University Institute of Lisbon

Department of Information Science and Technology

Intentional Control of Invasive Mobile Wireless Systems

João Pedro de Castro Ponte

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Telecommunications and Computer Engineering

Supervisor

Dr. Francisco António Bucho Cercas, Tenured Professor

ISCTE-IUL

Co-Supervisor

Dr. Pedro Joaquim Amaro Sebastião, Assistant Professor

ISCTE-IUL

December 2019

Resumo

Recentemente, *drones* operados remotamente ou de funcionamento autónomo têm surgido no domínio dos produtos eletrónicos para consumidores e começam a popular o espaço aéreo das áreas populacionais. Como tal, o número de incidentes envolvendo estes dispositivos tem sofrido um aumento acentuado.

Neste sentido, o presente trabalho visa explorar um método que permita a falsificação dos sinais Global Positioning System (GPS) utilizados por muitos destes dispositivos para navegar, com o intuito de desenvolver uma forma de alterar a sua rota para longe das áreas desejadas.

A hipótese em estudo é a de que, alterando os parâmetros usados pelos recetores GPS para corrigir erros de relógio nos sistemas de navegação, é possível alterar a posição calculada pelo dispositivo de uma forma mensurável e facilmente replicável.

Para testar esta hipótese, foi desenvolvido um simulador que permite testar diferentes desvios aplicados aos valores dos coeficientes de correção do relógio presentes nas mensagens de navegação GPS. As posições resultantes de cálculos dependentes destes parâmetros foram depois traçadas num mapa da área circundante e analisadas. Como esperado, as posições são eficaz e previsivelmente alteradas de acordo com os desvios aplicados.

Por forma a validar os resultados das simulações, foram realizados testes físicos usando uma plataforma de Software Defined Radio (SDR) e um gerador de sinais GPS *open source* que foi modificado para gerar sinais com base nos dados alterados das simulações. Estes testes sustentam a hipótese de que os sinais falsificados são capazes de provocar, consistentemente, a deteção errónea de posições por parte dos recetores de forma análoga à das simulações.

Palavras-chave: GPS (Global Positioning System), GNSS (Global Navigation Satellite Systems), SDR (Software Defined Radio), UAV (Unmanned Aerial Vehicle), Spoofing.

Abstract

Within recent years, remotely operated or autonomous drones have been encroaching on the realm of consumer electronics and are beginning to crowd the airspace in populated areas. As such, the number of incidents involving drones has seen a sharp increase and concerns are being raised.

In this sense, the current work aims to explore a method which enables spoofing of the Global Positioning System (GPS) many of these devices use to navigate, and thus provide a way to shift them off course and away from the intended areas.

The proposed hypothesis is that, by altering the parameters by which GPS receivers correct for clock errors in the navigation systems, it is possible to shift the device's perceived position in a measurable and easily replicable way.

To test this hypothesis, a simulator was developed to test different offsets applied to the clock correction coefficients of a GPS navigation message. The positions resulting from calculations using these altered parameters were then plotted on a map of the surrounding area and analysed. As expected, the positions are effectively and predictably altered according to the offsets applied.

In order to validate the results from the simulations, real world tests were conducted using a Software Defined Radio (SDR) platform and an open source GPS Signal Generator which was modified to generate a signal based on the altered data from the simulations. With these tests it was asserted that the spoofed signals were able to consistently cause receivers to miscalculate their positions analogously to the simulations.

Keywords: GPS (Global Positioning System), GNSS (Global Navigation Satellite Systems), SDR (Software Defined Radio), UAV (Unmanned Aerial Vehicle), Spoofing.

Acknowledgements

I would first like to thank my Supervisor, Prof. Francisco Cercas for helping define the subject of this dissertation, for all his continued guidance and support throughout the course of the project and for pushing me to explore further. I would also like to acknowledge my Co-Supervisor, Prof. Pedro Sebastião.

A special thank you to Prof. José Sanguino for lending his expertise on the subject and for advising me and clarifying my doubts.

I would also like to thank my colleague and friend Rui Dias, for all the work we've done together during our Master's Degrees and at the helm of the IEEE ISCTE-IUL Student Branch.

To all the friends I've made in these last few years, a big thank you for all the moments shared and for being part of this pivotal, formative stage in my life, with a special cheers to my closest friends from home who shared with me their own paths to becoming engineers.

To Rita, for standing by me at all times and making dull moments vanish, thank you, with all my love.

Last, but certainly not least, I have my family to thank, especially my parents, who always pushed me to strive for excellence, who taught me the value of integrity and perseverance, and for their unwavering support and confidence, my deepest, heartfelt thank you.

Contents

List of Figures

List of Tables

Abbreviations

ADS-B	**A**utomatic **D**ependent **S**urveilance-**B**roadcast
API	**A**pplication **P**rogramming **I**nterface
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
BPSK	**B**inary **P**hase **S**hift **K**eying
C/A	**C**oarse/**A**cquisition
DOP	**D**ilution **O**f **P**recision
DSP	**D**igital **S**ignal **P**rocessor
DSSS	**D**irect **S**equence **S**pread **S**pectrum
ECEF	**E**arth **C**entred **E**arth **F**ixed
ECI	**E**arth **C**entered **I**nertial
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
FTP	**F**ile **T**ransfer **P**rotocol
GDOP	**G**eometric **D**ilution **O**f **P**recision
GLONASS	**GLO**balnaya **NA**vigatsionnaya **S**putnikovaya **S**istema
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
IF	**I**ntermediate **F**requency
IMU	**I**nertial **M**easurement **U**nit
LLH	**L**atitude **L**ongitude **H**eight
LO	**L**ocal **O**scillator
P	**P**recision
PLL	**P**hase **L**ocked **L**oop
PRN	**P**seudo **R**andom **N**oise

PV	**P**osition **V**elocity
RF	**R**adio **F**requency
RHCP	**R**ight **H**and **C**ircular **P**olarized
RINEX	**R**eceiver **IN**ndependent **EX**change
SDK	**S**oftware **D**evelopment **K**it
SDR	**S**oftware **D**efined **R**adio
SV	**S**pace **V**ehicle
SVN	**S**pace **V**ehicle **N**umber
TEQC	**T**ranslation **E**diting **Q**uality **C**heck
TOC	**T**ime **O**f **C**lock
TOE	**T**ime **O**f **E**phemeris
USRP	**U**niversal **S**oftware **R**adio **P**eripheral

Chapter 1

Introduction

1.1 Motivation

With the advent of affordable consumer model drones there has been a dramatic increase in the number of autonomous mobile wireless systems roaming the airspace all around the world in the last couple of years [1].

This leads to the unfortunate side effect of drones wandering into areas they shouldn't, such as the no-fly zones in and around airports, power lines, highly populated areas, or simply our own backyards.

Whether the intrusion in these areas is intentional or accidental it will always have consequences. It can lead to material or personal damage, infringe on people's privacy and cause the person responsible for the drone to incur criminal charges.

Devising a system that tricks the drone into switching from its original navigation signal to one developed to make it steer away from the area or perform a landing provides a non-destructive, safe and effective way to capture it and ensure that none of the above situations occur.

1.2 Context

Many of the most commonly used drones currently rely on Global Navigation Satellite Systems (GNSS) such as GPS or Galileo to calculate their position and navigation paths [2], [3]. These systems are widely used and, in the case of GPS, are the standard in situations where geolocation of a device is required. According to [2], these systems are commonly used in conjunction with an onboard inertial measurement unit (IMU) to create a state estimator which enables reliable navigation. Since these GNSS are susceptible to external interference (intentional or otherwise), the present work will focus on exploiting this shortcoming. The goal is to attempt to make the drone follow a custom signal which spoofs its current location rather than the original signal it was using to navigate, allowing it to be captured.

To achieve this, a radio communication system is devised in order to first jam the original signal, so the drone locks on to the new one which will then spoof the location of the vehicle. As described in [2], this first step could be achieved by ensuring the power advantage of the spoofer ($\eta = P_s/P_a$, where P_s is the received spoofing signal power and P_a is the authentic signal power), is above a certain threshold. After successful lock on to the spoofing signal, the second stage is to attempt to send the spoofed location data to the system in order to alter the navigational estimator in such a way that the drone alters its navigational parameters without triggering any tampering detection systems.

The most versatile approach to building such a system is to use Software Defined Radio (SDR) to test and implement the components of a Radio Emitter & Receiver System [4], [5], [6], which will interface with a dedicated Universal Software Radio Peripheral (USRP) platform to send/receive the necessary signals. This system will serve as a proof of concept of the work pertaining to the dissertation and allow us to analyse the data both in a controlled environment and in real-world scenarios.

1.3 Research Questions

After conclusion of the research work, it is expected that the following basic research questions will be answered:

- Is the proposed system an effective tool to spoof the target drone's location?
- How does the implementation compare to other already available solutions?
- Can the system be designed to work more efficiently and, if so, in what way?
- Can the implementation be further improved upon by adapting it to enable compatibility with other GNSS besides GPS?

1.4 Goals

The focus of the project is to develop a practical implementation of the GNSS spoofing system described in the Context Section above, which should be robust enough to allow for reliable use as a mechanism to control invasive drones and, in doing so, possibly assert its viability for use as a plug and play, standalone solution.

To achieve the intended goals, a review of the literature must first be conducted in order to understand how GNSS systems are used by the drones and how the different signals can be replicated and subsequently altered to suit the purposes of our project.

Once a hypothesis on how this is to be achieved has been formulated, the next step is to design and implement a simulator, to emulate the operation of the proposed Radio Communication System. This system enables the collection of some initial data and provides a testbed for current and future research on GNSS signal modification.

In order to validate the results obtained from experimentation using the simulator developed, the final goal of the current work is to set up a physical implementation of the proposed system for spoofed signal transmission directly to the target receiver, using Universal Software Radio Peripheral (USRP) boards and Software defined Radio (SDR). This implementation will serve as a test platform to assess the proposed hypothesis' validity and observe the impact of the generated spoofed signal on GPS receivers.

1.5 Contributions

As a means to test the hypothesis proposed, a Matlab simulator was developed to calculate a GPS receiver position using data parsed from Receiver Independent Exchange (RINEX) Navigation and Observation files and test the effects on calculated positions of alterations to the navigation data. This simulator also allows plotting of the different positions on either satellite imagery or Google Maps renditions of the surrounding area.

Part of the algorithms developed for the simulator were also converted from Matlab code to C, in order to enable their use in an already existing open source GPS signal generator.

As part of the research for the present work, an article entitled: "GPS Data Alteration for Use in a Position Spoofing System" was submitted and presented at the 11th Conference on Telecommunications - ConfTele 2019:

- Ponte, J., Cercas, F., Sebastião, P., Sanguino, J., Dias, R, "GPS Data Alteration for Use in a Position Spoofing System", 11th Conference on Telecommunications - ConfTele 2019, Lisbon, June 28th, 2019.

Chapter 2

State of The Art

2.1 Global Navigation Satellite Systems

Global Navigation Satellite Systems are satellite navigation systems that enable geolocation of a device on a global scale by using constellations of satellites to transmit position and timing data to specific receivers. These systems include but are not limited to: GPS, Galileo and GLONASS [7].

On a base level, all GNSS use trilateration to calculate a device's position. The device receives its distance d relative to four or more satellites and by intersecting the four spheres of radius d centred on each satellite the resulting intersection point corresponds to where the device is located in relation to the satellites. Even though at least four satellites are needed, using only three satellites to calculate the receiver's position, the spheres would still intersect, regardless of how incorrect the measurements are. This is due to the fact that the receiver calculates the distances to each satellite using its internal clock to measure the time it takes to transmit the signal, so all of them will be proportionally incorrect. The fourth satellite is, thus, used to ensure that the measured distances are correct, since the intersection of four spheres is only possible if all the distances have been correctly measured [7], [8].

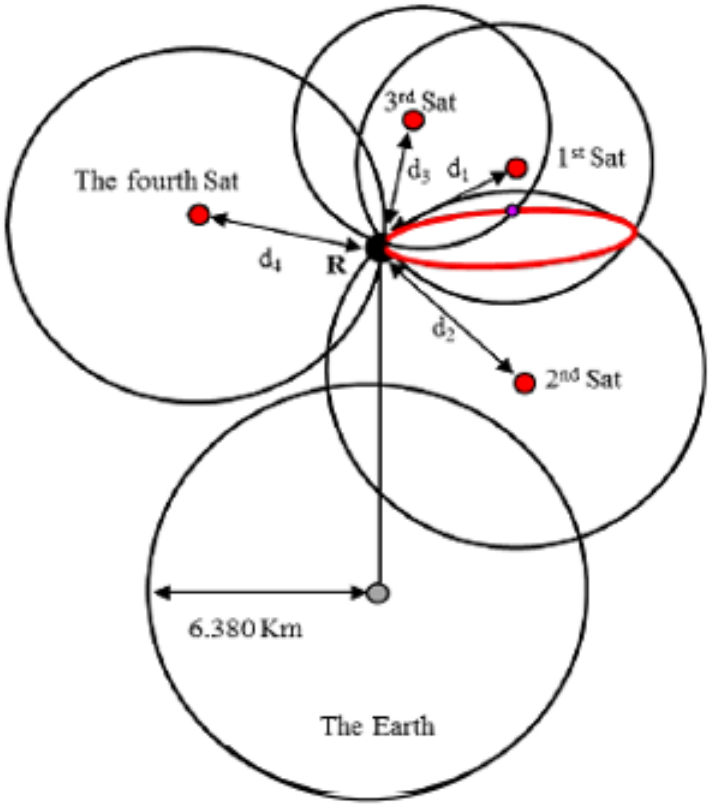

FIGURE 2.1: GNSS user position trilateration using four satellites [9]

Many of the most commonly used drones currently rely on GNSS to calculate their position and navigation paths, [2], [3]. These systems are widely used and, in the case of GPS, are the standard in situations where geolocation of a device is required. According to [2], these systems are generally used in conjunction with an onboard inertial measurement unit (IMU) to create a state estimator which enables reliable navigation. As stated previously these GNSS are susceptible to external interference (intentional or otherwise), as such, they present a weak link in the system which will be exploited in order to trick the drone into navigating using the spoofed navigation data rather than the genuine GPS signal.

2.2 Global Positioning System

The Global Positioning System (GPS) is the American GNSS which has seen widespread adoption worldwide. The system is maintained by the United States Government and is free to use with the caveat that the US Military can degrade or otherwise block the signal at any time if deemed necessary [7].

GPS signals use three types of Pseudorandom Noise (PRN) ranging codes: the publicly available Coarse/Acquisition or C/A code, the restricted Precision (P) and its Y-code variant for anti-spoofing mode. Since the latter are reserved for military applications and are therefore not widely adopted by the mobile devices in study, the present paper will focus exclusively on the C/A ranging codes [7], [8].

PRN codes present spectral properties analogous to random binary sequences and may look just like a random sequence of binary bits, but they are, in fact, known combinations of bits, of which there are 37 unique variants [8].

2.3 GPS Ranging Signals

GPS signal transmissions rely on direct sequence spread spectrum (DSSS) modulation of a carrier wave to enable the transmission of ranging signals as well as navigation data. The ranging signals are PRN codes that are multiplied by the binary phase shift keyed (BPSK) satellite carrier signal frequencies. The carrier wave frequencies used are 1575.42 MHz (L1) and 1227.60 MHz (L2). C/A code is transmitted on the L1 frequency while P(Y) code uses both L1 and L2. This use of two different frequencies to transmit the code is what provides P(Y) code with increased resistance to jamming as well as other benefits such as providing redundancy and allowing for the measurement and subsequent correction for ionospheric delay [7], [8].

PRN codes follow a predictable pattern and can thus be replicated. When determining the geometric range r from the satellite to the user, the receiver will generate a replica of the code and time shift it by Δt in order to achieve correlation with the code received from the satellite. Here Δt is the difference between the moment T_U when the code was received by the user and the moment T_S when the code was sent from the satellite. Multiplying Δt by the speed of light c, returns the geometric range r:

$$r = c\Delta t = c\,(T_U - T_S) \tag{2.1}$$

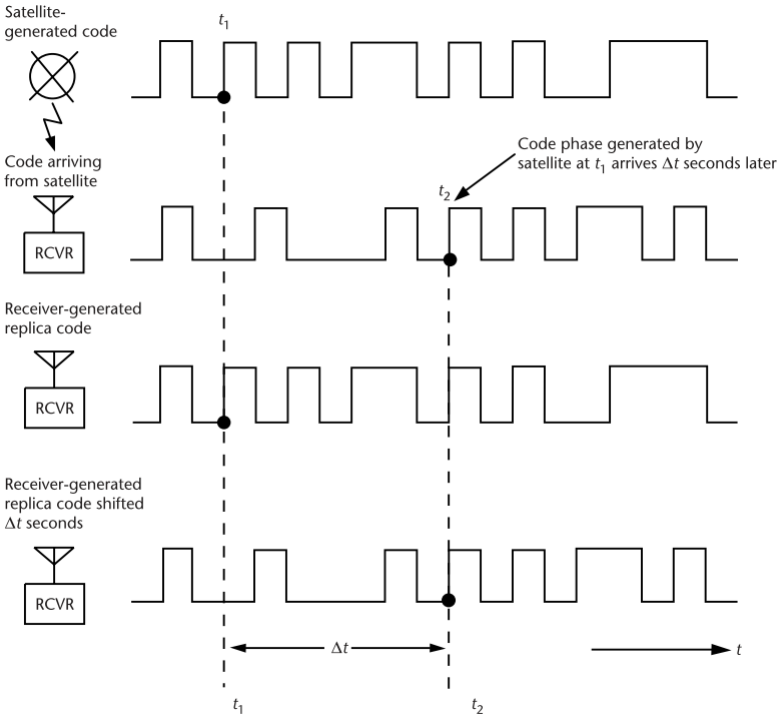

FIGURE 2.2: Satellite code reception delay representation [7]

Considering the Earth centred Earth fixed (ECEF) Cartesian coordinate system shown in Figure 2.3, the magnitude of the range vector r is given by:

$$\vec{\boldsymbol{r}} = ||\vec{\boldsymbol{s}} - \vec{\boldsymbol{u}}|| \tag{2.2}$$

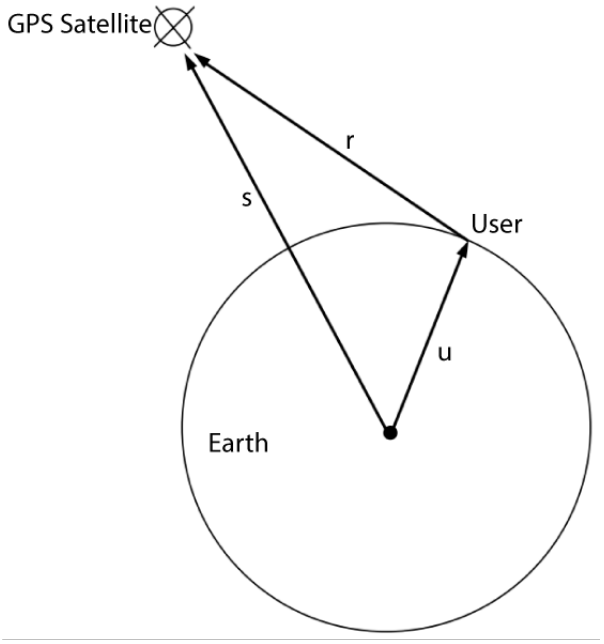

FIGURE 2.3: User position vector representation in ECEF Cartesian System [7]

This calculation assumes that both the satellite's and the receiver's internal clocks are perfectly synchronized, however, this is not the case in most real-world scenarios. As such, to get an accurate value of the distance from satellite to receiver the pseudorange ρ must be considered [7]. The pseudorange can be determined using the following equation, where t_u is the offset of the receiver clock from system time and δt is the offset of the satellite clock from system time [8]:

$$\rho = c[(T_U - t_u) - (T_S - \delta t)] \tag{2.3}$$

which can be simplified to:

$$\rho = c\,(T_U - T_S) + c(tu - \delta t) \tag{2.4}$$

$$= r + c(tu - \delta t)$$

Substituting r in equation 2.2:

$$\rho - c\,(t_u - \delta t) = ||s - u|| \tag{2.5}$$

The receiver's three-dimensional position (x,y,z) can now be calculated using the measurements of the pseudorange ρ and offset t_u from each of the four satellites connected to the receiver. Since the offset δt is compensated for on the user side it can effectively be removed from the equation and the system of equations that must be solved to get the user position becomes:

$$\rho_j = c\,(t_u) + ||s_j - u||, \quad j = 1, 2, 3, 4, (\dots) \tag{2.6}$$

where $\boldsymbol{j}$ denotes the $\boldsymbol{j}$th satellite (1 to **N**), $\boldsymbol{\rho_j}$ is the calculated pseudorange for the $\boldsymbol{j}$th satellite, $\boldsymbol{c}$ is the physical constant for the speed of light, $\boldsymbol{t_u}$ is the offset of the receiver clock from system time, $\boldsymbol{s_j}$ is the $\boldsymbol{j}$th satellite's position and $\boldsymbol{u}$ is the user (receiver) position. Both $\boldsymbol{s_j}$ and $\boldsymbol{u}$ are in Earth Centered Earth Fixed (ECEF) (x,y,z) coordinates [7].

The equation system in its expanded form can have up to **N** equations depending on the number of satellites present but, effectively, only four satellites are needed to have enough data to solve for the (x,y,z) position $\boldsymbol{u}$ in (2.6) so for most cases (2.6) can be expanded as:

$$\begin{cases} \rho_1 = \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} \;+\; c\mathrm{t_u} \\ \rho_2 = \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} \;+\; c\mathrm{t_u} \\ \rho_3 = \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} \;+\; c\mathrm{t_u} \\ \rho_4 = \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} \;+\; c\mathrm{t_u} \end{cases} \tag{2.7}$$

The solution to this equation can be obtained by closed-form solutions, Kalman Filtering or iterative techniques based on linearization, such as Least Squares iterative methods [7].

2.4 C/A Codes

GPS Coarse/Acquisition (C/A) ranging codes are Gold codes with a duration of 1 millisecond and 1023 kbps chipping rate. These codes are generated by a modulo-2 addition of two 1023 chip long linear pattern sub-sequences, G1 and G2, each generated by a 10-bit shift register [8].

