オープンソースドローンのシミュレーションを活用したセキュリティテストの検討

惣島 雅樹[1,a)]　須崎 有康[1,b)]

概要：近年のドローンの普及に伴い、複数のオープンソースのフライトコントローラが開発されている。中でも広く利用されているプラットフォームは、NuttX 上で動作する PX4 と、ChibiOS 上で動作する ArduPilot である。これらは通常、ARM Cortex-M4（168 MHz / 256 KB RAM/ 2 MB Flash）などのマイコンを搭載した Pixhawk 等のハードウェアで動作する。Gazebo Classic は、これらのフライトコントローラの挙動を再現可能なシミュレータであり、飛行制御に必要なジャイロスコープ、加速度センサ、磁力計、気圧計などのセンサ入力と、制御結果に基づくドローンの物理的な動作を仮想的に再現できることから、フィードバック制御や飛行安定性の検証に活用されている。本研究では、Gazebo Classic を用いたセキュリティテスト手法の検討を目的とし、フライトコントローラをシミュレータと同一ホスト上で実行する SITL（Software-in-the-Loop）環境と、実機の Pixhawk に実装して動作させる HITL（Hardware-in-the-Loop）環境を構築した。両環境に対してセンサ入力へ異常値を注入し、急激な落下や上昇といった挙動を誘発した結果、実行環境の違いに起因する挙動の差異が観測された。本稿では、センサデータの更新周期の差異などに注目し、SITL、HITL の違いがシミュレーション上のセキュリティテストの結果に与える影響等を考察するとともに、シミュレーションを用いたセキュリティテストの信頼性を向上させるための方法について議論する。

キーワード：ドローン、オープンソースフライトコントローラ、SITL、HITL

Exploring Security Testing Using Drone Simulation

Masaki Soshima[1,a)]　Kuniyasu Suzaki[1,b)]

Abstract: With the increasing popularity of drones, several open-source flight controllers have been developed. Among the most widely used platforms are PX4 running on NuttX and ArduPilot running on ChibiOS. These systems typically run on Pixhawk hardware equipped with microcontrollers such as the Arm Cortex-M4 (168 MHz, 256 KB RAM, 2 MB Flash). The behavior of these flight controllers can be simulated using Gazebo Classic, which supports both Software-in-the-Loop (SITL) and Hardware-in-the-Loop (HITL) configurations. In SITL, the controller runs entirely in a simulated environment, whereas HITL uses a real flight controller. Gazebo simulates four key sensors—gyroscope, accelerometer, magnetometer, and barometer—as well as the drone's physical responses, enabling the verification of feedback control and flight stability. In this study, we constructed both SITL and HITL environments and injected abnormal values into sensor inputs to induce rapid ascent or descent. As a result, we observed behavioral differences between the two environments. This paper analyzes the impact of such differences, including factors such as sensor update rates, on security testing outcomes, and discusses methods for improving the reliability of simulation-based security testing using Gazebo Classic.

Keywords: Drone, Open Source Flight Controller, SITL, HITL

1. はじめに

近年、ドローンの利活用は急速に拡大しており、日本国内の市場規模は 2028 年度に 9000 億円を超えると予測されている [1]。その用途は、物流、農業、点検、土木・建設、防犯などの産業用途に加え、個人による空撮等多岐にわたり、今後もますます普及してくことが見込まれる。国土交通省では有人地帯での目視外の飛行を「レベル 4 飛行」と定め、ドローンによる山間部や離島などへの物資の配送、災害時の救助活動、橋梁・工場設備などの保守点検などを実現するために、ドローンの機体の安全性を確保する認証制度の創設などを進めている [2]。

ドローンの普及に伴い国内外でドローンに関連する事件・事故が発生しており、国土交通省によると、ドローンに関する重大な事故等の報告が義務付けられた 2022 年 12 月以降、2025 年 3 月 18 日までの時点で 190 件を超える事故等が報告されている [3]。このようなドローンを取り巻く事件・事故の大半は運用者の誤操作や、天候等の影響、機体の故障等に起因するものであるが、悪意ある操作による事件や、悪意ある第三者が故意に他者のドローンの運用を妨害する事例も報告されている。特に、近年のドローンは複数のセンサや GPS、カメラの映像等を用いた高度な自動飛行制御に加え、ネットワークへの接続による目視外からの操作・情報収集等多様な機能を提供する一方で、それらを起点とした、サイバー攻撃のリスクが指摘されている。例えば、Pleban ら [4] はフランス Parrot 社の AR.Drone2.0 において、操縦用に接続するための Wi-Fi にパスワードが設定されていないうえ、複数の未保護ポートが解放されていることを確認しており、これらの脆弱性を利用して telnet 接続によりパスワードなしで root シェルにアクセスし、飛行中のドローンのを制御・奪取できることを示した。他にも、ドローンを遠隔から操作するための RC 電波を起点とした攻撃 [5] やセンサ類 [6]・カメラ [7]・GPS[8] を起点とした攻撃などが報告されており、これらのセキュリティリスクへの対策はドローンの普及に不可欠となっている。

