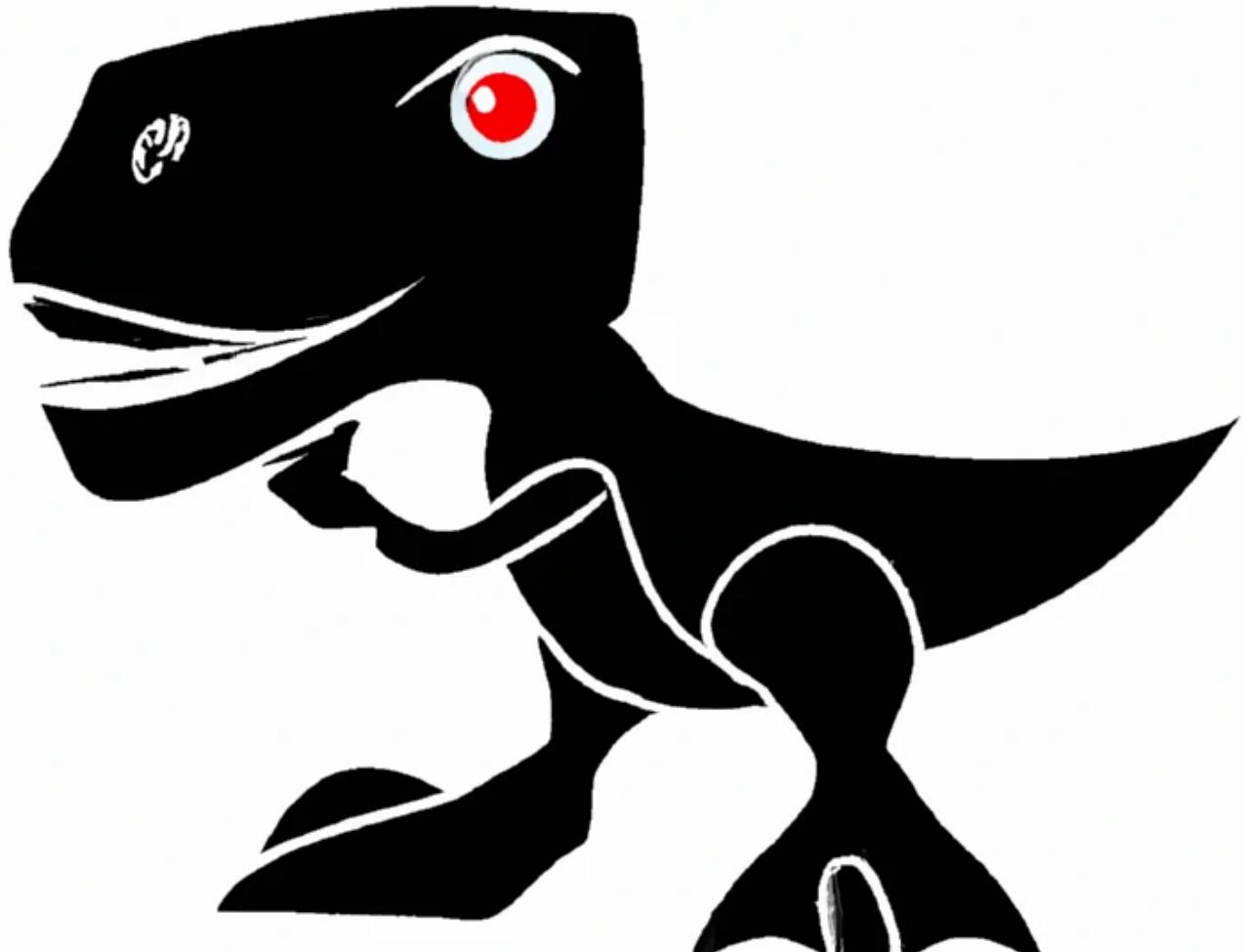


1. Rapport de projet : Banc de mesures de la consommation électrique pour application web



Candidat: Cyril Tobler

Proposé par: SINABE Sarl

Personne de contact: Benoit Vianin

Enseignant-e: Fabien Maire

Lieu de travail: Ecole CPNE-TI SIS2

Nombre de périodes: 300

Durée du travail : 14.08.23 - 22.09.23

2. Sommaire

- 1. Rapport de projet : Banc de mesures de la consommation électrique pour application web
- 2. Sommaire
- 3. Introduction
- 4. Objectifs
 - 4.1. Objectif principal
 - 4.2. Objectifs spécifiques
 - 4.2.1. Génération de trafic web
 - 4.2.2. Mesure de la consommation
 - 4.2.3. Génération de rapports sur la performance
- 5. Caractéristiques
 - 5.1. Génération de trafic web
 - 5.2. Mesure de la consommation électrique (INA219)
 - 5.3. Mesure de la consommation
- 6. Matériel
 - 6.1. Nomenclature
 - 6.2. Volt
 - 6.3. Nidus
 - 6.4. INA219
 - 6.5. RJ45
- 7. Budget
- 8. Planification
- 9. Instalation physique
 - 9.1. Nidus
 - 9.1.1. TODO Capture quand installation definitive
 - 9.2. Volt
 - 9.2.1. TODO Capture quand installation definitive
- 10. Schéma de principe
- 11. Systèmes d'exploitation (OS)
 - 11.1. Ubuntu
 - 11.2. Raspbian
 - 11.3. Première installation
 - 11.4. Seconde instalation Ubuntu Server
 - 11.4.1. Configuration post instalation
 - 11.4.2. Instalation Apache
 - 11.4.3. Script MQTT
- 12. Node-RED
 - 12.1. Instalation
 - 12.2. Configuration
 - 12.2.1. Installation des plugins
 - 12.2.2. Sécurisation de Node-Red
 - 12.2.3. Suivi Git
- 13. Gatling
 - 13.1. Installation
 - 13.1.1. Prerequis
 - 13.1.2. Download
 - 13.2. Vérification de l'installation
 - 13.3. Scripts
- 14. Apache et Site Web
 - 14.1. Installation
 - 14.2. Mise en place d'un site Web
- 15. MQTT
 - 15.1. Installation de Mosquitto sur Nidus
 - 15.2. Ouverture des port sur Nidus
 - 15.3. Script MQTT
 - 15.3.1. Script
 - 15.3.2. Description détaillée du script
 - 15.3.3. Conclusion
 - 15.4. Installation
 - 15.5. Utilisation du script
 - 15.5.1. Vérification
- 16. INA219
 - 16.0.1. Installation physique
 - 16.0.1.1. Branchement SANS VOLT

- 16.0.1.2. Branchement AVEC VOLT
- 16.0.2. Vérification de la présence du INA219
- 16.1. Obtention des données
 - 16.1.1. Test avec le script python A vide
- 17. Noeud Node-Red
 - 17.1. Dashboard
 - 17.2. INA219
 - 17.3. Monitoring
 - 17.4. PDF
 - 17.4.1. Base
 - 17.5. Images de graphiques et de tableaux
- 18. Stress Test V1.0
 - 18.1. Écran d'Accueil
 - 18.2. En Exécution
 - 18.3. Résultat
 - 18.4. Purge
- 19. Gatling Test V2.0
 - 19.1. But
 - 19.2. Étapes à Atteindre
 - 19.3. Exécution d'un Test Préétabli sur Gatling depuis Node-Red
 - 19.4. Envoi de Commande avec une Durée
 - 19.5. Récupération des Informations
 - 19.6. Traitement des données
 - 19.6.1. Création des graphiques
 - 19.6.2. PDF
 - 19.7. Refactoring
- 20. Gatling V3.0
 - 20.1. UI
 - 20.1.1. Ventilateur
 - 20.1.2. UI
- 21. Remerciement
- 22. Sources

3. Introduction

Le développement de sites web et d'applications web requiert une attention particulière à la performance et à la consommation des ressources. Afin de réduire l'impact écologique des solutions digitale et d'optimiser la charge des systèmes, il est essentiel de mesurer et d'analyser la consommation de ces systèmes dans des conditions de charge réalistes. Dans ce contexte, il est nécessaire de mettre en place un banc de mesures de la consommation capable de générer du trafic web, de mesurer la consommation électrique et de fournir des rapports sur la performance du code ou de l'architecture testée.

4. Objectifs

4.1. Objectif principal

L'objectif principal de ce projet est de concevoir et développer un banc de mesures de la consommation électrique pour site web ou application web. Ce banc de mesures devra être capable de générer du trafic web, de mesurer la consommation des ressources et de générer des rapports détaillés sur la performance du code ou de l'architecture testée par rapport à une référence.

4.2. Objectifs spécifiques

Les objectifs spécifiques du projet sont les suivants :

4.2.1. Génération de trafic web

Le système devra être en mesure de simuler le trafic web en générant des requêtes HTTP réalistes. Il devra pouvoir reproduire des scénarios de charge variable afin d'évaluer les performances du site web ou de l'application web dans des conditions réelles.

4.2.2. Mesure de la consommation

Le banc de mesures devra être équipé d'un mécanisme de mesure précis et fiable de la consommation des ressources, telles que la consommation d'énergie, l'utilisation du processeur, la consommation de mémoire et la consommation du réseau.

4.2.3. Génération de rapports sur la performance

Le système devra être capable de collecter les données de mesure et de générer des rapports détaillés sur la performance du code ou de l'architecture testée. Les rapports devront inclure des métriques telles que le temps de réponse, la consommation d'énergie par requête, l'utilisation du processeur, etc.

5. Caractéristiques

Le banc de mesures de la consommation pour site web ou application web devra présenter les caractéristiques suivantes :

5.1. Génération de trafic web

Utilisation de l'outil Gatling comme générateur de trafic web. Gatling permettra de simuler des requêtes HTTP réalistes, de configurer des scénarios de charge et d'évaluer les performances du système testé.

Gatling est un outil de test de charge open source basé sur Scala, conçu pour évaluer les performances des applications et des sites Web. Gatling simule des utilisateurs virtuels qui envoient des requêtes HTTP vers le système cible.

5.2. Mesure de la consommation électrique (INA219)

Utilisation d'un chip INA219 ou autre connectée via le bus I2C pour mesurer la consommation de manière précise et fiable. La chip INA219 fournira des informations détaillées sur la consommation d'énergie en mesurant la tension et le courant du système testé.

5.3. Mesure de la consommation

Utilisation d'un serveur Node-Red pour son environnement low-code pour les applications évènementielles. Il facilitera le relever des mesures de consommation des ressources. du banc de tests (consommation d'énergie, utilisation du processeur, consommation de mémoire, bande passante et temps de réponses, ...).

6. Matériel

- 2x Radiateur pour Raspberry Pi 4
- 2x Raspberry Pi 4 /4GB RAM / 64GB SD
- 2x Bloc d'alimentation Raspberry Pi 4
- 2x Carte Micro SD 64GB
- 2x cable RJ45 violet
- 1x cable Micro HDMI - HDMI
- 1x Plaque d'essai
- 1x set de câbles de connexion
- 2x Platine de mesure INA219
- 2x câble USB-C Femelle
- 2x câble USB-C Mâle

6.1. Nomenclature

Pour simplifier la lecture du rapport ainsi que le travail, les Raspberry Pi seront nommés comme suit :

- Volt : Serveur Web
- Nidus : Serveur de monitoring

Au niveau des hostname, les Raspberry Pi seront nommés comme suit :

- Volt : volt.s2.rpn.ch
- Nidus : nidus.s2.rpn.ch

En plus des raspberry il y a ma machine de développement qui se trouve être mon laptop personnel qui tourne sous un Kubuntu 22.04. Cette machine est nommé comme suit :

- LPT-UNIX-USB-CT

6.2. Volt

Au niveau de la répartition du matériel, Volt est composé de :

1. Cable HDMI - Micro HDMI
2. Raspberry PI 4
3. Bloc d'alimentation pour Raspberry PI 4
4. Carte Micro SD 64GB
5. Radiateur pour Raspberry Pi 4
6. Set de vis et de gomme pour le radiateur



6.3. Nidus

Le matériel de Nidus est composé de :

1. Set de câble de connexion
2. Plaque d'essai
3. Alimentation pour Raspberry Pi 4
4. Rallonge USB-C coupé en deux
5. INA219
6. Raspberry Pi 4 et sa carte Micro SD 64GB
7. Radiateur pour Raspberry Pi 4
8. Set de vis et de gomme pour le radiateur



6.4. INA219

Il s'agit d'une seconde puce INA219 qui a été commandé pour le projet, pour pouvoir changer la puce INA219 en cas de problème. Au cas ou le projet prend de l'ampleur, il est aussi possible à terme de mettre plusieurs INA219 sur le même bus I2C.



