Simulating GPS-spoofing of consumer UAVs as a side-channel remote control method

Botan Sinayic
bs477@kent.ac.uk

School of Computing
University of Kent

Word Count: 6,000

April 2, 2025

Contents

Abstract

With the rapid growth of UAV technology, applications in civil domains, cargo transport, and other fields have expanded. As we find new applications for these technologies, new cybersecurity threats emerge. It is essential to protect such technology from cyber attacks, to ensure safe, secure, and reliable usage for all.

This framework can serve as a dual use simulation tool that can allow drone manufacturers to evaluate the resilience of Commercial off the Shelf (COTS) positioning components against GPS spoofing attacks by evaluating known, modelled hardware and software. However, for the development of new models, such as adjustments to the Extended Kalman Filter for fine tuning will require real world validation. This simulation may not fully capture potential false positives compared to real world as it does not include environmental conditions such as weather. Ultimately, the simulation demonstrates how differential GPS spoofing combined with the use of controller APIs can allow an attacker to remote control a drone in some scenarios by translating X-Y coordinates into spoofed differential GPS coordinates. Due to legal, safety and ethical constraints, all experiments were conducted in a safe, hardware free environment.

Although carrying out experiments on real hardware would yield more accurate results, simulation offers an alternative which is reflective of real hardware conditions, legally compliant, and safe approach. Our results suggest that a stealthy GPS spoofing attack against Arducopter can enable an attacker to gain full remote control of the drone without the need to hijack the control channel, which is typically encoded and encrypted, making it a poor target for real-time attacks. Instead, the attack will leverage manipulated GPS signals to mislead the drone about its true position.

1 Introduction

A GPS spoofing attack involves an attacker crafting a malicious GPS signal that tries to replicate a legitimate signal to deceive the flight controller to miscalculate its location. This is because GPS signals are not encrypted, due to the difficulty of coordinating key exchanges between satellites and ground receivers. Receivers have insufficient power to respond to signals received from satellites in a GPS, GLONASS or other positioning constellations. GPS receivers lock onto any GPS signal that is sufficiently powerful, well-crafted and typically require a minimum of 8 confirmed satellites, with 13 being optimal to ensure accurate positioning and reduce errors. Arducopter, however, required a minimum of 6 satellites. In this project, the Software-in-the-Loop (SITL) simulation framework will be employed to evaluate the resilience of UAV flight controllers against malicious GPS signals. The simulation uses open-source Arducopter module part of the Ardupilot suite as the test drone and will replicate real-world sensor and flight behaviours in a controlled, hardware free environment.

At the core of our methodology is the exploitation of the Extended Kalman Filter (EKF), which integrates sensor fusion techniques to detect discrepancies by observing changes in GPS, magnetometer, an Inertial Measurement Unit (IMU) data over time. This fusion enables the EKF to determine whether the drone's position has shifted without corresponding instructions from the flight controller, a potential indication of a dangerous fault. By combining X-Y coordinates with incremental offsets to the legitimate GPS data, the framework demonstrates how the EKF's tolerance threshold can be exploited without triggering its safety measures.

GPS spoofing represents a serious security and safety threat to consumer UAVS and any technology that relies on GPS data for positioning. For instance, a drone on an autonomous mission can be hijacked or misdirected for a malicious purpose. Detailed abuse scenarios of such attacks will be discussed in the conclusion section.

A contribution of this project is the integration of a PS4 controller to provide a user-friendly interface for employing GPS spoofing to manually control and idling drone. This contribution will yield insights into how an adversary might combine GPS spoofing with real-time controller inputs to remotely hijack a UAV. This framework offers a legally compliant and safe method of testing UAV security measures and contributes to the development of powerful and effective countermeasures against cyber threats.

2 Background & Related Work

Prior to undertaking the simulation, we first explored the possibility of using the USRP B210. A detailed explanation of this attempt can be found in Appendix C.

The Global Positioning System (GPS) is a critical navigation system that relies on GPS signals received from satellites on the L band. Civil GPS signals are not encrypted, largely because encrypting them is a hard design problem given the extensive infrastructure required for reliable ground-to-space communication. As a result, it is not feasible to encrypt this data. Moreover, the integration of sensor fusion such as the EKF in conjunction with methods like Google's positioning infrastructure and cellular triangulation makes encryption redundant. Drones do not benefit from 5G connectivity yet, position verification must rely on EKF's ability to combine position related sensor data to identify anomalies.

GPS operators work by receiving signals from multiple satellites in medium Earth orbit, each satellite being approximately 20,000 km away. By measuring the time delay between when the signal was sent and when it was received, the GPS receiver computes the pseudo-distance, or the approximate distance to each satellite. With signals from at least four satellites, the receiver uses trilateration to determine its three-dimensional position.

