UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

Dipartimento di Ingegneria e Scienze
dell'Informazione e Matematica

Università degli Studi dell'Aquila

Smart Greenhouse System

2021 / 2022

Corso : Software Engineering for Internet of Things

Docente : *Prof. Davide Di Ruscio*

Project GitHub Repository: https://github.com/federix98/SE4IoT

Progetto realizzato da:

Rea Gianluca	Matr. ***278722***	gianluca.rea@student.univaq.it
Pietro Ciammaricone	Matr. ***274427***	pietro.ciammaricone@student.univaq.it
Federico Di Menna	Matr. ***275332***	federico.dimenna@student.univaq.it

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

Project Specification

A smart greenhouse can be defined as a greenhouse equipped with home automation technologies capable of allowing crop curators to remotely program, automate and manage the electronic devices present.

The first part of managing a greenhouse takes place through the control and measurement of various vital parameters for the cultivation of various vegetables. Since plants grown in greenhouses are normally out of season, the first parameter that is checked is the ambient temperature. Another important feature is humidity, which is measured in the ground and the air through specific sensors in the greenhouse. For the vitality of the plants, it is also necessary to constantly monitor the quality of the soil and that of the air by controlling respectively the PH of the soil and the amount of carbon dioxide inside the greenhouse. The impact of light on plants is also interesting, which must maintain the natural cyclical nature of day and night.

Measurement sensor list:

- Smart thermometer
- Air humidity sensor
- Ground humidity sensor
- Light sensor
- PH sensor
- Air quality sensor

These sensors make possible the control of the current situation in the greenhouse by his proprietary. Probably at some point, one of the sensors will inform us that one of the measures is out of the prefixed range and we will need to activate the actuators.

The system should also allow the user (greenhouse owner) to interact with the smart greenhouse system. So, firstly, the irrigation system should be able to be activated manually and automatically (using presents for specific cultivations).

For example, we can imagine that the irrigation must be activated when the soil humidity is above a certain value. Same for the temperature system and the light system. This means that the smart greenhouse system will have acting functions in addition to sensing ones.

The user will interact with the system through a dashboard deployed as a web app that contains all the parameters monitored in real-time and a section that will allow users to act on the system (consider OpenHab here).

Another requirement with lower priority than the previous ones is to insert in the dashboard the presets of greenhouse parameters settings to initialize the greenhouse for a specific plant. For sake of simplicity, we assume that a greenhouse will contain plants that can grow in similar settings. If we were to grow plants that require different settings, just consider the different greenhouse sections as different systems.

The project will be deployed in the cloud as SaaS to delegate to the provider the management of some non-functional aspects of the system such as the security, horizontal and vertical scalability of the (server-side) hardware and so on.

Functional Requirements

1 - Monitoring environment

Priority: **High**

Monitoring and showing in real-time the following environmental parameters:

- Air humidity
- Temperature
- Light
- Soil humidity
- PH Soil
- Air quality

2 - Warning Users

Priority: **High**

Warn the user when the parameters go beyond the established thresholds.

3 - Act manually on the greenhouse

Priority: **High**

In the dashboard, the user must be able to act manually on the following functions:

The temperature of the greenhouse

- Light
- Irrigation system (on/off)

4 - Presets for specific plants

Priority: **Low**

The system should provide presets for initializing automatically the settings of the greenhouse based on the plant.

