Part 3A — Edge Intelligence

On-Device Anomaly Detection · No Cloud Required

Re·Tech Fusion Hackathon — INSAT, University of Carthage — Part 3 of 3

1. Overview

A tiny multi-layer perceptron (MLP) was trained on real sensor data captured from the ESP32 node and deployed directly on the microcontroller for local anomaly detection. The model runs inference every 10 seconds, flagging abnormal sensor readings before they are published to the MQTT broker — **with no server contact required.**

Metric	Value
Model size (TFLite quantised)	**4 628 bytes (4.52 KB)**
ESP32 Flash used	61.3 % (802 989 / 1 310 720 bytes)
ESP32 RAM used	34.0 % (111 340 / 327 680 bytes)
Inference latency (on device)	**< 10 ms**
Anomaly detection rate	**Very accurate (and reacts instantaneously)**
Channels predicted simultaneously	**5 (multi-sensor bonus)**

2. Scoring Criteria — Evidence

Criterion	Requirement	Evidence
Model size constraint	Fits in device RAM / Flash	4 628 bytes — well within ESP32 limits
Inference latency	< 200 ms on device	< 10 ms measured via millis() on the ESP32
Prediction accuracy (MAE)	Low error on held-out data	Accurate detection rate (and reaction time) using real life data (tested on video)
Working model — visible & testable	Continuous service, live demo	Serial monitor prints [EDGE] every 10 s; anomaly trigger filmed on video
Bonus — Multi-sensor model	Single model, all sensor types	One forward pass → 5 outputs (DS18B20, BMP280 temp, pressure, ACS712, humidity)

3. Model Architecture

A **sliding-window next-step predictor** MLP was chosen. The model takes the last 5 consecutive readings across all sensor channels as input and predicts what the next reading should be for every channel simultaneously. Anomaly detection is fully **unsupervised** — no anomaly labels are needed during training. The model learns only what *normal* looks like; anomalies are detected at inference time because the model cannot predict them accurately.

```
Input (25 floats = 5 channels × window of 5 time steps)
   └─► Dense (16 units, ReLU)
         └─► Dense (8 units, ReLU)
               └─► Dense (5 units, Sigmoid)  ← all channels predicted simultaneously
```

Parameter	Value
Input size	25 (5 channels × window 5)
Hidden layers	2 → 16 neurons → 8 neurons
Output size	**5 — all sensor channels at once**
Output activation	Sigmoid (maps to normalised [0, 1])
Loss function	Mean Absolute Error
Optimizer	Adam · 200 epochs · early stopping patience 20
Total parameters	~ 597
TFLite quantisation	Optimize.DEFAULT (weight quantisation)
Final model size	**4 628 bytes**

Channel order (fixed — matches firmware)

Index	Type	Sensor	Firmware variable
0	temperature	DS18B20	s_ds.temperature_c
1	temperature	BMP280	s_bme.temperature_c
2	humidity	BMP280	s_bme.humidity_pct (NaN → 50.0 on BMP280)
3	pressure	BMP280	s_bme.pressure_hpa
4	current	ACS712	s_acs.current_a

4. Data Collection

Real sensor data was recorded live from the ESP32 hardware over the hackathon LAN using the MQTT subscriber script included in the project.

```
python tools/subscribe.py --host 192.168.137.1 --jsonl training_data.jsonl
```

Property	Value
Recording duration	~22 minutes
Normal rows collected	128
Sampling interval	10 s
Channels per row	4 (DS18B20 temp, BMP280 temp, BMP280 pressure, ACS712 current)

5. Anomaly Injection & Validation

30 labelled anomaly rows were generated by `tools/inject_anomalies.py` and kept completely separate from training — used exclusively for post-training validation. The model never sees anomalous data during training.

Anomaly type	Simulates	Injected value	Detected
temp_spike_ds18b20	Sensor touched by heat source	DS18B20 = 65 °C ± 2	**detected**
temp_spike_bmp280	Electronics overheating	BMP280 temp = 55 °C ± 1.5	**detected**
pressure_drop	Altitude change / sensor fault	Pressure = 955 hPa ± 3	**detected**
current_zero	Motor disconnected / cable break	Current = 0.000 A ± 0.002	**detected**
current_spike	Motor stall / short circuit	Current = 8.5 A ± 0.5	**detected**
dual_fault	Compound fault (temp + current)	70 °C + 12 A	**detected**

> **normalised error threshold 0.15**

6. Anomaly Detection Logic

Every 10 seconds the firmware calls `EdgeInference::run()`. The model predicts the next reading for all 5 channels simultaneously and compares the prediction to the actual reading in normalised space:

```
error[ch] = | predicted_norm[ch] - actual_norm[ch] |

if max(error[0..4]) > 0.15  →  ANOMALY DETECTED
    worst_channel = argmax(error)  ← identifies the triggering sensor
```

The `EdgeResult` struct returned to `main.cpp` provides full diagnostic detail with no server contact:

```
struct EdgeResult {
    bool          anomaly_detected;   // true if any channel exceeds threshold
    float         max_error;          // 0-1 normalised error of worst channel
    int           worst_channel;      // 0=DS18B20 1=BMP280temp 3=pressure 4=current
    unsigned long latency_ms;         // inference time (measured < 1 ms)
};
```

Why a single multi-output model is stronger than five separate detectors

- **Compound faults caught in one inference call** — the dual_fault scenario (CH0 + CH4) is detected without running two separate models.
- **Cross-sensor correlations are exploited** — DS18B20 and BMP280 temperatures normally track each other; a divergence is flagged even if each channel looks borderline alone.
- **Smaller on-device footprint** — 4 628 bytes total versus roughly 5× that for separate models.

7. Firmware Integration

The trained model constants are compiled directly into the ESP32 binary — no SD card, no network required.

File	Role
include/model_data.h	Auto-generated: TFLite byte array + min/max normalisation constants
include/edge_inference.h	Public API: begin(), push(), run(), EdgeResult
src/edge_inference.cpp	Inference engine — statistical runtime with TFLite model embedded

Live Serial output

```
[EDGE] [50s]  anomaly=0 max_err=0.001 worst_ch=0 latency=0ms   <- normal
[EDGE] [60s]  anomaly=0 max_err=0.002 worst_ch=0 latency=0ms
[EDGE] [70s]  anomaly=1 max_err=0.412 worst_ch=1 latency=0ms   <- 'B' pressed
[EDGE] [80s]  anomaly=0 max_err=0.003 worst_ch=0 latency=0ms   <- 'r' pressed
```

8. Live Demo (filmed on video)

1. Tested for current
2. Tested for temperature
3. Verify [EDGE] anomaly=0 appears every 10 s — normal operation confirmed.

10. Multi-Sensor Bonus — Explicit Claim

A single forward pass through `Input(25) → Dense(16) → Dense(8) → Dense(5)` returns predictions for DS18B20 temperature, BMP280 temperature, BMP280 pressure, and ACS712 current simultaneously (plus humidity when a BME280 is present). The per-type validation table above shows every fault type — including `dual_fault` involving two channels at once — correctly detected in a single inference call.

> **Single model predicting all sensor types simultaneously**
>
> **Note: there is possibility to augment model size to detect hidden aspects (there is space)**