Cyber-Resilient Digital Twin Framework for Detecting GPS Spoofing in PX4 Drones

Virginia Polytechnic Institute and State University

Blacksburg, Virginia

Walid Saad, Daniel Stilwell, Randy Marchany

Faculty Advisors

Alex Matthew A. Valencia

2025

A project report submitted to

the Faculty of the Graduate School of

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical and Computer Engineering

Bradley Department of Electrical and Computer Engineering

Contents

List of Figures

Abstract

With the rise of new drones being developed for various roles like reconnaissance, surveillance, logistics, FPV racing, and even cinematic reasons, drones are becoming more vulnerable to cyber threats such as Global Position System (GPS) spoofing. Therefore, with widely used autonomous drones, they have become more attractive targets for hackers to exploit for various motives. That being said, GPS spoofing attacks involve these drones being misled to an incorrect location during navigation, resulting in different losses and even mission failures. This paper will provide an overview of how a digital twin model detects these GPS spoofing attacks in PX4 drones to predict future attacks. Thus, the digital twin mirrors the behavior of a "physical" drone in a virtual environment, providing for a real-time analysis of these security threats. Moreover, upon using a machine learning framework, this model, as previously mentioned, predicts GPS spoofing attacks and implements a countermeasure to maintain the drone's security. This paper will provide a review of various test scenarios adopted for showcasing the effectiveness of the digital twin and machine learning model in PX4 drones.

1 Motivation

1.1 Drones in Military

Drones have become such an important asset in military operations, with services ranging from strikes, combat support, constant surveillance, and even gathering vital intel. As they become more critical on the battlefield, the need for different defense mechanisms against GPS spoofing is very critical. According to an article about the Department of Defense and their interest in this area, they mentioned, "The Air Force recently expanded its efforts to adopt digital twin technology with the award of a $100 million contract to Wichita State University's National Institute for Aviation Research last month" [1]. A little more into this, even the Naval Air Systems Command launched a program to use cases for digital twins and their capability with new technologies. Overall, drone usage in the military has significantly increased, raising new challenges in security and autonomous system reliability.

1.2 Driving Factor

As an engineer with a strong interest in the Department of Defense and cybersecurity, the milestone of digital twin development is an impactful step toward securing mission-critical systems. By utilizing digital twins in military defense, it is possible to analyze different attacks, such as GPS spoofing, but also test the resilience of other various cyber-threats in the field. The motivation to enhance security utilizing digital twins in the military is so intriguing because it opens the doors to critical defense mechanisms that can be used in real-time, allowing engineers, military personnel, and even analysts to observe how these drones may respond under certain conditions. In essence, my view is that the use of digital twins will and continues to be a reliable asset not only in military operations but also in the broader scope of national security and beyond.

2 Background

2.1 What is a Digital Twin?

Firstly, a digital twin is a virtual representation that reflects a physical object, person, system, or process in a digital environment 1. It can be a virtual replica that can be used to simulate its behavior to understand how it works in real life. They can help simulate situations and outcomes, model for analysis, optimization, and even facilitate maintenance.

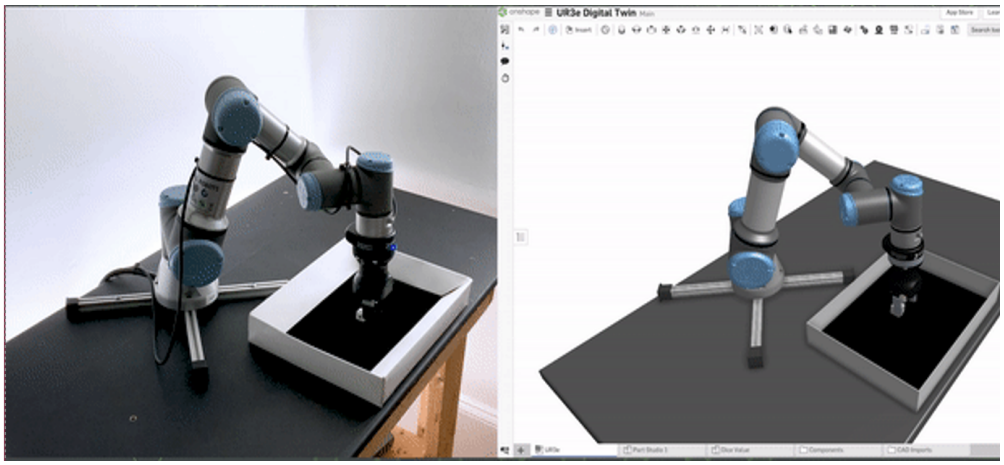

Figure 1: Digital Twin Representation

It allows the people using such as engineers, scientists, and decision-makers, to simulate, analyze, and watch the performance of the entity without having to actually work with that physical system. Moreover, the digital twin is built from various things from the object, such as its sensors, smaller components/systems, and other sources embedded within that system, to help retrieve the data to build the digital twin.

2.2 How they Work?

The way the digital twin works is that it receives continuous data from the physical object and the simulation. This is how the digital twin is primarily built. The sensors, for example, capture various information such as location data, temperature, motion, and energy use. With this data, it's transmitted to the digital twin to simulate the behavior. By analyzing this data, it can look at past patterns which can be used to help predict potential failures, look at various performances, and even look at specific areas for improvement. In some cases, it can use machine learning, artificial intelligence, and reasoning from this data, which can be used to help make decisions as shown in Figure 2.

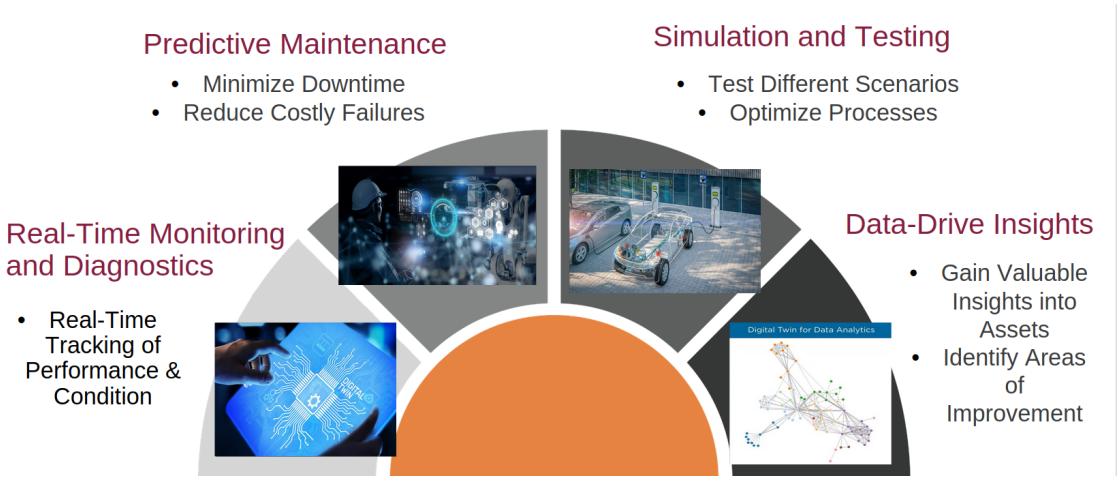

Figure 2: Digital Twin Applications

2.3 Brief History

As for the history of digital twins, the concept was first practiced by NASA in the 1960s. They used to use this idea to twin simulators to evaluate the Apollo 13's oxygen tanks. Though this was more of the idea of "digital twin" being born in this era, as it seemed to be more of a "living model" during the mission. It wasn't until 2010 did NASA officially introduced the term "digital twin" as an integrated simulation of a vehicle or system that mirrors its

physical twin. A little more into this, the motivation was to essentially monitor these complex systems without directly interacting with the physical object. In 2018, Microsoft launched Azure Digital Twins to expand the platform for digital twin development. It can be used to design a digital twin architecture in a wider cloud solution, which is an awesome touch to include IoT and cloud.

Digital twins can be used in various industries as they enable real-time monitoring, predictive maintenance, predictive analysis, and even optimization processes. They are found in manufacturing, healthcare, aerospace, architecture/construction, and even energy departments. In many of these industries, they are used for monitoring equipment, simulating systems, testing algorithms, and optimizing various functions. In the cyber realm, they are widely used to simulate and test attacks to help mitigate risks, strengthen defenses, and ensure security within cyber-physical systems.

Above all, digital twins are important because they provide a bridge between the physical world and the digital world. They open opportunities for real-time insight, predictive analysis, and system optimization. Additionally, they provide a safe environment for testing scenarios that can be too risky, expensive, or even impractical to conduct in real life. While serving all previously mentioned purposes, digital twins play a long-term role in enhancing business productivity, innovation, and supporting data-driven decisions, which lead to more efficient and adaptive systems over time. Also, digital twins support various integrations with artificial intelligence, machine learning, and IoT devices.

3 State of the Art

3.1 Digital Twins in Cyber-Physical Systems

This section reviews the state of the art, as well as the work and research that have been done in this area. Going into digital twins in cyber-physical systems, digital twins have allowed visibility into these systems' behaviors at a high scale, allowing for predictive analysis and real-time decision making. Within the robotics and unmanned aerial vehicles (UAVs) discipline, digital twins have gained a solid foundation as powerful tools for monitoring, control algorithms, and predictive maintenance. The integration of digital twins in UAV systems provides a strong basis for virtual test environments. It facilitates the communication between parts of the robot to create real-time, synchronized simulations for physical robots and cyber-physical systems. The Robot Operating System (ROS) is an open-source framework that simplifies the development of robot software. Gazebo, which is commonly used for robotics, provides physics simulations and sensor data emulation for ideal engineering applications. For instance, one study developed a five-degree-of-freedom industrial robotic manipulator using Gazebo and ROS, which enabled accurate forecasting of performance through integrated sensors. [2]. Recent research explored the development of digital twins for robotics systems using ROS. The work in this study looked at sensor integration, real-time data exchange, and simulation feedback, not the same as this study, but similar in the approach of using cybersecurity monitoring for predictive analysis.

