UNIVERSITÀ DI PISA

Internet of Things project

<IoT for Smart-Agricolture>

Developed by:

- Giuseppe Aniello — #643032 — g.aniello@studenti.unipi.it
- Leonardo Bellizzi — #643019 — l.bellizzi@studenti.unipi.it

PROJECT OUTLINE

Project's goal is to build and set up an automatic system to protect grapes from bad weather conditions or may be used to save other agricultural products.

Application turns out to be useful in the countryside, in particular in Italy, where bad weather conditions are not so frequent and when they do occur they usually produce economic and environmental disasters.

To solve this problem, the physical part of the project is composed of two mechanical covers that are closed when bad weather comes in.

The two covers are side-located to grape marquees and both have a 90° opening radius.

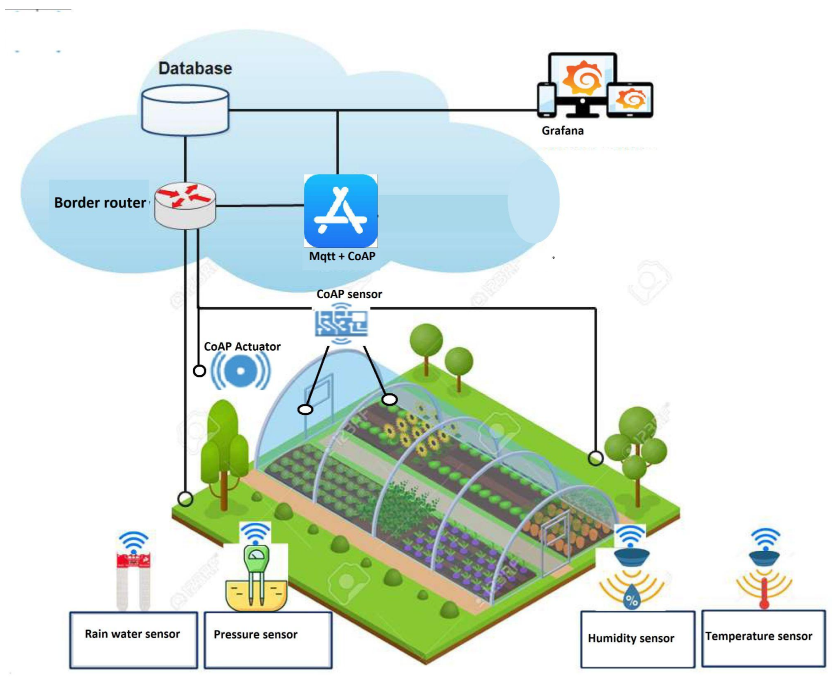
Application is composed by two main units:

- **MQTT sensors (3)** worry about ground real time conditions measuring temperature, rain water, humidity, pressure and actual weather;
- **COAP sensors (2)** interface with MQTT sensors to activate the two covers when bad conditions are detected. Based on how bad the weather is, covers can reach different opening degrees.

  When covers start to move an alarm is triggered. This alarm can be turned off with a button.

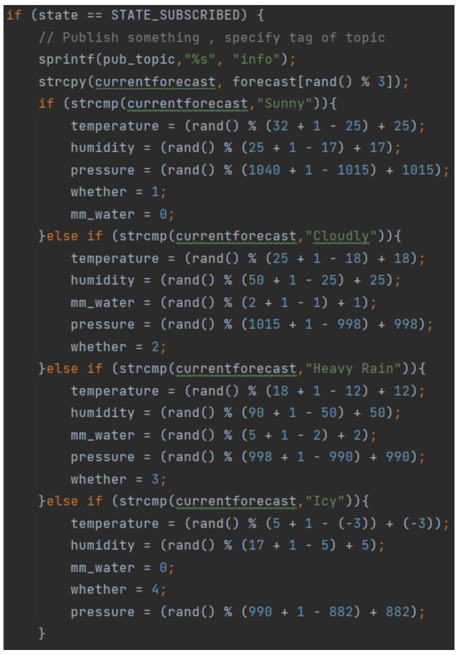
Sensors interact with a python application that works as a server collecting data and saving them into a **MySQL** database. Moreover, server deals with sensors sending response data.

