LINGI2146 - Mobile and Embedded Computing Report

Group 2:
Matthieu Baerts, Benoît Baufays
Julien Colmonts, Ludovic Vannoorenberghe

May 25, 2015

Introduction

Some embedded systems are designed to be used as regulators in large environments. Our objective for this project is to design such a regulator using the temperature sensor available on the Z1 modules.

A real application of this system is easily conceivable. Imagine a large hangar that has to be kept at a certain temperature. Wiring the all building would be quite expensive face to the usage of a few wireless modules communicating to each other to perform the cooling job. Some modules would be used to measure the temperature at critical spots of the building and some would be connected to the extractor system. Depending on which spot is reaching the temperature limit, specific fans need to be turned on. A last module could be connected to a central server which could compute statistics, monitor all spots and turn on manually some fans.

For the implementation phase of the project, since we had only two modules, we used one as temperature sensor and border router and one to control the fan. We had no electronic material to connect it to a real fan so we used LEDs as signal to observe the state of the electrically operated switch.

For the measurement part of the project, we had two interesting ideas directly related to the main concern of the initial problem. First, we would use the received signal strength indication (RSSI) to measure the maximum distance where it is possible to have a communication between modules. Indeed, if a large numbers of Z1 are required just to act as routers between endpoints, the project would be less interesting in economical terms. Moreover, the distance between the motes could be used as regulators for the fan power. Secondly, temperature sensor can differ in their measurements depending on several factors. Some measurements would need to be very accurate in work conditions. For employees, a difference of two Celsius degrees could result in a really bad work environment. Our idea is to identify the difference between real temperatures and the output of the temperature sensors in different conditions.

1 Development part

1.1 Our components and solution

Since we had at our disposal two Z1 motes, we balanced the features to make them able to solve the problem described in the introduction.

The first mote will act as a border router. With this responsibility, it will try to connect to all surrounding motes. It will even make the link between Z1 motes world and outside world such as Internet or an external server application. After the connection between the two Z1 motes, the second mote will subscribe to temperature measurements of the first one via REST/CoAP API. When this request is performed, the second will wait for answers from the border router. It has been configured to send the temperature and timestamps value every five seconds. The second mote will then parse these information and compute a mean between the four last received values. Then, it will compute a delta value depending on the threshold — that can be configured by external user via a REST/CoAP API — and the last RSSI value using the following formula:

$$\Delta = (mean\_temp - threshold) * (2 - RSSI/100) \tag{1}$$

We used LEDs to show the activation of the fans. Since we disposed of three of them, they are also used to show this Δ value. The RSSI is used to adjust the fan power depending on the distance between motes. This value is in a range from 0 to 100, the highest value meaning a better reception power level. If we are closed to the temperature sensor, no need to increase the fan speed. A blinking frequency is also set to the Δ value. For instance :

- the threshold is set to 20°C (default value),
- the detected temperature is 23°C,
- the reception is perfect ($RSSI \approx 100$)

It means that $\Delta \approx 3$ based on equation 1. Then 3 is displayed as binary value (red LED means 2^2, blue LED means 2^1 and green LED means 2^0), this is why blue and green LEDs are lighted up 3 times in one second (3 Hz).

If the $mean\_temp$ value is shorter than the threshold, no LEDs are lighted up. It means that the airing is shut down.

Note that we had to backport `ER-COAP` and `REST` engine libraries from Contiki Github repository into the `release-2-7` branch to both support `REST` server with `Observe` feature. With this feature motes can do only one request and regularly receive data. These modifications plus other ones to fix bugs have been pushed to our Github public repository[1]. One bug was still not fixed in the development branch of Contiki and a pull request has been created and accepted by Contiki's developers[2].

Using CoAP/REST server is interesting to use a standard protocol specially designed for this kind of devices. Moreover, we can easily interact with it because applications and libraries exist. For instance, we can easily change the threshold value from a computer connected to the border router with the help of Copper's Mozilla Firefox extension[3]. But if the goal was to use a very light implementation, it would maybe be better to implement a specific solution. E.g. sending JSON content as we did uses some resources that can be seen as useless. We could just send the temperature and timestamp in $1 + 4$ data bytes. But the JSON format is a standard, supported by libraries in many languages and data are human-readable.

1.2 Encountered problems and implementation choices

This is the first time this kind of project was done at UCL. The development environment was so in test too. For students under a Linux based OS, it is easier to directly install Contiki environment on their machine instead of the VM. It seems problems of transmission occur between the VM port bound to the USB physical port. Some sort of buffer would change the transmission rate between the mote and the virtualized Contiki environment that leads to a fail at pushing new applications on the Z1. Behaviour of the transmission module seemed to be random. After doing many tests by changing Z1's scripts, we recommend to use Contiki without virtualization to avoid wasting time. On MacOS X systems, it seems we have less problems with Parallels than with VirtualBox.

