	**Spoofing Feature**	**Complexity**	**Effectiveness**	**Receiver required capability**	**Spoofing scenario generality**
C/N_0 monitoring	Higher C/N_0	Low	Medium	C/N_0 monitoring	Medium
Absolute power monitoring	Higher amplitude	Low	Medium	Absolute power monitoring	High
Power variation versus receiver movement	Higher power variations due to proximity	Low	Low	Antenna movement/ C/N_0 monitoring	Low
L1/L2 power comparison	No L2 signal for spoofer	Medium	Low	L2 reception capability	Low
Direction of arrival comparison	Spoofing signals coming from the same direction	High	High	Multiple receiver antennas	High
Pairwise correlation in synthetic array	Spoofing signals coming from the same direction	Low	High	Measuring correlation coefficient	High
TOA discrimination	Inevitable delay of spoofing signal	Medium	Medium	TOA analysis	Low
Signal quality monitoring	Deviated shape of authentic correlation peak	Medium	Medium	Multiple correlators	Low
Distribution analysis of the correlator output	Perturbed amplitude distribution due to spoofing-authentic interaction	Low	Medium	Distribution analysis of correlator outputs	Medium
Consistency check with other solutions	Inconsistency of spoofing solution	High	High	Different navigation sensors	High
Cryptographic authentication	Not authenticated	High	High	Authentication	High
Code and phase rate consistency check	Mismatch between artificial code and phase rate	Low	Low	-	Low
GPS clock consistency check	Spoofing/authentic clock inconsistency	Low	Medium	-	Medium

TABLE 3.3: Summary of Spoofing Mitigation Methods from [18]

Anti-Spoofing method	**Spoofing Feature**	**Complexity**	**Effectiveness**	**Receiver required capability**	**Spoofing scenario generality**
Vestigial signal detection	The authentic signal is still present and can be detected	High	Medium	Multiple receive channels	Medium
Multi-antenna null steering	Spoofing signals coming from same direction	Medium	High	Multiple receive antennas	High
RAIM	Higher residuals for spoofed measurements	Medium	Medium	-	Medium

Chapter 4

Methodology

This chapter covers the hardware and software setup used for performing the GPS spoofing experiments. Rationalising the choices made and discussing the benefits and limitations of these selections.

4.1 Testing Methodology

In this section the method of creating a GPS signal spoofing device is detailed. The main part of this device is the upgradeability of the SDR platform and in particular the USRP N210 SDR. As previously mentioned the benefit of using an SDR over using an ASIC is that with a new version of software new capabilities are available to the device. This could be beneficial to either the spoofer or spoof defense.

The success of the spoofing was dictated by whether the receiver was able to lock onto the signal and calculate the same location as expected or travel the same path depending on whether the test is a static or dynamic spoofing test.

Testing was also performed on the reception of GPS signals. Such that they could be used in Meaconing attacks. Initially there was an attempt to receive a GPS signal using the GPS-SDR software as proof that the antennae were operational. This was then changed such that the reception was handled directly through GNURadio and the raw digitised version of the signal was saved to a file such that it could be replayed elsewhere.

4.2 Data Collection

To see the effectiveness of the GPS spoofing methods experiments were carried out and results were recorded. The success of the experiments were

dictated by whether or not the receiver was reporting false location or timing information. Using an Android phone there is access to the raw GPS information which can be used to determine if the spoofing signal is being accepted. However, the use of a "maps" program was also used as a way to determine if there was any form of software/hardware anti-spoofing technique being used post GPS receiver. A simple COTS was also used to log the NMEA sentences from its serial interface. A custom program was generated in order to graph various outputs from these sentences.

For the purpose of this project just three sentences were used when parsing the log files, "GPGGA", "GPGSV" and "GPRMC". The GP mnemonic is a designation that the received signal was from GPS. The other constellations have their own designations when using NMEA sentences as dictated by Table 2.3. GGA (Global Positioning System Fix Data) [1] sentences contain information related to the fix of the GPS signal. The calculated latitude and longitude as well as whether DGPS is being used in the fix, as well as others, as shown in Table 4.1. However, only the latitude and longitude information was used, see Listing 4.1.

An example of a GGA sentence is:

```
$GPGGA,000056.000,3500.5533,S,13834.1337,E,1,10,0.87,-14.7,M,-0.7,M,,*74
```

```
try:
    with open(infile, "r") as nmeaFile:
        for count, line in enumerate(nmeaFile):
            tmp = line.split('*')
            string = tmp[0].split(',')
            msgType = string[0][3:]
            currentTime = datetime.now()
            timestamp2 = (currentTime - startTime).microseconds
   /1000000

            #
            # Parsing GGA sentences
            #
            elif msgType == "GGA":
                if dataValid:
                    try:
                        latitude = degMin2deg(string[2])
```

[1] https://www.nmea.org/Assets/20130801 0183 identifier list.pdf

```
                        longitude = degMin2deg(string[4])
                        if string[3] == 'S':
                            latitude = latitude * -1
                        if string[5] == 'W':
                            longitude = longitude * -1
                        coordinates.append([timestamp2, latitude,
    longitude])
                    except ValueError:
                        print("No valid coordinates")
                        continue
```

LISTING 4.1: Parsing GGA sentences for lattitue and longitude information

The use of GSV (GNSS Satellites in View) sentences allowed for tracking how many and which satellites were in view (SVID) and what the carrier to noise ratio ($\frac{C}{N_0}$) was. This was used to generate the carrier to noise graphs and were stored in key:value dictionaries, see Listing 4.2. Since the sentences can only be a set length as per the standard, multiple successive GSV sentences may need to be sent.

An example of a GSV sentence that is part of a group of 4 is:

```
$GPGSV,4,1,13,23,70,156,52,10,69,284,52,20,50,138,,18,49,117,52*73
$GPGSV,4,2,13,27,37,224,52,16,32,276,52,15,26,127,51,26,23,308,51*77
$GPGSV,4,3,13,29,19,038,17,32,18,001,18,44,05,279,,13,02,148,47*75
$GPGSV,4,4,13,193,,,*40
```

```
    try:
        with open(infile, "r") as nmeaFile:
            for count, line in enumerate(nmeaFile):
                tmp = line.split('*')
                string = tmp[0].split(',')
                msgType = string[0][3:]
                currentTime = datetime.now()
                timestamp2 = (currentTime - startTime).
    microseconds/1000000

                #
                # Parsing GSV sentences
                #
                elif msgType == "GSV":
```

TABLE 4.1: NMEA GGA Sentence structure

Field	**Example**	**Description**
Message ID	$GPGGA	GGA Protocol header
UTC time	000056.000	hhmmss.sss
Latitude	3500.5533	ddmm.mmmm
N/S Indicator	S	N = North, S = South
Longitude	13834.1337	dddmm.mmmm
E/W Indicator	E	E = East, W = West
Position Fix Indicator	1	0: Fix not available 1: GPS SPS mode, fix valid 2: DGPS SPS mode, fix valid
Satellites used	10	
HDOP	0.87	Horizontal dilution of precision
MLS Altitude	-14.7	Meters
Units	M	Meters
Geoid separation	-0.7	Meters
Units	M	Meters
Age of differential correlation		Seconds
Differential reference station ID		
Checksum	*74	
<CR><LF>		End of message termination

```
14                  NumofMessages = int(string[1])
15                  sequenceNumber = int(string[2])
16                  satellitesView = int(string[3])
17                  satellitesToCalc = satellitesView-(
NumofMessages-1)*4 # this will dictate the number the index
numbers of final sentence of group
18                  if sequenceNumber == 1:
19                      timestamp = (currentTime - startTime).
microseconds/1000000
20                  try:
```

```
21                          if sequenceNumber < NumofMessages:
22                              if string[4] not in svSNR:
23                                  svSNR[string[4]] = [string[7]]
24                              else:
25                                  svSNR[string[4]].append(string
    [7])
26
27                              if string[8] not in svSNR:
28                                  svSNR[string[8]] = [string[11]]
29                              else:
30                                  svSNR[string[8]].append(string
    [11])
31
32                              if string[12] not in svSNR:
33                                  svSNR[string[12]] = [string[15]]
34                              else:
35                                  svSNR[string[12]].append(string
    [15])
36
37                              if string[16] not in svSNR:
38                                  svSNR[string[16]] = [string[19]]
39                              else:
40                                  svSNR[string[16]].append(string
    [19])
41                          else:
42                              if satellitesToCalc >= 1:
43                                  if string[4] not in svSNR:
44                                      svSNR[string[4]] = [string
    [7]]
45                                  else:
46                                      svSNR[string[4]].append(
    string[7])
47                              if satellitesToCalc >= 2:
48                                  if string[8] not in svSNR:
49                                      svSNR[string[8]] = [string
    [11]]
50                                  else:
```

```
51                                  svSNR[string[8]].append(
    string[11])
52                          if satellitesToCalc >=3:
53                              if string[12] not in svSNR:
54                                  svSNR[string[12]] = [string
    [15]]
55                              else:
56                                  svSNR[string[12]].append(
    string[15])
57                          if satellitesToCalc == 4:
58                              if string[16] not in svSNR:
59                                  svSNR[string[16]] = [string
    [19]]
60                              else:
61                                  svSNR[string[16]].append(
    string[19])
62                          if satellitesToCalc > 4:
63                              raise OverflowError
64
65                  except IndexError:
66                      pass
67                  except OverflowError:
68                      print("Computational error occured.
    Access to a value higher than available attempted")
69                  if sequenceNumber == NumofMessages:
70                      if svSNR:
71                          satellites[timestamp] = svSNR
72                      svSNR = {}
73
```

LISTING 4.2: Parsing GSV sentences for SVID carrier to noise ratio informaiton

As shown in Table 4.3 RMC (Recommended Minimum Specific GNSS Data) sentences contain a lot of useful information. However, for the purposes of this project it was only used in order to check when there is valid positional information, Listing 4.3.

```
$GPRMC,000118.000,A,3500.4918,S,13834.1369,E,53.66,12.07,300321,,,A*78
```

TABLE 4.2: NMEA GSV Sentence structure

Field	**Example**	**Description**
Message ID	$GPGSV	GSV Protocol header
Number of messages	4	Number of messages in group
Sequence Number	1	Message number
Satellites in view	13	Total number of satellites in view
Satellite ID 1	23	PRN number of satellite
Elevation 1	70	Elevation in degrees (0-90)
Azimuth 1	156	Azimuth in degrees (0-359)
SNR 1	52	SNR in dBHz (0-99)
Satellite ID 2	10	PRN number of satellite
Elevation 2	69	Elevation in degrees (0-90)
Azimuth 2	284	Azimuth in degrees (0-359)
SNR 2	52	SNR in dBHz (0-99)
Satellite ID 3	20	PRN number of satellite
Elevation 3	50	Elevation in degrees (0-90)
Azimuth 3	138	Azimuth in degrees (0-359)
SNR 3		SNR in dBHz (0-99)
Satellite ID 4	18	PRN number of satellite
Elevation 4	49	Elevation in degrees (0-90)
Azimuth 4	117	Azimuth in degrees (0-359)
SNR 4	52	SNR in dBHz (0-99)
Checksum	*73	
<CR><LF>		End of message termination

```
1    try:
2        with open(infile, "r") as nmeaFile:
3            for count, line in enumerate(nmeaFile):
4                tmp = line.split('*')
5                string = tmp[0].split(',')
6                msgType = string[0][3:]
```

```
7                currentTime = datetime.now()
8                timestamp2 = (currentTime - startTime).
  microseconds/1000000
9
10               #
11               # Parsing RMC sentences
12               #
13               if msgType == "RMC":
14                   if not dataValid:
15                       if not firstFix:
16                           timetofix += 1
17                   if string[2] == 'A':
18                       dataValid = True
19                       if not firstFix:
20                           firstFix = True
21                   else:
22                       dataValid = False
23
```

LISTING 4.3: Parsing RMC sentences for data and fix validity checks

4.3 Hardware Setup

For all experiments the same hardware was used. This included the USRP SDR, laptop and GPS receiver. The laptop was an important piece of equipment since it needed to be powerful enough to be able to feed the data to the SDR quick enough to avoid the aforementioned under run issues. Towards the end of the project there was a hardware fault with the Pixel XL phone receiver. This was replaced by the Pixel 4a 5G.

There was no formal decision process for choosing the hardware components since they were all parts that had already been purchased and were in possession of the supervisor. Having said that some research went into the suitability of the equipment, mainly the SDR. It was found that this was one of the most powerful and most importantly had one of the most stable clock oscillators available in software defined radios. This made is well suited to GNSS based tasks. Alternatives, like the RTL-SDR are capable devices for the

TABLE 4.3: NMEA RMC Sentence structure

Field	**Example**	**Description**
Message ID	$GPRMC	RMC Protocol header
UTC time	000118.000	hhmmss.sss
Status	A	A = Valid, V = Not Valid
Latitude	3500.4918	ddmm.mmmm
N/S Indicator	S	N = North, S = South
Longitude	13834.1369	dddmm.mmmm
E/W Indicator	E	E = East, W = West
Speed over ground	53.66	knots
Course over ground	12.07	degrees
HDOP	0.87	Horizontal dilution of precision
Date	300321	ddmmyy
Magnetic Variation		Meters
Mode	A	A = Autonomous, D = DGPS, E = DR
Checksum	*78	
<CR><LF>		End of message termination

cost, but only offer reception. Duplex options like the HackRF are also suitable for transmitting high bandwidth signals, however the on board clock is not stable enough for highly accurate signals like that of GNSS. There was a workaround for this, which included de-soldering the included oscillator and soldering a more stable part. In future, if access to USRP based SDRs is not available then this would be a very good second option. However, there was access to a better tool, being the N210, therefore it was chosen. From Ettus, there were two daughtercards that allow access to the desired 1575.42MHz band of the L1 GPS signal, the SBX and WBX. The SBX was chosen because it was already available. It should be noted that there would be no theoretical difference between either daughtercard since the desired frequency was not at the extreme of either cards bands. Either card would also be capable of transmitting or receiving on the other bands of GNSS constellations.

