DATABASE MOD B PROJECT

Smart Environmental IoT Data Pipeline

But why?

IoT devices generate massive amounts of sensor data

Storing and processing this data efficiently is a challenge

Different data types need different database solutions

Goal: Build a modular system to receive, process, and store IoT sensor data in real-time

Objectives

- Simulate environmental sensor data (temperature, humidity, air quality, network)
- Route data through an **MQTT broker**
- Process and store based on topic:
  - SQLite for structured data
  - MongoDB for semi-structured data
  - Neo4j for device relationships

System Architecture

Components:

- Sensor Simulator (Python)
- MQTT Broker (Mosquitto)
- Data Router (Python Subscriber)
- Databases: SQLite, MongoDB, Neo4j
- Docker Compose manages the whole system

All components are containerized and networked via Docker

Data Flow

[Sensor Simulator]
    └──→ Publishes to MQTT Topics (e.g., env/temperature)

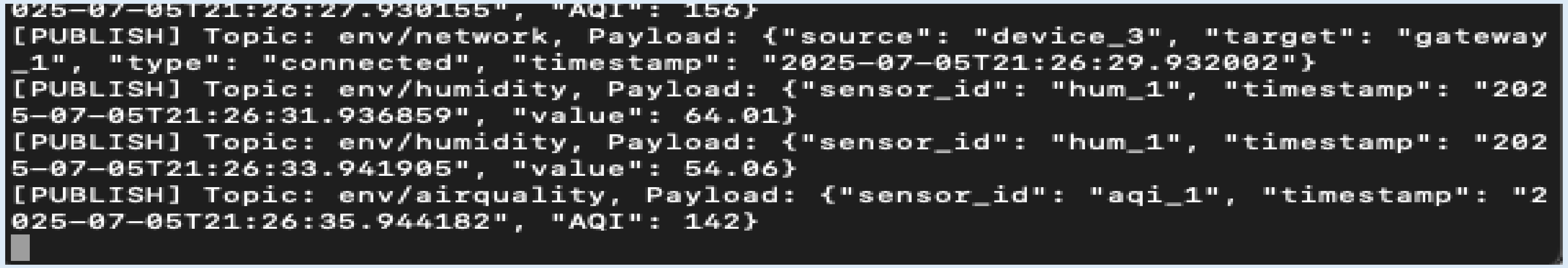
[MQTT Broker]
    └──→ Forwards messages to Listener

[Data Router (Python)]
    └──→ Analyzes topic & routes to correct DB:
        SQLite (temperature/humidity)
        MongoDB (air quality)
        Neo4j (network graph)

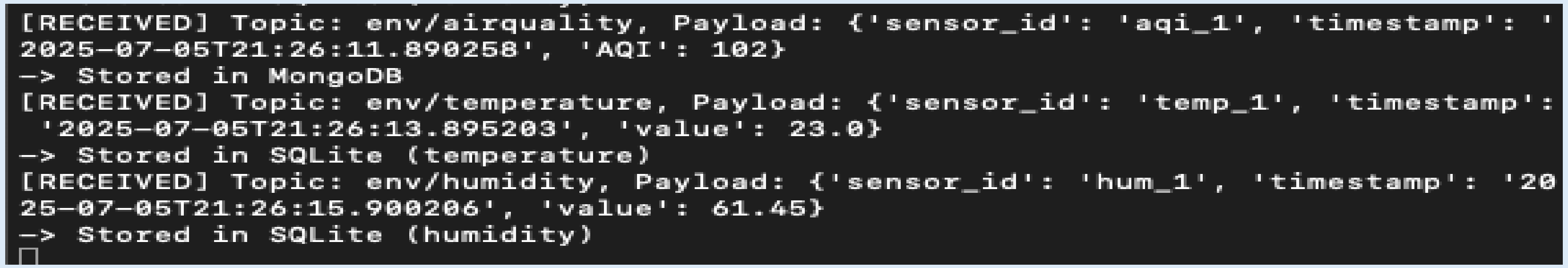
Database used

Database	Purpose
SQLite	Stores structured sensor values
MongoDB	Stores air quality in JSON format
Neo4j	Stores network/device relationships

Live Demo/Outputs



```
025-07-05T21:26:27.930155", "AQI": 156}
[PUBLISH] Topic: env/network, Payload: {"source": "device_3", "target": "gateway
_1", "type": "connected", "timestamp": "2025-07-05T21:26:29.932002"}
[PUBLISH] Topic: env/humidity, Payload: {"sensor_id": "hum_1", "timestamp": "202
5-07-05T21:26:31.936859", "value": 64.01}
[PUBLISH] Topic: env/humidity, Payload: {"sensor_id": "hum_1", "timestamp": "202
5-07-05T21:26:33.941905", "value": 54.06}
[PUBLISH] Topic: env/airquality, Payload: {"sensor_id": "aqi_1", "timestamp": "2
025-07-05T21:26:35.944182", "AQI": 142}
```



```
[RECEIVED] Topic: env/airquality, Payload: {'sensor_id': 'aqi_1', 'timestamp': '
2025-07-05T21:26:11.890258', 'AQI': 102}
-> Stored in MongoDB
[RECEIVED] Topic: env/temperature, Payload: {'sensor_id': 'temp_1', 'timestamp':
 '2025-07-05T21:26:13.895203', 'value': 23.0}
-> Stored in SQLite (temperature)
[RECEIVED] Topic: env/humidity, Payload: {'sensor_id': 'hum_1', 'timestamp': '20
25-07-05T21:26:15.900206', 'value': 61.45}
-> Stored in SQLite (humidity)
```

And why Docker?

- **Isolation**: Each service runs in its own container
- **Reproducibility**: Same setup on any machine
- **Orchestration**: One-command launch via docker compose

Technologies used

- Python
- Paho MQTT
- SQLite3
- MongoDB & PyMongo
- Neo4j & Py2Neo
- Eclipse Mosquitto
- Docker + Docker Compose

What I learned

- Setting up and using MQTT protocols
- Integrating Python with 3 different DBs
- Using Docker Compose to manage services
- Data routing based on message topics

So what's next?

- Add error handling and validation
- Replace SQLite with PostgreSQL
- Deploy to Raspberry Pi or cloud for real sensors