UNIVERSITÀ DI PISA

Msc. in Artificial Intelligence and Data Engineering

Smart Wine Cellar

Internet of Things project

Leonardo Turchetti
Ludovica Cocchella

Academic Year 2021/2022

Contents

1 Introduction

Internet of things is revolutionizing every industry in the world today and the wine industry is one of them.

Wine quality management is extremely important to create ideal conditions for storing and aging wine. In particular it is fundamental to keep under constant control environmental conditions as temperatures, humidity, quality of the air etc. Temperature is particularly important as even slight fluctuations impact the oxidation of the wine, which strongly affects the quality, and the level of humidity also has a significant effect on their longevity. Obviously it almost impossible to monitor all these values manually in real time to be able to act in a timely manner to safeguard the quality of the product. It is here that the internet of things comes to our aid.

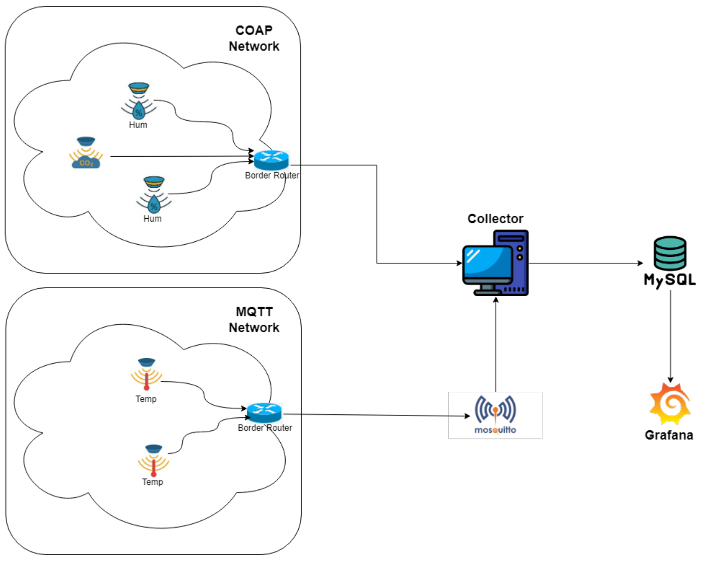
The context of our project is a wine cellar. We have integrated sensors for temperature sensing that integrate actuators to regulate the latter, sensors for humidity sensing that integrate actuators to initiate dehumidifier systems and then CO2 sensors that integrate actuators to initiate ventilation system and in certain condition to start an alarm.

The goal of this project is to design and implement an IoT control system for wine cellar in order to monitor independently temperature level, humidity level and air quality of the cellar and if necessary to act immediately to restore ideal conditions.

2 Design and Implementation

2.1 Overview

The system architecture is as shown in the following image



The system is composed by two different network of IoT devices. This two network are different because they use two different type of application protocol to communicate: humidity sensors and CO2 sensor exploit the CoAP application protocol while the temperature sensors use the MQTT application protocol. The system also includes a collector, a SQL database, and a Grafana web interface to view the data.

All these devices are IoT devices, and they are equipped with the Contiki-NG operating system. This is a Low Power and Lossy Networks (LLNs) using the IEEE 802.15.4 standard and the IPv6 protocol. Also, they exploit the RPL protocol which enables the multi-hop communication within the network. Finally, with the help of a border router, it is possible to send the data out of the LLN. These IoT devices exchange their data with a collector. So, the job of the collector is to collect the data coming from the IoT devices, and storing them in a database and also it can perform some task like sending actuating commands to the IoT devices.

2.2 Temperature Sensors

To manage the temperature level are needed two temperature sensors and two conditioner. One sensor and one actuator are dedicated to monitor and modifying the temperature level of red wine and the other do the same work but for white wine. The conditioner is used when the temperature level is out of the acceptable range in order to reestablish acceptable level.

