POLITECNICO
MILANO 1863

Smart Thermostat

Bacarella Mattia 928345
Gallone Emanuele 914114

Release date: 23/07/2019

Version 1.0

1 Setup

1.1 Setup

In order to have a successful execution of our project you have to:

1 - Save the code in folder Contiki-2.7/examples/<custom-folder>. If you want to use another path, modify the Makefile editing the path to Contiki.

2 - Start Cooja, create a RPL-BORDER-ROUTER using a Sky Mote, open the server socket on port 60001 and run in a terminal "make connect-border-cooja"

3 - Create 4 Sky motes using the file TemperatureMoteProject.c making sure that all the 4 nodes are in the TX range of the RPL-BORDER-ROUTER.

4 - Start NodeRed and import the content of the Node-Red-Dashboard.txt file

2 Problem Solving Approach

2.1 Server Side

We have decided to use an RPL-BORDER-ROUTER as router of our projects. All our motes connects to this router sending or receiving data. A variable called "current ratio" is used to simulate the behavior of the temperature in the room:

- if cooling is active, the variable is set to -1;
- if heating is active, the variable is set to +1;
- otherwise the variable is set to 0;

We lead to create two thread in our Temperature Motes. The first one is the main thread, where all our resource are activated and the temperature of our rooms is set. The main operation that is performed in this thread is to control the actual value of the thermostat (in which of the three status options the thermostat is in that moment), and accordingly to this value an operation is performed:

- If heating or cooling options are active, the variable "current ratio" is set accordingly to the state and this process create a new parallel thread, called "update temperature".
- If the new state of the thermostat is different from the previous state of the thermostat, the "update temperature" thread is killed.
- Otherwise the variable "current ratio" is set to 0.

The second thread is going to manage the update of the temperature. The duration of this thread is 20 seconds. When a new process is called, in this thread the current ratio is saved in a new variable. This permits to control that during the execution of this process, the state of the thermostat is not going to be changed. If this happen, the process is killed by the main thread. Otherwise, the temperature is incremented, looking also at the ventilation option.

We create three different resources for each mote:

- STATUS: Using a coap GET method, it is going to send the actual status of our thermostat.
- THERMOSTAT: Using a coap POST method, the user could select the desired state of the thermostat.
- TEMPERATURE: Usign a coap GET method, it is going to send the actual temperature of the room. This could be used also as OBSERVATION MODE.

2.2 Client Side

In Node-Red, we have created one flow for each temperature sensor. In each of this flow we have the control of our thermostat. Firstly, there are three buttons that control the three state of the thermostat. In each button we have set the path of the COAP request that is going to be performed. Each request converge in a single COAP request node. A notification is displayed for each request.

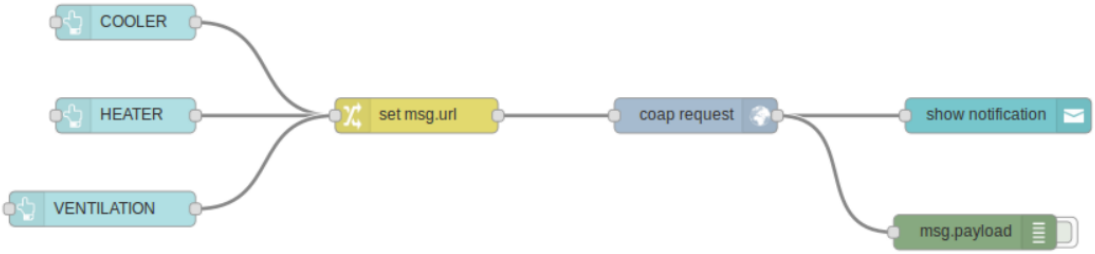


Figure 1: Status button control

Using a coap GET request, we retrieve the actual status of our thermostat, and display it on the dashboard.

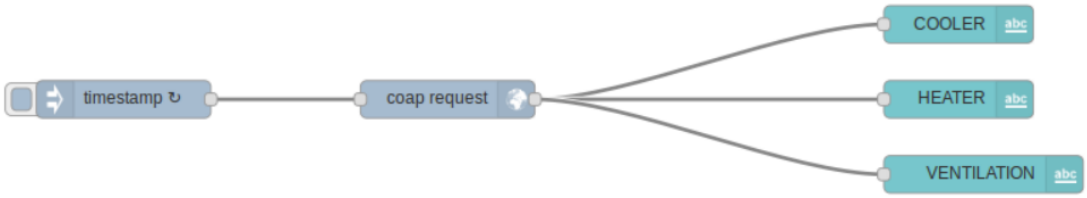


Figure 2: Retrieve actual status of the thermostat

In each mote flow, there is also a subscription to the ThingSpeak MQTT channel, where data of the avg per minute of the temperature of the room are retrieved, displayed on a chart in the dashboard.

Figure 3: MQTT subscription and chart creation

In another flow, the control of our room is managed. For each mote, we retrieve the actual temperature of the room using the OBSERVATION MODE of COAP and display it on the dashboard. Each value is read and stored in an array for 1 minute, then the average is computed by a function node. At the same time, a function controls that the actual value of the temperature do not overcome the boundaries of the desired temperature range. If the temperature is out of range, an email is sent.

