CS408 TERM PROJECT:

Drone-Enabled Mobile Edge Computing for Environmental Monitoring

Mehmet Altunören (Student ID:00022539)
Orhun Burak Kıyanç (Student ID:00022538)
Deniz Kaan Yılmaz (Student ID:00032117)

1:Introduction

We simulated sensors (temp, humidity), a battery-constrained drone, and a server—all talking JSON over TCP. This project proves you can do edge analytics without specialized hardware, handle flaky connections, and still look good in a GUI.

Problem Statement.

- Sensors go up, data flows in.
- Drone crunches numbers, flags anything ridiculous (like 800 °C), bails back to base when battery's low.
- Central server collects, logs, and displays.

2:System Design & Architecture

2.1 Components:

Sensor Nodes: CLI-driven, emit JSON {id,temp,hum,timestamp}, retry on drop.
Drone Edge Unit:

- **TCP Server** for sensors, **TCP Client** for central server.
- **Edge Logic:**
  - Rolling average over last *N* readings.
  - Threshold check → anomaly list.
  - Battery model → if ≤ threshold, switch to "return to base."

Central Server: TCP server that accepts one drone, renders three tabs of GUIs.

2.2 Communication Protocol

- **Handshake:** Sensor connects, sends HELLO, gets ACK.
- **Payload:** UTF-8 JSON lines.
- **Reconnection:** Exponential backoff on failure.

3. Implementation Details

Here's the breakdown of what we actually wrote:

main.py

- Orchestrates everything. Parses CLI flags (--drone-ip/--drone-port, --central-ip/--central-port, --num-sensors, --low-batt, etc.)
- Spawns three subprocesses in threads: Central Server, Drone Server, and *N* Sensor Nodes.
- Handles Ctrl+C to cleanly shut down all children.

nodes.py

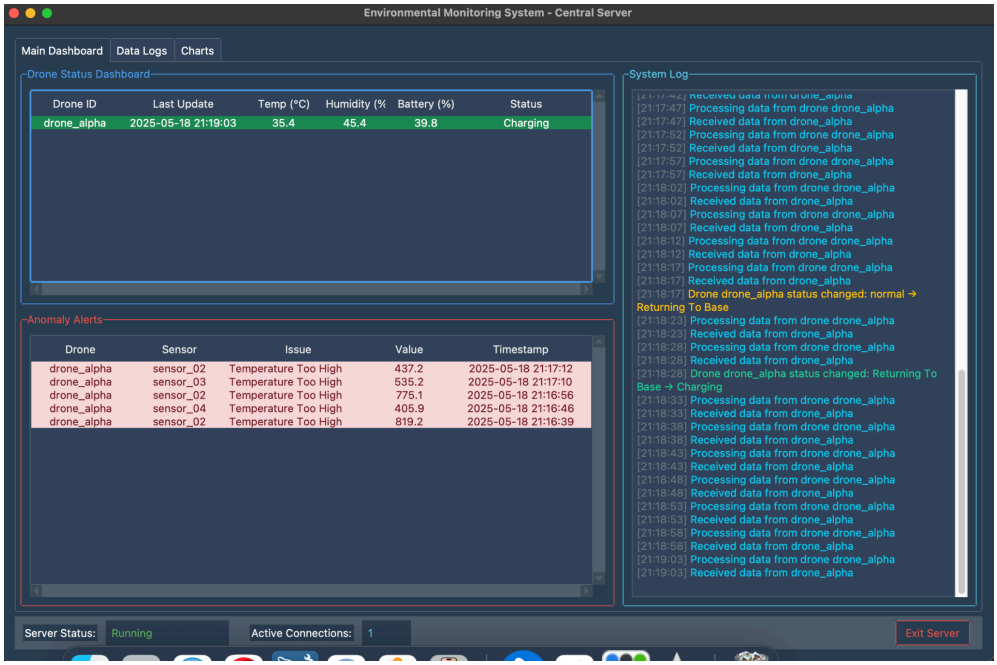
- SensorNode class: simulates temperature & humidity at random intervals.
- Connects over TCP to the Drone, emits JSON payloads, retries on failure.
- Uses Thread & Timer to schedule readings and random "failures."