3.2 Machine Learning for Drone Threat Detection

Predictive analysis in drone security has become increasingly important as drone development continues across different industries. By utilizing machine learning models, such as random forests (RF), support vector machines (SVM), and artificial neural networks (ANN),

certain threats can be mitigated in effective systems based on specific needs. Unmanned aerial vehicles are increasingly susceptible to cyber threats, including:

- **GPS Spoofing**: Tricking a drone's GPS receiver into thinking it's in a different location by sending fake GPS signals, leading it off-course.
- **GPS Jamming**: Disrupting or overwhelming the GPS signal to prevent the drone from determining its location.
- **Sensor Manipulation**: Interfering with or altering sensor data, either by tampering with the sensors or modifying the environment they monitor.

In one study, a team from Xidian University looked at how machine learning can be used to detect spoofing using perception data. In short, a team ensured that the UAV's position is not affected by GPS spoofing signals. Through their performance and evaluation, they used effective methods to conclude a detection rate of 99.69%. In doing so, the authors employed models like decision trees and support vector machines (SVMs) to differentiate the legitimate and spoofed signals.

The paper's machine learning-based approach aligns with the digital twin framework performed in this project by offering techniques for both physical and virtual drones [3]. Through the use of the machine learning model, the digital twin framework, and spoofing enabled scenarios which can be effectively simulated to asses the drone's response to external threats. Thus, the real-time integration of the machine learning model and predictive analysis enhances the overall security of he drone's system.

3.3 Real-Time Implementation of Predictive Models

Deploying machine learning models in real-time systems on drones has been a flourishing area of research. Though with the rise of cyber attacks, they have become a critical threat to drones. There are different kinds of machine learning algorithms, such as:

- **Supervised Learning**: Learns from labeled data, trained on input-output pairs.
- **Unsupervised Learning**: Works with unlabeled data to find patterns or groupings.
- **Semi-Supervised Learning**: Uses small amounts of labeled data with large amounts of unlabeled data.
- **Reinforcement Learning**: Learns through trial-and-error by receiving rewards or penalties.
- **Deep Learning**: Used for handling large-scale, complex data like logs or payloads.

With numerous machine learning algorithms, there have been many studies, though the one focused on this project is supervised learning. That is, being able to learn spoofed GPS attacks from labeled data and applying them in real-time. One recent study looked at how phishing threats use machine learning techniques to analyze web content [4]. The main idea here is that the authors employed algorithms like Random Forest to detect such malicious content to achieve a high F1-score, proving effective detection capabilities.

3.4 Relation to this Project

Emphasizing a little more on how this relates to the work, the system they used also looked at a cyber attack (phishing) on URLs and HTML content. With that being said, machine

learning applications were used for classification tasks. Then, there's a strong emphasis on how their team implemented the system in real-time for immediate threat identification.

With previous work on cybersecurity threats and real-time machine learning implementation, the digital twin framework can be further developed to integrate advanced threat detection and mitigation strategies, allowing real-time monitoring and adaptation to a cyber-physical system. By embedding such machine learning mechanisms, digital twins can simulate and respond to various cyber threats effectively in different applications.

In the context of digital twins, this study aligns with being able to create accurate real-time responses to cyberattacks on drones. Furthermore, by implementing real-time data and machine learning into the digital twin framework, one can enhance the twin's capability to respond to cyber threats. The integration of machine learning frameworks in real-time into digital twins is a huge step in enhancing security in not only drones but broader cyber-physical systems.

4 Problem Definition

Since drones have become such an important asset in various applications, the need for secure and protected drones grows in parallel. Cyber-physical systems rely heavily on real-time data and processing. Yet, with drones growing in complexity and a wide variety of interconnectivity, they are exposed to a wide range of cyber threats. Thus, with traditional cybersecurity approaches for real-time and machine learning applications, it is often overlooked to account for the consequences of cyber attacks on these systems.

4.1 Objective

The main objective of the project is to create a digital twin of the PX4 drone, which can be controlled to reflect the physical drone's behavior and observe how GPS spoofing attacks can jeopardize the drone's operation. This project addresses the need for a simulation-based environment using a virtual representation, enabling a predictive analysis of its vulnerabilities. Using the digital twin in a safe environment, not testing on the physical drone, allows for safe experimentation without risking real-world hardware.

4.2 Central Problem

The central problem is understanding how these spoofing attacks occur, when introduced in real-time, and how they can propagate to compromise the drone's operation. For example, what happens when the GPS receives data that is spoofed? Can the drone's corrupted real-time data mislead the drone to different locations? By looking at the effects of spoofed data in real-time, the twin receives this data and prevents such movement. This project looks at these vulnerabilities that might remain undetected until exploitation, ultimately leading to the development of a robust defense mechanism for cyber-physical systems.

5 Project Goals

5.1 Primary Goal

The primary goal of the project is to develop a real-time digital twin of the PX4 drone that can be controlled and analyze how GPS spoofing attacks can jeopardize the drone's operation. This digital twin will accurately mirror the drone's behavior and environment, enabling safe testing of attack scenarios without risking the physical system. By injecting spoofed GPS data into the digital twin, the project will examine how the manipulated location information can affect the drone's path, navigation, and stability.

The twin serves as a virtual representation to identify the vulnerabilities in the drone's reliance on positional data to evaluate the corrupted data in real-time. Ultimately, the project aims to provide insights that can guide the development of effective countermeasures against GPS spoofing threats. This is accomplished entirely within a digital twin simulation environment, which replicates the drone's physical dynamics and sensor feedback using software, allowing controlled injection of spoofed GPS signals and real-time observation of their impact—without exposing the actual hardware to damage or risk.

6 Contributions

6.1 Deliverable of Project

This project delivers a functional digital twin framework for a PX4-based drone, focused on real-time synchronization, a simulation environment, and cybersecurity aspects. The system was built using ROS2 and Gazebo, enabling a virtual twin to represent the "physical" PX4 drone in simulation. This foundation allowed for communication and tracking between the real and simulated environments, which is imperative for monitoring the drone, testing the cyber attacks, and securing the cyber-physical systems. Overall, the project demonstrates how digital twins can not only be used for visualization but also as a defense layer against cyber threats.

6.2 Key Contributions

Key contributions for the project include the architecture and deployment of the simulation, integration with the physical PX4 drone, the ground control system, and development of the predictive framework to assess GPS spoofing and risks. The project also explores the use of the machine learning algorithm, the Random Forest model, to detect anomalies related to spoofing. While the primary focus was on creating the digital twin, the work sets the stage for future efforts like defense responses. Ultimately, the project contributed to the research on cyber threats in a digital twinning framework.

7 Why This Project?

7.1 Evolving Role of Drones in Military

While drones continue to play crucial roles in civilian and military applications, reliability and security have become pivotal. These systems rely on Global Positioning System (GPS) data for navigation. Though GPS signals are vulnerable to spoofing attacks, attacks where false signals are sent to the drone, mislead the drone into thinking it's in a different location. Therefore, these attacks can result in mission failure, loss of control, and even crashes. With such attacks, it can make it harder to test GPS spoofing on physical drones, which can lead to unsafe, expensive tests even in real-world conditions with external factors like wind.

7.2 Reason for the Digital Twin of the PX4

This project aims to fill that gap by creating a digital twin of the PX4 drone. In short, it uses an open-source autopilot platform widely used in research and academic purposes. The digital twin acts as a safe testing environment for the PX4 drone to test the GPS spoofing attacks without risking any actual hardware. It uses a node to take in the data in real-time to simulate how the GPS spoof signals are received and observe how the drone reacts in various flight paths, like straight paths, far spoofed signals, various paths, and different locations, providing valuable insight into the system's vulnerabilities.

7.3 Groundwork

By developing this digital twin, we can explore how the spoofing affects the planned and manual flights to develop robust anomaly detection. Moreover, the project lays the groundwork for cyber-physical system security, which can be used for broader research into digital

threats.

7.4 Importance of the Digital Twin Framework

Ultimately, we want to do this project because it provides a low-risk way to understand and improve drone security. As the world progresses with more autonomous vehicles and systems, being able to anticipate cyber threats before they happen and anticipate them can protect against potential failures, breaches, and mission-critical disruptions, ensuring safe and reliable operations in real-world environments. The insights gained from this project can help future designs, cybersecurity frameworks, and contribute to safer, resilient operations, not only for drones but for other possible cyber-physical systems like cars and robots.

8 Project Details

8.1 Project Setup

The development environment for the digital twin framework was configured with the following softwares and platforms to ensure compatibility and performance for simulation and testing. Here is what was chosen and why:

- **Operating System**: Ubuntu 24.04 LTS was selected for its stability and compatibility with robotics software, ideal for cyber-physical system projects, offering smoothness and security
- **ROS 2 Distribution**: ROS 2 Jazzy Jalisco was installed as it is built for Ubuntu 24.04. ROS 2 Jazzy Jalisco provides tools and libraries for robot applications that support real-time systems and robot communication, which can also be integrated with ROS 2 middleware. It is an open-source robot software framework enabling real-time communication between the physical drone and its simulated twin.
- **Simulation Environment**: Gazebo Harmonic was chosen as it has been tested and runs well with Ubuntu 24.04. It supports ROS 2 integration, like Jazzy, for testing robots in virtual worlds. Gazebo Harmonic is a simulation software that offers advanced simulations like physics and sensors, supports Python, and is also a long-term support (LTS) release.
- **Visualization Tool**: RViz is used to visualize sensor data, robot states, and the environment. It was used to visualize the real-time state of the drone to monitor movement and system behavior.

