

## GPS Spoofing on UAV Simulation using Ardupilot

David Li, University of Maryland College Park

Prof. Houbing Herbert Song, University of Maryland Baltimore County

Houbing Herbert Song (M'12–SM'14–F'23) received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, in August 2012.

He is currently a Professor, the Founding Director of the NSF Center for Aviation Big Data Analytics (Planning), the Associate Director for Leadership of the DOT Transportation Cybersecurity Center for Advanced Research and Education (Tier 1 Center), and the Director of the Security and Optimization for Networked Globe Laboratory (SONG Lab, [www.SONGLab.us](http://www.SONGLab.us)), University of Maryland, Baltimore County (UMBC), Baltimore, MD. He has been the Founding Chair of Trustworthy Internet of Things (TRUST-IoT) Working Group within IEEE IoT Technical Community since 2024. He was a Distinguished Visiting Fellow of the Scottish Informatics and Computer Science Alliance (SICSA) in 2024. He is currently the Co-Editor-in-Chief (Co-EiC) of IEEE Transactions on Industrial Informatics (2025-present). He serves as an Associate Editor for IEEE Transactions on Artificial Intelligence (TAI) (2023-present), IEEE Transactions on Intelligent Transportation Systems (2021-present), and IEEE Journal on Miniaturization for Air and Space Systems (J-MASS) (2020-present). He was an Associate Technical Editor for IEEE Communications Magazine (2017-2020) and an Associate Editor for IEEE Internet of Things Journal (2020-2024). He is the editor of 10+ books, the author of more than 100 articles and the inventor of two patents. His research interests include AI/machine learning/big data analytics, cyber-physical systems/internet of things, and cybersecurity and privacy. His research has been sponsored by federal agencies (including National Science Foundation, National Aeronautics and Space Administration, US Department of Transportation, and Federal Aviation Administration, among others) and industry. His research has been featured on popular news media outlets, including IEEE Spectrum, IEEE GlobalSpec's Engineering360, IEEE Transmitter, insideBIGDATA, StateTech Magazine, Association for Uncrewed Vehicle Systems International (AUVSI), Security Magazine, CXOTech Magazine, Solutions Review, EdTech Magazine, PYMNTS, Fox News, U.S. News & World Report, The Washington Times, and New Atlas. Dr. Song is an IEEE Fellow, an ACM Distinguished Member, and a Full Member of Sigma Xi. Dr. Song has been a Highly Cited Researcher identified by Web of Science since 2021. He is an ACM Distinguished Speaker (2020-present), an IEEE Computer Society Distinguished Visitor (2024-present), an IEEE Communications Society (ComSoc) Distinguished Lecturer (2024-present), an IEEE Intelligent Transportation Systems Society (ITSS) Distinguished Lecturer (2024-present), an IEEE Vehicular Technology Society (VTS) Distinguished Lecturer (2023-present) and an IEEE Systems Council Distinguished Lecturer (2023-present). Dr. Song received Research.com Rising Star of Science Award in 2022, IEEE 2021 Harry Rowe Mimno Award, and 10+ Best Paper Awards from major international conferences. He has been an IEEE Impact Creator since 2023.

# Teaching GPS Spoofing on UAV Simulation using Ardupilot

## Abstract

In this paper, we show a technique which can be employed in simulating GPS spoofing attacks through the use of Ardupilot, which is an open-source software allowing for the simulation of UAV (Unmanned Aerial Vehicles). As concerns about GPS spoofing are raised, the harm it poses to navigation reliant systems and applications become an increasingly pressing issue. In order to resolve this problem, it is a necessity to understand how a GPS spoof works in practice and learn how to combat it through testing. Our simulated framework will greatly assist in this endeavor. In our methodology, we utilize the Ardupilot simulation system and the MAVProxy system to artificially send GPS data to the UAV's navigation system, essentially simulating a GPS spoof. We created a Python script to perform this GPS injection and make the UAV fly in a circular motion. The aim of the proposed system is to be able to operate in a virtual environment, thus reducing required hardware, and minimizing the risks that come with testing actual UAV systems. Throughout the progress of the project, a GPS spoofing script was created, incorporated into the ArduPilot simulation system, and tested on how effectively it is able to affect the movement of the UAV. This simulation will help in creating a framework to help detect a GPS spoof attack as well as allow us to understand the effects of a GPS spoof on a UAV. In addition to this, we evaluate the extent to which a remote attacker is able to exert control over a UAV through GPS spoofing.

