



Research Internship Report

GPS Spoofing Attacks on Drones, Detection Methods and Countermeasures

Made by

Youssef Mammou

Supervisors:

Mr. Guy Cogniat. Professor and Vice-Dean of the Faculty of Science and Engineering, University of South Brittany

Ms. Maria Real Mendez. Junior Professor Chair at University of South Brittany

Mr. Yvan Eustache. FPGA Research Engineer



Table of contents

1. Introduction	4
2. Background and Related Work	4
2.1. Global Navigation Satellite Systems (GNSS)	4
2.2. Principles of GPS Positioning	6
2.3. Threats to GNSS : Jamming – Spoofing	6
3. Internship Objectives	7
4. Materials	8
4.1. Hardware Components	8
4.2. Software Components	9
5. Performing The GPS Spoofing Attack	10
5.1. Single Position Spoofing	10
5.2. Dynamic Spoofing	12
5.3. Receiver Behavior and Signal Interpretation	13
5.4. Cold Start	14
5.5. Outdoor Testing	14
5.6. Discussions	16
6. Analytical Detection of GPS Spoofing Attacks	16
6.1. Detection Implementation Strategy	16
6.2. Real Time Kinematic Spoofing Detection	18
6.3. Detection of Spoofing Through Satellite Anomalies	20
6.4. Pseudorange Consistency Checks (<i>RAIM Outlier Detection</i>)	23
6.5. Doppler and Clock Drift Monitoring (<i>Dynamic Timing Consistency</i>)	24

7. AI-Based Spoofing Detection Implementation	26
7.1 Dataset Construction	26
7.2 Model used – XGBoost	28
7.3 Internal Insights from the Trained XGBoost Model	29
7.4 Normal Scenarios Testing	31
7.5 Challenges and limitations: Edge-Case Scenarios Evaluation	32
8. Counter-Spoofing Response Implementation	35
9. Countermeasures Against GNSS Spoofing	37
9.1 Anomaly Detection in Signals	37
9.2 Multi-Constellation Positioning	38
9.3 Multi-Frequency Receivers	39
9.4 GNSS Signal Authentication (<i>Cryptography</i>)	41
9.5 Interference Filtering Against Jamming (<i>Notch Filters</i>)	42
9.6 Directional Detection	43
9.7 Sensor Fusion	44
9.8 Receivers with Multi-Layer Protection (<i>Septentrio Mosaic</i>)	46
10. Conclusion	48
11. References	49

1. Introduction

Global Navigation Satellite Systems, nowadays referred as GNSS, including the Global Positioning System GPS, have become essential components of modern life. Critical infrastructure and many applications depend heavily on the accuracy and integrity of GNSS data. These systems provide real-time position, velocity, and time information by receiving signals from satellites orbiting the Earth.

While widely used, GNSS technologies were not originally designed with strong security features, which makes them vulnerable to certain types of signal interference and manipulation which sometimes means “attacks”!! like GPS jamming or even more GPS spoofing which is our report’s focus, to be explained later. As technology becomes more accessible, more possible does the ability to interfere with GNSS-based systems in subtle and potentially dangerous ways.

This report presents the work conducted during a research internship in Lorient, France, focused on the practical implementation and analysis of GPS spoofing and its detection on a GPS receiver of maritime drone. However, it will be broadened in certain cases to all types of drones. The internship involved building a working spoofing setup, conducting controlled attack experiments, collecting and analysing GNSS data, and evaluating various detection strategies, including real-time monitoring and AI detection approach.

The aim of this project is twofold: to demonstrate how GPS spoofing can be executed in practice using affordable tools, and to explore methods that could help detect and mitigate such threats. The report is written to be accessible to both technical and non-technical readers, with concepts explained clearly and gradually. References are introduced by clickable links in the form of [x] throughout the document.

2. Background and Related Work

2.1. Global Navigation Satellite Systems (GNSS)

Global Navigation Satellite Systems (GNSS) are space based systems that provide users on Earth with essential information about their position, speed, and time. These systems work by using a network of satellites that continuously broadcast radio signals from orbit. Receivers on the ground pick up these signals and use them to calculate where they are and what time it is.

There are several GNSS constellations currently in operation as we speak. The most well-known is GPS “Global Positioning System”, developed and maintained by the United States. Other systems include Russia’s GLONASS, the European Union’s Galileo, and China’s BeiDou. Although these systems are managed by different countries, they operate on similar principles and often complement one another as we will discuss later on.

Most of GNSS satellites, but not all, orbit the Earth at Medium Earth orbit “MEO”, typically around 20,000 km altitude, and travel at an average speed of approximately 14,000 km/h which means 3,889 km/s. These signals are transmitted on specific frequency bands. The most commonly used for civilian GPS is the L1 band at 1575.42 MHz, which carries the Coarse/Acquisition C/A code *. [\[1\]](#)

* C/A code helps receivers identify satellites and measure the time it takes for the signal to arrive, which is essential for calculating position. It is the civilian-accessible code, unlike the encrypted military P(Y) code.

Other bands like L2 at 1227.60 MHz and L5 at 1176.45 MHz are also used, especially for professional or dual-frequency receivers and many of other frequencies are reserved to military uses. Check this link for more frequencies => [\[2\]](#)

Examples of most common and used GNSS (2025):

GNSS system	Number of satellites	Used frequency band terminology and corresponding carrier frequency (MHz)
GPS (US)	31 satellites	L1 : 1575.42 L2 : 1227.60 L5 : 1176.45
Galileo (EU)	27 satellites	E1 : 1575.42 E5a : 1176.45 E5b : 1207.14 E6 : 1278.75
Glonass (Russia)	24 satellites (out of 26, 24 operational)	G1 : 1602 G2 : 1246
BeiDou (China)	35 satellites	B1I : 1561.098 B2I : 1207.14 B3I : 1268.52

Table 1: Overview of Major GNSS Constellations and Frequencies (2025)

The accuracy of GNSS for civilian users is typically around 3 to 10 meters in open-sky conditions. However, with advanced techniques like Real-Time Kinematic “RTK” * positioning, this accuracy can be improved to just a few centimeters, which is essential in many cases that need more precision and can’t abide by normal use like for some autonomous systems.

We should know that GNSS signals are extremely weak when they arrive at the Earth’s surface! Typically, only between –130 dBm and –160 dBm; which is much lower than many other radio signals like those used by mobile phones.

This low power makes GNSS vulnerable to interferences, including **jamming** and **spoofing** explained later in details.

This was a summary about the least of what we should know about GNSS, let’s proceed now to learn how they really work and how the receiver fixes the position.

* RTK (Real-Time Kinematic) enhances GNSS accuracy by using carrier-phase measurements and real-time corrections from a nearby base station. It resolves integer ambiguities between satellite and receiver signals to achieve **centimeter-level** positioning.

2.2. Principles of GPS Positioning

The Global Positioning System GPS, part of the broader GNSS family, determines a receiver's location using a method called *Trilateration*. In simple terms, GPS satellites orbiting the Earth each transmit a signal that includes their exact position and the time the signal was sent. When a GPS receiver picks up these signals, it calculates how long they took to arrive. The receiver then can estimate its distance from each satellite based on the time delay. [\[3 \]](#)

By calculating the distance from at least four satellites, the receiver can determine its three dimensional position; latitude, longitude, and altitude (3 Satellites), as well as the precise time (the fourth satellite). This process is possible thanks to the constant movement of satellites and the use of synchronized atomic clocks onboard each one. The signal carries the basic navigation message and a pseudorandom code that helps identify which satellite the signal is coming from.

2.3. Threats to GNSS: Jamming - Spoofing

As mentioned before, due to the distance to the satellites and the noise added to the signal, GNSS signals are so weak when they arrive at ground level that they are vulnerable to interference. Two of the most common threats are **jamming** and **spoofing**. While both involve radio signals that disrupt the normal functioning of a GPS receiver, they operate in very different ways.

Jamming is the simpler of the two threats. It works by sending out a strong noise signal on the same frequency as GPS, which drowns out the real satellite signals. When this happens, the receiver is unable to detect or lock onto any satellites, which usually results in a loss of positioning service. Jamming is relatively easy to detect because the GPS device will simply stop working. Although disruptive, jamming does not try to deceive the user, it only blocks access to the real GPS data.

Spoofing can be done in several ways. One technique is **meaconing**, where real satellite signals are recorded and replayed with a delay. Another method, known as **signal synthesis**, uses software and radio hardware to generate entirely fake signals. These synthetic signals can simulate fake positions, speeds, or even entire routes, deceiving the receiver into following a false trajectory. Because spoofing can happen without warning and without obvious errors, it is considered one of the most dangerous threats to GNSS.

The consequences of spoofing can range from mild to severe. It can affect civilian navigation apps, and mislead autonomous drones. Given how much modern technology relies on GPS data, spoofing is a very serious issue. Let's take an example to understand further:

Scenario: Let's imagine the attacker begins by broadcasting fake GPS signals. The drone's receiver locks onto these counterfeit signals, unaware of the deception. The attacker then gradually alters the signal to increase the manipulated satellite distance, making the drone believe it has drifted off its intended course! In response, the autonomous drone's flight controller calculates a correction to reach its wanted final destination. It then adjusts its heading and speed to return to what it thinks is the correct waypoint. However, that waypoint is now misaligned due to the spoofed position.

As a result, the drone physically turns and flies in the wrong direction, convinced it is self-correcting. The attacker continues to manipulate the fake position data in small increments, slowly redirecting the drone toward a chosen location, all while maintaining a valid-looking GPS fix. In figure 1, one of the effects of GPS signal compromise is presented.

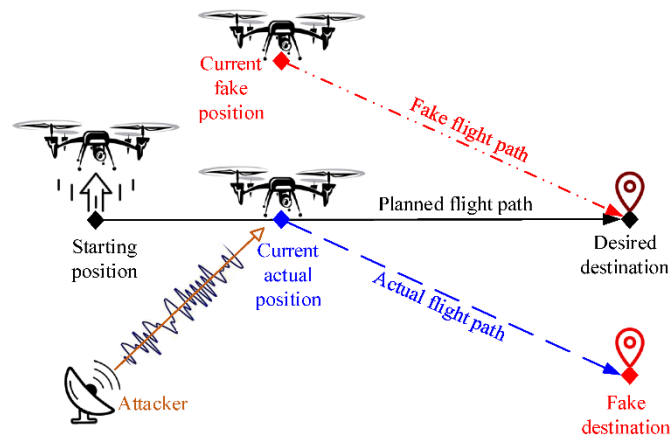


Figure 1: Illustration of a GPS spoofing attack on a drone

3. Objectives of the Internship

The main objective of this internship is to execute a spoofing attack on a maritime drone's GPS receiver and to develop methods for detecting such attack in real time and studying some existent solutions to counter these attacks. The project focuses on building a complete testing and analysis environment to simulate GPS spoofing scenarios, observe their effects on GNSS receivers. A key goal is to make the entire process, from spoofing to detection, observable, and understandable.

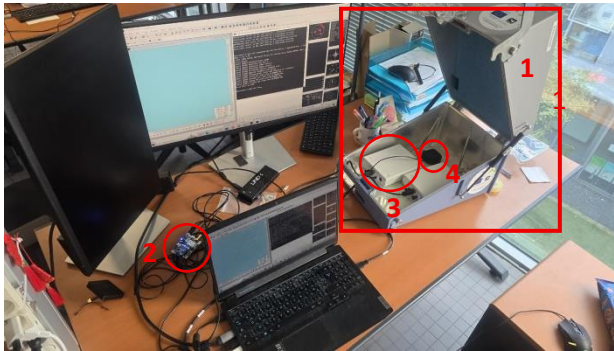
To achieve this, the work began with setting up the necessary hardware and software tools. A software-defined radio "USRP" was used to transmit spoofed GPS signals, generated using simulation executable tools such as "gps-sdr-sim". The target of the spoofing was a commercial GNSS receiver, "u-blox" module, used for drones, and its behavior was studied with a specific interface "U-center" to observe GPS information in real time.

A major part of the internship focused on developing detection methods. Two main approaches were explored: first, rule-based methods that looked for anomalies in speed, position jumps, signal strength, or satellite visibility; and second, a machine learning model that could classify spoofed versus normal conditions based on patterns learned from the collected data.

Finally, we will also study different countermeasures used to protect GPS systems from spoofing. The goal is to understand how these techniques work, how they detect or block fake signals, and how effective they are in real situations.

4. Materials

4.1. Hardware Components



1. Faraday Cage
2. U-blox GNSS Receiver
3. USRP
4. GPS Antenna

Figure 2: Experimental setup of the GPS spoofing test environment

Faraday Cage:

The main reason we use a Faraday cage is to stop our spoofed GPS signals from escaping the test area. If these fake signals go outside the room, they can be picked up by real GPS devices nearby. This can cause those devices to show the wrong location or lose their GPS fix, which is both dangerous and illegal!

The Faraday cage blocks radio signals from going out or coming in. This way, we can run controlled experiments and several tests where the GPS receiver only receives the signals we send, and we don't risk disturbing anything outside. It's an essential safety measure in any spoofing setup if it is done remotely by antennas. One other way could have been to experiment with cables which is extremely safe. But, in that case, be sure to add attenuators to avoid damaging the receiver or overloading its input with strong RF signals!

U-blox GNSS Receiver:

The u-blox module serves as the target GNSS receiver in the spoofing experiments used on many drones. It is a commercially available, multi-constellation-capable receiver that outputs detailed GNSS information such as latitude, longitude, altitude, number of visible satellites PRN, signal strength SNR (C/N0) *: as the most useful information! There are many other types of messages.

USRP: Universal Software Radio Peripheral:

It is a flexible software-defined radio "SDR" device used in this project to transmit spoofed GPS signals. It acts as the physical emitter that broadcasts the simulated GNSS signals generated by the software part. The model used supports operation in the L1 band 1575.42 MHz. By adjusting the transmit gain, sample rate, and central frequency, the USRP was able to create spoofing signals.

Since the Faraday Cage also blocks signals coming from outside, we use two USRPs, one for simulating real GPS signals (which are still spoofed, but represent the receiver's initial trusted position) and another one to simulate the spoofing signals with the second position.

* C/N₀ (Carrier-to-Noise Density Ratio) measures the strength of a satellite signal compared to background noise, expressed in dB-Hz. It indicates how clearly the GPS receiver can "hear" the signal. Higher and moderate values mean better signal quality

GPS Antenna:

Two types of antennas were used during testing: GPS one connected to the u-blox receiver to capture incoming GPS signals, and another simple one connected to the USRP to emit spoofed signals.