Extended Kalman Filter in navigation systems is a crucial component in all modern flight controllers like Arducopter. It is an algorithm that estimates the drone's position, velocity, and altitude. The EKF not only tracks where the drone is through this continuous estimation process but also identifies any sensor inconsistencies. Comparing the predicted value and the latest (non-IMU) value every 100 ms, the EKF is able to calculate the innovation value and the variance value. The variance value reflects how much trust the EKF has in its current estimate. If the innovation values are positive and the variance value is 1 or higher, a fail-safe is triggered. A sudden GPS jump without corresponding movement detected by the IMU indicates a potential sensor fault, in such cases the flight controller triggers a fail-safe.

Arducopter requires a minimum of 6 confirmed satellites to obtain an accurate GPS fix. This redundancy is required as it compensates for signal loss, interference, and environmental factors. For example, multipath effect occurs when two radio signals reach the receiver via multiple paths, and this causes interference and signal degradation, impacting the reliability of positioning. Even though, environmental factors do no exist in the simulation environment, the simulation tries to mimics this type interference through configurable sensor noise and error injection models. For example, our simulation setup includes parameters such as `SIM_ACC_RND` to add noise to the accelerometer, `SIM_WIND_SPD` and `SIM_WIND_DIR` to simulate wind effects. These types of models deliberately introduce noise, random errors and distortions to replicate real world conditions like signal delays, sudden drops, and inaccuracies. These models and approach enables manufacturers to test and refine the autopilot's filtering and fault detecting algorithms like the EKF.

For instance, if wind causes the copter to drift, the flight controller will rely on the EKF to fuse GPS and other sensor data to update the vehicles position estimate so that path corrections can be made. An attacker can advantage of this behaviour by injecting fake incremental drifts in the spoofed GPS packet that mimics natural sensor noise and remains below the EKF's threshold, by doing this the flight controller will interpret this changes as normal drifts and will attempt to correct it course based on its last known position. By continuously transmitting these spoofed GPS packets the attacker will be able to steer the copter in their desired location. It is important to note that the EKF itself does not directly control complex/correctional behaviours, it provides an estimate of position based on fusion of sensor data that the flight controller uses to make the course corrections. If the attacker exceeds this threshold, the EKF will trigger a fail-safe and land. So the success of a GPS spoofing attack depends on keeping under this threshold set by the EKF

Most modern commercial drones on the market rely on sensor fusion algorithms like the EKF, however, while the EKF is widely implemented, its configurations like the threshold for errors is highly dependant on the specific sensor suite and hardware characteristic of the drone. For instance, a larger more powerful drone will introduce more vibrational noise, therefore, the IMU sensors would need have more enhanced shielding and a higher polling rate for the EKF to effectively filter out the noise while still producing a precise positional estimate. Consequently, the drone manufacturers tend to keep these specific configurations and behaviours of their sensor fusion algorithms confidential and also put in countermeasures to make reverse engineering more difficult for attackers. As a good understanding of these mechanisms will equip the attacker with the necessary knowledge to develop bypass methods.

The inertial measurement units, such as the accelerometer and gyroscope measure the linear acceleration and angular velocity. It estimates short-term changes in position and altitude. The challenge with this is that the accelerometer suffers from drift, over time the raw data leads to errors. The magnetometer measures the orientation of the drone relative to the earth magnetic field and this is susceptible to electromagnetic interference. These are the reasons why a tolerance threshold is implemented as these drifts occur naturally and the flight controller needs to consider them as they can accumulate to greater errors.

GNSS systems are associated with specific geopolitical regions, they are controller by various political entities. For example, the United States operate GPS, Russia controls GLONASS, and the European Union manages Galileo. These entities have the ability to disable and modify signals, this drives the development and development of independent GNSS satellite systems.

2.1 Prior research

In the well-known UT Austin Humphreys et al. (2012) experiment, researchers successfully managed to capture a drone using GPS spoofing using real hardware. They also managed to make the UAV have a large number of unrecoverable navigation errors that caused the UAV to crash. Gupta et al. (2023) adopted the simulation-based approach, by leveraging the pre-built module called `fake_gps`. The keys features of this module include:

- **MAVLink integration:** This module uses the MAVLink protocol to communicate GPS data, which allows for precise injection of offsets into the telemetry stream. This capability facilitates a controller simulated of GPS spoofing attack. Even though big manufacturers tend to opt out of propriety software like DJI's occusync, MAVLink integration is still common in hobbyist drones and those that leverage open source autopilot systems like Ardupilot and PX4.

- **Offset injection:** Now that the drone communicates via MAVLink, the module now acts as a bridge to the GPS telemetry stream, this enables us to simulate various GPS spoofing scenarios in a controlled manner.

- **Ease of use:** Because it is already built in, it wont require extensive custom development, facilitating a quick setup and experimentation.

In the project Noh et al. (2019), the authors took the approach of directly modifying the GPS emulator part of the HAL abstraction layer of the SITL. This approach allowed the researchers to interact with the underlying simulation at a low level, this ensures that any modification effect the system in a manner that closely mimics how changes would manifest in real hardware. By making modifications to the HAL abstraction layer, the entire data flow between the simulated sensors and the control algorithm is influenced, enhancing realism.

