UNIVERSIDADE DE COIMBRA
UNIVERSITY OF COIMBRA

DEI
DEPARTMENT OF INFORMATICS ENGINEERING

Sensing and Actuation Networks and Systems [2023-2024]

Project (Part II) – Process and visualize IoT data

Introduction

This assignment aims to create a full IoT solution from data collection to processing and visualisation. Students will have to program MQTT publishers and subscribers, process the data gathered, store it in a database, and finally visualize it on a dashboard.

Objectives

Students successfully concluding this work should be able to:

- Produce and collect real-time data
- Transfer data using MQTT
- Process data coming from sensors and save it in a database
- Display and analyse real-time data in a dashboard

Support Material

- MQTT broker running on a Raspberry Pi available at the Department of Informatics Engineering
- Files and examples from previous classes
- Python JSON: https://www.w3schools.com/python/python_json.asp

Scenario

This project simulates the automatic control of **5** rooms of a museum. Each room has three different sensors - temperature, humidity, and motion - connected to a smart device. The goal is to control the environmental parameters in each room throughout the day, maintaining ideal temperature and humidity conditions, and preventing any intrusion once the alarms are activated. All the information collected is stored in a time-series database and is available for analysis at any time.

Network configuration

The network configuration to be used is shown in Figure 1. It consists of **8** different parts:

1. **Rooms**: each room has a set of sensors, connected to a specific smart device (Raspberry Pi); the data collected will be published to a **MQTT Broker**.
2. **MQTT Broker**: provides the connection between publishers and subscribers and is located at a Raspberry Pi; a Mosquitto MQTT broker will be provided.
3. **Data Processing Unit**: located in the students' computer, this unit processes data and stores it to an InfluxDB database; it also sends selected data to the **Control Central**.
4. **Control Central**: program in the students' computer that receives alerts from the **Data Processing Unit** when sensor readings are outside the healthy range, controls actuators (air-conditioned and humidity controller), and switches motion alerts on/off.
5. **Alarm Console**: located in the students' computer, allows the user to turn alarms on/off.
6. **MQTT Debugger**: located in the students' computer, displays all MQTT messages exchanged on the network in a console.
7. **Storage**: all data is stored in an InfluxDB time series database (a InfluxDB Cloud free plan, located at https://www.influxdata.com/, will be used).
8. **Control Dashboard**: the information is visualized in a Grafana dashboard (a Grafana Cloud free plan, located at https://grafana.com, will be used).

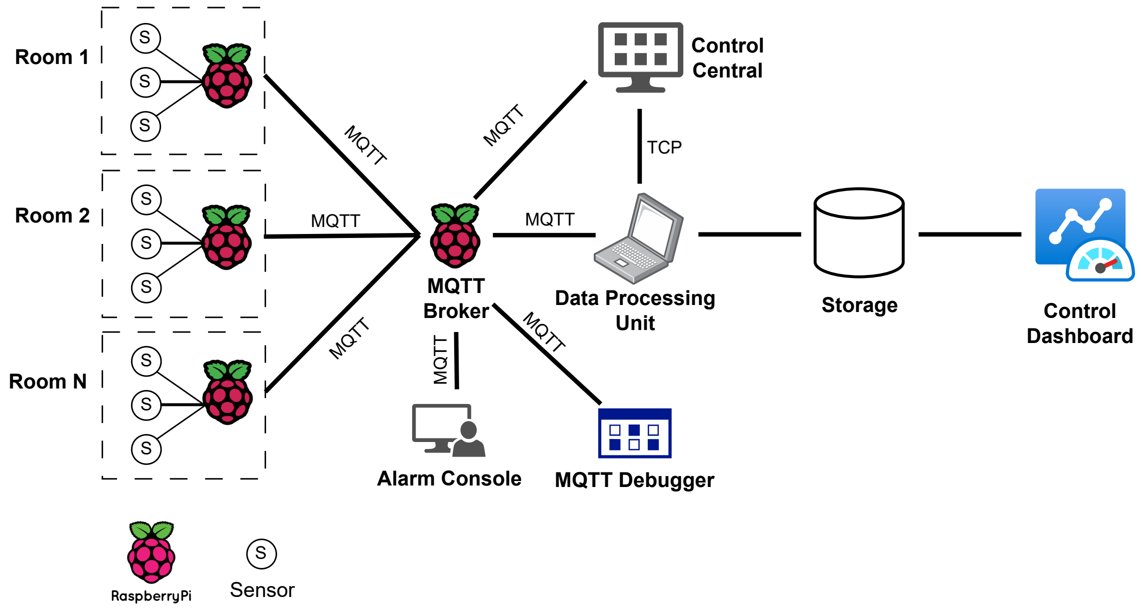

Figure 1. Network configuration

Each group of students (team) will have access to a Raspberry Pi during the work defence. The Raspberry Pi where the MQTT broker is located will only be accessible using MQTT, no other access is available to students.
To access the Raspberry Pi inside DEI premises, it is necessary to be connected to the "DEI" network. **Do not use the "eduroam" network!** To access the Raspberry Pi from outside DEI premises, you will need to connect to the department's VPN (Virtual Private Network). Please refer to the instructions provided by the helpdesk to connect to DEI's network using the VPN[1].
Both InfluxDB Cloud and Grafana Cloud are available with an Internet connection (registration in both sites will be required).

