

AFLORA

Autonomous Farming Life-cycle Observation
with Real-time Analytics





UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA



DISIM
Dipartimento di Ingegneria
e Scienze dell'Informazione
e Matematica

Department of Information Engineering, Computer Science and Mathematics

University Of L'Aquila, Italy

Professor Davide Di Ruscio

Software Engineering for Autonomous Systems

Studente	Matricola	Email
Calogero Carlino	302154	calogero.carlino@student.univaq.it
Luca Francesco Macera	302123	lucafrancesco.macera@student.univaq.it

Table of contents

1. Introduction	3
2. Goals	3
3. Managed Resources	4
4. Sensors and effectors	5
5. Architectural Pattern	6
Components Implementation	6
Monitor	6
Analyzer	6
Planner	6
Executor	7
Knowledge	7
System Architecture	8
System Functionalities	9
Sensor configuration	9
Status thresholds	9
Sensor data generation	10
Data visualization	10
Alert notification	11
Levels of Intervention	13
6. Adaptation goals	14
7. Autonomic Decision Process	15
8. Technologies used	15
Message Broker: MQTT	15
Database: InfluxDB	16
User Interface: Grafana	16
Node-RED	16
9. Functional Requirements	17
10. Non-Functional Requirements	18

1. Introduction

Our project monitors data related to the greenhouse environment and plant health, including parameters such as temperature, humidity, and CO2 levels, as well as plant-specific metrics like soil moisture.

The primary goal of the system is to automate crop management, optimizing resources such as water and energy consumption while improving crop quality and yield.

The system is based on a **MAPE-K LOOP** architecture. It uses data gathered by sensors to store and analyze the current state of the crops, enabling it to decide which actuators to activate when detected levels exceed or fall below predefined thresholds.

2. Goals

The goals of the systems are:

- Reliable IoT system for real-time monitoring of crops and fields.
 - Develop a robust IoT system capable of tracking the temperature, humidity, and air quality
- Maintaining sensor data below the thresholds
 - The system will activate the correct actuators such as smart windows or plant sprinklers to maintain the sensed data below the fixed thresholds for the sensors used.
- Smart alarm system to alert the farmers in case of plant hazards.
 - Integrate an alarm system that alarms the users when the registered levels go above the configured thresholds.
- Smart notification system to notify the farmers when an action was taken
 - Integrate a notification system that notifies the users when the registered levels go above the configured thresholds and an action was taken to correct these levels.
- User-friendly web-based dashboard accessible to the farmers
 - Create a dashboard that provides real-time updates on plant health and greenhouses' environmental-related condition

3. Managed Resources

The managed resources of the system are:

- Irrigation System
 - The system uses soil moisture data to determine the irrigation needs of the plant terrain. These will maximize irrigation efficiency by reducing water waste, while maintaining plant health and quality.
 - If the system detects a fire in the greenhouse, it activates all the sprinklers for the greenhouse plants to extinguish it.
- Ventilation System
 - In greenhouses, the system detects high levels of CO₂, fine dust, or temperature, which are harmful to plant health, and opens or closes the greenhouse windows to allow fresh air from outside.
 - If the system detects smoke or fire in the greenhouse, it opens all the windows.
- Dehumidification System
 - In greenhouses, the system detects high humidity levels, which are harmful to plant health, and reduces the water content in the air, thereby lowering the overall humidity.
- Notification System
 - The system will notify the farmers when a particular action was taken, for example, irrigating a particular plant or opening a window in a greenhouse
- Alarm System
 - The system will use devices capable of alerting the farmers present in the area in case of high CO₂, fine dust and temperature levels. This allows the personnel to act on quickly in order to resolve the problem that may arise.

4. Sensors and effectors

- Irrigation System
 - Effectors:
 - One actuator for each plant to pour water
 - Sensors:
 - One soil moisture sensor per plant that perceives if the plant needs irrigation
- Ventilation System
 - Effectors:
 - One actuator for each greenhouse window to open/close them
 - Sensors:
 - One CO2 level sensor per greenhouse that perceives if the CO2 level of the greenhouse is too high
 - One smoke detector sensor per greenhouse that perceives if there is smoke (and eventually a fire) in the greenhouse
 - One fine dust sensor per greenhouse that perceives the level of fine dusts in the air
 - One temperature sensor per greenhouse that detects if the temperature is too high
- Dehumidification System
 - Effectors:
 - One actuator for each dehumidifier in the greenhouse.
 - Sensors:
 - One humidity sensor per greenhouse that senses if the air humidity levels are too high.
- Notification System and Alert System
 - Effectors:
 - One sound alarm for each greenhouse to alert the farmers
 - Sensors:
 - All the sensors described in the other Systems

5. Architectural Pattern

The **fully decentralized adaptation** architectural pattern was used for the adaptation control of the system, therefore each sub-system has its own adaptation logic functionality and adapts itself.