Another approach includes using the pre-built parameters/commands, `SIM_GPS1_GLTCH_X`, `SIM_GPS1_GLTCH_Y` which applies a apply offsets to the latitude and longitude. This was built by Arducopter developers to test the effects of wind and other types of drifts. We will be taking this approach in our first implementation. The key features of taking this approach includes:

- **Ease of integration:** As these parameters are already built into Arducopter framework, they can be called without the need for extensive code modifications or additional custom modules.
- **Precise offset control:** These parameters allow the injection of exact offset values to the latitude and longitude.
- **Reproducibility:** This approach allows experiments to be consistent and repeatable, which is crucial for systematic testing and validation.

3 Design

This section will entail the design and execution of the simulation-based experiment, describing the simulation setup, rationale for using Software-in-the-Loop (SITL) environment and the attack implementation strategies.

3.1 Simulation setup

3.2 Why SITL?

To be able to conduct this experiment in a safe, ethical and legally compliant manner, we employ the ArduPilot SITL (Software-In-The-Loop) simulator[1]. This framework can utilise the open source Arducopter module from the

[1]https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

Ardupilot suite as the test drone, enabling us to replicate real sensors and flight behaviours in a controlled, hardware free environment. This setup will enable us to have precise control over the experiment conditions and a safe way to inject malicious GPS data directly into the simulated flight controller, without altering the drones actual simulated world position. The physical location simulation will remain consistent, while the flight controller responds to spoofed GPS readings as though they were genuine. This approach ensures we can precisely manipulate the drone's perceived location and evaluate its behaviour and the effectiveness of the safety measures in place.

Using this framework comes with several advantages. It ensures legal and safety compliance by eradicating any risk associated with the emission of unauthorized signals on real hardware. It also provides flexibility to the project, enabling us to make quick iterations and adjustments to the attack parameters such as offsets and rate. Finally, SITL enables a high-fidelity replication of flight dynamics and sensor behaviour, making it a powerful proxy for real-world testing.

3.3 Environment components

- Arducopter [2]

  Arducopter will be utilised as the flight control firmware.

  - Open-source nature: Arducopter is a widely used open source module that enables us to include extensive modification and realistic of UAV behaviour.

  - Built-in EKF: More close to real world conditions as modern drones utilise countermeasures like the Extended Kalman Filter algorithm.

- QGroundControl[3]

  We will use this as our ground controller.

[2] https://ardupilot.org/copter/ Arducopter documentation
[3] https://qgroundcontrol.com/

- Easy quick setup
- Visualization tools: Allows us to view telemetry data, parameters and sensor outputs in real time.
- Easy installation of flight logs

- Packages

  We will use these packages for our spoofing script.

  - PyGame[4] - utilised to retrieve analogue inputs from the PS4 controller.
  - DroneKit[5] - This will serve as the communication bridge between the simulation and Arducopter's flight controller.

3.4 Simulation environment

For this project we will be using Ubuntu 22.04 virtual machine running on an Apple M1 Pro. To launch the Arducopter simulation run,

```
sim_vehicle.py -v ArduCopter --console --map --osd
```

`sim_vehicle.py` is a startup script that configures the current code branch of the SITL firmware, this includes the vehicles position, altitude, orientation, and velocity. This process defines both the simulation world space and the entities within it. The world space is incorporated with global physic parameters such as gravitational forces, wind conditions and air resistance, these parameters effect all of the objects within the simulation. The copter itself is an individual object that has specific state parameters and interacts with the environment according to the simulated physic laws.

Once the simulation is running, the vehicles state is continuously updated by the SITL physics simulator, it computes the current state of the copter using motor output values. This data is then retrieved by the `SIM_GPS_GPS::read()` function and forwarded to the GPS manager `AP_GPS`, updating its GPS object

[4] https://devdocs.io/pygame/
[5] https://dronekit.io/

every 200 ms with fresh positional data. The EKF then retrieves this object and fuses it with IMU sensors to generate a state estimate which takes account for noise and sensor variances. The fine tuned state estimate produced by the EKF is then provided to the positional controller within Arducopter flight code which determines the motor outputs based on control gains and other parameters tuned for a stable flight. This is where the decision of triggering a correction is made because of the offsets applied to the GPS data. These motor outputs are then relayed to the motor driver modules to be be converted into actionable commands for the simulated UAV actuators. This motor activity is then fed back to the SITL physics simulator, updating the vehicle's state in the world space and thus completing the loop as seen in Figure 2.