## Introduction

GPS technology has become essential for the functioning of numerous autonomous systems such as drones, self-driving cars, and other vehicles. These systems heavily depend on precise GPS data to navigate and carry out their tasks. However as of recent, the increased reliance on GPS has brought about notable security vulnerabilities and concerns. There are a variety of cybersecurity attacks which specifically target the GPS system, such as denial-of-service, man-in-the-middle, and GPS jamming attacks. These attacks can heavily disrupt the functionality of the vehicles and result in potentially disastrous outcomes. Researching these attacks through rigorous experimentation is vital for researchers to understand these attacks and prepare proper countermeasures.

For our research, we primarily focused on GPS spoofing in relation to UAVs (Unmanned Aerial Vehicles). GPS Spoofing happens when a hacker creates false GPS signals to deceive a GPS receiver into displaying an incorrect location. This could cause the UAV to veer off track, head to an incorrect location, or even crash, resulting in devastating outcomes in crucial areas such as aviation, military activities, and public infrastructure, where accurate positioning is essential.

While GPS spoofing is a known vulnerability, conducting real-world experiments to explore its impacts and develop countermeasures is challenging due to the risks and high cost associated with broadcasting counterfeit signals. Therefore, we chose a simulation-based method, which has become a vital tool for researching and testing GPS spoofing on UAVs in a safe and controlled environment. The ability to simulate GPS spoofing allows us to study how navigation systems on UAVs respond to GPS spoofing attacks and to develop detection and mitigation strategies

without interfering with actual GPS signals. Therefore, we created a simulated framework which can effectively simulate a GPS spoof. The GPS spoof will actively hijack the UAV and take control of the system, causing it to follow a different path than intended. This simulation will allow us to educate researchers on how GPS spoofs occur.

In the remainder of this paper, we will first discuss background information to properly define a GPS spoof and discuss previous work. Next, we'll introduce the development process in which we explain the entire process that we went through to create our simulation, including the python script that we created as well as technical errors that arose. Following this, we'll explain our results and the observations for the simulation. Finally, we'll conclude and summarize our results and their implications.

## **Background**

The Global Positioning System (GPS) is a navigation system which makes use of satellites to transmit location data to users, allowing them to find their position. This system has become paramount for both critical infrastructure and civilian use alike. One of the most common uses for GPS is in vehicles, such as cars, planes, and drones, where the GPS navigation system is vital for proper functioning and pathfinding of the vehicle. However, the dependency of vehicles on GPS proves to be a critical vulnerability in the face of GPS spoofing attacks.

GPS spoofing is a form of air-traffic deception where fake signals are sent to fool a GPS receiver into believing it is at a different location or to be traveling along a different path than it actually is. This causes the vehicle to store inaccurate location data, which could result in various detrimental effects for the UAV's flight. GPS spoofing works by taking advantage of the GPS satellite's weak signals and overriding it with falsified signals. This results in the vehicle receiving inaccurate location data and thus being fooled. As mentioned in the introduction, GPS spoofing has a multitude of negative implications. GPS spoofing could cause a UAV to veer off course, reveal the location of a UAV to an attacker, or mislead the operator of the UAV about the current location of a vehicle. Thus, experimentation on what a GPS spoof looks like, as well as what preventative measures can be taken against a GPS spoof must be done. We aim to better understand the effects of a GPS spoof on a UAV through our simulated framework.

Early research on GPS spoofing was done by Humphreys et al. who conceptualized the idea of GPS spoofing. His work demonstrated that GPS signals could be manipulated to mislead the receiver. He performed an experiment on a UAV, successfully spoofing it and causing it to take a different path than intended. This highlighted the high level of vulnerability that UAVs are exposed to, and raised many concerns about the security of GPS technology inside vehicles. His research opened up further exploration on the topic of GPS spoofing.

A paper done by Kerns et al. performed various experiments to determine the necessary conditions for UAV capture as well as the range of control possible following a capture. They found that attackers can effectively and reliably gain control of a UAV through GPS spoofing. In addition to this, they found that attackers can use GPS spoofing to force the UAV to follow a path that they desire. This sets a precedent for our research, which simulates a forced path

through GPS spoofing, and also highlights the extent to which GPS spoofing has the ability to take control of a UAV and effect its functionality.