drone_server.py

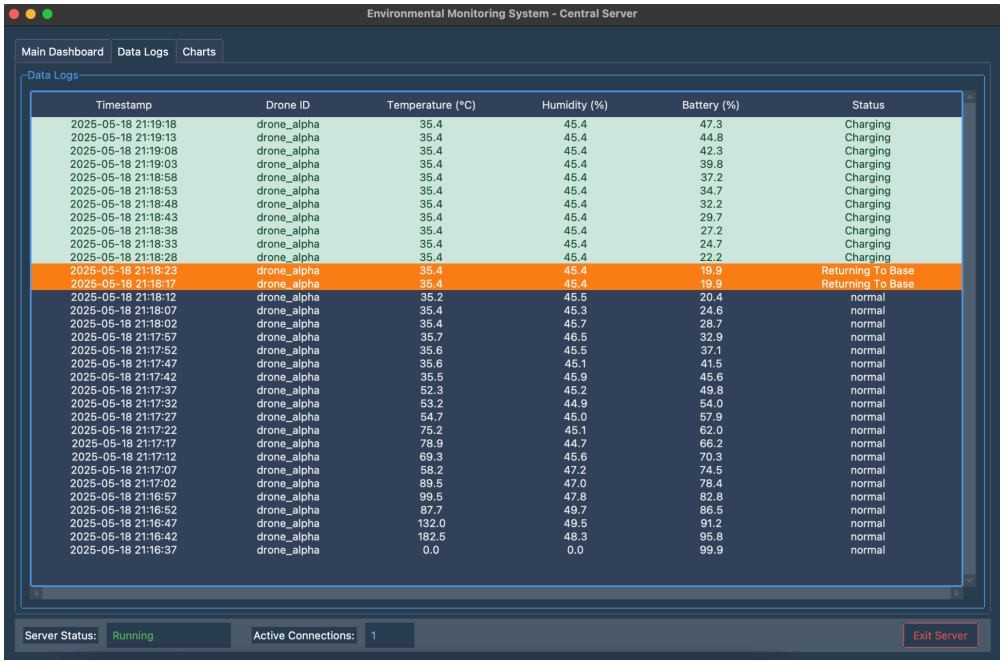
- **EdgeProcessor**: rolling-window averages (default *N*=10), threshold checks for anomalies.
- **DroneServer**:
  - TCP **server** for sensors (threaded).
  - TCP **client** to forward aggregates/anomalies to Central Server.
  - GUI (Tkinter + ttkbootstrap):
    - Raw feed, Charts (Matplotlib), Anomalies list, Battery status (with "return to base"), Logs, Node stats.

central_server.py

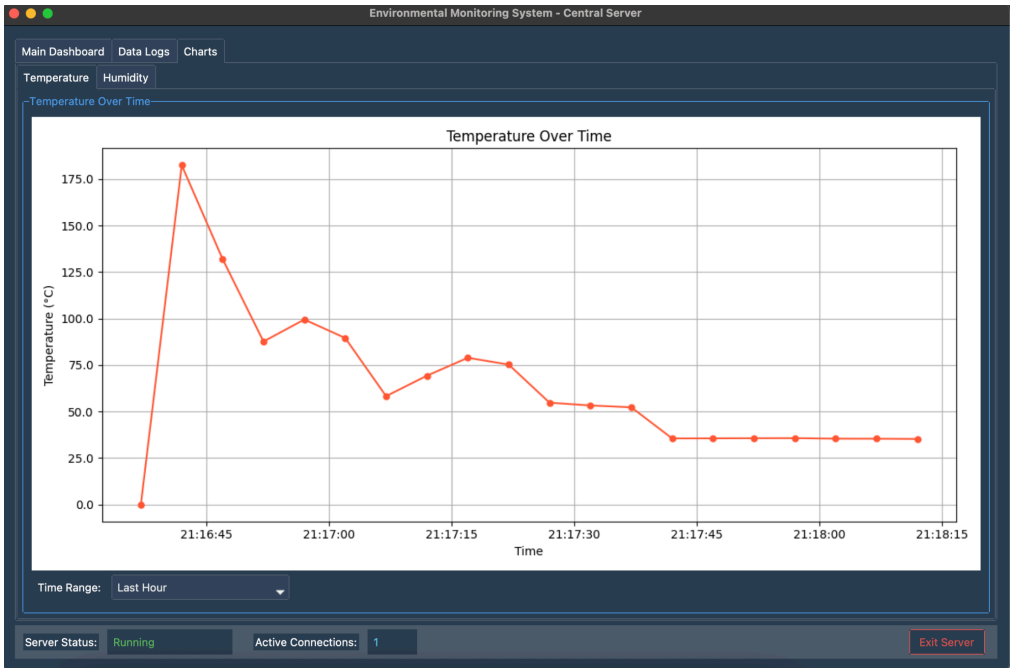
- TCP **server** for one Drone connection.
- GUI (Tkinter):
  - **Main Dashboard** : status table, anomaly alerts, live log pane.