Components Implementation

Monitor

It monitors the MQTT topics for new messages and utilizes them to update the system's knowledge base. The class establishes a connection to the MQTT broker, subscribes to the MQTT topics:

- “/greenhouse_{greenhouse_ID}/plant_{plant_ID}/soil_mosture”
- “/greenhouse_{greenhouse_ID}/co2”
- “/greenhouse_{greenhouse_ID}/smoke/status”
- “/greenhouse_{greenhouse_ID}/fine_dust”
- “/greenhouse_{greenhouse_ID}/humidity”
- “/greenhouse_{greenhouse_ID}/temperature”

Then it uses the appropriate method to update the knowledge base in response to the messages received.

With this implementation, the monitor ensures continuous tracking of events, enabling alarm management and data collection for analysis.

Analyzer

This component retrieves sensor readings data from Knowledge, analyzes, processes and compares them with the threshold limits, all this on a specific time interval to actively identify and detect environmental symptoms. The determined symptoms are then communicated to the Planner through specific messages (one for each symptom) published to the following topics:

- /greenhouse_{greenhouse_ID}/plant_{plant_ID}/status
- /greenhouse_{greenhouse_ID}/status

Planner

The Planner component is responsible for retrieving system symptoms (determined by the Analyzer) and formulating adaptation strategies. In other words, it defines

action plans to bring the parameters back within safe ranges. The defined strategy is then published in the relevant topics:

- /greenhouse_{greenhouse_ID}/plant_{plant_ID}/plan
- /greenhouse_{greenhouse_ID}/plan

Executor

This component is responsible for retrieving the plans defined by the Planner and executing them by correctly operating the actuators based on what was retrieved. This component also connects to the MQTT Message Broker and subscribes to the topic in which execution plans are posted. Each message received is analyzed and, based on the content, actuators are activated by publishing messages on a dedicated topic. String defining executions being made are then posted in the relevant topics:

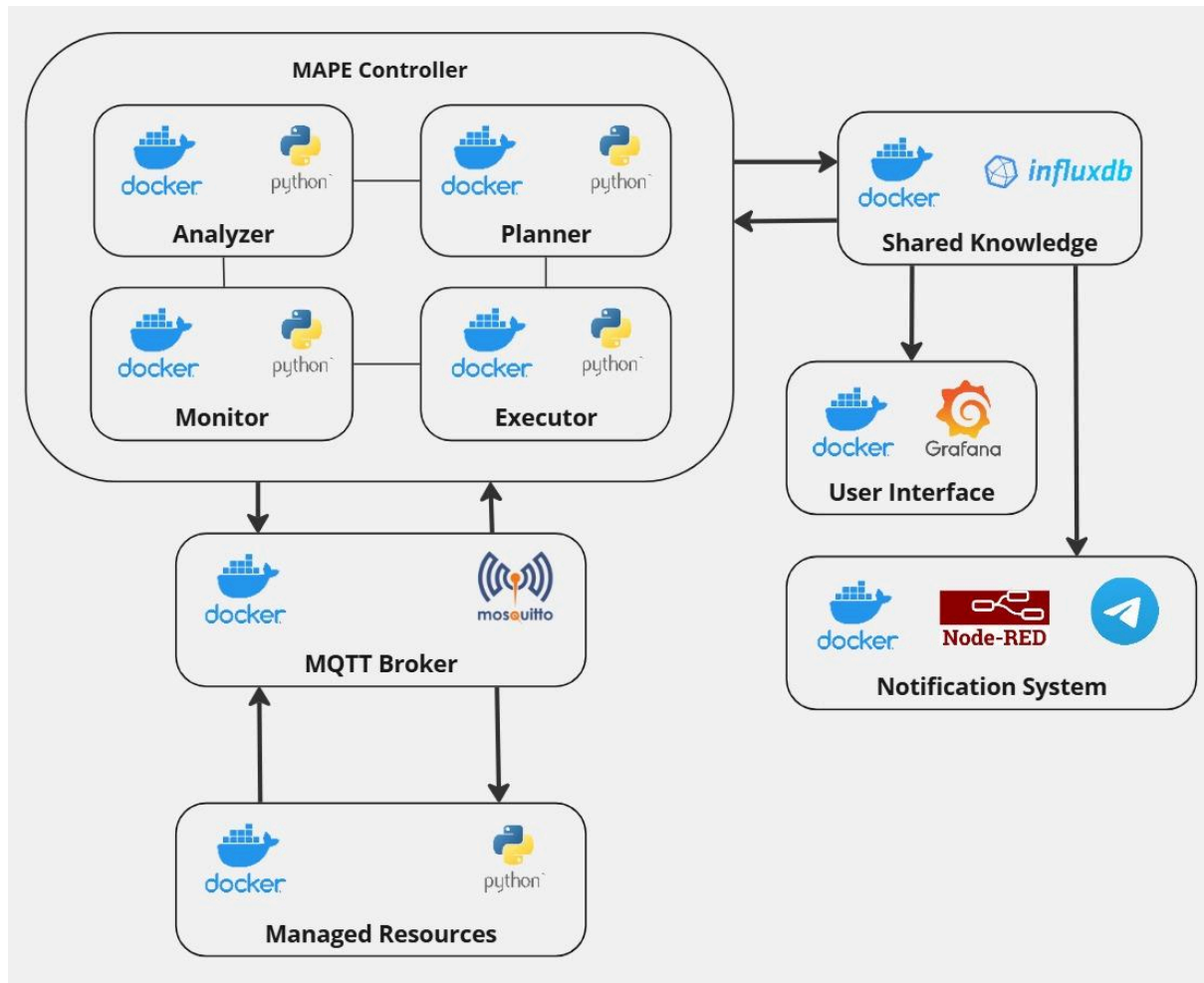
- /greenhouse_{greenhouse_ID}/plant_{plant_ID}/{managed_resource}
- /greenhouse_{greenhouse_ID}/{managed_resource}

Any changes in the status of managed resources are published to the /action_notification topic to notify users that an action has been taken.

Knowledge

The Knowledge component is responsible for storing measured parameters collected by the sensors (so the Monitor component), alarm status (general and specific for each parameter), and safety limits for each parameter.

