INTERGICIEL

Abderrazzak Mahraye

INSA TOULOUSE ISS B1

I MQTT LAB

I.1 Questions answers

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

MQTT protocol is based on a publish/subscribe architecture. In fact, a broker is set on a computer, and this broker enables components to publish or subscribe to a topic. Thus, a sensor can publish its values on a topic, and a component (as a monitor) can subscribe to this same topic to receive the values of the sensor.

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

MQTT is under the protocol TCP. TCP is more expensive in terms of bandwidth than UDP but the choice of TCP is that MQTT can be deployed in a restricted environment and it might be some packet loss.The use of TCP can avoid this loss and provide a better quality of service.

• What are the different versions of MQTT?

MQTT versions are:
MQTT v3.1.0
MQTT v3.1.1
MQTT v5 (for limited use)
MQTT-SN (under udp)

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

MQTT uses SSL/ TLS

• Suppose you have devices that include one button, one light and luminosity sensor. You would like to create a smart system for you 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?

Necessary topics: topic for the button, and topic for the sensor. The button will publish its value on its topic “buttonTopic” and the sensor will publish its values on “sensorTopic”. So the light will subscribe to these topics and change its state depending on the values received on each topic.

I.2 Practical Part

We used mqtt arduino libraries to compute a led depending on a button state, and a value of a sensor.