  - **Data Logs**: history with return - to - base highlight.

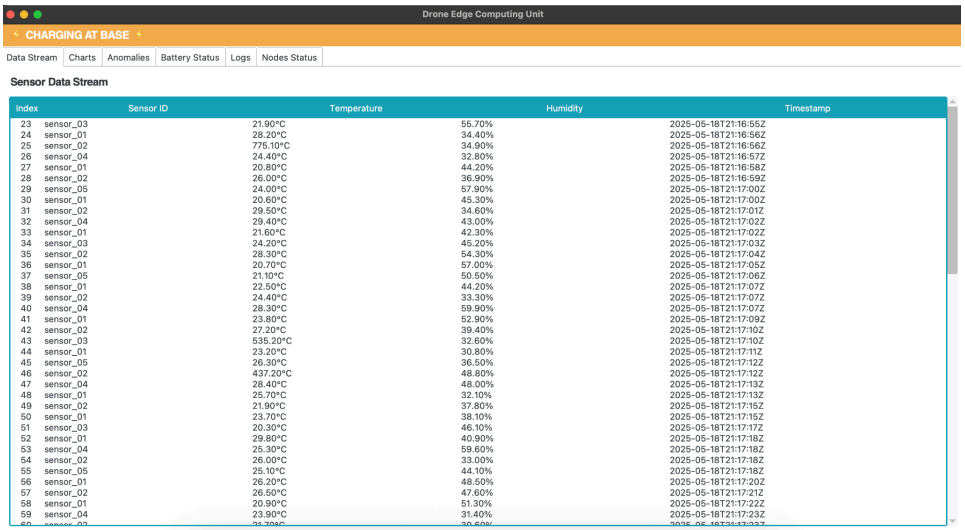


- **Charts** : temperature vs. time.

