

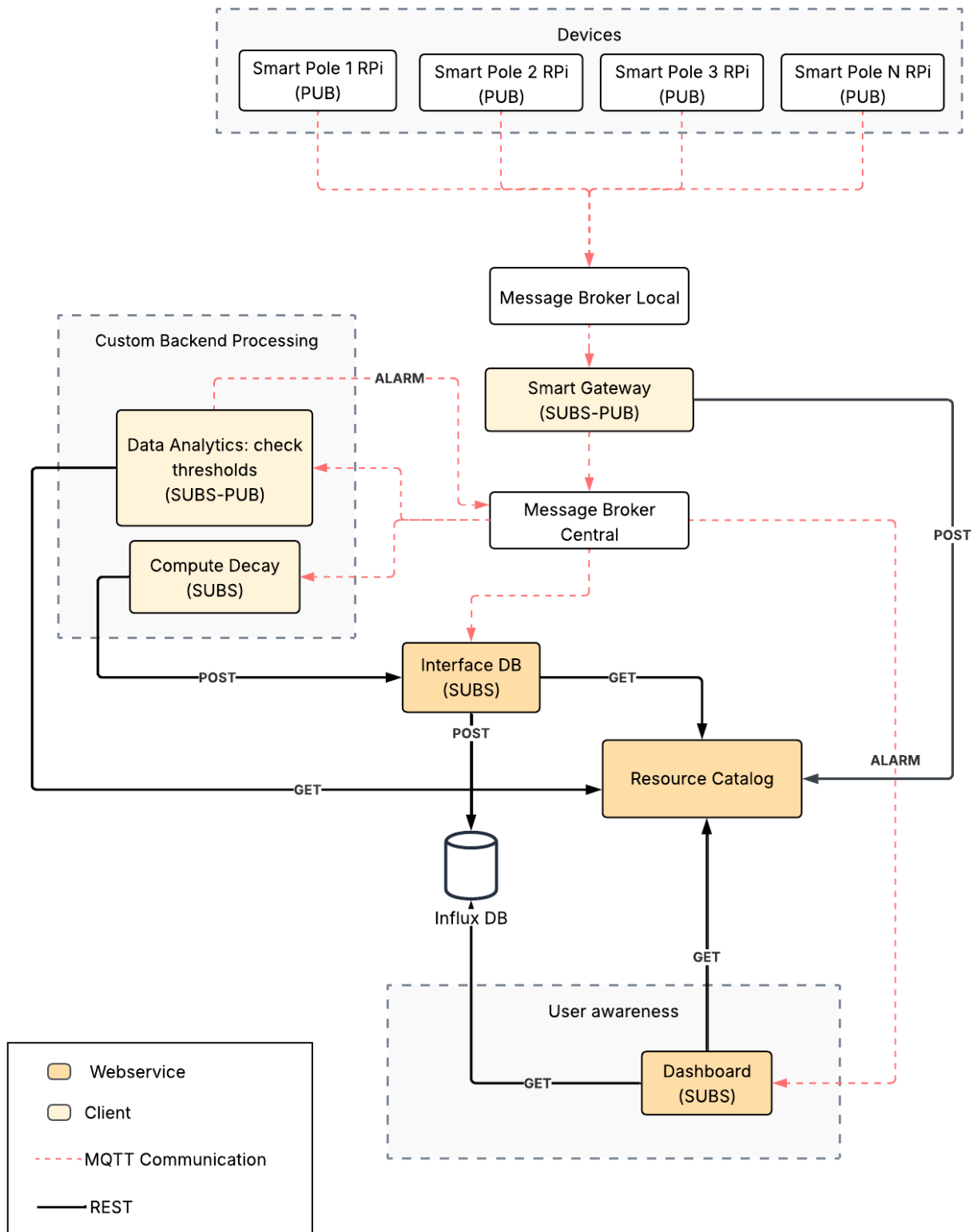
1 Name of Use Case

Name of the Use Case	Smart Poles Monitoring System
Version No.	V0.2
Submission Date	15/01/2026
Team Members (with student ids)	Tommaso Alesso (s337250), Chiara Difino (s343806), Natalia Mojica (s334735)

