UNIVERSITÀ DEGLI STUDI DI L'AQUILA

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DISIM
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

Autonomous Greenhouse using MAPE-K

Software Engineering for Autonomous Systems

Professor: Davide Di Ruscio

Students:

Mariama Celi S. de Oliveira - 297854

Motunrayo Osatohanmen Ibiyo – 297859

Table of Contents

Introduction

A greenhouse is a structure designed to create ideal conditions for plant growth [1]. It is typically constructed from galvanized iron and covered with transparent plastic or glass, allowing natural light to enter. Being a closed environment, a greenhouse enables the control of factors such as temperature, humidity, and carbon dioxide (CO2) levels, which are essential for optimal plant development [2]. As a result, greenhouses are essential tools for promoting plant growth, controlling pests, reducing environmental pollution, and lowering carbon emissions [2]. Therefore, determining the optimal parameters for operating a greenhouse is crucial for efficiently using resources such as energy, land, and water [2].

Considering this context, we propose an automated greenhouse system designed to sustain and regulate the greenhouse environment, optimizing conditions for crop growth in a sustainable manner. The proposal is structured into six sections. The first sections, *System Goals* and *System Requirements*, outline the main objectives of the proposed system and its functionalities. Following this, in the sections *Managed Resources*, we describe the sensors and devices utilized within the environment that directly modify the context. The *Autonomic Manager* section explains the system choices based on a taxonomy. Next, the *MAPE-K Loop* describes the system's behavior according to the MAPE-K loop framework. Finally, the *System Architecture* section provides details about the system components, technologies utilized, and information flows.

By the end of this proposal, we aim to present a comprehensive overview of the automated greenhouse system, ensuring it is detailed and practical enough for implementation.

Goals of the system

The primary purpose of the automated greenhouse system is to optimize crop growth conditions through autonomous operations. Its main goals are:

- Maintain key variables (**temperature, relative humidity, light, and CO2 concentration**) at appropriate levels for healthy crop development.
- Deploy a flexible solution that is adaptable for more than one crop type.
- Develop an interface where the user can monitor the key variables and resources of the greenhouse.

System Requirements

To align with the objectives of the system, we have identified both functional and non-functional requirements.

Functional Requirements

Table 1: Functional Requirement of the system

ID	Name	Description
FR-01	Real-Time Environmental Monitoring	The system must continuously monitor greenhouse variables, including temperature, humidity, CO_2 concentration, and light intensity, using sensors that transmit real-time data to the system and send it to the Knowledge module.
FR-02	Managed Resource Control	The system must control managed resources, such as fans, heaters, motorized hatches, water pumps, LED lights, and CO_2 injectors, to maintain optimal environmental conditions for plant growth.
FR-03	Customizable thresholds	The environmental variable thresholds should be configurable in a file or UI allowing the system to support different crops environmental needs.
FR-04	Real-Time Data Visualization	The system must provide a user dashboard that displays real-time greenhouse data, including temperature, humidity, CO_2 levels, and light intensity, alongside the current status of managed resources.
FR-05	Historical Data Access	The dashboard must allow users to view historical environmental data for analysis and monitoring purposes.
FR-06	Predictive Analysis	The system must predict future values for temperature, humidity, and CO_2 to act proactively.

Non-Functional Requirements

Table 2: Non-functional Requirements of the System

ID	Name	Description
NFR-01	Scalable Architecture	The system must be scalable, allowing integration of additional sensors, actuators, or greenhouses without significant architectural modifications.
NFR-02	Maintainable Microservice Design	The system must adopt a modular microservice architecture to ensure maintainability, enabling independent updates, replacements, or improvements to each service without affecting the entire system.
NFR-03	Threshold Flexibility for Crop Types	The system must allow flexibility for different crop types, ensuring that threshold values for environmental variables can be easily adjusted based on user requirements.

NFR-04	Platform Compatibility	The system must be compatible with standard hardware and software platforms, ensuring smooth deployment and accessibility.

Managed Resources

To ensure optimal operation and productivity, the greenhouse system relies on the management of key resources. These resources directly influence critical environmental variables that impact plant growth, such as **temperature**, **humidity**, **light levels**, and **CO2 concentration**. Below, we outline the primary managed resources and the variables they control:

1. **Temperature Control System:** The system dynamically maintains the temperature of the greenhouse within specified thresholds by controlling the state of the fan, hatch and heater.
2. **Humidity Control System–** The system dynamically regulates the state of the pumps to maintain the humidity of the green house.
3. **Lightening System –** The system regulates the available lights by controlling the states of the LED lighting systems to achieve optimal lightening for the plant based on the available sunlight.
4. **CO2 levels–** The system regulates the CO_2 levels by controlling the state of the CO_2 injector based on the specified thresholds for CO_2.

