*14th International Defense and Homeland Security Simulation Workshop, 002*
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0363 © 2024 The Authors.
doi: 10.46354/i3m.2024.dhss.002

# Synthetic threat Dataset Generation by UAV fleet Simulation

Gennaro Pio Rimoli[1,*], Francesco Palmieri[1] and Massimo Ficco[1]

[1]University of Salerno, Via Giovanni Paolo II 132, Fisciano, Salerno, Italy

[*]Corresponding author. Email address: grimoli@unisa.it

## Abstract

In the coming years, institutions, such as law enforcement, rescue agencies, and companies, will increasingly use drone fleets to support inspection, logistics, and surveillance applications, in scenarios that may also be highly sensitive for managed data, performed tasks, or the contexts in which they operate. Security-by-design and machine learning-based models may represent the main paradigms for the design of systems resistant to cyber attacks. However, the lack of open datasets and simulation frameworks may represent the main challenge in this direction. Therefore, this paper aims to propose an open source software stack and a schema for creating synthetic standard missions, attacks, and component failures, for the generation of scenarios and data useful for addressing security threats against drone fleets. The produced data set may be used to implement machine learning-based security intrusion detectors on a centralized ground station, as well as on-board agents that may be deployed on a drone to detect anomalous behaviors of compromised drones of the fleet in real time.

**Keywords**: Cyber security, drone fleet simulation, UAV threat data generation, UAV threat dataset, Machine Learning, TinyML.

## 1. Introduction

The rapid development of drones, *i.e*, aircraft that do not have a pilot and can be controlled remotely or autonomously, has significantly impacted several fields of daily life. Drones demonstrate remarkable versatility, from transforming agricultural practices to improving surveillance capabilities, helping with disaster response, inspection of critical infrastructures, etc. [Kim et al. (2019); Donatella Dominici and Massimi (2017); Zhang et al. (2023)]. More recently, fleets of drones have been used in several fields, such as delivery services, surveillance, and military. Their simplicity of deployment and low cost make them valid technologies for supporting coordinated missions in large and hostile areas.

The increasing of drones, and in particular fleet of small Unmanned Aerial Vehicles (UAVs), has led to the development of specialized traffic management systems, such as U-space and regulations like the European Regulation 2021/664. The regulation text highlights the importance of control operations in improving a nation's internal security. These operations include police services, such as cross-border law enforcement, traffic surveillance, pursuit missions, crowd event surveillance, environmental control missions, and search and rescue missions [SES (2014)].

Despite their great potential, the use of multiple drones in real-world scenarios presents several challenges and safety risks in the operational environments. The fleet must coordinate in order to avoid collisions with the other drones or obstacles, as well as achieve the mission objectives in the shortest possible time. The weather conditions and the energy constraints can represent the main limita-

tions, which may compromise the mission if they are not taken into consideration appropriately. Moreover, due to their resource-constraint capabilities, as well as a lack of specialized security measures implementation, they may be prone to cyber attacks, which may represent serious threats to the achievement of the mission and the safety of people.

Research and development (R&D) teams have spent significant effort to improve UAV devices, with a focus on miniaturizing hardware and sensors onboard. Moreover, to manage complex missions that involve many UAVs and articulated scenarios, an increasingly massive use of machine learning and Artificial Intelligence (AI) techniques is being made. However, little attention was paid to security aspects.

During critical missions, the resistance to cyber attacks by a fleet of drones would require greater ability to make real-time on-board decisions, especially in scenarios with high transmission latency, or even lack of communications between UAVs and ground. Therefore, Tiny Machine Learning (TinyML) techniques may be exploited to implement on-board security countermeasures.

However, the use of such ML/AI techniques requires a significant amount of data to be used and to achieve satisfactory results. Given the objective difficulties in collecting data from real scenarios, and given the lack of public datasets, data is often obtained from simulated missions. In particular, when dealing with scenarios that require testing of malfunctions or attacks that could lead to the failure of existing equipment, simulators may help mitigate the problem. Therefore, specialized open tools should be provided to the scientific community, capable of generating reliable dataset that can be used to test real world scenarios.