This is our code:

```
/*
 ********************************************************************************
 *
 * Purpose: Example of using the Arduino MqttClient with Esp8266WiFiClient.
 * Project URL: https://github.com/monstrenyatko/ArduinoMqtt
 *
 ********************************************************************************
 * Copyright Oleg Kovalenko 2017.
 *
 * Distributed under the MIT License.
 * (See accompanying file LICENSE or copy at http://opensource.org/licenses/MIT)
 ********************************************************************************
 */

#include <Arduino.h>
#include <ESP8266WiFi.h>

// Enable MqttClient logs
#define MQTT_LOG_ENABLED 1
// Include library
#include <MqttClient.h>

// capteurs
#define pinButton 5
const int pinLightSensor = A0;
#define pinLED 16

#define seuil 300

int prevButtonState = 1;
int buttonState = 0;
int luminosite;


#define LOG_PRINTFLN(fmt, ...)          logfln(fmt, ##__VA_ARGS__)
#define LOG_SIZE_MAX 128
void logfln(const char *fmt, ...) {
          char buf[LOG_SIZE_MAX];
          va_list ap;
          va_start(ap, fmt);
          vsnprintf(buf, LOG_SIZE_MAX, fmt, ap);
          va_end(ap);
          Serial.println(buf);
}

#define HW_UART_SPEED 115200L
#define MQTT_ID     "TEST-ID"

const char* MQTT_TOPIC_SUB = "test/" MQTT_ID "/sub";


const char* MQTT_TOPIC_PUB_LUM = "test/" MQTT_ID "/lum";
const char* MQTT_TOPIC_PUB_BUT = "test/" MQTT_ID "/but";

static MqttClient *mqtt = NULL;
static WiFiClient network;

// ============== Object to supply system functions ============================
class System: public MqttClient::System {
public:

          unsigned long millis() const {
                    return ::millis();
          }

          void yield(void) {
                    ::yield();
          }
};

// ============== Setup all objects ==========================================
void setup() {
          // Setup hardware serial for logging
          Serial.begin(HW_UART_SPEED);
          while (!Serial);
```

```
          // Setup WiFi network
          WiFi.mode(WIFI_STA);
          WiFi.hostname("ESP_" MQTT_ID);
          WiFi.begin("Cisco38658"); //Wifi Configuration
          LOG_PRINTFLN("\n");
          LOG_PRINTFLN("Connecting to WiFi");
          while (WiFi.status() != WL_CONNECTED) {
                    delay(500);
                    LOG_PRINTFLN(".");
          }
          LOG_PRINTFLN("Connected to WiFi");
          LOG_PRINTFLN("IP: %s", WiFi.localIP().toString().c_str());

          // Setup MqttClient
          MqttClient::System *mqttSystem = new System;
          MqttClient::Logger *mqttLogger = new MqttClient::LoggerImpl<HardwareSerial>(Serial);
          MqttClient::Network * mqttNetwork = new MqttClient::NetworkClientImpl<WiFiClient>(network, *mqttSystem);
          //// Make 128 bytes send buffer
          MqttClient::Buffer *mqttSendBuffer = new MqttClient::ArrayBuffer<128>();
          //// Make 128 bytes receive buffer
          MqttClient::Buffer *mqttRecvBuffer = new MqttClient::ArrayBuffer<128>();
          //// Allow up to 2 subscriptions simultaneously
          MqttClient::MessageHandlers *mqttMessageHandlers = new MqttClient::MessageHandlersImpl<2>();
          //// Configure client options
          MqttClient::Options mqttOptions;
          ////// Set command timeout to 10 seconds
          mqttOptions.commandTimeoutMs = 10000;
          //// Make client object
          mqtt = new MqttClient(
                    mqttOptions, *mqttLogger, *mqttSystem, *mqttNetwork, *mqttSendBuffer,
                    *mqttRecvBuffer, *mqttMessageHandlers
          );

  pinMode(pinButton, INPUT);
  pinMode(pinLightSensor, INPUT);
  pinMode(pinLED, OUTPUT);
}


// ============== Subscription callback ===================================
void processMessage(MqttClient::MessageData& md) {
  const MqttClient::Message& msg = md.message;
  char payload[msg.payloadLen + 1];
  memcpy(payload, msg.payload, msg.payloadLen);
  payload[msg.payloadLen] = '\0';
  LOG_PRINTFLN(
    "Message arrived: qos %d, retained %d, dup %d, packetid %d, payload:[%s]",
    msg.qos, msg.retained, msg.dup, msg.id, payload
  );
}


// ============== Main loop ================================================
void loop() {
          // Check connection status
          if (!mqtt->isConnected()) {
                    // Close connection if exists
                    network.stop();
                    // Re-establish TCP connection with MQTT broker
                    LOG_PRINTFLN("Connecting");
                    network.connect("192.168.1.147", 1883);
                    if (!network.connected()) {
                              LOG_PRINTFLN("Can't establish the TCP connection");
                              delay(5000);
                              ESP.reset();
                    }
                    // Start new MQTT connection
                    MqttClient::ConnectResult connectResult;
                    // Connect
                    {
                              MQTTPacket_connectData options = MQTTPacket_connectData_initializer;
                              options.MQTTVersion = 4;
                              options.clientID.cstring = (char*)MQTT_ID;
                              options.cleansession = true;
                              options.keepAliveInterval = 15; // 15 seconds
                              MqttClient::Error::type rc = mqtt->connect(options, connectResult);
                              if (rc != MqttClient::Error::SUCCESS) {
                                        LOG_PRINTFLN("Connection error: %i", rc);
                                        return;
                              }
                    }
                    {
                              // Add subscribe here if required
     MqttClient::Error::type rc = mqtt->subscribe(
       MQTT_TOPIC_SUB, MqttClient::QOS0, processMessage
      );
```

```
      if (rc != MqttClient::Error::SUCCESS) {
        LOG_PRINTFLN("Subscribe error: %i", rc);
        LOG_PRINTFLN("Drop connection");
        mqtt->disconnect();
        return;
                    }
            }
          }else {
                    {
                              // Add publish here if required
     //button
     const char* buf;
     if(digitalRead(pinButton)){
      buf = "bouton ON";
     }
     else{
      buf = "bouton OFF";
     }
     MqttClient::Message message;
     message.qos = MqttClient::QOS0;
     message.retained = false;
     message.dup = false;
     message.payload = (void*) buf;
     message.payloadLen = strlen(buf);
     mqtt->publish(MQTT_TOPIC_PUB_BUT, message);


    //luminosite
    luminosite = analogRead(pinLightSensor);
    if(luminosite < seuil){
      buf = "luminosite faible";
    }else{
      buf = "luminosite haute";
    }

    //MqttClient::Message message;
     //message.qos = MqttClient::QOS0;
     //message.retained = false;
     //message.dup = false;
     message.payload = (void*) buf;
     message.payloadLen = strlen(buf);
     mqtt->publish(MQTT_TOPIC_PUB_LUM, message);


                    }
                    // Idle for 30 seconds
                    mqtt->yield(1000L);
          }
}
```

