CONFUSENSE: Sensor Reconfiguration Attacks for Stealthy UAV Manipulation

Alessandro Erba[†]
KASTEL Security Research Labs, Karlsruhe Institute of Technology

John H. Castellanos[‡]
Hitachi Energy Research, Germany

Sahil Sihag[*]
CISPA Helmholtz Center for Information Security

Saman Zonouz
Georgia Institute of Technology

Nils Ole Tippenhauer
CISPA Helmholtz Center for Information Security

Abstract

Unmanned Aerial Vehicles autonomously perform tasks using state-of-the-art control algorithms. These control algorithms rely on the freshness and correctness of sensor readings. Incorrect control leads to catastrophic process destabilization.

In this work, we propose the CONFUSENSE attack, aiming to impact process control via stealthy sensor reconfiguration. In the first part, the attacker will inject messages on buses that connect to the sensor. The injected message reconfigures the sensors. The reconfiguration primitives are selectively used to affect the controller (e.g., stall the control computations) transparently to the data consumer. Consequently, the manipulated sensor values lead to unwanted control actions (e.g., a drone crash). We experimentally demonstrate CONFUSENSE and investigate its system-level effects and consequences. Our findings show that i) reconfiguring sensors can have surprising effects on reported sensor values, and ii) the attacker can stall the overall Kalman Filter state estimation, leading to a halt in control computations. This leads to stealthily crashing or deviating the UAV over 30 meters.

Our work shows that attacks on sensors are not limited to continuously inducing random measurements, and demonstrates that sensor reconfiguration can completely stall the drone controller. In our experiments, state-of-the-art UAV controller software and countermeasures do not handle such manipulations. Hence, we discuss new countermeasures.

1 Introduction

In modern society, Robotic Vehicles (RVs) such as Unmanned Aerial Vehicles (UAVs) are deployed to accomplish a number of tasks. UAVs have been used to deliver medicines [48] and have been tested for life-saving deliveries [18]. UAVs

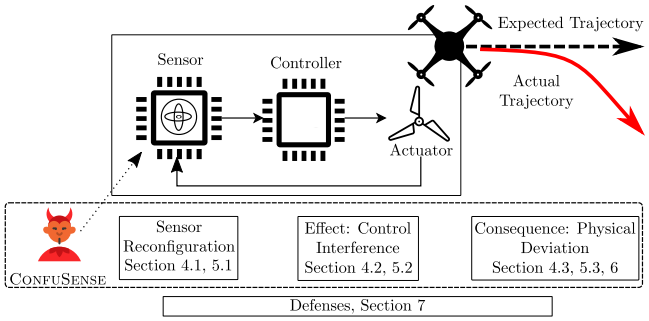


Figure 1: With CONFUSENSE, an adversary remotely reconfigures a flying drone's onboard sensor (e.g., via IEMI). The drone starts operating with the reconfigured sensor and makes wrong control decisions, which leads to deviation and crash. With a series of reconfigurations of the sensor, the attacker can even remotely control the drone.

are used for consumer goods delivery [50, 51] and are deployed in military operations [36], often relying on open-source flight controllers [46]. Given their importance, security concerns have been raised. Several recent papers confirmed these concerns by showing critical security issues in flight stacks [23, 24, 40, 43, 49]. Prior work in the Cyber-Physical Systems (CPS) domain proposed using GPS spoofing for remote UAV manipulation [40]. An attacker can take over the victim's UAV by precisely and continuously injecting the spoofed GPS signal. Alternatively, it was proposed to use intentional electromagnetic interference (IEMI) to inject noise in the MCU-sensor bus and continuously perturb the sensor readings received at the UAV controller [23]. By doing so, the attacker can crash the drone. Continuous spoofing of victim vehicle sensor readings will trigger anomaly detection as the readings will suddenly be noisy and inconsistent [17].

In UAVs, the Micro Controller Unit (MCU) continuously queries sensors for the latest readings. Sensors are configured to update their readings with fixed sampling rates. As sensors are in dedicated chips, they transmit the data to the MCU via the local bus system or analog signals.

[†]Part of this work was done while the author was with CISPA Helmholtz Center for Information Security and Saarbrücken Graduate School of Computer Science.

[‡]Part of this work was done while the author was with CISPA Helmholtz Center for Information Security.

[*]Also with Saarbrücken Graduate School of Computer Science.

In this work, we introduce the CONFUSENSE attack and investigate its impact on modern UAVs (see Figure 1). CONFUSENSE is a new attack that manipulates the sensor update rate (in the most extreme case, temporarily disabling a particular sensor) to influence controller decisions. We demonstrate that this reconfiguration can be achieved with a single malicious message, unlike sensor spoofing attacks [23], which require the continuous injection of malicious traffic. Notably, CONFUSENSE only requires intermittent access to the bus and remains effective even after the attacker is out of range.

We formalize the CONFUSENSE attack, investigate realizations of these attacks on Commercial-Off-The-Shelf (COTS) flight controllers, and evaluate their system security implications at hardware, Real-Time Operating System (RTOS), control, and anomaly detection. Finally, we propose a new attack synthesis algorithm enabling malicious drone control.

Our results show that CONFUSENSE induces severe consequences on UAV stability, quickly leading the drone to deviate from the planned trajectory or crash. Moreover, our novel attack remains stealthy from state-of-the-art Anomaly Detector [12, 26]. Our takeaway is that although sensors are fundamental for enabling autonomy in current UAV architectures, their insecure configuration poses a significant trust issue, affecting the system's security and functionality. Finally, we discuss potential countermeasures.

The contributions of the paper are:

- We propose a novel approach in which the attacker indirectly manipulates sensor readings via stealthy reconfiguration of the sensor via a shared bus. We show that this reconfiguration can lead to unexpected results on the reported readings and is transparent to the controller.
- We experimentally implement CONFUSENSE attack, evaluate its effect and consequences in different environments, demonstrating its practicality. We show the impact of reconfigurations on five different hardware platforms and two different control software. We demonstrate that our manipulations can stall the control loop.
- We propose an attack synthesis methodology to optimize CONFUSENSE for deterministic manipulation of drone behavior. This is the first time that drone manipulations do not only lead to uncontrolled crashes but control of the drone without active sensor spoofing.

Our artifact is available at https://github.com/scy-phy/droneDeprivation.

2 UAV Security: Background and Motivation

2.1 Architecture of UAVs

Autonomous UAVs have sensors and actuators connected to a flight controller (Figure 2). Sensors observe the UAV state and report it to the flight controller. The flight controller has a microcontroller unit (MCU), which is used to run the control and communication software. The received sensor readings are then used for state estimation via sensor fusion algorithms (e.g., Kalman Filter). Based on the estimated state and the planned trajectory, the flight controller computes the control action, and transmits it to the actuators (motors).

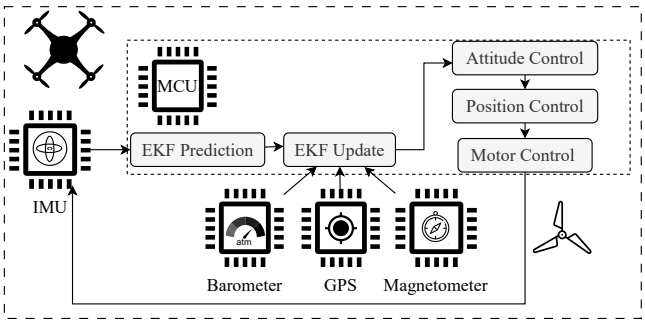


Figure 2: Flight control—sensors, GPS receivers and their contribution to the state estimation.

Sensors for autonomous navigation. Inertial Measurement Unit (IMU) is an essential sensor for flight stabilization. It uses an accelerometer and a gyroscope to report the proper acceleration and angular velocity to the flight controller. UAVs utilize this information for attitude estimation (estimating the pose of the drone). Other sensors required for autonomous navigation are the compass, the magnetometer, and the barometer. Moreover, a GPS receiver is necessary to locate the drone. Readings from all these sensors and the GPS receiver are input to the Kalman Filter for state estimation.

EKF architecture. Extended Kalman Filter (EKF) is the non-linear extension of the Kalman Filter [25]. It operates in two steps: prediction and update. During the prediction step, the state estimate is computed (e.g., roll, pitch, yaw), and during the update, the estimate is improved by sensor fusion.

Popular open-source flight control software such as Ardupilot applies state-of-the-art Extended Kalman Filter state estimation for autonomous navigation. According to the documentation [5], state prediction relies on IMU data, while the state update is done by fusing the other available sensor readings. State prediction is calculated every time new IMU data is received (IMU has a high sampling rate), while state update is performed less frequently (as the other sensors are sampled less often). The IMU sensor reading is the most important and has the highest priority in the state estimation procedure.

2.2 Serial Buses for Sensor Communication

In UAV systems, sensors connect to the microcontroller via unauthenticated, unencrypted serial buses like SPI, I2C, CAN, and UART. These buses transmit sensor data and allow the microcontroller to configure sensors using serial APIs detailed in publicly available datasheets. The microcontroller configures sensor registers to meet real-time application needs.

We focus on I2C, a common two-wire protocol with a clock and data line. The microcontroller acts as the controller, coordinating communication. The controller mandates communication via the clock signal. All other bus members listen to the data line and respond to the specific controllers' requests.

2.3 Attacks on UAVs

Nassi et al. [33] systematized UAV attacks and defenses. Son et al. [49] showed sound injection attacks on gyroscopes, while Jeong et al. [24] proposed countermeasures. Sathaye et al. [40] studied GPS spoofing, showing drone takeover is challenging in practice. Dayanıklı et al. [16] demonstrated false PWM actuation via IEMI, and Jang et al. [23] disrupted IMU readings by blocking serial channels electromagnetically. Selvaraj et al. [45] manipulated analog signals via IEMI. Dayanıklı et al. [15] and Zhang et al. [55] showed IEMI-induced bit flips in UART, I2C, and CAN. Xie et al. [53] demonstrated UART attacks and proposed a defense.

2.4 Anomaly Detection for UAVs

Open-source autopilots like Ardupilot implement an EKF failsafe [4], which triggers when EKF residuals exceed a threshold for over 1 second, switching the drone to land mode. In security research, Quinonez et al. [37] used an additional EKF and CUSUM for anomaly detection. Choi et al. [12] proposed using control invariants derived via system identification. More recently, Khan et al. [26] proposed to detect UAV attacks by monitoring the microcontroller's low-level peripheral bus registers to identify any abnormal access patterns, and by moving the Kalman filter outside the RTOS.