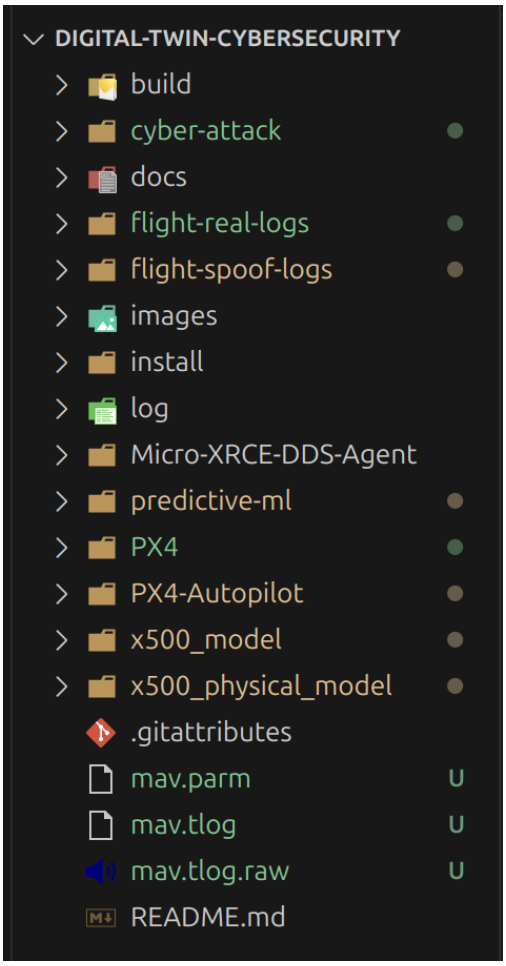

Figure 3: Directory Setup

8.2 System Architecture and Design

This section reviews the digital twin framework's high-level and low-level architecture. Both are essential designs that show the thought and progress put into the framework's underlying back structure. The high-level architecture looks at the major system components and their interactions from a broader view. In comparison, the low-level architectures dive into the specific implementation details such as UDP ports, communication protocols, tools used, and subsystem interactions. Overall, these systems highlight the planning, decisions, and complexity involved in developing the digital twin framework.

8.3 High-Level Architecture Design

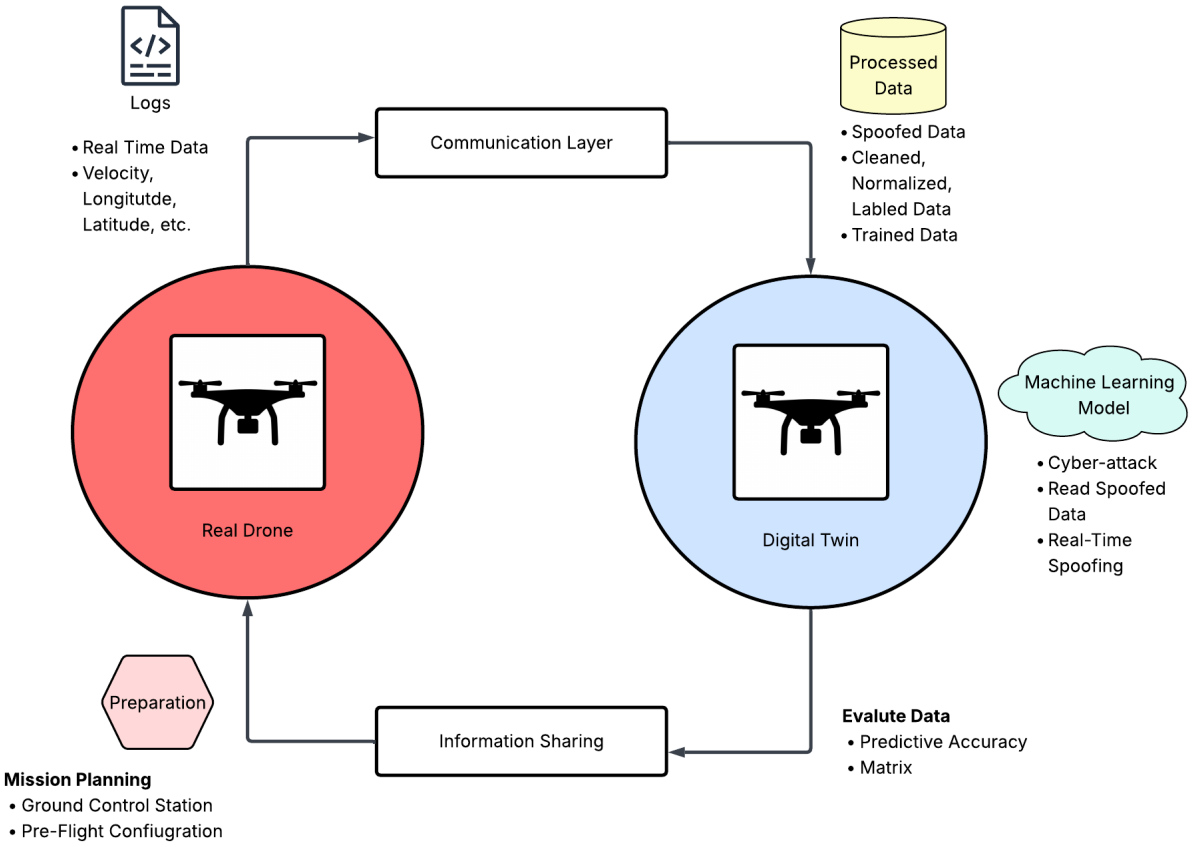

Figure 4: High-level architecture of digital twin framework

The high-level diagram provides a detailed breakdown including:

- **Real World Drone**: Holybro X500 V2 PX4 used for data collection and flight operations.
- **Digital Representation**: Digital twin model of the PX4 drone ROS 2, Rviz, Gazebo
- **Communication Layer**: Data transferred and analyzed using Python libraries; Real-time data such as the velocity, longitude, latitude, and other parameters.

- **Information Sharing**: Evaluation of collected data for accuracy and predictive performance, with emphasis on predictive analysis matrix and logs.

8.4 Low-Level Architecture Design

The low-level diagram provides a detailed breakdown including:

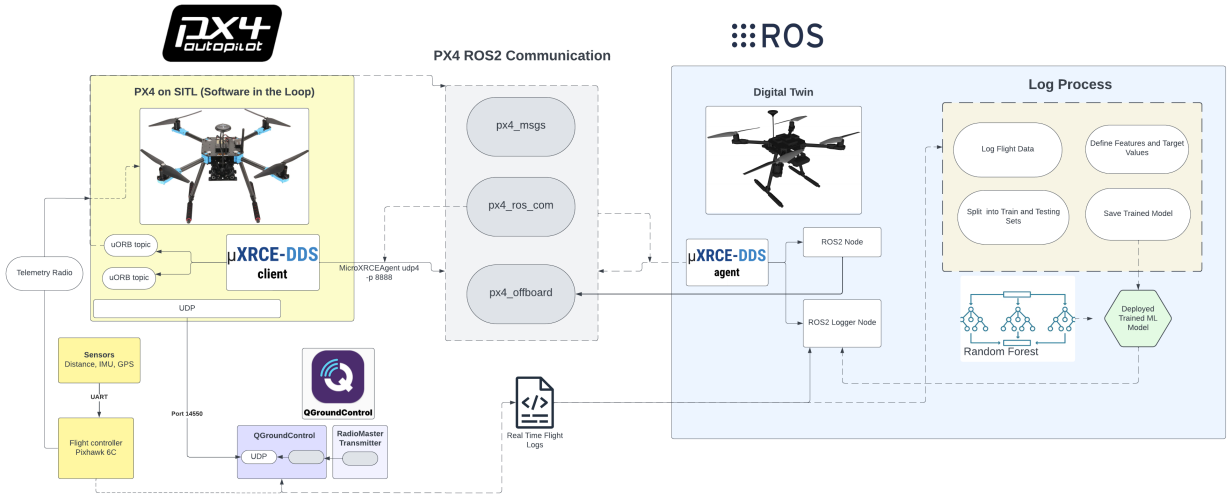

Figure 5: Low-level architecture of digital twin framework

- **PX4 on SITL (Software-in-the-Loop) Simulation**: Acts as the virtual drone model and serves as the physical counterpart in the digital twin framework. Here is the GitHub reference for the Software-in-the-loop used in this project [5].

- **uORB Topics & uXRCE-DDS Client**: PX4 communicates internally through uORB topics and externally over UDP using the uXRCE-DDS client.

- **Telemetry and Flight Controller**: The telemetry radio handles wireless communication, including sensor data (e.g., IMU, GPS, distance). The **Pixhawk 6C** is the physical flight controller connected via UART.

- **QGroundControl (QGC)**: The ground control station (GCS) used to operate both the simulated and physical PX4 drones. Connects via **UDP port 14550** and supports manual control through the **RadioMaster Transmitter**.
- **PX4 ROS 2 Communication Bridge**: Built on the following three packages:
  - `px4_msgs` – defines PX4 message formats.
  - `px4_ros_com` – establishes the communication bridge.
  - `px4_offboard` – handles offboard control commands.
- **Digital Twin Subsystem**: Represents the virtual counterpart of the drone, processing data from ROS 2 nodes. Includes a logger node that stores flight data for further analysis.
- **Log Processing and Machine Learning**:
  - Logs are recorded in real time from QGroundControl.
  - Features and target variables are defined for the ML model.
  - The dataset is split into training and test sets.
  - A **Random Forest** algorithm is used to train the model.
  - The trained model is deployed in real-time within the digital twin via a ROS 2 node.

8.5 Digital Twin Setup

This section provides an overview of the digital twin setup, beginning with the initial build process, including the integration of the CAD file and the simulation environment. It includes the initial setup for both the simulated physical drone and the physical Holybro PX4 drone,

which I created/modified. It allows the drone to be virtually represented for real-time data processing and control.

