UNIVERSITÀ DI PISA

Artificial Intelligence and Data Engineering
Internet of Things

Smart GreenHouse

Tommaso Amarante, Edoardo Morucci

Github repository:
https://github.com/EdoardoMorucci/iot-SmartGreenhouse

Contents

Chapter 1

Introduction

Nowadays greenhouses are used to grow crops during all the year. It is important to take under constant control parameters like soil PH or environment temperature, for this reason IoT devices could be used since they do not need much energy, they could run for long period of time without specific maintenance.

Chapter 2

Architecture

The application is made up of two networks, one composed of IoT devices that use MQTT to retrieve data, the other network uses CoAP to communicate.
The MQTT network is deployed on remote testbeds whereas the CoAP network is simulated in Cooja. The application logic is performed by a collector that is implemented in Java, it receives data from both MQTT and Coap and stores them in an mySQL database. The collector is responsible for the proper operation of the whole system. For the visualization of data is carried out by a web interface based on Grafana.
The collector also exposes a simple command line interface to retrieve information from the sensors.

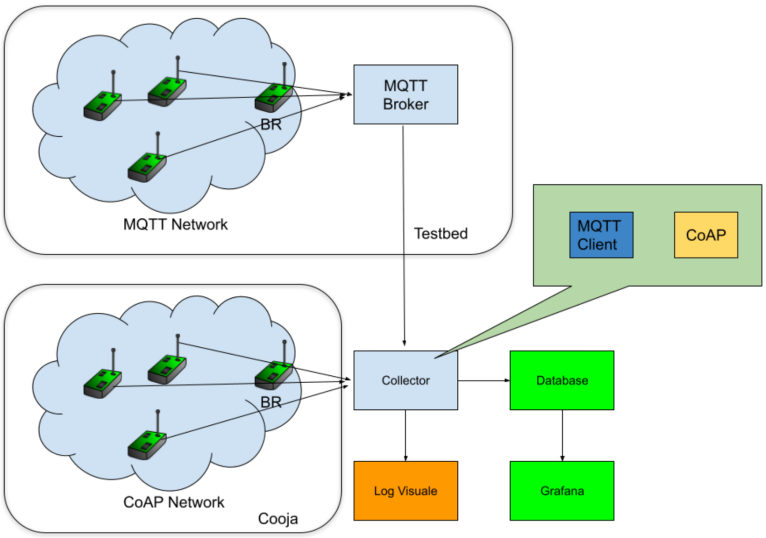

Figure 2.1: System architecture

2.0.1 MQTT Network

The MQTT network is responsible for the collection of values of temperature and soil humidity. The greenhouse has windows and lights that are connected with IoT devices that act as MQTT actuators to control the environmental temperature. There is also an irrigation system connected to a water valve that can be opened and closed based on the soil humidity percentage.

Temperature Sensor

The Temperature sensor senses the environmental temperature and sends it to the Java collector, the code flashed on the remote testbed simulates the changes in temperature based on the status of the windows and the lights inside the greenhouse. The initial temperature starts from 15°C and according to the application logic it decreases or increases.

Window and Light Actuators

Those actuators are controlled by the Java collector. In particular if the temperature senses is above the threshold of 25°C the window is opened and the temperature starts to decrease. When the temperature reaches the lower bound of 10°C the lights are switched on so that the temperature can increase.

Humidity Sensor

The Humidity sensor collects the value for the soil humidity percentage. Also in that case the code flashed on the remote testbed simulates the behaviour of a real soil. The initial percentage is 20% and according to the application logic it increases or decreases.

Valve Actuator

This actuator is controlled by the Java collector. In particular when the humidity percentage reaches the upper bound of 95% the valve is closed and the soil humidity starts to decrease, when the percentage is lower than 20% the valve is opened and the irrigation system starts to work.