The different C/A codes are obtained by delaying the G2 sub-sequence relative to G1, the delay is unique to each satellite so each PRN number will have a different associated delay which eliminates the need for a delay register. PRN codes 1 through 32 are reserved for the space segment and PRN codes 33 through 37 are meant for other uses such as ground transmitters [7], [8].

These civil ranging codes are easy to predict and their documentation is publicly available, which makes them exceptionally vulnerable to signal spoofing [2], [7] [10].

2.5 GPS Receiver Arquitecture

In this section, the basic components of an SDR GPS receiver will be presented. To aid in detailing each component's function a generic GPS receiver block diagram is shown in Figure 2.4. The GPS Radio Frequency (RF) signals from all the satellites in view of the receiver are received by a right hand circular polarized (RHCP) antenna with near hemispherical gain coverage. These signals are then amplified by a low noise preamplifier to set the base noise figure and down converted to intermediate frequency (IF) using mixing frequencies obtained from local oscilators (LO). Mixing with the LO signal produces upper and lower sidebands of the satellite signal, so in order to select the lower sideband and discard the rest of the signals a bandpass filter is used. Finally, the IF signal goes through analog to digital (A/D) conversion as well as an automatic gain control process.

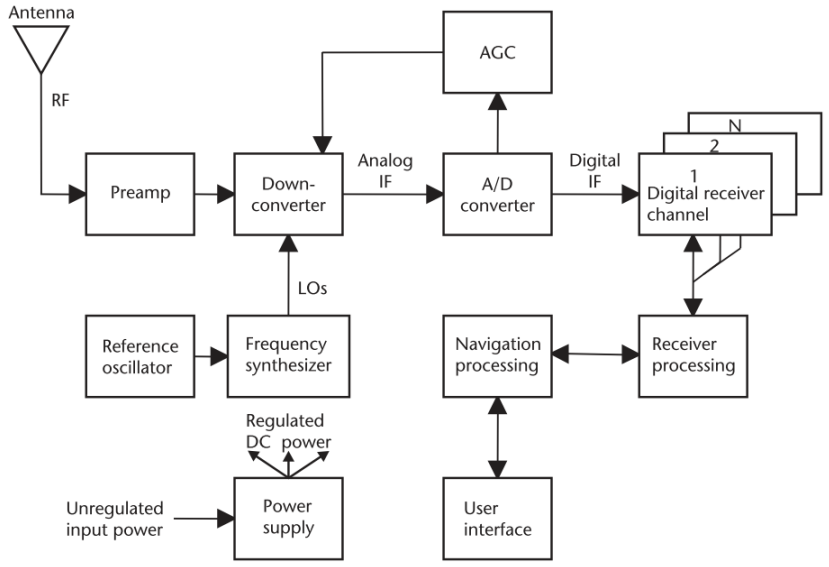

FIGURE 2.4: GPS Receiver Block Diagram[7]

The resulting IF signals from the process so far are now ready to be fed into the digital receiver channels where the actual demodulation will take place. These N channels are typically implemented using application specific integrated circuits (ASICs) or field programmable gate arrays (FPGAs) [7].

The signals are then demodulated and, following the conversion to baseband and code-stripping, they undergo an integration and dump process to allow for filtering and resampling and are then ready to be handled by the receiver processor [7], [8].

The receiver takes the processed signal and uses the data contained therein to calculate the necessary variables for position estimation in relation to the satellite which transmitted the signal. This process is done for each of the satellites within line of sight of the receiver.

2.6 Unmanned Aerial Vehicles

Unmanned Aerial Vehicles (UAV), more commonly known as drones, are aircraft which are controlled remotely and can therefore function without a human pilot

onboard. They are one of the components of Unmanned Aerial Systems (UAS), these systems are, in their most basic configuration, the combination of a UAV, a ground based controller and the communications system between the two [11]. UAS may also include GNSS receivers for autonomous UAV navigation and geolocation.

The most prevalent UAV navigation systems are based on using a state estimator which relies on an Inertial Measurement Unit (IMU) and a GPS Receiver to gather positional and geolocation data [2], [3].

2.7 Radio Frequency Interference

Since GNSS receivers rely on Radio Frequency (RF) signals, they are particularly vulnerable to RF Interference whether it is unintentional, like intersystem interference between GPS and Galileo, or intentional, as is the case with jamming.

While jamming aims to completely block or degrade the received signal in order to disable geolocation entirely, which is essentially a denial of service approach to the interference, spoofing consists in the creation of an interfering signal that mimics a genuine GNSS signal but contains information which generates a false position [2], [7], [10], [12].

2.8 GNSS Spoofing

The transmission of false GNSS signals in order to produce a false position in the receiver is known as spoofing [7].

GNSS signal spoofing can be accomplished through a variety of methods which have varying degrees of complexity, from simple meaconing attacks which replay captured GNSS signals, to more advanced solutions which generate an entirely new signal [13], [14], [15]. These methods usually follow one of two different strategies: overt spoofing [2], in which the spoofer does not take into consideration

any spoofing detection systems the target GNSS receiver may have; and covert spoofing [2], which considers the aforementioned detection systems and attempts to evade detection and, in doing so, circumvent any defensive measures which rely on spoofing attempt detection to activate [16].

An overt spoofing attempt will consist, in its most basic form, of increasing the spoofer's power advantage η above a certain threshold in order to force the receiver to lock on to the spoofed signal [2], [10], [16].

A successful covert spoofing attack, however, entails the capture of both the GPS receiver tracking loop and the navigational state estimator. Firstly, it must be ensured that no frequency unlock is detected in any of the Phase-Locked Loops (PLL), this is achieved by aligning the spoofed signals as closely as possible with the genuine signals in code phase τ(t) as well as Doppler frequency $f_D(t)$. If these conditions have been met, the spoofed signals' power can theoretically be raised above the genuine signals' power without raising alarm and thus capture the GPS tracking loop [2].

Secondly, to capture the navigation state estimator, the spoofer must evade innovations testing within the navigation state estimator. This is a complex and computationally costly procedure as it requires a degree of accuracy in calculating the Position Velocity (PV) estimate far greater than required for the previous step. The most effective way to obtain such accurate estimations is to intercept the device's Automatic Dependent Surveillance-Broadcast (ADS-B). These contain accurate position and velocity data which, when compared to the calculated PV estimate values of the navigation state estimator, will not trigger an alarm due to being very similar. This is particularly apparent when considering that innovations testing will have inflated thresholds to account for false positives due to multipath or poor satellite visibility [2], [17], [10], [15].

2.9 Spoofing Detection Strategies

A myriad of spoofing detection strategies has been proven to be effective in detecting spoofing attempts on UAVs and a successful covert spoofing attack must be designed to circumvent them. Some of the most common strategies are detailed below:

2.9.1 J/N Monitoring

J/N Monitoring is based on measuring the received power in the GPS receiver and if it exceeds that of the system in the same bandwidth under quiescent conditions (i.e., the voltages and current levels of the inputs and components of the circuit are fixed), then an alarm is triggered. This, by itself, can mean a spoofing attempt has been made, but to be thorough, it should be considered that if the alarm has been triggered due to the received power exceeding the threshold and the receiver continues to track GPS signals even though the signal appears to be jammed, then it is likely that a spoofing attempt is underway. This can effectively prevent an attack in which the GPS Signal is jammed to force the receiver to lock on to the spoofed signal. In order to bypass this strategy, [2] states that, for certain tested scenarios, maintaining the spoofer power advantage $\eta \leq 12dB$ is enough to avoid triggering the alarm [2].

2.9.2 Innovations Testing

Innovations Testing measures the PV output from the GPS receiver and compares it with the PV estimate given by the onboard navigation state estimator. If both don't match then, most likely, the device is being spoofed [2].

2.9.3 Drift Monitoring

Drift Monitoring consists of monitoring the receiver's position and clock fix for unusually quick changes in the reported values. For instance, if the clock error changes too quickly, the receiver might detect that such an alteration will produce an abnormally high rate of clock drift for the type of oscillator used [17].

2.9.4 Signal-Geometry-Based-Defences

Signal-Geometry-Based-Defences considers the direction of arrival of the GPS signals by monitoring the received beat carrier phase, modelled by:

$$\frac{\lambda\phi_i}{2\pi} = p_0^i + (\hat{p}^i)^T \Delta d + c\left(\delta_r - \delta^i\right) + \frac{\lambda\beta^i}{2\pi} \tag{2.8}$$

where $\hat{p}^i$ is the unit vector pointing from satellite to receiver and Δd is the displacement of the receiving antenna from the nominal receiver location [17].

Using interferometry (i.e., the superimposition of electromagnetic waves to intentionally cause interference in order to measure differences in said waves) the receiver can measure the direction-of-arrival vector $\hat{p}^i$ of three satellites with different Δd offsets. In a non-spoofed scenario, the $\hat{p}^i$ vectors will be spread out across the sky but in most spoofing scenarios, since a single emitter is used, the signals will be broadcast from the same spot and the $\hat{p}^i$ vectors will all have the same direction [17],[18].

The strategies described above are not, by any means, foolproof and, as such, are often used in parallel with other solutions so as to complement each other and provide a more robust spoofing detection system. They are also not the only available methods, in fact, [19] provides a different set of recommended strategies (with some in common with those on the list above) and [14] suggests usage of countermeasures based on message authentication.

2.10 Software Defined Radio

Software Defined Radio is an approach to building radio communication systems using software to emulate the functionality of traditional hardware components such as amplifiers, filters and modulators. It provides an easy, cost-effective and versatile way to design, build and test radio systems [4].

According to [7], "Most modern GPS receiver designs are digital receivers. These receiver designs have evolved rapidly towards higher and higher levels of digital component integration (...) Also, microprocessors and their specialized cousins DSPs, are becoming so powerful and cost effective that software defined radio receivers (SDRs) are being developed that use no custom digital components.".

One of the most popular SDR Software Development Toolkits (SDK) is GNU Radio [20], an open source toolkit that enables development of radio systems using a graphical interface based on signal processing blocks which can be linked to compatible external radio hardware [21]. The systems implemented using this SDK can effectively perform the functions of a GNSS receiver, as well as all the other required functionalities of the proposed GNSS signal generator.

Another commonly used SDK for SDR is Matlab's Simulink graphical block diagramming toolkit, which was proven by [4], [22] to be an effective tool for developing SDR based GNSS Systems.

2.11 Universal Software Radio Peripherals

Universal Software Radio Peripherals (USRP) are software defined radios which connect to a computer and provide the necessary hardware so that the emulated components can send and receive radio signals. They are a cost effective and versatile option when compared to traditional radio components.

For this project an Ettus Research USRP N210 USRP was used. These boards have a robust set of features ideally suited to this project, most notably a large FPGA, high bandwidth connectivity to the host pc and a selection of daughter-boards which can be installed for different applications [23].

FIGURE 2.5: Ettus Research N210 USRP [24]

2.12 Receiver Independent Exchange

The Receiver Independent Exchange format (RINEX) [25] is an ASCII text based format for GNSS data designed to be the standard for GNSS data storage and exchange, facilitating inter-operability of different systems, independent of receiver type or manufacturer. The format was developed hinging on the notion that most receivers use a well-defined set of observables, most notable of which are: carrier-phase measurement, pseudorange code measurement and the observation time [25], [26]. Consequently, most programs that use the receivers require only these observables along with some station-related information to perform their intended function such that the data stored can, in effect, be summarized to the essentials for each type required.

RINEX version 2.11 is the most widely used by the GNSS community at the time of writing, as such, this version will be the main one used for the purposes of this research throughout. This version of RINEX consists of seven different ASCII file types [25] pertaining to different sets of data gathered from the satellites, namely:

- 1. Observation Data File
- 2. Navigation Message File
- 3. Meteorological Data File
- 4. GLONASS Navigation Message File
- 5. GEO Navigation Message File
- 6. Satellite and Receiver Clock Date File
- 7. SBAS Broadcast Data File

Each of the files is comprised of a header section and a data section. The latter contains the data itself, while the former provides a description of the parameters contained within the data section and will thus be used to determine what data to look for when parsing the files.

The data required for the purposes of the present work is available, in its entirety, within the navigation message files and the observational data files, so the RINEX files employed either in simulations or experimental tests will be limited to these two types of file.

```
     2.11           N: GPS NAV. MESSAGE                     RINEX VERSION / TYPE
gLAB                gAGE / UPC          17-MAR-10 12:14     PGM / RUN BY / DATE
EXAMPLE OF A NAVIGATION RINEX FILE                          COMMENT
THIS FILE IS PART OF THE gLAB TOOL SUITE                    COMMENT
FILE PREPARED BY: ADRIA ROVIRA GARCIA                       COMMENT
PLEASE EMAIL ANY COMMENT OR REQUEST TO: glab.gage @ upc.edu COMMENT
     0.1676D-07     0.2235D-07    -0.1192D-07    -0.1192D-07 ION ALPHA
     0.1208D+06     0.1310D+06    -0.1310D+06    -0.1966D+06 ION BETA
   0.133179128170D-06   0.107469588780D-12   552960     1025 DELTA-UTC: A0,A1,T,W
    15                                                      LEAP SECONDS
                                                            END OF HEADER
 6 99  9  2 19  0  0.0 -.839701388031D-03 -.165982783074D-10  .000000000000D+00
    .910000000000D+02  .934062500000D+02  .116040547840D-08  .162092304801D+00
    .484101474285D-05  .626740418375D-02  .652112066746D-05  .515365489006D+04
    .409904000000D+06 -.242143869400D-07  .329237003460D+00 -.596046447754D-07
    .942817490922D+00  .326593750000D+03  .206958726335D+01 -.638312302555D-08
    .307155651409D-09  .000000000000D+00  .102500000000D+04  .000000000000D+00
    .000000000000D+00  .000000000000D+00  .000000000000D+00  .910000000000D+02
    .406800000000D+06  .000000000000D+00
13 99  9  2 19  0  0.0  .490025617182D-03  .204636307899D-11  .000000000000D+00
    .133000000000D+03 -.963125000000D+02  .146970407622D-08  .292961152146D+01
   -.498816370964D-05  .200239347760D-02  .928156077862D-05  .515328476143D+04
    .414000000000D+06 -.279396772385D-07  .243031939942D+01 -.558793544769D-07
    .986307770581D+00  .271187500000D+03 -.232757915425D+01 -.619632953057D-08
   -.785747015231D-11  .000000000000D+00  .102500000000D+04  .000000000000D+00
    .000000000000D+00  .000000000000D+00  .000000000000D+00  .389000000000D+03
    .410400000000D+06  .000000000000D+00
```

FIGURE 2.6: Example of the data in a RINEX 2.11 navigation message ASCII file [27]

2.12.1 RINEX Navigation Message Files

RINEX navigation data files contain the GPS satellite ephemerides data required to calculate each satellite's position at the time of data capture. This time might range from a single instance (though this is not a common occurrence when gathering data for generation of a navigation file) to several hours' worth of data capture in individual instances, usually limited to 24h cycles, at least as far as the files publicly available from NASA [28] are concerned. The data parsed by the simulator software developed for the present research is detailed in Table 2.1.

Line	Parameter		Units
1	-	SV PRN Code	Dimensionless
2	$af2$	SV Clock Drift Rate Correction Coefficient	sec/sec^2
3	M_0	Mean Anomaly at reference time	Semi-circles
4	$\sqrt{A}$	Square Root of the semi-major axis of the orbital ellipse	Meters
5	$\Delta\eta$	Correction term to the satellite mean angular velocity	Semi-circles/sec
6	e	Orbit Eccentricity	Dimensionless
7	ω	Argument of perigee	Semi-circles
8	C_{uc}	Amplitude of the cosine harmonic correction terms to the argument of latitude	Radians
9	C_{us}	Amplitude of the sine harmonic correction terms to the argument of latitude	Radians
10	C_{rc}	Amplitude of the cosine harmonic correction terms to the orbit radius	Meters
11	C_{rs}	Amplitude of the sine harmonic correction terms to the orbit radius	Meters
12	i_0	Orbit inclination angle (relatively to the equator) at the ephemeris reference time	Semi-circles
13	$IDOT$	Rate of the orbit inclination angle	semi-circles/s
14	C_{ic}	Amplitude of the cosine harmonic correction terms to the orbit inclination angle	Radians
15	C_{is}	Amplitude of the sine harmonic correction terms to the orbit inclination angle	Radians
16	Ω_0	Longitude (measured East) of ascending node at weekly epoch	Semi-Circles
17	$\dot{\Omega}$	Correction term to the right ascension of the ascending node	Semi-circles/s
18	toe	Ephemeris data reference time of week	Seconds
19	af_0	SV Clock Bias Correction Coefficient	Seconds
20	af_1	SV Clock Drift Correction Coefficient	sec/sec
21	toe	Ephemeris data reference time of week	Seconds

Table 2.1: GPS Navigation data parameters parsed from RINEX navigation data files

2.12.2 RINEX Observation Data files

RINEX observation files contain, among other assorted data relative to measurements taken by the receiver, the aforementioned observables: time, pseudo-range and phase. According to the standard format in [25], these are "(...) three fundamental quantities that need to be defined".

Time is considered the time measured by the receiver at the moment of each signal reception expressed in GPS time. It is identical for the phase and range measurements, as well as for all satellites at the same epoch [25].

Pseudo-range is defined as the distance in meters from the receiver antenna to the satellite antenna including offsets for both receiver and satellite clocks as well as any biases introduced in the signal transmission process [25].

Phase is the carrier wave phase measured in whole cycles [25].

```
     2.11           OBSERVATION DATA    M (MIXED)           RINEX VERSION / TYPE
BLANK OR G = GPS,  R = GLONASS,  E = GALILEO,  M = MIXED    COMMENT
gLAB                gAGE                17-MAR-10 12:14    PGM / RUN BY / DATE
EXAMPLE OF A MIXED RINEX FILE                               COMMENT
MRKR                                                        MARKER NAME
9080.1.34                                                   MARKER NUMBER
gAGE                UPC: Technical University of Catalonia  OBSERVER / AGENCY
THIS FILE IS PART OF THE gLAB TOOL SUITE                    COMMENT
FILE PREPARED BY: ADRIA ROVIRA GARCIA                       COMMENT
PLEASE EMAIL ANY COMMENT OR REQUEST TO: glab.gage @ upc.edu COMMENT
IR2200716006        ASHTECH UZ-12       CQ00                REC # / TYPE / VERS
482                 AOAD/M_T            NONE                ANT # / TYPE
   4789028.4701    176610.5133   4195017.0310               APPROX POSITION XYZ
      0.9030         0.0000        0.0000                   ANTENNA: DELTA H/E/N
     1     1                                                WAVELENGTH FACT L1/2
     1     2     3   G14   G18   G19                        WAVELENGTH FACT L1/2
     7    L1    L2    P1    P2    C1    S1    S2           # / TYPES OF OBSERV
    30.000                                                  INTERVAL
  2010     3     5     0     0     0.0000000     GPS         TIME OF FIRST OBS
  2010     3     5    23    59    30.0000000     GPS         TIME OF LAST OBS
     1                                                      RCV CLOCK OFFS APPL
    15                                                      LEAP SECONDS
    14                                                      # OF SATELLITES
   G07   815   815   815   815   815   815   815            PRN / # OF OBS
   G09   246   246   246   246   246   246   246            PRN / # OF OBS
   G12   687   687   687   687   687   687   687            PRN / # OF OBS
   G13   762   762   762   762   762   762   762            PRN / # OF OBS
   G15   454   454   454   454   454   454   454            PRN / # OF OBS
   G20   599   599   599   599   599   599   599            PRN / # OF OBS
   G21   636   636   636   636   636   636   636            PRN / # OF OBS
   G26   210   210   210   210   210   210   210            PRN / # OF OBS
   G31   874   874   874   874   874   874   874            PRN / # OF OBS
   G32   457   457   457   457   457   457   457            PRN / # OF OBS
   R11   907                     907   907                  PRN / # OF OBS
   R19   348                     348   348                  PRN / # OF OBS
   R23   936                     936   936                  PRN / # OF OBS
   S24   198                     198   198                  PRN / # OF OBS
                                                            END OF HEADER
10  3  5  0  0  0.0000000  0 14G18R19G32G TR23G31G20R11G12G26G 9G21    -0.12945
                                 G15824
121367582.20508  94572194.49208  23095489.677 9  23095481.949 9  23095483.463 7
        42.000          40.000
134357446.85408                                              23095483.463 7
        51.000
 94357446.85408 104694102.10708  25567381.585 9  25567371.841 9  25567379.659 7
        76.000          54.000
118767195.32608  91018570.22508  22600658.277 9  22600648.232 9  22227666.760 7
        57.000          52.000
132798857.20508                                              22600648.288 7
        22.000
130586822.29708 101785719.95608  24849779.954 9  24849799.921 9  24849797.341 7
        56.000          24.000
135891004.29908 105889081.83208  25859215.981 9  25859205.736 9  25859205.875 7
        44.000          46.000
```

FIGURE 2.7: Example of the data in a RINEX 2.11 observation data ASCII file (cropped) [29]

These observables are not corrected for external effects and any adjustment made by software on either the receiver or the transmitter end must be applied in such a way as to maintain the consistency of the 3 quantities described above [25].

Chapter 3

Simulator Implementation

3.1 Polynomial Coefficients for Satellite Clock Correction

According to [7], [8], in order to accurately calculate user position, the receiver must determine the effective Space Vehicle (SV) PRN code phase time offset referenced to the phase centre of the antennas ($\boldsymbol{\Delta t_{sv}}$) with respect to GPS system time ($\boldsymbol{t}$) at the time of data transmission. So, the receiver must correct the effective SV PRN code phase time at message transmission time received from the SV ($\boldsymbol{t_{sv}}$) using the following equation:

$$t = t_{sv} - \Delta t_{sv} \tag{3.1}$$

with $\boldsymbol{\Delta t_{sv}}$ given by:

$$\Delta t_{sv} \ = af_0 + af_1\,(t - toc) + af_2(t - toc)^2 + \Delta tr \tag{3.2}$$

Where $\boldsymbol{af}_0$, $\boldsymbol{af}_1$ and $\boldsymbol{af}_2$ are the polynomial correction coefficients for the satellite clock bias, clock drift and clock drift rate, respectively, $\boldsymbol{toc}$ is the clock data reference time in seconds and $\boldsymbol{\Delta tr}$ is the relativistic correction term given by:

$$\Delta tr = Fe^{\sqrt{A}}\sin(E_k) \tag{3.3}$$

$\boldsymbol{\Delta tr}$ can be calculated entirely from data contained in the GPS navigation message [7], [8]. The satellite's orbital parameters of eccentricity ($\boldsymbol{e}$), square root of the semi-major axis ($\boldsymbol{\sqrt{A}}$) are contained in subframes 2 and 3 of the GPS navigation message, while the eccentric anomaly ($\boldsymbol{E_k}$) can be computed by solving Kepler's equation for eccentric anomaly using iteration. $\boldsymbol{F}$ is a constant, equal to $-\ 4.442807633\ (10)^{-10} \frac{sec}{\sqrt{meter}}$ [7], [8], [30].

