

Wireless Network

Master IP Paris – Cyber-Physical Systems

Building an IoT application

Émilie THOMÉ

December 2020

Contents

1	Introduction to the IoT application	2
1.1	IoT-LAB M3 board	2
1.2	Design of challenges	2
1.2.1	Automated lamp	2
1.2.2	Meteorological prediction	2
1.2.3	Runner watch	3
2	Architecture of the applications	4
2.1	Automated lamp	4
2.2	Meteorological prediction	4
2.3	Runner watch	4
3	Experiment	5
3.1	Automated lamp	5
3.2	Meteorological prediction	7
3.3	Runner watch	8
4	HTTP vs CoAP	9

1 Introduction to the IoT application

In order to build an IoT application, we have to know the board where the application will be deployed.

1.1 IoT-LAB M3 board

The IoT-LAB M3 board is the one we used in TPs and that we are using in this project. It is based on a STM32 (ARM Cortex M3) micro-controller with an ATMEL radio interface in 2.4 GHz, 3 LEDs (red, green and orange) and 4 sensors :

- **Light sensor:** this measures ambient light intensity in lux.
- **Pressure sensor:** this measures atmospheric pressure in hPa.
- **Accelerometer/Magnetometer:** this provides feedback on an object's acceleration, and can be used to detect movement.
- **Gyroscope:** this measures the orientation of an object in space and can be used, for example, to determine the orientation of the screen of a tablet or a smartphone.

The goal is to sens information and, in reaction to this information, react by printing in the terminal or using LEDs.

1.2 Design of challenges

I want to make a smart home with smart accessories in it. First, I thought about an automated lamp that turn the light on when someone arrive in the front door. Next, I want to develop a simple meteorological tool using the pressure sensor. Finally, I want to make a speed alarm in the watch of runners. I also added the weather prediction to the watch when the running mode is activated.

I think this kind of smart tools are very useful in the day to day life. The lamp helps to see the lock of the door when opening with keys. The meteorological station helps to adapt our day according to the weather (outfit, outdoor activity, running, etc). The watch is very useful for runners who wants to have a constant speed and to be more enduring.

This application responds to current issues but those daily problems will still be the same in the future: we will need to see (whether keys or code or digital door) the entrance of our house, we will need to predict the weather and runners will still need to outperform themselves.

1.2.1 Automated lamp

This automation function is simple but necessary. It detects when it becomes dark and turn the light on. As there is no object detection sensor in the IoT-LAB M3 board I will leave the "when someone arrive in the front door" criteria aside.

1.2.2 Meteorological prediction

I can use the pressure sensor to make daily prediction on the weather using one of the Moreux's tables (North wind during Spring):

- **> 1020 hPa:** Good or fairly good weather. Warm days, cool nights, morning frosts.
- **1006 - 1020 hPa:** Low or high rainfall, of short duration. Cool temperatures.
- **< 1006 hPa:** Rain or snow with wind. Low temperatures.

