

# Middleware for the IOT

## Practical Work Report

Luan TRUONG - Michael EJIGU  
Groupe B1 5 ISS 2020

---

### Table of contents

<b>Part 1 : MQTT</b>	<b>2</b>
MQTT	2
NodeMCU ESP8266 Board	2
Use case	3
Video	4
Code	4
Analysis	4
Ressources	4
<b>Part 2 : OM2M</b>	<b>5</b>
OM2M architecture	5
OM2M architecture binary :	9
Analysis	10
<b>Part 3 : Node-Red</b>	<b>10</b>
Node-Red	10
Workflow	12
Setup to use Workflow	13
Analysis	13
<b>Conclusion</b>	<b>13</b>

## Part 1 : MQTT

### MQTT

- **What is the typical architecture of an IoT system based on the MQTT protocol?**

Broker (Server) - Clients with Publish and Subscribe system. In a Pub/Sub system, a device can publish a message on a topic or it can be subscribed to a particular topic to receive messages.

- **What is the IP protocol under MQTT? What does it mean in terms of bandwidth usage, type of communication, etc ?**

Usually TCP/IP, MQTT is a simple messaging protocol, designed for devices with low-bandwidth, therefore it's good for the transport of telemetry data (sensor data) and for M2M protocol, IoT application... . MQTT can send commands to control outputs, read and publish data from sensor nodes... MQTT has a lot of overhead and extra packets.

- **What are the different versions of MQTT?**

There are several versions and several variants of MQTT

MQTT v3.1.0

MQTT v3.1.1

MQTT v5

We also have MQTT-SN (Sensor Network) or MQTTS (secured version of MQTT)

For each topic, MQTT allows us to manage the desired QoS (Quality of Service). There are 3 QoS.

- **What kind of security/authentication/encryption are used in MQTT?**

MQTT encryption in the MQTTS version

Client authentication possible in some implementations of Brokers

Access Control on Topics possible in some implementations of Brokers

- **Suppose you have devices that include one button, one light and luminosity sensor. You would like to create a smart system for your house with this behavior:**

- you would like to be able to switch on the light manually with the button

- the light is automatically switched on when the luminosity is under a certain value

**What different topics will be necessary to get this behavior and what will the connection be in terms of publishing or subscribing?**

We will need 2 topics. Let's say we have 2 topics:

- Topic LuminosityValue in home/bedroom/LuminosityValue

- Topic ButtonValue in home/bedroom/ButtonValue

The light will be a client that is subscribed to these 2 topics.

The button will be the client that publishes in topic home/bedroom/ButtonValue.

The light sensor will be the client that publishes in topic home/bedroom/LuminosityValue.

We can use Mosquitto as MQTT Broker.