2.0.2 CoAP Network

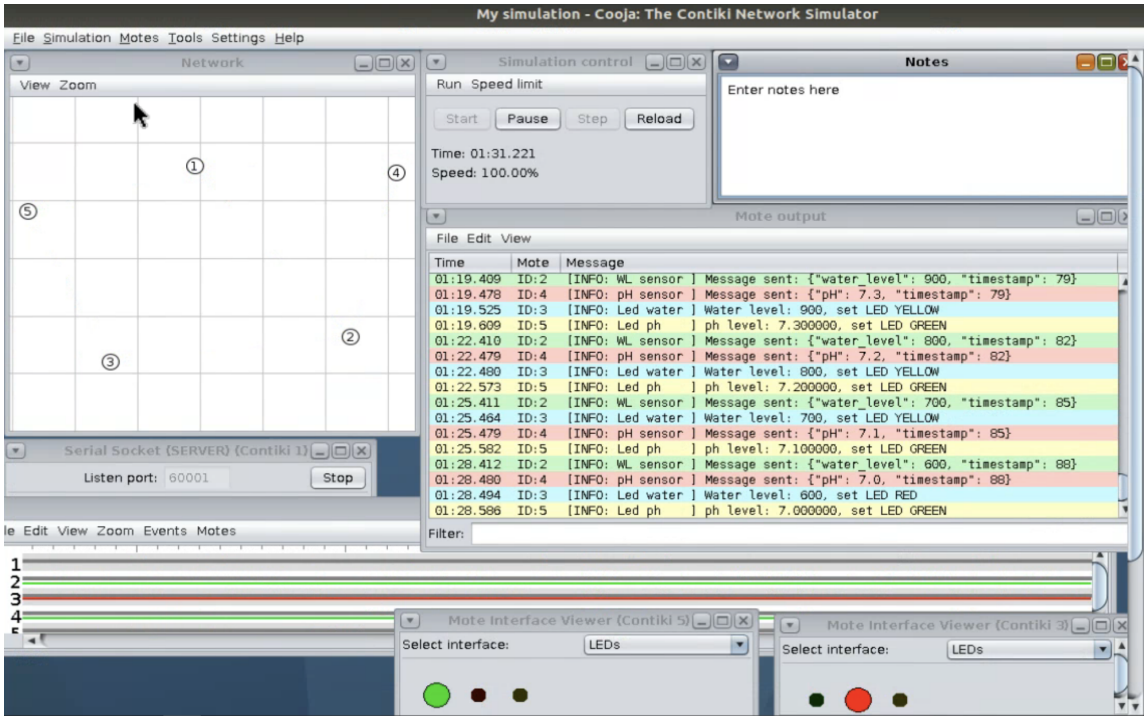

Figure 2.2: Coap Network simulated on Cooja

Furthermore, the implementation of IoT devices can automate some repetitive processes that arise in every greenhouse. Think of the daily need for water that plants need to survive, this process can be completely automated by placing a sensor in the ground and an actuator that will irrigate when necessary.

Another fundamental sensor that can help the growth and survival of plants is that of temperature, in fact, this greatly affects the growth process. We decided to implement a sensor of this type to guarantee a constant temperature inside the greenhouse. This is possible thanks to a Java application that receives the temperature values and sends them to two actuators, one to open the window and another to turn on a light and heat the room, in this way the temperature will be controlled and adjusted depending on the climate inside the greenhouse.

Finally, we decided to use a sensor inside the water boiler, this will notify the user via a series of LEDs, in this way the farmer will know when it is necessary to fill it, ensuring the plants their daily water supply.

Water level sensor and led

Since not all greenhouses are equipped with a well with ”guaranteed” running water, we thought that sometimes it is necessary to have a water boiler as a primary resource or as a backup in case of drought. Knowing the water level inside is certainly helpful, so we decided to insert a sensor for this purpose, a series of LEDs will then show the water level to the farmer.

The water level sensor exposes an observable resource called water_level_sensor, this measures (a dummy value in our simulation) the level, in liters, and makes it available to clients. The resource accepts GET requests and will notify all observers when it has new data available.