Another paper by Mendes et al. studies the effects of a GPS spoof on UAVs in regards to its deviation from the original trajectory and attack success. They created a simulation framework similar to ours, using Ardupilot and MAVProxy to perform the simulation. They found that attempts to take control of or even crash a simulated Quadcopter were largely successful. This underscores the necessity to perform further simulation and study the ability for GPS spoofs to affect UAVs.

Various protective measures have been studied as a response to these studies revealing that GPS spoofing is a major concern for UAVs. One of the key methods developed is the use of multi-receiver setups to detect inconsistencies in the signals received from GPS satellites. By comparing signals from multiple GPS receivers, spoofing attacks can be identified when discrepancies in the location data appear. For instance, Tippenhauer et al. investigated the conditions under which GPS spoofing attacks can succeed and suggested that placing receivers in different physical locations helps detect spoofed signals because the false signals would arrive simultaneously at multiple receivers, unlike legitimate signals.

More recently, machine learning techniques have been applied to spoofing detection. In one study, Bose et al. used machine learning algorithms to analyze the statistical features of GPS signals and detect subtle changes introduced by spoofed signals. This method has shown promise in identifying spoofing attempts in real-time, with a high accuracy for correct classification.

## **Development Process**

We will now detail the entire process of creating the simulation, from installation to testing. This is done so future researchers who may attempt to perform similar experiments can guide themselves through the process smoothly and with as minimal issue as possible.

We first proposed to use a Gazebo-based simulation using a custom UAV model. This would all be done on the Linux operating system. However, we found that designing a custom model for the Gazebo was extremely difficult and complex, and because our project is focused on the cybersecurity / GPS spoof aspect of the simulation, we decided to pivot towards using a pre-existing model in order to simulate. We chose Ardupilot as the solution, which contained all the necessary predesigned models.

At first, we attempted to use Gazebo in tandem with Ardupilot, but various difficulties cropped up when this was attempted. We encountered many unix version errors, with our version of unix being incompatible with Gazebo. Thus we had to reconfigure our unix setup to fix these compatibility errors. After, due to various file path errors resulting from a mismatched home directory variable, we attempted to switch to a virtual machine environment to have a fixed unix setup and resolve the file path errors. However, this did not succeed, as Gazebo refused to run on the virtual machine. Some research led us to the conclusion that virtual machines lack the processing power to run Gazebo properly.

After trying and failing to fix the problem by adjusting the parameters on the virtual machine, we decided to abandon this altogether and instead opt for a dual boot with Linux. However, this also proved unsuccessful as Gazebo again refused to work even in the Linux OS, and various partitioning issues forced us to adjust our solution once more.

We finally turned to using WSL to install a Linux distribution and perform the simulation. In the process of finding these solutions we also dropped the use of Gazebo, as Ardupilot has its own simulation software known as Software In The Loop (SITL) which facilitates independent simulation without the need to incorporate external software.

Using this new approach, we were able to successfully simulate a GPS spoof on a UAV, specifically the ArduCopter model provided by the ArduPilot repository. We preemptively simulated a vehicle without the GPS spoof to confirm that the simulation did indeed work. Then, we created a python script which injects GPS coordinates to the navigation system.

The script begins by establishing a connection to the vehicle. The script then defines a method which takes parameters of latitude, longitude, and altitude. It creates a message containing the necessary GPS data based on the latitude, longitude, and altitude, then sends the message through MAVProxy.

The script follows this by generating parameters for a circular motion and repeatedly runs the above method with a loop and continuously increments the angle of the flight in order to maintain a circular path. It then sends these parameters to the UAV, effectively simulating a GPS spoof attack where the attacker overtakes the UAV.

```
master = mavutil.mavlink\__connection__('udp:127.0.0.1:14551')

def send_gps(latitude, longitude, altitude):
    # Create a new message with the GPS data
    msg = master.mav.gps_input_encode(
        0, 0, # Time of week and fix type
        0, # GPS time
        latitude, longitude, altitude, # Latitude, Longitude, and Altitude
        0, 0, 0, # Horizontal and vertical speeds, GPS ground speed
        0, 0, # Horizontal and vertical speed accuracy
        0, # Number of satellites visible
        0, 0 # H/V accuracy
    )
    # Send the message
    master.mav.send(msg)

# Parameters for circular motion
center_lat = 37.7749 * 1e7 # Center latitude center_lon = -122.4194 * 1e7 # Center longitude
radius = 0.0001 * 1e7 # Radius of the circle
altitude = 10 * 1e2 # Altitude
angle = 0

# Spoofing loop
while True:
    # Calculate new coordinates
    spoofed_lat = center_lat + radius * math.cos(math.radians(angle))
```

```

spoofed_lon = center_lon + radius * math.sin(math.radians(angle))
send_gps(spoofed_lat, spoofed_lon, altitude)
# Increment angle for circular motion
angle += 10 if angle >= 360: angle = 0
time.sleep(1)