Grafana, that is a visual-real time dashboard is then used to plot out what is saved on MySQL.



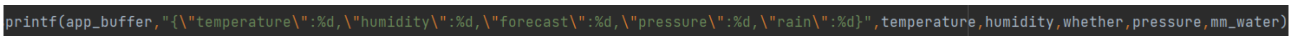
MQTT EXECUTION

MQTT sensors are used to monitor ground real time conditions measuring temperature, rain water, humidity, pressure and actual weather.
After the starting phase, they are connected to a **border router** that has been setup on a physical sensor.
In the communication a **message broker** is used. In particular a **Mosquittos** broker has been chosen and deployed on the VM**.**
If the connection has been made successfully a Green Led turn on for a short time period to signal the correct connection.
After the connection, data are sent in a JSON format on the topic "Info".
Real ground sensors are not available so data are randomly generated with some logic to produce data similar to real data.



```c
if (state == STATE_SUBSCRIBED) {
    // Publish something , specify tag of topic
    sprintf(pub_topic,"%s", "info");
    strcpy(currentforecast, forecast[rand() % 3]);
    if (strcmp(currentforecast,"Sunny")){
        temperature = (rand() % (32 + 1 - 25) + 25);
        humidity = (rand() % (25 + 1 - 17) + 17);
        pressure = (rand() % (1040 + 1 - 1015) + 1015);
        whether = 1;
        mm_water = 0;
    }else if (strcmp(currentforecast,"Cloudly")){
        temperature = (rand() % (25 + 1 - 18) + 18);
        humidity = (rand() % (50 + 1 - 25) + 25);
        mm_water = (rand() % (2 + 1 - 1) + 1);
        pressure = (rand() % (1015 + 1 - 998) + 998);
        whether = 2;
    }else if (strcmp(currentforecast,"Heavy Rain")){
        temperature = (rand() % (18 + 1 - 12) + 12);
        humidity = (rand() % (90 + 1 - 50) + 50);
        mm_water = (rand() % (5 + 1 - 2) + 2);
        pressure = (rand() % (998 + 1 - 990) + 990);
        whether = 3;
    }else if (strcmp(currentforecast,"Icy")){
        temperature = (rand() % (5 + 1 - (-3)) + (-3));
        humidity = (rand() % (17 + 1 - 5) + 5);
        mm_water = 0;
        whether = 4;
        pressure = (rand() % (990 + 1 - 882) + 882);
    }
```

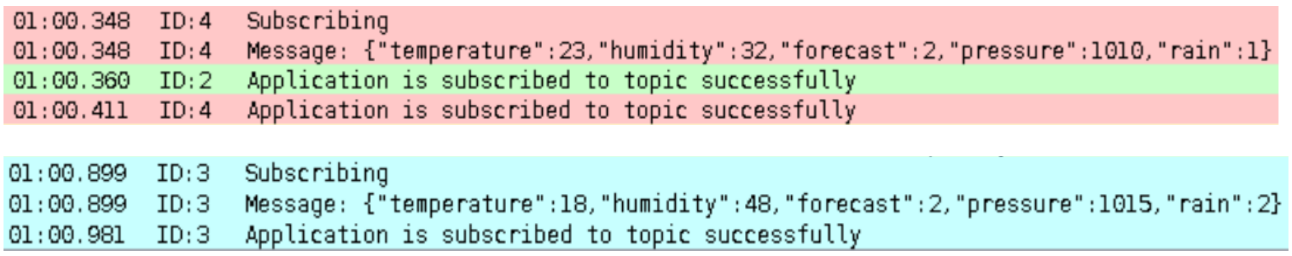
The message sent to Broker has the following structure:



```c
printf(app_buffer,"{\"temperature\":%d,\"humidity\":%d,\"forecast\":%d,\"pressure\":%d,\"rain\":%d}",temperature,humidity,whether,pressure,mm_water)
```

Once the app receives data via JSON from MQTT sensors, it will extract data and it will make a query to MySQL database to insert them into "**mqttsensors**" table. Inside the table a timestamp column is inserted to plot afterward graphs with Grafana.