6.5. RJ45

En réserve, deux câble RJ45 violet ont été commandé pour le projet.



7. Budget

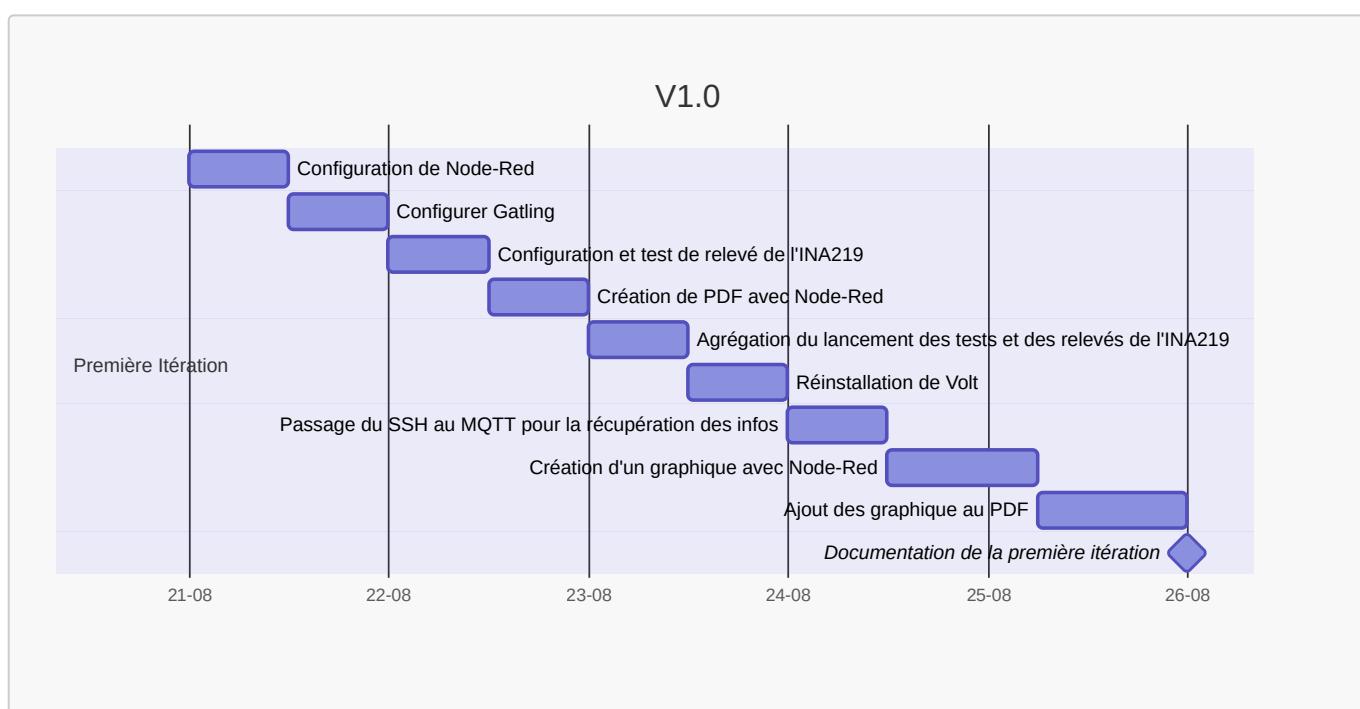
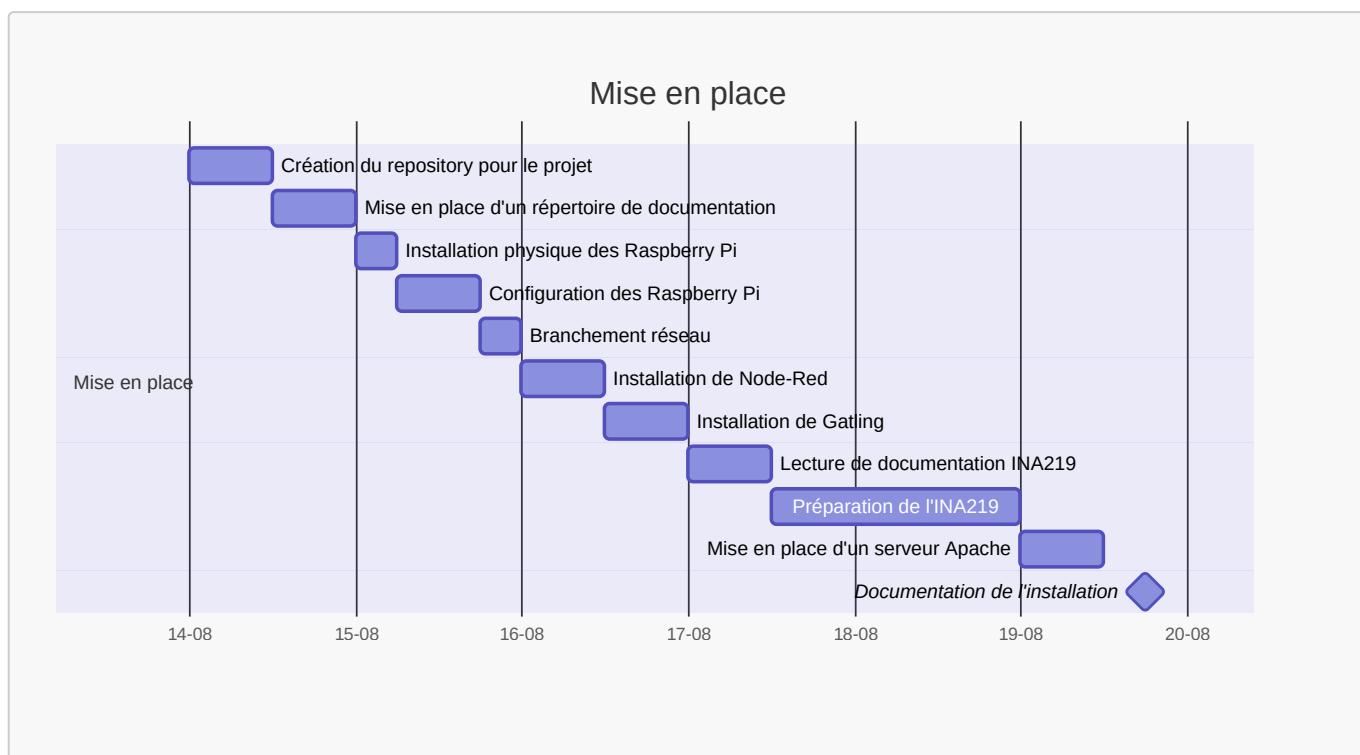
En ce qui concerne le budget, je vais fournir les prix de Digitec/Galaxus pour donner une idée des coûts si l'on souhaite reproduire le projet. Dans mon cas, c'est M. Viannin qui a effectué les achats auprès de ces fournisseurs. Il est possible que les prix ne correspondent pas exactement à ceux que j'ai indiqués.

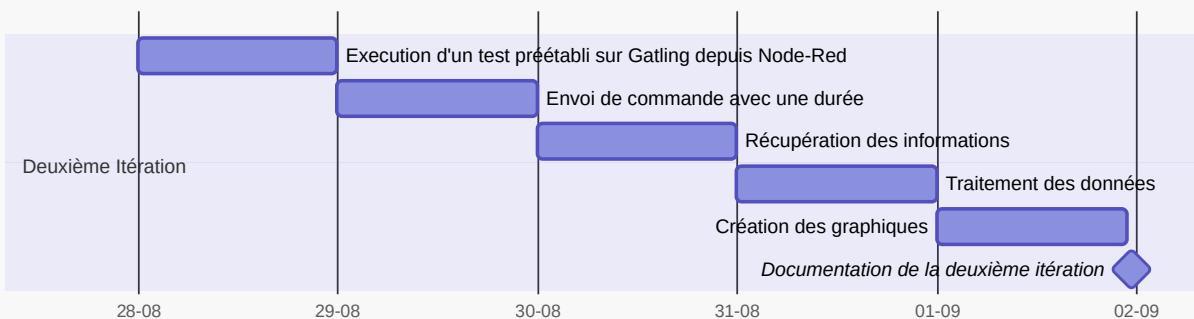
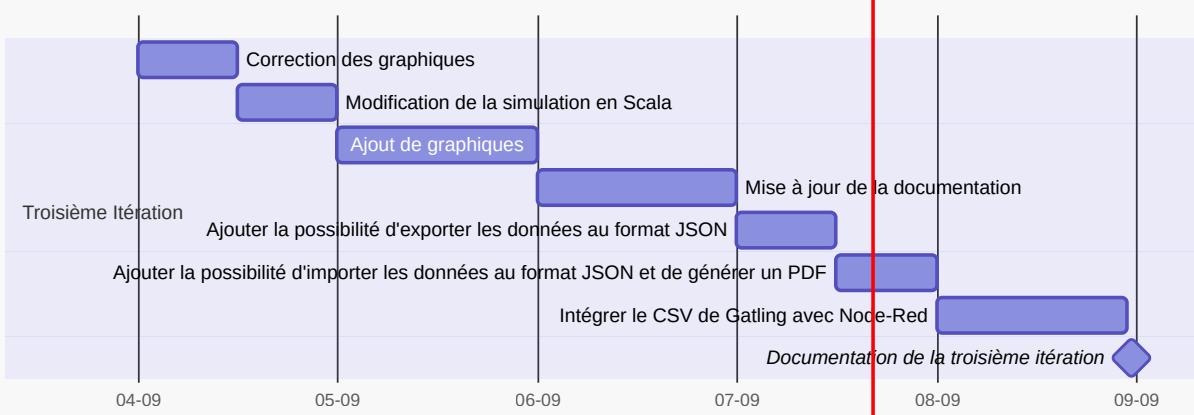
Matériel	Quantité	Prix Par Unité	Prix Total
Raspberry Pi Câble officiel blanc Micro-HDMI vers HDMI 1M pour Raspberry Pi 4	1X	12.90	12.90
RASPBERRY Radiateur	2X	25.80	51.60
Raspberry Pi 4 4G Modèle B ARMv8	2X	87.-	174.-
Official Raspberry Pi 4 Power Adapter	2X	13.90	27.80
microSDXC, 64 Go, U3, UHS-I	2X	14.90	29.80
Carte enfichable à grille de trous Breadboard ZY-60	1X	18.90	18.90
INA219	2X	17.90	35.80
MikroElektronika Fils de connexion	1X	13.-	13.-
Cable RJ45 violet court	2X	9.75	19.50
Cable USB-C Mâle-Femelle	2X	14.95	29.90
Total			433.20

Une fois le matériel pris en compte, je tiens également à calculer le coût horaire de travail. Pour cela, je prends le salaire d'un technicien en informatique, soit 30.- de l'heure. Je prends également en compte le temps de travail, soit 300 périodes de 45 minutes, ce qui équivaut à 225 heures. Cela nous donne un total de 6750.- de salaire. En ajoutant le coût du matériel, nous arrivons à un total de 7183.- .

Bien sûr, dans le cadre de ce projet, le salaire n'est pas pris en compte, mais il est important de le considérer si l'on souhaite reproduire le projet dans un contexte professionnel.