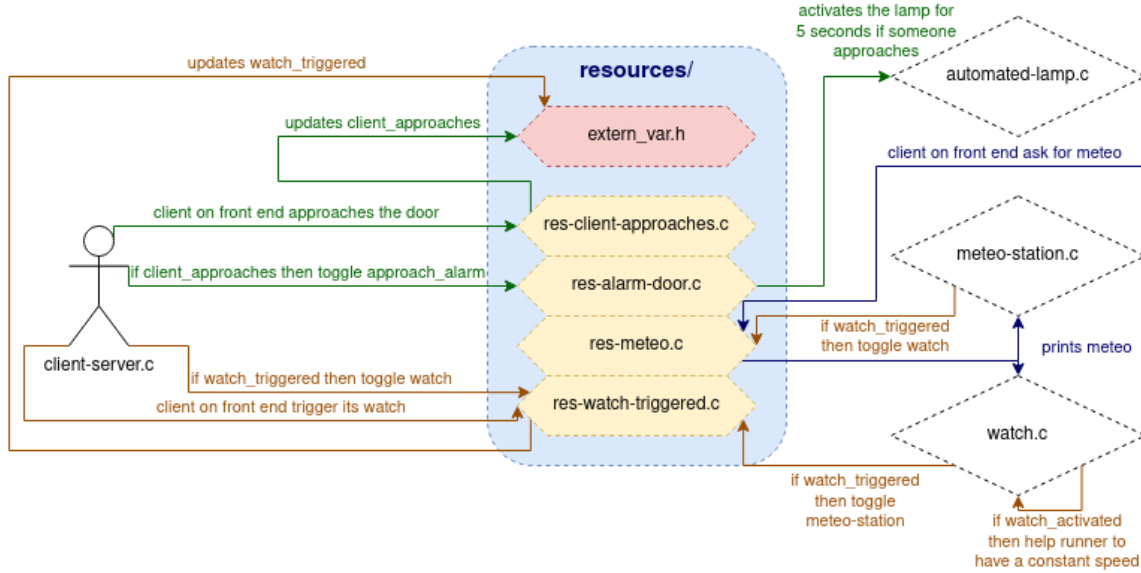
This tool should be combine with a temperature sensor but the IoT-LAB M3 board does not seem to communicate such information.

1.2.3 Runner watch

It is better to have a constant speed when running. This watch will use the accelerometer to detect changes in the speed of the runner when the changes are not suggesting a start or end of the exercise. It will send immediately a message to the watch alarming the runner that its speed decreases or increases. It is also linked to the meteorological station that, when the watch is activated, send to the watch the meteorological prediction in order to aware the user of the weather.

2 Architecture of the applications

As there is no object/person detection with the IoT-LAB M3 board, I use a client-server node to simulate an user. The global architecture is the following:



2.1 Automated lamp

The client send the alarm "Someone approaches." to the lamp server which turn on the light for 5 seconds (I had to make it short for me to test but 30 seconds would be better). If someone else approaches during the 5 seconds, the timer is restarted.

2.2 Meteorological prediction

The station can be used in front-end but it also send prediction to the watch when receiving an alarm informing that the watch is triggered to be activated. It uses the pressure sensor on a resource called **res-meteo.c**, so the prediction can be sent to the watch.

2.3 Runner watch

When receiving a trigger sent by the client, the watch is activated or deactivated (I wrote "disactivated" in the code but it was a mistake). Then it send the same alarm to the meteorological station in order to have weather prediction.

3 Experiment

In this section I will illustrate and explain the experiments realized with my IoT application with screenshots of my computer screen.

3.1 Automated lamp

The first and second experiments show how the automated lamp works.

To recall, it should turn on the light when someone approaches the lamp and it is dark, and so for 5 seconds.

In order to show the lamp status, the terminal representing the automated lamp node prints every second the light and the LEDs status ("Lamp state = on." implies that all LEDs are on). And when the lamp receives the alarm it prints "Someone approaches." and runs the following function:

```
static void turn_lamp_on()
{
    /* Activate sensors. */
    light_sensor.configure(LIGHT_SENSOR_SOURCE, ISL29020_LIGHT_AMBIENT);
    light_sensor.configure(LIGHT_SENSOR_RESOLUTION, ISL29020_RESOLUTION_16bit);
    light_sensor.configure(LIGHT_SENSOR_RANGE, ISL29020_RANGE_1000lux);
    SENSORS_ACTIVATE(light_sensor);

    etimer_set(&lamp_timer, 5*CLOCK_SECOND);
    float light = (float)light_sensor.value(0);
    light = light / LIGHT_SENSOR_VALUE_SCALE;
    printf("light = %f\n", light);
    if ((!lamp_activated) && (light < 0.1)) {
        leds_on(LED_ALL);
        printf("The lamp is on for 5 secondes.\n");
        lamp_activated = 1;
        etimer_restart(&lamp_timer);
    }
}
```

When the global timer `lamp_timer` reaches 5 seconds, the lamp is turned of and it prints "The lamp is turned off".

