Industrial Machinery Monitoring with IIoT
Meta 2 Project

Pedro Silva, Ramyad Raadi
uc2023235452@student.uc.pt, uc2023205634@student.uc.pt

University of Coimbra, Faculty of Science and Technologies
Bachelor of Data Science and Engineering, SRSA Course

May 2025

1 Introduction

This project simulates an Industrial Internet of Things (IIoT) scenario where industrial machinery is monitored in real-time. The system collects sensor data (RPM, oil pressure, coolant temperature, battery potential, and fuel consumption) from different machine models, processes it via MQTT, stores it in an InfluxDB database, and visualizes it on a Grafana dashboard. The goal is to ensure efficient machine management by detecting anomalies, issuing control commands, and triggering alerts when critical conditions appear.

2 System Implementation

2.1 Machine Simulation

Each machine was simulated using a Python script that generates sensor data that is according to predefined operational ranges. The script:

- Updates sensor values every `MACHINE_UPDATE_TIME` seconds.
- Publishes data to the MQTT topic: `v3/(GroupID)@ttn/devices/{machine_id}/up` in a JSON format compliant with TheThingsNetwork (TTN) standards.
- Subscribes to control (`push_actuator`) and alert (`push_alert`) topics to receive commands from the Data Manager Agent.

2.2 Data Manager Agent

This module acts as the intermediary between machines and the Machine Data Manager. Its functionalities include:

- Subscribing to machine data topics and extracting sensor values.
- Standardizing/destandardizing units (e.g., converting °F to °C, bar to psi) for consistency.
- Sending processed data to the Machine Data Manager via a custom MQTT topic: `{group_id}/internal/machine_data` and receiving control messages from via a custom MQTT topic: `{group_id}/internal/control_commands`
- Encoding control and alert messages into byte-structured commands and sending them commands/alerts to the machines.
- Forwarding shutdown commands from the Alert Manager via UDP.

2.3 Machine Data Manager

This component:

- Reads healthy operating ranges from `intervals.json`.
- Monitors sensor data and issues control commands if values exceed thresholds.
- Publishes alarms to the Data Manager Agent via MQTT when adjustments are needed.

2.4 Alert Manager

The Alert Manager evaluates machine health over time. In our implementation:

- A machine enters a `CRITICAL` state if it receives 5 control commands within 2 minutes.
- Upon critical status detection, a UDP message is sent to the Data Manager Agent to shut down the machine.

2.5 MQTT Debugger

A Python script logs all MQTT messages in the format: `[time]:[topic]:[message]` to assist in debugging and monitoring.

2.6 Storage and Dashboard

- InfluxDB Cloud stores sensor data, control/alert messages, and signal metrics.
- Grafana Cloud provides a complete dashboard that presents:
  - **Summarized metrics:** Displays key metrics and its healthy ranges.
  - **Detailed metrics:** Shows historical trends, healthy ranges, and signal quality.

3 Dashboard Explanation

The dashboard created to visualize the data is represented in Figure 1. There, we can check all the real-time metrics of the machine in study with gauge charts (upper charts). Each color represents its normal range (green), the cautious range (yellow), and the danger range (red). Also, the time series graphs (lower graphs) demonstrate not only the real-time values of each metric, but also the historical values, healthy operating intervals (same colors as the gauge charts range) and signal transmission information. This last type of time-series, in the bottom-right corner - combines the `RSSI` (yellow), `SNR` (green) and `Channel_RSSI` (blue) metrics into one time-serie graph.

Figure 1: Dashboard of an Industrial Machine

4 Conclusion

The project successfully demonstrated a full IoT pipeline, from data collection to visualization. The system effectively monitors machinery, enforces operational limits, and triggers alerts for critical conditions.

Acknowledgements

We would like to mention that AI [ChatGPT] was used to correct some bugs in the source code and to help us create the report.