このようなドローンを安全に運用するためには安定した飛行の制御が重要となるが、ドローンの飛行の制御はフライトコントローラと呼ばれるシステムが担っており、PX4 や Ardupilot に代表されるようなソフトウェア、Pixhawk に代表されるハードウェアが、それぞれオープンソースのコミュニティで開発され、継続的に改善・保守されている。さらに、これらのオープンソースのフライトコントローラは Gazebo などのオープンソース 3D ロボティクスシミュレータを用いて、飛行特性やセンサ挙動を高精度に再現し、その飛行制御アルゴリズムを安全かつ効率的に検証することができる。

本研究では、これらのオープンソースのフライトコントローラシステムとシミュレータを用いて、ドローンのフライトコントローラシステムのサイバー攻撃耐性を安全かつ、効率的に検証するセキュリティテストの手法について検討する。具体的には、フライトコントローラのソフトウェアをシミュレータと同じコンピューター上で動作させる SITL（Software-In-The-Loop）と、PIxhawk 等の実機のハードウェア上で動作させる HITL（Hardware-In-The-Loop の環境を構築し、それぞれの環境においてドローンの主要なセンサに異常値を注入するテストを実施したところ、攻撃後の挙動に差異が生じることを明らかにした。本稿では、この要因として、ソフトウェアの実行環境やセンサ値の更新周期などの時間依存要因に注目し、シミュレーションを用いたセキュリティテストの信頼性を向上させるための方法について議論する。

2. 構成要素

ドローンのフライトコントローラシステムの概要を図 1 に示す。飛行制御をおこなうソフトウェアは RTOS 上で動作し、センサや GNSS からの入力をリアルタイムに処理してモーターの出力を制御することで機体の動作を制御している。本稿で取り扱うドローンは以降、４つのモーターで動作するクアッドローター型のものを指すこととする。

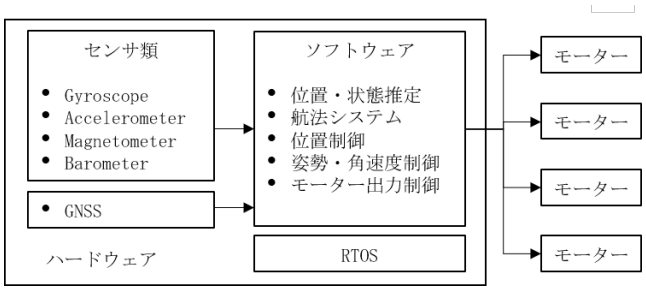

図 1: フライトコントローラシステムの概要

2.1 ソフトウェア

オープンソースフライトコントローラのソフトウェアとしては、PX4、Ardupilot、Paparazzi、LibrePilot、Betafilght、iNAV などがあり [9]、特に多様な機体タイプをサポートしている PX4 と Ardupilot は主要なソフトウェアとして、産業用途も含め幅広く利用されている。PX4 は Linux Foundation 傘下の Dronecode Foundation が中心となり開発・保守しており [10]、モジュール分割アーキテクチャを採用している。各モジュール間の内部通信には publish/subscribe 型のメッセージバスである uORB を利用し [11]、各モジュール間の疎結合化を実現することで、モジュールの追加・交換や機能変更を他の部分への影響を最小限で行えるよう

[1] 情報セキュリティ大学院大学
Graduate School of Information Security, Institute of Information Security
a) mgs247502@iisec.ac.jp
b) suzaki@iisec.ac.jp

にしている。外部との通信はオープンスタンダードの軽量メッセージプロトコルである MAVLink(Micro Air Vehicle Link) を利用し、フライトコントローラと地上局ソフトウェア間でテレメトリ情報の取得や制御コマンドの送信を行うことができる [12]。さらに、地上局ソフトウェアを利用することで、ドローンの飛行モードの変更やパラメータ設定、自動飛行の飛行経路作成・機体へのアップロードが可能である。また、PX4 は SITL および HITL のシミュレーションに対応しており、Gazebo、jMAVSim などのシミュレータと地上局ソフトウェアの QGroudControl を PX4 に接続して動作する [13]。

一方で、Ardupilot はコミュニティ主導で開発・保守されており [14]、主に ChibiOS RTOS 上で動作する。アーキテクチャは、ハードウェア依存部分を抽象化する AP_HAL(Hardware Abstraction Layer)、共通的に利用される基盤ライブラリ群（shared libraries)、機体タイプごとの飛行制御用フライトコードというレイヤ構造で構成されている。姿勢推定、センサ管理といった共通機能は shared libraries にまとめられ、これらが機体タイプごとの飛行制御用フライトコードから直接呼び出されて機体全体の制御システムを構成する [15]。また、モジュール間の連携は関数呼び出しや共有データ構造を介して行われ、タスクスケジューラが各処理の実行タイミングを制御している [16]。外部との通信には、PX4 と同様に MAVLink プロトコルが用いられ、テレメトリ情報の取得や制御コマンドの送信を行うことができる。また、Ardupilot は SITL のみに対応しており、Gazebo,AirSim などの外部シミュレータと連携し、ハードウェア抽象化層を PC 向けの SITL 実装に置き換えることで、飛行制御コードや共通ライブラリを直接実行する [15][17]。

