INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INNOVATIVE SMART SYSTEMS

INSA Toulouse

Innovative Smart Systems

Middleware for IoT

Alex NOIZE - Mathieu RAYNAUD - Group B2
Karima KHADIR

Promo 2019

January 16, 2019

Contents

Chapter 1

The OM2M standard

1.1 The main standards of the Internet of Things

The OM2M standard is an IoT communication standard which is dedicated to Machine to Machine exchanges. It means that the data exchanged on a OM2M network is not dedicated to a user but to a machine.

OM2M is produced and maintained by a mixed consortium which name is oneM2M. This consortium regroups several standardisation organisms like the European Telecommunications Standards Institute (ETSI) and the Telecommunications Industry Association (TIA), and several manufacturers like Amazon, Huawei, Cisco or Intel.

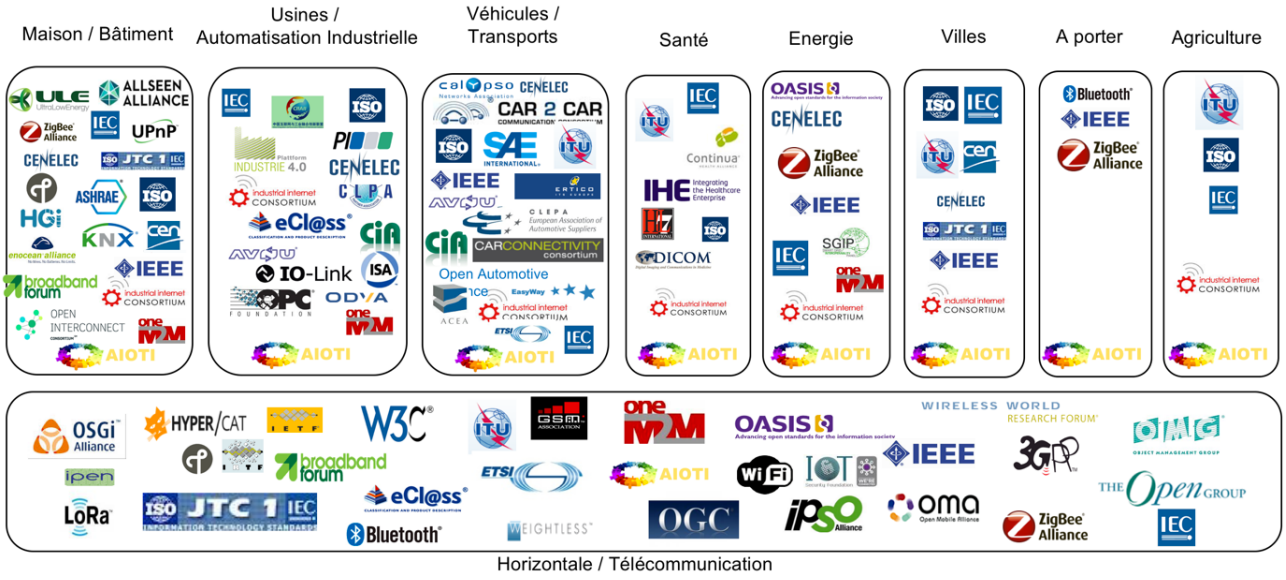

Figure 1.1: IoT communication standards per category

The objective of the OM2M standard is to provide a unique communication standard able to interact with all the existing communication protocols. That's why it is applicable to any domain of the IoT, as we can see in the picture above.

The figure below presents the layers of the OM2M architecture:

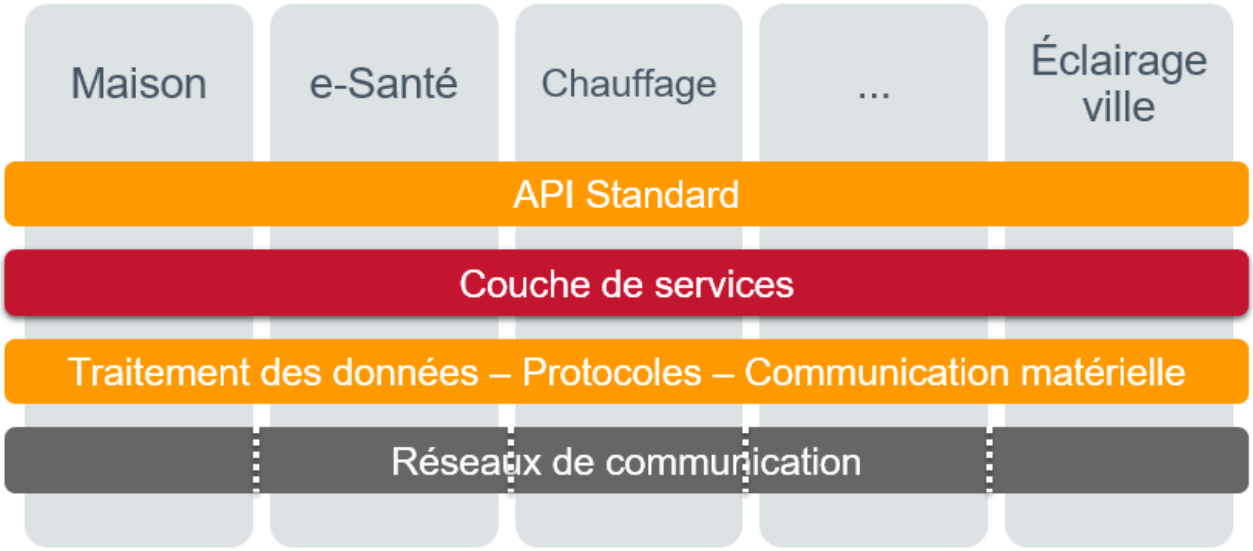

Figure 1.2: Layers of the OM2M architecture

OM2M uses a standard API and a services layer to abstract totally from the different communication protocol of each sensors network. Consequently, the OM2M technology is adaptable to any activity domain, and offers an homogeneous vision of the system, independently from the domain or the technology used.

It means that you can monitor the intensity of the connected heater in your bathroom and the connected doors of your kennel with the same interface.

Chapter 2

Deploy an architecture consistent with a standard and set up a system from sensors network to services

2.1 Deploy and configure an IoT architecture using OM2M

The OM2M architecture contains several entities.
First, the server, in which the OM2M main service is running, is called the Infrastructure Node (IN).
Then, the Application Entities (AE) represent devices which are going to be monitored thanks to the OM2M standard.
Several AE are connected to a gateway OM2M, and this gateway is called a Middle Node (MN), and is connected to the IN.
Each Application Entity can contain one or several Containers (CNT). These containers are kind of boxes in which you can find several Content Instances (CIN).

The Middle Node and the Infrastructure Node are implementing Common Services Entities (CSE). These services are for example the discovery of the connected nodes, the registration of an AE to the CSE, or the subscription to a node and its status to send notifications on status changes.

The figure below shows an example of an OM2M system for a smart meter which is monitored by an end user from his smartphone:

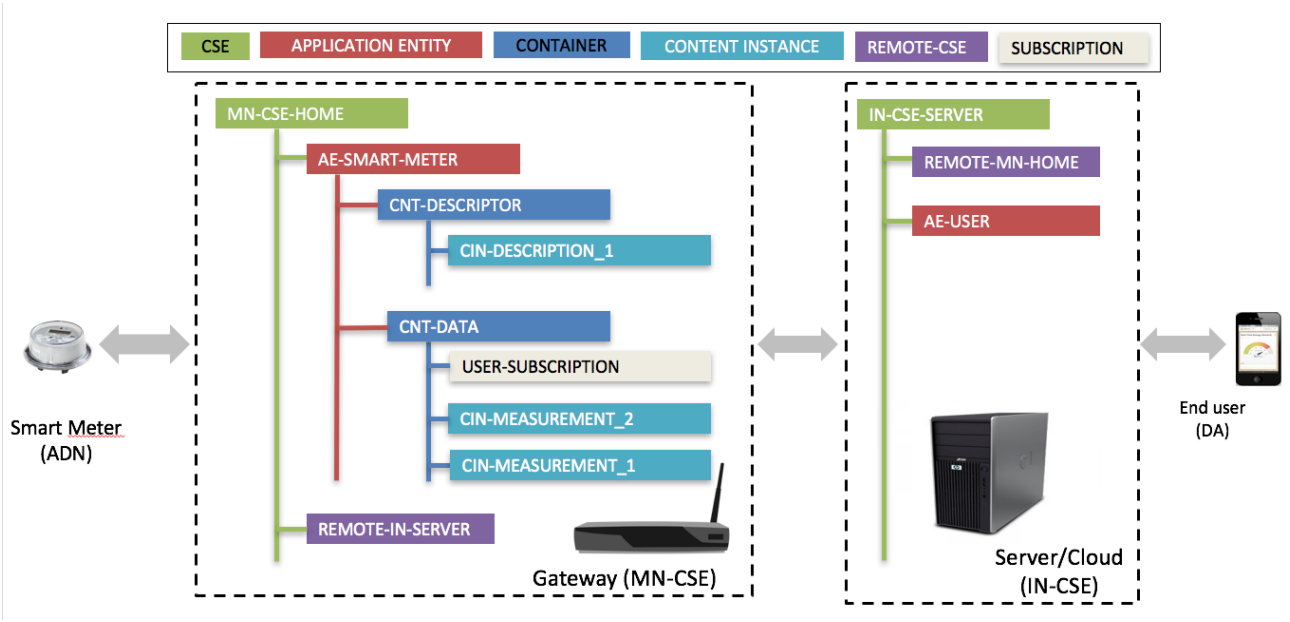

Figure 2.1: An example of OM2M instances and OM2M resources tree

In a first time, we created Application Entities, Containers and Content Instances by sending HTTP requests to the IN-CSE using Postman. We observed that we needed to use different types of requests depending on what we wanted to do:

- GET requests to obtain an information
- POST requests to create something in the OM2M resources tree
- UPDATE requests to update the information of a node in the resources tree
- DELETE requests to delete a node from the resources tree

Then, we developed a RESTful client using Java to interact with our OM2M IN-CSE, and retrieve some information, create and update AE.

Finally, we developed an Interworking Proxy Entity (IPE) in Java to interact with a Philips lamp using OM2M. See the next section to get more information about this topic.

2.2 Interact with the objects using a REST architecture

To interact with objects, we first had to launch the IN-CSE server. In fact, this is the main server of the OM2M architecture, and it is necessary to first launch it before doing something else, otherwise nothing could work. Then, we launched the MN-CSE server to link application entities to it, and to be able to monitor them.

First, we created Application Entities in the IN-CSE and in the MN-CSE using HTTP Post requests. Then, we created Containers and Content Instances.

We also developed a RESTful client able to create resources in the OM2M resources tree , like AE, CT, CIN for example and retrieve data.

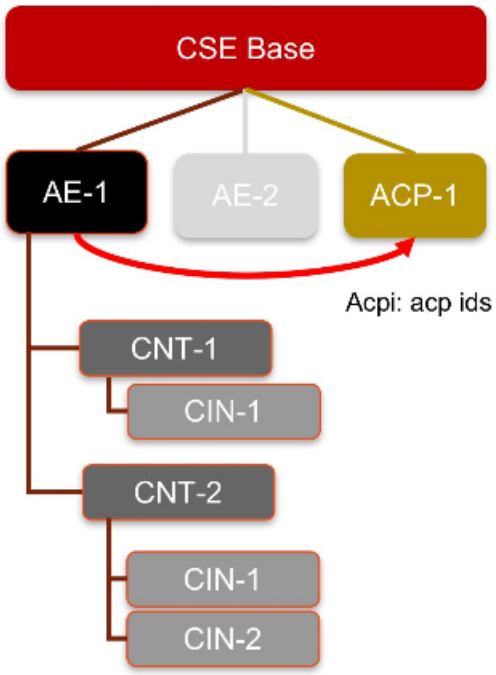

Figure 2.2: Nodes structure in OM2M

If we wanted to add a new device to our architecture, we created a new AE in the MN-CSE server. We added a DATA CT to the AE, and if needed, when we wanted to store information about the device, we created a CIN under the DATA CT.