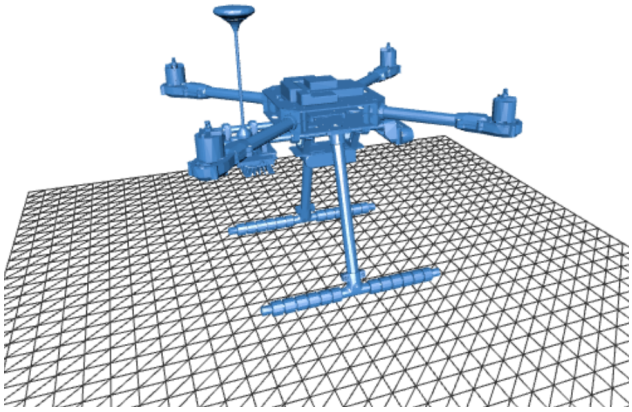

Figure 6: STL (CAD) Model of PX4 Drone

In Figure 6 an STL model of the PX4 drone was modified from [6]. An STL model is a 3D model of the PX4 drone that describes its surface geometry using a mesh of connected triangles. While the STL file only contains the drone's geometry, it does not include additional information such as colors, joints, or sensors; it solely represents the physical shape.

Next, the STL model was integrated into a URDF (Unified Robot Description Format) file to describe the robot's structure. This URDF model references the STL file for the drone's visual and collision elements. RViz then reads the URDF file and visualizes the drone using the STL model to simulate the drone's behavior.

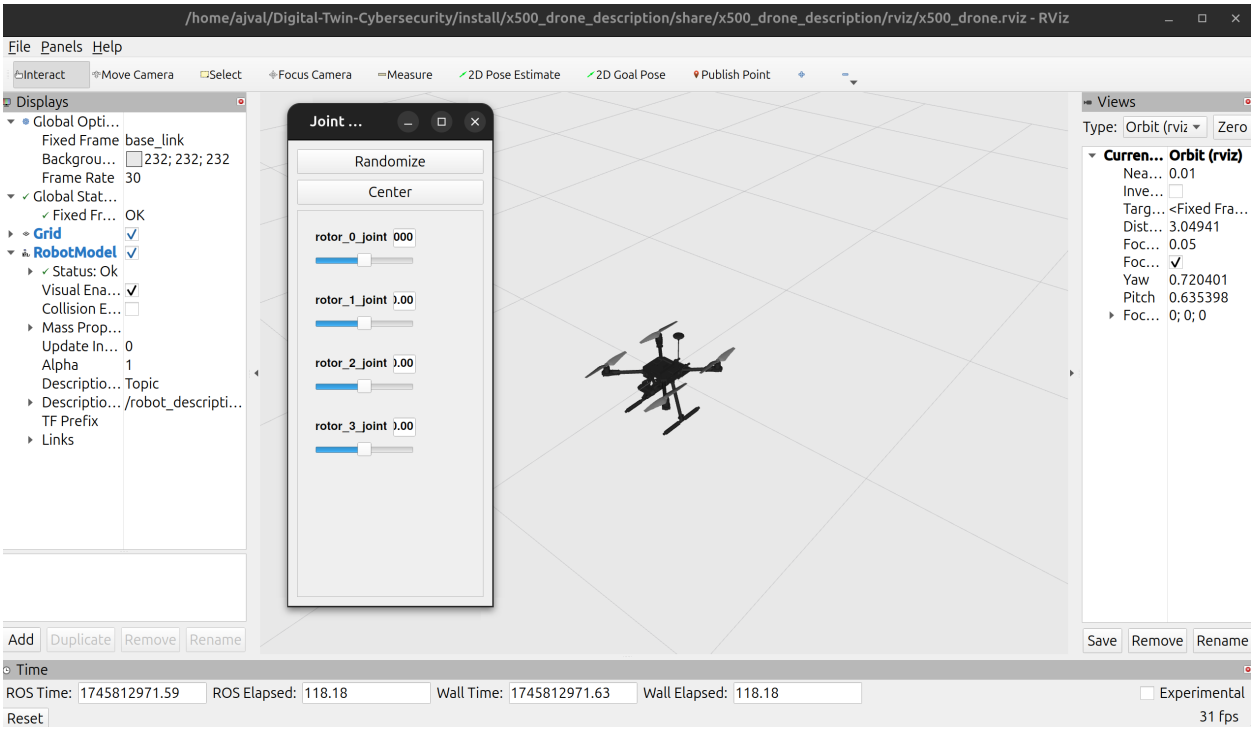

Figure 7: PX4 Drone with Propellers

For the four rotors, each rotor was added as its link and attached to the body (base_link) using a joint modeled to replicate a realistic propeller. It has an origin and XYZ positions and an rpy (roll-pitch-yaw), so it can rotate the propeller if needed. The fixed joint is used because the rotors are visibly attached. After loading the URDF into ROS 2, RViz now reads the robot's description parameters. Now that the four rotors are attached and linked, it's loaded in ROS 2 so it can be visualized assembled with the body and rotors.

8.6 Integration and Communication

In this section, the translation of ROS 2 messages is imperative to understanding how different systems communicate with one another. The translation of nodes interworks with versions of PX4 to ensure data exchange between the flight controller and the ROS 2 envi-

ronment.

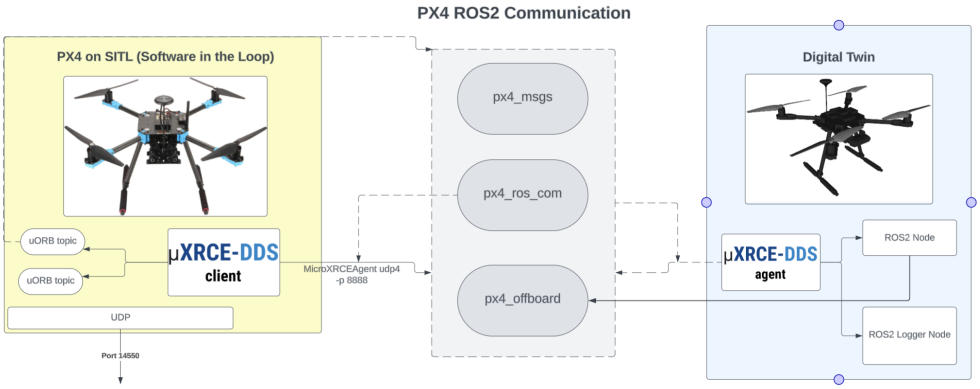

Figure 8: Digital Twin and Physical Drone Communication

In the above Figure 8, the communication between the digital twin and the physical drone is described from a higher level. The ROS 2 packages msgs, ros_com, and offboard integrate the digital twin and the physical drone. The next figure will describe how the packages are used in more detail.

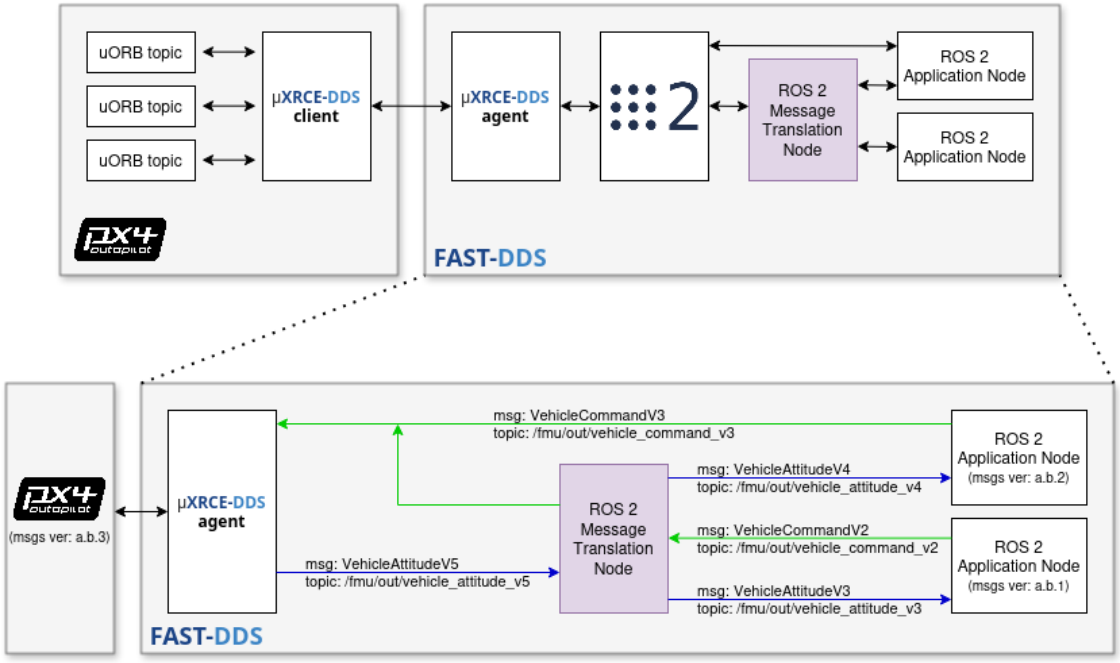

Figure 9: PX4 Message Translation

From the PX4 documentation, Figure 9 describes how both message translation operates in the digital twin framework. This figure is taken from PX4's ROS 2 Message Translation Node documentation [7]

Message translation nodes allow ROS 2 applications to work with PX4 messages and different versions of PX4. Communication for the digital twin framework in this project is built on this structure. Essentially, the translation node has access to the clients, agents, publications, and subscriptions. It converts the messages to versions readable by PX4, ensuring communication and compatibility. Thus, in this project, the translation node manages the message translation for the drone's topics, such as the altitude, velocity, and location.