2.2 ハードウェア

オープンソースのフライトコントローラのハードウェアとしては、Pixhawk 系シリーズ（CUAV を含む)、Lisa、Apogee、Naze、CCD32、Kakute、Omnibus などがある [9]。特に Pixhawk 系シリーズは各センサからのデータ取得、制御演算、通信インターフェイスなどを担う中核処理ユニット FMU（Flight Management Unit）のシリーズごとに複数のバージョンが開発されており、PX4、Ardupilot といった主要なソフトウェアをサポートしていることから、幅広く利用されている。また、Pixhawk 系シリーズはボード内にプロセッサと Accelerometer、Gyroscope、Magnetometer、Barometer などのセンサを複数搭載しており、シリーズごとにその特性が異なる。Pixhawk 系シリーズの FMU は世代を重ねるごとにプロセッサ性能、メモリ容量、センサ冗長性などの面で進化しており、例えば、Pixhawk 系シリーズで初めて市販化された FMUv2 の Pixhawk はセンサ取得、状態推定、姿勢・位置制御などの飛行に関する制御一式を担当する Main プロセッサとして STM32F427 (Cortex-M4,180MHz,256KB RAM) と PWM 出力の生成や RC 入力の取得等を担当する I/O プロセッサとして STM32F100(Cortex-M3,24MHz, 8KB RAM) を搭載しており、Gyrocope と Accelerometer のセンサをそれぞれ 2 系統搭載した構成となっている [18]。また、最新世代の標準モデルである FMUv6x の Pixhawk6x はメインのプロセッサとして、STM32H753 (Cortex-M7,480MHz,1MB RAM) を I/O プロセッサとして STM32F103 (Cortex-M3,72MHz,64KB RAM) を搭載、3 系統の Gyroscope、Accelerometer と 2 系統の気圧計を搭載した構成となっている [19]。これらの性能向上により姿勢制御や航法計算の精度が高まり、外乱に対する応答性や安定性が大きく改善されており、複雑な自立飛行ミッションや長時間・長距離の運用、高信頼な飛行が可能となっている。

3. シミュレーション

3.1 SITL(Software In The Loop)

SITL の構成図を図 2 に示す。SITL は、フライトコントローラの実機ハードウェアを用いず、飛行制御を行うフライトスタックそのものをホスト PC 上で実行し、センサ・アクチュエータ I/O をシミュレータが生成/消費するデータで代替し、飛行制御アルゴリズムやシステム全体の動作を、物理環境を模擬しながら安全かつ迅速に検証することを可能とする手法である。

PX4 では、シミュレータがセンサ値を MAVLink メッセージで送信すると、PX4 はこれを受信し uORB を用いて各処理モジュールに渡す。その後、スケジューラが設定した制御ループ周期に従い、状態推定や制御計算を行い、出力結果を再度 MAVLink メッセージでシミュレータに送

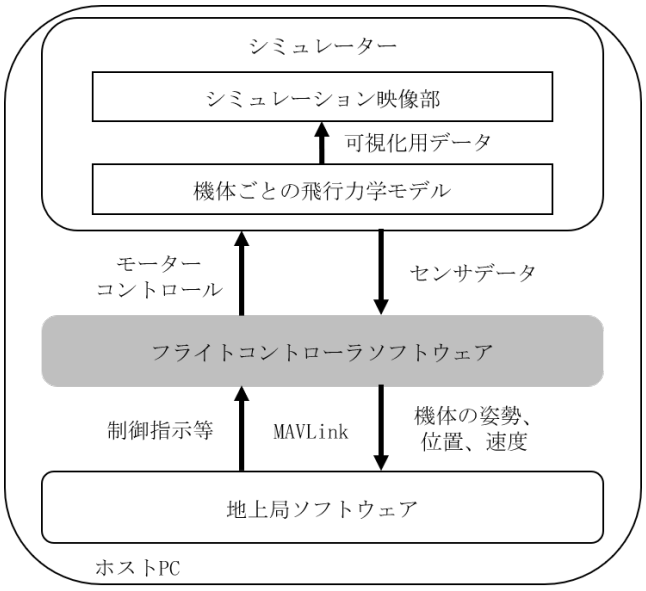

図 2: SITL の構成　（著者作成：参考 [13][20]）

信する [13]。Nuttx RTOS 上でではカーネルレベルのタスク・ワークキューと RTOS タイマーにより優先度と周期制御を行う [11]。対して、SITL（POSIX ／ Linux）環境では px4_task_spawn_cmd() により pthread スレッドを生成し、POSIX のリアルタイムスケジューリングポリシー（例：SCHED_FIFO）を設定することで、RTOS におけるタスクモデルの動作を再現している [11]。さらに、PX4 と Gazebo Classic を用いた SITL ではロックステップ機構により PX4 とシミュレータ間で理想的に時間同期されるため [13]、センサ値や制御入力の伝達に遅延やジッタが生じない。

一方 Ardupilot では、外部シミュレータが生成した物理状態量を、UDP インターフェースを通じて受信し、SITL 内のシミュレーション専用センサライブラリに入力して IMU・GPS・磁気・気圧などの擬似センサ値を生成して、共通の基盤ライブラリ群、飛行制御用フライトコードに渡された後、状態推定や制御計算を行いモータ出力をシミュレータに返す。STIL ではホスト PC 上の実行ファイルとして動作し、SITL 用のハードウェア抽象化層が pthread によって各スレッドを生成し、これらはリアルタイム優先度をもつスレッドとして動作する [17][16]。

