Environmental Monitoring using IoT Devices

Davide Remondina, Marco Rodino, Andrea Zanin

Project overview

The aim of the project is to measure some parameter, such as temperature and humidity, from the environment through sensor and then communicate the readings or the average of the readings through regular internet to a backend, which stores the maximum and the minimum of each month on a persistent log and periodically communicates the updated log via email.

Node-Red implementation

First thing the Flow produces a timestamp of the current time from which the actual month is extracted. The flow searches on a specified directory if a log with the current bests already exist and, if it does, reads it and stores it in a JS Object called as the name of the month plus the year. This is necessary to ensure data durability even in the case of rebooting after a failure.

The flow subscribes to the topic "iot/sensor" on the server mqtt.neslab.it:3200 and processes every message as follows:

1. Wait 2 seconds to ensure that the log has been read from disk in case of reboot. In practice we have no guarantee on which one of the two parallel stream of instruction is executed first, but 2 seconds is a reasonable time to wait.
2. It converts the object read from the MQTT connection into a JS object.
3. Extract from the timestamp included in the message the day and the month.
4. A switch block checks which is the topic of the message. We worked using only humidity and temperature, but other type of measures can be easily added.
5. If this is the first reading of the month, a new JS object called as the month + year is created with dummy maximum and minimum. This object is saved as a flow variable.
6. The current maximum and minimum of the month is compared with the value of the reading and updated if necessary.
7. Even if not changed, the JS object keeping track of maximum and minimum is overwritten on a local files in order to have a persistent storage.

Furthermore every 3 days an e-mail with the log of the month is sent to a specific e-mail address using the additional set of nodes contained in the "node-red-node-email" package.

Contiki-NG implementation

For the contiki part two programs, which are a modified version of the mqtt-demo program, have been implemented. One program simulates a sensor that reads temperature from the environment while the other one simulates a humidity sensor.

The program flow is the following:

1. A connection to mqtt.neslab.it:3200 is established. If the connection cannot be established for some reason the program retries after a set number of seconds, this number is increased for each connection failure.
2. The program register to the topic “iot/sensor”.
3. Each program starts to generate its own samples at a rate of 1 sample every 10 seconds and store them in an array of 6 elements. Each sample is a data structure which contains the generated value and the Unix timestamp of when it has been generated.
4. Once the array is filled with measurements, the average is calculated.
5. If the average is higher than a given threshold the program publishes on the mqtt connection all the values contained in the array, otherwise it only publishes the average with the last generated timestamp. Before the publish mechanism takes place the data are converted and formatted into a JSON string that can be easily converted into a JS object at the backend. The conversion is made as follows:

```
len = snprintf(buf_ptr, remaining,
               "{"
               "\"type\":\"T\","      // T for temperature, H for humidity
               "\"value\":\"%d\","
               "\"timestamp\":\"%d\"",
               temp.value, temp.timestamp);
```

6. Between each publish the program wait 2 seconds, this has been made to make sure that the publish actually completes.
7. The program restarts generating samples and reuses the same array as before to store them.

COOJA Simulation

To simulate the IoT environment a Cooja simulation has been made. Inside the simulation two cooja motes have been inserted, one is the temperature sensor and the other one is the humidity sensor. To make them connect to the internet a border router has been also inserted in the visibility range of the other two motes. The sensor motes connects automatically to the border router in their range, if the router is not in range they try to reach it through the other motes. Others motes can be easily added representing others temperature or humidity sensors.

MQTT configuration

From the MQTT point of view achieving an exactly once semantic is very expensive and it is not already implemented in the library. We have then decided to opt for an at least once semantic, which is easily achieved by setting QoS=1. This is enough for our purposes

since we are only interested in keeping track of maximum or minimum therefore reading a duplicate message does not affect the result.

The Cooja simulation is not able to reach the outside world without a broker, for this reason a mosquitto broker has been installed. Finally to make the MQTT connection with neslab work in the configuration file of the broker the following lines have been added:

```
connection bridge-01
address mqtt.neslab.it:3200
topic # out 0
topic # in 0
```