To ensure the SDR and PC were able to communicate, an ethernet cable was connected between each device and the IP address of the PC was set to 192.168.10.2/24 as per the default configuration from the setup documentation. To check if the IP address was correctly set, a command line program was run that would poll the appropriate interfaces of the PC and check for USRP devices attached, see Figure 4.1. There was also a command line program that allowed for the probing of a radio to gather information on what daughtercard was installed and other properties, see Figures 4.2, 4.3 and 4.4.

```
alastair@alastair-Inspiron-7570:~/Desktop/gps-sdr-sim-master$ uhd_find_devices
[INFO] [UHD] linux; GNU C++ version 9.2.1 20200304; Boost_107100; UHD_3.15.0.0-2build5
--------------------------------------------------
-- UHD Device 0
--------------------------------------------------
Device Address:
    serial: EFR19S6UP
    addr: 192.168.10.2
    name:
    type: usrp2
```

FIGURE 4.1: Using the UHD daemon to find connected USRP SDR devices

```
/
|       Device: USRP2 / N-Series Device
|     /
|     |       Mboard: N210r4
|     |   hardware: 2577
|     |   mac-addr: a0:36:fa:25:39:f6
|     |   ip-addr: 192.168.10.2
|     |   subnet: 255.255.255.255
|     |   gateway: 255.255.255.255
|     |   gpsdo: none
|     |   serial: EFR19S6UP
|     |   FW Version: 12.4
|     |   FPGA Version: 11.1
|     |
|     |   Time sources:  none, external, _external_, mimo
|     |   Clock sources: internal, external, mimo
|     |   Sensors: mimo_locked, ref_locked
```

FIGURE 4.2: Using the UHD program to probe the hardware details of any connected USRP device. This part of the output shows the main board details

- Laptop
  - Dell Inspiron 15
  - CPU: Core i7 Quad Core/ 8 thread
  - RAM: 32GB

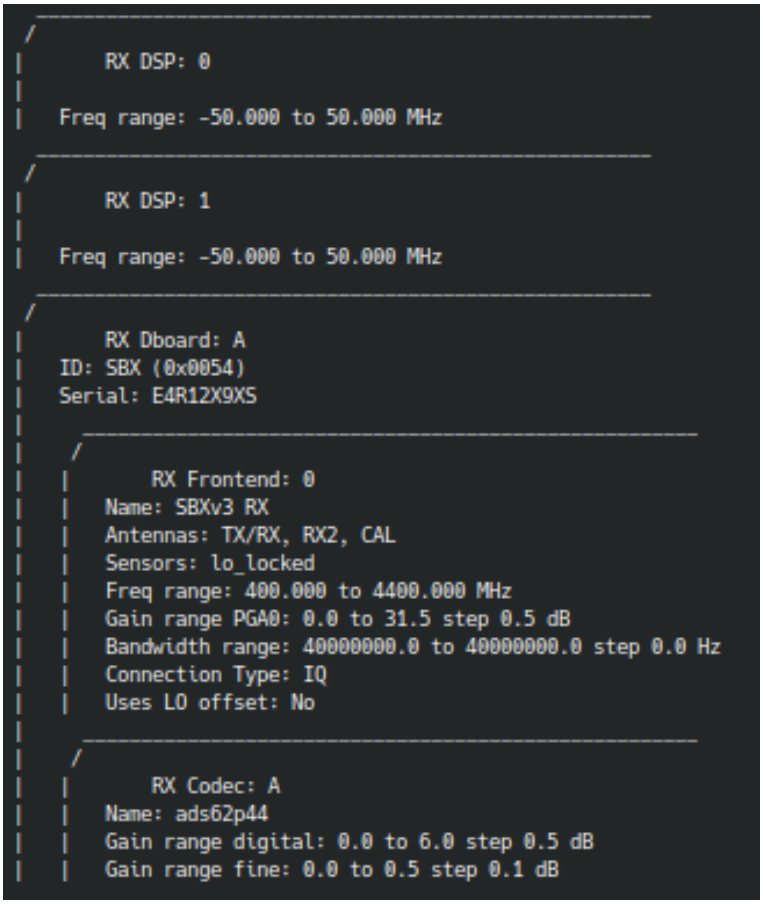

FIGURE 4.3: Using the UHD program to probe the hardware details of any connected USRP device. This part of the output shows the reception details

  - Ethernet Connection: gigabit

- Software defined radio
  - USRP N210
  - SBX-40 daughtercard
  - 30dB attenuator
  - Full duplex
  - Gigabit Ethernet connection
  - high performance FPGA
  - omnidirectional antenna

- GPS Receiver (Phone 1)
  - Google Pixel XL
  - Android 10

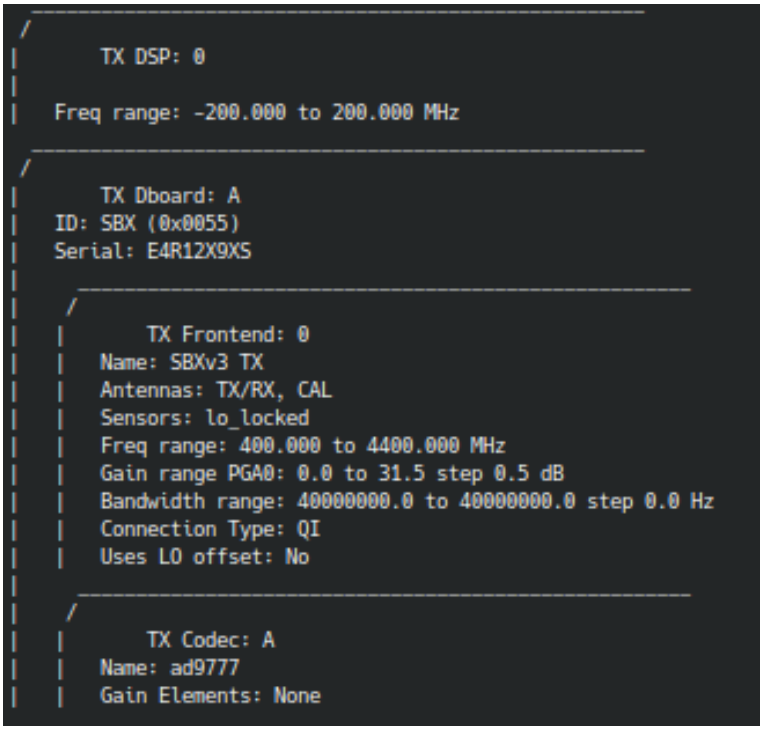

FIGURE 4.4: Using the UHD program to probe the hardware details of any connected USRP device. This part of the output shows the transmission details

  - Multi constellation GNSS support (GPS+QZSS, GLONASS, BeiDou)

- GPS Receiver (Phone 2)
  - Google Pixel 4a 5G
  - Android 11
  - Multi constellation GNSS support (GPS+QZSS, GLONASS, BeiDou)

- GPS Receiver (COTS)
  - GPS L1 support only
  - NMEA output

4.3.1 SDR Setup for GNSS Reception

Reception of GNSS signals is a complicated process which involves synchronising time values and solving simultaneous equations for position, therefore

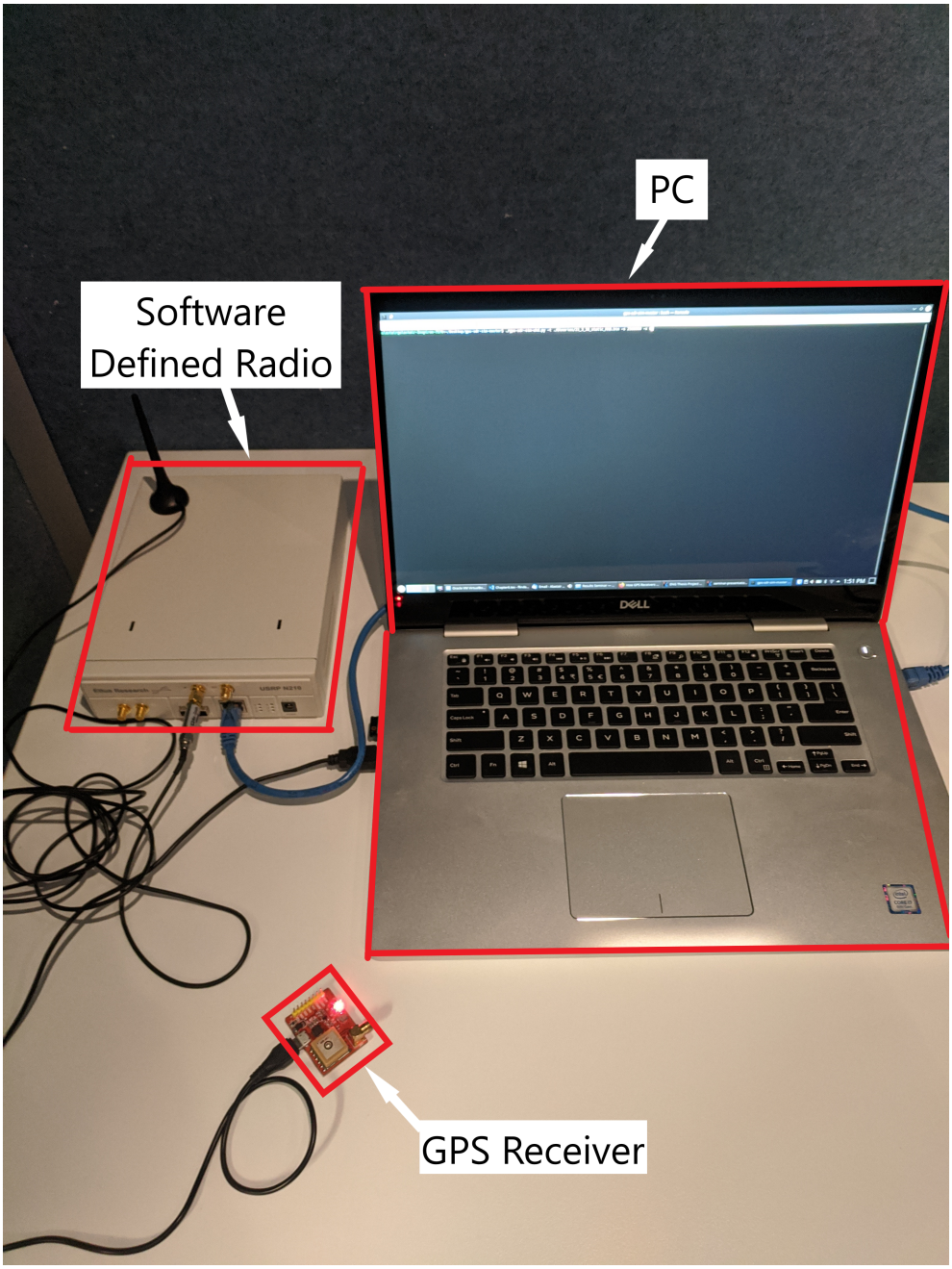

FIGURE 4.5: Hardware setup for testing spoofing transmission within Faraday cage

instead of developing the software from scratch it was decided that the open-source program GNSS-SDR would be used to perform all of these functions. This software has been built over a number of years and is able to receive different GNSS signals and translate them into position.

The hardware setup for this was different. Since the signal strength of a GNSS transmission is so low when it reaches the earths surface ($\approx -160dBm$) an active antenna is typically required for best performance, especially if

there is no clear view of the sky or if there are many buildings that add multipath signals into consideration.

4.4 Software Setup

The laptop was the interface between software and hardware. It was used to generate the binary files and then to control the SDR. The SDR would not operate without the PC attached to the ethernet port.

The software used for this project was a combination of open source and custom software. The software was chosen while performing the literature review. There were examples ***Insert citations that used GPS-SDR-sim*** of using open source programs for the purpose of GPS spoofing. This was the best step forward since there were credible results with that software.

Custom software was created for the reporting and results of the experiments.

- GPS-SDR-sim
- GNURadio
- GNSS-SDR
- Python3

4.5 Faraday Cage

A Faraday cage was used for testing purposes for two main reasons. It will isolate the target device from existing legitimate GNSS signals and stop any transmitted radiation from propagating into the local environment. Isolation from receiving legitimate signals is important since a receiver that is tracking a satellite already is harder to jam or spoof than one that is not. More important is ensuring that the radiation does not enter the environment since transmitting any signals on the frequency band for GNSS systems is illegal in Australia.

Since the received signal strength from a GNSS satellite is so low ($\approx -150dBm$) any signal that is transmitted from Earth's surface will be able overpower these signals for up to 85km, assuming an omnidirectional antenna, as shown in equation 4.1. This calculation was performed with the assumed maximum transmission power of the SDR of $15dBm$ coupled with

a $30dB$ attenuator and transmitting on the L1 GPS band. In practise the effective range will be much less due to attenuation due to objects between transmitter and receiver, but this calculation shows that performing the experiments within a controlled environment was required.

$$\begin{aligned} Att_{dB} &= 10\log_{10}\left(\frac{c}{4\pi df}\right)^2 \\ d &= \frac{c}{4\pi 10^{\left(\frac{Att_{dB}}{20}\right)} f} = \frac{3\times 10^8}{4\pi 10^{-6.75} 1.57542\times 10^9} \approx 85km \end{aligned} \tag{4.1}$$

4.6 Testing Workflow

Some preliminary testing was done with different software setups to see which would be the best for reproducing spoofing results. A combination of meaconing and signal generating techniques were investigated. Initially meaconing was chosen as the best way to perform an attack, therefore an attempt was made to record real time GPS signals and store them for later transmission. Initial testing using a passive log periodic antenna resulted in no data being properly captured. Since a log-periodic antenna was used there was a mismatch in the polarisation of the wave. The transmitted GPS signal is polarised as RHCP, whereas by its nature log-periodic antennas are linearly polarised. This equated to a $3dB$ attenuation of the signal. This coupled with the lack of signal gain from the passive antenna and the directional nature of the antenna meant that the data within the signal was unrecoverable and an active antenna should be used. Unfortunately, none of the daughtercards on hand were able to feed an active antenna. A bias-tee was used in order to feed the antenna with the $5V$ required for its operation while filtering out the DC to feed into the SDR. To interface with the bias-tee a USB cable was cut and used to connect to a perf board with a soldered SMA connector. Unfortunately this was unsuccessful. The GNSS-SDR program was unable to find or lock onto any of the GPS satellites at any time. It was found that another opensource program, GPS-SDR-sim, could be used to create binary files that replicate the received signals from the satellites.

Each of the nodes of Figure 4.6 will be expanded upon below.

4.6.1 Choose Coordinates

Regardless of whether a static or dynamic spoofing attack is desired, the best way to choose the coordinates was through the use of SatGen3. There was

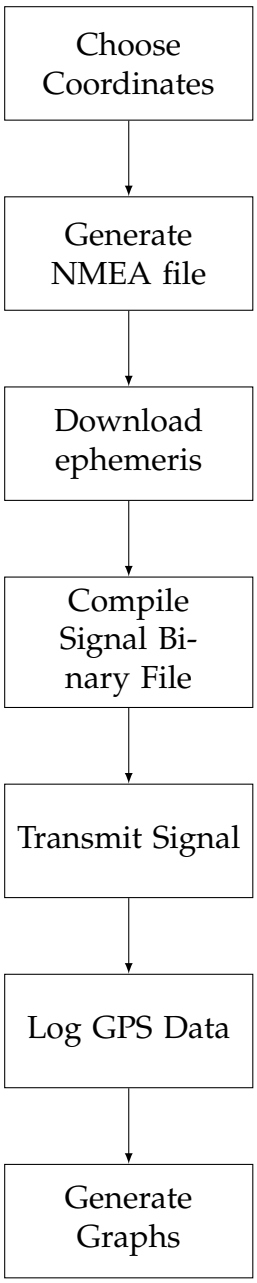

FIGURE 4.6: Flowchart of performing experimentation

an alternative for static scenarios which did remove the need for installing SatGen3. This alternative was through the use of GPS-SDR-sim itself. It has an attribute that allows for inputting either latitude longitude or ECEF coordinates. So while this does save one step, it still requires using a map software to find the desired coordinates, which is considerably simplified using SatGen3. To compile dynamic situations a GGA NMEA stream is always required. Considering for this project there was a desire to have both static and dynamic spoofing it made sense to maintain a more consistent workflow.

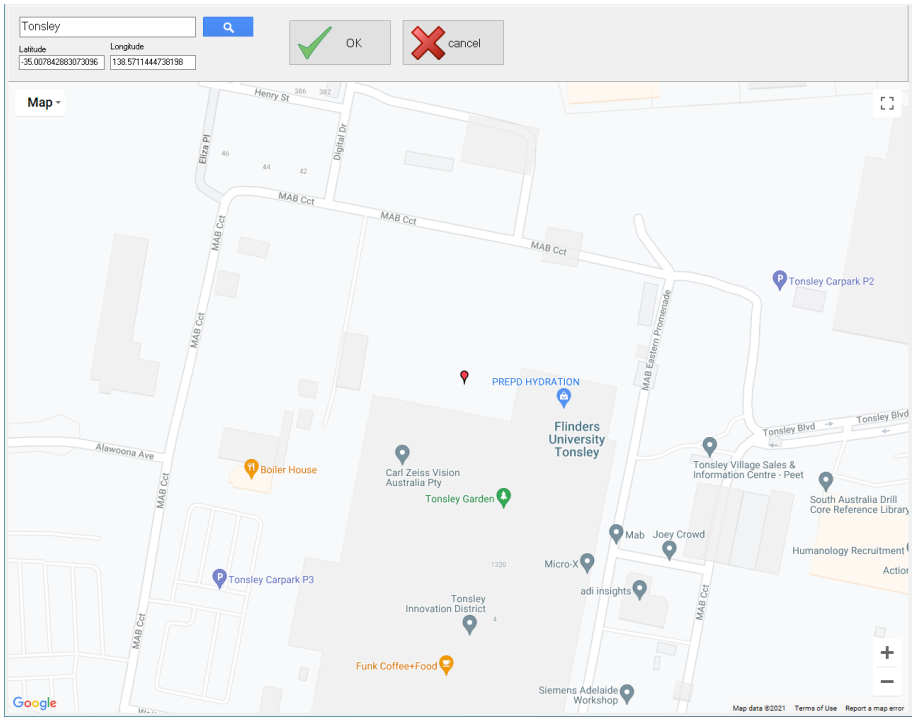

FIGURE 4.7: Using google maps within SatGen3 to choose a static coordinate to be used as desired spoof location

Figure 4.7 shows how the searching and selecting of a coordinate for use in a static spoof attack was done within the SatGen3 application. The SatGen3 program has access to the google maps API and allows for searching. This makes finding the desired location for spoofing attack easy. Figure 4.8 shows how mapping out of a path is performed within SatGen3 in order to generate a dynamic spoof attack.

