Internet of Things
Project Report

Patti Alberto - 895416
Sciamanna Filippo - 898515

Smart Thermostat

Project description and requirements

The objective of this project was to implement a set of sensors, ideally placed in each room of a house, which manage and monitor the temperature using Contiki and simulating it in Cooja.
The requirements are as follows:

- turn on/off the air conditioning, the heating and the ventilation unit;
- capture, store and visualize the temperature of each room;
- send an alert when the average temperature is above/below a certain threshold.

Development

The development of the project is split in two parts:

- Contiki and the simulation in Cooja;
- Node-RED.

Contiki and Cooja

Contiki and Cooja objective is to manage the thermostats and their interactions with the user.
All the nodes are implemented in the simulation using Sky motes, one of the simplest motes available in Cooja, as the operation required are not too complex and demanding to require more powerful nodes. These nodes are divided in two groups based on their role and the operation performed in the network:

- RPL Border router (1 mote);
- Sensors (N motes, in our simulation we chose to use 4).

The RPL Border router node creates a network with the sensors using the RPL algorithm allowing the relaying of message to the right node and allowing the user (node-RED) to reach and retrieve the information of the simulation.

The sensor nodes task is to simulate the change in temperature in the room as specified in the project specification and allowing the access to the data to the user.
The simulation of the temperature is simply done by using a timeout (20 seconds) after which the value of the temperature is updated based on the state of the node:

- +1°C if the heating is active;
- -1°C if the air conditioning is active;
- x2 multiplier if the ventilation unit is active.

In order to avoid reaching unreasonable values it was decided to keep the temperature in the range [0°C, 50°C].
The communication with the user is implemented using the CoAP protocol by defining two resources:

- heating_opt (actuators/heating);
- temperature (sensors/temp).

The first one is related to the heating options of the house. A GET request returns the state of the heating device while a POST request allows to manage which device is turned on or off based on a parameter (opt). The accepted value are:

0. Turn on/off the air conditioning if the heating is off;
1. Turn on/off the heating if the air conditioning is off;
2. Turn on/off the ventilation;
3. Turn on the air conditioning and turn off the heating;
4. Turn off the air conditioning;
5. Turn on the heating and turn off the air conditioning;
6. Turn off the heating;
7. Turn on the ventilation;
8. Turn off the ventilation.

The options {0, 1, 2} are meant to be used on a single node while the options {3, 4, 5, 6, 7, 8} are meant to be used on multiple nodes in order to override the status. This distinction was made in order to allow Node-RED to control either the single nodes or all the nodes with one operation.
The temperature resources is related to the value of the temperature read by the sensors. It is able to handle GET requests that are answered with the current value and it is also an observable resource that sends the data every 5 seconds.

Node-RED

Node-RED allows the user to see and interact with the sensor network by connecting to the border router and sending CoAP request. In order to keep everything easier to understand, it was decided to take advantage of the possibility to put everything in multiple flows.

Room flows

For each "room" there is a flow where an OBSERVE request on the temperature resource is sent to the corresponding node in order to start receiving the values. Every value received is stored in the room specific global array variable (needed to calculate the average values) and displayed on a gauge. Each room offers the possibility to send a POST request in order to change the status of the heating devices. These requests use the {0, 1, 2} values for the "opt" parameter. The status of the room is visible in two ways:

1. through the color of the buttons(grey buttons indicate that the corresponding services are off)
2. through the stats button, a GET request is sent and the result is displayed as text.

House flow

The house flow simply evaluate the average temperature in the house by periodically taking the last value received by the sensor of each room. The obtained value is then stored in a global variable. There is also the possibility to send POST requests to all the nodes in order to decide the overall behavior of the heating devices by using the value {3, 4, 5, 6, 7, 8} for the "opt" parameter.

Settings flow

The settings flow defines multiple forms that allow the user to set the parameters required for the email alerts: a range of value can be set for each room and the house so that an email is sent to the set address when the corresponding temperature value exceeds the thresholds set.

Average flow

The average flow is tasked with the evaluation of the average temperature of each room and the house over one minute by taking the values from the global variable set in the previous flows.
The average of the house is immediately displayed on the dashboard and published to a Thingspeak channel while the average of each room are just published to the corresponding field in the Thingspeak channel. These values are retrieved by subscribing to the same Thingspeak channel and then displayed.
The average value are checked (the house average is checked immediately while the room average values are checked after retrieving them through MQTT) to see whether they are in the range set for the alarm and eventually a email is sent.

Problems encountered

During the development of the project, only one major problem arose:
the MQTT subscribe node wasn't able to connect to the channel on one of the machine where the project was tested while the publish nodes work correctly on both. This is probably related to a problem regarding the ThingSpeak MQTT broker.
The video was taken on the machine that didn't show this issue to show that the application works correctly.