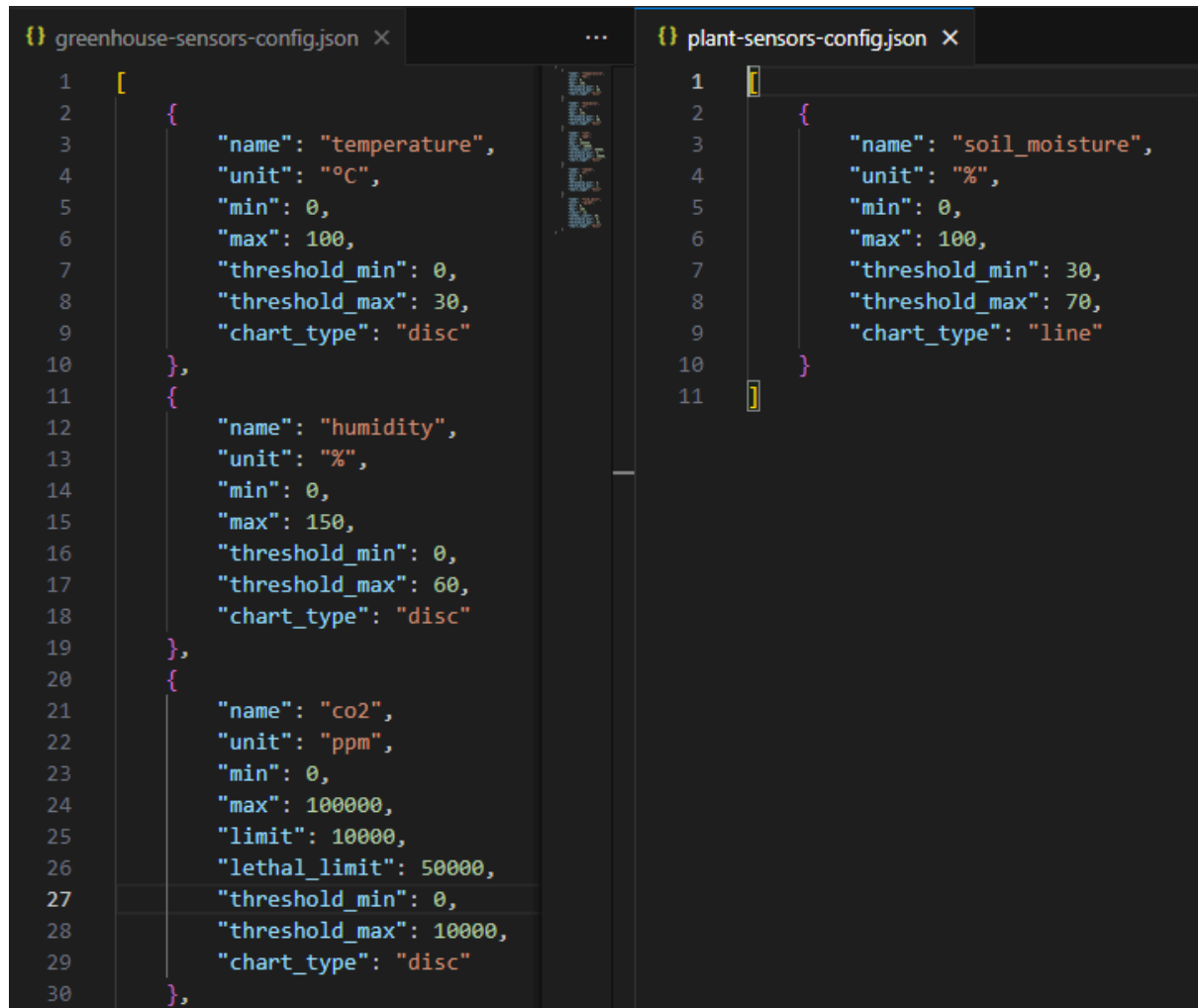
System Architecture



System Functionalities

Sensor configuration

The different types of greenhouse and plant sensors and their respective threshold limits are defined in JSON configuration files.



The image shows a code editor with two JSON configuration files side-by-side. The left file, 'greenhouse-sensors-config.json', contains a list of three sensor configurations: temperature (unit: °C, min: 0, max: 100, threshold_min: 0, threshold_max: 30, chart_type: 'disc'), humidity (unit: %, min: 0, max: 150, threshold_min: 0, threshold_max: 60, chart_type: 'disc'), and CO2 (unit: ppm, min: 0, max: 100000, limit: 10000, lethal_limit: 50000, threshold_min: 0, threshold_max: 10000, chart_type: 'disc'). The right file, 'plant-sensors-config.json', contains a single sensor configuration: soil_moisture (unit: %, min: 0, max: 100, threshold_min: 30, threshold_max: 70, chart_type: 'line').

```
{
  "name": "temperature",
  "unit": "°C",
  "min": 0,
  "max": 100,
  "threshold_min": 0,
  "threshold_max": 30,
  "chart_type": "disc"
},
{
  "name": "humidity",
  "unit": "%",
  "min": 0,
  "max": 150,
  "threshold_min": 0,
  "threshold_max": 60,
  "chart_type": "disc"
},
{
  "name": "co2",
  "unit": "ppm",
  "min": 0,
  "max": 100000,
  "limit": 10000,
  "lethal_limit": 50000,
  "threshold_min": 0,
  "threshold_max": 10000,
  "chart_type": "disc"
}
],
```

```
{
  "name": "soil_moisture",
  "unit": "%",
  "min": 0,
  "max": 100,
  "threshold_min": 30,
  "threshold_max": 70,
  "chart_type": "line"
}
]
```

Status thresholds

The different greenhouse and plant status and their respective threshold limits are defined in JSON configuration files.

```
{} greenhouse-status-thresholds.json × ... {} plant-status-thresholds.json ×
1 {
2   "NORMAL": {
3     "temperature": [0, 30],
4     "humidity": [0, 60],
5     "co2": [0, 10000],
6     "smoke": false,
7     "fine_dust": [0, 15]
8   },
9   "HIGH_CO2": {
10    "co2": [10000, 50000]
11  },
12  "EXTREME_CO2": {
13    "co2": [50000, 100000]
14  },
15  "SMOKE": {
16    "smoke": true
17  },
18  "FIRE": {
19    "temperature": [50, 100],
20    "co2": [10000, 50000],
21    "smoke": true
22  },
23  "HIGH_DUST": {
24    "fine_dust": [15, 50]
25  },
26  "EXTREME_DUST": {
27    "fine_dust": [50, 100]
28  },
29 }
```

```
1 {
2   "NORMAL": {
3     "soil_moisture": [40, 100]
4   },
5   "THIRSTY": {
6     "soil_moisture": [20, 40]
7   },
8   "DEHYDRATED": {
9     "soil_moisture": [0, 20]
10  }
11 }
```