These devices are IoT devices which exploit the MQTT application protocol. They act as MQTT client, subscribing and publishing messages to an MQTT Broker. The temperature sensors dedicated to red wine publish the sample detected to topic "current_temperature_red" and wait for the command on the topic "temperature_regulator_red" while the other sensor publish to topic "current_temperature_white" and wait on topic "temperature_regulator_white". The message publish by the sensor to the topic has the following format:

```
    {"temperature": [value]}
```

while the message published by the collector to the topic can have the following format:

```
    {"ON"} to turn on the conditioner
    {"OFF"} to turn off the conditioner
```

The sensor sends to the collector the current temperature. The collector firstly check if the temperature is outside the acceptable range and will turn on or off the conditioner or will do any action based on the detected temperature.

2.3 Humidity Sensors

The humidity sensors monitor the humidity level inside the cellar. The two sensors are dedicated one for red wine and the other for white wine. These devices also act as an actuator of a dehumidifier which has the task of restoring the optimal conditions of the humidity level of cellar when the humidity level is outside the acceptable range.

These IoT devices exploiting CoAP as application protocol, acting both as client, since they continuously report their readings and as server since they also receive commands from the collector.

The two devices act as a CoAP server by exposing the observable resources which are the following: *humidity_redwine sensor* for red wine and *humidity_whitewine sensor* for white wine. The resources just mentioned allow the collector to become an observer and receive status updates through a GET request in json format as follows:

```
    {"concentration": [value]}
```

The actuators can be trigger by a PUT request from the collector which the payload of the request must contain ”mode=ON” or ”mode=OFF”.

2.4 CO2 Sensor

The CO2 sensor control air quality inside the cellar. This device also act as an actuator of a ventilation system which contains an alarm. The ventilation comes into action when the level of CO2 is greater or equal than a certain value and turns off when it returns less while the alarm is activated when the level of CO2 is greater or equal than the maximum allowed and turn off it when it returns acceptable.

This sensor also use the CoAP protocol for communication with the application. They act as both client and server by exposing *Co2 sensor*. The collector acts as a server by observing the resource and receives through GET requests periodically the CO2 level through a json with the following format:

```
    {"concentration": [value]}
```

The actuators can be trigger by a PUT request from the controller which the payload contain one of the following expressions: ”mode=ON1”, ”mode=ON2” or ”mode=OFF”. If the actuator receive ”mode=ON1” the ventilation system is activated and the alarm is turned off while if it receive ”mode=ON2” both the ventilation system and the alarm are activated then with ”mode=OFF” both the ventilation system and the alarm are deactivated.

2.5 Collector

The collector is responsible for receiving data from sensors whether these use CoAP or MQTT thanks to respectively *Californium* and *Paho* libraries. It communicates with the database to save the sample received and take decisions activating the actuators whenever the environmental values are not within the established range.

Before starting receiving any updates from CoAP, IoT devices have to register to the collector. After the registration, the collector sets up the observe resources to receive updates from the device; the observe resources are instantiated only once and they are turn down only if no data arrived since a certain amount of time, in this way the Collector can independently understand when a resource is no longer active. After that the collector start to analyses the data of sensors and if the value are not acceptable start to make the action mentioned in the paragraph concerning the sensor.

It also implements a CLI to make it easier for the user to manage. The functions made available to the user through the CLI are the following:

```
!exit: exit the program
!commands: list possible commands
!checkTemp: get current temperature
!checkHumidity [type]: get current humidity level
!checkCo2: get current Co2 level
!changeRangeTemp [low] [up]: set new acceptable range for the ...
   temperature
!changeRangeHum [low] [up]: set new acceptable range for the humidity
!changeMaxCo2 [new_value]: set new max value for the Co2 level
!stopAlarm: stop Co2 alarm
```

2.5.1 Package Structure

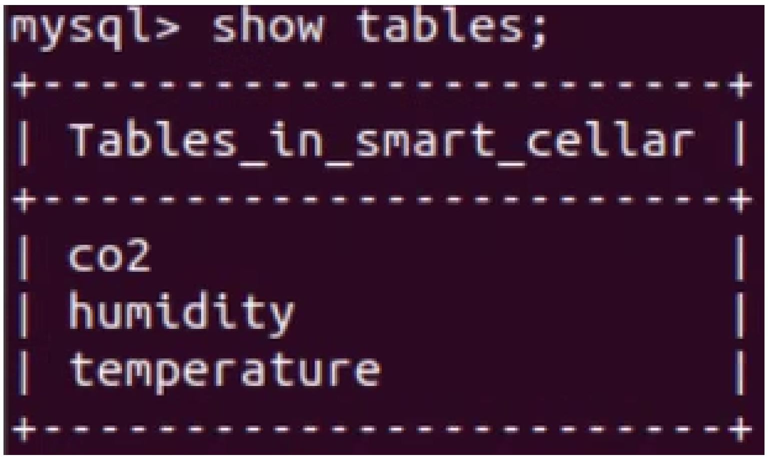
The collector of the Smart Cellar System is composed of the following packages and classes:

- **Collector.java**: it contains the main function in which are defined the available commands.
- **CoapNetworkHandler.java**: it contains the functions to handle the CoAP sensors.
- **RegistrationServer.java**: it contains the function to handle the registration of the CoAP sensors to the java application.
- **MqttClientCollector.java**: it contains the function to handle MQTT sensors.
- **SmartCellarDB.java**: it contains the function to upload the sample detected into the database.

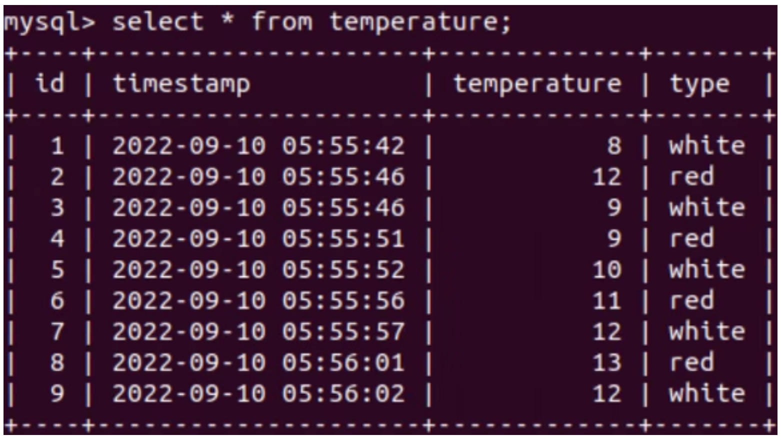
2.5.2 Database

The data captured by the sensors are stored in a MySQL database by the collector in order to be able to visualize them in Grafana.

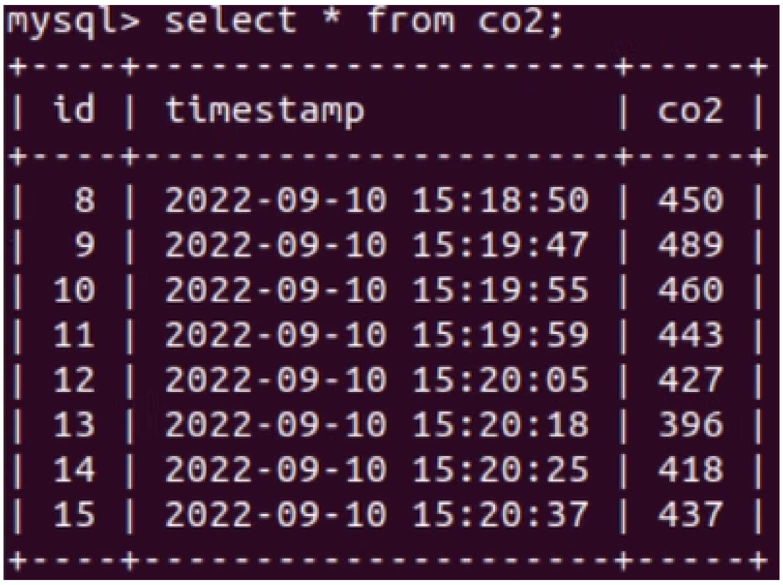
In particular, three tables were defined, one for each type of IoT device present in the system: temperature, humidity, CO2. The database has tables organized as following:



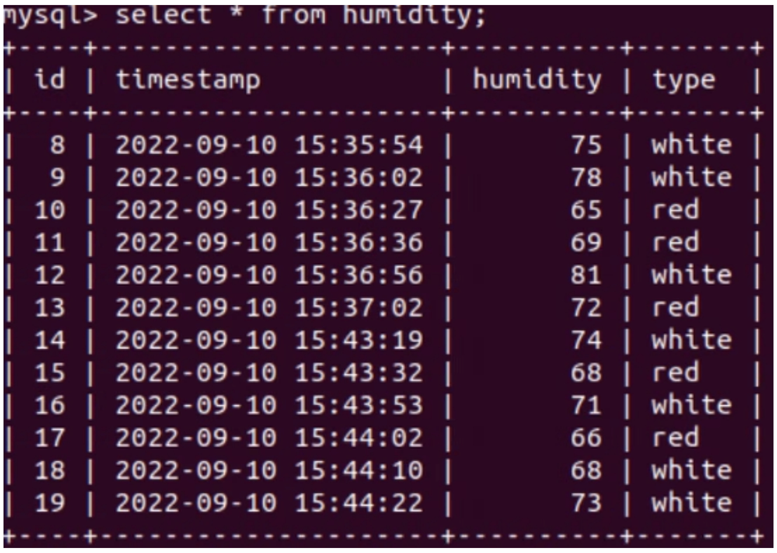
```
mysql> show tables;
+-------------------------+
| Tables_in_smart_cellar |
+-------------------------+
| co2                     |
| humidity                |
| temperature             |
+-------------------------+
```



```
mysql> select * from temperature;
+----+---------------------+-------------+-------+
| id | timestamp           | temperature | type  |
+----+---------------------+-------------+-------+
|  1 | 2022-09-10 05:55:42 |           8 | white |
|  2 | 2022-09-10 05:55:46 |          12 | red   |
|  3 | 2022-09-10 05:55:46 |           9 | white |
|  4 | 2022-09-10 05:55:51 |           9 | red   |
|  5 | 2022-09-10 05:55:52 |          10 | white |
|  6 | 2022-09-10 05:55:56 |          11 | red   |
|  7 | 2022-09-10 05:55:57 |          12 | white |
|  8 | 2022-09-10 05:56:01 |          13 | red   |
|  9 | 2022-09-10 05:56:02 |          12 | white |
+----+---------------------+-------------+-------+
```



```
mysql> select * from co2;
+----+---------------------+-----+
| id | timestamp           | co2 |
+----+---------------------+-----+
|  8 | 2022-09-10 15:18:50 | 450 |
|  9 | 2022-09-10 15:19:47 | 489 |
| 10 | 2022-09-10 15:19:55 | 460 |
| 11 | 2022-09-10 15:19:59 | 443 |
| 12 | 2022-09-10 15:20:05 | 427 |
| 13 | 2022-09-10 15:20:18 | 396 |
| 14 | 2022-09-10 15:20:25 | 418 |
| 15 | 2022-09-10 15:20:37 | 437 |
+----+---------------------+-----+
```



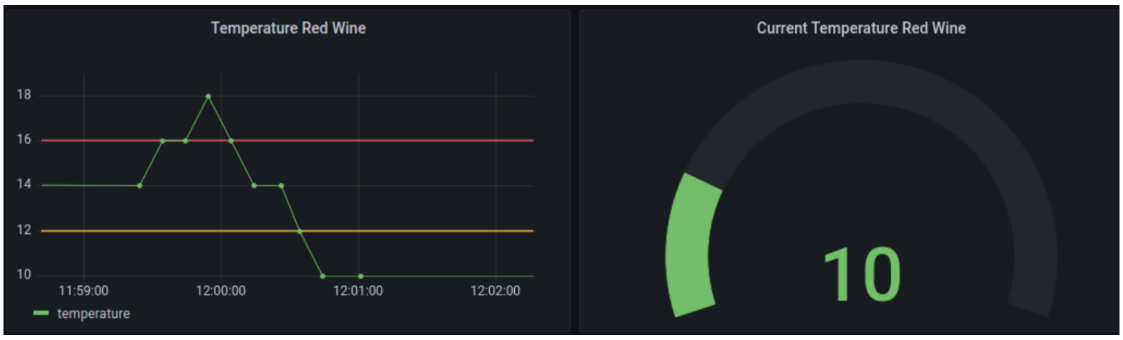
```
mysql> select * from humidity;
+----+---------------------+----------+-------+
| id | timestamp           | humidity | type  |
+----+---------------------+----------+-------+
|  8 | 2022-09-10 15:35:54 |       75 | white |
|  9 | 2022-09-10 15:36:02 |       78 | white |
| 10 | 2022-09-10 15:36:27 |       65 | red   |
| 11 | 2022-09-10 15:36:36 |       69 | red   |
| 12 | 2022-09-10 15:36:56 |       81 | white |
| 13 | 2022-09-10 15:37:02 |       72 | red   |
| 14 | 2022-09-10 15:43:19 |       74 | white |
| 15 | 2022-09-10 15:43:32 |       68 | red   |
| 16 | 2022-09-10 15:43:53 |       71 | white |
| 17 | 2022-09-10 15:44:02 |       66 | red   |
| 18 | 2022-09-10 15:44:10 |       68 | white |
| 19 | 2022-09-10 15:44:22 |       73 | white |
+----+---------------------+----------+-------+
```

