Documentation - IoT platform planning and design for analysis of environmental conditions in smart cities

Table of Contents

Documentation - IoT platform planning and design for analysis of environmental conditions in smart cities 1
Introduction 1
Authors 1
General Considerations 2
Implementation 2
Database Flows 2
Flows responsible for requesting, parsing and storing the data 2
Near-Realtime Graphs 3
Statistical Graphs 3
Conclusion 4

Introduction

This document contains the documentation analysis, considerations and thorough explanation of the implementation regarding the project "IoT platform planning and design for analysis of environmental conditions in smart cities". Defines the work that has been done in order to implement the project's specifications, explains the decisions taken behind the implementation and gives a brief overview of the Node Red framework which has been used in order to conclude this project.

Authors

Stavros Alexakis
Ioannis Brant-Ioannidis

General Considerations

Based on the project requirements and options available we have made the following decisions:

- Instead of using FRED, we chose to setup Node Red locally in order to implement the specifications and generate the results.
- We are collecting the data and generating the results from the locations Volos and Kriti. This can be easily changed if you wish to collect the data from different location using this project in the future.
- The data are collected from the open API: https://maps.sensor.community/#2/0.0/0.0. The documentation of this API can be found on GitHub: https://github.com/opendata-stuttgart/meta/wiki/EN-APIs.
- Installation steps can be found on the document *Documentation/Installation-Guide.docx*.

Implementation

The flows we are using can be separated into two large categories:

Database Flows

These are the flows which are creating the database tables of the SQL database. The main use of these tables is to store the data retrieved from the API in order to enable us to create the required graphs, generate results and be able to provide a general, data-based overview regarding the environmental conditions of the chosen locations.

The tables are created and queried by making use of the sqlite node provided by the Node Red framework and are receiving the SQL Query from the previous node's msg. topic property, responding accordingly to each command.

The tables that we are using in order to store data are:

TABLE Volos: Stores the temperature, humidity, pressure.
TABLE VolosPM: Stores the PM2.5, PM10.
TABLE Kriti: Stores the temperature, humidity, pressure.
TABLE KritiPM: Stores the PM2.5, PM10
TABLE VolosAVG: Stores the average temperature, humidity and pressure.
TABLE KritiAVG: Stores the average temperature, humidity and pressure.

Flows responsible for requesting, parsing and storing the data

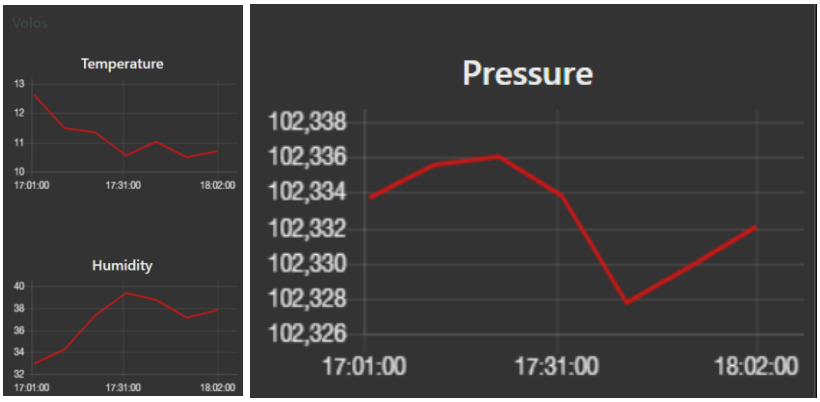
This category of flows refers to the flows that are responsible for performing the http requests to the mentioned API, parsing the returned response in order to extract the useful information and store them into the respective SQL tables that were created using the Database Flows.

Near-Realtime Graphs

The http requests are performed by using inject nodes, with an interval currently set to 10 minutes (i.e., every request is performed every 10 minutes). After each request and parsing of the results, each result is provided to the appropriate graph which is displayed on the Dashboard.

Note: *As mentioned on the APIs docs, due to performance reasons the data from the sensors are not immediately available which is the reason why the implemented system cannot be considered a realtime system, but provides an accurate reflection.*

Examples:

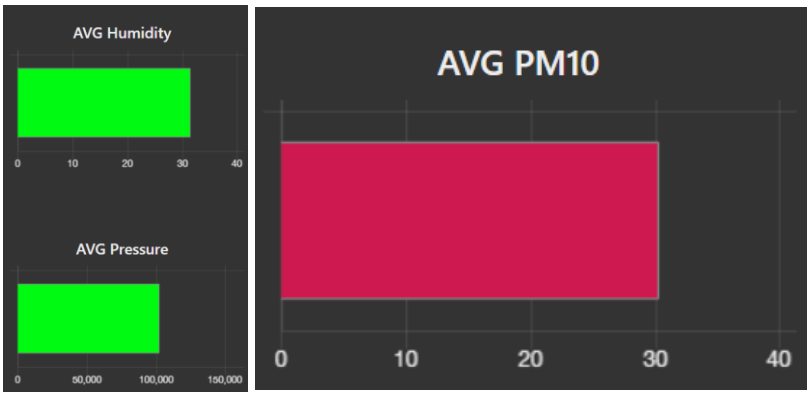


Statistical Graphs

Apart from the above mentioned, near-realtime graphs, the application provides also graphs that display the average of each environmental measurement that is collected. This is done by using the data that are collected and stored into the database tables every 10 minutes, as described above.

These graphs are updated every 2 hours and the result is provided by selecting the AVERAGE (using SQL queries) of each measurement that is collected and stored into the respective SQL Table.

Examples:



Conclusion

The above describe the current implementation of the IoT platform that can be used in order to collect certain environmental measurements and display them for specific locations.

As next steps, we can expand the functionality of the current implementation by providing dynamically the locations and measurements that we want to receive data from. In order to achieve that we could use configuration files, where the locations and environmental measurements are specified.

The project is not currently hosted online. In order to run it and generate results one needs to setup and run the project locally following the installation steps on Installation-Guide.docx.