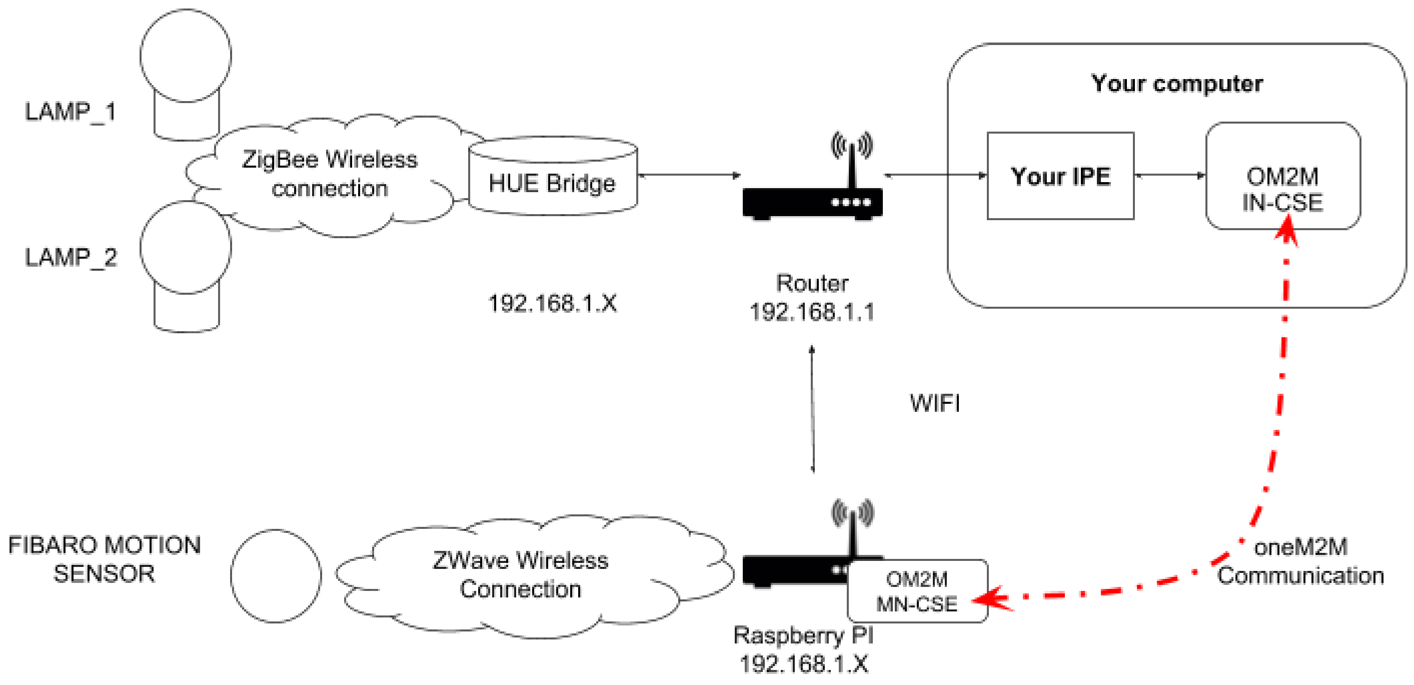
Middleware for the IoT - TP 4

Develop a high level application based on real IoT architecture thanks to *OM2M* & *Node-RED*

Objectives

- deploy a high level application
- deploy a concrete architecture with real devices
- learn to use NODE RED to develop faster the app

The aim of this last session is for you to integrate fully what you have done in the TP2 to the TP3 and have a complete application that will interact with real devices: a multi sensor (Fibaro motion/luminosity/temperature/alarm sensor, connected in ZWave technology) and the HUE Lights connected thanks to your IPE.



The idea here is to have a real life use case where you have devices connected to a middleware. Then you will create a high level application that will use different sources and actuators together thanks to the interoperability provided by a oneM2M middleware.

Deploy the architecture

Connect to the router in WiFi.
First, launch the IN-CSE (remember to check the configuration of you **<u>CSE BASE IP ADRESS</u>**) on your computer and connect your **HUE IPE ADN** to it (if not already done).

Deploy the FIBARO sensor

Start the Raspberry Pi you were given.
Connect the ZWave Sigma sitck (USB).
Connect in ssh to the raspberry pi.
Configure the MN-CSE-PI on the raspberry (ip address of you IN-CSE) to enable the mn to register to your IN-CSE.
Start the MN-CSE-PI
Start the ZWave IPE (given)

Power up the sensor and press the button 3 times. Then, when the device is connected, press the button another time to initiate the data transfer.

High level application

To develop a high level application you can do it two ways:

- implement everything with your HTTP client developed in TP 2 and used in TP 3, but it will be a bit time consuming.
- or use a tool to do so: e.g. NODE RED.

Now that you know how to interact with OM2M through the HTTP REST interface, you will learn to use a tool to develop faster high level applications.

Use the provided documentation on NODE RED to configure the tool, then use it to create your high level application that will implement the proposed scenario.

Configure and use Node RED

Node-RED is a programming tool for wiring hardware devices, APIs and online services in new and interesting ways.
It provides a browser-based editor that makes it easy to wire together flows using a wide range of nodes in the palette that can be deployed to its runtime in a single-click.

Installation

Before installing Node-RED, you must have a working installation of Node.js. We recommend you use the version **LTS 8.x** of Node.js. https://nodejs.org/en/download/

Once installation of Node.js has finished, we are going to proceed to the installation of Node-RED. The easiest way is to use the node package manager, npm, that comes with Node.js. To use it as a global module, add the command *node-red* to your system path.

```
npm install -g --unsafe-perm node-red
```

Integration of node-red-contrib-IDE_OM2M in Node-RED:

As node-red-contrib-ide-iot is published in npm, we can directly use the following command to integrate it into Node-RED:

```
npm install node-red-contrib-ide-iot
```

After the installation of Node-RED and the integration of IDE-IOT, you can run it using the command:

```
node-red
```

You can then access the Node-RED editor by pointing your browser at http://localhost:1880.

Information about node-red-contrib-ide-iot:

Available at: https://homepages.laas.fr/monteil/drupal/content/node-red-ideom2m

OM2MEntity: allows the user to enter the parameters URL, login and password describing the OM2M platform solicited.
NamedSensor: allows the user to retrieve a data instance produced by a sensor designed by its name.

LabeledSensor: allows the user to recover a data instance produced by a sensor described by a list of labels.
NamedActuator: to act on the state of a device by sending a simple command to an actuator designated by its name.
LabeledActator: to act on the state of a device by sending a simple command to an actuator described by a list of labels.
DataExtractor: it is used to extract the useful value of the recovered data instance (device state, luminosity value, etc.)
SimpleCondition: to make a simple condition on the recovered data.
NotificationsHandler: it acts as an HTTP server to receive notifications from the resources to which the user subscribed.

You can also use other pre-existing nodes in Node-RED like:
Switch: routes messages to specific outputs based on received data. It makes it possible to compare the latter with certain values introduced by the user to activate an output or block it.
inject: to allow the user to launch his application: on demand, on a specific date and at specific time intervals.
Debug: to display the results returned by the created user application.
boolan_logic: to perform operations of Boolean logic.

Applications:

- Retrieve the last data instance produced by the luminosity sensor. For that, use the nodes: **OM2MEntity**, **inject**, **NamedSensor** or **LabeledSensor** and **Debug**.
- Extract the luminosity value from the already recovered data instance by using **DataExtractor** node.
- Perform a simple test between this value and a threshold (choose an arbitrary value) using **SimpleCondition** node.
- Depending on the previous result you can turn Off /On your lamps using **NamedActuator** or **LabeledActuator** node. Use the query strings stored in the operation (Descriptor CIN) as the "Command".
- Recover the state of one of your lamps.
- If the first lamp is On and the brightness is below a certain threshold, switch on the other lamp.
- You need to subscribe to the luminosity sensor (Create a subscription resource under the sensor data descriptor using Postman for example), then launch the **NotificationsHandler** node to receive notifications from this sensor whenever a new data instance is produced. Then, you can extract the useful values from these notifications to perform tests and therefore control the status of the lamps.

Deploy a composite app

Now you know how to use Node-RED you can develop your high level app.
This is a first example you can implement. Feel free to create new use cases if you have time.

Connected objects :

- Sensor
- HUE Lights

When a new sensor value is received (presence, luminosity, temperature...), you can implement the following scenario:

- If luminosity value is too high: switch off the lamp
- If medium: turn on the lamp with a smooth brightness
- if too low: switch on the lamp full brightness

- if alarm detected: switch lamp to red light

Once you implemented this example, feel free to create others (even with simulated sensors / actuators).

In a nutshell

The idea here is that you can modify the scenario as you need but the communication will not change even if you integrate new devices to OM2M.
Once you know how to interact with a oneM2M platform, you can do anything with any technology connected and communicate with several platforms in a uniform and standardised way.

What you learned

- deploy a concrete architecture with heterogeneous devices
- deploy several oneM2M nodes (IN, MN, ADN)
- interconnect heterogeneous devices through oneM2M API
- develop a high level application thanks to node RED