8. Planification



V2.0**V3.0**

9. Instalation physique

9.1. Nidus

9.1.1. TODO Capture quand installation definitive

9.2. Volt

9.2.1. TODO Capture quand installation definitive

10. Shéma de principe

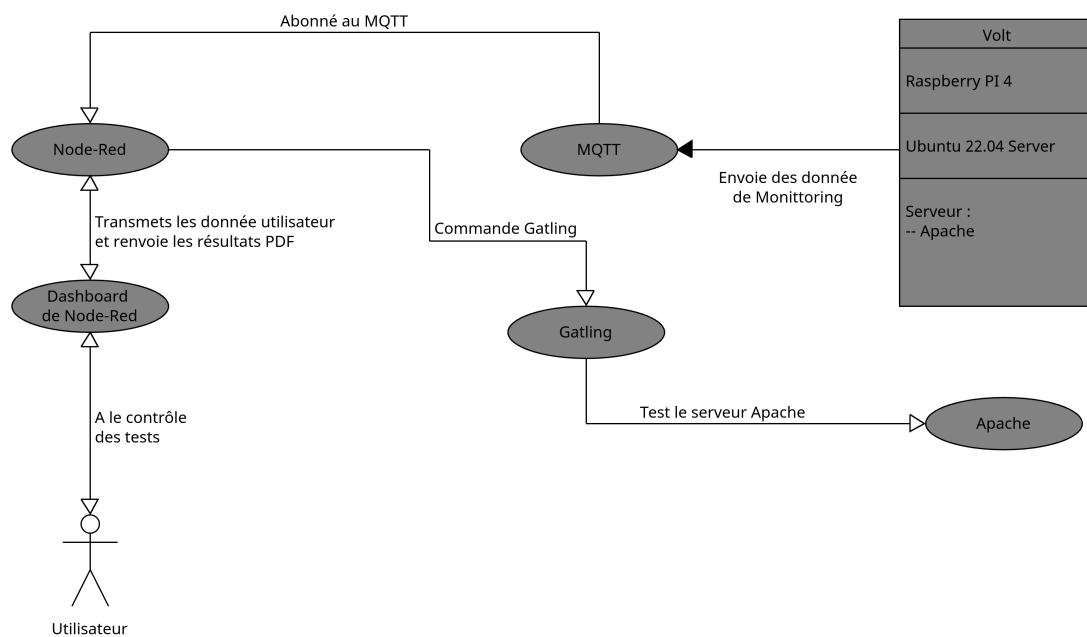
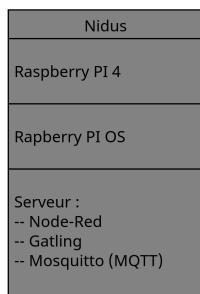
Explication : Actuellement le but est que Nidus offre tout les outil pour le monittring incluant le MQTT, Node-Red, Gatling et l'INA219. Volt lui ne sert que de serveur web pour le site web. Le but est de pouvoir faire des test de charge sur le site web et de pouvoir mesurer la consommation électrique du serveur web.

De fait toute interaction de l'utilisateur se fait avec Nidus.

Nidus envoie ses donnée de monittring sur le serveur MQTT installé sur Nidus, et Node-Red installé sur Nidus récupère les données du serveur MQTT et les envoie dans des noeud fais pour le traiter et fournir ensuite les sortie approprié :

- Dashboard : Pour l'utilisateur
- PDF : Pour l'utilisateur

Nidus peut dans un second temps lancer des stresstest via Node-Red sur lui même et sur Volt. Il peut aussi lancer des stresstest sur Volt via Gatling.



11. Systèmes d'exploitation (OS)

Dans le cadre de ce projet, plusieurs systèmes d'exploitation seront utilisés. Pour commencer, nous utiliserons Ubuntu.

11.1. Ubuntu

Ubuntu est un système d'exploitation largement utilisé pour les serveurs et les ordinateurs de bureau. Il est livré avec un ensemble d'outils de développement et de productivité, notamment un navigateur Web, un éditeur de texte, des logiciels de programmation, des outils de calcul, des jeux et des logiciels de productivité. Ubuntu propose un environnement de bureau léger et réactif, conçu tant pour les ordinateurs de bureau que pour les serveurs.

11.2. Raspbian

Raspbian est un système d'exploitation libre basé sur Debian, spécialement optimisé pour le Raspberry Pi. Depuis 2015, Raspbian est livré avec un ensemble d'outils appelé Pixel. Pixel offre un environnement de bureau comprenant un navigateur Web, un éditeur de texte, des logiciels de programmation, des outils de calcul, des jeux et des logiciels de productivité. Pixel est un environnement de bureau léger et réactif, conçu spécifiquement pour les ordinateurs monocarte Raspberry Pi.

11.3. Première installation

Dans un premier temps, nous allons installer la version bureau d'Ubuntu sur Volt. Cette décision est motivée par le fait qu'il est plus simple de travailler dans un environnement de bureau pour tester rapidement tous les concepts du projet.

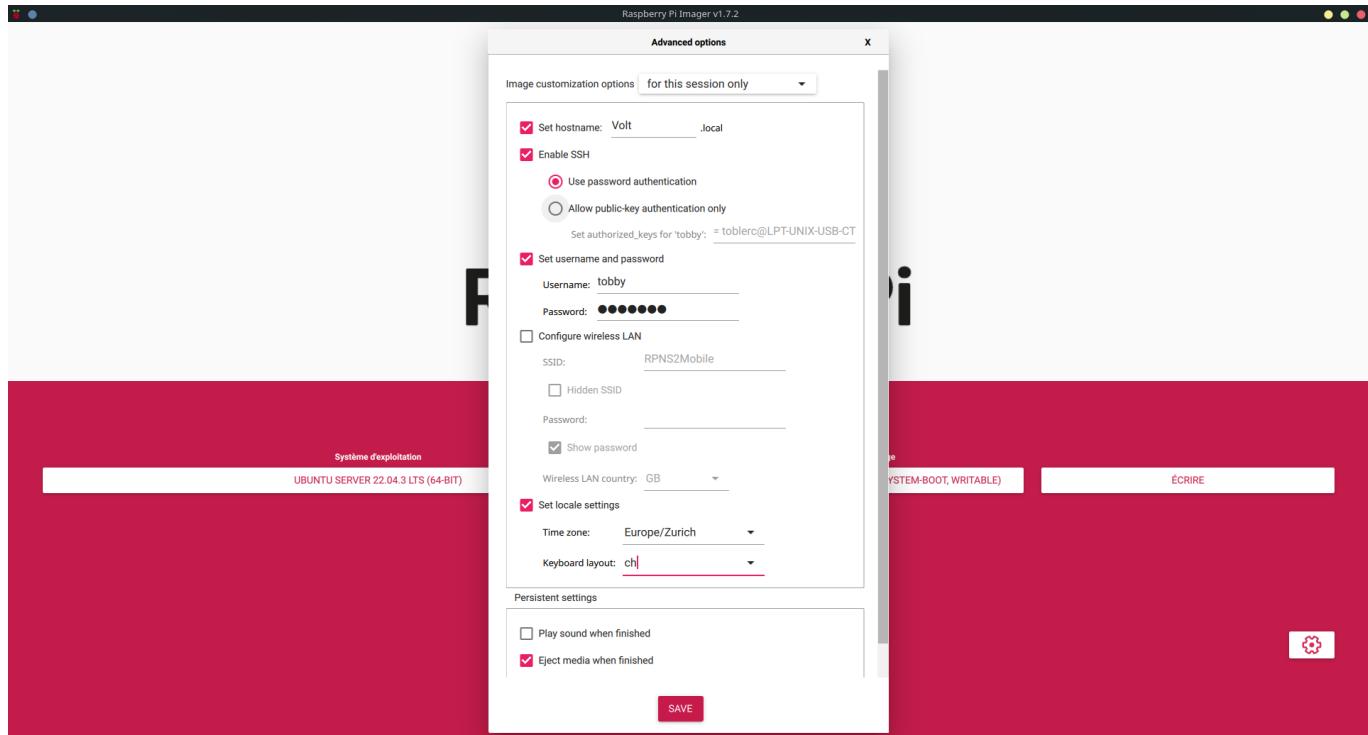
Sur Nidus, Raspbian en version bureau sera installé pour des raisons similaires à celles de Volt.

Un élément crucial à noter est que, étant donné que l'INA219 sera connecté à Nidus, il est plus pratique d'installer Raspbian sur Nidus afin d'avoir accès aux broches GPIO.

Dans un second temps, pour obtenir des mesures plus précises, nous installerons les versions « core » d'Ubuntu et de Raspbian.

- Adresse IP de Volt : 157.26.228.77
- Adresse IP de Nidus : 157.26.251.185

11.4. Seconde instalation Ubuntu Server



11.4.1. Configuration post instalation

- Première connexion en SSH

```
toblerc@LPT-UNIX-USB-CT:~$ ssh toddy@157.26.228.77
The authenticity of host '157.26.228.77 (157.26.228.77)' can't be established.
ED25519 key fingerprint is SHA256:/5raLlKqk0A4AnFwNLP9bagNS3zKE9rFPqn5vA5pc+M.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '157.26.228.77' (ED25519) to the list of known hosts.
tobdy@157.26.228.77's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-1034-raspi aarch64)

[...]
```

- Copie des clé SSH pour permettre la connexion sans mot de passe

```
tobdy@Volt:~$ cd .ssh/
tobdy@Volt:~/ssh$ sudo vi authorized_keys
[sudo] password for toddy:
```

- Vérification en se déconnectant et en se reconnectant

```
tobdy@Volt:~/ssh$ exit
logout
Connection to 157.26.228.77 closed.
toblerc@LPT-UNIX-USB-CT:~$ ssh toddy@157.26.228.77
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-1034-raspi aarch64)
[...]
Last login: Wed Aug 23 09:30:02 2023 from 157.26.215.31
```

- Mise à jour du système

```
tobdy@Volt:~$ sudo apt update && sudo apt upgrade -y && sudo apt dist-upgrade -y
&& sudo apt autzo-remove -y
[sudo] password for toddy:
[...]
```

- Vérification

```
tobdy@Volt:~$ sudo apt update && sudo apt upgrade -y && sudo apt dist-upgrade -y
&& sudo apt auto-remove -y
[...]
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

11.4.2. Instalation Apache

- Instalation de Apache

```
tobby@Volt:~$ sudo apt install apache2  
[...]
```

Copie du site web entre la machine de développement et Volt

```
toblerc@LPT-UNIX-USB-CT:~/Documents/ES_2024/banc-de-mesures-de-la-consommation-electrique/siteWeb/www/html  
tobby@157.26.228.77://home/tobby  
[...]  
toblerc@LPT-UNIX-USB-CT:~$  
tobby@Volt:~$ sudo cp -r /home/tobby/html /var/www/
```

11.4.3. Script MQTT

- Copie du script MQTT depuis la machine de développement vers Volt

```
toblerc@LPT-UNIX-USB-CT:~/Documents/ES_2024/banc-de-mesures-de-la-consommation-electrique$ scp ./mqtt.sh toby@157.26.228.77:/home/tobby  
mqtt.sh  
100% 2522      1.7MB/s   00:00
```

- Copie du script MQTT dans le dossier /usr/local/bin/

```
tobby@Volt:~$ sudo cp ./mqtt.sh /usr/local/bin/
```

- Vérification

```
tobby@Volt:~$ ls -la /usr/local/bin/  
total 12  
drwxr-xr-x  2 root root 4096 Aug 23 10:26 .  
drwxr-xr-x 10 root root 4096 Aug  7 17:23 ..  
-rw-r--r--  1 root root 2522 Aug 23 10:26 mqtt.sh
```

- Execution du script

```
tobby@Volt:/usr/local/bin$ sudo ./mqtt.sh  
Installation de mosquitto-clients...  
Hit:1 http://ports.ubuntu.com/ubuntu-ports jammy InRelease  
Get:2 http://ports.ubuntu.com/ubuntu-ports jammy-updates InRelease [119 kB]  
Hit:3 http://ports.ubuntu.com/ubuntu-ports jammy-backports InRelease  
Get:4 http://ports.ubuntu.com/ubuntu-ports jammy-security InRelease [110 kB]  
Fetched 229 kB in 2s (133 kB/s)  
Reading package lists... Done  
Reading package lists... Done  
Building dependency tree... Done  
[...]
```

```
Activation du service...
Service activé.
```

12. Node-RED

Node-RED est un outil de programmation visuelle open source conçu pour faciliter la connexion de périphériques, d'API et de services en ligne. Il propose un éditeur de flux basé sur un navigateur, permettant ainsi de connecter des nœuds à l'aide de simples glisser-déposer. Ces nœuds peuvent être exécutés dans un environnement Node.js. Ils peuvent être des fonctions JavaScript ou des modules npm, tels que node-red-contrib-gpio, node-red-contrib-sqlite, node-red-contrib-modbustcp, etc. En plus des nœuds de base fournis, Node-RED offre plus de 2000 nœuds supplémentaires créés par la communauté et prêts à être utilisés.

12.1. Instalation

L'instalation de Node-Red se fait via le script officiel de Node-Red. Ce script permet d'installer Node-Red et Node.js, de configurer Node-Red et de l'activer comme service.

Qui plus est on peut définir des utilisateurs et des mots de passe pour sécuriser Node-Red.