3.2 HITL(Hardware In The Loop)

HITL の構成図を図 3 に示す。HITL はフライトコントローラの実機ハードウェアをシミュレータに接続し、フライトスタックを実機ハードウェア上で実行し、センサ・アクチュエータの入出力は SITL と同様にシミュレータが生成/消費するデータで代替し、実機と同じ実行環境・制約条件下での挙動を安全かつ効率的に評価することを可能とする手法である。HITL に対応する PX4 では、起動時に Accelerometer や GPS などのセンサドライバや物理 PWM ドライバが立ち上がらず、シミュレータから送られるメッセージを受信して uORB トピックに流し込み、その後は実機と同様に処理されたのち、MAVLink メッセージとして、モータの出力結果をシミュレーションに送信する [21]。PX4 の HITL では実機と同じ Nuttx RTOS 上で動作するため、その優先度や周期実行の制御は実機と同じしくみであるが、各センサの周期実行に伴うハードウェアタイマの割り込みなどは発生しない。

4. セキュリティテスト

シミュレーション環境における信頼性の高いセキュリティテスト手法を検討するため、SITL,HITL それぞれの環境でドローンの正常な飛行を妨げるようなテストを実施し、その挙動を比較する。環境の差異がセキュリティテストの結果に与える影響を比較するため、各テストは PX4、Ardupilot の SITL と PX4 の HITL（FMUv2：HITL に対応）の 3 環境で実施した。また、セキュリティテストでは、

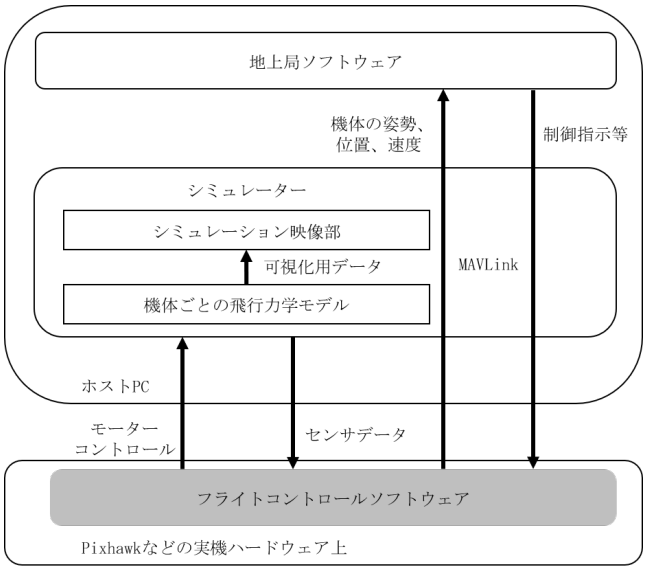

図 3: HITL の構成　（著者作成：参考 [21]）

PX4 の HITL 及び PX4、Ardupilot の SITL の全てに対応する Gazebo Classic をシミュレータとして使用した。セキュリティテストの環境を表 1 示す。

表 1: セキュリティテスト環境

ホスト PC	AMD RYZEN 9 9900X, Memory 64GB
ホスト OS	Ubuntu 24.04
シミュレーション環境	Ubuntu 20.04 on VirtualBox, 12 CPU, Memory 32GB
ソフトウェア	PX4 ver1.16.0,ArduPilot ver4.7.0dev
ハードウェア	FMUv2(Pixhawk2.4.8)
地上局ソフトウェア	QGroundControl v4.4.4

Gyroscope、Accelerometer はドローンの自律飛行における必須のセンサとなっており、これらのセンサ値に異常値が注入された場合、ドローンの安定的な飛行に影響を及ぼすことが想定される。図 4 にドローンの自律飛行における制御の概要 [22] を示す。

ドローンの制御は Accelerometer と Gyroscope のセンサ値とそこから推定した値を用いた PID 制御が基本となっている。PID 制御は目標値と現在の値の差を比例（Proportional）・積分 (Integral)・微分（Derivative）の三つの要素で補正することで、迅速かつ正確に安定したフィードバック制御を行う手法である。オープンソースのドローンは、まず外側のループで目標位置と現在位置の差から P 制御で速度目標を生成し、速度目標と実速度の差から加速度の目標を PID 制御で生成する。この加速度と機体の向きの目標をもとに、内側のループでクォータニオンと呼ばれる状態数と推力の要求値に変換され、クォータニオンの目標値が P 制御され、角速度の目標値を生成し、角速度の目標値が PID 制御され各軸のトルクの要求値を生成し、推力の目

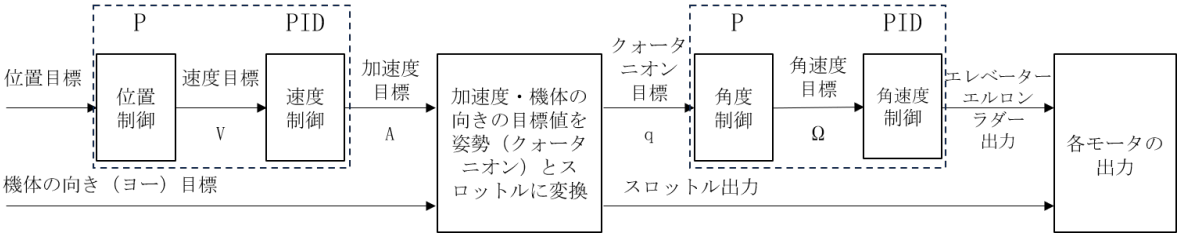

図 4: ドローンの自律飛行制御の概要 [22]