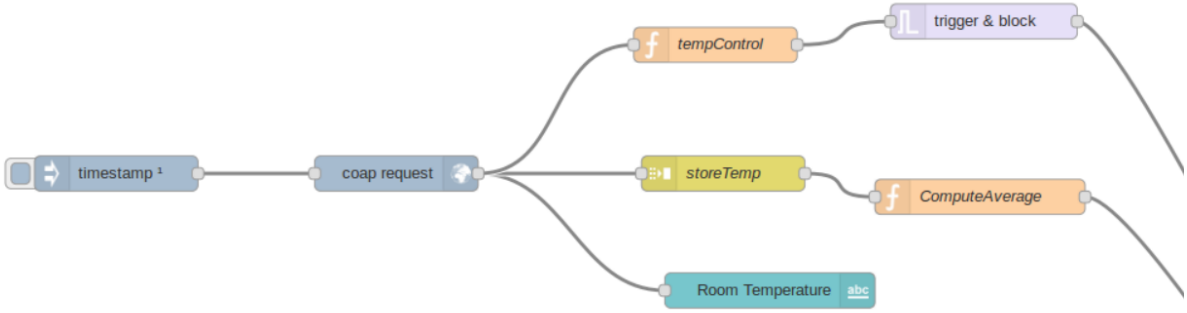


Figure 4: Control center of each mote

When the average is computed, it is sent to a join node, that is used to create the payload of the MQTT publish to the ThingSpeak channel, containing all the motes. This is done also for the entire home, where the total average is calculated and published on another channel.

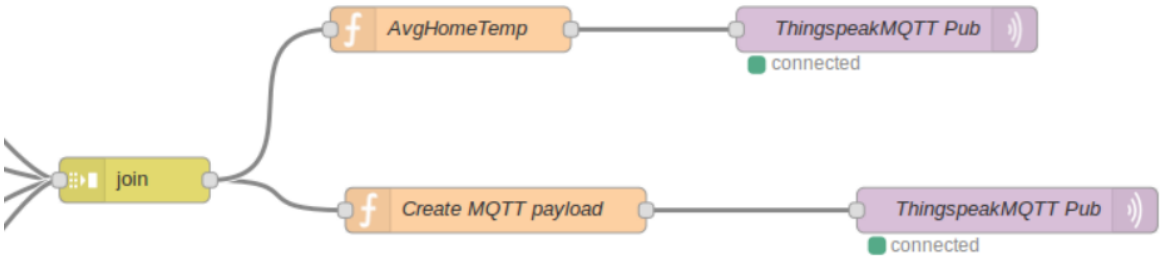
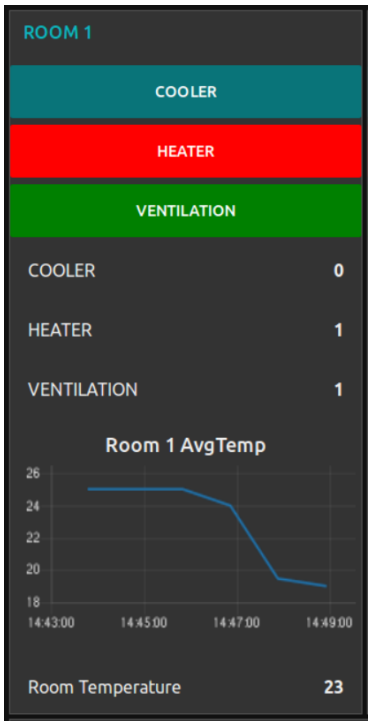
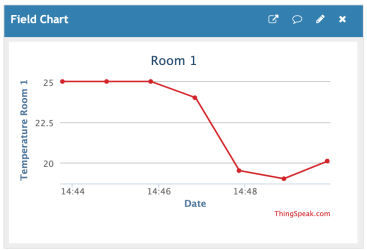


Figure 5: MQTT publish

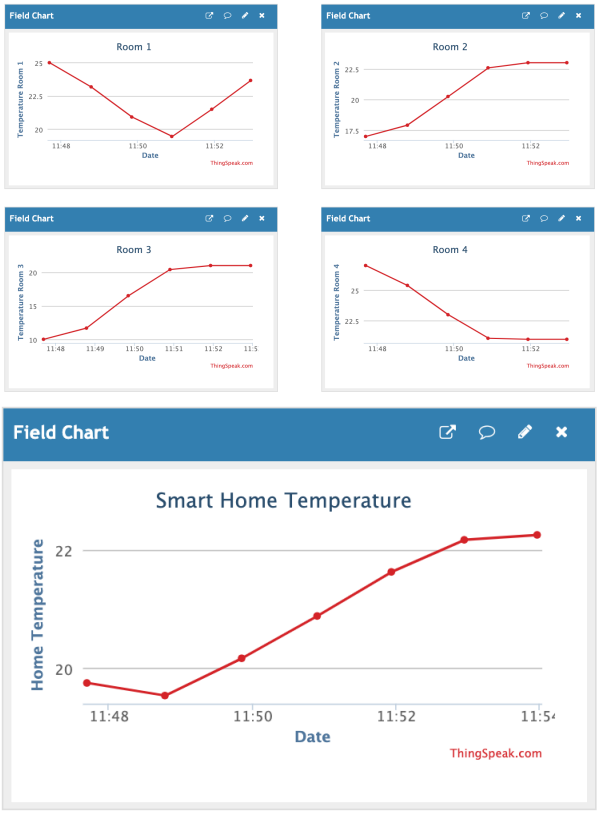
The dashboard results in the following figure, where the chart is retrieved directly from the thingspeak channel.



(a) Room Control dashboard



(b) Corresponding channel field in ThingSpeak



(c) Room channel and Home channel

Figure 6: Dashboard and ThingSpeak