During the development phase, we faced one of the main problem related to embedded systems. Indeed, these system have strict limitation in terms of space and computation power. We tried to multiply the features available on a single node and tried to add a HTTP server on the mote already measuring the temperature and doing routing operation. We were able to display statistics about last measurements and show a chart built using Google API directly to users connecting to it. But even if we took care at restricting the number of kept measurements in memory, the application itself couldn't fit into the Z1 memory when using it with a border router and a `REST` engine. We were so

[1]See: https://github.com/ludov04/contiki/tree/release-2-7

[2]See: https://github.com/contiki-os/contiki/pull/1084

[3]See: https://addons.mozilla.org/en-US/firefox/addon/copper-270430/

forced to remove this feature and we imagined that it would be easily available on a real server which would not suffer from these limitations.

Debugging and resolving those bugs plus testing different libraries and reading documentations took us some time. We guess it was not a waste of time because students of next years will now be aware about that and advised to use some protocols specially designed for the IoT. Then, they will have more time to write code, implements more features and test them.

2 Measurements part

2.1 Temperature accuracy

To validate our system, we tested first the accuracy of the temperature sensor. With a real thermometer, we compared the temperature measured using the Z1 mote and the physical tool. Each measure follows the same methodology : we put, in the same area, the mote and the thermometer. We wait approximately 10 minutes before making measurements. We repeated these measurements in different areas with a temperature's range between 7°C to 24°C. We discovered that the temperature given by the mote is always higher than the temperature obtained by the thermometer. This difference comes partially from the fact that we used a USB cable to power up the mote. Another source of the difference can be the presence, near to the sensor, of chips and electric circuits.

With this set of measurements, we were able to calibrate the application. We chose to remove 5.76°C, decomposed in two: 2°C for the USB issue [4] and 3.76°C from our measurements.

2.2 RSSI

We measured also the received signal strength indicator (RSSI). This indicator can help us to provide a better fan system. We will be also able to detect the best position of motes in one area.

Let's assume that L is the distance between the mote and the server. The methodology of this measurement is simple: we began with a L equal to 0 and every 5 seconds, when a new data is sent by the border router, we increased the L value by 10 centimetres. This measurements were done 4 times in Reaumur computer labs hall.

The figure 1 on the following page shows our results. For each RSSI value, we took the average of values observed during the 4 measurements. To validate

[4]See http://zolertia.sourceforge.net/wiki/images/e/e8/Z1_RevC_Datasheet.pdf, page 13

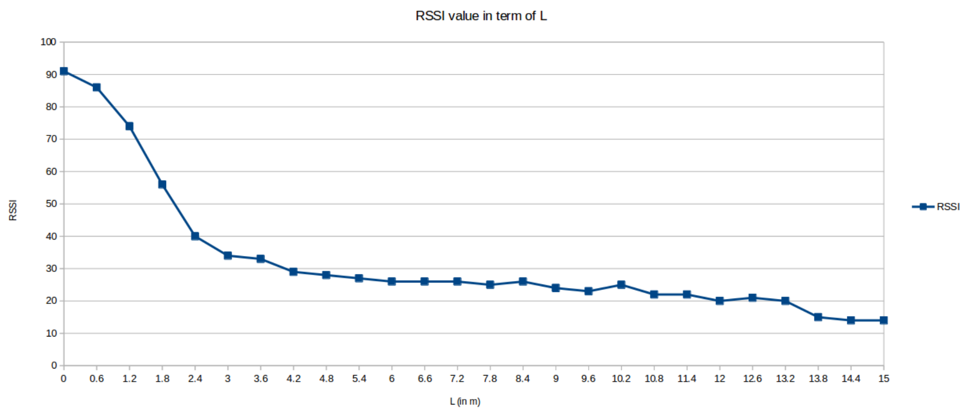

Figure 1: RSSI value in therm of L

this approach, we computed the standard deviation of this average : 0.35, which is very small. The general form of the chart is a decreasing line where the beginning decreases more than the end. After 210 cm, we didn't always received the RSSI value. Note that having electronics devices like computers next to the motes seemed to decrease the RSSI value. Of course, adding an antenna to these Z1 devices will increase RSSI values and cause a higher power range.

This RSSI value can then be used to estimate more precisely the distance and increase the fan speed.

Conclusion

This project helps us understanding the future of remote objects via a entire system dedicated to a temperature control. We divided the system into parts: a server and a temperature mote. With this approach, it will be very easy to add more motes to increase the monitored area or the measurements precision.