**TP 4 : Développement Node-Red**

**Notions abordées :**

* Mise en œuvre de node-red
* Acquisition de données via MQTT sur mosquitto.org ou serveur oneM2M
* Mise en œuvre d’un dashboard
* Sauvegarde de données

**Objectif :**

• Installer **node-red**.

• Connecter **node-red** aux serveurs The Things Network via MQTT.

• Réaliser un dashboard pour présenter les données a l’utilisateur final.

• Enregistrer les données dans une base de données.

**Rendus :**

Vous allez rendre ce TP dans la boite de dépôts *rendusTP3* de moodle.

Ce rendu comprend :   
1- le présent fichier word dans lequel vous mettrez des copies pour vérifier l’avancée du TP

# Installation node.js et node-red

Installation de Node-JS

Node.js doit être installer sur votre machine.

Les procédures d’installation selon votre OS sont disponibles ici : <https://nodejs.org/en/download/package-manager/>

**Sous Windows** : Lancez un powershell et exécutez les commandes suivantes

# installs fnm (Fast Node Manager)

winget install Schniz.fnm

# configure fnm environment

fnm env --use-on-cd | Out-String | Invoke-Expression

# download and install Node.js

fnm use --install-if-missing 20

# verifies the right Node.js version is in the environment

node -v # should print `v20.17.0`

# verifies the right npm version is in the environment

npm -v # should print `10.8.2`

**Sous Linux** : Lancez un terminal bash et exécutez les commandes suivantes

# installs fnm (Fast Node Manager)

curl -fsSL https://fnm.vercel.app/install | bash

# activate fnm

source ~/.bashrc

# download and install Node.js

fnm use --install-if-missing 20

# verifies the right Node.js version is in the environment

node -v # should print `v20.17.0`

# verifies the right npm version is in the environment

npm -v # should print `10.8.2`

Installation de Node-red

En cas d’installation préalable, vérifiez que la version de node.js est compatible avec la version de Node-Red que vous allez utiliser.

<https://nodered.org/docs/getting-started/local>

**Sous windows :**

>npm install -g --unsafe-perm node-red

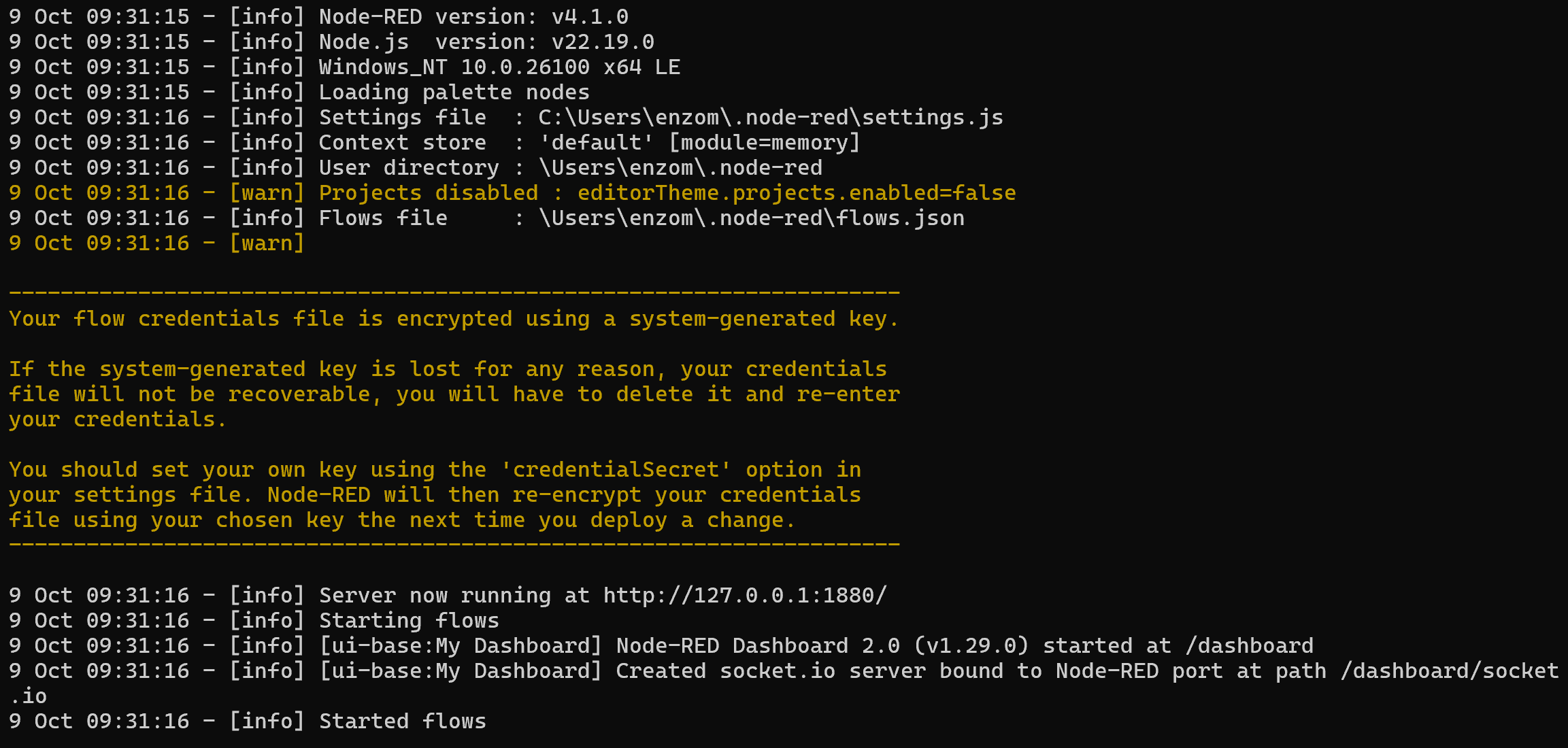
**Sous Linux** :