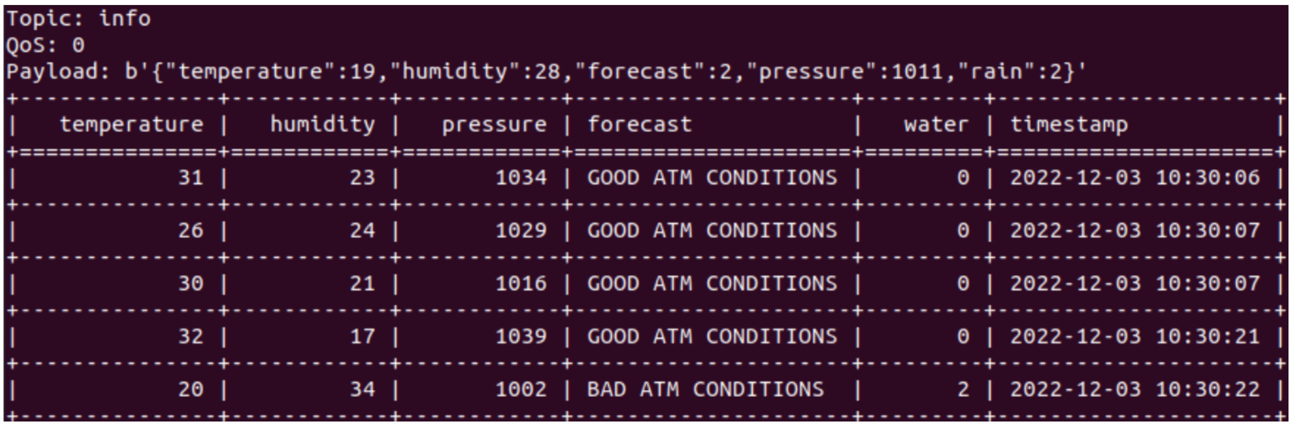
If data are correctly transmitted Cooja will display a window like the following one:



```
01:00.348  ID:4  Subscribing
01:00.348  ID:4  Message: {"temperature":23,"humidity":32,"forecast":2,"pressure":1010,"rain":1}
01:00.360  ID:2  Application is subscribed to topic successfully
01:00.411  ID:4  Application is subscribed to topic successfully

01:00.899  ID:3  Subscribing
01:00.899  ID:3  Message: {"temperature":18,"humidity":48,"forecast":2,"pressure":1015,"rain":2}
01:00.981  ID:3  Application is subscribed to topic successfully
```

Concerning MQTT, DB data will be displayed in this way:



```
Topic: info
QoS: 0
Payload: b'{"temperature":19,"humidity":28,"forecast":2,"pressure":1011,"rain":2}'
+-------------+-----------+----------+-----------------------+--------+---------------------+
|  temperature |  humidity |  pressure | forecast               |  water | timestamp            |
+=============+===========+==========+=======================+========+=====================+
|           31 |        23 |      1034 | GOOD ATM CONDITIONS |      0 | 2022-12-03 10:30:06 |
+-------------+-----------+----------+-----------------------+--------+---------------------+
|           26 |        24 |      1029 | GOOD ATM CONDITIONS |      0 | 2022-12-03 10:30:07 |
+-------------+-----------+----------+-----------------------+--------+---------------------+
|           30 |        21 |      1016 | GOOD ATM CONDITIONS |      0 | 2022-12-03 10:30:07 |
+-------------+-----------+----------+-----------------------+--------+---------------------+
|           32 |        17 |      1039 | GOOD ATM CONDITIONS |      0 | 2022-12-03 10:30:21 |
+-------------+-----------+----------+-----------------------+--------+---------------------+
|           20 |        34 |      1002 | BAD ATM CONDITIONS  |      2 | 2022-12-03 10:30:22 |
+-------------+-----------+----------+-----------------------+--------+---------------------+
```

COAP EXECUTION

For COAP part two sensors have been deployed in order to open or close the mechanical cover. To accomplish this aim we have two resources, a **motion resource** and an **alert resource**. Motion resource generates data according to a prefixed logic since physical sensors are not available.