```
tobby@Nidus:~ $ bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
Running Node-RED install for user tobyy at /home/tobby on debian
```

This can take 20-30 minutes on the slower Pi versions - please wait.

Stop Node-RED	✓
Remove old version of Node-RED	✓
Remove old version of Node.js	✓
Install Node.js 18 LTS	✓ v18.17.1 Npm 9.6.7
Clean npm cache	✓
Install Node-RED core	✓ 3.0.2
Move global nodes to local	-
Npm rebuild existing nodes	✓
Install extra Pi nodes	✓
Add shortcut commands	✓
Update systemd script	✓

[...]

Would you like to customise the settings now (y/N) ? y

```
Node-RED Settings File initialisation
=====
This tool will help you create a Node-RED settings file.
```

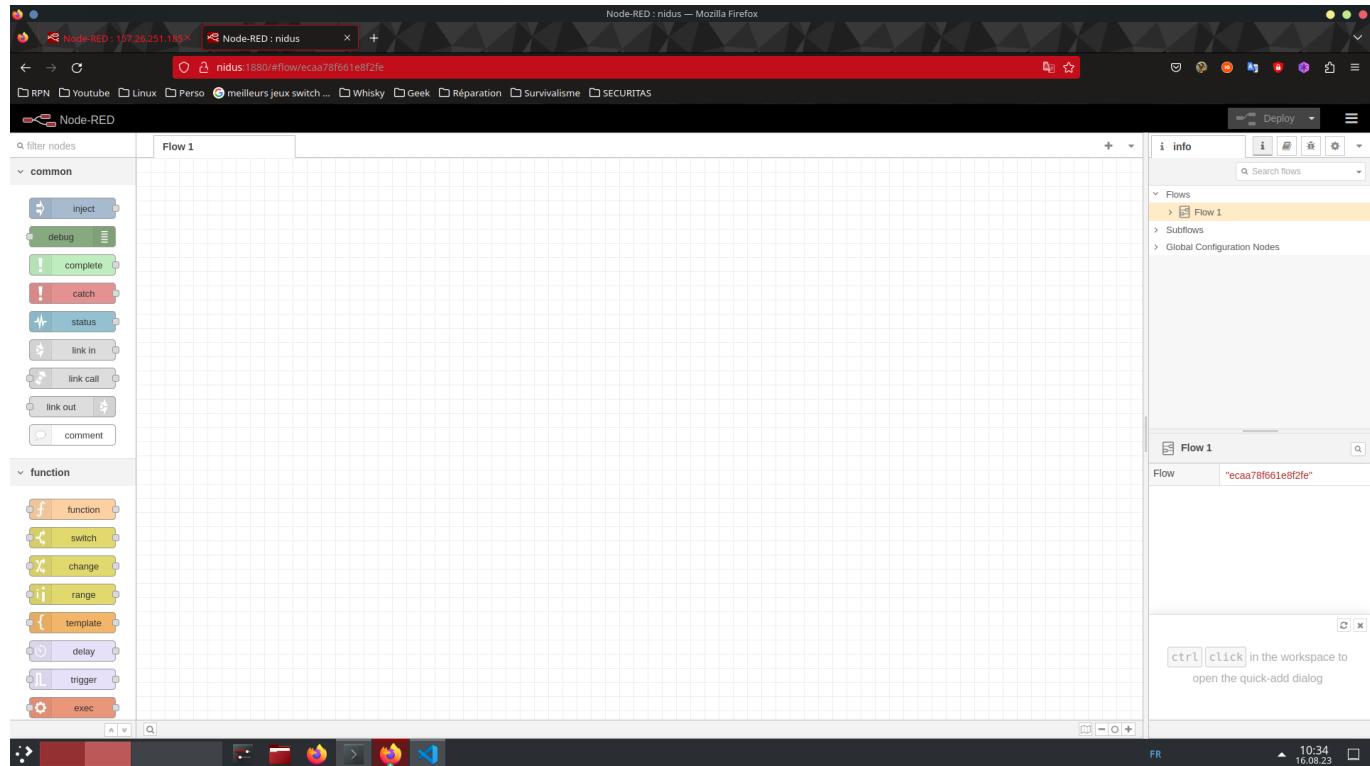
✓ Settings file · /home/tobby/.node-red/settings.js

```
User Security
=====
✓ Do you want to setup user security? · Yes
✓ Username · Tobby
✓ Password · *****
✓ User permissions · full access
✓ Add another user? · Yes
✓ Username · FMA
✓ Password · ***** (Pa$$w.rd)
✓ User permissions · read-only access
✓ Add another user? · Yes
✓ Username · BVI
✓ Password · ***** (Pa$$w.rd)
✓ User permissions · read-only access
```

```
✓ Add another user? · No  
[ . . . ]
```

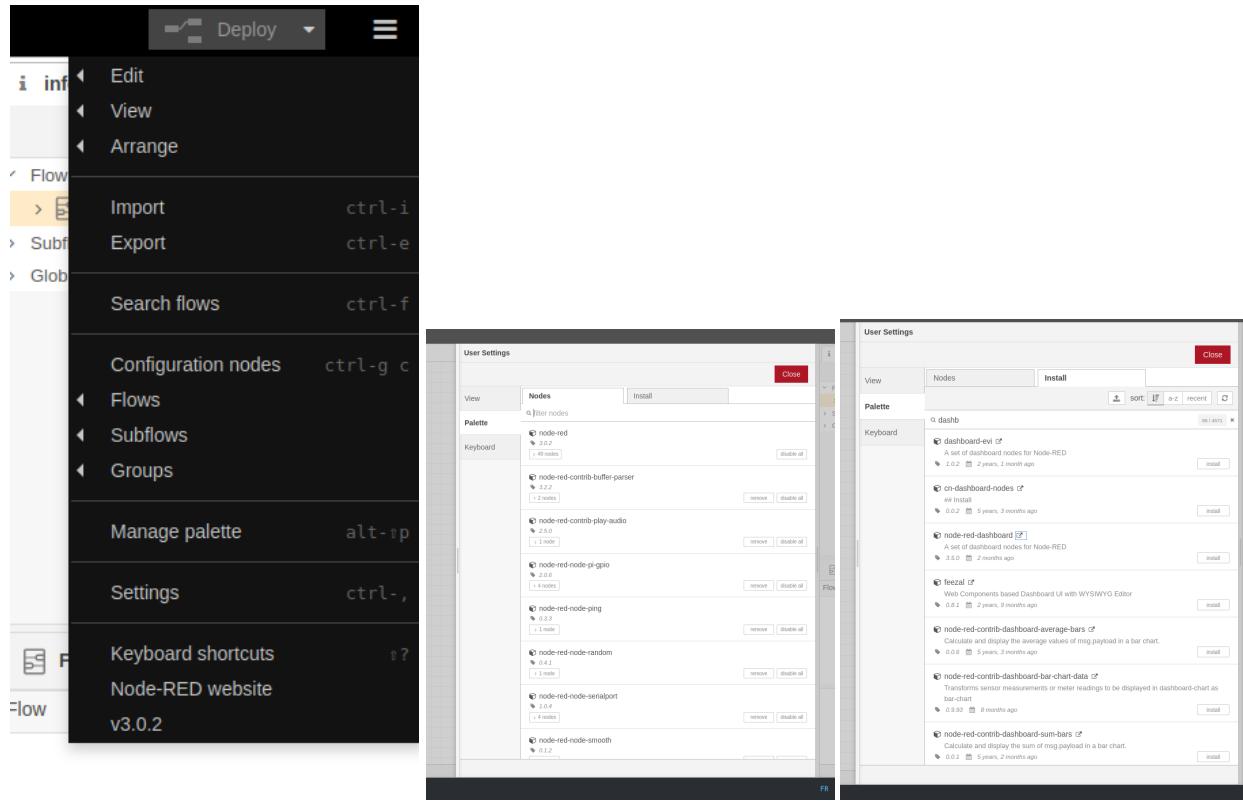
- Activation de Node-Red comme service

```
tobby@Nidus:~ $ sudo systemctl enable nodered.service  
Created symlink /etc/systemd/system/multi-user.target.wants/nodered.service →  
/lib/systemd/system/nodered.service.
```



12.2. Configuration

12.2.1. Installation des plugins



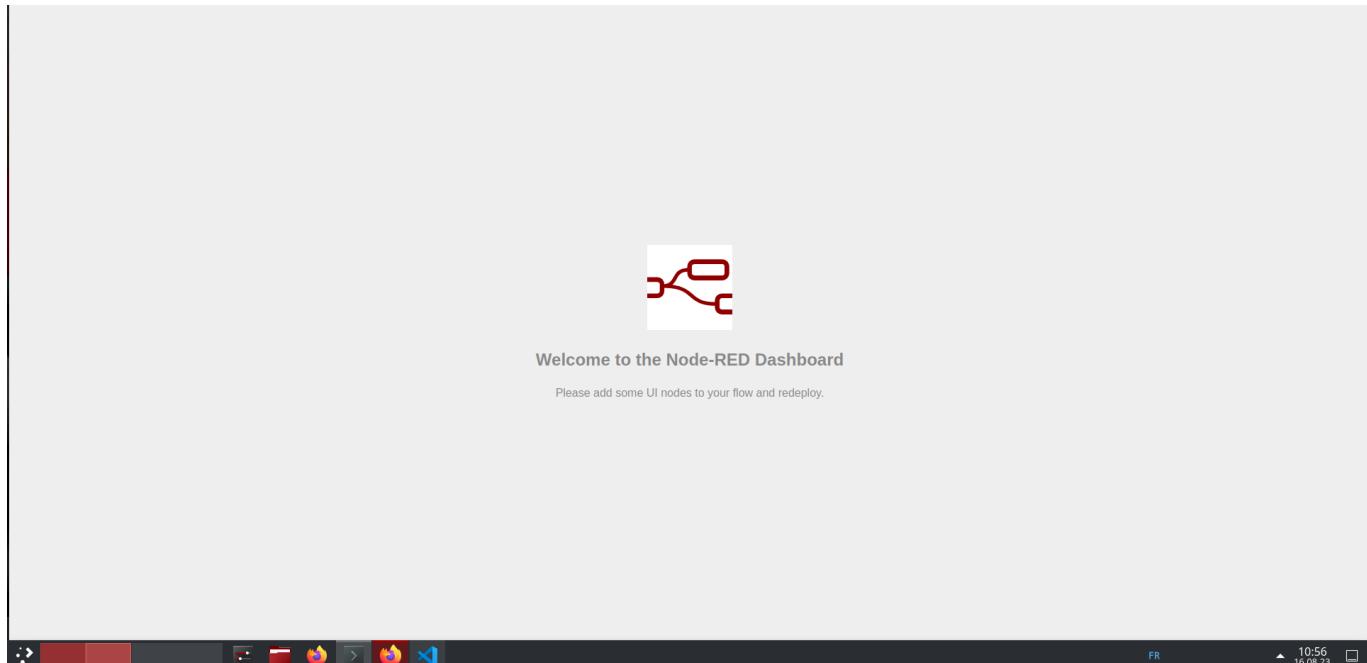
Installing 'node-red-dashboard'

Before installing, please read the node's documentation. Some nodes have dependencies that cannot be automatically resolved and can require a restart of Node-RED.

Cancel

Open node information

Install



12.2.2. Sécurisation de Node-Red

Pour sécuriser Node-Red, il convient de modifier le fichier `settings.js`. Dans notre cas, nous utilisons la commande `node-red admin init`, ce qui permet, par exemple, de créer des paires utilisateur/mot de passe.

De plus, il est recommandé, si nécessaire, d'ajouter un login au *Dashboard*. #TODO

12.2.3. Suivi Git

Afin de suivre le projet sur Git, il est nécessaire de configurer un utilisateur, générer des clés SSH, puis effectuer un *clone* du projet.

The three screenshots illustrate the configuration steps for Git integration:

- User Settings:** Shows the "Git config" section where "Workflow" is set to "Manual". It also displays an SSH key named "Node-Red" under "SSH Keys".
- Projects:** Shows the "Clone Repository" button highlighted. It includes fields for "Project name" and "Git repository URL" (set to `https://git.example.com/path/my-project.git`).
- Project Settings:** Shows the "Files" tab with "package.json" selected. It also displays "Dependencies" (flows.json), "Settings" (flows_cred.json), and "Version Control" (with a single branch named "main").

Comme il s'agit d'un *clone*, il faudra ajouter les fichiers manquants et ajuster les droits d'accès.

