Politecnico di Milano

Internet of Things Project
Smart Thermostat

Andrés Felipe Rodríguez

10675968 - 915761

Internet of things

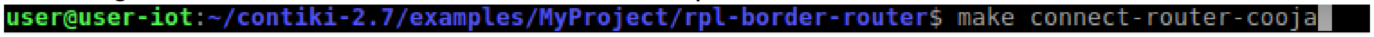
2019

Creating Motes using Contiki.

The topology used in this project consists in the creation of 5 sky motes; The first one, *rpl-border-router*, included in the examples folder of Contiki 2.7, is already working for our purposes. The rest of the motes are defined as *sensor-mote,* this mote is created using two main methods, *GET* and *POST*, for each resource that I use in this application. The resources are defined as *BLUE, RED, GREEN, TOOGLE* and *TEMP*, corresponding to the air conditioning, the heating, the ventilation, the actuators and temperature systems, respectively.
The *GET* method is implemented four times, one per each service *(AirCond, Heating and Ventilation),* and one for the temperature sensor, in the temperature one, it is defined as a *PERIODIC_RESOURCE* and configured in order to have the temperature value every 20 seconds. The temperature handler function is implemented according to the requirements, taking in to account the value of the Boolean variables for the three main services and according to this increase or decrease the temperature value.
The *POST* method is implemented three times, one per each actuator *(toogleAirCond, toogleHeating* and *toggleVentilation),* in the functions of the AirCond and the Heating system, the implemented algorithm is implemented in order to make them mutually exclusive.
In the main process the temp value is initialized with a random value between 10 and 30.

Cooja Simulation.

Once started cooja, I create a new simulation, including, first the *rpl-border-router* sky mote and then including four *sensor-mote* sky motes. Then I need to open a *Serial Socket (SERVER)* window for the router mote, and in a command window I start the CoAP connections with the following command in the folder direction of our rpl-border-router:



```
user@user-iot:~/contiki-2.7/examples/MyProject/rpl-border-router$ make connect-router-cooja
```

The command window will ask the password of the user (*user*).

After this I can start running the Cooja simulation, the topology implemented is the following.

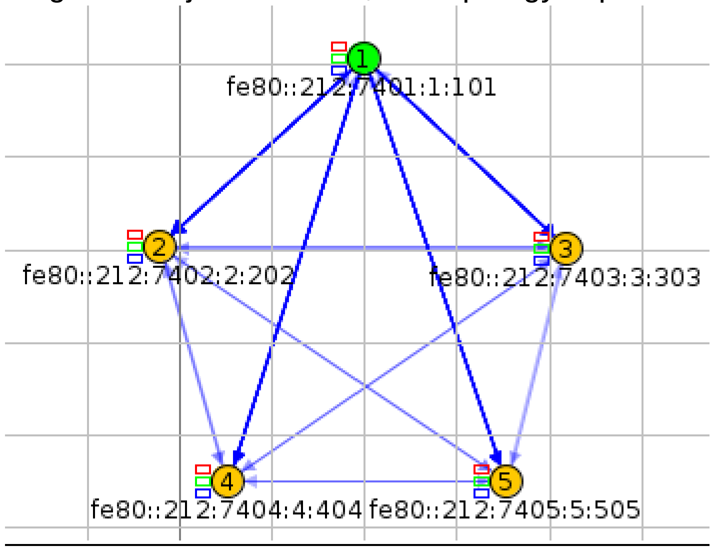
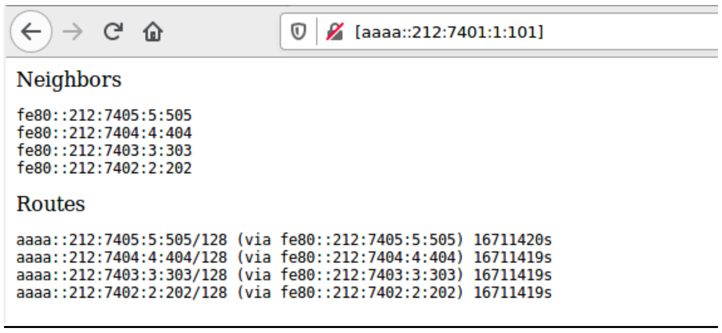


Figure 1. Cooja Simulation Topology

In the browser I can go to the Server IPv6 address corresponded to the router, and I will verify the routes and the neighbors of the router that are correspondent to the IPv6 addresses of the sensors.