2.3 Integrate a new or another technology in an IoT architecture

Each sensors network uses its own communication protocol to communicate. But if we want to be able to monitor a sensor using OM2M, we have to use an interface between the two communication standards. This interface is the Interworking Proxy Entity (IPE).

In the figure below, you may find a representation of the OM2M architecture including an IPE:

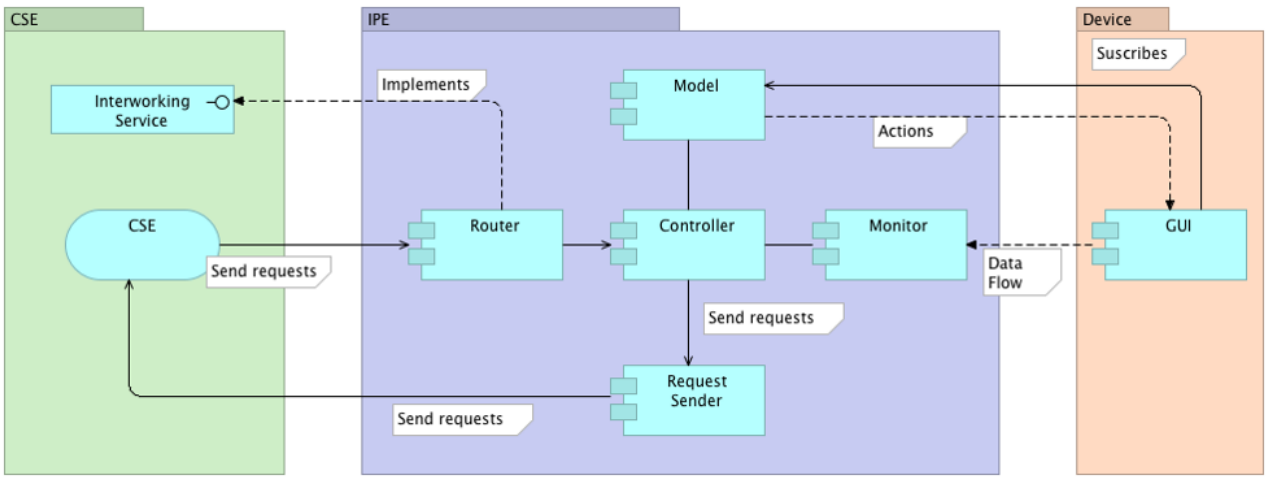

Figure 2.3: The IPE architecture in oneM2M

Thanks to the IPE, the end user does not have to care about the technology used by the sensors network, he only has to use the OM2M interface, and the access to the integrated technology is completely transparent.

The IPE works as following: when it detects a new device, it creates an AE node in the OM2M MN CSE. It is a registration of the device in the OM2M server. Then, it creates relevant containers below the AE, and needed content instances below each container.

Finally, the IPE can also implement additional functionalities to the monitored sensors network.

During the third practical course, our objective was to implement an IPE to monitor a lamp produced by Philips. This is a picture of the lamp to monitor:

Figure 2.4: Philips Hue Go and Philips Hue Bridge

The bridge (bottom right of the picture) communicates directly with the lamp. Our IPE had to interface the communication between the OM2M IN CSE and the bridge.

We implemented the IPE using Java. The program sent HTTP requests to the OM2M server like GET, POST, UPDATE or DELETE when an event occurred. It was also using the Philips Hue libraries to communicate with the bridge, and subscribe to occurring events (like state changes for example).

Our IPE created an AE in the Middle Node CSE called LAMP_1, with two containers: DESCRIPTOR and DATA. These containers contain content instances. In the DESCRIPTOR, the CIN allows the user to switch on, switch off or change the color of the lamp. In the DATA, each CIN registers a state of the lamp: each time the lamp's state changes, a CIN with the new state of the lamp is created in the DATA container.