```

To run our simulation as a whole, first install WSL. Then, follow the instructions to set up the build environment using WSL in the Ardupilot documentation. After this is finished, cd into the ardupilot/Arducopter directory, and run the simulation using

```

../Tools/autotest/sim_vehicle.py --map
--console

```

Three windows should appear: a command prompt to enter commands to, a console displaying the status of the vehicle, and a map which shows the vehicle in its environment. In the console, enter the following commands to allow the copter to take flight:

```

mode guided
arm throttle
takeoff 40

```

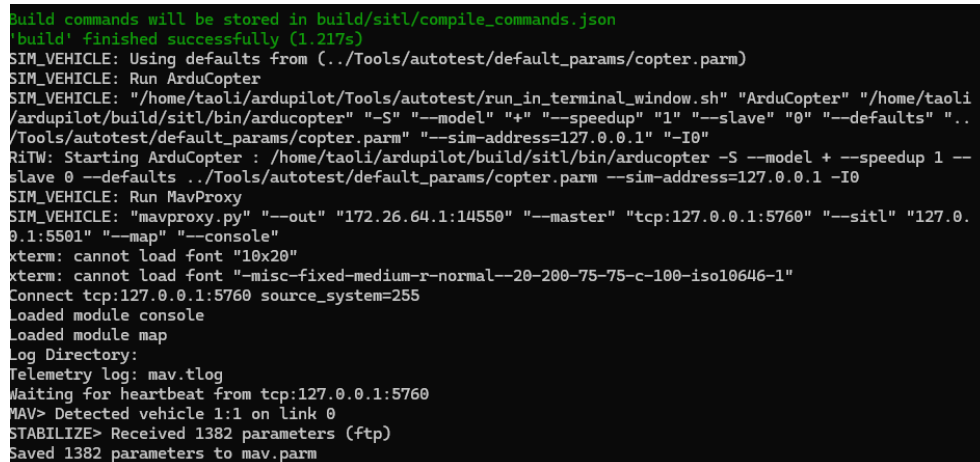
Next open another terminal and run this command:

```

mavproxy.py --master=127.0.0.1:14550 --out=127.0.0.1:14551

```

This will connect MAVProxy to the Ardupilot SITL instance and allow for the transmission of GPS spoofing data. Finally, we'll run the python script which was included above. This will start the GPS spoof, and the results will be observable on the map.



```

Build commands will be stored in build/sitl/compile_commands.json
'build' finished successfully (1.217s)
SIM_VEHICLE: Using defaults from (../Tools/autotest/default_params/copter.parm)
SIM_VEHICLE: Run ArduCopter
SIM_VEHICLE: "/home/taoli/ardupilot/Tools/autotest/run_in_terminal_window.sh" "ArduCopter" "/home/taoli/ardupilot/build/sitl/bin/arducopter" "-S" "--model" "+" "--speedup" "1" "--slave" "0" "--defaults" "..../Tools/autotest/default_params/copter.parm" "--sim-address=127.0.0.1" "-I0"
RiTW: Starting ArduCopter : /home/taoli/ardupilot/build/sitl/bin/arducopter -S --model + --speedup 1 --slave 0 --defaults ../Tools/autotest/default_params/copter.parm --sim-address=127.0.0.1 -I0
SIM_VEHICLE: Run MavProxy
SIM_VEHICLE: "mavproxy.py" "--out" "127.26.64.1:14550" "--master" "tcp:127.0.0.1:5760" "--sitr" "127.0.0.1:5501" "--map" "--console"
xterm: cannot load font "10x20"
xterm: cannot load font "-misc-fixed-medium-r-normal--20-200-75-75-c-100-iso10646-1"
Connect tcp:127.0.0.1:5760 source_system=255
Loaded module console
Loaded module map
Log Directory:
Telemetry log: mav.tlog
Waiting for heartbeat from tcp:127.0.0.1:5760
MAV> Detected vehicle 1:1 on link 0
STABILIZE> Received 1382 parameters (ftp)
Saved 1382 parameters to mav.parm