Neighbors

```
fe80::212:7405:5:505
fe80::212:7404:4:404
fe80::212:7403:3:303
fe80::212:7402:2:202
```

Routes

```
aaaa::212:7405:5:505/128 (via fe80::212:7405:5:505) 16711420s
aaaa::212:7404:4:404/128 (via fe80::212:7404:4:404) 16711419s
aaaa::212:7403:3:303/128 (via fe80::212:7403:3:303) 16711419s
aaaa::212:7402:2:202/128 (via fe80::212:7402:2:202) 16711419s
```

Figure 2. Server IPv6 address

Node-Red.

First of all, it is necessary to start node-red in a new cmnd window. In Node-Red I started implementing the four first requirements regarding the Dashboard visualization and modification, followed by the implementation of the email alert, the steps of the implementation are the following:

- Using the *inject* node, *coap request* node and the *gauge* node, I can visualize the current temperature of each room value observing the temperature resource by implementing a GET request *(i.e. coap://[aaaa::212:7402:2:202]:5683/sensors/temperature)* of the temperature for each *sensor-mote* each 5 second, specified in the injection time.

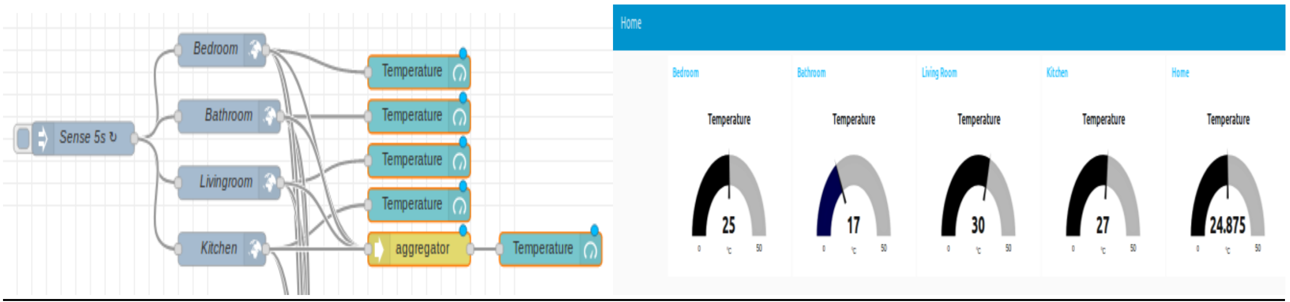


Figure 3, current state of the thermostat in each room

- For send email alerts when the temperature is not in the desired range, we use the *email* node and a *switch* node for each coap GET request.

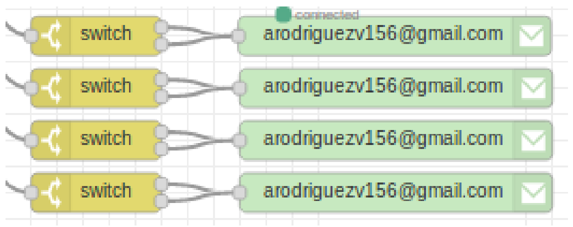


Figure 4, Node red implementation send emails

- Using the *button* node and using again the *coap request* node this time set to a POST request for each actuator *(coap://[aaaa::212:7402:2:202]:5683/actuators/toggleAirCond)*, we can change the value of the variable that controls if the service is ON or OFF, we see this value using again another *coap request* node set to a GET request and a *text input* node. *(coap://[aaaa::212:7403:3:303]:5683/services/Ventilation).*