This is an overview of the OM2M interface after the creation of the several Content Instances:

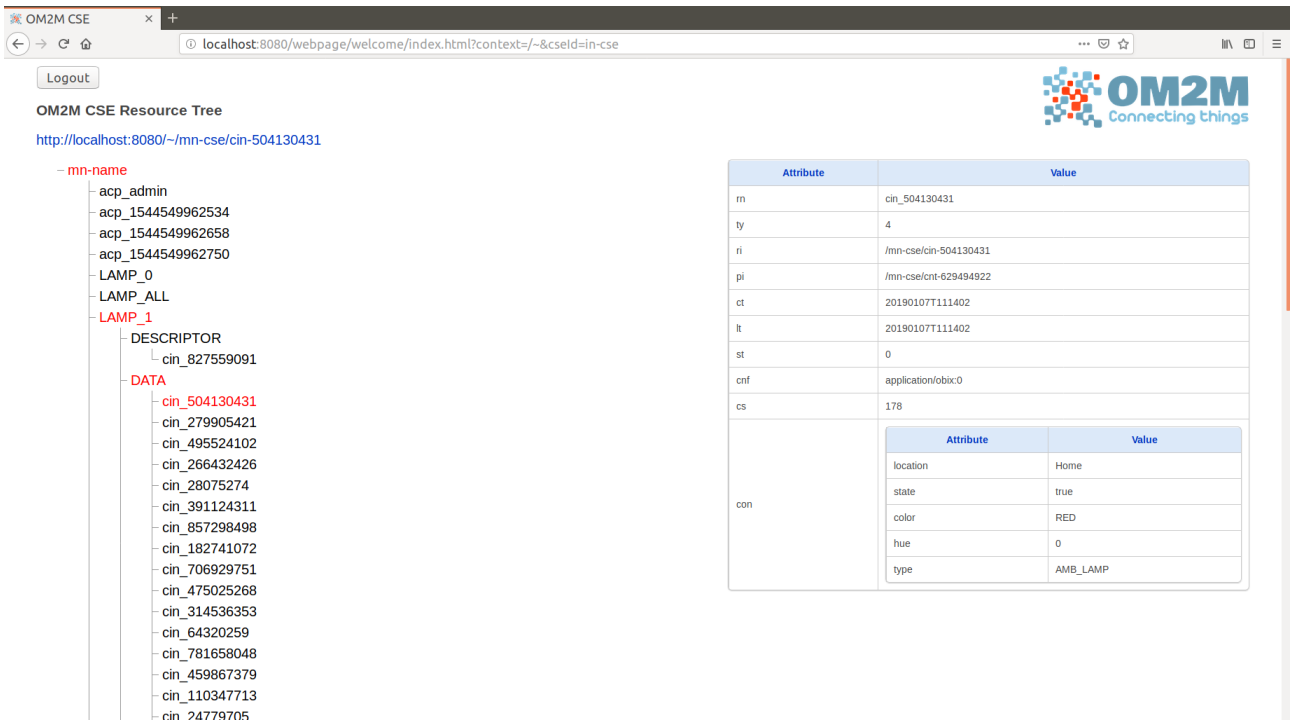

Figure 2.5: OM2M interface with the Philips lamp registered

Chapter 3

Deploy a composite application between several technologies with node-red based on a standardised middleware

Node-RED is a flow-based programming tool and can be easily used because it provides a web browser-based flow editor. It has been designed to wire hardware devices, APIs and online services. All the main types of IoT (MQTT, HTTP) are included in the base nodes provided but what really makes Node-RED powerful is the community developed nodes, for today, oneM2M package but there are a lot more of packages for many purposes.

Our application is pretty simple : we have a Fibaro sensor running on a raspberry Pi with an MN-CSE so we can easily retrieve the sensors data through oneM2M. In the other hand, we're running another MN-CSE on our PC with a little app that emulates two lamps. Thanks to oneM2M and node-RED, we will retrieve the data and compute them to have the desired behavior.