The main polynomials chosen to be altered in this case are the clock bias ($\boldsymbol{af_0}$) and clock drift ($\boldsymbol{af_1}$) which are used to correct the satellite clock values when calculating pseudorange. By altering these values for each of the satellites being used to calculate receiver position, the perceived position can be shifted according to the bias now introduced in calculated pseudorange values for each of the satellites. Thus, it becomes possible to, effectively, control the direction in which the perceived position drifts relative to the actual position of the receiver.

Clock Drift Rate ($\boldsymbol{af_2}$) was often 0 in tested instances. A choice was therefore made not to rely too much on altering this parameter for future experiments since it would be very easy for a receiver that is closely monitoring navigation message values to detect the new position as a spoofing attempt. Furthermore, of the three coefficients ($\boldsymbol{af_0}$, $\boldsymbol{af_1}$ and $\boldsymbol{af_2}$) in study, altering this last one seemed to produce the smallest variation in the calculated position.

For each of the experiments conducted, the polynomial coefficients were altered by adding either a positive or negative offset between 1x10^{-7} and 1x10^{-5} to the parameter in study. Applying this offset equally to all the satellites within line of sight of the receiver will result in a different position being calculated for each of the SV data chosen to solve the user position equation.

To assess the impact in calculated position for each satellite, all the possible shifted positions consequence of altering the polynomials by a certain offset for each satellite are plotted on an accurate satellite image of the area surrounding the receiver as shown in Figure 3.1. This allows for visual assessment of where exactly on the map the position can be shifted to, by applying the offset to a given SV or even a combination of multiple SVs. By doing this, it becomes apparent

that choosing a combination of any 2 SVs to alter effectively allows placement of the spoofed position anywhere in the surface defined by the 2 vectors created by plotting a straight line from the original receiver position to the position calculated using an offset applied to the satellite in question.

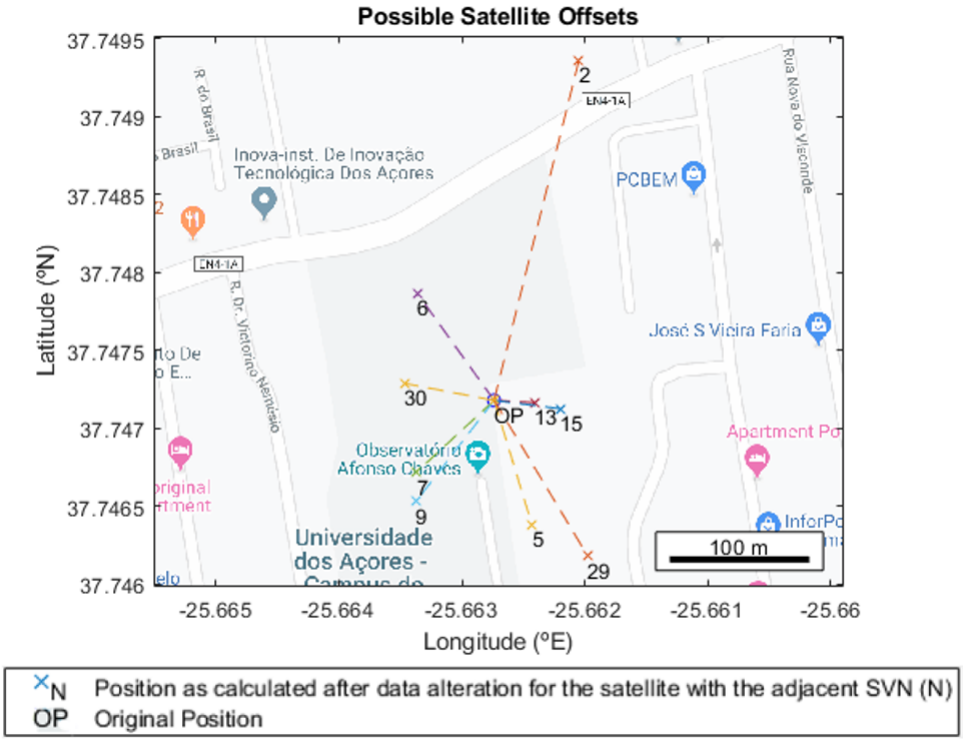

FIGURE 3.1: Example of possible spoofed receiver positions when positive offsets of $1x10^{-6}$ seconds and $1x10^{-6}$ sec/sec are applied to both af0 and af1 parameters respectively for each satellite within line of sight of the receiver

To allow for easier visualisation of the area where the spoofed position can be placed, the satellites chosen for the experiments are, for the most part, those whose vectors are as close to orthogonal between them as possible.

The orthogonality between the vectors is evaluated visually from preliminary renderings of possible offsets such as Figure 3.1, meaning that there is an associated imprecision to determining how orthogonal the vectors actually are, but for the purposes of the experiments pertaining to the present work, the vectors don't have to be precisely orthogonal, just close enough that they form as close to a 90 degree angle between them as possible. Using this approach allows for two dimensional control over the spoofed position, since each of the offsets applied

shifts the position in a different direction and it is essentially possible to treat the two vectors as axes in a Cartesian reference system for increased ease of use and better perception of possible position shifts.

Once this has been established, defining the spoofed position becomes as straightforward as inputting the right offset value combination for the polynomial correction coefficients into their corresponding values for each of the chosen satellites. This is done on a trial and error basis, manually altering the offset values and then recalculating the receiver position to assert its placement until the intended position has been successfully set.

As an example, by altering the polynomial correction coefficient for the satellites with numbers 6 and 7 from the example in Figure 3.1, it is apparent that the lines plotted between the receiver's original position and the positions affected by the introduction of positive offset values to the clock correction coefficients form a near 90 degree angle between them (Figure 3.2).

Furthermore, additionally considering the positions affected by negative offsets of equal magnitude and plotting the lines between them and the original receiver position, the aforementioned Cartesian-like chart begins to form (Figure 3.3).

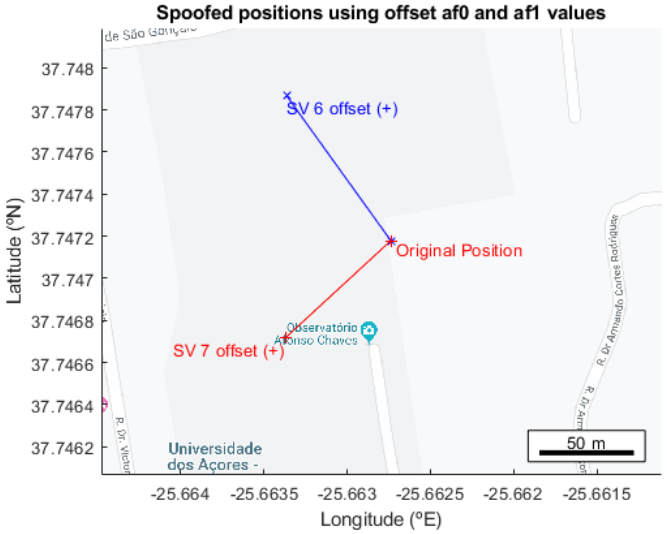

FIGURE 3.2: Example of possible spoofed receiver positions when positive offsets of $1x10^{-6}$ seconds and $1x10^{-6}$ sec/sec are applied to both af0 and af1 parameters for satellites numbers 6 and 7

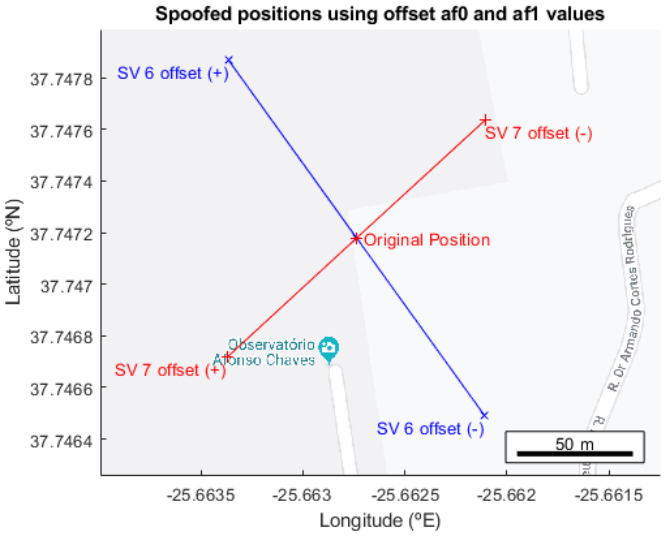

FIGURE 3.3: Example of possible spoofed receiver positions if either positive or negative offsets of $1x10^{-6}$ seconds and $1x10^{-6}$ sec/sec are applied to both af0 and af1 parameters respectively for satellites numbers 6 and 7

With a selection of satellites such as the one shown in Figure 3.3, it is essentially possible to place the location of the spoofed position anywhere within the surface defined by the plotted lines by applying negative or positive offsets to create a bias in a certain direction. For instance, if the intended spoofed position in this particular instance were to be east of the receiver, the offsets applied to each of the satellites would both have to be negative. The magnitude of the applied offsets will, in turn, define the spoofed position's distance from the actual receiver position, with greater magnitudes corresponding to greater distances as will be discussed in the following sections 3.3.1 through 3.3.4.

The approach detailed above should, in theory, allow for flexible experimentation possibilities with different polynomial coefficient offsets as a means to alter the GPS receiver's perceived position in a predictable and replicable manner.

For the experimental examples detailed in the following sections, the RINEX navigation and observation files used were downloaded from NASA's FTP server [28] and were generated from GPS Signals captured in the campus of Universidade dos Açores in São Miguel, Azores Islands on the 20th of June 2019.

Prior to starting the simulation, the RINEX observation files are trimmed using the WinTEQC editor, a graphical interface developed for use with Unavco's Translation Editing Quality Check (TEQC) GNSS data toolkit. This is done to remove GLONASS and Galileo observational data, as well as any GPS observation types not required for the experimental procedures. By doing so, the simulator's scope can be simplified as it is no longer necessary to redesign the simulator's file parsing algorithm to account for data irrelevant to the present work.

The editor parses the RINEX observation file's header to determine which satellite types are present and what observational data was captured to the file. By comparing this information to the input provided of what observation data is to be kept, the editor can parse the remainder of the file to eliminate all the unnecessary data from each epoch. Since most RINEX observation files used for this work are several thousand lines long, employing this editor significantly reduces the time required to trim the file and ensures that no information is wrongly eliminated.

In the case of future research aiming to apply the present approach to GNSS other than GPS, the files may be used unaltered, providing that the simulator's parser is adapted to recognize the remaining observational data required.

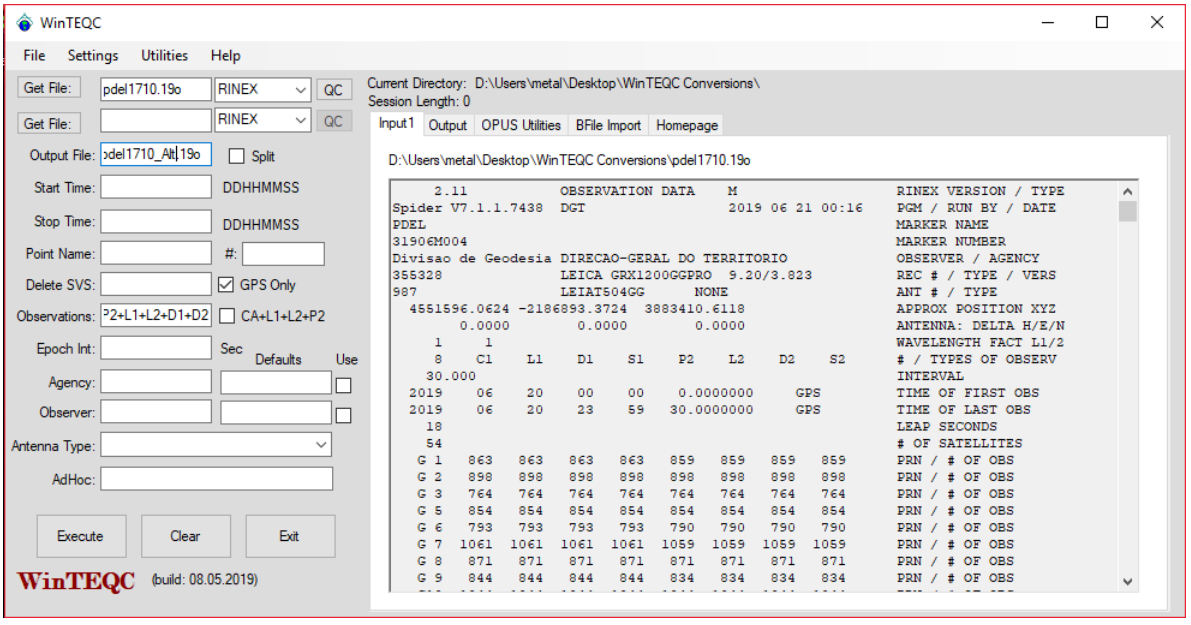

FIGURE 3.4: WinTEQC interface showing a RINEX observation file and the data to be trimmed

3.2 Basic Algorithm Functionality

For the purposes of testing the hypothesis of altering GPS navigation data parameters in order to produce an altered perceived position differing from the receiver's actual position, a Matlab simulator was developed. The main components of the code discussed in the present section are available in appendices A and C.

The program reads and parses RINEX navigation message and observation data files (Appendix A.2) in order to obtain the necessary satellite ephemerides and pseudorange data required to calculate a GPS receiver's position by using a least-squares iteration method (Appendix A.1). Sections of the code in [31], [32] were used to develop the simulator components which deal with parsing the RINEX files and calculating the actual receiver position from the data contained within. The required files are sourced from a publicly accessible FTP server hosted by NASA [28] and providing RINEX files of all different types introduced in section 2.12, which are compiled daily from receivers in locations across the globe. This variety of files allows selection from a variety of different locations and times to simulate a receiver position.

The simulator starts by parsing the RINEX navigation file line by line and storing the ephemerides data for each of the N satellites present in the file in a 21xN matrix where each line corresponds to a different parameter of the satellite ephemerides. The navigation data stored in this matrix is detailed in Table 2.1.

Next, the RINEX observation file for the corresponding location and time range is parsed to obtain the data observed by the receiver, namely, observation types, antenna delta, pseudorange and GPS time.

Using the data from these two files, there is now enough information to estimate the receiver position using an iterative least squares method. With the pseudorange and ephemerides data from four or more of the satellites available it becomes possible to solve the system of equations described in Section 2.3, equations 2.6 and 2.7.

To solve for the correct estimated receiver position, the aforementioned least squares iteration method is employed, starting by assuming the receiver's approximate position is (0,0,0) which corresponds to the Earth's centre in an ECEF system, then iterating through possible positions to calculate the offset of the true position relative to each of these approximated positions by a displacement, coming closer to the actual receiver position with each iteration [33], [34].

The basics steps for each cycle of the iterative least squares algorithm used to calculate the receiver are detailed in the following paragraphs.

Before starting the actual calculation of the position, some variables must be calculated. First, each satellite's time of clock (TOC) is checked for over or underflow according to half the duration of a week in seconds. This time is then corrected for clock errors and relativistic effects, as described in section 3.1. With this corrected time, each satellite's position is then calculated, using the ephemerides data obtained from the RINEX navigation files.

Having all the necessary variables, the algorithm can now begin the iterative process to determine an approximation of the receiver position as close as possible to the actual position.

First, the latitude and longitude coordinates are converted from degrees to radian. In the first iteration these coordinates will each be, as previously mentioned, zero and for each subsequent iteration run they will be updated to the most recent output coordinates obtained from the previous iteration.

The next step is creating a grid of possible positions around the globe for each latitude and longitude, which is achieved by calculating the azimuth between the current position and a series of regularly spaced latitudes around the Earth every 20 degrees until a full rotation has been achieved and repeating this process for each of the 90 degrees of possible longitude (for both North and South) also divided in 20 degree intervals. Each of the azimuths in this grid will then be used to compute a selection of coordinates which span the surface of the Earth. A visual representation of the created grid is shown in Figure 3.5, where each point

of intersection between latitude and longitude lines represents one of the positions mentioned above.

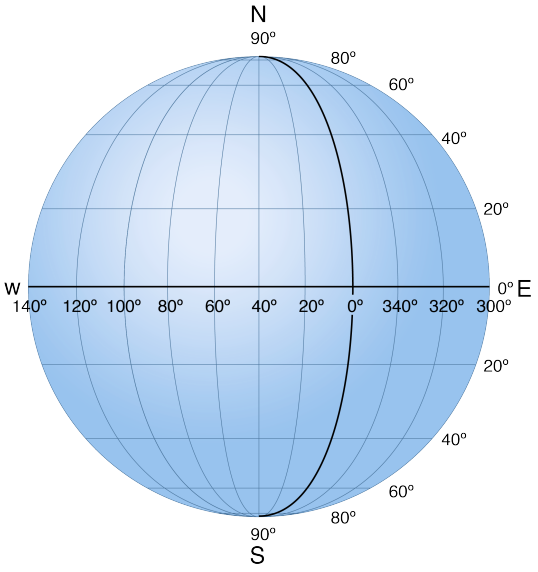

FIGURE 3.5: Representation of the grid created by the position calculation algorithm

Each of these coordinates will serve as an approximated starting point to check which of them is closest to the actual receiver position. The algorithm will then compute the distance from each of these positions to every one of the satellites within line of sight, which have been previously identified from the RINEX Observation file and whose positions relative to the Earth have been calculated before the beginning of the cycle:

$$distance = ||X_s - X_r|| \tag{3.4}$$

For this equation, X_s represents each satellite's cartesian coordinates and X_r represents the cartesian coordinates of the receiver's approximated position.

Due to rotation of the Earth during the signal transmission time, a relativistic error known as the Sagnac effect would be introduced to any positions calculated considering an ECEF coordinate system [7], [30]. Since this effect is not present

when computing in an ECI (Earth Centered Inertial) coordinate systems and the receiver position is the same in both systems at the signal reception time, the solution is to convert satellite coordinates to the ECI system and use them to compute the receiver position [7]. This relativistic effect is illustrated in Figure 3.6.

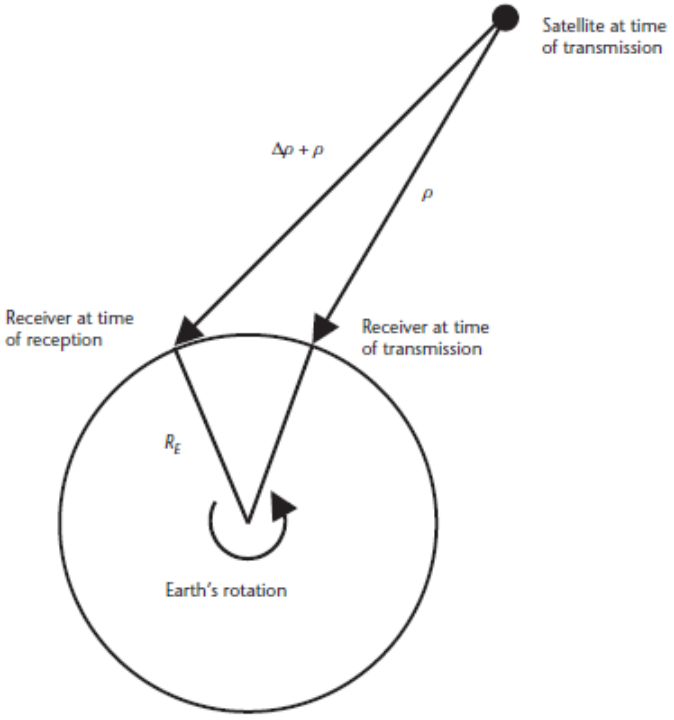

FIGURE 3.6: The Sagnac effect [7]

The longitude of the ascending node is subsequently determined to convert the calculated distances and positions from the ECEF coordinates to ECI coordinates. To do so, the right ascension of the reference meridian (represented by $\Omega(t)$ in equation 3.5) is required to build the rotational matrix $Rot_z(-\Omega(t))$ used for the conversion. This $\Omega(t)$ is obtained thusly:

$$\Omega(t) = \Omega(t_0) - \dot{\Omega} \cdot (t - t_0) \tag{3.5}$$

where $\boldsymbol{t_0}$ is the time at the start of the signal transmission, $\boldsymbol{t}$ is the time of signal reception by the receiver, $\boldsymbol{\Omega(t_0)}$ is the right ascension of the reference meridian at

the moment of signal transmission and $\dot{\boldsymbol{\Omega}}$ is the rotation rate of the Earth, which is a constant of value 7.292115147x10^{-5} rad/s [7], [8].