4.2. Software Components

gps-sdr-sim:

This tool is the most important software component: it serves as the core GPS signal generator in the project. It produces simulated GPS signals (IQ data *) based on a predefined single position or a whole spoofed trajectory. It is important to know that it only uses GPS satellites! It doesn't take into consideration other types of satellites like Galileo and BeiDou

(More advanced tools can spoof all types of GNSS: Skydel GSG-8 Advanced GNSS Simulator/ Safran BroadSim Genesis / Spirent, but these tools are commercial and not open-source tools)

GNURadio (as backend via Miniconda):

Although initially tested through its graphical interface, GNU Radio was chosen to act only as a backend library in the final setup. It is installed via Miniconda and provides support for radio communication, allowing gps-sdr-sim to interface with the USRP through UHD. No flowgraphs or visual tools are used explicitly, they are programmed in the python file.

Python Scripts:

Python plays a central role in this project. It handles:

- Sending the implemented signal to the USRP for emission
- Live data collection from the u-blox receiver using pyserial and pyubx2,
- Parsing and formatting of GNSS data (position, speed, PRNs, C/N₀, etc.),
- Rule-based spoofing detection, identifying abnormal jumps or suspicious values,
- Dataset creation and labelling based on collected GNSS conditions,
- and AI-related tasks, where Python supports early testing of models that learn to distinguish spoofed signals from normal ones.

U-Center:

U-Center is the official software interface for monitoring u-blox receivers. It provides real-time visual feedback on satellite visibility, position, signal strength (C/N₀), and fix status. It also provides an API map for showing the position.

* IQ data refers to In-phase (I) and Quadrature (Q) components of a radio signal. It represents the signal's amplitude and phase information in digital form, allowing precise reconstruction and modulation of complex waveforms like GPS signals.

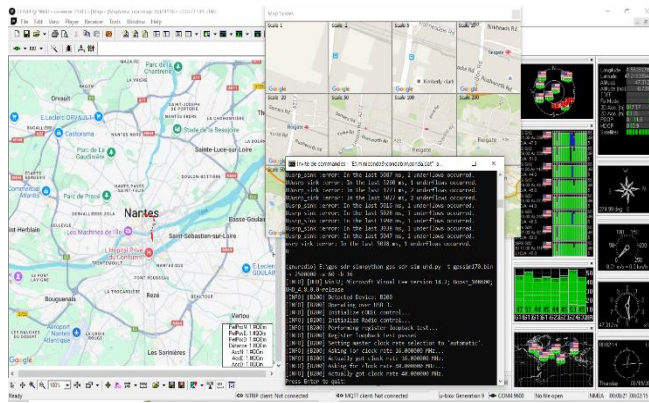


Figure 3: U-center 1 software interface

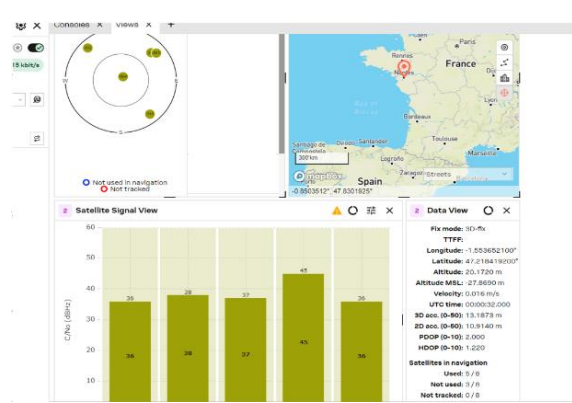


Figure 4: U-center 2 software interface

5. Performing the GPS Spoofing Attack

5.1. Single Position Spoofing

Concept: When the spoofed signal is transmitted, the GPS receiver perceives it as a real signal one: The signal should contain valid satellite IDs, timing information, and positioning data. What the receiver actually does is measure the time it takes for each satellite signal to arrive, which it then converts into distance estimates. These distances are used to compute the position by intersecting spheres in space, the method we called trilateration. So, one idea is that the spoofed signal is carefully designed so that the satellite positions and the transmission times are shifted in a way that makes the receiver believe it is farther from or closer to each satellite than it really is. By adjusting the timing of each satellite's signal, the attacker manipulates the receiver's perceived distance to each satellite, the receiver then computes a position that is consistent with those fake distances, which leads to a completely false location. This is completely possible using the GPS-SDR-SIM tool which helps in generation this signal from a set of satellites and a chosen position.

These are the two essential commands to execute the spoofing. They should be executed after the Gnuradio environment has already been activated.

```
gps-sdr-sim -e brdc1680.25n -l 47.2184,-1.5536,30 -d 180 -s 2500000 -b 16 -o gpssim.bin
```

```
python gps-sdr-sim-uhd.py -t gpssim170.bin -s 2500000 -x 50 -b 16
```

The first command serves to create the binary file gpssim.bin which contains the signal in IQ data. We start by generating this fake GPS signal based on real satellite data (called ephemeris data) contained in the RINEX file brdcXXX0.XXn. The first step is to download this file. This information allows us to build a realistic looking signal that a GPS receiver will accept as valid. The RINEX file for each day is available for free on NASA's CDDIS archive [here](#) [4] after signing in for free.

You simply choose the correct year, day of the year (like 168 for June 16), and download the file from the XXn folder (25n for the year 2025). The file is compressed so we extract it and get a file like brdc1680.25n.

Once we have the RINEX file, gps-sdr-sim tool generates the GPS signal using the first command, this tool only uses GPS satellites that starts with “G” in the RINEX file:

- -e brdc1680.25n: use the RINEX navigation file we downloaded.
- -l 47.2184,-1.5536,30: latitude, longitude, altitude of the spoofed location (Nantes in this case).
- -d 180: duration of the spoofing in seconds.
- -s 2500000: sampling rate, set to 2.5 MSPS (standard for USRP).
- -b 16: output with 16-bit resolution for better quality.
- -o gpssim.bin: name of the binary file created.

To actually broadcast this spoofed signal, we can either use the gnuradio-compagnion tool or, as done in this case, we can use a Python script called gps-sdr-sim-uhd.py (second command), which interfaces directly with our USRP. This script takes the binary file and streams it in real time at the correct sampling rate and frequency. We specify the signal file, set the sample rate to match what we used in generation, and choose a transmit gain (-x ..): typically between 40 and 60 dB with the USRP 2900 model. Once the script starts, the USRP begins broadcasting the GPS signal at 1575.42 MHz. If the u-blox receiver is nearby and isolated from real satellite signals, it will lock onto the fake signal and display the spoofed position

The spoofed location detected by u-blox is shown in U-Center in figures 3 and 4. As illustrated in the following screenshot (figure 5), the same spoofed position was also detected by my phone using Google Maps.

We will analyze the receiver’s behavior in more detail in a later section.

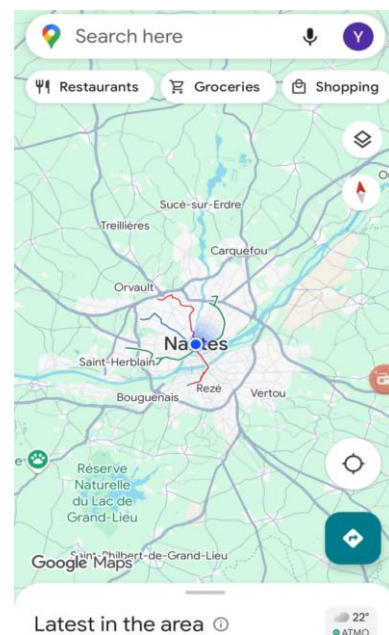


Figure 5: Spoofed position displayed on Google Maps.

5.3. Receiver Behaviour and Signal Interpretation

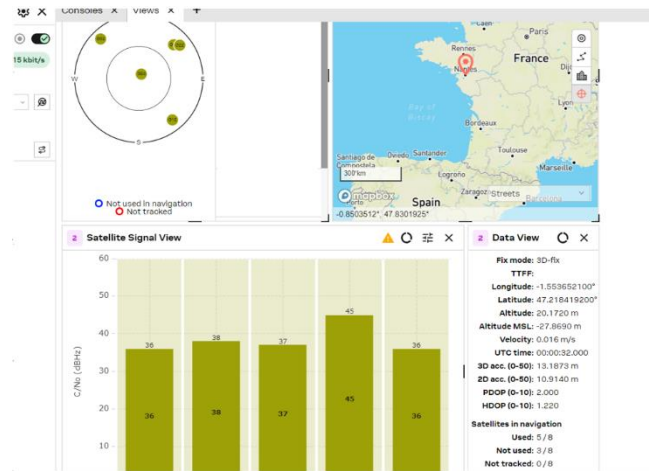


Figure 4: U-center 2 software interface

Before conducting the spoofing experiment, we manually configured the u-blox to accept only GPS signals by disabling all other constellations (Galileo, GLONASS, BeiDou, etc.) in U-center. So, now, the spoofed GPS signal is the sole source of positioning information, making the test consistent. Our spoofing system only transmits signals in the L1 band for GPS, so allowing other constellations would mean the receiver continues to receive genuine signals from those in open space, which would interfere with or weaken the spoofing effect. For more advanced GNSS receivers, if we allowed multiple GNSS systems, the receiver could cross-check the satellite data across different constellations in open air, which might lead to inconsistencies, loss of fix, or even automatic rejection of the spoofed data (To be explained in countermeasures part).

After launching the spoofed signal using our setup, we observed the spoofed location and signal characteristics in U-center. As shown in figure 4, the receiver correctly locks onto the spoofed position, showing coordinates longitude = -1.55368, latitude = 47.21819200, which is in Nantes France, matching the coordinates defined in the spoofing command. This confirms that the spoofing attack is successful and that the receiver is fully deceived.

In the Satellite View, we can see that the receiver tracks 8 GPS satellites but only uses 5 of them, with SNR (C/N_0) values ranging from 36 dBHz to 45 dBHz. To better understand this, we use the standard formula:

$$C \text{ (dBm)} = C/N_0 \text{ (dB-Hz)} + N_0 \text{ (dBm/Hz)}$$

Here, C is the carrier power, C/N_0 is the carrier-to-noise density ratio shown by the receiver (for example: 40 dB-Hz), and N_0 is the noise power spectral density, which is calculated around -174 dBm/Hz at room temperature.

For example, if we observe a C/N_0 value of 40 dB-Hz, then:

$$C = 40 + (-174) = -134 \text{ dBm}$$

As mentioned earlier, real GPS signals received at ground level are extremely weak, typically ranging from -160 dBm to -125 dBm. This falls well within the expected range of real GPS signal levels, confirming that our observations and calculations are consistent with known theoretical values. [5]

These signal strengths are consistent with a good quality signal, as expected from a nearby spoofing emitter inside a Faraday cage. The system shows a valid 3D fix, with PDOP = 1.09 and HDOP = 0.89 *, indicating a stable and accurate (but false) position solution.

All status indicators are normal: the fix is 3D, velocity is zero (consistent with a static spoofed signal), and 5 satellites are used in the navigation solution. This visual and numerical feedback in U-center validates the success of our spoofing setup and reinforces the need for advanced spoofing detection techniques, which are explored in the following sections.

5.4. Cold Start

When a GNSS receiver is powered on after being fully reset or unused for a long time, it execute what's called a cold start. In this mode, the receiver has no prior information about satellite positions, timing, or its own location. As a result, it must search blindly for satellite signals, decode their data (ephemerides), and compute a fix from scratch, which can take several minutes. In contrast, a receiver that has already acquired satellite data in recent use retains this information in short-term memory (also known as warm or hot start), allowing it to reconnect to satellites much faster and more reliably to those same satellites and position. Changing satellites (PRNs) or position brutally requires a longer wait for the receiver to stabilize on them. Cold start makes the receiver more vulnerable to spoofing, because it trusts the first valid-looking signals it receives; potentially locking immediately onto a spoofed position without detecting any inconsistencies.

5.5. Outdoor Testing

After confirming in a Faraday cage that our spoofed GPS signals had power levels comparable to real GPS signals (typically between -160 dBm and -125 dBm), and were therefore under control, we proceeded with a series of outdoor tests. These experiments were carried out with safety precautions, including real time signal monitoring to ensure that the spoofed signal remained confined to a very limited area. We always start with very low transmission gains (for example, 0 dB without cold start, or 15 dB with cold start), gradually increasing the gain depending on the observed results.

The range and effectiveness of the spoofing attack depend directly on the transmission gain configured on the USRP as well as the distance between the USRP and the u-blox receiver. It is important to note that the useful gain range is not a fixed value; it varies depending on the specific USRP model, antenna quality, surrounding environment, and device positioning. In our case, we used the USRP 2900 and tested various gain levels ranging from 0 to 60 dB at distances from 0 meters (direct proximity) up to 4 meters.

We observed two distinct behaviors:

- **Without restarting the receiver (already has a fix):** When the receiver already has a valid GPS fix (on Lorient), introducing the spoofing signal first causes a visible jamming phase, where the fix is lost or destabilized, followed by a switch to the spoofed location (Nantes). This transition can take several seconds to over a minute, depending on gain level, distance, and generated signal stability.
 - At **0 meters**, spoofing starts to succeed around 25 dB, typically preceded by a jamming effect before the position switches.
 - At **0.5 meters**, no fix jamming starts at gain 40 and spoofing becomes visible from 50 dB, often long jamming and then with a transition to the spoofed location.
 - At **1 meter to 3 meters**, a gain of at least 55-60 dB is required to force a position change after a long jamming.
- **With cold start (receiver powered on under spoofing):** In this scenario, the receiver is powered on while already under the influence of the spoofed signal. It immediately obtains a 3D fix on the spoofed location, without any jamming phase or attempt to lock onto the real position. This is particularly dangerous because no anomaly is apparent to the user. For example, even with a very low gain of just 5 or 10 dB at 10 centimeters, the receiver instantly displays the spoofed coordinates for Nantes.
 - At **0.5 to 2 meters**, even low gains between 5 and 25 dB are sufficient to immediately spoof the receiver. This demonstrates that an attacker in close proximity does not need much power to deceive the device, especially if the attack is performed just before startup.

This underlines a major difference between the two scenarios: when rebooted, the receiver blindly trusts the available signal, while during active operation, it may detect inconsistencies or try to maintain its original fix until the spoofed signal overwhelms it with a higher and more powerful signal.

The jamming phase observed during spoofing attempts where the receiver temporarily loses its GPS fix or experiences abnormal fluctuations in signal strength can be a valuable clue for detecting spoofing activity. This disruption often occurs when the spoofed signal starts to overpower the genuine satellite signals but has not yet fully taken control of the receiver.

In our experiments, we noted that this moment of signal confusion can trigger noticeable anomalies in satellite visibility, SNR levels, or positional stability. These irregularities can be used as indicators or triggers in spoofing detection algorithms.

