UNIVERSITÀ DI PISA

IoT PROJECT

< Smart Home >

Project made by**:**

- **Domenico Armillotta -** badge : 643020 - email: d.armillotta@studenti.unipi.it
- **Stefano Dugo -** badge : 564370 - email: s.dugo@studenti.unipi.it

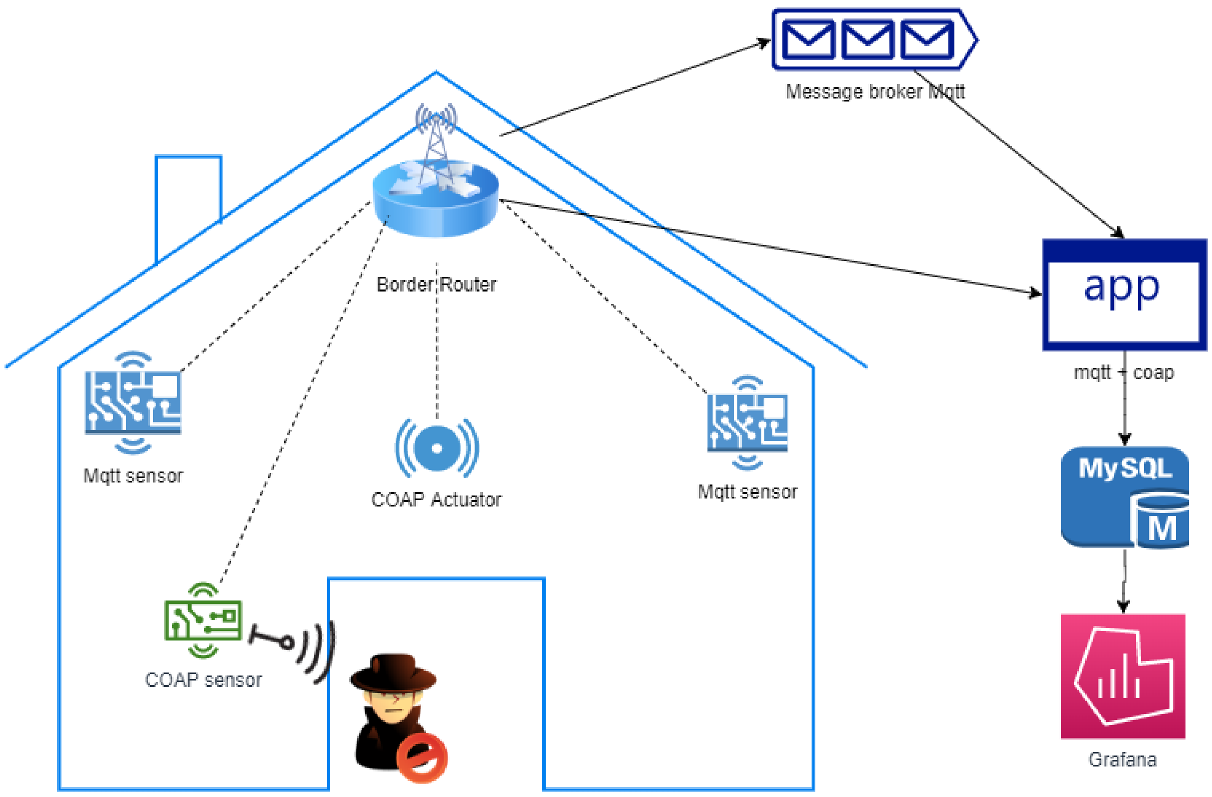
Project Overview

Our project aims to create a home monitoring system , environmental factors and an intrusion detection system. All the data are then saved in a database.

"mqtt" sensors take care of the monitoring part of gas , lighting , temperature and humidity.

"Coap" sensors deal with the intrusion system, as soon as the motion sensors sense the intruder , the alarm is triggered and gradually increases in intensity. To manually stop the alarm, just press the button of the sensor emitting the alarm.

The sensors interact with a python application, which receives data and saves them into a MySQL database and send a response to the sensors.