```
tobby@Nidus:~/.node-red/projects/banc-de-mesures-de-la-consommation-electrique $ touch ~/.node-red/projects/banc-de-mesures-de-la-consommation-electrique/flows_cred.json
tobby@Nidus:~/.node-red/projects/banc-de-mesures-de-la-consommation-electrique $ chmod 600 ~/.node-red/projects/banc-de-mesures-de-la-consommation-electrique/flows_cred.json
```

13. Gatling

Gatling est un outil de test de charge open source basé sur Scala, conçu pour évaluer les performances des applications et des sites Web. Gatling simule des utilisateurs virtuels qui envoient des requêtes HTTP vers le système cible. Il enregistre les temps de réponse des requêtes et les présente sous forme de graphiques. Gatling est doté d'un éditeur de scénarios basé sur navigateur, permettant aux utilisateurs de créer des scénarios de test de charge à l'aide d'un langage de domaine spécifique (DSL) appelé *Gatling DSL*. Ce langage, basé sur Scala, permet de définir des scénarios de test de charge à l'aide de mots-clés tels que `exec`, `pause`, `feed`, etc.

La version la plus récente de Gatling est la 3.9.5, compatible avec Java 8 et Java 11. Dans ce projet, nous opterons pour Java 11 pour exécuter Gatling.

13.1. Installation

13.1.1. Prerequisites

- Installation de Java 11

```
tobby@Nidus:~ $ sudo apt install default-jdk
```

- Vérification de la version de Java

```
tobby@Nidus:~/.node-red $ java -version
openjdk version "11.0.18" 2023-01-17
OpenJDK Runtime Environment (build 11.0.18+10-post-Debian-1deb11u1)
OpenJDK 64-Bit Server VM (build 11.0.18+10-post-Debian-1deb11u1, mixed mode)
```

13.1.2. Download

- Création du répertoire d'installation

```
tobby@Nidus:~ $ mkdir .gatling
```

- Vérification de la création du répertoire

```
tobby@Nidus:~ $ ls -la
total 104
drwxr-xr-x 18 toby toby 4096 16 aoû 15:10 .
drwxr-xr-x  3 root  root 4096 16 aoû 13:58 ..
[...]
drwxr-xr-x  2 toby toby 4096 16 aoû 15:10 .gatling
[...]
```

- Téléchargement de Gatling

```
tobby@Nidus:~ $ wget -O ~/.gatling/gatling-charts-highcharts-bundle-3.9.5-
bundle.zip https://repo1.maven.org/maven2/io/gatling/highcharts/gatling-charts-
highcharts-bundle/3.9.5/gatling-charts-highcharts-bundle-3.9.5-bundle.zip
[...]
2023-08-16 15:12:47 (12.8 MB/s) - « /home/tobby/.gatling/gatling-charts-
highcharts-bundle-3.9.5-bundle.zip » sauvegardé [77080673/77080673]
```

- Dézippage de Gatling

```
tobby@Nidus:~ $ unzip ~/.gatling/gatling-charts-highcharts-bundle-3.9.5-
bundle.zip -d ~/.gatling/
Archive:  /home/tobby/.gatling/gatling-charts-highcharts-bundle-3.9.5-bundle.zip
[...]
```

- Vérification du dézippage

```
tobby@Nidus:~ $ cd .gatling/
tobby@Nidus:~/gatling $ ls -la
total 75288
drwxr-xr-x  3 toby toby      4096 16 aoû 15:12 .
drwxr-xr-x 18 toby toby      4096 16 aoû 15:10 ..
drwxr-xr-x  7 toby toby      4096 10 mai 11:19 gatling-charts-highcharts-bundle-
3.9.5
-rw-r--r--  1 toby toby  77080673 10 mai 11:19 gatling-charts-highcharts-bundle-
3.9.5-bundle.zip
tobby@Nidus:~/gatling $ cd gatling-charts-highcharts-bundle-3.9.5/
tobby@Nidus:~/gatling/gatling-charts-highcharts-bundle-3.9.5 $ ls -la
total 48
drwxr-xr-x  7 toby toby  4096 10 mai 11:19 .
drwxr-xr-x  3 toby toby  4096 16 aoû 15:12 ..
drwxr-xr-x  2 toby toby  4096 10 mai 11:19 bin
drwxr-xr-x  2 toby toby  4096 10 mai 11:19 conf
drwxr-xr-x  2 toby toby 12288 10 mai 11:19 lib
-rw-r--r--  1 toby toby 11367 10 mai 11:19 LICENSE
drwxr-xr-x  2 toby toby  4096 10 mai 11:19 results
drwxr-xr-x  5 toby toby  4096 10 mai 11:19 user-files
```

13.2. Vérification de l'installation

Pour vérifier l'installation de Gatling, il suffit de lancer le script `gatling.sh` situé dans le répertoire `bin` de Gatling. Ce faisant, Gatling affichera un menu permettant de choisir entre plusieurs options. Dans notre cas, nous choisirons l'option 1, qui permet de lancer une simulation de test de charge.

Ensuite, il y a la possibilité de choisir la simulation à exécuter. Dans notre cas, il n'y a qu'une seule simulation, donc nous n'avons pas besoin de la choisir.

Cette simulation est celle créée par Gatling et vas aller tester un serveur web de test de Gatling.

```
tobby@Nidus:~/gatling/gatling-charts-highcharts-bundle-3.9.5/bin $ ./gatling.sh
GATLING_HOME is set to /home/tobby/.gatling/gatling-charts-highcharts-bundle-
3.9.5
Do you want to run the simulation locally, on Gatling Enterprise, or just package
it?
Type the number corresponding to your choice and press enter
[0] <Quit>
[1] Run the Simulation locally
[2] Package and upload the Simulation to Gatling Enterprise Cloud, and run it
there
[3] Package the Simulation for Gatling Enterprise
[4] Show help and exit
1
août 16, 2023 4:28:28 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
computerdatabase.ComputerDatabaseSimulation is the only simulation, executing it.
Select run description (optional)
InstallVerif
Simulation computerdatabase.ComputerDatabaseSimulation started...

[...]

Simulation computerdatabase.ComputerDatabaseSimulation completed in 17 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...

=====
---- Global Information -----
> request count                                108 (OK=105   KO=3    )
> min response time                            108 (OK=108   KO=111    )
> max response time                            1563 (OK=1563  KO=114    )
> mean response time                           162 (OK=163   KO=112    )
> std deviation                               168 (OK=170   KO=1      )
> response time 50th percentile                115 (OK=115   KO=112    )
> response time 75th percentile                120 (OK=121   KO=113    )
> response time 95th percentile                351 (OK=352   KO=114    )
> response time 99th percentile                620 (OK=620   KO=114    )
> mean requests/sec                          6.353 (OK=6.176 KO=0.176 )
---- Response Time Distribution -----
> t < 800 ms                                    104 ( 96%)
> 800 ms <= t < 1200 ms                      0 ( 0%)
> t >= 1200 ms                                 1 ( 1%)
> failed                                       3 ( 3%)
---- Errors -----
> status.find.is(201), but actually found 200          3 (100, 0%)
=====

Reports generated in 0s.
Please open the following file: file:///home/tobby/.gatling/gatling-charts-
highcharts-bundle-3.9.5/results/computerdatabasesimulation-
20230816142907884/index.html
```

13.3. Scripts

Pour Gatling, j'ai d'abord choisi le Java comme language de programmation mais je me suis tournée ensuite vers le Scala car il est plus adapté à Gatling. J'ai donc créé un script Scala qui permet de faire un test de charge sur le site web. Ce script est très simple, il se contente de faire une requête GET sur la page d'accueil du site web. Il est possible de modifier le nombre d'utilisateur et le temps de test dans le script. J'ai aussi créé un script bash qui permet de lancer le script Scala.

```
import io.gatling.core.Predef._
import io.gatling.http.Predef._
import scala.concurrent.duration._

class CuriusTREx_Bash extends Simulation {

    val httpProtocol = http
        .baseUrl("http://volt.s2.rpn.ch") // Il s'agit de l'adresse du site web
        .inferHtmlResources() // Permet de récupérer les ressources HTML
        .acceptHeader("image/avif,image/webp,*/*") // Accepte les images
        .acceptEncodingHeader("gzip, deflate") // Accepte la compression
        .acceptLanguageHeader("fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3") // Accepte les langues
        .userAgentHeader("Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/114.0") // User Agent

    val headers_0 = Map(
        "Accept" -> "text/css,*/*;q=0.1",
        "If-Modified-Since" -> "Thu, 17 Aug 2023 08:18:41 GMT",
        "If-None-Match" -> "\"dc3-6031a0f5b4a47-gzip\""
    )

    val headers_1 = Map(
        "Accept" ->
            "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8",
        "Upgrade-Insecure-Requests" -> "1"
    )

    val headers_4 = Map(
        "If-Modified-Since" -> "Thu, 17 Aug 2023 07:26:33 GMT",
        "If-None-Match" -> "\"164ac-6031954e8df3b\""
    )

    val headers_6 = Map(
        "If-Modified-Since" -> "Thu, 17 Aug 2023 07:26:33 GMT",
        "If-None-Match" -> "\"14c4c-6031954e8cf9b\""
    )

    val headers_7 = Map(
        "Accept" ->
            "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8",
        "If-Modified-Since" -> "Thu, 17 Aug 2023 08:08:51 GMT",
        "If-None-Match" -> "\"2129-60319ec28bede-gzip\"",
        "Upgrade-Insecure-Requests" -> "1"
    )

    val scn = scenario("CuriusTREx")
        .exec(
            http("request_0")
                .get("/styles.css")
                .headers(headers_0)
        )
        // Start
        .exec(

```