Non Functional Requirements

- **High Performance Database** (es. Time Series)
    - The system is supposed to contain a large amount of data, and it should be able to perform aggregation functions and real time queries on them, so we need a database which is designed for these purposes.
- **Usability**
- **Reusability**
- **Cross-platform system**

SYSTEM ARCHITECTURE

Component Diagram (With technologies)

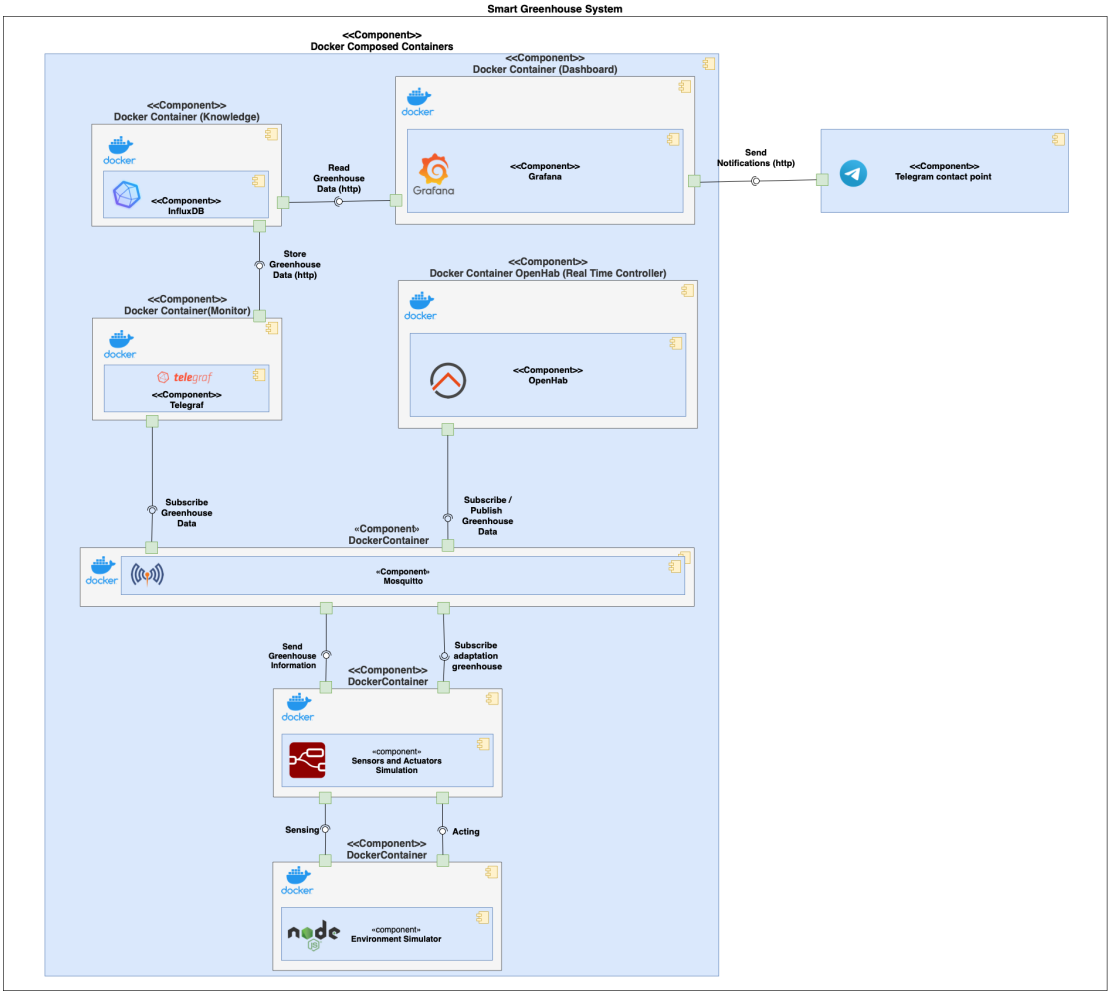


Figure 2: Smart Greenhouse System - Component Diagram

Greenhouse Environment Simulator

The simulation of the greenhouse systems environment has been made using a **node js web service** accessible via HTTP. In this way we can call the GET endpoints to retrieve the data (this simulates exactly the physical sensing act), instead with a POST request we can act on the environment (this simulates the physical acting on the environment).

```
GET http://localhost:8000/greenhouse/potato-cultivation/temperature/1

200 OK {"timeOfMeasurement":1645914608811,"data":21}
```