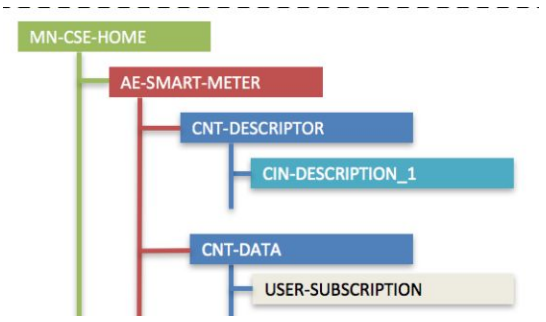
II ONEM2M RESTFUL ARCHITECTURE

II.1 Deployed infrastructure and interaction between component

The OM2M is composed of four components based on a tree (CSE, AE, CNT, and CIN) . The Common Service Entity is the root ressource. It has child ressources and thus allows to stock the new ressources, and information. The Application Entity is one child of CSE. It represents the link between an application and a CSE. Then, CNT are the containers that allow the structure of the resource data tree. Finally, the content instance (CIN) are the resources that enable the storage of values.

Thus, in this lab we created an infrastructure which contains 3 AEs representing each sensor: SmartMeter, LuminositySensor and temperatureSensor. In addition, these sensors will have two containers (DESCRIPTOR and DATA). Then, the DESCRIPTOR container is composed of a description CIN, and the DATA contains the measurement values of the sensor. Finally, the main aim is to be able to monitor the data measured by a sensor.

For example, the smartMeter will have the following tree:



II.2 Main Requests created

To do this, we used the rest API to create these different components by using the POSTMAN tool.

As a sample, here is the creation of an AE (SmartMeter sensor):

URL	http://127.0.0.1:8080/~/in-cse/
Method	POST
HEADER	X-M2M-Origin: admin:admin Content-type: application/xml;ty=2
BODY	<m2m:ae xmlns:m2m = "http://www.onem2m.org/xml/protocols" rn = "SmartSensor"> <api>app-sensor</api> <lbl>Type/sensor</lbl> <rr>false</rr> </m2m:ae>

Then, here is the creation of an the containers DESCRIPTOR and DATA:

URL	http://127.0.0.1:8080/~/in-cse/in-name/Sensor_Name
Method	POST
HEADER	X-M2M-Origin: admin:admin Content-type: application/xml;ty=3
BODY	<m2m:cnt xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="DATA or DESCRIPTOR"> </m2m:cnt>

And finally, the creation of content instance sample (for the DATA container) :

URL	http://127.0.0.1:8080/~/in-cse/in-name/Sensor_Name/DATAorDESCRIPTOR
Method	POST
HEADER	X-M2M-Origin: admin:admin Content-type: application/xml;ty=4
BODY	<m2m:cin xmlns:m2m="http://www.onem2m.org/xml/protocols"> <cnf>message</cnf> <con> <obj> <str name="Category" val="Light"/> <str name="Data" val="300" /> <str name="Unit" val="Lux" /> <str name="Location" val="Home" /> </obj> </con> </m2m:cin>

The same request can be done for the DESCRIPTOR with different characteristics in the part <obj>.