This equation can be simplified by substituting $(\boldsymbol{t} - \boldsymbol{t_0})$ for the previously calculated Euclidean distance between satellite and approximated receiver position from Equation 3.4, divided by the speed of light $\boldsymbol{c}$, to become the span of time between the moment of signal reception and the moment it was sent. When also considering $\boldsymbol{\Omega}(\boldsymbol{t_0})$ to be null, the simplified equation becomes:

$$\Omega(t) = -\dot{\Omega} \cdot \frac{distance}{c} \tag{3.6}$$

As mentioned previously, this will be used to build the rotational matrix:

$$Rot_z(-\Omega(t)) = \begin{bmatrix} \cos(\Omega(t)) & \sin(\Omega(t)) & 0 \\ -\sin(\Omega(t)) & \cos(\Omega(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Which, in turn, will allow us to compute the coordinates on the ECI frame (x_{ECI}, y_{ECI}, z_{ECI}) from the coordinates on the ECEF frame (x_{ECEF}, y_{ECEF}, z_{ECEF}):

$$\begin{bmatrix} x_{ECI} \\ y_{ECI} \\ z_{ECI} \end{bmatrix} = \begin{bmatrix} x_{ECEF} \\ y_{ECEF} \\ z_{ECEF} \end{bmatrix} \cdot Rot_z \tag{3.8}$$

From the converted set of coordinates (X'_s), the distance from satellite to receiver is recalculated for each satellite, just as before

$$distance' = ||X'_s - X_r|| \tag{3.9}$$

And, taking this new distance, along with the signal travel time corrected for clock errors multiplied by the speed of light $\boldsymbol{c}$, as well as the pseudorange for each of

the SVs in use, the corrected distance to the satellites ($distance_{res}$) to be used in calculating the residuals is given by the equation:

$$distance_{res} = pseudorange - distance' + corrected\ travel\ time \cdot c \tag{3.10}$$

Finally, the residual distances are calculated by subtracting this corrected distance from the distance calculated from the rotational matrix conversion and multiplying the matrix of said residuals by its own transposed matrix.

$$residuals = distance' - \ distanceRes$$

Since this calculation is performed for each satellite, the residuals are stored in a **1xN** array, where **N** is the number of satellites within line of sight. So the final calculation is between a line matrix and a column matrix, resulting in the sum of all the residuals for each satellite squared:

$$Sum\ of\ residuals\ squared = residuals \cdot residuals' =$$

$$= [residual(SV_1) \quad residual\,(SV_2) \quad \cdots \quad residual(SV_N)] \cdot \begin{bmatrix} residual\,(SV_1) \\ residual\,(SV_2) \\ \vdots \\ residual\,(SV_N) \end{bmatrix} \tag{3.11}$$

$$Sum\ of\ residuals\ squared = \sum_{n=1}^{N} residual(SV_n)^2 \tag{3.12}$$

Having this value, the algorithm simply checks to see if the new sum of the squared residuals is lower than the previous one (which starts with an initial value of $1x10^{20}$ in the first iteration). If such is the case, then the position computed for the set of coordinates X_r determined in this iteration is closer to the actual

receiver position than before and the coordinates for the next iteration become the current set; otherwise, the position is farther from, or at equal distance to the actual position, so the coordinates for the next iteration remain unchanged.

The cycle is repeated for a set number of iterations, once all have been completed, the coordinates stored at the end of the cycle are considered the closest approximation to the actual receiver position as computed by the least squares algorithm.

The final calculated position is returned in ECEF cartesian (x,y,z) as well as geographical (latitude, longitude, height) forms. Validation of the accuracy of the position calculated is achieved by comparing it to the approximate (x, y, z) coordinates stated in the comments of the RINEX observation file, in the line which ends with the tag: "APPROX POSITION XYZ". These coordinates correspond to the actual position of the GPS receiver used to generate the RINEX file, so, by comparing this given position with the one calculated using the algorithm it is possible to assert how accurate the calculation is relative to the known position.

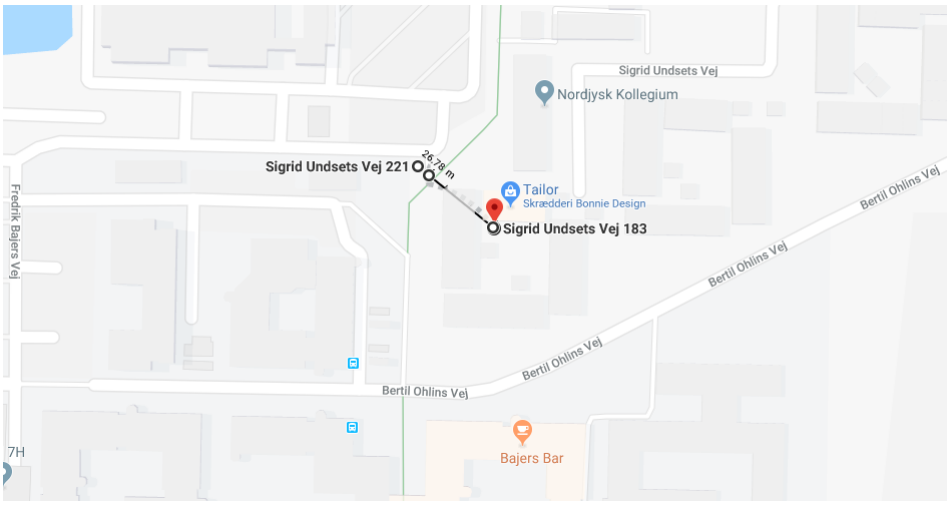

FIGURE 3.7: Example of the difference between calculated position (unmarked) and actual receiver position (marked with red pin) measured and plotted on Google maps image

Throughout the experimental procedures it was determined that the calculated position was usually within a distance of under 30 m from the approximated position stated in the observation file, with a few exceptions going as far as 66

m in cases where global, non-location specific, RINEX navigation files were used to obtain satellite ephemerides data, but never above this value. Many of the calculated positions, when overlaid on the map, were close enough to be separated by less than a meter, which made them essentially overlap each other for virtually all map scales used when plotting the positions.

An example of common grouping of calculated receiver positions for multiple RINEX observation files as seen in tested scenarios is shown in Figure 3.8, for this particular case, the observation files used to calculate receiver position were all taken from the same physical location (at the University of the Azores' campus) and the navigation files where an even split of 50% location specific files with respect to the indicated position and 50% global, non-location specific, navigation data files. All files were created from data taken during the current year of 2019, ranging from the months of February to July, and spanning periods of exactly 24 hours each.

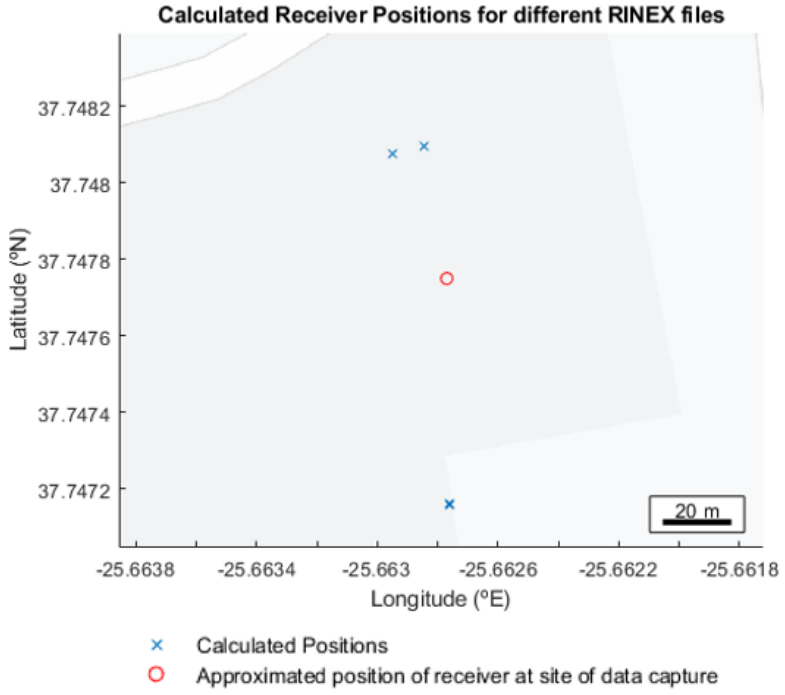

FIGURE 3.8: Calculated receiver position grouping plotted on map around approximated data capture position of physical GPS Signal receiver

These observations suggest that the system is capable of estimated receiver position calculation with acceptable accuracy for the intended purposes of the research and above average precision between runs with different observation files.

Given that the navigation and observation files are always paired by date for every experiment run, the difference in calculated position for each of these pairs does not affect simulations and only constitutes a factor to be accounted for during the procedure should the date vary greatly from the actual date/time of day at which the physical part of the experiment is being conducted. Throughout the experimental stages of the research, it was found that using old files whose date and time don't not match the actual time at which the experiments are being run for real world testing seemingly decreased attack success probabilities since it hinders the chance of receivers locking on to the spoofed signal.

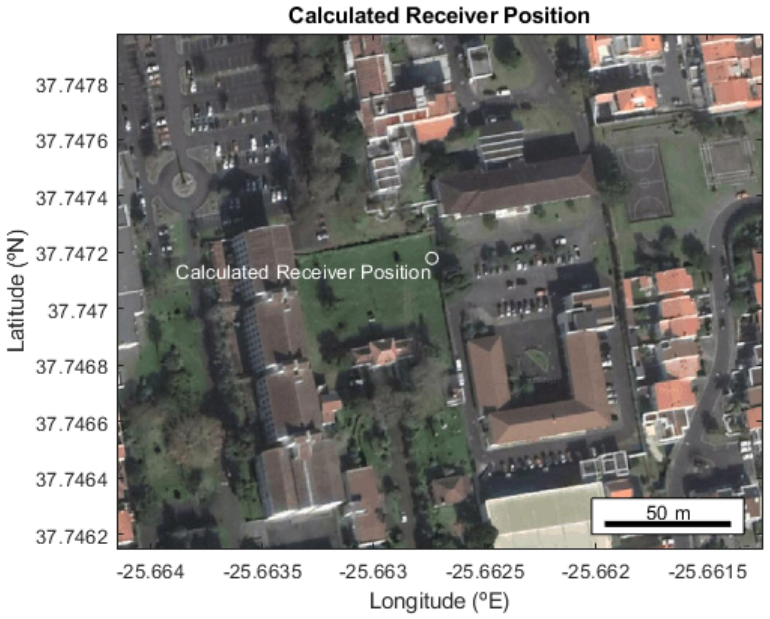

FIGURE 3.9: Calculated receiver position plotted on an accurate satellite image composition of the surrounding area

Having successfully calculated the receiver's position using the GPS navigation data, the aforementioned position is then plotted on a graphical representation of the area around it. The background image of this plot can be either a composition of satellite imagery (as shown in Figure 3.9) or a Google Maps representation of the area (as shown in Figure 3.10). Generation of these images is achieved by

using a set of Matlab scripts which take the position in Longitude/Latitude coordinates of the receiver's position and, using calls to the Google Maps application programming interface (API), overlay the plot's data on an accurate portrayal of the surrounding area [35].

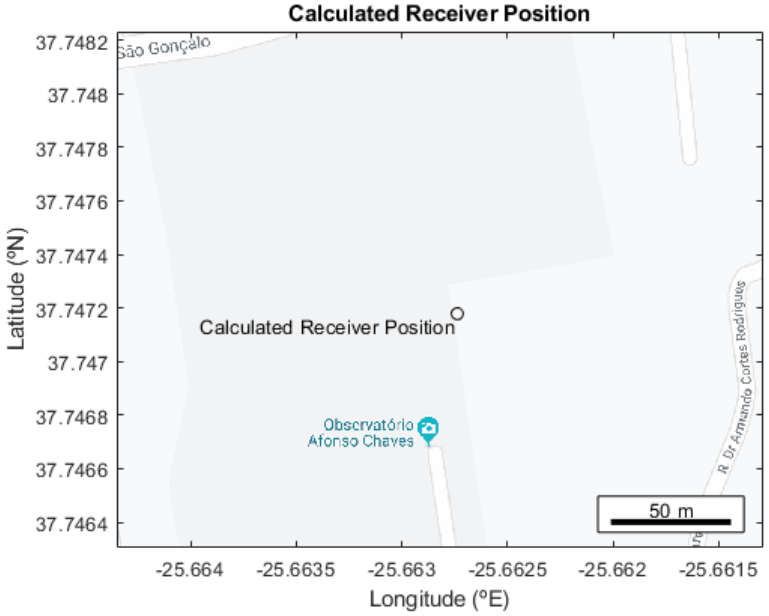

FIGURE 3.10: Calculated receiver position plotted on an accurate map-like representation of the surrounding area

These enhanced plots present the possibility of manipulating the viewport being displayed, which essentially enables scrolling through the map and zooming in and out while updating the displayed area and scale on-the-fly in a user experience akin to browsing for a specific location on Google Maps.

Plotting the position on the map makes it easier to visualise where on the globe the receiver that gathered the GPS navigation data is situated and also facilitates detection of any subsequent alterations to this position produced by the experimental results of the spoofing attempts.

The next step in the simulation is to perform alterations to the GPS navigation data, as described in greater detail in section 3.1, and to plot all variations of the altered position in relation to the original, unaltered, position previously calculated (Appendix A.3). This is done for the data of each of the satellites within line of

sight of the receiver. An example is shown in Figure 3.11 where each new position is plotted around the original calculated position and represented alongside the space vehicle number (SVN) of the satellite whose data was altered to produce the position.

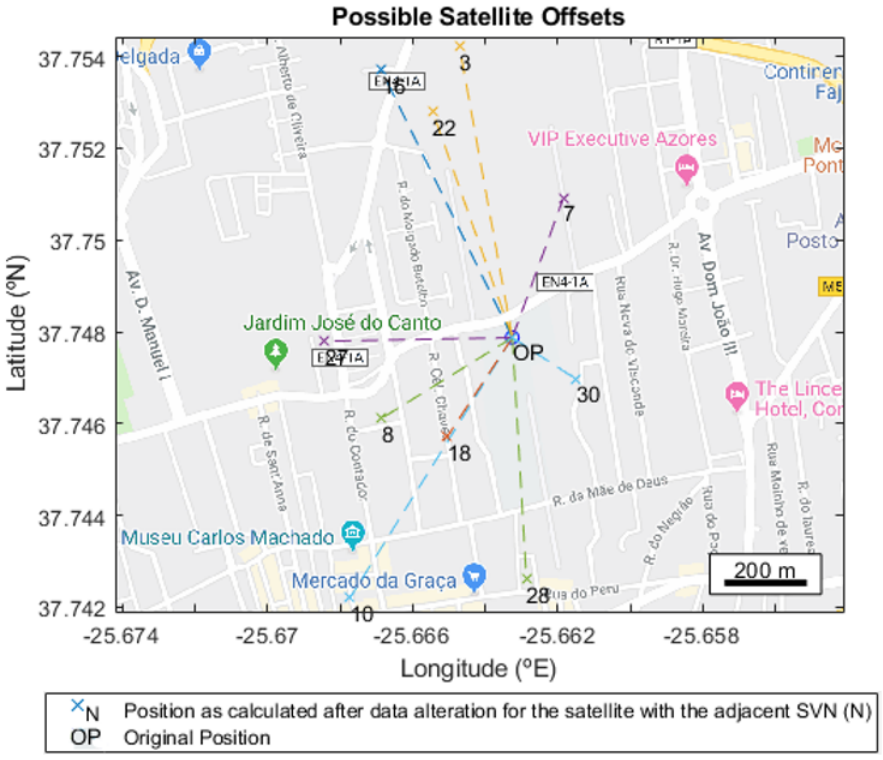

FIGURE 3.11: Visual representation of altered positions around the original receiver position due to alteration of clock correction polynomials for each satellite within line of sight

Alterations to the position can now be manually inspected and evaluated. Once evaluation is concluded, and a final spoofed position has been chosen for the experiment, the data is saved in .txt files to be parsed by an external python script, which concludes the Matlab simulator's part of the experimental workflow.

The experiment up to this point allows us to assert the validity of proposed hypotheses regarding alteration of GPS navigation data as a means to alter a GPS receiver's perceived position in a controlled and simulated environment. To further test these hypotheses and assert their validity in a real-world scenario, the altered data must be used to generate a valid GPS signal, which, in turn, will be transmitted to actual GPS receivers for analysis and position calculation.

At this stage the altered navigation data to be transmitted is now ready to be rewritten into new RINEX Navigation files to be used in generating a spoofed GPS signal. This is achieved by using a python script to parse this data, along with the original RINEX files used to generate it and alter the latter to reflect any and all changes done during the course of the simulation (Appendix C.1). Once the new RINEX files have been produced, they are used as an input to a GPS signal simulator (GPS-SDR-SIM) [36]. This generated GPS signal is then transmitted using an Ettus Research N210 USRP SDR peripheral [23].

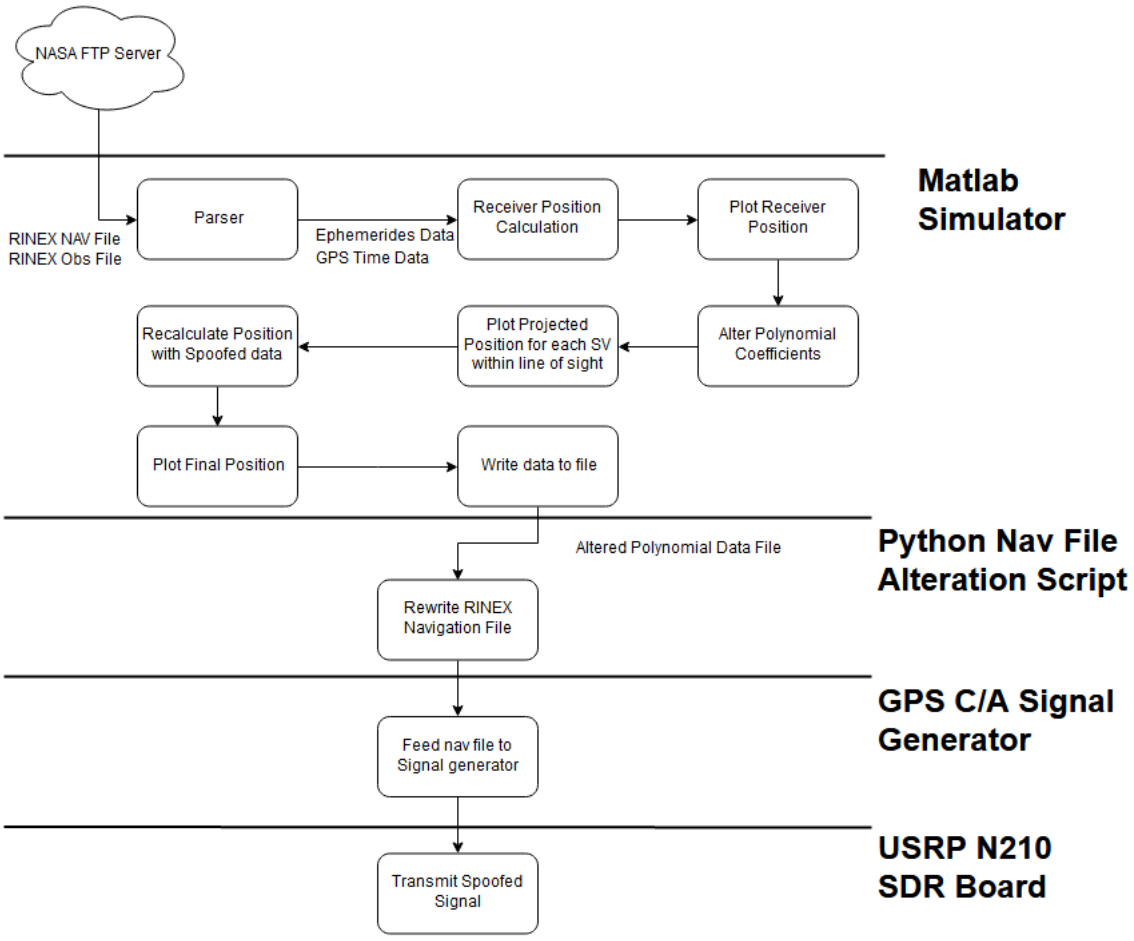

FIGURE 3.12: Basic simulator and experimental system flowchart

3.3 Simulator Experimentation

The main hypothesis in study for the current dissertation is that by altering how a GPS receiver corrects for the delay in the transmission of GPS satellite ephemerides data messages, the pseudoranges used to calculate user position will be impacted, effectively causing the perceived position to be shifted from its actual location.

In order to assert the impact of altering GPS ephemerides data on the receiver's perceived position, several tests were conducted with different parameters being altered for each of these tests. The parameters in question are all, in some way, part of the equations used to calculate receiver position in relation to the satellites.

3.3.1 Experimentation With Offsets in the SV Clock Bias Correction Coefficient(af0)

The results described in this section refer to the experiments in which only the af_0 coefficient was altered by the addition of an offset. The offset values used were in the order of 1x10^{-5}, 1x10^{-6} and 1x10^{-7} seconds.

As a first analysis, the values of altered pseudoranges can be compared to the original, untampered, values of the pseudoranges calculated from each SV to the receiver. Examining tables 3.1 and 3.2, it is clear that, for small offsets applied to the coefficients, the difference between these values is almost indistinguishable at a first glance. One would have to actively be looking for unexpected alterations in the pseudoranges and still, it would, most likely, be attributed to minor variations due to expected, normal error in measurement or due to corrections applied on the receiver end.

If the difference in these values were to vary greatly from the initial value, such as if the first applied offset was in the order of 1x10^{-5} seconds, for instance, then it would, most certainly, be noticeable. Yet, the perceived position would, most certainly, have shifted from the original location so suddenly that this alteration in and of itself would raise suspicions far quicker than in the case of monitoring

pseudorange values, making it a moot point to try and account for very sudden and large-scale changes in the coefficients.

So, while the difference in meters between pseudoranges is kept to a minimum, providing that the chosen offsets do not exceed a few microseconds (in theory meaning the range to work with for successful covert spoofing should be below $1x10^{-6}$ s in magnitude), these small changes to the pseudoranges seem to be enough to shift the perceived position by a distance ranging from a few meters to several hundred meters depending on the magnitude of the offset applied, as evidenced in figure 3.13.

Offset (af_0)	**Spoofed ρ (m)**	**$\Delta\rho$ (m)**	**% Relative to Original ρ**
$1x10^{-5}$	22394653.474654	516.050810	100.0023
$1x10^{-6}$	22394188.912845	51.489001	100.0002
$1x10^{-7}$	22394142.514155	5.090311	100

Original SV 6 Pseudorange = 22394137.423844 m

TABLE 3.1: Spoofed pseudorange values for different offset values applied to the af_0 coefficient of the satellite with SV 6

Offset (af_0)	**Spoofed ρ (m)**	**$\Delta\rho$ (m)**	**% Relative to Original ρ**
$1x10^{-5}$	22359493.928134	785.818698	100.0035
$1x10^{-6}$	22358836.818615	128.709179	100.0006
$1x10^{-7}$	22358770.673235	62.563799	100.0003

Original SV 7 Pseudorange = 22358708.109436 m

TABLE 3.2: Spoofed pseudorange values for different offset values applied to the af_0 coefficient of the satellite with SV 7

Comparing the original position with the spoofed positions for different offset values, it is clear that the distance between these two points increases proportionally to the applied offsets, which is to say, the greater the offset, the greater the distance of the spoofed position from the original location.

