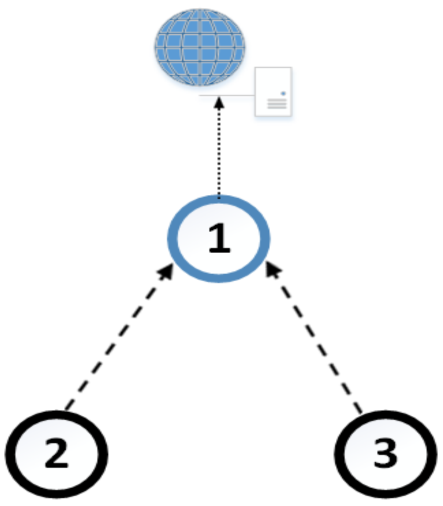
NODE-RED PROJECT - Paper

Politecnico di Milano - Internet of Things (2014/2015)

INTRODUCTION

This project is about the creation of an ad-hoc system to manage data coming from a wireless sensor network, exploiting the Node-RED tool. In the network there is a sink node (mote 1), which aim is just to collect data coming from the peripheral motes, in particular temperature from mote 2 and humidity from mote 3. Moreover, the sink node is attached to a Node-RED socket in order to make data available on the web. After exploring Node-RED functionalities, I succeeded in storing, plotting and monitoring the incoming data.



CHOICES AND ISSUES

Firstly, in order to make the first part of the project easier to develop, I decided to use TinyOS as operating system and its specific programming language nesC. Then, regarding the motes and the network simulator, TelosB (Sky mote) and Cooja were the only possible choices since MicaZ does not work well on Cooja and communication between NODE-Red and TOSSIMLive through serial port is not yet fully supported/tested. During the development of the TinyOS part of the project, since debug statements (dbg) are not supported by Cooja output console, I decided to use the Printf service provided by the PrintfC component that in some way (the output is still distorted but readable) allowed me to verify the correctness of the application on runtime, from data sensing to data reception. There is a big issue, though: if the serial port is turned on, printf statements are identified as serial messages and sent out to Node-RED, along with the temperature and humidity data. Therefore, I turned all the Printf statements and the component into comments right after having seen that data were correctly sensed by the peripheral nodes and received by the sink. Then, using Node-RED debug nodes and checking its output console, I verified the correct reception of serial messages from the simulation.

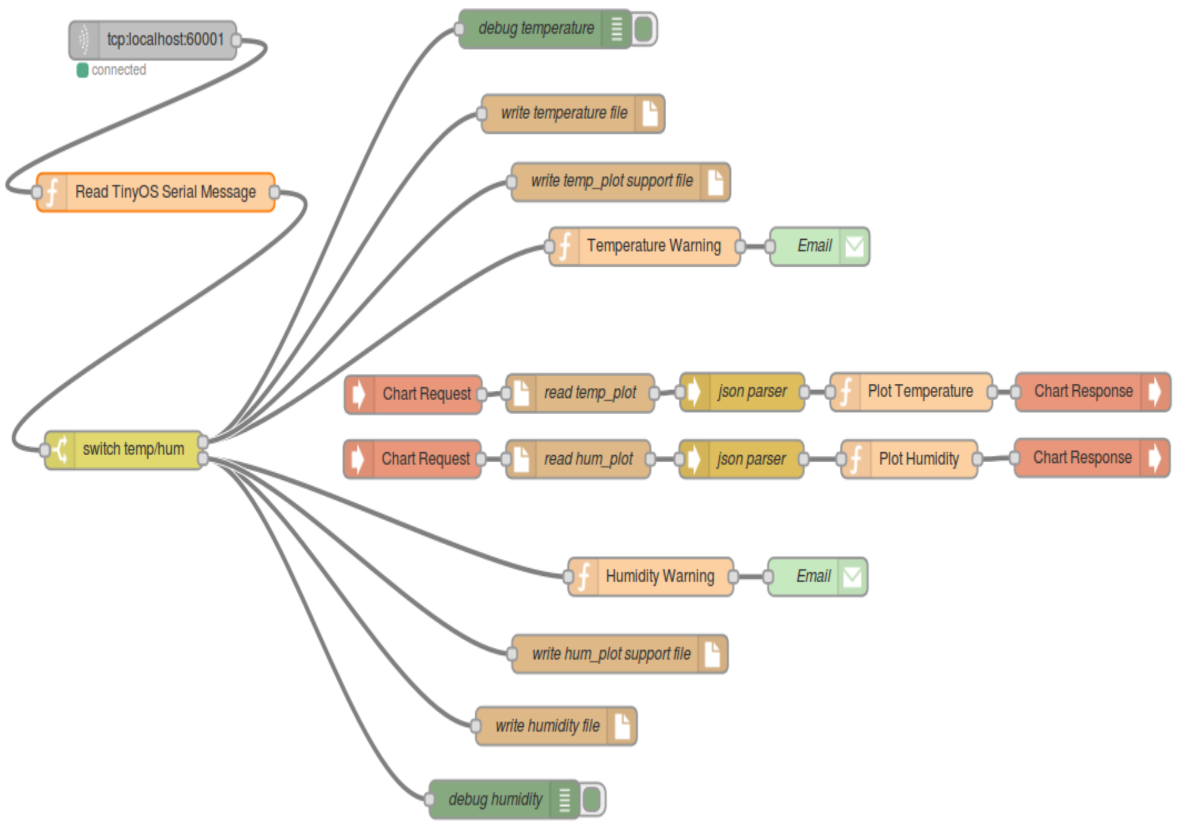
Secondly, regarding the Node-RED part of the project, I decided not to use the suggested Plotly library because navigating *nodered.org* and *npmjs.com* websites and the official Node-RED Google Group, I found out that a Google Chart Node-RED node has recently been developed.