We can now subscribe to a sensor to receive its data on the monitor:

URL	http://127.0.0.1:8080/~/in-cse/in-name/MY_SENSOR/DATA
Method	POST
HEADER	X-M2M-Origin: admin:admin Content-type: application/xml;ty=23
BODY	<m2m:sub xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="SUB_SmartSensor"> <nu>http://localhost:1400/monitor</nu> <nct>2</nct> </m2m:sub>

Once a data is received by a sensor, it will be notified on the monitor.

Let's now introduce the ACP:
ACP enables us to specify rights. It specifies what an user is enabled to do.
For example: Guest:Guest has the rights to Retrieve and Create because his acop equals to 3 (1-> create + 2-> retrieve), and admin:admin has all the rights.

URL	http://127.0.0.1:8080/~/in-cse/
Method	POST
HEADER	X-M2M-Origin: admin:admin Content-type: application/xml;ty=1
BODY	<m2m:acp xmlns:m2m="http://www.onem2m.org/xml/protocols" rn="ACP_SENSOR"> <pv> <acr>

	<acor>guest:guest</acor> <acop>3</acop> </acr> <acr> <acor>admin:admin</acor> <acop>63</acop> </acr> </pv> <pvs> <acr> <acor>admin:admin</acor> <acop>63</acop> </acr> </pvs> </m2m:acp>

Then each ressources has to be linked to an ACP: we attach the smartSensor to the two ACP via each ACPIs

URL	http://127.0.0.1:8080/~/in-cse/in-name/SmartSensor
Method	PUT
HEADER	X-M2M-Origin: admin:admin Content-type: application/json;ty=3

BODY	{
	"m2m:cnt": {
	"acpi": [
	"/in-cse/acp-585995077",
	"/in-cse/acp-534323853"
	]
	}
	}

III TECHNOLOGIES COMPARISONS

III.1 MQTT and oneM2M

The main difference between MQTT and OM2M is their architecture. MQTT is based on a client/server mecanisme while OM2M is based on a ressource mecanisme. Consequently, using OM2M we have to get the resource (as a sensor value) from the platform while using MQTT the resource can be published by the sensor on a topic and get this resource from this topic.

III.2 oneM2M and MangoH Yellow

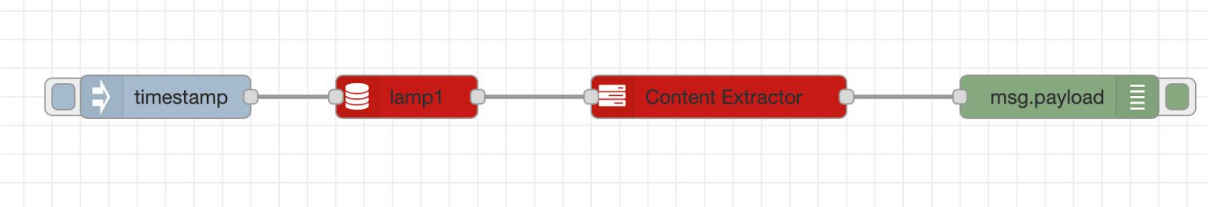
The main advantage of Sierra MangoH Yellow is that we have access to the component/sensor directly on the Sierra platform. We can compute and monitor them directly on the platform. Unlike OM2M we must set up all the components thanks to its restfull API and then compute and monitor them.

IV HIGH LEVEL APPLICATION NODE RED

In this part we used both the MQTT technology and OM2M in an node red application to compute a lamp function of the value of a light sensor.

