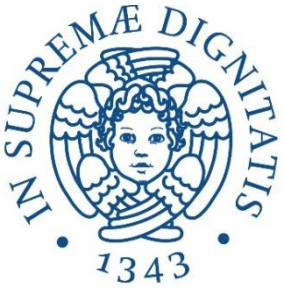


UNIVERSITÀ DI PISA

COMPUTER ENGINEERING

Internet of Things

PROJECT DOCUMENTATION

Design and development of *"Smart GreenHouse"*:

an IoT Telemetry and Control System

Student

Tommaso Giorgi

Academic Year 2021/2022

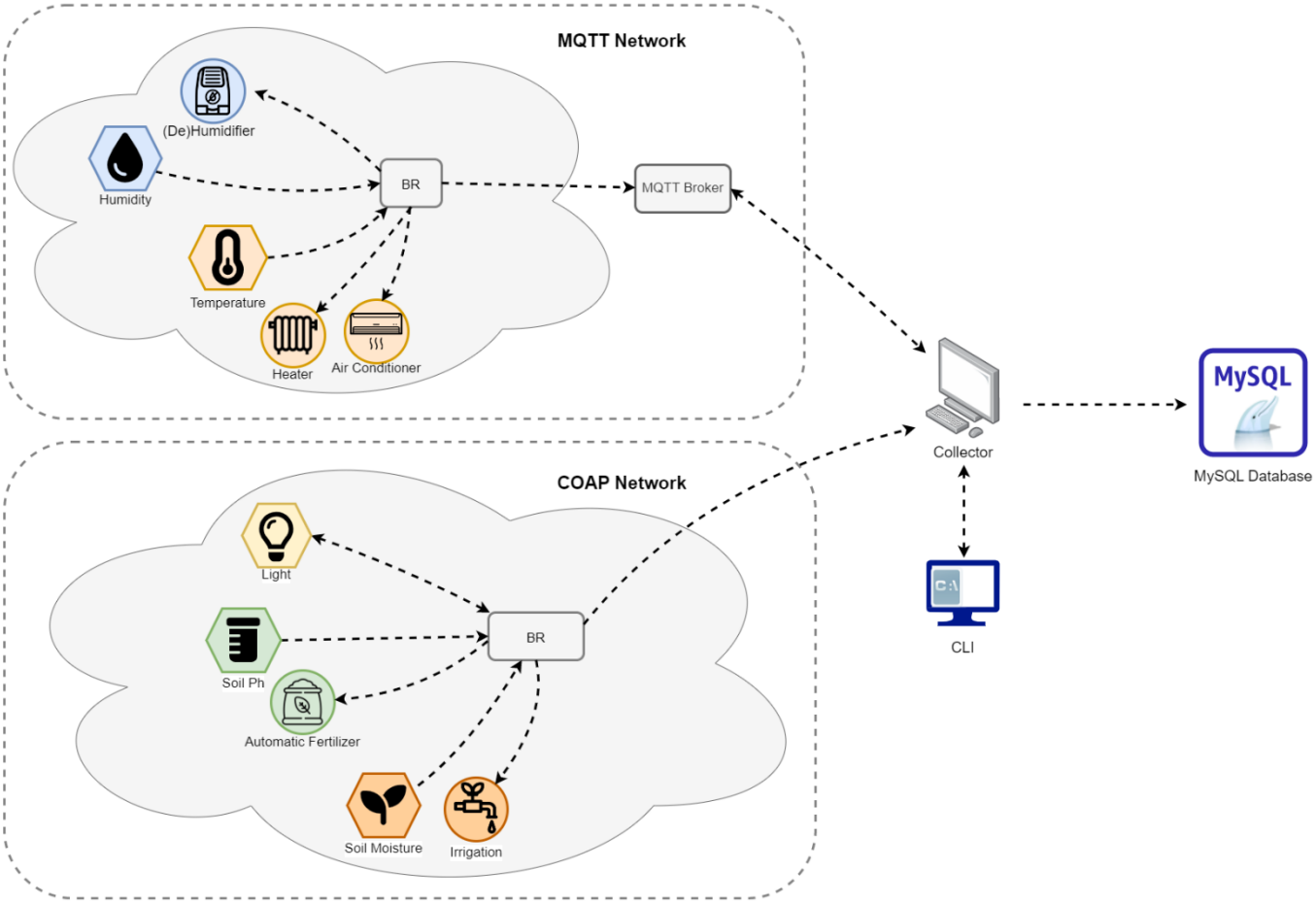
Sommario

Introduction

I decided to develop a smart greenhouse for the IoT Project. There are interesting and important sensors inside a smart greenhouse that we can use. Analyzing different values recorded by these sensors, the user can understand easily everything related to crops growing and react to them in order to change them in a proper way. In the next pages I will express every sensor used and its respective actuator in details starting from the system architecture.

Architecture

The system architecture is the following:



As we can see we have two networks, one MQTT Network that has been developed on real nodes (using the testbed) and a COAP Network simulated using Cooja. Each network has a border router that is used for the external communication. The MQTT broker instead is developed on the testbed using Mosquitto. Additional details about the sensors chosen and their respective actuators are expressed in details in the next paragraphs.