## NodeMCU ESP8266 Board

Communication : Wifi b/g/n(can be used as Wifi access point), Bluetooth

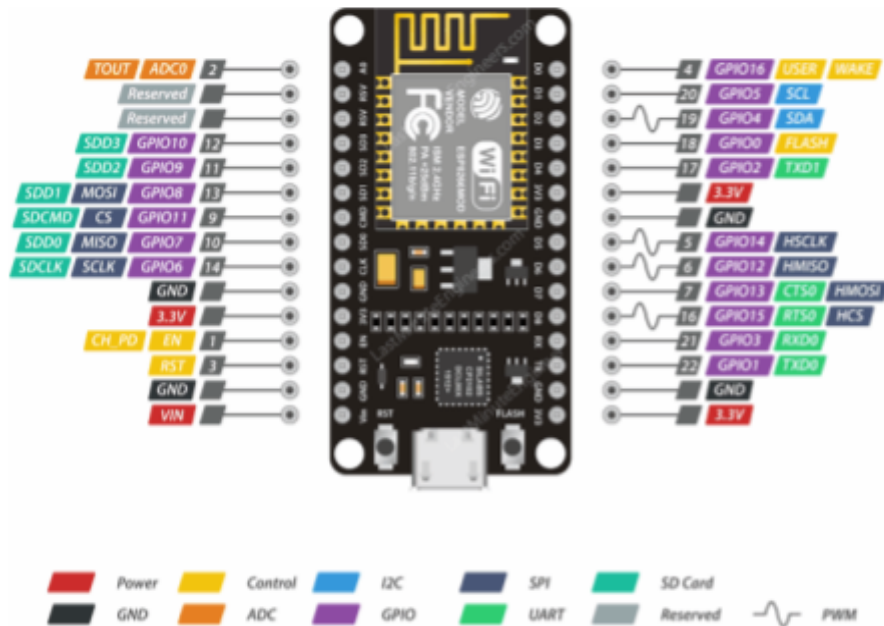
UART : 1

SPIs: 1

I2Cs:1

Protocoles : HTTP, MQTT

Langage de programmation : C/C++ on Arduino IDE



NodeMCU ESP8266 Pinout

## Detailed input/output characteristics:

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	<p><b>Micro-USB:</b> NodeMCU can be powered through the USB port</p> <p><b>3.3V:</b> Regulated 3.3V can be supplied to this pin to power the board</p> <p><b>GND:</b> Ground pins</p> <p><b>Vin:</b> External Power Supply</p>
Control Pins	EN, RST	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.

## Use case

The objective here is to use an MQTT broker to share data of measurements.

The system subscribes to this data and uses it as it wishes.

We use 2 Topics: LuminosityValue and ButtonValue

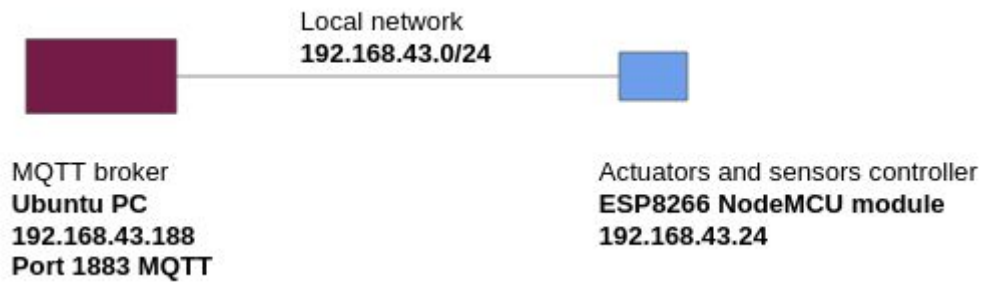
Light sensor publishes value on topic LuminosityValue

The system is subscribed to this topic, then if the value is as expected, it can switch the light off

Button sensor publishes value on topic ButtonValue

The system is also subscribed to this topic, and if the button is on , it can switch the light on.

Here is the hardware architecture



We verify with mosquitto client the published values of the sensor

```
Michael@Michael-X455LJ:~$ mosquitto_sub -h localhost -t LuminosityValue
877
878
880
881
883
886
```

```
Michael@Michael-X455LJ:~$ mosquitto_sub -h localhost -t LuminosityValue
877
878
880
881
883
886
```

Terminal shows Luminosity's value

## Video

[https://drive.google.com/file/d/19R2m5-UV2Wg7laBNV1e1y8Ur\\_ljuUqIL/view?usp=sharing](https://drive.google.com/file/d/19R2m5-UV2Wg7laBNV1e1y8Ur_ljuUqIL/view?usp=sharing)

## Code

[https://drive.google.com/file/d/1ze\\_-w6fjEs3xQLNe9U3xFD1zqPc411j/view?usp=sharing](https://drive.google.com/file/d/1ze_-w6fjEs3xQLNe9U3xFD1zqPc411j/view?usp=sharing)

## Analysis

We here used an MQTT broker to share information between two functionalities of one single system (the publishing sensors, and the one controlling the LED.).

