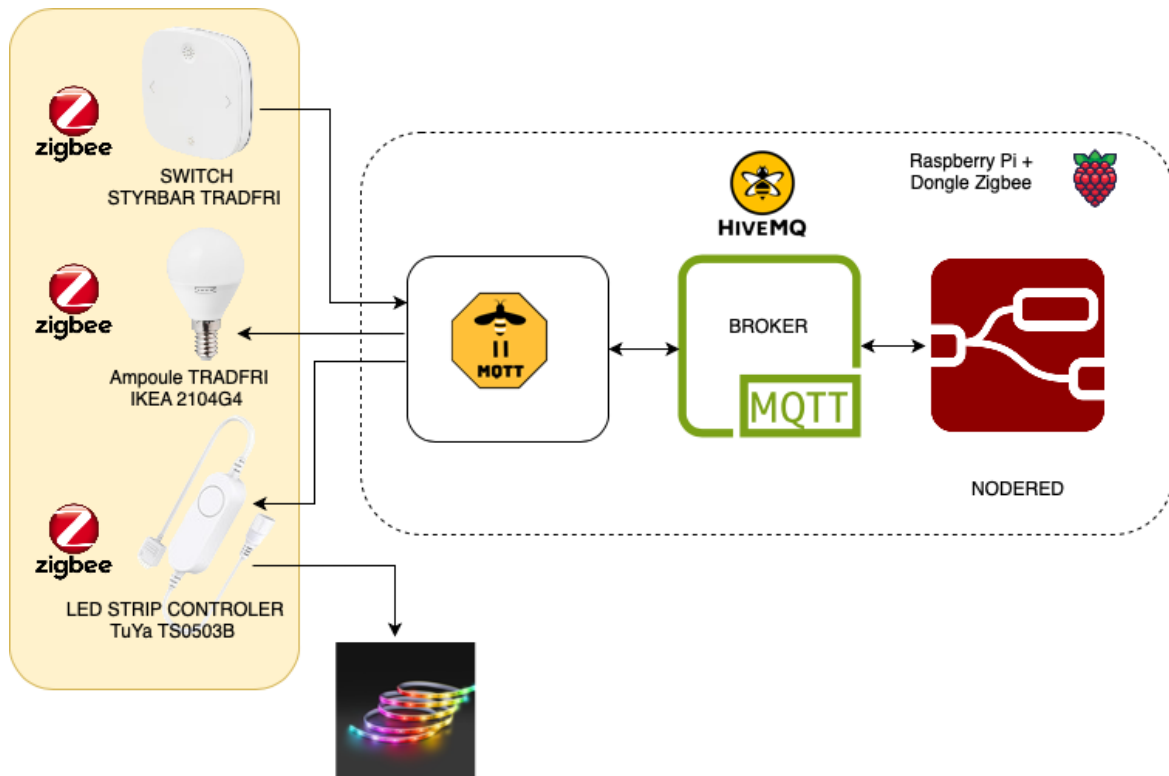


# TP 2 ZIGBEE

## COMMANDE D'ÉCLAIRAGE



### 1. CAHIER DES CHARGES

#### Objectif général :

Concevoir une interface interactive comprenant un **dashboard** et une **télécommande**, permettant de contrôler une ampoule et un LED Strip.

#### Spécifications fonctionnelles :

##### Côté télécommande :

1. **Commande ON/OFF :**
  - Les boutons **haut** et **bas** de la télécommande doivent permettre d'allumer (ON) ou d'éteindre (OFF) la lampe sélectionnée.
2. **Sélection de lampe :**
  - Les boutons **gauche** et **droite** permettent de sélectionner l'ampoule ou le LED Strip.
  - La lampe sélectionnée doit être reflétée en temps réel sur le dashboard.

##### Côté dashboard :

1. **Sélection de couleur :**
  - Le dashboard doit inclure un **color picker** permettant de sélectionner une couleur pour le LED Strip.
2. **Réglage de la luminosité :**

- Un **slider** doit permettre d'ajuster la luminosité :
  - Pour l'ampoule.
  - Pour le LED Strip.
- 3. **Affichage de la lampe sélectionnée :**
  - Un **text input** doit afficher la lampe actuellement sélectionnée par les boutons gauche/droite de la télécommande.

### Scénario d'utilisation :

1. L'utilisateur sélectionne la lampe (ampoule ou LED Strip) à l'aide des **boutons gauche/droite** de la télécommande.
2. Il allume ou éteint la lampe sélectionnée avec les **boutons haut/bas**.
3. Depuis le **dashboard** :
  - L'utilisateur choisit une couleur pour le LED Strip via le **color picker**.
  - Il ajuste la luminosité de l'ampoule ou du LED Strip via le **slider**.
  - La lampe sélectionnée s'affiche dans le **text input**.

## 2. 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 STYRBAR
- Un spot LED IKEA 2104G4
- Un controleur LED Strip TuYa
- Un bandeau de LED

## 3. INSTALLATION SUR RASPBERRY PI

3.1. Connexion ssh au raspberry pi depuis le terminal :

```
ssh ciel@cielRpX.local (RpX correspond à votre raspberry ex : Rp1)
```

3.2. 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
```

3.3. Installation du broker mosquitto :

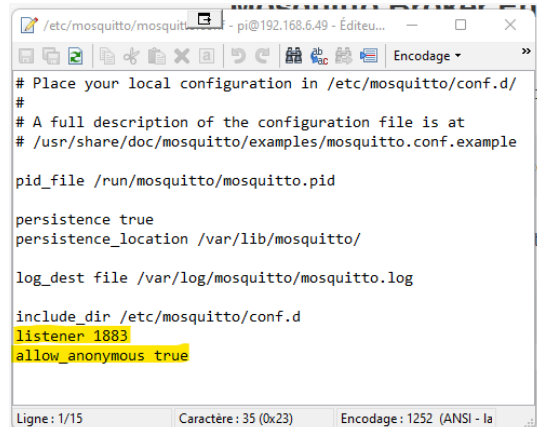
```
sudo apt update
```