COAP Network

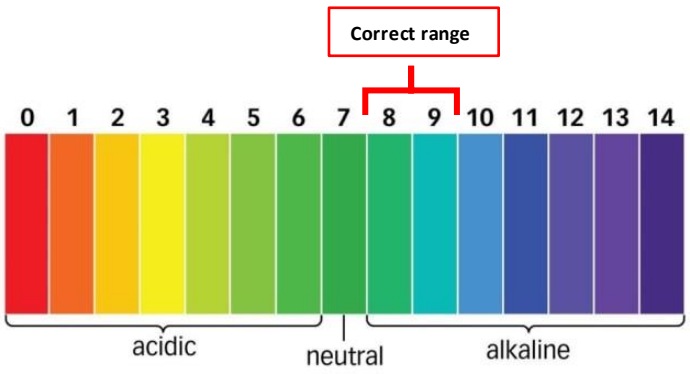
Light

A sensor is attached to all the lights inside the greenhouse. Depending on the different color associated to the leds in the cooja simulation we have different states for the lights in the greenhouse. When the led is red it means that the lights are turned off, when the led is yellow it means that the lights are on but with a low intensity and when led is green, the lights are on with the highest intensity. Lights are very important inside a greenhouse and the intensity is also foundamental, proper greenhouse lighting can maximize plant growth

and development while at the same time minimizing energy consumption. Depending on the plants inside the greenhouse and the season considered, the exposure of the plants to light becomes vital. Lights inside the greenhouse are managed manually by the operators using the CLI, specific commands are used to turn on or off the lights also specifying their intensity.

Soil PH Sensor

Another important parameter to consider inside a greenhouse is the PH of the soil. The effect of soil pH on crop growth is mainly manifested in the appearance and shape, material metabolism, growth and development, quality, and yield of plants. If the soil is too acid or alkaline, it will affect the root growth of plants to a certain extent, thereby affecting the normal growth and development of plants. Soil acid-base imbalance can also reduce the availability of nutrients in the soil and affect soil fertility. Therefore, it is necessary for the growth of crops to monitor the pH of the soil through the soil PH sensor, understand the soil quality, apply fertilizer reasonably, speed up soil improvement, and improve fertility. Paired with the Soil PH Sensor we have an **automatic fertilizer (actuator)** that is used to re-establish the soil ph to the correct ph range. I decided to use as correct ph range the one between 8 and 9. So when the ph of the soil is lower than a specific threshold decided in the configuration file or higher than a specific value, the automatic fertilizer must be turned on to higher or lower the ph of the soil in order to reach a value inside the correct ph range.



To simulate the real behaviour of the sensor, the soil ph changes over time increasing or decreasing randomly. When this value is below a target threshold, the fertilizer is automatically turned on by the application and stopped when the ph value has reached the correct range. The same happen when it exceedes a target value. Using the Java application, the user can check the current soil ph value and enable or disable manually the fertilizer, the user can also specify a **quantity** in the fertilizer. There are 3 modes that the user can select as quantity: 1,2 or 3. The quantity effects how fast the fertilizer will be in order to restore the correct ph value. With a quantity of 1, the fertilizer will lower or increase the ph of the soil **slowly** than the fertilizer turned on with a quantity of 2 or 3. When it is turned on or off by the application, the quantity chosen is 2 and cannot be modified unless the fertilizer is stopped and then re-enabled manually.

Leds are used in the simulation to check if the automatic fertilizer is on (green led) or off (red led).

Soil Moisture Sensor

This sensor is used to monitor the quantity of the water inside the soil. When the water content in the soil is relatively high, the water in the soil will enter the body of the plant through the membrane of the root system, accompanied by a large number of inorganic nutrients in the soil. However, when the water content in the soil is insufficient, the concentration in the root system of the plant is lower than the growth environment of the outside world. This will have an impact on growing needs of plants. Associated with this sensor we have is respective actuator, **an irrigation system**. The irrigation system can be manually turned on or off using the

CLI or it is turned on or off automatically by the application when the soil moisture sensor detects a very low (or high) value of water in the soil.

Values returned by the soil moisture sensor are between 1 and 2 and they are expressed as:

$$Soil\ moisture\ value\ = \frac{Wet\ Soil\ Weight}{Dry\ Soil\ Weight}$$

So, when the soil is full of water it has a weight doubled than the dry soil. Otherwise if it has no water, it will have a weight very similar to the dry soil and so a value near 1.

To simulate the real behaviour of the sensor, the soil moisture value increases linearly when the irrigation system is ON and decreases linearly over time when the irrigation system is off (it cannot go beyond 2 or below 1). Leds are used in the simulation to check if the irrigation system is on (green led) or off (red led).

MQTT Network

Humidity Sensor

An humidity sensor is used to monitor the humidity, a very high humidity promotes mold and pest problems in greenhouses. Associated to the humidity sensor, we have an actuator, a **(De)humidifier**.