```

**Fig. 1. Output of running simulation**

To generate Fig. 3 (shown below), the computed difference between spoofed and expected coordinates, I stored the latitude and longitude coordinates of the altered flight path into a csv

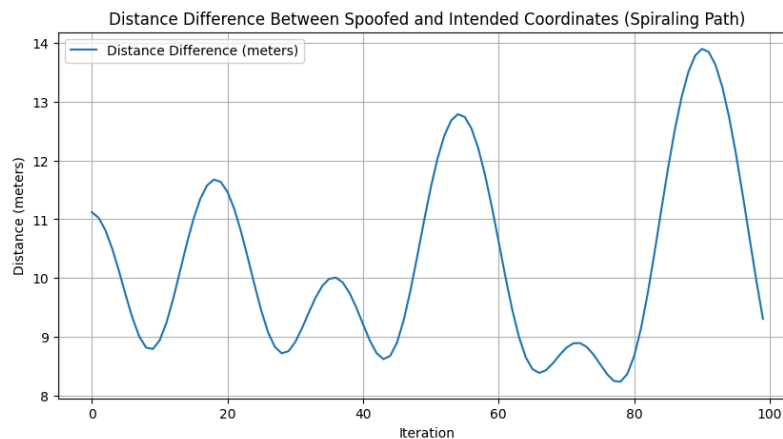
file. I stored the latitude and longitude coordinates of the intended flight path into the second column of that csv file. Then I used a python script to compute the difference between the respective latitudes and longitudes. Then, I used the haversine formula to calculate the straight line distance between the coordinates, leaving me with a list of distances. Then, I used the pyplot package to simply plot these distances on a graph.

## Results



**Fig. 2. UAV moving in the map**

As can be seen from the images, the UAV first takes off, moves in a straight line, then when the GPS spoof is sent, the UAV begins a circular path. The effect is immediate, taking place right after the GPS spoofing attack begins. The console displays that a message was received remotely, then following this the UAV begins the irregular path based on the spoofed GPS coordinates sent by the script.



### **Fig. 3. Difference between spoofed and expected coordinates**

The graph shows the difference between the spoofed coordinates and the expected straight line coordinates. As can be seen, the difference is quite stark, with the UAV continuously moving further away from the spoofed vehicle.

We can also see that the effect is extremely potent as the UAV is completely hijacked by the attacker in this situation, moving in the circular pattern and not being interfered with by the original flight commands. This is quite telling of the dangers of a GPS spoof attack on a UAV, since the attacker was able to exert complete control of the UAV and have it completely turn around.

Through this simulation, we are also able to roughly understand how a GPS spoof attack can be performed on a UAV. Through injection of GPS coordinates, the attacker can essentially fool the UAV into taking a different path than it should be.

This simulation can be used to educate others regarding how GPS spoofing works. Through our simulation, researchers can gain firsthand experience into how GPS spoofs take place. The results of this simulation also allow researchers to understand how effective GPS spoofing is and familiarize them with GPS injection and ground control systems, as well as introduce them to simulation software.

### **Review of open-source UAV simulation programs pertaining to cybersecurity simulation**

When reviewing open-source UAV simulation software programs for cybersecurity scenarios, it is important to consider several categories, including their capabilities, flexibility, and support for simulating security attacks, network vulnerabilities, and defensive mechanisms. These platforms can be used to model scenarios in which UAVs interact with various communication networks, sensors, or other UAVs, making them ideal for studying cybersecurity vulnerabilities such as in the case of spoofing.

#### *Gazebo-based ROS*

I first attempted to use Gazebo in tandem with ROS to simulate the GPS spoof. I attempted to use ROS to create my own model, but found it to be extremely complex and took an extremely long time to understand the tutorials. In a scenario where I wanted to focus on the cybersecurity aspect of the simulation, it was not ideal. Learning to configure ROS, understand its nodes and topics, and integrate it with Gazebo is not trivial. ROS tutorials often focus on basic concepts and might not quickly address advanced concepts such as cybersecurity-specific attacks. In addition, ROS has a modular structure and users often need to develop custom packages or plugins to implement more complex systems, such as GPS spoofing. Even with clear tutorials, setting up an



effective simulation might have taken months. This was problematic since the goal is to test and study specific vulnerabilities without delving into unrelated areas like the UAV's basic control systems or environment modeling.