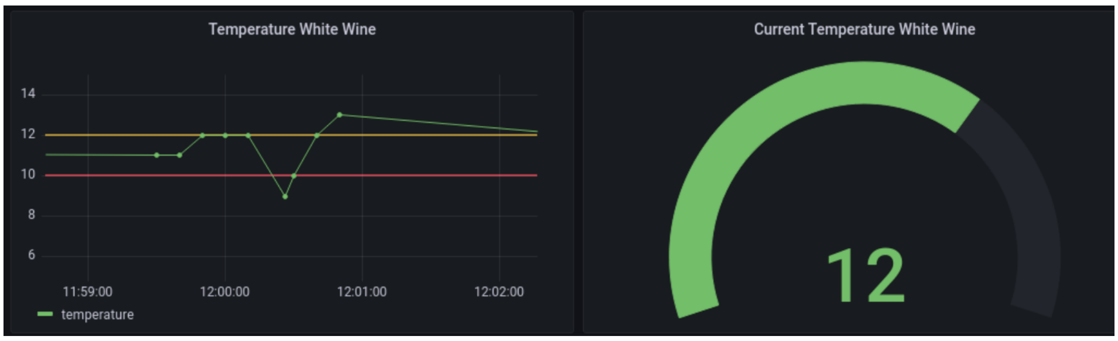
2.6 Grafana Dashboard

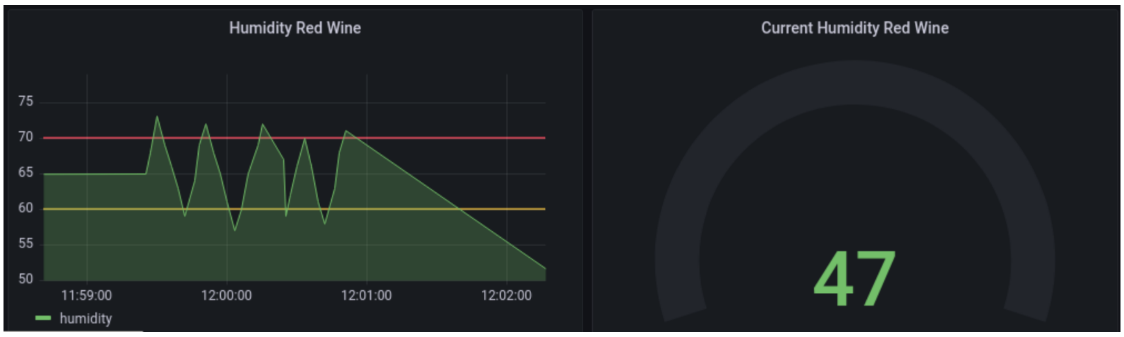
A dashboard is a set of one or more panels organized and arranged into one or more rows. Grafana ships with a variety of Panels. Grafana makes it easy to construct the right queries, and customize the display properties so that you can create the perfect dashboard for your need.

