Critical Sensor Failure Analysis for Proactive Mitigation in Secure System-on-Chip for UAVs

Yashrajsinh Parmar*, Junior Sundar†, Rafail Psiakis*, Florian Caullery*, Martin Andreoni*, Ari Kulmala*
*Secure Systems Research Center - Technology Innovation Institute (TII), Abu Dhabi, United Arab Emirates
†Unikie MEA Ltd., Abu Dhabi, United Arab Emirates

***Abstract*—Unmanned Aerial Vehicles (UAVs) have revolutionized various industries by offering versatile surveillance, delivery, agriculture, and more capabilities. However, ensuring their safe operation remains a paramount concern. This paper delves into analyzing critical time-to-failure scenarios stemming from drone sensor malfunctions. We explore the performance thresholds demanded by the System-on-Chip (SoC) to swiftly detect and mitigate impending failures, thereby averting catastrophic flight incidents. We demonstrate that the SoC must promptly identify any faults in the drone sensor within a mere 0.56 seconds to prevent the risk of flight failure. We take an open-source RISC-V-based SoC as a reference and identify key factors influencing failure criticality through empirical analysis and simulation.**

I. Introduction

The rapid rise of UAVs has revolutionized industries like surveillance, delivery, and agriculture. With their ability to operate autonomously, they offer new efficiencies and possibilities. For instance, the global drone taxi market is expected to hit USD 869.9 million by 2030[1]. However, this surge in UAV usage raises significant safety concerns [3] that must be addressed to prevent accidents, ensure public safety, and maintain trust in the technology.

The safe operation of UAVs relies heavily on the reliability of their onboard sensors, which provide critical flight data such as altitude, velocity, and orientation. Any malfunctions or inaccuracies in sensor data can lead to disastrous flight incidents if not promptly detected and addressed [12]. Thus, fault detection and sensor reliability are crucial for ensuring UAV safety. Robust fault detection mechanisms help identify potential issues early on, enhancing the resilience and dependability of autonomous systems. Prioritizing sensor reliability significantly reduces the risks associated with UAV operations, thereby fostering greater trust and wider adoption of drone technologies.

Previous studies on sensor fault detection have primarily focused on model-driven [7, 10, 11] and data-driven approaches [16, 8, 18, 19]. With advancements in deep learning algorithms like Long-Short Term Memory (LSTM)[14] and Convolution Neural Network (CNN)[2], data-driven methods are increasingly preferred over traditional model-specific techniques. Unlike model-driven methods requiring precise physical models, data-driven approaches utilize historical flight data to detect anomalies. Neural network analysis of flight data enables more accurate and efficient fault detection, enhancing UAV safety. This shift towards data-driven solutions highlights the potential of machine learning to transform sensor fault detection, providing a more adaptable and robust framework for ensuring UAV safety.

Understanding the architecture of modern UAV platforms is crucial for analyzing the impact of sensor failures. One example is the Shaheen open-source RISC-V-based SoC for nano-UAVs [17]. Shaheen SoC functions include processing sensor data, executing control algorithms, collecting flight logs, and monitoring sensor data for fault detection. Upon detecting a sensor fault, the system applies mitigation strategies to ensure safe operation [15]. Sensor data fault detection can be achieved using data-driven models on either on-chip AI accelerators [9] or external AI accelerators like the NVIDIA Jetson Orion AGX. The choice of processing unit depends on the complexity of fault detection models. For simpler models, processing within the SoC is efficient. However, for more complex models requiring higher computational power, external accelerators may be utilized. This processing flexibility enables UAV platforms to maintain high reliability and safety standards by effectively detecting and mitigating sensor faults.

The computation of data-driven models must consider the Lead Time. Lead Time refers to the interval between the occurrence of a fault and its manifestation as a system failure. Despite its importance, very few studies have examined Lead Time across sensor failure scenarios. Authors in [4] focused solely on the Lead Time for GPS sensors. However, their study did not account for the reduction in Lead Time resulting from mitigation latency—a crucial factor in preventing a flight transition into a Failed state. Additionally, the study overlooked the impact of other sensor failures, such as those in the gyroscope, accelerometer, magnetometer, barometer, and distance sensor, on Lead Time. Understanding and addressing these gaps is vital for developing comprehensive fault detection and mitigation strategies that enhance the reliability and safety of UAV operations.