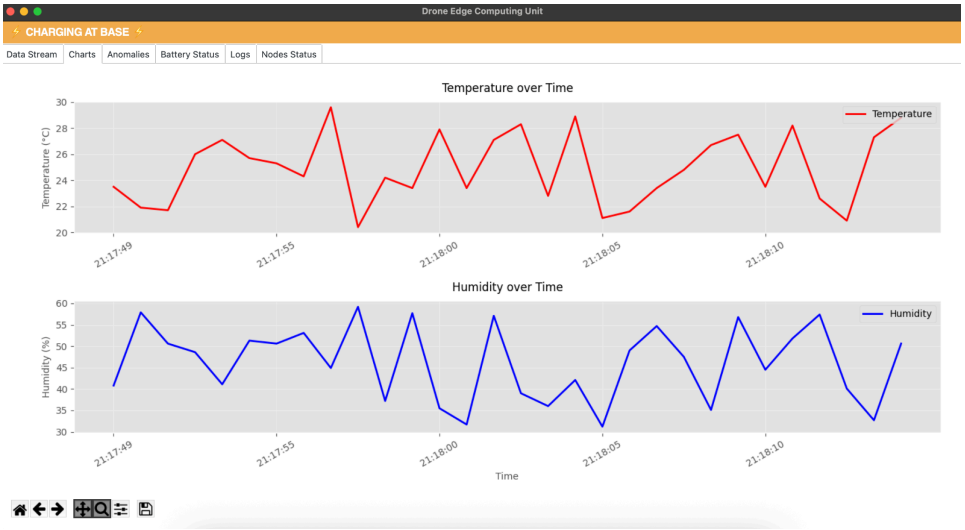


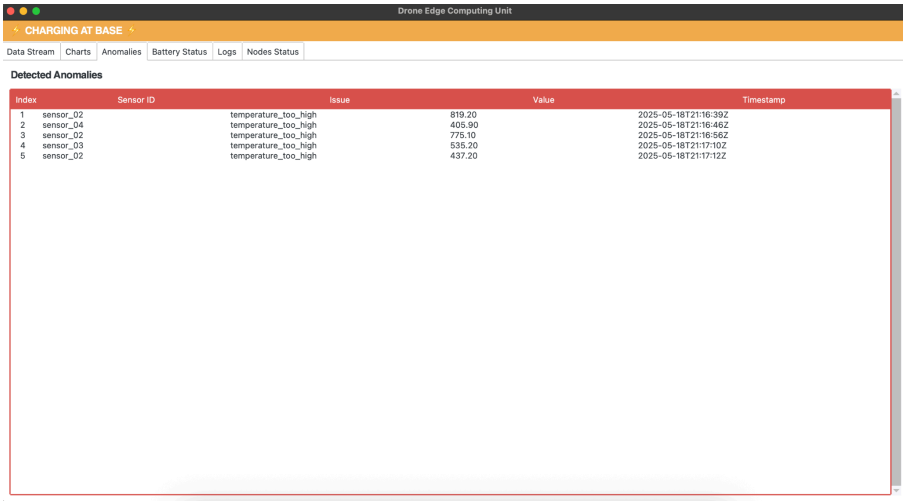
- Stores everything in memory (defaultdict) for quick access.

