POLITECNICO
MILANO 1863

Final Project

Project documentation (2021/2022)

Daniele Sinigaglia (10574224) - Mattia Siriani (10571322) - Matteo Visotto (10608623)

Contents

1 Introduction

1.1 Purpose

This document is meant to provide a detailed explanation about the technical details of the project we have decided to implement. We have chosen the second project "Smart Buildings and Neighborhoods".

1.2 Scope

The scope of this project is to design and implement a system to manages sensors deployed in different buildings. The system should be able to collect the data, moreover temperature and humidity, from the different sensors and use those data to trigger commands on nearby actuators. Lastly, once the data are collected and stored, some further analysis are done on this data to retrieve statistics about the data.

1.3 Technologies used

For our project we have used the following technologies:

- Contiki NG - To manage sensor motes.
- Node-Red - To manage connection flows such as sensor configuration and data storing.
- Apache Spark - To analyze the data and retrieve the required statistics.
- MQTT - Communication

1.4 Reference document

- Slides of Middleware technologies for distributed systems
- Apache Spark documentation https://spark.apache.org/docs/latest/
- NodeRed documentation https://nodered.org/docs/
- Contiki NG documentation https://contiki-ng.readthedocs.io/en/develop/

1.5 Github project and Telegram channel

Github repository: https://github.com/matteovisotto/MTDS-FinalProject
Telegram channel: https://t.me/+0byQR2Z1k_JiNjJk

2 Architecture

The architecture of our project is divided in three different parts. First of all, we have IoT sensors that retrieve data from the environment and then they publish those data by means of MQTT protocol.
Then, the data are received from the Node-Red flow that is subscribed to the root sensor topic to stores those data on a back-end.
The control logic for actuators is contained in the actuator itself that can subscribe to a specific MQTT topic and perform operations using data collected by interested sensors.
Finally, the data are analyzed and some statistics are achieved.
Each technology previously described (Technologies used) cover a different part described above.
Below, a more in depth description is done:

2.1 IoT sensors - Contiki NG

The data retrieved from our sensor are faked (the temperature and humidity are achieved through the use of a random function) and operation of the HVAC is emulated by a python script that implements some basic control logic to print on/off state changes based on received values.
The structure of our data is composed in this way:

- Location: represents the location where the sensor is located. The structure of the location is the following (A.B.C.D), where each letter represents the corresponding element:
  - A: Neighborhood-scale.
  - B: Building.
  - C: Floor.
  - D: Room.

Same division is used to build the MQTT topic where data are published (ex: mtds/sensor/data/A/B/C/D). In this way a subscriber can decide to get data with different levels of granularity.

- Date Time: represents the time at which the sample is collected.
- Temperature: represents the temperature sensed from the sensor, using as unit of measure Celsius degree.
- Humidity: represents the humidity sensed from the sensor, using as unit of measure kg/m^3