$sudo npm install -g --unsafe-perm node-red

# Démarrage et arrêt de Node-RED

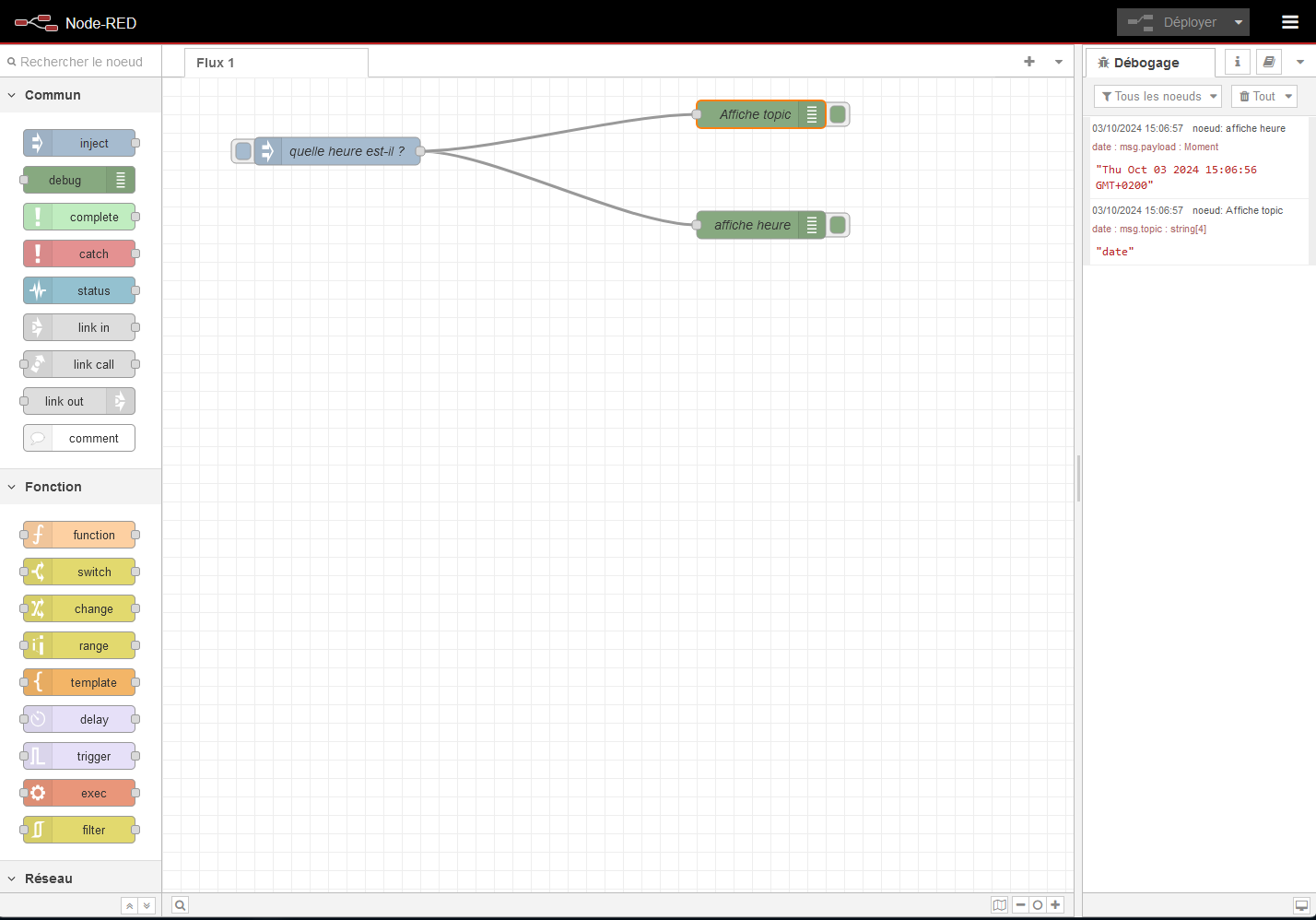
Démarrage de Node-RED

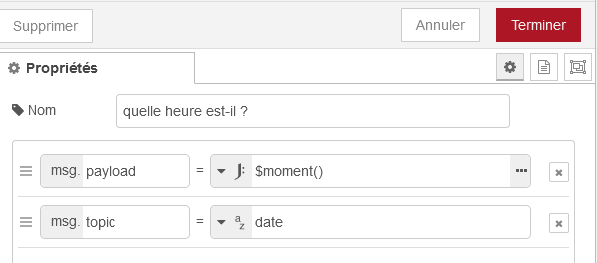
$ node-red

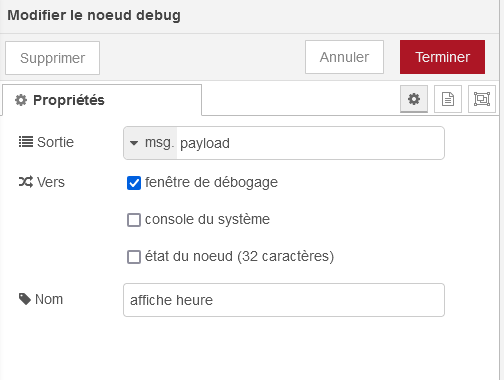


Comme indiqué dans le terminal on accède à l’interface de développement par un navigateur en utilisant

l’URL : <http://127.0.0.1:1880>







Arrêt de Node-RED

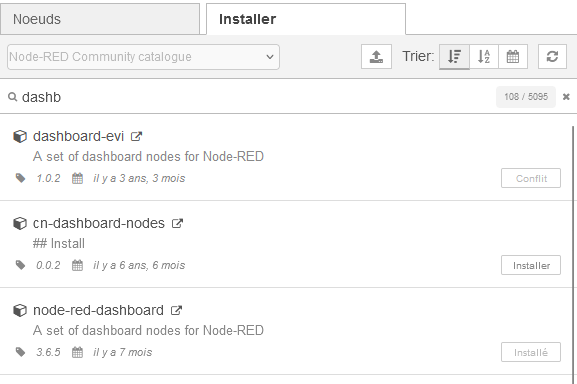
Pour arrêter node-red, utilisez la combinaison de touche [Ctrl] + [C]

# Ajout de nouveaux nœuds dans la palette

Nous allons ajouter des catégories de nœuds permettant de réaliser un dashboard.

Ajoutez la catégorie correspondante :

Dans le menu,  cliquez sur *Gérer la Palette* -> *Installer* -> dans la barre de recherche recherchez *dashb et choisissez node-red-dashboard*



# Programmer votre première application node-red

La programmation consiste à glisser-déplacer les nodes sur le flow et à les relier entre eux. Les

données échangées sont des objets JSON et doivent posséder la propriete payload (charge utile).

Le payload peut être un nombre, une chaine de caractères ou un objet JSON.

Certains nodes peuvent avoir plusieurs sorties. Dans ce cas, le node retourne un tableau d’objets

JSON dont le premier élément sort par la premiere sortie, le deuxième par la seconde, …

Commençons par réaliser un flow très simple composé de :

* un Slider
* une Gauge.