Sensor data generation

The sensor readings are automatically generated using a Python script that simulates normal values or occasional values exceeding or falling below thresholds. This approach was chosen to best simulate real-world conditions.

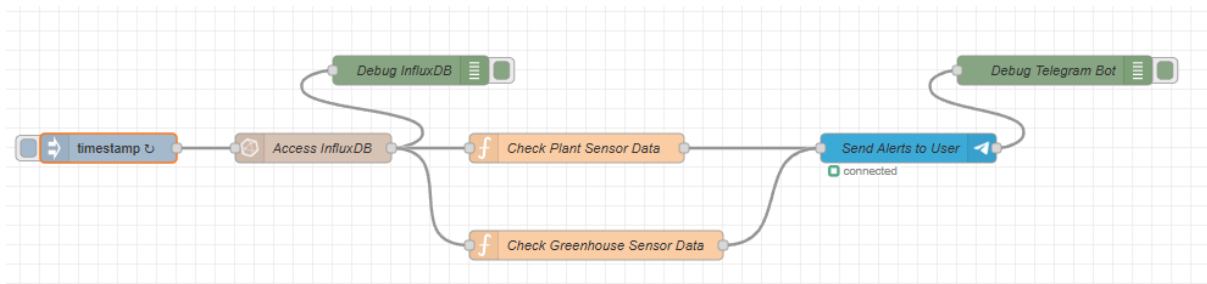
Data visualization

Sensor readings stored in the database can be displayed on two Grafana dashboards: one for greenhouse data and another for plant data. In the greenhouse data dashboard, the data can be filtered by greenhouse ID, while in the plant dashboard, it can be filtered by both greenhouse ID and plant ID. This enables users to focus more effectively on the incoming values for a specific plant or greenhouse.

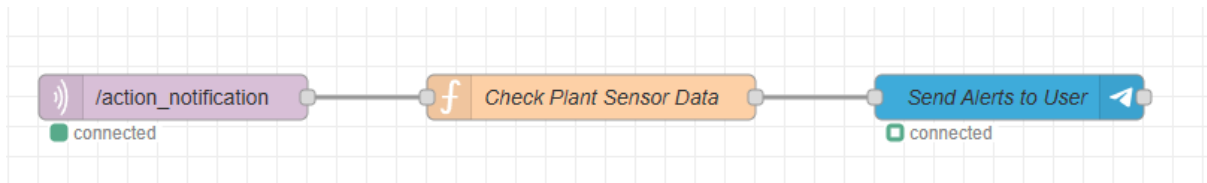


Alert notification

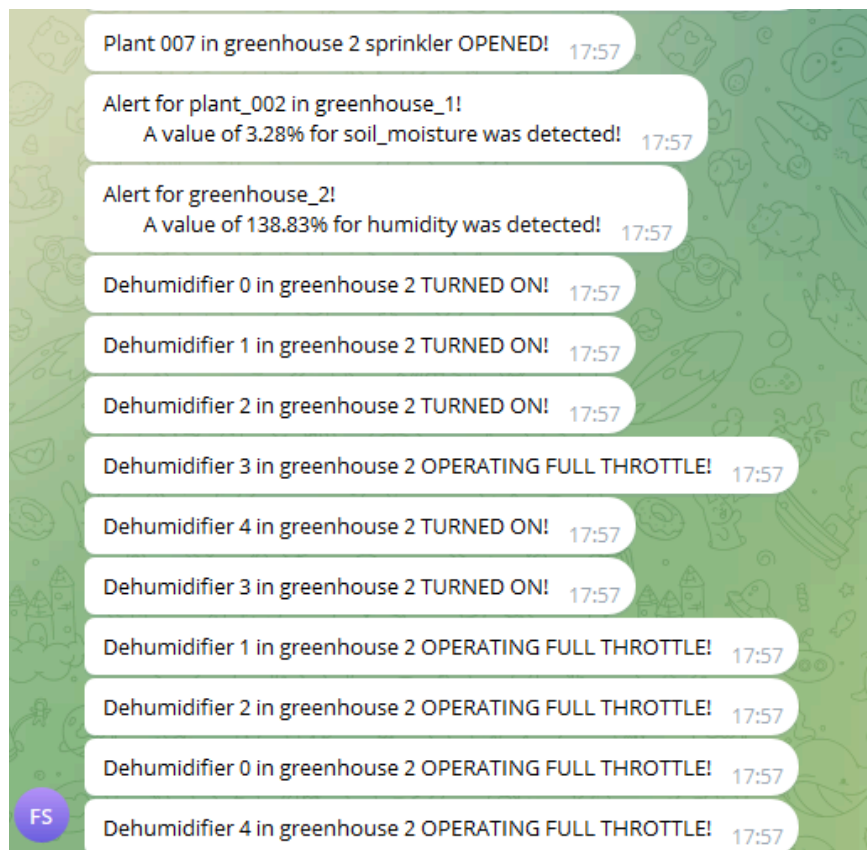
Every 10 seconds, the system checks the latest values stored in the database and sends an alert if a reading exceeds the safe limits set for that specific type of sensor.



