SMART FACTORY

Software Engineering for Adaptive Systems project
Giordano Tinella 258520
11/07/2023

Introduction

The *Smart Factory* project showcases the potential of IoT and self-adaptive systems in revolutionizing industrial environments. This project aims to create a highly advanced system that monitors and optimizes various aspects of factory operations. At its core, the project embraces the concept of the Internet of Things (IoT), which facilitates the seamless connection and communication between physical sensors. By establishing a network of interconnected components, the system enables the efficient exchange of data, enabling real-time monitoring, analysis, and adaptive decision making. By integrating the principles of self-adaptive systems, the project empowers the system to dynamically adjust its behavior based on environmental feedback. This adaptability enables the system to autonomously respond to changing conditions, optimize performance, and improve overall efficiency. The project encompasses a wide range of IoT sensors and actuators strategically deployed throughout the factory environment. By combining IoT technologies and self-adaptive systems, the project demonstrates the potential of intelligent and interconnected systems in industrial settings.

Devices (sensors and actuators)

Real-time data collection, analysis, and adaptive decision making enable the system to optimize processes, enhance energy efficiency, improve productivity, and maintain a safe and secure working environment. These sensors enable real time data collection, providing valuable insights into crucial parameters such as temperature, air quality, and movement.

- Temperature control is a key focus of the project. Through the utilization of temperature sensors in each room, the system continuously monitors the ambient temperature. Leveraging actuators, such as hot conditioners and cool conditioners, the system proactively regulates room temperatures to ensure optimal working conditions and energy efficiency.
- Ensuring optimal air quality is another vital aspect. With the integration of air sensors, the system constantly monitors the air quality in real time. Whenever the air quality falls below predefined thresholds, the system activates the fan actuator to improve air circulation, creating a healthier and more productive working environment.
- To enhance security measures, movement sensors are being incorporates in each room. During locked hours, if movement values exceed predefined thresholds, the system triggers the alarm actuator, ensuring the prompt detection and response to potential security breaches. Each room has a locker actuator that will block the access to the corresponding room.

Used Technologies

MQTT is a lightweight messaging protocol that plays a pivotal role in project. It provides efficient communication between devices and facilitates the transfer of real time data. MQTT's publish/subscribe architecture enables seamless data transmission, ensuring timely updates from sensors to the system. By leveraging MQTT, we achieve efficient and reliable data exchange, enhancing the overall performance and responsiveness of the system. For the project Mosquitto is used as MQTT broker and then the following are the used topics:

- rooms/{room}/sensors/movement
- rooms/{room}/sensors/temperature
- rooms/{room}/sensors/air
- rooms/{room}/actuators/alarm
- rooms/{room}/actuators/locker
- rooms/{room}/actuators/hot_conditioner
- rooms/{room}/actuators/cool_conditioner
- rooms/{room}/actuators/fan

InfluxDB is a powerful time series database, it provides a scalable and efficient solution for storing and retrieving time stamped data. InfluxDB's ability to handle large volumes of data with high write and query performance is crucial for managing the continuous influx of sensor data in real time. By utilizing InfluxDB, we can effectively store and analyze sensor readings, enabling data driven decision making and historical analysis.