Motion sensors are triggers of the physical cover that it will close with a variable degree radius and an alarm will be triggered.

Sensor client was done by splitting up threads for recording and detecting the state change to avoid threads synchronization.

If registration to Border Router has been successful, LEDs are turned on for a short period of time to indicate “Success”. After registration to Python Server, sensors are instantiated with resource class and observation of the sensor “Motion” begins.

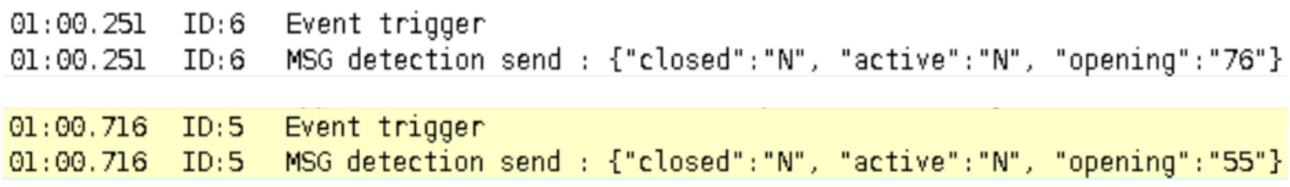
When resources change their state, a query is made to the MySQL database of the sensors, and a POST is made to set opening degree radius.

If the system is active, cover is activated, alarm is triggered and covers start to close with different radius degrees.

Even for this part, JSON messages are used.

To disable/enable the alarm and motion detection in manual mode, a button on one of the two sensors can be pressed.

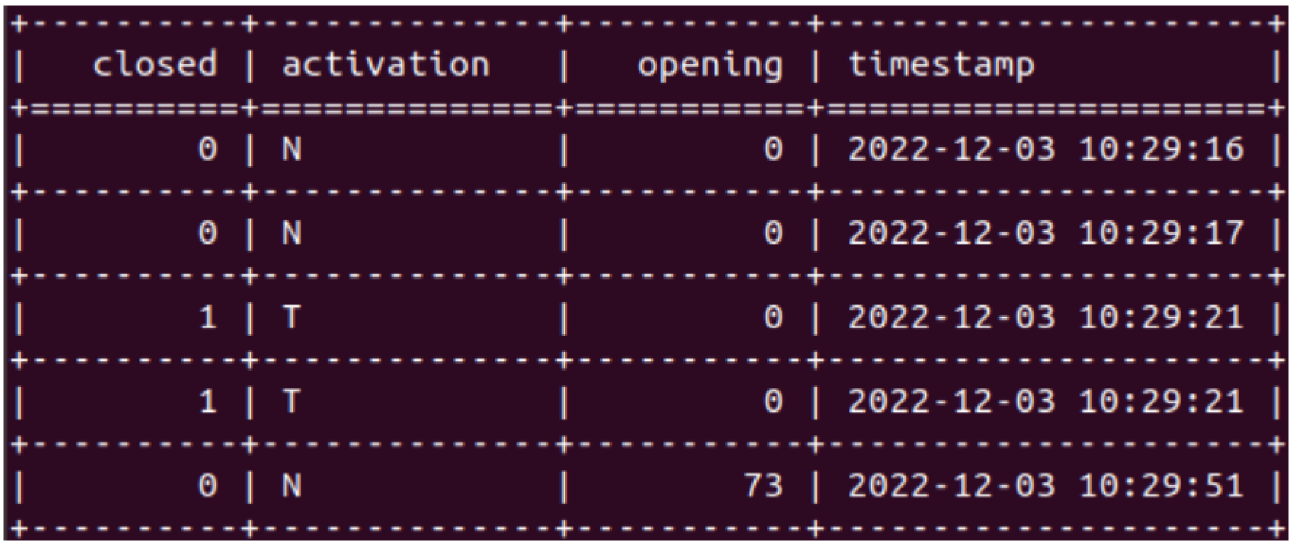
If data are correctly transmitted Cooja will display a window like the following one:



```
01:00.251   ID:6   Event trigger
01:00.251   ID:6   MSG detection send : {"closed":"N", "active":"N", "opening":"76"}

01:00.716   ID:5   Event trigger
01:00.716   ID:5   MSG detection send : {"closed":"N", "active":"N", "opening":"55"}
```