IMPLEMENTATION

I decided to assign to mote 2 the task of sensing temperature and to mote 3 humidity, in order to simulate a real sensor network inside a room for example, because it would have had no sense to collect both data from two separate motes with the same sampled data arrays. So the entire project is based on this division, as I will explain.

When the network starts, the radio is turned on for all motes of the network, and for the sink also the serial port interface. If boot is successful, each peripheral node starts a periodic timer that fires every 2 seconds to read a value from the equipped sensor (mote 2 and 3 have a different base time so that inside the network their timer will expire in different moments, avoiding collisions). This is done thanks to the provided TempHumSensorC component, which simulates a sensor returning values taken sequentially from the files DATA/TEMP_DATA.h or DATA/HUM_DATA.h (measurements over a week, sampled at 15 minutes apart). I made sure that the read values were returned (and then transmitted) over 16-bits unsigned integers in the TempRead.readDone and HumRead.readDone events. Right after a value is read, it is immediately transmitted to the sink with the task SensorTx, in which the packet to be sent via radio is created with type (2 if the value comes from mote 2, that identifies temperature; 3 if it comes from mote 3, so humidity), id (sequential) and value. As soon as the sink receives a radio packet, it creates a serial packet to be sent over the serial port to Node-RED, with the corresponding type, id and value.

Now, the flow that I created on Node-RED (in particular the TCP node), after deployment, is ready to receive the TinyOS serial messages from the Cooja simulation of the network.



I modified the provided "Read TinyOS Serial Message" Function node to match the structure of the payload of the messages coming from my simulation, in particular I set the constant PAYLOAD_BYTES equal to 5 (my data structure has 3 field: unsigned 8-bit integer type, unsigned 16-bit integer id, unsigned 16-bit integer value. So 5 bytes in total). Then, I added useful lines of code in order to get the single fields from the payload and create a message (in JavaScript Object format) to be passed on as output to the following nodes depending on type, id and value of the received serial message. Then, I used a Switch node to manipulate temperature and humidity data separately. In fact, for both I have a debug node to check every message in the debug console, a File node to store them into a file separated by a newline and a "Warning" Function node to monitor the data, followed by an Email node to send alarms in case an upper or lower threshold is violated (26°C - 18°C, 52% - 48%). The email will have the id of the temperature/humidity as object and the current value that has violated a threshold as body text. Thanks to the "Warning" Function, if an alarm has already been sent for a specific value and the next value is equal, the email will not be sent again (exploiting the "context" variable that keeps hold of a previous value of a message that arrives in a Function node).

The plot creation was the most difficult part of the project and took the major part of my efforts, since I am a beginner in programming on the internet. I tried to use Plotly with the "http request" and "websocket" Node-RED nodes, but without success since it has a NodeJS API library that can be used in the terminal after starting the NodeJS service, but nobody till now has tried to make it work with data coming from Node-RED (sending dynamically HTTP POST requests with a token to the Plotly Streaming service), so I found no support on the Internet (*groups.google.com*, *stackoverflow.com*, *developer.ibm.com*). Everybody is using Xively or proprietary solutions to plot their data.
Eventually, I worked around the problem installing and using two new nodes ("Chart Request" and "Chart Response") that exploit Google Chart API to plot data and now I will explain how. Firstly, I tried to take the data directly from the Switch Node to create two separate graphs for temperature and humidity, but this solution was unstable while the simulation was going on. Therefore, I decided to create a support file (one for temperature, another for humidity) in which a message at a time is stored always overwriting the previous one. This message is then read from the file and parsed to match again the JavaScript Object format ("File in" + "JSON" nodes). An ad-hoc "Plot" function takes every message that arrives as input and stores their id and value fields in two arrays that will be passed to the "Chart Response" node to draw the chart. I made two separated plots located in two different paths (/temperature and /humidity), that are updated by the "Chart Request" nodes every second, so basically the data is streamed in real time and displayed as soon as it is posted.
I tried a lot to fit both types of data in a single chart, trying different functions that I wrote to cope with the asynchronous nature of Node-RED flows when there are multiple inputs to a single node, but without success. Therefore, the two charts can be watched only opening two different webpages.

CONCLUSIONS

Even though there have been some issues, all the functionalities requested by the project specification have been correctly implemented.