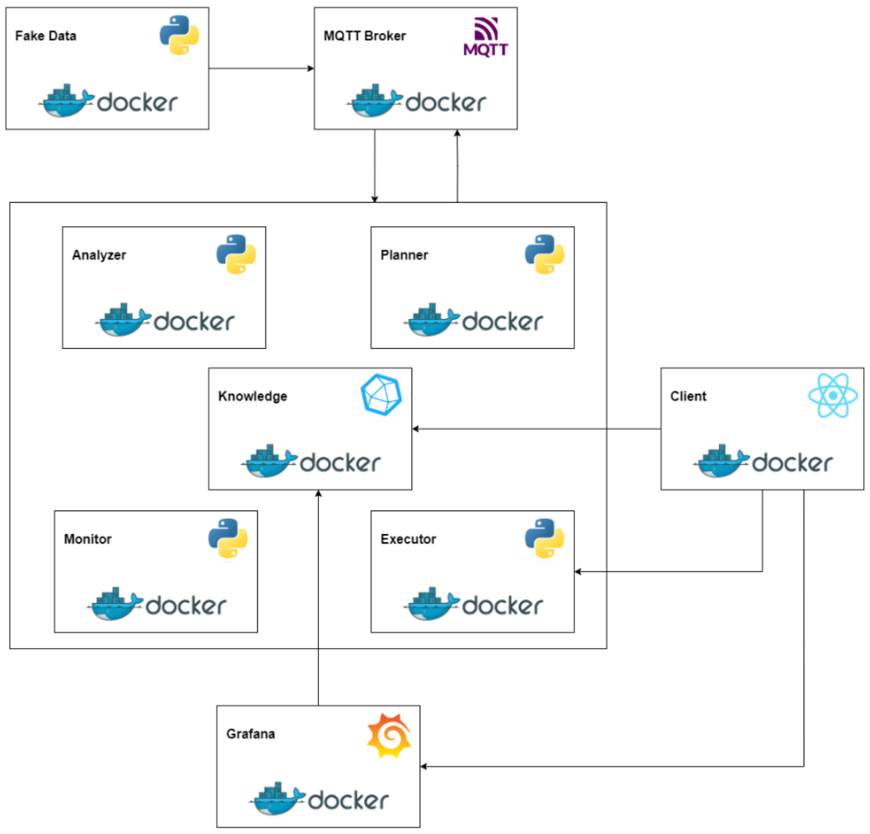
Grafana allows data visualization and monitoring, with Grafana it is possible to create dashboards and graphs that represent data from various sources. Grafana is an ideal choice for visualizing the real-time and historical data collected by the system.

NumPy is a Python library widely used for numerical computing. It provides essential capabilities for handling large arrays and matrices, along with a comprehensive collection of mathematical functions. NumPy's efficient numerical operations and array manipulation functions enable advanced data processing and statistical analysis. Its integration empowers the project with robust computational capabilities and facilitates data driven decision making.

React is a JavaScript library used to build user interfaces. It uses a component-based architecture, JSX syntax, and a virtual representation of the DOM for efficient rendering. It follows a unidirectional data flow and provides hooks for managing state in functional components. React has a large ecosystem of libraries and tools for different purposes, making it popular for creating interactive web applications.

Docker is a containerization platform that plays a significant role in the project's deployment and management. By utilizing Docker, the project achieves a standardized and efficient environment for running applications and services. Docker containers encapsulate all the necessary dependencies and configurations, ensuring consistency and portability across different environments. This allows for easy deployment, scalability, and isolation of components.

Architecture

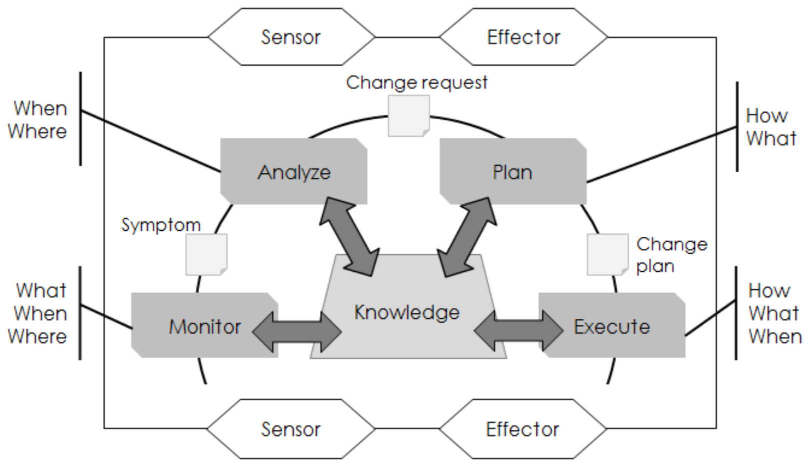


The project's architecture is well structured and modular to effectively monitor and optimize factory operations. The architecture consists of 9 Docker containers, each serving a specific purpose within the system:

- The Fake Data container generates simulated sensor data for temperature, air quality, and movement values. It publishes this data to the Mosquitto container for further processing.
- The MQTT broker container, facilitate the communication between different components in the system. It receives the sensor data published by the Fake Data container and ensures seamless data transfer.
- The Monitor container subscribes to the MQTT broker and receives the sensor data transmitted by the Fake Data container. It analyzes the incoming data and store them into InfluxDB.
- The Knowledge container utilizes InfluxDB to store and manage the sensor data received from the Monitor container. It provides efficient storage and retrieval of time stamped data for further analysis.
- The Analyzer container retrieves sensor data from the InfluxDB database and performs advanced analysis operations. It calculates statistical measures, detects patterns, and identifies critical situations based on predefined rules.
- The Planner container receives information from the Analyzer container regarding critical aspects identified in the sensor data. It generates a list of potential actions to address the identified issues, based on predefined strategies and rules.

