Immagine che contiene schizzo, disegno, Line art, Elementi grafici

Descrizione generata automaticamente

University of Pisa

*Internet of Things - Academic Year 2022/2023*

**SmartWinemaking**

**Students**:

*Emmanuel Piazza*

*Edoardo Focacci*

***Table of Contents:***

[**Introduction** 3](#_Toc140071343)

[**Architecture** 4](#_Toc140071344)

[CoAP Network 4](#_Toc140071345)

[Bypass 4](#_Toc140071346)

[Cooling 4](#_Toc140071347)

[MQTT Network 5](#_Toc140071348)

[Temperature 5](#_Toc140071349)

[Float 5](#_Toc140071350)

[Co2 5](#_Toc140071351)

[**Database** 6](#_Toc140071352)

[**Data Encoding** 7](#_Toc140071353)

[**Collector** 7](#_Toc140071354)

[CLI 7](#_Toc140071355)

[Packaging and Classes 8](#_Toc140071356)

[**Grafana** 9](#_Toc140071357)

# **Introduction**

//TODO intro

# **Architecture**

The system is composed of two different IoT device networks. One network is composed of IoT devices that use MQTT to report data, while the other network uses CoAP as application protocol.

Both MQTT and CoAP network is deployed using real sensors from the IoT testbed. Both networks are connected to a border router that allows them external access.

In this project were used 3 sensors (*Co2, Temperature* and *Float*) and 2 actuators (*Bypass* and *Cooling*).

The collector collects data from both MQTT and CoAP sensors and stores them in a MySQL database. The collected data can then be visualized through a web-based interface developed using *Grafana*.

A simple control logic is executed on the collector to modify the external environment based on the data that has been collected from temperature and float sensors. The collector also exposes a simple command line interface through which the user can retrieve the last measured values or change the ranges within the actuators will turn on or off.

## CoAP Network

### Bypass

This sensor is used to move up or down the cork linked to the barrel. This cork is used to make the air change inside the barrel and to make progress on the vinification.

By using the data received from the *Float* sensor and by checking the bounds defined by the user, the collector can assess whether it is necessary to move the cork.

The cork will be moved **UP** if the float level of the wine of the barrel surpasses a defined percentage, meanwhile it will turn **DOWN** if the float level returns on a stable percentage, still defined by the user.

To visualize the movement of the cork, a message is printed inside the console to show the action sent to the actuator.

### Cooling

This sensor is used to turn on or off the hydraulic cooling system built outside the barrel.

By using the data received from the *Temperature* sensor and by checking the bounds defined by the user, the collector can assess whether it is necessary to activate the cooling system.

The Cooling system will be **ON** if the inner temperature of the barrel surpasses a defined bound, meanwhile it will turn **OFF** if the temperature’s registered values return stable and are lower that a defined bound.

To visualize the state of the actuator, a message is printed inside the console to show the action sent to the actuator.

## MQTT Network

### Temperature

### Float

### Co2

# **Database**

//todo

# **Data Encoding**

As data encoding, we decided to use the *JSON* format. All the generated data is transmitted to the collector as JSON objects. This choice was taken because the sensors have limited resources, and XML has a too complex structure. JSON is more flexible and less verbose, resulting in less overhead.

# **Collector**

The collector takes care of receiving the data, communicating with the database to save the samples received and decide which actions to take.

It also implements a **CLI** to make it easier for the user to manage. We decided to implement the collector in Java, using the *Paho* and *Californium* libraries.

## CLI

The function available through the CLI are the following:

* **set lowTemp <integer>**
  + Used for setting the temperature sensor’s lower bound. This bound will be checked by the collector before turning OFF the Cooling system.
* **set upTemp <integer>**
  + Used for setting the temperature sensor’s upper bound. This bound will be checked by the collector before turning ON the Cooling system.
* **set activationLow <integer>**
  + Used for setting the float sensor’s lower bound. This bound will be checked by the collector before turning DOWN the Bypass system.
* **set activationUp <integer>**
  + Used for setting the float sensor’s upper bound. This bound will be checked by the collector before turning UP the Bypass system.
* **get temp**
  + Used for checking the temperature sensor’s current bounds.
* **get activation**
  + Used for checking the float sensor’s current bounds.
* **get co2**
  + Used for checking the co2 sensor’s last registered value.
* **exit**
  + Used for closing the application.

## Packaging and Classes

# **Grafana**

//todo