Although ROS and Gazebo are highly flexible and can indeed simulate cybersecurity scenarios such as GPS spoofing, the process often involves a lot of customization. This requires you to modify or create custom plugins for spoofing GPS signals, message filters, and network configurations to model attack scenarios, or network traffic interception to study the vulnerabilities in communication systems. However, without a prebuilt framework specifically designed to simulate cybersecurity scenarios, such as GPS spoofing, we would end up spending too much time on the underlying system infrastructure rather than focusing directly on the attack scenario itself.

In addition to this, the Gazebo and ROS systems required a very good understanding of the unix-based operating system. The installation and the running of the simulation all was done in a complex unix environment, which is not friendly towards those without a unix background. Gazebo and ROS also resulted in various file path errors which are very difficult to resolve and may significantly hamper attempts to run the simulation. Overall, despite it being a very flexible tool, it is not very suited towards cybersecurity simulation due to its very high barriers to entry and the complexity of making modifications.

### *Ardupilot Simulation Software*

We then pivoted towards using Ardupilot, which provided built in UAV models so that we did not have to custom build one from scratch. It did share several of the same problems as ROS and Gazebo, including its heavily unix-based system and a complicated documentation, but greatly simplifies the simulation process itself, providing a prebuilt model and simple, easy to follow flight commands. It also used a MAVProxy ground control station, making it easy to simulate a spoof by sending false coordinates and forcing the UAV to take a different route. The combination of these two allowed me to focus much more efficiently and effectively on simulating a cybersecurity attack.

However, the simplicity came with a price. The lack of Gazebo integration meant that the simulation was much simpler. It was a flat 2d map with a monitoring console, rather than a 3d graphic environment like Gazebo based ROS provided. The simulation also did not consider other environmental factors like wind, obstacles, etc. making the simulation less reflective of a real life flight. It was difficult to control the flight path due to the lack of a direct control system, instead relying on a separate console to send simple commands to the flight control system. This made it difficult to perform complicated maneuvers. It was also difficult to track the various parameters of the flight, such as fuel, coordinates, altitude, etc. due to them being displayed once

more on a separate monitor window. This made running the simulation more arduous and inconvenient.

Despite these drawbacks, the convenient setup and ease of running cybersecurity tests using ArduPilot made running the simulation infinitely more efficient. It enabled a more focused analysis of the impact of GPS spoofing attacks without getting tied down by the complexity of the setup process as well as managing a more complex simulation environment with more variables to account for. Its strength lies in its simplicity, allowing us to hone in specifically on simulating cybersecurity instead of getting too caught up in the simulation of the UAV itself. However, for future studies aiming to achieve a higher degree of realism and account for environmental factors, a hybrid approach using both ArduPilot's simplicity and Gazebo's realism might strike a better balance. Ardupilot would also benefit greatly from having a more unified monitoring / control system, making controlling the simulation much more user-friendly.

### *Other software*

There are other prevalent UAV simulation systems which should also be considered in this review of the current available software.

### PX4 Software In The Loop

PX4 is an open-source flight control software widely used in the UAV community. Its ability to simulate a variety of scenarios makes it a strong contender for testing UAV systems. One of its strengths is its realistic simulation environment, since PX4 pairs very well with simulators like Gazebo. This allows for highly detailed UAV simulations where GPS spoofing or other cybersecurity attacks can be simulated realistically by spoofing GPS inputs, allowing for more detailed analysis of the UAV's response to the attack. Additionally, PX4 allows for users to modify the software or create specific scripts simulating specific GPS spoof threat scenarios. It supports external software and scripts via MAVLink, further facilitating GPS spoof simulation, and making it suitable for testing countermeasures against spoofing attacks. PX4 also provides a wide variety of drone models, allowing for versatility in the simulation.

However, PX4 does have several weaknesses when it comes to simulating GPS spoofing attacks. The simulation setup process can be complex and requires significant technical skill, making use of PX4 internals, MAVLink, and UNIX to modify GPS data or inject spoofed coordinates. Without built-in cybersecurity attack simulation software, users are forced to write separate scripts themselves, or rely on external tools to create their cybersecurity attack. This may cause the simulation to be even more complicated. Additionally, how detailed the simulation is depends largely on the abilities of the simulator. If the simulator does not have effective GPS modeling or interference simulation, the accuracy of the attack scenario may be decreased.