4.6.2 Generate NMEA File

The coordinate and NMEA file creation step is essentially the same step within the SatGen3 software package. Once the coordinate or path was chosen, it was a matter of choosing to save the output as a text file. An example of this is shown in Figure 4.9. In order to keep track of the source NMEA files and the results files the following naming scheme was employed:

```
YY_MM_DD_<static/dynamic>_LOCATION.nmea
YY_MM_DD_<static/dynamic>_LOCATION.bin
YY_MM_DD_<static/dynamic>_LOCATION.txt
```

Where .nmea file was the source file, the .bin file was the signal binary file and the .txt file was the log file produced from the GPS receiver.

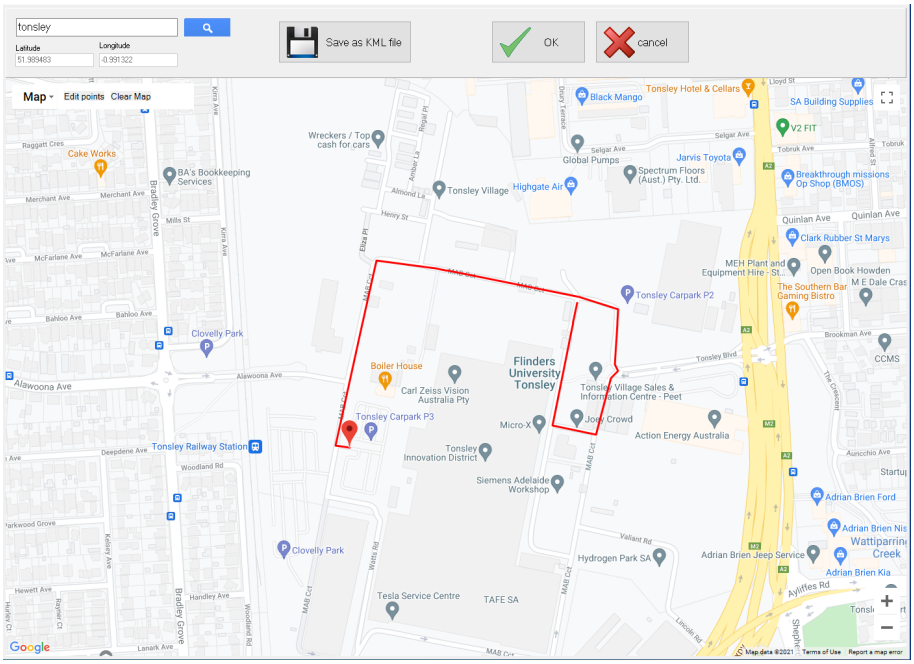

FIGURE 4.8: Using google maps within SatGen3 to choose a set of coordinates to be used as desired spoof path for a dynamic spoof attack

When generating a dynamic file there is an option to set the amount of "stationary" time. This will put a buffer of unmoving coordinates for a set amount of time. This was used to allow the spoofing victim time to lock onto the spoofed signal before the simulated movement began.

To generate the binary bit stream for use with an SDR RF front end a GGA NMEA data stream was created using the SatGen3 software package. This NMEA stream was used to make the binary file using the GPS-SDR-sim command line interface program. SatGen3 replaces the need for capturing the raw GNSS signals or GGA sentence stream thus increasing the flexibility of scenarios that can be tested. Although there is no guarantee that the simulated stream of information is going to be correct. Thus a capture replay attack should still be considered as more reliable.

4.6.3 Download Ephemeris

From the NASA website, hourly versions of the GPS constellation ephemeris, in the form of a RINEX file, are available. These are compiled into a daily file. These daily files are available all the way from 1992. An account is required to access these files, but there are sample programs provided for retrieving the files programatically using different methods. A script was generated that was able to get the latest version from the website using Python. The full

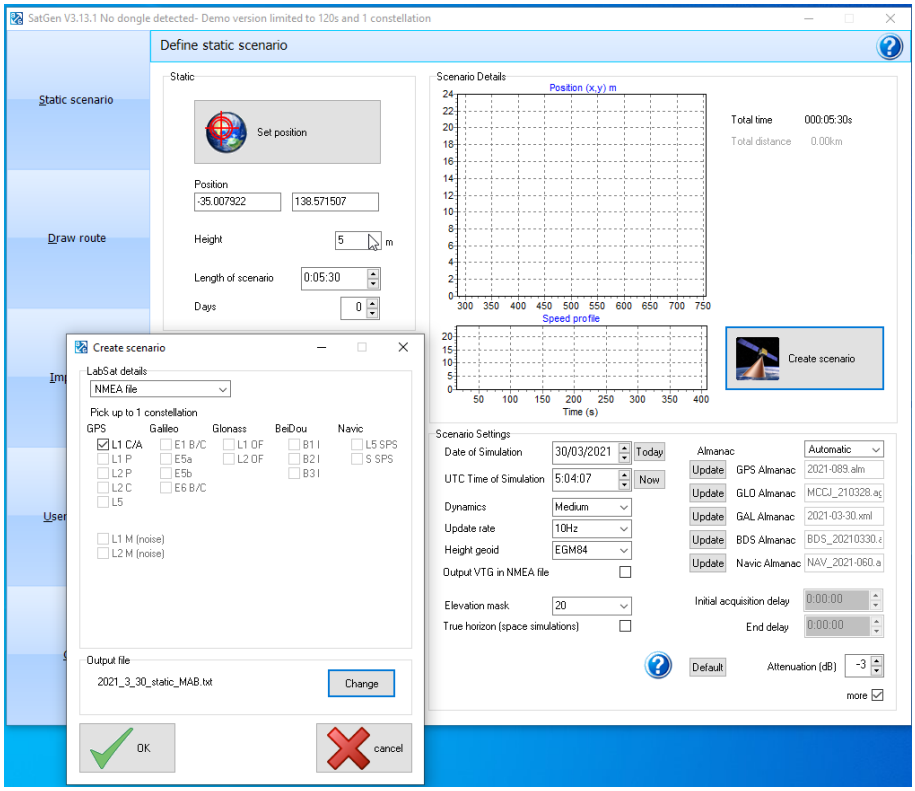

FIGURE 4.9: Using static coordinate to generate NMEA file

version of this script is shown in Appendix A.2. There were initially some issues with getting the script to connect to the remote server to gain access to the files. It was found that there was a bug on Linux with a certain version of openssl. A newer version was manually installed and this solved the issue.

The downloaded file was a compressed version of the text file, and thus it was required to be uncompressed using the tar program.

4.6.4 Compile Signal Binary File

GPS-SDR-sim compiles the signals into a 16bit data format where the first 8bits represents the in-phase data and the next 8 bits represents the Quadrature data information. The minimum requirements to compile a binary file using GPS-SDR-SIM was a source coordinate in either ECEF or latitude/-longitude format and the ephemeris for the desired date and time that the spoofer will transmit. This did not need to match the date and time that the signal was being transmitted on. If the attack was to be a dynamic spoofing attack then it was required that there be a "user motion file" or GGA sentence stream. This stream was required to be 10Hz. It was decided that even for static scenarios a GGA stream would be used. This simplified the workflow overall. Using this method meant that the program was able to calculate

the duration of the transmission itself. The output was altered in order to conform with the naming convention stated in Section 4.6.2. The sampling rate was also modified to be 2.5Msps instead of the default 2.6Msps. When compiling some of the dynamic spoofing scenarios the maximum size was reached. This is a limit that is hardcoded when the GPS-SDR-sim program is compiled on the host machine. Therefore, the program was cleanly recompiled in with the command to increase the maximum size of output file.

Using this information, the place, time and ephemeris the program is able to calculate which of the satellites will be in view and what their expected pseudoranges would be. This is shown in Figure 4.11. The signal modulation is applied within the compilation of the file, simplifying the process of transmitting the file.

```
Usage: gps-sdr-sim [options]
Options:
  -e <gps_nav>     RINEX navigation file for GPS ephemerides (required)
  -u <user_motion> User motion file (dynamic mode)
  -g <nmea_gga>    NMEA GGA stream (dynamic mode)
  -c <location>    ECEF X,Y,Z in meters (static mode) e.g. 3967283.154,1022538.181,4872414.484
  -l <location>    Lat,Lon,Hgt (static mode) e.g. 35.681298,139.766247,10.0
  -t <date,time>   Scenario start time YYYY/MM/DD,hh:mm:ss
  -T <date,time>   Overwrite TOC and TOE to scenario start time
  -d <duration>    Duration [sec] (dynamic mode max: 800, static mode max: 86400)
  -o <output>      I/Q sampling data file (default: gpssim.bin)
  -s <frequency>   Sampling frequency [Hz] (default: 2600000)
  -b <iq_bits>     I/Q data format [1/8/16] (default: 16)
  -i               Disable ionospheric delay for spacecraft scenario
  -v               Show details about simulated channels
```

FIGURE 4.10: Options available for compiling spoofed GPS signal using the GPS-SDR-SIM program

```
alastair@alastair-Inspiron-7570:~/Desktop/gps-sdr-sim-master$ ./gps-sdr-sim -e Ephemeris/brdc0830.21n
 -g ./NMEA_Files/2021_3_30_dynamic_mab.txt -o ./Binaries/21_3_30_dynamic_MAB.bin -s 2500000
Start time = 2021/03/24,00:00:00 (2150:259200)
Duration = 141.1 [sec]
10  305.3  58.8  20726733.0   2.0
13  140.6   8.3  25021561.1   5.5
15  116.7  31.6  23024059.3   3.3
16  261.8  34.8  22312925.9   2.5
18  134.4  55.3  21097016.2   2.1
20  147.4  67.8  20497201.3   1.9
23  173.8  82.4  20232409.7   1.8
26  297.8  30.2  22625795.3   2.9
27  222.9  26.5  23258508.8   2.8
29   44.7  29.3  22864991.4   3.7
32  357.7   7.7  24909771.1   5.9
Time into run = 141.1
Done!
Process time = 27.1 [sec]
```

FIGURE 4.11: Example of running GPS-SDR-sim with required arguments to generate dynamic spoof around the Tonsley MAB

4.6.5 Transmit Signal

Once the signal has been compiled into a binary file it is sent to the USRP SDR via GNURadio. GNURadio has an SDK that allows for text based programming. A script that was capable of connecting to USRP devices automatically through the UHD daemon was distributed with the GPS-SDR-sim program. This was used to simplify the development process.

This was a simple program that connected the source file to the SDR via some simple blocks as shown in Figure 4.12. The file is read 16bits at a time into the "IShort to Complex" block. This converts the interleaved I/Q data into a complex file type which is then passed to the SDR via an amplifier block.

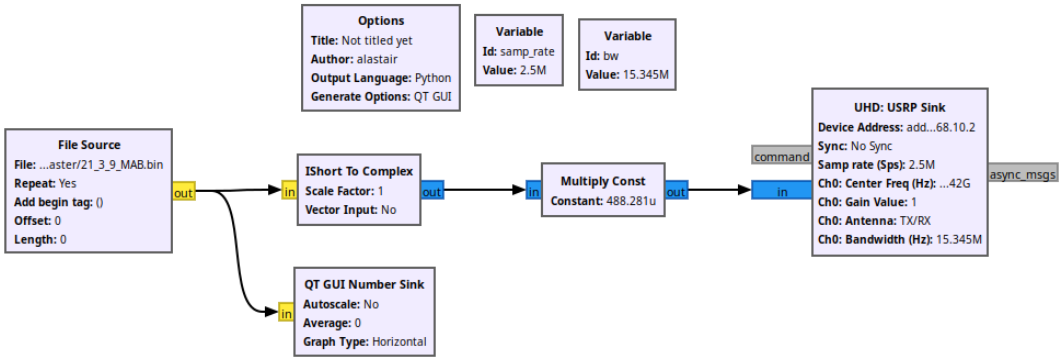

FIGURE 4.12: GNURadio flowchart for transmitting binary files with USRP

An important part of the transmission was that it needed to be performed within a Faraday cage for legal purposes. This was discussed in section 4.5.

4.6.6 Log GPS Data

To log the transmitted GPS signals from the SDR a COTS receiver and a smartphone were set up. The COTS receiver has a USB interface that outputs a stream of NMEA sentences at a 1Hz frequency. This can be modified to 5Hz, but this was not required for these experiments. Using a script the USB port was opened and logged to a text file, see Appendix A.3. The log file was passed as an argument and could be stored wherever it was needed. Each time a spoof attempt was made, the GPS receiver was powered off to ensure there was not lock when starting the experiment. This ensured results were as consistent as possible.

When using a smartphone Android was required for logging since the OS allows access to the raw properties of the received signal. There is also an

intuitive application "GNSS Logger" that allows for comprehensive visualisation of current GNSS tracking as well as the ability to log data. This data was logged to a text file using a non standard layout, but there was an option to log the NMEA sentences as well.

4.6.7 Generate Graphs

Once there was a log file, graphs were generated to visualise the various parameters of the location resolution as interpreted by the receiver. The generation of the graphs that make up the results section was done in python and was based upon the NMEA sentences logged from the receivers. GPS receivers have the ability to check the validity of signals they receive. When they are able to resolve a location the value of one of the fields within the RMC sentence is changed to reflect this. The number of sentences before initial lock were counted. This was used to determine the time to first fix of each test since it was known that the GPS receiver would output at 1Hz. Alterations would need to be made if the receiver was outputting at a higher frequency. As such this should not be considered accurate and viewed as an estimate only.

The carrier to noise ratio graphs were generated directly from the output of the GSV sentences. The SVID and associated $\frac{C}{N_0}$ ratios were stored as the value in a key value pair. The key was a timestamp used to ensure each sentence was represented.
The latitude and longitude were retrieved from the GGA sentences.

Using the NMEA stream created in section 4.6.2 a desired plot was put on both the static and dynamic graphs. For the static graphs this reference was also used as the origin when calculating the error of the calculated location.

The error was calculated by finding the difference in latitude and longitude from the desired origin point and converting the ΔLat and $\Delta Long$ into meters. This assumes that the distances being dealt with are so small that they are essentially flat. Thus the equations can be simplified to remove any angles.

The earth is not perfectly spherical therefore when converting latitude and longitude into x and y coordinates in meters the average of the polar plane and equatorial planes were taken, as shown in equation 4.2. The Δx and Δy values from Equation 4.3 and 4.4 represent the error of the coordinate in the x and y direction respectively.
When finding the total error value the law of Haversines was used.

$$\begin{aligned} C_{Polar} &= 40007.863km \\ C_{Equatorial} &= 40075.017km \end{aligned} \tag{4.2}$$

$$\Delta x = \Delta Lat \times \frac{C_{Polar}}{\frac{2}{180}} \times 1000m \tag{4.3}$$

$$\Delta y = \Delta Long \times \frac{C_{Equatorial}}{\frac{4}{90}} \times 1000m \tag{4.4}$$

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{4.5}$$

Below (Listing 4.4) is the implementation of the law of Haversines in python used for the calculation of distance between two latitude and longitude points. The output of the function was stored into an array and used to graph the total error over time.

```
def latlon2m(lat1, long1, lat2, long2):
    r = 6371000 # this is the radius of the earth
    dLat = radians(lat2) - radians(lat1) # change in latitude in
   radians
    dLong = radians(long2) - radians(long1) # change in longitude
    in radians
    a = math.sin(dLat/2) * math.sin(dLat/2) + math.cos(radians(
   lat1))*math.cos(radians(lat2))*math.sin(dLong/2)*math.sin(
   dLong/2)
    d = 2*r*math.asin(math.sqrt(a)) # distance in meters
    return d
```

LISTING 4.4: Python implementation of the law of Haversines used to calculate the distance between two points on a sphere

There is a PC companion application for the GNSS Logger app that allows for comprehensive graphing and reports to be generated for a particular log file.