To simulate the real behaviour of the sensor, the humidity percentage will increase or decrease over time randomly. If the humidifier is on, the humidity percentage will grow otherwise if the dehumidifier is on it will decrease (humidifier and dehumidifier cannot be on at the same time). A target range is set in the configuration file to automatically stop/start the actuator when target values are detected.

Testbed device (*/dev/ttyACM9*) is used to periodically register and send values from the sensor to the java application. From the java application, commands can be used to manually read a value,enable or disable the actuator.

Values recorded are written in the `humidity_s` topic, instead the topic used for enabling/disabling the actuator is called `humidity_act`.

Commands that can be received in the last topic are:

- **ON_HUM** : Enable the humidifier, the humidity will increase over time
- **ON_DEH** : Enable the dehumidifier, the humidity will decrease over time
- **OFF** : Actuator (humidifier or dehumidifer) is turned off, the humidity will increase/decrease randomly over time

Temperature Sensor

The biggest advantage of using greenhouses to grow crops is that they can provide ideal temperatures for plant growth and development. In greenhouse planting, the adjustment of indoor meteorological parameters is an important factor affecting crop growth and yield, and people need to adjust the indoor environment regularly. This sensor is very similar to the humidity one, it will record temperature values and will publish them into the `temperature` topic. Instead the `actuator` topic is used to send commands for the actuator that is a refrigerator or heater depending on if we want to decrease or increase the temperature.

Commands for the actuator are the following:

- **ON_HEA** : Enable the heater, the temperature will increase over time
- **ON_REF** : Enable the refrigerator, the temperature will decrease over time

- **OFF** : Actuator (heater or refrigerator) is turned off, the temperature will increase/decrease randomly over time (real behaviour simulation)

The system automatically disables and enables the actuator depending on the values recorded. If the temperature is below a target threshold specified inside the configuration file, the heater will be turned on. The same happen with the refrigerator when a too cold value is recorded.

Data Encoding

I decided to use **JSON** to encode data sent from the sensors to the Java Application. JSON is lighter (less verbose and more flexible) than XML and so it is the best choice for IoT nodes that must rely on a little batteries and power saving techniques. There are better approaches that can be used than JSON, for example non text-based formats, but are very recent in Contiki and not already completed.

Database

A very simple MySQL relational database is used to store values from the sensors. There are no dependencies between the tables. Tables used are the following:

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
node_id	int(11)	NO	
humidity_percentage	int(11)	NO	
timestamp	timestamp	NO	

Table 1 – humidity

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
node_id	int(11)	NO	
moisture_value	float	NO	
timestamp	timestamp	NO	

Table 2 - soilmoisture

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
node_id	int(11)	NO	
ph_value	float	NO	
timestamp	timestamp	NO	

Table 3 - soilph

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
node_id	int(11)	NO	
temperature_value	int(11)	NO	
timestamp	timestamp	NO	

Table 4 - temperature

Collector

I used Java to develop the collector. The collector is very important because it is the way in which a user can communicate with the iot sensors and receive or send information from/to them in an easy and readable way.

Available commands

The commands available in the CLI are:

- **!help <command>** : shows the details of a command
- **!get_light_intensity** : returns if the lights are turned on or off and also the intensity
- **!set_light_intensity <0,1 or 2>** : 0 to turn off lights, 1 for yellow leds, 2 for green leds
- **!set_fertilizer_quantity <0,1,2 or 3>** : 0 to turn off the fertilizer manually; 1,2 and 3 to turn on the fertilizer manually with that quantity
- **!get_soil_ph y** : returns the ph of the soil in the greenhouse
- **!get_fertilizer_status** : returns if the automatic fertilizer is turned on or off and also the quantity provided
- **!switch_irrigation <ON, OFF>** : switch on or off the irrigation system
- **!get_soilmoisture** : returns the soil moisture value in the greenhouse
- **!get_humidity** : returns the humidity percentage in the greenhouse
- **!set_humidity <low bound> <high bound>** : change the humidity threshold
- **!get_temperature** : returns the temperature value in the greenhouse
- **!set_temperature <low bound> <high bound>** : change the temperature threshold
- **!exit** : exit the program

Main Classes

- **SmartGreenhouseCollector**: main class, it shows all the available commands and makes them available to the users.
- CoapRegistrationServer:
- **CoapDevicesHandler:** It helps managing the different IoT devices and to redirect requests to the device classes
- **Light**: Class used for handling the lights inside the greenhouse (*COAP*)
- **SoilMoisture**: Class used for handling soil moisture in the greenhouse and the irrigation system (*COAP*)
- **SoilPH:** Class used for handling the soil ph in the greenhouse and the automatic fertilizer (*COAP*)
- **MQTTUtility**: Manages the connection with the MQTT Broker, it also has methods for handling publishes on different topics. Also , it automatically enables or disables actuators when the values recorded are too high or too low.
- **Humidity**: Class for the humidity sensor (*MQTT*)
- **Temperature**: Class for the temperature sensor (*MQTT*)
- **DBDriver**: Manages the connection with the MySQL database and has functions for inserting the values recorded from the sensor inside their respective tables.
- **ConfigurationParameters**: Class for handling the XML configuration file, this file is validated using the corresponding XML Schema.