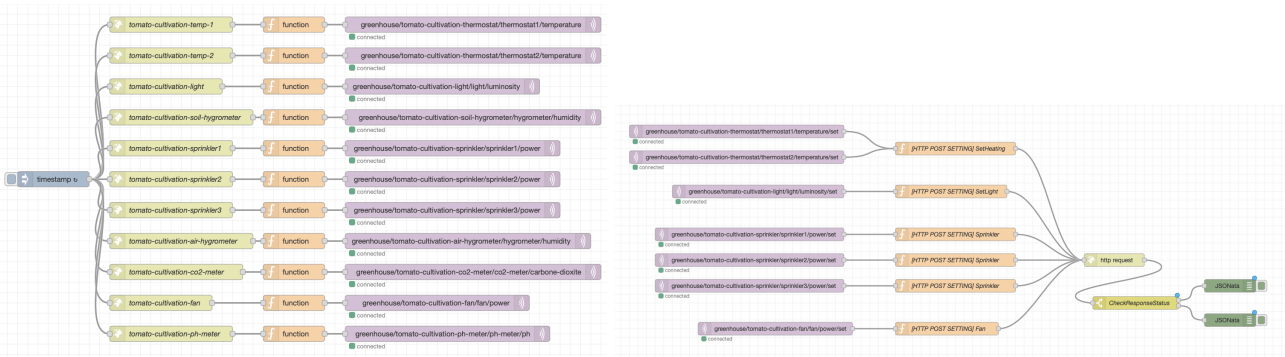
Sensors and Actuators

Sensors and Actuators have been simulated using **NodeRed** flows. For the sensing we have a time scheduled job which checks (make sensing on) the environment through HTTP GET requests and pushes the data on the related topic. The actuators subscribe and listen to the set topics and make HTTP POST requests to act on the environment.

Topics have been structured with the [The Homie convention]:

- `greenhouse/tomato-cultivation-thermostat/thermostat1/temperature`
- `greenhouse/tomato-cultivation-thermostat/thermostat1/temperature/set`
- `greenhouse/tomato-cultivation-thermostat/thermostat2/temperature`
- `greenhouse/tomato-cultivation-thermostat/thermostat2/temperature/set`
- `greenhouse/tomato-cultivation-light/light/luminosity`
- `greenhouse/tomato-cultivation-light/light/luminosity/set`
- `greenhouse/tomato-cultivation-soil-hygrometer/hygrometer/humidity`
- `greenhouse/tomato-cultivation-sprinkler/sprinkler1/power/set`

  ...