Pour modifier la configuration par défaut des nodes, il suffit de double-cliquer dessus.

Pour que ces composants soient bien disposés sur la page web, il faut les mettre dans un Tab, puis

dans un Groupe. Donc pour chacun de ces nœuds :

* Dans la boite de sélection du groupe, cliquez sur *Add new ui\_group*,
* Créer un groupe (ici : "**Le groupe**") et faite de même pour le Tab (ici : "**Test**").
* Cliquez ensuite sur le bouton Done, puis

Déployer le flow en cliquant sur le bouton Déployer.

On accède au site web génère par l’url : <http://127.0.0.1:1880/ui>

Vous devriez obtenir ceci

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Pour conserver un historique des valeurs affichées, on peut utiliser un node Chart :

Nous allons maintenant rajouter une information textuelle en bas du groupe. Cette information contiendra la valeur du slider ainsi que sa date et heure de production.

* Ajoutez un node Text qui permet d’ajouter un label : "Valeur = " et le contenu d’un message value format : "{{msg.payload}}"
* Intercalez un node Function pour insérer la date dans le message à afficher. Le code de la fonction date() est :

var now = new Date();

var year = now.getFullYear();

var month = now.getMonth()+1;

var day = now.getDate();

var hour = now.getHours();

var minute = now.getMinutes();

var second = now.getSeconds();