Figure 1: Greenhouse lateral view. The ceiling features hatches at the top and fans at the bottom.

By managing these resources, the system creates an environment where key variables are optimized, ensuring healthy and sustainable plant growth while minimizing resource wastage.

Sensors and the Actuators for managed resources

Sensors

To be able to appropriately control the managed resources is necessary to measure the variables desired to be controlled. The sensor list includes:

- Temperature sensors (to monitor air temperature).
- Humidity sensors (to measure air moisture levels);
- CO2 sensors (to track carbon dioxide concentration);
- Light intensity sensors (to measure available natural light levels).

Actuators

These are the devices used to act upon the environment to manage the resources:

- Fans (to control air circulation, temperature and ventilation)
- Heaters (to increase temperature as needed)
- Motorized hatches (to open or close ventilation openings and impact temperature)
- Pumps (to distribute water through irrigation systems)
- LED lights (to provide artificial light during low natural light conditions)
- CO2 injectors (to release carbon dioxide into the greenhouse)

Sectors

Since a greenhouse encompasses a large area, the sensors and actuators are distributed across different sectors of the greenhouse. Therefore, the system described will be configured with different areas depending on how the user defines it. Each area will have a sensor for each environmental variable and an actuator.

Autonomic Manager

The developed system was structured based on the taxonomy illustrated in Figure 2. This section will explore the characteristics of the autonomic manager according to this framework.

Architectural pattern

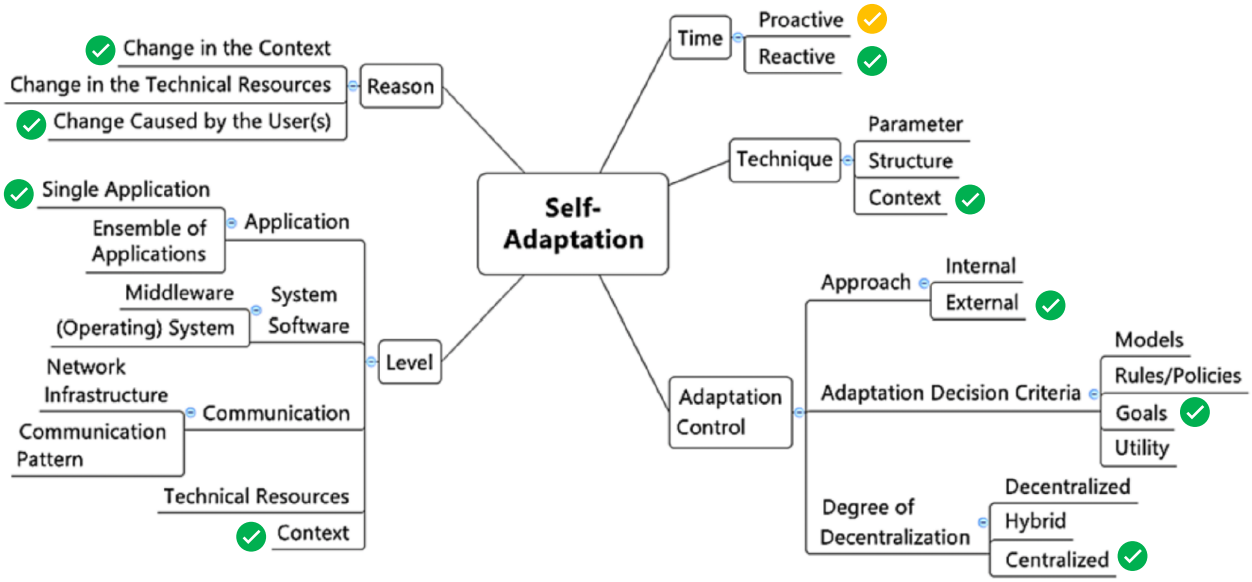


Figure 2 - System Taxonomy

Time

Proactive and Reactive – Depending on the variable to be adapted, the system will act in a proactive or reactive manner. Variables such as temperature, humidity, and CO2 levels, where it is possible to run predictions regarding their future behaviour, allow the system to perform the adaptation before critical levels are reached. However, for light levels, the system will detect when daylight is over to activate the artificial lights.