Separating the offsets into different plots based on their order of magnitude (which in this case will produce three different plots for offsets within the $1x10^{-5}$ s, $1x10^{-6}$ s and $1x10^{-7}$ s ranges) illustrates the fact that each order of magnitude produces a different range of distances from the original position (Figure 3.13). Offsets within the $1x10^{-5}$ s range allow for spoofed positions over 1km away from the original while smaller offset values, such as $1x10^{-7}$ s only allow for distances up to 10m and offsets in the $1x10^{-6}$ s ranges stand in a sort of middle ground with distances ranging between 10m to 100m.

Since the distance varies proportionately to the applied offset, each range of offsets presents a different possible range to work with: larger offsets can be applied for sheer distance from the position and then smaller offsets can be applied to fine tune the position depending on the desired distance range.

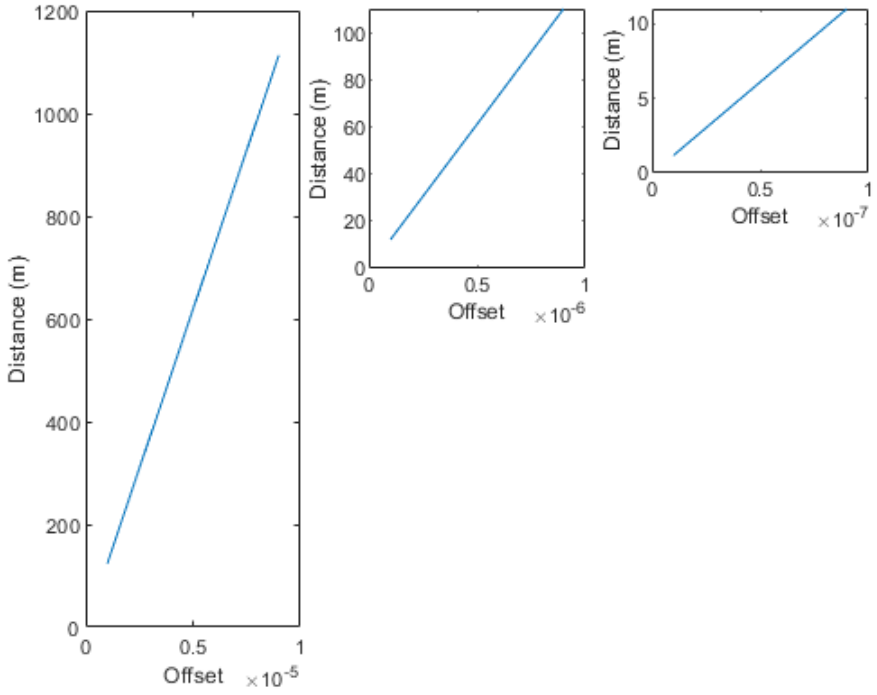

FIGURE 3.13: Distance of spoofed positions from the original calculated receiver position in relation to offsets applied to the af_0 correction coefficient

When considering the fact that a spoofing operation should, in most scenarios, be carried out covertly (depending on the purpose and conditions of course), it should be noted that it is in the best interest of the spoofer not to raise any

suspicion concerning neither the shifting position of the drone nor the varying parameters of the navigation message being received. So, bearing this in mind, larger offset values which produce massive variations in distance from the actual position should be used in moderation. A better approach would be to gradually and moderately adjust the applied offset in order to slowly shift the position, rather than doing so in a single instant. While it is advantageous from the perspective of a spoofer which is mindful of the possibility that the receiver can detect shifts in the navigation message parameters, to try and produce a different position without noticeably altering a great number of the numerals in a given parameter, it should be noted that a large shift in the perceived position will, arguably, be much more noticeable even by an unsuspecting receiver.

The distances presented in Figure 3.13, and in subsequent figures representing distance from this point forward, are calculated using a Haversine formula [37] to determine the distance over the surface of the Earth between the original, unaltered, position and each subsequent spoofed position. This allows accurate calculation of the distance on a spherical representation of the Earth between two points using their respective latitude and longitude values.

A Haversine (or half-versed sine) can be defined by the squared sine of half the angle in question:

$$haversine\,(\theta) = \sin^2(\theta/2) \tag{3.13}$$

Taking this formula, it is possible to rework it in order to obtain the square of half the chord length between the points ($a = (\frac{l}{2})^2$) from their respective latitude (φ) and longitude (λ) coordinates:

$$a = \sin^2\left(\frac{\varphi B - \varphi A}{2}\right) + \cos\left(\varphi A\right)\ \cdot \cos\left(\varphi B\right)\ \cdot \sin^2\left(\frac{\lambda B - \ \lambda A}{2}\right) \tag{3.14}$$

In this case, $\boldsymbol{A}$ denotes the original receiver position while $\boldsymbol{B}$ represents the spoofed position to where the distance is being calculated (Figure 3.14).

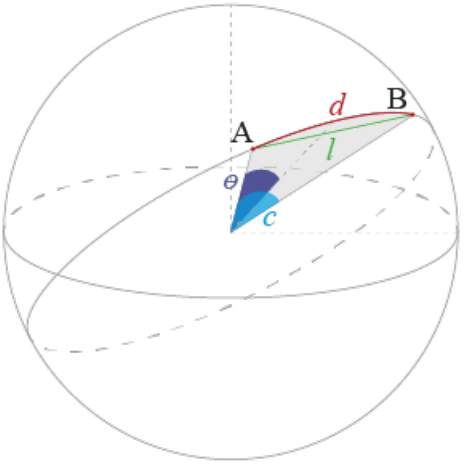

FIGURE 3.14: Representation of the variables required for application of a Haversine formula to determine the great circle distance between two points on a sphere

Having the value of ($\boldsymbol{a}$), it becomes possible to calculate the angular distance in radians between the two points ($\boldsymbol{c}$) by using the two-argument arctangent (atan2) to determine the angle in the Euclidean plane between $\sqrt{a}$ and $\sqrt{(1-a)}$. Or, equivalently, for broader use in case the atan2 function is not available, the arcsin of $\sqrt{a}$, with a maximum value of 1.

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right) = 2 \cdot \arcsin\left(\min(1, \sqrt{a})\right) \tag{3.15}$$

Multiplying $\boldsymbol{c}$ by the radius of the Earth ($\boldsymbol{R}$) returns the great-circle (or orthodrome) distance between the two positions in question ($\boldsymbol{d}$), which is to say, the distance between the two positions, along the shortest arc of the circle resulting from the intersection of the spherical representation of the Earth and a plane which passes through the centre point of this sphere, dividing it into two equal hemispheres [38], as illustrated in Figure 3.15. This distance is also commonly referred to as: "as the crow flies".

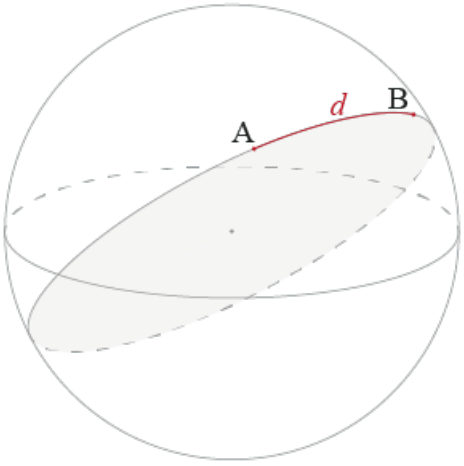

FIGURE 3.15: Great circle distance between points A and B

$$d = R \cdot c \tag{3.16}$$

Any distances corresponding to spoofed positions from this point forward will also be calculated using this method.

3.3.2 Experimentation With Offsets in the SV Clock Drift Correction Coefficient(af1)

The results described in this section refer to the experiments in which only the af_1 coefficient was altered by the addition of an offset. The offset values used were in the orders of $1x10^{-5}$, $1x10^{-6}$ and $1x10^{-7}$ seconds per second.

For these experiments, the offsets used were in the same order of magnitude as those used in the experiments with af_0. This was done in order to enable direct comparison of the results from simulations and real-world testing alike.

Comparing the spoofed pseudoranges with the original pseudoranges for each satellite it is, once again, clear that the higher the offset applied, the greater the gap between pseudoranges. While not as pronounced as the changes derived from

alteration of the af_0 coefficient, altering af_1 also has a measurable impact on the pseudorange measurements and, consequently, perceived receiver position, as shown in Figure 3.16.

Offset (af_1)	**Spoofed ρ (m)**	$\Delta\rho$ (m)	**% Relative to Original ρ**
1x10^{-5}	22394099.310235	-38.113609	99.99982981
1x10^{-6}	22394133.791546	-3.632298	99.99998378
1x10^{-7}	22394137.180489	-0.243355	99.99999891

Original SV 6 Pseudorange = 22394137.423844 m

TABLE 3.3: Spoofed pseudorange values for different offset values applied to the af_1 coefficient of the satellite with SV 6

Offset (af_1)	**Spoofed ρ (m)**	$\Delta\rho$ (m)	**% Relative to Original ρ**
1x10^{-5}	22358708.832329	0.722893	100.0000032
1x10^{-6}	22358757.756060	49.646624	100.000222
1x10^{-7}	22358762.658039	54.548603	100.000244

Original SV 7 Pseudorange = 22358708.109436 m

TABLE 3.4: Spoofed pseudorange values for different offset values applied to the af_1 coefficient of the satellite with SV 7

Much like the previous experiments with offsets in af_0, the distance of spoofed positions from the original increases proportionally with the increase in the values of applied offsets but the rate of this increase is vastly inferior to the previous cases. For offsets in the 1x10^{-7} seconds/sec range the maximum distance is under 1 meter and changes in the order of 1x10^{-5} seconds/sec will only produce a spoofed position up to around 80 meters from the original. Furthermore, the direction in which the position has been shifted is opposite the direction that was observed in the previous section, as evidenced by the negative value of the pseudorange difference in Table 3.3. This indicates that for the same satellite, different polynomial correction

coefficients will have similar effects on the perceived receiver position but with some variation in regard to distance and direction of the position shift.

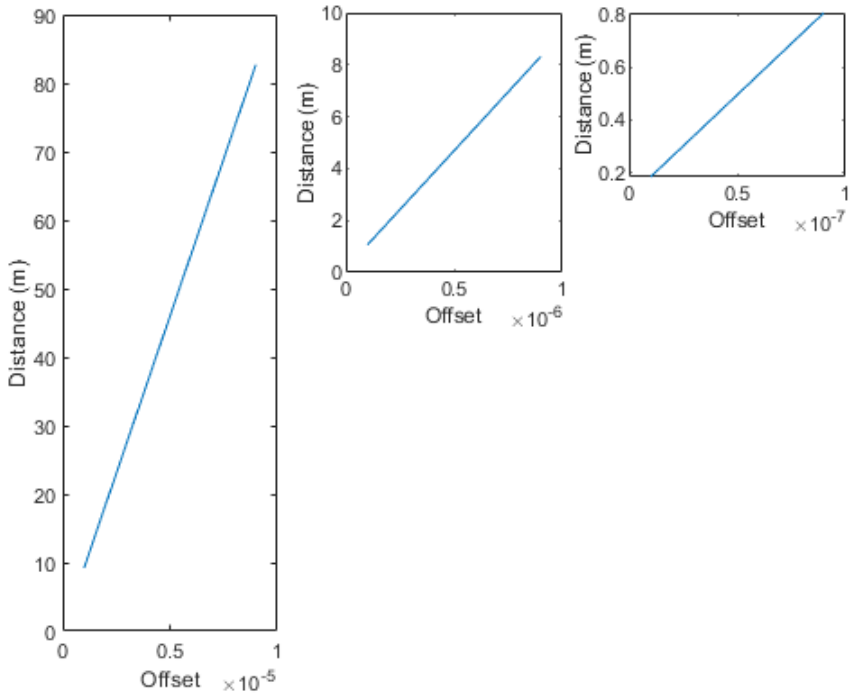

FIGURE 3.16: Distance of spoofed positions from the original calculated receiver position in relation to offsets applied to the $\boldsymbol{af_1}$ correction coefficient

The preliminary results from these experiments suggest that, while altering the $\boldsymbol{af_0}$ coefficient produces more noticeable and more effective changes to the calculated receiver position, altering $\boldsymbol{af_1}$ may prove to be useful in situations where the position is not meant to be shifted so drastically and may even be the more sensible approach in such a scenario, since it may allow for greater control over the smaller distances of the position shift without altering the values in navigation files in an easily perceptible way.

3.3.3 Experimentation With Offsets in the SV Drift Rate Correction Coefficient(af2)

As previously mentioned, altering $\boldsymbol{af_2}$ produced very minimal variation in the calculated receiver position and the value for this coefficient was, in fact, null

for many of the Ephemerides in the RINEX files used for the simulations. This essentially resulted in simulations where the spoofed positions were almost imperceptibly shifted from the actual position, even though the coefficient had, very noticeably, been altered in the RINEX file, since stored the value shifted from "0.000000000000D+00" prior to the spoofing attempt, to anywhere between "0.000000010000D+00" and "0.000001000000D+00" post spoofing.

As such, alterations to the $\boldsymbol{af_2}$ coefficient were not taken into account when testing the hypothesis considering real world scenarios.

3.3.4 Experimentation With Offsets in Multiple Parameters

The offsets applied to individual parameters can, naturally, also be applied to multiple parameters simultaneously, so some experiments were conducted using offsets for both clock bias and clock drift correction coefficients to determine if the effects were, in any way, different, or if the cumulative effects of the alterations were a sum of the effects described in sections 3.3.1 and 3.3.2, as expected. What was determined was that the latter option proved to be correct and this approach does, in fact, produce changes in position analogous to the previous experiments, but with the final position being determined by a sum of both of the applied offsets' effects.

While one might argue that altering both parameters at once presents no immediately apparent benefit over altering a single parameter and adjusting the offset value to achieve the same result, there are two major reasons why this may be advantageous.

Firstly, altering more than one parameter may allow for smaller offsets being applied to each of the parameters, making variations between genuine and spoofed signal harder to detect, while still keeping the end result essentially the same. This may prove to be more beneficial than inputting a larger offset to a single parameter in a scenario where the receiver is actively monitoring the data in GPS signals, either as a means of detecting a spoofing attempt or otherwise.

Secondly, if the receiver is actively correcting for these kinds of alterations, altering more than one parameter may serve to further increase the robustness of the spoofing attempt by exploiting shortcomings in the correction algorithm. If the receiver fails to correct for all the alterations in the spoofed signal, then those not accounted for might just be enough to conduct a successful spoofing attempt regardless, albeit with negative impact on the efficiency of the spoofing procedure.

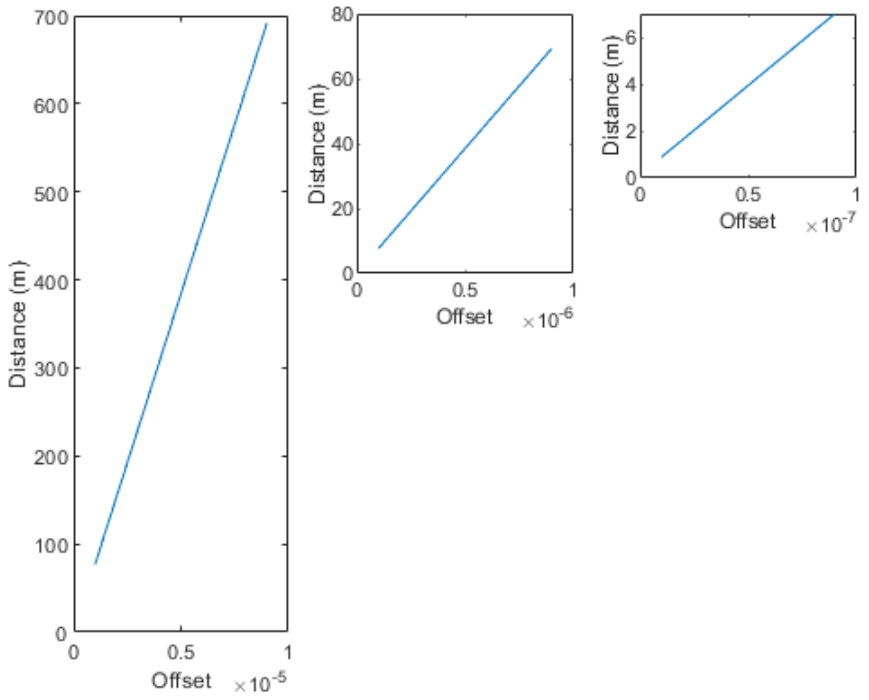

FIGURE 3.17: Distance of spoofed positions from the original calculated receiver position in relation to offsets applied to both $\boldsymbol{af_0}$ and $\boldsymbol{af_1}$ correction coefficients

As evidenced in Figure 3.17, the distance of the spoofed positions from the original unaltered one is impacted in a similar way to that of previous scenarios where only $\boldsymbol{af_0}$, ranging from variations of a few meters for offsets of magnitude $1\text{x}10^{-7}$ to several hundred meters for larger offsets of $1\text{x}10^{-5}$ of magnitude.

The similarity between these two different experimental scenarios is due to the fact that, as outlined in the previous sections, $\boldsymbol{af_0}$ has a much greater impact on the final spoofed position than $\boldsymbol{af_1}$ so, naturally, when using both parameters in conjunction and applying the same offset to each to produce variations in the

receiver position, $\boldsymbol{af_0}$ will produce the most noticeable effects on the spoofed position.

In this particular scenario, with $\boldsymbol{af_1}$ actually producing a drift in calculated position in the opposite direction of that produced by $\boldsymbol{af_0}$, the combination of offsets in the two parameters in question creates a minor shortening effect in the distance from the actual position.

Chapter 4

Real World Test Scenarios

4.1 Experimental Setup

In this section, an overview of the hardware and software used for the experimental procedures of the real world test scenarios will be made, along with an explanation of the process itself. The main components of the code described in the present section are available in Appendices B and C.

The experimental procedure begins with running the simulator described in the previous sections. Once the RINEX navigation file containing the altered clock correction coefficients has been altered, it can be used to produce a spoofed GPS signal.

The altered RINEX file is generated by taking the offsets used for the final position in the simulation and adding them to the corresponding parameters of the satellites used. Since the RINEX files are ASCII text based, altering the data on them can be accomplished by simple text manipulation between two files, parsing the information to find the correct location for the values on one file and then replacing them with the new data from the other.

To do so, a python script was produced to parse the original RINEX navigation file used to perform the simulation and search for values to be altered in accordance

to simulation results (Appendix C.1). The new values for the altered data are obtained from a .txt file created upon simulator runtime completion, they are formatted to conform to the standards of the data in the navigation file and then copied into the position of the data they are to replace.

Having made the required alterations to the RINEX files, the spoofed GPS signal can be generated by using the GPS-SDR-SIM GPS signal generator. Using the data from the altered navigation file, this open source software is able to accurately determine satellite ephemerides, from which it can compute the pseudorange measurements required to effectively place a GPS receiver's perceived position on a given set of coordinates. These coordinates are provided to the software, either as a single stationary point or a set of coordinates which change over the course of the generated signal's time span. The latter option essentially mimics a moving user, so a stationary receiver would perceive its position to be shifting as if it were moving, but for the purposes of validating the results obtained from the Matlab simulation, the spoofed position for each scenario should be static, therefore the spoofed signal will be generated with a single position.

GPS-SDR-SIM was built with the purpose of simulating a valid GPS signal for a given position, as such, the core functionality of its underlying algorithms is built upon the assumption that the creation of data which will be measured by the target receiver should be done using the provided position. When determining the pseudorange values, which will effectively simulate what a receiver might perceive for the specified position, rather than performing the calculations as described in section 3.2, the software will, instead, take advantage of the fact that the receiver position is no longer an unknown variable but in fact a known set of ECEF (x, y, z) coordinates and directly compute the Euclidean distance between this position and the SV positions determined using the provided RINEX navigation file (Figure 4.1). While this is advantageous when trying to reduce the complexity of the computations and thus allow for faster spoofed signal generation, it essentially bypasses most of the calculations dependent on the polynomial clock correction coefficients in study. These are still taken into account for satellite position calculations, which will produce some observable difference in the calculated position,

but not as much as if the entire process of receiver position determination was conducted.

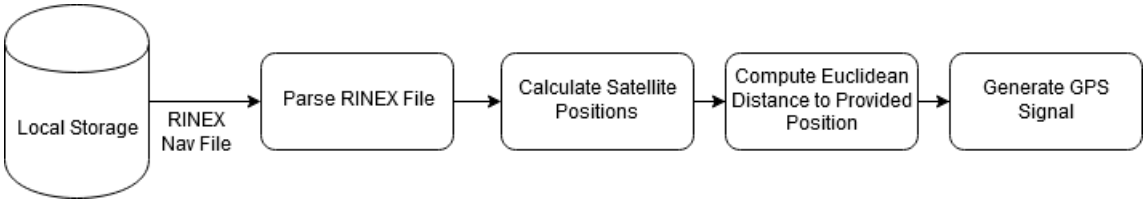

FIGURE 4.1: Simplified model of the functionality of GPS-SDR-SIM

This fundamental design decision means that for the purposes of the experiments conducted to validate the present work's hypothesis, some modifications had to be done to the software since the altered parameters would, otherwise, not be considered. These alterations were made to ensure the software takes into account the altered clock correction coefficients during signal generation, thus allowing the calculation of the spoofed position to be affected by the altered RINEX files and enabling evaluation of the impact of the proposed hypothesis in a real-world test scenario.

The solution was to take part of the algorithm that calculates the receiver position for the Matlab simulator, refactoring to convert it from Matlab code to C so it can be run within the signal generator software during its normal runtime operation and inserting the calculated position where the static, provided position would have been on the original signal generator (Figure 4.2).