- The Executor container acts as an intermediary between the Planner container and the actuators in the system. It receives the list of actions or solutions from the Planner container and communicates them to the appropriate actuators.
- The Grafana container serves as a powerful data visualization and monitoring platform. It retrieves data from the InfluxDB database and provides real-time and historical insights through visually appealing dashboards and graphs.
- The Client container is a React-based client that display data generated by Grafana and provide a manual mode for adjusting actuator values.

MAPE-K loop



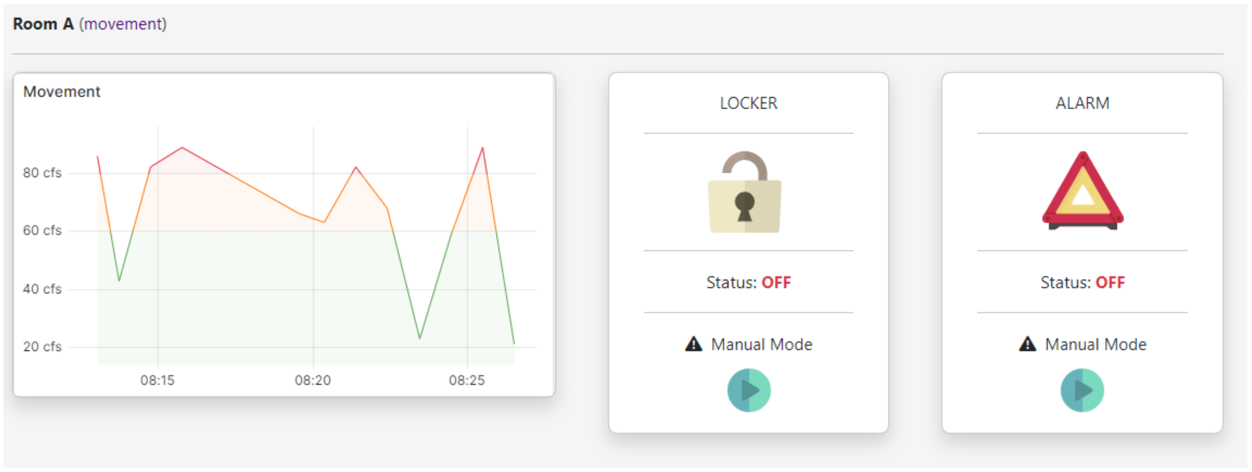
As we examine the system architecture of the project, it becomes evident that the adopted design follows the principles of a MAPE-K loop. This architectural framework, consisting of the Monitor, Analyzer, Planner, Executor, and Knowledge components, forms a closed feedback loop. Through this loop, the system continuously monitors the factory environment, analyzes the collected data, plans appropriate actions, executes them through actuators, and updates its knowledge.
We can summarize saying that:

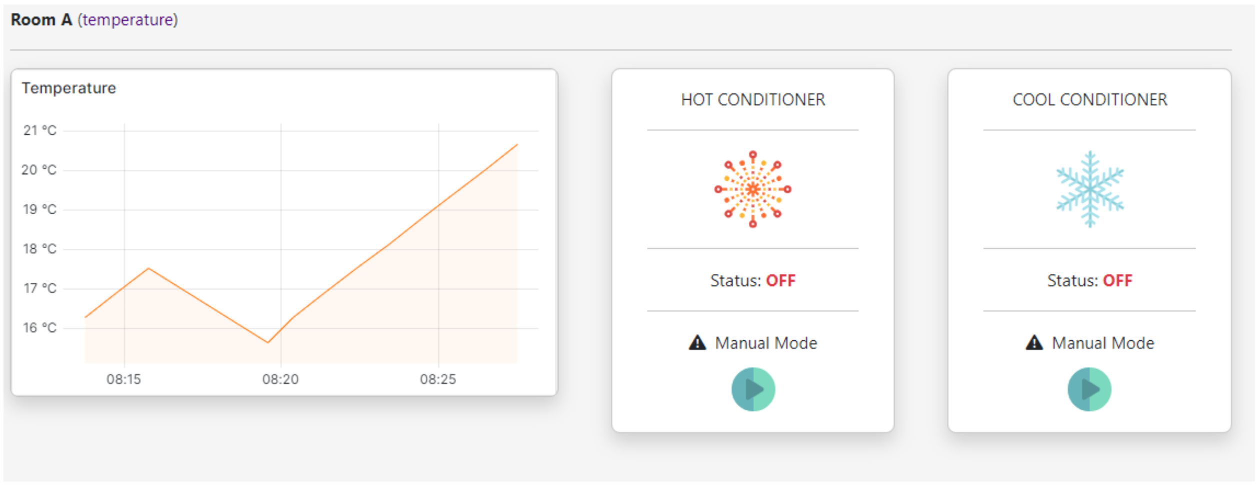
- The Monitor stage collects data from sensors in real time, monitoring the factory environment for any changes or anomalies.
- In the Analyze stage, the collected data is analyzed and processed to identify patterns, trends, and potential issues or critical situations.
- Based on the analysis results, the Plan stage generates a set of actions or strategies to address the identified issues or optimize factory operations.
- The Execute stage involves the execution of the planned actions. This stage communicates the actions to the relevant actuators, triggering the necessary responses in the factory environment.
- The Knowledge component stores and updates information about the factory operations, historical data, and rules for analysis and decision-making. It serves as a knowledge base for the system.

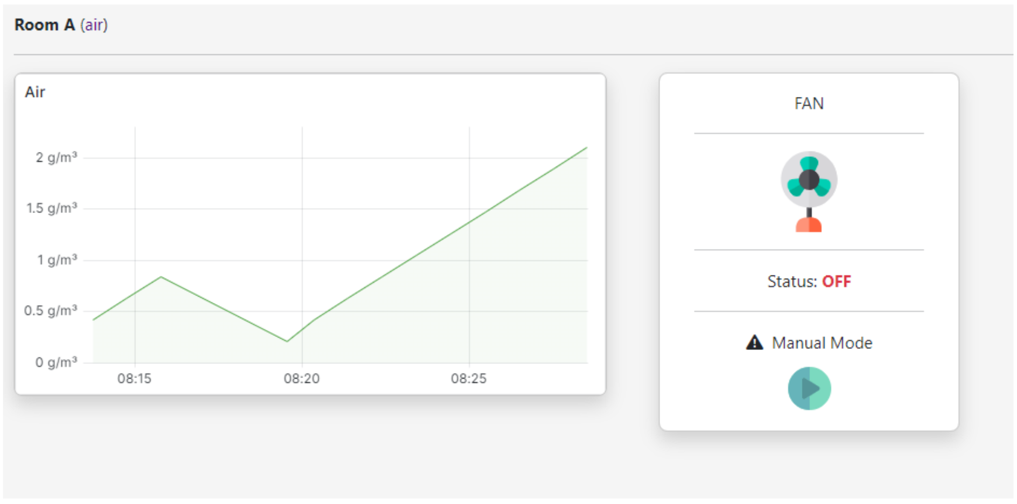
The MAPE-K loop enables a continuous feedback loop where the system adapts and improves its operations based on real-time data analysis, planning, and execution. By leveraging this loop, the project achieves efficient monitoring, analysis, optimization, and adaptation to enhance overall factory performance.

Data Visualization

In the *Smart Factory* project, data visualization and monitoring play a crucial role in gaining insights into the factory operations. Let's delve into the details of how the client empowers us to gain valuable insights into our system, following pictures represent views for one of the factory's rooms.







Conclusion

In conclusion, the *Smart Factory* system exemplifies a high level of autonomy by autonomously initiating appropriate actions based on the current factory conditions. During the reactive phase, the application leverages contextual analysis to make well informed decisions while ensuring the smooth functioning of the system. As a result, the application effectively meets all the essential criteria for creating an autonomous system within the specified context. The system's ability to autonomously adapt and respond to real time conditions showcases its efficiency and effectiveness in optimizing factory operations.

Instructions

- Clone the repository: https://github.com/GiordanoT/smart-factory.git
- Run: docker compose up -d
- Go to: http://localhost:3001