IV.1 Get the Lamp state

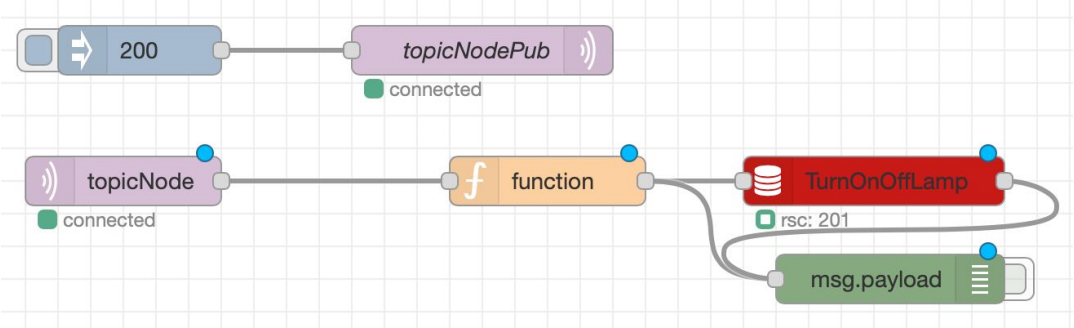
Firstly, we tried to get the lamp value:



Thus, this flow return the lamp value “false” : the lamp is turned off

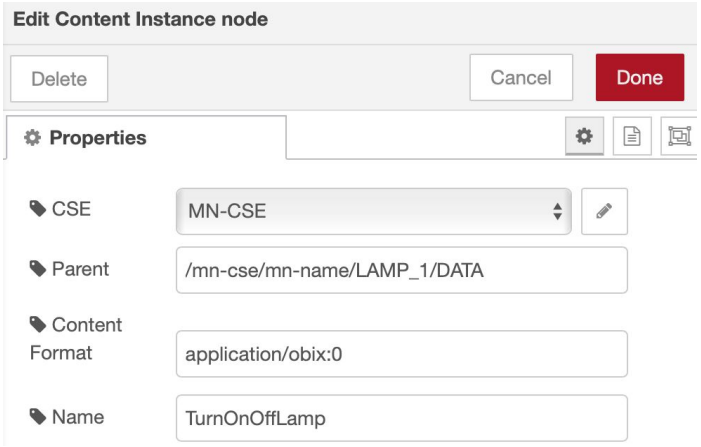
```
msg.payload : string[215]
"<?xml version="1.0"
encoding="UTF-8" standalone="yes"?
> <obj>     <str val="LAMP"
name="type"/>     <str val="Home"
name="location"/>     <str
val="LAMP_1" name="lampId"/>
<bool val="false" name="state"/>
</obj>"
```

IV.2 Get the sensor value and compute the lamp depending on this value



In this part, we used the MQTT technology to handle the sensor. In fact, the sensor and its values are published in the topic “topicNode”. Unfortunately, during the practical work we did not have the sensor. So, we replaced the MQTT publications by an injected values (for example 200 in the previous screenshot).
Then, we subscribed to the same topic to compute the values. We test these values to compute the lamp in the function block:

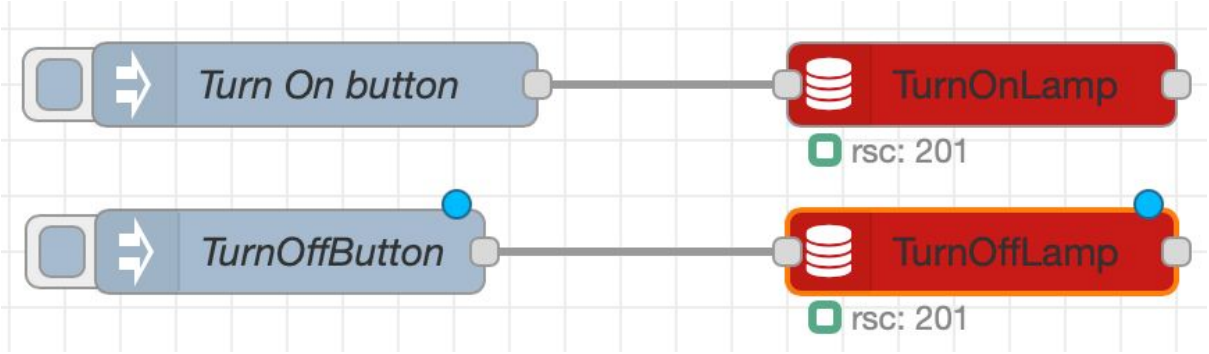
Then depending on the value, the returned obix object is different (turn on or off the lamp).
Finally, we create an new content instance from the obix object in order to change the state of the lamp:



As an example, for the sensor value 200 we returned an new content instance with a “false” value because 200 is lower than 300:

IV.3 Dashboard

We created buttons that turn on or off the lamp:



To do that, we created new content instances with a different obix object as input (value false or true: turn off/turn on) : they are the same objects as before.

IV.4 Benefits and drawbacks to build an application with Node-Red

The main benefits:
- Node-Red application allows us to use several and different technologies easily without incompatibilities (MQTT and OM2M in this practical work).
- Node-Red application does not require programming skills and it is user friendly.

The main drawback:
Hard to use for complicated use (a lot of “block”, the user can be easily lost)

IV.5 Node-Red flows exportation


```
[{"id":"8b297297.2ffa38","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"586606a7.6f60e","type":"Named Sensor
Data","z":"8b297297.2ffa38","name":"lamp1","cseConfig":"51c20198.d4617","aeConfig":"397a98bc.6ad94","cntName":"DATA","cin":"la","x":330,"y":140,"wires":[["7e4769a9.2cd8e8"]]},{"id"
:"1f7f30f7.94bb97","type":"inject","z":"8b297297.2ffa38","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":
"","payload":"","payloadType":"date","x":160,"y":140,"wires":[["586606a7.6f60e"]]},{"id":"7e4769a9.2cd8e8","type":"Content Extractor","z":"8b297297.2ffa38","name":"Content
Extractor","x":550,"y":140,"wires":[["62c9d1af.8c0c9"]]},{"id":"62c9d1af.8c0c9","type":"debug","z":"8b297297.2ffa38","name":"","active":true,"tosidebar":true,"console":false,"tosta
tus":false,"complete":"payload","targetType":"msg","statusVal":"","statusType":"auto","x":790,"y":140,"wires":[]},{"id":"afc0679d.16a31","type":"mqtt
in","z":"8b297297.2ffa38","name":"","topic":"topicNode","qos":"2","datatype":"auto","broker":"ea436a7.1a60118","x":140,"y":360,"wires":[["32ff387e.b9efb"]]},{"id":"8140375b.88e958"
,"type":"mqtt
out","z":"8b297297.2ffa38","name":"topicNodePub","topic":"topicNode","qos":"2","retain":"","broker":"ea436a7.1a60118","x":360,"y":280,"wires":[]},{"id":"32ff387e.b9efb","type":"fun
ction","z":"8b297297.2ffa38","name":"","func":"var val = msg.payload\nif(val>300)\n{\n    msg.payload =\"<?xml version=\\\"1.0\\\" encoding=\\\"UTF-8\\\"
standalone=\\\"yes\\\"?><obj><str val=\\\"LAMP\\\" name=\\\"type\\\"/><str val=\\\"Home\\\" name=\\\"location\\\"/><str val=\\\"LAMP_1\\\" name=\\\"lampId\\\"/><bool
val=\\\"true\\\" name=\\\"state\\\"/></obj>\"\n    \n\n}\nelse\n{\n    msg.payload =\"<?xml version=\\\"1.0\\\" encoding=\\\"UTF-8\\\" standalone=\\\"yes\\\"?><obj><str
val=\\\"LAMP\\\" name=\\\"type\\\"/><str val=\\\"Home\\\" name=\\\"location\\\"/><str val=\\\"LAMP_1\\\" name=\\\"lampId\\\"/><bool val=\\\"false\\\"
name=\\\"state\\\"/></obj>\"\n}\n    \nreturn msg
;","outputs":1,"noerr":0,"initialize":"","finalize":"","x":400,"y":360,"wires":[["8eaaae2c.e156c","c61b4ad6.e64b88"]]},{"id":"8eaaae2c.e156c","type":"Content