PX4 is a solid candidate for simulating GPS spoofing attacks on UAVs due to its flexibility and support of custom software. However, without specific cybersecurity attack simulation modules, and possessing a rather complex setup process, it demands a high amount of effort to properly set up, create, and maintain a simulation of a cybersecurity attack on a UAV. Adding built-in attack scenarios or user interfaces for simulating cybersecurity threats would greatly improve its ease of use in UAV cybersecurity research.

## MATLAB and Simulink

MATLAB and Simulink are good for modeling, simulating, and analyzing complex systems, including UAVs and their behavior under cybersecurity threats like GPS spoofing. It possesses strong computational power and a large library of modules with which it is effective for simulating and analyzing cybersecurity scenarios. One of MATLAB/Simulink's biggest strengths is its ability to simulate UAV dynamics with high precision. Libraries for aerospace as well as support for custom models allow users to simulate UAV flight behavior with high detail. The Simulink environment's graphical interface also simplifies the process of designing models and allows users to track UAV responses to cybersecurity attacks in real-time. Furthermore, MATLAB's scripting capabilities make it easy to automate simulations, inject spoofed GPS signals into the system, and analyze the results comprehensively.

However, MATLAB/Simulink is not without its weaknesses. The steep learning curve can be a barrier for new users, particularly those unfamiliar with the Simulink environment or aerospace modeling. While MATLAB offers extensive documentation, it can be dense and difficult to navigate, requiring significant time and effort to master the platform. Another drawback is its high cost, which can be prohibitive for smaller research teams or individuals, especially when compared to free, open-source alternatives like PX4 or ArduPilot. Additionally, MATLAB/Simulink lacks the immersive 3D simulation environments provided by platforms like Gazebo. While it excels in numerical precision and system-level modeling, it does not natively provide a visually rich environment to observe UAV behavior under GPS spoofing attacks. This can make it harder to analyze certain spatial dynamics or visually demonstrate the impact of spoofing in presentations or collaborative research.

In conclusion, MATLAB/Simulink is a highly capable platform for simulating GPS spoofing attacks on UAVs, particularly for researchers focused on detailed system modeling and numerical analysis. Its ability to integrate with HIL systems and other external tools adds significant value for advanced testing. However, its steep learning curve, lack of immersive visualization, and high cost may limit accessibility and usability for some users. For those seeking a balance

between realism and system-level modeling, combining MATLAB/Simulink with complementary visualization tools could be a powerful approach.

### *State of the Art Overview*

While UAV simulation tools are powerful and effective for simulating UAVs, they often struggle with specifically simulating a cybersecurity attack due to limited capabilities and difficulty of modification. Both softwares did not have built in functions to simulate these cybersecurity attacks and required separate custom modifications in order to do such. In addition, the simulation tools have very high barriers to entry, meaning they require significant unix experience to make use of them and require a significant amount of time to understand them. There is also no easy user interface with which the user can run simulations conveniently, which could also hamper less technologically adept users. This makes it very costly from a time perspective to build a simulation for a cybersecurity scenario.

I also observed a tradeoff between simulation complexity and difficulty of implementation: creating a more complex and realistic simulation requires a much more difficult implementation, while the easier-to-implement simulations sacrificed simulation complexity. Striking a balance point between these two factors is very important for effective and efficient simulation, and this tradeoff should be considered in future simulation attempts of cybersecurity attacks on UAVs. The table below compares the capabilities of the simulation software in various categories:

	Gazebo & ROS	Ardupilot	PX4	MATLAB & Simulink
Visual detail	5	2	4	3
User Interface	3	3	3	4
Parameter detail	5	4	4	5
Ease of setup	1	4	3	3
Support of external software	4	3	4	5
Documentation	4	4	3	5
Customization options	5	2	4	5
Logging tools	4	4	5	5
Scalability	5	3	4	3

**Fig. 4. A table rating each of the software on several categories from 1 (weak) to 5 (strong)**

## **Conclusion**

In this research project, we have created a simulation of a UAV in order to evaluate the ability of a remote attacker to take control of and hijack a UAV. Through observation of the UAV following the beginning of the GPS spoof attack, we were able to see that the GPS spoof attacks were effective in taking control of the vehicle and allowed for attackers to send the UAV on a completely new path.

How effective the GPS spoof is at controlling the UAV's actions is definitely concerning when it comes to a cybersecurity standpoint. Due to the reliance of UAVs on GPS systems, our results indicate that GPS spoof attacks could compromise the security of UAVs as a whole. This also has implications for other fields that use GPS navigation, including cars, airplanes, and ships, which could also potentially be compromised by GPS spoof attacks. This research highlights the needs for robust preventative measures against GPS spoofing.