In detection section, we will explore how such symptoms of jamming and transition can be systematically monitored and used to build effective real-time detection methods.

5.6. Discussions

Having validated the effectiveness of spoofing in various outdoor configurations, it becomes essential to reflect on different variations of these results. Beyond simply demonstrating that spoofing can succeed, our experiments reveal different results on how receivers behave under different conditions.

One important observation concerns the receiver's behaviour during the spoofing transition. When spoofing is introduced while the receiver already has a fix, we always observed erratic behaviours: the receiver may oscillate between a "no fix" and a "3D fix", or often long waiting on jamming mode before fully locking onto the spoofed location. These transitions are often accompanied by instability in SNR values and satellite count, offering useful indicators that can be exploited for real time spoofing detection algorithms.

The time to fix under spoofing conditions also differs depending on the receiver's startup state. In cold start scenarios; where the receiver boots directly under the influence of the spoofed signal, a 3D fix is often obtained very quickly (typically within 15 to 30 seconds), as the receiver blindly trusts the stronger available satellite data. In contrast, if the spoofing starts after a genuine fix is already established, the transition to the spoofed position can take significantly longer (40–90 seconds or more for a phase of instability or loss of fix).

We also define spoofing success based on objective criteria: let's say a spoofing attack is considered effective when the receiver maintains a stable 3D fix on the spoofed coordinates for enough spoofing time with true fix (like 30 seconds), with no fluctuation in position and consistent satellite tracking. This typically includes observing stable SNR values, no position jumps, and usage of the spoofed satellite list, indicating that the receiver is no longer attempting to re-acquire real signals.

Finally, environmental conditions are another critical component that affect spoofing performance. Reflections, obstacles can attenuate or distort the spoofed signal. In real world environments, walls, metallic structures, or even nearby objects can introduce multipath effects or signal absorption, requiring higher gain values to achieve the same spoofing effect observed in open field conditions.

6. Analytical Detection of GPS Spoofing Attacks

6.1. Detection Implementation Strategy

After successfully conducting multiple spoofing experiments and observing how the receiver behaves under our created attack, the next important step is to build a system capable of detecting spoofing attacks in real time.

Since GPS spoofing can deceive the receiver without triggering any errors or warnings, it becomes essential to monitor specific signal behaviours and identify symptoms that reveal the presence of a false signal.

First, we implement a detection strategy focuses on real-time monitoring of factors, which are typically altered and well observed during a spoofing attempt. These include:

1/ a sudden changes in position, where the calculated location jumps unexpectedly despite no physical movement

2/ Unrealistic speeds, such as instant acceleration beyond what is possible in the real world.

3/ Jamming phases, where the receiver temporarily loses its fix or fluctuates between fix/no-fix states before locking onto the spoofed signal.

4/ Satellite anomalies, such as loss of familiar satellites, sudden appearance of new PRNs, their disappearance or abrupt changes in SNR (C/N_0) levels.

These detections were implemented after returning to the use of Faraday Cage and two USRP 2900, first one used to simulate the real GPS signal (although not real), to simulate the first position and the second one used to simulate the attack and change of signal (position / satellites). Yes, the Faraday Cage blocks the signal coming from outside and so the need for two USRPs

Second, we go beyond handcrafted rules and introduce machine learning based detection, which allows the system to learn from data. Using real spoofing and normal signal datasets, we train a model to recognize patterns in the signal, such as fluctuations in SNR, elevation, and azimuth of satellites.

By continuously monitoring these physical and signal-level indicators, we aim to detect spoofing attempts as soon as they begin or during the transition phase.

False Positives and Receiver Instabilities: One of the major challenges encountered during simulation and real-time detection is the occurrence of false positives; instances where the system alerts a spoofing attempt despite the absence of any malicious signal. These can be particularly frustrating, as they reduce the trust in the detection mechanism and complicate analysis.

In our tests, u-blox receivers occasionally exhibited brief instabilities. Such events may cause temporary loss of fix, sudden shifts in the reported position, or fluctuations in speed, even under normal operation. These erratic behaviours can closely resemble the effects of spoofing. As a result, the system may detect what appears to be a spoofing pattern, such as an abrupt jump followed by re stabilisation, when in fact it is simply a side effect of receiver sensitivity. Managing with specific parameters and thresholds and minimizing these false positives is critical for ensuring that the detection system remains both sensitive to real attacks and robust against real fluctuations. The majority of false positives happen when starting the receiver so we always need to neglect an interval of time when starting the simulation (60 to 90 seconds) and then wisely choose the thresholds after executing several tests.

6.2. Real Time Kinematic Spoofing Detection

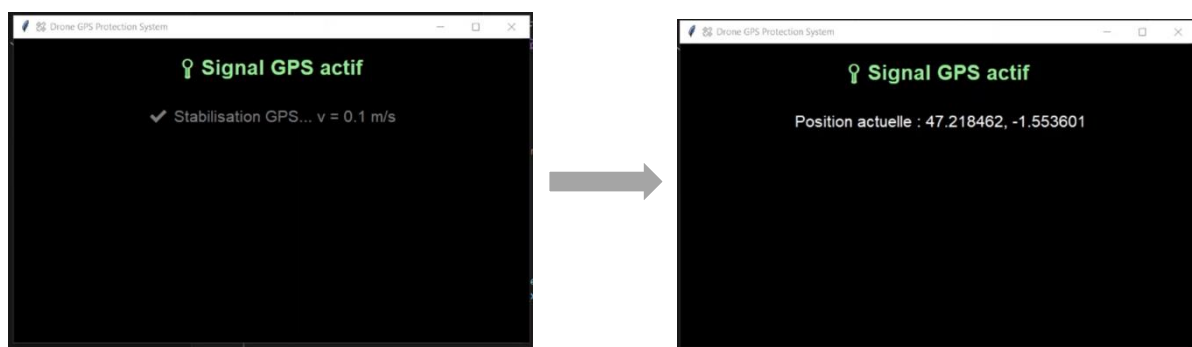
Once a GPS receiver is compromised, the attacker often seeks to override its positional awareness by simulating satellite signals that lead the receiver to report a false location. However, this transition does not always occur smoothly! In many cases, the reported position shifts abruptly and inconsistently, producing displacements that are physically implausible.

Such inconsistencies, especially sudden jumps of several tens or hundreds of meters in just one second, are a great sign of manipulation. A drone cannot instantaneously relocate without a corresponding motion pathway. These impossible transitions, form the foundation of our first detection method.

The approach uses only the GNSS derived position and time information to conclude whether the observed motion is physically realizable. By tracking the calculated displacement and its consequent velocity at that small interval of time, we can highlight moments where the receiver's behaviour deviates from logical physics

• Step 1: Monitoring Natural Stabilization

The system begins by observing the behaviour of the receiver to ensure it is in a stable physical state before any spoofing detection is attempted. Stabilization is defined as the condition where the calculated speed remains logical, either for static position (velocity ≈ 0) or dynamic movement (logical speed as chosen for the drone). This reflects either a stationary state or controlled motion. This phase is critical for defining a baseline of normal physical behaviour, used later to contrast against anomalous patterns.



• Step 2: Detecting Sudden Velocity Jumps

The stabilization is now confirmed. After launching the second signal for another position, the system begins detecting abrupt and implausible increases in speed. Any object in motion must obey continuity of movement and limits of acceleration. A sudden increase in the distance between two GNSS fixes, occurring over a short time window, will manifest as a sharp spike in calculated speed.

- Such jumps, like from 0 m/s to 30 m/s within one second, are physically improbable in many applications (drones, ground robots).

- These artifacts indicate teleportation-like transitions which do not correspond to any realistic physical movement.
- After losing the fix for the first position an alert saying that a spoofing tentative has been detected appears.



Figure 8: Real-time spoofing alert displayed by the detection system

• Step 3: Identifying Re Stabilization on Spoofed Signals

The spoofing phase causes the receiver to accept a false location. As a result, the system may again observe low and constant speeds following the initial velocity anomaly.

- This sudden re-stabilization, when occurring immediately after an unphysical spike, is highly suspicious, especially with a difference of position.
- The system flags this pattern, *sudden, high-speed jump followed by low-speed stability in a second position*, as a strong indicator of spoofing.

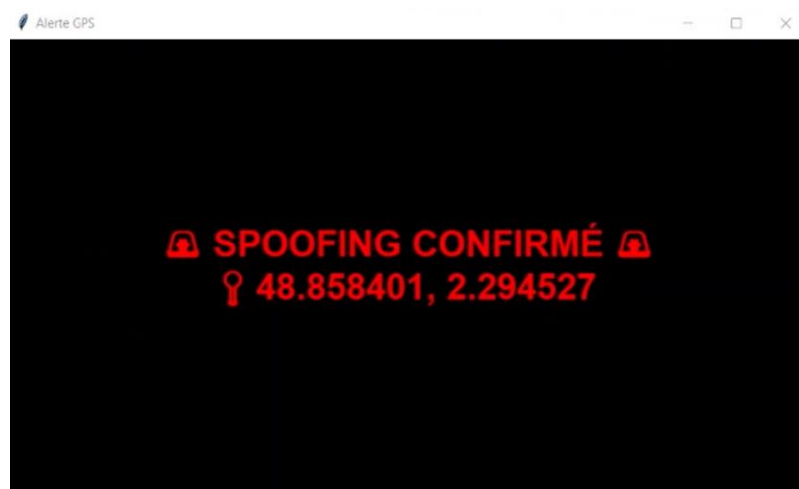


Figure 9: Spoofing confirmation alert with detected coordinates

Conclusion: Alert Triggering Based on Physical Inconsistency

When the system observes a combination of the following:

- Sudden, non-physical velocity jump;
- Followed by a re-stabilization at a displaced location;

it interprets this as confirmation of a spoofing attack. A detection event is then raised immediately and persistently displayed.

However, maybe you have already asked yourself the question, a more advanced attacker may choose to gradually manipulate the victim's position, avoiding the sudden physical jumps that typically trigger position/speed-based detection. By slowly drifting the spoofed signal and carefully shaping the trajectory over time, the attacker can the system inside the receiver without violating obvious physical constraints. In such scenarios, the transition from the authentic to the spoofed location may appear smooth and physically plausible, making velocity-based methods insufficient on their own. Moreover, a sophisticated spoofing setup can even mimic realistic movement patterns or match prior dynamics.

To counter these stealthier forms of attack, the detection must also focus on other parameters of the signal. Here comes some characteristics of the satellite data itself. Specifically, we examine how spoofing often introduces anomalies in the satellite constellation, including the sudden appearance of new PRNs, the disappearance of some previously tracked satellites, or abrupt shifts in received signal power (C/N_0). These signal irregularities, even when positional continuity is preserved, can reveal the presence of a falsified source.

6.3. Detection of Spoofing Through Satellite Anomalies

In most practical spoofing scenarios, the attacker uses pre-generated satellite signal files, often created using specific tools varying from simple to advanced ones (such as `gps-sdr-sim` in our case), based on previously recorded ephemerides. These datasets define the identities PRNs and expected signal characteristics of satellites at specific locations and times. While these spoofed signals can successfully trick a GPS receiver into computing a false position, they often rely on a static or partially outdated satellite configuration.

As a result, the spoofed satellite constellation may not perfectly match the real and the actual dynamic satellite environment at the receiver's location. The ejection of the spoofing signal introduces some variations, such as sudden appearances of new PRNs, sometimes the disappearance of some previously tracked satellites, or especially and most common, unexpected fluctuations in signal strength, the variations, if well observed and exploited, can be used as reliable indicators of a spoofing event.

• Logical Basis of the Detection Strategy

A legitimate GNSS environment typically evolves gradually: satellites rise and set in predictable patterns, SNR values change smoothly as elevation angles shift, and the constellation varies slightly over time. Conversely, during spoofing as shown in figure 11:

- The attacker may introduce PRNs that were present in the pre-recorded file but not in actual view of the receiver, resulting in the sudden appearance of new satellites from the receiver's perspective.
- Satellites that the receiver had been tracking may disappear abruptly if they are not included in the spoofed dataset, especially when the signal is much stronger than the real one.
- SNR levels may fluctuate sharply due to transmission artifacts, misaligned power calibration, or rapid switching between authentic and spoofed signals.

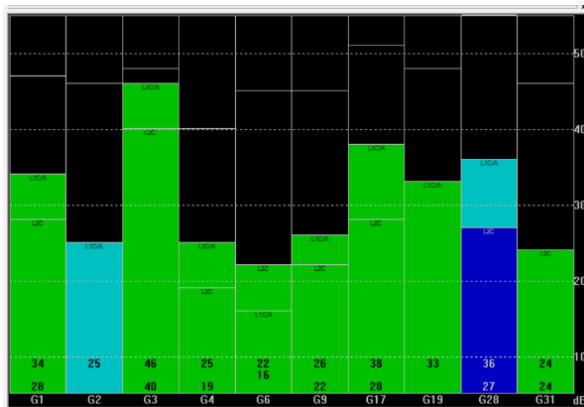


Figure 10: Satellites from first signal:
real GPS signal simulation

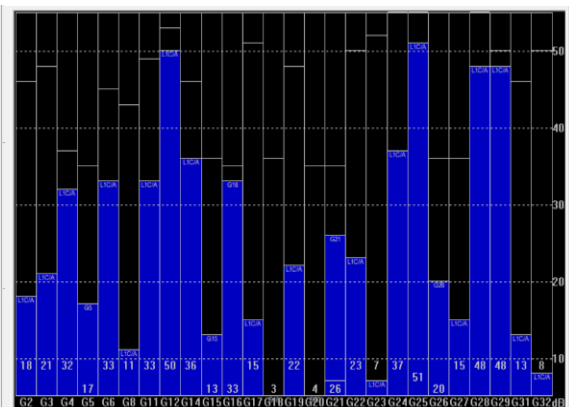


Figure 11: Jamming after including the
second signal as tentative of Spoofing

• Real-Time Monitoring of PRNs and Signal Strength

To detect these anomalies in practice, our system performs continuous observation of the satellite environment, specifically focusing on:

- **New PRNs:** Satellites not previously seen by the receiver, but which appear suddenly in the satellite list.
- **Disappeared PRNs:** Satellites that had been tracked consistently but vanish abruptly.
- **SNR Jumps:** Sudden changes in signal strength for individual satellites, exceeding a defined threshold.

The system maintains a small history of each satellite's SNR values, allowing it to identify both instantaneous jumps and instability patterns. This history is then used to compare current satellite visibility and signal strength against previous observations.

The comparison needs to be well studied and the thresholds must be deduced from several tests. Let's not forget about false positives !!