Continuous monitoring – The monitoring will be constant regardless of the variable values.

Reason

Change in the Context – Adaptation is necessary due to changes in key variables—temperature, relative humidity, light, and CO2 concentration—to prevent any of them from exceeding expected ranges or behaviours.

Change Cause by the User(s) – Adaptation can also occur due change in the type of crop selected by the user. Each crop requires a different threshold for the key variables.

Technique

- **Context Adaptation** – The system keeps track of the key variables, and when one of them goes out of the expected range, one of the actuators (fans, heaters, hatches, pump, LED lights, and CO2 injector) is activated to modify the context to a desired state.

Level

- **Application** - ***Ensemble of Applications*** – The solution is deployed in a set of microservices, each responsible for one functionality of the solution. Since each service is specialized, each service can easily be replaced by another if needed.
- **Context** – The system responds to changes in the environment to maintain the variables within desirable ranges for optimal crop yield through its actuators.

Adaptation Control

- **Approach** - ***External Approach*** – The system adaptation is separated from the managed resources. Sensors observe the key variables and send the values to a controlled central system that makes decisions based on the adaptation goals and predictions.
- **Adaptation Decision Criteria** – ***Goals*** - The adaptation must happen based on a set of adaptation goals.
- **Degree of Decentralization – *Centralized*** - The adaptation control is centralized. In other words, a unique subsystem will be responsible for implementing the adaptation logic based on monitoring the context and resources.

Adaptation goals

Below are listed the goals that must be achieved by the system. The values highlighted in bold should be configured in a specific file depending on the type of crop monitored.

Goal	Description	Evaluation Metric
Maintain an ideal temperature range for the crop	The temperature must be between an upper limit and a lower limit.	**Lower value <** temperature (Co) **< upper value**
Maintain relative humidity levels range for the crop	The humidity must be between an upper limit and a lower limit.	**Lower value <** humidity % **< upper value**
Provide X amount of light exposure to the crop	A specified light intensity must be provided to the crop	**Available light >=** Lower Light intensity limit
Maintain CO2 levels in a range for the crop	The CO2 levels must be between an upper limit and a lower limit.	**Lower value <** CO2 value in ppm **< upper value**

Decision Function

The decision-making functionality of the autonomic manager is built on a **rule-based** system. While the system is designed to achieve specific goals, it operates by relying on predefined rules that trigger actions in response to monitored variables, characterizing it as a rule-based approach.

System Architecture

The system architecture is illustrated in Figure 3, showcasing a microservice-based design with each component running in its own Docker container. For the sake of abstraction, the Broker component surrounds most of the other components, meaning that communication between these components is mediated by the Broker component which implements the MQTT protocol. The nature and order of the communication between the components is clarified by the Sequence Diagrams presented in Figures 4 and 5 in the section below.