Description of work

Each team has to write the code to simulate one room, the data processing unit, the control central, the alarm console, and the MQTT debugger. Data is stored in a time series database, and processed data is displayed in a control dashboard.

The name of any MQTT topic used by the team must follow the format: ***GroupID_TOPIC***. E.g., "03_ROOM_DATA". Some of the MQTT topics are explicitly referenced in this assignment (e.g., ***GroupID*_ROOM_DATA**, ***GroupID*_ACT**), but additional ones can be created as needed for MQTT data exchange.
To obtain a *GroupID*, a previous registration via email, sent to the PL teacher, is required.

Rooms

Each **Room** simulates a museum room, in which sensor data is collected. The requirements for the simulation are:

- The data is sent to the **MQTT Broker** using a JSON (JavaScript Object Notation) object. E.g.:

```
room_data = {
        "temperature": 22.3,    # Celsius or Fahrenheit (float)
        "humidity": 10,         # % (integer)
        "motion": 0,            # 0 = no motion detected; 1 = motion detected
        "ac_working": 0,        # 1/0 = air conditioner working/not working
        "hc_working":0,         # 1/0 = humidity controller working/not working
        "sensor_type": 1,       # integer
        "room_number": 2        # integer
}
```

- The **Room** sensor data is simulated using a Python script.
- To simulate the change in temperature, every TEMP_TIME seconds the temperature increases between [-0.1, +0.5] ºC and the humidity between [-5,+10] % (must not fall bellow 0% or exceed 100%).
- To simulate intrusions, the motion alarm is triggered every MOV_TIME seconds.
- The room_number is equal to 5.

[1] https://helpdesk.dei.uc.pt

- *GroupID*, TEMP_TIME and MOV_TIME will be provided as line parameters of the program.
  ```
  $ python3 room.py {GroupID} {TEMP_TIME} {MOV_TIME}
  ```
- All data collected is sent to the ***GroupID*_ROOM_DATA** MQTT topic.
- The **Room** also subscribes the ***GroupID*_ACT** (Actuators) MQTT topic, where it receives information from the **Control Central** indicating that it must switch on the air conditioner or the humidity controller to increase/decrease the values until they reach the ideal values sent.
  - When the air conditioner is switched on, the temperature for that room starts to go up/down 0.5 ºC every TEMP_TIME until it reaches the ideal temperature.
  - When the humidity controller is switched on, the humidity for that room starts to go up/down 5% every TEMP_TIME until it reaches the ideal humidity.
  - A tolerance can be added when assessing whether the temperature has reached the ideal value, to avoid the air conditioner or humidity controller not stopping because an exact number is not reached.
- The temperature is given in Celsius, the humidity in percentage and the "sensor_type" will be 1.

During the defence, the script simulating the room will run in a Raspberry Pi. During the implementation students can run it on their own computer.

MQTT Broker

A Raspberry Pi with a Mosquitto v2.0.11 MQTT broker, located at IP **10.6.1.9 (srsa-pi-8.dei.uc.pt),** port **1883**, will be available for students. This broker will manage all the students' topics. Additionally, the topic **OTHER_ROOMS** will receive synthetic data, generated by other rooms (students will have access to the data but are not responsible for its generation).

Note:

- The format of data received on **OTHER_ROOMS** will be the same as the **room_data**.

Data processing unit

The **Data Processing Unit** will run on each team computer. It is a server program that has the following functionalities:

- Reads a configuration file with healthy sensor intervals and ideal values (`intervals.cfg`); the values are defined for all rooms and concern temperature and humidity.
  E.g.:
  ```
  15 30 23   # temperature low hi ideal
  40 60 50   # humidity low hi ideal
  ```
- Subscribes to each team MQTT topics, and to the additional **OTHER_ROOMS** topic.
- Communicates with the **Control Central** using TCP.
- Generates alarms when the values received from sensors, through the MQTT broker, are outside the healthy interval defined:
  - The alarm is sent to the **Control Central** indicating the room, if temperature/humidity must be higher or lower, and the ideal value.
- If nothing is received from a **Room** for 15 seconds, an alarm is sent to the **Control Central**; the event must be stored in the database.