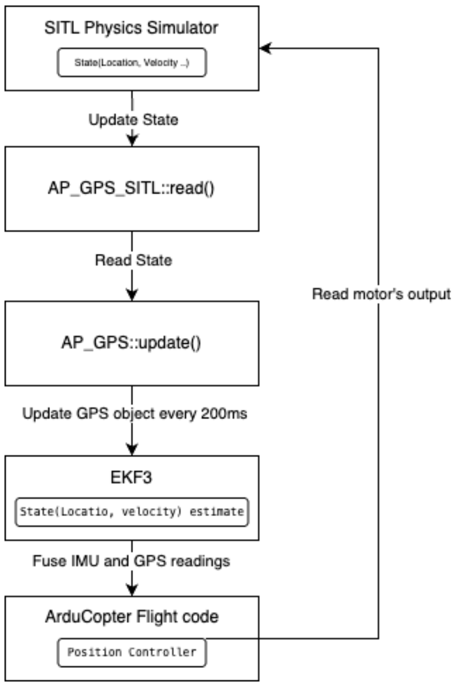


Figure 1: Execution loop of SITL environment

In the context of the GPS spoofing attacker, rather than altering the vehicles state provided by the physics simulator, we alter the updates made to the GPS object by the `AP_GPS` module. This approach attempts to simulate real-world conditions where spoofed sensor inputs must pass through validation checks, such as the Extended Kalman Filter sensor fusion process before being able to influence the flight controller.

4 Attack Implementation

Further detailed explanation of earlier experiments and an analysis of results can be found in Appendix A and B.

The primary objective of this attack is to manipulate Arducopter perceived position by injection spoofed GPS signals in a way that a path correction is triggered by the flight controller causing it to steer in the attackers desired location. All while maintaining stealth and within the tolerance limits of the EKF to bypass safety mechanisms such as a fail-safe. We will attempt to achieve out objectives through two different approaches, all essentially attempting to inject spoofed GPS data to the GPS telemetry stream.

Early experiments which included attempts to spoof the GPS with arbitrarily large leaps resulting in immediate detection, this insight resulted in further investigation into the role of the EKF and the process of fusing GPS data with Inertial Measurement Unit sensor outputs. It became clear that the EKF is designed to filter out sudden, unrealistic changes by comparing different sensor outputs every 100 ms and triggering safety mechanisms in case of accumulation of discrepancies which exceed the predefined threshold. This led the attack implementation to adopt a strategy of gradual, incremental offsets that mimic natural sensor noise, thereby gaining stealth.

First Implementation: In the first implementation, we will be using pre-built parameters called `SIM_GPS_GLTCH_X` and `SIM_GPS_GLTCH_Y`, each of which adds a specific pre-determined offset to the GPS data being sent to the flight controller. We will keep track of each offset applied and apply the offset to the spoofed location, which will cause the effect of *ArduCopter* chasing the spoofed perceived location. The script will be run while the UAV is in *Auto* and *Guided* modes, both of which rely heavily on GPS.

Second Implementation: The second implementation involves using a pre-built module within the Ardupilot suite called `fake_gps`. This will allow us to inject spoofed GPS data via MAVLink communication. To force Arducopter to use MAVLink as its communication protocol we use the command `param set GPS1_TYPE 14`. We will be modifying the script to integrate the controller API and make minor adjustments.

Step by step instructions on how to reproduce these attacks have been included in this project's repository README file, the link can be found in the footnote[6].

4.1 EKF and Offset

To carry out a successful GPS spoofing attack against consumer drones, we first need to have a good understanding of the fail-safe mechanism in place. The Extended Kalman Filter plays a crucial role in this, by monitoring and keeping track of discrepancies between the GPS data and IMU sensor outputs, it triggers a safety measure if it exceeds a pre defined threshold.

By fusing the IMU output with the GPS data, the EKF algorithm predicts velocity, position, and magnetic field. IMU sensors inherently have errors, which the EKF algorithm takes into account. For instance, the accelerometer has a constant bias error which causes a position error that accumulates over time. The gyroscope has angle random walk error, it will exhibit high-frequency y white noise due to thermoelectric reactions. Due to these errors, EKF has to count for them. It does this by comparing the data every 100ms and a fail-safe will be triggered if the error count exceeds 10. If this threshold was not implemented, the EKF algorithm would constantly have false positives and trigger fail-safes. The accuracy of the sensor outputs has a large influence on the effectiveness of the EKF and different sensors on different drones exhibit varying levels of accuracy. Therefore, different drones have different degrees of errors and this needs to be taken into consideration for the most effective EKF.

An attacker can exploit this tolerance by injecting spoofed GPS data that fall in line with the threshold accepted by the EKF. This allows us to inject fake GPS data that seem normal to the EKF, thus, deceiving it to believe that it is at a certain position triggering a path correction in the attacker's desired location.

Prior research suggested that Arducopter changes flight mode on activation of a fail-safe, however, this was not the case for our first implementation and we will dive deeper on to the reasons why. The second implementation,

[6] https://github.com/Botii/gpsSpoofingProject

however, aligns with the findings from prior research and Arducopter did switch to Land mode on the trigger of a fail-safe.

4.2 First implementation

Preliminary experiments related to the second implementation, first implementation and the controller feature are detailed in the Appendix. Please refer to the Appendix for additional background and supporting information on these early experiments.

Results from the first implementation showed that the path-following algorithm detected a deviation, and the Extended Kalman Filter (EKF) switched sensor reliance from GPS to IMU. A lane-switch corrective action was observed seven times during the flight before reaching the waypoint. This is what we call a soft failure, when the EKF switches from relying on GPS to IMU sensors without detecting a critical error sufficient to trigger a full fail-safe. Only one sensor became unreliable, while others, such as the IMU, remained a reliable alternative for maintaining an accurate state estimate. Even though the EKF relied more on the IMU sensors, the drone carried on being spoofed. I believe this is a simulation bug, as the GPS data still had enough weight to alter the state estimate.