2 Scope and Objectives of Function

Scope and Objectives of Use Case	
Scope	The scope of our IoT application is to allow remote monitoring of the health of wooden utility poles
Objective(s)	<p>The objectives are:</p> <ul style="list-style-type: none">• Preventing telecommunication interruptions due to faulty wooden poles.• Avoiding landscape degradation and dangers due to fallen utility poles.• Cutting the economic costs of companies that own utility poles, allowing them to know their poles' conditions without needing physical presence.
Domain(s)	Smart Monitoring, Smart Industry, Smart Cities
Stakeholder(s)	Telecommunication companies, citizens, and public administration
Short description	<p>This IoT application sets out to monitor remotely the conditions of decay of wooden utility poles used in telecommunications. These poles may not be functional anymore, and when this happens, urgent intervention is needed. This comes at great cost for telecommunication companies because they need to constantly physically monitor their huge and vast network of poles. A remote application would allow them to cut monitoring costs and make interventions timelier, preventing inconveniences. Our solution will tackle this issue by weekly monitoring remotely the most important variables to understand wood decay (humidity, temperature, tilting). This information will be analysed through a decentralised system that will understand the quality of each wooden pole, checking its tilting, to make sure that it has not fallen, and the other variables to compute its rate of decay. The user will be informed through a dashboard, where urgent importance will be given to the fallen poles, and decayed poles will be highlighted.</p>

3 Diagram of Use Case



4 Complete description of the system

The proposed IoT platform for Smart Poles Monitoring follows the microservices design pattern. It also exploits two communication paradigms: i) publish/subscribe based on the MQTT protocol and ii) request/response based on REST Web Services. This architecture includes custom backend processing for decay computation, threshold checking and a user interface based on a dashboard.

In this context, nine main actors have been identified and introduced in the following:

- *The Message Broker* provides asynchronous communication based on the publish/subscribe approach. It exploits the MQTT protocol to act as the central communication bus between the data acquisition layer, the analytics layer, and the user interface. In particular, at the data acquisition layer, two brokers can be found, corresponding to two different MQTT connections: the first, connects each Raspberry board to its closest gateway, while the second connects all of the gateways to the rest of the system.
- *The Resource Catalog* works as a service and device registry system for all the actors in the system. It provides information about endpoints and configuration settings.
 - o The *Smart Gateway* registers itself and the connected poles here via *REST POST*.
 - o *Data Analytics* retrieves configuration settings (e.g., threshold values) from here via *REST GET*.
 - o *The Dashboard* retrieves system information via *REST GET* from here to correctly visualize resources.
 - o *Interface DB (Writer Service)* retrieves InfluxDB credentials and agent information via *REST GET*.
- *The Smart Poles* are edge devices located in the target area. They gather local sensor data and transmit it via a local MQTT instance ("MQTT Local"). These act as the primary data sources.
- *The Smart Gateway* acts as a bridge between the local field devices and the central infrastructure. It aggregates data from the Smart Poles and forwards it to the central system.
 - o Works as an *MQTT Subscriber* to listen for configuration and measurements from poles given by the *Local Message Broker*.
 - o It works as an *MQTT Publisher* to send collected telemetry data to the *Central Message Broker*.
 - o It uses *REST POST* to register devices with the *Resource Catalog* and acts as *REST Client* too to registers itself via *POST*.
- *Data Analytics* is a backend service responsible for processing incoming telemetry.
 - o It works as an *MQTT Subscriber* to receive "Data" topics from the Central Message Broker.
 - o It implements a "*Check TH*" (*Threshold*) logic, comparing incoming data against configurations fetched from the Resource Catalog.
 - o If a threshold is breached, it acts as a *MQTT Publisher*, sending an "*ALARM*" message back to the Message Broker.

- *"Compute Decay"* is a specialized backend processing unit, which processes data streams to calculate decay metrics.
 - o It works as a *MQTT Subscriber* to receive the temperature and humidity data the Central Message Broker.
 - o It works as a *REST Client* to *POST* the computed results into the *Interface Database*.
- *Interface DB / Writer Service* is a dedicated backend component that joins telemetry data with the computed analytics and sends results to the external database.
 - o Works as a *MQTT Subscriber* that listens to the sensor's measurements from the Central Message Broker.
 - o It receives data (via *POST*) from the *Compute Decay* service.
 - o Writes complete data points (with all fields: temperature, humidity, tilt, decay) to the external *database*.
- *The Influx Database (DB)* is an external cloud-based storage service.
 - o It receives processed data (via *POST*) from the *Interface DB*.
 - o It provides historical data to the *Dashboard* via *REST GET* requests.
 - o Supports time-range queries for analytics and visualization.
- *The Dashboard* is the user interface for the system.
 - o It works as an *MQTT Subscriber* to receive real-time *"ALARM"* notifications from the Message Broker.
 - o It communicates with the *Resource Catalog* via *REST GET* to obtain system metadata.
 - o It retrieves processed analytics and decay data from the external *Database* via *REST GET* to visualize the status of the monitored area.

5 Desired Hardware components (only among those we can provide)

Device Name	Quantity	Needed for...
Accelerometer	1 x pole	Tilting and decay formula
Humidity sensor	1 x pole	Decay formula
Temperature sensor	1 x pole	Decay formula
Raspberry pi	1 x pole	Process data and send it
Antenna LoRa	1 x pole	Communication
Central gateway	1 x 100 poles (estimation)	Data handling