3 System Model and Assumptions

System Model We consider a UAV running state-of-the-art autopilot software (see Figure 2 and Section 2.1). A monitoring system like M2MON [26] is in place to detect attacks on the UAV by monitoring the Kalman filter and serial register. In its most basic form, the monitor will check for unexpected large differences in sensor readings (e.g., bad data detection [9]). If an attack is detected, the drone will switch to the failsafe mode or execute recovery [14].

Threat Model An attacker is assumed to access a copy of the victim's hardware (e.g., by buying the same UAV drone or flight controller). The attacker's goal is to deviate the victim drone from its trajectory or crash it without being detected.

We assume the attacker can interact with local bus communication (e.g., eavesdropping, manipulating, or injecting their own messages). Realizations of such attacks have been demonstrated, such as: i) a supply chain attack, implanting malicious behavior into one component connected to the bus [7,54], ii) by remotely injecting or manipulating messages on the bus via IEMI [15, 16]. We use the IEMI channel in our later experiments for demonstration purposes but consider in-depth discussion on this as out of scope (see Section 3.1).

Our threat model does not allow the attacker to execute (or manipulate) code on the main MCU, no vulnerabilities in the MCU would allow remote code execution (e.g., via the bus).

Table 1: Compared to prior work attacks against UAV sensors, our attack does not require continuous signal injection, active sensor value spoofing, or sensor corruption. Instead, our attacker requires a single reconfiguration message injection, which persists even when an attacker is out of range.

Spoofing Type	Continuous injection	Active spoofing	Sensor corruption	Single injection	Persistent Out of Range
GPS, SEC'22 [40]	●	●	○	○	○
IEMI, NDSS'23 [23]	●	○	●	○	○
Acoustic, NDSS'15 [49]	●	○	●	○	○
Acoustic, NDSS'23 [24]	●	○	●	○	○
ConfuSense	○	○	○	●	●

Research Goal and Challenges The idea of the attack is twofold: i) the attacker reconfigures sensors altering its operational behavior (e.g., sensor operating frequency), and ii) using this capability, the attacker forces the victim controller to perform wrong control decisions, leading the drone on the wrong path or crashing. We introduce the term ConfuSense attack to capture the intuition of our novel attack.

There are two main challenges: **C1)** It is not clear from prior work which manipulations are possible for the proposed attacker model without raising errors at the controller and **C2)** Depending on the attacker impact on the sensors, deterministically controlling the drone behavior is challenging.

3.1 Gaps in Prior Research

Unlike classical denial of service attacks, our contribution lies in identifying an under-explored class of cyber-physical attacks where the adversary deliberately and persistently denies timely sensor updates. While control theory has addressed the impact of stochastic deadline misses due to unreliable communication channels [31,41,42], these timing issues have not been studied from a security viewpoint.

Existing UAV architectures remain vulnerable to remote sensor manipulation attacks. While process-based anomaly detection systems [12, 26, 37] can identify and recover from conventional spoofing attempts [14], our proposed attack introduces a novel threat that bypasses such defenses. Unlike prior sensor spoofing or false data injection attacks that continuously tamper with sensor values [23,24,40,49], our approach reconfigures the sensor permanently, allowing the attack to persist even after the adversary is no longer in range. As illustrated in Table 1, this enables control disruption through legitimate traffic without compromising communication.

Our work investigates the security implications of such an attack across both hardware and software layers of UAV

systems. While previous studies have demonstrated that intentional electromagnetic interference (IEMI) can destabilize UAVs [15, 16, 23, 53, 55], we explore how IEMI can be used to induce sensor reconfigurations. However, we do not claim novelty in the use of IEMI itself, as it is already recognized in the literature as a viable attack vector.

4 ConfuSense Attack

4.1 ConfuSense: Sensor Reconfiguration

Our attack approach is to deny timely sensor updates in order to crash the target vehicle (or manipulate the path). To achieve this, we leverage two variants of sensor reconfiguration: a) sensor suspension, and b) sampling frequency reduction.

a) Sensors allow suspension (or power-save mode, depending on the manufacturer) for power-saving purposes. In this mode, the sensor stops updating the sensor reading register. The control software will continue its normal execution, but the information received from the external sensor will be no longer reliable (e.g., outdated, wrong, or absent).

b) Sensors allow modifications of the operating frequency. Modern flight controllers (e.g., the Pixhawk 6c [21]) rely on general-purpose IMUs that can be programmed for different applications (i.e., the same IMU chip can be used in various contexts, for example, in smartphones and activity trackers). In flight control software, sensors are configured to run at the highest sampling frequency allowed by the sensor to enable precise control of the aircraft. An attacker can delay sensor updates and compromise control accuracy by reducing the sampling frequency. This can be modeled as intermittent suspension with a duration inversely proportional to the frequency.

Sensor Reconfiguration Commercially available sensors used in UAVs are configured by issuing (not cryptographically secured) bus write commands into specific register addresses (serial API, Section 2.2). All the commercially available sensors that we reviewed allow configuration via serial APIs, and to the best of our knowledge, configuration commands for a sensor cannot be flushed to avoid configuration overriding. An attacker altering the sensor configuration influences the drone's state estimation, leading to incorrect actuation. We demonstrate that modern controllers implicitly trust sensors.

During the normal operations of a drone, the sensor is periodically queried from the microcontroller to pull the fresh data (see Section 2.2). The attacker can leverage such periodic messages or create new traffic on the bus to achieve target sensor reconfiguration. In the evaluation (Section 5.1), we provide two case studies showing how the attack can be practically launched on a victim's vehicle.

4.2 ConfuSense Effect: Control Interference

Possible Behaviors Consider the system in Figure 3. Based on sensor readings, the controller computes the actuator signal to control the physical process. Intuitively, the proposed ConfuSense involves attacker actions to 'interrupt' the feedback from the physical process to the controller (the 'switch' symbol in Figure 3 represents the attacker's action). During an attack, an attacked set of observations reaches the controller.

In control theory, prior work proposed control strategies for lossy control packets [41, 42, 47], by stochastically modeling missing observations. In contrast, we assume the attacker deterministically triggers the sensor reconfiguration to destabilize the process. We identify three possible behaviors resulting from a ConfuSense-induced sensor reconfiguration and their resulting missing observations. In Section 5.2, we verify which of the hypothesized behaviors occur in practice in COTS IMU sensors and their impact on the control.

Absent data. No sensor values are received at the controller. This situation occurs when the sensor is disabled.

Default value data. The value that reaches the controller is a constant. This situation occurs when the sensor is in an error state and replies to queries with a default value.

Stale data. The sensor stops updating, and the last observation is retransmitted to the controller. Krotofil et al. introduced this setup in an ICS scenario [27].

Due to the influence of ConfuSense on the sensor readings, the controller reacts depending on the received observations. In the case of *absent data*, two control strategies were proposed by Schenato et al. [41]. *Zero-input:* Stop actuating the system. *Hold-input:* Keep actuating based on the last observed value (or keep the last control action). In the case of *default data* and *stale data*, the controller will continue to actuate the system based on the received data. In a nutshell, ConfuSense affects the Kalman filter state estimation, propagating to the controller to produce the wrong actions.

Attack Formalization Consider a linear discrete-time system, Equation 1.

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \tag{1}$$

where A, B, and C are matrices that describe the coefficient physical model. x_k denotes the current system's state, y_k are the sensor readings, and u_k denotes the actuation command computed based on the current sensor readings.

In the case of ConfuSense, the observations that reach the controller are affected by the attacker (y_k^a), which can decide to start the attack on the system ($\gamma_k \in \{0, 1\}$).

In the case of absent data, no sensor data is received at the controller during an attack. The sensor observation model is:

$$y_k^a = (1 - \gamma_k) y_k + \gamma_k \varnothing \tag{2}$$

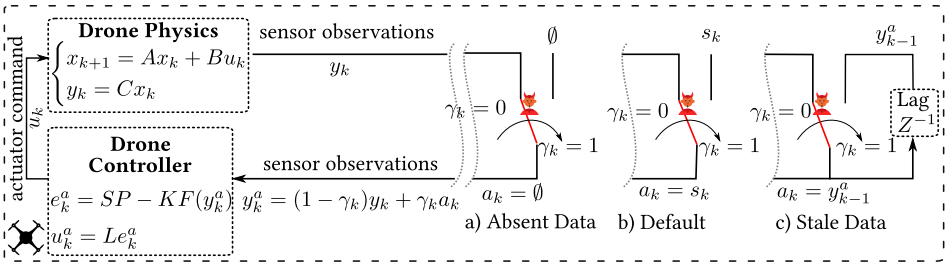


Figure 3: CONFUSENSE attack abstraction. An attacker reconfigures sensors to influence the readings. This can have different effects in the control loop, a) Absent data, where no data is received b) Default data where s_k is constant c) Stale data where last observation is received by the controller.

In the case of default value data, the observation model is:

$$y_k^a = (1-\gamma_k)y_k + \gamma_k s_k \tag{3}$$

where s_k is constant, while for . Finally, in the case of stale data, the observation model is:

$$y_k^a = (1-\gamma_k)y_k + \gamma_k y_{k-1}^a \tag{4}$$

where y_{k-1}^a is the last observation before the attack.

Before reaching the controller (e.g., PID controller), y_k^a goes through the Attitude and Heading Reference System (AHRS) and Kalman filter (where it gets fused with the other sensor sources to refine the system state and the sensor readings estimate, e.g., yaw, pitch, and roll) (see Figure 2). The attack-induced sensor reading propagates through the filters to reach the controller, where the wrong state estimate influences the error value e_k^a (between the observation and the control target). Formally,

$$\begin{aligned} & u_k^a = Le_k^a \\ \text{where} \quad & e_k^a = SP - KF(y_k^a) \\ & y_k^a \in \{y_k, \text{absent, default, stale}\} \end{aligned} \tag{5}$$

L are the controller parameters that operate based on the error, SP is the control target (set point), and KF is the Kalman filter.

4.3 CONFUSENSE Consequence: Physical Deviation

CONFUSENSE induces the state estimation to be incorrect or delayed. This causes the system to have the wrong actuation. In real-world control systems, the effect of measurement and environmental noises makes the PID controllers perform an error correction w.r.t. a target value (as explained in Equation 5). We are interested in understanding if this wrong actuation can be adversarially exploited to deviate the drone to an adversary's desired location.

