



Rapport de TP de SAM

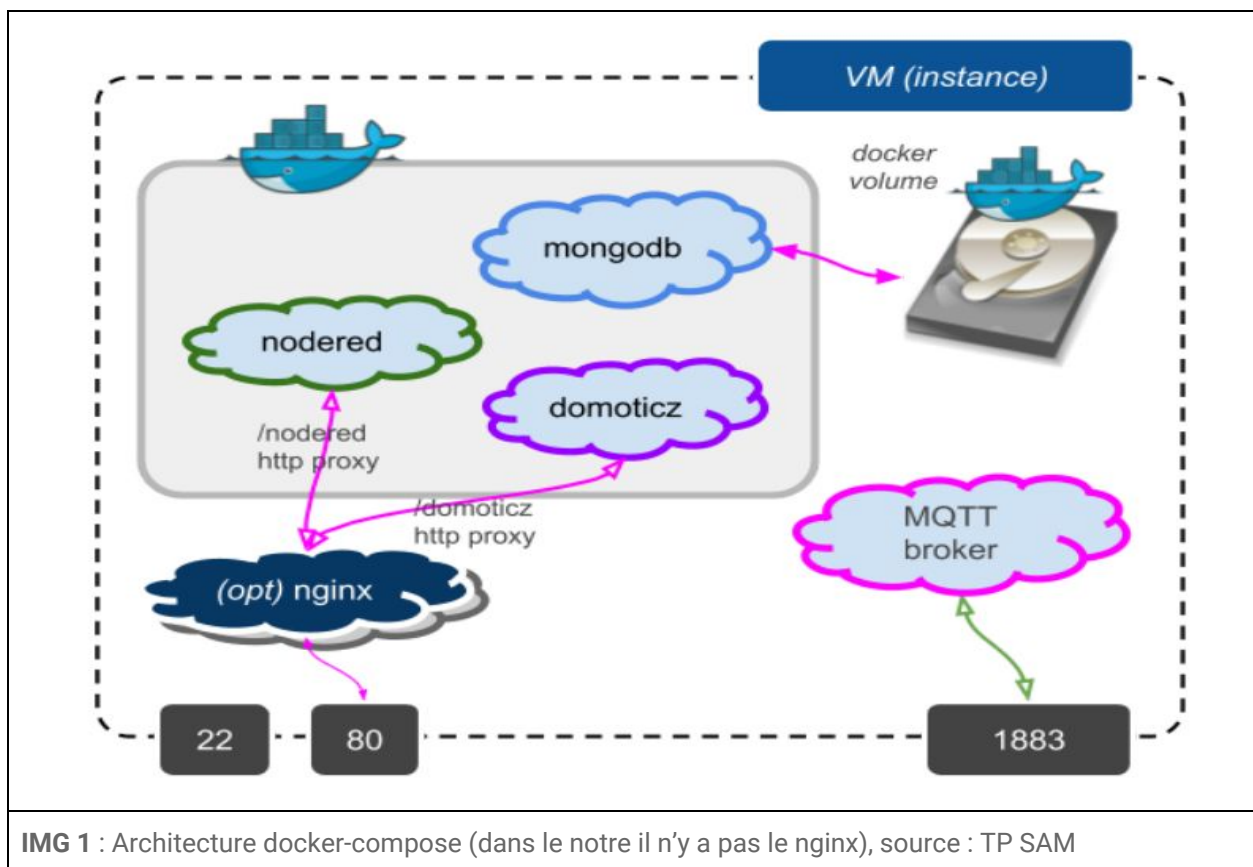
KRUGLOV Nikita, MMADI Anzilane
20/12/2019

Introduction

Ce document a pour but d'expliquer les technologies utilisées et les solution mise en oeuvre tout au long des séances de TP de SMAE. Dans le cadre de ces TPs, on va apprendre à manipuler un Raspberry Pi B et des capteurs sur breadboard pour simuler une maison intelligente. On doit donc être capable de remonter la température, monitorer la luminosité de la maison en jouant avec les lumière ou les volet roulant et de façon automatique ou manuel.

Configuration de l'environnement de déploiement

Création du fichier de Docker-compose.yaml et configuration de Domoticz et de NodeRed



DOMOTICZ-MQTT configuration

Ici on présente une configuration de MQTT sur domoticz et on configure aussi un lampe

nous permettant la gestion de la lumière dans la salle R14.

Configuration de docker

La configuration de tous les containers de docker peuvent être trouver dans le dossier *docker* de ce projet. Pour initialiser le docker, il suffit de saisir

```
$ sudo docker-compose up
```

pour que docker-compose puisse initialiser nodred, domoticz et mongodb.

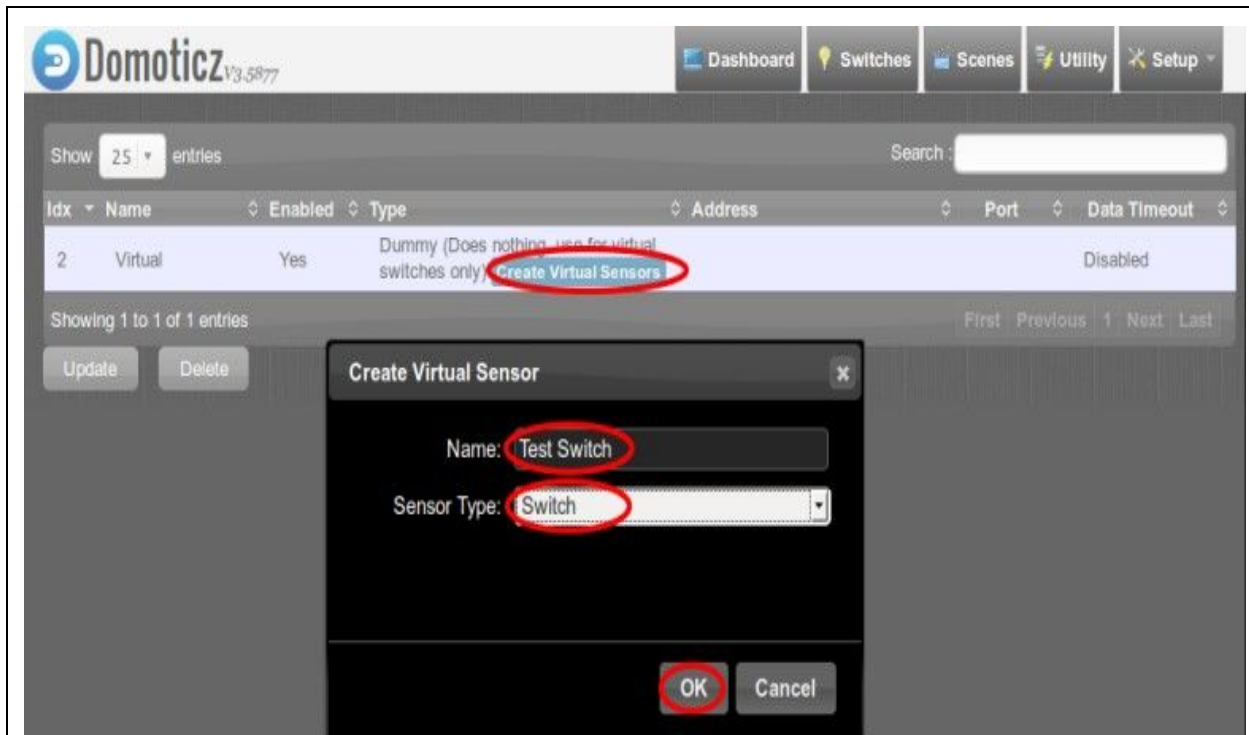
Configuration de Domoticz

Depuis setup aller sur hardware pour avoir cette fenêtre de configuration

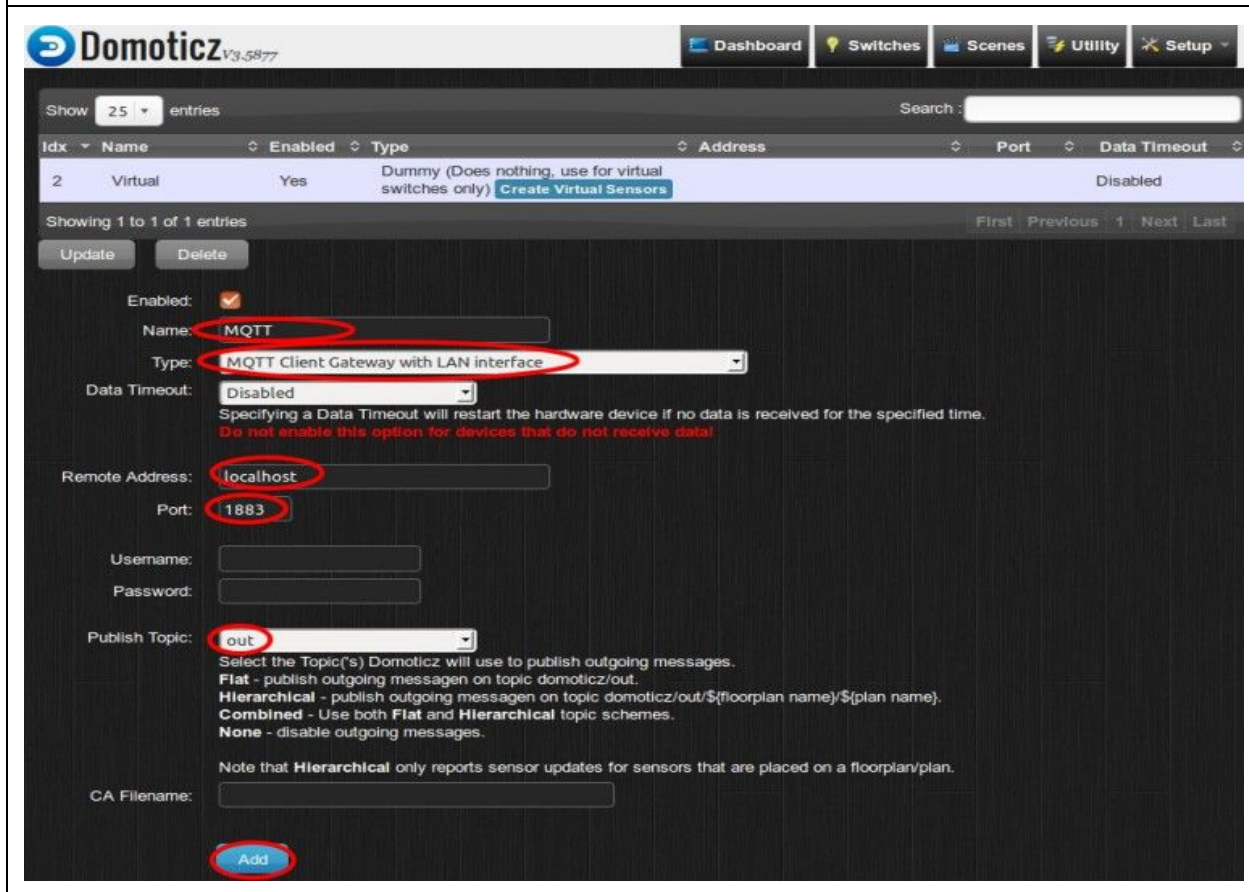
The screenshot shows the Domoticz web interface in the 'Setup' section. The 'Hardware' tab is active. A table lists hardware devices, but it is empty. Below the table, there are fields for adding a new device: 'Enabled' (checked), 'Name' (set to 'Virtual'), 'Type' (set to 'Dummy (Does nothing, use for virtual switches only)'), and 'Data Timeout' (set to 'Disabled'). The 'Add' button is highlighted with a red circle.

Cliquez sur le bouton **Create Virtual Sensors** dans le nouveau matériel **Virtual**.

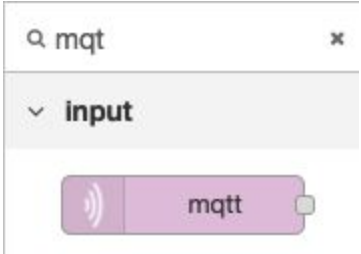

Remplissez le champs **Name**, sélectionnez **Switch** dans la liste déroulante de **Sensor Type** dans la fenêtre **Create Virtual Sensor**.



Configuration de MQTT



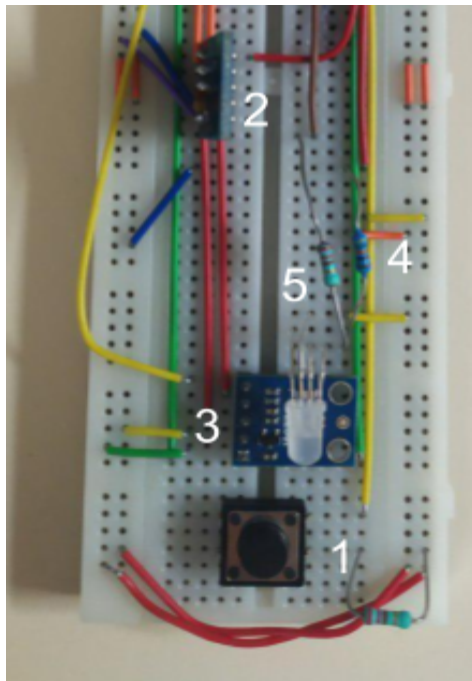
Configuration de Node-red-MQTT

	
Node MQTT au flow courant	Configurer la connexion à Mosquitto

Installation des capteurs

Avant de commencer, il nous faut installer le système sur notre Raspberry Pi, et d'installer tout le soft nécessaire.

Ensuite, on a installé `pip3`, `paho-mqtt` avec `mosquitto` et `MQTTbox` installés sur l'ordinateur de la fac.



Breadboard

Sur notre breadboard on a placé 3 *capteurs* (la bouton pour simuler la présence dans la salle (1), un capteur de température (2) et de lumière (3)).

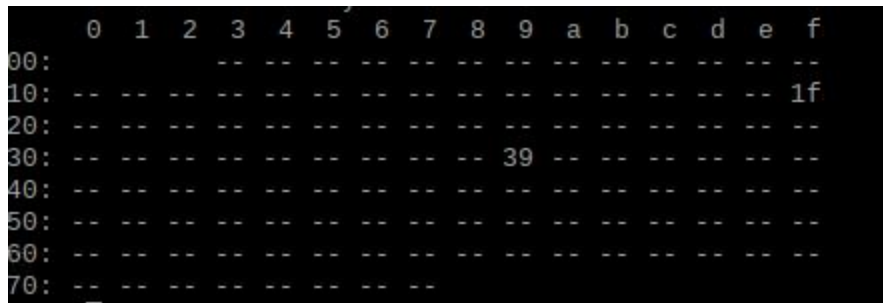
Aussi, on a les *actionneurs* : Une led RGB branché avec deux résistances de 440 Ohm (4) et de 220 Ohm (5). Les résistances différentes sont choisies pour simuler l'intensité de la lumière : $R = 440 \text{ Ohm}$ représente la lumière venant de l'extérieur (plus faible $\sim 670 \text{ lux}$), $R = 220$ représente les lampes (plus fort $\sim 1200 \text{ lux}$). Les 3 couleurs activés en même temps donne autours de 1600 lux. Sachant que le lumière de base dans la salle est autours de 200 lux, avec ce led on peut simuler le comportement lié à la lampes dans la salle et le comportement des volets roulants (si les roulés sont ouvert, le lumière de l'extérieur passe et si non on se

base sur les lampes seulement).

I2C Sensor integration (TP4)

Dans notre disposition on a utilise deux capteurs : MCP9808 (Capteur de température) et TSL2561 (Capteur de luminosité). Les adresses I2C de base sont 1F et 39 respectivement avec les adresses de lecture 1F et 39.

<Les résultats avec i2cscan -y 1>



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
10:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	1f
20:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
30:	--	--	--	--	--	--	--	--	39	--	--	--	--	--	--	--
40:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
50:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
60:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
70:	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Ensuite, pour confirmer le, on a implémenté notre propre i2cScan fonction suivante :

```
def i2c_scan():
    _I2C_ADDR_RANGE = 127

    try :

        bus = SMBus(1)

    except:

        return []

    devices = []

    for addr in range(0, _I2C_ADDR_RANGE) :

        try :

            bus.write_quick(addr, 0x00)

            devices.append(addr)

        except:

            pass
```

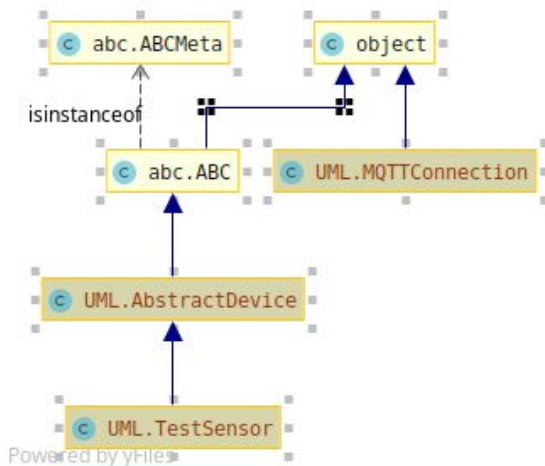


```
return devices
```

Pour interagir avec les capteurs, on a utilisé les librairies proposées par Adafruit (avec son circuit python). Il faut les installer grâce aux commandes : `sudo pip3 install adafruit-blinka`, `sudo pip3 install adafruit_mcp9808` et `sudo pip3 install adafruit_tsl2561`. Avec ces librairies, on peut lire les données des capteurs directement sans les manipulations avec le `smbus`. Vu que les interruptions n'étaient pas intéressantes dans le contexte du TP7, on ne les a pas implémentées.

LE TP 7

Code des capteurs



Du point de vue du code, c'est le langage python qui est utilisé et cela nous permet de bénéficier de la puissance de l'orienté objet :

- class **MQTTConnection** qui représente la connexion d'un agent à une broker MQTT. Cette class est responsable de transfert des messages , mais aussi à cacher la complexité liée au manipulation avec le connecteur MQTT
- class abstract **AbstractDevice** nous permet d'unifier les actions communes pour tous les agents connectés au broker MQTT telle que la publication de message et le traitement de commandes communes pour tous les agents : publication immédiate des statuts,

mis à jour lors d'une publication régulière. Pour transformer l'AbstractDevice en capteur il suffit d'implémenter la fonction `read_data()` et `get_address()` ainsi que l'appelle du constructeur de la super-class. Dans le cas de l'actionneur il faut aussi implémenter la fonction `message_handler(payload)` pour traiter le message JSON qui contient l'ordre particulier pour cet actionneur.

- class Test sensor représente une des implémentations possibles des capteurs.

mqtt-tp7 [mqtt-project] ~/P3

AbstractDevice.py

Clock.py

connection.py

LampController.py

LuminositySensor.py

PresenceDetector.py

ShuttersController.py

start.py

start_fake_sensor.py

TemperatureSensor.py

Implementation

Tous ces agents décrits ci-dessus sont implémentés chacun dans un fichier particulier.

1. Clock - un simple pseudo-capteur qui publie la date et le pseudo-temps à chaque interval de temps (20 sec = 1 heure par défaut) ⇒ dans `/time/data`
2. LampController - Représente la lampe (les couleurs vert et bleu d'une LED branché au pin 16)

3. LuminositySensor - Lit les données du capteur de luminosité
4. Presence detector - Détecte les interruptions venu du bouton branché au pin 20)
5. ShuttersController - Simule le comportement des volets roulants : il s'occupe des commandes du "shutter". Aussi, il allume ou non la led rouge en fonction du temps reçu par le système et le statut des volets. Il doit donc recevoir le temps publié dans /time/data.
6. TemperatureSensor - Lit les données du capteur de température.

Tous les capteurs se lancent avec notre setup, grâce au script start.py.

Comportement souhaité du système

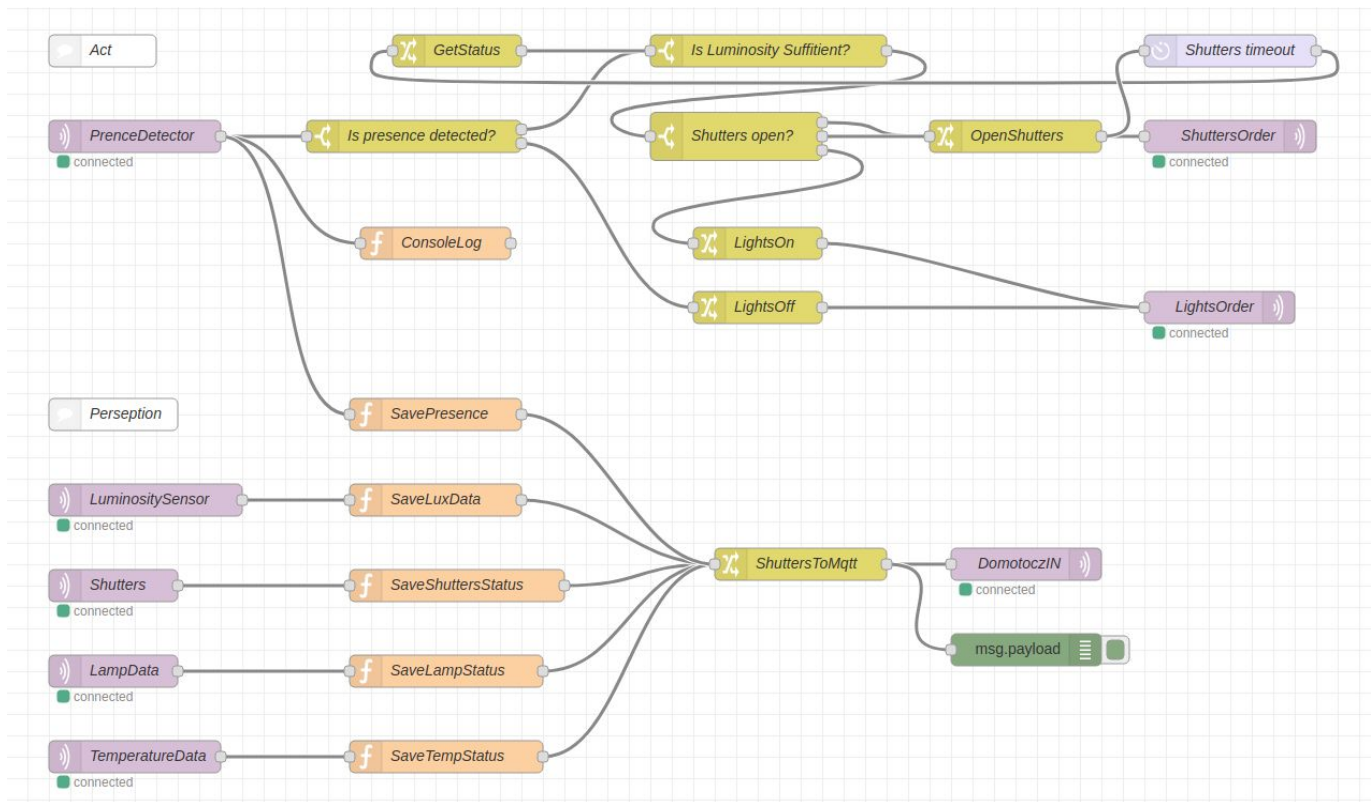
Le rôle de notre système est en grande partie la gestion de la luminosité dans la "salle". Et la logique de fonctionnement est super-simple et gérée par Nodered :

1. Tous les capteurs publient ses valeurs à intervalle de temps régulier (cette intervalle varier en fonction du capteur).
2. Si le niveau de luminosité dans la salle est suffisant, on ne fait rien
3. Si le niveau de luminosité est inférieur à un certain seuil :
 - a. Si une présence dans la salle est détectée :
 - i. Si les volets sont fermés et qu'on est entre 7h et 20h
 1. Ouvrir les volets (cela a pour effet d'augmenter la luminosité de la salle sans utiliser les lampes).
 2. Si malgré tout la luminosité reste inférieur au seuil :
 - a. Allumer les lampes (lumières)
 - ii. Si les volets sont ouverts et qu'on est entre 7h et 20h
 1. Allumer les lampes (lumières)
 - iii. Si les volets sont ouverts et qu'on n'est pas entre 7h et 20h
 1. Fermer les volets et allumer les lumières
 - iv. Sinon
 1. allumer les lumières
 - b. Si aucune présence dans la salle n'est détectée : on ne fait rien.

Chaîne des tests

Pour tester notre programmes qui correspond aux différentes parties du TP, on a décidé de tester d'abord sur notre ordinateur (aussi équipé de Mosquitto et MQTTbox) avec les "placeholders" à la place des fonctions qui doivent renvoyer des valeurs de part des capteurs. Puis, on les va tester sur le RaspberryPi pour debugger le code lié au capteurs.

Création d'un flow sur Node Red



Sur l'image ci-dessus on peut observer le flow des donnée construit grâce à nodered. Il s'agit d'algorithm présenté dans la section "*Comportement souhaité du système*". Il s'agit de 2 partie principales :

- **perceive** - dans cette partie on reçoit les données des capteurs (topics de genre /1R1/014/<device>/data) dont on a besoin pour faire la décision et on stocke ces données dans les variables globales. En même temps on transforme ces données en format json de domoticz et on monte ce packets transformé vers le domoticz
- **act** - dans cette partie on utilise les données reçus pour diriger vers les actionneurs correspondants dans les topics /1R1/014/<device>/orders (d'abord vers les volets roulants, puis, si toujours pas de lumière - vers la lamp)

Le sauvegarde des données dans le mongodb n'était pas fait encore, mais sera effectuer dans les plus courte délais.

Domiticz

Avant de relié les capteurs à Dotomicz, on doit faire la travail préliminaire pour ajouter la connection de MQTT ainsi que de créer l'ensemble des capteurs virtuelles. Après, on va pusher le données dans ses capteurs via la connection à broqueur MQTT.

Show **25** entries

Search:

Idx	Name	Enabled	Type	Address	Port	Data Timeout
11	Sensors	Yes	Dummy (Does nothing, use for virtual switches only) Create Virtual Sensors			Disabled
9	MQTT_Central	Yes	MQTT Client Gateway with LAN interface	192.168.96.1	1883	Disabled

Showing 1 to 2 of 2 entries

First Previous 1 Next Last

Update Delete

Enabled: ☒

Name:

Type:

Data Timeout:

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.
Do not enable this option for devices that do not receive data!

Add

Ensuite, on va rajouter 5 capteurs (devices) des types correspondants:

Show **25** entries

	Idx	Hardware	ID	Unit	Name	Type	SubType	
	6	Sensor	14056	1	Temperature	Temp	LaCrosse TX3	0.0 C
	5	Sensor	00014055	1	Presence	Light/Switch	Switch	Off
	4	Sensor	00014054	1	Lamp	Light/Switch	Switch	Off
	3	Sensor	00082003	1	Shutters	General	Text	Hello World
	2	Sensor	82002	1	LuminosityData	Lux	Lux	400 Lux

Chaque capteur est adressable par son "idx". Pour écrire la valeur il faut avoir dans notre json le champs "nvalue" (numeric value) si les données est une chiffre ou 0/1 logique et le champ "svalue" (string value) si les données est un string (comme dans le cas de l'état de de shutters).

MongoDB

...

Conclusion

Pendant le travail sur ce TP on a appris à utiliser le stack des technologies IoT sur plusieurs couches : perception, data management, data storage, data representation. On a pris en main en pratique le Docker, NodeRed, Domoticz, MongoDB. Aussi, on a continuer notre pratique avec les GPIO de RaspberryPI et les capteurs à l'interface I²C. En fin, avec tout ces connaissances on a créé la système de manipulation avec le lumière dans notre moquette de salle.