In the first experiment, I only approach the lamp one time in order to see if it works properly.

The top terminal represents someone approaching the door where the lamp is located, it is the `client-server` node. The bottom terminal represents the automated lamp:

```
comasic3@grenoble: ~
* or the bug-tracker: https://github.com/iot-lab/iot-lab/issues
Last login: Fri Jan  8 09:49:31 2021 from 192.168.1.254
comasic3@grenoble:~$ nc m3-215 20000

Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
Starting 'Receiver' 'Transmitter'
Client approaches the door.
Client approaches the door.
Watch triggered.

Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
Starting 'Receiver' 'Transmitter'
Client approaches the door.
--Toggle alarm door--
[2001:0660:5307:3110:0000:0000:0000:b369] : 5683
|
--Done--
█

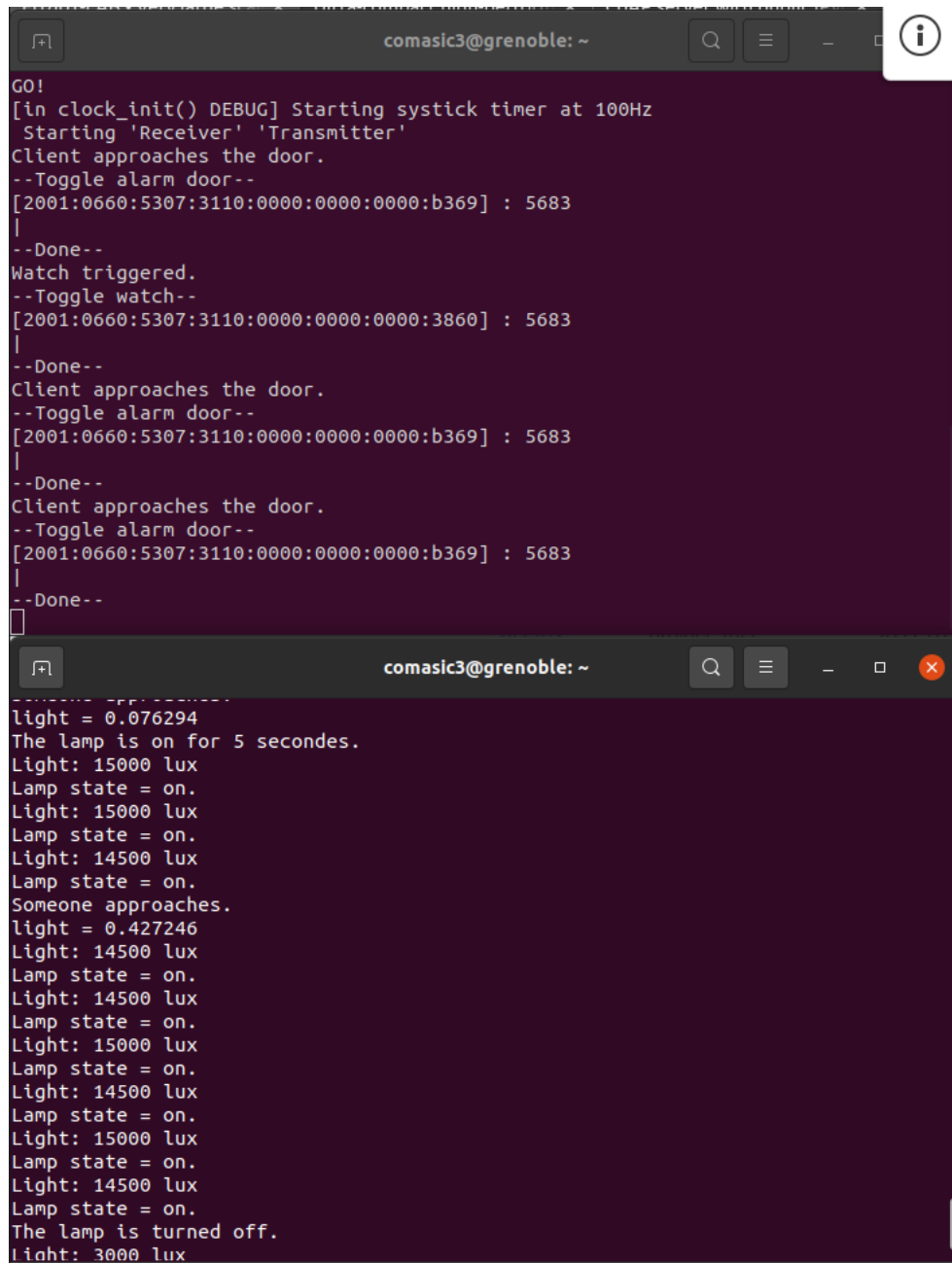
comasic3@grenoble: ~
Lamp state = off.
Light: 3000 lux
Lamp state = off.
Someone approaches.
light = 0.076294
The lamp is on for 5 secondes.
Light: 15000 lux
Lamp state = on.
Light: 15000 lux
Lamp state = on.
Light: 14500 lux
Lamp state = on.
Light: 15000 lux
Lamp state = on.
Light: 15000 lux
Lamp state = on.
Light: 15000 lux
Lamp state = on.
The lamp is turned off.
Light: 3000 lux
Lamp state = off.
Light: 3000 lux
Lamp state = off.
█
```