Instance","z":"8b297297.2ffa38","cseConfig":"51c20198.d4617","parentUri":"/mn-cse/mn-name/LAMP_1/DATA","contentFormat":"application/obix:0","name":"TurnOnOffLamp","x":600,"y":360,"
wires":[["c61b4ad6.e64b88"]]},{"id":"4580d79.ada4e28","type":"inject","z":"8b297297.2ffa38","name":"","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","o
nce":false,"onceDelay":0.1,"topic":"","payload":"200","payloadType":"num","x":150,"y":280,"wires":[["8140375b.88e958"]]},{"id":"52ff8ae9.63a574","type":"inject","z":"8b297297.2ffa3
8","name":"Turn On button","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":false,"onceDelay":0.1,"topic":"","payload":"<?xml version=\"1.0\"
encoding=\"UTF-8\" standalone=\"yes\"?> <obj>     <str val=\"LAMP\" name=\"type\"/>     <str val=\"Home\" name=\"location\"/>     <str val=\"LAMP_1\" name=\"lampId\"/>     <bool
val=\"true\" name=\"state\"/> </obj>","payloadType":"str","x":150,"y":620,"wires":[["3186b856.7bf858"]]},{"id":"3186b856.7bf858","type":"Content
Instance","z":"8b297297.2ffa38","cseConfig":"51c20198.d4617","parentUri":"/mn-cse/mn-name/LAMP_1/DATA","contentFormat":"application/obix:0","name":"TurnOnLamp","x":390,"y":620,"wir
es":[[]]},{"id":"cab23206.660ff8","type":"inject","z":"8b297297.2ffa38","name":"TurnOffButton","props":[{"p":"payload"},{"p":"topic","vt":"str"}],"repeat":"","crontab":"","once":fa
lse,"onceDelay":0.1,"topic":"","payload":"<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?> <obj>     <str val=\"LAMP\" name=\"type\"/>     <str val=\"Home\"
name=\"location\"/>     <str val=\"LAMP_1\" name=\"lampId\"/>     <bool val=\"false\" name=\"state\"/>
</obj>","payloadType":"str","x":141,"y":680,"wires":[["e2f7f3c3.e3112"]]},{"id":"e2f7f3c3.e3112","type":"Content
Instance","z":"8b297297.2ffa38","cseConfig":"51c20198.d4617","parentUri":"/mn-cse/mn-name/LAMP_1/DATA","contentFormat":"application/obix:0","name":"TurnOffLamp","x":390,"y":680,"wi
res":[[]]},{"id":"c61b4ad6.e64b88","type":"debug","z":"8b297297.2ffa38","name":"","active":false,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType"
:"msg","statusVal":"","statusType":"auto","x":610,"y":420,"wires":[]},{"id":"51c20198.d4617","type":"CSE_CONFIG","cse":"MN-CSE","poa":"http://127.0.0.1:8282","cseId":"mn-cse","cseN
ame":"mn-name","adminOriginator":"3fe9ff83.72b428"},{"id":"397a98bc.6ad94","type":"AE_CONFIG","aeName":"LAMP_1"},{"id":"ea436a7.1a60118","type":"mqtt-broker","name":"","broker":"lo
calhost","port":"1883","clientid":"","usetls":false,"compatmode":false,"keepalive":"60","cleansession":true,"birthTopic":"","birthQos":"0","birthRetain":"false","birthPayload":"","
closeTopic":"","closeQos":"0","closeRetain":"false","closePayload":"","willTopic":"","willQos":"0","willRetain":"false","willPayload":""},{"id":"3fe9ff83.72b428","type":"ORIGINATOR
_CONFIG","originatorName":"admin","originator":"admin:admin"}]
```