Distributed systems

Project 1: Energy Management Software

Nabil Abdennadher, Nizar Bouchedakh

Smarthepia is an IoT platform deployed on the 4th and 5th floors of hepia's "A" building. This platform is composed of sensors (temperature, humidity, luminance, motion) and actuators (controlling stores, radiators and light). The sensors' measurements and actuators' control are accessible through RESTful web services:

- **Sensors:** As you can see on this page (http://lsds.hesge.ch/smarthepia/), the sensors are divided into 3 separate sensor networks. Each network has its own REST server deployed on a Raspberry Pi. The three Raspberry Pis' IP addresses are:
  - Pi1 (5th floor): 129.194.184.124
  - Pi2 (5th floor): 129.194.184.125
  - Pi3 (4th floor): 129.194.185.199

  All Raspberry REST servers are identical (same API) and listen on the port 5000.
- **Actuators:** Actuators (store and radiators controllers) are managed through another RESTful web service.

A documentation of these REST servers is available on cyberlearn.

The goal of this project is to develop an energy management software that allows to:

1. lower the temperature of a room to a given threshold when it is empty,
2. increase the temperature of a room to a given threshold when it is occupied,
3. close the stores when the humidity is high,
4. open the store at day time, when the luminance is low and the room is occupied,
5. display the status of a given store and/or a given radiator,
6. provide statistics.

Figure 1 presents the global architecture of the energy management software to design and develop:

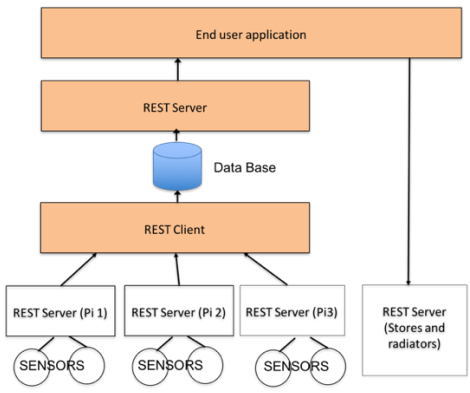

Figure 1: Global architecture of the energy management software.

The modules to develop within this project are those in salmon colour. The development of this tool will be done in 4 steps.

Step 1: Creating the database and developing the REST client (Deadline: 8th November 2017, 10:00)

Raspberry servers only return the last measurement of every sensor. To keep a record of sensors' data, you are asked to develop a REST client. This REST client will periodically (every 4 minutes) get the last measurement of every sensor and store it in a Database.

Deliverables:

1. REST client which retrieve data from the three raspberries and store them in a database (python code)
2. Document explaining the database structure: which database management system, which tables to use?
3. A demonstration must be made to the Professor/assistant. You must show that the database is well fed by the data coming from the 3 raspberries.

Step 2 (Deadline: Tuesday 21th November 2017, 23:55)

You are asked to develop a REST server supporting these functionalities:

1. get the last measurement of a specific sensor. Sensor is identified by the couple: controller ID-sensor ID.
2. process the average of the *x* last measurements of the sensors in a specific room. Rooms are identified by room ID. *x* must be a parameter of the request.
3. get the measurements of a specific sensor between two dates.

Before starting the development, you must design the routes, parameters and return format of your REST server. You will be using apidoc tool (http://apidocjs.com/) to generate the documentation. These routes must be validated by the professor/assistant.

Deliverables:

1. *apidoc* documentation
2. REST server (python program)
3. A demonstration must be made to the Professor/assistant.

Step 3 (Deadline: Tuesday 28th November 2017, 23:55)

Develop a Web or mobile application on top of the REST server developed in Step 2.

The Web/mobile application should:

1. Display the last measurement of a specific sensor.
2. Display the average of the *x* the last measurements of sensors in a specific room.
3. Display graphs of measurement of a specific sensor between two dates.

Step 4 (Deadline: Tuesday 12th December 2017, 23:55)

Extend the web/mobile application developed in step 3 to support the REST server used to manage the stores and radiators (Figure 1). In addition to the statistics of step 3, the application must provide these functionalities:

1. lower the temperature of a room to a given threshold when it is empty
2. increase the temperature of a room to a given threshold when it is occupied
3. close the stores when the humidity is high
4. open the store at day time, when the luminance is low and the room is occupied
5. display the status of a given store and/or a given radiator.