4.3 Second implementation

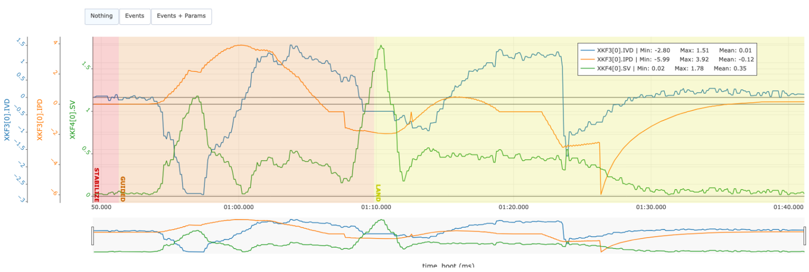


Figure 2: Large offset being detected by the EKF

In our first experiment, we applied very large offsets, which immediately caused the EKF to trigger a fail-safe. For example, applying a sudden offset of 0.0001 was quickly recognized by the EKF, resulting in a mode switch to landing. As shown in Figure 3, the green line (square root of the velocity variance), blue line (innovation in velocity), and orange line (innovation in position) all exceed the EKF threshold of 0.8. This triggers a fail-safe, causing the copter to switch to land mode and landing. In contrast, Figure 4 illustrates a normal flight in guided mode without any spoofing, the values are much lower with a maximum of only 0.08 compared to a peak of 3.9 during the spoofed flight in Figure 3.

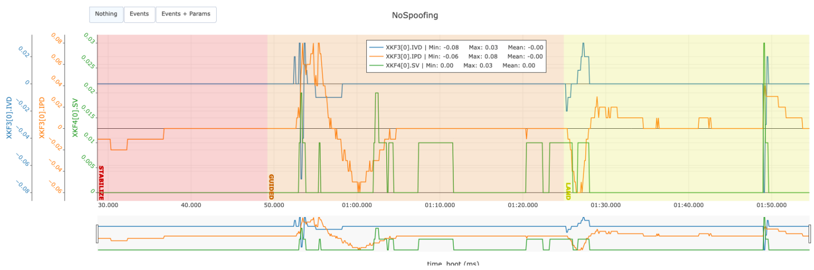


Figure 3: Regular flight with no offsets

The EKF threshold can be adjusted via `FS_EKF_THRESH parameter`, the EKF's fail-safe action can also be changed via `FS_EKF_ACTION parameter`. Land fail-safe action may not always be the best option, an alternative can be AltHold.

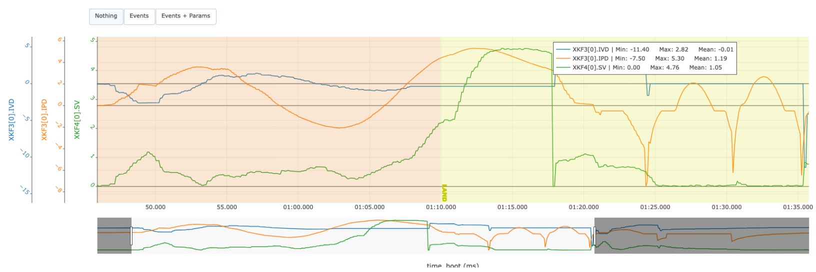


Figure 4: Incremental offsets

This initial result prompted further investigation into reducing the mean of the EKF values to similar values seen in Figure 4. One approach is using incremental small offsets, with an incremental offset of 0.000001 we managed to reduce the mean to 1.19 allowing the attacker to spoof the copter some distance before a trigger of a fail-safe. This gradual increase removed sudden spikes and maintained control of the copter until the variance eventually reached the threshold, at which point the EKF triggered a fail-safe as seen in Figure 5.

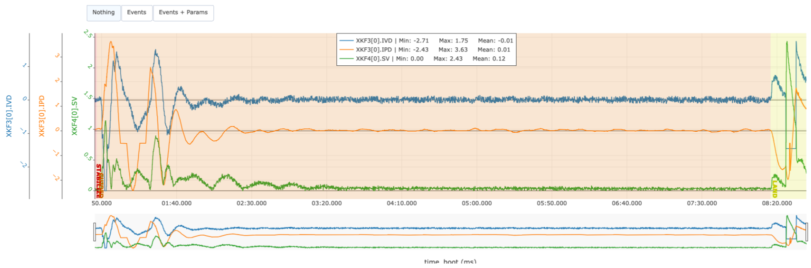


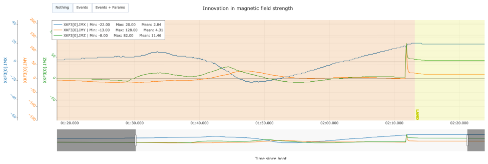
Figure 5: Applying offsets incrementally and in bursts

Further investigation led the approach to applying the offsets in bursts to remove the cumulative error. Through trial and error with various burst interval, we determined that burst intervals of 1.8 provided the best stealth performance. As shown in Figure 6, none of the three monitored EKF parameters exceeded the threshold at any point. This allowed us to achieve a mean value similar to Figure 4, the mean remained below 0 during the whole flight. This approach effectively hijacks the arducopter until battery depletion. However, the limitation is that this technique only works when modifying the latitude.