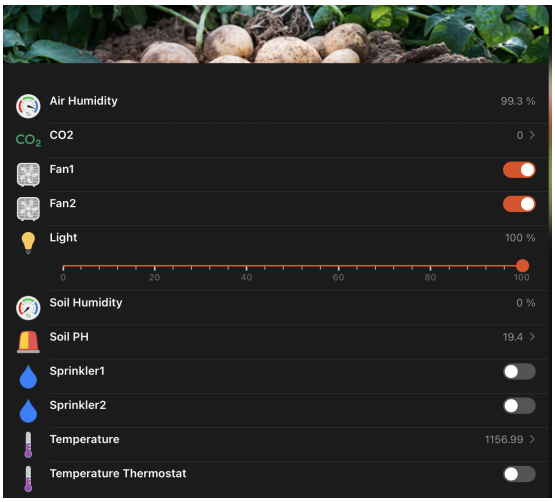
MQTT Broker

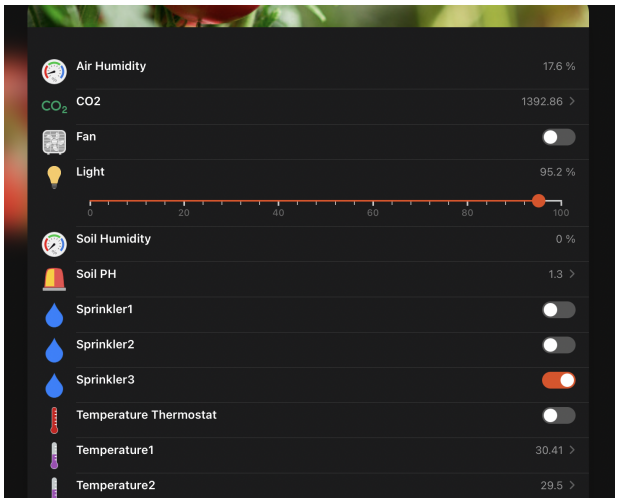
Sensors and actuators communicate with OpenHab and Telegraf using the MQTT protocol. For this purpose we used the containerized version of **Mosquitto MQTT Broker** in order to set up the connections.

Real Time display and interaction

The system allows the user to monitor and interact in real-time with the sensors and actuators (and so with the environment/greenhouse). To make it possible, we used **OpenHab**, in which we found different cultivations we can act on.

Clicking on the specific environment we can monitor the sensed data and switch on/off the actuators (heating on, light on, sprinkler...).