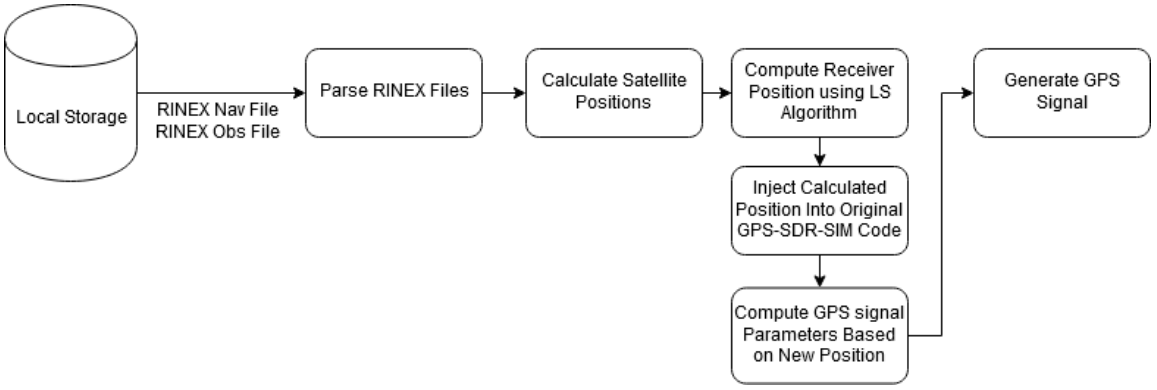

FIGURE 4.2: Simplified model of the functionality of GPS-SDR-SIM with the proposed modifications

This approach uses some data which is already being either parsed or calculated by the program, but two major algorithms must be added to the code, namely, the least squares iteration method to properly determine the position (Appendix B.1), and a RINEX observation data file parser to obtain the calculated pseudoranges to each of the SVs as measured from the target location (Appendix B.2).

As detailed in 3.2, to determine the receiver position using a least squares iterative method, pseudoranges to each of the satellites being used must be calculated and these must be corrected using the satellite clock correction equations 3.1, 3.2. The software already parses the provided RINEX navigation file, so clock correction coefficients, as well as any other ephemerides data required, are already present and readily accessible without needing to alter the generator's source code. As for obtaining the necessary pseudorange values for each of the satellites within line of sight, the aforementioned RINEX observation data parser was developed. This algorithm takes the file in question as an input, parses through it to determine which satellites are within line of sight and the values of their respective pseudoranges to the receiver.

Having collected all the necessary data for position calculation, focus then shifts to the least squares iteration algorithm which is, as far as possible, a one to one replication of the code used in the Matlab simulation described in section 3.2, converted to C. Upon completion of the iterative calculation process, the resulting calculated position is passed to the main signal generator operation as if it were the static position to be simulated.

By doing so, it is ensured that the coefficients in study are taken into account during signal generation and, consequently, that the positions perceived by receivers using this signal as a basis to calculate their location will accurately reflect the impact altering the coefficients has on said calculations. The generated signal is stored in a .bin file with approximately 3Gb, which is essentially a 300s long simulation of all the data GPS satellites would transmit to a receiver located at the intended spoofed position. This file can be "played back" and transmitted, second by second, using a variety of SDR peripherals.

For the following test scenarios, the signal was transmitted using an ETTUS N210 USRP platform, connected via ethernet to a laptop running the script which enables the replay of the generated signal file, the antenna chosen for this experiment was a passive, omnidirectional antenna.

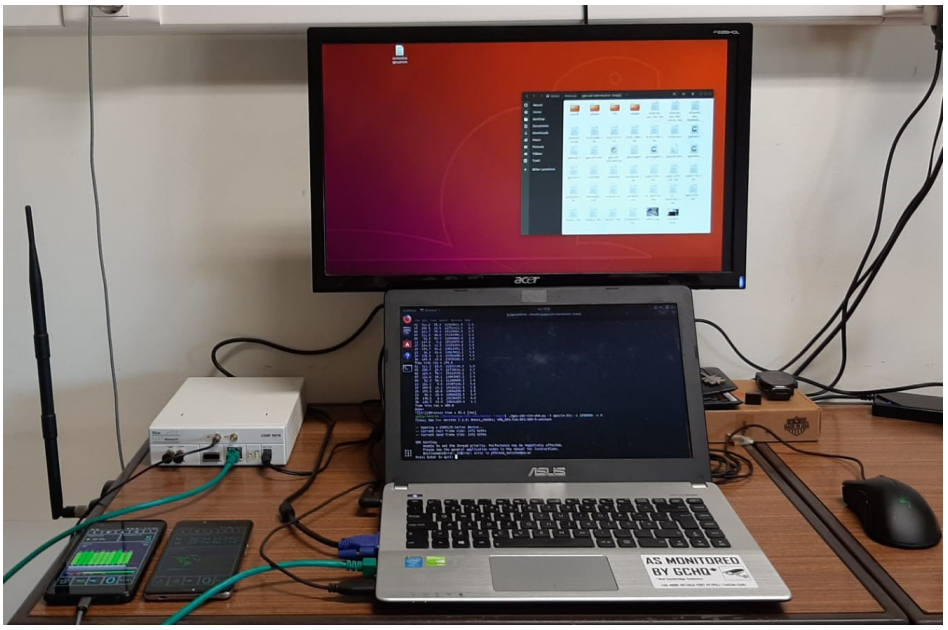

FIGURE 4.3: Experimental hardware setup

Once the signal is transmitted using the hardware detailed above, it can then be received by any commercially available GPS receiver. For the purposes of the current work, two android smartphones running the GPS Test app [39] were used to validate the effectiveness of the spoofing signal. This app shows, in real time, all the data pertaining to the device's GPS receiver which includes, but is not limited to: the device's perceived coordinates, heading and speed; the satellites within line of sight, their respective received signal power and the GPS time as perceived from the received signals.

This experimental setup forms the test suite used for the experiments detailed in the following sections. For each scenario tested, the process is as follows:

- 1) Run the Matlab simulator to assert the required alterations to the RINEX files and generate the data;

- 2) Run the python script to alter the RINEX navigation file based on generated data from the simulator;
- 3) Produce the spoofed GPS signal using the RINEX navigation file mentioned above and its corresponding observation file as input for the signal generator;
- 4) Run the python script to playback the signal and transmit it using the USRP N210;
- 5) Capture the signal using the two aforementioned smartphones, paying close attention to the position coordinates, accuracy and detected satellite constellation;
- 6) Verify the position by using the Google Maps app to assert that the device's position has been changed to that of the intended spoofed position.

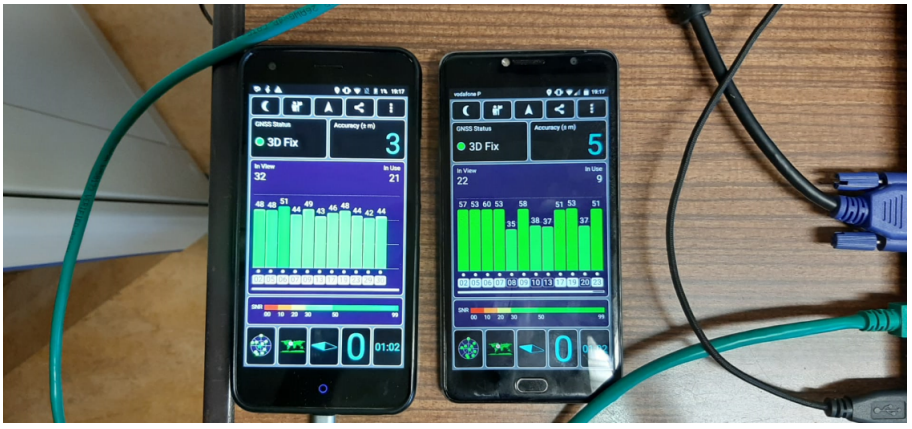

FIGURE 4.4: Smartphone receivers during spoofed signal capture

As a way to facilitate data analysis, all the positions of the captured signals are then plotted on a map of the surrounding area using a modified version of the Matlab algorithm described in section 3.2.

Since the laboratory where the experiments were conducted is effectively isolated from outside interference, the generated signal easily overpowers any legitimate GPS signal received by the smartphones.

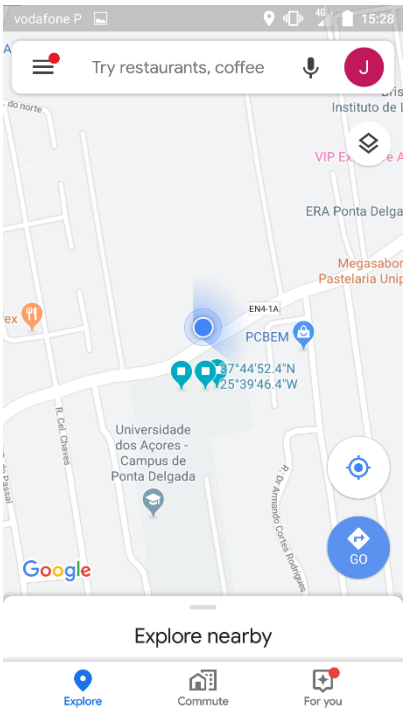

FIGURE 4.5: Example of a spoofed position perceived by one of the receivers (shown in the Google Maps app)

In the early stages of testing, all mobile data, Wi-Fi and cellular connections on the smartphones were also disabled to prevent position prediction based on network services or base station triangulation but it soon became apparent that the spoofed signal was effective enough to trick the device even with the auxiliary position calculation methods provided by these services enabled. Since no discernible interference to the results was detected, both Wi-Fi and cellular data connections were left enabled for the remainder of the experiments, providing an environment closer to a real world scenario where the drone's receiver will, in all likelihood, have more than one way to determine its position.

The data used for these experiments was downloaded from NASA's FTP server and consists of RINEX navigation and observation files generated from GPS Signals captured in Universidade dos Açores' Campus in São Miguel, Azores Islands on the 27th of May 2019. The data contained in these files spans a 24h period from midnight of the 27th to midnight of the 28th.

The satellites chosen to illustrate the effect on perceived receiver position

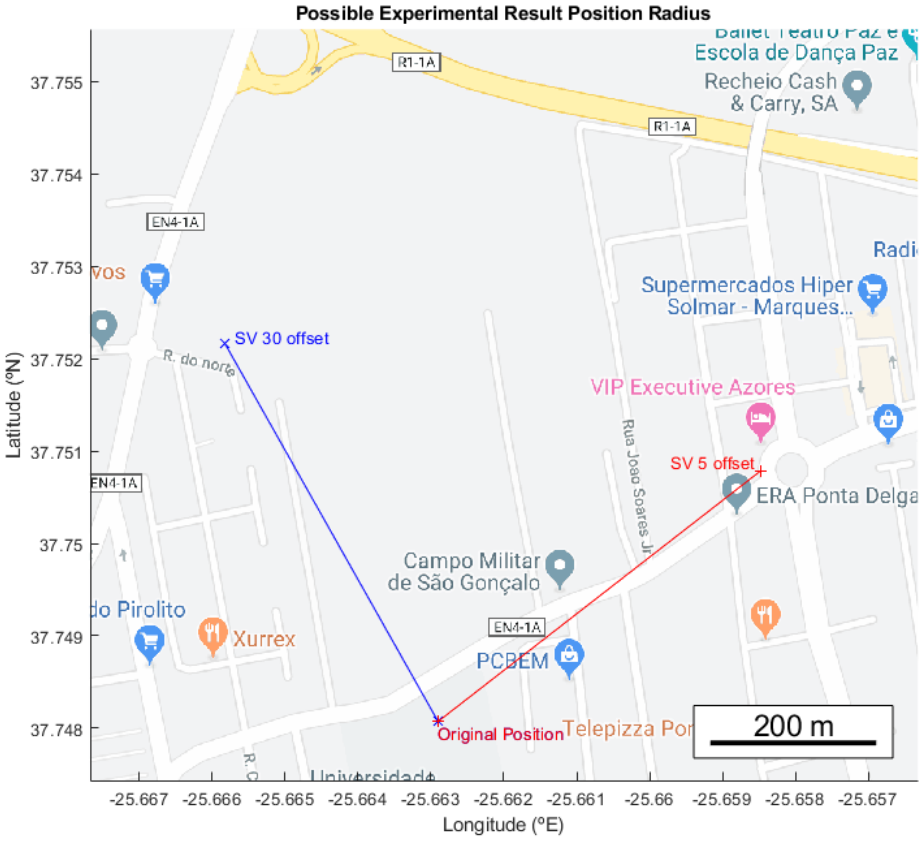

FIGURE 4.6: Simulator plot of the area for possible experimental result positions with visual representation of the impact of applied offset for satellites with SVN 5 and SVN 30

of altering the clock correction coefficients for this scenario were SVs 5 and 30, since they present the ideal conditions for spoofing the receiver position using the methodology proposed in chapter 3.1. The vectors created by plotting a line from the original position to the spoofed position generated by each of the SVs present almost the same length, meaning that the impact of applied offsets will be equivalent for both satellites, and the angle between them is very close to 90°, which makes it so that the spoofed position can be moved freely within the surface defined by these two vectors by considering the offset applied to each satellite's data as a shift in a given direction.

4.1.1 Experimentation With Offsets in the SV Clock Bias Correction Coefficient (af_0)

The results described in this section refer to the experiments in which only the af_0 coefficient was altered by the addition of an offset. The offset values used were in the ranges of $1x10^{-5}$, $1x10^{-6}$ and $1x10^{-7}$ seconds.

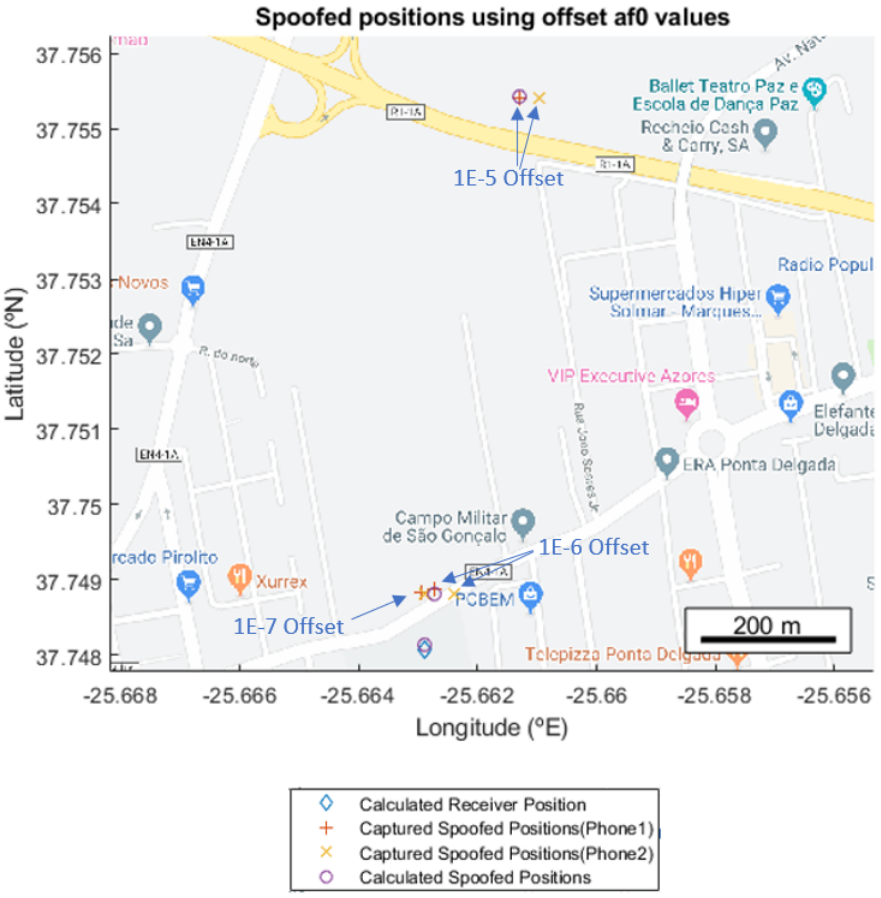

FIGURE 4.7: Captured experimental spoofed positions for offsets applied to af_0

In general terms, the effect of altering polynomial coefficients used for clock correction seems to be analogous to the results observed in Matlab simulations with similar parameters.

Comparing the maximum distances of the spoofed positions relative to the actual position, the results seem to follow what was observed in section 3.3.1.

Offset	**Distance from Original Position (m)**
$1x10^{-5}$	830.9864
$1x10^{-6}$	89.3814
$1x10^{-7}$	8.4984

TABLE 4.1: Maximum distance from original position for each range of offsets applied to af_0

Larger offsets will produce a spoofed position farther away from the original calculated receiver positions, while smaller offsets lead to spoofed positions closer to the original. As presented in Table 4.1, offsets in the $1x10^{-5}$ s gamut are able to produce shifts in perceived position up to approximately 830m away from the actual position, while offsets of magnitude $1x10^{-7}$ present shifts of only a couple of meters.

The maximum distances are a bit shorter than observed in the simulations presented as examples in section 3.3.1, but still within the margin of error expected. These values are susceptible to change due to factors inherent to each scenario and variation of up to 30% was observed not only between experimental scenarios, but also within the same scenario, when selecting a different pair of satellites for the experiment, even in simulations.

As observed in Figure 4.7, the captured positions from real world test scenarios fall, for the most part, very close to their simulated counterparts, which seems to validate, on a physical level, the results previously observed in the simulations.

Some anomalous results are present though, such as the values for applied offsets of range $1x10^{-7}$ being closer to the expected position for offsets of range $1x10^{-6}$. This was a common occurrence for most of the experiments run, and there are a few options worth exploring to attempt to find the cause.

Since the position is wrongly shifted in equal measures for both receivers rather than just one, a device malfunction is very unlikely to be the cause of these anomalous results and can therefore be ruled out as a hypothesis.

Analysing the accuracy reported on the measurements taken from each device, the mean accuracy is approximately 5m with a few instances of greater inaccuracy but not very far from this value. While this certainly contributes to the inaccuracy of the perceived position, it is not enough to be the sole cause of the issue.

There is also the possibility of external interference from other signals whether they are being actively being used by the receiver to correct the position or they are simply interfering with the spoofed signal's transmission. The first hypothesis was debunked by re-running the experiments with all the device's communication systems disabled, save for the GPS receiver, to be met with the same exact results as previously observed. The second hypothesis was ruled out by the fact that the testing environment is very efficiently isolated from outside interference, as stated previously.

Another possibility for the anomalous results is Dilution of Precision (DOP), which is a term used to evaluate the propagation of errors when calculating receiver position. Geometric Dilution of Precision (GDOP), in particular, is greatly affected by the positions of satellites relative to the receiver. For satellites in close proximity to each other, the overlapping areas for possible receiver locations are greater and thus the GDOP will also be greater, leading to errors in the position calculation [40].

Since each spoofed position is, as previously detailed, a combination of shifts in a given direction by altering the data for a set of satellites, and since these shifted positions are so close to each other, the approximation of the receiver position as well as each satellite's position relative to the receiver are being impacted. As such, the geometry of the satellites-receiver system may become less ideal, increasing GDOP and the error associated with position calculation, meaning that DOP might account for some of the offset between calculated and captured positions.

The most likely cause of this miscalculation, however, seems to be an issue with the generated signal itself, which suggests some fault in the algorithm's code. During the development of the position calculator segment of this algorithm, it was discovered that the precision of calculated positions was, in some instances,

not as good as that of the Matlab simulations. After tinkering with the data formats used, this fault was mitigated for most situations, but some combinations of variables seemed to present higher inaccuracies than others, and dealing with Latitude Longitude Height (LLH) coordinates, a shift of 0.0001° in the final calculated position coordinates is enough to shift the position by up to a few hundred meters relative to the correct position. The root cause of why exactly only some combinations of values produced errors of this magnitude was not conclusively determined, yet they seem to mostly affect the latitude coordinates and occur mostly with offsets in the $1x10^{-6}$ range. Of the previously mentioned, this seems to be the most probable hypothesis for the errors observed in the experiments detailed in this chapter, as seen in figures 4.7, 4.8 and 4.9.

4.1.2 Experimentation With Offsets in the SV Clock Drift Correction Coefficient (af_1)

The results described in this section refer to the experiments in which only the af_1 coefficient was altered by the addition of an offset. The offset values used were in the ranges of $1x10^{-5}$, $1x10^{-6}$ and $1x10^{-7}$ seconds/second.

Comparing the maximum distances of spoofed positions for each offset with the values obtained from the corresponding simulations, it is apparent that these distances vary, much in the same way for both cases.

Offset	**Distance from Original Position (m)**
$1x10^{-5}$	65.0389
$1x10^{-6}$	6.3595
$1x10^{-7}$	0.4510

TABLE 4.2: Maximum distance from original position for each range of offsets applied to af_1

As in the previous experiment with offsets in af_0, the maximum values for the distances are lower than those found in simulations but, once again, this is due to expected variations in the test conditions.

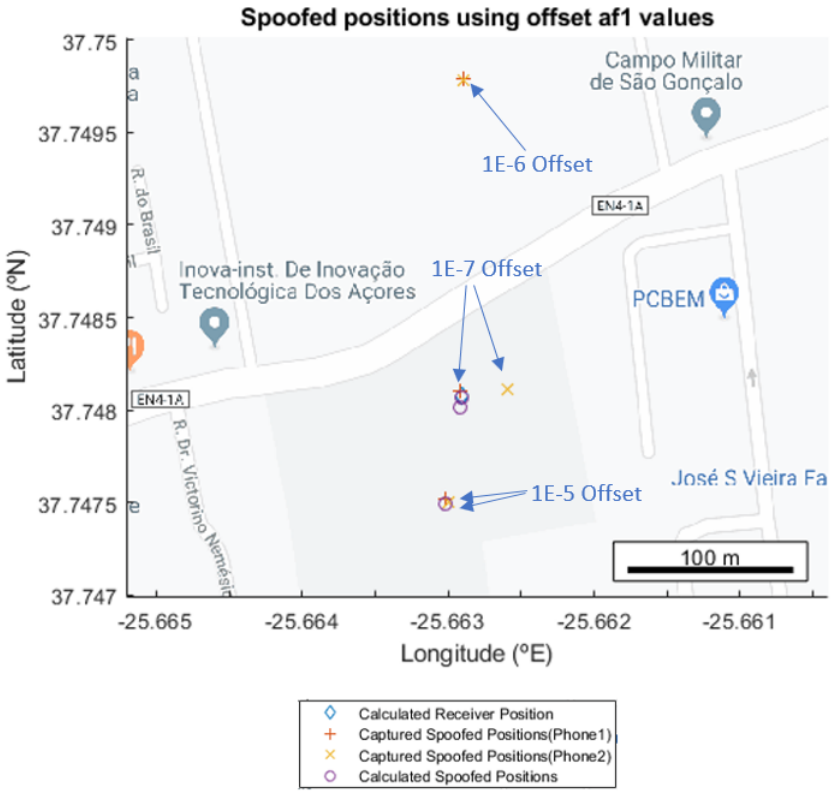

FIGURE 4.8: Captured experimental spoofed positions for offsets applied to af_1

Similarly to what was observed in the simulations detailed in section 3.3.2, exclusively altering the af_1 coefficient produces a shift in position in the direction opposite that of scenarios with offsets being applied only to the af_0 coefficient. Furthermore, the impact on the distance is, similarly, lessened to approximately a tenth of the maximum distance observed for offsets in af_0, with higher offsets of $1\text{x}10^{-5}$ magnitude producing a maximum position shift distance of around 65m versus 830m and lower offsets in the $1\text{x}10^{-7}$ range producing less than a half meter shift versus over 8m in the previous scenario.