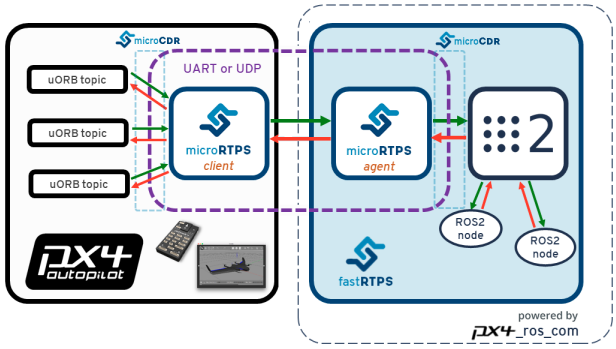

Figure 10: Integration of Packages

In the framework, ROS 2 uses both px4_msgs and px4_ros_com to ensure that message definitions are created for both the client (the physical drone) and the agent (the digital twin). The application pipeline for ROS 2 uses a middleware (DDS/RTPS). Essentially, this is a data distribution service (DDS) and real-time publish-subscribe (RTPS), which are both used for real-time, reliable communication between systems widely used in robotics, drones, and other IoT devices. Very shortly, it uses a publish-subscribe model in which a publisher sends data, and the subscriber receives that data, while supporting QoS (Quality of Service) for durability and reliability.

At a higher level the px4_ros_com package translates the PX4 uORB messages, which are

just internal PX4 messages, into ROS 2 topics. Then, the package is used to communicate with the digital twin connected on UDP port 8888. One important thing worth noting, the real-life Holybro PX4 drone connects through UART: " /dev/serial/by-id/usb-FTDI_FT231X_USB_UART_D30JCTDI-if00-port0 –baudrate 57600" through two radio transmitters. For the majority of this project, on the PX4 Autopilot simulation, the UDP port was used. In Figure 10, we see that ROS2 uses both packages: px4_msgs, which contains the client message definitions, while px4_ros_com builds the px4_msgs and uses the files to create and build the agent.

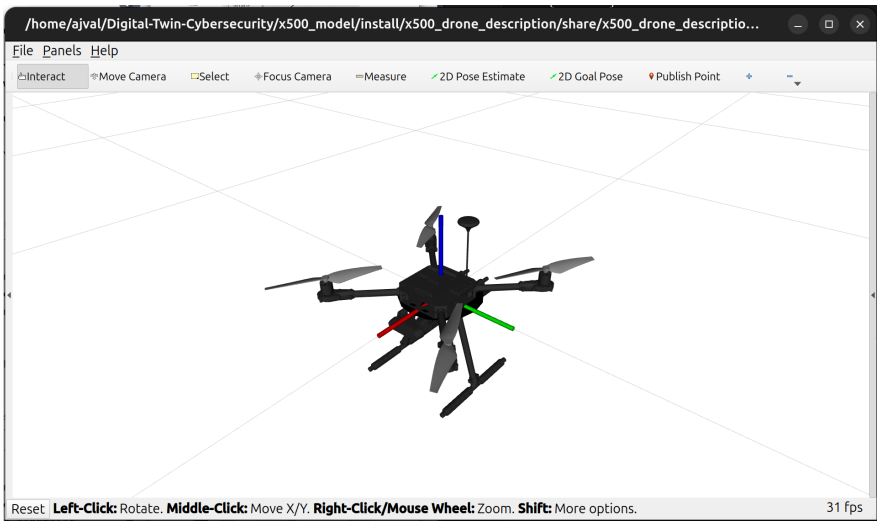

Figure 11: Offboard Visualizer

As for the last ROS2 package used, px4_offboard is referenced from Jaeyoung-Lim's px4-offboard GitHub with a slight modification [9]. PX4 Offboard allows for a companion PC to take direct control of the PX4 drone by sending it position, velocity, or altitude commands in real time, instead of relying on onboard autonomous modes. In this project, PX4 Offboard was used to simulate the drone by sending ROS2 messages simulated in the PX4 Autopilot, allowing the digital twin to dynamically update the drone's behavior.

8.7 Attack Simulation (GPS Spoofing)

In the context of the project's digital twin, GPS spoofing was emulated by injecting fake positions (longitude, latitude, and altitude) via offboard control using MAVSDK over UDP. Due to the nature of the PX4 Autopilot system, being open-source and modular, it's vulnerable to spoofing attacks because offboard APIs are enabled without proper authentication or encryption, and MAVLink communication is unsecured. This illustrates how a malicious actor could override a real GNSS (Global Navigation Satellite System) signal and take control. Such vulnerabilities are critical in cyber-physical systems, emphasizing the need for encrypted MAVLink channels.

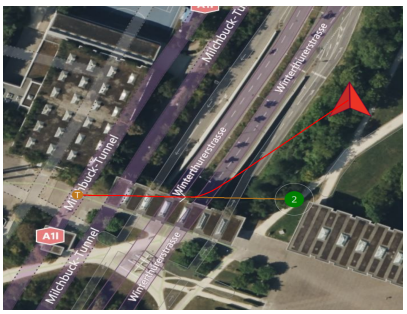

Figure 12: Straight Spoof

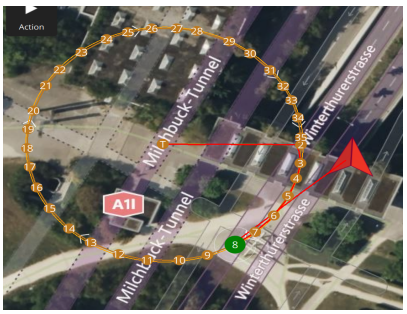

Figure 13: Circular Spoof

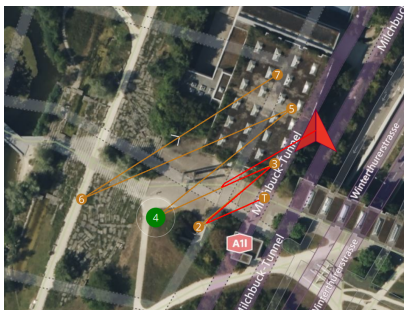

Figure 14: Arbitrary Spoof

In the above three figures, the controlled GPS spoofing experiments were tested on the PX4 Autopilot system using the digital twin setup. To simulate cyberattacks with fake coordinates to observe the drone's behavior, three path profiles were tested: straight-line trajectory, circular path, and arbitrary maneuvering. Because machine learning can be trained on diverse data, such as different flight profiles, it was decided to perform various flight paths to improve model generalization. Testing across various profiles ensures the model recognizes spoofing in a wide range of flight behaviors, such as different longitudes, latitudes, and altitudes, ultimately making it more effective in real-world situations.

Flight Number	Spoofed Latitude	Spoofed Longitude	Spoofed Altitude
1	37.7749	-122.4194	50
5	50	19	20
6	100	100	100
24	49.2201	12.4573	5.345
33	24.797	70.593	3.017
37	126.789	119.098	66.891
71	346.864	268.537	10.947
92	63.737	72.567	15.236
130	68.641	72.692	49.952
136	-225.697	-156.359	480.648
150	126.487	197.853	170.452

Figure 15: Log of Various Flight Tests

Upon collecting both real flight logs and spoofed logs from the digital twin and the physical drone, 150 flights were conducted, with approximately less than 100,000 instances collected for model training. The digital twin recorded separate logs with spoofed data, while the physical drone with the same flight was recorded and obtained from the QGroundControl flight logs. Both were recorded into the two different directories, one for spoofed data and the other for real flight logs from the same flight.

In Figure 15, the table shows a simplified version of the recorded data tracked during the spoof attacks. Each spoofed flight included altered GPS coordinates—specifically, changes in latitude, longitude, and altitude. Some contain extremely large or even negative coordinate values that wouldn't occur during normal flight. By training the model with this range of spoofed data, the model was able to learn how to distinguish between genuine and manipulated GPS signals.