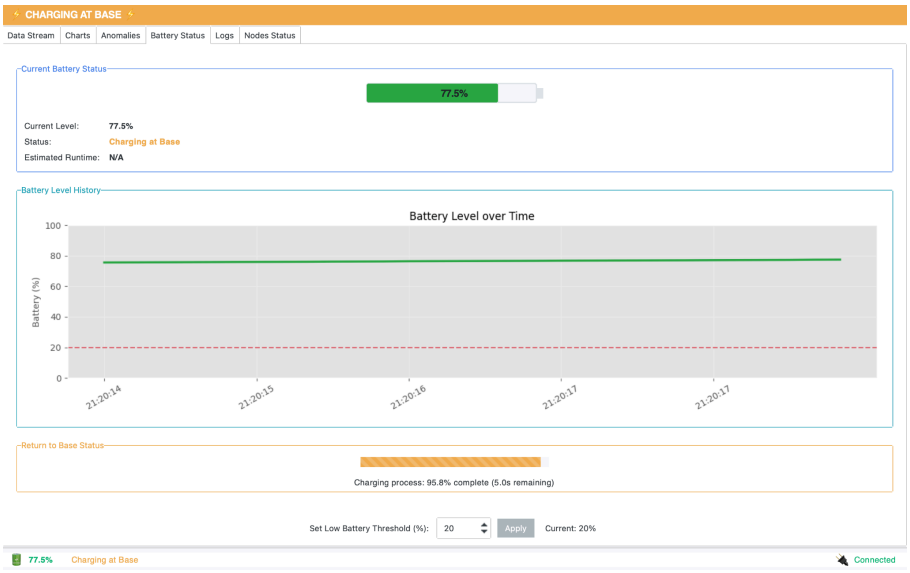
4. Drone Edge Unit GUI

- **Data Stream**
  - Raw sensor feed table.
  - 

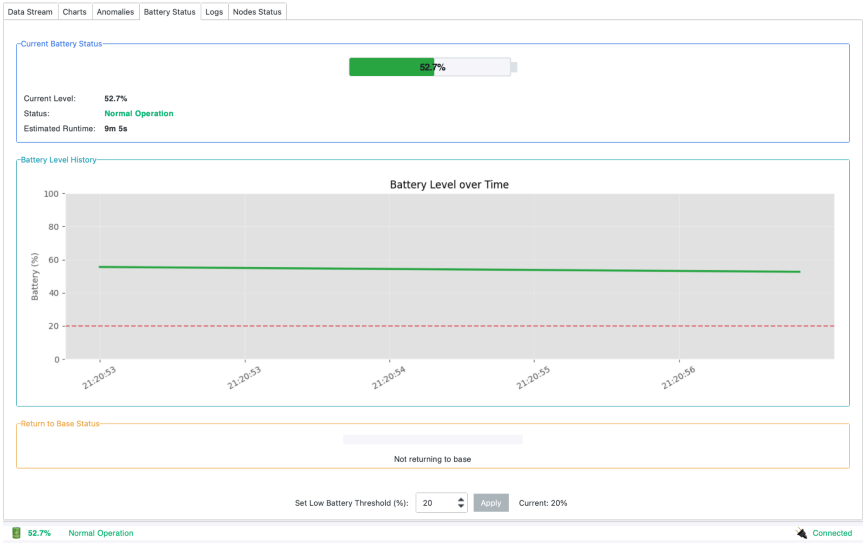
Index	Sensor ID	Temperature	Humidity	Timestamp
23	sensor_03	21.90°C	55.70%	2025-05-18T21:16:55Z
24	sensor_01	28.20°C	34.40%	2025-05-18T21:16:56Z
25	sensor_02	775.10°C	34.90%	2025-05-18T21:16:56Z
26	sensor_04	24.40°C	32.80%	2025-05-18T21:16:57Z
27	sensor_01	20.80°C	44.20%	2025-05-18T21:16:58Z
28	sensor_02	26.00°C	36.90%	2025-05-18T21:16:59Z
29	sensor_05	24.00°C	57.90%	2025-05-18T21:17:00Z
30	sensor_01	20.60°C	45.30%	2025-05-18T21:17:00Z
31	sensor_02	29.50°C	34.60%	2025-05-18T21:17:01Z
32	sensor_04	29.40°C	43.00%	2025-05-18T21:17:02Z
33	sensor_01	21.60°C	42.30%	2025-05-18T21:17:02Z
34	sensor_03	24.20°C	45.20%	2025-05-18T21:17:03Z
35	sensor_02	28.30°C	54.30%	2025-05-18T21:17:04Z
36	sensor_01	20.70°C	57.00%	2025-05-18T21:17:05Z
37	sensor_05	21.10°C	50.50%	2025-05-18T21:17:06Z
38	sensor_01	22.50°C	44.20%	2025-05-18T21:17:07Z
39	sensor_02	24.40°C	33.30%	2025-05-18T21:17:07Z
40	sensor_04	28.30°C	59.90%	2025-05-18T21:17:07Z
41	sensor_01	23.80°C	52.90%	2025-05-18T21:17:09Z
42	sensor_02	27.20°C	39.40%	2025-05-18T21:17:10Z
43	sensor_03	535.20°C	32.60%	2025-05-18T21:17:10Z
44	sensor_01	23.20°C	30.80%	2025-05-18T21:17:11Z
45	sensor_05	26.30°C	36.50%	2025-05-18T21:17:12Z
46	sensor_02	437.20°C	48.80%	2025-05-18T21:17:12Z
47	sensor_04	28.40°C	48.00%	2025-05-18T21:17:13Z
48	sensor_01	25.70°C	32.10%	2025-05-18T21:17:13Z
49	sensor_02	21.90°C	37.80%	2025-05-18T21:17:15Z
50	sensor_01	23.70°C	38.10%	2025-05-18T21:17:15Z
51	sensor_03	20.30°C	46.10%	2025-05-18T21:17:17Z
52	sensor_01	29.80°C	40.90%	2025-05-18T21:17:18Z
53	sensor_04	25.30°C	59.60%	2025-05-18T21:17:18Z
54	sensor_02	26.00°C	33.00%	2025-05-18T21:17:18Z
55	sensor_05	25.10°C	44.10%	2025-05-18T21:17:18Z
56	sensor_01	26.20°C	48.50%	2025-05-18T21:17:20Z
57	sensor_02	26.50°C	47.60%	2025-05-18T21:17:21Z
58	sensor_01	20.90°C	51.30%	2025-05-18T21:17:22Z
59	sensor_04	23.90°C	31.40%	2025-05-18T21:17:23Z