Figure 4 shows an example: while a drone is following a specific mission (e.g., moving along waypoints), the attacker is launching targeted attacks to suppress specific (or all) new

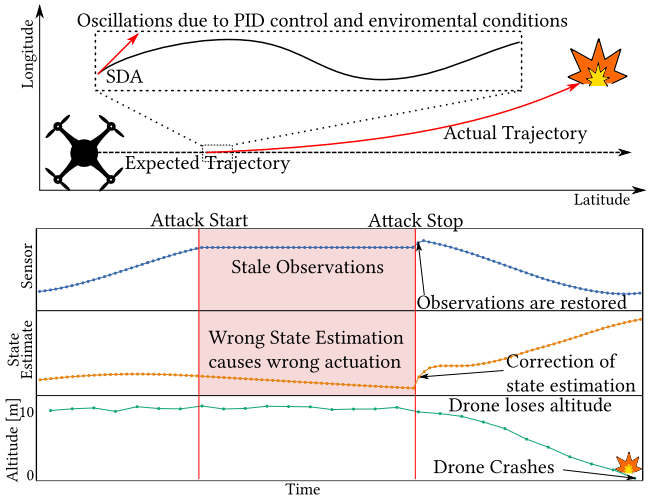


Figure 4: Motivating example, a drone flies on a straight trajectory. The flight is influenced by sensor and environmental noise which results in continuous correction applied to the flight. An attacker starts CONFUSENSE, targeting one of the drone sensors. The sensor stops updating the transmitted value to the MCU. Consequently, the attitude estimation is compromised, causing wrong actuation commands that result in the drone's deviation or crash.

sensor readings (i.e., through sensor reconfiguration). When the CONFUSENSE attack starts, the last computed attitude is the last correct estimate of the drone's pose. The actuation command will reflect the correction w.r.t. the expected trajectory of the drone. Such attacks cause the drone controller to operate on outdated sensor readings with constant actuation – which causes the drone to go on the wrong trajectory. When the attack stops, the drone will compute the correct state estimation and attempt to re-stabilize the body frame. In Figure 4, the drone crashes on the ground while recovering from the CONFUSENSE attack.

In contrast with a sensor spoofing attack, where the attacker can arbitrarily substitute the sensor readings to induce the desired behavior, with CONFUSENSE the attacker does not

actively decide the value received at the controller, instead it would follow one of the three behaviors described in the previous section (absent, default, stale data).

To deviate the victim vehicle towards the adversary's desired position, the attacker intermittently triggers CONFUSENSE to influence the sequence of actuations (u_k) to minimize the distance between the drone location and adversary's desired position (*adversarial control* problem).

$$\begin{aligned} \text{minimize} \quad & \sum_{k=0}^{t} \sqrt{(\text{DronePos}_k - \text{AttackerGoalPos}_k)^2} \\ \text{where} \quad & \text{DronePos}_{k+1} = \sum_{k=0}^{t} A(\text{DronePos})_k + Bu_k^a \end{aligned} \tag{6}$$

In Section 6, we propose an attack synthesis methodology to solve the adversarial control problem.

5 Experimental Evaluation

We comprehensively study CONFUSENSE attack. The first part involves the realization of CONFUSENSE in hardware, followed by the effects on the UAV controller, and finally, a general perspective of CONFUSENSE controllability by studying the consequences of such attacks on UAVs. Specifically, we answer the following research questions. **RQ1**: *How can* CONFUSENSE *be practically launched on modern flight controllers?* **RQ2**: *Which practical interactions between an IMU and the control program are observed and what is their relation to the hypothesized behaviors for* CONFUSENSE*?* **RQ3**: *What is the impact of the proposed* CONFUSENSE *attack on UAVs?*

5.1 Case Study: CONFUSENSE in Hardware

To study CONFUSENSE attack realizations in hardware, we designed a set of experiments that help us to answer **RQ1**. In this section, we demonstrate how an attacker can practically launch the proposed CONFUSENSE on hardware. Consistent with prior work [23], we set up the same testbed with a main controller (Raspberry Pi3) that connects to an IMU (MPU6050) through a shared bus (I2C). A Python script using the `smbus2` library controls the I2C bus communication.

We offer two case studies via local and remote attacks to showcase CONFUSENSE for sensor suspension. For space constraints, we demonstrate the sensor frequency reconfiguration and its effects in Appendix A.

Local sensor reconfiguration *Attack Setup.* The attacker relies on a peripheral connected to the I2C bus. The main controller continuously reads data from the IMU. I2C uses pull-up resistors in both lines to ensure a logical '1', which means only '1' can be turned into '0' and not vice versa – making message manipulation challenging, as the attacker

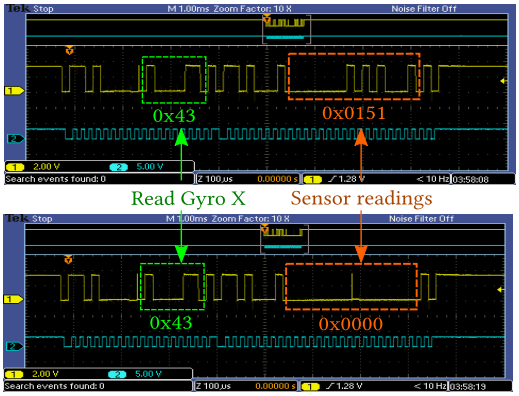


Figure 5: Reading Gyro X on the shared bus. Top: Readings before the malicious command injection. Bottom: Readings after the malicious command injection, no data is transmitted due to the device reconfiguration.

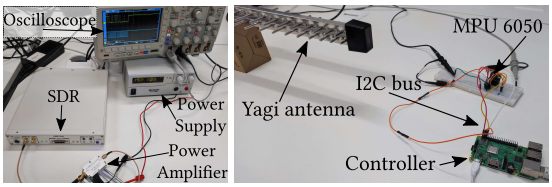


Figure 6: IEMI injection testbed. On the left: the SDR, the power amplifier (connected to the power supply), and the oscilloscope. On the right, the Yagi antenna, the Raspberry Pi connected to the IMU.

lacks full control of the data line. We found that the clock signal is generated only when the main controller sends a request. This presents an opportunity for an attacker to inject commands. When the bus is idle, any device could generate the clock signal and maliciously impersonate the legitimate controller. To avoid collisions, the attacker listens to reading patterns on the bus and determines when to inject bus commands. The attacker-controlled peripheral injects commands by reconfiguring its I2C interface from peripheral to controller mode.

Injection result. The attacker has a time window before the next communication with the legitimate controller occurs (legitimate communications occur at 400Hz in Ardupilot). Within this time window, our attacker sends requests to the target peripheral on the bus to change its configuration. After the injection, the sensor stops answering the controller's requests. For example, before the attack, the read gyro X command `0x43` is followed by a sensor reading response `0x0151`, and after the attack, the sensor responds `0x00`. Figure 5 shows the impact of the CONFUSENSE on the sensor.

Remote sensor reconfiguration *Attack Setup.* The attacker injects wireless electromagnetic emission on the victim bus. Prior work has shown that random values can be injected in

the shared bus via IEMI [23]. We now demonstrate that an attacker can use IEMI for targeted sensor reconfiguration.

For IEMI injection, we use a software-defined radio (SDR, USRP X310) connected to a power amplifier (WYDZ-PA-1M-750MHz) that provides power to a Yagi antenna (see Figure 6). We monitor the IEMI effects on an oscilloscope (Tektronix MSO2024B) connected to the data and clock lines of the I2C bus. The I2C controller (Raspberry Pi3) controls the sensor's operation by changing the values on particular registers. To reconfigure a sensor, an attacker needs to inject a valid writing command in the shared bus. A valid I2C writing command is composed of a device address, followed by a register address and a payload (see Figure 7). To induce the desired reconfiguration, the attacker needs to bit-flip parts of the register address or payload in the write command.

To perform a successful IEMI injection, we must find the correct frequency that induces the power required to cause a bit flip. After performing a frequency sweep, we found that the range between 120MHz and 150MHz injects enough power to disrupt the bus communication. We used 126MHz as the carrier frequency which showed the highest injection power. Prior work [23] investigated and characterized the relationship between power and distance (Figure 18 in [23]). Their result shows that by using a higher power amplification, the attacker can induce IEMI at a higher distance (Jang et al. [23] results are offered up to 8 meters. As explained in Section 3.1, our paper's contribution is not the use of IEMI. Instead, we rely on IEMI to demonstrate our attack.) Following the base case from prior work [23], we test the injection at 10 cm. The maximum output power of our power amplifier is 34.8 dBm (3W).

Disrupting bus communication triggers errors such as collisions, EKF failsafe, or bad-data detection (e.g., Control Invariant monitors [12]), making continuous IEMI injection unsuitable due to its lack of stealth. Instead, the attacker must inject IEMI in a timely, controlled manner to achieve sensor reconfiguration. Prior work [15, 16] discusses synchronization with the victim bus and demonstrates both synchronous and asynchronous IEMI injections via IEMI leakage. As a proof of concept, we demonstrate on-the-fly modification of payload values. By exploiting the periodicity of bus communication, the attacker can time the IEMI duration to selectively override specific segments, such as IMU register values, while passively eavesdropping to synchronize injections.

Injection result. Our goal is to achieve the proposed sensor suspension and sampling frequency reconfiguration via IEMI. According to the sensor specification [22], to achieve the sensor suspension in the MPU6050, we have to write in the `Power Management 1` register `0x6B`, the value `0x40`. Figure 7, represents the collected power trace at the oscilloscope, with and without injection. The collected power traces contain a register write followed by a register read. As we can observe when the IEMI injection is performed (lasting 800μs, i.e., 5 clock cycles), the response from the register changes

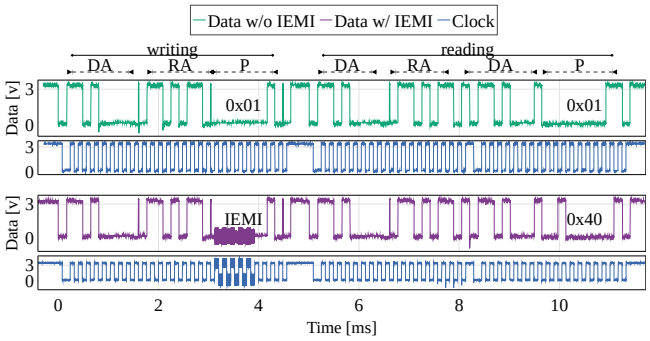


Figure 7: I2C bus oscilloscope capture. Comparison of the stored register value before and after triggering the suspend mode in the sensor via IEMI attack on MPU6050. The IEMI modifies the payload of the message from 0x01 to 0x40. DA: Device address, RA: Register address, P: Payload.

reporting `0x40` (our target value for suspend mode).