Even if the broker is separated from the system (broker on PC and system on the ESP8266 module) and they are interconnected on a local network, this technology can be better used

when the functionalities are separated on different machines, and share data through a broker as done here.

## Ressources

<https://www.instructables.com/NodeMCU-ProjectButton-Control-LED/>  
[https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/Programming ESP8266 ESP-12E NodeMCU Using Arduino IDE - a Tutorial : 7 Steps \(with Pictures\) - Instructables](https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/Programming ESP8266 ESP-12E NodeMCU Using Arduino IDE - a Tutorial : 7 Steps (with Pictures) - Instructables)

---

## Part 2 : OM2M

### OM2M architecture

The objective here is to use the OM2M architecture to set up some Application Entities on a Common Service Entity (IN).

We used the given architecture of a Infrastructure Node CSE (IN CSE) with a Middle Node CSE(MN - CSE).

On our MN-CSE we have quite a few Application Entities.

3 Lamp AEs are set up using the MN-CSE's built in example which launches a GUI to control the state of the lamps and post their state on the corresponding containers in the AEs.

3 AEs (SmartMeterAE, Luminosity Sensor and Temperature Sensors) are set up using Postman the REST client. Each of these AEs have a DATA container and DESCRIPTION container ready to post some Content Instances.

We have a monitoring AE which we will explain later.

All these AEs have ACPs (Access Control Policies) that control their access by users.

Figure below shows this architecture:

<http://localhost:8282/~mn-cse/CAE788795898>

```
– mn-name
  – acp_admin
  – acpae-287584268
  – acpae-162980442
  – acpae-965423736
  – acpae-570824657
  – acpae-475309183
  – acpae-788795898
  – acpae-158610115
  – LAMP_0
  – LAMP_1
  – LAMP_ALL
  – SmartMeterAE
  – LuminositySensor
    – DESCRIPTION
    – DATA
  – TemperatureSensor
    – DESCRIPTION
    – DATA
  – MonitoringAE
  – in-name
```

We tried using the POSTMAN client to add some Content Instances in the DESCRIPTION containers using the obiX model. Put the formatting wouldn't work right and gave us some errors. The body used for the content and the result are shown below:



POST
http://localhost:8282/~mn-cse/mn-name/LuminositySensor/DESCRIPTION

Params
Authorization
Headers (11)
Body
Pre-request Script
Tests
Settings

none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
XML

```

1 <mzm:cin xmlns:mzm="http://www.onemzm.org/xml/protocols">
2   <cnf>application/xml</cnf>
3   <con>
4     <obj>
5       <str name="Type" val="Sensor"/>
6       <str name="Category" val="Light"/>
7       <str name="Unit" val="Lux"/>
8       <str name="Model" val="1142_0"/>
9       <str name="Location" val="Home"/>
10      <str name="Manufacturer" val="PHIDGETS"/>
11      <str name="ConsumptionMax" val="27mA"/>
12      <str name="VoltageMin" val="4.8VDC"/>
13      <str name="VoltageMax" val="5.3VDC"/>
14      <str name="OperatingTemperatureMin" val="0C"/>
15      <str name="OperatingTemperatureMax" val="70C"/>
16    </obj>
17  </con>
18 </mzm:cin>

```

Attribute	Value
ty	4
ri	/mn-cse/cin-132275029
pi	/mn-cse/cnt-22464092
ct	20201121T164611
lt	20201121T164611
st	0
cnf	application/xml
cs	1083
con	<obj>????????? <str?? name="Type"?? val="Sensor"?/?> ?????????<str?? name="Category" ??val="Light"?/?> ???????? <str?? name="Unit"?? val="Lux"?/?> ?????????<str?? name="Model"?? val="1142_0"?/?> ?????????<str?? ??name="Location"?? val="Home"?/?>????????? <str?? name="Manufacturer" ??val="PHIDGETS"?/?>????????? <str?? name="Consumption??Max" ??val="27??mA"?/?>????????? <str?? name="Voltage??Min" ??val="4.8??V??DC"?/?> ?????????<str ??name="Voltage??Max" ??val="5.3??V??DC"?/?>????????? <str?? name="Operating??Temperature??Min"??val="0??C"?/?> ?????????<str ??name="Operating??Temperature??Max"?? val="70??C"?/?> </obj>