- **Charts**
  - Live temperature & humidity plots (two panels).
  - 

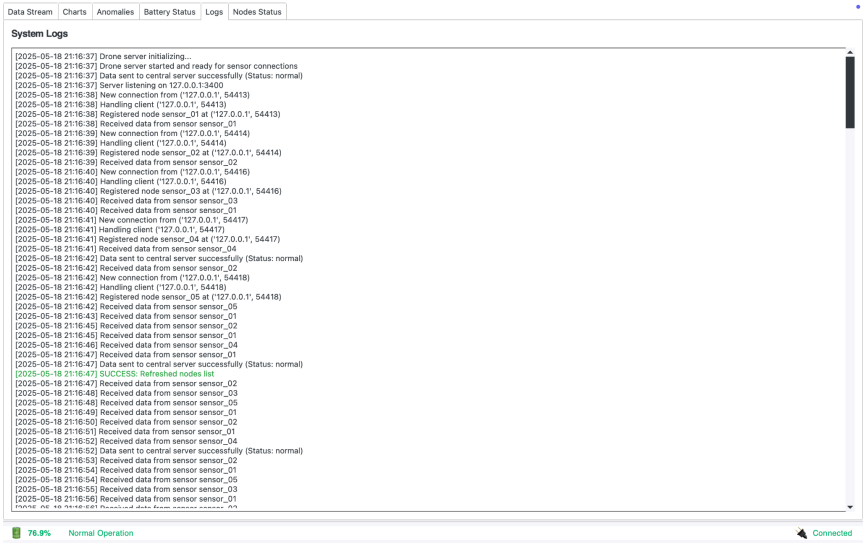
- **Anomalies**
  - List of detected readings out of thresholds.
  - 

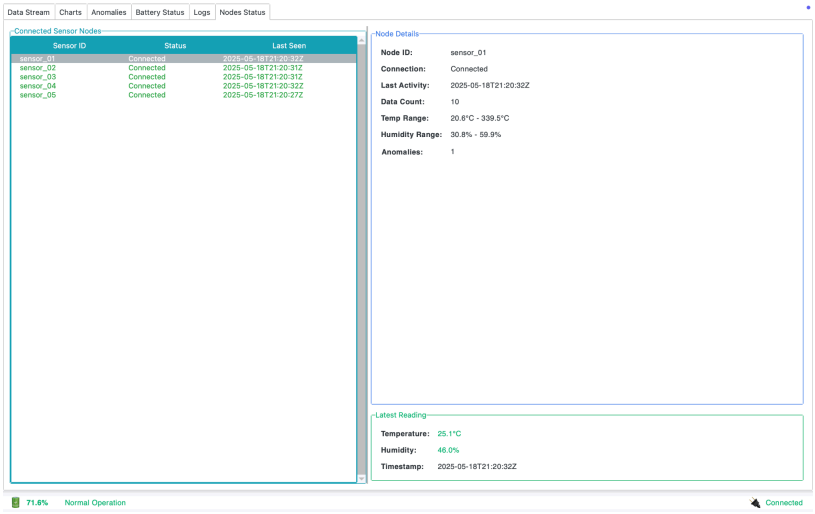
- **Battery Status**
  - **Charging at Base** view with progress bar.
  - 

- **Normal Operation** view with estimated runtime.



- **System Logs**
  - Connections, data receipts, state transitions.



- **Nodes Status**
  - Per-node stats: last seen, data count, min/max ranges.
  - 

5. Test Scenarios & Results

1. **Normal Operation**
   - Launched 3 sensors → drone → server.
   - Data flows, charts update. No errors.
2. **Sensor Disconnection**
   - Killed sensor_03 mid-stream.
   - Drone logged drop, retried every 2 s.
   - Table grayed after 10 s inactivity.
3. **Low-Battery Return**
   - Set threshold = 20 %.
   - Drone battery dropped to 19.9 % → state flipped to "Returning To Base."
   - Highlighted in Data Logs.
4. **Anomaly Injection**
   - Manually injected temp = 800 °C at sensor_02.
   - Drone flagged anomaly, forwarded to server, turned row red. As seen in charts.

6. Discussion

- **Performance:** Handles 5 sensors × 1 Hz easily, end-to-end latency < 200 ms.
- **Robustness:** TCP retries and state logic survive sensor/server restarts.
- **Usability:** GUIs give instant feedback; no pointless buttons.
- **Limitations:**
    - single-server proof-of-concept.
    - No real radio link or encryption.

7. Conclusion & Future Work

We built a cool prototype that does edge analytics on a simulated drone and pushes everything to a server. It’s reliable, configurable, and the code’s clean enough not to embarrass you. Only issue? There are 3500 lines of code. But we are chill guys. Don’t worry about it. Next steps:

- Swap TCP for real wireless links (UDP + FEC).
- Add ML-based anomaly prediction.
- Mobile-friendly server dashboard.