These results are very similar to what was observed in the simulations, where altering af_0 similarly had a much more noticeable impact on the calculated position than altering af_1.

As observed in figure 4.8, the positions, once again fall very close to the expected, simulated positions, with the exception of those produced by offsets in

the 1x10^{-6} range which seem to have been erroneously shifted to a higher latitude than expected, in all likelihood, due to position calculation imprecision in the signal generation code, as previously discussed.

To an extent, the results of this experiment seem to validate their simulated counterparts, with positions shifting in the predicted direction and with variations in range analogous to those observed in the simulations.

4.1.3 Experimentation With Offsets in both the SV Clock Bias Correction Coefficient (af_0) and the SV Clock Drift Correction Coefficient (af_1)

In this section, both of the coefficients discussed in previous sections were altered simultaneously to assess whether this approach brings any advantage to the purposes of position spoofing versus altering only one of the coefficients. The offsets tested were in the range of 1x10^{-5}, 1x10^{-6} and 1x10^{-7} (seconds for the af_0 coefficient and seconds/second for the af_1 coefficient respectively).

As expected, the results are, much like in the simulations, a conjunction of the results from altering each of the considered coefficients individually.

Offset	**Distance from Original Position (m)**
1x10^{-5}	767.31
1x10^{-6}	76.9545
1x10^{-7}	7.8453

TABLE 4.3: Maximum distance from original position for each range of offsets applied to both af_0 and af_1

Greater offset ranges once again present the greatest shift in distance from the original position and the maximum values are stunted by the fact that now each coefficient will pull the position in an opposite direction. Despite this, since the af_0 coefficient has a much greater impact on the position calculation than af_1, all the values for this experiment present a clear bias towards the results of section 4.1.1,

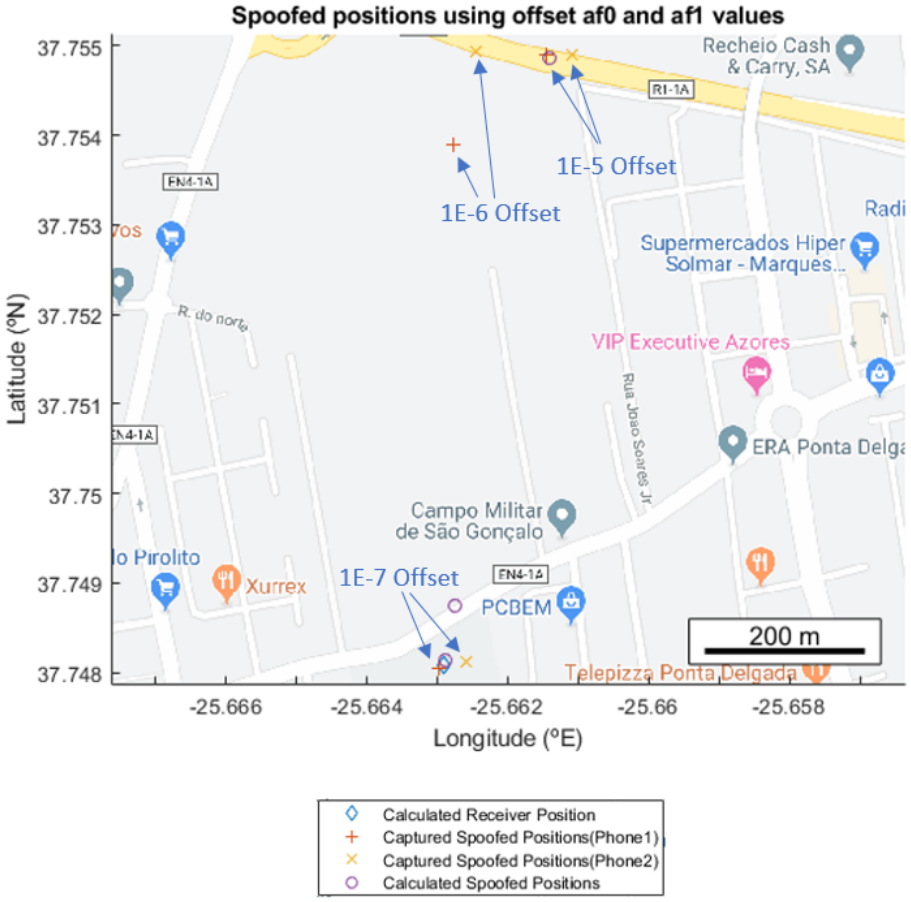

FIGURE 4.9: Captured experimental spoofed positions for offsets applied to both af_0 and af_1

as was the case with the simulator experiments of the same conditions presented in section 3.3.4, whose results were heavily biased towards those obtained in 3.3.2.

As was noted in the previous experiments, the latitude for the positions with offset $1\text{x}10^{-6}$ is wrongly shifted North by a few hundred meters but the remaining captured positions are very close to the simulated positions for the corresponding offset values, with an average accuracy of approximately 5m.

Confirming the results obtained in section 3.3.4, altering both coefficients simultaneously presents no immediately perceptible advantage over altering only af_0, although, as previously detailed in the same section, there may be some benefits in situations where offsets are being monitored. Since this approach allows

for similar results to altering af_0 but with smaller offsets across two different coefficients, rather than one large offset applied to a single coefficent, it may prove more difficult to detect, making this approach more robust against some spoofing detection measures.

4.2 Analysis of Results

The results in chapters 3 and 4 seem to confirm the hypothesis that altering the clock correction parameters by which GPS receivers account for the difference between their internal clock and the reported time from GPS satellites does, in effect, cause the receiver to miscalculate its position, as intended. Furthermore, from the tests conducted, it is apparent that both the distance of spoofed positions from the original position and the direction of these positions' shift are easily and predictably calculated on a case by case basis. This allows for reliable replication of the results and thus enables straightforward selection of the intended final position to be calculated by the receiver.

As discussed in section 3, different orders of magnitude for the offset values applied to each parameter will produce a different gamut of distances from the actual receiver position. These ranges will vary slightly based not only on the choice of which satellites' data to alter during the course of the simulation but also what RINEX navigation files were used to obtain the original ephemerides data, as will the direction of the shift in position from actual to spoofed. The variations in results, consequence of using different RINEX files, are due to differences in the date, geographical location and duration of the capture of GPS signals used to produce the files, which, subsequently, determines which satellites are within line of sight of the receiver and their position relative to it.

During the experimental procedures, it was determined that, in order to ensure the accuracy of the results obtained, RINEX observation files should be paired with the corresponding navigation file, regarding position, as well as date.

As far as matching the positon is concerned, it is possible to use an observation file captured in a certain location (for instance a pdel0560.19o which corresponds to data captured on the 56th day of 2019 in Ponta Delgada, Azores) paired with a brdc RINEX navigation file (such as brdc0560.19n), which essentially contains satellite ephemerides data for all satellites across the globe, since the program will check which satellites are within line of sight and use only the data related to these. Using a file concerning a different location altogether, however, will result in incorrect position calculation, since, unless the location is geographically very near the matching position for the corresponding observation file, the set of satellites described in one file will not match those observed in the other.

Attempting to use a pair of files whose dates do not match up presents a similar problem. While it may be possible that the satellite data described in each file matches up by chance, since they are from the same location, most likely the satellites over a certain area will be different depending on date and time of day, in which case the position calculations will not be correct since the data from both files does not match up. When moving from simulation to physical, real world experimentation, further considerations to this variable should be made regarding not only if both files pertain to the same date but also if said date matches as closely as possible with the actual time of the experiment. Using out of date files for the experiments proved to result in greatly diminished probability of the receivers locking on to the spoofed signal, furthermore, when lock on was achieved, the satellites detected were frequently different from those detected from genuine GPS navigation signals.

As evidenced by the results from the experiments in Chapter 4, the simulator's results are able to be replicated in real world test scenarios. Most of the positions captured during the physical test scenario experiments fall very close to the locations calculated in the simulator experiments, so the results are within the realm of outcomes expected, based on the previously obtained simulation results. Some errors were found to occur under some testing conditions, causing some positions' latitude coordinates to be miscalculated. The most probable cause for these errors

was narrowed down to precision errors in the calculation of the spoofed position during signal generation.

Even though each situation and its correspondent simulation and/or physical transmission study must be analysed on a case-by-case basis to precisely assess the impact of the alterations to the navigation messages, the experiments present consistently comparable results, suggesting that they are replicable and usable as a solution for GPS location spoofing. Moreover, the few situations which produced anomalous or invalid results in the physical testing stages were motly due to limitations of the GPS signal generation tool used and can, therefore, be mitigated by using a different tool or taking advantage of the open source nature of this software to adapt it to suit the intended purpose. In each experiment conducted with successful outcomes, it was also found that the signal generated with the spoofed data was able to deceive all the receivers used in testing.

Chapter 5

Conclusions

5.1 Summary

Having established that the results from sections 3 and 4 confirm the main hypothesis of the current work, comparisons can be made between the effectiveness of each of the polynomial coefficients altered for a given experimental procedure, and, to a certain extent, the results allow for conjecture on the detectability of each.

Of the three parameters in question from the clock correction equations 3.1 and 3.2, the drift rate correction coefficient ($\boldsymbol{af_2}$) seems to be the only one unable to produce the intended results on a regular basis, let alone with guaranteed efficacy as is the case for the clock bias and clock drift correction coefficients ($\boldsymbol{af_0}$ and $\boldsymbol{af_1}$ respectively).

Comparing these two coefficients, the results unequivocally show that the clock bias correction coefficient has a much greater impact on the receiver's position while still allowing a comprehensive degree of control over how the final spoofed position intended to be calculated by the receiver.

As for the clock drift correction coefficient parameter, altering it produces similar results with greatly diminished efficiency and should, therefore, be considered

only in situations where altering $\boldsymbol{af_0}$ is not a viable option due to risk of detection or when greater granularity of control over the spoofed position is required.

It should be noted that if $(t - t_{oc})$, as described in equation 3.2, were to equal zero at any moment, the impact of altering either $\boldsymbol{af_1}$ or $\boldsymbol{af_2}$ would essentially be negated for that instant, meaning $\boldsymbol{af_0}$ is the only coefficient of the three to have a guaranteed impact on perceived time even in situations with no apparent offset between the receiver's and the satellites' clocks.

The results also indicate that very minor changes in either of these clock correction coefficients are able to produce variations in the corresponding satellite's pseudorange up to several hundred meters, which in turn translates to even greater shifts in the receiver's calculated position. This establishes that although the spoofed data is, in essence, unnoticeably different from normal variation expected in the values throughout the process of data acquisition by the receiver, the consequences of altering said data can be quite effective and produce a very wide range of spoofed positions from the actual receiver position. Thus, the proposed method for position alteration could prove to be very well suited for use in a GPS position spoofing tool.

Bearing the above conclusions in mind and, considering that the results of both simulation and experimental work seem to validate the viability of the hypothesis as a mechanism for a spoofing tool, a few conditions for the successful use of the functionalities required of such a tool can be outlined.

First and foremost, a solution based on the proposed hypothesis should presume that the spoofing device will be stationary and is intended to produce an area where invasive drones will be much more likely to lock on to a false, spoofed signal than to genuine GPS navigation signals. This will serve as a means to ensure that the GPS navigation data required to generate the spoofed signal pertains to a set, known area and is readily available as soon as a spoofing attempt is required to deter an invader.

The satellite ephemerides data required should also be captured as close to possible to the actual time used for the spoofed signals and should be transmitted in the same time intervals as genuine GPS navigation messages to ensure both lower chances of detection and increased efficacy of the solution.

In addition, the drone's position should also be used as an input parameter to ensure greater accuracy in the final spoofed position. This can be achieved using ranging detection techniques such as radar or laser-based distance measurements to determine the drone's position relative to the spoofing device's position and thus correct the expected initial location for use in the calculation of the shifted position. The importance of this step is due to the intrinsic functionality of the position calculation itself. During the course of the experiments, all analysis was conducted with the precondition that the initial, actual, target position is known, and all spoofed positions are obtained as a transformation of this initial position, as detailed throughout section 3.2.

Finally, it should be noted that the underlying mechanisms of this proposed tool are effective as long as the transmitted signal strength is enough to overpower genuine GPS signals in the intended area. This should not be hard given that GPS signals are very low power signals in actuality, yet it does mean some precautions should be taken in order to avoid self-interference from the transmitter's own spoofed signal.

As soon as the drone leaves the area of effect of the device, since normal operation of GPS geolocation relies on regular recalculation of position based on available signals, the drone will most likely be able to re-acquire a genuine signal lock and recalculate their perceived position to their actual position in relation to the satellites. Before the drones have a chance to leave the area of effect, the spoofer should aim to direct the invader to an location within this area where it might be captured or disabled, otherwise an autonomous drone or a drone operator will, most likely, be able to detect the spoofing attempt and try to compensate for it.

5.2 Future Work

Regarding future work, a number of potential improvements to the current system have been identified. These stem from either intended functionalities which were not implemented due to time constraints or conjectures elicited by preliminary results which point towards interesting new possibilities for research hinging on the results of the current hypothesis.

As far as improvements to the system are concerned, the most relevant involve automation and general streamlining of the simulation/position calculation process. The proposed improvements start with developing the simulator from scratch to increase efficiency and decrease runtimes. The current Matlab implementation is extremely useful for research purposes since all variable monitoring and graphics generation are perfectly integrated into the core functionality of the software, but this comes at a cost of requiring greater computing power overhead which could be reduced if not almost entirely eliminated.

Having a purpose built solution would allow for further modifications to the software and eliminate the need for parameter adjusting through code manipulation, a graphical interface could be developed to show the map around the intended area and with a click on the location for the intended spoofing position, the code could be rewritten to calculate pseudoranges from said position to the satellites, evaluate which satellites would produce the greatest difference when altering the parameters and determine the offset necessary to the clock correction parameters required to produce the shift from the actual position. These changes would, essentially, remove any sort of guesswork from the workflow and make it so that the simulation has to be run only once to produce an accurate, altered navigation file to be used for spoofed signal generation.

Still on the subject of building the software from the ground up, the signal generator should also be constructed with the specified purpose in mind, during the course of the research, adaptations were made to the open source generator

used but the efficiency of such modifications is below what could be achieved with a custom solution.

Although some of the process required to build this custom solution has been started in the present work, in the form of the position calculator algorithm conversion to C and its subsequent implementation into the source code of GPS-SDR SIM (as detailed in chapter 4.1), further modifications are required to the signal generator in order to streamline the process and there is still a great number of Matlab scripts which must be converted to C in order to remake the simulator. Adding this to the changes proposed above means that building the proposed tool will take more time than initially anticipated. As mentioned, the Matlab implementation is still ideal for research but for practical application of the explored concepts as a solution, the tool should be custom built for the task at hand.

As for further research, three topics stood out as interesting possibilities: exploration of the hypothesis as a means to spoof GPS time for applications which rely on GPS to set device times or otherwise synchronize clocks; application of machine learning algorithms to parse location data and determine the most efficient and least detectable alterations to the clock correction coefficients able to produce equivalent results to the manually selected ones; and, finally, exploring the viability of applying a similar sort of time correction spoofing mechanism to other GNSS systems as a way to circumvent anti-spoofing measures relying on shifting to other systems once GPS is found to have been compromised.

Appendices

Appendix A

Matlab Algorithms

A.1 Least Squares Position Calculation

```
function [PositionCart,PositionECEF, clkOffset] = calcRecPos(P,prns,time,Eph)
%calcRecPos Least-squares searching for receiver position.
%      Given 4 or more pseudoranges and ephemerides.
%       Adapted from Kai Borre's code, returns position in ECEF and LLH
%       coordinates

vlight = 299792458;           % vacuum speed of light in m/s
Omegae_dot = 7.292115147e-5;  % rotation rate of the earth in rad/s
dtr = pi/180;
P
[m,n] = size(P);
sv = prns;

for t = 1:m

   icol = find_eph(Eph,sv(t),time);
%    if icol == 0, continue;
```

```
%    end
   tx_RAW = time-P(t,1)/vlight;



   TOC = Eph(21,icol);
   dt = check_t(tx_RAW-TOC);
   tcorr(t) = (Eph(2,icol)*dt + Eph(20,icol))*dt + Eph(19,icol)%%;
   tx_GPS = tx_RAW-tcorr(t);
   XS(:,t) = satpos(tx_GPS, Eph(:,icol));
   [phi(t),lambda(t),h(t)] = togeod(6378137,298.257223563,...
                                          XS(1,t),XS(2,t),XS(3,t));
end

clkOffset = tcorr;
% close all

% Satellite positions are now known in the ECEF system in form of
% Cartesian (X,Y,Z) and geographical (phi,lambda).
% First guess for receiver's position is (phi_old,lambda_old)
phi;
lambda;
phi_old = mean(phi) ;
lambda_old = mean(lambda) ;
scale = 900;
acc_p = [];
acc_l = [];

Old_Sum = 10^20;
tic

phi_old = 0;
```

```
lambda_old = 0;


for iter = 1:8  % You may find a nicer upper bound for "iter"
   scale = scale/10;
   sin_phi0 = sin(phi_old*dtr);
   cos_phi0 = cos(phi_old*dtr);
   ndiv = 8;
   for b = 0:ndiv
      psi = b*scale/ndiv;    % distance
      sin_psi = sin(psi*dtr);
      cos_psi = cos(psi*dtr);
      for alpha = 0:20:340    % azimuth
         sin_phi2 = sin_phi0*cos_psi...
                              +cos_phi0*sin_psi*cos(alpha*dtr);
         phi2 = asin(sin_phi2)/dtr;
         if cos(phi2) == 0
            sin_dlambda = 0;
         else
            sin_dlambda = sin(alpha*dtr)*sin_psi/cos(phi2*dtr);
         end;
         dlambda = asin(sin_dlambda)/dtr;
         lambda2 = lambda_old + dlambda;
         [XR(1,1),XR(2,1),XR(3,1)] = frgeod(6378137,...
                              298.257223563, phi2, lambda2, 0);
         for t = 1:m
            cal_one_way(1,t) = norm(XS(:,t)-XR);
            sat_clock(t) = tcorr(t)*vlight;
            omegatau = Omegae_dot*cal_one_way(1,t)/vlight;
            R3 = [ cos(omegatau) sin(omegatau) 0;
                  -sin(omegatau) cos(omegatau) 0;
                           0      0      1];
```

```
            X_ECF(:,t) = R3*XS(:,t);
            cal_one_way(1,t) = norm(X_ECF(:,t)- XR);
            one_way_res(1,t) = ...
                        P(t,1)-cal_one_way(1,t)+sat_clock(t);
         end;
         resid_t = one_way_res(1,:)-one_way_res(1,1);
         New_Sum = resid_t*resid_t';
         b=resid_t';
         if New_Sum < Old_Sum
            Old_Sum = New_Sum;
            phi_save = phi2;
            lambda_save = lambda2;
            acc_p = [acc_p phi2];
            acc_l = [acc_l lambda2];
         end;
      end %alpha
   end %b
   fprintf('Sum of residuals squared: %6g\n', Old_Sum);
   phi_old = phi_save
   lambda_old = lambda_save
end %iter

fprintf('Final value for latitude  %10.6f\n', phi_old);
fprintf('Final value for longitude %9.6f\n', lambda_old);



%%Conversion to cartesian coord
% [X,Y,Z] = geodetic2ecef(phi_old,lambda_old,48.4,wgs84Ellipsoid('kilometer'));
% PositionCart= lla2ecef([phi_old lambda_old 48.4]);
PositionCart=XR;
PositionECEF=[phi_old lambda_old];
```

```
X=XR(1)*.001;
Y=XR(2)*.001;
Z=XR(3)*.001;

fprintf('Cartesian Coordinates are:\n');
fprintf('X= %f\n', X);
fprintf('Y= %f\n', Y);
fprintf('Z= %f\n', Z);
%%%%%%% end recpos.m %%%%%%%%%%%%%%%%%%%%
```

A.2 RINEX Navigation and Observation File Parser

```
%%Read RINEX Navigation file
rinexe('pdel1470.19n','eph.dat');

Eph = get_eph('eph.dat');

obsFile='pdel1470_alt.19o';
file = fopen(obsFile, 'rt');

% Parse Header for Observation types and antenna delta

eof = 0;
ifound_types = 0;
Obs_types = [];
ant_delta = [];

time = 0;
dt = 0;
```

```
sats = [];
NoSv = 0;
eof = 0;

while 1
    line = fgetl(file);
    %     Find End of HEADER
    ans = findstr(line, 'END OF HEADER');
    if ~isempty(ans), break;end;
    if (line == -1), eof = 1; break; end;
    ans = findstr(line,'ANTENNA: DELTA H/E/N');
    if ~isempty(ans)
        for i=1:3
            [delta, line] = strtok(line);
            del = str2num(delta);
            ant_delta = [ant_delta del];
        end;
    end
    answer = findstr(line,'# / TYPES OF OBSERV');
    if ~isempty(answer)
        [NObs, line] = strtok(line);
        NoObs = str2num(NObs);
        for k = 1:NoObs
            [ot, line] = strtok(line);
            Obs_types = [Obs_types ot];
        end;
        ifound_types = 1;
    end;
end;

% eof
```

```
% ifound_types
% Obs_types
% ant_delta

% Parse epochs with flag 0 to get time and number of SVs
while 1
    line = fgets(file);

    % Skip comment lines
    ans = findstr(line, 'COMMENT');
    if ~isempty(ans);
        line = fgetl(file);
    end;

    % Check if end of file has been reached
    if (feof(file) == 1);
        eof = 1;
        break
    end;

    % If epoch has flag different from 0, break
    if ((strcmp(line(29),'0') == 0) & (size(deblank(line),2) == 29))
        eof = 1;
        break
    end;


    % Grab data from each epoch with flag 0
    if ((strcmp(line(2),'0') == 1)||((strcmp(line(2),'1') == 1))  &...
    (strcmp(line(29),'0') == 1))
```

```
ll = length(line)-2;
if ll > 60, ll = 60; end;
linp = line(1:ll);
%fprintf('%60s\n',linp);
[year, line] = strtok(line);
year
[month, line] = strtok(line);
month
[day, line] = strtok(line);
day
[hour, line] = strtok(line);
hour
[minute, line] = strtok(line);
minute
[second, line] = strtok(line);
second
[OK_flag, line] = strtok(line);
h = str2num(hour)+str2num(minute)/60+str2num(second)/3600;
jd = julday(str2num(year)+2000, str2num(month), str2num(day), h);
[week, sec_of_week] = gps_time(jd);
%jd;
time = sec_of_week;
[NoSv, line] = strtok(line,'G');

for k = 1:str2num(NoSv)

    sat=[];
    if contains(line, 'G')
        [sat, line] = strtok(line,'G');
    end
```

```
            if isempty(sat)==0
                prn(k) = str2num(sat)
            end

            %Check to see if any sat numbers are in next line
            if (isempty(sat)) && (str2num(NoSv)>12)
                line = deblank(fgetl(file))
                lineF = deblank(line(end:-1:1));
                line = lineF(end:-1:1);
                if contains(line, 'G')
                    [sat, line] = strtok(line,'G');
                    if contains(sat, 'R')
                        sat = strtok(sat,'R');
                    end
                end
            end
            if isempty(sat)==0
                prn(k) = str2num(sat)
            end
        end

%         prn(end)=[];
        sats = prn(:);
        dT = strtok(line);
        if isempty(dT) == 0
            dt = str2num(dT);
        end
        break
    end

end;
```

```
datee=[str2num(year) str2num(month) str2num(day) str2num(hour)...
str2num(minute) str2num(second)]

NoSv = size(sats,1);
NoObs = size(Obs_types,2)/2

% Read observations of all NoSV

Obs = zeros(NoSv, NoObs);

if NoObs <= 5   % This will typical be Turbo SII data
    for u = 1:NoSv
        lin = fgetl(file);
        for k = 1:NoObs
            Obs(u,k) = str2num(lin(2+16*(k-1):16*k-2));
        end
    end
else         % This will typical be Z12 data
    Obs = Obs(:,[1 2 3 4 5]); % We cancel the last two columns 6 and 7
    NoObs = 5;
    for u = 1:NoSv
        lin = fgetl(file);
        lin_doppler = fgetl(file);
        for k = 1:NoObs  %%-1
            if size(lin,2)>=78 && isempty(str2num(lin(1+16*(k-1):16*k-2))) == 1,...
            Obs(u,k) = nan;

            else                                %
                if size(lin,2) < 78, lin = fgetl(file);
                else
                    Obs(u,k) = str2num(lin(1+16*(k-1):16*k-2));
```

```
                end
            end
            % Obs(u,NoObs) = str2num(lin(65:78));
        end
    end
end

Obs;

%NoSvs = size(sats,1)

fclose(file);
```

A.3 Ephemerides data alteration in the data structure used during simulation

```
function [Eph] = alterEph(ephemData, offset0, offset1, index);

ephemerisData = ephemData;
svprn = ephemerisData(1,:);
af0 = ephemerisData(19,:);
af1 = ephemerisData(20,:);

ephemerisData(19,index) = af0(index)+offset0;
ephemerisData(20,index) = af1(index)+offset1;

Eph=ephemerisData;
```

A.4 Haversines formula calculation of Great Circle distance between two sets of coordinates

```
%Haversine Formula for great circle distance calculation between two coordinates
%Distance returned in m

function [distance]=distanceCalc(coordA, coordB)

%Radius of the Earth
R = 6371E3;

phi_1 = deg2rad(coordA(1));
phi_2 = deg2rad(coordB(1));
deltaPhi = deg2rad(coordB(1)-coordA(1));
deltaLambda = deg2rad(coordB(2)-coordA(2));

%square of half the chord length between the points
a = sin(deltaPhi/2)^2 + cos(phi_1) * cos(phi_2) * sin(deltaLambda/2)^2;

%angular distance in radians
c = 2 * asin(min(1,sqrt(a)));

%Distance in m
distance=R*c;
```

Appendix B

C Algorithms

B.1 Least Squares Position Calculation

```
void calcRecPos(double *pseudoranges, double *pos, double *vel, double *clk,
double *recPosXYZ){

double tcorr[MAX_SAT];
double sat_clock[MAX_SAT];
double tx_GPS=0;

double recPosLLH[3]={0,0,0};

//SV within LoS HAVE to be rewritten for every RINEX file,
// cross-reference with Matlab Simulation

int satList[9] = {1,4,5,6,8,16,18,22,29};

for(int j = 0; j < 9; j++){
int i = satList[j];
```

```
double tx_RAW=eph[0][i].toc.sec - pseudoranges[j]/vlight;
double dt = tx_RAW-eph[0][i].toc.sec; //check_t removido
tcorr[j]=((eph[0][i].af2*dt + eph[0][i].af1)*dt + eph[0][i].af0);
tx_GPS = tx_RAW-tcorr[i];
}

double XS[MAX_SAT][3];

for(int j = 0; j<numsats; j++){
int i =satList[j];
//  grx = incGpsTime(eph[0][i].toc, 0.0);
grx.week=0;
grx.sec=tx_GPS;

satpos(eph[0][i], grx, pos, vel, clk);
// printf("Sat %d Position=          %f, %f, %f \n", i, pos[0], pos[1],pos[2]);
XS[j][0]=pos[0];
XS[j][1]=pos[1];
XS[j][2]=pos[2];
// printf("Sat %d Position in Array= %f, %f, %f \n", i, XS[i][0], XS[i][1],XS[i][2]);
}

long double resid_t[MAX_SAT];

long double phi2=0;
long double lambda2=0;
long double phi_save =0;
long double lambda_save =0;
// double acc_p=0;
// double acc_l=0;
long double old_sum = pow(10,20);
```

```
long double one_way_res[MAX_SAT];
long double X_ECF[MAX_SAT][3];
long double cal_one_way[32];
double XR[3] ={0,0,0};
long double scale = 900;
dtr = M_PI/180;

double phi_old=0;
double lambda_old=0;

for (int iter=0; iter<9; iter++){
scale = scale/10;
long double sin_phi0=sin(phi_old*dtr);
long double cos_phi0=cos(phi_old*dtr);
long double ndiv=8;

for (int b = 0; b<=ndiv; b++){
long double psi = b*scale/ndiv; //Distance
long double sin_psi = sin(psi*dtr);
long double cos_psi = cos(psi*dtr);
for(int alpha=0; alpha<341; alpha+=20){ //azimuth
long double sin_phi2 = sin_phi0*cos_phi0+cos_phi0*sin_psi*cos(alpha*dtr);
phi2 = asin(sin_phi2)/dtr;
long double sin_dlambda;
if (cos(phi2)==0){
sin_dlambda = 0;
}else{
sin_dlambda = sin(alpha*dtr)*sin_psi/cos(phi2*dtr);
}
long double dlambda = asin(sin_dlambda)/dtr;
lambda2 = lambda_old+dlambda;
```

```
frgeod(6378137,298.257223563, phi2, lambda2, 0, XR);

for(int t=0; t<numsats; t++){
cal_one_way[t]=sqrt((XS[t][0]-XR[0])*(XS[t][0]-XR[0]) +
(XS[t][1]-XR[1])*(XS[t][1]-XR[1]) + (XS[t][2]-XR[2])*(XS[t][2]-XR[2]));
sat_clock[t] = tcorr[t]*vlight;
long double omegatau=Omegae_dot*cal_one_way[t]/vlight;
long double R3[3][3];
R3[0][0]= cos(omegatau); R3[0][1]=sin(omegatau); R3[0][2]=0;
R3[1][0]=-sin(omegatau); R3[1][1]=cos(omegatau); R3[1][2]=0;
R3[2][0]=0    ; R3[2][1]=0    ; R3[2][2]=1;
// X_ECF[t] = R3*XS[t];
X_ECF[t][0]= R3[0][0]*XS[t][0]+R3[0][1]*XS[t][1];
X_ECF[t][1]= R3[1][0]*XS[t][0]+R3[1][1]*XS[t][1];
X_ECF[t][2]= XS[t][2];

cal_one_way[t] = sqrt((X_ECF[t][0]-XR[0])*(X_ECF[t][0]-XR[0]) +
(X_ECF[t][1]-XR[1])*(X_ECF[t][1]-XR[1]) + (X_ECF[t][2]-XR[2])*(X_ECF[t][2]-XR[2]));
one_way_res[t]=pseudoranges[t]-cal_one_way[t]+sat_clock[t];
}
for(int i = 0; i<=MAX_SAT;i++){
 resid_t[i]= one_way_res[i]-one_way_res[0];
}
}

recPosLLH[0]=phi_old;
recPosLLH[1]=lambda_old;
recPosLLH[2]=0;

long double new_sum = 0;
```

```
for (int i=0; i<numsats; i++){
new_sum += resid_t[i]*resid_t[i];
}
if(new_sum<old_sum){
old_sum = new_sum;
phi_save = phi2;
lambda_save = lambda2;
}
printf("\n");
printf("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");
printf("Sum of residuals squared: %f \n", old_sum);
phi_old=phi_save;
lambda_old =lambda_save;
llh2xyz(recPosLLH,recPosXYZ);
printf("Phi= %f \n", phi_old);
printf("Lambda= %f \n", lambda_old);
printf("***************************************************************\n");
printf("LLH Coordinates= [%f,%f,%f] \n", recPosLLH[0],recPosLLH[1],
recPosLLH[2]);
printf("ECEF Coordinates= [%f,%f,%f] \n", recPosXYZ[0],recPosXYZ[1],
recPosXYZ[2]);
printf("Google Maps Paste: %f %f \n", phi_old, lambda_old);
}
}
}
```

B.2 RINEX Observation File Parser

```
    void readObsFile(double *pseudoranges){

// int numSV;
```

```
// int SV;
// double pseudos[32];
// double pseudorange;
char line[100];

// int year, month, day, hour, min;
// int sec;
// int flag;
char *satString[8];

FILE *filePointer;

if ((filePointer = fopen("pdel1470_alt.19o", "r")) == NULL) {
printf("Error! No RINEX observation file present!");
// Program exits if file pointer returns NULL.
exit(1);
}

filePointer = fopen("pdel1470_alt.19o", "r");

// while (EOF != fgets(line, 100, filePointer)) {
// while (strstr(line, "END OF HEADER") == NULL) {
// fgets(line, 100, filePointer);
// }
// printf(line);
// }
while (1) {
fgets(line, 100, filePointer);
if (strncmp(line + 60, "END OF HEADER", 13) == 0) {
break;
}
```

```
}

fgets(line, 100, filePointer);
// printf("%s", line);

char lineS[100];
strcpy(lineS, line);

char delim[] = "  ";
char *ptr = strtok(lineS, delim);

for (int i =0; i<8; i++) {
// printf("'%s'\n", ptr);
satString[i]=ptr;
ptr = strtok(NULL, delim);
}
// printf("satString= %s\n", satString[7]);

char satsList[50];

strcpy(satsList, satString[7]);
// printf("satString2= %s\n", sats);

char delimG[] = "G";
ptr = strtok(satsList, delimG);

int j=0;
numsats=atoi(ptr);

char *sats[MAX_SAT];
```

```
while (ptr){
// printf("'%s'\n", ptr);
if (j != 0){
sats[j-1] = ptr;
}else{
numsats=atoi(ptr);
}
j++;
ptr = strtok(NULL, delimG);
}
// printf("%d\n",numsats);

for(int i=0; i<numsats; i++){
fgets(line, 100, filePointer);
double buffer;
sscanf(line, "%lf", &buffer);
// printf("Buffer= %.3f\n", buffer);
pseudoranges[i]=buffer;
printf("Pseudorange %d= %.3f\n", i+1, pseudoranges[i]);
fgets(line, 100, filePointer);
// pseudoranges[i]=atof();
}
fclose(filePointer);
}
```

Appendix C

Python Algorithms

C.1 Script for RINEX Navigation File Modification

```
#!/usr/bin/env python3

import re

print("Reading polynomial coefficient files...")

f=open("af0_values.txt","r")
lines0 = f.readlines()
f.close

f=open("af1_values.txt","r")
lines1 = f.readlines()
f.close

print("All files read")
```

```
print("Reformatting notation...")


f=open("af0_values_Corrected.txt","w+")
for i in lines0:
i=re.sub('e','D',i)
i=re.sub('000000000000','0.000000000000D+00',i)
if i[0] != '-':
i= ' ' + i
#print(i)
f.write(i)
lines0 = f.readlines()
f.close
print("af0 reformated")

f=open("af1_values_Corrected.txt","w+")
for i in lines1:
i=re.sub('e','D',i)
i=re.sub('000000000000','0.000000000000D+00',i)
if i[0] != '-':
i= ' ' + i
#print(i)
f.write(i)
lines1 = f.readlines()
f.close
print("af1 reformated")

#f=open("copy_of_pdel0560.19n", "r")
f=open("pdel1470.19n", "r")
linesN = f.readlines()
f.close
```

```
#Count head lines
headlines=0
for i in linesN:
match = re.search("END OF HEADER",i)
if match:
headlines+=1
#print(headlines)
break
else:
headlines+=1
#print(headlines)

#Find all ephemerides
print("Creating new RINEX Nav file...")
step = 8
noEph = 0
line = 0

f0 = open("af0_values_Corrected.txt","r")
f1 = open("af1_values_Corrected.txt","r")

lines0 = f0.readlines()
lines1 = f1.readlines()

f0.close
f1.close

ff= open("C1_1470.19n", "w+")

for i in linesN:
```

```
#if linesN.index(i)>100:
# break
if linesN.index(i) <= (headlines-1):
ff.write(i)

if linesN.index(i)>(headlines-1):
if (linesN.index(i)+(step-headlines))%(step)==0:
s=i[:22] + lines0[line].strip('\n') + lines1[line].strip('\n') + i[60:]

#print(line)
#print(i)
#print(s)
#print("\n")

ff.write(s)

line+=1
noEph+=1

#print(noEph)
else:
ff.write(i)
ff.close

print("Done")
```

Bibliography

[1] Federal Aviation Administration, "FAA Aerospace Forecast, FY2018-38," tech. rep., Record No. 2018 ASI 7504-6, Mar. 15, 2018. [Online]. Available: https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2018-38_FAA_Aerospace_Forecast.pdf. [Accessed: 2018-11-13].

[2] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via gps spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[3] V. Dey, V. Pudi, A. Chattopadhyay, and Y. Elovici, "Security vulnerabilities of unmanned aerial vehicles and countermeasures: An experimental study," in *2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID)*, pp. 398–403, IEEE, 2018.

[4] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A software-defined GPS and Galileo receiver: a single-frequency approach*. Springer Science & Business Media, 2007.

[5] K. Wang, S. Chen, and A. Pan, "Time and position spoofing with open source projects," *Black Hat Europe*, vol. 148, 2015.

[6] H. N. Viet, K. R. Kwon, S. K. Kwon, E. J. Lee, S. H. Lee, and C. Y. Kim, "Implementation of GPS signal simulation for drone security using Matlab/Simulink," *Proceedings of the 2017 IEEE 24th International Congress on Electronics, Electrical Engineering and Computing, INTERCON 2017*, 2017.

[7] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech House, 2005.

[8] Global Positioning System Wing (GPSW) Systems Engineering & Integration, "NAVSTAR GPS Space Segment/Navigation User Segment L1C Interfaces IS-GPS-800, Revision J." 2018.

[9] N. Van Thang, C. D. Trinh, and T. D. Tan, "Application of street tracking algorithm in a feedback configuration for an integrated ins/gps navigation system," pp. 279–288, 2014.

[10] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 75–86, ACM, 2011.

[11] International Civil Aviation Organization, "Unmanned Aircraft Systems (UAS)," Tech. Rep. 1, Cir 328 AN/190, 2011. [Online]. Available: http://dx.doi.org/10.1016/B978-0-12-374518-7.00016-X. [Accessed: 2018-11-13].

[12] T. Rosa, "GPS Radio Hacking – What the Hell Time Is It?," in *IS2 - Information Security Summit*, (Prague), pp. 138–152, 2016.

[13] B. Motella, M. Pini, M. Fantino, P. Mulassano, M. Nicola, J. Fortuny-Guasch, M. Wildemeersch, and D. Symeonidis, "Performance assessment of low cost gps receivers under civilian spoofing attacks," in *2010 5th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, pp. 1–8, IEEE, 2010.

[14] C. Günther, "A Survey of Spoofing and Counter-Measures," *Navigation, Journal of the Institute of Navigation*, vol. 61, no. 3, pp. 159–177, 2014.

[15] J. R. Van Der Merwe, X. Zubizarreta, I. Lukčin, A. Rügamer, and W. Felber, "Classification of Spoofing Attack Types," *2018 European Navigation Conference, ENC 2018*, no. July, pp. 91–99, 2018.

[16] B. Hermans and L. Gommans, "Targeted GPS spoofing," *M.S. thesis, University of Amsterdam*, 2018.

[17] M. L. Psiaki and T. E. Humphreys, "Gnss spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, 2016.

[18] J. Dampf, T. Pany, W. Bär, J. Winkel, L. Mervart, J. Ávila-Rodríguez, R. Ioannides, and G. Hein, "Real world spoofing trials and mitigation," *InsideGNSS*, vol. 12, pp. 55–65, 2017.

[19] T. E. Humphreys, B. M. Ledvina, V. Tech, M. L. Psiaki, B. W. O. Hanlon, and P. M. Kintner, "Assessing the Spoofing Threat : Development of a Portable GPS Civilian Spoofer," *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008) September 16 - 19, 2008 Savannah International Convention Center Savannah, GA*, pp. 2314–2325, 2009.

[20] B. Stewart, K. Barlee, D. Atkinson, and L. Crockett, *Software Defined Radio Workflow Using MATLAB & Simulink and the RTL-SDR*. Strathclyde Academic Media, 2015.

[21] GNU Radio Foundation, "About GNU Radio." https://www.gnuradio.org/about/. [Accessed: 2018-12-27].

[22] M. Elhawary, G. Gomah, A. Zekry, and I. Hafez, "Simulation of the e1 and e6 galileo signals using simulink," *International Journal of Computer Applications*, vol. 975, p. 8887, 2014.

[23] Ettus Research, "USRP N210." https://www.ettus.com/product/details/UN210-KIT, 2019. [Accessed: 2019-09-10].

[24] Ettus research, "USRP N210." http://www.ettus.com/all-products/un210-kit/, 2019. [Accessed: 2019-09-10].

[25] W. Gurtner, L. Estey, and B. Fields, "RINEX: The Receiver Independent Exchange Format Version 2.11," Tech. Rep. June, 2012.

[26] P. Teunissen and O. Montenbruck, *Springer handbook of global navigation satellite systems*. Springer, 2017.

[27] gAGE Research group of Astronomy and GEomatics of the Technical University of Catalonia, "GPS Navigation RINEX 2.11 Format." https://gage.upc.es/sites/default/files/gLAB/HTML/GPS_Navigation_Rinex_v2.11.html, 2019. [Accessed: 2019-09-10].

[28] NASA, "NASA Daily GNSS Data Files Server." ftp://cddis.nasa.gov/gnss/data/daily/2019/. [Accessed: 2019-09-10].

[29] gAGE Research group of Astronomy and GEomatics of the Technical University of Catalonia, "Observation RINEX 2.11 Format." https://gage.upc.es/sites/default/files/gLAB/HTML/Observation_Rinex_v2.11.html, 2019. [Accessed: 2019-09-10].

[30] R. R. Hatch, "Relativity and gps," *Galilean Electrodynamics*, vol. 6, no. 3, pp. 52–57, 1995.

[31] K. Borre, "The GPS Easy Suite–Matlab code for the GPS newcomer," *GPS Solutions*, vol. 7, no. 1, pp. 47–51, 2003.

[32] K. Borre, "The EASY Suite." http://kom.aau.dk/ borre/easy/, 1997. [Accessed: 2019-02-20].

[33] M. Mosavi, S. Azarshahi, I. Emamgholipour, and A. Abedi, "Least squares techniques for gps receivers positioning filter using pseudo-range and carrier phase measurements," *Iranian Journal of Electrical and Electronic Engineering*, vol. 10, no. 1, pp. 18–26, 2014.

[34] Y. He and A. Bilgic, "Iterative least squares method for global positioning system," *Advances in Radio Science*, vol. 9, no. C. 5-2, pp. 203–208, 2011.

[35] Z. Bar-Yehuda, "plot_google_map." https://www.mathworks.com/matlabcentral/fileexchange/27627-zoharby-plot_google_map, 2019. [Accessed: 2019-02-11].

[36] T. Ebinuma, "GPS-SDR-SIM." https://github.com/osqzss/gps-sdr-sim, 2015. [Accessed: 2019-02-15].

[37] C. Veness, "Calculate distance, bearing and more between Latitude/Longitude points." https://www.movable-type.co.uk/scripts/latlong.html, 2019. [Accessed: 2019-09-11].

[38] E. W. Weisstein, ""Great Circle."." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/GreatCircle.html. [Accessed: 2019-09-11].

[39] Chartcross Limited, "Gps test." https://play.google.com/store/apps/details?id=com.chartcross.gpstest. [Accessed: 2019-09-20].

[40] R. B. Langley *et al.*, "Dilution of precision," *GPS world*, vol. 10, no. 5, pp. 52–59, 1999.