msg.payload = day+"/"+

month+"/"+

year+" a "+

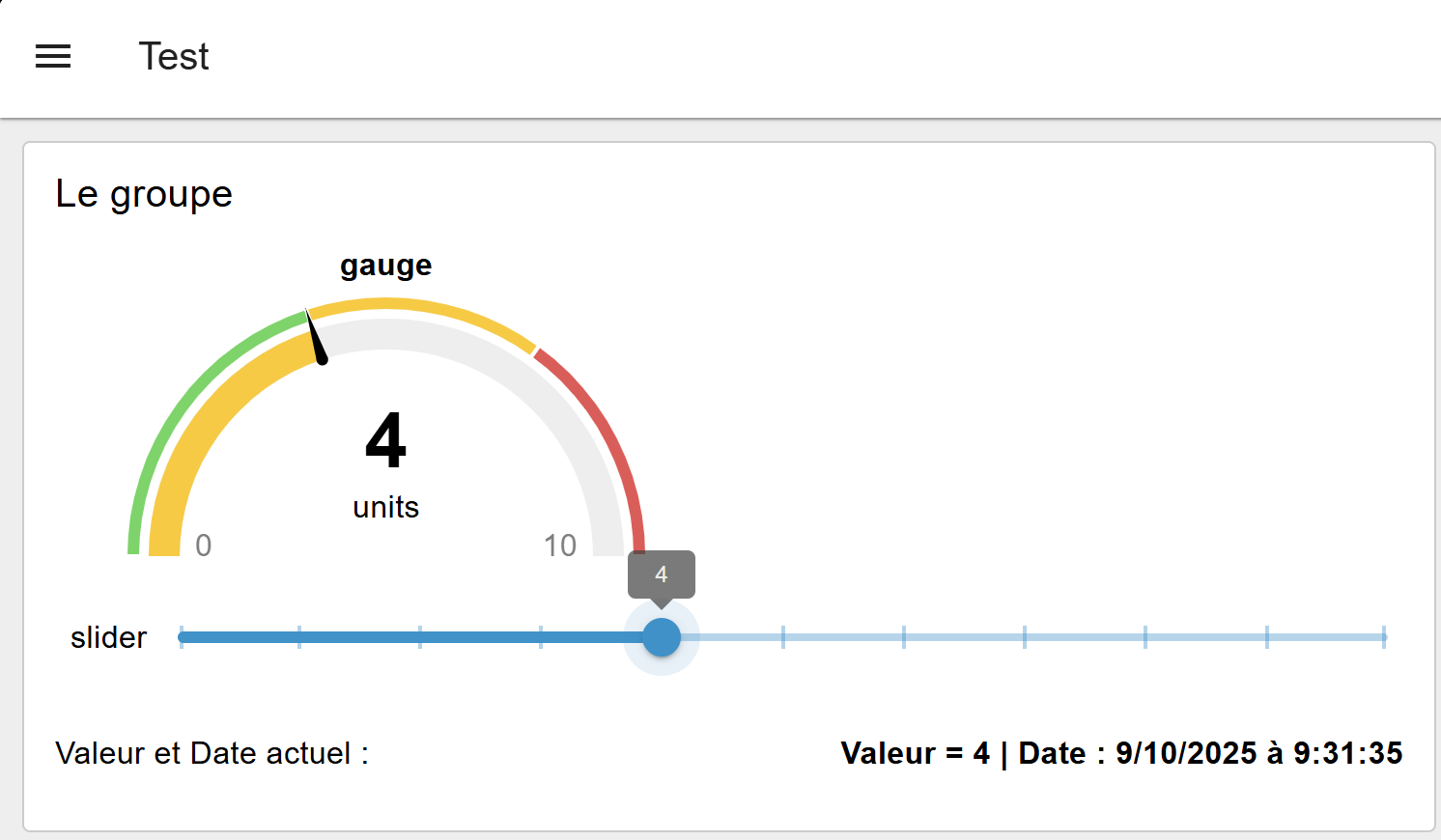
hour+":"+

second+" => "+

msg.payload;

return msg;

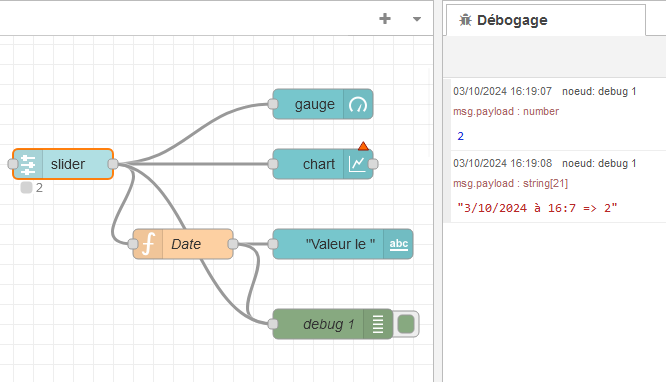
* Déployez observez le résultat.



|  |  |
| --- | --- |
|  |  |

* Debug d’une application node-red

Les messages transmis dans les flux d’entrée et de sortie des nodes sont des objets. C’est l’attribut payload, qui contient l’information à transmettre. Pour observer cette charge utile de message vous pouvez utiliser un node Debug.



# Réalisation d’une application de bout en bout en explorant des applications et capteurs publics

Nous voulons un objet connecté qui transmet à intervalles réguliers certaines valeurs de capteurs.

Nous prélevons des données produites par des capteurs sur test.mosquitto.org.

Ces derniers diffusent ces données via le protocole MQTT aux applications qui y souscrivent. Nous allons donc réaliser un flow qui se connecte à ce serveur afin d’obtenir les mesures et les afficher dans un dashboard puis les enregistrer dans un fichier cvs.

## Souscription à test.morsquitto.org MQTT

* Insérer et configurez des node mqtt in afin qu’ils écoutent le serveur *test.mosquitto.org.*
* Les nodes souscriront chacun à un des topics suivants :
  + Capteur luminosité sur le topic /merakimv/Q2GV-LNY7-QTMZ/light
  + Niveau de cuve sur topic *BTB\_Utilities/Water/Bulk\_Tank\_1/Level*
  + Niveau sonore sur topic */merakimv/Q2SV-FM73-WPJM/audio\_analytics*
* Déployez et observez les traces debug.

|  |  |
| --- | --- |
|  |  |

## Extraction des informations

Le payload du message reçu est une string. Pour accéder aux propriétés et aux valeurs qu’elle contient, nous allons la transformer en objet JSON en utilisant la classe javascript JSON et sa méthode parse().

NB vous pouvez aussi utiliser le node json

• Ajouter un node Function avec le code suivant :

msg.payload = JSON.parse(msg.payload);

return msg;

• Déployez et observez les traces debug.