標値とともに各モータの出力制御に用いられる。ドローンは 4 つのモータの出力のみで飛行にかかわる全動作を制御しており、各モーターの強弱のバランスで図 5 に示す方向の姿勢と全体の出力を制御することで、上昇下降、前後左右への移動、回転などの運動を実現している。

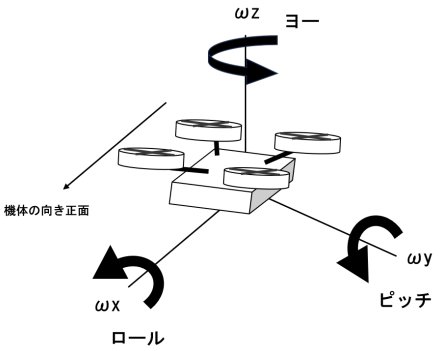

図 5: ドローンの向きと姿勢角

したがって、これらのフィードバック制御に用いられるセンサ値へ異常値を注入した場合、機体の安定的な制御に大きな影響を及ぼすものと考えられる。

実際に、Son ら [6] は MEMS ジャイロスコープが特定の音響振動に共振すると誤った値を出力する特性を利用し、ドローンに異常な角速度データを注入できることを確認しており、この異常値によりドローンの飛行制御が乱れ、短時間で制御不能・墜落させることが可能であると報告している。また、ドローンの飛行中にセンサが何かしらの原因で故障した場合、当該センサからの入力を失うケースや、異常値が注入され、同様に飛行に異常をきたすようなケースも想定される。このようなセンサへの異常値が注入された場合のドローンの挙動をシミュレーション上で検証するため、Gyroscope、Accelerometer の 2 種類のセンサに対して物理的な外乱要因や故障による異常値の注入を想定したテストを実施する。同じ条件下で挙動を比較するため、ドローンが離陸後高度 10m でホバリング（高度維持モードで飛行）している安定状態で固定の異常値を注入し、その後の挙動を確認する。

4.1 Gyroscop への異常値注入

PX4 や Ardupilot の自律飛行では 3 軸の Gyroscope で計測された角速度ベクトル（ω x, ω y, ω z）を状態推定アルゴリズム EKF（Extended Kalman Filter）を用いた機体の状態推定やリアルタイム姿勢制御に使用している。特に姿勢制御は PX4 や Ardupiot が制御する実機のドローン上では最大 1kHz 程度で動作しており、異常値の注入は即座に安定したホバリング状態の維持に影響を与えることが推測される。異常値注入は、故障や物理的な外乱で一定の値に固定されるケースを想定し、各軸を一定の値に固定した際の挙動を確認する。ホバリング状態のドローンは高度を維持してその地点から動かないような動作をするため、理想的には姿勢が一定に保たれ、全軸の角速度が 0 となるはずである。したがって、0 を基準として異常値の注入テストを実施する。

4.2 Acclerometer への異常値注入

3 軸 Accelerometer で計測された加速度ベクトル（ax, ay, az）は EKF を用いた速度や位置などの機体の状態推定に用いられる。EKF で推定された速度は、飛行地点、飛行速度に到達するために必要となる各 3 軸の速度の PID 制御に用いられ、姿勢の制御に必要となる目標加速度を計算する。図 4 で示したように、これらの制御はモータの出力を直接制御する角速度の PID 制御の上位に位置しており、異常値注入による影響は Gyroscope ほど直接的な影響は受けないと予測されるが、EKF の状態推定に用いられることから、安定したホバリング中のドローンに対して、誤った推定結果が緩やかにホバリングの維持に影響を与えると推測される。Accelerometer への異常値注入も Gyrosocpe と同様に各軸をホバリング中の理想値である（0,0,-9.8) を基準として、各値の加速度の値を固定した際の挙動を確認する。なお、重力加速度についてはシミュレータ側で定義された値として-9.8 を用いた。

5. 結果及び考察

Gyroscope や Acclerometer に固定の異常値を注入した結果、すぐに墜落する、攻撃開始後は影響がないが一定時間後に急降下し墜落する、一時的に制御を失うものの高度を保とうとする、全く影響がないといった様々な挙動が観測できた。それぞれの結果の比較と考えられる要因を本章に記載する。

5.1 Gyroscop への異常値注入結果

Gyroscope の各軸を 0 に固定する異常値注入攻撃の結果を表 2 に示す。ここで×は攻撃開始後に制御を失い、最終的に急降下により墜落したことを示す。それぞれの軸の角速度を 0 に固定する異常値を注入したところ、すべて墜落する結果となった。

	ω_x	ω_y	ω_z
Ardupilot SITL	×	×	×
PX4 SITL	×	×	×
PX4 HITL (FMUv2)	×	×	×

表 2: Gyroscope への異常値注入攻撃（各軸を 0 に固定）の結果

図 6 に角速度の PID 制御の概要 [22] を示す。Gyroscope から得た角速度の実測値はモーターの出力に直結する角速度の PID 制御に直接用いられており、ホバリング中の機体の微小振動による理想値とのずれも、Gyroscope の実測値のフィードバック制御で打ち消すことで姿勢が保たれている。この微小な振動の状態を反映せずに 0 に固定してしまうと、特に、ロール、ピッチ方向については、飛行するための十分な推力を受ける姿勢を保つことができなくなるため、即座に墜落してしまったものと考えられる。

