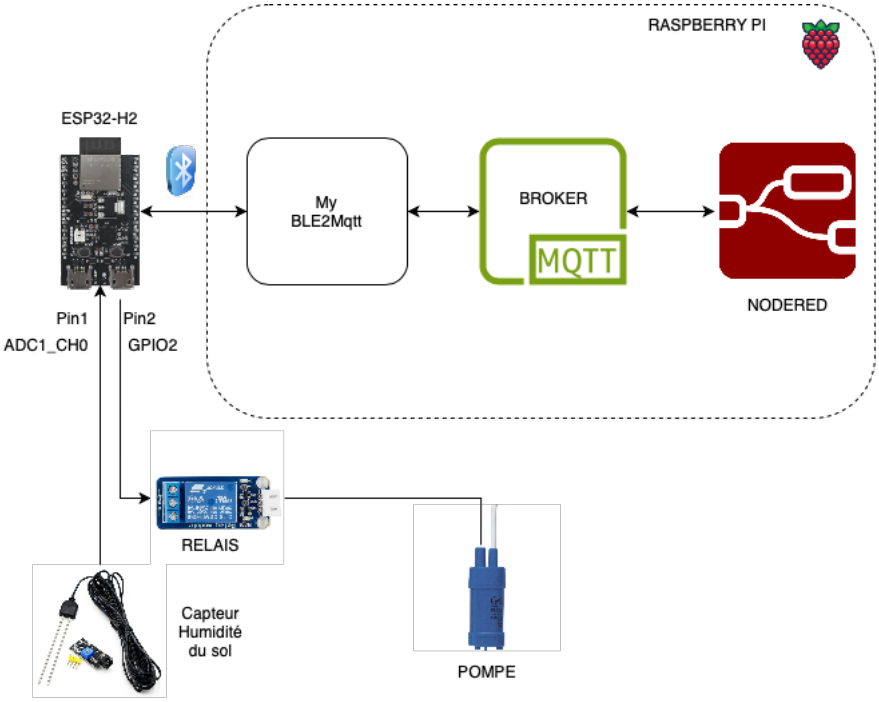
TP3 BLE
SYSTEME D'IRRIGATION INTÉRIEURE



1. MATERIEL

- 1 raspberry pi
- Un capteur d'humidité du sol.
- Une carte ESP32 H2
- Une carte relais
- Une pompe immergée 12V

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
```

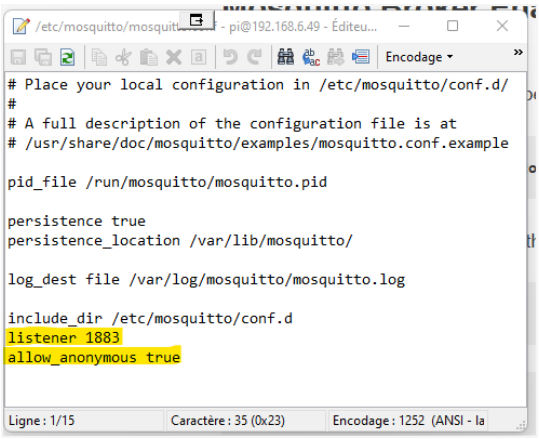
```
npm --version  # Should output 10.X or more
```

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
```

Installation des outils « python, git et bluetooth » :

```
sudo apt install git python3 python3-pip bluetooth bluez
```

Dans le cas d'un OS raspberry pi 64 bits :

```
sudo apt install libglib2.0-dev
```

Installation du module MyBLE2Mqtt :

```
sudo git clone https://github.com/bouhenic/BLE2MQTT.git /opt/BLE2MQTT
```

```
cd /opt/BLE2MQTT
```

Vous trouverez trois fichiers :

- Un fichier python qui fait passerelle BLE2MQTT
- Un fichier Arduino pour le device BLE.
- Un fichier yaml de configuration

1. installer bleak :

```
sudo apt install python3-bleak
```

2. installer paho-mqtt :

```
sudo apt install python3-paho-mqtt
```

3. installer python3-yaml :

```
sudo apt install python3-yaml
```

4. Connectez-vous sur https://www.uuidgenerator.net et générer trois UUID pour le service et les caractéristiques.

5. Recopiez les à l'emplacement désigné dans le fichier Arduino.

```
// UUIDs pour les services et les caractéristiques
#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define SENSOR_CHARACTERISTIC_UUID "beb5483d-36e1-4688-b7f5-ea07361b26a8"
#define RELAY_CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"
```

6. Modifiez le nom du BLEDevice :

```
// Create the BLE Device
BLEDevice::init("MyBleDevice");
```

7. Uploadez votre fichier dans la carte ESP32-H2.

8. Depuis le terminal du Raspberry, réalisez un scan BLE :

```
sudo hcitool lescan
```

```
LE Scan ...
48:31:B7:C0:35:0B (unknown)
48:31:B7:C0:35:0B MyBleDevice
CE:94:80:94:13:7C (unknown)
C4:7C:8D:6A:DA:F5 (unknown)
44:96:7D:F3:71:F8 (unknown)
44:96:7D:F3:71:F8 (unknown)
```

9. Relevez l'adresse MAC de votre device BLE.

10. Dans le fichier yaml, modifiez le nom du device, le service_uuid et les characteristic_uuid, vos topics.

```
mqtt:
  broker: 'localhost'
  port: 1883
  topic_sensor: 'ble/sensor'
  topic_relay_command: 'ble/relay/command'

ble:
  device_name: 'MyBleDevice'
  sensor_characteristic_uuid: 'beb5483d-36e1-4688-b7f5-ea07361b26a8'
  relay_characteristic_uuid: 'beb5483e-36e1-4688-b7f5-ea07361b26a8'
```

11. Exécutez votre fichier python en tâche de fond :

```
nohup python3 ble2mqtt.py &
```

12. Testez en publiant sur le broker :

Pour actionner l'arrosage :

```
mosquitto_pub -h localhost -t ble/relay/command -m 1
```

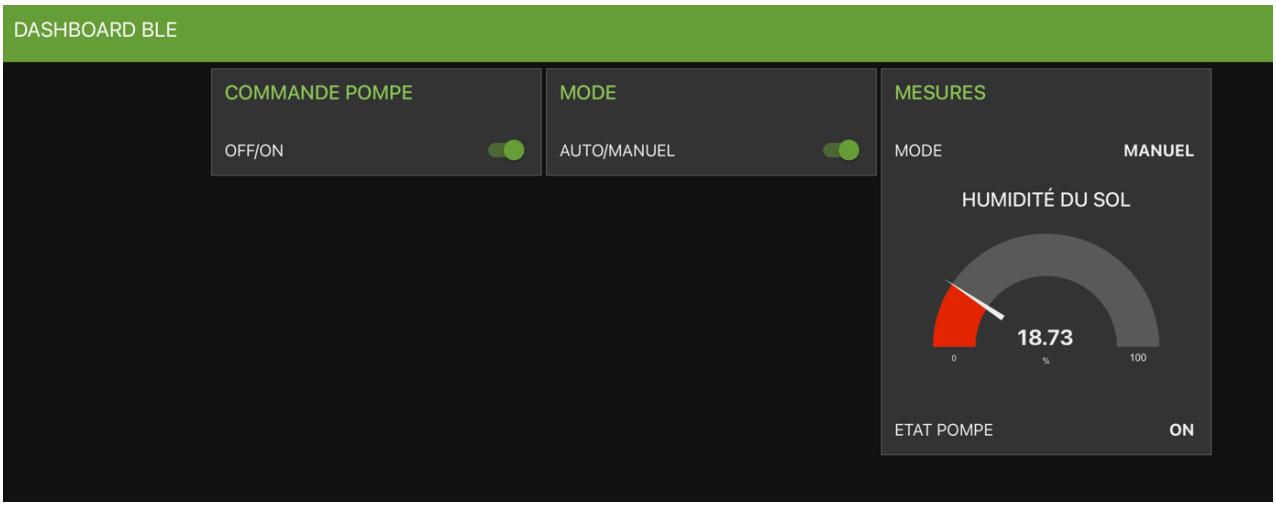
Pour stopper l'arrosage :

```
mosquitto_pub -h localhost -t ble/relay/command -m 0
```

13. Testez en s'abonnant sur le broker :

```
mosquitto_sub -h localhost -t ble/sensor
```

3. UTILISATION DE NODE-RED



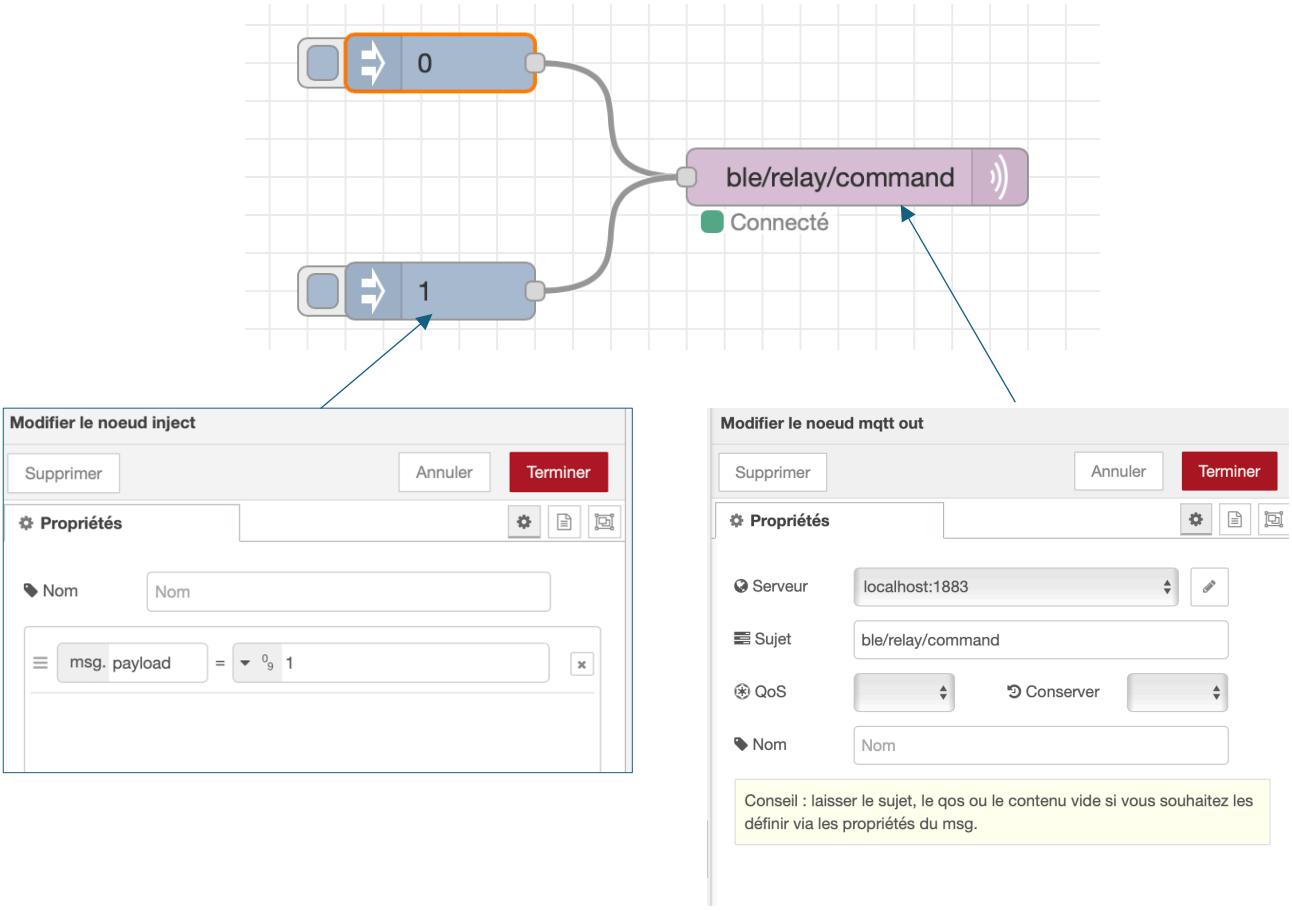
L'objectif est de faire le dashboard ci-dessus. Il est constitué de:

- Un switch OFF/ON pour commander la pompe en mode manuel.
- Un switch AUTO/MANUEL pour choisir le mode de fonctionnement. Le mode automatique dévalide le fonctionnement du switch ON/OFF
- Une gauge pour visualiser la valeur de l'humidité.
- Une entrée texte pour indiquer le mode.
- Une entrée texte pour indiquer l'état de la pompe.

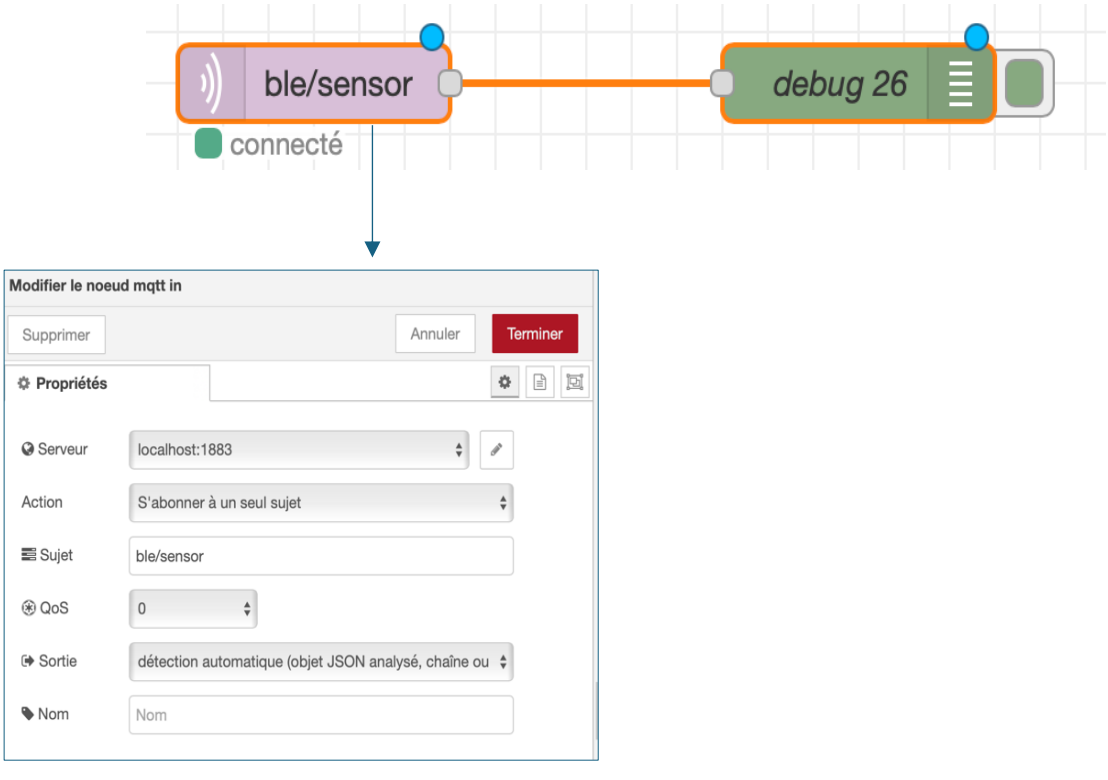
Lancer Node-red :

```
sudo node-red start
```

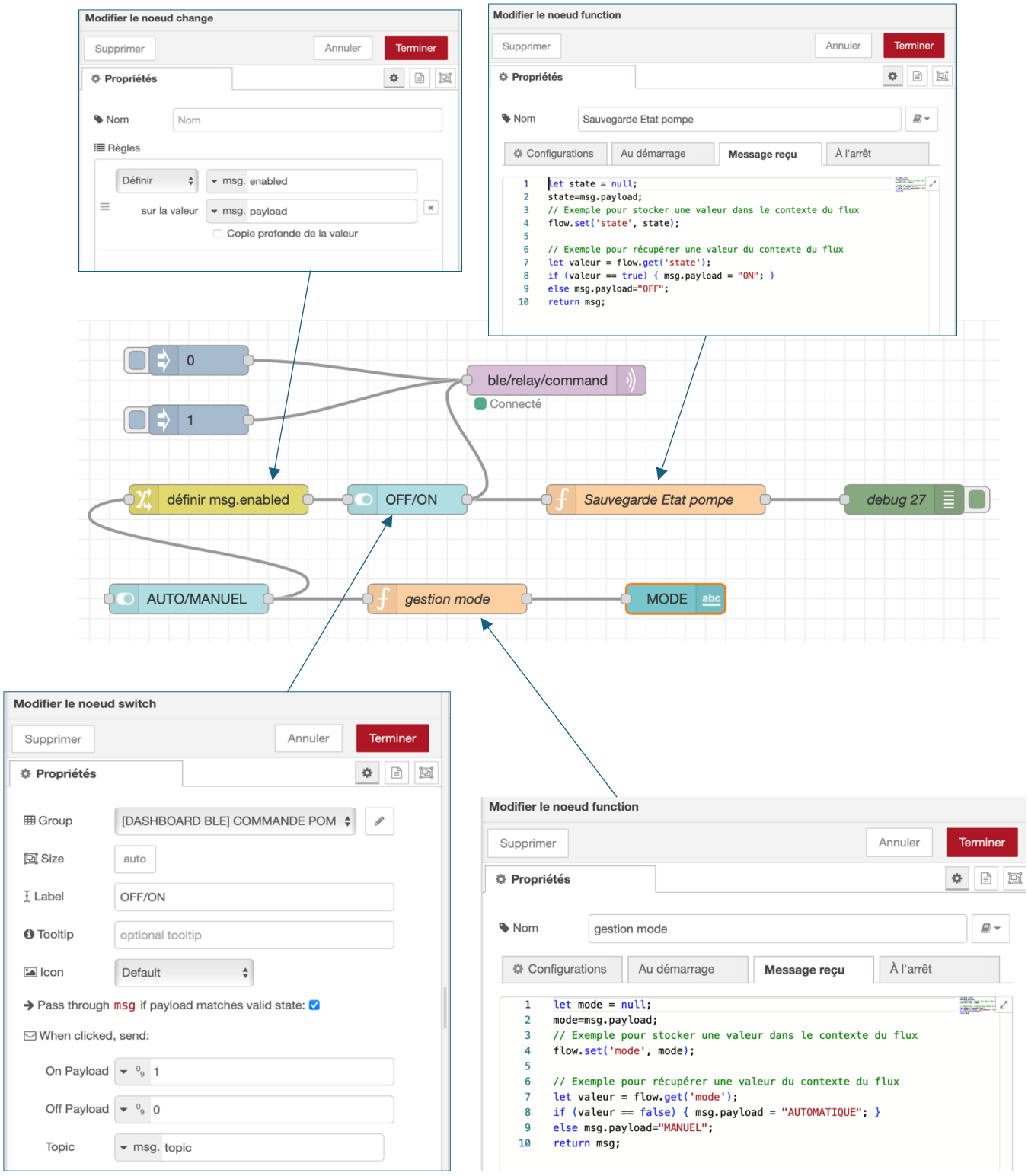
Test du publish, commande de la pompe :



Test du subscribe, mesures de l'humidité du sol :



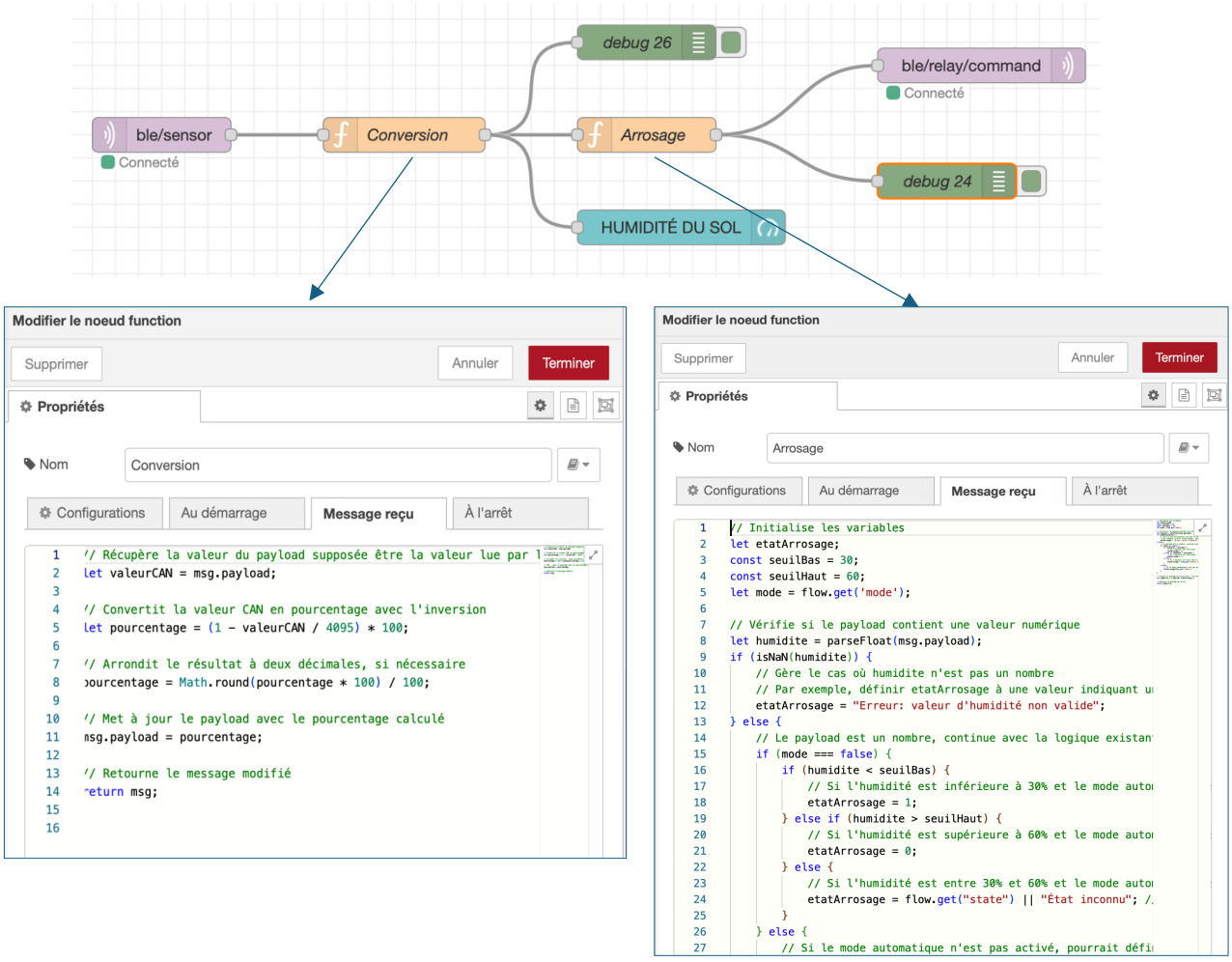
Ajout des switchs ON/OFF et AUTO/MANUEL. Dévalidation du switch ON/OFF en fonction du mode sélectionné.



Source fonction gestion mode :
https://github.com/bouhenic/FormationIOT/blob/main/Tp3Ble/gestionMode.js

Source fonction Sauvegarde état pompe :
https://github.com/bouhenic/FormationIOT/blob/main/Tp3Ble/SauvegardeEtatPompe.js

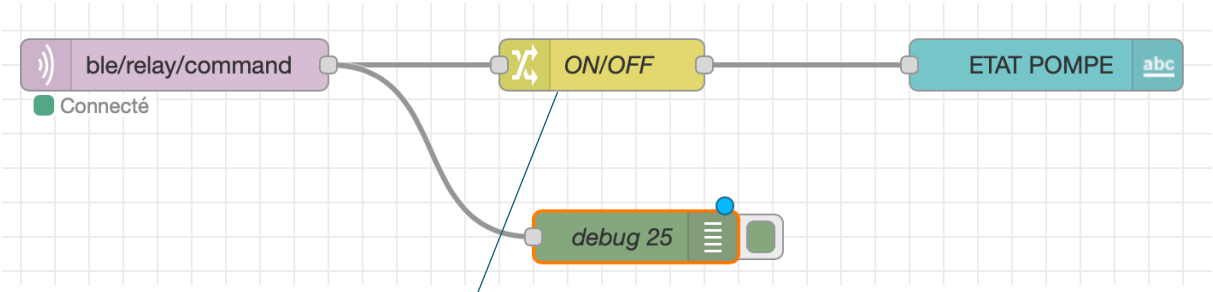
Codage de l'automatisation de l'arrosage en fonction du mode et de la température mesurée.

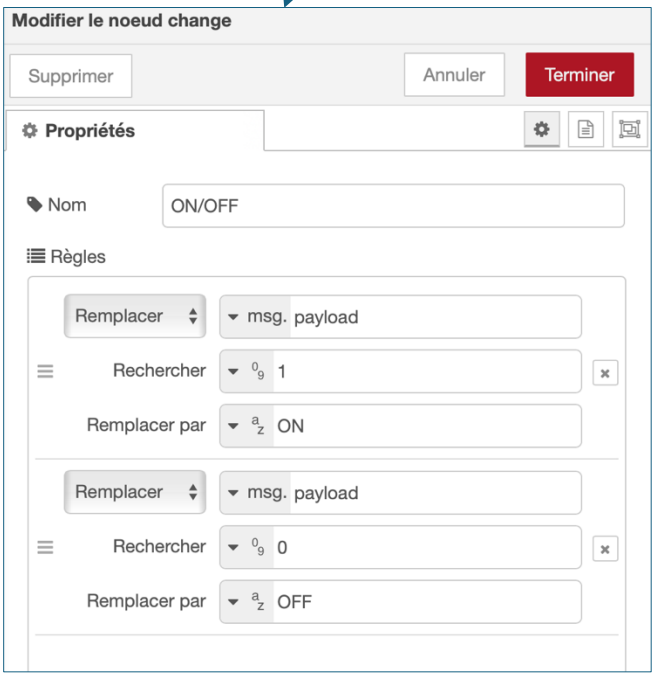


Source fonction arrosage :
https://github.com/bouhenic/FormationIOT/blob/main/Tp3Ble/arrosage.js

Source fonction conversion :
https://github.com/bouhenic/FormationIOT/blob/main/Tp3Ble/conversion.js

Gestion de l'affichage de l'état de la pompe





Fonction Conversion :

```
// Récupère la valeur du payload supposée être la valeur lue par le CAN
let valeurCAN = msg.payload;

// Convertit la valeur CAN en pourcentage avec l'inversion
let pourcentage = (1 - valeurCAN / 4095) * 100;

// Arrondit le résultat à deux décimales, si nécessaire
pourcentage = Math.round(pourcentage * 100) / 100;

// Met à jour le payload avec le pourcentage calculé
msg.payload = pourcentage;

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

Fonction gestion mode :

```
let mode = null;
mode=msg.payload;
// Exemple pour stocker une valeur dans le contexte du flux
flow.set('mode', mode);

// Exemple pour récupérer une valeur du contexte du flux
let valeur = flow.get('mode');
if (valeur == true) { msg.payload = ="AUTOMATIQUE"; }
else msg.payload="MANUEL";
return msg;
```

Fonction Stockage état pompe :

```
let state = null;
state=msg.payload;
// Exemple pour stocker une valeur dans le contexte du flux
flow.set('state', state);

// Exemple pour récupérer une valeur du contexte du flux
let valeur = flow.get('state');
if (valeur == true) { msg.payload = "ON"; }
else msg.payload="OFF";
return msg;
```

Fonction Arrosage :

```
// Initialise les variables
let etatArrosage;
const seuilBas = 30;
const seuilHaut = 60;
let mode = flow.get('mode');

// Vérifie si le payload contient une valeur numérique
let humidite = parseFloat(msg.payload);
if (isNaN(humidite)) {
    // Gère le cas où humidite n'est pas un nombre
    // Par exemple, définir etatArrosage à une valeur indiquant une erreur ou ignorer la mise à jour
    etatArrosage = "Erreur: valeur d'humidité non valide";
} else {
    // Le payload est un nombre, continue avec la logique existante
    if (mode === false) {
        if (humidite < seuilBas) {
            // Si l'humidité est inférieure à 30% et le mode automatique est activé, déclenche
l'arrosage
            etatArrosage = 1;
        } else if (humidite > seuilHaut) {
            // Si l'humidité est supérieure à 60% et le mode automatique est activé, stoppe
l'arrosage
            etatArrosage = 0;
        } else {
            // Si l'humidité est entre 30% et 60% et le mode automatique est activé, ne change rien
            etatArrosage = flow.get("state") || "État inconnu"; // Utilise l'état actuel de
l'arrosage ou un message par défaut si non disponible
        }
    } else {
        // Si le mode automatique n'est pas activé, pourrait définir etatArrosage à une valeur
spécifique ou le laisser indéfini
        etatArrosage=flow.get('state');
    }
}

// Prépare le message de sortie avec l'état de l'arrosage
let msgSortie = { payload: etatArrosage };

// Retourne le message de sortie
return msgSortie;
```