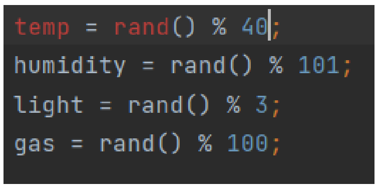
MQTT implementation

The mqtt sensors take care of monitoring some parameters including gas, light , temperature and humidity in the room.

As soon as they started, they connect to the border router deployed on a physical sensor , while the message broker used was deployed on the Vm , Mosquittos broker was chosen.

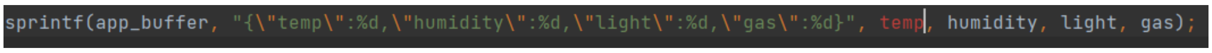
If the sensor connects to the border router , the LEDs are turned on for a short time to signal that the connection has been made.

Once the connection is made, the data is sent in json format , on the topic "info". Not having the actual sensors , the data are randomly generated with the rand() function.



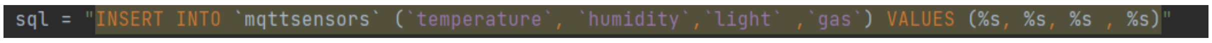
```
temp = rand() % 40;
humidity = rand() % 101;
light = rand() % 3;
gas = rand() % 100;
```

This is the message sent to the broker:



```
sprintf(app_buffer, "{\"temp\":%d,\"humidity\":%d,\"light\":%d,\"gas\":%d}", temp, humidity, light, gas);
```

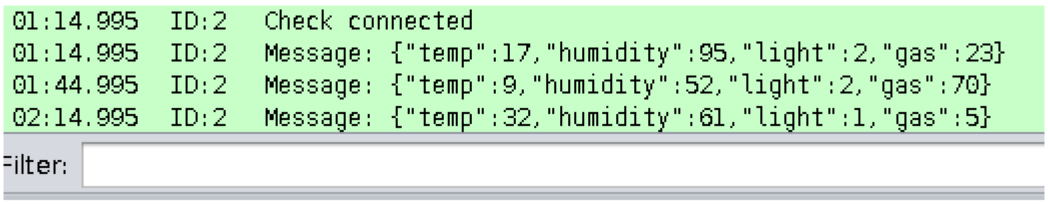
As soon as the app receives the data via json (onMessage()) from the mqtt sensors , it extracts the data and makes a query to the SQL Database to insert it into the table "mqttsensors" , with an associated timestamp , useful for graph visualization.



```
sql = "INSERT INTO `mqttsensors` (`temperature`, `humidity`,`light` ,`gas`) VALUES (%s, %s, %s , %s)"
```

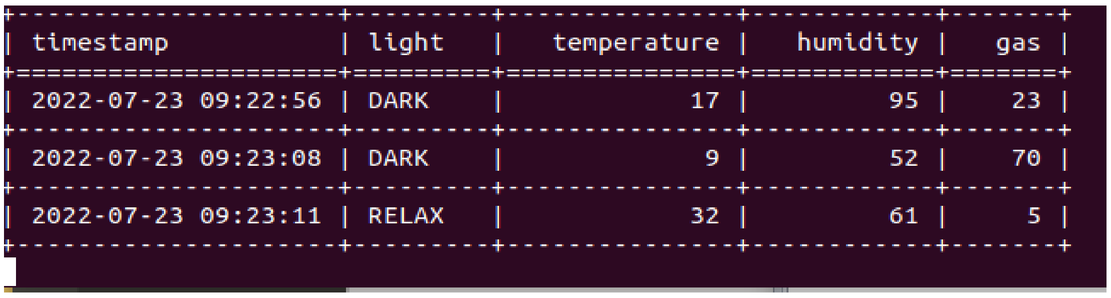
A thread is created for the mqtt client.