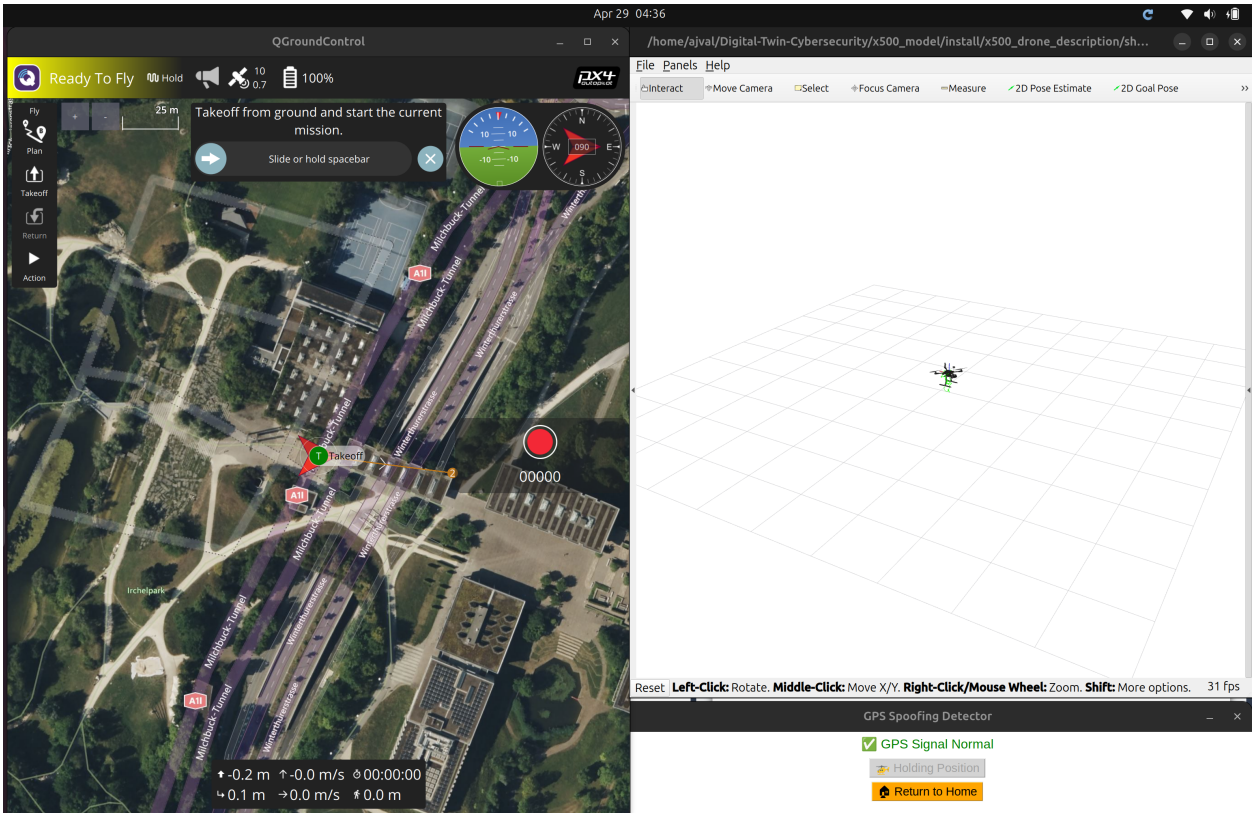

Figure 16: Digital Twin Environment

In the above Figure 16, the image shows the digital twin following a straight path during a normal flight. This illustrates that the GPS signal remains consistent and accurate, with no anomalies present, indicating a legitimate and stable flight pattern.

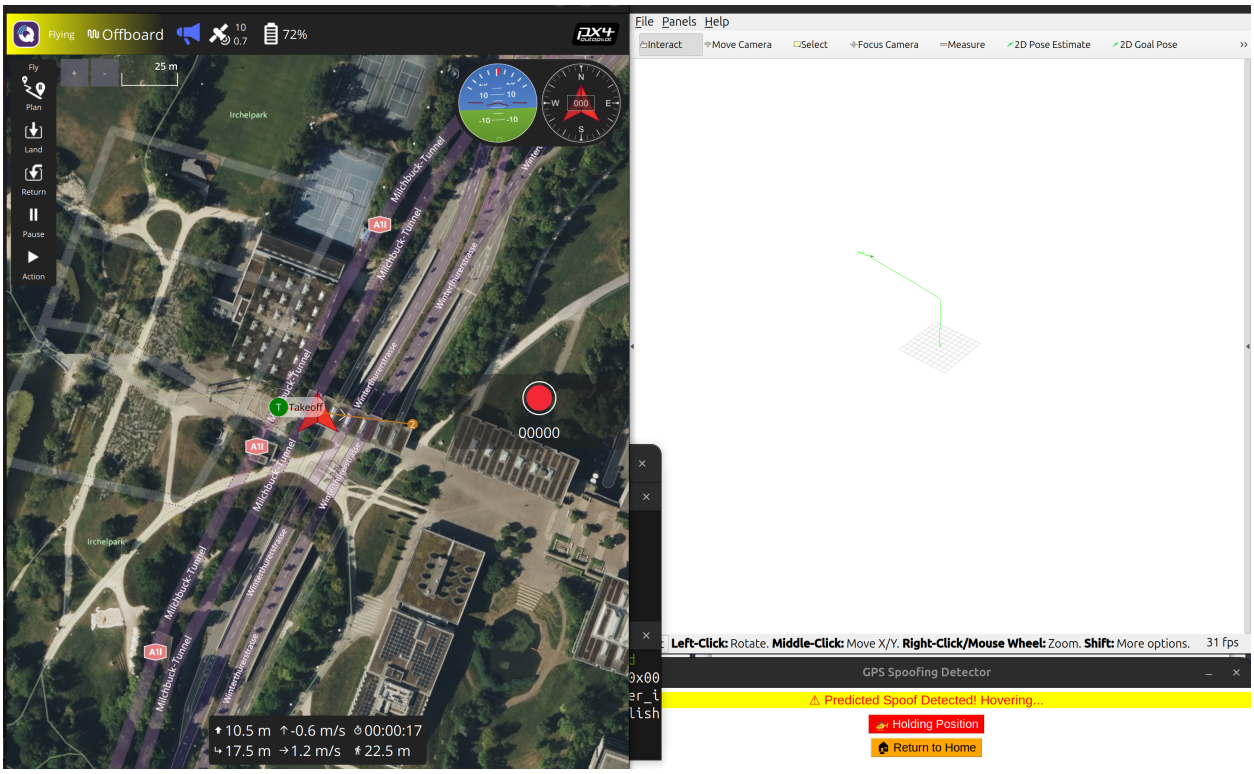

Figure 17: Predicted Spoof

In Figure 17, the digital twin is shown following a straight flight path; however, this path is the result of a GPS spoofing attack. In this case, specifically a spoof attack on `SPOOFED_LAT = 126.487`, `SPOOFED_LON = 197.853`, `SPOOFED_ALT = 170.452`. Unlike the normal behavior in Figure 16, the system detects inconsistencies in the GPS data. As a result, the digital twin displays a warning: "Predicted spoof detected! Hovering...". This response indicates that the digital twin successfully identified the spoofed signal and triggered a safety protocol, causing the drone to stop and hover in place to prevent further navigation under false data. Moreover, a Return to Home button is optional if the user wants to redirect the drone back to its launch position safely.

8.8 Challenges and Debugging

Throughout this project, several challenges and roadblocks were encountered with software tools, debugging, and up to learning curves. While QGroundControl (QGC) is a basic flight control and telemetry tool for drones, integrating it with both the physical drone model and the simulated physical drone required significant trial and error. For one, the physical drone communicated over serial UART through radio transmitters. Meanwhile, the simulated physical drone in Gazebo is connected to QGC over UDP. Moreover, learning to configure the sensor data in RViz and render the URDF model demanded proper synchronization for the ROS 2 topics. As seen in Figure 18, the ROS2 topics are core communication for publishing and subscribing to messages between nodes. These were used to update the physical drone's location, for example, to the subscriber to the companion computer understands its exact coordinates. Other topics can be measured like motors, attitude, flags, status, etc.

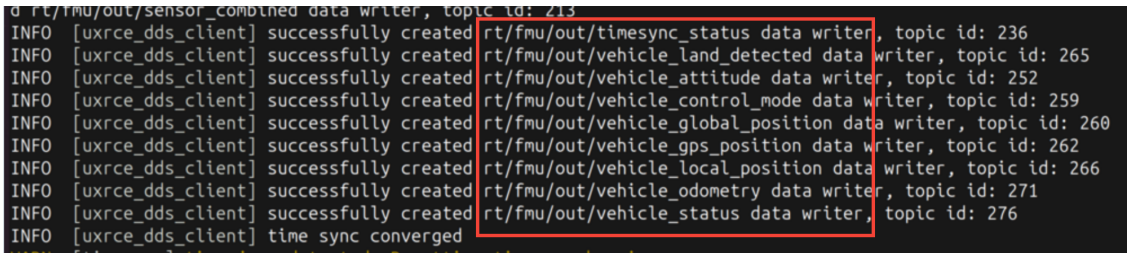

Figure 18: ROS2 Topics List

Additionally, communication was probably the biggest challenge aside from the learning curve of the new software. Understanding the seamless communication between the physical drone, ROS 2 nodes, and Gazebo required an in-depth understanding of DDS/RTPS, serial telemetry, and launch configurations. Debugging the issues proved very time-consuming. Lastly, machine learning integration within the digital twin requires careful features, cleaning, and labeling. Ensuring the balanced datasets fit involved repeated testing and tuning of the random forest model. Despite these challenges, overcoming them contributed to the depth

and realism of the digital twin, improving the system's robustness.

8.9 Results and Observations

The implementation of the digital twin for the PX4 Autopilot system successfully simulated the behavior of the physical drone in a variety of environments under different attack scenarios. The initial creation of the digital twin—focusing on GPS spoofing, detection mechanisms, and machine learning-based tests—demonstrated its ability to replicate real-world behavior and provide a solid foundation for predictive cybersecurity analysis.

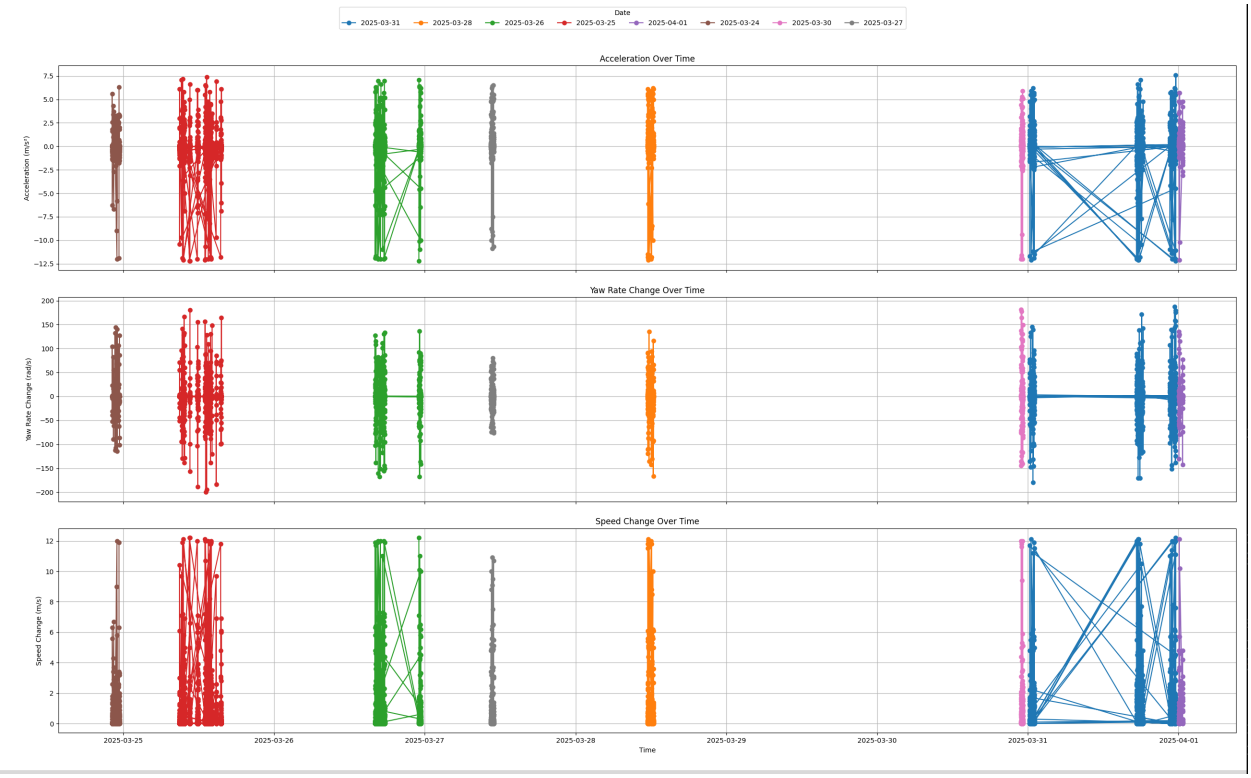

Figure 19: Changes in Acceleration, Yaw, Speed over 150 Tests

Figure 19 above represents the results of the 150 drone flights throughout testing are represented. The flight tests are designed to capture telemetry under normal and spoofed

conditions. The top plot visualizes the acceleration over time, where noticeable spikes indicate a rapid thrust observed during GPS spoofing injections. As for the middle plot, yaw rate changes reveal the angular instability and student rotational shifts, indicating the spoofed navigation path. Lastly, the last plot reveals speed over time, in which abrupt transitions and irregular intervals demonstrate variability during spoofing events. Overall, over the 150 tests conducted, with less than 100,000 instances, this graph demonstrates the variability introduced during the cyber attacks, highlighting how such features were extracted and analyzed for machine learning.

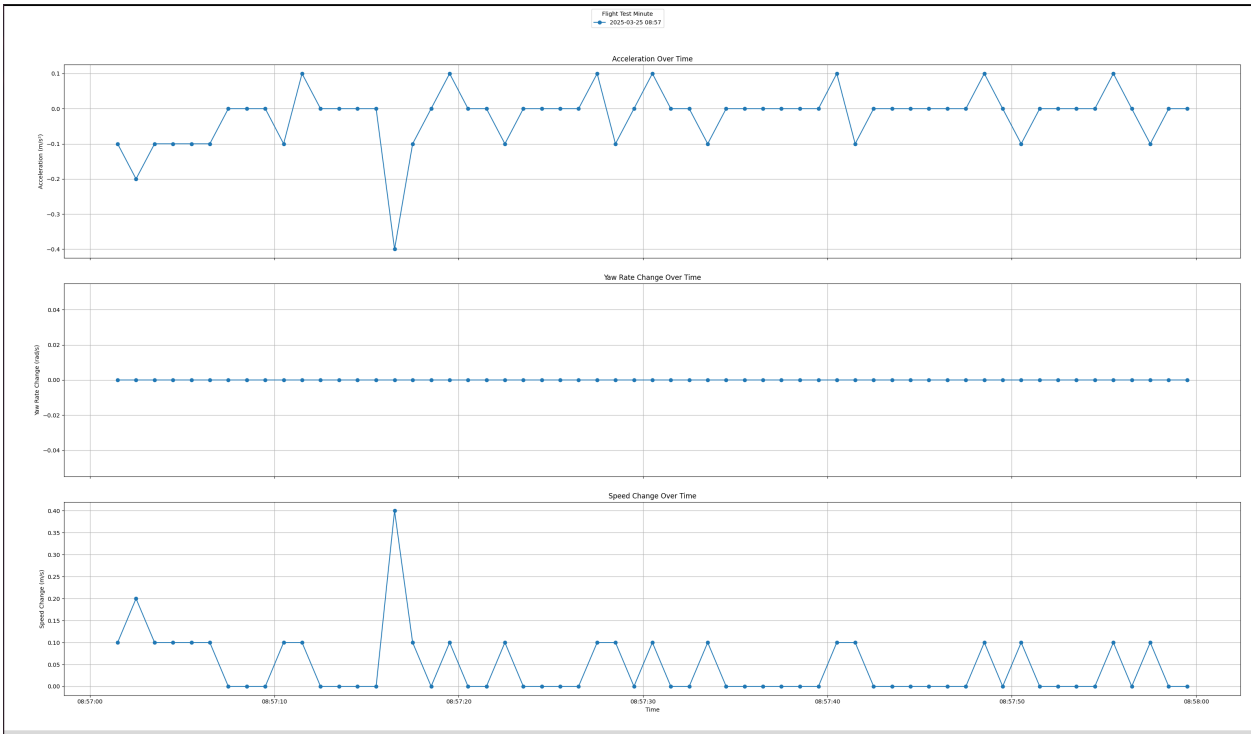

Figure 20: Single Test of Flight Log, Changes in Speed

Figure 20 shows a detailed telemetry record of a flight to visualize, simpler focusing on acceleration, yaw rate change, and speed. Throughout the flight, the yaw rate remained constant due to the control from QGroundControl through the ROS2 node. However, acceleration fluctuated between -0.38 m/s^2 and 0.11 m/s^2. Meanwhile, the speed peaked at roughly 0.39

m/s, with multiple points showing zero change. These abnormal movements were flagged by the machine learning model as spoofed behavior. Similarly, as this is a single flight, the same testing was recorded across all flights to ensure consistent spoof detection and model training.

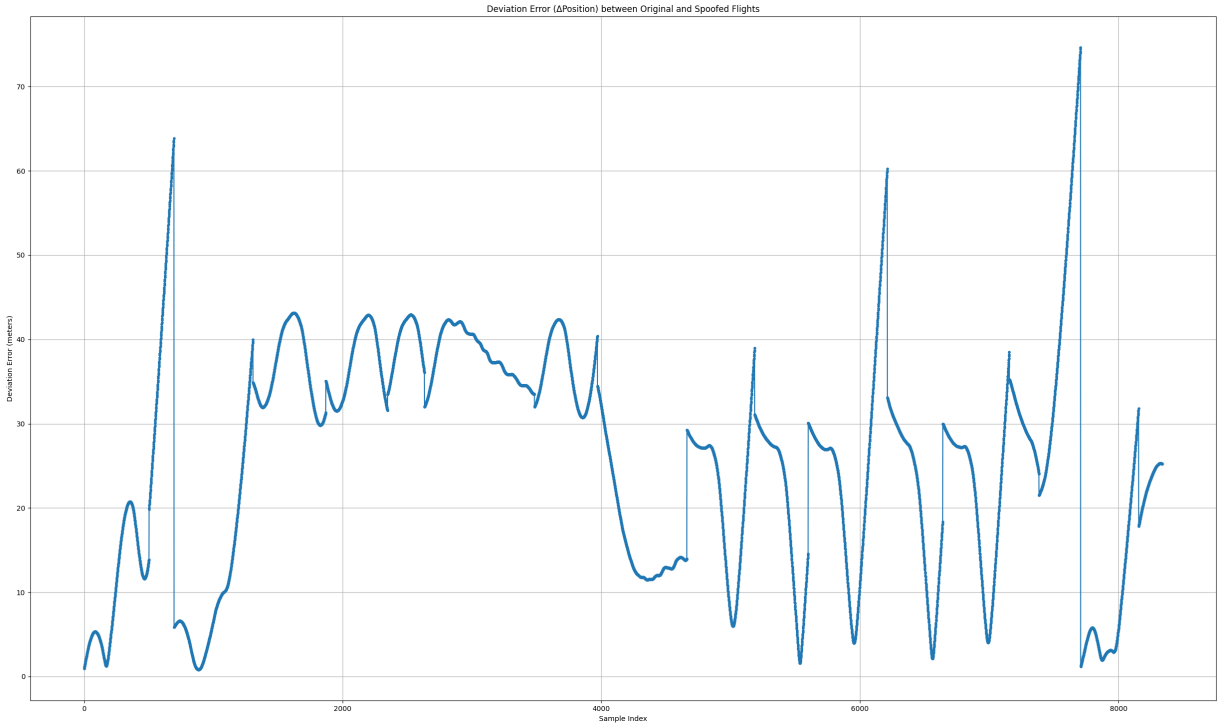

Figure 21: Deviation Error Between Spoofed and Original Flight

To quantify the effects of the GPS spoofing on the original paths of the flights, a deviation error analysis was conducted on non-spoofed and spoofed flight logs using the Haversine formula. The formula calculates the great-circle distance between latitude and longitude points.

The deviation error ΔPosition was computed using the Haversine distance:

$$\Delta d = 2R \cdot \arctan\left(\sqrt{a}, \sqrt{1-a}\right)$$

where:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

Here, ϕ and λ represent latitude and longitude in radians, respectively, and $R = 6{,}371{,}000$ m is the Earth's radius. This formula was applied point-by-point between corresponding entries in the real and spoofed flight datasets. Since Gazebo uses real-world latitude and longitude coordinates in the simulations, the magnitude used in meters is appropriate.

Figure 21 illustrates the deviation error for the change in position across all 150 test flights. The graph compares the original and spoofed positions over time. Each point represents the position error between the datasets using the Haversine formula. The graph shows different things. First, an initial deviation around 2 meters but rises, averaging roughly 30-45 meters, with frequent spikes exceeding 60 meters. These sharp increases in positional deviation correspond to moments when the spoofed GPS signals misled the drone's perceived location. The nature of these wave-like patterns fluctuates between 2000 and 8500, highlighting the consistent spoofing over the various flights, like straight, circular, and arbitrary path profiles. The deviation map across the diverse scenarios reinforces the deviation error as a critical feature in spoof detection, helping validate the digital twin's ability to track and quantify behavior in real-time.