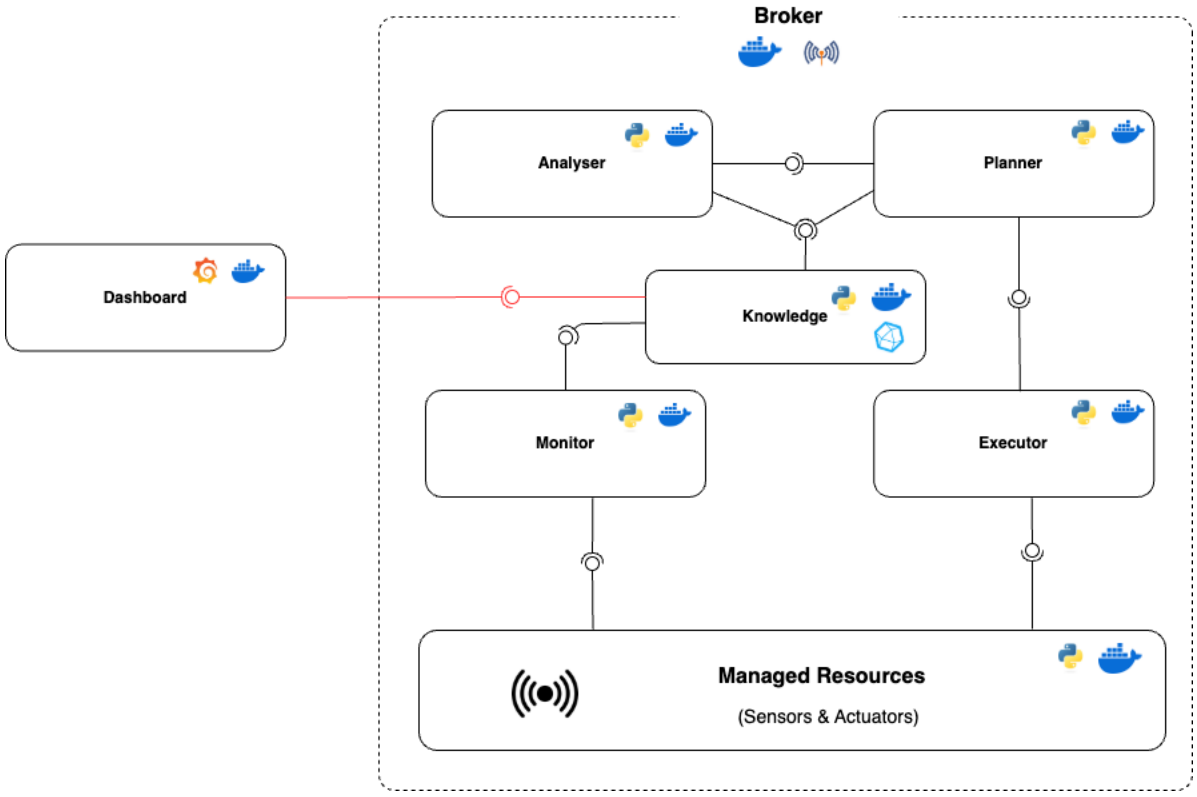


Figure 3 System Architecture Overview

Sequence Diagrams

The sequence diagram below shows two system use cases. The first, illustrated in Figure 4, depicts the sequence of actions in the MAPE-K Loop, from capturing the sensor data to the end when the actuator receives the course of action decided by the system. The second use case, shown in Figure 5, describes Dashboard Visualization, where the user can continuously view the sensor measurements and the status of actuators. Both use cases operate in a loop within the system.