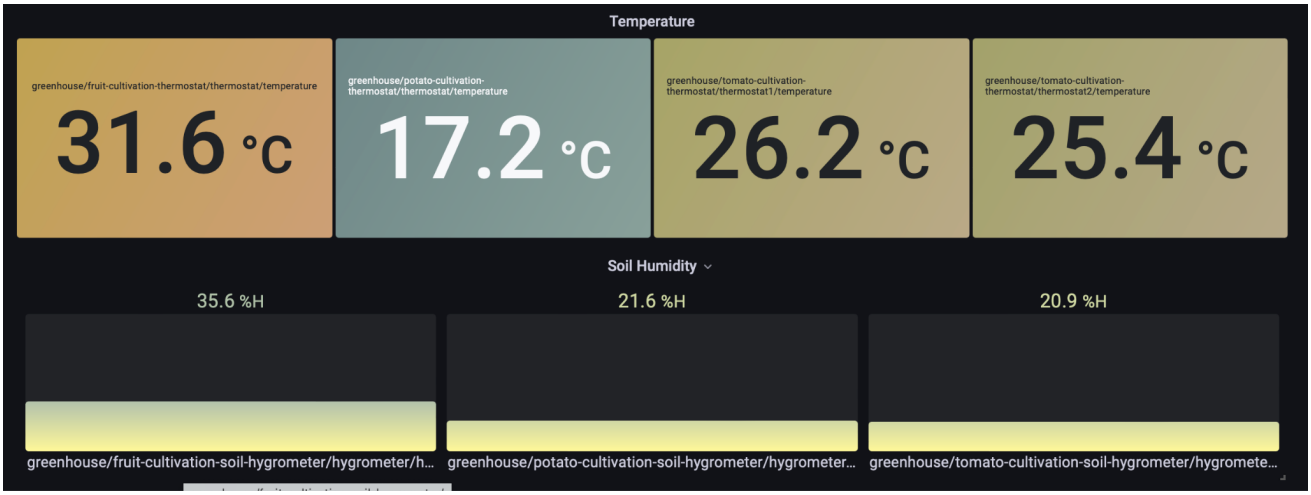


Dashboard for historic data

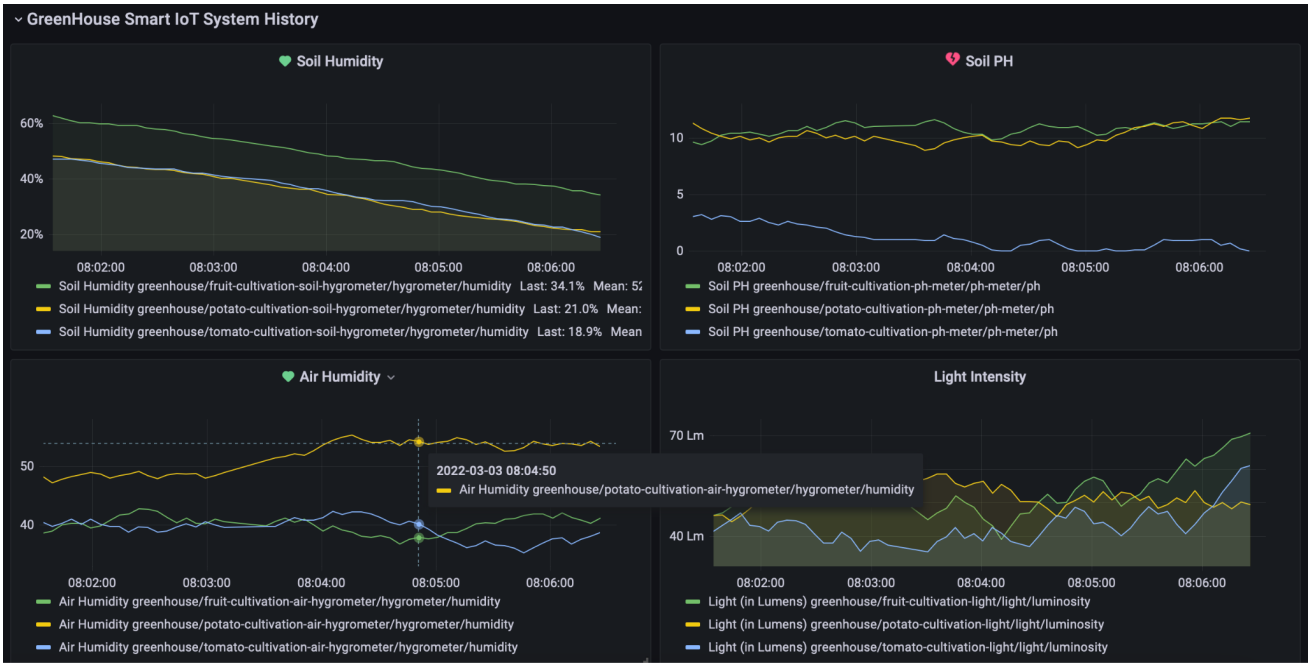
In addition to monitoring and acting on the real time data, it is worth keeping track of the historical data which can be used to make statistical analysis and this kind of thing. Furthermore, we want to notify the user in case the sensed data goes over specified thresholds.

For doing that we used the following stack:

- **Telegraf**: the containerized version of telegraf allows us to automatically monitoring all the topics under the greenhouse/ (greenhouse/# wildcard) and store the data on the configured database.
- **InfluxDB**: With this time series database we have an efficient approach to store time series sensed data. It also has many functions that allow you to aggregate data in different ways.
- **Grafana**: To complete the stack, we need a technology that allows us to graph the data entered in the database and support alert rules in case of exceeding the thresholds. The notification point we have chosen is **Telegram** given its simplicity and practicality in use.



Grafana - Real Time Dashboard



Grafana - Historical Analysis Dashboard

How Functional Requirements have been addressed

Functional Requirement #1 - Monitoring environment

The NodeRed sensor flows and OpenHab UI allows the user to actively monitor the greenhouse status in real time. The sensors are all the required ones.

Functional Requirement #2 - Warning Users

The user will be automatically notified in case of the sensed data exceeding the specified thresholds. This is possible thanks to the Grafana alert rules system: there exists a specific rule for each property.

Functional Requirement #3 - Act manually on the greenhouse

The user can act manually in real time on the greenhouse environment though the OpenHab UI. As illustrated above, the user can switch on / off the actuators in order to set the conditions he wants on the greenhouse.

Functional Requirement #4 - Presets for specific plants

This has been realized with OpenHab scripts. We have a script which automatically switches on / off the actuators based on the sensor's current status / value. In this way it's easy for the user to set the conditions most favorable to the desired crop.

TECHNOLOGIES