- Receives commands from the **Alarm Console** to switch the motion alarms on and off and sends the information to the **Control Central**. The event is stored in the database.
- If the motion alarm is switched on and motion is detected in a room, an alarm is sent to the **Control Central** so that a siren can be activated (not simulated), and the event is stored in the database. If the motion alarm is switched off, the data is discarded.
- When a **Room** changes the status of its air conditioner or humidity controller device (working/not working), that information is processed to be able to calculate the price of the operation.
    - The air conditioner consumes 3000W per hour and the humidity controller 1000W per hour; the price of energy is 0.15 eur /kWh.
    - The time when the device started and finished its operation, and the total energy cost must be stored in the database, also indicating the room number and any other necessary information.
- Sensors with "sensor_type"=2 send data in Fahrenheit, which has to be converted in Celsius.
- Data from sensors with "sensor_type" different from 1 or 2 should be discarded. The discard information should be sent to the **Control Central**.
- Saves data to an InfluxDB database:
    - Room temperature values in ºC and humidity in %;
    - Motion detections when motion alarm is switched on;
    - Disconnected rooms (if no information was received for more than 15 seconds);
    - All other processed values found necessary for the dashboard.

Control Central

This console receives alarms from the **Data Processing Unit**, using TCP sockets. The alarms are displayed on the console screen with information about the sensor, room, date/time of receipt. The following alarms are received:

- Sensor values outside healthy intervals – receives the Room, the sensor, if temperature/humidity must be higher or lower, and the ideal value;
    - In response, **Control Central** sends a MQTT message to the **Room** to switch on the air conditioner or the humidity controller until the values reach the ideal value; both the ideal value and the indication to increase/decrease the value are sent.
- Disconnected room – if a room does not send information for more than 15 seconds;
- Motion alarms switch on/off;
- Motion detected in a room – only received if the motion alarm is switched on;
- Discarded sensor values – on detecting an unusual sensor type.

Alarm Console

This console receives commands from the user that may switch on or off the motion alarms. The command is then sent by MQTT to the **Data processing unit**.

MQTT Debugger

Writes in the console every message that is transferred through the MQTT broker. The information should be displayed as: [time]:[topic]:[message]

Storage

An InfluxDB database will be used to store data. This database must be created in InfluxDB Cloud (https://www.influxdata.com/). The registration of a free plan is required (registration has already been made in previous classes, students can use the same one). The database should hold all the necessary values to enable the following analysis of the data in the Control dashboard.

Control Dashboard

Grafana will be used to create a control dashboard. This dashboard must be created in Grafana Cloud (https://grafana.com). The registration of a free plan is required (registration has already been made in previous classes, students can use the same one).
Each team must prepare a dashboard to display the data collected from the sensors and processed in the processing unit. The data analysis should include three separate sections:

Section 1: Summary dashboard

- Students should select the most important information to present in the dashboard and use the best Grafana charts and graphs. Information about the status of each individual air conditioner and humidity controller devices (1 of each per room) should be displayed.

Section 2: Detailed dashboard

- Last values from sensors;
- Healthy intervals;
- Values history;
- Moving averages.

Section 3:

- This section will focus the energy consumption. The variation along time, and the cost associated with it, must be visualized.
- Data visualized will help the management understand when more energy is consumed and by which device.

Project delivery

This section describes the delivery conditions and submission policy for the project.

Delivery conditions

The project should be done in groups of two (2) students. While the project is a group work, each student will have its own grade, corresponding to the individual performance in the final defence. Python programming language must be used.

Project submission

Each group of students must create a zip file containing all the necessary files for their project to work. Include all source and configuration files needed. Do not include any files not needed for the execution of the programs. Also add a report (maximum 3-page A4 in pdf format) that includes information about the data sent to the MQTT broker, stored in the database, and the analysis made to the data (dashboard screenshots are required). The final zip file must be submitted to Inforestudante (only one zip file per group). **Submissions via email will not be accepted!**

Submission date: 11/May/2024
Defence date: 13, 14, 20 and 21/May/2024

Evaluation

This practical assignment is worth six (6) points of the final grade. The evaluation will consist of the categories listed in Table 1. The final grade will be weighted according to the individual defence of the project Not being present for the defence will result in a grade of 0 for that student. Students may be asked, during defence, to alter some of data processing features or the graphics shown in the dashboard.

Table 1. Evaluation criteria

Module	Value
Room	20%
Data Processing Unit	20%
Control Central	20%
Alarm Console	5%
MQTT Debugger	5%
Storage	10%
Control Dashboard	20%
Total	100%

Plagiarism

There will be no tolerance for plagiarism or any other type of fraud. Attempts to do so will be graded ZERO and will imply the failure of the course. Depending on the severity, the situation may lead to disciplinary proceedings. Code will be subject to inspection and comparison between all students.