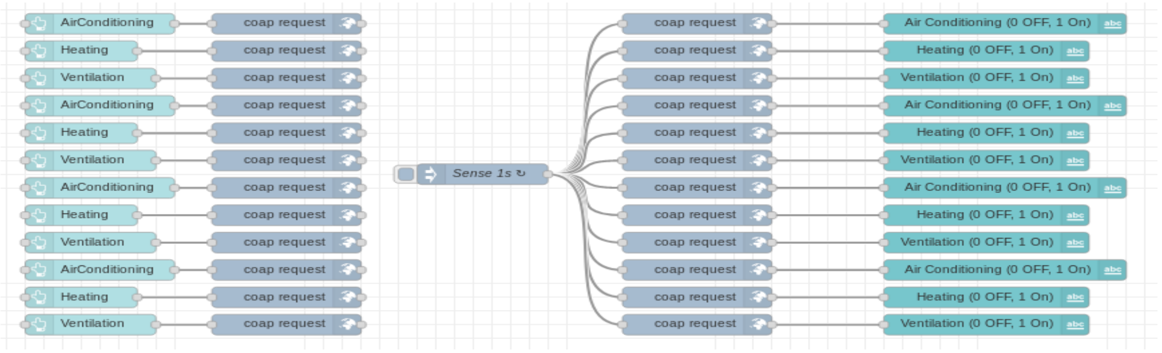
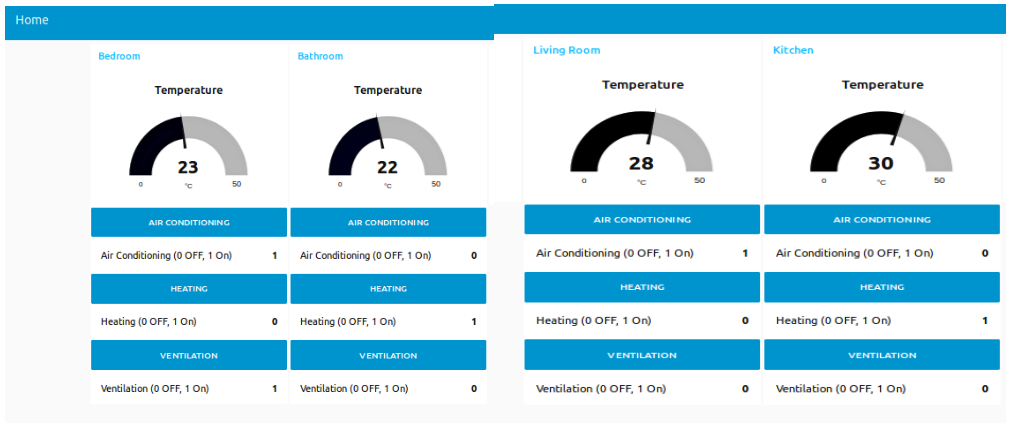


Figure 5, Node red implementation for set and visualise the actuators

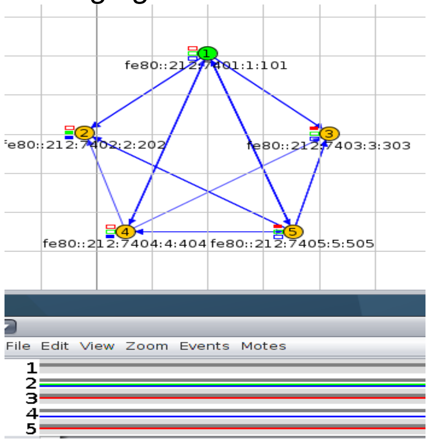
By clicking the buttons we can see how the values of the temperature of each room change. Here we can confirmed that the Air Conditioned and the Heating are mutually exclusive and when the ventilation is on the value of the temperature increases or decreases by two degrees depending on wich actuator (*toogleHeating* or *toogleAirCond*) is ON, otherwise, it just change by one degree if one actuator in ON.


Home

	Bedroom	Bathroom	Living Room	Kitchen
Temperature	23	22	28	30
Air Conditioning (0 OFF, 1 On)	1	0	1	0
Heating (0 OFF, 1 On)	0	1	0	1
Ventilation (0 OFF, 1 On)	1	0	0	0

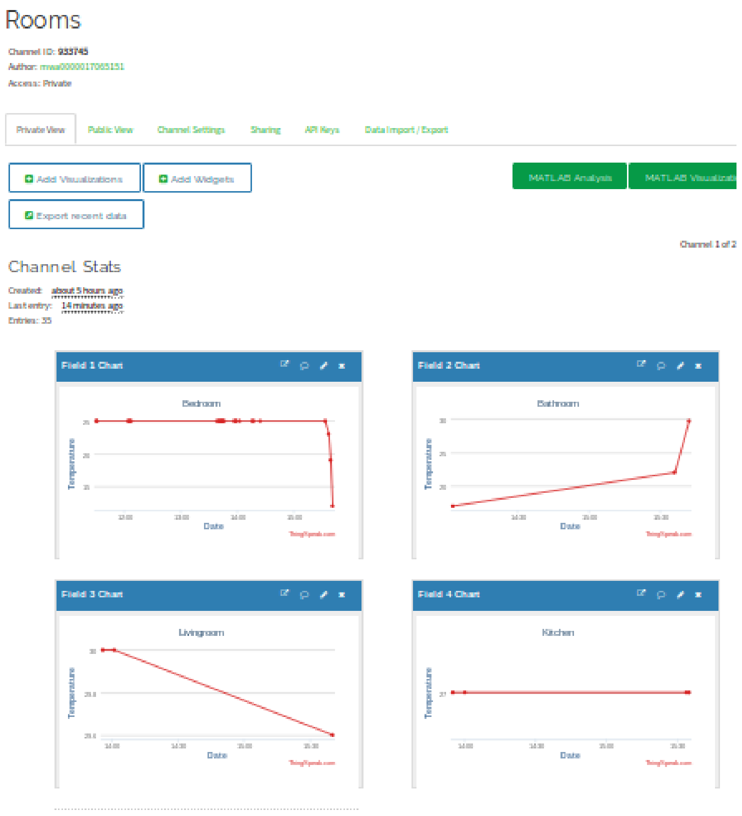
Figure 5, Dash Board changing with buttons