```
http("request_1")
    .get("/contact.html")
    .headers(headers_1)
)
.exec(
    http("request_2")
        .get("/about.html")
        .headers(headers_1)
    .resources(
        http("request_3")
            .get("/capture/Home.jpg"),
        http("request_4")
            .get("/capture/Test_Complet.jpg")
            .headers(headers_4),
        http("request_5")
            .get("/capture/Donn%C3%A9es.jpg"),
        http("request_6")
            .get("/capture/Test.jpg")
            .headers(headers_6)
    )
)
.exec(
    http("request_7")
        .get("/about.html")
        .headers(headers_7)
)
.exec(flushHttpCache)
.exec(flushSessionCookies)
.exec(flushCookieJar)

val nbUsers = java.lang.Long.getLong("users", 1).toDouble
val myRamp = java.lang.Long.getLong("ramp", 0)
println(s"Nombre d'utilisateurs : $nbUsers")
println(s"Temps de montée : $myRamp")

setUp scn.inject(constantUsersPerSec(nbUsers).during(myRamp
seconds))).protocols(httpProtocol)
}
```

14. Apache et Site Web

14.1. Installation

Pour installer et activer Apache, il suffit d'exécuter les commandes suivantes :

```
sudo apt install apache2  
sudo systemctl enable apache2
```

- Vérification de l'installation

```
sudo systemctl status apache2
```

14.2. Mise en place d'un site Web

J'ai créé un site web très simple reprenant le README du projet. Et il comporte trois pages ainsi que du CSS.

En premier lieu je le copie donc sur le serveur web avec la commande suivante :

```
scp -r /home/toblerc/Documents/ES_2024/banc-de-mesures-de-la-consommation-  
electrique/siteWeb/www/html toby@Volt:/var/www/html/
```

15. MQTT

Le MQTT, ou Message Queuing Telemetry Transport, est un protocole de messagerie exceptionnellement léger qui repose sur le protocole TCP/IP. Sa conception vise à répondre aux besoins des appareils dotés de ressources limitées en termes de calcul et de bande passante, en faisant un choix privilégié pour l'Internet des objets (IoT). L'architecture de ce protocole repose sur le principe fondamental de publication et d'abonnement.

Au cœur du MQTT se trouve un élément essentiel : le broker MQTT. Celui-ci joue le rôle de médiateur en recevant les messages émis par les clients puis en les diffusant aux clients abonnés.

Le protocole MQTT s'appuie sur le protocole TCP/IP et utilise deux ports principaux : le port 1883 pour les communications non sécurisées et le port 8883 pour les communications sécurisées. La sécurité des communications MQTT est assurée par le protocole TLS/SSL.

Il existe deux catégories de clients MQTT : les clients légers et les clients complets. Les clients MQTT légers ne prennent pas en charge le protocole TLS/SSL, tandis que les clients MQTT complets le prennent en charge. Les clients MQTT légers sont fréquemment utilisés pour les dispositifs IoT en raison de leur faible empreinte mémoire et de leur simplicité, tandis que les clients MQTT complets sont davantage adaptés aux applications de bureau.

Ma décision d'utiliser MQTT découle de mon expérience antérieure. J'ai eu l'occasion d'employer ce protocole lors de mon travail de diplôme pour mon CFC. À l'époque, j'utilisais MQTT pour l'échange de données entre un Raspberry Pi et les puces de développement de Microchip. Cette expérience m'a permis de développer une solide compréhension de MQTT.

En utilisant MQTT, nous évitons la complexité liée au transfert de requêtes via SSH ainsi que l'utilisation de clés SSH. Cela se traduit par un gain significatif en termes de performances et de sécurité. Sur le plan des performances, MQTT se distingue par sa légèreté, étant environ dix fois plus léger que SSH. Cette légèreté en fait un choix judicieux pour la transmission de données de consommation.

Au niveau du soft que j'ai choisi de mettre en place, il s'agit de Mosquitto, un broker MQTT open source. Il est disponible sur les dépôts officiels de Debian, ce qui facilite son installation. De plus, il est très simple à configurer et j'ai déjà eu l'occasion de l'utiliser dans le cadre de mon travail de diplôme de CFC.

15.1. Installation de Mosquitto sur Nidus

- Installation du paquet Mosquitto

```
tobby@Nidus:~/.ssh $ sudo apt install mosquitto
Lecture des listes de paquets... Fait
[...]
```

- Vérification du status de Mosquitto

```
tobby@Nidus:~/.ssh $ sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
    Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor
preset: enabled)
      Active: active (running) since Tue 2023-08-22 16:01:58 CEST; 7s ago
        Docs: man:mosquitto.conf(5)
               man:mosquitto(8)
    Process: 22571 ExecStartPre=/bin/mkdir -m 740 -p /var/log/mosquitto
(code=exited, status=0/SUCCESS)
    Process: 22572 ExecStartPre=/bin/chown mosquitto /var/log/mosquitto
(code=exited, status=0/SUCCESS)
    Process: 22573 ExecStartPre=/bin/mkdir -m 740 -p /run/mosquitto (code=exited,
status=0/SUCCESS)
    Process: 22574 ExecStartPre=/bin/chown mosquitto /run/mosquitto (code=exited,
status=0/SUCCESS)
   Main PID: 22575 (mosquitto)
     Tasks: 1 (limit: 3933)
       CPU: 42ms
      CGroup: /system.slice/mosquitto.service
              └─22575 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

aoû 22 16:01:58 Nidus systemd[1]: Starting Mosquitto MQTT Broker...
aoû 22 16:01:58 Nidus systemd[1]: Started Mosquitto MQTT Broker.
```

15.2. Ouverture des port sur Nidus

Pour que le broker MQTT soit accessible depuis l'extérieur, il faut ouvrir les ports 1883 sur Nidus. Dans le fichier de configuration de Mosquitto, il faut aussi autoriser les connexions anonymes.

Il faut donc modifier le fichier de conf comme suit :

```
tobby@Nidus:~ $ sudo vim /etc/mosquitto/mosquitto.conf
```

- Vérification du fichier de conf

```
tobby@Nidus:~ $ sudo cat /etc/mosquitto/mosquitto.conf
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 1883
allow_anonymous true
```

15.3. Script MQTT

J'ai développé un script MQTT relativement sophistiqué, spécialement conçu pour publier efficacement les données de consommation sur le broker MQTT. Ce script, lors du démarrage de la machine, s'exécute en boucle continue, assurant ainsi une collecte et une publication régulières de ces données. Son objectif principal est d'optimiser les performances tout en garantissant la fiabilité du processus.

Comme nous l'avons évoqué à plusieurs reprises, l'objectif de ce banc de test est de minimiser les exigences envers Volt. Idéalement, il devrait être possible de simplement flasher une image sur une carte SD, de l'insérer dans Volt, et de connecter le banc de test sans contraintes majeures liées à un système d'exploitation spécifique ou à une configuration particulière.

Cependant, il est presque impossible de créer un système entièrement "plug and play". Par conséquent, le script ci-dessous accomplit une tâche cruciale en mettant en place toutes les dépendances et configurations nécessaires pour assurer le bon fonctionnement du script et l'envoi réussi des données vers le broker MQTT.

15.3.1. Script

```
#!/bin/bash
### BEGIN INIT INFO
# Provides: mqtt
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Script MQTT de collecte de données
# Description: Ce script collecte la charge CPU, la charge RAM
# et le nombre de processus, puis publie ces données
# sur un broker MQTT.
### END INIT INFO

# Pour ajouter les droits d'exécution :
# chmod +x mqtt.sh
# Pour le copier depuis Nidus vers Volt :
# scp ./mqtt.sh toobby@volt:/usr/local/bin/mqtt.sh
# Emplacement du script (doit être dans /usr/local/bin)
INSTALL_DIR="/usr/local/bin" # Nom du script
SCRIPT_NAME="mqtt.sh" # Adresse du broker MQTT
MQTT_BROKER="nidus" # Sujets MQTT pour les différentes données
MQTT_TOPIC_CPU="benchmark/cpu"
MQTT_TOPIC_RAM="benchmark/ram"
MQTT_TOPIC_PROCESSES="benchmark/processes"

# Vérification si le script est dans le bon dossier d'installation
if [ "$(dirname "$(readlink -f "$0")")" != "$INSTALL_DIR" ]; then
    echo "Erreur : Le script doit être installé dans $INSTALL_DIR"
    exit 1
fi

# Vérification et installation des dépendances (mosquitto-clients)
if ! command -v mosquitto_pub &> /dev/null; then
    echo "Installation de mosquitto-clients..."
    sudo apt-get update
    sudo apt-get install mosquitto-clients
    echo "Installation terminée."
fi

# Vérification si le lien symbolique vers init.d existe
if [ ! -e "/etc/init.d/$SCRIPT_NAME" ]; then
    echo "Création du lien symbolique dans /etc/init.d..."
```

```
sudo ln -s "$INSTALL_DIR/$SCRIPT_NAME" "/etc/init.d/$SCRIPT_NAME"
echo "Lien symbolique créé."
fi

# Vérification et activation du service init.d
if ! sudo service "$SCRIPT_NAME" status &> /dev/null; then
    echo "Activation du service..."
    sudo update-rc.d "$SCRIPT_NAME" defaults
    echo "Service activé."
fi

# Boucle principale pour la collecte et la publication des données
while true; do
    # Collecte des données
    CPU_LOAD=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')
    RAM_LOAD=$(free | awk '/Mem/{printf("%.2f\n", $3/$2*100)}')
    PROCESS_COUNT=$(ps aux | wc -l)

    # Publication des données sur MQTT
    mosquitto_pub -h $MQTT_BROKER -t $MQTT_TOPIC_CPU -m "$CPU_LOAD"
    mosquitto_pub -h $MQTT_BROKER -t $MQTT_TOPIC_RAM -m "$RAM_LOAD"
    mosquitto_pub -h $MQTT_BROKER -t $MQTT_TOPIC_PROCESSES -m "$PROCESS_COUNT"