Furthermore, a message is sent whenever an action is taken.



The alert is sent using Telegram as the messaging service.



Levels of Intervention

The system is designed to operate at three distinct levels of intervention based on the data coming from the different sensors placed in the environment and the gravity of the situation that may happen.

Following is the description of each intervention level explaining their purpose and their operation in different scenarios:

1. Normal Level

Description: In this state, all the information coming from the sensors, like humidity, CO₂, temperature and others, are inside the safety thresholds. The system continues to monitor the situation inside the greenhouses without activating any effectors or alarms.

2. Critical Level

Description: In this state, the system registers some data that is above the safety thresholds, or predicts that they will, and activates the correct actuators to try and bring the system back to the **Normal Level**. Moreover, the system will send some notifications to notify the farmers of the critical levels detected.

3. Alert Level

Description: In this state the system is detecting/predicting harmful levels that greatly exceed the pre-established thresholds. It activates all the necessary effectors at full power and sounds the alarms to alert the personnel present of the current dangers.

6. Adaptation goals

GOALS	DESCRIPTION	EVALUATION METRIC
Maintain the humidity level (%RH) within the limits of harmful values	The humidity concentration must remain inside the optimal interval	%RH < 60
Maintain the quality of CO ₂ below harmful levels	The concentration of CO ₂ must remain below the harmful threshold	Limit: CO ₂ <10000 ppm Lethal limit: CO ₂ <50000 ppm
Maintain the quantity of fine dust matter (PM) between harmful levels	The concentration of fine dust must remain below the harmful threshold	Limit: PM < 15 µg/m ³ Lethal limit: PM < 50 µg/m ³
Maintain the temperature level (°CT) of the greenhouse within the limits of harmful values	The temperature must remain inside the optimal interval	°CT < 30
Prevent/Extinguish Fire	Fire must be extinguished or avoided as soon as possible	No fire
Prevent/Clear Smoke	Smoke must be cleared or avoided as soon as possible	No smoke
Notify the farmers through the notification when an action was taken	Farmers must be notified when an action was taken in order to preserve plant health	%RH < 60 CO ₂ <10000 ppm PM < 15 µg/m ³ °CT < 30
Alert the farmers through the alarm when the parameters exceed the lethal safety limits	It is important to alert the farmers for preserving plant health	CO ₂ <50000 ppm PM < 50 µg/m ³

7. Autonomic Decision Process

The system uses a **Model-Based** approach to carry out predictions of what may happen in the monitored environment based on the gathered data and the one stored in the Knowledge component, and then define a sequence of actions to be taken.

This means that the autonomic manager progressively updates the system state with new predictions and reasoning based on the data coming from the sensors and the historical information that was stored over a period of time.

Specifically, the model uses a **linear regression** algorithm to predict what may be the next values for a specific sensor reading.

According to the generated predictions, the possible actions are, for example, activating the ventilation system actuators, or sending different alerts.

8. Technologies used

System components are wrapped into lightweight containers, which, as the name suggests, contain everything needed to run the application, and therefore allow not rely on what's installed on the host.

The system has been entirely implemented using the Python programming language and the following technologies.