In the Cooja simulation we also can verify the LEDS by changing the buttons in node red.



Thingspeak.

In Node-Red using the *MQTT out* node I publish in one channel the average temperature of each room and in a diferent chanel the temperature of the Home, also making use of the agregator. The info on the MQTT is included according with the channel ID the Write API key and the thingspeak server. i.e *channels/933745/**publish**/fields/field1/8NDE48BXP5ZZFB9I*



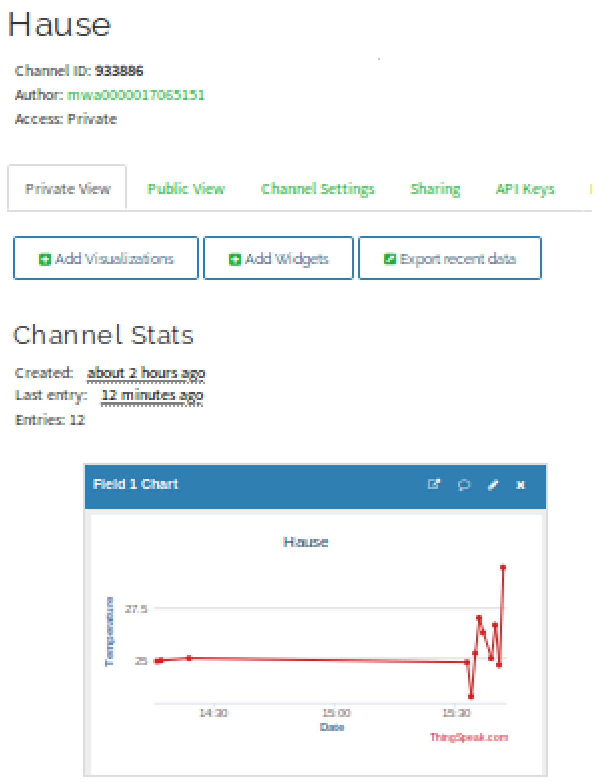


Figure 6, ThingSpeak charts.

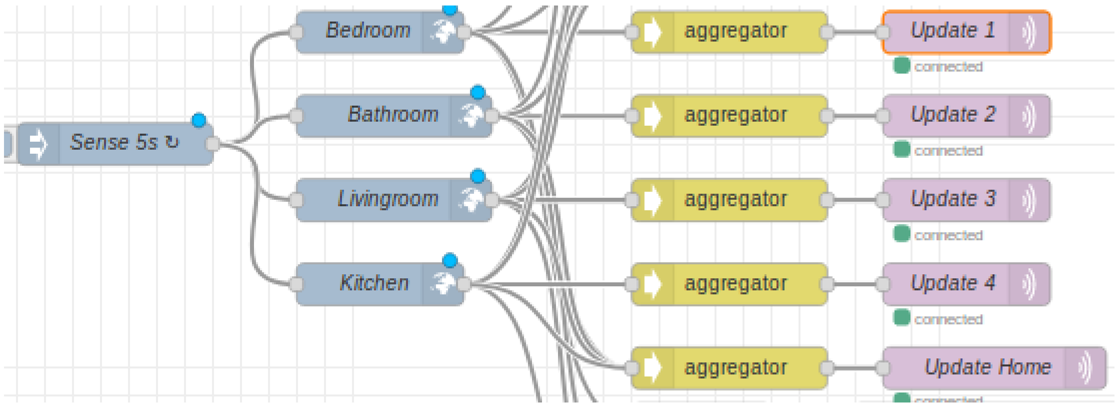


Figure 7, Flow to update chart in Thingspeak

For the subscribe part I use the *mqtt in* node and the *chart* node. The info on the MQTT is included according with the channel ID the Write API key and the thingspeak server.
*i.e channels/933745/**subscribe**/fields/field1/4HVV38KE1TN5AH9C*