In this paper, we delve into analyzing critical time-to-failure scenarios stemming from drone sensor malfunctions. We explore the performance thresholds demanded by the System-on-Chip (SoC) to swiftly detect and mitigate impending failures, thereby averting catastrophic flight incidents. Using an open-source RISC-V-based SoC as a reference [17], we identify key factors influencing failure criticality through empirical analysis and simulation. The major contributions of this work are (i) Using Hardware in The Loop (HiTL) setup, we measure the Lead times for various faults for the Gyroscope, Accelerometer, Magnetometer, and Barometer,

[1]https://www.emergenresearch.com/industry-report/drone-taxi-market

TABLE I
LEAD TIME (ALL VALUES ARE IN SECONDS) FOR VARIOUS SENSOR FAULTS IN A PX4 FLIGHT CONTROLLER. GYRO: GYROSCOPE, ACCEL: ACCELEROMETER, BARO: BAROMETER & MAG: MAGNETOMETER; *: GPS FAULT INCLUDES SIMULTANEOUS FAULT INJECTION TO LATITUDE, LONGITUDE & ALTITUDE

Fault Type	Gyro	Accel	GPS*	Baro	Mag
Maximum Value	0.92	2.11	2.84	No Fail	112.89
Minimum Value	0.91	1.73	5.60	No Fail	No Fail
Sensor Failure	1.56	3.22	5.62	No Fail	No Fail

thereby resulting in performance requirements for the System-on-Chip (SoC). (ii) We discuss the challenges of executing fault detection algorithms on external AI Accelerator (iii) We present a detailed performance analysis of various data-driven fault detection models [16, 8, 18, 19] for processing within the SoC and using the external AI Accelerator.

The remainder of this paper is structured as follows. Section II identifies common drone sensor types and examines potential failure scenarios and their impact on flight safety. Section III gives an overview of the configuration for executing Fault detection algorithms within the SoC or external AI Accelerator. Section IV discusses the performance requirements of SoC in flight controllers. Section V concludes the paper.

II. CRITICAL SENSOR FAILURES IN DRONES

This section describes the setup of Hardware in The Loop (HiTL), which simulates the sensor fault and flight failure conditions in real time. We then identify potential failure scenarios for each type of sensor and assess the impact of sensor failures on Lead Time.

Figure 1 illustrates the HiTL simulation setup. Conducted on a Lenovo ThinkPad T14 Gen 4 with a 13th Gen Intel i7-1355U (12) @ 5.0 GHz processor, the experiment employs a Pixhawk v5 flight controller. This controller features an ARM Cortex-M7@ 216MHz flight controller CPU and supports 3x Inertial Measurement Unit (IMU), Barometer, and Magnetometer. Utilizing jMAVSim as the simulation environment, it seamlessly integrates with PX4-Autopilot for hardware-in-the-loop implementation. In HiTL, PX4 firmware operates on genuine flight controller hardware, with controller inputs and outputs interfaced between the simulation environment. Custom parameters in the sensor module source code simulate sensor fault lead time. Flight failure categorization includes three qualitative state tags: `ACCEPTABLE`, `UNACCEPTABLE`, and `FAILURE`. `ACCEPTABLE` signifies the drone remaining within a 0.5m radius of its hovering location, `UNACCEPTABLE` denotes violation of this boundary within a 2.0m radius, and `FAILURE` indicates complete boundary breach, including scenarios like drone overturning or forced landing. Lead time calculation is based on fault injection time and flight time to `FAILURE`.

From the work [4], the following faults show worst-case flight failure lead times:

- Maximum Value: The input value of the sensor is set to maximum

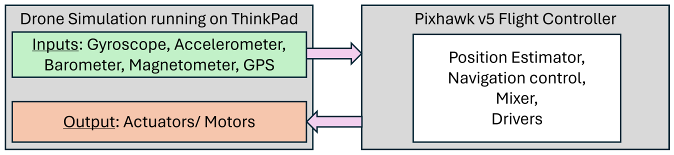
Drone Simulation running on ThinkPad
Inputs: Gyroscope, Accelerometer, Barometer, Magnetometer, GPS
Output: Actuators/ Motors
Pixhawk v5 Flight Controller
Position Estimator, Navigation control, Mixer, Drivers

Fig. 1. Setup for HiTL simulation

- Minimum Value: The input value of the sensor is set to minimum
- Sensor Failure: There is no input value available from the sensor

Table I compares the lead times of different sensor failures in a typical Drone running a PX4 flight controller. The lead time is measured by performing simulation using HiTL setup. The GPS sensor failure fault lead time aligns with the results from [4]. It is worth noting that the worst-case Lead Time for the gyroscope's Minimum Value fault is 0.909 sec.