4.7 Experiments

4.7.1 Static Spoofing

Within this thesis static spoofing is defined as producing a signal that produces a calculated location that does not change over time. While in practise there was some minor movement caused by the uncertainty in trilateration, this change in position is minor and within the range of error of GPS positioning. Using the SatGen3 software a location was chosen as the spoof location. After setting the desired time and date of spoofing the scenario was created. This

4.7.2 Dynamic Spoofing

Dynamic spoofing refers to the production of a signal that when used to calculate position, will be shown to change over time. The path traced by the receiver will be set at the time of production of the binary file, but there will be perceived motion.

4.7.3 Real-Time Spoofing

Due to time constraints a real-time algorithm was not produced as part of this project. However, utilising the open source projects that have been used to complete this thesis it would be probable to be able to create a real time spoofing device that would be able to react to the positional changes of the receiver instead of following a pre-determined path when generating the binary file. The signals generation algorithm could be ported to a GNURadio block in C++ to allow for easy access to real-time GPS signal spoofing. If an SDR is a full duplex radio, such is the case for the USRP N210, then one port can be receiving the real GNSS signal and the other can be transmitting the spoofed signal. Care would need to be taken in the set up of this arrangement since any transmitted signal would also be picked up by the receiving antenna. Therefore a directional transmitting antenna and physical distance should be employed to allow for legitimate GNSS signals to reach the receiver port.

Chapter 5

Results

This section covers the results of the GNSS reception and GPS transmission experiments. Section 5.1 covers the attempts at receiving authentic GNSS signals and accurately calculating a location. Section 5.2 will cover the attempts and results of performing spoofing attacks on GPS receivers.

5.1 GNSS Reception

Before testing the GPS transmission performance of the SDR, reception was tested, with less success. Three antenna setups were chosen each failing to lock and track the satellites. A log linear, an omnidirectional and an active patch antenna were all used. The active antenna should have given best results since it includes amplifiers and filters within the antenna module itself. The failure could be to do with the bias-t used to inject the 5V DC into the antenna, however testing with a multi-meter showed that there was 5V on the RF+DC port of the Tee.

5.2 GPS Transmission

Experimentation was performed at the Tonsley campus of Flinders University, which has coordinates of $-35.007650, 138.572030$. At first it was decided to use the centre of the Adelaide CBD as the coordinates for spoofing, that is $-34.5571732282, 138.3599516878$. Therefore it would be considered successful if the GPS receiver was to show that the current location was in the CBD. This was extended to include other locations, Davis Station in Antarctica and the MAB at Flinders University Tonsley Campus with coordinates of $-68.5762449235, 77.9696166515$ and $-35.00792163316667, 138.5715065625$ respectively.

Dynamic spoofing attacks were set to the road surrounding the MAB and around the CBD of Adelaide. All spoof attack dates were set to the same date, which was chosen to be the 30th March 2021. This was arbitrary and only chosen since this was when official testing began.

5.2.1 Meaconing Attack

Since the GPS reception failed with the SDR, there was no way of performing a meaconing attack. Therefore, the only viable attack strategy was to generate a binary file of the intended location. It would be ideal to perform this kind of attack in the future since this is the only basic attack type that can be used on the encrypted military band.

5.2.2 Spoofing Attack

Once the workflow and hardware setup for the SDR was properly configured it was found that a smartphone was able to be spoofed. When attempting to spoof devices in the wild, the Pixel XL smartphone was susceptible to attack while more modern smartphones that were tested were not susceptible. These included the iPhone 12 Pro, Samsung Galaxy S10 and Google Pixel 3XL. The spoofing setup was not able to fool any of these devices. This could be down to software based anti-spoofing algorithms that have been implemented including the usage of multiple constellations for position resolution.

5.2.3 Static Spoofing

5.2.3.1 Tonsley MAB

The grassed area outside of the Flinders University Tonsley campus was chosen as one of the points to spoof to. As mentioend above the desired coordinates that the receiver were $-35.00792163316667, 138.5715065625$.

The $\frac{C}{N_0}$ ratio graph for the static MAB spoof attack, see Figure 5.1, shows that there was a good quality signal being received throughout the test, with no evidence of under running. In this example the time to first fix was calculated as being 37 seconds. This is quite good for a cold start situation. During experimentation the time to first fix has been variable between 30 seconds and 180 seconds. A trend has been the quicker the fix the better the overall accuracy of the spoofed location. At approximately the time of first fix it can be seen that there was a reduction in number of satellite PRNs being received.

Then throughout there was spikes from various satellites. These spikes offer no benefit since they only lasted a few seconds.

From Figures 5.3 and 5.2 it can be seen that there are not many GPS location points on this graph, so while the spoof was successful in the sense that the receiver was able to interperate a location. The calcualted locationw had a high error and there was clearly not a consistent lock. This experiment should have been redone ensuring that there was no interefecence in the Faraday cage causing the lock to be lost. The extent of the error and lack of data points are more evident in Figures 5.4 and 5.5.

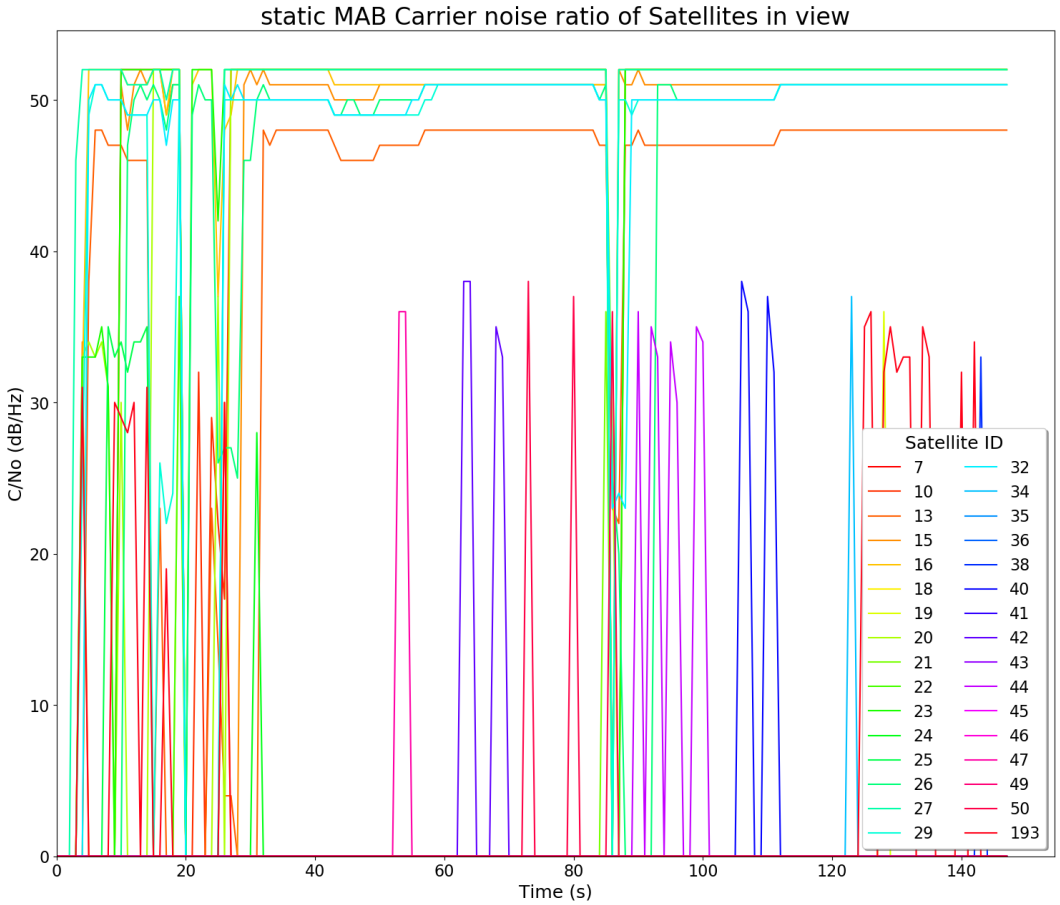

FIGURE 5.1: Carrier to noise ratio from unique SVIDs as broadcast by SDR and received by GPS receiver at MAB in Tonsley on 30th March 2021. Each SVID has been given a unique colour.

5.2.3.2 Antarctica

Davis station in Antarctica was chosen as a location for spoofing. As mentioned above the coordinates for this location was $-68.5762449235, 77.9696166515$. The COTS receiver was used as a spoof victim and it was successfully spoofed. Figure 5.6 shows the carrier to noise graph over the spoof attack. There is a

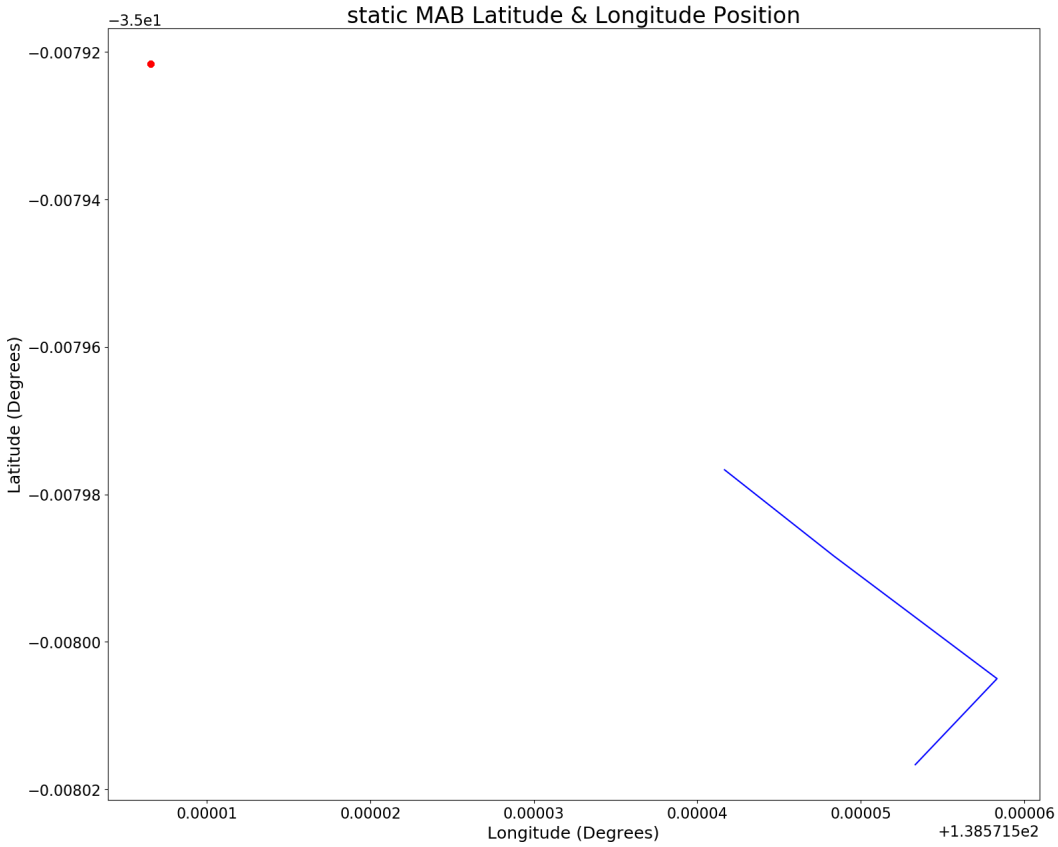

FIGURE 5.2: Trace of latitude and longitude over time as interpreted from the GPS receiver with expected location noted with red dot

consistent value of $\approx 50 dBHz$ plus signals that are consistently at $\approx 40 dBHz$. These values indicate low interference and high signal quality. There were more SVIDs (or PRN codes) than expected, with more SVIDs recoreded than there are GPS satellites in operation. Time to first fix was shown to be approximately 60 seconds, this is the lower end of the fix times from experiments for cold starts. It can be see that there is consistent signals for the first 50 seconds. For the time pereiod between 70 and 200 seconds there was more volatile signal quality. Figure 5.9 shows that there was a high, but consistent level of error between first fix and 140 seconds. After this there was a spike, which is associated with a drop of lock, and the high error remained until the end of the test. This high error is evident when viewing the raw position in Figure 5.7 and with respect to the desired location, Figure 5.8. Figure 5.10 shows this error in position in more context placed over a satellite image. The inaccuracy is higher than was expected from this experiment.

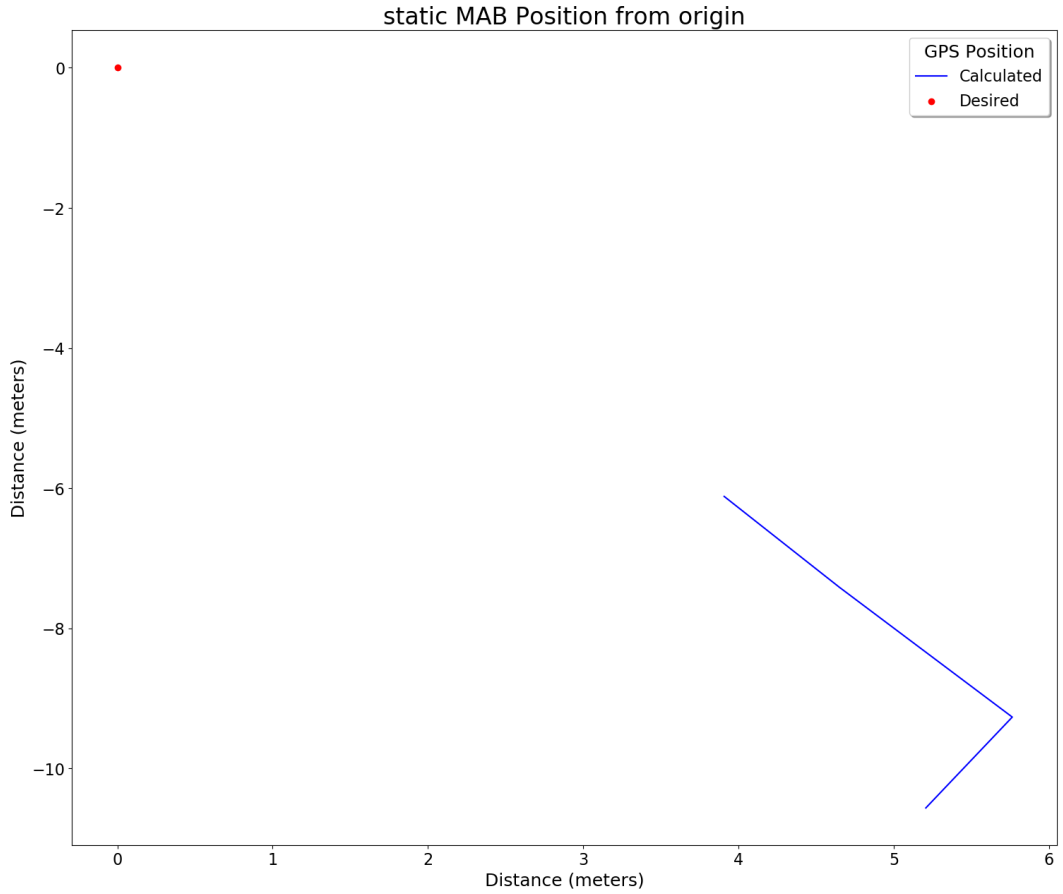

FIGURE 5.3: Graph of recorded position with respect to the intended spoofed location

5.2.4 Dynamic Spoofing

Dynamic spoofing attacks were also performed. These attacks do not have a set coordinate that the spoofer is attempting to make the receiver appear at, rather these attacks try to get the receiver to "follow" a path that has been set, see Section 4.6.1.

5.2.4.1 Tonsley MAB

There is a road that circles around the MAB (Mass Assembly Building) in which the Flinders UniversityTonsley campus is situated. This was used as a path to spoof the GPS receiver, see Figure 5.11.