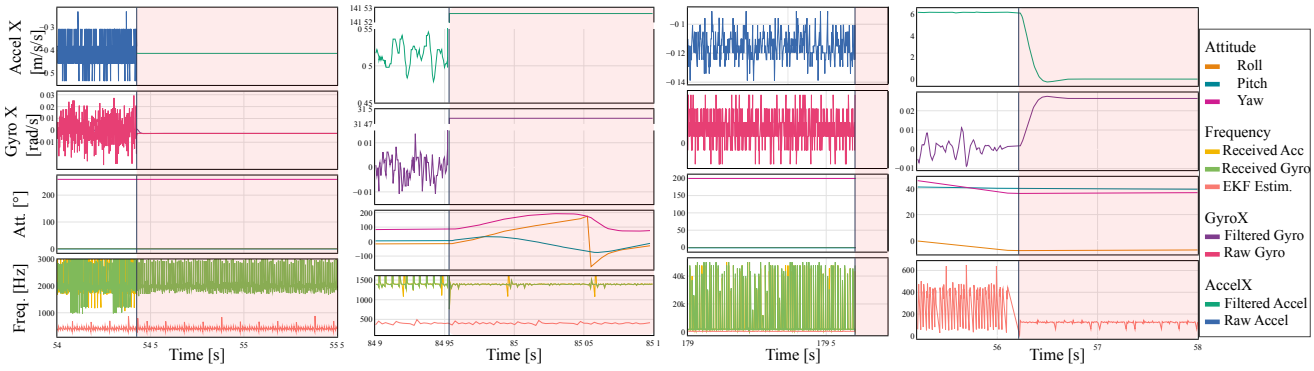
Generalizability to other protocols and CPS As described in Section 2.2, the serial protocols exploited for the attack are general and used in several CPS domains; for this reason, the proposed sensor reconfiguration method can be used to attack any CPS architecture that relies on similar sensors (e.g., autonomous vehicles). Moreover, the proposed approach is general; the same attack applies to other serial protocols (e.g., UART). Independent from the bus protocol, two factors make the CONFUSENSE possible: i) the sensor's reconfigurability, and ii) the use of unauthenticated buses.

To answer **RQ1**, we practically realize and demonstrate the reconfiguration command injection on an I2C bus testbed. Our case studies show that it is possible to reconfigure the sensor (locally or remotely) maliciously. The proposed reconfiguration approach brings advantages compared to spoofing of the sensor readings that could lead to detectable bus collisions.

5.2 Control Interference: CONFUSENSE Effects in Hardware and Software

Motivated by the practicality of reconfiguration attacks, we now address **RQ2** by investigating the effect on IMUs of such reconfigurations and the consequences on flight control.

To verify the behavior of CONFUSENSE over the control algorithm, we run experiments on multiple COTS flight controllers and evaluate how different IMUs respond to the reconfiguration. We also evaluate how the control software reacts to those reconfigurations. We rely on the Ardupilot, a widely-used flight control software for commercial UAVs. For our evaluation of system-level effects of CONFUSENSE, using either PX4 or Ardupilot yields similar results as both rely on the same Extended Kalman Filter implementation [38].



(a) BMI055 (b) BMI270 (c) ICM42688 (d) MPU6000

Figure 8: Suspend attack behavior on the four tested IMUs. Red background: IMU suspended.

Evaluation Setup. We test ConfuSense with three flight controllers running Ardupilot and launch ConfuSense during the flight control software execution. We then analyze the collected log files to assess the impact of sensor reconfiguration. The flight controllers used in our evaluation are i) Kakute H7 Mini (BMI270 IMU); ii) Pixhawk 6C with a dual-IMU (BMI055 IMU and ICM-42688-P IMU); iii) Omnibus F4 (MPU6000 IMU). Moreover, we assess the end-to-end impact of ConfuSense on the Crazyflie V2, equipped with BMI088 IMU and the optical flow sensor.

Effects of IMU Suspend For each IMU, we identify the register/value pair used for device reconfiguration and verify its effect on the control software. We now detail our findings.

BMI055. IMU is composed of two sensor modules: the accelerometer and the gyroscope. The accelerometer reacts by stopping sending values to the MCU (absent data), while the gyroscope reacts to the configuration change by reporting a constant value (default value data). Table 2, reports the default value that was sensed on each axis during the attack, while Figure 8a shows the results of the suspend reconfiguration on the BMI055 sensor. The Ardupilot controller reacts to the (partially) absent data by applying the hold strategy on the last observed value, and the attitude estimation continues despite no accelerometer samples being received. As we can notice from the bottom plot in Figure 8a, the frequency of the EKF estimation is unaffected by the reconfiguration attack.

BMI270. In this case, both the accelerometer and gyroscope react to the reconfiguration with the default value data, and the reported value for each module is an out-of-range value (see Table 2). Figure 8b, shows the sensed behavior when starting the suspend attack on the BMI270. Ardupilot accepts this received value as legitimate, which induces an error in attitude estimation. The attitude estimator believes that the drone is flipping (instead, the drone is steady on a table). As in this previous case, the reconfiguration does not affect the sensor and EKF frequency, leaving the scheduler unaltered.

ICM-42688-P. In this case, the IMU reacts to the suspend attack by stopping transmission of both the gyroscope and the accelerometer sensor updates. Since no data is received at the controller (for both sensors), no state estimation is performed (see Figure 8c). This result shows the resource dependency between the IMU and the state estimation. In Ardupilot, the state estimation is part of the `fast_loop()` tasks, executed with the highest priority by the scheduler with a frequency of 400Hz. The control flow inside the flight controller remains unaltered, but IMU data unavailability makes the highest priority task in Ardupilot not executed.

MPU6000. As reported in Table 2 the sensor behaves similarly to the BMI270 and starts reporting default values; those values are close to zero. In this case, after the reconfiguration, the sensor increases its responsiveness time (the attitude and EKF frequency drops from 400Hz to 100Hz, see Figure 8d),

Effects of IMU Sampling Frequency Reducing the IMU sampling frequency below 400Hz leads to intermittent stale data, causing significant delays in the Kalman Filter updates. This slowdown in sensor data processing directly impacts the RTOS task scheduling, causing tasks to be delayed or missed. As a result, the drone's ability to compute fresh control commands is impaired, compromising its stability. A full discussion of these results and implications is available in Appendix A.2.

Our ConfuSense uncovers a critical controller design flaw, where an untrusted sensor can induce delays in task execution within the MCU. This flaw exposes a weakness in the firmware's handling of sensor data and task scheduling.

End-to-end Attack Demonstration We evaluate the impact of the reconfiguration attack strategies on the Crazyflie V2 platform equipped with the control firmware `2023.02` and BMI088 IMU sensor. An attack demonstration video is available at https://youtu.be/7vgxqLVDyUA. The Crazyflie device is hovering above the ground and will continue doing so (baseline in the video) unless an attack is started. In BMI088, the suspension produces absent data behavior (similar to BMI055). The impact of the suspended attack results in

Table 2: Summary of IMU behaviors observed during Suspend mode attack

	Accelerometer Value							Gyroscope Value						
		X		Y		Z			X		Y		Z	
IMU Model	Behavior	μ	σ	μ	σ	μ	σ	Behavior	μ	σ	μ	σ	μ	σ
BMI055	Absent Data	-	-	-	-	-	-	Default	-0.0025	0.0	-0.002	0.0	0.0012	0.0
BMI270	Default	141.53	0.0	141.53	0.0	141.53	0.0	Default	31.48	0.0	31.49	0.0	31.49	0.0
ICM-42688-P	Absent Data	-	-	-	-	-	-	Absent Data	-	-	-	-	-	-
MPU6000	Default	0.2477	1.058	68.9905	14.563	-0.3031	1.295	Default	0.0254	0.0	-0.0346	0	0.1211	0

the drone crashing on the ground. Since the sensor updates are suspended, this attack is undetected by a bad data detector (see Section 7.1).

To answer **RQ2**, we characterize the behavior of IMUs when targeted with the proposed reconfiguration and their implications on the control algorithm (data unavailability blocks execution of RTOS tasks). Our results show that the proposed techniques will result in three main behaviors: absent, stale, and default data (consistently with the behaviors hypothesized in Section 4.2), and expose fundamental weaknesses in the firmware's handling of sensor data and task scheduling.

5.3 Physical Deviation: ConfuSense Consequence

To address **RQ3**, we investigate the impact of ConfuSense on UAVs. For our evaluation, we assess the effect of ConfuSense on the Ardupilot [2] autopilot system (version 4.3.3), running in the Software In The Loop (SITL) simulator. SITL is a fully functional environment to run Ardupilot software in simulated physical surroundings. All the Ardupilot features are available within the SITL framework; for this reason, it is well suited to test the impact of ConfuSense on the control algorithms without safety hazards. To emulate the effects of the ConfuSense, we target the virtual sensor driver for the IMU (`AP_inertial_sensor_SITL.cpp`, available in our repository). In turn, we emulate the stale data by storing the last observation and re-transmitting it to the attitude controller, the default data by transmitting the default values observed for the MPU6000 and BMI270 (see Table 2), and the absent data by skipping the data transmission during the attack time (in practice we emulate it as a frequency reduction, as otherwise the physical simulation would not advance). To evaluate the attack's impact, we create a straight-line mission at a desired altitude of 50 meters, flying it 11 times without attacks to establish a baseline trajectory, and 11 times for each observed ConfuSense-induced behavior, lasting 1 second. The ConfuSense is launched when the drone reaches the desired altitude and follows the straight trajectory. We evaluate the impact of the attacks in terms of deviation D (Equation 7) as the instantaneous mean squared difference between the attack trajectory A and the reference trajectory R. The distance is computed over the three-dimensional space axis, x (longitude displacement), y (latitude displacement), and z (altitude displacement).

Table 3: Emulation of ConfuSense attack with Ardupilot SITL. Behavior characterization of Ardupilot flight control after launching ConfuSense for 1 second. Each attack is repeated 11 times.