```
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
```

### 3.4. Installation de zigbee2mqtt

#### 3.3.1 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**

#### 3.3.2 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 install
```

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: 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
# choisir un canal entre 11 et 26.
channel: 11
device_options:
  legacy: false
passlist:
#01
- '0x60b647fffeb7ddf6'
- '0xa4c138ec52b22b07'
- '0x84b4dbfffed15697'
devices: {}
```

Source TP2 :

<https://github.com/bouhenic/FormationIOT/blob/main/journée2/TP2Zigbee/ConfigurationTp2.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 :

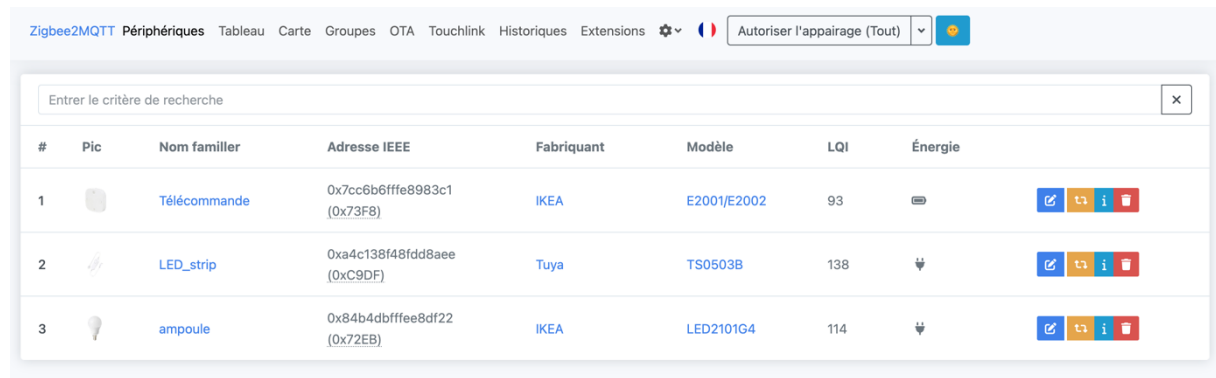
```
info 2023-01-27 16:27:24: Logging to console and directory:
'/opt/zigbee2mqtt/data/log/2023-01-27.16-27-24' filename: log.txt
info 2023-01-27 16:27:24: Starting Zigbee2MQTT version 1.29.2 (commit #1402139)
info 2023-01-27 16:27:24: Starting zigbee-herdsman (0.14.83-hotfix.0)
info 2023-01-27 16:27:30: 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 un démarrage comme un service avec systemctl voir ici :

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

## 4 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](http://<IP_RASPBERRY>:8080) ou <http://<cielrp1.local>:8080>



The screenshot shows the Zigbee2MQTT web interface. At the top, there's a navigation bar with links: Zigbee2MQTT, Périphériques, Tableau, Carte, Groupes, OTA, Touchlink, Historiques, Extensions, and a settings icon. A button 'Autoriser l'appairage (Tout)' is on the right. Below the navigation bar is a search bar 'Entrer le critère de recherche'. The main content is a table of devices:

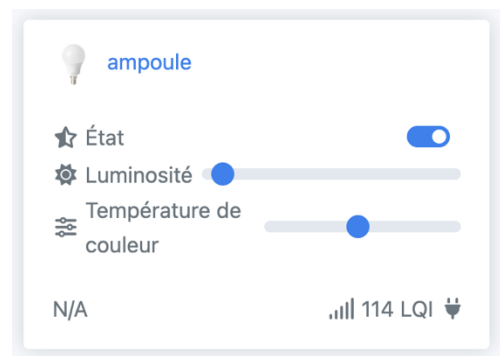
#	Pic	Nom familier	Adresse IEEE	Fabricant	Modèle	LQI	Énergie
1		Télécommande	0x7cc6b6fffe8983c1 (0x73F8)	IKEA	E2001/E2002	93	
2		LED_strip	0xa4c138f48fdd8aee (0xC9DF)	Tuya	TS0503B	138	
3		ampoule	0x84b4dbfffee8df22 (0x72EB)	IKEA	LED2101G4	114	

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 « ampoule » génère le message suivant dans le fichier log.



```
[2025-01-25 18:56:21] info:      z2m:mqtt: MQTT publish: topic 'zigbee2mqtt/ampoule', payload  
'{"brightness":10,"color_mode":"color_temp","color_temp":347,"linkquality":117,"state":"ON"  
,"update":{"installed_version":69635,"latest_version":16842784,"state":"available"}}'
```

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

```
{"brightness":10,  
"color_mode":"color_temp",  
"color_temp":347,  
"linkquality":117,  
"state":"ON",  
"update":  
{  
  "installed_version":69635,  
  "latest_version":16842784,  
  "state":"available"}  
}
```

Il s'agit d'un objet qui indique la luminosité de l'éclairage , la température de couleur, la qualité de 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 2025-01-25 18:56:21 z2m:mqtt: MQTT publish: topic 'zigbee2mqtt/ampoule', payload  
'{"brightness":10,"color_mode":"color_temp","color_temp":347,"linkquality":117,"state":"ON","update":  
{"installed_version":69635,"latest_version":16842784,"state":"available"}}'
```

## 5 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/ampoule/set -m '{"state":"ON"}'
```

Peut être lu après une commande subscribe

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

Peut aussi être lu après un /get en publish :