The time to first fix was found to be approximately 49 seconds which is considered quick for a cold start. The $\frac{C}{N_0}$ ratio graph, see 5.12 shows a common trend of having a high average of $\approx 50 dBHz$ average with some lower values from spikes. These spikes would not add any value to the quality of the fix and given more time the cause of the spikes would be found. From

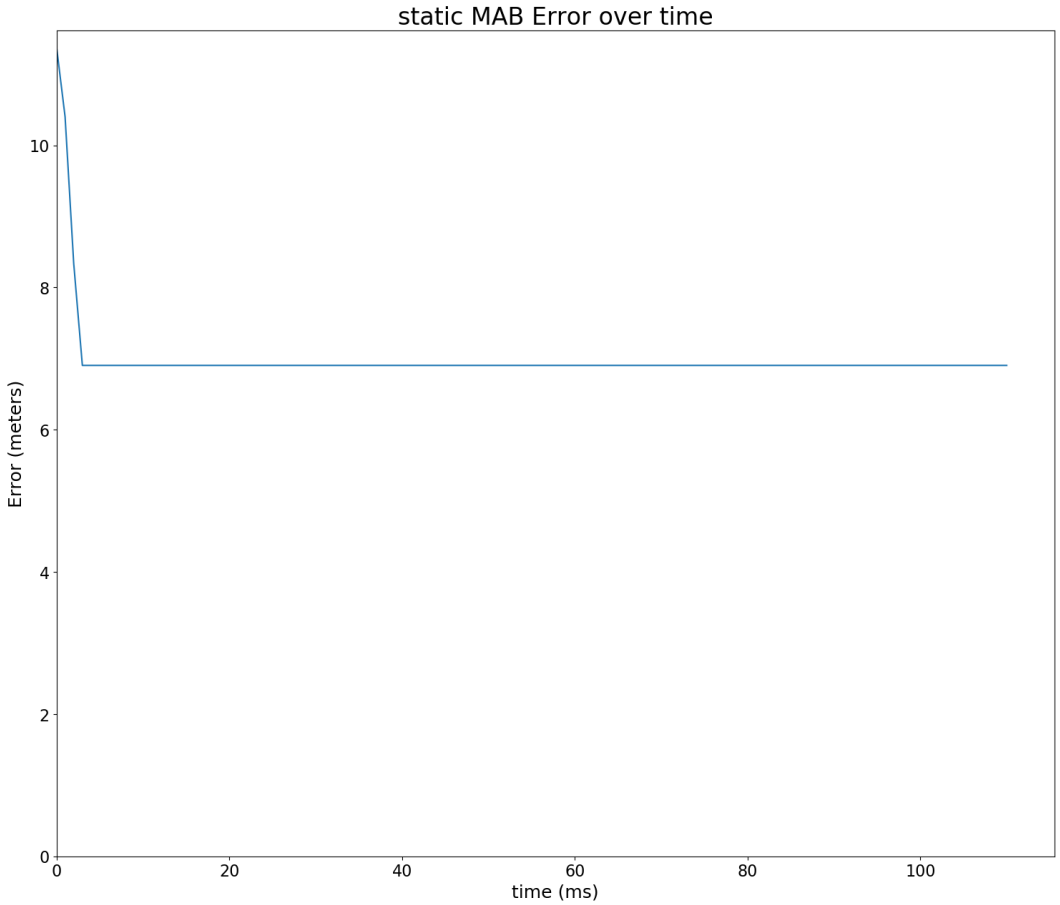

FIGURE 5.4: Error between calculated location and expected location over time

Figure 5.13 it can be seen that initially there was high amounts of error while getting the inital fix. Though when the movement began there was good tracking. This is evident in Figure 5.14 where the reference path and recevied path were overlayed. Overlaying onto a satellite image yielded 5.15 which shows how well the spoof attack was able to track the refence path. This produced results that would be expeccted from authentic GPS signals, with the exception being the inital part of the experiment.

5.3 Experimental Issues

While performing experiments there were issues that were run into that were required to be overcome. Initially a Log Periodic antenna was used for transmission since its frequency range was appropriate for use transmitting L1 band signals. After experiments failed to pick up any signals it was swapped for a omnidirectional antenna. This was able to produce signals that were

FIGURE 5.5: Satellite image of the MAB area next to Flinders University with expected GPS position shown with red cross and recorded position shown with blue line

picked up by the GNSS Logger application. Observing the graphs of the carrier to noise figure showed that when there was an under run issue the $\frac{C}{N_0}$ would drop to 0 and the GPS receiver would lose connection to the 'satellites'. This meant that no GPS lock was achievable.

The under run issue was more or less solved by increasing the buffer size to 40$\times$ greater than the sampling size. There was an issue towards the end of the project where there was considerable leakage of EM radiation into the Faraday cage that was causing the GPS receiver to produce wildly inaccurate position (up to and over 500m error). This was much different to what had previously been recorded within the Faraday cage. This amount of error does not constitute a successful spoof since the time to first lock was much greater than a real signal, or even spoofing attempts previously.

During initial testing of the spoofing there was an inability to get the spoofer to correctly spoof the receiver. It was found that there is a requirement that the ratio between the sampling frequency and clock frequency of the radio must be an integer value. Therefore when compiling the signal using GPS-SDR-sim the sampling frequency value was required to be changed to 2.5Msps.

Just after the testing phase of the project the smartphone that was being used for some of the logging and graphing was rendered unusable. While

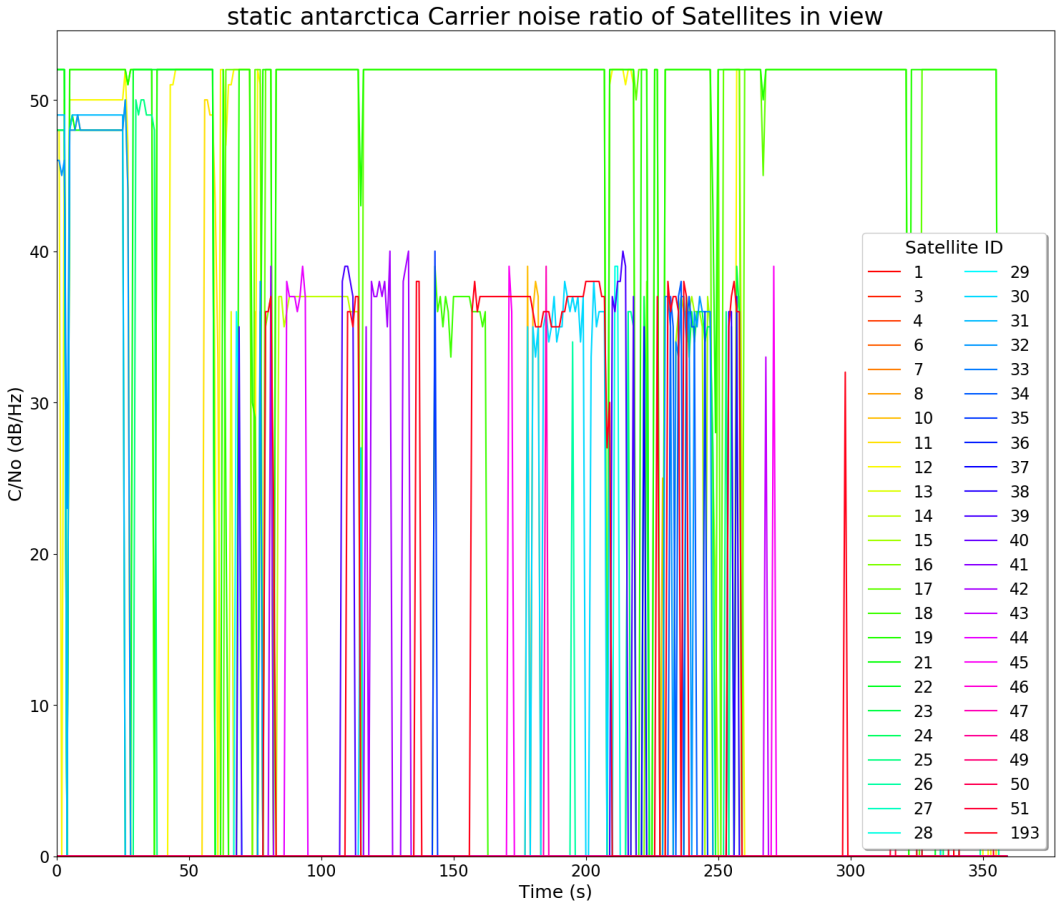

FIGURE 5.6: Carrier to noise ratio from unique SVIDs as broadcast by SDR and received by GPS receiver at Davis Station in Antarctica on 30th March 2021. Each SVID has been given a unique colour.

there was no data lost, it was replaced with a newer model that experimentally was much harder to spoof than the previous model. This could be due to many factors including anti-spoofing algorithms [22]. It would be a fair assessment that the newer device is able to receive more GNSS signals including augmented systems as well as other signals such as Wi-Fi. The connection to the Wi-Fi signal was such that notifications were coming up on the phone while performing the experiments.

There were a number of unterminated cables that were running from outside of the cage to inside. This was seen as being the cause of the issues that were being faced. The suspect cables were removed or terminated and the results were more consistent with what was being achieved previously. Even with the new phone the GPS location was able to be spoofed.

A common result from spoofed attempts was to have GPS satellites with an SVID of 193. GPS satellites have PRN codes from 1-63 and SBAS have

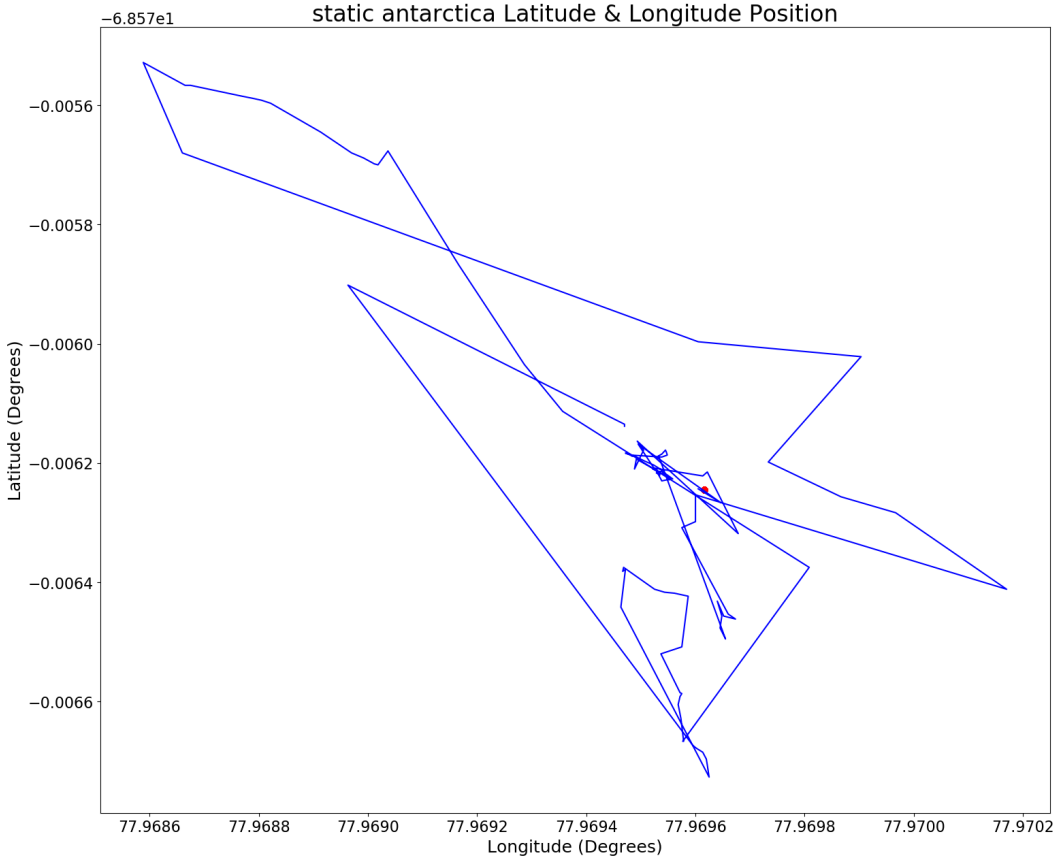

FIGURE 5.7: Trace of latitude and longitude over time as interpreted from the GPS receiver with expected location noted with red dot

PRNs from 64 to 158 [35]. From the list of PRN codes [35] it can be seen that PRN 193 is used by one of the QZSS satellites. This is likely a result of stray EM waves penetrating the Faraday cage. It is unlikely that the GPS-SDR-sim program includes augmenetation systems when geneating spoofed signals. The orbit of the QZSS satellites does include Australia so seems most likey an issue where an unterminated cable is allowing the signal to bleed into the cage.

In an attempt to overcome the under run issue that was plaguing the experiments, a new PC was brought in with Ubuntu installed directly instead of via a VM. This resulted in the radio working straight away. The under run issues resurfaced when trying to read the serial data from a COTS GPS receiver. This was less than ideal and required a second computer to act as a datalogger.

insert graph of under run

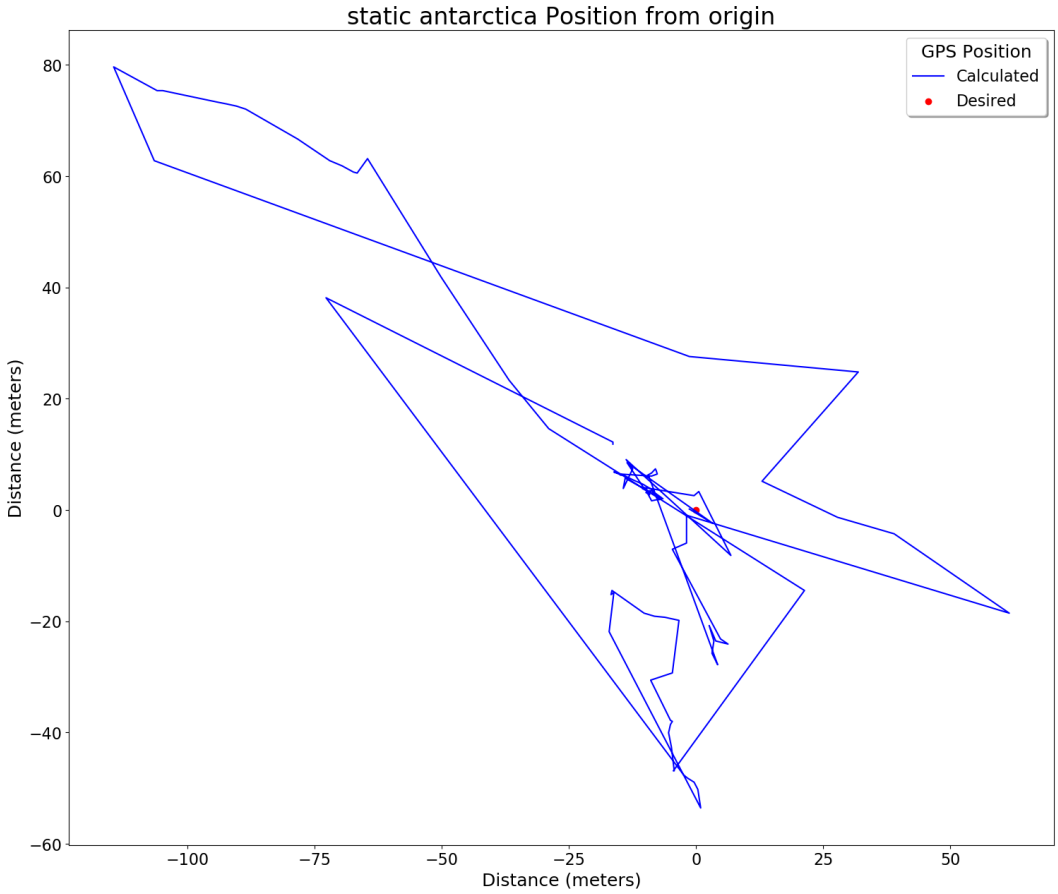

FIGURE 5.8: Graph of recorded position with respect to the intended spoofed location