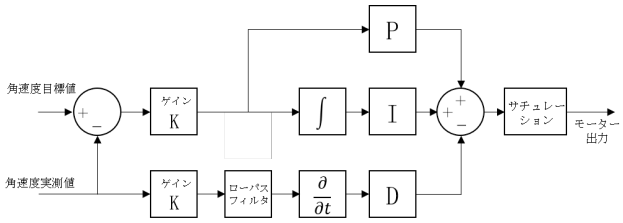

図 6: 角速度の PID 制御の概要 [22]

一方で、図 7 に示すように、ω z の固定化は一定の時間の後に急激な墜落を招く結果となった。ヨー方向の固定は、機体のホバリング地点でのヨー方向への回転を招くが、この状態においても機体は飛行するための上向きの一定の推力を維持することができる。その後、回転をし続けることにより姿勢が破壊され最終的には急降下に至るが、ω x、ω y 方向と比べると、敏感な影響を受けないことが分かる。

5.2 Acclerometer への異常値注入結果

Accelerometer を理想値（0,0,-9.8）に固定する異常値注入攻撃の結果を表 3 に示す。それぞれの軸の理想値に固定する異常値を注入したところ、z 軸を除いて影響を受けない結果となった。ここで z 軸の結果における〇は影響がないことを、×は機体が上昇し続け制御不能になったことを示し、△は制御不能に陥ったものの最終的に着陸に至ったことを示す。

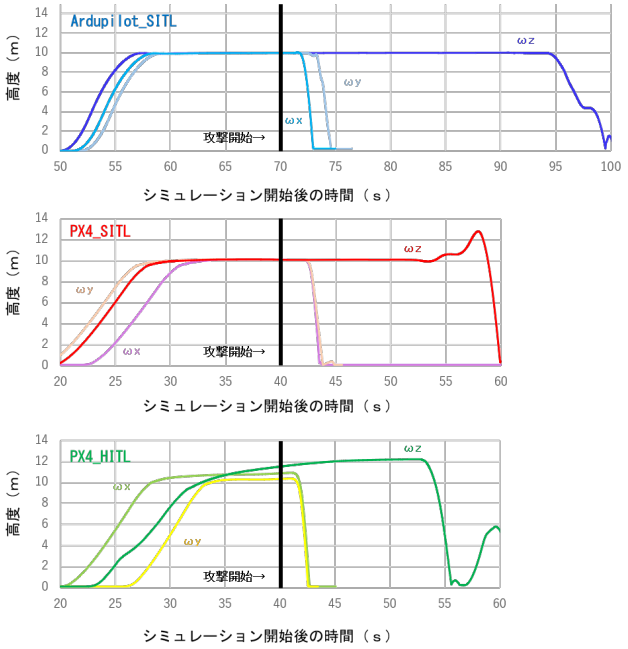

図 7: Gyroscope への異常値注入攻撃（各軸を 0 に固定）の際の高度推移

	x	y	z
Ardupilot SITL	〇	〇	△
PX4 SITL	〇	〇	×
PX4 HITL (FMUv2)	〇	〇	×

表 3: Accelerometer への異常値注入攻撃（(0,0,-9.8）に固定）の結果

z 軸の固定については x、y 方向に制御を失うことはなかったが、図 8 に示すように z 軸方向に不安定に上昇や下降をする挙動が確認された。

x、y 方向への移動は主に姿勢の制御によって実現されており、このとき、これらの方向の加速度センサの値は最終的に角速度の目標値を出力するために用いられ、姿勢の直接的な制御には Gyroscope が用いられる。したがって、Gyroscope が正常に動作している状態であれば、実際の各軸の加速度との乖離が小さければ飛行動作に影響は生じないものと考えられる。一方で、z 軸については上下の動作、すなわち機体の推力を決めるモータの制御に直接的に影響する。したがって、実際の加速度に近い理想値に固定したとしても、変動する加速度をリアルタイムに正確に取得できなければ安定した動作をすることができない。また、PX4 では SITL、HITL ともに機体が上昇をつづけ制御不能になる結果となったが、Ardupilot の SITL では攻撃開始から高度に影響が出るまで約 5 秒あり、一時的に 10m 程度機体が上昇するが、その後徐々に高度を下げ、着陸する

結果となった。この PX4 と Ardupilot の違いは、両者の状態推定や機体の制御の仕組みに起因し、Ardupilot の方が z 軸の方向の加速度の異常値に対して耐性を有する可能性がある。