After performing many tests, a combination of the following was chosen:

- ≥ 4 new satellites appearing suddenly
- ≥ 4 previously tracked satellites disappearing
- unusual SNR fluctuations affecting multiple PRNs: exceeding 6 dB or 25% relative variation is considered abnormal, particularly when observed across three or more

satellites within a single second. In natural conditions, SNR typically varies gradually (often within 1–2 dB per second) making abrupt multi-satellite shifts highly suspect.

To avoid false alarms due to some arbitrary signal jumps because of the instability of ublox or the generated signal, the system should wait for anomaly confirmation over multiple cycles (2 to 3 consecutive detections) before triggering a spoofing alert.

• Visual Interface and Operational Feedback



Spoofing detected because 6 PRNs have disappeared: 02, 04, 05, 06, 10, 24



Spoofing detected because of an SNR jump of 6 dB detected for PRN 10



Spoofing detected because 6 PRNs have appeared: 01, 02, 03, 05, 12, 23



Spoofing detected because of two causes at the same time:

- 4 PRNs have appeared: 02, 03, 22, 32
- an SNR jump of 6 dB detected for PRN 24

As in the previous detection module, the spoofing alert system includes an integrated Tkinter-based graphical user interface designed for operational monitoring. Upon confirmation of spoofing:

- The window appears centered in the foreground, but in reality, it should be styled for visibility in a command control.
- The alert is categorized by cause, allowing operators to identify whether the spoofing was likely due to new satellite injection, SNR instability, or constellation mismatch.

Meanwhile, a real-time plot of satellite SNRs is continuously updated using Matplotlib, providing visual context for the ongoing analysis and showing the stability or volatility of each signal.

As attackers become more sophisticated, they may design spoofing signals that evolve gradually. That's why, it is always good to implement the most possible number of detection methods. To address this, we explore additional real-time detection techniques that analyse deeper properties of the GPS signals and their timing behaviour. These methods rely only on the data provided by the GPS receiver and can reveal subtle forms of spoofing that appear physically smooth but logically or mathematically inconsistent.

6.4. Pseudorange Consistency Checks (RAIM Outlier Detection)

In a properly functioning GNSS system, each satellite provides the receiver with a distance estimate, called a *pseudorange*, based on how long the signal took to arrive. These distances, when combined with the known positions of the satellites, must all point to the same physical location on Earth. If one of the measurements is incorrect; because it was manipulated or artificially introduced, it creates a contradiction: the set of distances no longer intersect at a single logical point. This internal contradiction is what we use to detect spoofing.

This method, often inspired by techniques used in aviation systems like RAIM (Receiver Autonomous Integrity Monitoring), continuously checks whether the current set of satellite signals agrees on a single position solution. If one or more satellites are inconsistent with the rest, their contribution will cause a sudden increase in the error margin or “residual” distance during position calculation. The detection algorithm tracks these residuals and raises a warning when they become too large to be explained by noise or environmental conditions.

This type of analysis is particularly useful when only a subset of satellites is being spoofed, which is common in resource-limited spoofing scenarios. A spoofer may attempt to spoof the navigation solution by manipulating just a few satellite signals, enough to shift the position, but not all satellites. In such cases, the affected measurements begin to conflict with the rest, and the system can spot this disagreement as soon as it happens.

It is also complementary to the speed and SNR-based approaches: while a smart spoofer might avoid triggering speed anomalies by slowly drifting the signal, they often struggle to produce pseudoranges that are both believable and consistent with all other satellite paths, especially over long periods.

This method may be not practical (because we need to recompute our own position using all satellites and it is like ignoring the benefit of having an already calculated position from u-blox) but it is a very powerful tool to detect spoofing attempts. To use this detection method, the system must collect the pseudorange values from all the satellites computed ourselves using:

$$\text{Pseudorange} = (\text{Receiver Time}) - (\text{Satellite Transmission Time}) \times c$$

A pseudorange is just an estimate of how far each satellite is from the receiver, based on how long its signal took to arrive. The receiver also knows the position of each satellite at that moment (the satellite transmission time and its position are given in the navigation message).

1. The first step is to calculate the receiver's position using all the satellites together. This is the normal GPS position fix.
2. Then, the system repeats the calculation several times, each time leaving out one satellite from the group.
3. If the result changes a lot when one satellite is removed, this means that satellite may not agree with the others, it may be giving a fake or incorrect distance.
4. The system measures how much each satellite's distance "disagrees" with the expected value. This difference is called a residual.
5. If one or more satellites have very high residuals, this could mean spoofing. The system can raise a spoofing alert if the disagreement is too large.

This check runs in real time, using the data already provided by the GPS receiver (no need for extra equipment). It works best when the receiver sees at least 5 satellites, so it can compare multiple combinations and find out which ones are suspicious. [\[6\]](#) [\[7\]](#)

6.5. Doppler and Clock Drift Monitoring (Dynamic Timing Consistency)

In addition to measuring distances from satellites, a GPS receiver also tracks how the satellite signals change over time. Two key elements are monitored: the Doppler shift (the change in frequency due to satellite movement) and the receiver's internal clock drift (how the local timing deviates from GPS time). These values follow predictable patterns during normal operation. However, during spoofing, these patterns can become unnatural or inconsistent.

Doppler Monitoring:

Each satellite in orbit moves rapidly with respect to the Earth, and this motion causes a measurable Doppler shift: a change in the frequency of the GPS signal due to relative velocity. The GPS receiver tracks this shift continuously for each satellite, and the values it observes should match the physical motion of both the satellite and the receiver. In a normal scenario, Doppler shifts vary across satellites and evolve smoothly over time. However, when a spoofer tries to deceive the receiver by transmitting signals from a fixed location on the ground or from a moving platform simulating a false trajectory, the resulting Doppler behavior often becomes unrealistic exposing the attack.

For example:

- If most or all satellites show very similar Doppler shifts, it is unlikely to be natural, since satellites are distributed across the sky.
- If the receiver is stationary, but the Doppler values suggest it is moving rapidly, this creates a contradiction that is physically impossible.

If you want to go much deeper into this technique. Here is the best research link for you: [\[8\]](#)

Clock Drift Monitoring:

Do you remember the fourth satellite used for computing the clock bias? A GPS receiver has a small internal clock to keep track of time. However, this clock is not highly accurate because it typically relies on a low-cost oscillator, which can drift over time. To stay synchronized with Coordinated Universal Time (UTC), the receiver continuously corrects its internal clock using the precise timing information from GNSS satellites, which are equipped with extremely stable atomic clocks.

Under normal conditions, the receiver's clock bias (the difference between its local clock and true GPS time) drifts slowly and predictably. The rate of this drift depends on the quality of the receiver's oscillator and the signal environment.

However, during a spoofing attack, an adversary sends fake satellite signals that manipulate the arrival time of the signals (This is what tricks the receiver into computing an incorrect position and time). To accept these fake signals, the receiver must adjust its internal clock to match the manipulated timing. This adjustment often results in sudden jumps or unnaturally smooth corrections to the clock bias.

For example:

- The clock offset might change by several milliseconds very quickly.
- Or the clock might appear to drift linearly and perfectly over time, which is too smooth compared to natural oscillator behavior.

If we continuously monitor how the clock bias evolves, we can detect such abnormal patterns. Signs of spoofing include:

- Rapid or large changes in clock offset.
- Unnaturally low noise in the clock bias evolution.
- The way the clock shifts might suddenly change direction or move in straight, unnatural steps.

These features can serve as indicators that the GPS receiver is under a spoofing attack. [\[9\]](#)

7. AI-Based Spoofing Detection Implementation

In addition to the real-time detection mechanisms based on logical rules and physical anomalies (such as speed jumps, SNR anomalies), this work also explores a data-driven approach using machine learning for GPS spoofing detection. This section describes the implementation of an AI-based classification model designed to distinguish between legitimate and spoofed GNSS signal snapshots based purely on signal characteristics. Unlike the real-time detection systems developed in this project, the AI model was applied not in real time, it operates on recorded data and analyses entire datasets after acquisition, rather than making alerts live as the receiver operates.

7.1. Dataset Construction

The training dataset for this model was based initially on a large open-source GNSS signal dataset retrieved from the internet ([\[10 \]](#) link for real GPS data and [\[11 \]](#) link for spoofed data). The dataset was created after well organised by a python script (**each row is 12 recorded satellites data for one second**), it included a wide variety of measurements collected under normal and spoofed conditions, each satellite is represented by its most 3 important features, so each row (second) has 36 pieces of data. The features used for each satellite are:

- **C/N₀** (Carrier-to-Noise density ratio)
- **Elevation angle** (in degrees)
- **Azimuth angle** (in degrees)

These three features were chosen because they offer a rich and compact representation of each satellite's signal and spatial geometry, without relying on satellite-specific identifiers such as PRN.

First, C/N₀ reflects the signal quality and strength, which tends to be inconsistent or unnaturally uniform in spoofed scenarios. Spoofed signals often show signal strengths that are either too similar or too strong across all satellites, unlike genuine GNSS conditions, where signal strength varies depending on satellite position, obstructions, and atmospheric effects. We can notice the difference in the next photos (the spoofing signal sometimes appears like a real GNSS but most of the cases and after stabilisation it is as shown in figure 12)

Second, this value of C/N₀ is influenced by many factors, especially the position of the satellite in the sky **Elevation/Azimuth**. For instance, satellites that are low on the horizon usually suffer from atmospheric attenuation and multipath, resulting in lower C/N₀, while those, overhead, tend to have stronger signals. That's why elevation and azimuth are not just geometric descriptors, they provide essential context that helps interpret the plausibility of each C/N₀ value. If a satellite appears to be at a low elevation but shows an abnormally high signal strength, it may indicate an incoherent and possibly spoofed signal.

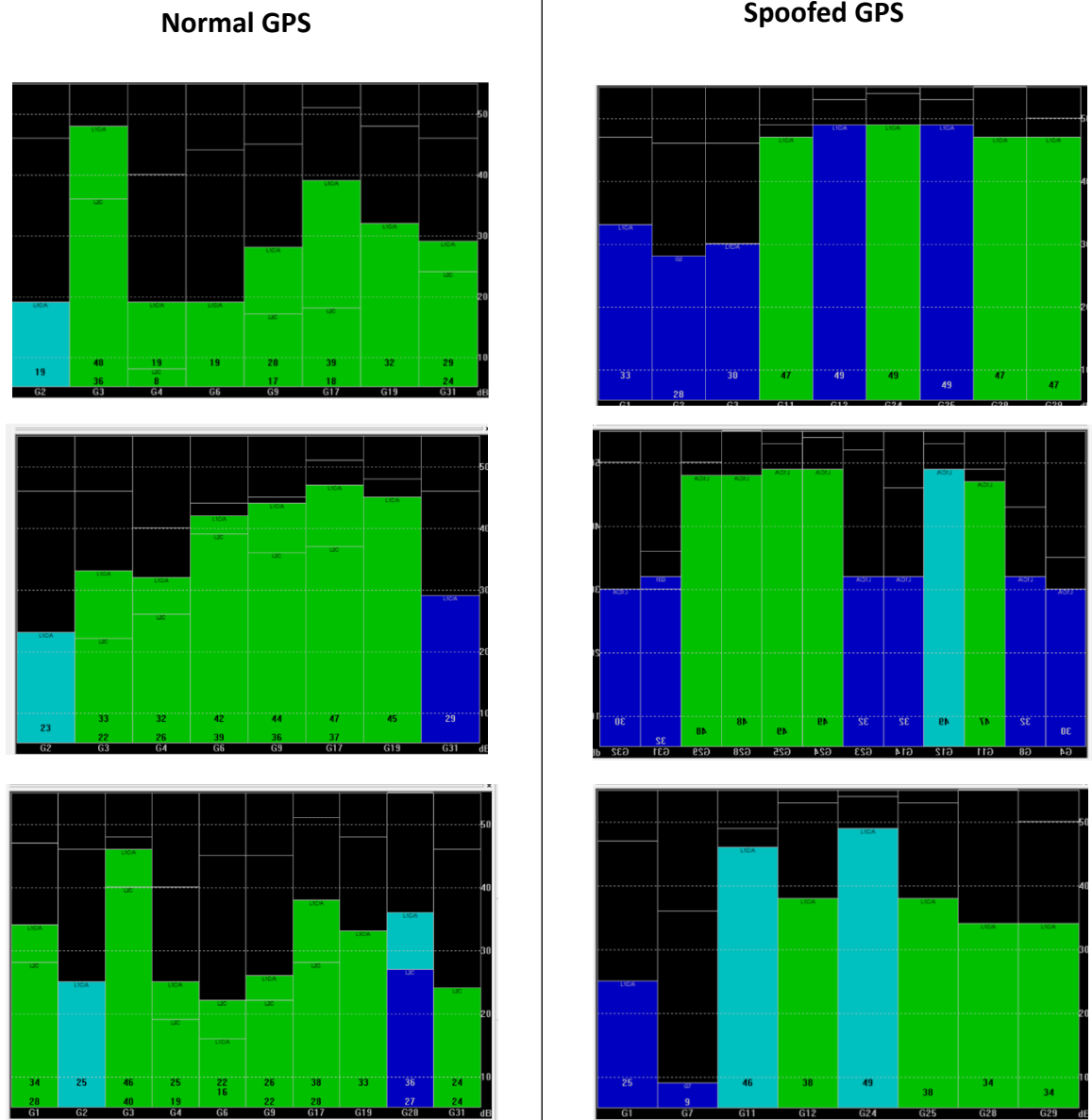


Figure 12: Comparison of satellite signal strength between normal GPS and spoofed GPS

The figure 12 compares the distribution of signal strengths (C/N_0) from a real GPS signal (left column) versus a spoofed signal (right column). In the normal GPS recordings, we observe a wide variation in signal strength across different satellites. Some satellites show high C/N_0 , while others are significantly weaker. This variation is expected in real-world conditions due to the natural spread of satellites in the sky, differences in elevation angles, atmospheric effects, and potential partial obstructions.