There are several limitations to the study that we performed. The simulation only tested one UAV under controlled conditions. The results may very well be different if tests were performed in more complex environments. Additionally, the spoofing attack was relatively simple, with only simulation of circular motion. More sophisticated attacks, involving dynamic manipulation of GPS signals, could yield different results. Another limitation is the simulation itself, as due to it not being a real UAV, the results are limited to the scope of the simulation.

Because of the replicable simulation, this experiment can be used to help others understand how GPS spoofs function, and have them gain insight into how GPS spoof attacks operate. Additionally, the review of open source UAV simulation software gives other researchers insight into the strengths and weaknesses of the available UAV simulation software, so that if they choose to run UAV simulations themselves, they understand which software is suitable. We encountered various challenges in making this simulation suitable for education purposes. First, the simulation requires a lot of unix knowledge, rendering inexperienced users unable to utilize the simulation. Second, it is difficult to understand the results of the simulation without specifically tracking the coordinates on the monitor. These factors may hinder this experiment from being as versatile an educational tool as we would have liked.

The results of this study highlight the critical need for strengthening the security of GPS-dependent systems. Without robust detection and mitigation strategies, autonomous systems will remain vulnerable to potentially devastating GPS spoofing attacks. Finding detection methods, preventative methods, and methods to regain control of a UAV during GPS spoofing attacks will be crucial to maintain security of vehicles as a whole.

## **Acknowledgement**

This research was supported in part by the U.S. National Science Foundation under Grant No. 2309760 and Grant No. 2317117.

## References

- [1] A. Larcom and H. Liu, "Modeling and characterization of GPS spoofing," 2013 IEEE International Conference on Technologies for Homeland Security (HST), Waltham, MA, USA, 2013, pp. 729-734, doi: 10.1109/THS.2013.6699094
- [2] Kerns, A. J., Shepard, D. P., Bhatti, J. A., & Humphreys, T. E. (2014). Unmanned aircraft capture and control via GPS spoofing. *Journal of field robotics*, 31(4), 617-636.
- [3] Mendes, D., Ivaki, N., & Madeira, H. (2018, December). Effects of GPS spoofing on unmanned aerial vehicles. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)* (pp. 155-160). IEEE.
- [4] Warner, J. S., & Johnston, R. G. (2003). GPS spoofing countermeasures. *Homeland Security Journal*, 25(2), 19-27.
- [5] Tippenhauer, N. O., Pöpper, C., Rasmussen, K. B., & Capkun, S. (2011, October). On the requirements for successful GPS spoofing attacks. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 75-86).
- [6] Zhang, T., & Zhu, Q. (2017). Strategic defense against deceptive civilian GPS spoofing of unmanned aerial vehicles. In *Decision and Game Theory for Security: 8th International Conference, GameSec 2017, Vienna, Austria, October 23-25, 2017, Proceedings* (pp. 213-233). Springer International Publishing.
- [7] Seo, S. H., Lee, B. H., Im, S. H., & Jee, G. I. (2015). Effect of spoofing on unmanned aerial vehicle using counterfeited GPS signal. *Journal of Positioning, Navigation, and Timing*, 4(2), 57-65.
- [8] Khan, S. Z., Mohsin, M., & Iqbal, W. (2021). On GPS spoofing of aerial platforms: a review of threats, challenges, methodologies, and future research directions. *PeerJ Computer Science*, 7, e507.
- [9] Gaspar, J., Ferreira, R., Sebastião, P., & Souto, N. (2018, November). Capture of UAVs through GPS spoofing. In *2018 Global Wireless Summit (GWS)* (pp. 21-26). IEEE.
- [10] Humphreys, T.E. (2012). STATEMENT ON THE VULNERABILITY OF CIVIL UNMANNED AERIAL VEHICLES AND OTHER SYSTEMS TO CIVIL GPS.
- [11] Qiao, Y., Zhang, Y., & Du, X. (2017, December). A vision-based GPS-spoofing detection method for small UAVs. In *2017 13th International Conference on Computational Intelligence and Security (CIS)* (pp. 312-316). IEEE.
- [12] Bose, S. C. (2021). GPS spoofing detection by neural network machine learning. *IEEE Aerospace and Electronic Systems Magazine*, 37(6), 18-31.