Figure 2 shows the actual Lead time available for the system to detect fault. The mitigation latency to initiate emergency landing of the UAV, estimated using the HiTL simulation, is approximately 0.350 sec. The actual lead time available is $0.909 - 0.350$ sec $= 0.56$ sec. This implies that the system needs to detect the fault and take necessary mitigation actions within 0.56 sec only.

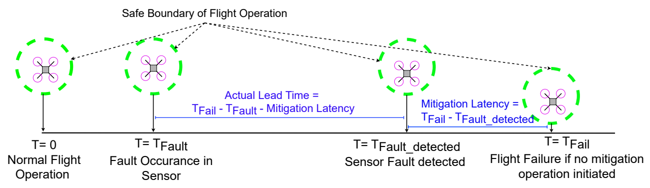
Safe Boundary of Flight Operation
Actual Lead Time = T_{Fail} - T_{Fault} - Mitigation Latency
Mitigation Latency = T_{Fail} - $T_{Fault\_detected}$
T= 0 Normal Flight Operation
T= T_{Fault} Fault Occurance in Sensor
T= $T_{Fault\_detected}$ Sensor Fault detected
T= T_{Fail} Flight Failure if no mitigation operation initiated

Fig. 2. Lead Time for Sensor Fault to Flight Failure

III. SoC ARCHITECTURE FOR FLIGHT CONTROLLER AND SENSOR FAULT DETECTION

This section explores the SoC architecture for executing flight controller and sensor fault detection functions, while enabling secure data transmission to prevent unauthorized manipulation.

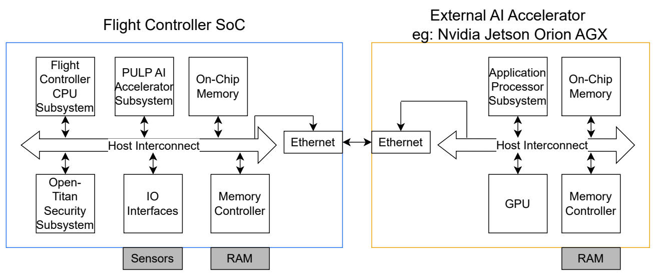
Flight Controller SoC
Flight Controller CPU Subsystem
PULP AI Accelerator Subsystem
On-Chip Memory
Host Interconnect
Ethernet
Open-Titan Security Subsystem
IO Interfaces
Memory Controller
Sensors
RAM
External AI Accelerator eg: Nvidia Jetson Orion AGX
Application Processor Subsystem
On-Chip Memory
Ethernet
Host Interconnect
GPU
Memory Controller
RAM

Fig. 3. SoC Block Diagram with an external AI Accelerator

Figure 3 displays the SoC block diagram, leveraging a design from [6]. It features a 64-bit RISC-V CVA6 CPU core

[20], supporting real-time execution of the flight controller stack. Peripheral data is managed via on-chip memory and various peripherals, facilitated by a dedicated μDMA [13]. Additionally, an on-chip AI Accelerator, PULP [9], with 8x CV32E-processor cores, aids in executing ML algorithms for sensor data fault detection [16, 8, 18, 19].

The on-chip AI Accelerator's processing capability reaches up to 7.9 GFLOp/s on 16-bit FP kernels while operating at max 500MHz frequency[9]. For more intensive tasks, an external AI Accelerator like NVIDIA Jetson Orion AGX can be employed, connected via Ethernet interface for real-time computational requirements (as shown in Figure 3).

The Ethernet interface serves two functions: securely transmitting flight logs, sensor outputs, and other critical data for fault detection models, while also handling non-secure data streams for untrusted applications. Although the Ethernet controller supports dedicated FIFOs for multiple data streams, it lacks a mechanism for generating unique Stream-IDs per data stream. Consequently, this deficiency prevents the implementation of access control measures to prevent unauthorized access to the secure data stream, jeopardizing its confidentiality and integrity.

To safeguard the confidentiality and integrity of critical data, it is imperative to encrypt the secure data and generate hashes on the plain data as a means of authentication. We use the crypto accelerator implemented using the Opentitan framework [6, 17] for encrypting and generating the hash of the message.

Two data paths are utilized for sensor fault detection based on computational complexity of the data-model.

- For fault detection models with low-computation requirement, the The PULP AI Accelerator conducts ML model inference, with the resulting output stored in on-chip memory for CPU processing.
- For fault detection models with high-computation requirement, Opentitan executes the Sensor data encryption and hash generation. Encrypted data and hash are transmitted to the external AI Accelerator via Ethernet. The external AI Accelerator verifies the hash, decrypts data, executes fault detection on GPU, and encrypts the result. The processed result is then transmitted back to the SoC, where it's verified, decrypted, and stored in on-chip memory for CPU action.