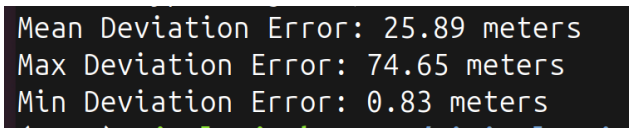

Figure 22: Deviation Statistics

The results in Figure 22 summarize the deviation across the 150 flights. On average, the spoofed positions deviated 25.89 meters, showing consistent spoofing interference. The 74.65

max deviation shows that the drone was severely misled in the worst case. This is possibly due to the sudden jump or spoof injection to the coordinates, or even a GPS correction delay. And the 0.83 meters min deviation indicates that there were moments with little or no spoofing impact. This value could be due to the spoofing not yet being activated, or even the GPS filtering in QGC aligning with the spoofed data to the real path.

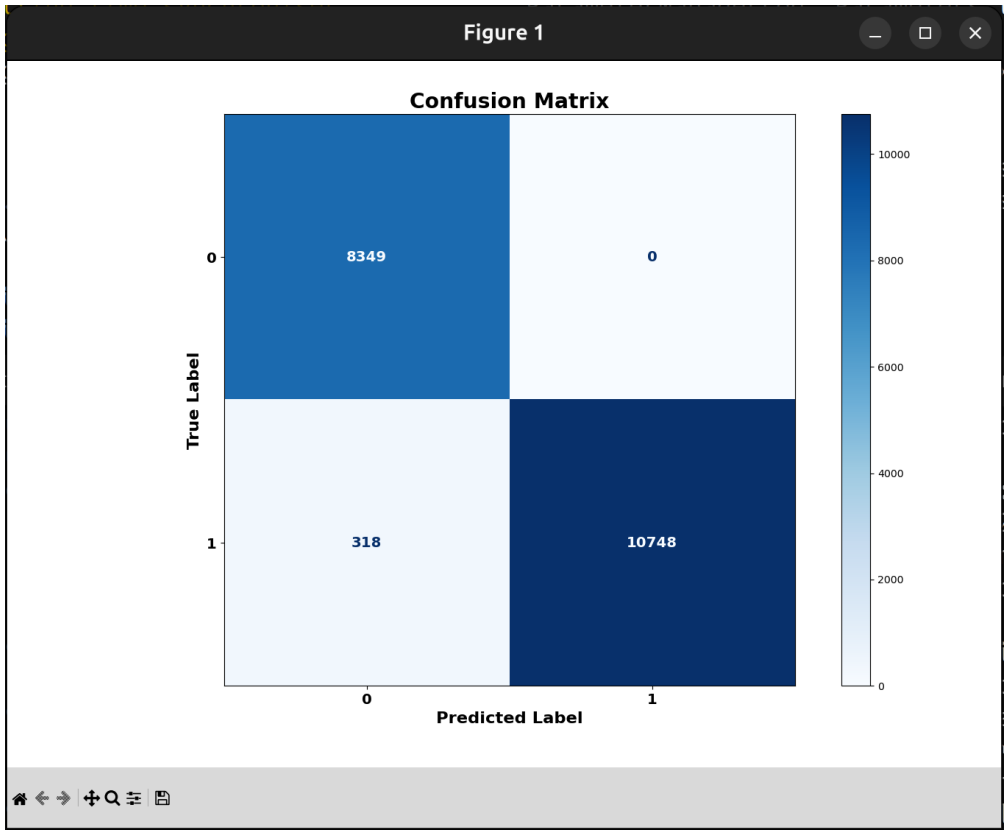

Figure 23: Confusion Matrix from Machine Learning Model

The confusion matrix in Figure 23 demonstrates the effectiveness of the Random Forest model in detecting GPS spoofing for the real and spoofed data sets. Out of the data sets, 10,748 spoofed instances were correctly identified as spoofed data. There were 8,349 non-spoofed instances, which meant they were correctly identified as normal (non-spoofed data). Moreover, for the false negatives, there were 318 instances in which spoofed data were incorrectly predicted as normal. Lastly, 0 false positives means the model never falsely labeled a

normal flight as spoofed, which is excellent for safety-critical applications.

```
(venv) ajval@aj-ubuntu:~/Digital-Twin-Cybersecurity/predictive-ml$ ./trainedResults.py
Accuracy: 0.9836209116662374
Precision: 1.0
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      8349
           1       1.00      0.97      0.99     11066

    accuracy                           0.98     19415
   macro avg       0.98      0.99      0.98     19415
weighted avg       0.98      0.98      0.98     19415
```

Figure 24: Accuracy and Precision

This report shows the Random Forest ML model achieved an overall accuracy of 98.36% with a precision score of 1.0 for spoofed data. This high performance is due to a possible clear separation in data patterns. There was a distinct exhibition of spatial and velocity-based patterns. Moreover, the six input features: latitude, longitude, altitude, velocity north/east/down, can directly relate to how the strong indication helps the model differentiate between spoofed and non-spoofed behavior. With nearly 20,000 labeled instances, it provided a balanced distribution of data, reducing the risk of overfitting. Possible reasons as to why there were some errors may be due to the spoofed behavior may closely mimic legitimate flight characteristics due to certain spoofed paths, making it slightly harder for the model to distinguish between spoofed and real data, leading to 318 false negatives.

9 Future Work

Future work for this digital twin project includes expanding capabilities beyond GPS spoofing, such as motor faults, abnormal power consumption, and even incorporating wind and external disturbances in the simulation, which would help improve the digital twin's robustness.

Here are some expanded potential future works for the digital twin project that seem interesting to possibly dive into:

- **Honeypots in the Digital Twin:** Extend the digital twin to simulate decoy UAVs or fake telemetry paths, acting as honeypots that lure attackers into revealing themselves while monitoring their behavior without endangering the actual drone.
- **Game-Theoretic Framework for Jamming/Deception:** Model a 2-player jamming game (e.g., attacker chooses channel A or B to jam; defender chooses avoidance or counteraction), and simulate it in real time using the digital twin. With telemetry input, it's possible to build on the communication channel and implement different strategies.
- **Hypergame Theory Applications:** Leverage the twin to simulate asymmetric knowledge, where the attacker and defender have different perceptions of the system. Possibly, use the GPS spoofing framework to show how attackers may misjudge drone responses, and vice versa.
- **Metaverse Security Prototyping:** Use the drone twin framework as a representative CPS node inside a simulated metaverse and explore how UAVs might interact with digital assets in a connected virtual world—then simulate attacks like digital forgery or identity spoofing.

10 Conclusion

To conclude, this digital twin project demonstrates a different approach to enhancing security and situational awareness in cyber-physical systems through real-time synchronization, predictive modeling, and anomaly detection. Through simulating GPS spoof attacks and training a supervised machine learning model in the digital twin framework collected through various flight patterns, the system successfully identified spoofed behavior with high accuracy. The integration of the physical drone with a virtual twin created a dynamic testbed for certain attack scenarios and allowed for potential countermeasures without risking real-world assets. Overall, the research highlights the potential of digital twins not only as a mirror but as a defense tool capable of learning, adapting, and responding to threats. This work lays the foundation for more intelligent, adaptive, and secure systems, highlighting the potential of digital twins in enhancing the resilience of autonomous platforms.

11 Code Repository

The complete source code, simulation environment, and ROS 2 packages developed for this project are publicly available at the following GitHub repository:

https://github.com/alexmvalencia/Digital-Twin-Cybersecurity

This repository includes all scripts, launch files, CAD models, and machine learning components used in the development of the PX4 digital twin framework for GPS spoofing detection.

12 References

[1] I. Degges, "DOD Eyeing Digital Twins to Strengthen Warfighting Capabilities," *GovCon Wire*, Sep. 13, 2023. https://www.govconwire.com/2023/09/dod-eyeing-digital-twins-to-strengthen-warfighting-capabilities/

[2] C. Hitchens, "US Army to Build Digital Twins of Key Weapons Systems," *Defense News*, Sep. 12, 2023. https://www.defensenews.com/pentagon/2023/09/12/us-army-to-build-digital-twins-of-key-weapons-systems/

[3] J. Ma, X. Zhang, W. Yu, X. Zhang, L. Ge, and Q. Liu, "A Remote Sensing Image Change Detection Method Based on an Improved U-Net Network," *Remote Sensing*, vol. 14, no. 19, p. 4925, 2022. https://www.mdpi.com/2072-4292/14/19/4925

[4] A. S. R. Srinivasan, M. A. V. S. Prasad, and B. Eswara Reddy, "Real-Time Content-Based Cyber Threat Detection with Machine Learning," *ResearchGate*, 2021. https://www.researchga
Time_Content-Based_Cyber_Threat_Detection_with_Machine_Learning

[5] PX4 Development Team, "PX4-Autopilot," GitHub Repository, 2025. https://github.com/PX4/PX4
Autopilot

[6] AVL-TMU Team, "sitl_gazebo_x500," GitHub Repository, 2025. https://github.com/AVL-TMU/sitl_gazebo_x500/tree/new-branch/models/x500

[7] PX4 Autopilot Team, "PX4 ROS 2 Message Translation Node," *PX4 Developer Guide*. https://docs.px4.io/main/en/ros2/px4_ros2_msg_translation_node.html

[8] PX4 Autopilot Team, "PX4 ROS 2 Communication," *PX4 Developer Guide*. https://docs.px4.io/v1.1

[9] J. Lim, "PX4 Offboard Control Example," GitHub Repository, 2025. https://github.com/Jaeyoung-Lim/px4_offboard