Several academic papers in the literature discuss simulations of individual drones or fleets, using different stacks of protocols, software, and algorithms [Chen et al. (2023); Mairaj et al. (2019); Horri and Pietraszko (2022); Xavier et al. (2023); de Melo Miranda de Oliveira et al. (2019)]. They have all described the general characteristics of the simulators adopted, identifying the platform or programming language used and the available sensors from which information can be extracted. However, several works presented in the literature converge on the difficulty to use such simulators because they are often not provided with clear execution instructions and run in environments that are no longer supported, leading to their abandonment.

The objective of this paper will be to investigate the various combinations of software and protocols found in the literature and other sources, which can be exploited to implement simulations of drone fleets in terms of missions, sensors, and system data. After establishing the software stack, we will propose a schema for creating synthetic standard missions, threats, and component failures. Particular attention will be paid to the generation of scenarios and data useful for addressing security threats. The presented software stack will be specialized to produce datasets, which may be used to implement ML-based security intrusion detectors on a centralized ground station and used to detect attacks against the fleet, as well as TinyML-based on-board agents that may be used by an appropriately specialized UAV to detect anomalous behaviors of compromised fleet UAVs [Ficco et al. (2024b); Fusco et al. (2024); Rimoli et al. (2024)].

The paper is structured as follows. Sect. 2 will provide a background of what a fleet of drones is, and an overview of the software and protocols used in the literature to simulate drones, as well as the main results obtained from their use. This section will also describe the state-of-the-art threats related to the use of drones and possible attacks on on-board systems. Sect. 3 presents a generic simulation environment and its constituent elements. Sect. 4 will provide an overview of the identified software stack, its characteristics, and the decisions that led to its selection. This section also covers the system architecture and adjustments implemented to facilitate communication between multiple drones on a single platform. In addition, it explains the methodology for generating synthetic scenarios based on a mission definition. Sect. 6 presents the conclusions drawn from the study.

## 2. Background and State of the Art

### 2.1. Drones fleets

A fleet or swarm of UAVs consists of multiple drones that operate together, coordinate, and communicate with each other as a single cohesive unit to achieve a common goal. The characteristics of each drone in a swarm may be distinct, allowing us to refer to it as a nonhomogeneous swarm of drones or a heterogeneous swarm of drones if they share the same characteristics.

The characteristics of a drone are classified according to different attributes, including hardware specifications or flight dynamics. A common categorization is based on the type of wings used: fixed-wing or rotary-wing, or a combination of both, as in the case of vertical take-off and landing drones (VTOL). Wings are airfoils that help the aircraft lift off by moving through the air. Fixed-wing aircraft are designed to lift off by the forward motion of the aircraft and the aerodynamic shape of the wings. These can be subdivided into glider and powered fixed-wing. Gliders operate through the use of air currents, whereas powered fixed-wing drones depend on engine thrust for propulsion. In contrast, rotary-wing drones use rotating wings or blades to achieve lift. Both fixed- and rotary-wing drones present unique benefits and drawbacks and are applied in various contexts. For example, rotary-wing drones excel in agility compared to fixed-wing models. They possess the ability to maneuver effortlessly in any direction and can hover in one spot, which is particularly advantageous for tasks like aerial photography. However, they generally have limited flight time and, due to their design features, are not suitable for operation under adverse weather conditions or at elevated altitudes. The most common afford-

able drones on the market today are rotary-wing, such as quadcopters and bicopters, which belong to the multirotor category [Chhaya et al. (2019)].

As illustrated in Tab. 1, the operation of a drone fleet can be carried out with full or semi autonomy. These operational modes are further categorized into single- or multi-layered decision-making structures. The primary distinction between full and semi autonomy, is based on the need of external control to complete the mission. Secondary classification, into single- or multi-layered structures, examines whether decision making is distributed between individual drones or centralized in one unit per layer [Mohsan et al. (2022); Tahir et al. (2019)].

**Table 1.** List of potential missions for a swarm of drones.

| Fleet of Drones | |
|---|---|
| Fully Automated Single Layer | Fully Automated Multi Layered |
| Semi Automated Single Layer | Semi Automated Multi Layered |

Several studies have explored different command and control strategies for managing multiple drones [Saffre et al. (2021); Vásárhelyi et al. (2018)]. Among these, direct control is the easiest to implement and can be employed to direct the swarm's trajectory via formation flying. In this approach, a human operator controls one unit drone, while the rest of the swarm employs basic autonomous functions, such as relative positioning, to disperse and create a formation around the leader. This control strategy also includes preprogrammed missions, which are composed of specific flight routes predetermined by the operator at a ground control station and then uploaded to drones prior to takeoff. Alternatively, indirect control involves the swarm autonomously completing its goal without direct human supervision. This method requires advanced autonomous capabilities, including decentralized resource management and collective decision-making [ Moreno et al.]. Typically, these features are enabled by distributed control and artificial intelligence algorithms, which are facilitated by communication and collaboration between drones based on shared data.