Testing on cooja , you can see that messages are sent and received correctly:



```
01:14.995  ID:2  Check connected
01:14.995  ID:2  Message: {"temp":17,"humidity":95,"light":2,"gas":23}
01:44.995  ID:2  Message: {"temp":9,"humidity":52,"light":2,"gas":70}
02:14.995  ID:2  Message: {"temp":32,"humidity":61,"light":1,"gas":5}
```

And they are also correctly entered on the DB:



```
+---------------------+---------+--------------+-----------+-------+
| timestamp           | light   |  temperature |  humidity |   gas |
+=====================+=========+==============+===========+=======+
| 2022-07-23 09:22:56 | DARK    |           17 |        95 |    23 |
+---------------------+---------+--------------+-----------+-------+
| 2022-07-23 09:23:08 | DARK    |            9 |        52 |    70 |
+---------------------+---------+--------------+-----------+-------+
| 2022-07-23 09:23:11 | RELAX   |           32 |        61 |     5 |
+---------------------+---------+--------------+-----------+-------+
```

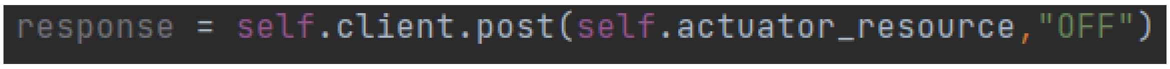
Coap implementation

"Coap" sensors take care of the alarm system; we have the motion sensors placed in various parts of the house, which detect movement and start an alarm when movement is detected. In order to do that, we have two resources, a motion resource and an alert resource. The motion resource randomly generates motion data with a timer, having no physical sensors.

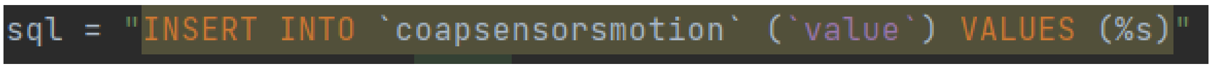
The sensor client was done by dividing the threads for recording and detecting the state change , so as to avoid problems caused by synchronization of threads , detected in the testing phase.

When the sensors register to the Python Server , they are instantiated with the resource class, and the observation of the sensor "Motion" begins , which detects any change in state.

As soon as the state of the resource changes , a query is made to the MySQL database of the sensor , and a POST is made to the alarm to turn the alarm on or off.



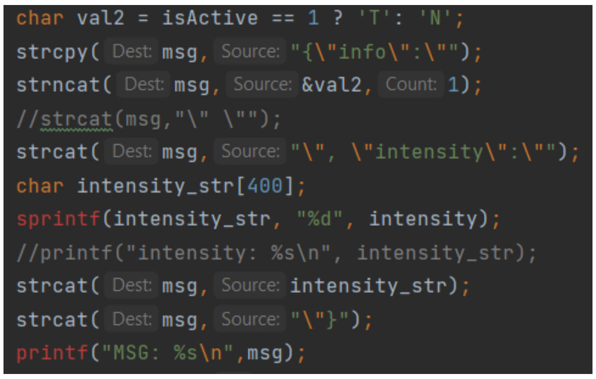
```
response = self.client.post(self.actuator_resource,"OFF")
```



```
sql = "INSERT INTO `coapsensorsmotion` (`value`) VALUES (%s)"
```

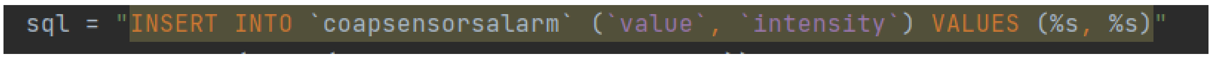
When an intruder is detected, the alarm is activated. It emits a variable intensity sound , in fact the intensity is increased according to the time of the intruder detection.

Communication via json was also used for these sensor types , here is an example of creating the GET message of the alarm:



```
char val2 = isActive == 1 ? 'T': 'N';
strcpy( Dest: msg, Source: "{\"info\":\"");
strncat( Dest: msg, Source: &val2, Count: 1);
//strcat(msg,"\" \"");
strcat( Dest: msg, Source: "\", \"intensity\":\"");
char intensity_str[400];
sprintf(intensity_str, "%d", intensity);
//printf("intensity: %s\n", intensity_str);
strcat( Dest: msg, Source: intensity_str);
strcat( Dest: msg, Source: "\"}");
printf("MSG: %s\n",msg);
```

It is saved in the Database , the signal intensity and actual alarm activation combined with a timestamp, useful data for graphing in Graph.