Message Broker: MQTT



MQTT, which stands for Message Queuing Telemetry Transport, is a lightweight and scalable messaging protocol. The system makes use of the MQTT protocol to obtain data from sensors and enable communication between some of the components of the MAPE-K cycle. The data are sent through Python on the dedicated topic and subsequently processed by other components.

Database: InfluxDB



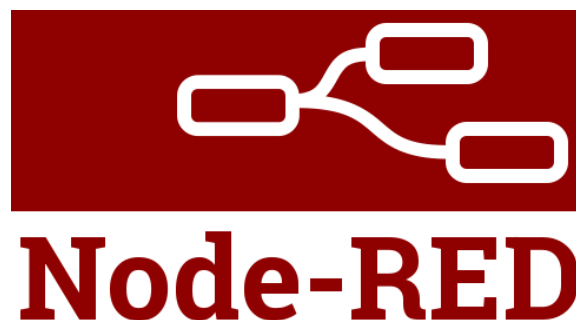
InfluxDB not only can handle large volumes of data from environmental sensors and other data sources but is also reliable and flexible, allowing efficient data analysis and management within our project, with the advantage of easy reads and writes.

User Interface: Grafana



Grafana is used primarily as a dashboard and was therefore used to visualize and understand the data. The main advantages of Grafana are the simple information management and data display options.

Node-RED



Node-RED is a flow-based programming tool that provides a visual representation of an application flow to wire together hardware devices, APIs, and online services in innovative ways. A Node-RED flow operates by passing messages between nodes.

The messages in Node-RED are simple JavaScript objects that can have any set of properties. The system makes use of this technology to notify the farmers when sensor measurements are above safe thresholds.

9. Functional Requirements

- **FR1: Greenhouse plant soil moisture level**
Description: The system monitors each greenhouse plant soil moisture level and sends the data to Knowledge
Type: Functional Requirement - Monitoring
Priority: **High**
- **FR2: Greenhouse CO2 level**
Description: The system monitors each greenhouse CO2 level and sends the data to Knowledge
Type: Functional Requirement - Monitoring
Priority: **High**
- **FR3: Greenhouse fine dust level**
Description: The system monitors the presence of fine dusts of each greenhouse and sends the data to Knowledge
Type: Functional Requirement - Monitoring
Priority: **Medium**
- **FR4: Greenhouse humidity level**
Description: The system monitors the humidity of each greenhouse and sends the data to Knowledge
Type: Functional Requirement - Monitoring
Priority: **Medium**
- **FR5: Greenhouse temperature level**
Description: The system monitors each greenhouse temperature and sends the data to Knowledge
Type: Functional Requirement - Monitoring
Priority: **High**
- **FR6: Greenhouse smoke detection**
Description: The system check the presence of a fire in each greenhouse and sends the data to Knowledge
Type: Functional Requirement - Monitoring
Priority: **High**
- **FR7: Data analysis**
Description: The system analyzes data stored in the Knowledge component
Type: Functional Requirement - Analysis
Priority: **High**

- **FR8: Plan definition**
Description: The system proactively plans an adaptation strategy to keep monitored parameters within safe thresholds
Type: Functional Requirement - Planning
Priority: **High**
- **FR9: Plan execution**
Description: The system communicates with the actuators to put Planner defined plans into motion
Type: Functional Requirement - Execution
Priority: **High**
- **FR10: Alert**
Description: The system alarms the farmers if the sensors readings are above safe thresholds
Type: Functional Requirement - Alarm
Priority: **High**
- **FR11: Notification**
Description: The system notifies the farmers when a particular action was taken like for example, irrigating a particular field or opening a window in a greenhouse
Type: Functional Requirement - Notification
Priority: **Low**

10. Non-Functional Requirements

- **NFR1: Scalability**
Description: The system must scale horizontally and vertically to accommodate an increasing number of sensors and users.
- **NFR2: Reliability**
Description: High system availability is critical, with redundant components to ensure continuous operation during peak loads or failures.
- **NFR3: Performance**
Description: The system must process and store data at an adequate rate with minimal latency.
- **NFR4: Usability**
Description: The user interface should be intuitive, providing clear visual feedback and easy navigation.
- **NFR5: Maintainability**
Description: The system architecture should allow easy maintenance, including updates and scaling, with minimal downtime.
- **NFR6: Energy Efficiency**
Description: The system must be optimized for energy consumption,

especially concerning the sensors deployed in the field and the central data processing units.