1 Name of Use Case

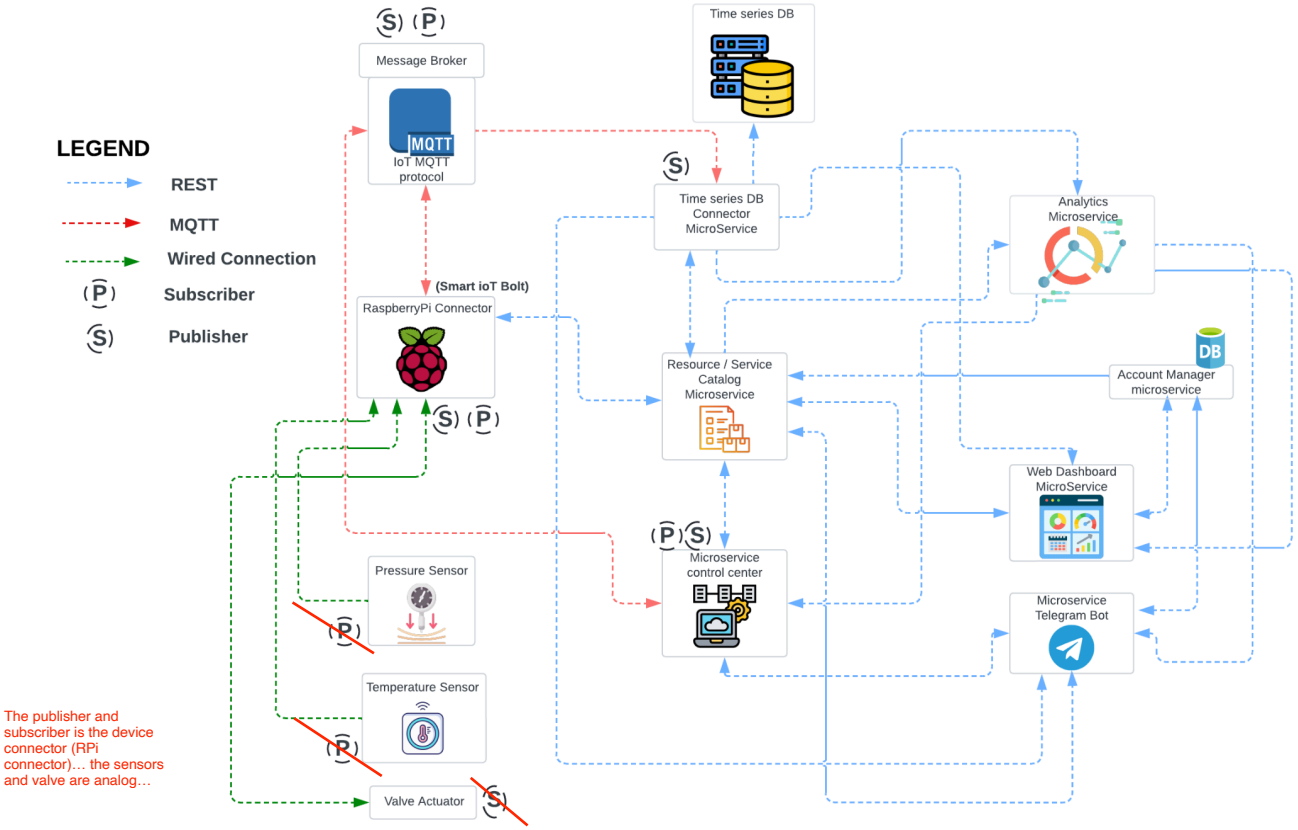
Name of the Use Case	**IoT platform Smart Bolt for PipeLines**	
Group Name	**31**	
Version No.	v1.3	
Date	19/11/2024	
Team Members (with student IDs)	Mohammad Eftekhari pour - s307774 Mahdi Rajaee - s308497 Hamid Shabanipour – s314041 Tanin heidarloui moghaddam - S308784	s307774@studenti.polito.it s308497@studenti.polito.it s314041@studenti.polito.it s308784@studenti.polito.it

2 Scope and Objectives of Function

Scope and Objectives of Use Case	
Scope	The proposed IoT platform aims to provide services for monitoring pressure and temperature in heavy machinery using Smart IoT Bolts.
Objective(s)	The expected results consist of providing a smart control of machinery by monitoring pressure levels, minimizing potential failures, and promoting predictive maintenance via user-awareness applications.
Domain(s)	Industrial Machinery, IoT Monitoring, Predictive Maintenance.
Stakeholder(s)	Industry managers, Maintenance engineers, and Machinery operators.
Short description	The proposed IoT platform is designed to optimize industrial operations by monitoring pipeline pressure and temperature using Smart IoT Bolts. These sensor-enabled bolts collect real-time data, allowing for efficient control strategies that prevent failures or breakdowns. The platform leverages historical data stored in a Time Series Database to enable predictive analysis through an Analytics Microservice. By analyzing data trends and detecting anomalies, the system identifies patterns in pressure and temperature fluctuations, predicting potential issues before they become critical. This proactive approach ensures pipeline stability and supports predictive maintenance, with preemptive alerts to users about possible failures or safety risks. The platform offers unified interfaces via REST and MQTT protocols, enabling comprehensive monitoring and management of pipeline integrity. It integrates seamlessly with broader industrial monitoring systems, providing enhanced awareness of pipeline conditions.