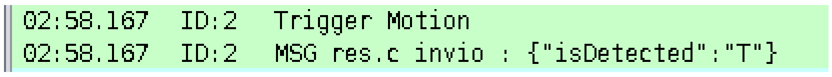
```
sql = "INSERT INTO `coapsensorsalarm` (`value`, `intensity`) VALUES (%s, %s)"
```

To disable the alarm and motion detection in manual mode, you can press the button of any sensor.

After the connection and registration phase , the sensor emits a light signal , to signal that the sensor is actually activated.

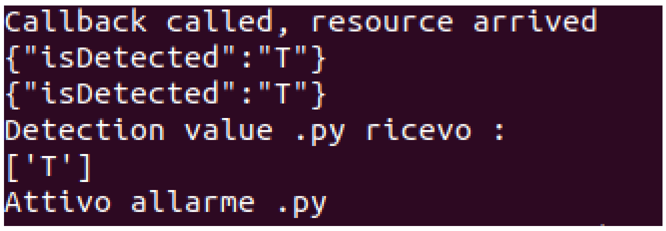
Testing on cooja:

1. we see the motion sensor changing state:



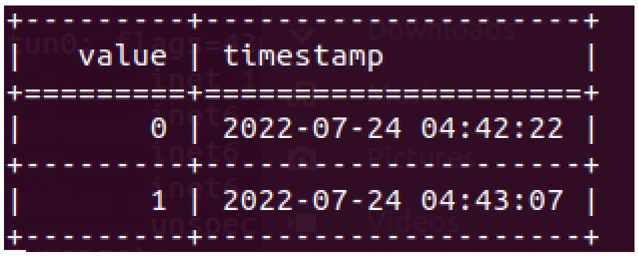
```
02:58.167   ID:2   Trigger Motion
02:58.167   ID:2   MSG res.c invio : {"isDetected":"T"}
```

2. The server receives the status change and activates the alarm sensors:



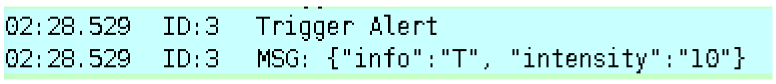
```
Callback called, resource arrived
{"isDetected":"T"}
{"isDetected":"T"}
Detection value .py ricevo :
['T']
Attivo allarme .py
```

3. The server makes a query to the database with "isDetected = 1" and the timestamp:



```
+---------+---------------------+
|   value | timestamp           |
+=========+=====================+
|       0 | 2022-07-24 04:42:22 |
+---------+---------------------+
|       1 | 2022-07-24 04:43:07 |
+---------+---------------------+
```

4. The alarm is activated with an intensity that will vary over time:



```
02:28.529   ID:3   Trigger Alert
02:28.529   ID:3   MSG: {"info":"T", "intensity":"10"}
```

MySql Server

The Python library "pymysql" was used to create the SQL server with the following access data:

host='localhost',

user='root',

password='PASSWORD',

database='collector'.

With 2 tables inside:

mqttsensors = for temperature , gas, light , humidity info of mqtt sensors.

coapsensorsmotion = for tracking detected movements and associated alarms over time.

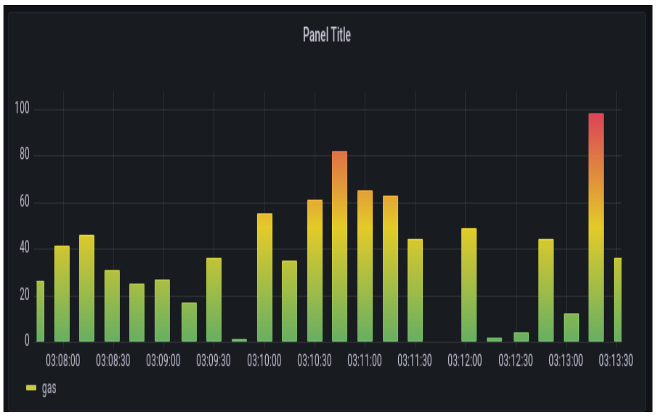
Grafana

We used Grafana for visualization of the data collected from the sensors and saved in a SQL database , coupling them with timestamp.