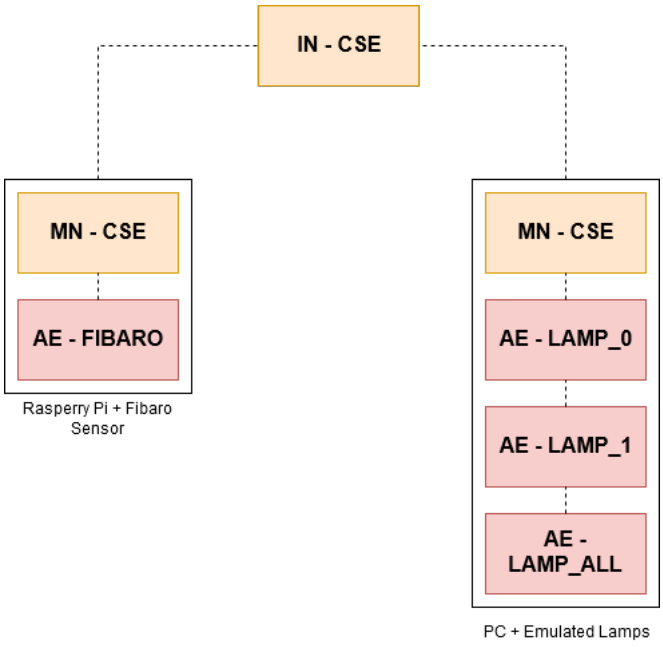

Figure 3.1: Application OM2M architecture

3.1 First application

This first application is really simple, we just retrieve the luminosity data from the Fibaro sensor, test it and if it's below a certain threshold we turn on a lamp. You can see the Node-RED flow below :

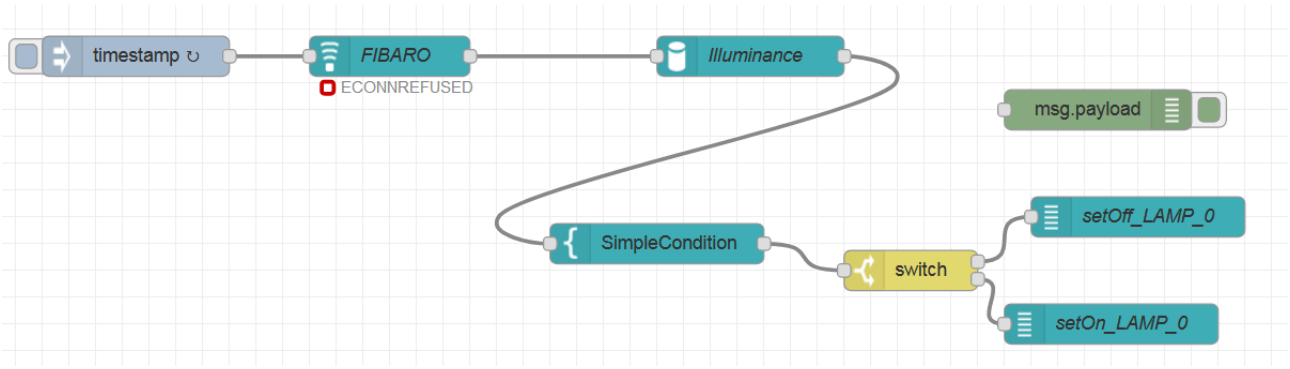

Figure 3.2: First application Node-RED flow

In details, we retrieve the data thanks to a **NamedSensor** node with the SensorName and the wanted Container, these data goes through a **DataExtractor** node to get only the raw data of the illuminance. After this, we compare the data with a threshold thanks to a **SimpleCondition** that goes into a **Switch** node to make the right action with **NamedActuator** node with the right command.

3.2 Second application

The second application a little more elaborated, we still retrieve the luminosity data and compare it with a threshold but we turn on the second lamp only if the first one is on. So we need to get the state of the first lamp.