IV. Performance Analysis for Sensor Fault Detection

Now that we have discussed the possible data paths for sensor fault detection, we present the performance analysis of each data path in this section. We then identify the performance bottlenecks for each data path and propose potential SoC architecture updates for timely failure detection and mitigation, as discussed in Section II.

For our analysis, we consider the following sensors being interfaced with the Flight Controller SoC: 1)IMU: Dual ICM-42688-P + ICM-20649, 2) Barometer: Dual BMP388, 3) Magnetometer: BMM150 + IST8310 and 4) GPS: u-blox NEO-M9N. Based on the output of these sensors, the number of inputs for ML-based fault detection model is 19. The max sensor sampling frequency and PWM update rate for the motors is predefined by the PX4 Flight controller specification [1]. For executing fault detection on the sensor data, the sensor data is aggregated in the on-chip memory.

TABLE II
Performance of PULP AI Accelerator. TCN: Temporal Convolution Network, CNN: Convolution Neural Network, LSTM: Long-Short Term Memory Network, A: Accelerometer, G: Gyroscope, B: Barometer, Alt: Altitude, PWM: Actuator input; Latency numbers are in milliseconds

Work	ML Network	Model Input	Accuracy	Latency
[19]	TCN	G, B, PWM	94.76%	1.57
[18]	LSTM-RF	A, G, Alt	86.80%	0.74
[8]	1D CNN + LSTM	A, G, PWM	92.74%	6.51
[16]	LSTM+FC	A, G, PWM	96.30%	743.24

A. Performance Analysis of Fault Detection using On-Chip PULP AI Accelerator

The major computational bottleneck of fault detection using the on-chip AI Accelerator (refer to section III) is the low throughput of the PULP. In Table II, we present the performance achieved by the PULP AI Accelerator to detect sensor faults. We use the cycle-accurate GVSoC simulator for estimating the latency [5].

The works in [8, 18, 19] use the pre-trained ML-based fault detection model. These models consider a sliding window approach to generate the input data for the fault detection model. The size of the sliding window is restricted to very limited history of the sensor data (approximately 50 msec). The total data aggregated within 50 msec, from the above mentioned sensors, is approximately 14 KBytes. The advantage of this approach is faster inference time, as shown in Table II. However, since the view of the history is very limited, these models suffer from low accuracy. The work in [16], is based on detecting flight log anomalies that could result in physical instabilities online, while the drone is conducting a flight mission. The advantage of detecting faults based on the inputs of previous flight controller logs (of approximately 120 msec) and current sensor data is to achieve higher accuracy in fault detection. However, the disadvantage of such an approach is the latency of the inference. As shown in Table II, [16] requires 0.75 sec to perform a single inference. This violates the actual lead time of 0.56 sec (refer to Section IV).

Also, it is beneficial to execute multiple inferences within the boundary of the actual lead time to reduce the possibility of a False Positive or False Negative prediction result. Because of this reason, it is mandatory to execute the fault detection algorithm of [16] using an external AI Accelerator with higher computational capabilities.

B. Performance Analysis of Fault Detection using On-Chip PULP AI Accelerator + AI Accelerator SoC

To execute sensor fault detection with a very high accuracy model, like [16], we use an external AI Accelerator SoC. For the performance analysis, we use NVIDIA Jetson Orion AGX 64GB AI Accelerator, with total throughput of 275 TOPS. This accelerator is interfaced to the flight controller SoC using the Ethernet interface (Figure 3).

TABLE III
PERFORMANCE OF EXECUTING FAULT DETECTION ALGORITHM ON EXTERNAL AI ACCELERATOR;
ALL LATENCY NUMBERS ARE IN MILLISECONDS

SoC	Datapath	Latency
Flight Controller SoC	Opentitan: AES-256 Encryption	1329.22
	Opentitan: SHA-256 Hash Generation	645.2
	Ethernet Latency Tx	0.286
AI Accelerator: Nvidia Jetson	SHA verification & AES Decryption	0.01
	GPU latency	0.046
	Hash generation and AES Encryption	0.01
	Ethernet Latency Rx	0.0000112
Flight Controller SoC	Opentitan: Hash verification	0.003
	Opentitan: AES-256 Decryption	0.005
	Total Latency	1974.78

Table III displays the computational latency of the datapath for executing the fault detection algorithm in [16]. With an aggregated data size of approximately 30 KBytes over a 120 msec window, it's notable that the ML algorithm's inference time is only 0.046 msec (compared to 746 msec using PULP). However, Opentitan's performance, operating at max 350 MHz, poses a significant bottleneck for this solution. Over 99% of computation time is spent on AES-256 encryption and SHA-256 hash generation for the 30 KBytes payload. From [17], it's evident that about 73% of the overall crypto-latency is dedicated to data movement to/from external RAM, managed by the Ibex processor within Opentitan.