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

Peut être allumée avec éclairage défini :

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

Peut être allumé avec un extinction programmée :

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

- Effet :

On peut choisir un effet :

Exemple avec l'effet breathe :

```
mosquitto_pub -h localhost -t zigbee2mqtt/ampoule/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].eu.hivemq.cloud:8883/mqtt

Websocket URI

[REDACTED].eu.hivemq.cloud:8884/mqtt

### Current Usage

MQTT Client Sessions		Data Traffic	
0 / 100		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	Publish and Subscribe	DELETE

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
```

```
mqtt:
  base_topic: zigbee2mqtt
  server: 'mqtt://[redacted]eu.hivemq.cloud:8883'
  user: '[redacted]'
  password: '[redacted]'
  ca: '/etc/ssl/certs/ISRG_Root_X1.pem'
  keepalive: 60
  reject_unauthorized: false
```

On relance zigbee2mqtt depuis /opt/zigbee2mqtt

```
npm start
```

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

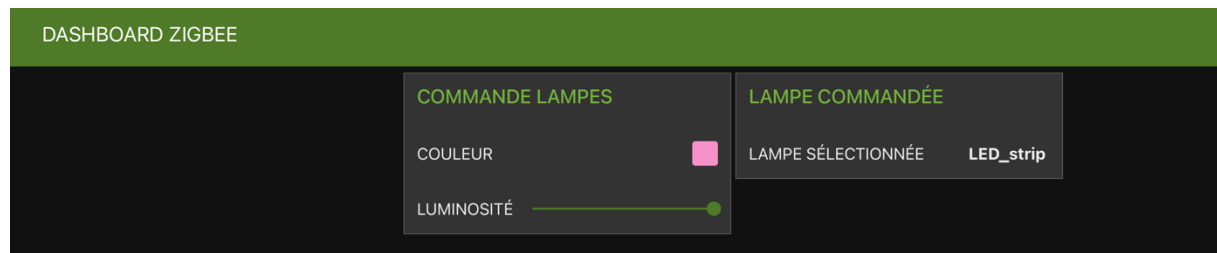
```
> zigbee2mqtt@1.33.2 start
> node index.js

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/ampoule/set -m '{"state":"TOGGLE"}'
```

## 6 UTILISATION DE NODE-RED



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

Côté télécommande :

- Les boutons haut et bas commande l'allumage ON/OFF
- Les boutons droit et gauche permettent la sélection des lampes (Ampoule ou LED),

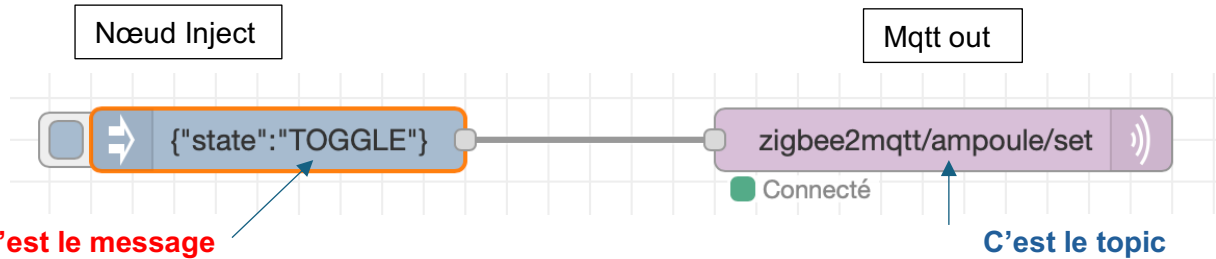
Côté Dashboard :

- Un color picker permet la sélection de la couleur du LED Strip.
- Un slider permet le réglage de la luminosité. Pour l'ampoule et le LED Strip
- Un text input affiche la sélection de lampe.

L'idée du scénario est de sélectionner l'ampoule ou le LED Strip à partir des boutons gauche/droite de la télécommande. D'allumer ou éteindre la lampe sélectionnée à partir des boutons haut/bas. La couleur du LED Strip et luminosité des deux lampes sont réglées depuis le dashboard. Celui-ci affiche également la lampe sélectionnée.

**Lancez une seconde fenêtre de terminal et connectez-vous de nouveau sur le raspberry en ssh. Cela nous permettra de lancer nodered.**

Test de la commande du spot :



Modifier le nœud inject

Supprimer Annuler Terminer

Propriétés

Nom

msg. payload = {"state": "TOGGLE"}

Modifier le nœud mqtt out

Supprimer Annuler Terminer

Propriétés

Serveur 5e771d8634234d8db8dfddd75efd80fd.s

Sujet zigbee2mqtt/ampoule/set

QoS Conserver

Nom

Pour configurer les paramètres de connexion au serveur mqtt Hivemq

Modifier le nœud mqtt out > Modifier le nœud mqtt-broker

Supprimer Annuler Sauver

Propriétés

Nom

Connexion

Serveur 5e771d8634234d8db8dfddd75efd80fd.s1.eu.l Port 8883

☒ Se connecter automatiquement

☒ Utiliser TLS Ajouter un nouveau tls-config...

Protocole MQTT V3.1.1

Client ID Laisser vide pour s'auto générer

Resteur en vie 60

Session ☒ Utiliser une session propre

Modifier le nœud mqtt out > Modifier le nœud mqtt-broker

Supprimer Annuler Sauver

Propriétés

Nom

Connexion Sécurité Messages

Nom d'utilisateur sbouhenic

Mot de passe .....