	Key features include: - Remote monitoring and control of pipeline pressure and temperature - Predictive maintenance through intelligent analysis and control strategies for pipeline integrity - Unified REST and MQTT interfaces for real-time pipeline monitoring and alert notifications - End-user applications that enhance visibility and awareness of pipeline systems, contributing to improved industrial operations

Diagram of Use Case



3 Complete description of the system

The proposed IoT platform for industrial machinery follows the **microservices design pattern**. It also exploits two communication paradigms:

1. **Publish/subscribe** based on **MQTT** protocol.
2. **Request/response** based on **REST Web Services**.

In this context, the following actors are identified:

- **Message Broker**: Provides an asynchronous communication based on the publish/subscribe approach via the MQTT protocol. It manages the data transmission from Smart IoT Bolts to the platform.

- **Resource/Service Catalog**: The catalog works both as a device (resource) and service registry. It provides system components with details for applications and control strategies, including REST Web Services for accessing **historical data**, sensor, and actuator lists. At start-up, each actor retrieves necessary configurations from the Catalog. Additionally, it maintains up-to-date statuses of all devices, receiving updates from the Raspberry Pi Connector. This information is then **retrieved** with the Microservice Control Center for control actions, the Web Dashboard and Telegram Bot for monitoring and alerts, and the Analytics Microservice for data processing and analysis.

- **Raspberry Pi Connector:** The emulated Raspberry Pi Connector acts as a data publisher, collecting temperature and pressure readings from sensors attached to the pipeline and monitoring pipeline conditions. It connects to the valve actuator (wired connection) to respond during emergencies. This data is published to the Message Broker via MQTT, enabling real-time monitoring of pipeline health and safety. Additionally, it updates the Catalog with sensor and actuator statuses over REST, providing system-wide visibility. It can also retrieve the configuration about the frequency of data publication, and threshold limits from the Catalog. The connector receives control commands from the Control Center (MQTT), allowing for automatic or manual valve operations based on safety parameters. The Raspberry Pi Connector enables comprehensive monitoring and control by tracking both sensor data and actuator status.

- **Control Center:** Its role is to process incoming sensor data, apply control rules or logic, and then send appropriate control commands to actuators (e.g. Valve), it receives real-time sensor data (pressure and temperature readings) from the Message Broker via the MQTT communication protocol. Based on predefined rules or algorithms (such as thresholds, timing intervals, or conditional logic), the Control Center evaluates sensor data to determine if any action is required. For instance, if the temperature or pressure exceeds a certain threshold, it may need to open or close the valve. Sends commands (via MQTT) to the Raspberry Pi Connector to operate the Valve Actuator and Sends alerts if manual intervention is required or if anomalies are detected (forwarded to Telegram Bot).

- **Time Series Database**: to store sensor time series measurements, we will use InfluxDB.

    - **InfluxDB**: is an IoT-friendly database, specifically designed for seamlessly storing and querying time-stamped sensor measurements. It excels in managing dynamic data streams from IoT devices, especially sensors, where accurate timestamps are crucial for contextual analysis and decision-making. Its optimized architecture ensures efficient handling of continuous, high-frequency data updates typical in IoT scenarios. It is a preferred choice for applications demanding real-time insights and historical trend analysis based on sensor readings.

- **Time Series Database Connector**: The Time Series DB Connector acts as an interface between the sensor data(data published by message broker using MQTT protocol) and the

Time Series Database via REST web services (**InfluxDB**). It ensures that data streams from the Message Broker are stored efficiently in InfluxDB with accurate timestamps. This connector handles data aggregation and provides APIs to retrieve historical data for the Analytics Microservice and the Web Dashboard**(both Web Dashboard and Analytics using REST)**.

- **Web Dashboard**: The Web Dashboard serves as the user interface, allowing users to view sectors with Smart IoT Bolts in place. It displays sensor and actuator statuses from the Resource Catalog using REST. It visualizes data insights from the Analytics Microservice (using REST) along with historical data from the Time Series DB Connector through detailed plots. Access is restricted to authorized users through integration with the Account Manager.

- **Telegram Bot**: It also acts as an interface that retrieves the latest pressure and temperature average from the Catalog via REST, and it also has the ability to send a command to the actuator. This is done by sending a command to the control center through REST API, and then the control center sends this command to the message broker via MQTT.

  Maybe can send this commands using MQTT directly ? Also, it might receive "alerts" of the analytics (via message broker) using MQTT !!

- **Account Manager**: The Account Manager handles user authentication and authorization for restricted services on the Web Dashboard and Telegram Bot through the REST API, interfacing with the Resource Catalog via REST API. It allows a typical user to create and modify sectors and manage their Smart IoT Bolts, ensuring secure access and control over the IoT devices within their sector.

- **Analytics**: This microservice uses temperature and pressure data from time series DB connector via REST and applies a prediction method using historical data to anticipate future risks. It evaluates forecasted values against predefined safety thresholds, assessing whether temperature or pressure is likely to exceed safe limits. If a potential hazard is detected, the microservice generates an alert and sends it to both the Web Dashboard and Telegram Bot via RESTful API. This ensures that users are promptly notified of possible issues through dashboard updates and Telegram notifications, allowing them to take preventive action as needed.

Maybe send it to telegram using MQTT instead of REST?

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

Device Name	Quantity	Needed for…