	ConfuSense behavior							
	Undetected		Detected (failsafe)		TtD		TtC	
	%crash	%compl.	%crash	%land	μ(s)	σ	μ(s)	σ
Stale	0.00	0.00	0.00	100.00	4.99	1.80	N.A.	N.A.
Default	0.00	0.00	100.00	0.00	1.22	0.00	4.56	0.00
Absent	100.00	0.00	0.00	0.00	N.A.	N.A.	4.22	0.01

$$D_{t(x,y,z)} = \sqrt{(A_{t(x,y,z)} - R_{t(x,y,z)})^2} \tag{7}$$

In particular, we consider Maximum deviations Equation 8 induced by the attack (i.e., between the events *attack start* and the *crash*, *failsafe* or *landing* event timestamps).

$$\max \quad D_{t(x,y,z)} \quad t \in \{\text{attack start, failsafe/crash}\} \tag{8}$$

Moreover, we are interested in evaluating the impact of the attacks on the Ardupilot failsafe mechanism (see Section 2.4). For this reason, we measure how often the attack triggers the EKF failsafe, how often the drone crashes, how often the drone completes the mission, and how often it performs an emergency landing. For the attacks that trigger the EKF failsafe or crash the drone, we compute the average Time to Detect (TtD) and Time to Crash (TtC). We note that with the Ardupilot default EKF failsafe, the minimum Time to Detect is 1 second (see Section 2.4).

$$TtD = T_{\text{failsafe}} - T_{\text{attStart}} \tag{9}$$

$$TtC = T_{\text{crash}} - T_{\text{attStart}} \tag{10}$$

Emulation Results Table 3 and Table 4 report the summary of the emulation carried out over the possible ConfuSense induce behaviors. We measure the deviation induced by the ConfuSense before the EKF failsafe triggers.

Table 4: Emulation of CONFUSENSE with Ardupilot SITL. Maximum deviation distribution in meters until failsafe or crash. Attack duration is 1 second.

	x		y		z	
	μ(m)	σ	μ(m)	σ	μ(m)	σ
Stale	16.7	15.8	7.6	7.9	1.2	1.2
Default	4.3	0.2	39.7	0.0	4.9	0.0
Absent	57.8	0.4	138.1	0.0	53.1	0.0

Stale Data. In the case of stale data attacks, the attack triggers the EKF failsafe 100% of times, which successfully lands the drone 100% of times. The EKF failsafe requires an average TtD of 5 seconds, which is a significant amount of time (recalling that the minimum time to detect is 1 second). The attack deviates the drone from its trajectory, 1 to 16 meters on average. The standard deviation reveals that attack effects vary with the stale data value. Due to sensor noise, each emulation run differs, leading to non-deterministic behavior.

Default Value Data. Table 4 reports the simulated deviation observed when using the MPU6000 default values. In this scenario, the CONFUSENSE-induced behavior triggers the EKF failsafe. The drone crashes as the controller is unable to stabilize the vehicle post-attack. This behavior is consistent across repeated experiments, evidenced by a low standard deviation. This consistency can be attributed to the transmitted data remaining identical across runs. The time to detection (TtD) is also low, with the EKF failsafe activating after 1.22 seconds, compared to a minimum detection threshold of 1 s.

When using the BMI270 default data, the simulation logger stops recording the drone's status, making it unclear whether the drone activated the failsafe, crashed, or landed. Nonetheless, analysis of the altitude logs indicates significant deviation: 10.63m (σ = 0.73) along the x-axis, 22.02m (σ = 0.01) along the y-axis, and 51.66m (σ = 0.00) along the z-axis—strongly suggesting the drone reached the ground and crashed. The time to crash (TtC = 0.74s) is below the 1-second threshold required to trigger the EKF failsafe. This result is excluded from Table 4 as the logger fails to capture critical post-attack data, making it unsuitable for comparison.

Absent Data. In this case, the controller stops receiving sensor updates. In our emulations, the drone crashes 100% of the time without triggering the EKF failsafe. The induced behavior is consistent among the runs (i.e., a low standard deviation of the induced maximum deviation). Notably, the attack will induce a deviation of 138 meters on the y axis and above 50 meters on the x and z axis.

Regarding **RQ3**, the emulation results of CONFUSENSE demonstrate a severe impact on the physical system, consistently causing the drone to deviate from its intended path and crash. Our analysis shows that stale data attacks are detected by the EKF failsafe, forcing the drone to land. In contrast, attacks involving default value or absent data always result in a crash: either the failsafe triggers too late to prevent it, or the attack goes entirely undetected. These findings highlight how different forms of sensor data manipulation undermine the controller's ability to respond and recover, ultimately causing mission failure.

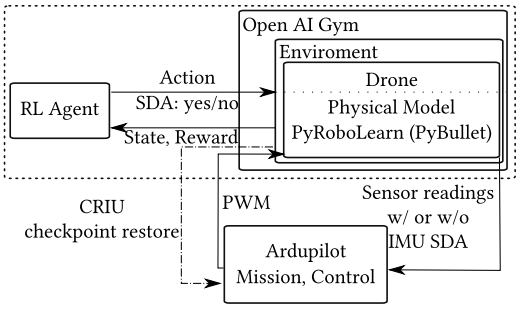


Figure 9: Controllable attack synthesis framework

6 Targeted Attack Synthesis

In the previous sections, we introduced CONFUSENSE and evaluated their impact on drone operations and COTS flight controllers. In this section, we answer the following question,**RQ4**: *How can UAVs be adversarially controlled via* CONFUSENSE*?* To answer **RQ4** and solve the adversarial control problem identified in Equation 6, we investigate the controllability of the proposed CONFUSENSE and propose an attack synthesis methodology to fly the drone. The proposed framework, by using Reinforcement Learning (RL), allows the attacker to direct a drone toward the attacker's desired location. To reach the desired location, the attacker decides when the CONFUSENSE is performed ($\gamma_k \in \{0,1\}$, see Section 4.2).

The attack synthesis relies upon the following elements:

Goal. The goal is an X, Y, and Z coordinate to which we aim to deviate the drone. We randomize the goal at the beginning of each episode to guarantee learned policy generalizability.

Action Space. The action space is a boolean variable reporting whether the IMU readings should be attacked.

State Observations. The observation space is 25 dimensional and continuous, with drone position, quaternion orientation (4 values), Roll, pitch, and yaw angles in radians, the velocity vector in m/s (3 values), Angular velocity in radians/second (3 values), Motors' speeds in RPMs (4 values), the roll, pitch, and yaw first derivative and goal coordinates.

Reward Function. Our reinforcement learning agent for drone deviation learns using a custom reward function (Algorithm 1, Appendix B) designed for the dual of the adversarial control problem (Equation 6). This function evaluates the Euclidean distance to the goal (`goalDist`) and its change

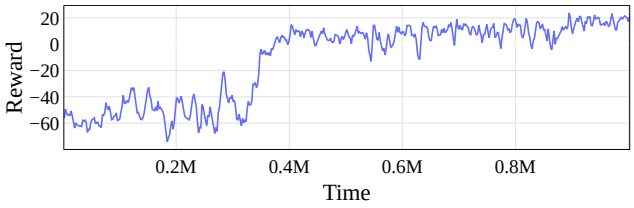


Figure 10: Attack synthesis: training reward, 1M iterations

(ΔDist). It rewards proper drone orientation (*droneUP()*) and penalizes flipping or being upside down. Additionally, it discourages following the controller's path and penalizes EKF failsafe triggers while rewarding movement towards the goal.

6.1 Attack Synthesis Framework

Framework Implementation. Figure 9, summarizes the framework architecture for our evaluation. We create an Open AI Gym [8] environment to allow the interaction between the Reinforcement Framework agent and Ardupilot. Ardupilot SITL allows for an external physical emulation [3] via UDP socket; we rely on this feature to enable the interaction between the reinforcement learning environment and Ardupilot. To initialize the Ardupilot controller in the environment, we use Checkpoint/Restore In Userspace (CRIU) to restore the Ardupilot process. The Ardupilot process image was taken while the drone was flying over a straight trajectory.

Learning is based on episodes. Each episode is composed of steps. In each step, the agent computes the ConfuSense decision based on the last received state and transmits it to the environment. The physical model processes the PWM input, simulates the corresponding physical response, and sends the normal or attacked sensor readings to Ardupilot, updating the drone's state. This update prompts the agent to begin the next step in the learning process.

Evaluation of the attack synthesis framework. We train the attack synthesis agent for 1M iterations (19 hours) using the TRPO algorithm [44], Figure 10 reports the reward evolution over the evaluation set. Over time, the model increases the average reward function value, which means that the agent policy learned how to control the drone via ConfuSense.

We assess the agent's ability to manipulate the drone's trajectory. Figure 11 presents sample traces demonstrating how the learned policy deviates the drone to the left or the right from the Ardupilot-planned path. Under normal conditions, the drone follows a straight trajectory unless ConfuSense is executed. The figure highlights the steps where the synthesis framework applies ConfuSense actions. The results confirm that the synthesis method can time ConfuSense to divert the drone without triggering a failsafe.

To answer **RQ4**, we propose an attack synthesis methodology. Our evaluation results show that the target UAV can deviate from the planned trajectory to the attacker's desired location. Compared to prior work attacks, the proposed framework represents an advancement in the attacker capability as previously demonstrated attacks will imply drone crashing [23, 49].

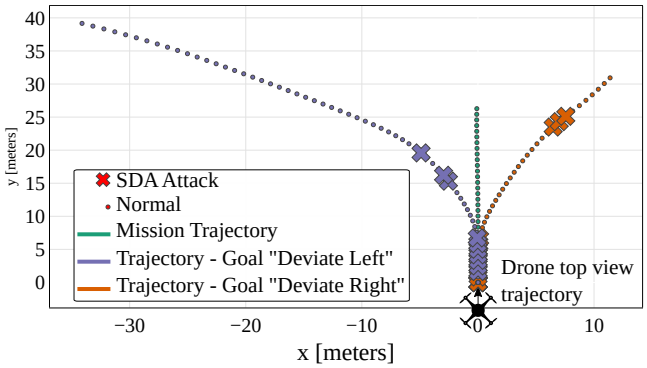


Figure 11: Drone Traces, the attack synthesis learned policy. Given the objective, the drone will turn left or right via intermittent ConfuSense

7 Detection and Mitigations

7.1 Detecting ConfuSense

In previous sections, we have presented and evaluated a novel attack vector against UAVs. We are now interested in understanding the efficacy of detection mechanisms to prevent ConfuSense attack. Specifically, we address the following question **RQ5**: *Is the induced* ConfuSense *attack detected by State-of-the-Art detectors for UAVs?*

In Section 5.3, we assessed the impact of ConfuSense on the EKF failsafe algorithm implemented on Ardupilot, showing that the attacks are often undetected. To answer **RQ5**, we assess the impact of ConfuSense on UAV anomaly detectors. We evaluate ConfuSense on flight controllers with the state-of-the-art security reference monitor, M2MON [26] (see Section 2.4). To detect an attack, M2MON continuously monitors the MMIO (Memory-mapped I/O) address range of the MCU with three different metrics: *Access list* (list of MMIO address access), *Access chain* (order of MMIO address access), *Access frequency* (frequency of MMIO address access). Any deviation from regular access patterns indicates the presence of malicious activity. Additionally, to secure the EKF from attacks on the RTOS, M2MON runs EKF separately from the RTOS. We show that ConfuSense remains undetected or delays detection.