    echo "Données publiées sur MQTT"
    sleep 1 # Attente d'une seconde
done
```

15.3.2. Description détaillée du script

Le script commence par vérifier si l'emplacement d'installation est correct, s'assurant qu'il est placé dans le répertoire défini par **INSTALL_DIR**. Ensuite, il vérifie la présence et l'installation des dépendances requises, notamment **mosquitto-clients**, en l'installant si nécessaire.

Une autre vérification importante concerne l'existence d'un lien symbolique vers **/etc/init.d**, qui est nécessaire pour exécuter le script au démarrage de la machine. Si le lien symbolique n'existe pas, le script le crée.

Ensuite, le script s'assure que le service init.d correspondant est activé. Si ce n'est pas le cas, il active le service en utilisant la commande **update-rc.d**.

La section la plus importante du script est la boucle principale, où les données de consommation sont collectées et publiées en continu sur le broker MQTT. Pour chaque itération de la boucle, les taux de charge CPU, de charge RAM et le nombre de processus en cours sont mesurés et enregistrés.

Ces données sont ensuite publiées sur le broker MQTT à l'aide de la commande **mosquitto_pub**. Chaque type de données est publié sur un sujet MQTT spécifique (**\$MQTT_TOPIC_CPU**, **\$MQTT_TOPIC_RAM**, **\$MQTT_TOPIC_PROCESSES**), ce qui permet de les organiser de manière claire.

Le script affiche également un message indiquant que les données ont été publiées sur MQTT, et ensuite attend une seconde avant de reprendre une nouvelle itération de la boucle.

15.3.3. Conclusion

Ce script MQTT élaboré et bien structuré offre un moyen efficace de collecter et de publier les données de consommation sur le broker **MQTT**. Son fonctionnement en boucle continue, combiné à des vérifications et des actions préliminaires, garantit une gestion fiable et optimisée des données, contribuant ainsi à la réussite globale du projet.

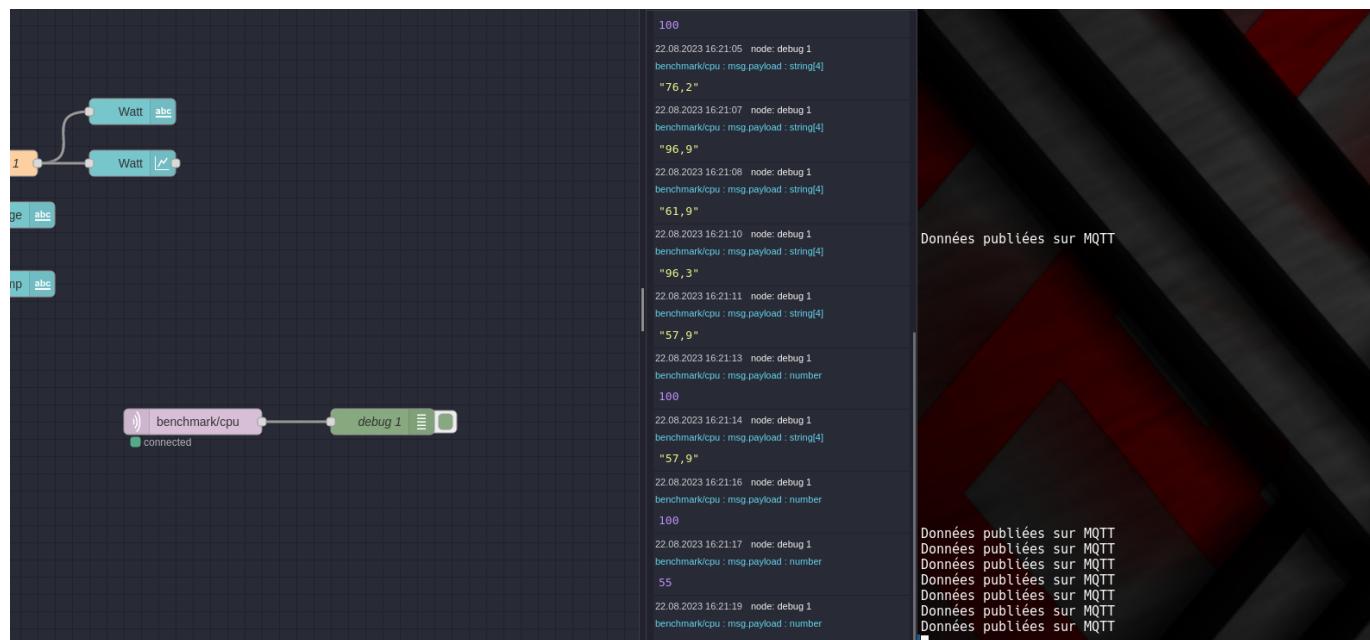
15.4. Installation

```
toblerc@LPT-UNIX-USB-CT:~/Documents/ES_2024/banc-de-mesures-de-la-consommation-electrique$ scp ./mqtt.sh toby@volt:/usr/local/bin/mqtt.sh
mqtt.sh
100% 2526      2.1MB/s   00:00
```

15.5. Utilisation du script

```
tobby@Volt:/usr/local/bin$ sudo ./mqtt.sh
Installation de mosquitto-clients...
[...]
Il est nécessaire de prendre 136 ko dans les archives.
Après cette opération, 568 ko d'espace disque supplémentaires seront utilisés.
Souhaitez-vous continuer ? [0/n] 0
[...]
Installation terminée.
Création du lien symbolique dans /etc/init.d...
Lien symbolique créé.
Activation du service...
Service activé.
```

15.5.1. Vérification



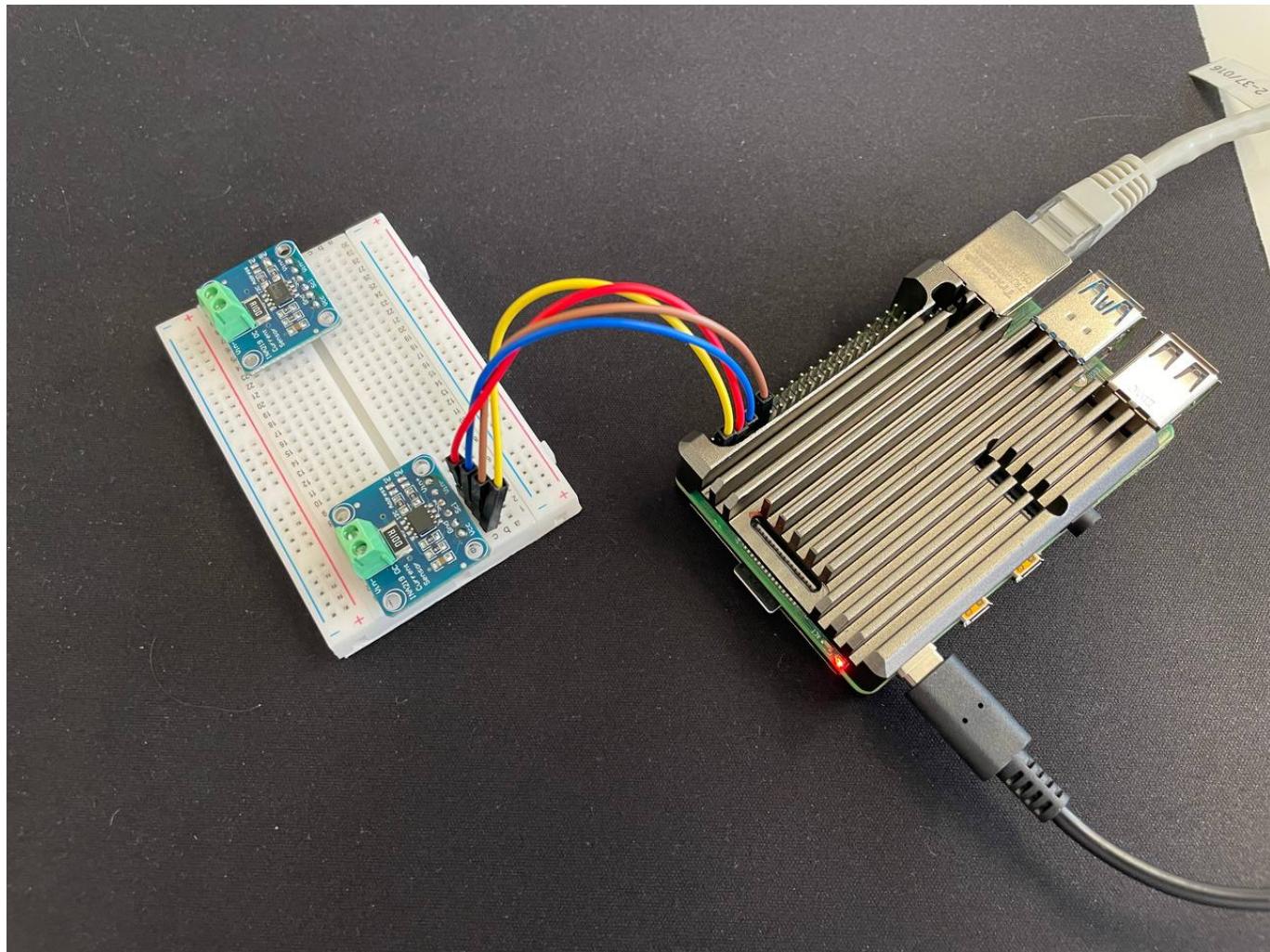
16. INA219

Dans ce chapitre, nous explorerons la puce **INA219**, qui joue un rôle essentiel dans la mesure de la consommation. Il est important de noter que nous utilisons deux puces INA219 dans ce projet : l'une pour la mesure proprement dite et l'autre en tant que pièce de rechange en cas de problème. Pour les différencier, nous avons effectué des soudures pour attribuer des adresses I2C spécifiques à chaque puce. L'adresse de la puce de mesure est réglée sur *0x40*, tandis que l'adresse de la puce de remplacement est réglée sur *0x41*.

16.0.1. Installation physique

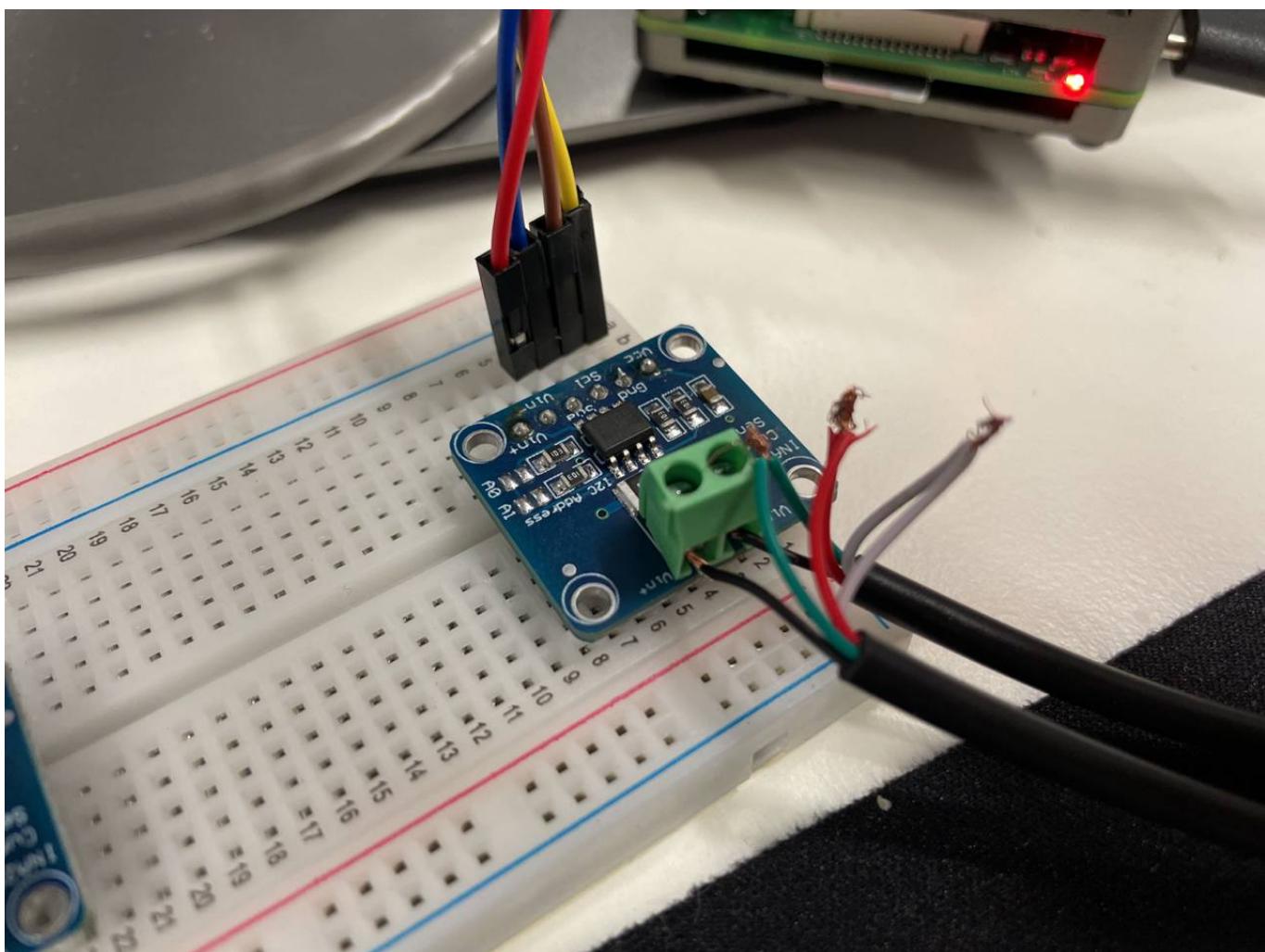
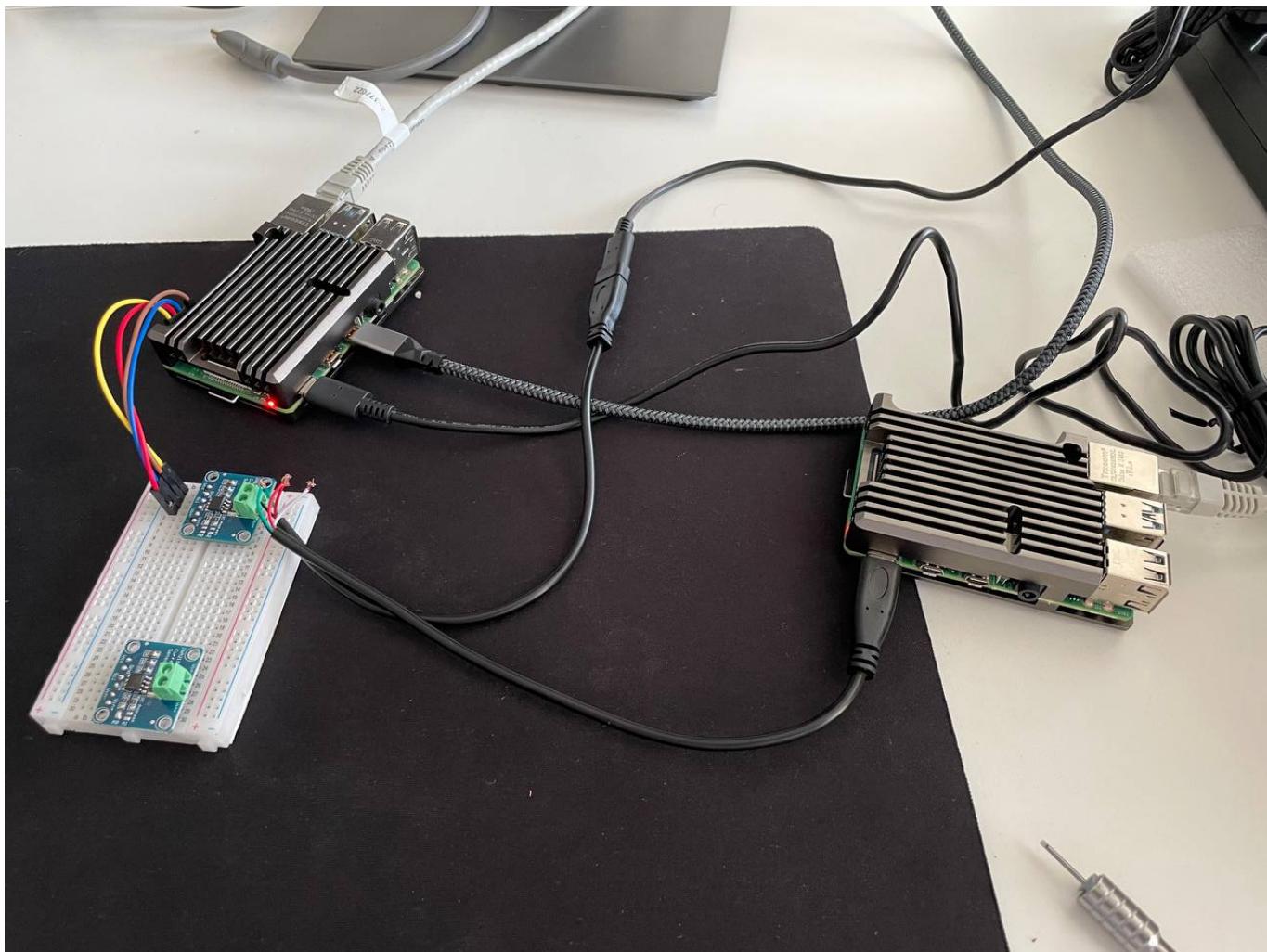
L'installation physique du **INA219** implique des branchements spécifiques en fonction des scénarios : avec ou sans le dispositif Volt. Voici les détails de chaque configuration :

16.0.1.1. Branchement SANS VOLT



16.0.1.2. Branchement AVEC VOLT

Le branchement avec le dispositif Volt ajoute une complexité supplémentaire. Voici un aperçu détaillé de ce branchement :



16.0.2. Vérification de la présence du INA219

Avant de pouvoir commencer à utiliser le **INA219** pour mesurer la consommation, il est crucial de vérifier la présence de la puce et de s'assurer qu'elle est correctement détectée par le système. Cette étape est essentielle pour garantir des mesures précises et fiables tout au long du projet.