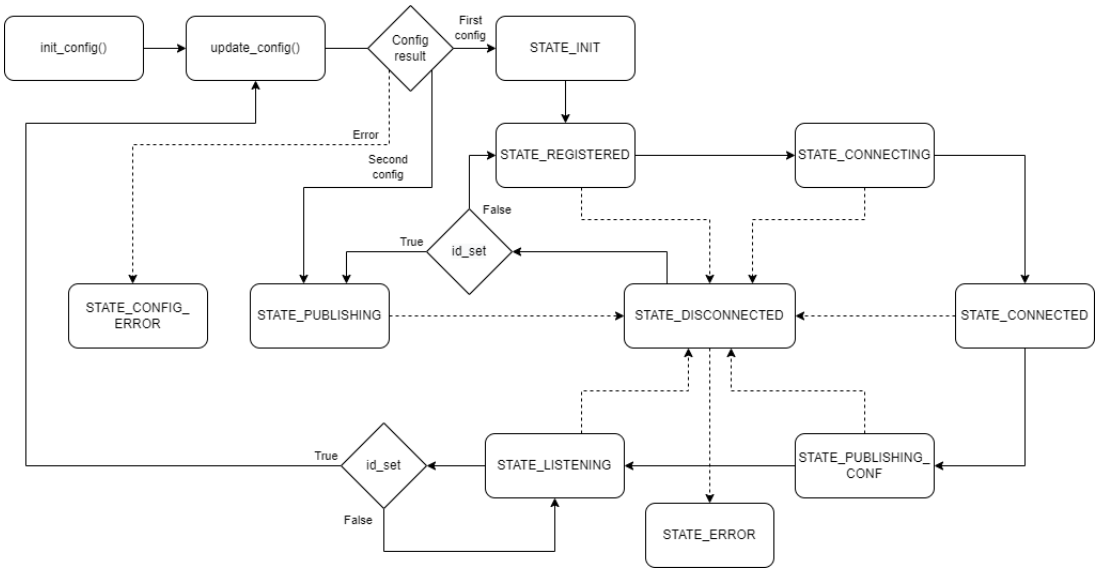

Figure 2.1: State machine

2.2 Flow of data and storage - Node-Red

Two Node-Red flows have been used.
A first flow (figure: 2.2) is used to configure a sensor location according to its identification number. Each sensor "serial number" is associated to a location and when the sensor connect the broker a configuration message request is sent to a generic topic (mtds/sensor/conf) and it receive its configuration as response by subscribing its own conf topic (mtds/sensor/conf/{{sensor_id}}).

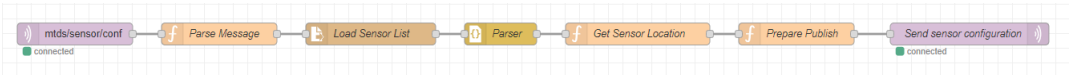

Figure 2.2: Mote Configurator

A second flow (figure: 2.3), instead, is used to collect messages from all the sensors of the network. A MQTT client subscribe to all the location (mtds/sensor/data/#) and for each received message the content is parsed and saved in a structured CSV file to later accomplish an analysis of those data.

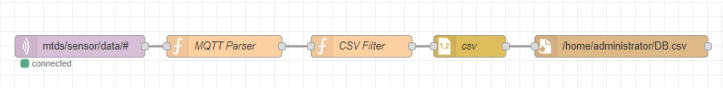

Figure 2.3: Data Saver

2.3 Analysis of data - Apache Spark

Here the data stored at the previous point, are analyzed, exploiting the Apache Spark SQL API. Different types of analysis have been conducted:

- Hourly, daily and weekly moving average for both temperature and humidity. Those analysis have been done grouping the location in the following ways:
- Daily-night temperature difference.
- Month of the year with the highest average night-day temperature difference.

All the statistics conducted have been done, grouping the location in the following ways:

- Neighborhood-scale.
- Building.
- Floor.
- Room.

In order to avoid re-computation of the data, the Dataset useful for other analysis have been cached and unpersisted only when there are no more needed.

3 Design Choices

Below are present the design choices we have done.

3.1 Contiki NG

We have decided to use Contiki NG because it was the most suited technology to handle IoT sensors, since it is explicitly designed to manage those devices.

3.2 Node-Red

We used Node-Red to handle connection flows and the configuration one due to the simplicity of the logic and the availability of all the pre-configured node we need.

3.3 Apache Spark

We have decided to use Apache Spark to handle the analysis of the data, because, thanks to its SQL API, it permits to achieve different statistics of the data in a smoother and easier way. Forse aggiungere anche qualche considerazione sul fatto di dividere il lavoro per i vari worker.

3.4 MQTT - Mosquitto

We used MQTT since it is the most suitable message oriented publish/subscribe protocol for IoT devices.

3.5 Why not Kafka?

In this project also Kafka has been taken into consideration but, even if it is a produce/consumer message event driven communication protocol, it is most suitable

for data exchanges between business application in back-end processing. In addition, Kafka it is heavier than MQTT making it not so suitable for IoT motes.
A possible use of Kafka within this project could have been the use of this technology to merge the Spark analysis into the flow, making them real time.
However, we decided to carry out a data storage and perform the calculation of the statistics asynchronously, in fact they provide very large time windows and, in a real scenario, a real-time output would not have been as relevant as the persistence of the individual readings.