• Créez les fonctions nécessaires pour accéder a :

* la luminosité : getLum()
* le niveau de la cuve : getTankLevel()
* le niveau audio : getAudioLevel()

Exemple : code de la fonction getLum()

temp= msg.payload.lux;

    msg.payload = temp;

    return msg;

## Horodatage des informations

* Pour les deux premiers sensors qui ne sont pas horodatés vous allez leur associer une date et une heure de collecte,

Exemple 2 : code de la fonction TimeStampée : TSTankLevel

var now = new Date();

var year = now.getFullYear();

var month = now.getMonth() + 1;

var day = now.getDate();

var hour = now.getHours();

var minute = now.getMinutes();

var second = now.getSeconds();

var msg1 ={};

var msg2={};

var date = day+"/"+month+"/"+year;

var heure = hour+":"+minute+":"+second;

var temp = {"date": date, "heure" : heure , "tankLevel": msg.payload };

msg1.payload = "Le " + temp.date + " à " + temp.heure + " tankLevel=" + temp.tankLevel

msg2.payload = temp;

return [msg1, msg2];

La fonction TSTankLevel() possède deux sorties. Ne pas oublier de configurer le node en conséquence. La sortie 1 fournie un objet qui contient la date et l’heure et le niveau. La sortie 2 fournie une chaine de caractères.

* Pour le capteur audioLevel il y a un timestamp associé à la création de la donnée et vous allez exploiter cette valeur de ts.

var msg1 ={};

var msg2={};

var temp = {"timeStamp": msg.payload.ts, "audioLevel": msg.payload.audioLevel };

msg1.payload = "Le " + temp.timeStamp + " audioLevel=" + temp.audioLevel

msg2.payload = temp;

return [msg1, msg2];

## Production du Dashboard

Pour le dashboard, nous allons utiliser deux nodes chart et une jauge pour tracer l’évolution des grandeurs mesurées. Nous utiliserons également un node text pour afficher la date et l’heure et de la mesure.

|  |  |
| --- | --- |
|  |  |

Une image contenant texte, capture d’écran, Parallèle, diagramme

Le contenu généré par l’IA peut être incorrect.

# Enregistrement des données

Il est possible d’enregistrer les données dans un **fichier texte CSV** ou dans une **base de données**. Il suffit d’utiliser le node approprié.

## Fichiers CSV

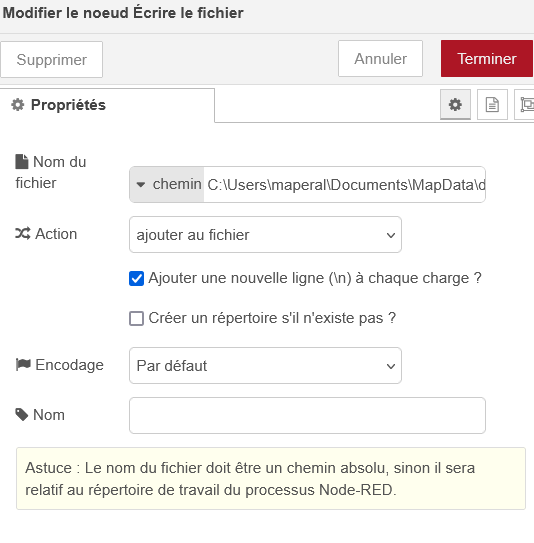
Le node csv permet de formater une string en csv à partir d’un objet JSON :

* Glissez le node csv sur le flow et connectez le aux fonction TS\*().

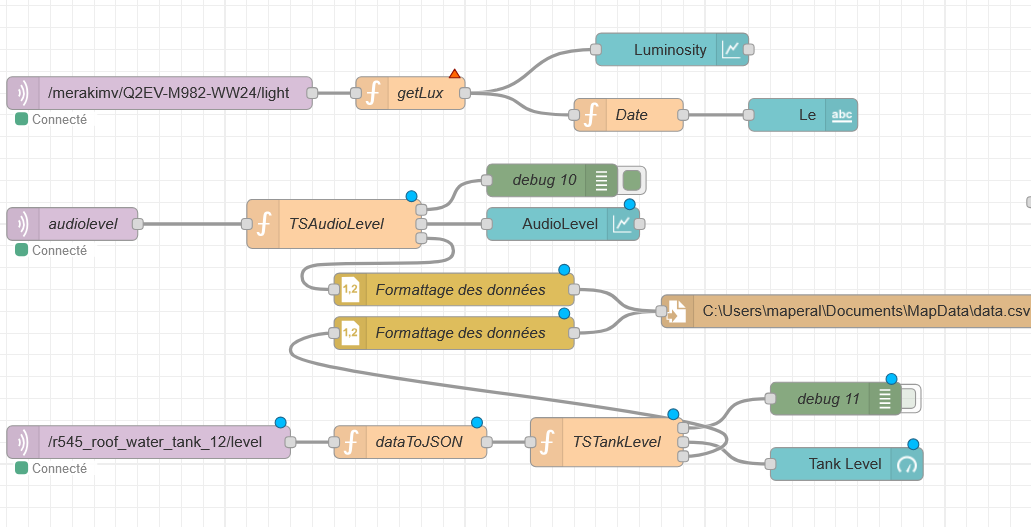
• Configurez le node csv comme suit :

|  |  |  |
| --- | --- | --- |
|  |  |  |

* Glissez un node fichier et configurez-le comme suit :