Url du serveur mqtt sur le port 8883

Indiquer le nom d'utilisateur et le mot de passe

### Test du bandeau de LED :



**Modifier le noeud inject**

Supprimer Annuler Terminer

**Propriétés**

Nom

msg. payload = `{ "state": "TOGGLE" }`

**Modifier le noeud mqtt out**

Supprimer Annuler Terminer

**Propriétés**

Serveur: 5e771d8634234d8db8dfddd75efd80fd.s

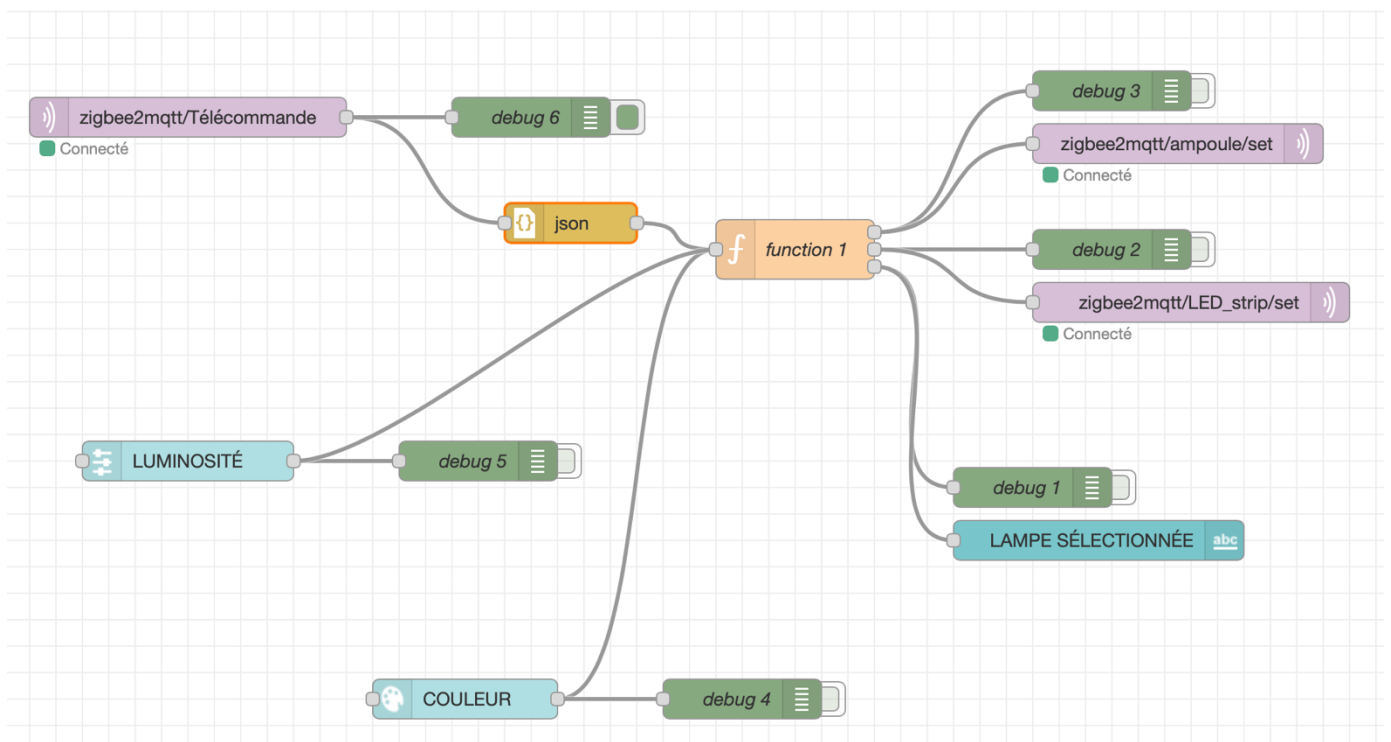
Sujet: zigbee2mqtt/ampoule/set

QoS: Conserver

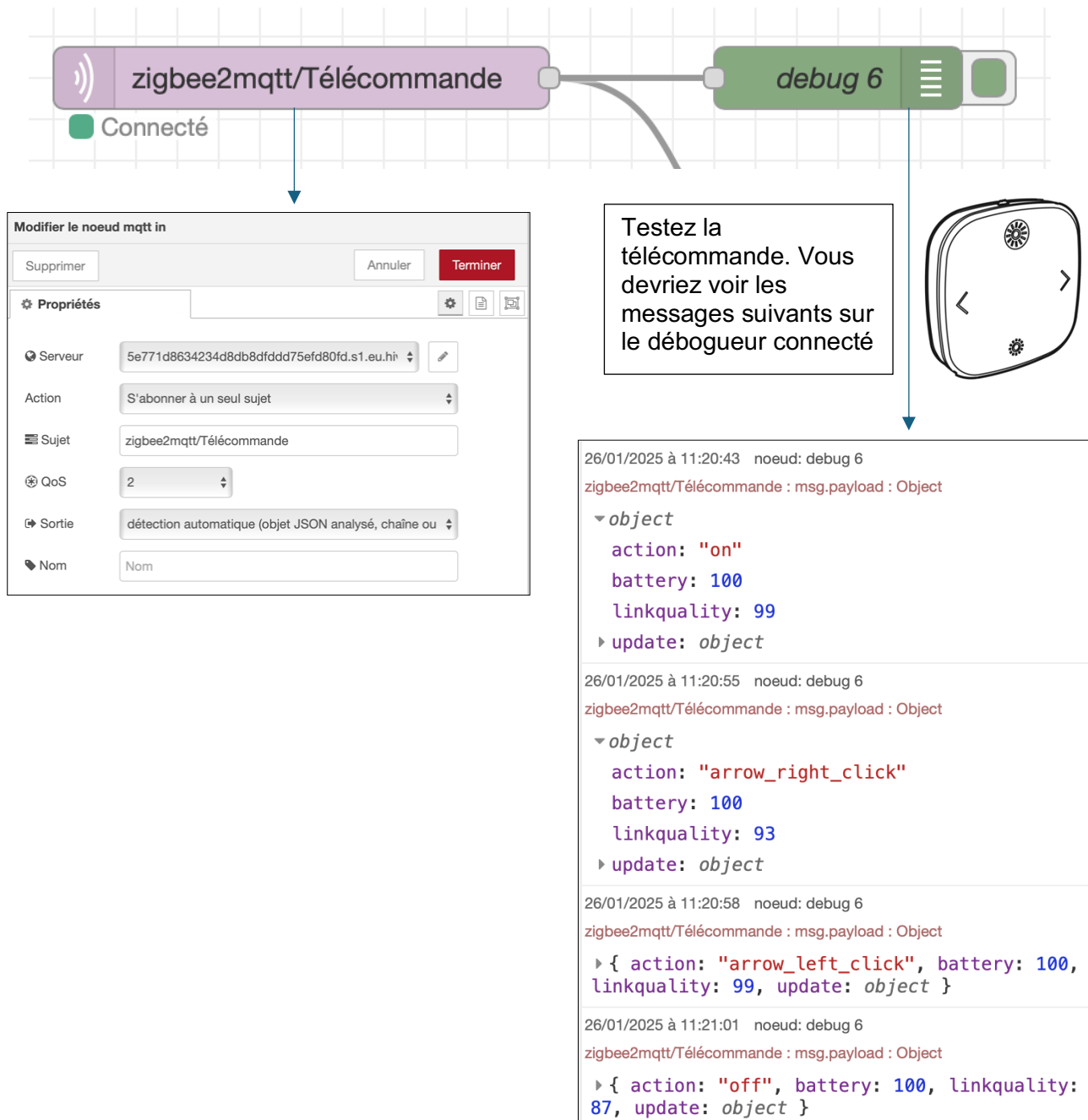
Nom

Choisir le serveur Mqtt configuré précédemment

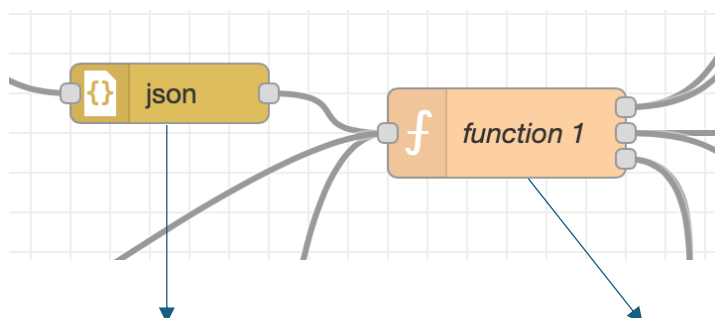
### Scénario complet :



## Test de la télécommande et du mqtt in (mqtt subscribe) :



## Mise en forme de l'objet json reçu et automatisation du scénario.



Modifier le noeud json

Supprimer Annuler Terminer

Propriétés

Action: Toujours convertir en objet JavaScript

Propriété: msg. payload

Nom: Nom

### Source fonction1 :

<https://github.com/bouhenic/FormationIOT/blob/main/journée2/TP2Zigbee/function1.js>

Le code expliqué en détail en annexe

Modifier le noeud fonction

Supprimer Annuler Terminer

Propriétés