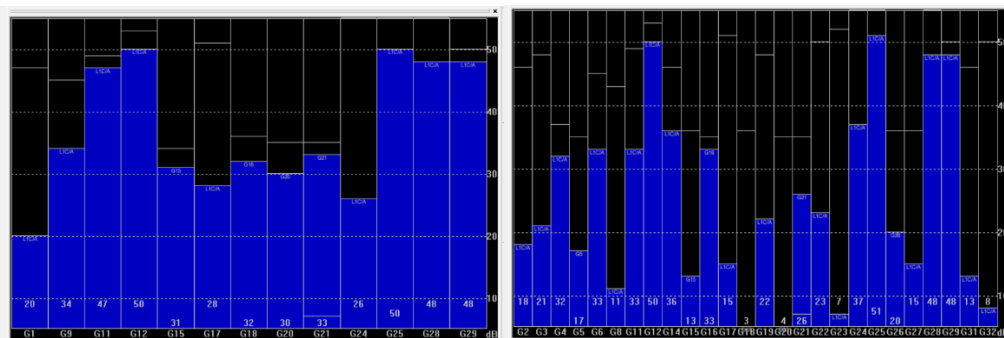
On the other hand, the spoofed GPS recordings tend to display a much more uniform distribution of signal strength. Many of the spoofed satellites have near C/N_0 values, it is like a more “perfect” signal. This is a common artifact of GPS spoofing, where all fake satellite signals are generated by a single transmitter with equal or nearly equal power levels controlled by the “gain” option we mentioned before, resulting most of cases in an unrealistic uniformity across the satellite SNRs.

While this pattern is not guaranteed in every spoofing second record, it is common and repetitive for the seconds (rows) recorded during our experiments. That’s why the AI model, should not rely on any single second of data.

Instead, we focus on a long-recorded time, (the longer is the recording time, the better is the accuracy!) and often work with average values over multiple seconds to get a more reliable view of the signal's authenticity. This approach helps reduce false positives.

To ensure generalization and avoid overfitting to specific satellite identifiers or constellations, satellite IDs such as PRN were deliberately excluded from the model inputs. Instead, the model relies solely on physical signal features like C/N₀, elevation, and azimuth. It allows the model to focus on detecting spoofing based on the physical coherence between signal strength and satellite geometry without looking at which satellite is in question. Since C/N₀ is naturally influenced by the satellite's position in the sky, the combination of signal strength with elevation and azimuth provides a way to verify the plausibility of each signal. For example, a high C/N₀ reported for a satellite near the horizon (where attenuation is expected) may suggest unnatural behaviour and thus a spoofed signal.

Remark: As shown before, spoofing, most of cases, especially when using simple emitting tool or due to the instability of the receiver, is preceded by Jamming which often produces SNR patterns that hugely differ significantly from those of real GNSS satellites. Two examples of Jammed SNRs are shown below.



Note !! All the results analysed in the following sections are based on signal files where the receiver had an established 3D fix and a clearly detected position. No jamming was involved in the generation of testing files !! With Jamming involved, False signals detection would be much easier, faster and much more accurate!

7.2. Model used - XGBoost

To build the spoofing detection model, I chose **XGBoost** (Extreme Gradient Boosting). XGBoost is a machine learning algorithm that works by combining many simple decision trees to make very accurate predictions. A **decision tree** is a model that makes decisions by asking a series of yes/no questions about the data. XGBoost builds many of these small trees in sequence, each one learning from the errors of the previous one, and then combines them to give a strong final prediction. This technique is called **boosting**, and XGBoost is a highly optimized version of it.

It is widely used because it gives strong results, especially when the input data is organized in tables (like in our case, where each row contains satellite signal features for one second). XGBoost is a convenient choice in this case.

In fact, it works very well for binary classification problems, where we want to decide between two options (like 0 or 1). In this case, whether the signal for that second (row) is **normal (1)** or **spoofed (0)**. It is good at learning patterns, which is important because it is very hard to make spoofed signals that look very similar to real ones, much more using open source tools.

After training the model, I used it to predict the spoofing status for each second of a given GPS test file. The model processes the file line by line, where each line represents one second of signal data, and gives a prediction: As mentioned, **1** if the signal looks normal, or **0** if it looks spoofed. Since spoofing may not be obvious in a single second, and many misclassifications can occur, I did not rely on just one prediction. Instead, I computed the average of all the predictions over the entire file (typically over 100 seconds, so 100+ line predictions). If the **final average** prediction is below 0.5, it means that the majority of seconds were classified as spoofed, and the final result is spoofed. If the average is 0.5 or above, the file is classified as normal.

7.3. Internal Insights from the Trained XGBoost Model

- **Number of Trees (Estimators):**

The trained XGBoost model uses **100 trees**, which is the default value in `xgb.XGBClassifier()` when the `n_estimators` parameter is not explicitly set.

Each tree is a weak learner (typically a shallow decision tree) that focuses on correcting the errors made by the previous ones. These trees are built sequentially, and together they form a strong predictive model.

- **Maximum Tree Depth**

The XGBoost model used a maximum tree depth of **6**, which is the default value. This means each decision tree can split up to 6 levels deep. It lets the model learn useful patterns in the data without memorizing noise. In our case, it was enough to capture the small but important differences between **normal** and **spoofed** GNSS signals.

- **Learning Rate**

The learning rate controls how much each tree's output affects the final prediction. In this model, the learning rate was set to **0.3**, which is the default moderate value. This means each new tree adds only 30% of its prediction to the overall result (the value predicted by the tree is multiplied by 0.3 before being taken into account) helping the model learn efficiently without overreacting to small patterns. Because the spoofing signals in the dataset show strong and consistent patterns, this value allowed the model to reach high accuracy using only 100 trees

- **Average Number of Leaves per Tree**

Each decision tree in the model consisted of many leaf nodes. A "leaf" in a decision tree represents a final decision based on the conditions along the tree's path. The number of leaves gives insight into how many unique rules or signal patterns the model is using. A higher number of leaves means the model is distinguishing finer details.

In our case the average number of leaves among the tree was **28.14**

- **Feature Importance Bias:**

XGBoost builds many decision trees to make predictions. At each step, it asks:

"Which feature helps me make the best split between spoofed and normal?"

It tries out different features (like C/No, elevation, azimuth of each satellite), and it chooses the one that helps reduce the number of mistakes the most. The feature that improves the result is used in the tree.

Over all the trees, XGBoost keeps track of:

- **How often a feature is used**
- **How much each use improves the model**
- **How many data points are affected when the feature is used**

The most common method to measure importance is called **"Gain"**.

That just means:

"How much does this feature help reduce errors?"

So each time we execute the training of the model we find different values of feature importance assigned to features input:

Example:

cn0_1: 0.0091, elev_1: 0.0238, azim_1: 0.0150, cn0_2: 0.0167, elev_2: 0.0229, azim_2: 0.0345, cn0_3: 0.0135, elev_3: 0.0175, azim_3: 0.0206, cn0_4: 0.0429, elev_4: 0.0114, azim_4: 0.0230, cn0_5: 0.1430, elev_5: 0.0104, azim_5: 0.0104, cn0_6: 0.3283, elev_6: 0.0033, azim_6: 0.0071, cn0_7: 0.0063, elev_7: 0.0168, azim_7: 0.0191, cn0_8: 0.0518, elev_8: 0.0148, azim_8: 0.0308, cn0_9: 0.0040, elev_9: 0.0086, azim_9: 0.0087, cn0_10: 0.0137, azim_10: 0.0063, cn0_11: 0.0211, elev_11: 0.0048, azim_11: 0.0032, cn0_12: 0.0164, elev_12: 0.0064, azim_12: 0.0061

This happens because XGBoost learns from patterns that consistently help reduce classification error. If a specific satellite's C/No value consistently differs between spoofed and normal conditions, the model will learn to rely more heavily on that satellite's signal. Conversely, features that don't show a clear pattern across the training data, or whose values fluctuate in both classes, will be considered less informative and receive lower weight.

This feature importance "bias" is not arbitrary; it is the result of repeated data-driven decisions during training. It reflects the model's ability to detect which signals are most important in identifying spoofing attempts.

- **Confidence Output**

The model reached **99.96% accuracy** using only raw features: C/N₀, elevation, and azimuth. No additional indicators like satellite count or speed were used, showing that spoofing impacts these basic values and can be concluded from these pieces of information.

7.4 Normal scenarios Testing

Open space (Normal GPS signals):

```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 99.94%  
Probabilité que le signal soit SPOOFED : 0.06%
```

Open space

In open space conditions (where the GNSS receiver has a clear and unobstructed view of the sky) the model performs as expected, identifying the signal as genuine with extremely high confidence. As shown in the result, the probability of the signal being normal is 99.94%, while the probability of it being spoofed is only 0.06%. This confirms that in ideal environments, the signal characteristics (such as C/N₀ distribution and satellite geometry) are fully consistent with what the model has learned as authentic GNSS behavior.

Spoofed area using the USRP 2900 with different Gain values gain values:

```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 1.54%  
Probabilité que le signal soit SPOOFED : 98.46%
```

Gain = 40

```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 2.56%  
Probabilité que le signal soit SPOOFED : 97.44%
```

Gain = 45

```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 11.00%  
Probabilité que le signal soit SPOOFED : 89.00%
```

Gain = 50

```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 1.26%  
Probabilité que le signal soit SPOOFED : 98.74%
```

Gain = 55

The results shown above demonstrate how using the USRP 2900 demonstrates the model's ability to detect spoofed GNSS signals. At all tested gain levels (40, 45, 50, and 55) the model consistently classified the signal as highly likely to be spoofed, with spoofing probabilities ranging from 89.00% to 98.74%. Even at Gain = 45, where the spoofed probability slightly decreased to 89.00%, the spoofing confidence remained dominant.

7.5. Challenges and limitations: Edge-Case Scenarios Evaluation

One of the major challenges in machine learning based spoofing detection lies in the ambiguity of certain environments. For instance, in some real world scenarios, such as near windows, inside buildings with partial satellite visibility, or even beside metal structures, the receiver may exhibit behaviours similar to spoofed conditions: erratic SNR values, weak signal coverage, or sudden fix losses. These cases often confuse simple analysis detectors and may lead to false positives or missed detections.

Let's see the examples below for files recorded for 6 mins

Bike parking (with a roof above)

```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 46.47%  
Probabilité que le signal soit SPOOFED : 53.53%
```

Model output:

- Probability of being NORMAL: **46.47%**
- Probability of being SPOOFED: **53.53%**

Although this environment is outdoors and the model is testing on a real GPS signal, the presence of a roof directly above the GNSS receiver (despite open sides) creates a misleading signal environment. Satellites that are directly overhead are blocked, while those at medium-to-low elevation are still visible. This changes the C/N_0 distribution and satellite geometry, making it look incoherent to the model. As a result, the model classifies it as spoofed, even though it's a legitimate signal. This is a classic false positive caused by unusual but legitimate environmental conditions.

In the window

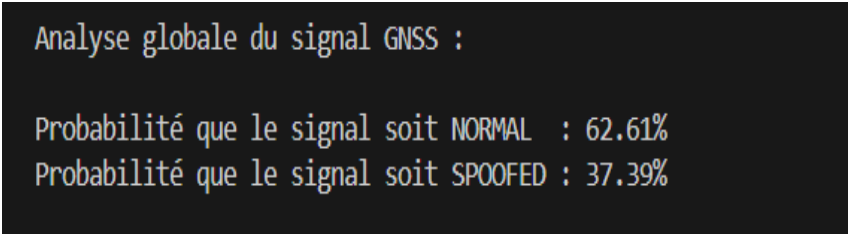
```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 53.20%  
Probabilité que le signal soit SPOOFED : 46.80%
```

Model output:

- Probability of being NORMAL: **53.20%**
- Probability of being SPOOFED: **46.80%**

This environment is located indoors near a window, where some satellite signals are partially received. The model's prediction is nearly balanced, indicating uncertainty. The signal quality in such areas often exhibits reduced SNR and unusual distribution, that can mimic spoofing-like behaviour. While the model leans toward classifying the signal as normal, the high spoofed probability (46.8%) shows that this is a highly ambiguous scenario that might trigger false positives.

Near the wall



```
Analyse globale du signal GNSS :  
  
Probabilité que le signal soit NORMAL : 62.61%  
Probabilité que le signal soit SPOOFED : 37.39%
```

Model output:

- Probability of being NORMAL: **62.61%**
- Probability of being SPOOFED: **37.39%**

This test was performed outdoors but close to a wall, which also like other cases, may reflect or partially block GNSS signals. Despite being outside, the proximity to a solid surface can distort the expected signal geometry and SNR patterns. The model still considers the signal mostly normal, but the spoofing score remains notable. This also suggests that even outdoor locations with limited obstruction can produce partially degraded signals that resemble artificial conditions, although in this case, the model is more confident in the signal being genuine.

To be noted:

Another important factor in this AI spoofing detection is the duration of the GPS signal recording. Since the AI model classifies the signal second by second, and the final decision is based on the average prediction over all those seconds, the length of the test file directly influences the accuracy of the result. In short recordings, a few seconds of weak or noisy data can heavily influence the outcome. But in longer recordings, the model has more time to observe consistent patterns and produce a more confident decision.

This effect is illustrated in the following two examples:

Analyse globale du signal GNSS :

Probabilité que le signal soit NORMAL : 53.20%
Probabilité que le signal soit SPOOFED : 46.80%

6-minute-long recording
near the window

Analyse globale du signal GNSS :

Probabilité que le signal soit NORMAL : 95.23%
Probabilité que le signal soit SPOOFED : 4.77%

10-minute-long recording
near the window

In the first case, the test we already discussed, was recorded over a 6-minute in the window, resulting in a nearly borderline classification: 53.20% normal vs. 46.80% spoofed. In contrast, the second test, recorded in the exact same place but extended to 10 minutes, led to a much more confident prediction: 95.23% normal vs. only 4.77% spoofed. This shows how longer recordings reduce anomalies, reduce the impact of transient errors, and increase model confidence.

However, it's important to recognize that our ultimate goal is to detect spoofing as quickly as possible, not just accurately. In real world applications such as autonomous maritime drone's navigation, waiting 10 minutes to detect an attack is unacceptable. Therefore, while long recordings help in offline evaluation, our detection logic must aim for both speed and reliability. This challenge motivated the need to train the model with a wide variety of real-world conditions, including edge cases, so it can make faster decisions, even with shorter signals.

To address this, many other records were made during the work to learn subtle statistical patterns from real and spoofed signals. This model if trained well and given an enough time, is capable of learning from edge cases conditions and improving overall robustness.

These additional recordings included:

- **Especially genuine GNSS signals** captured in challenging or ambiguous environments (like windows, balconies, near walls with partial sky visibility),
- **Spoofed signals** generated and recorded under controlled conditions using the USRP 2900 and gps-sdr-sim, simulating various spoofing trajectories.

We finally obtain a dataset with 71,467 lines. But it should be extended as much as possible, the more lines, the better is the model in misleading areas.

Limitations:

Unlike real-time spoofing detection methods based on SNR and speed (which operate instantly on live GNSS data), this AI-based detection approach uses recorded GNSS log files to make predictions. The model analyses samples of data collected over a short period and determines whether spoofing is present. This method is well suited for post processing GNSS recordings like analysing logs from the drones.