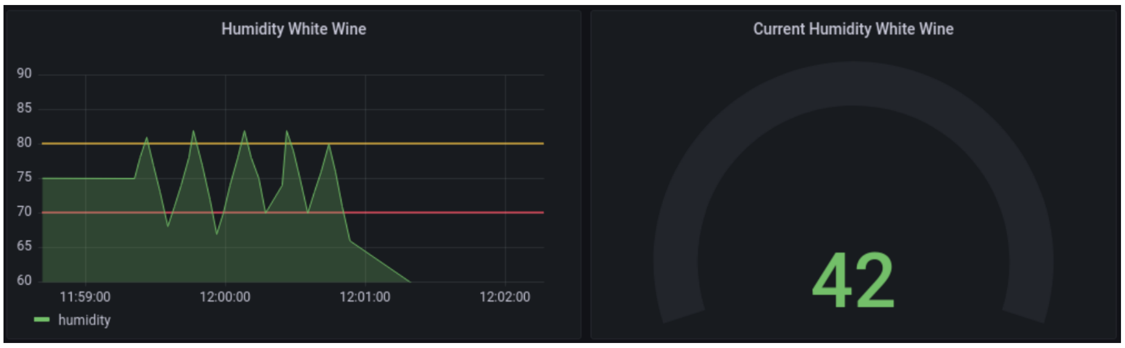
We have implemented a dashboard on Grafana in order to be able to monitor in real time the data that is stored in the database, and therefore to be able to view the trend of the monitored parameters.

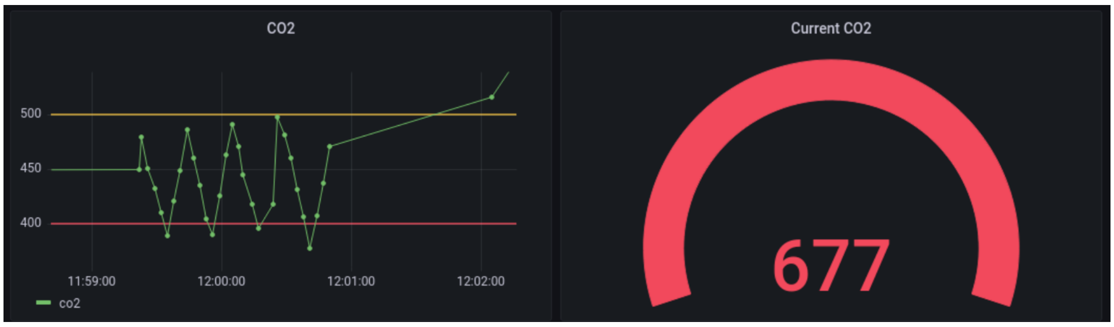
Through the dashboard we can see one row for each sensor. Each row contains two panels one that display the trend of the environmental quantities with the thresholds which help to see if the value remain within the established range and the other panel is for monitoring the current value of the respective environmental quantity.











3 Testing

The Smart Wine Cellar system has been tested with the Contiki Cooja Simulator environment and then on the sensors provided to us. The sensor are organized as following:

- *nRF52840 Dongle*: border router
- *nRF52840 Dongle*: temperature sensor for red wine in cellar
- *nRF52840 Dongle*: temperature sensor for white wine in cellar
- *nRF52840 Dongle*: humidity sensor for red wine in cellar
- *nRF52840 Dongle*: humidity sensor for white wine in cellar
- *nRF52840 Dongle*: CO2 sensor in cellar

The configuration of the environment is the following:

- we start the Mosquitto MQTT broker which runs locally and listens on port 1883
- we start the collector which contacts the broker at *localhost:1883*
- the temperature sensors contacts the MQTT broker at *[fd00::1]:1883*
- regarding COAP, the collector exposes the resources on the address *[fd00::1]:5683*, so humidity sensors and CO2 sensor contact the collector at this address
- the database is reachable at *localhost:3306*

The flow of execution is the following:

- The devices connect to the border router.
- The temperature sensors send the current temperature to the collector and if they are out of the acceptable range, the collector sends the command to turn on or off the conditioner.

- the humidity sensors send the current humidity value to the collector and if they are out of range, the collector sends the command to activate or deactivate the dehumidifier.
- the CO2 sensor notified to the collector the current CO2 value and depends on the value it activate the ventilation system and in some case it active also the alarm.