PROJECT SHIELD: SMART HARDWARE INSPECTION FOR EARLY LATENCY DETECTION

STUDENTS: SPARSH DADHICH, THOMAS GLASS, JOCELYN HE, SAM KOROSTOV, HOUSER ZHANG

Problem Statement

- Sensors are used in many industries, including aerospace, robotics, industrial automation, and healthcare.
- As systems become more autonomous, reliable sensor data becomes increasingly important.
- A single sensor failure can lead to serious consequences in safety-critical systems.
- **Project SHIELD aims to monitor sensor behavior and predict failures early to reduce risk.**

Requirements

Criteria	Goal
Sensor modalities	8 supported sensor types
Sensor identity	Unique {sensor_type, sensor_unit_id} per run
Sampling architecture	Fast / medium / slow tiered sampling
Timestamping	≤ 1 ms resolution
Storage	Binary logs + JSON metadata
Firmware pipeline	FreeRTOS buffering + async SD writing
Dataset output	CSV files for visualization and ML
Final ML target	Predict failure ≥ 20 s before functional loss

System Diagram

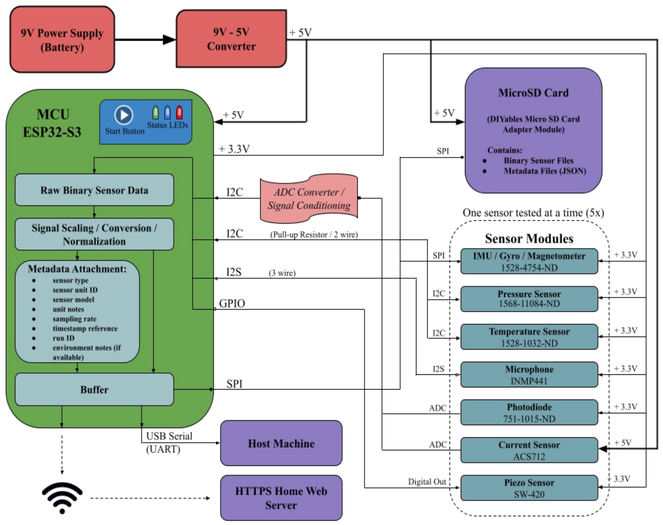

Figure 1: Project SHIELD system architecture and sensor data flow.

Firmware & Software Design

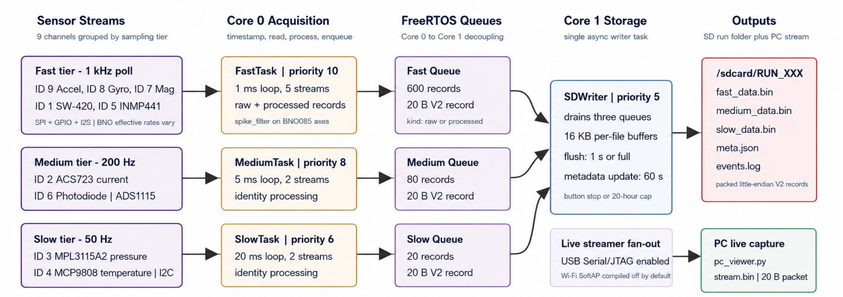

Figure 2: Dual-core FreeRTOS pipeline

- Core 0 acquires sensor data across three sampling tiers and enqueues records; Core 1 asynchronously writes to MicroSD while streaming live to PC

Hardware Design

8 Sensors type supported

Custom PCB DAQ board

Dual-core MCU ESP32-S3

Faraday Cage EMI shielding

- Custom DAQ PCB replaced the breadboard prototype to reduce wiring noise, stabilize power delivery, and keep the hardware setup consistent across long-duration data runs
- Faraday cage setup to reduce external electromagnetic interference during testing