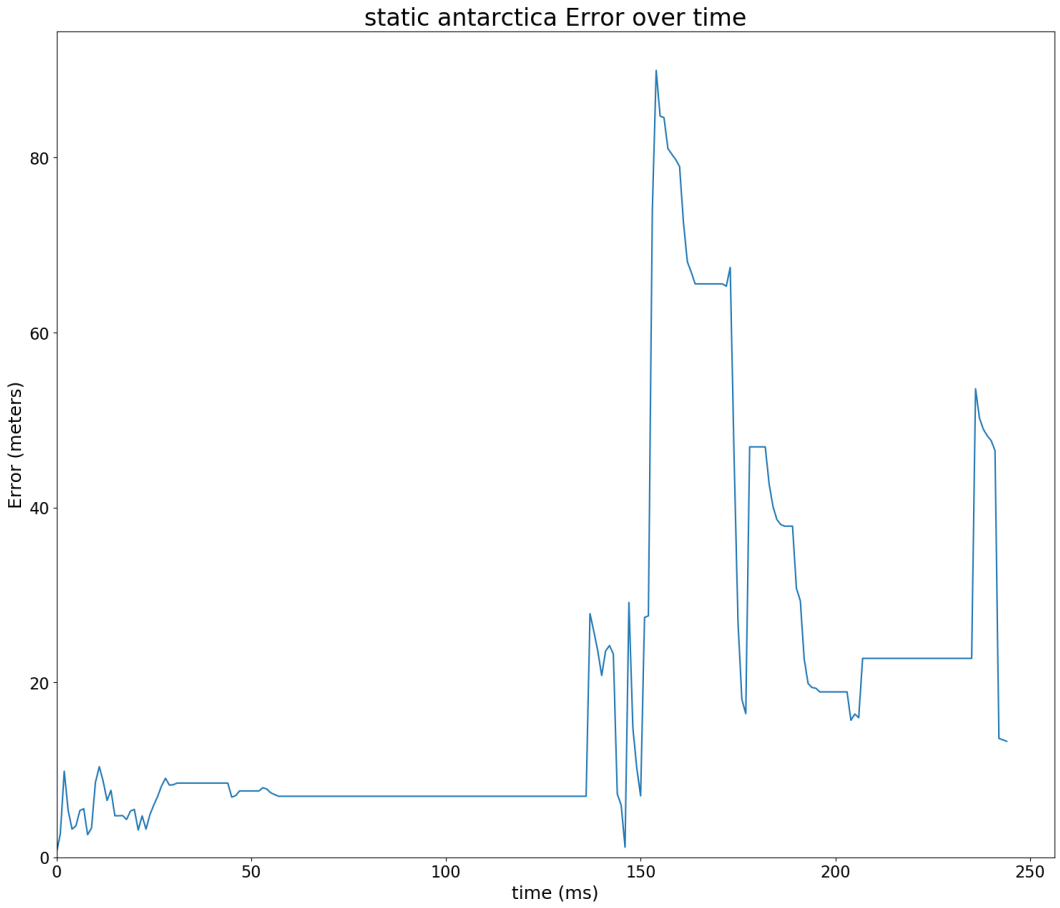

FIGURE 5.9: Error between calculated location and expected location over time

FIGURE 5.10: Satellite image with expected GPS position shown with red cross and recorded position shown with blue line

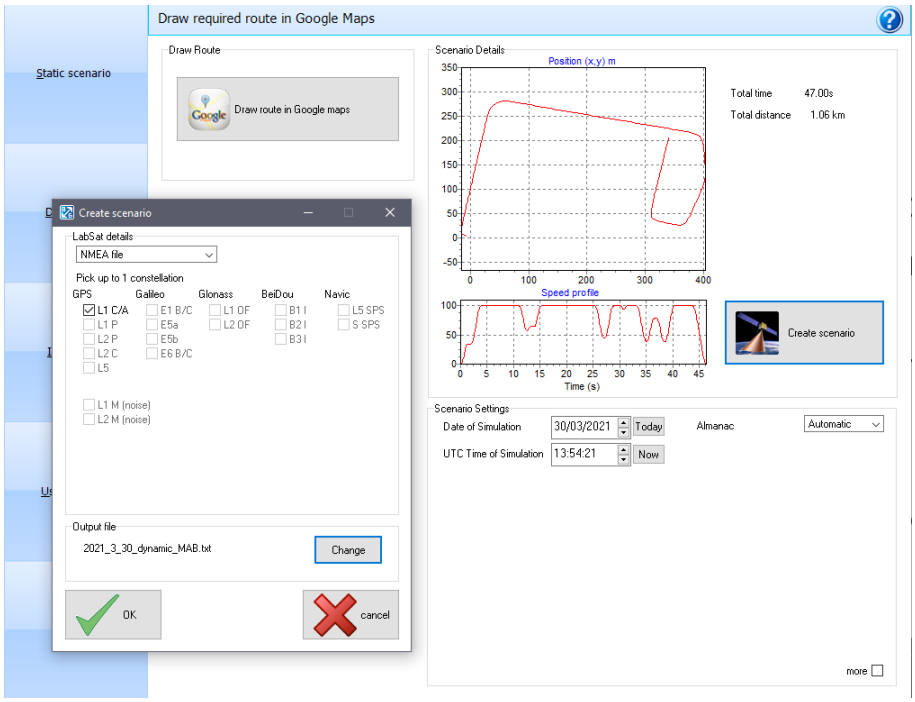

FIGURE 5.11: Using road around the MAB at Tonsley, Adelaide as path for dynamic spoofing attack. The file was saved as a text file to be compiled into a binary.

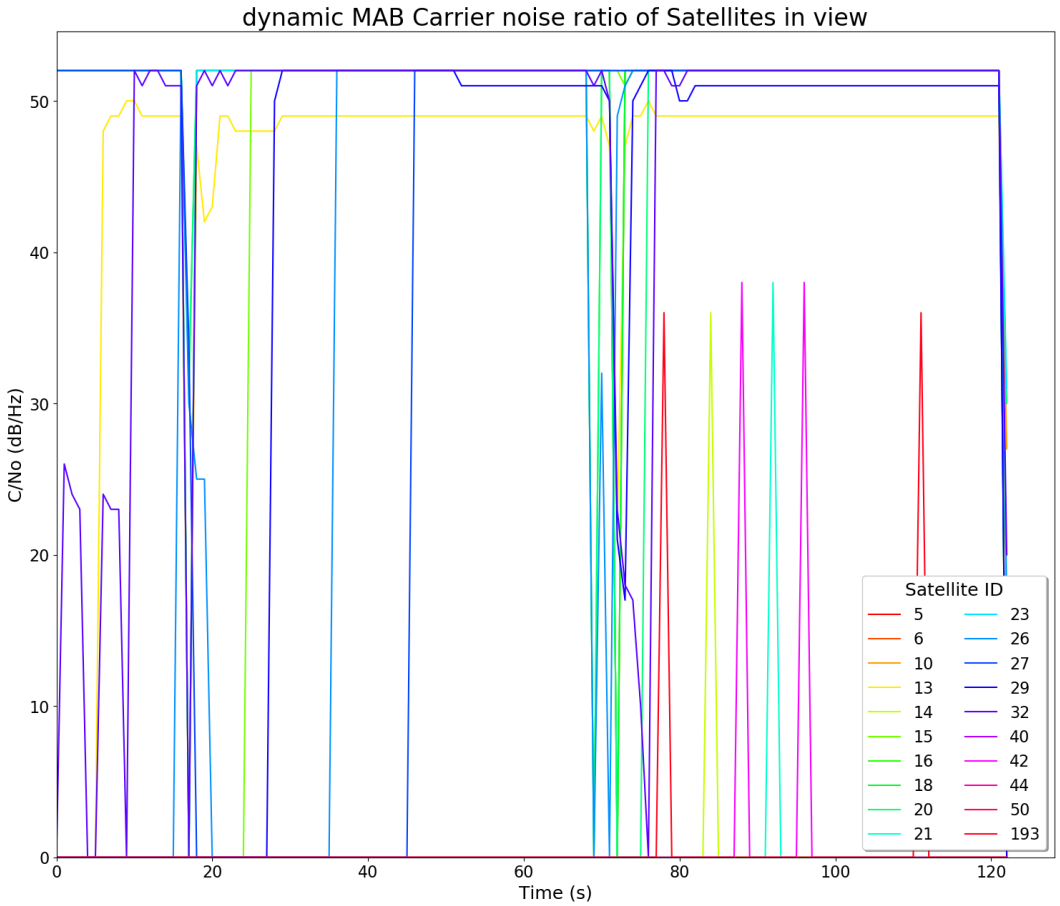

FIGURE 5.12: Carrier to noise ratio from unique SVIDs as broadcast by SDR and received by GPS receiver at MAB in Tonsley on 30th March 2021. Each SVID has been given a unique colour.

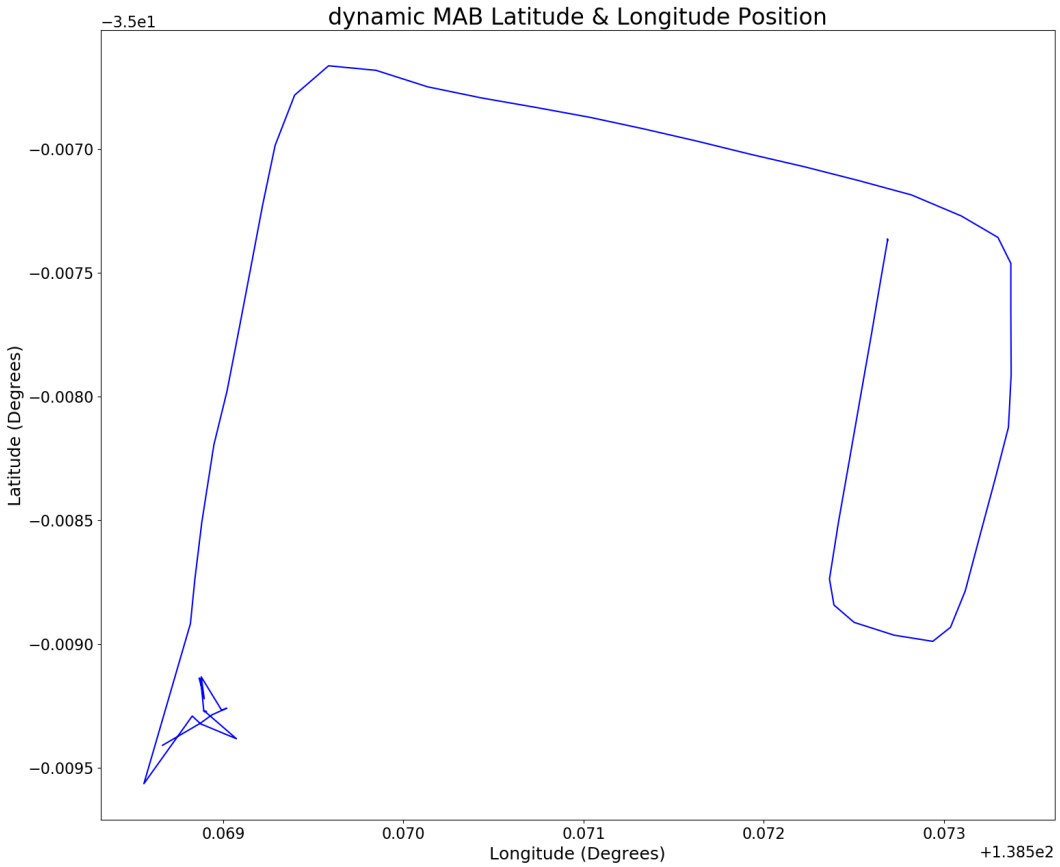

FIGURE 5.13: Trace of latitude and longitude over time as interpreted from the GPS receiver with expected location noted with red dot

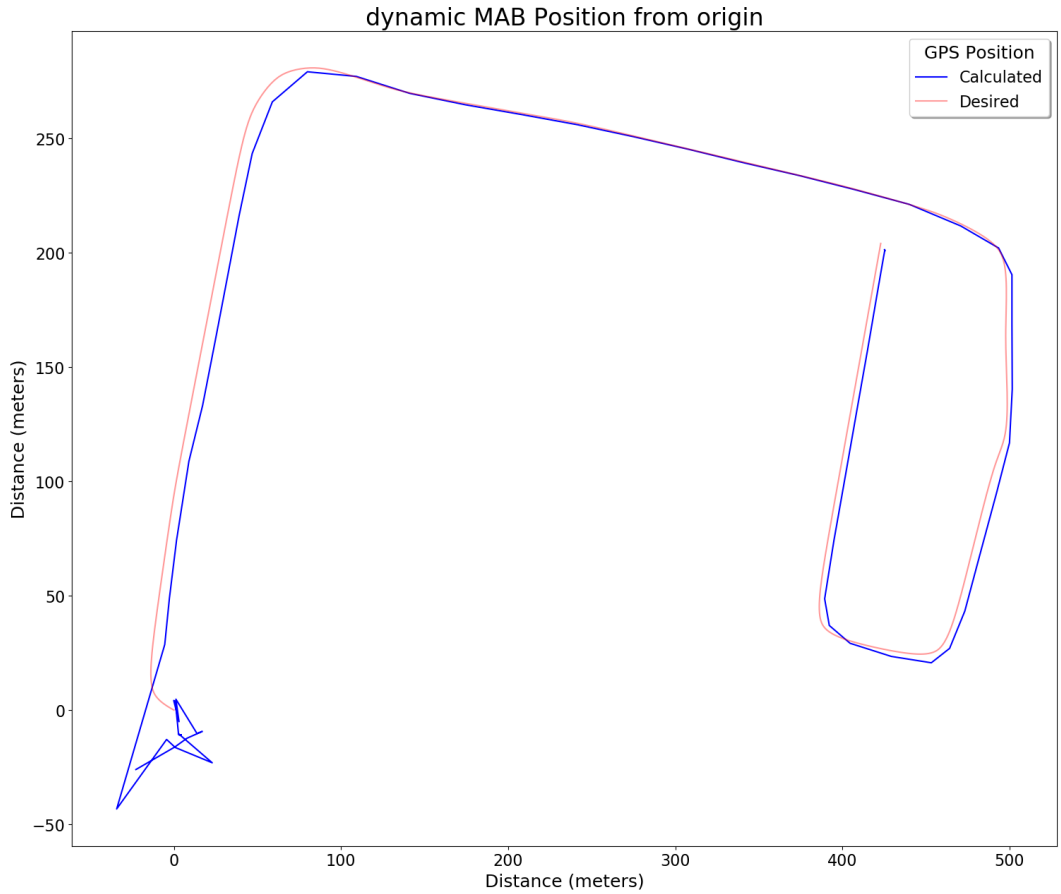

FIGURE 5.14: Graph of recorded position with respect to the intended spoofed location

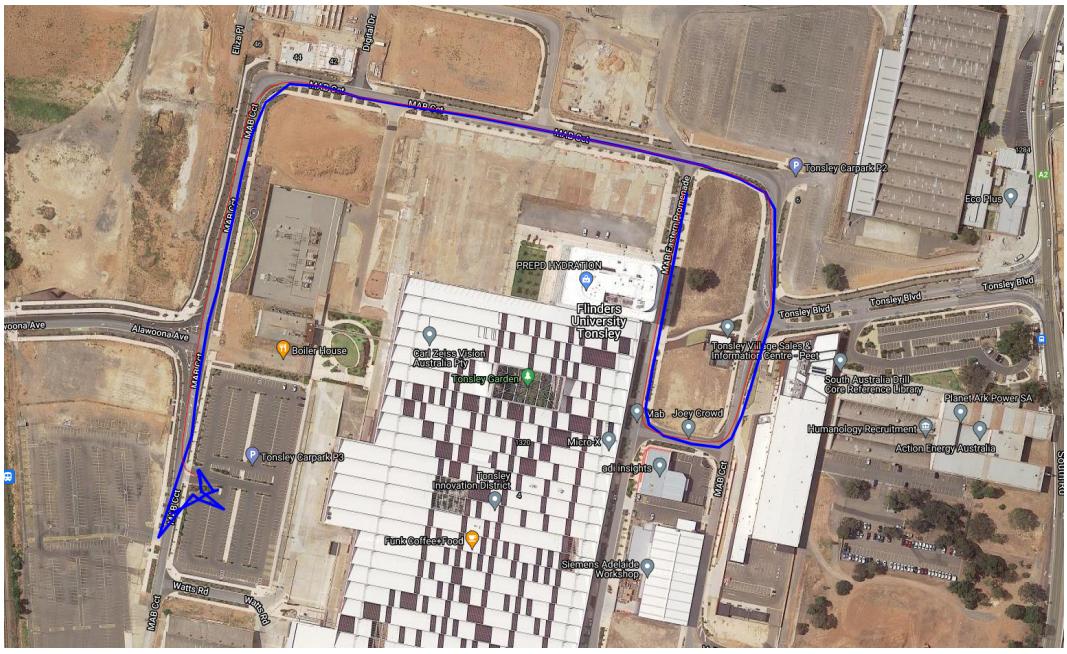

FIGURE 5.15: Satellite image with expected GPS position shown with red cross and recorded position shown with blue line

Chapter 6

Discussion

This section will discuss the results gathered from the previous section 5. Any broad findings or trends will be noted and explanations of why results are the way they are.

6.1 Results

Initially it was decided that the locations that would be spoofed would be locations that are not easily accessible as a means of proving the validity of the results. This was made easier due to the ongoing pandemic. Although this idea was later changed to make it easier to compare with real signals if needed.

Using the developed workflow, it was simple to create and deploy spoofing attacks for any location and at any elapsed time within minutes. This includes static or dynamic spoofing attacks. Not enough time was invested into the reception of GNSS signals and thus meaconing attacks were not performed. The GPS-SDR-sim program was able to estimate and simulate the ionic effects. While it is clear that this method resulted in successful spoofing attacks, the hypothesis is that having a received signal that includes ionic delays and other interference would make it easier to fool more complicated receivers. Further testing and investigations should be carried out to find the answer to this.

The results from the static Antarctica spoof attack, specifically the carrier to noise graphs show a high number of unique SVIDs. Based on the number of satellites in the GPS constellation this must be an error. Although there was no explanation for the error since the resultant location was correct. One hypothesis was that the compilation software added some incorrect data. Another is the prevalence of SBAS satellites. There was a known issue for some time with the Faraday cage where stray EM waves were getting in. This would explain the consistency in which SVID 193 appears. From [35]

it can be seen that SVID 193 is a QZSS satellite. These satellites orbit over Australia and part of Asia.

Consistently for both static and dynamic spoofing the $\frac{C}{N_0}$ ratio was a high value (between 40 and 50 dBHz). During experimentation it was found that authentic GPS signals would be received with more variable $\frac{C}{N_0}$ ratio but with an average of arround 30dBHz. This discrepency can be attributed to a combination of the controlled environment of the Faraday cage as well as the reletive distance between the spoofer and victim. In theory the spoofed $\frac{C}{N_0}$ ratio could be higher, but during compilation estimated Ionospheric effects are included. This shows that there is still room for improvement for the GPS-SDR-sim software, especially if it is to be used in controlled environments for testing purposes. Just from the $\frac{C}{N_0}$ ratio graphs the spoofed signals are easy to identify.

On the $\frac{C}{N_0}$ ratio graphs there were consistently more SVIDs than expected. The trend was that static spoof attacks had more SVIDs than the dynamic spoofs, with the most coming from the Antarctic spoof attack. It can also be seen that the length of attack was longest for the Antarctic spoof, which may account for this discrepency. Even still, the MAB static spoof included more than expected SVIDs.

SDRs are very sensitive to the processing pipeline of the computer that they are connected to. It is not uncommon to receive an under run error (where 'U' is printed to the output) when transmitting a signal with high SPS (samples per second). This error is caused by the host computer not being able to feed the data to the radio quickly enough.

This was solved by increasing the UDP buffer size manually to at least the sample rate, and over for better results. From limited experimentation this completely resolved the issue, and opened up opportunities to perform spoofing with low powered embedded devices like the Raspberry Pi single board computer.

6.2 Future Work

All of the tests that have been conducted for use in this thesis have either been generating a signal based off a predetermined location (coordinate) or a predetermined path (group of connected coordinates). None of these options have a mechanism for position feedback. For example you cannot create a signal that has a linear offset from the actual position using these methods.

This is something that could be achieved through combination and modification of the existing open source projects used in this thesis. This would allow for properly real-time GPS spoofing, and could be extended further to have intelligent algorithms.

Current methods, as detailed in Chapter 4, requires downloading the Ephemeris for the date and time of the proposed spoofing attack. This makes real-time spoofing attacks not possible since there is a delay between the current time and the associated ephemeris. A potential fix for this could be to perform some analysis on the orbits of each satellite within the constellation over an arbitrary period of time to create a machine learning algorithm. This algorithm would be able to predict the location of the satellites ahead of time, thus binaries could be compiled for future attacks, or could be used for real-time attacks without the need for downloading/ retrieving the ephemeris from legitimate sources.

The logging program could be extended to provide real time feedback on position and signal quality. This would be achieved by implementing the serial connection within the Python code itself rather than relying on the logging function of screen. Extended to work on anti-spoofing for PhD. If there was more time for this project, a software based anti spoofing program would have been easy to implement based on the $\frac{C}{N_0}$ ratio alone. This could be implemtnted in conjucntion with real time graphing as mentioned above.

The hardware used for this project was successfully used for spoofing attacks, however, the software must be altered in order to achieve the accurate results that were expected of the project. However ensuring software capabilities was out of the scope of this thesis.

Chapter 7

Conclusion

One of the aims of this project was to create hardware that is capable of successfully spoofing a GPS receiver. From the results it can be seen that the method of spoofing using signals generated from coordinates was successful. There were issues that impacted the results at times, however, these were overcome to produce good results in both static and dynamic spoofing. The hardware that was used for this is capable of performing any spoofing attack, however the software was not cable to produce the results at times, where anti-spoofing was at play. Overall the accuracy of the position gathered from the GPS receivers was poorer than expected, although to be able to spoof anything at all was considered a success. Given these results were gathered in a controlled environment more care will need to be taken if performing these experiments in the field. The project was able to provide a foundation for which further research into GNSS attack and defence strategies can be performed. SDR devices are prevalent in EW research due to their combination of hardware level performance and software level flexibility. This will allow for further research into other areas of EW. There were no results gathered from performing Meaconing attacks since the GPS signal reception was unsuccessful and therefore signal capture was unavailable.

Through this thesis it has been shown that with relatively low amounts of technical knowledge successful spoofing attacks can be performed. Some basic computer literacy was required in order to install and configure the software needed to perform the attacks. The simplest way of using the software is with Linux which does add a level of difficulty. Moving forward into the future the software could be given a graphical interface and integrated into one software package, or simplify the transmission interface and use pre compiled binaries.

Appendix A

Project Code

A.1 GPS Sentence Parsing and Graphing

Filename: NMEA.py
Description: This code will take a NMEA file as input and calculate the time to first fix, then plot the carrier to noise ratio of every satellite that was within view of the receiver, then plot the position given the output of the GPS receiver.

```
#!/bin/python3

import sys
import csv
from datetime import datetime, time
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import numpy as np
import gmplot
import math


def degMin2deg(x):
    x = float(x)
    return (int(x/100)) + ((x%100)/60)


def radians(degrees):
    return degrees*math.pi/180

```

```
22
23 def latlon2m(lat1, long1, lat2, long2):
24     r = 6371000 # this is the radius of the earth
25     dLat = radians(lat2) - radians(lat1) # change in latitude in
   radians
26     dLong = radians(long2) - radians(long1) # change in longitude
    in radians
27     a = math.sin(dLat/2) * math.sin(dLat/2) + math.cos(radians(
   lat1))*math.cos(radians(lat2))*math.sin(dLong/2)*math.sin(
   dLong/2)
28     d = 2*r*math.asin(math.sqrt(a)) # distance in meters
29     return d
30
31
32 def plotPath(coordiantes, origin=None): # coordinates stored as
   timestamp | Lat | Long
33     x = []
34     y = []
35     refX = []
36     refY = []
37
38     for coordinate in coordiantes:
39         y.append(coordinate[1])
40         x.append(coordinate[2])
41
42     fig, ax = plt.subplots(figsize=(18 ,15))
43     ax.plot(x, y, color='blue')
44     if origin:
45         if not dynamic:
46             for coordinate in origin:
47                 refY.append(coordinate[1])
48                 refX.append(coordinate[2])
49             ax.scatter(refX, refY, color='red')
50
51     plot_title = title + " Latitude & Longitude Position"
52     ax.set(xlabel = 'Longitude (Degrees)', ylabel = 'Latitude (
   Degrees)', title = plot_title)
```

```
53      fig.savefig(("../Images/" + datetitle + " Lat long position.
    png"), bbox_inches='tight')
54      plt.tight_layout()
55      plt.show()
56
57
58  def plotDistance(coordiantes, reference=None): # coordinates
    stored as timestamp | Lat | Long
59      poleCircumference = 40007.863 # circumference of earth along
    polar plane in kms
60      equaCircumference = 40075.017 # circimference of earth along
    equatorial plane in kms
61      lats = [] # distance from origin lat measurement in m
62      longs = [] # distance from origin long measurement in m
63      refLats = []
64      refLongs = []
65
66      if reference: # if the refernce file has been provided use
    that for reference
67          origin = [reference[0][1], reference[0][2]]
68          if dynamic:
69              for coords in reference:
70                  dLat = (coords[1] - origin[0]) * ((
    poleCircumference/2)/180) * 1000
71                  dlong = (coords[2] - origin[1]) * ((
    equaCircumference/4)/90) * 1000
72                  refLats.append(dLat)
73                  refLongs.append(dlong)
74      else:
75          origin = (coordinates[0][1], coordiantes[0][2])
76
77      for coordinate in coordiantes:
78          dLat = (coordinate[1] - origin[0]) * ((poleCircumference
    /2)/180) * 1000
79          dlong = (coordinate[2] - origin[1]) * ((equaCircumference
    /4)/90) * 1000
80          lats.append(dLat)
81          longs.append(dlong)
```

```
82
83      fig, ax = plt.subplots(figsize=(18 ,15))
84      ax.plot(longs, lats, color='blue', label="Calculated")
85      if dynamic:
86          ax.plot(refLongs, refLats, color='red', alpha=0.4, label=
        "Desired")
87      else:
88          ax.scatter(0, 0, color='red', label="Desired")
89      plot_title = title + " Position from origin"
90      ax.set(xlabel = "Distance (meters)", ylabel = "Distance (
        meters)", title = plot_title)
91      ax.legend(title="GPS Position", loc='upper right', shadow=
        True)
92      fig.savefig(("../Images/" + datetitle + " Position from
        origin.png"), bbox_inches='tight')
93      plt.tight_layout()
94      plt.show()
95
96
97  def plotError(coordinates, reference):
98      error = []
99      time = range(len(coordinates))
100     origin = [reference[0][1], reference[0][2]]
101     if dynamic:
102         return
103
104     for coord in coordinates:
105         error.append(latlon2m(origin[0], origin[1], coord[1],
        coord[2]))
106     fig, ax = plt.subplots(figsize=(18 ,15))
107     ax.plot(time, error)
108     plot_title = title + " Error over time"
109     ax.set(xlabel = "time (ms)", ylabel = "Error (meters)", title
        = plot_title)
110     ax.set_ylim(ymin=0)
111     ax.set_xlim(xmin=0)
112     fig.savefig(("../Images/" + datetitle + " error over time.png
        "), bbox_inches='tight')
```

```
113     plt.tight_layout()
114     plt.show()
115 
116 def plotMap(coordiantes, reference=None):
117     lats = []
118     longs = []
119     apikey = 'AIzaSyDmP3BcjpPRtxNKqPjX9Q7sSyAW96afCB4'
120 
121     for coordinate in coordiantes:
122         lats.append(coordinate[1])
123         longs.append(coordinate[2])
124 
125     avgLats = (min(lats) + max(lats)) / 2
126     avgLongs = (min(longs) + max(longs)) / 2
127 
128     print("Reference point for map: Lat: {}, Long: {}".format(
    avgLats, avgLongs))
129 
130     filename = title + "googleMap"
131 
132     gmap = gmplot.GoogleMapPlotter(avgLats, avgLongs, 18, apikey=
    apikey)
133     gmap.plot(lats, longs, edge_width=4, color='blue')
134     if reference:
135         if dynamic:
136             refLats = []
137             refLongs = []
138             for coords in reference:
139                 refLats.append(coords[1])
140                 refLongs.append(coords[2])
141             gmap.plot(refLats, refLongs, color='red')
142         else:
143             refLats = [reference[0][1]]
144             refLongs = [reference[0][2]]
145             gmap.scatter(refLats, refLongs, symbol='+', marker=
    False, color='red', size=2)
146     gmap.draw('../Images/{}.html'.format(filename))
147 
```

```
148
149 def plotCNo(satellites):
150     fig, ax = plt.subplots(figsize=(18 ,15))
151     timestamps = list(satellites.keys())
152     x = range(len(timestamps))
153     SatIDs = satellites.values()
154     tmp = []
155
156     for sats in SatIDs:
157         for sat in sats:
158             if int(sat) not in tmp:
159                 tmp.append(int(sat))
160     tmp.sort()
161
162     ax.set_prop_cycle(plt.cycler("color", plt.cm.hsv(np.linspace
    (0,1,len(tmp)))))
163     for count, sv in enumerate(tmp):
164         y = []
165         for sats in SatIDs:
166             for sat in sats:
167                 if str(sv) in sats:
168                     if sat == str(sv):
169                         try:
170                             y.append(float(sats[sat][0]))
171                         except ValueError:
172                             y.append(0)
173                         break
174                 else:
175                     y.append(0)
176                     break
177         ax.plot(x, y, label=sv)
178     plot_title = title + " Carrier noise ratio of Satellites in
    view"
179     ax.set(xlabel = 'Time (s)', ylabel = 'C/No (dB/Hz)', title =
    plot_title)
180     ax.set_ylim(ymin=0)
181     ax.set_xlim(xmin=0)
```

```
182     ax.legend(title="Satellite ID", loc='lower right', shadow=
        True, ncol=2)
183     fig.savefig(("../Images/" + datetitle + " Carrier noise ratio
        .png"), bbox_inches='tight')
184     plt.tight_layout()
185     plt.show()
186 
187 if len(sys.argv) > 2:
188     sourceFile = sys.argv[2] # this is the file that was used to
        generate the binary file for the SDR
189 infile = sys.argv[1] # this is the file from the GPS receiver
190 timetofix = 0
191 dataValid = False
192 firstFix = False
193 svSNR = {}
194 satellites = {} # Make this a dict in a dict. {Timestamp: {SVid:
        SNR[]}}
195 currentTime = 0
196 startTime = datetime.now()
197 coordinates = [] # coordiantes stored as timestamp | Lat | Long
198 referenceCoords = [] # coordinates that act as reference
199 dynamic = False # is the file being graphed a static or dynamic
        spoof scenario
200 title = ""
201 
202 if '.txt' in infile:
203     datetitle = infile[:-4]
204 elif '.nmea' in infile:
205     datetitle = infile[:-5]
206 else:
207     raise NameError("Input file is not in correct format")
208 
209 chunkedtitle = datetitle.split("_")
210 
211 if 'dynamic' in infile:
212     dynamic = True
213     indexer = chunkedtitle.index("dynamic")
214 else:
```

```
215     indexer = chunkedtitle.index("static")
216 
217 title = chunkedtitle[indexer] + " " + chunkedtitle[indexer + 1]
218 
219 try:
220     with open(infile, "r") as nmeaFile:
221         for count, line in enumerate(nmeaFile):
222             tmp = line.split('*')
223             string = tmp[0].split(',')
224             msgType = string[0][3:]
225             currentTime = datetime.now()
226             timestamp2 = (currentTime - startTime).microseconds
    /1000000
227             #
228             # Parsing RMC sentences
229             #
230             if msgType == "RMC":
231                 if not dataValid:
232                     if not firstFix:
233                         timetofix += 1
234                 if string[2] == 'A':
235                     dataValid = True
236                     if not firstFix:
237                         firstFix = True
238                 else:
239                     dataValid = False
240             #
241             # Parsing GSV sentences
242             #
243             elif msgType == "GSV":
244                 NumofMessages = int(string[1])
245                 sequenceNumber = int(string[2])
246                 satellitesView = int(string[3])
247                 satellitesToCalc = satellitesView-(NumofMessages
    -1)*4 # this will dictate the number the index numbers of
    final sentence of group
248                 if sequenceNumber == 1:
```

```
249                     timestamp = (currentTime - startTime).
    microseconds/1000000
250                 try:
251                     if sequenceNumber < NumofMessages:
252                         if string[4] not in svSNR:
253                             svSNR[string[4]] = [string[7]]
254                         else:
255                             svSNR[string[4]].append(string[7])
256 
257                         if string[8] not in svSNR:
258                             svSNR[string[8]] = [string[11]]
259                         else:
260                             svSNR[string[8]].append(string[11])
261 
262                         if string[12] not in svSNR:
263                             svSNR[string[12]] = [string[15]]
264                         else:
265                             svSNR[string[12]].append(string[15])
266 
267                         if string[16] not in svSNR:
268                             svSNR[string[16]] = [string[19]]
269                         else:
270                             svSNR[string[16]].append(string[19])
271                     else:
272                         if satellitesToCalc >= 1:
273                             if string[4] not in svSNR:
274                                 svSNR[string[4]] = [string[7]]
275                             else:
276                                 svSNR[string[4]].append(string
    [7])
277                         if satellitesToCalc >= 2:
278                             if string[8] not in svSNR:
279                                 svSNR[string[8]] = [string[11]]
280                             else:
281                                 svSNR[string[8]].append(string
    [11])
282                         if satellitesToCalc >=3:
283                             if string[12] not in svSNR:
```

```
284                                 svSNR[string[12]] = [string[15]]
285                             else:
286                                 svSNR[string[12]].append(string
    [15])
287                         if satellitesToCalc == 4:
288                             if string[16] not in svSNR:
289                                 svSNR[string[16]] = [string[19]]
290                             else:
291                                 svSNR[string[16]].append(string
    [19])
292                         if satellitesToCalc > 4:
293                             raise OverflowError
294 
295                 except IndexError:
296                     pass
297                 except OverflowError:
298                     print("Computational error occured. Access to
     a value higher than available attempted")
299                 if sequenceNumber == NumofMessages:
300                     if svSNR:
301                         satellites[timestamp] = svSNR
302                     svSNR = {}
303 
304             #
305             # Parsing GGA sentences
306             #
307             elif msgType == "GGA":
308                 if dataValid:
309                     try:
310                         latitude = degMin2deg(string[2])
311                         longitude = degMin2deg(string[4])
312                         if string[3] == 'S':
313                             latitude = latitude * -1
314                         if string[5] == 'W':
315                             longitude = longitude * -1
316                         coordinates.append([timestamp2, latitude,
    longitude])
317                     except ValueError:
```

```
318                         print("No valid coordinates")
319                         continue
320
321 except IOError:
322     print("could not open file. try again")
323     sys.exit()
324
325 try:
326     with open(sourceFile, "r") as refFile:
327         for count, line in enumerate(refFile):
328             tmp = line.split('*')
329             string = tmp[0].split(',')
330             msgType = string[0][3:]
331             currentTime = datetime.now()
332
333             if msgType == "GGA":
334                 try:
335                     latitude = degMin2deg(string[2])
336                     longitude = degMin2deg(string[4])
337                     if string[3] == 'S':
338                         latitude = latitude * -1
339                     if string[5] == 'W':
340                         longitude = longitude * -1
341                     referenceCoords.append([(currentTime -
    startTime).microseconds/1000, latitude, longitude])
342                 except ValueError:
343                     print("No valid coordinates")
344                     continue
345 except IOError:
346     print("Reference File not provided")
347     pass
348
349 print("Time to first fix was {} seconds".format(timetofix))
350 text_params = {'axes.labelsize': 18, 'axes.titlesize' : 24, '
    xtick.labelsize': 16, 'ytick.labelsize': 16, 'legend.
    title_fontsize' : 18, 'legend.fontsize' : 16} # text
    parameters for plotting
351 plt.rcParams.update(text_params)
```

```
352 plotCNo(satellites)
353 plotPath(coordinates, referenceCoords)
354 plotDistance(coordinates, referenceCoords)
355 plotMap(coordinates, referenceCoords)
356 plotError(coordinates, referenceCoords)
```

LISTING A.1: Python code to parse the receiver NMEA sentence and produce graphs

A.2 GPS Ephemeris download script

Filename: remote_ephemeris_utc.py
Description: This code will take the current time and and use that to request the latest ephemeris from the CDDIS portal.

```
1  import requests
2  import sys
3  from datetime import datetime
4  from datetime import timezone
5
6  """
7  Ensure that the .netrc file is in the home directory for the user
       otherwise this will not work.
8  The file should be named _netrc for use on windows systems
9  This script will download the latest BRDC file from NASA
10
11 This can be extended to allow for supplying a specific date in
      the command line paramerters to set
12 specific dates that the ephemeris will be downloaded for
13 """
14 today = datetime.utcnow().date()
15 base_url = "https://cddis.nasa.gov/archive/gnss/data/daily/"
16 YYYY = today.year # current year in a four digit format #replace
      with utc bnased date
17 YY = str(YYYY)[2:] # current year in two digit format
18 DDD = today.timetuple().tm_yday # day of the year in 3 digit
      format
19 Z = "gz" # unix compression type
```

```
20 url = "{base}{yyyy}/{ddd:03}/{yy}n/brdc{ddd:03}0.{yy}n.{z}".
       format(base=base_url, yyyy=YYYY, ddd=DDD, yy=YY, z=Z)
21 filename = "./ephemerisData/{}".format(url.split('/')[-1])
22
23 r = requests.get(url)
24 print(r.status_code)
25 if (r.status_code == '404'):
26     print("File not found, may not be uploaded yet. Try again
       later")
27 else:
28     with open(filename, 'wb') as fd:
29         for chunk in r.iter_content(chunk_size=1000):
30             fd.write(chunk)
31         print("Ephemeris saved as: {}".format(filename))
```

LISTING A.2: Python code to retreive the latest RINEX file from the CDDIS servers

A.3 Connect to GPS Receiver

Filename: Connect_receiver.sh
Description: Script that was used to set the permissions on the USB GPS receiver and then connect to the serial interface using the screen program. The script would take 1 argument that was used to save the logfile.

```
1 #!/bin/bash
2
3 filename=$1
4
5 chmod 777 /dev/ttyUSB0
6 screen -L -Logfile $filename /dev/ttyUSB0
```

LISTING A.3: Bash script to connect to USB GPS receiver

Bibliography

[1] Geoffrey Blewitt. "Basics of the GPS technique: observation equations". In: *Geodetic applications of GPS* (1997), pp. 10–54. ISSN: 0280-5731.

[2] Norman Bonnor. "A Brief History of Global Navigation Satellite Systems". In: *Journal of navigation* 65.1 (2012), pp. 1–14. ISSN: 0373-4633. DOI: 10.1017/S0373463311000506.

[3] Government Of Japan Cabinet Office. *Overview of the Quasi-Zenith Satellite System (QZSS)*. 2021. URL: https://qzss.go.jp/en/overview/services/sv01_what.html.

[4] Government Of Japan Cabinet Office. *What is the Quasi-Zenith Satellite System (QZSS)?* 2021. URL: https://qzss.go.jp/en/overview/services/sv02_why.html.

[5] N. Carson et al. "GPS spoofing detection and mitigation using Cooperative Adaptive Cruise Control system". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. 2016, pp. 1091–1096. DOI: 10.1109/IVS.2016.7535525.

[6] Robert J Danchik. "An overview of transit development". In: *Johns Hopkins APL technical digest* 19.1 (1998), p. 19.

[7] Gonzalo De La Torre, Paul Rad, and Kim-Kwang Raymond Choo. "Driverless vehicle security: Challenges and future research opportunities". In: *Future Generation Computer Systems* 108 (2020), pp. 1092–1111. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2017.12.041. URL: http://www.sciencedirect.com/science/article/pii/S0167739X17315066.

[8] G Arul Elango and GF Sudha. "Design of complete software GPS signal simulator with low complexity and precise multipath channel model". In: *Journal of electrical systems and information technology* 3.2 (2016), pp. 161–180. ISSN: 2314-7172.

[9] Carles Fernandez-Prades et al. "GNSS-SDR: An open source tool for researchers and developers". In: *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, pp. 780–794.

[10] I. Fernández-Hernández and G. Seco-Granados. "Galileo NMA signal unpredictability and anti-replay protection". In: *2016 International Conference on Localization and GNSS (ICL-GNSS)*. 2016, pp. 1–5. ISBN: 2325-0771. DOI: 10.1109/ICL-GNSS.2016.7533686.

[11] Mahsa Foruhandeh et al. "Spotr: GPS Spoofing Detection via Device Fingerprinting". In: *arXiv preprint arXiv:2005.08787* (2020).

[12] M. T. Gamba, M. Nicola, and B. Motella. "Galileo OSNMA: an implementation for ARM-based embedded platforms". In: *2020 International Conference on Localization and GNSS (ICL-GNSS)*. 2020, pp. 1–6. ISBN: 2325-0771. DOI: 10.1109/ICL-GNSS49876.2020.9115539.

[13] G. X. Gao et al. "Understanding the GIOVE-B broadcast codes of the Galileo system". In: *2008 42nd Asilomar Conference on Signals, Systems and Computers*, pp. 2086–2090. ISBN: 1058-6393. DOI: 10.1109/ACSSC.2008.5074800.

[14] Yaqi Hu. "GNSS SDR Signal Generator Implementation Based on USRP N210". In: *Journal of Physics: Conference Series* 1314 (2019), p. 012016. ISSN: 1742-6588 1742-6596. DOI: 10.1088/1742-6596/1314/1/012016. URL: http://dx.doi.org/10.1088/1742-6596/1314/1/012016.

[15] Todd Humphreys, Jahshan Bhatti, and Brent Ledvina. "The GPS Assimilator: a method for upgrading existing GPS user equipment to improve accuracy, robustness, and resistance to spoofing". In: (2010).

[16] Todd E Humphreys et al. "Assessing the spoofing threat: Development of a portable GPS civilian spoofer". In: *Radionavigation laboratory conference proceedings*. 2008.

[17] IAC. *Glonass History*. 2021. URL: https://www.glonass-iac.ru/en/guide/index.php.

[18] Ali Jafarnia-Jahromi et al. "GPS Vulnerability to Spoofing Threats and a Review of Antispoofing Techniques". In: *International Journal of Navigation and Observation* 2012 (2012), p. 127072. ISSN: 1687-5990. DOI: 10.1155/2012/127072. URL: https://doi.org/10.1155/2012/127072.

[19] K. Jansen et al. "Crowd-GPS-Sec: Leveraging Crowdsourcing to Detect and Localize GPS Spoofing Attacks". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 1018–1031. ISBN: 2375-1207. DOI: 10.1109/SP.2018.00012.

[20] Malek Karaim et al. "GNSS error sources". In: *Multifunctional Operation and Application of GPS* (2018), pp. 69–85.

[21] Andrew J. Kerns et al. "Unmanned Aircraft Capture and Control Via GPS Spoofing". In: *Journal of Field Robotics* 31.4 (2014), pp. 617–636. ISSN: 1556-4959. DOI: https://doi.org/10.1002/rob.21513. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21513.

[22] Sherman Lo et al. "Test of crowdsourced smartphones measurements to detect GNSS spoofing and other disruptions". In: *Proceedings of the 2019 International Technical Meeting of The Institute of Navigation*, pp. 373–388. ISBN: 2330-3646.

[23] Damian Miralles et al. "Android Raw GNSS Measurements as the New Anti-Spoofing and Anti-Jamming Solution". In: (2018), pp. 334–344. DOI: 10.33012/2018.15883.

[24] R. Morales-Ferre et al. "A Survey on Coping With Intentional Interference in Satellite Navigation for Manned and Unmanned Aircraft". In: *IEEE Communications Surveys & Tutorials* 22.1 (2020), pp. 249–291. ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2949178.

[25] Gabriele Oligeri et al. "Drive me not: GPS spoofing detection via cellular network: (architectures, models, and experiments)". In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*. 2019, pp. 12–22.

[26] JA Pierce. "An introduction to Loran". In: *Proceedings of the IRE* 34.5 (1946), pp. 216–234. ISSN: 0096-8390.

[27] M. L. Psiaki and T. E. Humphreys. "GNSS Spoofing and Detection". In: *Proceedings of the IEEE* 104.6 (2016), pp. 1258–1270. ISSN: 1558-2256. DOI: 10.1109/JPROC.2016.2526658.

[28] J.A Ávila Rodríguez. *Galileo Signal Plan*. 2011. URL: https://gssc.esa.int/navipedia/index.php/Galileo_Signal_Plan.

[29] J.A Ávila Rodríguez. *GLONASS Signal Plan*. 2011. URL: https://gssc.esa.int/navipedia/index.php/GLONASS_Signal_Plan.

[30] J.A Ávila Rodríguez. *GPS Signal Plan*. 2011. URL: https://gssc.esa.int/navipedia/index.php/GPS_Signal_Plan.

[31] J.A Ávila Rodríguez and University FAF Munich. *BeiDou Signal Plan*. 2011. URL: https://gssc.esa.int/navipedia/index.php/BeiDou_Signal_Plan.

[32] Christian Titus Sanders. "Localizing Spoofing Attacks on Vehicular GPS Using Vehicle-to-Vehicle Communications". Thesis. 2020.

[33] Desmond Schmidt et al. "A Survey and Analysis of the GNSS Spoofing Threat and Countermeasures". In: *ACM Comput. Surv.* 48.4 (2016), Article 64. ISSN: 0360-0300. DOI: 10.1145/2897166. URL: https://doi.org/10.1145/2897166.

[34] E. Schmidt et al. "Software-Defined Radio GNSS Instrumentation for Spoofing Mitigation: A Review and a Case Study". In: *IEEE Transactions on Instrumentation and Measurement* 68.8 (2019), pp. 2768–2784. ISSN: 1557-9662. DOI: 10.1109/TIM.2018.2869261.

[35] Navigation National Coordination Office for Space-Based Positioning and Timing. *Pseudorandom Noise Code Assignments*. 2021. URL: https://www.gps.gov/technical/prn-codes/.

[36] Navigation National Coordination Office for Space-Based Positioning and Timing. *Selective Availability*. 2018. URL: https://www.gps.gov/systems/gps/modernization/sa/.

[37] Peter F Swaszek et al. "Analysis of a simple, multi-receiver GPS spoof detector". In: (2013).

[38] Nils Ole Tippenhauer et al. "On the Requirements for Successful GPS Spoofing Attacks". In: *Ccs '11* (2011), 75–86. ISSN: 9781450309486. DOI: 10.1145/2046707.2046719. URL: https://doi.org/10.1145/2046707.2046719.

[39] P. Waller et al. "The In-Orbit performances of GIOVE clocks". In: *IEEE Trans Ultrason Ferroelectr Freq Control* 57.3 (2010), pp. 738–745. ISSN: 0885-3010. DOI: 10.1109/TUFFC.2010.1472.

[40] Kang Wang, Shuhua Chen, and Aimin Pan. "Time and position spoofing with open source projects". In: *black hat Europe* 148 (2015).

[41] Hengqing Wen et al. "Countermeasures for GPS signal spoofing". In: *ION GNSS*. Vol. 5. 2005, pp. 13–16.

[42] K. Wesson, M. Rothlisberger, and T. Humphreys. "Practical Cryptographic Civil GPS Signal Authentication". In: *NAVIGATION* 59.3 (2012), pp. 177–193. ISSN: 0028-1522. DOI: https://doi.org/10.1002/navi.14. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/navi.14.

[43] Z. Wu et al. "Spoofing and Anti-Spoofing Technologies of Global Navigation Satellite System: A Survey". In: *IEEE Access* 8 (2020), pp. 165444–165496. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3022294.

[44] Safoora Zaminpardaz, P. J. G. Teunissen, and Nandakumaran Nadarajah. "IRNSS/NavIC l5 attitude determination". In: *Sensors (Basel)* 17.2 (2017), p. 274. ISSN: 1424-8220. DOI: 10.3390/s17020274.

[45] Kexiong Curtis Zeng et al. "A practical GPS location spoofing attack in road navigation scenario". In: *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications*. 2017, pp. 85–90.

[46] Kexiong Curtis Zeng et al. "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems". In: *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1527–1544. ISBN: 1939133041.

[47] Xian-Chun Zheng and Hung-Min Sun. "Hijacking Unmanned Aerial Vehicle by Exploiting Civil GPS Vulnerabilities Using Software-defined Radio". In: *Sensors and Materials* 32.8 (2020), pp. 2729–2743.