Monitoring MMIO Access. The Kakute H7 Mini flight controller (STM32H743 MCU and BMI270 IMU) uses SPI for MCU-to-IMU communication. Upon referencing the

datasheet of STM32H743, we find that the transmit data register (TXDR) for SPI is located at address `0x40013420` and the receive data register (RXDR) for SPI is located at address `0x40013430`. These registers are responsible for exchanging data with the IMU. Therefore, we inject counters into low-level SPI drivers for the ArduPilot firmware and count the number of accesses for the specified SPI MMIO addresses.

Anomaly Detection. CONFUSENSE does not alter the list and order of MMIO address access. Since the CONFUSENSE does not modify the control firmware, the MCU will continue querying the IMU. Therefore, the CONFUSENSE remains undetected under the first two M2MON metrics, i.e., access list and access chain. Furthermore, we investigate the change in the frequency of MMIO address access. To achieve this, we first calculate the maximum and minimum MMIO access frequencies over 300 seconds during the regular IMU operation. To validate if MMIO access always remains within the calculated boundary, we compare it with a 60 seconds trace of regular IMU operation. The detector has an accuracy of 0.999988 when monitoring an idle MCU. Finally, we perform sensor suspend on our IMU for 60 seconds and use the previously derived boundaries to detect CONFUSENSE.

Sensor Suspend Attack Detection. In this case, the detector has a low accuracy of 0.0002 when applied on suspend attacks. For this reason, we argue that this attack remains undetected. We attribute the undetected suspend attacks to the fact that there is no change in communication patterns with the BMI270 IMU. Moreover, monitoring the values set into the registers will not be sufficient to detect the device reconfiguration attack, as the attacker issues the reconfiguration through an out-of-bound method (e.g., IEMI) and not through the monitored MCU registers.

Sensor Frequency Attack Detection. In this case, the reconfigurations cause late detection. A sampling frequency attack reduces the communication frequency between the IMU and MCU, causing delayed detection of frequency changes due to 128 missing observations when the IMU frequency drops from 1600 Hz to 12.5 Hz. As noted by Jang et al. [23], detection delays reduce the chance of attack remediation, leading to a drone crash. A complete discussion of these results and implications is provided in Appendix A.3.

Process-based detection. Similarly to what we showed in Section 5.3 about the impact of CONFUSENSE on the EKF failsafe mechanism, the time to detect would also increase in the case of other proposed process-based anomaly detectors [12, 37] which rely on the same sensor observations used by the failsafe to detect anomalies. The delay induced by the frequency attack prevents the anomaly detection signal from reaching the detector, delaying or preventing the alarms.

To answer **RQ5**, we evaluate the proposed attacks against prior work anomaly detectors and show that the proposed suspend attack evades M2MON (accuracy drops from 0.999988 to 0.0002) and induces delays in the process-based anomaly detection, while the frequency attack will slow down the state estimation and delay detection.

7.2 Countermeasures

Chip-level Mitigation Mitigating the proposed attacks at the sensor chip level requires: i) enabling the chips' secure configuration with an authentication layer on the serial protocol. This is challenging as it induces computational and power overheads on resource-constrained devices and cannot be retrofitted to legacy chips. ii) Removing configuration functionalities from the sensors or allowing configuration only at sensor boot to impede the device configuration overwriting.

Software-level Mitigation While software mitigations do not address the root cause of sensor reconfiguration attacks (i.e., these approaches do not prevent the attacker from reissuing the malicious configuration commands), they are easier to deploy on legacy hardware. Periodic attestation of sensor configurations or bus monitoring for anomalous requests can enable attack detection. Upon detection, the microcontroller will restore the correct sensor configuration or initiate a recovery procedure [13].

After sharing our results with the ArduPilot community, the core developers suggested leveraging an existing ArduPilot feature for our attestation use case. To guard against sensor brownouts, ArduPilot uses *checked registers*, which synchronously verify that critical registers for supported sensors remain unchanged upon data availability. This feature is only enabled for specific sensors experiencing brownouts and is not uniformly implemented among them. Moreover, the current design does not perform this check when a sensor's output is suspended (preventing data availability). To address both limitations and cover our threat model to enable CONFUSENSE detection, we propose extending the *checked register* mechanism to run asynchronously and independently of the sensor. While this extension would provide full coverage, it introduces additional bus traffic that may conflict with the system's real-time requirements.

Proactive Solutions Under legacy and computational constraints, hardware mitigation is needed to prevent supply chain and IEMI attacks.

Hardware Supply Chain Security. Modern circuits have billions of components, making their verification a complex task. Recent large-scale hardware supply chain attacks involved the modification of devices to contain explosives [6, 20]. The detection of Hardware Trojans has been studied in prior work by proposing hardware verification techniques [7, 10, 34, 39]. In response to these risks, governments have proposed control over hardware supply chains [1, 35].

IEMI Defenses. Prior research has proposed solutions to mitigate the effects of intentional electromagnetic interference (IEMI) on analog signals [28]. These countermeasures

fall into two main categories: analog defenses (such as shielding, differential comparators, and filters) and digital defenses (such as IEMI detection via *signal contamination monitoring*). However, to the best of our knowledge, such countermeasures are not widely implemented. Moreover, analog solutions such as shielding may be deemed ineffective [23].

Attack Recovery Even if the reconfiguration attack is detected through attestation or bus monitoring, recovery is not straightforward. Reissuing the correct configuration commands would be insufficient, as the attacker could repeat the attack. Khan et al. [26] proposed opening the parachute immediately upon attack detection. Choi et al. [11] suggested using software sensing techniques to recover from attacks. Dash et al. proposed diagnosis-based [14] and reinforcement learning-based [13] recovery strategies. We argue that the appropriate recovery strategy depends on the context in which the RV is deployed. While software sensing may allow drone recovery, parachute opening represents a conservative fallback option.

8 Related Work

We discussed relevant related work in Section 2. In this section, we report complementary related works.

Control in Presence of Missing Deadlines. Control under missing observations or commands has been widely studied. Sinopoli et al. [47] analyzed Kalman Filter behavior under network faults, showing that excessive packet loss leads to unbounded state covariance. Schenato et al. [41, 42] explored control under lossy networks using zero-input and hold-input strategies. Maggio et al. [31] examined stability under sporadic deadline misses. While prior work focuses on stochastic faults, our setting involves attacker-induced misses. Li et al. [30] introduced TimeTrap, a fault injection tool to identify exploitable timing patterns in control software.

Adversarial Reinforcement Learning. Recently, Reinforcement Learning techniques have been applied to security-related tasks for attacks and defenses. On the defensive side, Landen et al. [29] recently proposed a deep reinforcement learning agent to operate the power grid and perform anomaly detection, while Maiti et al. [32] proposed a reinforcement learning agent to perform vulnerability assessment of the power grid. On the attack side, Wu et al. [52] and Gong et al. [19] proposed policies to train deep reinforcement learning adversarial agents to attack state-of-the-art reinforcement learning agents. We are the first to use RL to control the victim adversarially.

9 Discussion

Feasibility of Controlled Attacks In our study, the controlled attack was evaluated in simulation to explore the theoretical upper bound of its potential impact. In this setting, the attacker has access to sufficient state feedback (e.g., position, velocity) to steer the drone toward a targeted direction. This state feedback may be approximated in scenarios where external localization leaks (e.g., compromised telemetry links or visual tracking systems).

Our implementation does not assume that the RL agent runs directly on the RV. Instead, the agent is trained offline and produces a control policy that can be executed with minimal computational overhead. In practice, the RL policy could be deployed on an attacker's nearby companion device or on compromised onboard components. Recent work [13] has demonstrated the deployment of RL-based policies for attack recovery on drone-embedded hardware, suggesting that RL-enabled control of RVs is becoming increasingly viable.

Redundant and Heterogeneous Sensor Setups A multi-sensor attack strategy would be required in systems with redundant sensors connected over independent communication channels or using different protocols. This increased attack surface complexity does not render the attack infeasible.

Ethical Considerations and Vulnerability Disclosure Our research does not involve human subjects or the use of data derived from them, and it poses no direct risk to individuals. The sensor reconfiguration explored in our work leverages an intended feature of the sensors rather than exploiting any vulnerability or flaw. To promote responsible research practices and contribute to system improvements, we shared our findings with the ArduPilot developer community (https://github.com/ArduPilot/ardupilot/issues/29545). This will help encourage enhanced handling of sensor reconfiguration, ultimately leading to more robust, safe, and reliable system behavior.

10 Conclusion

We presented the novel ConfuSense attack, which exploits sensor reconfiguration to achieve process destabilization. The proposed attacks differ from prior work attacks on UAVs (e.g., GPS spoofing) as the attacker does not need to continuously spoof sensor readings. The proposed attack is practically demonstrated and evaluated on COTS flight controllers. We investigate and describe the effects of the attack at the hardware and control level, showing that the attacks severely delay the frequency of attitude estimation as the EKF estimation depends on IMU frequency. We demonstrate how to use ConfuSense to control the victim's vehicle. We show that the proposed attack can evade state-of-the-art UAV anomaly detection techniques (accuracy drops from ~1 to 0.0002) or delay attack detection while deviating the victim vehicle more than 30 meters from its trajectory. Our findings demonstrate how the proposed ConfuSense threatens modern UAVs and their consequences on state-of-the-art control software. To defend against such attacks, we discuss countermeasures ranging from chip design to software-level mitigations.

Acknowledgments

The authors gratefully acknowledge funding from the Helmholtz Association (HGF) within the topic "46.23 Engineering Secure Systems." In addition, this project was funded by the Federal Ministry for Digital and Transport (BMDV) under the funding code 45AVF5A011. The responsibility for the content of this publication lies with the authors. This work was also supported by the National Science Foundation (NSF) under the Cyber-Physical Systems (CPS) and Secure and Trustworthy Cyberspace (SaTC) programs.

References

[1] 117TH CONGRESS. Chips and science act. https://www.congress.gov/bill/117th-congress/house-bill/4346#:~:text=Public%20Law%20No%3A%20117%2D167%20(08%2F09%2F,of%20the%20federal%20science%20agencies., Accessed January 2025.