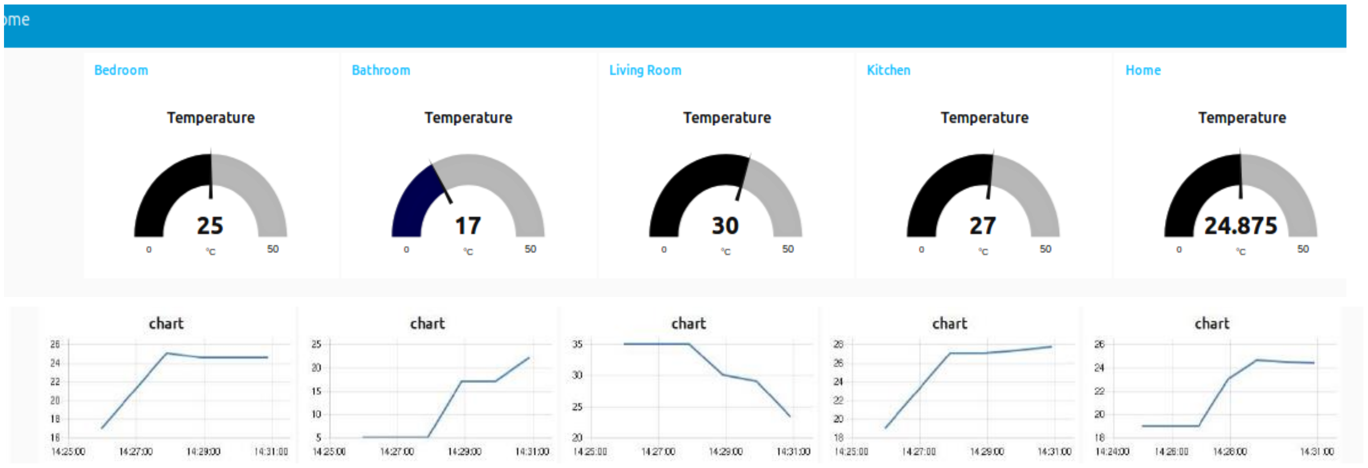


Figure 8, Dash Board with thinkspeak charts

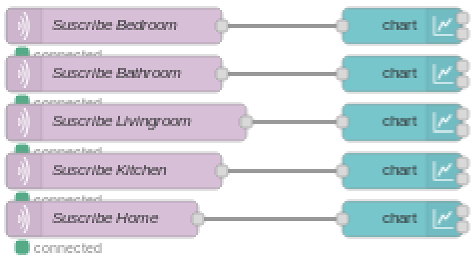
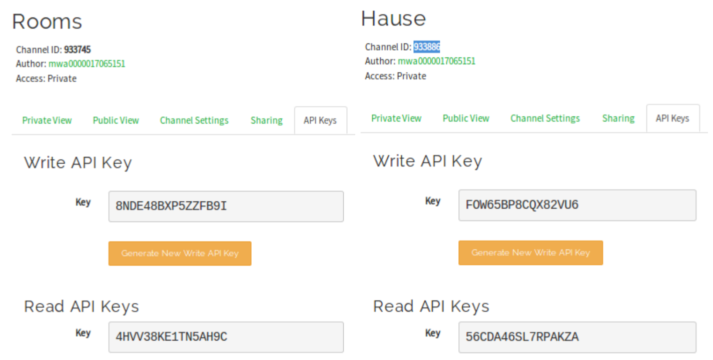


Figure 9, Flow to subscribe chart in Thingspeak



Rooms

Channel ID: **933745**
Author: mwa0000017065151
Access: Private

Write API Key

Key 8NDE48BXP5ZZFB9I

Read API Keys

Key 4HVV38KE1TN5AH9C

Hause

Channel ID: 933886
Author: mwa0000017065151
Access: Private

Write API Key

Key F0W65BP8CQX82VU6

Read API Keys

Key 56CDA46SL7RPAKZA

Figure 10, Thingspeak Channels Info