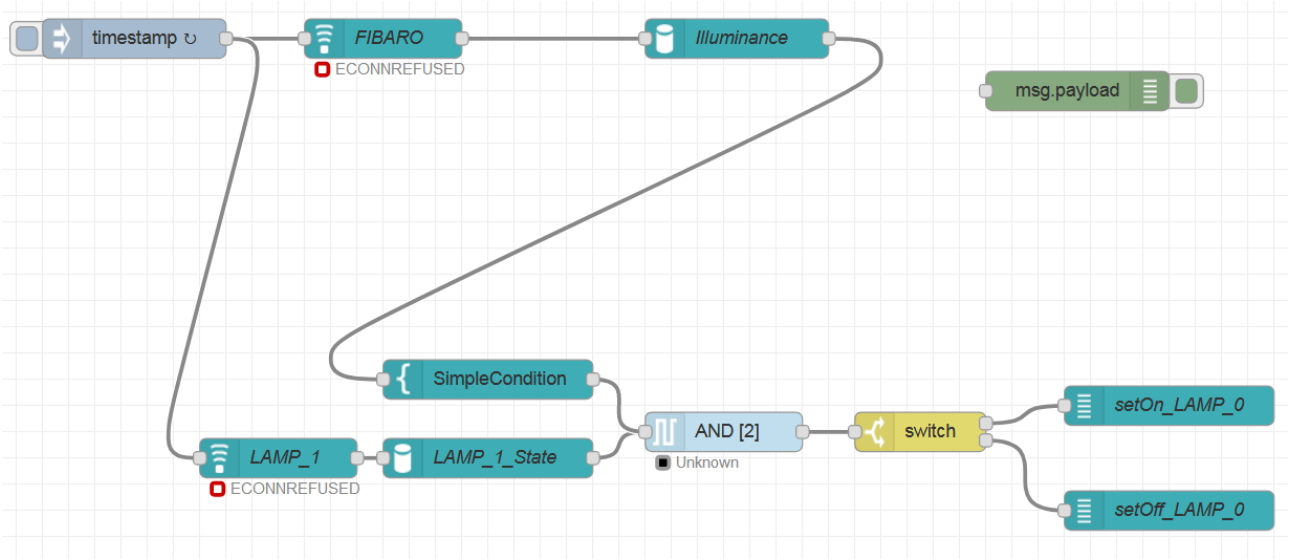

Figure 3.3: Second application Node-RED flow

We retrieve the data of the illuminance and compare it with a threshold exactly the same way as the first application. To get the status of the lamp we also use a **NamedSensor** node and a **DataExtractor** node with

the desired container and data name. An **Boolean logic** node from the boolean logic package is used to make the AND gate and then the actuators are used as in the first application.

Conclusion

During this project, we've applied the concepts seen during the MOOC: we deployed an oneM2M architecture with different types of nodes and different hardware. We have first used a REST Client to interact with the oneM2M resource tree and develop a monitoring application, then, we used an interworking proxy entity in order to connect a Philips HUE lamp that uses the ZigBee technology to an oneM2M system. And finally, we have fully integrated what we've done in labs and developed a high-level application thanks to Node-RED and oneM2M that will interact with real devices.

Thanks to these lab, we are now able to :

- 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.

Bibliography

[1] Fnac.com. Lampe à poser connectée philips hue go led nomade + pont de connexion philips hue bridge. https://static.fnac-static.com/multimedia/Images/FR/NR/14/c1/76/7782676/1505-1/tsp20160129080019/Lampe-a-poser-connectee-Philips-Hue-Go-Led-Nomade-Pont-de-connexion-Philips-Hue-Bridge.jpg.

[2] Thierry Monteil. Node-red for om2m: node-red-contrib-ide-iot. https://homepages.laas.fr/monteil/drupal/content/node-red-ideom2m.

[3] OpenClassrooms. Mettez en place une architecture pour objets connectés avec le standard onem2m. https://openclassrooms.com/fr/courses/5079046-mettez-en-place-une-architecture-pour-objets-connectes-avec-le-standard-onem2m.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INNOVATIVE SMART SYSTEMS

INSA Toulouse
135, avenue de Rangueil
31077 Toulouse Cedex 4 - France
www.insa-toulouse.fr

Université Fédérale Toulouse Midi-Pyrénées

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE
DE L'ÉDUCATION NATIONALE,
DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE