

INTERGICIEL FOR INTERNET OF THINGS

REPORT OF PRACTICAL WORK – 5 ISS INSA TOULOUSE

Authors

Ting Chen
David Laille

Group: B1
Teacher: KHADIR Karima

Introduction	1
1. Be able to position the main standards of the Internet of Things	2
2. To deploy an architecture in accordance with a standard and install a network system from sensors to services	3
2.1. To deploy and configure an IoT architecture using OM2M	3
2.1.A/ Different types of node	3
2.1.B/ Interaction with oneM2M	4
2.2. Interact with objects using REST architecture	5
2.2.A/ Postman	5
2.2.B/ REST Client	7
2.2.C/ Marshal and Unmarshal	8
2.3. Integrate a new technology in an IoT architecture IoT	9
2.3.A/ Presentation of the technology to integrate: HUE Philips lamps and HUE bridge	9
2.3.B/ Retrieving data from the lamps or changing their state	9
2.3.C/ Changing the state of the lamps	10
3. To deploy a composite application thanks to NODE-RED	10
3.1 Presentation of the system	10
3.2 Application with Node-Red	11
Conclusion	11

INTRODUCTION

During the practical sessions, we can experiment a lot of technologies linked with IoT, and especially with the oneM2M standard. This report will examine several topics. The standard oneM2M will be explained, with its advantages and its drawbacks. Then we will show some works done with rest architecture, from sensors or actuators to oneM2M platform, by way of Middleware Node (MN) and Infrastructure Node (IN). We will explore technologies like HUE bridge and HUE lamps (Philips) and finally implement a high level application with Node-Red.

After reading this paper, we hope that the reader could embrace the main concepts of IoT and know better how a connected object can work with the oneM2m standard.

1. BE ABLE TO POSITION THE MAIN STANDARDS OF THE INTERNET OF THINGS

The objective is to explain briefly the principle of the **oneM2M** standard and how it is positioned compared with other standards and existing technologies.

OneM2M offers a generic and horizontal vision of the connected systems, that is to say whatever the domain of application, it will position itself in a transverse way. The idea is that through oneM2M, we will be able to have a homogeneous vision of the system. The figure below shows the structure of a service developed with oneM2M.

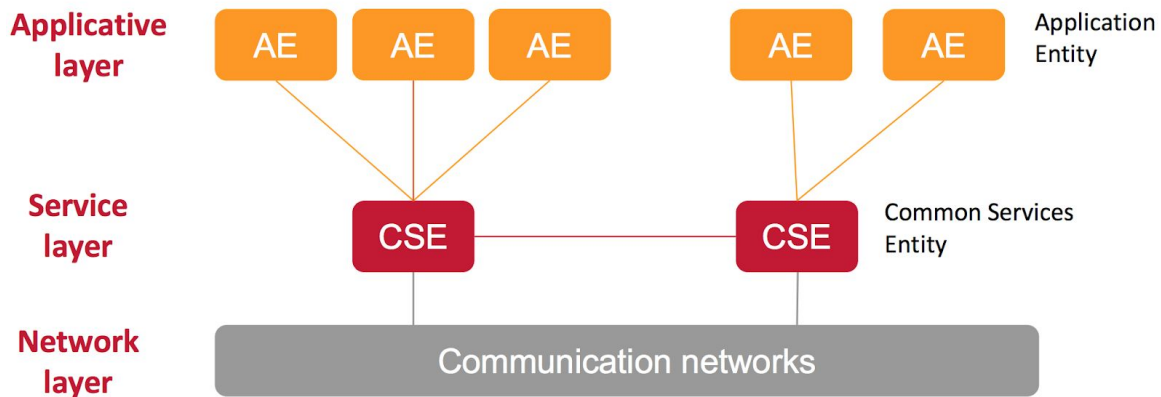


Figure 1: Structure of the different layers in oneM2M

The system will be modeled by different entities. At the application level, the Application Entities (AEs) will be used to represent the different applications connected to our system, the high-level applications, user or even if the objects themselves. Then, the service layer provided by the different systems is represented by CSE (Common Services Entities). These entities will allow applications to register in their system and will provide the various services related to the standard. Finally, the latest layer linked the services to the physical transport of data. The communication networks will be articulated with the layers of services that will abstract it.

	Standard cover	Deployment architecture	Data model	Security
oneM2M	Service: discovery, communication, group management, data, equipment management, network, etc	Based on 3 layers: application, server and network. Deploying on objects gateways and the cloud	Based on a REST architecture and a resource tree	Management authentication, authorization and encryption of communications
Homekit	Connected objects compatible, smartphone and cloud Apple	Framework that communicates and controls connected objects in a home	Database to store various information: houses, rooms, areas, accessories, services	Level of security offered by IOS
LWM2M	Management of connected equipment	Applications, a server and clients on objects	Based on a REST architecture and a resource tree	Secure communication via DTLS
CoAP	Communication on protocol for constrained equipment	Client Server model	Information only on addresses and communication ports	Encryption of communication possible with DTLS

Figure 2: Comparison between several protocols in IoT

Here, we will detail how to deploy an **IoT architecture** with **OM2M**: Types of nodes used, entities interacting with the intergiciel, the level of interaction between objects/sensors and the system, as well as applications interactions with the system.

2.1.B/ INTERACTION WITH ONEM2M

To interact with onem2M platform, we need to follow certain steps.

First, we have to launch the IN-CSE and the MN-CSE (here we run them on a unique computer).

```
in-cse — java -bash — 80x24
~/Documents/OM2M/TP_INTERGICIEL_17_18_BIN/in-cse — java -bash

[INFO] - org.eclipse.om2m.webapp.resourcesbrowser.xml.Activator
Register /webpage http context
osgi> [INFO] - org.eclipse.om2m.persistence.eclipselink.internal.DBServiceJPAImp
1
DataBase initialized.
[INFO] - org.eclipse.om2m.persistence.eclipselink.Activator
Registering Database (JPA-EL) Service
[INFO] - org.eclipse.om2m.core.Activator
DataBase persistence service discovered
[INFO] - org.eclipse.om2m.core.thread.CoreExecutor
Creating thread pool with corePoolSize=5 & maximumSize=50
[INFO] - org.eclipse.om2m.core.CSEInitializer
Initializing the cseBase
[INFO] - org.eclipse.om2m.core.CSEInitializer
cseBase already initialized
[INFO] - org.eclipse.om2m.core.Activator
Registering CseService...
[INFO] - org.eclipse.om2m.binding.coap.Activator
CSE Service discovered
[INFO] - org.eclipse.om2m.binding.http.Activator
CseService discovered
[INFO] - org.eclipse.om2m.core.Activator
CSE Started

mn-cse — java -bash — 80x24
...RGICIEL_17_18_BIN/in-cse — java -bash ...
...RGICIEL_17_18_BIN/mn-cse — java -bash

[INFO] - org.eclipse.om2m.core.router.Router
Response in Router= ResponsePrimitive [statusCode=2000,
content=<?xml version="1.0" encoding="UTF-8"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="LAMP_1">
  <ty>2</ty>
  <ri>/mn-cse/CAE829056886</ri>
  <pi>/mn-cse</pi>
  <ct>20190111T203500</ct>
  <lt>20190111T203500</lt>
  <acpi>/mn-cse/acp-945664834</acpi>
  <et>20200112T203500</et>
  <api>LAMP_1</api>
  <aei>CAE829056886</aei>
  <poa>sample</poa>
  <ch rn="DESCRIPTOR" ty="3">/mn-cse/cnt-873178373</ch>
  <ch rn="DATA" ty="3">/mn-cse/cnt-873924342</ch>
  <rr>true</rr>
</m2m:ae>
,
to=admin:admin,
from=/mn-cse,
contentType=application/xml,
]
```

Then, we can access the Web interface for the Infrastructure Node.

Rapport - Goo

Créez vos pre

RapportTPInt

Google 翻译

Open-Moodle

Cours : Intern

P2_en_activi

OM2M CSE

127.0.0.1:8080/webpage/welcome/index.html?context=/~&cseId=in-cse

应用 苹果中国 必应 iCloud 中国雅虎 新浪微博 Accueil - INSA Google Developers 从 Safari 中导入 Connexion / Conn... Mes projets - Sha...

Logout

OM2M CSE Resource Tree

http://127.0.0.1:8080/~in-cse

- in-name

acp_admin

acpae-889604050

MY_SENSOR

mn-cse

Attribute

Value

ty

5

ri

/in-cse

ct

20190111T173220

lt

20190111T173220

acpi

AccessControlPolicyIDs

/in-cse/acp-819630142

cst

1

csi

in-cse

srt

1 2 3 4 5 9 14 15 16 17 23

poa

Point Of Access

http://127.0.0.1:8080/

After that, we can also access the Web interface to see the Middle Node.

OM2M CSE Resource Tree

<http://127.0.0.1:8080/~mn-cse>

— mn-name

- acp_admin
- acpae-916133441
- acpae-829056886
- acpae-380826795
- acpae-648896811
- LAMP_0
- LAMP_1
- LAMP_ALL
- LuminositySensor
- in-name

Attribute	Value
ty	5
ri	/mn-cse
ct	20190112T202233
lt	20190112T202233
acpi	<div>AccessControlPolicyIds</div> /mn-cse/acp-859715981
cst	1
csi	mn-cse
srt	1 2 3 4 5 9 14 15 16 17 23
poa	<div>Point Of Access</div> http://127.0.0.1:8282/

2.2. INTERACT WITH OBJECTS USING REST ARCHITECTURE

*In this part, we will show how we have been able to interact with objects, visualize the data sent by them, such as resources generated by IPE sample and simulated lamps. We will also explain which oneM2M resources we have used (**CSE-BASE**, **AE**, **CNT**, **CIN**, **SUB**, **REMOTE CSE**), and the way we have interacted with them (HTTP client + server)*

2.2.A/ POSTMAN

Firstly, we have used the software postman to send request to the server IN-CSE. There are four type of requests, GET, PUT, POST and DELETE. GET is for retrieving data, PUT is for modifying the already existed data, POST is for creating a data and DELETE is for removing the already existed data.

Here is an example of the GET request with the postman. We have written the url and two parameters in the Headers. When we sent the request to the server, we could get the response, which is below the request image, encapsulated in XML because of the Headers of the request. We have already defined the type XML. The response gives us the information about IN-CSE. Here we used CSE-BASE (IN-CSE).

GET <http://127.0.0.1:8080/~in-cse> Send

Params Authorization Headers (2) Body Pre-request Script Tests Cookies Code

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	X-M2M-Origin	admin:admin	
<input checked="" type="checkbox"/>	Accept	application/xml	
	Key	Value	Description

Figure 5: GET request using Postman


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <m2m:cb xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="in-name">
3   <ty>5</ty>
4   <ri>/in-cse</ri>
5   <ct>20190111T173220</ct>
6   <lt>20190111T173220</lt>
7   <acpi>/in-cse/acp-819630142</acpi>
8   <cst>1</cst>
9   <csi>in-cse</csi>
10  <srt>1 2 3 4 5 9 14 15 16 17 23</srt>
11  <poa>http://127.0.0.1:8080/</poa>
12 </m2m:cb>
```

Figure 6: Result of the GET request in XML, we received information about IN

Another example is the POST request. Here we want to create a AE, named “TEST_FOR_POST” in IN-CSE. This time we need to complete the body when we send this request. The information is shown in the picture below. After sending this request, we have got the response telling me the AE “TEST_FOR_POST” has been created successfully and the response has returned the information about TEST_FOR_POST. Here we used CSE-BASE (IN-CSE) and AE.

```
1 <m2m:ae xmlns:m2m = "http://www.onem2m.org/xml/protocols" rn="TEST_FOR_POST">
2   <api> app-sensor </api>
3   <lbl> Type/sensor Category/temperature Location/home </lbl>
4   <rr> false </rr>
5 </m2m:ae>
```

Figure 7: POST Request using Postman

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="TEST_FOR_POST">
3   <ty>2</ty>
4   <ri>/in-cse/CAE325265978</ri>
5   <pi>/in-cse</pi>
6   <ct>20190115T215805</ct>
7   <lt>20190115T215805</lt>
8   <lbl>Type/sensor Category/temperature Location/home</lbl>
9   <acpi>/in-cse/acp-793348575</acpi>
10  <et>20200115T215805</et>
11  <api> app-sensor </api>
12  <aei>CAE325265978</aei>
13  <rr>false</rr>
14 </m2m:ae>
```

Figure 8: Result of the POST request in XML, the Application Entity “TEST_FOR_POST” has been created in the Middle Node

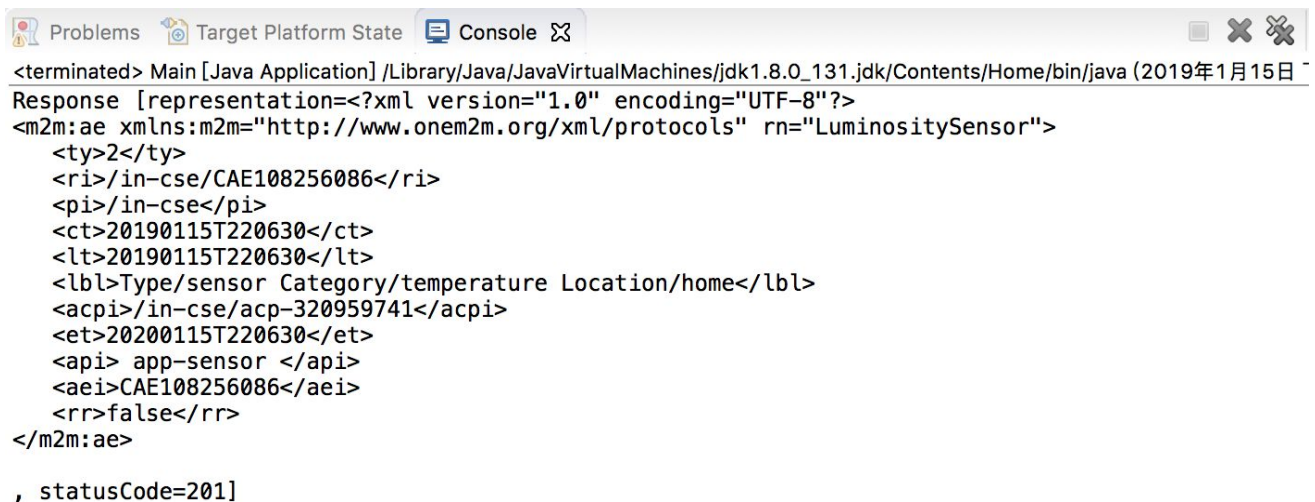
After using postman, we have used the java code to do the REST request. Here in the Client class, we have four functions (GET, PUT, POST, DELETE). And we have an example of GET function, we need to finish the rest three functions. Here this my "create" function which is for POST. We use the provided class HttpClient to do POST. So we just need to write the parameters before doing POST.

We tested to create "LuminositySensor" in server IN-CSE. And in the console, it returns the result about this "LuminositySensor" and the status code. Actually, we already finished and tested all this four functions. They work well.

Below you can see CSE-BASE (IN-CSE) and AE.

```
String url = "http://127.0.0.1:8080/~in-cse";
String originator = "admin:admin";
String representation =
    "<m2m:ae xmlns:m2m = \"http://www.onem2m.org/xml/protocols\" rn=\"LuminositySensor\">\n" +
    "  <api> app-sensor </api>\n" +
    "  <lbl> Type/sensor Category/temperature Location/home </lbl>\n" +
    "  <rr> false </rr>\n" +
    "</m2m:ae>";
String type = Integer.toString(2);
Client cl = new Client();
Response resp = cl.create(url, representation, originator, type);
System.out.println(resp);
```

Figure 9: Java code to initiate a POST request using a Http Rest client



```
<terminated> Main [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (2019年1月15日
Response [representation=<?xml version="1.0" encoding="UTF-8"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="LuminositySensor">
  <ty>2</ty>
  <ri>/in-cse/CAE108256086</ri>
  <pi>/in-cse</pi>
  <ct>20190115T220630</ct>
  <lt>20190115T220630</lt>
  <lbl>Type/sensor Category/temperature Location/home</lbl>
  <acpi>/in-cse/acp-320959741</acpi>
  <et>20200115T220630</et>
  <api> app-sensor </api>
  <aei>CAE108256086</aei>
  <rr>false</rr>
</m2m:ae>
, statusCode=201]
```

Figure 10: Result of the POST request, an Application Entity has been created

Sometimes we need to transform an object to XML format, and sometimes we need to do the reverse action. These two functions, marshal and unmarshal, are used to do that. Here is a test about marshal and unmarshal. Firstly we transform an object to XML format and we print the XML format. Secondly, we transform the XML format to an object, then we change a parameter and transform the new object to XML format and print it. We can see the result of this test goes well.

```

public static void main(String[] args) throws IOException {
    MapperInterface mapper = new Mapper();

    // example to test marshal operation
    AE ae = new AE();
    ae.setRequestReachability(false);
    ae.setAppName("test");
    System.out.println(mapper.marshal(ae));

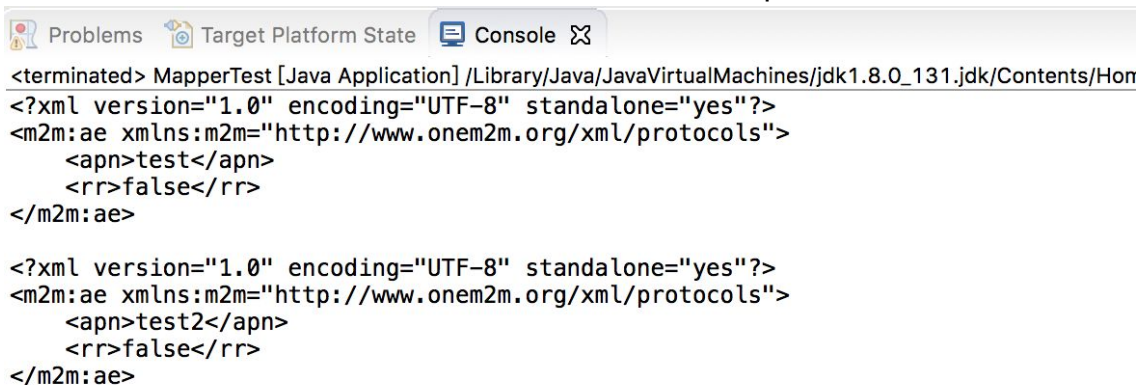
    // get the XML representation, parse it with unmarshal operation

    AE ae1 = new AE();
    ae1 = (AE) mapper.unmarshal(mapper.marshal(ae));
    ae1.setAppName("test2");
    System.out.println(mapper.marshal(ae1));
}

```

Figure 11: Code example to test our marshal and unmarshal functions

Below, you can see the result of the marshal and unmarshal operations.



```

<terminated> MapperTest [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Hon
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols">
  <apn>test</apn>
  <rr>false</rr>
</m2m:ae>

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:ae xmlns:m2m="http://www.onem2m.org/xml/protocols">
  <apn>test2</apn>
  <rr>false</rr>
</m2m:ae>

```

Figure 12: View of the java console after executing the marshal and unmarshal tests

With the previous parts, we have seen how we can interact with the Middle Node and the Infrastructure Node. We also learned that we can use different formats to transfer data. The only problem we have to think of is the compatibility and the ability to understand the data sent. And in the standard oneM2M, it is preferable to use XML.

We also saw that it is possible to send request by several ways. We can use a dedicated software like Postman or create a REST client.

2.3. INTEGRATE A NEW TECHNOLOGY IN AN IoT ARCHITECTURE IoT

To explain the principle of Interworking Proxy Entity of oneM2M, we will see what architecture is used to integrate into the intergiel a new technology, not initially compatible with the standard (software architecture, oneM2M nodes and resources used to build the interface)

2.3.A/ PRESENTATION OF THE TECHNOLOGY TO INTEGRATE: HUE PHILIPS LAMPS AND HUE BRIDGE

As we have seen before, communicating from a lamp (or sensors or actuators) is quite mastered with oneM2M standard. In the following part we will see how to integrate an other technology to ensure the communication between a device and the oneM2M interface. In our case, we use lamps HUE Philips. To connect them to the network, we use the ZigBee communication protocol coupled with a HUE bridge. That technology allows us to link the lamps with a router and then with the oneM2M interface (MN and IN). For more details, you can see the illustration below.

Architecture to deploy

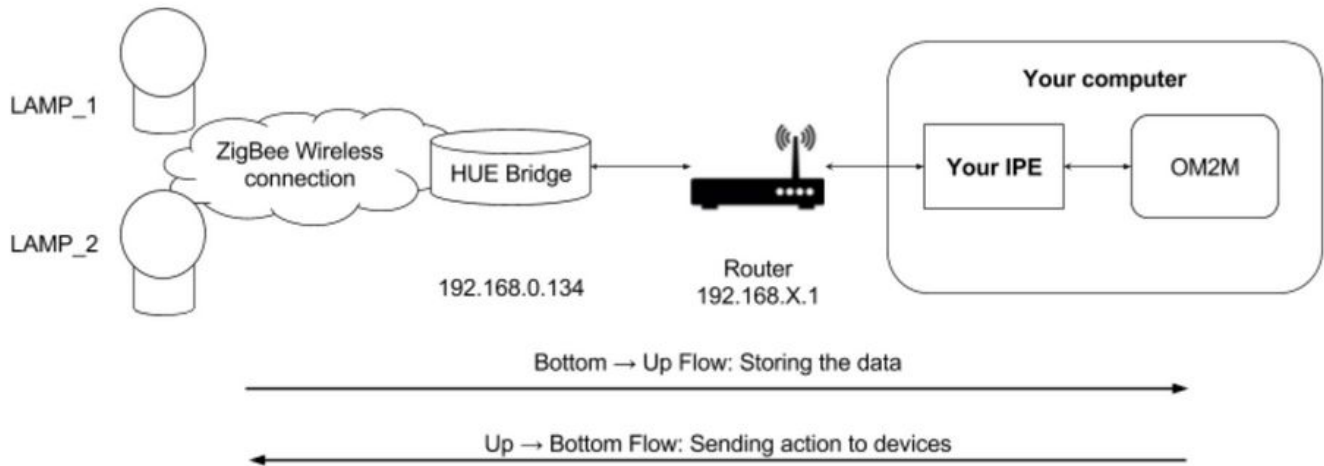


Figure 13: Illustration of the connection between HUE lamps and oneM2M platform

All the work will consist in designing and coding the interactions between oneM2M platform, through the IPE. To make the system function, we need to program an IPE which make the link, which translate oneM2M instructions into an understandable language for the HUE lamp. To do that, the constructor, Philips, provides a SDK (Software Development Kit) in java. It contains some packages and functions useful to configure the communication and transmit the different messages.

2.3.B/ RETRIEVING DATA FROM THE LAMPS OR CHANGING THEIR STATE

First, we connect the bridge to the oneM2M platform thanks to the router. To do that, we need the IP address of the bridge and other information gathered in the package *HueProperties*. That step make the connection between the bridge and the PC (infrastructure oneM2M).

After that, we have to create the resources, here the lamps. That step is done by some methods of the class *HueSdkListener*. These methods are simply *getAllLights* and *createResources*. These methods use HTTP requests to create the AEs “Lamp_1” and “Lamp_2” on the oneM2M platform.

When the bridge is connected and the resources are created, we will listen the changes thanks to *onCacheUpdate* which receive all the changes on the cache of the bridge. We can notice changes with the method *getStates* (in the class *HueUtil*).

2.3.C/ CHANGING THE STATE OF THE LAMPS

To modify the state of a lamp, we need its identifier and the changes we want to apply. Then, we can use the controller or the class *HueUtil*.

3. TO DEPLOY A COMPOSITE APPLICATION THANKS TO NODE-RED

In this final part, we will briefly explain the principle of functioning of NODE-RED and the development of a composite application using different technologies.

3.1 PRESENTATION OF THE SYSTEM

The aim of the following part is to have a complete application that will interact with real devices. This application will be done with Node-Red and will connect a multi sensor (Fibaro motion/luminosity/temperature and alarm sensor) thanks to a Z-Wave wireless connection. It will also connect 2 HUE lamps by a ZigBee connection, a HUE bridge, and an IPE. Below, you can see a view of the system.