In this scenario, several challenges arise due to the limited memory and computational power of drones, complicating tasks such as sensor/data fusion, autonomous navigation, machine learning, and remote operation technology. In situations where indirect control is preferred within a fully automated, single-layer drone swarm, each unit should have the ability to communicate via ad hoc networks and protocols that uphold the CIA triad (Confidentiality, Integrity, Availability). This involves merging data from various sensors throughout the UAV swarm to achieve a thorough perception of the surroundings. Subsequently, each drone should relay its plans to others to avoid overlaps or possible conflicts. An AI algorithm, capable of decision-making based on its learned experiences and environmental interactions, must manage all these operations, continuously enhancing its efficacy.

### 2.2. Related work and Security Threats

In the context of drone simulation, there are two primary methods to emulate a real drone: Software-in-the-Loop (SITL) and Hardware-in-the-Loop (HITL). The key distinction between them is the physical presence of a flight controller. In SITL, both the flight controller and the firmware are simulated, while in HITL, the flight controller operates on actual hardware. HITL approaches are less frequently used due to their complexity, variable costs, and scalability problems when considering large drone fleets [Zhang et al. (2020)].

The literature mainly discusses two main categories of SITL autopilots: open-source platforms such as PX4 and Ardupilot, and proprietary systems developed in MATLAB/Simulink [Horri and Pietraszko (2022); Xavier et al. (2023)] or other languages such as Python [Diller et al. (2022)]. The latter generally need to interface with a ground control station (GCS) to retrieve important data, such as sensor information, or with the flight controller through a companion computer. This article focuses on the first category of software because, like other open-source products, they are more commonly used in production applications. In simulation contexts, the academic sector often prefers Gazebo, in conjunction with PX4 or Ardupilot, as the primary dynamic flight simulator. This preference comes from the simplicity of handling the drone model, physics, the surrounding environment and meteorological conditions [Tazir et al. (2023); Nair et al. (2022); Moon et al. (2020)]. In contrast, other simulators such as AirSim [Shah et al. (2017)], CoppeliaSim, JSBSim, and FlightGear are outdated or lack native support from open-source autopilots, complicating their use.

Another aspect considered in this study are the potential threats and vulnerabilities that an individual drone or drone swarm can face. All possible threats that could affect a single drone could also impact a swarm. Such threats include sensor-related attacks such as jamming, spoofing, and interception or fabrication of data [Sergeevna et al. (2022); Rong-xiao et al. (2020)]. Furthermore, the references examine security compromises that could affect the companion computer or the artificial intelligence algorithms. Additional research has focused on various forms of attacks aimed at reducing flight time, such as those targeting battery or charging systems [Tlili et al. (2022)]. Given this study's emphasis on developing a dataset of threats for simulated drones, we reviewed articles that employ artificial intelligence algorithms to determine if similar studies already exist. Firstly, sensors have been discovered to be used in image acquisition to identify unusual activities in other drones [Svanström et al. (2021); Baig et al. (2022)], and datasets of common computer network attacks, which may include attacks not typically encountered by drones in their operational environment [Ouiazzane et al. (2023)].

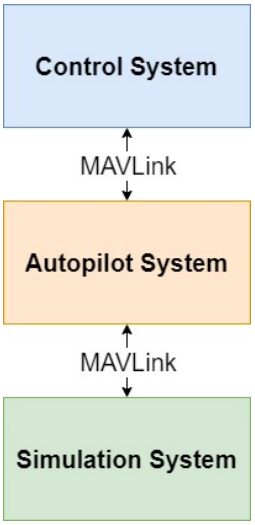


**Figure 1.** A general overview of a simulated drone environment.

## 3. Drone simulation overview

Before presenting the architecture and tools used to build the synthetic data scenarios generation system, it is important to note that the main architecture is divided into three key subsystems: the control system, the autopilot and the simulation system. Fig. 1 shows an overview of a proposed schema used to manage the simulations. The software solutions used are reflective of actual scenarios in which drone manufacturers test their products before they are marketed.

### 3.1. Autopilot System

The first system described is the autopilot, deployed within a flight controller, such as those developed by Bitcraze, Pixhawk, Sky-drones, etc. [bit; pix; sky]. The autopilot reads sensor data and adjusts the motor function to move the drone in the desired direction. The main open-source alternatives are Ardupilot and PX4, two C++-written autopilot systems that support various types of vehicle, including aircraft, ground vehicles, and water vehicles. These software packages utilize protocols such as CAN/PWM for motor communication and I2C/UART/SPI for sensor communication. The MAVLink protocol is the dominant external communication protocol, a lightweight messaging protocol based on a hybrid publish-subscribe and point-to-point design, developed by Dronecode Foundation, a vendor-neutral community dedicated to drone technologies [mav (b); dro]. MAVLink messages are XML-based, with the latest version of the protocol having just 14 bytes of overhead and supporting up to 255 systems concurrently. Telemetry data are multicast to all topic subscribers, while system configuration and mission protocols requiring guaranteed delivery are communicated point-to-point with retransmission. A distinctive feature of PX4 is its ability to develop supplementary modules using a proprietary protocol, uORB [PX4], for interthread/interprocess communication, a rapid asynchronous publisher/subscriber messaging API.

### 3.2. Control System

Autopilots can be controlled by an operator using a handheld radio, using a Ground Control Station, or through external software. As mentioned above, all autopilots communicate externally using the MAVLink protocol, which can be interfaced with using most major programming languages. To facilitate programming, Dronecode Foundation, in collaboration with MAVLink, has also developed MAVSDK – a collection of libraries that enable programming mainly in C++, but also in other languages, like JAVA, Swift, Python, and more using the gRPC protocol. Initially created by Google for the efficient invocation of Remote Procedure Calls, the gRPC protocol is managed by an instance of MAVSDK_server that handles communication with a single device [grp; mav (c)]. An alternative to manual drone control or programming is the use of a GCS, a program that can display real-time UAVs data [ qgr; ard]. A GCS can also control a drone in flight by uploading mission plans or commands and setting parameters. It is often used to monitor live video streams from UAV cameras.

### 3.3. Simulation System

The final component of the simulation architecture is the simulation environment, which can consist of various software, including Gazebo, jMAVSim, JSBSim, AirSim, and others [gaz; git (a,b); mic]. This environment comprises a communication plugin between the simulator and the flight controller, the drone model (including physical characteristics, such as weight, motor power, and flight patterns), the main sensors, and possibly additional sensors (such as cameras and radar, etc.), as well as a physics engine managing gravity, magnetic fields, and collisions with other objects in the scene. In addition, it may comprise a companion computer, which is typically equipped with a real-time operating system, such as ROS2 or Linux. This enables the computer to process the data and issue commands to the flight controller.

## 4. Proposed architecture

The proposed framework, Fig. 2, leverages the latest open source technologies developed by the drone community. One of the objectives of this article was to investigate the feasibility of using any autopilot (PX4 or Ardupilot) within the Gazebo simulator. Gazebo, an open-source software developed since 2002, has in recent years, with the support of Open Robotics, modernized its code, moving away from its monolithic architecture to a collection of loosely coupled libraries. By modifying the configuration files of the worlds in which UAVs are simulated, it was possible to preload both drone models made publicly available by autopilot manufacturers, as well as all dependencies in

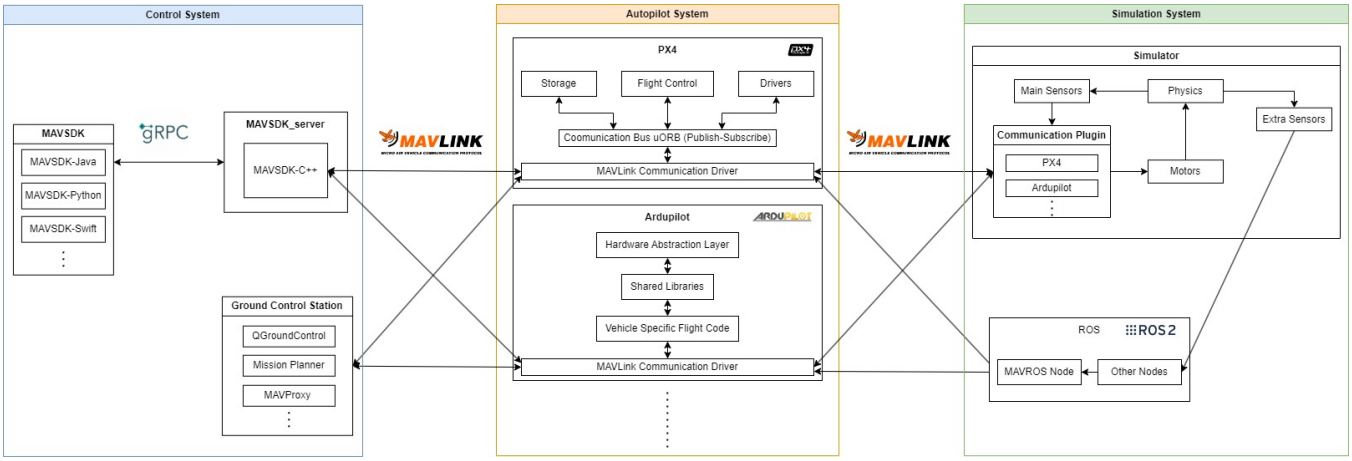


**Figure 2.** Gazebo/(Ardupilot/PX4)/QGroundControl software stack.

terms of additional plugins and settings related to general environmental conditions, including magnetic field, wind, gravity force, and the number of satellites available for position management. This, unlike other research, allows us to generate data for various types of autopilot and enhances the realism of the data.

This research also evaluates additional simulators. In particular, the suitability of simulators for the management of multiple types of drones engaged in simultaneous missions was evaluated. Tab. 2 enumerates the simulators deemed appropriate for this purpose. Considerations included the potential to enhance drone functionalities with extra sensors via the ROS system, compatibility with flight controllers, and the software's lifecycle status. Gazebo was selected, even though it matches the same capabilities of JSBSim. The primary distinction lies in Gazebo's ability to incorporate a variety of model types, as opposed to JSBSim's limitation to quadcopters only. Furthermore, Gazebo provides more precise simulations and supports a range of features with its existing modules.

**Table 2.** Simulator comparison.

| Simulator | Flight Controller | ROS | Discontinued |
|---|---|---|---|
| *Gazebo* | PX4/Ardupilot | Y | N |
| *JSBSim* | PX4/Ardupilot | Y | N |
| *Gazebo Classic* | PX4/Ardupilot | Y | N (Soon 2025) |
| *AirSim* | PX4/Ardupilot | N | Y |
| *JMAVSim* | PX4 | N | N |

To manage and generate various types of mission, it was decided to use the Dronecode Foundation QGroundControl software, which, through the MAVLink protocol, allows managing waypoints, altitude, and drone flight speed [qgr]. Each drone was then connected to an MAVLink server managed by a self-produced Java application that injected system faults. The generation of faults was handled through the "inject" method of the Failure class of the MAVSDK plugin [mav (a)]. The latter takes a failure unit and a failure type as input and returns a result. The potential options are shown in the Tab. 3 and Tab. 4, each briefly described and can be freely combined to achieve various results.

**Table 3.** A list of units that can be employed to inject failure.

| | FailureUnit |
|---|---|
| **System** | Gyro, Accelerometer, Barometer, GPS, Optical flow, Visual inertial odometry, Distance sensor, Airspeed |
| **Sensors** | Airspeed, Battery, Motor,Servo, Avoidance, RC signal, MAVLink signal |

**Table 4.** A list of failure types that can be injected.

| Failure Type | |
|---|---|
| **Value** | **Description** |
| Ok | Reset all failure |
| Off | Sets off the unit |
| Stuck | Report always the last value |
| Garbage | Report random value |
| Wrong | Report possible but inconsistent data |
| Slow | Report correct data, but slowly |
| Delayed | Report correct data, but with a packet time delay |
| Intermittent | Report correct data, but intermittently |

## 5. Experiments and results

In this work, an experimental dataset has been produced by using the proposed Gazebo/(Ardupilot/PX4)/QGroundControl software stack, to support security-oriented research activities on drones fleets. It consists of a set of missions experiencing injected malfunctions in one of the fleet drones.

The QGroundControl software, Fig. 3, expedited the creation of missions by offering a platform to manage waypoints or scan pattern missions, such as survey, corridor,


**Figure 3.** QGroundControl view of a simulated mission with three drones flying over the University of Salerno.

or structure. This software allows users to select a flight area and automatically create a navigational path, which can be adjusted later as necessary. In addition, the Region of Interest (ROI) feature enables the drone to stop at designated points for camera sensors to capture environmental data. Each mission was set to run on a simulation time that exceeded real time, which, as observed, did not compromise the data quality and facilitated more simulations in less time.

The log data was recorded and stored on GitHub [1], including the configuration files necessary to replicate the experiments. The saved logs included those of the flight controllers, MAVLink communications managed by QGroundControl, and summary logs at a 1Hz frequency in CSV format for each vehicle in the mission. These logs, while less detailed than telemetry logs, are simpler to handle and faster for data extraction.

Each scenario in the Git repository was generated using a set of repeatable processes. These scenarios vary in terms of the units targeted, the nature of the failure/attack, and the routes taken by the drones. The operations performed are as follows:

- Define $S = \{d_0, d_1, ..., d_n\}$ as the collection of drones for the simulation;
- Initiate the Gazebo simulator, which loads a modified world file to manage multiple flight controllers, and start each of the $|S|$ flight controllers separately;
- Utilize QGroundControl to set the flight routes for all drones and upload the respective missions to each;
- Select drone $d_x \in S$ as the target for the compromise and establish a connection via MAVSDK;
- Begin the simulation, integrating all loaded missions, and wait for the appropriate time to introduce the simulated attack.

Additionally, scenarios have been constructed based on identified security threats in the communication between the Ground Control Station (GCS) and the UAV using the MAVLink protocol [Ficco et al. (2024a); Xu et al. (2021); Hadi and Cao (2022); Kwon et al. (2018)]. Officially, MAVLink lacks any form of message encryption and only supports a message signing feature that verifies the authenticity of the messages. Researchers have introduced a cryptographic layer using ChaCha20 on the Ardupilot flight controller [Cha], which showed improved performance and efficiency over other encryption methods, such as AES-CBC, AES-CTR, and RC4 [Allouch et al. (2019)]. The message signing process modifies the transmitted packets by activating the incompatibility flag bit and adding an extra 13 bytes of data. This additional data includes a linkID (8 bits), a timestamp (48 bits), and a signature (48 bits), the signature being the initial 48 bits of a SHA-256 hash of the entire packet, incorporating the

1 https://github.com/iotresearchunisa/drone-fleet-simulation

secret key, header, payload, CRC, link-ID, and timestamp. MAVLink signing is not fully implemented on PX4 and works only with the Ardupilot flight controller and the ground control station Mission Planner.

As result, are also selected scenarios in which Ardupilot/PX4 flight controllers use the MAVLink protocol with the message signing functionality disabled and are compromised through message replay attacks [Hadi and Cao (2022)], ICMP DoS, and generic DoS [Kwon et al. (2018); Xu et al. (2021)]. By performing packet sniffing within the drone network, an attacker is able to capture MAVLink packets in clear and may alter the payload according to the protocol or replay particular messages until the UAV ceases communication. This second type of scenario involved a swarm of fully automated single-layer drones in which the attacker, leveraging the previously described vulnerability, successfully modified the drones' behavior, specifically making them land.

## 6. Conclusions and future works

The results of this study underscore the significant role of simulation in enhancing the security measures of UAVs. The utilization of the described stack software enables the creation of realistic scenarios of fleet operations and cyber threats, thereby facilitating the generation of a comprehensive dataset that can be exploited as a foundation for the development of robust security and AI models.

The future step include designing an open source distributed ML-based Security Information and Event Management (SIEM) system, which is able to identify and mitigate the effects of unauthorized intrusions against fleets of drones.

## 7. Acknowledgements

This work is part of the research activities realized within the projects SERICS (CUP PE00000014, under the NRRP MUR program funded by the EU - NGEUm) and FLEGREA (Federated Learning for Generative Emulation of Advanced Persistent Threats - CUP E53D23007950001, Bando PRIN 2022).

## References

Bitcraze. https://bitcraze.io/. [Accessed 14-05-2024].

Chacha20 - an authenticated encryption with additional data (aead) algorithm. https://en.wikipedia.org/wiki/ChaCha20-Poly1305. [Accessed 12-05-2024].

Class Failure. https://mavsdk.mavlink.io/main/en/cpp/api_reference/classmavsdk_1_1_failure.html. [Accessed 14-05-2024].

Gazebo. https://gazebosim.org/home. [Accessed 14-05-2024].

gRPC. https://grpc.io/. [Accessed 14-05-2024].

Home - AirSim — microsoft.github.io. https://microsoft.github.io/AirSim/. [Accessed 14-05-2024].

jMAVSim. https://github.com/PX4/jMAVSim. [Accessed 14-05-2024].

JSBSim. https://github.com/JSBSim-Team/jsbsim. [Accessed 14-05-2024].

MAVLink Developer Guide. https://mavlink.io/en/. [Accessed 14-05-2024].

mavsdk. https://mavsdk.mavlink.io/. [Accessed 14-05-2024].

Mission Planner. https://ardupilot.org/planner/. [Accessed 14-05-2024].

Pixhawk. https://pixhawk.org/products/. [Accessed 14-05-2024].

PX4 Autopilot User Guide | PX4 Guide (main) — docs.px4.io. https://docs.px4.io/main/en/index.html. [Accessed 14-05-2024].

QGC - QGroundControl. http://qgroundcontrol.com/. [Accessed 14-05-2024].

Sky-Drones Technologies Ltd. https://sky-drones.com/. [Accessed 14-05-2024].

The Dronecode Foundation. https://dronecode.org/. [Accessed 14-05-2024].

(2014). U-space conops 4th edition. https://www.sesarju.eu/node/4544. Accessed: 2024-03-11.

Allouch, A., Cheikhrouhou, O., Koubâa, A., Khalgui, M., and Abbes, T. (2019). Mavsec: Securing the mavlink protocol for ardupilot/px4 unmanned aerial systems. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 621–628.

Baig, Z., Syed, N., and Mohammad, N. (2022). Securing the smart city airspace: Drone cyber attack detection through machine learning. *Future Internet*, 14(7).

Chen, Z., Yan, J., Ma, B., Shi, K., Yu, Q., and Yuan, W. (2023). A survey on open-source simulation platforms for multi-copter uav swarms. *Robotics*, 12(2).

Chhaya, B., Jafer, S., and Proietti, P. (2019). An ontology for threat modeling and simulation of small unmanned aerial vehicles. In *Proceedings of THE 9TH INTERNATIONAL DEFENCE AND HOMELAND SECURITY SIMULATION WORKSHOP, DHSS 2019*, DHSS 2019. CAL-TEK srl.

de Melo Miranda de Oliveira, N. S., Moreira, E. M., and Rosa, P. F. F. (2019). Particle swarm optimization algorithm implementation for multiple drones control in continuous task simulation. In *The Latin American Robotics Symposium (LARS)*.

Diller, J., Hall, P., Schanker, C., Ung, K., Belous, P., Russell, P., and Han, Q. (2022). Iccswarm: A framework for integrated communication and control in uav swarms. In *Proceedings of the 2022 workshop on Eighth Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, MobiSys '22. ACM.

Donatella Dominici, M. A. and Massimi, V. (2017). Uav photogrammetry in the post-earthquake scenario: case studies in l'aquila. *Geomatics, Natural Hazards and Risk*, 8(1):87–103.

Ficco, M., Granata, D., Palmieri, F., and Rak, M. (2024a). A systematic approach for threat and vulnerability analysis of unmanned aerial vehicles. *Internet of Things*, 26:101180.

Ficco, M., Guerriero, A., Milite, E., Palmieri, F., Pietrantuono, R., and Russo, S. (2024b). Federated learning for iot devices: Enhancing tinyml with on-board training. *Information Fusion*, 104:102189.

Fusco, P., Rimoli, G. P., and Ficco, M. (2024). Tinyids - an iot intrusion detection system by tiny machine learning.

Hadi, H. J. and Cao, Y. (2022). Cyber attacks and vulnerabilities assessment for unmanned aerial vehicles communication systems. In *2022 International Conference on Frontiers of Information Technology (FIT)*, pages 213–218.

Horri, N. and Pietraszko, M. (2022). A tutorial and review on flight control co-simulation using matlab/simulink and flight simulators. *Automation*, 3(3):486–510.

Kim, J., Kim, S., Ju, C., and Son, H. I. (2019). Unmanned aerial vehicles in agriculture: A review of perspective of platform, control, and applications. *IEEE Access*, 7:105100–105115.

Kwon, Y.-M., Yu, J., Cho, B.-M., Eun, Y., and Park, K.-J. (2018). Empirical analysis of mavlink protocol vulnerability for attacking unmanned aerial vehicles. *IEEE Access*, 6:43203–43212.

Mairaj, A., Baba, A. I., and Javaid, A. Y. (2019). Application specific drone simulators: Recent advances and challenges. *Simulation Modelling Practice and Theory*, 94:100–117.

Mohsan, S. A. H., Khan, M. A., Noor, F., Ullah, I., and Alsharif, M. H. (2022). Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6(6).

Moon, S., Bird, J. J., Borenstein, S., and Frew, E. W. (2020). A gazebo/ros-based communication-realistic simulator for networked suas. page 1819 – 1827. Cited by: 11.

Moreno, A., de la, L., Risco-Martín, J. L., Besada-Portas, E., and Aranda, J. Devs-based validation of uav path planning in hostile environments.

Nair, N., Sareth, K., Bhavani, R. R., and Mohan, A. (2022). Simulation and stabilization of a custom-made quadcopter in gazebo using ardupilot and qgroundcontrol. *Smart Innovation, Systems and Technologies*, 292:191 – 202. Cited by: 2.

Ouiazzane, S., Addou, M., and Barramou, F. (2023). A zero-trust model for intrusion detection in drone networks. *International Journal of Advanced Computer Science and Applications*, 14(11).

Rimoli, G. P., Boi, B., Fusco, P., Esposito, C., and Ficco, M. (2024). Tiny federated learning with blockchain for privacy and security preservation of mcu-based iot applications.

Rong-xiao, G., Ji-wei, T., Bu-hong, W., and Fu-te, S. (2020). Cyber-physical attack threats analysis for uavs from cps perspective. In *2020 International Conference on Computer Engineering and Application (ICCEA)*, pages 259–263.

Saffre, F., Hildmann, H., and Karvonen, H. (2021). The design challenges of drone swarm control. In Harris, D. and Li, W.-C., editors, *Engineering Psychology and Cognitive Ergonomics*, pages 408–426, Cham. Springer International Publishing.

Sergeevna, B. E., Dmitrievna, M. V., Igorevich, S. O., Alexandrovich, L. A., and Bogdanovich, M. A. (2022). Survey of security threat and attack scenario for commercial uavs. In *2022 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–6.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles.

Svanström, F., Alonso-Fernandez, F., and Englund, C. (2021). A dataset for multi-sensor drone detection. *Data in Brief*, 39:107521.

Tahir, A., Böling, J., Haghbayan, M.-H., Toivonen, H. T., and Plosila, J. (2019). Swarms of unmanned aerial vehicles — a survey. *Journal of Industrial Information Integration*, 16:100106.

Tazir, M. L., Mancas, M., and Dutoit, T. (2023). From words to flight: Integrating openai chatgpt with px4/gazebo for natural language-based drone control. page 215 – 222. Cited by: 0; All Open Access, Bronze Open Access.

Tlili, F., Fourati, L. C., Ayed, S., and Ouni, B. (2022). Investigation on vulnerabilities, threats and attacks prohibiting uavs charging and depleting uavs batteries: Assessments & countermeasures. *Ad Hoc Networks*, 129:102805.

Vásárhelyi, G., Virágh, C., Somorjai, G., Nepusz, T., Eiben, A. E., and Vicsek, T. (2018). Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20).

Xavier, F. C. S., Santos, S. R. B. d., and Givigi, S. N. (2023). Multifixed-wing-uav software-in-the-loop simulation platform for system design. *IEEE Systems Journal*, 17(4):6724–6735.

Xu, H., Zhang, H., Sun, J., Xu, W., Wang, W., Li, H., and Zhang, J. (2021). Experimental analysis of mavlink protocol vulnerability on uavs security experiment platform. In *2021 3rd International Conference on Industrial Artificial Intelligence (IAI)*, pages 1–6.

Zhang, Y., Wang, Q., Shen, Y., Wang, T., Dai, N., and He, B. (2023). Multi-auv cooperative search method based on dynamic optimal coverage. *Ocean Engineering*, 288:116168.

Zhang, Z., Yang, W., Shi, Z., and Zhong, Y. (2020). Hardware-in-the-loop simulation platform for unmanned aerial vehicle swarm system: Architecture and application. In *2020 39th Chinese Control Conference (CCC)*, pages 58–64.