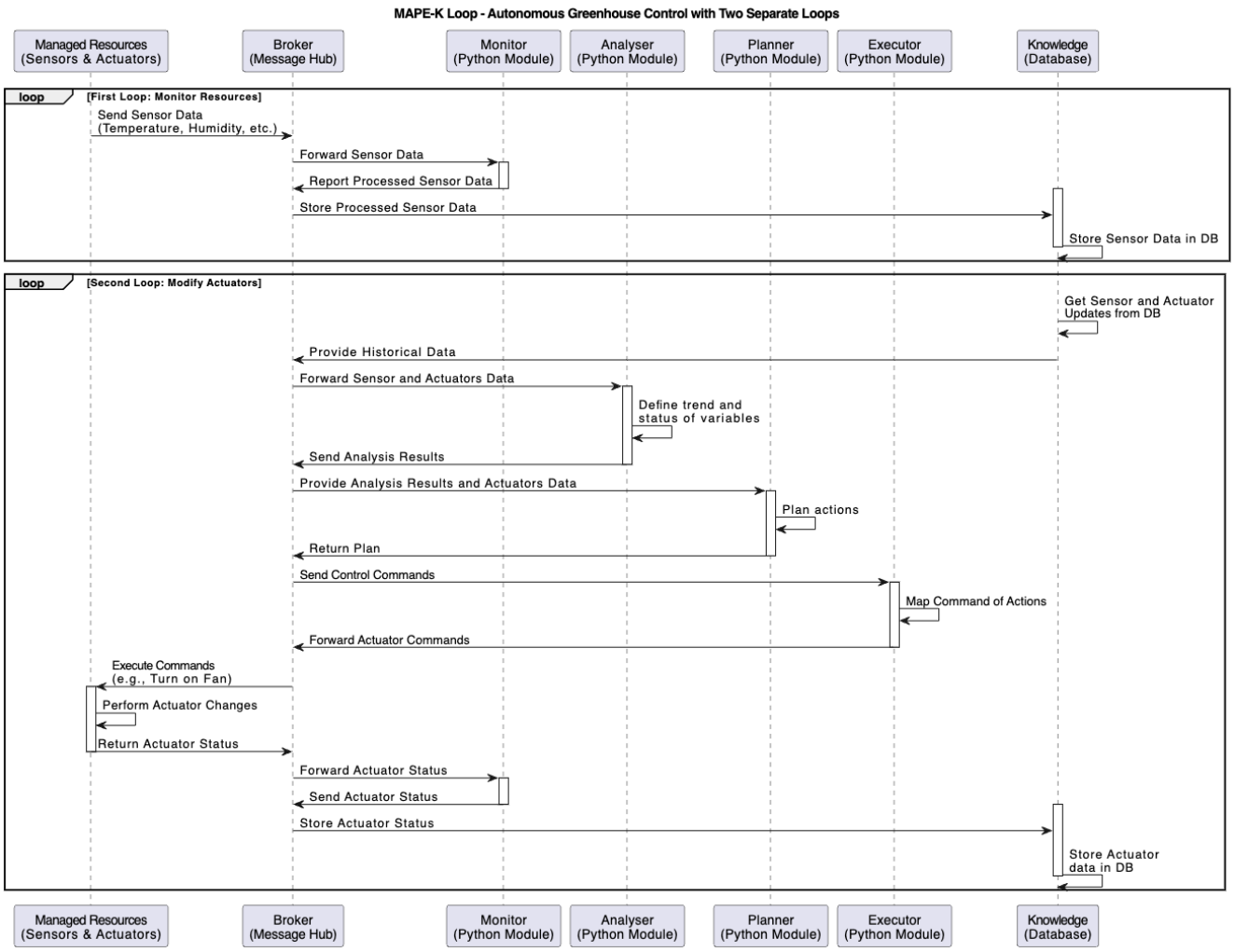


Figure 4 MAPE-K Sequence Diagram

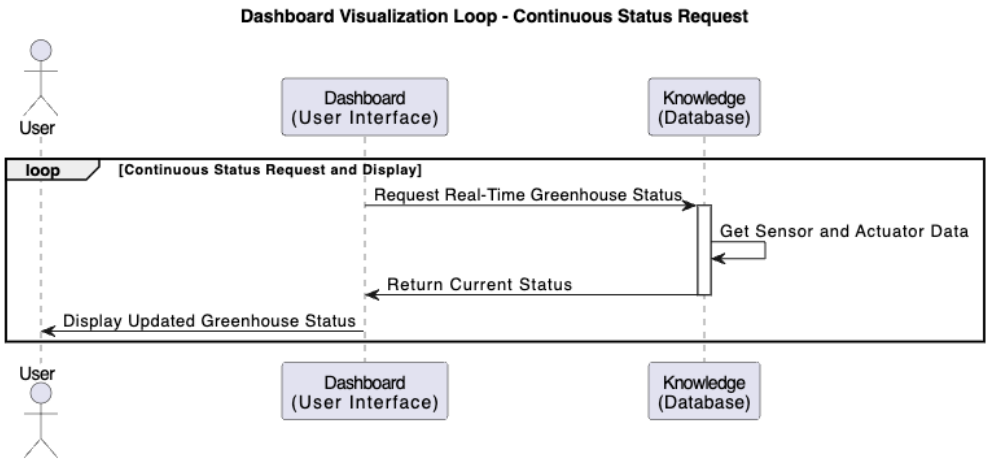


Figure 5 Dashboard Visualization Sequence Diagram

The explanation of each component presented in the System Architecture and Sequence Diagrams will be detailed in the sections below.

System Components

Broker

The broker is responsible for managing asynchronous communication between services, as shown in Figures 4 and 5. We used Mosquitto, an open-source server implementation of the MQTT protocol, as our broker. All components, except for the dashboard that communicates with the knowledge (highlighted in red), use this broker. Each component publishes data on a topic that starts with "greenhouse/..." followed by a unique identifier. This structure allows other services interested in the information to subscribe to the relevant topic.

Managed Resources

This component includes the sensors and actuators of the system. In our implementation, we simulated both sensors and actuators using Python code. To initialize the simulation, a .json file named `sector_config.json` is read, containing the desired configuration. Utilizing these files allows flexibility to the system configuration since neither the number of sectors in the greenhouse nor the environmental conditions are hard coded in the system.

Sensors

In our system, the data collection from five sensors was simulated: temperature, humidity, CO2 levels, and light intensity (both sunlight and internal light). These values were periodically updated and published asynchronously via MQTT to the monitor for storage in the knowledge base.