However, if the model is trained properly and reaches high confidence levels with minimal false positives, it becomes possible to apply it in a **quasi-real-time** detection: by testing short samples of data (such as the last 1–3 minutes) at regular intervals. The better the model is trained, the shorter the time window required for accurate predictions.

Another point is that we can make this more dynamic by using a technique called **sliding window** analysis. Instead of waiting for long recordings, we can continuously feed the model a small "window" of recent data (the last 20 or 60 lines), shifting this window forward as new data arrives. This enables more responsive spoofing detection, without processing the entire file each time. But it should be trained accordingly on slide windows not independent lines like our case.

Finally, while this version uses a machine learning tree-based model (XGBoost), more advanced deep learning models, such as **Recurrent Neural Networks (RNNs)** or **Temporal Convolutional Networks (TCNs)**, can also be trained on the same features.

8. Counter-Spoofing Response Implementation

What Should a Drone Do When GNSS Spoofing is Detected?

Imagine a drone in charge of a sensitive patrol mission, for example, monitoring the perimeter of a university campus. It's navigating through the streets, relying on GNSS signals for precise location tracking. But suddenly, something unusual happens: the drone starts to detect suspicious behaviour in the signal, unnatural jumps in speed or unexpected power levels from the satellites. What should it do next? Should it keep going? Should it stop completely? Or is there a smarter way to continue the mission safely?

These questions underlines a critical problem in autonomous systems: how to react intelligently to GNSS spoofing attacks in real time.

Scenario: Drone Patrol with Spoofing Threat

Let's consider a drone tasked with patrolling the area surrounding a faculty of engineering. As shown in figure 13, the drone begins its mission from the campus, moving along specific roads, and looping back across multiple points. This entire path is calculated and followed using GNSS data received from a u-blox receiver.

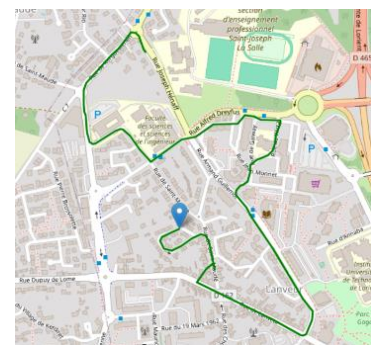


Figure 13: Drone patrol mission path

However, as the drone reaches a certain street, a spoofing attack is launched! The detection system, which constantly monitors both signal-to-noise ratio (SNR) and estimated speed, identifies anomalies. The system detects a sudden jump in position or an unexpected drop or spike in signal power! Both signs of GNSS spoofing!!

=> When a spoofing attack is detected with high confidence, using sudden jumps in speed or unrealistic SNR changes, the drone must react immediately to prevent being misled into a dangerous or false location. The very first step is to freeze all movement by stopping in place. This minimizes any risk of collision or unintended deviation while the system transitions to a safe mode.

Once movement is halted, the drone disables the GNSS module completely (Continuing to rely on false signals would only amplify the danger) At this stage, the drone activates its fallback logic: autonomous mode.

In this autonomous mode, the drone now considers several possible strategies. One option is simply to stop in place and do nothing, preserving battery and avoiding error until human intervention. Another option is to trace back its previous path using its recorded trajectory, a safe decision when the return path is known to be clear and reliable. A more proactive choice, however, is to compute a new path to the final wanted destination, bypassing the need for GPS entirely.

To carry out this computation, the drone relies on local maps and open-source tools such as OSMnx and NetworkX. OSMnx allows it to download detailed OpenStreetMap data for the local area and convert it into a graph of streets and intersections. NetworkX is then used to perform pathfinding algorithms such as Dijkstra or A* to determine the most efficient route from the drone's current position to the final destination. The resulting path is calculated purely using environmental data and estimated position, not GPS.

This process is illustrated in figure 14, where the green trajectory shows the path initially followed using GNSS signals, and the blue trajectory represents the new path calculated after spoofing detection. As we can clearly see, the blue route differs from the original and very direct, highlighting that the system is no longer following GPS but rather navigating independently in an urgent mode to come as quickly as possible to its final point

To follow this new path, the drone uses its onboard sensors: an IMU helps estimate its motion through acceleration and rotation, a compass ensures consistent orientation, and if available, visual SLAM (uses camera input to detect the environment and localize itself).

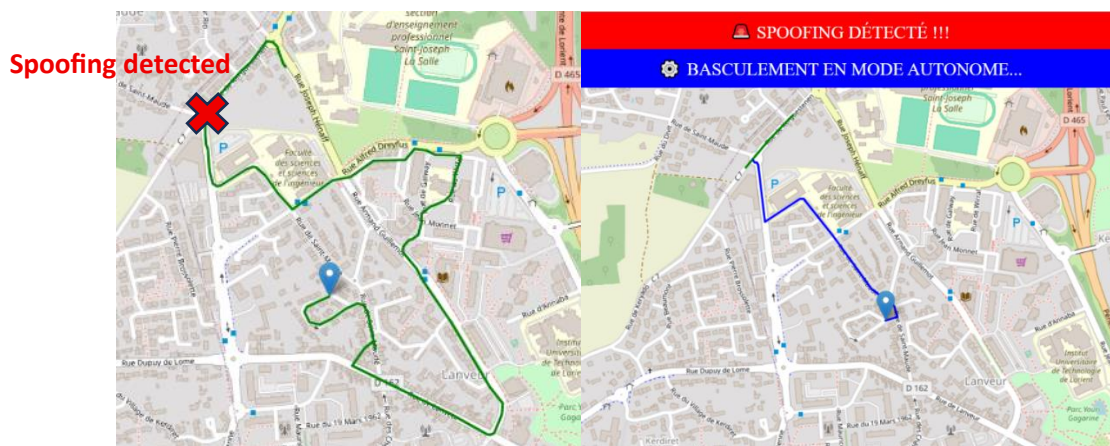


Figure 14: Trajectory correction after spoofing detection.

Summarizing:

After spoofing is confirmed, the drone performs the following sequence:

1. Stop movement temporarily to avoid entering a dangerous or manipulated zone.
2. Shut down GNSS input to prevent further misleading signals.
3. Switch to autonomous mode
4. Calculate a new shortest path to its destination (like an endpoint or a base).
5. Continue movement.

9. Countermeasures Against GNSS Spoofing

As we worked on some countermeasures that can be implemented and activated only after detecting Spoofing Attacks, there are many countermeasures that avoid being attacked in the first place. Modern GPS/GNSS (exp: Septentrio) receivers employ a range of countermeasures to achieve this objective. These techniques make it much harder for an attacker to fool the receiver into a false position. Below, we explain several major countermeasures in clear terms, with examples and notes on how attackers might try to overcome them. In the internship, many studies were conducted on the behaviour of **Septentrio** against Spoofing signals.

All references for the following sections are provided in the References section.

9.1. Anomaly Detection in Signals

The receiver continuously monitors incoming satellite signals for anything unusual. It checks if the signal characteristics (like strength, distribution, etc.) fall within normal ranges. If something looks “off”, for example, a signal is too strong or doesn’t match expected patterns, the receiver can flag or reject it as suspicious after very short time before it corrupts the position calculation.

How it works: Advanced GNSS receivers have internal algorithms that perform integrity checks on signals. For instance, they look at the correlation peaks for each satellite’s signal. Normally, each satellite should produce one clear correlation peak in the receiver’s signal processing. A spoofer, however, might create a second peak (often a stronger one, since the fake signal may arrive with higher power). Modern receivers can detect when **multiple signals for the same satellite** are present and isolate the true signal. The fake signal often shows up as a stronger secondary peak, which the tracking channels can recognize and reject so it never affects the position solution [\[12 \]](#)

Receivers also perform reasonableness checks on the data from satellites. For example, they keep a history of each satellite’s orbit data. If suddenly a satellite’s broadcast orbital parameters change in a way that doesn’t make sense (like jumping to an unexpected orbit!), this is a huge sign of sudden spoofing. It can cross-check the satellite’s information on multiple frequencies as well. For instance, it compares the info on GPS L1 vs L5 signal, to see if they agree [\[13 \]](#)

Any large inconsistencies could indicate spoofing, prompting the receiver to discard or distrust that satellite. These inconsistencies in different frequency intervals are explained in details later.

Let's imagine you are using a GPS receiver that implements this while surveying. If someone nearby turns on a spoofer that broadcasts a fake GPS signal with stronger power, your receiver's anomaly detection might notice that for one satellite PRN there are now two signals: one at normal power and one much stronger. A robust receiver will lock onto the authentic one and discard the suspicious strong signal. You as the user might not even notice and only see the real position on the controller.

Limitations: Simple anomaly detection can be foiled by a sophisticated spoofer that behaves like a "perfect fake." Instead of obvious anomalies like super strong signals, an advanced attacker could try to blend in with normal signals. For instance, they could gradually increase the power of their fake signals to smoothly take over from real ones (avoiding a sudden jump in strength). This is called a **synchronous spoofing** or **gradual takeover**. In practice, high-end receivers combine anomaly detection with other countermeasures to catch even these subtle attacks, but it shows that anomaly checks alone are not sufficient if the attacker is very careful.

Note that to perform such attack is very hard for a normal person and require much advanced tools.

9.2. Multi-Constellation Positioning

This method means that the receiver doesn't rely on just one satellite system (like GPS alone). Instead, it can use multiple GNSS constellations: for example, GPS (USA), Galileo (EU), GLONASS (Russia), BeiDou (formerly Compass: China), etc. In normal conditions, signals from all these constellations together improve accuracy and reliability. But importantly, using many constellations also helps security: if an attacker spoofs or disrupts one constellation (like we did with our GPS-SDR-SIM, the receiver can still check with the others.

How it helps against spoofing: A spoofer may target a specific system (GPS is the most commonly targeted since it's widely used). If a receiver is tracking, say, GPS and Galileo at the same time, and someone introduces fake GPS signals, the receiver can compare the solution it's getting from GPS satellites vs the solution from Galileo satellites. Under attack, the GPS-derived positions might start to disagree with the Galileo-derived positions. The receiver's integrity algorithms (like RAIM : Receiver Autonomous Integrity Monitoring) will notice the inconsistency. It can then exclude the suspicious signals. For example, if GPS data says you're 100 meters east but Galileo and BeiDou and others say you're still where you started, the receiver can conclude the GPS data is likely wrong or spoofed. It will then ignore GPS and rely on the other constellations that still agree with each other, maintaining a correct fix.

Figure 15: Multi-constellation positioning



For instance, **Trimble's** anti-spoofing scheme explicitly does this: if only GPS is under attack, the receiver can detect it by comparing against solutions from GLONASS, Galileo, BeiDou, etc, and then remove the GPS measurements from its calculation [12] [13]

In practice, multi-constellation support means an attacker has a much bigger task. Instead of spoofing, say, 4 satellites from one system, they might have to spoof a dozen or more satellites across different systems simultaneously; a far more complex operation.

So, an attacker aware of multi constellation usage has two main options, the first is to **spoof all constellations at once**, essentially creating an entire “false sky” of multiple satellite systems. This requires significantly more hardware and coordination, multiple transmitters or as mentioned in a previous part, very sophisticated simulator that can spoof all constellations. The second (more common in real incidents) is to **jam the constellations they aren't spoofing**. For example, a spoofer might generate fake GPS signals, but at the same time transmit wideband radio noise to knock out Galileo and GLONASS signals, so the receiver cannot use those as a reference. This combination of spoofing and jamming has been observed as a strategy to force a receiver into using only the fake signals [14]

However, employing jamming raises its own flags, the receiver might notice the sudden loss of all other constellations as a sign of interference. Advanced receivers often report such jamming attempts (showing an “Interference detected” status).

9.3. Multi-Frequency Receivers

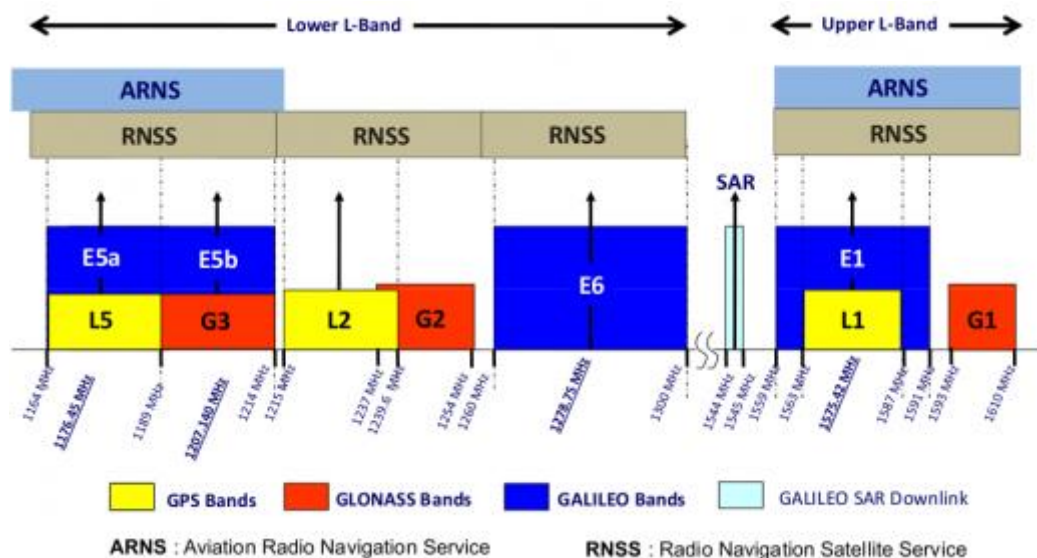


Figure 16: GNSS signals' frequencies

Traditional GPS devices often use a single frequency (L1 band, 1575.42 MHz). Modern receivers, especially high-precision ones, listen on **multiple frequency bands** (L1, L2, L5 for GPS; similarly, E1, E5a/E5b for Galileo, etc.). These are signals from the same satellites but at different radio frequencies. Multi-frequency usage improves accuracy and also helps detect interference or spoofing.

A spoofer now has to **mimic all the frequencies** to fool the receiver. If the attacker only spoofs the L1 frequency and your receiver is also listening to L5 = 1176.45 MHz, the L5 signal coming from the real satellite will contradict the fake L1 signal. The receiver can spot

the inconsistency. The timing or range measurements on L1 versus L5 won't add up correctly if one is fake. Many receivers routinely compare measurements from dual or more frequencies. An attacker who spoofs only L1 while the real L5 is present essentially can't fully convince the receiver; the receiver might reject that satellite or at least not get a usable solution. [\[13\]](#) [\[15\]](#)

Even if the attacker tries to jam the other frequencies (so that only their spoof on L1 is heard), the receiver can sense that. Suddenly losing all L2/L5 signals due to interference is a red flag in itself. Some receivers will output an interference alarm or at least note that multi-band consistency checks failed.

Just to explain more, consider a professional drone with a dual-frequency GNSS board. An attacker nearby plays a counterfeit GPS L1 signal. However, the drone's receiver also listens to GPS L5 and Galileo E5. It notices that while the L1 signals suggest a position shift, the L5 signals from the same GPS satellites either aren't present or still indicate the original position. Recognizing that genuine satellites wouldn't have such illogical outcome, it concludes these L1 signals must be false. The drone may then ignore the GPS L1 data and rely on L5 and Galileo signals, maintaining an accurate fix. The pilot or system might get a notification that spoofing is suspected, but the drone doesn't stray off course.

Attacker counter-methods: The obvious counter for the attacker is to **spooft all the frequency bands** that the receiver uses. This significantly raises the complexity of the attack. Instead of generating one signal per satellite, the attacker must generate two or three signals per satellite (spooft the L1, L2, and L5 components for each satellite) and keep them perfectly synchronized in time and content.

Doing this usually requires expensive, specialized hardware (or multiple coordinated transmitters) to cover the different frequency bands simultaneously. A low-cost hacker with one software-defined radio will find this extremely challenging. In fact, using cheap off-the-shelf tools, *multi-frequency spoofing was long thought to be infeasible*. However, some researchers have started to demonstrate it's **not impossible**. For instance, one team showed a clever way to spoof a multi-frequency survey receiver using a single SDR by rapidly switching or simulating a wide chunk of spectrum. But such attacks are far from trivial and require deep expertise. [\[16\]](#)

Another attacker tactic is, again, to jam the frequencies they can't spoof. If they spoof only L1 but jam L2/L5, the receiver is forced to use L1 alone. This can work, but it means the attack is no longer "stealthy". Jamming typically will be detected by the receiver's anti-jamming measures or at least cause a loss of signal lock on other bands, saying loudly to the system that something is wrong. Advanced receivers are often designed to gracefully degrade if some bands are jammed, and they may still refuse to produce a solution if only one band is in use.

*In summary, multi-frequency capability makes spoofing **much more complicated**. It doesn't make it utterly impossible (a very resourceful adversary could attempt multi-band spoofing), but it weeds out the simpler attacks that exploit single-frequency receivers.*

9.4. GNSS Signal Authentication (Cryptography)

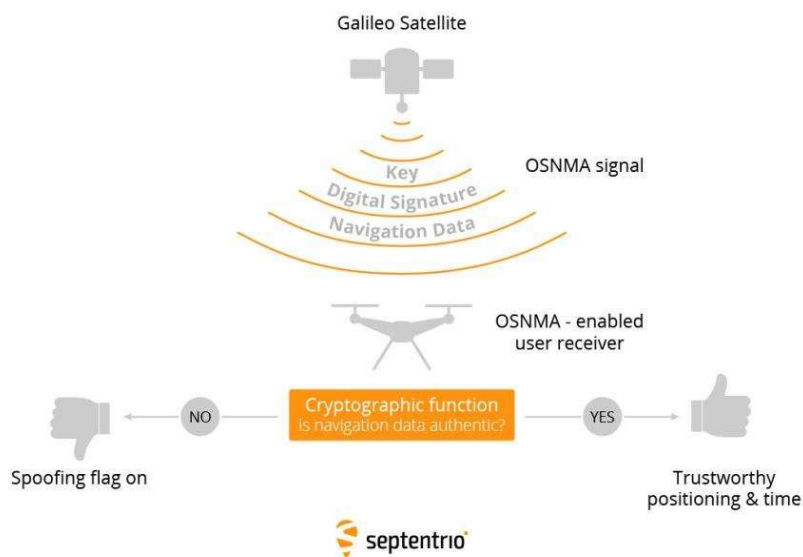


Figure 17: GNSS signal authentication using cryptography (OSNMA)

This is a cryptographic approach to verify that signals truly originate from genuine satellites and not an impostor. The idea is similar to how a secure website uses a digital certificate. Some navigation satellites transmit an **encrypted code or digital signature** embedded in their messages. Only someone with the correct key (usually the receiver or authorized users) can verify this code. And surely if a signal's authentication doesn't check out, the receiver knows it's fake and ignores it.

Current examples: The best known example is **Galileo's OSNMA** (Open Service Navigation Message Authentication). Galileo (the European GNSS) is introducing OSNMA on its civilian signals, which appends a cryptographic signature to the navigation data. Receivers that support OSNMA can verify that the data (and therefore the signal) came from a real Galileo satellite and wasn't tampered with. In April 2022, for instance, Septentrio announced it had integrated OSNMA into its receivers, making them among the first to use this new anti-spoofing security. With OSNMA, if a spoofer tried to broadcast a false Galileo signal, it would not have the correct digital signature (since the attacker doesn't have the secret keys). The receiver would detect the authentication failure and reject that signal. [\[17 \]](#)

Another form of authentication has existed in military GNSS: **encrypted military codes**. GPS, for example, has the P(Y) code on L1/L2 and the newer M-code on modern satellites. These signals are encrypted so that only receivers with the proper decryption keys (military or authorized users) can track them. If an adversary tries to spoof a military GPS receiver but doesn't know the secret code, the receiver can quickly tell something's wrong. Essentially, the spoofer cannot generate a valid military signal. This is why spoofing military-grade GNSS is exceedingly difficult. **the attacker would need to break encryption or somehow get the secret key code**

Attacker counter-method: However, while authentication raises the bar enormously, it's not an absolute cure. One known attack against authentication is **meaconing**, which is a term used for recording and rebroadcasting the genuine signals. If an attacker simply captures the real

satellite signal and transmits it again (perhaps with a slight delay or from a different location), the signal still carries a valid signature (since it originally came from the real satellite). The receiver will see the correct authentication and might accept the signal. **In effect, the attacker hasn't "forged" the signal; they've relayed a real one.** This can deceive the receiver, especially if the attacker can rebroadcast it in such a way that it arrives with a stronger signal or appropriate timing to shift the victim's position. The RNTF (Resilient Navigation and Timing Foundation) editor pointed out that authenticated signals won't *necessarily* protect against meaconing, because a rebroadcast includes the valid codes [\[18\]](#)

However, meaconing has its challenges for the attacker: they need to get a clean copy of the genuine signals (possibly requiring a receiver and transmitter setup), and any delay or offset they introduce can itself be noticed.

Advanced receivers might detect anomalies in timing (*the signal's time stamp or pseudo-range doesn't line up with others, or the same signal was seen twice*).

Also, schemes like OSNMA are being designed with some protections (such as delayed key release, etc.) that make simple replay less effective.

Additionally, authentication doesn't protect against *jamming*: an attacker could always jam the signals (including the authentic ones) to simply deny service. So, authentication is best used in combination with other defences.

I personally consider this method as the most powerful one against GNSS spoofing if timing anomaly detection is well optimised.

9.5. Interference Filtering Against Jamming (Band-Stop & Notch Filters)

Receivers can include filters **that remove suspicious frequencies or signals that shouldn't be there**. A band-stop filter (also known as a notch filter) is a technique to cut out a specific frequency band of incoming signals. If a receiver detects a strong interference at a particular frequency, it can dynamically apply a filter to "notch it out," while still listening to the rest of the band. This is part of a broader category of **anti-jamming and anti-interference** measures.

Many spoofing attacks are accompanied by jamming or interference. Advanced receivers can recognize these forms of interference by analyzing the frequency spectrum of what they receive. If a big spike or an unusual signal appears at a certain frequency, it places a notch filter at that frequency, essentially tuning out the jammer.

Septentrio's AIM+ (Advanced Interference Mitigation) technology, for instance, includes the ability to suppress various types of interference. Tests have shown it can **amazingly suppress single-tone jammers, frequency-sweeping (FSK) signals, and even wideband noise**. This means if an attacker tried to jam the L1 band with a continuous carrier or a chirp, the receiver could cut that part of the spectrum, reducing the jamming effect. The genuine satellite signals (spread across the rest of the band) can still be picked up. [\[14\]](#)

By filtering out interference, the receiver protects itself from jamming, and indirectly from spoofing attempts that rely on jamming. Recall that many spoofers jam the real signals first

before introducing fakes. If the receiver's filters nullify the jamming, the real signals remain available as a reference, and the spoof is far less likely to succeed.

Limitations: Filtering is excellent for narrowband or simple interference. If an attacker uses a single-frequency jammer which is the most common one using simple emitting tools, a few well placed notches can completely eliminate its effect. But if an attacker uses a very **broadband noise jammer** (one that floods the entire GNSS band with noise), you obviously can't notch out everything, otherwise you'd also remove the satellites' signals!

Receivers do have limits; wideband interference might still overwhelm the front-end if it's powerful enough. However, generating a high-power wideband jammer is more difficult and power intensive for the attacker

When it comes to spoofing specifically: if the fake signals are very well-made (spread-spectrum signals that look like real ones), the receiver can't notch them out simply based on frequency, because they occupy the same band as real signals. In that case, the receiver relies on the other spoofing detection methods described (anomaly detection, consistency checks, etc.). Filtering shines more in handling accompanying interference. For example, to spoof a multi-constellation receiver, an attacker might jam other constellations. The receiver's filters might nullify those jammers, preventing the attacker from "blinding" the receiver.

Septentrio notes that their AIM+ anti-jamming works hand-in-hand with their spoofing protection: it uses sophisticated filtering and signal processing techniques to ensure that when someone tries a spoof+jamming combo, the jamming part is dealt with. [\[19 \]](#)

In a test scenario, even when a spoofer increased power extremely high, the receiver's interference mitigation helped it still see some real signals until the power was absurdly high. Only at an extreme power level (at which point essentially all real signals were drowned) did the spoof finally succeed. This shows that filtering and mitigation can delay or prevent a spoofing success, and so it is forcing the attacker to use high power levels. [\[14 \]](#)

9.6. Directional Detection

This countermeasure uses the physics of signal arrival direction or polarization to detect spoofing. Normal GNSS signals come from satellites spread across the sky, so each signal comes from a different direction and angle. A spoofing device, on the other hand, often transmits all fake signals from one antenna (a single source on the ground): meaning they all arrive from essentially the same direction. Smart antenna systems exploit this difference using multi-antenna or phased array systems.

=> If a GNSS receiver has two or more antennas separated by some distance (or a single multi-element antenna array), it can compare the phase or time differences of incoming signals at the different antennas. **From these differences, the receiver can deduce the direction of arrival of each signal** (this is like how our ears locate a sound by the time difference between left and right ear).

In a legitimate scenario, signals from different satellites will arrive from different directions around the sky. In a spoofing scenario with one transmitter, all the counterfeit signals effectively come from one direction (the direction of the spoofer). By checking the antenna

array data, the receiver can spot if all signals have the same direction of arrival, which is a huge red flag!!!

In fact, experts note that if you have multiple antenna inputs, one of the most powerful tests is to see if all signals appear to emanate from one place. A legitimate set of satellite signals will never all line up like that.

Limitations: The directional detection via multi-antenna is extremely powerful, but it requires hardware (multiple antennas) that most consumer devices don't have due to cost/size. It's more common in military or high-end systems.

Attackers, in theory, **could try to defeat this by using multiple spoofing transmitters placed in different locations, each one spoofing a different satellite, so that the signals do arrive from the correct different directions.** However, coordinating multiple transmitters precisely to simulate a coherent set of satellite signals is extraordinarily difficult. They would have to synchronize their signals in time and phase perfectly and know the positions of the victim's antennas to get the geometric relationships right. This kind of "distributed spoofing" is more a theoretical possibility than a practical concern for most attackers. It would be an incredibly complex and expensive setup, likely only within reach of a state-level adversary, if at all.

Another counter the attacker might attempt is to **physically get near the victim antenna and broadcast from all around it turning with a higher speed than the antenna victim and be extraordinarily precise in time of broadcasting each signal to try to fool direction finding.** But again, this is impractical in most real scenarios (if they can get that close, maybe there are simpler ways to sabotage the system than trying to spoof the GPS!).

In short, smart antennas add a geometric/physical dimension to spoof detection that's very hard to fool with software alone. They force the attacker to replicate not just the signal content, but the spatial configuration of signals, which is a major hurdle. This is why for critical applications (like military), you often see multi-element antenna systems being used. For the average user, this might not be available, but it's good to know that such technology exists.

9.7. Sensor Fusion

Many navigation systems combine GNSS with other onboard sensors. This can include an **IMU** (Inertial Measurement Unit) that provides acceleration and rotation data, a digital compass for heading, wheel speed sensors in a car, barometric altimeters, odometers, etc. Sensor fusion means the system compares and blends these various inputs to get a more reliable and smooth navigation solution. In the context of spoofing, sensor fusion provides a way to cross-validate the GNSS data: do the other sensors agree with what the GPS is saying?

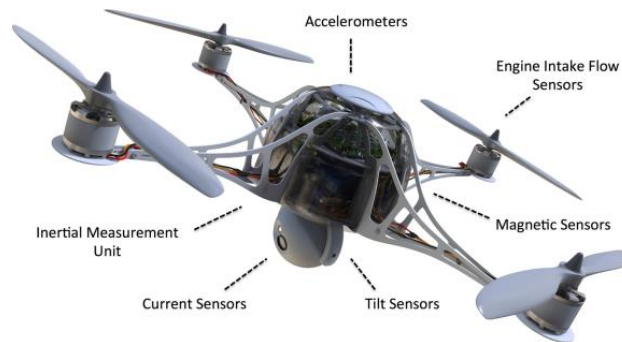


Figure 18: Integration of inertial sensors

Imagine if a spoofer causes the GNSS to suddenly show a large change in position or velocity that conflicts with inertial or other measurements, the system can detect an inconsistency. For example, your car's GPS suddenly indicates a jump to a location 500 meters away, but the car's wheel sensors and gyroscope indicate that you've been driving straight at 50 km/h! there's no way you could have teleported 500 meters in an instant or changed direction without the inertial sensors feeling it. The system can recognize: "GNSS must be wrong (or spoofed) because it contradicts the data coming from the IMU system"

Even on a smaller scale: if a spoofing attack is gradually drifting the GPS position when the drone is still, a high quality IMU will, for a short time, keep it honest. Let's say the spoofer slowly slides the drone's position north over several minutes. Meanwhile, the IMU doesn't sense that corresponding movement (drone is still). The fused solution can notice that the GNSS derived movement doesn't match the inertial-derived movement. It can then reject the GNSS data or at least raise a flag.

It can also play on direction: Consider the maritime drone's GPS says "it turned around" but the gyroscope and sensors say "it is still going straight," the drone's system can conclude the GPS info is wrong.