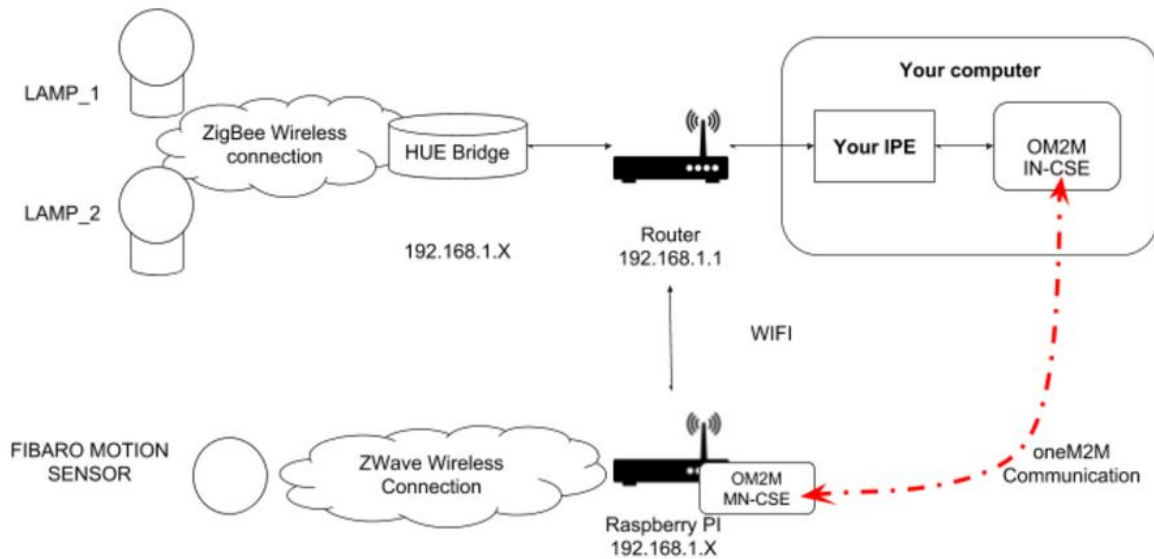


Figure 14: Illustration of the connections between HUE lamps and IN, and between Fibaro sensor, MN and IN

As you can notice, the middleware node (MN) is located in a Raspberry PI, and it provides interoperability, what is very useful to measure phenomenon from sensor, and also activate some actuators.

3.2 APPLICATION WITH NODE-RED

Node-Red is a programming tool which allows to link devices, APIs and high level services. Below, you can see an example of basic application done with Node-Red. The blocks “timestamp” and “msg_payload” are originally present in Node-Red. The other ones are located in a package node-red-contrib-ide-iot.

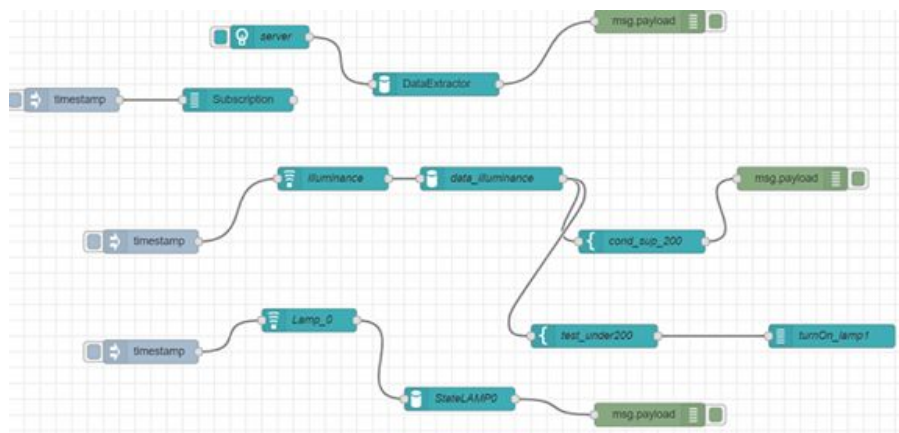


Figure 15: Schematic of the Node-Red application

This application has three main functions.

First, it makes a link from oneM2M platform by subscribing to it. The principle of a subscription is described as follow. Every application which subscribe receive data when the oneM2M platform receives a new data. For each sensor, there is an attribute saying which application or server subscribe to it. Like that, the platform can send data to subscribers when there is updates.

Then, the line with the block “illuminance” retrieve a measure of luminosity and if it is too low, we turn on the lamp. If not, we just display the data.

The last one allows to get the state of a lamp and display it.

The main advantage of Node-Red is that it allows to create a high level application. That means that we can visualize in a very simple view the connections and interactions.

Through the previous part, we have seen how to integrate new technologies into a oneM2M architecture, at which level and with what consequences. Thanks to a practical experience of some IoT devices and technologies, we have been able to embrace the main principles of the oneM2M standard, and most importantly the concepts and the way of reasoning in this field.

CONCLUSION

Through these practical session, we have seen that oneM2M standard cover a large amount of use cases and is adaptable to several technologies thanks to its horizontal architecture. The reader can also notice that it is easy, or at least feasible to interconnect heterogeneous devices and technologies through oneM2M API. And by creating some high level applications, we can better visualize the transmission of data and display it in a convenient way.

It was an interesting work to take up and apply some important concepts of the Internet of Things. The only question which still remains is the legitimacy of the standard oneM2M. And it is always the same issue with standards, who will use it and which one will gain international recognition.