IoT & Control Component – 1st Integration

EMSIB – Energy Management System in Intelligent Buildings

Authors: Aday García & Cristian Costa

11/12/2025

Index

Introduction

- Our module is responsible for managing IoT devices and collecting sensor telemetry within the EMSIB system.
- This component interacts with building sensors, HVAC devices, and optimization modules to support energy efficiency actions.
- This component is essential for enabling real-time decision-making in the EMSIB system.

Module Purpose

- Main goals:
  - Handle real-time telemetry from building sensors
  - Send commands to IoT devices (e.g., HVAC, lights)
  - Apply optimization scenarios generated by higher layers
  - Act as the first layer of the EMSIB energy-management pipeline

Architecture Overview

Sensors → IoT & Control Component → Devices

↕

Optimization Module

- We implemented the Device & Sensor Layer
- Communication simulated using print/MQTT
- FastAPI server provides access to the module

Implemented Endpoints (Overview)

- List:
  - POST /iot/telemetry
  - POST /iot/device-control/{deviceId}/command
  - POST /iot/optimization/apply

Each endpoint supports one of the module’s core responsibilities.

Implemented using FastAPI with lightweight logic and in-memory storage.

Endpoint 1: Telemetry Ingestion

- Purpose:
  - Receive measurement packets from sensors.

- What it does:
  - Stores data
  - Prints received telemetry
  - Prepares data for analysis layer

```
@app.post ( "/iot/telemetry" )
def ingest_telemetry ( payload: dict ):
    telemetry_db.save ( payload )
```

Endpoint 2: Device Control

- Purpose:
  - Send commands to devices (e.g., HVAC).

Example:

Set HVAC temperature to 20°C.

- What it does:
  - Accepts command JSON
  - Simulates MQTT publish
  - Shows execution in console

Endpoint 3: Optimization Scenario

- Purpose:
  - Apply optimization actions created by the Business Logic Layer.

Example:

```
{
  "scenarioId": "opt01",
  "actions": [...]
}
```

- What it does:
  - Receives scenario
  - Stores it
  - Prepares execution logic

Component Features Summary

- Real telemetry ingestion
- Device control simulation
- Low-level connection for higher-level modules
- Fully functional API built with FastAPI
- Ready for integration with the rest of EMSIB

Integration Workflow

Sensor Data → Telemetry API → Storage

↓

Optimization Scenario → Apply API → Device Control

- All communication occurs via REST endpoints
- Data is processed inside simple classes
- No external database used (memory simulation)

Testing and Validation

- Tools used:
  - FastAPI interactive docs (/docs)
  - Manual JSON tests
  - Console execution traces
- Validation:
  - Telemetry saved correctly
  - Commands executed as expected
  - Optimization scenarios accepted

Conclusion

- Key messages:
  - IoT & Control Component successfully implemented
  - Covers all required first-integration functionality
  - Fully ready for next integration phase with optimization and analysis layers

Next Steps

- Connect to real MQTT broker
- Store telemetry in external database
- Implement real device drivers
- Support bidirectional communication

THANK YOU