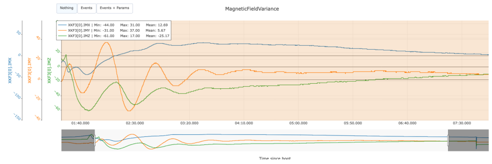
This is caused by the limitation of the magnetometer. Near the north and south poles, the Earth's magnetic field lines become nearly vertical. It relies on detecting variations in the horizontal component of the magnetic field to detect movement. When the copter is travelling in the north-south direction, the horizontal variance is minimal because the field lines are almost parallel. The magnetometer is unable to accurately calculate its relative position or detect movements, making it less sensitive to spoofing in these directions.

The pre-defined magnetic field variance threshold is set to 50 by default [7].

[7] https://ardupilot.org/copter/docs/parameters-Copter-stable-V4.5.7.html



(a) Magnetic field error threshold exceeded, spoofed towards east



(b) Magnetic field variance staying below the threshold, spoofed towards south

Figure 6: A side by side comparison of magnetic field variance varying based on direction of spoof

When this threshold is exceeded, a fail-safe is triggered. As shown in Figure 7a, the magnetic field innovations in both the X-Y and Z axis exceeds the threshold of 50, triggering a fail-safe. In contrast, when spoofing the copter in the southward direction, we can see from Figure 7b the magnetic field innovations never exceeds the threshold. This discrepancy occurs because the magnetometer is unable to accurately determine its relative position near the poles due to the nearly vertical orientation of the magnetic field. The lack of significant horizontal variation means that the EKF does not register a spoofing event in these conditions.

4.4 Controller

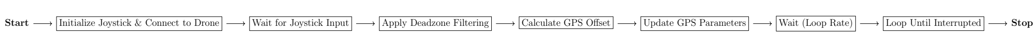


Figure 7: Flowchart of the GPS spoofing script with controller-based offset

Due to the success achieved from the second experiment in the appendix without the controller, I implemented the controller integration to determine if we can control the copter within a radius of 97.98 m without detection by the EKF.

I created a new function called `get_controller_input` which is responsible for retrieving the X and Y axis from the controller API. I applied a dead-zone to prevent unwanted drift from joystick noise, this is necessary as

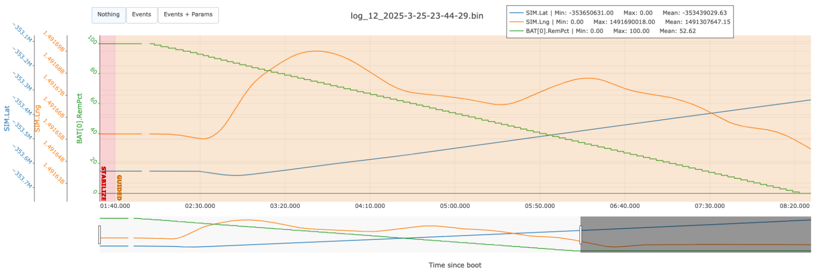


Figure 8: Latitude and longitude changes over time

it causes the EKF to trigger a compass error. I've also reduced the applied offset scale from 0.000001° to 0.0000005°. The script continuously adds latitude and longitude offsets based on the real-time controller input, allowing dynamic control of the spoofed location while maintaining stealth.

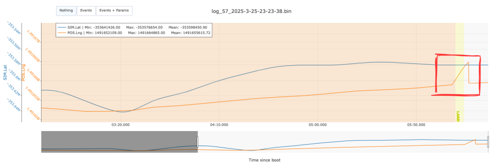


Figure 9: Sudden change in direction on the joystick causing fail-safe

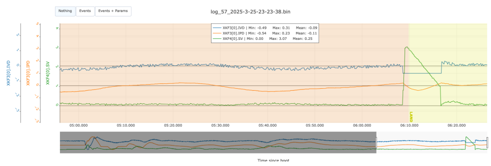


Figure 10: EKF values exceeding the threshold

I ran this script several times with varying joystick input. In the first experiment, sudden changes in the joystick input caused the EKF to detect the abrupt offsets. As shown in Figure 9 (red box), a sudden increase in longitude triggered a switch to land mode, a fail-safe response by the EKF. Looking at the EKF values for the same flight, which can be seen in Figure 10, the sudden increase in longitude caused the velocity variance to exceed the threshold of 0.8.

This occurs due to the controller-induced lag, during which the offsets accumulate, when the GPS data is finally processed, the accumulated offset is applied abruptly rather than as gradual changes.

However, with smooth joystick movement, gradual and consistent offsets were applied to latitude and longitude, as shown in Figure 8 enabled us to successfully take full control of Arducopter via GPS spoofing. From the take-

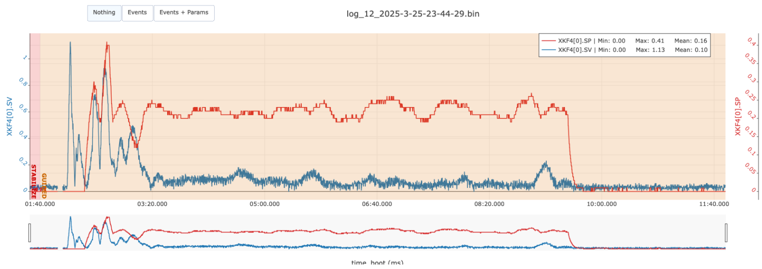


Figure 11: EKF values staying under the threshold

Figure 12: Achieving 3,189 m distance with GPS spoofing attack

off point until the UAV crashed, we managed to reach a distance of 3,180.4 m or 3 km. As shown in Figure 11, the EKF values stay below the threshold of 0.8 during the whole period of the attack.

5 Results & Analysis

In the first implementation, we were able to partially hijack arducopter, eventually triggering an EKF lane switch when a specific distance was reached. However, even though multiple lane switched would occur, we were still able to spoof Arducopter. I believe that this is a simulation bug, as the GPS data still held enough weight for the EKF to produce spoofed state estimates.

In the second implementation, we successfully spoofed Arducopter in the north and south direction until battery depletion using the burst technique. However, any other direction caused the EKF to trigger a fail-safe once a threshold was reached.

The second implementation with the controller feature (not using the burst technique), allowed the UAV to be fully hijacked without triggering any safety mechanisms if the attacker maintains smooth joystick movement eliminating sudden spikes in EKF values.

6 Discussion

6.1 Countermeasures

In this section we will be exploring the different ways in which we can defend against GPS spoofing attacks. I will be considering both technical solutions which will be built into the drones and higher-level policies and practices.

6.2 Technical countermeasures and defences

The most obvious answer would be to utilise cryptography and encrypt all GPS signals. This is not common practice in consumer drones as the cost of implementation is quite high and there is also processing time needed. This will prevent GPS spoofing attacks but still would be vulnerable to jamming.

Even though arducopter did utilise IMU sensor fusion with GPS, we were still able to go undetected. On top of this, we can incorporate an additional sensor which will make the flight controller less reliant on GPS. For example, we can incorporate a camera as a visual odometry system which can track the drones movement in relation to the horizon line by comparing different frames provided by the camera. If the GPS data is indicating that the drone is moving but the frame hasn't changes and the horizon level is the same the system will flag it as an inconsistency and reject the data. Now with multiple cross-checks, this makes it much more difficult for an adversary to fool all the sensors.

If the drone has a data link, it can take use of external API's to validate its position. For example, by using googles Geolocation API the drone can send pings to multiple different cell towers, and the algorithm will be able to calculate its location to a very good accuracy. While this may induce additional complexity and relies on a good connection, this will act as a second layer of verification.

6.3 Operational and policy strategies

Many of the drones on the market already have a geo-fence databases such as airports, military bases and prisons. Drone manufacturers need to have a up to data no fly zone and a pre defined behaviour in case of a GPS loss. This can reduce the impact caused by attacks. For example, if an adversary lures the drone to an airport, a well defined no fly zone will get in the way of travelling to such place.

Integration of spoofing detection in all UAV firmware as a standard will raise the baseline security of drones. Making anti-spoofing features incorporated in all drones can narrow the window of opportunity for attackers. This can include; standardised anomaly detection libraries or a certification requirement that a drone is resilient to basic GPS spoof scenario.

Human operators can be put in the loop as a countermeasure. Organisation can make human operators an operational protocol and their responsibility would be to monitor telemetry. If a drone starts to behave oddly or it goes off course, the human operator can intervene and manually take control. While this may not stop spoofing attacks, it provides situational awareness so that the right steps can be taken.

Even though the UK government outlaws unauthorised signal transmission and jamming, not all countries have such strictly enforced laws. For example, Ukraine is one of the few countries in the world that has legalised the use of jammers. Funnily, the reason was so that they can be used to stop students from using mobile phones during exams Phantom Technologies (2023).

Regulations alone cant stop a determined attacker, however, it can push towards safer designs and ecosystems. By combining both technical defences with operational policies, we can reduce the risk of GPS spoofing attacks substantially. In the next section we will be going through adversaries can benefit from GPS spoofing attacks.

6.4 Future works

In future work, I plan to explore this experiment on real hardware, potentially within an anechoic chamber to minimise signal interference and still maintaining within safety and ethical constraints. Additionally, I aim to refine the controller implementation to incorporate the burst technique we explored. However, given the existing delay, incorporating bursts could further complicate control.

I also intend to revisit the preliminary experiments to address the issues encountered with the script and the USRP B210. This will allow us to explore generating spoofed signals and simulating the communication based on the captured traces, which are crucial for achieving a realistic real-world attack conditions. Additionally, we will also explore the limitations imposed by rate of trace and the impact of environmental conditions.

The effectiveness of different countermeasure strategies can also be investigates in future work. The countermeasures can also be tested within the simulation environment we explored in this project.

7 Conclusion

GPS spoofing attacks against UAV's present a very serious and critical security threat when abused by malicious actors. Our projects success in hijacking a drone in a safe hardware free environment shows how adversaries can use GPS spoofing of consumer UAV's as a side-channel remote control method.

7.1 Adversarial threats and abuse scenarios

- Sabotaging delivery drones: An adversary can either a steal a payload or simply just prevent a successful delivery by making the UAV go of course. For example, a competitor might spoof the drone off its pre defined route, causing the drone to misdeliver or crash. This can have reputational damages but also a possibility of getting your path/route license revoked. This can cause a considerable amount of damage to the customers trust of delivering parcels.

- Public safety: because GPS spoofing attacks allows us to lure the drone to anywhere the attacker would like, an adversary can lure the drone to a restricted or dangers airspace. For example, a malicious actor can spoof a hobbyist drone to an airport such as Heathrow and cause disruption or even accidents. We have seen in the past how a drones sighting can halt all departures. In 2019, Heathrow airport was temporarily stopped after a drone was reported to be seen BBC News (2019).

- Attacks on critical infrastructure: drones that are being utilised for inspecting infrastructure or by police can be spoofed to crash into critical infrastructure such as power lines and bridges causing severe impact. An adversary could also break formations of swarms by simply making one drone collide with another without even requiring a large offset.

The application of GPS spoofing go beyond negative purposes, they also have defensive and ethical applications.

- Safe-hijacking as threat mitigation: The concept of safe-hijacking can be seen as an anti-drone defence method. Authorities can deliberately spoof a rogue UAV and take remote control to be able to safely make it land for further analysis. Simply jamming the rogue UAV can have the opposite effect as the behaviour of the drone becomes unpredictable and cause more damage. Noh et al. (2019) successful carried out a adaptive GPS spoofing attack against commercial drones and it showed that they can for the UAV to move in any direction with high accuracy in real world conditions. Our simulation proved this possible even further. With this accuracy, authorities can escort a rogue UAV out if harm's way.

- Testing and training for resilience: This safe simulation-based approach provides drone developers and researched a valuable tool to test their products. UAV manufactures can evaluate their drones robustness and fail-safe mechanisms under a GPS spoofing attack. For example, arducopter developers can adjust the EKF parameters or fine-tune the detection algorithm and run the spoofing scripts to see if the drone is still vulnerable to GPS spoofing attacks.

In conclusion, the project "Simulating GPS-spoofing of consumer UAVs as a side-channel remote control method" has illustrated the feasibility of GPS-based UAV hijacking, as well as to why it is important to guard against such attacks. While this research may not perfectly replicate the real-world conditions, it has demonstrated the drones behaviour under fake GPS data and has highlighted bypass techniques for the Extended Kalman Filter algorithm.

In order for such an attack to be executed successfully in the real world, the attacker would require specific technical equipment, detailed knowledge of drones flight controller and a malicious motive also including the factors of timing, rate of trace , signal precision and environmental conditions. Nevertheless, this project has effectively demonstrated the inherent vulnerability in UAV GPS navigation systems and has provided valuable insight and tools that can guide future developments in drone security and defensive measures.

7.2 Validation Framework for Enhancements

This simulation framework not only exploits the vulnerabilities in drone navigation under GPS spoofing but it can also be used to validate modifications to the navigation algorithm and sensor integration like the EKF. It enables developer to test incremental EKF adjustments and sensor fusion improvements, directly addressing the threats identified in previous sections.Researchers can benchmark different hardware upgrades like using better quality sensors and enhancements to the EKF algorithm to decide whether algorithm can be developed that allows the use less accurate (legacy) sensors with improved efficacy when used as part of a enhanced EKF. This tool allows these theories and designs to be tested more rigorously.

References

BBC News (2019). Gatwick drone attack possible inside job, say police. `https://www.bbc.co.uk/news/uk-46803713`, accessed: 2025-03-26.

Gupta, S., Patnala, S. K., W, W., Agrawal, N., Tripathi, N., N, P. K. B. and Raman, B. (2023). Gps spoof and detect in ardupilot simulating uavs. In *Proceedings of the 21st OITS International Conference on Information Technology (OCIT 2023)*, IEEE.

Humphreys, T. E., Ledvina, B. M., Psiaki, M. L., O'Hanlon, B. W. and Kintner, P. M. (2012). Assessing the spoofing threat: Development of a portable gps civilian spoofer. *IEEE Journal of Selected Topics in Signal Processing*, 5(3), pp. 574–581.

Noh, J., Kwon, Y., Son, Y., Shin, H. and Kim, Y. (2019). Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing. *ACM Transactions on Privacy and Security*, 22(2), pp. 12:1–12:26.

Phantom Technologies (2023). Legality of Jammers. Accessed: 2025-03-26.