Nom: fonction 1

Configurations Au démarrage Message reçu À l'arrêt

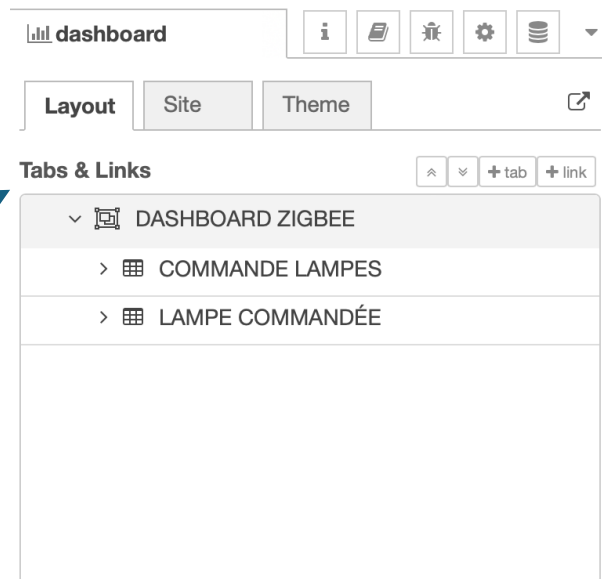
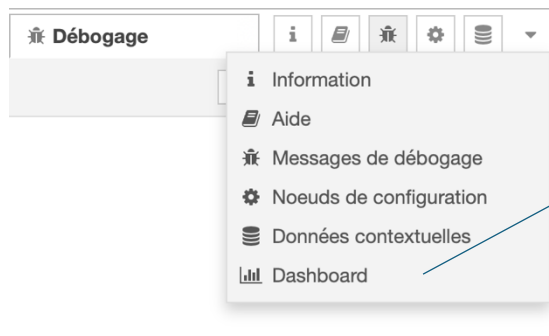
```
1 // Récupère les valeurs actuelles depuis le contexte
2 let selectedBulb = flow.get("selectedBulb") || null;
3 let brightness = flow.get("brightness") || 100;
4 let color = flow.get("color") || "#FFFFFF";
5 let spot = null;
6
7 // Debug : Affiche les valeurs actuelles
8 //node.warn('SelectedBulb: ${selectedBulb}, Brightness: ${brightness}, Color: ${color}, Spot: ${spot}');
9
10 // Mise à jour de la sélection de lampe
11 if (msg.payload.action === "arrow_left_click") {
12     selectedBulb = "left";
13     flow.set("selectedBulb", selectedBulb);
14     return [null, null, { payload: "Ampoule" }];
15 }
16
17 if (msg.payload.action === "arrow_right_click") {
18     selectedBulb = "right";
19     flow.set("selectedBulb", selectedBulb);
20     return [null, null, { payload: "LED_strip" }];
21 }
22
23 // Vérifie si une lampe est sélectionnée
24 if (!selectedBulb) {
25     return [null, null, { payload: "Aucune lampe sélectionnée" }];
26 }
27
```

Ce code gère la sélection et le contrôle de deux types de lampes (ampoule et LED Strip) à partir d'actions reçues via un message (msg.payload). Il utilise un contexte de données (flow) pour conserver l'état actuel de la sélection, de la luminosité et de la couleur. Les principales fonctionnalités sont :

1. **Sélection de lampe** : Les actions `arrow_left_click` et `arrow_right_click` mettent à jour la lampe sélectionnée (ampoule ou LED Strip).
2. **Mise à jour des paramètres** : La luminosité et la couleur peuvent être modifiées selon les commandes reçues, avec des traitements spécifiques pour chaque type de lampe.
3. **Commandes ON/OFF** : Les lampes peuvent être allumées ou éteintes, avec des états adaptés pour l'ampoule ou le LED Strip.
4. **Sorties** : Trois sorties sont configurées :
  - Première sortie pour l'ampoule.
  - Deuxième sortie pour le LED Strip.
  - Troisième sortie pour le nom de la lampe actuellement sélectionnée.

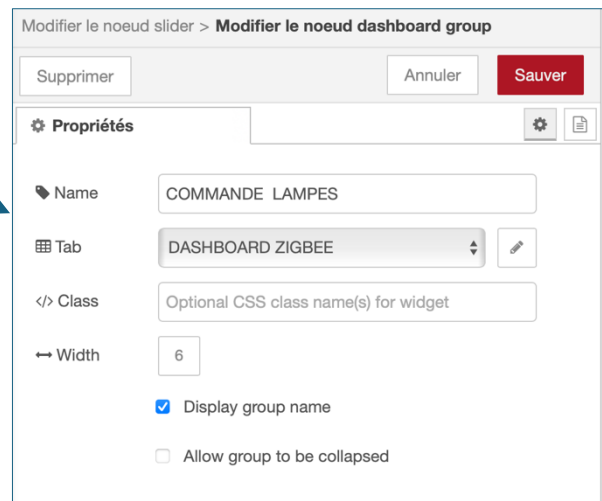
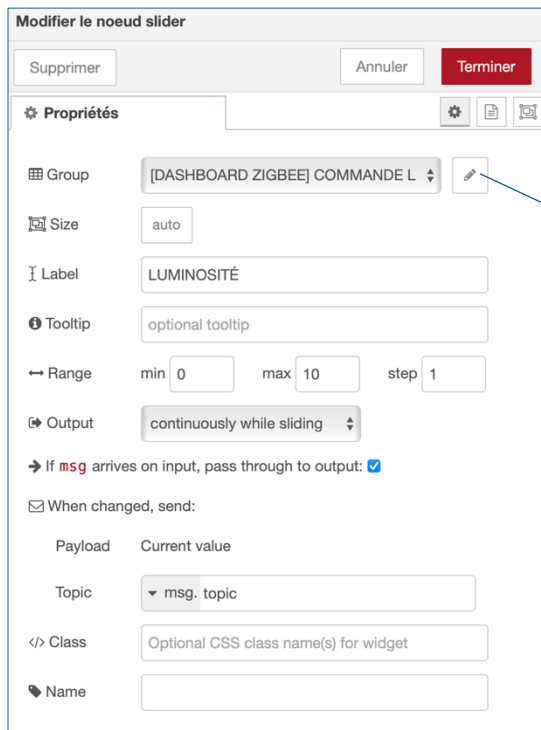
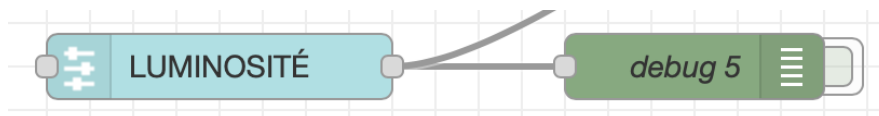
Chaque modification est sauvegardée dans le contexte, permettant une gestion persistante des états entre les interactions.

## Création du dashboard et des groupes :



Créez un dashboard et deux groupes

## Création du slider pour le réglage de la luminosité :

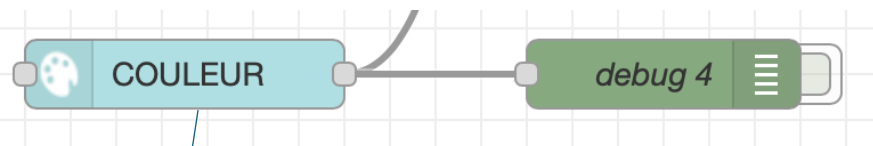


Testez le slider et vérifiez le résultat sur le débogueur connecté



```
26/01/2025 à 12:58:11 noeud: debug 5
msg.payload : number
5
26/01/2025 à 12:58:16 noeud: debug 5
msg.payload : number
4
```

## Création du color picker pour le réglage de la couleur du bandeau de LED :



The diagram shows a 'COULEUR' node (light blue) connected to a 'debug 4' node (green). An arrow points from the 'COULEUR' node to its configuration window.

**Modifier le noeud colour picker**

Supprimer Annuler Terminer

**Propriétés**

- Group: [DASHBOARD ZIGBEE] COMMANDE L
- Size: auto
- Label: COULEUR
- Format: hex round
- Show hue slider: ☐
- Show lightness slider: ☒
- If width is 4 or greater:
  - Always show switch: ☒
  - Always show picker: ☐
  - Always show value field: ☐
- If msg arrives on input, pass through to output: ☒
- Send: one value when released/closed
- Payload: current value as a string
- Topic: msg. topic
- </> Class: Optional CSS class name(s) for widget

**Modifier le noeud colour picker > Modifier le noeud dashboard group**


Supprimer Annuler Sauver

**Propriétés**

- Name: COMMANDE LAMPES
- Tab: DASHBOARD ZIGBEE
- </> Class: Optional CSS class name(s) for widget
- Width: 6
- ☒ Display group name
- ☐ Allow group to be collapsed

Testez le color picker et vérifiez le résultat sur le débogueur

26/01/2025 à 13:10:07 noeud: debug 4  
msg.payload : string[6]  
"1663e9"





## Création du text input pour afficher la lampe sélectionnée

The diagram illustrates the configuration of a text input widget. The configuration panel on the left, titled "Modifier le noeud text", shows the following settings:

- Group:** [DASHBOARD ZIGBEE] LAMPE COMM
- Size:** auto
- Label:** LAMPE SÉLECTIONNÉE
- Value format:** {{msg.payload}}
- Layout:** A grid of five labels with the text "label value".
- Style:** ☐ Apply Style
- Class:** Optional CSS class name(s) for widget
- Name:** (empty field)

The configuration panel on the right, titled "Modifier le noeud text > Modifier le noeud dashboard group", shows the following settings:

- Name:** LAMPE COMMANDÉE
- Tab:** DASHBOARD ZIGBEE
- Class:** Optional CSS class name(s) for widget
- Width:** 6
- Display group name:** ☒
- Allow group to be collapsed:** ☐

The preview on the right shows the widget's output: "LAMPE COMMANDÉE" in green text, "LAMPE SÉLECTIONNÉE" in white text, and "Ampoule" in white text.

Connectez l'ensemble du scénario et testez le text input en sélectionnant une lampe

## 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
# choisir un canal entre 11 et 26.
channel: 13
device_options:
  legacy: false
passlist:
#01
  - '0x60b647fffeb7ddf6'
  - '0xa4c138ec52b22b07'
  - '0x84b4dbfffed15697'
#02
  - '0xa4c138591e675d50'
  - '0x84b4dbfffec942a2'
  - '0x881a14fffed34e1a'
#03
  - '0xa4c138de63548d15'
  - '0x003c84fffea0a250'
  - '0x7cc6b6fffe89e8ed'
#04
  - '0xa4c1389c73550ed1'
  - '0x881a14fffed232bf'
  - '0x003c84fffea0acc9'
#05
  - '0x7cc6b6fffe8983c1'
  - '0xa4c138f48fdd8aee'
  - '0x84b4dbfffee8df22'
#06
  - '0xa4c138c4ccac0339'
  - '0x7cc6b6fffe89e994'
  - '0x84b4dbfffee8df1a'

devices: {}
```

## CODE DE LA FONCTION :

### Function 1 :

```
// Récupère les valeurs actuelles depuis le contexte
let selectedBulb = flow.get("selectedBulb") || null;
let brightness = flow.get("brightness") || 100;
let color = flow.get("color") || "#FFFFFF";
let spot = null;

// Debug : Affiche les valeurs actuelles
node.warn(`SelectedBulb: ${selectedBulb}, Brightness: ${brightness}, Color: ${color}`);

// Mise à jour de la sélection de lampe
if (msg.payload.action === "arrow_left_click") {
    selectedBulb = "left";
    flow.set("selectedBulb", selectedBulb);
    return [null, null, { payload: "Ampoule" }];
}

if (msg.payload.action === "arrow_right_click") {
    selectedBulb = "right";
    flow.set("selectedBulb", selectedBulb);
    return [null, null, { payload: "LED_strip" }];
}

// Vérifie si une lampe est sélectionnée
if (!selectedBulb) {
    return [null, null, { payload: "Aucune lampe sélectionnée" }];
}

// Gestion des modifications de luminosité ou de couleur
if (typeof msg.payload === "number") {
    // Mise à jour de la luminosité
    brightness = Math.round(msg.payload * 10); // Remap en 0-100 si nécessaire
    flow.set("brightness", brightness); // Sauvegarde dans le contexte
    node.warn(`Brightness updated: ${brightness}`);

    // Prépare le message pour transmettre immédiatement le changement
    if (selectedBulb === "left") {
        // Pour l'ampoule : uniquement `brightness`
        spot = { payload: { brightness: brightness } };
    } else if (selectedBulb === "right") {
        // Pour le LED strip : inclut `brightness` et `color`
        spot = { payload: { brightness: brightness, color: color } };
    }
}

if (typeof msg.payload === "string" && selectedBulb === "right") {
    // Mise à jour de la couleur via le color picker
    color = msg.payload.startsWith("#") ? msg.payload : `#${msg.payload}`; // Ajoute `#` si
    manquant
    flow.set("color", color); // Sauvegarde dans le contexte
    node.warn(`Color updated: ${color}`);

    // Prépare le message pour transmettre immédiatement le changement
    spot = { payload: { brightness: brightness, color: color } };
}

// Commandes on/off avec les valeurs actuelles
if (msg.payload.action === "on") {
    if (selectedBulb === "left") {
        // Pour l'ampoule : uniquement `state`
        spot = { payload: { state: "on" } };
    } else if (selectedBulb === "right") {
        // Pour le LED strip : inclut luminosité et couleur
        spot = { payload: { state: "on", brightness: brightness, color: color } };
    }
}

if (msg.payload.action === "off") {
    if (selectedBulb === "left") {
```

```

    // Pour l'ampoule : uniquement `state`
    spot = { payload: { state: "off" } };
  } else if (selectedBulb === "right") {
    // Pour le LED strip : inclut luminosité et couleur
    spot = { payload: { state: "off", brightness: brightness, color: color } };
  }
}

// Détermine la sortie
let output = [null, null, null];
if (spot) {
  output[selectedBulb === "left" ? 0 : 1] = spot;
}

// Toujours envoyer uniquement le nom de la lampe sélectionnée sur la troisième sortie
output[2] = { payload: selectedBulb === "left" ? "Ampoule" : "LED_strip" };
return output;

```

### 1. Initialisation et récupération des états du contexte

```

let selectedBulb = flow.get("selectedBulb") || null;
let brightness = flow.get("brightness") || 100;
let color = flow.get("color") || "#FFFFFF";
let spot = null;

```

- Les variables selectedBulb, brightness, et color récupèrent leurs valeurs actuelles depuis le contexte flow.
- Si aucune valeur n'est trouvée, des valeurs par défaut sont assignées :
  - selectedBulb : null (aucune lampe sélectionnée).
  - brightness : 100 (luminosité maximale par défaut).
  - color : #FFFFFF (blanc par défaut pour le LED Strip).
- La variable spot est initialisée à null et servira à stocker les données prêtes à être envoyées.

### 2. Sélection de lampe avec la télécommande

```

if (msg.payload.action === "arrow_left_click") {
  selectedBulb = "left";
  flow.set("selectedBulb", selectedBulb);
  return [null, null, { payload: "Ampoule" }];
}

if (msg.payload.action === "arrow_right_click") {
  selectedBulb = "right";
  flow.set("selectedBulb", selectedBulb);
  return [null, null, { payload: "LED_strip" }];
}

```

- Les actions arrow\_left\_click et arrow\_right\_click permettent de sélectionner respectivement l'ampoule (left) ou le LED Strip (right).
- Une fois la sélection effectuée :
  - La variable selectedBulb est mise à jour dans le contexte.
  - Un message avec la lampe sélectionnée (Ampoule ou LED\_strip) est envoyé sur la troisième sortie (index [2]).

### 3. Vérification d'une lampe sélectionnée

```
if (!selectedBulb) {  
    return [null, null, { payload: "Aucune lampe sélectionnée" }];  
}
```

- Si aucune lampe n'est sélectionnée (selectedBulb est null), un message indiquant "Aucune lampe sélectionnée" est envoyé sur la troisième sortie.

### 4. Gestion de la luminosité

```
if (typeof msg.payload === "number") {  
    brightness = Math.round(msg.payload * 10); // Remap en 0-100 si  
nécessaire  
    flow.set("brightness", brightness);  
    node.warn(`Brightness updated: ${brightness}`);  
  
    if (selectedBulb === "left") {  
        spot = { payload: { brightness: brightness } };  
    } else if (selectedBulb === "right") {  
        spot = { payload: { brightness: brightness, color: color }  
    };  
    }  
}
```

- Si le message reçu (msg.payload) est un nombre, il est interprété comme une mise à jour de la luminosité.
- La luminosité est multipliée par 10 pour convertir une plage de 0-10 en 0-100 si nécessaire.
- La valeur est sauvegardée dans le contexte.
- Un message contenant la nouvelle luminosité est préparé :
  - Pour l'ampoule (left), seul brightness est inclus.
  - Pour le LED Strip (right), brightness et color sont inclus.

### 5. Gestion de la couleur

```
if (typeof msg.payload === "string" && selectedBulb === "right") {  
    color = msg.payload.startsWith("#") ? msg.payload :  
`#${msg.payload}`;  
    flow.set("color", color);  
    node.warn(`Color updated: ${color}`);  
  
    spot = { payload: { brightness: brightness, color: color } };  
}
```

- Si msg.payload est une chaîne de caractères et que le LED Strip est sélectionné :
  - La couleur est mise à jour.
  - Si la chaîne ne commence pas par #, le symbole est ajouté.
  - La couleur est sauvegardée dans le contexte.
  - Un message contenant la luminosité et la nouvelle couleur est préparé pour le LED Strip.

## 6. Commandes ON/OFF

```
if (msg.payload.action === "on") {
  if (selectedBulb === "left") {
    spot = { payload: { state: "on" } };
  } else if (selectedBulb === "right") {
    spot = { payload: { state: "on", brightness: brightness,
color: color } };
  }
}

if (msg.payload.action === "off") {
  if (selectedBulb === "left") {
    spot = { payload: { state: "off" } };
  } else if (selectedBulb === "right") {
    spot = { payload: { state: "off", brightness: brightness,
color: color } };
  }
}
```

- Les actions on et off permettent d'allumer ou d'éteindre la lampe sélectionnée.
- Un message contenant l'état (on ou off) est préparé :
  - Pour l'ampoule : seul l'état (state) est inclus.
  - Pour le LED Strip : l'état, la luminosité et la couleur sont inclus.

## 7. Détermination des sorties

```
let output = [null, null, null];
if (spot) {
  output[selectedBulb === "left" ? 0 : 1] = spot;
}
output[2] = { payload: selectedBulb === "left" ? "Ampoule" :
"LED_strip" };
return output;
```

- Trois sorties sont définies :
  - La première sortie (output[0]) est pour l'ampoule.
  - La deuxième sortie (output[1]) est pour le LED Strip.
  - La troisième sortie (output[2]) envoie toujours le nom de la lampe sélectionnée.
- Si des données (spot) ont été préparées, elles sont envoyées sur la sortie appropriée (ampoule ou LED Strip).

### Source :

*Formation IOT (François Riotte)*