Limitations: Sensor fusion is a great add-on, but it's not that good proof against careful spoofing. **The main issue is that non-GNSS sensors have drift and limited accuracy over time.** An IMU, for example, is precise in the short term but tends to accumulate error (drift) when used over longer periods without correction. A clever spoofer can exploit this by introducing false position changes slowly, staying within the bounds of what the inertial sensor might perceive as normal drift.

As noted in a Septentrio insight article, if a spoofer increments the position gradually, those changes can be made small enough to be indistinguishable from the natural drift/errors: *"Using sensors other than GNSS such as an IMU or odometry can help flag spoofing by detecting inconsistencies between GNSS and the other sensors. While such sensors help reduce spoofing risks, they are not sufficient to provide full protection because they only output relative positioning which is subject to drift."* [\[20 \]](#)

9.8. Receivers with Multi-Layer Protection (Septentrio Mosaic)

What if we implement most or even all of the above Spoofing mitigation techniques? That would be of great level of security!

GNSS manufacturers are increasingly building receivers that incorporate many or all of the above countermeasures simultaneously. They often use specialized hardware, faster processors, and sophisticated software to guard against spoofing and jamming. A prime example mentioned is the *Septentrio Mosaic* series, similar high-end receivers are offered by others like Trimble for professional and industrial applications.

A multi-layered defence is far stronger than any single technique. A new-generation receiver might do the following *all at once*:

- Track multiple constellations (GPS, Galileo, GLONASS, BeiDou, etc.)
- Track multiple frequencies from each satellite (L1, L2, L5 bands) => adding more cross-checks.
- Continuously monitor signals for anomalies (extra peaks, sudden changes, inconsistent data) => providing real-time spoof detection.
- Use RAIM or other integrity algorithms to exclude bad measurements
- Employ some good advanced interference mitigation (notch filters, adaptive antennas) => nullifying jammers and simplistic attacks.
- Utilize cryptographic authentication when available (Galileo OSNMA, etc.) => verifying signals' authenticity.
- Possibly integrate with other sensors to help detect spoofing.

Because all these measures are in one package, the receiver can respond to an attack in a very robust way. For example, **Septentrio's AIM+ technology** (as part of their receivers) is described as a puzzle of hardware and software features: signal-level checks (including authentication and anomaly detection), measurement-level quality checks, and position solution checks (like RAIM+) all working together

Real-world performance:

Septentrio receiver rejected all my attempts of Spoofing using USRP 2900 that worked greatly on U-blox. As soon as the spoofing started, the receiver dropped the GPS signals and did not output any position solution rather than use the fake data.

Figures 19 and 20 show two screenshots I took for RxControl interface ; a tool to monitor Septentrio receiver (exactly like u-center for u-blox) where overall signal of 1/10 (low) with real GNSS detected the correct position and a higher overall signal 6/10 (good) with spoofed GPS signals didn't give any position at all, and blocked the whole signal (became 0/10) only after 15 seconds

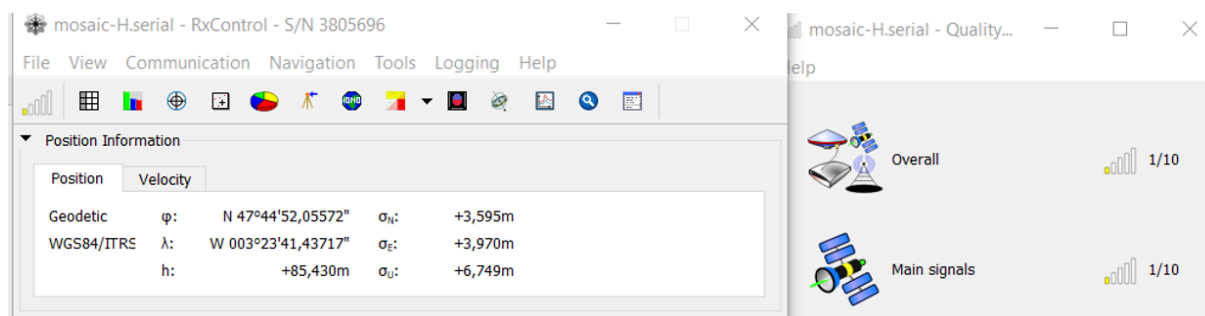


Figure 19: Normal detection of real GNSS signals (real position detected)

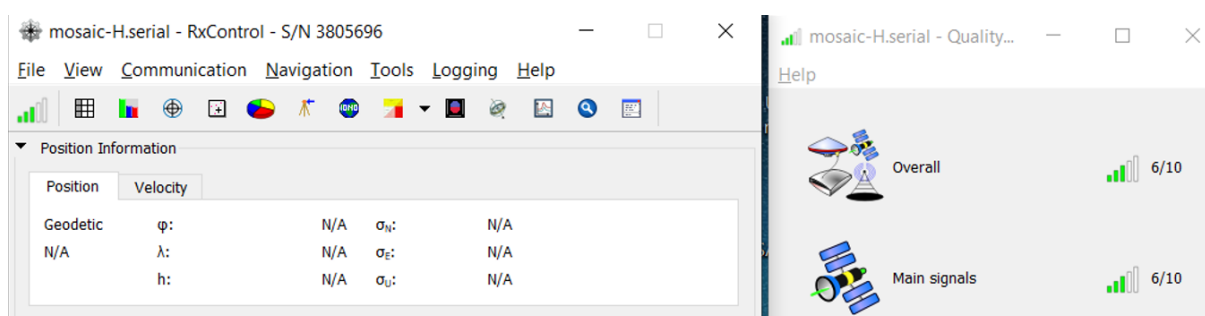


Figure 20: Spoofing resistance results (no position detected)

Additionally, researchers' tests have shown that these advanced receivers can be extremely hard to spoof. In a public test by researchers at GPSPatron, the Septentrio Mosaic-T was subjected to various spoofing attempts.

Key findings were: it was the first receiver they tested that could not be spoofed using a simple SDR (Software Defined Radio) and publicly available simulator. [\[14 \]](#)

Traditional methods like using a HackRF (an SDR like USRP 2900) with GPS-SDR-SIM failed entirely: the receiver detected something was wrong and refused to yield a wrong position *"No matter how hard we tried, all our attempts were unsuccessful!"*. It also says in the article *"Immediately after activating the spoofer, the receiver lost GPS satellites and stopped providing a navigation solution"*: The same as my case!

Trimble's latest receivers (with Maxwell 7 technology) likewise implement multi-layer defences, and they emphasize similar features: digital signal processing to reject spoofers, cross-checking multi-constellation data, RAIM, position sanity checks, etc. These receivers fuse GNSS with additional sensor data as well, as noted, to strengthen reliability

To sum up, defending against GNSS spoofing involves using a combination of methods: real-time anomaly detection, multi-constellation and multi-frequency verification, cryptographic authentication, antenna-based spatial filtering, and cross-checks with other sensors. Each adds a layer of security, and together they provide robust protection. With such receivers, the "bar" for a successful spoof becomes extremely high. The attacker would need to **simultaneously** fool multiple systems all the mitigation techniques above. For sure, nothing is 100% when it comes to security, but they make it so difficult that for most practical purposes, they deter and defeat all known spoofing methods.

10. Conclusion

This internship provided a comprehensive and practical journey into the world of GPS spoofing. Through hands-on experimentation using tools like USRP 2900, gps-sdr-sim, and u-blox receivers, we successfully demonstrated how GPS signals can be manipulated to mislead receivers into computing false positions, both static and dynamic

A major part of the work focused on the design and evaluation of real-time detection techniques, especially those based on sudden physical inconsistencies such as unrealistic jumps in speed or position, satellite signal anomalies like PRN changes or SNR fluctuations, and internal inconsistencies in Doppler or clock drift.

These techniques were complemented by a machine learning-based approach, which leveraged XGBoost to classify spoofed and normal using only C/N_0 , elevation, and azimuth information from visible satellites.

This report also discussed the practical challenges of deploying spoofing detection in real environments. False positives caused by weak or inconsistent GNSS signals near windows, walls, or partial obstructions remain a real concern. We explored how longer data collection; smarter threshold tuning can reduce such errors. Furthermore, we discussed the limitations of each detection method and the need for multi-layered, redundant protection, especially for critical systems.

Finally, an in-depth study of countermeasures implemented by advanced GNSS receivers, particularly from manufacturers like Septentrio, showed how a combination of signal anomaly checks, multi-frequency/multi-constellation verification, cryptographic authentication (like Galileo OSNMA), directional antennas, and sensor fusion can dramatically increase resistance to spoofing. Our spoofing tests, which successfully misled u-blox receivers, entirely failed to spoof the Septentrio Mosaic.

In conclusion, GPS spoofing is a real and growing threat. With the right combination of several anti spoofing measures, we can build systems that are can avoid being spoofed either entirely or at least make it very hard for the attacker to do so.

11. References:

- [1] **Pössel, M.** *Relativity and Satellite Navigation*. Einstein Online, Band 13 (2021), 1003. (An article explaining how GPS positioning works and its connection to Einstein's relativity). Available online: <https://www.einstein-online.info/en/spotlight/relativity-satellite-navigation>
- [2] **Global GPS Systems.** *Understanding Satellite Frequencies and GNSS Receiver Channels*. GlobalGPSSystems.com. (Web article describing GNSS satellite signal frequencies and receiver channel usage). Available online: <https://globalgpsystems.com/gnss/understanding-satellite-frequencies-and-gnss-receiver-channels/>
- [3] **GISGeography.** *How GPS Receivers Work : Trilateration vs Triangulation*. 2019. Available online: <https://gisgeography.com/trilateration-triangulation-gps>
- [4] **NASA CDDIS.** *GNSS Data – Daily*. NASA's Crustal Dynamics Data Information System (GNSS archive). Available online: <https://cddis.nasa.gov/archive/gnss/data/daily/>
- [5] **Angelo, J.** *Measuring GNSS Signal Strength*. Inside GNSS (GNSS Solutions), Nov/Dec 2010. Available online: <https://insidegnss.com/auto/novdec10-Solutions.pdf>
- [6] **everything RF.** *What is RAIM: Receiver Autonomous Integrity Monitoring? (Brief explanation of RAIM, a method by which a GPS receiver checks the consistency of signals to detect faults or spoofing)*. Available online: <https://www.everythingrf.com/community/what-is-raidm>
- [7] **Stanford GPS Lab.** *Receiver Autonomous Integrity Monitoring (RAIM)*. Stanford University GPS Lab, n.d. (Overview of RAIM technology developed in the 1990s to assess GPS signal integrity for aviation safety-critical applications). Available online : <https://gps.stanford.edu/research/early-gpspnt-research/receiver-autonomous-integrity-monitoring-raidm>
- [8] **Tu, J.; Zhan, X.; Zhang, X.; Zhang, Z.; Jing, S.** *A Low-Complexity GNSS Anti-Spoofing Technique Based on Doppler Frequency Difference Monitoring*. IET Radar, Sonar & Navigation, 2018. Available online : <https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-rsn.2018.5151>
- [9] **Truong, V.; Vervisch-Picois, A.; Rubio Hernan, J.; Samama, N.** *Characterization of the Ability of Low-Cost GNSS Receiver to Detect Spoofing*

Using Clock Bias. Sensors 2023, 23, 2735. Available online :
<https://www.mdpi.com/1424-8220/23/5/2735>

[10] **Wang, X.; Yang, J.; Huang, M.** *GNSS Dataset (with Interference and Spoofing) – Part I*. Mendeley Data, V1 (2024). Available online :
<https://data.mendeley.com/datasets/ccdgjcfvn5/1>

[11] **Wang, Yang, Huang, Peng.** *GNSS Dataset (with Interference and Spoofing) – Part III*. Mendeley Data, V3 (2024). Available online:
<https://data.mendeley.com/datasets/nxk9r22wd6/1>

[12] **Trimble** (Trimble Geospatial). *Your Trimble GNSS Receiver: Protection Against GNSS Spoofing*. Trimble Geospatial Blog. (Article introducing anti-spoofing features of Trimble’s Maxwell 7 GNSS receiver technology, explaining how advanced signal processing, RAIM, and other checks protect against false signals). Available online: [Your Trimble GNSS Receiver: Protection Against GNSS Spoofing](#)

[13] **Trimble** (OEM GNSS). *Protection Against GNSS Spoofing – Maxwell 7 Technology*. Trimble OEM GNSS Technical Note. (Technical bulletin describing Trimble’s anti-spoofing approaches in Maxwell 7 chipsets: detecting multiple peak correlations, checking orbital data consistency, RAIM, position sanity checks, etc.) Available online : [Protection against GNSS spoofing | OEMGNSS](#)

[14] **GPSPATRON**. *Septentrio Mosaic-T GNSS Spoofing Vulnerability Testing*. GPSPATRON Vulnerability Report, 30 July 2021. Available online:
<https://gpspatron.com/septentrio-mosaic-t-gnss-spoofing-vulnerability-testing>

[15] **Interview with Logan Scott**. “Spoofing”. Inside GNSS, 18 July 2013. Available online: <https://insidegnss.com/spoofing>

[16] **Curran, J. T.; Morrison, A.; O’Driscoll, C.** Feasibility of Multi-Frequency Spoofing. Inside GNSS, 14 June 2018. Available online:
<https://insidegnss.com/infeasibility-of-multi-frequency-spoofing/>

[17] **Septentrio**. *Septentrio Brings OSNMA Anti-Spoofing Security to Market*. Press Release (Leuven), 12 April 2022. Available online:
<https://amerisurv.com/2022/04/12/septentrio-brings-osnma-anti-spoofing-security-to-market/>

[18] Simsky, M. *What Is Spoofing and How Can You Ensure GPS Security?* Aerospace Testing International (Online Article), 5 November 2019. Available online: <https://rntfnd.org/2019/11/05/what-is-spoofing-and-how-can-you-ensure-gps-security-aerospace-testing-international/>

[19] **Septentrio**. *AIM+ Anti-Spoofing Protection : What Is GPS/GNSS Spoofing and How to Secure My Receiver?* Septentrio “Learn More” Series (Technical Insight), 2022. Available online: <https://www.septentrio.com/en/learn-more/advanced-positioning-technology/aim-anti-spoofing-protection>

[20] Septentrio. *Insight: Anti-Jamming and Anti-Spoofing for GNSS/INS*. Septentrio (Insight Article), 2022. Available online: [Insight: Anti-jamming and anti-spoofing for GNSS/INS](#)

XGBoost Documentation: Chen, T.; Guestrin, C. XGBoost : Extreme Gradient Boosting Library Documentation (stable release, 2023). (*Official documentation for XGBoost, describing the software, API, and parameters of this machine-learning algorithm*). Available online: <https://xgboost.readthedocs.io/en/stable>