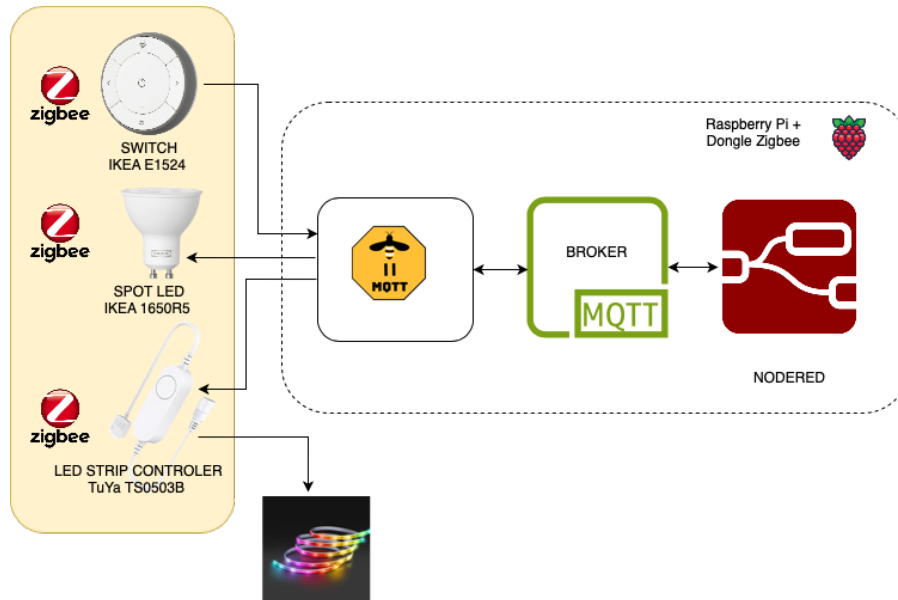


TP 2 ZIGBEE

COMMANDE D'ÉCLAIRAGE



1. MATERIEL

- 1 raspberry pi
- 1 dongle USB/zigbee
 - sonoff zigbee 3.0 (<https://amz.run/6LG0>)
 - Ou dongle CC2531 Zigbee Clé USB + **firmware pour OpenHAB ioBroker FHEM zigbee2mqtt** avec antenne SMA Boîtier Noir (<https://amz.run/6LFw>)
- Un switch IKEA E1524
- Un spot LED IKEA
- Un controleur LED Strip TuYa
- Un bandeau de LED

2. INSTALLATION SUR RASPBERRY PI

2.1. Installation de node-red : ouvrir une console sur le raspberry pi et saisir la commande

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

Cela installe nodejs, npm et node-red.

Si node-red a été correctement installé, nodejs et npm le sont aussi. On peut vérifier avec les commandes

```
# Verify that the correct nodejs and npm (automatically installed with nodejs)
# version has been installed
node --version # Should output v18.X or more
```

```
npm --version # Should output 10.X or more
```

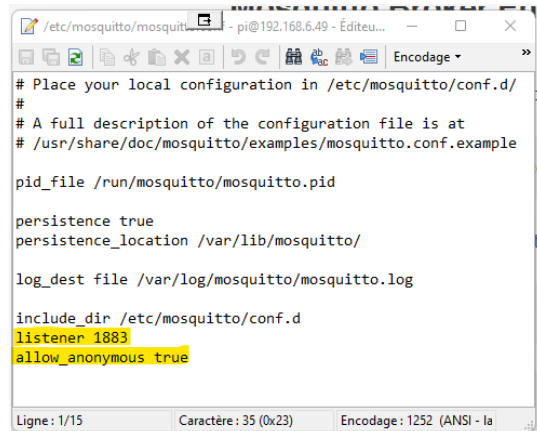
2.2. Installation du broker mosquitto :

```
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
sudo systemctl enable mosquitto.service
```

Éditer le fichier

/etc/mosquitto/mosquitto.conf et ajouter les 2 lignes suivantes à la fin :

```
listener 1883
allow_anonymous true
```



Redémarrer le service mosquitto :

```
sudo systemctl restart mosquitto
```

2.3. Installation de zigbee2mqtt

2.3.1 Détermination du port USB du dongle zigbee

Connecter le dongle ZigBee et déterminer le port série.

```
ls -l /dev/serial/by-id
```

exemple de réponse avec un dongle CC2531

```
total 0
lrwxrwxrwx. 1 root root 13 Oct 19 19:26 usb-
Texas_Instruments_TI_CC2531_USB_CDC_0X00124B0018ED3DDF-if00 -> ../../ttyACM0
```

dans ce cas le dongle sera localisé à **/dev/ttyACM0**

exemple de réponse avec un dongle sonoff ZigBee 3.0

```
total 0
lrwxrwxrwx. 1 root root 13 26 janv. 16:17 usb-
Silicon_Labs_Sonoff_Zigbee_3.0_USB_Dongle_Plus_0001-if00-port0 -> ../../ttyUSB0
```

Dans ce cas le dongle sera localisé à **/dev/ttyUSB0**

2.3.2 Installation de zigbee2mqtt

Création du dossier d'installation pour zigbee2mqtt et affectation de l'utilisateur pi comme propriétaire

```
sudo mkdir /opt/zigbee2mqtt
sudo chown -R ${USER}: /opt/zigbee2mqtt
```

Clonage du dépôt Zigbee2Mqtt

```
git clone --depth 1 https://github.com/Koenkk/zigbee2mqtt.git
/opt/zigbee2mqtt
```

Installation de dépendances

```
cd /opt/zigbee2mqtt
npm ci
```

À la fin cela indique combien de packages ont été installés.

Éditer le fichier de configuration (avec nano ou autre)

Il est localisé ici : **/opt/zigbee2mqtt/data/configuration.yaml**

```
homeassistant: false
permit_join: true
mqtt:
  base_topic: zigbee2mqtt
  server: mqtt://localhost
serial:
  # emplacement du dongle Zigbee
  port: /dev/ttyACM0 # ou /dev/ttyUSB0
advanced:
  network_key: GENERATE
frontend: true
```

Source TP3 :

<https://github.com/bouhenic/FormationIOT/blob/main/Tp2Zigbee/ConfigurationTp2.yaml>

Source TP4 :

<https://github.com/bouhenic/FormationIOT/blob/main/Tp2Zigbee/ConfigurationTp2Bis.yaml>

Lancement de zigbee2mqtt :

```
cd /opt/zigbee2mqtt
npm start
```

Si le démarrage se passe sans encombre on voit ce type de messages dans la console :

```
info 2023-01-27 16:27:24: Logging to console and directory:
'/opt/zigbee2mqtt/data/log/2023-01-27.16-27-24' filename: log.txt
info 2023-01-27 16:27:24: Starting Zigbee2MQTT version 1.29.2 (commit #1402139)
info 2023-01-27 16:27:24: Starting zigbee-herdsman (0.14.83-hotfix.0)
info 2023-01-27 16:27:30: zigbee-herdsman started (resumed)
info 2023-01-27 16:27:30: Coordinator firmware version:
'{"meta":{"maintrel":3,"majorrel":2,"minorrel":6,"product":0,"revision":20211115,"transportrev":2},"type":"zStack12"}'
```




Pour in démarrage comme un service avec systemctl voir ici :


https://www.zigbee2mqtt.io/guide/installation/01_linux.html#optional-running-as-a-daemon-with-systemctl

3 SERVEUR WEB DE CONFIGURATION (FRONTEND)

Si zigbee2mqtt est lancé, il est accessible à l'adresse du Raspberry sur le port 8080 :

http://<IP_RASPBERRY>:8080

Zigbee2MQTT Appareils							
Tableau de bord Schéma Paramètres Groupes MâJ OTA Touchlink Journaux Extensions							
Saisissez le critère de recherche							
#	Img	Nom simplifié	Adresse IEEE	Constructeur	Modèle	LQI	Alimentation
1		IKEA_GU10	0x90fd9ffffe14c33d (0x9BC2)	IKEA	LED1650R5	150	🔌
2		IKEA_SWITCH	0x000d6ffffe16ec06 (0xDB84)	IKEA	E1524/E1810	111	🔌
3		LED	0xa4c138de63548d15 (0xEF9A)	TuYa	TS0503B	189	🔌

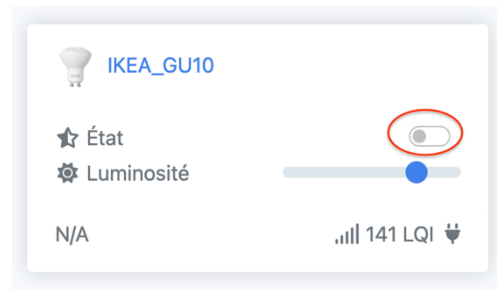
Pour découvrir un nouveau dispositif il suffit de cliquer sur **Activer l'appairage (tout)** et de mettre le dispositif à inscrire en mode découverte. Il apparait avec son adresse IEEE. On peut le renommer en cliquant sur .

Il est possible de piloter les dispositifs depuis cette page web en choisissant l'onglet **Tableau de bord**.

Chaque fois qu'une action est déclenchée sur un dispositif, les données sont publiées sur le broker.

Par exemple :

Le fait de cliquer sur l'interrupteur d'état pour le dispositif nommé **IKEA_GU10** génère le message suivant dans le fichier log.



```
Zigbee2MQTT:info 2024-02-25 11:41:09: MQTT publish: topic 'zigbee2mqtt/IKEA_GU10', payload '{
  "brightness":220,"linkquality":111,"state":"ON","update":{"installed_version":587810353,"latest_version":587810353,"state":"idle"}}'
```

Ce qui montre qu'un message a été publié sur le topic **zigbee2mqtt/IKEA_GU10**
Le message est un fichier Json qui une fois mis en forme donne

```
{
  "brightness": 220,
  "linkquality": 111,
  "state": "ON",
  "update": {
    "installed_version": 587810353,
    "latest_version": 587810353,
    "state": "idle"
  }
}
```

Il s'agit d'un objet qui indique la luminosité de l'éclairage , la qualité la connexion, l'état de la lampe et un objet indiquant des informations sur la mise à jour du firmware.
On peut suivre le journal des événements en cliquant sur l'onglet **Journaux**.

```
Info 2024-02-25 11:45:59 MQTT publish: topic 'zigbee2mqtt/IKEA_GU10', payload '{"brightness":220,"linkquality":117,"state":"OFF","update":{"installed_version":587810353,"latest_version":587810353,"state":"idle"}}'
Info 2024-02-25 11:46:03 MQTT publish: topic 'zigbee2mqtt/IKEA_GU10', payload '{"brightness":220,"linkquality":111,"state":"ON","update":{"installed_version":587810353,"latest_version":587810353,"state":"idle"}}'
```

4 PILOTAGE PAR MQTT

Fonctions exposées

Pour chaque dispositif, il faut se rendre sur la page de documentation qui lui est associée. Par exemple pour une ampoule spot GU10 de la marque IKEA, la documentation donne :

Expositions

Lumière Cette lumière prend en charge les fonctionnalités suivantes : état, luminosité.

état :

Pour contrôler l'état, publiez un message sur le sujet zigbee2mqtt/NOM_AMICAL/set avec le payload {"state": "ON"}, {"state": "OFF"} ou {"state": "TOGGLE"}.

Pour lire l'état, envoyez un message à zigbee2mqtt/NOM_AMICAL/get avec le payload {"state": ""}.

luminosité :

Pour contrôler la luminosité, publiez un message sur le sujet zigbee2mqtt/NOM_AMICAL/set avec le payload {"brightness": VALEUR} où VALEUR est un nombre entre 0 et 254.

Pour lire la luminosité, envoyez un message à zigbee2mqtt/NOM_AMICAL/get avec le payload {"brightness": ""}.

Allumé avec extinction programmée :

Lorsque vous réglez l'état sur ON, il pourrait être possible de spécifier une extinction automatique après un certain temps. Pour cela, ajoutez une propriété supplémentaire on_time au payload qui est le temps en secondes pendant lequel l'état doit rester allumé. De plus, une propriété off_wait_time peut être ajoutée au payload pour spécifier le temps de refroidissement en secondes pendant lequel la lumière ne répondra pas à d'autres commandes d'extinction programmée. La prise en charge dépend du firmware de la lumière. Certains appareils peuvent nécessiter à la fois on_time et off_wait_time pour fonctionner. Exemples : {"state": "ON", "on_time": 300}, {"state": "ON", "on_time": 300, "off_wait_time": 120}.

Transition :

Pour toutes les fonctionnalités mentionnées ci-dessus, il est possible de faire une transition de la valeur au fil du temps. Pour cela, ajoutez une propriété supplémentaire transition au payload qui est le temps de transition en secondes.

Exemples : {"brightness":156,"transition":3}, {"color_temp":241,"transition":1}.

Effet :

Déclenche un effet sur la lumière (par exemple, faire clignoter la lumière pendant quelques secondes). La valeur ne sera pas publiée dans l'état. Il n'est pas possible de lire (/get) cette valeur. Pour écrire (/set) une valeur, publiez un message sur le sujet zigbee2mqtt/NOM_AMICAL/set avec le payload {"effect": NOUVELLE_VALEUR}. Les valeurs possibles sont : blink, breathe, okay, channel_change, finish_effect, stop_effect.

Test de mqtt avec mosquitto :

Commande et configuration du spot GU10 :

- State (binary) :

L'état de la lampe peut être activée avec :

```
mosquitto_pub -h localhost -t zigbee2mqtt/IKEA_GU10/set -m  
'{"state":"ON"}'
```

Peut être lu après une commande subscribe

```
mosquitto_sub -h localhost -t zigbee2mqtt/IKEA_GU10
```

Peut aussi être lu après un /get en publish :

```
mosquitto_pub -h localhost -t zigbee2mqtt/IKEA_GU10/get -m  
'{"state":""}'
```

Peut être allumée avec éclairage défini :

```
mosquitto_pub -h localhost -t zigbee2mqtt/IKEA_GU10/set -m  
'{"state":"ON","brightness": 120}'
```

Peut être allumé avec un extinction programmée :

```
mosquitto_pub -h localhost -t zigbee2mqtt/IKEA_GU10/set -m  
'{"state":"ON","on_time": 300}'
```

- Transition :

On allume jusqu'à un éclairage défini sur une durée de transition.

```
mosquitto_pub -h localhost -t zigbee2mqtt/IKEA_GU10/set -m  
'{"brightness":220,"transition": 30}'
```

- Effet :

On peut choisir un effet :

Exemple avec l'effet breathe :

```
mosquitto_pub -h localhost -t zigbee2mqtt/IKEA_GU10/set -m  
'{"brightness":220,"transition": 30, "effect":"breathe"}'
```

Utilisation d'un serveur mqtt public sécurisé.

La société **HiveMQ** propose un broker dans le cloud gratuit limité à 100 sessions et 10GB par mois, ce qui est largement suffisant dans le domaine pédagogique.

Il faut créer un compte gratuit sur <https://console.hivemq.cloud> (identification par adresse email, compte gmail ou github)

Un fois connecté, il faut créer un **cluster** et choisir le type de serveur (AWS ou AZURE). Le choix n'entraîne aucune conséquence. Avec le compte gratuit, on peut créer 2 clusters.

Serverless
FREE

Running

URL

[redacted]s1.eu.hivemq.cloud

Port (TLS)

8883

Started

Wed, Feb 16, 2022, 11:52 AM

MANAGE CLUSTER

Quand on clique sur **MANAGE CLUSTER** on accède à la gestion du broker.

Dans l'onglet **OVERVIEW**, on a le rappel de l'URL et des ports par lesquels on peut accéder au broker.

Connection Settings

Cluster URL

EDIT

[redacted]s1.eu.hivemq.cloud

Port

8883

Websocket Port

8884

TLS URI

[redacted]s1.eu.hivemq.cloud:8883/mqtt

Websocket URI

[redacted]s1.eu.hivemq.cloud:8884/mqtt

Current Usage

MQTT Client Sessions

0 / 100

Data Traffic

0 B / 10 GB

Last update
19 seconds ago

Price per session
FREE

Last update
19 seconds ago

Price per GB
FREE

Dans l'onglet **ACCESS MANAGEMENT**, on peut gérer les informations d'identifications pour plusieurs utilisateurs.

Access Management

Credentials

Define one or more sets of credentials that allow MQTT clients to connect to your HiveMQ Cloud cluster. To learn more [check out our Security Fundamentals guide](#).

Username *

At least 5 characters

Password *

At least 8 characters, 1 digit, 1 uppercase character

Confirm Password *

Passwords must match

Permission *

Add permissions to limit access

> CREATE CREDENTIAL

Username	Permission type	Actions
bouhenic	<div><div></div> Publish and Subscribe</div>	<div>DELETE</div>

Il est à noter que les mots de passe **ne peuvent être modifiés ni revisualiser ultérieurement**, il faut donc prendre la précaution de les noter dans un lieu sûr.

Ajout du Broker à zigbee2mqtt :

On va modifier directement le fichier /opt/zigbee2mqtt/data/configuration.yaml :

```
sudo nano /opt/zigbee2mqtt/data/configuration.yaml
```

```
homeassistant: false
permit_join: true
mqtt:
  base_topic: zigbee2mqtt
  server: 's1.eu.hivemq.cloud:8883'
  user: 'sbouhenic'
  password: ' '
  ca: '/etc/ssl/certs/ISRG_Root_X1.pem'
  keepalive: 60
  reject_unauthorized: true
  version: 4
```

On relance zigbee2mqtt depuis /opt/zigbee2mqtt

```
npm start
```


Si tout se passe comme convenu, zigbee2mqtt se connecte sur le broker HiveMQ :

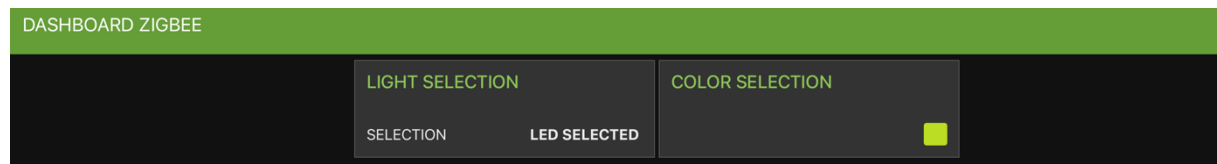
```
> zigbee2mqtt@1.33.2 start
> node index.js

Zigbee2MQTT:info 2024-02-25 11:37:36: Logging to console and directory: '/opt/zigbee2mqtt/data/log/2024-02-25.11-37-36' filename: log.txt
Zigbee2MQTT:info 2024-02-25 11:37:36: Starting Zigbee2MQTT version 1.33.2 (commit #311ea07)
Zigbee2MQTT:info 2024-02-25 11:37:36: Starting zigbee-herdsman (0.21.0)
Zigbee2MQTT:info 2024-02-25 11:37:39: zigbee-herdsman started (resumed)
Zigbee2MQTT:info 2024-02-25 11:37:39: Coordinator firmware version: '{"meta":{"maintrel":1,"majorrel":2,"minorrel":7,"product":1,"revision":20210708,"transportrev":2},"type":"zStack3x0"}'
Zigbee2MQTT:info 2024-02-25 11:37:40: Currently 3 devices are joined:
Zigbee2MQTT:info 2024-02-25 11:37:40: IKEA_GU10 (0x98fd9ffffe14c33d): LED1650R5 - IKEA TRADFRI LED bulb GU10 400 lumen, dimmable (Router)
Zigbee2MQTT:info 2024-02-25 11:37:40: IKEA_SWITCH (0x000d6ffffe16ec06): E1524/E1810 - IKEA TRADFRI remote control (EndDevice)
Zigbee2MQTT:info 2024-02-25 11:37:40: LED (0x4c138de63548d15): TS0503B - TuYa Zigbee RGB light (Router)
Zigbee2MQTT:warn 2024-02-25 11:37:40: 'permit_join' set to 'true' in configuration.yaml.
Zigbee2MQTT:warn 2024-02-25 11:37:40: Allowing new devices to join.
Zigbee2MQTT:warn 2024-02-25 11:37:40: Set 'permit_join' to 'false' once you joined all devices.
Zigbee2MQTT:info 2024-02-25 11:37:40: Zigbee: allowing new devices to join.
Zigbee2MQTT:info 2024-02-25 11:37:40: Connecting to MQTT server at mqtt://[REDACTED].s1.eu.hivemq.cloud:8883
Zigbee2MQTT:info 2024-02-25 11:37:41: Connected to MQTT server
Zigbee2MQTT:info 2024-02-25 11:37:41: MQTT publish: topic 'zigbee2mqtt/bridge/state', payload '{"state":"online"}'
Zigbee2MQTT:info 2024-02-25 11:37:42: Loaded MyExampleExtension1701438553314
Zigbee2MQTT:info 2024-02-25 11:37:42: MQTT publish: topic 'zigbee2mqtt/example/extension', payload 'hello from MyExampleExtension1701438553314'
Zigbee2MQTT:info 2024-02-25 11:37:42: MQTT publish: topic 'zigbee2mqtt/901/availability', payload '{"state":"online"}'
Zigbee2MQTT:info 2024-02-25 11:37:42: Started frontend on port 0.0.0.0:8080
```

Test avec les commandes mosquitto :

```
mosquitto_pub -h xxxxxxxxxxxxxxxxxxxxxxxxxxx.s1.eu.hivemq.cloud -p 8883 -u sbouhenic -P xxxxxxxxxx -t zigbee2mqtt/IKEA_GU10/set -m '{"state":"TOGGLE"}'
```

5 UTILISATION DE NODE-RED



L'objectif est de concevoir le dashboard ci-dessus :

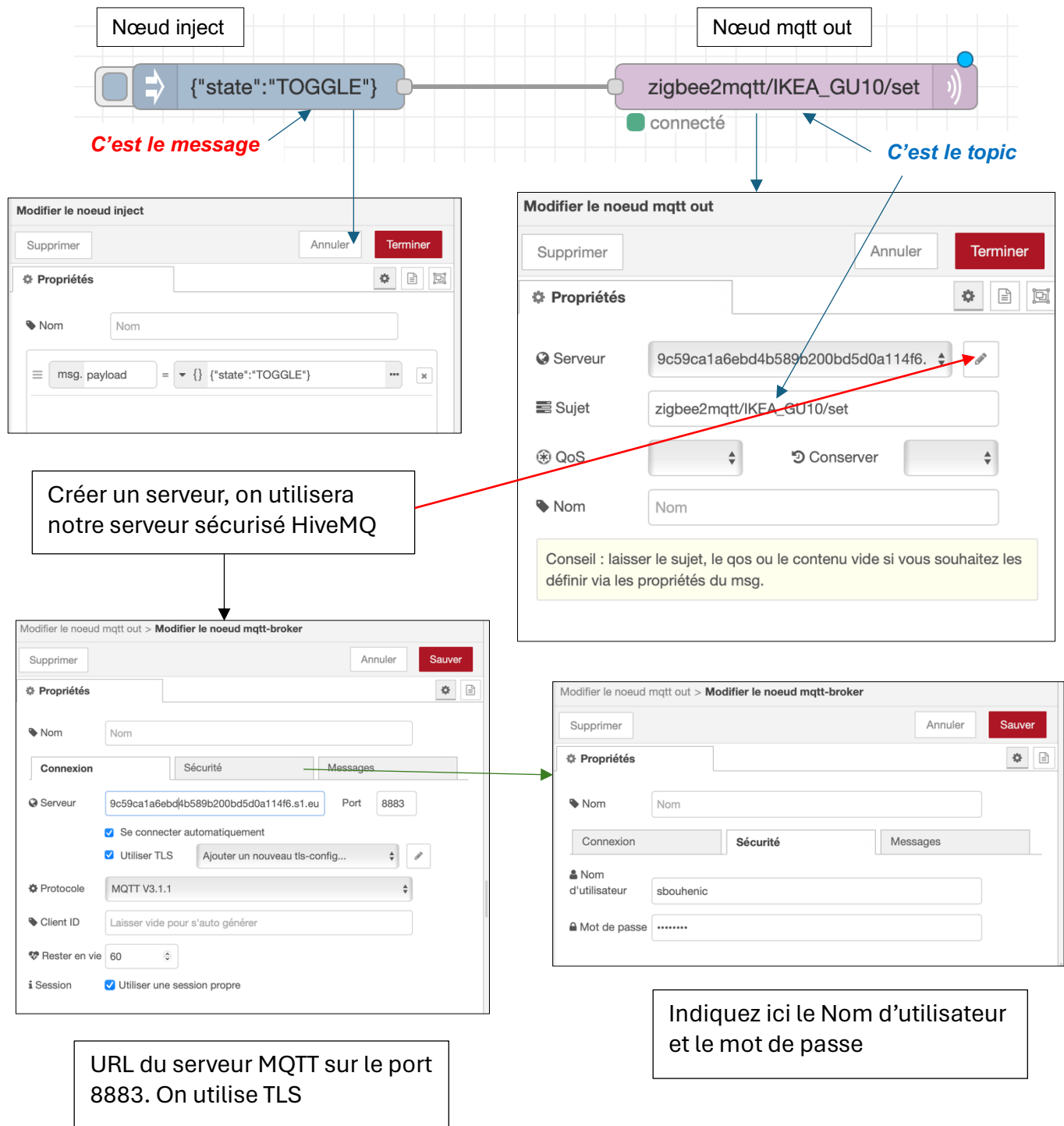
Côté switch :

- Le bouton central actionne les lampes (ON ou OFF).
- Les boutons haut et bas règlent la luminosité des lampes
- Les boutons droit et gauche permettent la sélection des lampes (spot ou LED), un text input affiche la sélection de lampe.
-

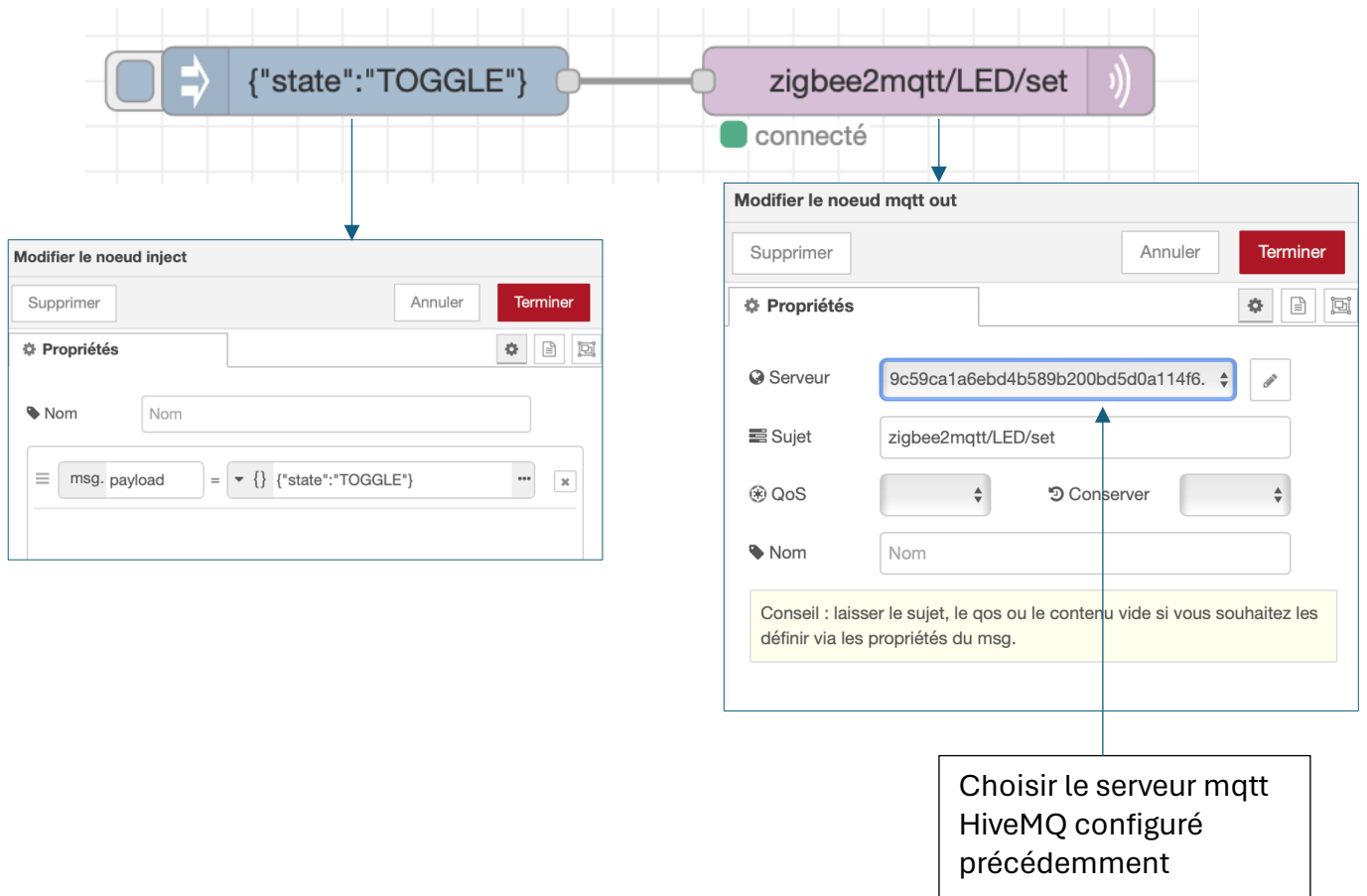
Côté LED :

- Un color picker permet la sélection de la couleur de la LED.

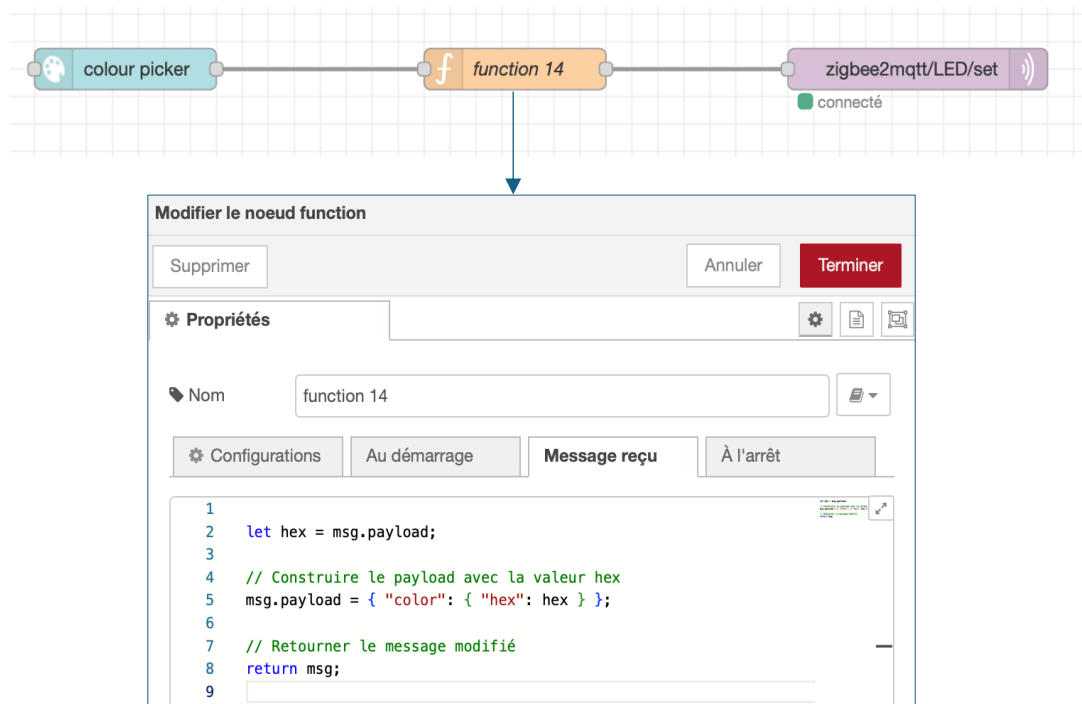
Test de la commande du spot :



Test du bandeau de LED :



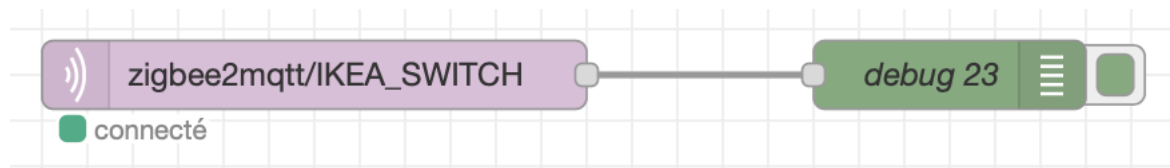
Commande de la couleur du bandeau de LED depuis le dashboard :



Source function14 :

<https://github.com/bouhenic/FormationIOT/blob/main/Tp2Zigbee/function14.js>

Réception des commandes issues du switch IKEA :



Modifier le noeud mqtt in

Supprimer Annuler Terminer

Propriétés

Seur 9c59ca1a6ebd4b589b200bd5d0a114f6.s1.eu.h

Action S'abonner à un seul sujet

Sujet zigbee2mqtt/IKEA_SWITCH

QoS 0

Sortie détection automatique (objet JSON analysé, chaîne ou

Nom Nom

```
03/03/2024 à 13:34:59 noeud: debug 23
zigbee2mqtt/IKEA_SWITCH : msg.payload : Object
  { action: "toggle", battery: 100,
    linkquality: 129, update: object }

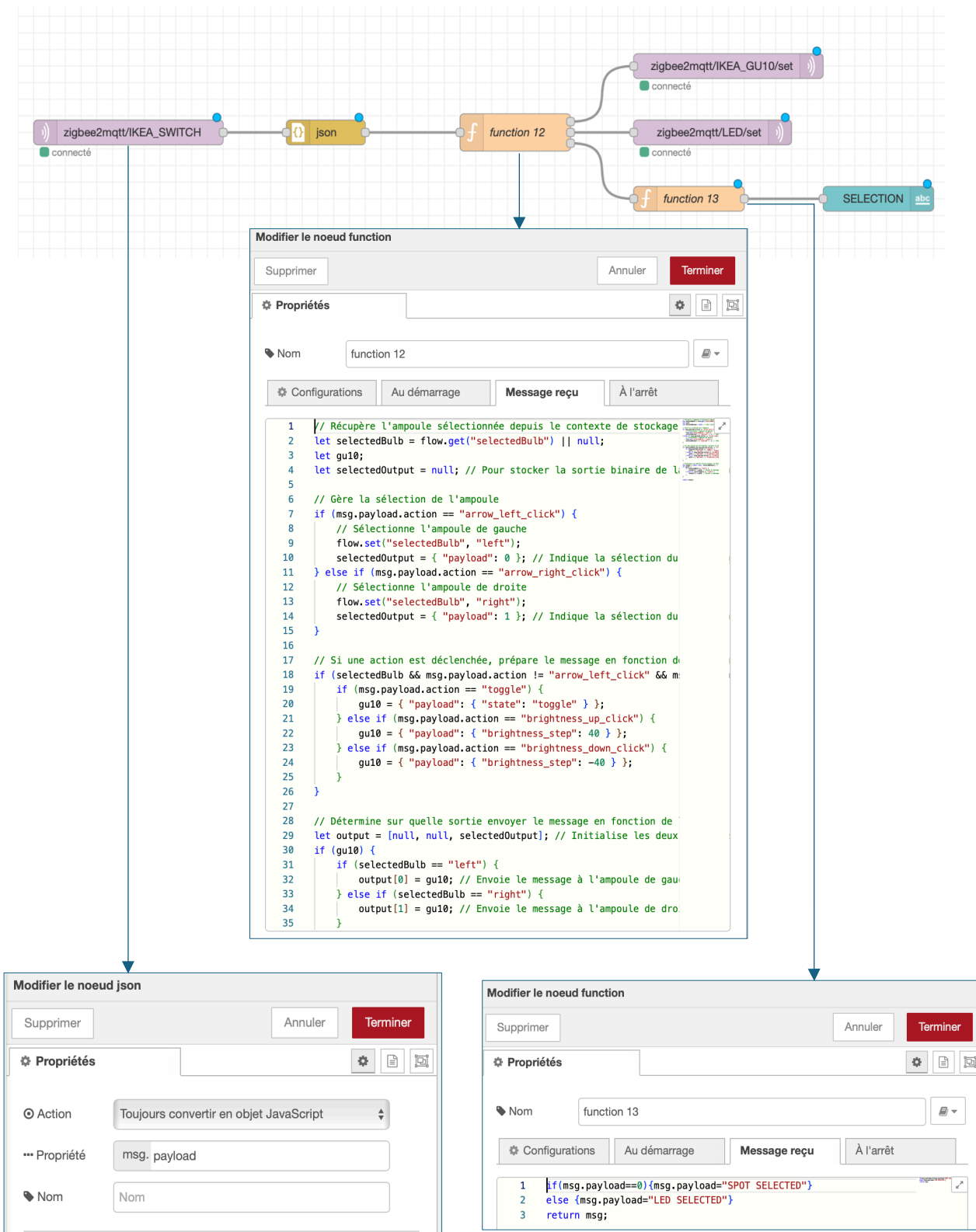
03/03/2024 à 13:35:03 noeud: debug 23
zigbee2mqtt/IKEA_SWITCH : msg.payload : Object
  { action: "arrow_right_click",
    battery: 100, linkquality: 126,
    update: object }

03/03/2024 à 13:35:05 noeud: debug 23
zigbee2mqtt/IKEA_SWITCH : msg.payload : Object
  { action: "arrow_left_click",
    battery: 100, linkquality: 126,
    update: object }

03/03/2024 à 13:35:07 noeud: debug 23
zigbee2mqtt/IKEA_SWITCH : msg.payload : Object
  { action: "brightness_up_click",
    battery: 100, linkquality: 129,
    update: object }

03/03/2024 à 13:35:09 noeud: debug 23
zigbee2mqtt/IKEA_SWITCH : msg.payload : Object
  { action: "brightness_down_click",
    battery: 100, linkquality: 129,
    update: object }
```

Commande des lampes (choix de la lampe, ON/OFF et luminosité) :



Source function12 :

<https://github.com/bouhenic/FormationIOT/blob/main/Tp2Zigbee/function12.js>

Source function13 :

<https://github.com/bouhenic/FormationIOT/blob/main/Tp2Zigbee/function13.js>

EXEMPLE DE FICHIER DE CONFIGURATION.YAML

```
homeassistant: false
permit_join: false
mqtt:
  base_topic: zigbee2mqtt
  server: mqtt://localhost
serial:
  port: /dev/ttyUSB0
frontend: true
advanced:
  log_output:
    - console
# mettre 675X ou X est le numéro du groupe
  pan_id: 6753
device_options:
  legacy: false
blocklist:
#01
  - '0xa4c138c35298bc87'
  - '0xa4c138a82a64f6f2'
#02
  - '0xa4c1381520886a8c'
  - '0xa4c138fcfcd1ab05'
#03
  - '0x90fd9ffffe14c33d'
  - '0x000d6ffffe16ec06'
  - '0xa4c138de63548d15'
#04
  - '0xd0cf5efffe18cb8c'
  - '0xa4c1389c73550ed1'
  - '0x90fd9ffffe193cfc'
devices: {}
```

CODE DES FONCTIONS :

Function 12 :

```
// Récupère l'ampoule sélectionnée depuis le contexte de stockage ou initialise
à null si non définie
let selectedBulb = flow.get("selectedBulb") || null;
let spot;

// Gère la sélection de l'ampoule
if (msg.payload.action == "arrow_left_click") {
    // Sélectionne l'ampoule de gauche
    flow.set("selectedBulb", "left");
    return [null, null]; // Ne transmet aucun message car c'est une action de
sélection
} else if (msg.payload.action == "arrow_right_click") {
    // Sélectionne l'ampoule de droite
    flow.set("selectedBulb", "right");
    return [null, null]; // Ne transmet aucun message car c'est une action de
sélection
}

// Si une action est déclenchée, prépare le message en fonction de l'action
if (selectedBulb) {
    if (msg.payload.action == "toggle") {
        spot = { "payload": { "state": "toggle" } };
    } else if (msg.payload.action == "brightness_up_click") {
        spot = { "payload": { "brightness_step": 40 } };
    } else if (msg.payload.action == "brightness_down_click") {
        spot = { "payload": { "brightness_step": -40 } };
    }
}

// Détermine sur quelle sortie envoyer le message en fonction de l'ampoule
sélectionnée
let output = [null, null]; // Initialise les deux sorties à null
if (spot) {
    if (selectedBulb == "left") {
        output[0] = spot; // Envoie le message à l'ampoule de gauche
    } else if (selectedBulb == "right") {
        output[1] = spot; // Envoie le message à l'ampoule de droite
    }
}

return output;
```

Function 14 :

```
let hex = msg.payload;

// Construire le payload avec la valeur hex
msg.payload = { "color": { "hex": hex } };

// Retourner le message modifié
return msg;
```

Function 13 :

```
if(msg.payload==0){msg.payload="SPOT SELECTED"};
else {msg.payload="LED SELECTED"};
return msg;
```

Source :

Formation IOT (François Riotte)