Programming for IoT applications – AY 2015/16
Lab 4

Exercise 1. Become confident with the Mosquitto clients:

- Use mosquitto_pub to publish json messages with a certain topic
- Use different (more than one) mosquitto_sub to receive the messages published by mosquitto_pub. Play with the wildcards (‘+’ and ‘#’) when you subscribe for the topic.

Use the raspberry pi as message broker.
Repeat this exercise using the message broker at seemp.polito.it (use it ONLY for this lab)

Exercise 2. Develop an MQTT publisher to send:

- every minute the date and time following the format dd-mm-yyyy hh:mm.
- every 30 seconds the unix timestamp.

The two messages must be sent in JSON with two different topics.

Develop a first MQTT subscriber to receive only the messages about date and time and print the information in a user friendly format (not the full JSON).

Develop a second MQTT subscriber to receive only the messages about unix timestamp and print the information in a user friendly format (not the full JSON).

Use the raspberry pi as message broker.

SUGGESTION: Exploit the Mosquitto clients to debug the communication among your applications.

Exercise 3. Using the raspberry pi, develop a RESTful-style Web Service and identify the proper HTTP methods (among GET, POST, PUT and DELETE) to:

- retrieve information about the temperature sensor
- retrieve information about the humidity sensor
- change the status of the relay to switch on or switch off a led
- retrieve information about the status of the relay (on or off)

Use the following senml+json data format (IETF draft, standard in progress):

```
{
 "bn": "<node_id>",
 "bt": "<unixtime_sec>",
 "e": [
   {
    "n": "measurement_type (e.g. Temperature or Humidity)",
    "u": "<units_string>",
    "v": "<value_float>"
   },
   {
    "n": "measurement_type (e.g. Temperature or Humidity)",
    "u": "<units_string>",
    "v": "<value_float>"
   }
 ]
}
```

Develop a **client** python application to retrieve such information and to change the status of the relay by invoking the RESTful web services.

SUGGESTIONS:

1. use the library *urllib2* to open ULRs and read their contents (handle exceptions properly)
2. use the library Adafruit_Python_DHT to manage the DHT11 sensor
3. use the library RPi.GPIO to manage the relay

Exercise 4. Using the raspberry pi, develop a first MQTT client, that works as publisher and subscriber, to:

- publish information about the temperature sensor every 30 seconds
- publish information about the humidity sensor every 60 seconds
- receive, as subscriber, commands to change the status of the relay and switch on or switch off a led
- publish the status of the relay every time it changes

Develop a second MQTT client to subscribe, receive and print all the information about temperature, humidity and relay status

Develop a third MQTT client to publish an actuation commands to change the status of the relay

Use the senml+json to exchange the information among the MQTT clients
Use the raspberry pi as message broker.

SUGGESTION:

1. use the library Adafruit_Python_DHT to manage the DHT11 sensor
2. use the library RPi.GPIO to manage the relay