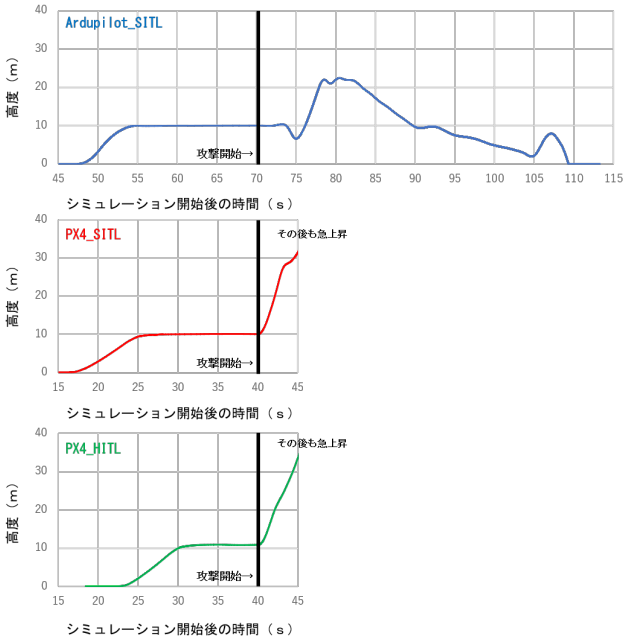

図 8: Z 軸の加速度を-9.8 に固定したときの挙動

さらに、平均値への固定で影響を受けなかった x 軸と y 軸について、値を変化させながら挙動を確認したところ、4.3m/s^2 で PX4 の SITL 環境では、制御を失い墜落に至ったのに対して Ardupilot や PX4 の HITL では高度を保とうとする挙動が確認できた。その様子を表 4 と図 9 に示す。特に PX4 の SITL と HITL で同じソフトウェアを用いているにも関わらず、その挙動に差異が生じたことに注目する。

	結果	説明
Ardupilot SITL	△	攻撃開始後制御を失うが、その後高度を保持しようとする
PX4 SITL	×	攻撃開始後制御を失いすぐに落下
PX4 HITL (FMUv2)	△	攻撃開始後制御を失うが、その後高度を保持しようとする

表 4: x 軸の加速度を 4.3m/s^2 に固定したときの攻撃結果

これらの差は、主に SITL と HITL におけるソフトウェアの実行環境、通信経路、および時間同期方式の違いに起因すると考えられる。PX 4 の SITL ではソフトウェアが

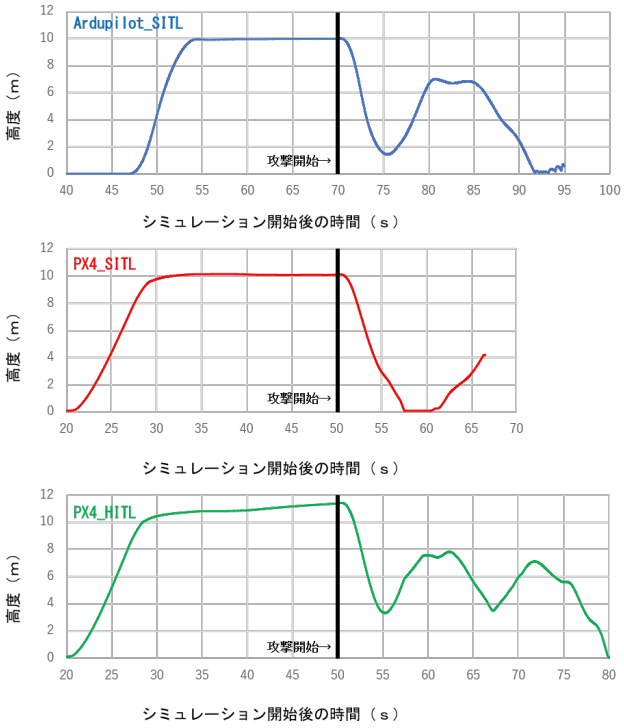

図 9: x 軸の加速度を 4.3m/s^2 に固定したときの HITL と SITL の挙動

ホスト OS 上のユーザープロセスとしてシミュレータと同一プロセス空間で動作し、ロックステップ機構により理想的に時間同期されるため、センサ値や制御入力の伝達に遅延やジッタが生じない。一方 HITL 上ではシミュレータで生成されたセンサデータが MAVLink を介して FMU に送信され、RTOS 上のタスクスケジューリングやシリアル通信のバッファ、処理負荷による遅延など複数の時間依存要因が存在する。フライトコントローラの状態推定に用いられている EKF では加速度・角速度に基づく状態予測値が GPS や気圧計、磁力計などの観測値と矛盾する場合、その影響を減じる仕組みとなっているが [23]、時間依存要因によりセンサ値の観測時間にばらつきがある HITL 環境では、予測値と観測値の矛盾が発生しやすく、固定加速度のような攻撃によるセンサ値が SITL よりも取り込まれにくい。その結果、SITL では異常値を制御に用いてすぐに墜落したのに対して、HITL では墜落しないという挙動の差が発生したものと考えられる。実機のドローンもこのような時間依存の要因が存在するため、機体のリアルタイム制御など時間依存要因が影響するセキュリティテストにおいては、HITL の環境の方がより実機に即したテストを実施できるものと考えられる。

6. まとめ

オープンソースのフライトコントローラである PX 4 や Ardupilot などのソフトウェアおよび Pixhawk などのハードウェアとシミュレータを用いて、主要なセンサに異常値