Another node in the network will receive the value via the PUT and will take care of setting the corresponding LED as follows:

- Less than 300 litres: RED led
- Between 300 and 1400 litres: YELLOW led
- Greater than 1400 litres: GREEN led

pH sensor and led

The pH of the water is another determining factor for the growth of some types of plants, keeping it under control allows you to know when to correct the value appropriately. Very abnormal pH values could affect an entire crop. Again we have an observable resource pH_sensor, which will send the respective values.

As in the previous case we have a led that indicates the value:

- Less than 6 or more than 8: RED led
- Between 6-6.5 and between 7.5-8: YELLOW led
- Between 6.5-7.5: GREEN led

2.1 MySQL

The values once received from the Java application are saved in a SQL database, in particular the database is called sensors and contains the following tables:

- temperatures (id, value, unit, timestamp)
- humidity (id, value, unit, timestamp)
- ph (id, value, timestamp)
- water level (id, value, unit, timestamp)

The id field is the primary key in each table and is automatically incremented for each data entered into each table.

2.2 User Interface

The following commands are available to the user through a CLI interface.

- **!help**: it prints the whole command list.
- **!current**<**value you want to sense**>: it gives you the current value received from the sensor.
- **!continuous**<**value you want to sense**>: it prints a continuous flow of data sesed by the remote sensor.
- **!waterLevelSensor** and **!pHSensor**: print the id address of the water level sensor or the pH sensor if they are present.
- **!printHystory**<**value you want**><**limit**><**offset**>: it gives you the measurement stored in the database based on the value of limit and offset.
- **!stop**: stops the continuous printing

2.3 Grafana

In order to visualize the data collected by all the sensors a web interface can be exploited. To per more precise we decided to visualize all the data sensed during the day. An example of the web interface is shown in the following image.

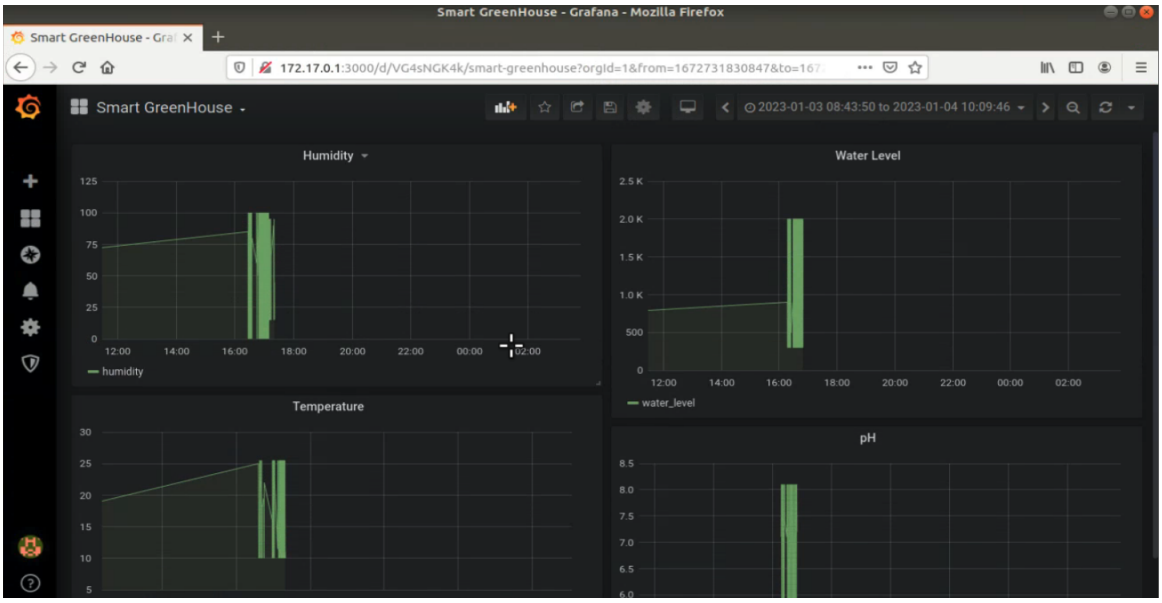

Figure 2.3: SmartGreenHouse dashboard on Grafana