When I made the user approach the lamp, by typing

```
aiocoap-client coap://[2001:660:5307:3110::9175]:5683/approach_door
```

in the terminal, we can see that the `client-server` node prints that the "Client approaches the door." and, as it is in debug mode, that it alarms the lamp of its presence. And then we can see that the automated lamp works properly as expected.

The second experiment is similar but we want to approach the lamp two times, in less than 5 seconds delay between the two times:



```
comasic3@grenoble: ~  
GO!  
[in clock_init() DEBUG] Starting systick timer at 100Hz  
Starting 'Receiver' 'Transmitter'  
Client approaches the door.  
--Toggle alarm door--  
[2001:0660:5307:3110:0000:0000:0000:b369] : 5683  
|  
--Done--  
Watch triggered.  
--Toggle watch--  
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683  
|  
--Done--  
Client approaches the door.  
--Toggle alarm door--  
[2001:0660:5307:3110:0000:0000:0000:b369] : 5683  
|  
--Done--  
Client approaches the door.  
--Toggle alarm door--  
[2001:0660:5307:3110:0000:0000:0000:b369] : 5683  
|  
--Done--  
[  
-----  
light = 0.076294  
The lamp is on for 5 secondes.  
Light: 15000 lux  
Lamp state = on.  
Light: 15000 lux  
Lamp state = on.  
Light: 14500 lux  
Lamp state = on.  
Someone approaches.  
light = 0.427246  
Light: 14500 lux  
Lamp state = on.  
Light: 14500 lux  
Lamp state = on.  
Light: 15000 lux  
Lamp state = on.  
Light: 14500 lux  
Lamp state = on.  
Light: 15000 lux  
Lamp state = on.  
Light: 14500 lux  
Lamp state = on.  
The lamp is turned off.  
light: 3000 lux
```

It works as expected: the timer is reset to 5 seconds when the lamp see another person approaching the door.

We can pinpoint the fact that there are six lamp status in the supposed 5 seconds. It is just scheduling processes, the info timer must block a second more than expected. I do not think is very important, as it is just a lamp and that the "error" is seen every time (it is deterministic).

3.2 Meteorological prediction

This simple experiment shows how the front-end part of the meteorological station works.