Concerning COAP, DB data will be displayed in this way:



```
+----------+-------------+----------+---------------------+
|   closed | activation  |  opening | timestamp           |
+==========+=============+==========+=====================+
|        0 | N           |        0 | 2022-12-03 10:29:16 |
+----------+-------------+----------+---------------------+
|        0 | N           |        0 | 2022-12-03 10:29:17 |
+----------+-------------+----------+---------------------+
|        1 | T           |        0 | 2022-12-03 10:29:21 |
+----------+-------------+----------+---------------------+
|        1 | T           |        0 | 2022-12-03 10:29:21 |
+----------+-------------+----------+---------------------+
|        0 | N           |       73 | 2022-12-03 10:29:51 |
+----------+-------------+----------+---------------------+
```

MSQL

“pymsql” allowed to link python and Mysql. MySQL has the following access data:

- **host**: “localhost”;
- **user**: “root”,
- **password:** “PASSWORD”
- **database**: “collector”

With the following two tables:

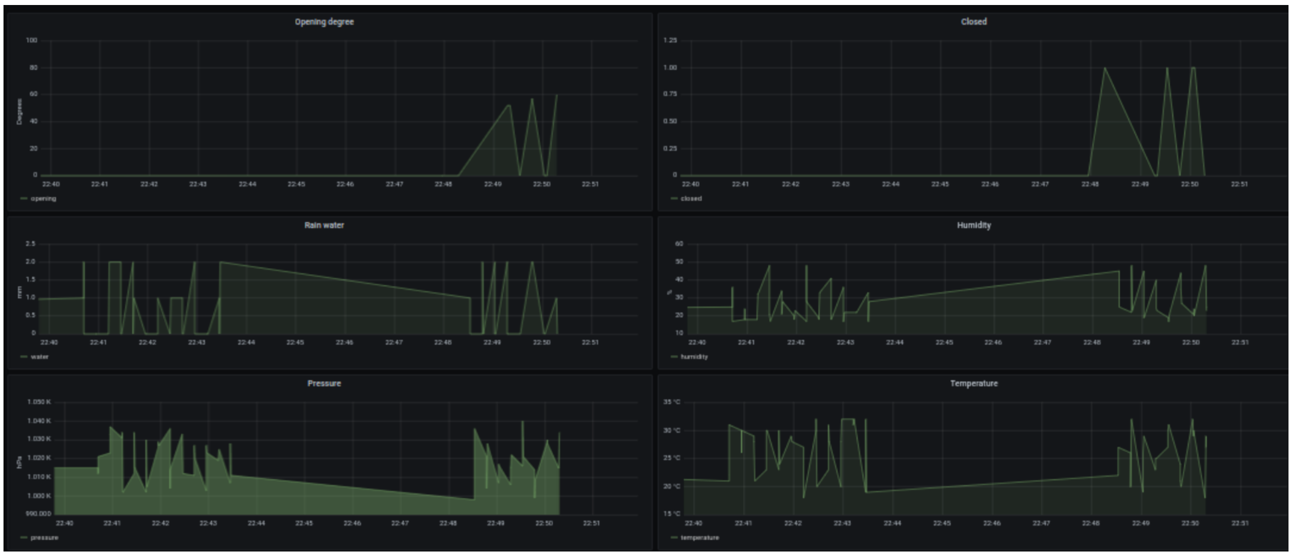
- **mqttsensors** to store ground data;
- **coapsensorsmotion** to store mechanical cover degree status over time

GRAFANA

Grafana was used to plot and display what is stored in "**collector"** in MySQL DB. On VM, Chromium has been installed, to access Grafana dashboard.

Dashboard auto update is set to **30** seconds.

Here some Grafana charts:



In the first row **CoAP sensors** graphs are displayed, while the others belong to **MQTT sensors**.