L’ensemble donnera cela :



* A ce schéma vous allez ajouter le stockage de la lumière également

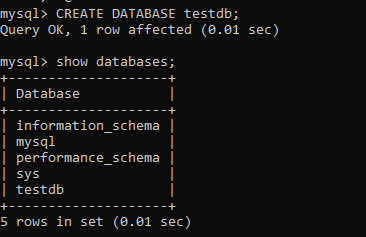
Une image contenant texte, capture d’écran, logiciel, Police

Le contenu généré par l’IA peut être incorrect.

## Base de donnees MySQL

Un node de gestion du classique SGBD MySQL est disponible.

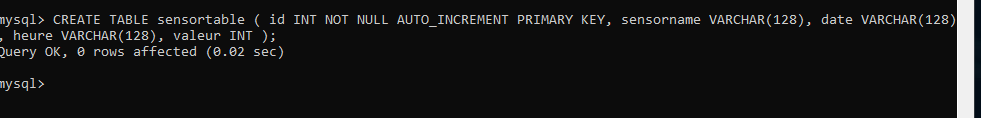
* Installez et démarrer MySQL avec une base de données accessible : <https://dev.mysql.com/downloads/mysql/#downloads>
* Configurez la base de données : (Compte root :root et user map : mapmap pour mon installation)
* Lancement mysql : Mysql



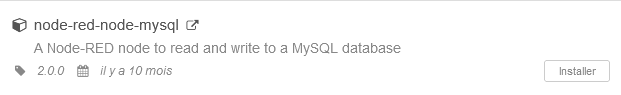


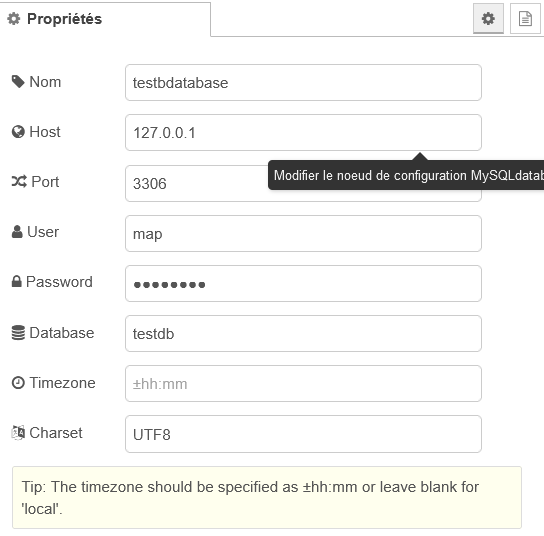


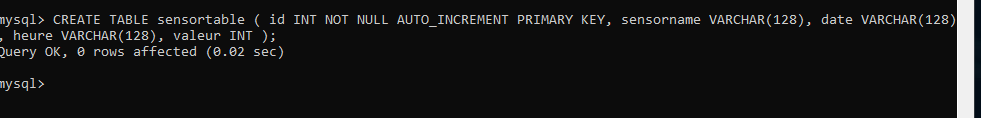
* Créer une table *sensortable* :



* Installer et connecter le node mysql à cette database:







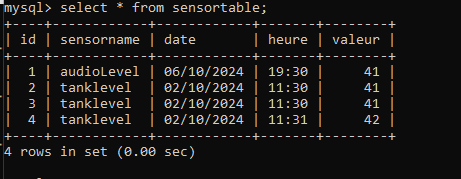
* Exemple de code de la fonction query

var query = "INSERT INTO sensortable VALUES (NULL, 'tankLevel', '" + msg.payload.date + "', '" + msg.payload.heure + "'," + msg.payload.tankLevel + ");";

msg.topic=query;

return msg;

* Vérifiez le stockage dans la base de données



* A ce schéma vous allez ajouter le stockage de la lumière également

Copie d’écran du fichier des codes des querys et copie d’écran de la table qui stocke les données des trois capteurs.