The left terminal is the front-end and the right the station:

```

comasic3@grenoble: ~
n3,216
{
  "0": [
    "n3-216.grenoble.lot-lab.info"
  ]
}
comasic3@grenoble:~$ lynx -dump http://[2001:660:5307:3110::b481]
Neighbors
fe80::3860
fe80::9175
fe80::b768
fe80::b369

Routes
2001:660:5307:3110::3860/128 (via fe80::9175) 1800s
2001:660:5307:3110::b369/128 (via fe80::9175) 1798s
2001:660:5307:3110::b768/128 (via fe80::9175) 55s
2001:660:5307:3110::9175/128 (via fe80::9175) 1785s
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/appro
ach_door
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::b768]:5683/meteo
comasic3@grenoble:~$

Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
Starting 'Meteo station receiver' 'Meteo station transmitter'

Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
Starting 'Meteo station receiver' 'Meteo station transmitter'
Watch activated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--
Rain or snow with wind. Low temperatures.

Platform starting in 1...
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
Starting 'Meteo station receiver' 'Meteo station transmitter'
Watch activated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--

```

We can see that, when typing

```
aiocoap-client coap://[2001:660:5307:3110::b768]:5683/meteo
```

in the terminal, the station prints the predicted weather. Very simple.

3.3 Runner watch

These experiments are quite more complex than the previous ones. They show the interactions between the runner, the watch and the meteorological station.

On the top-left corner we have the runner, the watch on the top right, the station on the bottom right. The bottom left is the front-end.

```

comasic3@grenoble: ~
GO!
[in clock_init() DEBUG] Starting systick timer at 100Hz
Starting 'Receiver' 'Transmitter'
Watch activated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--
Watch disactivated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--
Watch activated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--
Watch disactivated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--

comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$

Starting 'Smart watch receiver' 'Smart watch transmitter'
Watch activated.
--Toggle meteo station--
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683
|
--Done--
Rain or snow with wind. Low temperatures.
Watch disactivated.
--Toggle meteo station--
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683
|
--Done--
Watch activated.
--Toggle meteo station--
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683
|
--Done--
Rain or snow with wind. Low temperatures.
Watch disactivated.
--Toggle meteo station--
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683
|
--Done--

comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$ aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigg
er_watch
comasic3@grenoble:~$

Starting 'Meteo station receiver' 'Meteo station transmitter'
Watch activated.
--Toggle watch--
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683
|
--Done--
Watch disactivated.

```

We can see that, when typing

```
aiocoap-client coap://[2001:660:5307:3110::9175]:5683/trigger_watch
```

in the terminal, the runner trigger its watch (activates or deactivates). The triggered watch informs the station of its status by sending the same trigger alarm as the runner. If the watch is triggered to

The second watch experiment shows that when the watch is activated, the watch helps the runner to have a constant speed by alarming them of acceleration or deceleration:

The screenshot displays two terminal windows side-by-side, both running on a system named 'grenoble'.

Left Terminal Window:

```
comasic3@grenoble: ~  
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683  
--Done--  
Watch activated.  
--Toggle watch--  
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683  
--Done--  
Watch deactivated.  
--Toggle watch--  
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683  
--Done--  
Watch activated.  
--Toggle watch--  
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683  
--Done--  
Watch deactivated.  
--Toggle watch--  
[2001:0660:5307:3110:0000:0000:0000:3860] : 5683  
--Done--
```

Right Terminal Window:

```
comasic3@grenoble: ~  
Rain or snow with wind. Low temperatures.  
You are speeding up too much.  
You are speeding up too much.  
You are speeding up too much.  
You are speeding up too much.  
You are speeding up too much.  
You are speeding up too much.  
Watch deactivated.  
--Toggle meteo station--  
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683  
--Done--  
Watch activated.  
--Toggle meteo station--  
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683  
--Done--  
Rain or snow with wind. Low temperatures.  
Watch deactivated.  
--Toggle meteo station--  
[2001:0660:5307:3110:0000:0000:0000:b768] : 5683  
--Done--
```

During this application project I only used CoAP. As it is a bit complexe and laborious to re-do the same application using HTTP, I will pass the test of my application with HTTP. I am deeply sorry but the experiment setting and the code generation is too long for me to do it a second time.

9