Publishes: `greenhouse/sensor_raw/<section>/<sensor>`

Actuators

The actuators simulate the physical resources inside the greenhouse (e.g., fans, heaters, pumps, motorized hatches, and lights) and their current states. Their key responsibilities include modifying the environmental variable values accessed by the sensors in our system, as well as sending their current status to the monitor for storage in the knowledge base via MQTT.

Subscribes to:

- `greenhouse/execute/<section>`

Publishes:

- `greenhouse/actuator_status/<section>/<sensor>`

MAPE-K Services

The MAPE-K framework defines the flow of the greenhouse system's operation. The **Monitor Service** collects real-time data from sensors and stores it in the **Knowledge Service**. The **Analyzer Service** retrieves both real-time and historical data from the **Knowledge Service** and evaluates the current state against predefined thresholds, incorporating predictions of future states. The results of the analysis are passed to the **Planner Service**, which generates an action plan to address any deviations or anticipated issues. The action plan is then sent to the **Executor Service**, which communicates with the **Broker** to deliver commands to the **Managed Resources**, ensuring that the required adjustments are implemented. This cyclical flow enables the system to adapt dynamically and maintain optimal conditions.

Monitor:

The monitor is responsible for collecting data sent by the sensor and actuator in the system and later sending this information to the knowledge component in the system's database. In our system, the monitor subscribes to two types of data via the broker: the measurements from the sensor and the status data from the actuators. The monitor also publishes two topics asynchronously: measurements and status. The data from each sensor and actuator is handled independently by this component. Therefore, if any sensor or actuator is down, it does not interfere with the information from the others.

Subscribes to:

- `greenhouse/sensor_raw/#`
- `greenhouse/ actuator_status/#`

Publishes:

- `greenhouse/monitor/sensor_raw/<section>/<sensor>`
- `greenhouse/monitor/actuator_status/<section>/<sensor>`

Analyzer:

The Analyzer plays a critical role in evaluating and managing greenhouse resources by continuously assessing sensor data trends and classifications. It compares historical values against predefined plant-specific thresholds to determine whether a variable (e.g., temperature, humidity, CO_2 levels, or light intensity) is *too low, low, optimal, high, or too high*. Additionally, it applies slope analysis to detect trends, categorizing them as *increasing, decreasing, or stable*, which helps in predicting future environmental changes. By integrating threshold-based classification with trend detection, the analyzer enables semi-intelligent decision-making for automated control.

Analyzer Topics:

Subscribes to:

- `greenhouse/last_updates/#`

Publishes :

- `greenhouse/analyzer/<section>/past/change_status`
- `greenhouse/analyzer/<section>current/change_status`
- `greenhouse/analyzer/<section>/<managed_resources>`

Status Function: The table below describes the rules guiding the classification of the managed resources status:

Table 3: Rules guiding analyzer's state status determination function

State	Description	Analysis Function
Too_low	Below acceptable range	Less than minimum acceptable value configured for the plant
Low	Acceptable but below optimal range	Greater than minimum but less than optimal range.
Optimal	Optimal range	Within optimal range
high	Acceptable but above optimal range	Greater than the optimal range but less than the maximum threshold value
Too_high	Above acceptable range	Above the maximum threshold for the specified plant.

The optimal range is evaluated using the specified threshold in '`greenhouse_threshold.json`'

$$\frac{optimal\ value - \min\ \_threshold\_value}{3} <= Optimal\ range < \frac{\max threshold\ value - optimal\_value}{3}$$

Trend Function: The analyzer's trend function uses the last 3 values of the managed resources to determine if the trend is increasing, decreasing or stable. This function uses the mathematical function of determining a slope to verify if it is increasing or decreasing.

The function outputs:

- Increasing if the slope is *greater than* 0.2,
- *Decreasing* if the slope is *less than* -0.2,
- *Stable* if the slope is *between* -0.2 and 0.2 inclusive

Planner:

The planner uses the results of the analysis from the analyzer to make plans. Based on the current state and trend of the manage resources, the planner initiates a plan that is aimed at returning or keeping the state of each section of the greenhouse within optimal state. Using if else rules that analysis the state and trend, the planner decides what needs to be done by the executor.