Nvidia Jetson CPU supports ARM crypto extensions on Quad-A78 cores running at 1500MHz, offering minimal latency for AES-256 and SHA-256 crypto-extensions compared to Opentitan. Ethernet RGMII operates at 1000 Mbps speed. The output data-width of the fault detection model is 64 Bytes. Nvidia encrypts and generates hash of the output, transmitting it over the Ethernet interface back to the Flight controller SoC for hash verification and result decryption.

The bottleneck of Opentitan's high computational latency results in failure to complete fault detection execution using an external AI Accelerator within the 0.56 sec Lead time (as discussed in Section IV). To address this issue, potential solutions include: (i) Implementing a dedicated DMA mechanism for accessing data in external memory, or (ii) Modifying the CVA6 CPU core to support RISC-V Crypto extensions.

V. CONCLUSION

Key findings of this study include: (i) Gyroscope sensor Lead Time is notably inferior to other drone sensors, (ii) Ensuring confidentiality, integrity, and reliability for flight logs transmitted over non-secure interfaces to external AI Accelerators is imperative, and (iii) The prolonged computational latency of Opentitan's crypto-accelerator impedes fault detection execution within the critical 0.56-second lead time. For future work, we propose to integrate DMA into Opentitan or enable RISC-V crypto-extension for CVA6 CPU core, to reduce the computational latency of the fault detection using external AI Acclerator.

REFERENCES

[1] PX4 Autopilot: Parameter Reference. URL https://docs.px4.io/main/en/advanced_config/parameter_reference.html.

[2] S. Albawi et al. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[3] M. Andreoni Lopez et al. Towards secure wireless mesh networks for UAV swarm connectivity: Current threats, research, and opportunities. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 319–326. IEEE, 2021.

[4] O. Asghari et al. Lead Time Analysis for UAVs' Failure Prediction in U-space. In *2023 IEEE 28th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 123–133, 2023.

[5] N. Bruschi et al. GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors. IEEE, Oct. 2021.

[6] M. Ciani et al. Cyber Security Aboard Micro Aerial Vehicles: An Opentitan-Based Visual Communication Use Case. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2023.

[7] E. D'Amato et al. UAV Sensor FDI in Duplex Attitude Estimation Architectures Using a Set-Based Approach. *IEEE Transactions on Instrumentation and Measurement*, PP:1–11, 06 2018.

[8] J. Fu et al. A hybrid CNN-LSTM model based actuator fault diagnosis for six-rotor UAVs. pages 410–414, 2019.

[9] A. Garofalo et al. PULP-NN: Accelerating Quantized Neural Networks on Parallel Ultra-Low-Power RISC-V processors. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2164):20190155, Dec. 2019.

[10] D. Guo et al. Multisensor Data-Fusion-Based Approach to Airspeed Measurement Fault Detection for Unmanned Aerial Vehicles. *IEEE Transactions on Instrumentation and Measurement*, PP:1–11, 12 2017.

[11] L. Liu et al. Fault Detection and Isolation Based on UKFs for a Novel Ducted fan UAV. pages 212–218, 10 2016.

[12] P. Moosbrugger et al. R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems. *Formal Methods in System Design*, 51, 08 2017.

[13] A. Pullini et al. uDMA: An autonomous I/O subsystem for IoT endnodes. In *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 1–8, 2017.

[14] H. Sak et al. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 338–342, 01 2014.

[15] D. A. Santos et al. Enhancing Fault Awareness and Reliability of a Fault-Tolerant RISC-V System-on-Chip. *Electronics*, 12(12), 2023.

[16] L. K. Shar et al. DronLomaly: Runtime Detection of Anomalous Drone Behaviors via Log Analysis and Deep Learning. In *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, pages 119–128, 12 2022.

[17] L. Valente et al. A Heterogeneous RISC-V Based SoC for Secure Nano-UAV Navigation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5):2266–2279, May 2024.

[18] B. Wang et al. Multivariate Regression Based Fault Detection and Recovery of UAV Flight Data. *IEEE Transactions on Instrumentation and Measurement*, PP:1–1, 08 2019.

[19] J. You et al. An Adaptable UAV Sensor Data Anomaly Detection Method Based on TCN Model Transferring. pages 73–76, 05 2022.

[20] F. Zaruba et al. The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(11):2629–2640, 2019.