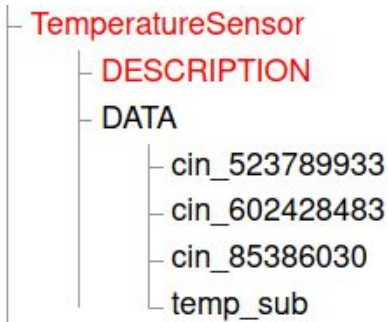
We then used the MonitorAE to which we connected the java application that prints Subscription Notifications.

Subscription Notifications are a way to get notified of new content instances in a container, instead of periodically trying to retrieve the content instances itself. The CSE here notifies the AE that it has new data in the container it is subscribed to.

Below, the concerned AE's information: \*

Attribute	Value
ty	2
ri	/mn-cse/CAE844130764
pi	/mn-cse
ct	20201121T171638
lt	20201121T171638
acpi	<div>AccessControlPolicyIDs</div> <div>/mn-cse/acp-594734824</div> <div>/mn-cse/acp-441294093</div>
et	20211121T171638
apn	MonitoringAE
api	MonitoringAE1
aei	CAE844130764
poa	<div>Point Of Access</div> <div>127.0.0.1:1400/monitor</div>
rr	true

And below the subscription of the AE to the DATA container of the TemperatureSensor AE:



Attribute	Value
ty	2
ri	/mn-cse/CAE844130764
pi	/mn-cse
ct	20201121T171638
lt	20201121T171638
acpi	<div>AccessControlPolicyIDs</div> <div>/mn-cse/acp-594734824</div> <div>/mn-cse/acp-441294093</div>
et	20211121T171638
apn	MonitoringAE
api	MonitoringAE1
aei	CAE844130764
poa	<div>Point Of Access</div> <div>127.0.0.1:1400/monitor</div>
rr	true

When we post data on the TemperatureSensor's DATA container, the java application is indeed notified:

```

Received notification:
<?xml version="1.0" encoding="UTF-8"?>
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
  <vrq>true</vrq>
  <sud>false</sud>
</m2m:sgn>

Received notification:
<?xml version="1.0" encoding="UTF-8"?>
<m2m:sgn xmlns:m2m="http://www.onem2m.org/xml/protocols">
  <nev>
    <rep rn="cin_840271103">
      <ty>4</ty>
      <ri>/mn-cse/cin-840271103</ri>
      <pi>/mn-cse/cnt-720542098</pi>
      <ct>20201121T172222</ct>
      <lt>20201121T172222</lt>
      <st>0</st>
      <cnf>xml:0</cnf>
      <cs>9</cs>
      <con>40
    </con>
    </rep>
    <rss>1</rss>
  </nev>
  <sud>false</sud>
  <sur>/mn-cse/mn-name/TemperatureSensor/DATA/temp_sub</sur>

```

OM2M architecture binary :

[https://drive.google.com/file/d/1lIFvKztBdlB0hmASUb\\_qGI8btIVBVbY2/view?usp=sharing](https://drive.google.com/file/d/1lIFvKztBdlB0hmASUb_qGI8btIVBVbY2/view?usp=sharing)

## Analysis

The OM2M architecture allows us to really structure our resources and to have richer information. We can also lighten the load on a machine organizing these resources and share it on distributed machines as done here with the IN-CSE and the MN-CSE.

However richer data means more data to transfer which might turn out more expensive.

---

## Part 3 : Node-Red

### Node-Red

The objective is with NodeRed is to create workflows to merge the MQTT architecture with the OM2M architecture, and to create higher level applications using data from all these sources.