Subscribes to:

- `greenhouse/analyzer/#`
- `greenhouse/last_updates/+/actuators`

Publishes :

- `greenhouse/planner_strategy/<section>`

Table 4: Rules guiding planners strategy

Managed Resources	Rules	Decision
Temperature	If temperature is too high, cooling must be increased.	Turn ON fan, OPEN hatch, Turn OFF heater
	If temperature is high and increasing	Check actuator state → If the heater is ON, turn it OFF; or turn it ON fan
	If temperature is within optimal range	Turn OFF all sensors to save energy and maintain current state
	If temperature is low and stable or decreasing	Turn OFF Fan, close hatch. If fan is already OFF, turn on heater.
	If temperature is too low, heating must be increased	Turn ON heater, Turn OFF fan, CLOSE hatch
Humidity	If humidity state is high or too high	Turn OFF pump
	If humidity state is low or too low	Turn ON pump

	If humidity state is optimal and the trend is stable, or increasing	Turn OFF pump
	If humidity state is optimal but the trend is reducing	Turn ON pump
CO_2 Level	If CO_2 level is high or too high:	Turn OFF CO_2 injector
	If CO_2 level is low or too low:	Turn ON CO_2 injector
	If CO_2 level is decreasing and status is low, too_low or optimal	Turn ON CO_2 injector
Light Intensity	If current light intensity is too high	Keep LED Lights OFF
	If current available is optimal, low or too low	Keep LED lights ON

Executor:

The executor gets the decisions of the planner and forwards the command to the actuators.

Subscribes to:

- `greenhouse/planner_strategy/#`

Publishes :

- `greenhouse/execute/<section>`

Knowledge:
This component stores the sensor measurements and actuator statuses in the database. The selected database is InfluxDB, as it is specifically designed to store and manage time series data. In InfluxDB, a bucket named 'greenhouse' has been created, which contains two types of measurements: actuator_state and sensor_data, both serving as tags for indexing the data. All data is obtained through a subscription to the topics published by the monitor in the system. To provide access to historical data, Knowledge periodically publishes the last three data points available in the system as well the actuator status via a topic to the broker.

Subscribe to:

- `greenhouse/monitor/#`

Publishes:

- `greenhouse/last_updates/<section>/sensors`
- `greenhouse/last_updates/<section>/actuators`

Dashboard

The dashboard was built using Grafana. The dashboard connects directly to the knowledge base. It serves as an interface for the user to monitor the state of the greenhouse. The dashboard is divided into two views.

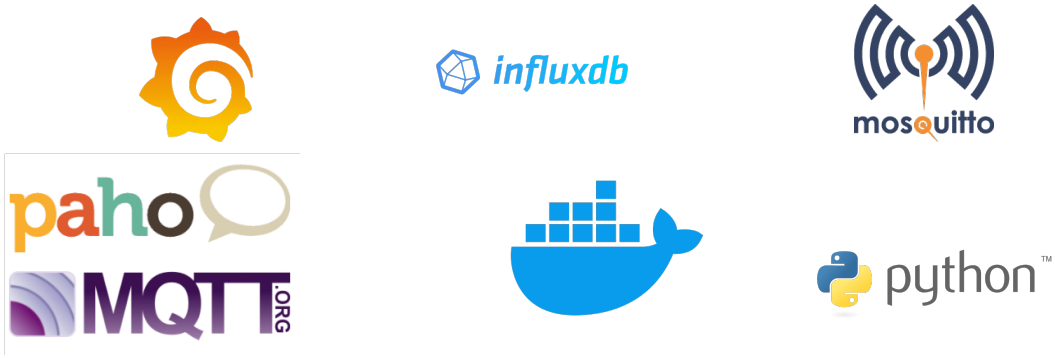
- **Current State View**: This part of the dashboard reports the current values from the sensor and actuators in real time. It displays the real-time state of the greenhouse variables (e.g., temperature, humidity, light intensity, CO2 levels) and Actuator state (ON/OFF) for all selected sections of the greenhouse.

Figure 6: Current state view of the greenhouse

- **Trends View**: This Shows the historical values of the greenhouse environment.

Figure 7:Trend view in the dashboard

Technologies Used



References

[1] Cosman, S. I., Bilatiu, C. A., & Marţiş, C. S. (2019, June). Development of an automated system to monitor and control a greenhouse. In *2019 15th International Conference on Engineering of Modern Electric Systems (EMES)* (pp. 1-4). IEEE.

[2] Li, H., Guo, Y., Zhao, H., Wang, Y., & Chow, D. (2021). Towards automated greenhouse: A state of the art review on greenhouse monitoring methods and technologies based on internet of things. Computers and Electronics in Agriculture, 191, 106558.