[2] ARDUPILOT. Ardupilot. https://ardupilot.org/ardupilot/index.html, Accessed January 2025.

[3] ARDUPILOT. Ardupilot: external emulation. https://ardupilot.org/dev/docs/sitl-with-JSON.html, Accessed January 2025.

[4] ARDUPILOT. Ekf failsafe. https://ardupilot.org/copter/docs/ekf-inav-failsafe.html, Accessed January 2025.

[5] ARDUPILOT. Extended kalman filter navigation overview and tuning. https://ardupilot.org/dev/docs/extended-kalman-filter.html#extended-kalman-filter, Accessed January 2025.

[6] BARKER, K., AND SCHWIRTZ, M. With explosive goggles, ukraine sought to blast russian drone operators. https://www.nytimes.com/2025/02/20/world/europe/urkaine-russia-explosive-goggles.html, Accessed February 2025.

[7] BHASIN, S., AND REGAZZONI, F. A survey on hardware trojan detection techniques. In *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS)* (2015), IEEE.

[8] BROCKMAN, G., CHEUNG, V., PETTERSSON, L., SCHNEIDER, J., SCHULMAN, J., TANG, J., AND ZAREMBA, W. Openai gym, 2016.

[9] CÁRDENAS, A. A., AMIN, S., LIN, Z.-S., HUANG, Y.-L., HUANG, C.-Y., AND SASTRY, S. Attacks against process control systems: risk assessment, detection, and response. In *Proc. of the ACM Asia Conference on Computer and Communications Security (CCS)* (2011).

[10] CHAKRABORTY, R. S., WOLFF, F., PAUL, S., PAPACHRISTOU, C., AND BHUNIA, S. MERO: A statistical approach for hardware trojan detection. In *Proc. of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (2009), Springer.

[11] CHOI, H., KATE, S., AAFER, Y., ZHANG, X., AND XU, D. Software-based realtime recovery from sensor attacks on robotic vehicles. In *Proceedings of International Symposium on Research in Attacks, Intrusions and Defenses (RAID)* (Oct. 2020), USENIX Association.

[12] CHOI, H., LEE, W.-C., AAFER, Y., FEI, F., TU, Z., ZHANG, X., XU, D., AND DENG, X. Detecting attacks against robotic vehicles: A control invariant approach. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2018).

[13] DASH, P., CHAN, E., AND PATTABIRAMAN, K. Specguard: Specification aware recovery for robotic autonomous vehicles from physical attacks. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2024).

[14] DASH, P., LI, G., KARIMIBIUKI, M., AND PATTABIRAMAN, K. Diagnosis-guided attack recovery for securing robotic vehicles from sensor deception attacks. In *Proc. of the ACM Asia Conference on Computer and Communications Security (CCS)* (2024).

[15] DAYANIKLI, G. Y., MOHAMMED, A. Z., GERDES, R., AND MINA, M. Wireless manipulation of serial communication. In *Proc. of the ACM Asia Conference on Computer and Communications Security (CCS)* (2022).

[16] DAYANIKLI, G. Y., SINHA, S., MUNIRAJ, D., GERDES, R. M., FARHOOD, M., AND MINA, M. Physical-Layer attacks against pulse width Modulation-Controlled actuators. In *Proc. of the USENIX Security Symposium* (Aug. 2022), USENIX Association.

[17] ERBA, A., AND TIPPENHAUER, N. O. Assessing model-free anomaly detection in industrial control systems against generic concealment attacks. In *Proc. of the Annual Computer Security Applications Conference (ACSAC)* (Austin, USA, Dec. 2022), ACM.

[18] FORSTER, V. Drone flies lungs between hospitals for transplant patient. https://www.forbes.com/sites/victoriaforster/2021/10/14/drone-flies-lungs-between-hospitals-for-transplant-patient, 2021.

[19] GONG, C., YANG, Z., BAI, Y., SHI, J., SINHA, A., XU, B., LO, D., HOU, X., AND FAN, G. Curiosity-driven and victim-aware adversarial policies. In *Proc. of the*

38th Annual Computer Security Applications Conference (New York, NY, USA, 2022), ACSAC '22, ACM, p. 186–200.

[20] HAWKINS, A., AND GEDEON, J. Middle east pager attacks ignite fear of supply chain warfare. https://www.politico.com/news/2024/09/19/pager-attacks-supply-chain-warfare-00180136, Accessed January 2025.

[21] HOLYBRO. Holybro pixhawk 6c, system diagram and pinout. https://docs.holybro.com/autopilot/pixhawk-6c/system-diagram-and-pinout, 2023.

[22] INVENSENSE. Mpu-6000 and mpu-6050 register map and descriptions revision 4.2. https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf, Accessed January 2025.

[23] JANG, J., CHO, M., KIM, J., KIM, D., AND KIM, Y. Paralyzing drones via emi signal injection on sensory communication channels. In *Proc. Network and Distributed System Security Symposium (NDSS)* (2023).

[24] JEONG, J., KIM, D., JANG, J., NOH, J., SONG, C., AND KIM, Y. Un-rocking drones: Foundations of acoustic injection attacks and recovery thereof. In *Proc. Network and Distributed System Security Symposium (NDSS)* (2023).

[25] KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering 82*, 1 (03 1960), 35–45.

[26] KHAN, A., KIM, H., LEE, B., XU, D., BIANCHI, A., AND TIAN, D. J. M2mon: Building an mmio-based security reference monitor for unmanned vehicles. In *Proc. of the USENIX Security Symposium* (Aug. 2021), USENIX Association.

[27] KROTOFIL, M., CÁRDENAS, A. A., MANNING, B., AND LARSEN, J. Cps: Driving cyber-physical systems to unsafe operating conditions by timing dos attacks on sensor signals. In *Proceedings of the 30th Annual Computer Security Applications Conference* (2014).

[28] KUNE, D. F., BACKES, J., CLARK, S. S., KRAMER, D., REYNOLDS, M., FU, K., KIM, Y., AND XU, W. Ghost talk: Mitigating emi signal injection attacks against analog sensors. In *Proc. of the IEEE Symposium on Security and Privacy* (2013), IEEE.

[29] LANDEN, M., CHUNG, K., IKE, M., MACKAY, S., WATSON, J.-P., AND LEE, W. DRAGON: Deep reinforcement learning for autonomous grid operation and attack detection. In *Proc. of the Annual Computer Security Applications Conference (ACSAC)* (2022), ACM.

[30] LI, A., WANG, J., BARUAH, S., SINOPOLI, B., AND ZHANG, N. An empirical study of performance interference: Timing violation patterns and impacts. In *Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2024), IEEE Computer Society Press.

[31] MAGGIO, M., HAMANN, A., MAYER-JOHN, E., AND ZIEGENBEIN, D. Control-system stability under consecutive deadline misses constraints. In *Euromicro Conference on Real-Time Systems (ECRTS 2020)* (2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[32] MAITI, S., BALABHASKARA, A., ADHIKARY, S., KOLEY, I., AND DEY, S. Targeted attack synthesis for smart grid vulnerability analysis. In *Proc. of the ACM Conference on Computer and Communications Security (CCS)* (2023).

[33] NASSI, B., BITTON, R., MASUOKA, R., SHABTAI, A., AND ELOVICI, Y. SoK: Security and privacy in the age of commercial drones. In *Proc. of the IEEE Symposium on Security and Privacy* (2021), IEEE.

[34] NGUYEN, L. N., CHENG, C.-L., PRVULOVIC, M., AND ZAJIĆ, A. Creating a backscattering side channel to enable detection of dormant hardware trojans. *IEEE transactions on very large scale integration (VLSI) systems 27*, 7 (2019).

[35] PARLAMENT, E. European chips act (semi-conductors). https://www.europarl.europa.eu/legislative-train/theme-a-europe-fit-for-the-digital-age/file-european-chips-act-(semiconductors)?sid=7301, Accessed January 2025.

[36] PRAVDA, U. Ukraine has 250-300 drones in the air simultaneously in some contact line areas. https://news.yahoo.com/ukraine-250-300-drones-air-064501794.html?, September 2023.

[37] QUINONEZ, R., GIRALDO, J., SALAZAR, L., BAUMAN, E., CARDENAS, A., AND LIN, Z. SAVIOR: Securing autonomous vehicles with robust physical invariants. In *Proc. of the USENIX Security Symposium* (Aug. 2020), USENIX Association.

[38] RISEBOROUGH, P. Inertialnav implementation. https://github.com/priseborough/InertialNav, Accessed January 2025.

[39] SALMANI, H., TEHRANIPOOR, M., AND PLUSQUELLIC, J. A novel technique for improving hardware trojan detection and reducing trojan activation time. *IEEE transactions on very large scale integration (VLSI) systems 20*, 1 (2011).

[40] SATHAYE, H., STROHMEIER, M., LENDERS, V., AND RANGANATHAN, A. An experimental study of GPS spoofing and takeover attacks on UAVs. In *Proc. of the USENIX Security Symposium* (Aug. 2022), USENIX Association.

[41] SCHENATO, L. To zero or to hold control inputs with lossy links? *IEEE Transactions on Automatic Control 54*, 5 (2009), 1093–1099.

[42] SCHENATO, L., SINOPOLI, B., FRANCESCHETTI, M., POOLLA, K., AND SASTRY, S. S. Foundations of control and estimation over lossy networks. *Proc. of the IEEE 95*, 1 (2007).

[43] SCHILLER, N., CHLOSTA, M., SCHLOEGEL, M., BARS, N., EISENHOFER, T., SCHARNOWSKI, T., DOMKE, F., SCHÖNHERR, L., AND HOLZ, T. Drone security and the mysterious case of dji's droneid. In *Proc. Network and Distributed System Security Symposium (NDSS)* (2023).

[44] SCHULMAN, J., LEVINE, S., ABBEEL, P., JORDAN, M., AND MORITZ, P. Trust region policy optimization. In *International conference on machine learning* (2015), PMLR.

[45] SELVARAJ, J., DAYANIKLI, G. Y., GAUNKAR, N. P., WARE, D., GERDES, R. M., AND MINA, M. Electromagnetic induction attacks against embedded systems. In *Proc. of the ACM Asia Conference on Computer and Communications Security (CCS)* (2018).

[46] SHARWOOD, S. Cardboard drones running open-source flight software take off in ukraine and beyond. https://www.theregister.com/2023/04/07/corvo_cardboard_drone/, 2023.

[47] SINOPOLI, B., SCHENATO, L., FRANCESCHETTI, M., POOLLA, K., JORDAN, M. I., AND SASTRY, S. S. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control 49*, 9 (2004).

[48] SNOUFFER, E. Six places where drones are delivering medicines. *Nat Med 28*, 5 (2022).

[49] SON, Y., SHIN, H., KIM, D., PARK, Y., NOH, J., CHOI, K., CHOI, J., AND KIM, Y. Rocking drones with intentional sound noise on gyroscopic sensors. In *Proc. of the USENIX Security Symposium* (Washington, D.C., Aug. 2015), USENIX Association.

[50] WALMART. Walmart drone delivery by the numbers. https://corporate.walmart.com/newsroom/2023/01/05/walmart-drone-delivery-by-the-numbers, Accessed January 2025.

[51] WIKIPEDIA. Amazon prime air. https://en.wikipedia.org/wiki/Amazon_Prime_Air, 2023.

[52] WU, X., GUO, W., WEI, H., AND XING, X. Adversarial policy training against deep reinforcement learning. In *Proc. of the USENIX Security Symposium* (Aug. 2021), USENIX Association.

[53] XIE, Z., YAN, C., JI, X., AND XU, W. Bitdance: Manipulating UART serial communication with IEMI. In *Proc. of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)* (2023).

[54] ZHANG, M., AND ZONOUZ, S. Control corruption without firmware infection: Stealthy supply chain attacks via plc hardware implants (maltag). In *Proc. of the International Conference on Cyber-Physical Systems (ICCPS)* (2024), IEEE.

[55] ZHANG, Y., AND RASMUSSEN, K. Electromagnetic signal injection attacks on differential signaling. In *Proc. of the ACM Asia Conference on Computer and Communications Security (CCS)* (2023).

A Alternative CONFUSENSE reconfiguration: Frequency Attack

To launch CONFUSENSE, an alternative to sensor suspension is reconfiguring the sensor to operate at a lower sampling frequency. Since the sensors commonly found on modern flight controllers are general-purpose devices (e.g., the same IMU chip can be used in various contexts, for example, in smartphones and activity trackers) they can be programmed for different purposes. The sensor's sampling frequency is one feature that can be reconfigured to serve a particular task. In flight control software, sensors are often configured to run at the highest sampling frequency, this allows precise control of the aircraft. An attacker can resort to this device configuration to decrease the update rate of the sensor, in this way the sensor readings are updated less frequently, and the controller will rely on outdated sensor readings or wait until the next sample is collected. The frequency reconfiguration can be modeled as an intermittent suspend reconfiguration where the suspension lasts $\frac{1}{\text{sensor frequency}}$.

A.1 Reconfiguration via IEMI

To achieve the sensor sampling frequency change in the MPU6050, we have to write in the `sampling rate divider` register `0x19`. The value contained in this register will be later used to determine the sampling rate as in Equation 11.

$$\text{Sample Rate} = \frac{\text{Gyroscope Output Rate}}{(1 + \text{Sampling Rate Divider})} \tag{11}$$

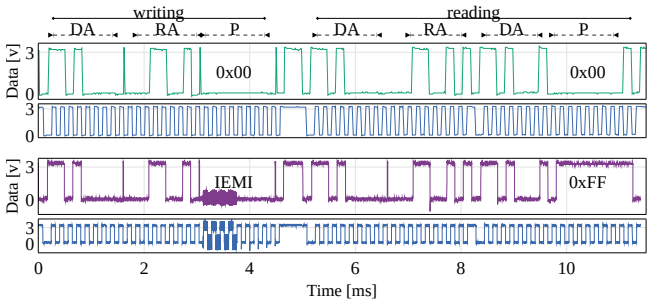


Figure 12: Sensor sampling frequency modification via IEMI attack on MPU6050, the IEMI modifies the payload of the message from 0x00 to 0xFF.

Table 5: Summary of IMU behaviors observed during IMU frequency attack

	Lowest frequency (Hz)			
	Accelerometer		Gyroscope	
IMU Model	Tested	Allowed	Tested	Allowed
BMI055	15.56	7.81	32	32
BMI270	12.5	0.78125	25	25
ICM-42688-P	12.5	1.5625	12.5	12.5
MPU6000	100	50	100	50

Figure 12, represents the collected power trace at the oscilloscope, with and without injection. As we can observe, when the IEMI injection is performed (lasting 640μs, i.e., 4 clock cycles), the response from the register changes, and the value that is stored in the register is `0xFF`, which is the highest 8-bit unsigned integer, corresponding to the lowest sampling frequency configurable (31.25Hz).

A.2 Effects of IMU Sampling Frequency

To test the effect of frequency change on the control software (Ardupilot), we set all the possible allowed sampling frequencies in the target IMU. The sensor is by default configured to operate at the highest sampling frequency (e.g., for the BMI 270, 3200Hz for the gyroscope, and 1600Hz for the accelerometer). Then, we reconfigure the sensor to operate at a lower frequency. Figure 13a, shows the impact of the frequency change compared to the Kalman Filter estimation frequency of the system when performing experiments on the the four considered IMUs that show consistent behavior. In Ardupilot, the Kalman Filter prediction is scheduled for computation at 400Hz. As long as the sensor sampling frequency exceeds 400Hz, the attack does not show any consequence on the controller. When the sensor frequency is reduced below 400Hz, we can observe that the Kalman Filter estimation gets slowed down. This has severe consequences on the scheduler and drone control. In a cascading effect, the drone attitude estimation will not produce fresh outputs and consequently, no control commands are computed. This effect can be seen as a realization of intermittent absent data values or as a stale data attack where the duration of the attack is $\frac{1}{\text{sensor frequency}}$. Moreover, we observed scheduled tasks in the ROTS being slowed down due to deadline misses (e.g., data transfer to the ground station will slow down, and barometer sensors will be sampled less frequently). In the following paragraph, we comment further on the observed consequences of CONFUSENSE.

In our experiments, we were able to slow down the sampling frequency of all the considered IMUs, impacting the Kalman Filtering operations. Table 5, reports the lowest possible frequencies that can be set on each IMU.

Explanation for observed delay effects. Given the severe consequences of IMU sampling frequency on EKF frequency and other sensors (e.g., the barometer), we investigated the root cause of this behavior by looking into the ArduPilot codebase. We observed that the scheduler loop runs continuously inside an infinite while loop. The scheduler loop waits for new data from IMU and manages 81 tasks (such as the EKF state estimator, attitude controllers, and checking if the UAV has landed or crashed). Differently from what is expected, although the reading of data from IMU is performed asynchronously, when the `sensor frequency < scheduler frequency` the scheduler loop waits for new IMU data to continue its execution. Consequently, all the tasks are not performed until a new reading is obtained from IMU.

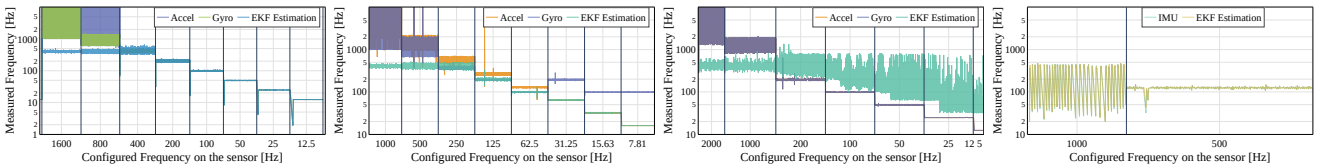
Our CONFUSENSE exposes a controller design flaw, which makes it possible to delay the tasks inside the firmware from a ROTS execution inside the MCU from an untrusted sensor.

Lowest frequency for control. Using the Ardupilot SITL simulator, we test the frequency change attack to verify the impact of the sensor frequency change on the flight and observe which is the frequency below which the flight controller will fail in controlling the drone. In our evaluation, we found that below 200Hz the drone becomes unstable but continues the mission. Instead, below 150Hz the controller will fail to stabilize the drone and crash on the ground. All the tested IMU allow frequencies lower than 150Hz.

Interestingly, we discovered that the sensor sampling frequency attack would force the attitude estimation to slow down to the frequency of the sensor and impact the whole RTOS scheduler. This complements our previous findings for **RQ2** and presents an effective manipulating alternative.

A.3 Sampling Frequency Attack Detection

In this case, we observe that the detector has an accuracy of 0.9377 for 100 Hz and 0.9923 at 12.5 Hz reconfiguration. This is because a sampling frequency attack leads to decreased communication frequency between the IMU and



(a) BMI270 (b) BMI055 (c) ICM42688 (d) MPU6000

Figure 13: Dependency between the IMU sampling frequency and the attitude estimation frequency. Below 400Hz, sensor frequency slows down to the state estimation.

the MCU. We argue that while the detector would be able to observe a change in access frequency, this detection will be delayed by the lower frequency of the IMU as there are 128 missing consequent observations when changing the IMU frequency from 1600Hz to 12.5Hz. We note that all the observations will be missing in the case of absent data. Moreover, M2MON-EKF would wait for subsequent IMU readings for state estimation, delaying the overall operations of M2MON (32 skipped state estimations, i.e., 12.5Hz vs 400Hz). As already noted by Jang et al. [23], induced higher time to detect the anomaly will consequently reduce the probability of attack remediation (failsafe/parachute) that leads the drone to crash. We note that since the implementation of M2MON-EKF is the same EKF implementation in ArduPilot, moving EKF implementation outside RTOS offers no additional security against ConfuSense.

B Reward Function

In this section, we present the reward function to train the RL agent to control the drone using ConfuSense. In a nutshell, the reward function aims to generate a policy that flies the drone to a desired position (GoalPos) without losing control (reduced amount of flips) and without triggering any safety mechanism (failsafe). The Reward function penalizes the trigger of the failsafe mechanism (line 22 in Algorithm 1) and reduces the reward when the drone flips (line 16).

Algorithm 1: Reward function

```
1  flips = 0
2  function Reward(state)
3      reward = 0
4      goalDist = sqrt((DronePos − GoalPos)^2)
5      ΔDist = prevDist - goalDist
6      if droneUP() and closeToGoal(ΔDist) then
7          reward = reward+1.5
8      else
9          reward = reward-0.5
10     if !droneUP() and closeToGoal(ΔDist) then
11         reward = reward+0.5
12     else
13         reward = reward-1.5
14     if droneFilps() then
15         flips = flips + 1
16         reward = -flips
17     else
18         flips = 0
19     if closeToPlannedPath() then
20         reward = reward - 2
21     if failsafeTriggered() then
22         reward = -40
23     if goalDist < 1 then
24         reward = 100
25     return reward
```