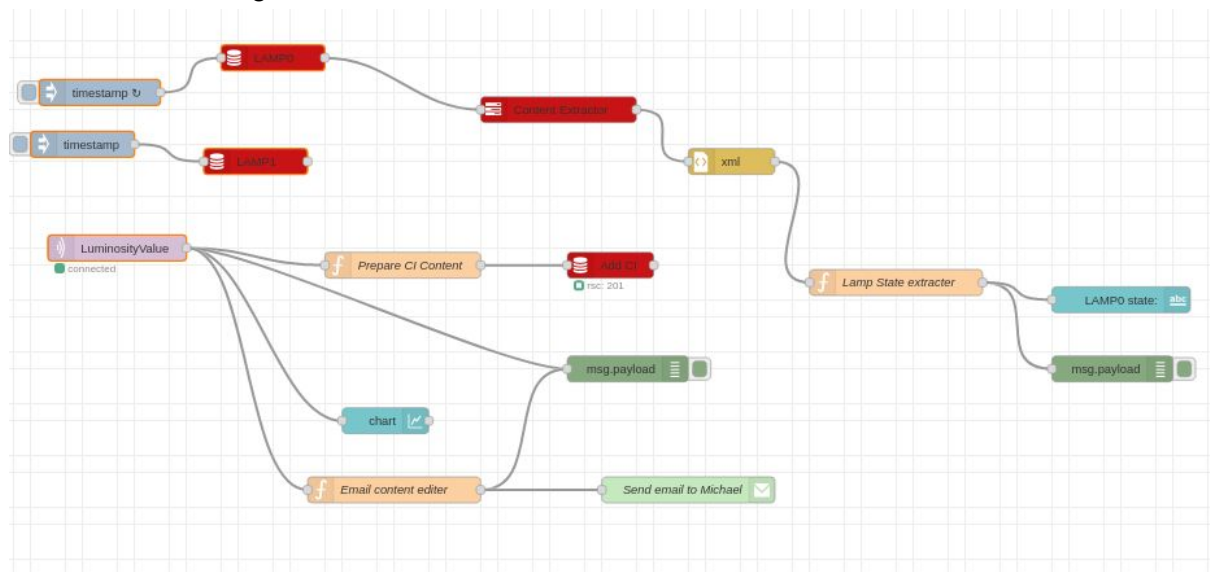
We implement the following use case: a Luminosity Sensor (simulated with the mosquitto ubuntu client) posts its values on a MQTT Broker (on the topic LuminosityValue).

On our OM2M architecture, we have an Application Entity called LAMP0, on which we can post the state of a Lamp (simulated using the inbuilt example in the MN-CSE).

The idea here is to switch the state of the lamp according to the value of the LuminositySensor (to control for example outside lighting or lighting for plants etc...).

We also use the Email Notification feature of NodeRed

Here is our resulting Workflow :



To do so we progress in a few parts.

First, to be able to post data of the Lamp State and to also be able to show its state, we extract the latest Content Instance in the LAMP0 DATA container.

This part of the Workflow is the top part using the Named Sensor Data, Content extractor, XML to Object, and Function Nodes.

To extract only the state of the Lamp, we put the following code in the Function Node (Lamp State Extracter):



Now knowing the structure of the content instance, we use a ContentInstance and a Function Node to be able to post lamp state Content Instances ourselves from the Workflow :



We put the following code in our Function Node Prepare CI Content:



As seen on top, the LuminosityValue node is the MQTT Subscriber Node which is subscribed to the LuminosityValue topic of our MQTT Broker

We verify that this chain works, we can see on the picture below, Content Instances being posted according to the data posted on the MQTT Broker:

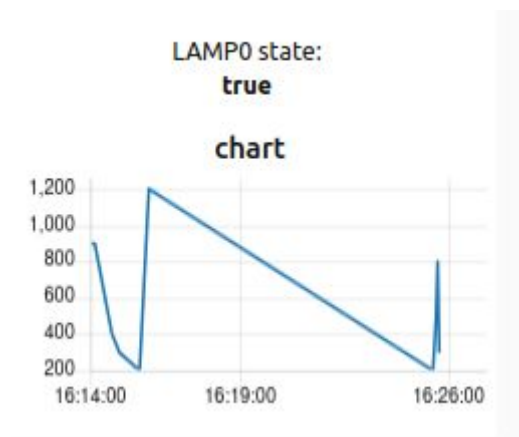
```

- LAMP_0
- DESCRIPTOR
- DATA
  - cin_998301876
  - cin_287484762
  - cin_718504853
  - cin_81306314
  - cin_387991130
  - cin_473671869
  - cin_74858261
  - cin_301991947
  - cin_61088282
  - cin_833214677

```

st	0										
cnf	text/xml										
cs	193										
con	<table border="1"> <thead> <tr> <th>Attribute</th><th>Value</th></tr> </thead> <tbody> <tr> <td>type</td><td>LAMP</td></tr> <tr> <td>location</td><td>Home</td></tr> <tr> <td>lampId</td><td>LAMP_0</td></tr> <tr> <td>state</td><td>false</td></tr> </tbody> </table>	Attribute	Value	type	LAMP	location	Home	lampId	LAMP_0	state	false
Attribute	Value										
type	LAMP										
location	Home										
lampId	LAMP_0										
state	false										

To visualize all this information we set up a ui using the Chart end Text Nodes.  
All works:



Finally we set up Email Notifications when the LuminosityValue reaches critical limits using the Function and Email Nodes (to be downloaded) .

Here below the content of the Email Content Editor Function,, the Email Node and the Email Notifications received :

Name: Email content editor

Setup
Function
Close

```

1 var Max={payload:"Light level has exceeded max critical limit of 1000, Value : "+msg.payload};
2 var Min={payload:"Light level has exceeded min critical limit of 300, Value : "+msg.payload};
3
4 if (msg.payload<300) {
5   return Min;
6 }
7 else if(msg.payload>1000) {
8   return Max;
9 }
10
11


```

Edit email node
Delete Cancel Done

Properties


To: michael.mesfin.97@gmail.com
Server: smtp.gmail.com
Port: 465 Use secure connection.
Userid: noderedissinsa2@gmail.com
Password: \*\*\*\*\*
Use TLS?
Name: Send email to Michael

---

**noderedissinsa2@gmail.com**16:16 (il y a 12 minutes) ☆



Light level has exceeded max critical limit of 1000, Value : 1200

---

**noderedissinsa2@gmail.com**16:25 (il y a 2 minutes) ☆ ↩ ⋮

À moi ▾  
⋮

Light level has exceeded min critical limit of 300, Value : 200



## Workflow

<https://drive.google.com/file/d/1GqvT7h-N1Swq77fW-L04ZMJ3UueziYPY/view?usp=sharing>

## Setup to use Workflow

- Need an MQTT broker, topic and address to be updated on the Node
- Need a CSE, address to be updated on the Node
- Need to download the OM2M, UI and Email Nodes.
- A Gmail Account with the security for less secure apps deactivated to send the email

## Analysis

With node-red we can without writing a single line of code, develop high level applications.,merge data from different sources ,monitor, notify and visualize information. We can also share these UIs by launching the workflow on an accessible machine and accessing it through a web browser.

Other tools such as Nifi offer an advantage of being able to handle more loads of data, offering queues and information on the processes in the workflow as well as the application itself.

## Conclusion

MQTT offers the simplicity of its use, and freedom to choose what date to share in what form. OM2M on the other hand is much more complex to get started with but by default offers very rich information.

We would recommend to use MQTT as done in our use case in Part 3, to collect simple data from sources with simple data (with maybe a timestamp on the data), that only has temporary information and doesn't need to be retained, and using OM2M to to organize resources for which data needs to be conserved and kept for further monitoring.

We would recommend NodeRed or Nifi for users who would like to build high level applications without writing code.