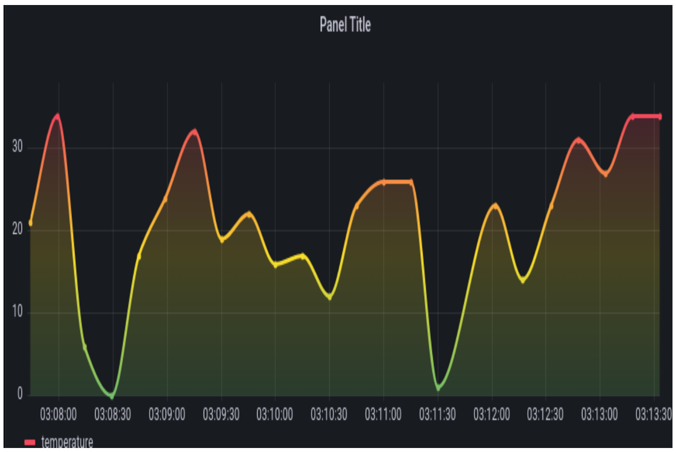
Recommended use of Chrome.

We update the graphs every 30 seconds.

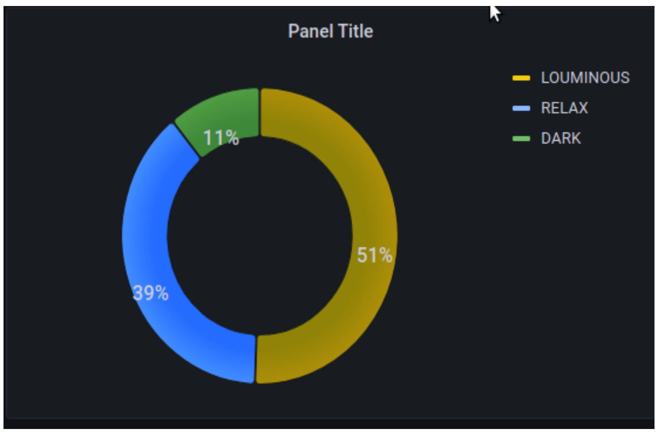
"Mqtt" dashboard:



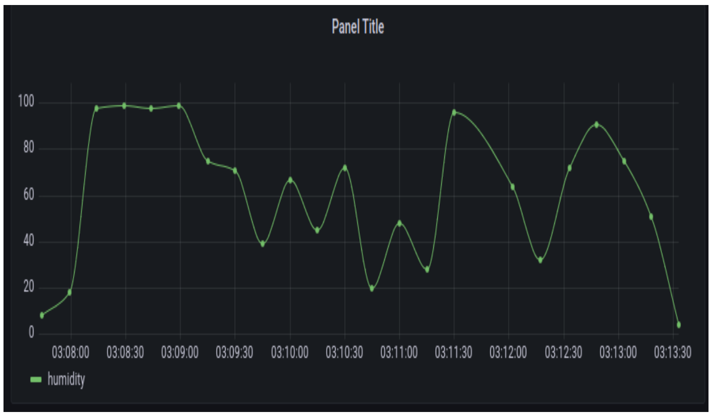
1. Chart for gas in the room



2. Chart for Temperature



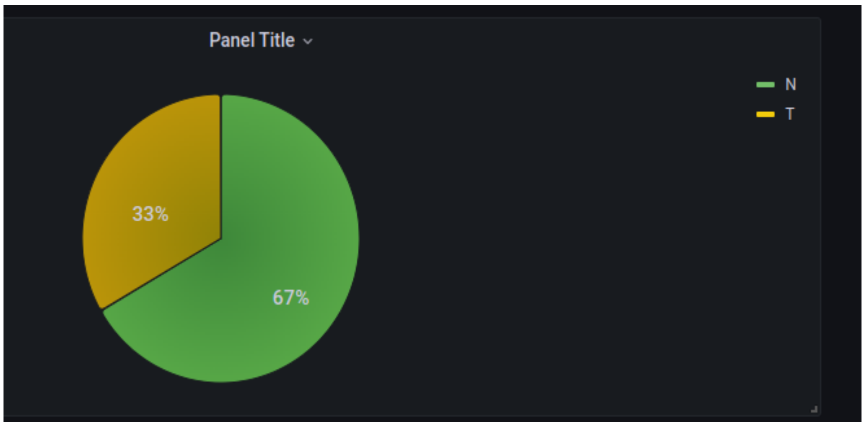
3. Chart for the luminosity of the room



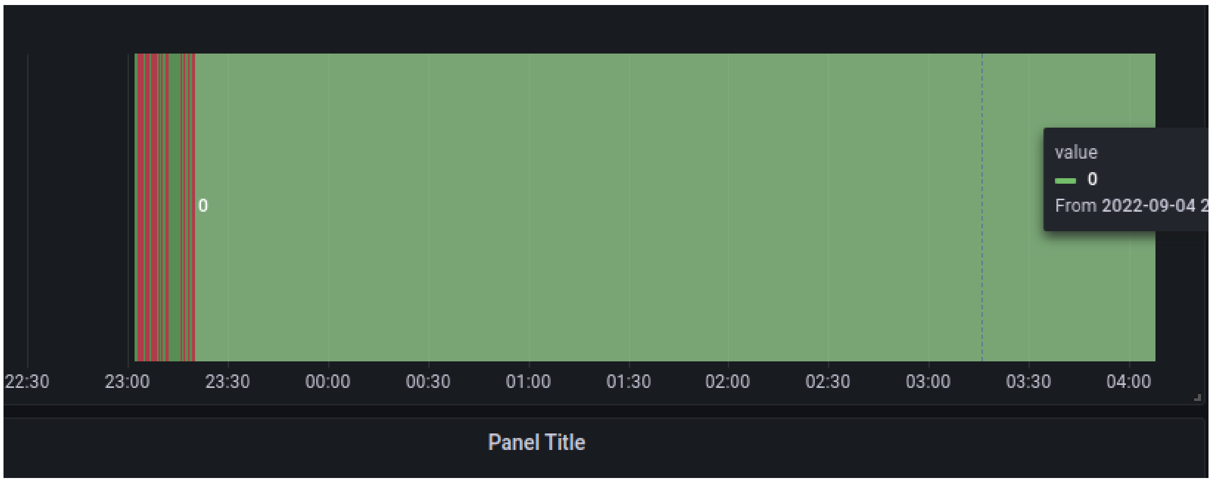
4. Chart for the humidity

“Coap” dashboard:

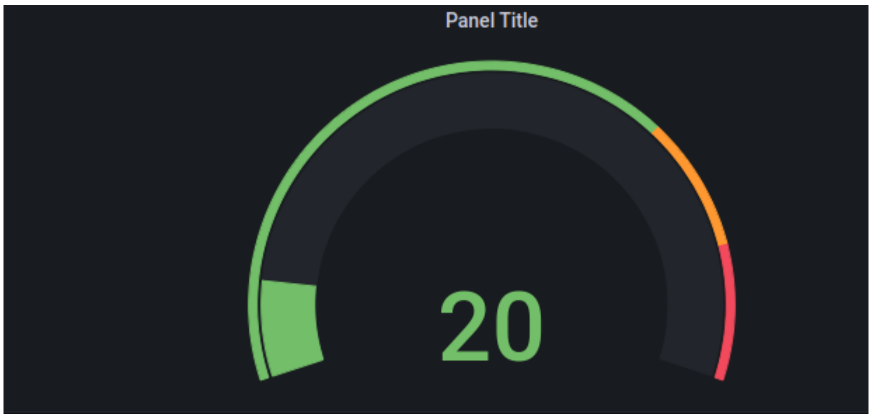
1. Chart for alarm % activation over the time



2. Chart for the detection of the intrusion



3. Dashboard for the intensity of the alarm updated real time value have a range 10-100.


Panel Title