を注入するセキュリティテストを実施したところ、センサの種別、環境に応じて異なる結果を得た。ドローンの姿勢制御に用いられる角速度を固定する攻撃に対しては、すべての環境で墜落を招く結果となった一方で、加速度を固定する攻撃では、加速度の向きと環境によって差が生じた。特に、平面方向（x 軸、y 軸方向）にドローンの実際の状態と大きく異なる加速度の固定値を注入したところ、PX4 の SITL では急激な墜落を招いたが、HITL では高度を維持しようとする挙動の差異が確認できた。この差異は、SITL でセンサデータの更新や制御入力の伝達が一定であるという実機との乖離に起因するものと考えられる。したがって、RTOS 上のタスクスケジューリングやシリアル通信のバッファ、処理負荷による遅延などの時間依存要因が影響するセキュリティテストについては、HITL を用いることで、実機に近いタイミングでの処理を再現できるため、時間依存要因を適切に考慮した高信頼なセキュリティ評価が可能となるものと考える。

参考文献

[1] インプレス総合研究所. 2023 年度ドローンビジネス市場規模は前年比 23.9%増の 3854 億円、2028 年度は 9000 億円超へ 橋梁、大規模建造物に加え狭小空間での点検が進む『ドローンビジネス調査報告書 2024』3 月 22 日発売. https://research.impress.co.jp/topics/list/drone/685, 2024. [Accessed 08-08-2025].

[2] 国土交通省. 無人航空機レベル 4 飛行ポータルサイト. https://www.mlit.go.jp/koku/level4/, n.d. [Accessed 08-08-2025].

[3] 国土交通省. 無人航空機に係る事故等報告一覧. https://www.mlit.go.jp/common/001585162.pdf, n.d. [Accessed 08-08-2025].

[4] Johann-Sebastian Pleban, Ricardo Band, and Reiner Creutzburg. Hacking and securing the ar. drone 2.0 quadcopter: investigations for improving the security of a toy. In *Mobile Devices and Multimedia: Enabling Technologies, Algorithms, and Applications 2014*, Vol. 9030, pp. 168–179. SPIE, 2014.

[5] Cyrille Morin and Leonardo S. Cardoso. Hacking the dsmx drone rc protocol. In *GNU Radio Conference Proceedings*, Vol. 2. GNU Radio, 2021.

[6] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. Rocking drones with intentional sound noise on gyroscopic sensors. In *24th USENIX security symposium (USENIX Security 15)*, pp. 881–896, 2015.

[7] C. Zhou, Q. Yan, Y. Shi, and L. Sun. Doublestar: Long-range attack towards depth estimation based obstacle avoidance in autonomous systems. In *Proceedings of the 31st USENIX Security Symposium (USENIX Security 22)*, pp. 1885–1902, August 2022.

[8] A. J. Kerns, D. P. Shepard, G. Bhatia, and T. E. Humphreys. Unmanned aircraft capture and control via gps spoofing. *GPS World*, pp. 51–56, July 2014.

[9] Nourdine Aliane. A survey of open-source uav autopilots. *Electronics*, Vol. 13, No. 23, p. 4785, 2024.

[10] Dronecode Foundation. Who we are. https://dronecode.org/who-we-are/, n.d. [Accessed 08-08-2025].

[11] PX4 Dev Team. Px4 architectural overview. https://docs.px4.io/main/en/concept/architecture.html, n.d. [Accessed 08-08-2025].

[12] PX4 Dev Team. Mavlink messaging. https://docs.px4.io/main/en/mavlink/, n.d. [Accessed 08-08-2025].

[13] PX4 Dev Team. Simulation. https://docs.px4.io/main/en/simulation/, n.d. [Accessed 08-08-2025].

[14] ArduPilot Development Team. Welcome to the ardupilot development site. https://ardupilot.org/dev/index.html, n.d. [Accessed 2025-08-21].

[15] ArduPilot Development Team. Apmcopter code overview. https://ardupilot.org/dev/docs/apmcopter-code-overview.html#apmcopter-code-overview, n.d. [Accessed 2025-08-21].

[16] ArduPilot Development Team. Threading — ap_scheduler system. https://ardupilot.org/dev/docs/learning-ardupilot-threading.html, 2024. [Accessed 2025-08-21].

[17] ArduPilot Development Team. Sitl simulator (software in the loop). https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html, 2024. [Accessed 2025-08-21].

[18] PX4 Dev Team. Pixhawk. https://docs.px4.io/main/en/flight_controller/pixhawk.html, n.d. [Accessed 2025-08-21].

[19] Holybro. Pixhawk 6x autopilot flight controller. https://holybro.com/products/pixhawk-6x, n.d. [Accessed 2025-08-21].

[20] ArduPilot Development Team. Simulation. https://ardupilot.org/dev/docs/simulation-2.html, 2024. [Accessed 2025-08-21].

[21] PX4 Dev Team. Hardware-in-the-loop (hitl) simulation. https://docs.px4.io/main/en/simulation/hitl, n.d. [Accessed 08-08-2025].

[22] PX4 Dev Team. Controller diagrams. https://docs.px4.io/main/en/flight_stack/controller_diagrams, 2025. Accessed: 2025-08-08.

[23] 野波健蔵, 鈴木智, 王偉, 三輪昌史. ドローンのつくり方・飛ばし方 ―構造、原理から製作・カスタマイズまで―. オーム社, 2022. 出版日: 2022 年 8 月 3 日; ページ数: 228 p; 判型: A5.