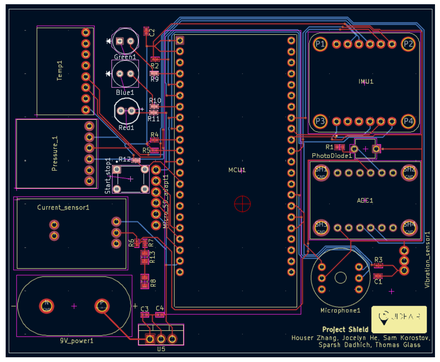

Figure 3: Our PCB layout

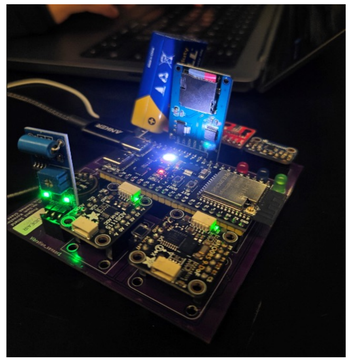

Figure 4: Our DAQ Board

Fault Injection

- **Software faults:** bias, drift, burst noise, dropouts, stuck-at, clipping, latency
- **Hardware faults:** power variation, EMI/analog noise, bus disturbance
- **Goal:** compare fault behavior against nominal baseline data
- **Output:** labeled dataset for fault detection and ML training

Machine Learning Pipeline Design

- Raw sensor telemetry is segmented into **fixed-length windows** and labeled using synthetically injected fault scenarios with randomized onsets and severities
- Each window is characterized by a feature vector combining **time-domain statistics**, **frequency-domain spectral features, and wavelet-domain coefficients**
- **Classification models** are trained on these features to distinguish healthy sensor behavior from fault conditions

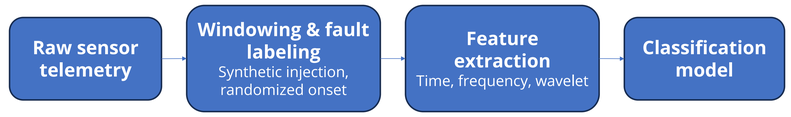

Figure 5: Machine Learning Pipeline

Result

- Breadboard prototype **100 hours of nominal data** successfully collected
- PCB prototype **100 hours of nominal data** successfully collected
- Spike-like artifacts were reduced after firmware cleanup and real-time processing.
- **Raw + processed data** saved simultaneously
- Live viewer shows raw and processed signals together

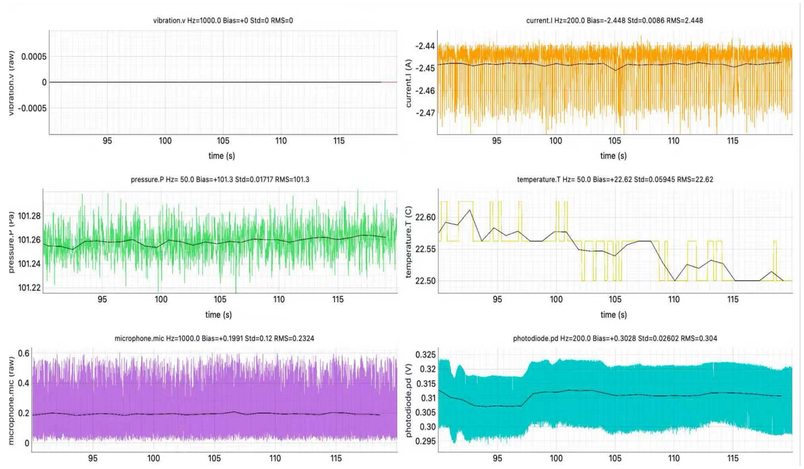

Figure 6: Clean data example for 6 sensors

Future Work & Acknowledgment

- Include AOA (Angle of Attack) sensor, proximity sensor, humidity sensor
- Add pressure in hardware fault injection
- Continuous run for 1 week on DAQ system
- Continue working on machine learning model training
- We would like to thank our industry and faculty mentors for their guidance throughout the project, as well as our TA, Zach Hao for his help and support.

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

INDUSTRY ADVISOR: ALPEREN CUCI

FACULTY ADVISOR: SEP MAKHSOUS

SPONSOR: GUIDE AIR LABS