```
tobby@Nidus:~ $ sudo i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          - - - - - - - - - - - - - - - - - -
10:          - - - - - - - - - - - - - - - - - -
20:          - - - - - - - - - - - - - - - - - -
30:          - - - - - - - - - - - - - - - - - -
40: 40          - - - - - - - - - - - - - - - -
50:          - - - - - - - - - - - - - - - - - -
60:          - - - - - - - - - - - - - - - - - -
70:          - - - - - - - - - - - - - - - - - -
```



```
tobby@Nidus:~ $ sudo i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          - - - - - - - - - - - - - - - - - -
10:          - - - - - - - - - - - - - - - - - -
20:          - - - - - - - - - - - - - - - - - -
30:          - - - - - - - - - - - - - - - - - -
40:          - - 41 - - - - - - - - - - - - - -
50:          - - - - - - - - - - - - - - - - - -
60:          - - - - - - - - - - - - - - - - - -
70:          - - - - - - - - - - - - - - - - - -
```

16.1. Obtention des données

16.1.1. Test avec le script python A vide

Installation de la bibliothèque python

```
tobby@Nidus:~ $ sudo pip3 install pi-ina219
[...]
```

Vérification de la présence de l'INA219

```
tobby@Nidus:~ $ i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          - - - - - - - - - - - - - - - - - -
10:          - - - - - - - - - - - - - - - - - -
20:          - - - - - - - - - - - - - - - - - -
30:          - - - - - - - - - - - - - - - - - -
40: 40          - - - - - - - - - - - - - - - -
50:          - - - - - - - - - - - - - - - - - -
60:          - - - - - - - - - - - - - - - - - -
70:          - - - - - - - - - - - - - - - - - -
```

- Création du dossier py et du script python

```
tobby@Nidus:~/Documents $ mkdir py
tobby@Nidus:~/Documents $ cd py
tobby@Nidus:~/Documents/py $ touch my_ina219.py
```

- Vérification de la présence du script

```
tobby@Nidus:~/Documents/py $ ls -la
total 8
drwxr-xr-x 2 toby toby 4096 22 aoû 10:19 .
drwxr-xr-x 3 toby toby 4096 22 aoû 10:18 ..
-rw-r--r-- 1 toby toby    0 22 aoû 10:19 my_ina219.py
```

- Modification du script

```
tobby@Nidus:~/Documents/py $ sudo vi ./my_ina219.py
```

```
#!/usr/bin/env python
from ina219 import INA219
from ina219 import DeviceRangeError

SHUNT_OHMS = 0.1

def read():
    ina = INA219(SHUNT_OHMS)
    ina.configure()

    print("Bus Voltage: %.3f V" % ina.voltage())
    try:
        print("Bus Current: %.3f mA" % ina.current())
        print("Power: %.3f mW" % ina.power())
        print("Shunt voltage: %.3f mV" % ina.shunt_voltage())
    except DeviceRangeError as e:
        # Current out of device range with specified shunt resistor
        print(e)

if __name__ == "__main__":
    read()
```

Execution du script

```
tobby@Nidus:~/Documents/py $ python ./my_ina219.py
Bus Voltage: 0.888 V
Bus Current: -0.195 mA
Power: 0.000 mW
Shunt voltage: -0.010 mV
```

17. Noeud Node-Red

17.1. Dashboard

Le **Dashboard**, en tant que centre de contrôle essentiel, rassemble tous les éléments nécessaires pour une visualisation optimale des données générées.

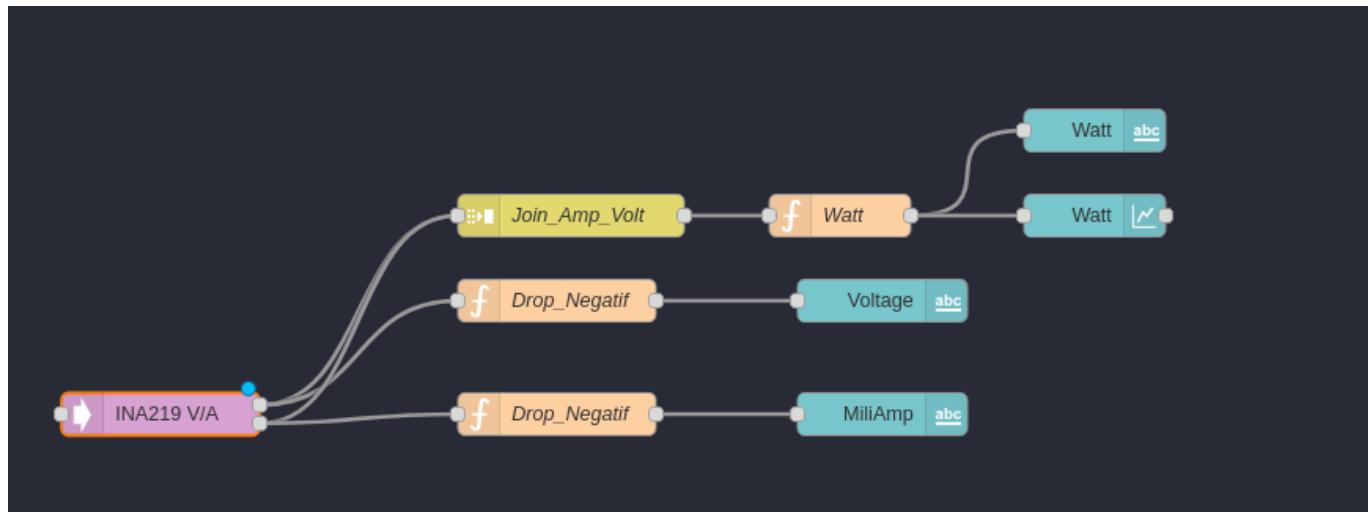
Il met à disposition un ensemble complet de noeuds spécifiques, créant une interface utilisateur intuitive et interactive. Ces noeuds proposent une gamme variée de fonctionnalités pour présenter, ajuster et transmettre les données. Voici quelques exemples des noeuds qui contribuent à cette expérience :

- **Bouton (Button)** : Permet aux utilisateurs d'interagir et de déclencher des actions de manière directe.
- **Liste déroulante (Dropdown)** : Offre un moyen de sélectionner parmi plusieurs options, permettant un contrôle structuré des paramètres ou des valeurs.
- **Interrupteur (Switch)** : Fournit une transition immédiate entre deux états, souvent utilisé pour activer ou désactiver des fonctionnalités.
- **Curseur (Slider)** : Permet un réglage précis d'une valeur numérique en glissant un curseur. Utile pour ajuster des paramètres continus.
- **Champ numérique (Numeric)** : Fournit une interface pour entrer des valeurs numériques avec précision.
- **Champ de texte (Text input)** : Permet aux utilisateurs d'entrer du texte, généralement pour des commentaires, des descriptions ou des valeurs personnalisées.
- **Sélecteur de date (Date picker)** : Facilite la sélection de dates et d'heures, souvent utilisé pour des enregistrements horodatés.
- **Sélecteur de couleur (Colour picker)** : Permet de choisir précisément une couleur pour des éléments visuels ou des codes couleur.
- **Formulaire (Form)** : Regroupe plusieurs champs de saisie et de contrôle en une entité logique, simplifiant ainsi la collecte de données.
- **Texte (Text)** : Affiche du texte ou des instructions pour guider l'utilisateur dans l'interprétation des données ou l'utilisation de l'interface.
- **Jauge (Gauge)** : Présente graphiquement une valeur numérique, offrant une visualisation rapide d'un état ou d'une mesure.
- **Graphique (Chart)** : Permet la création de divers types de graphiques pour illustrer visuellement les tendances et les relations entre les données.
- **Sortie audio (Audio out)** : Peut être utilisée pour fournir des commentaires auditifs ou des alertes sonores.
- **Notification (Notification)** : Affiche des messages d'information ou d'alerte à l'utilisateur pour des événements spécifiques.
- **Contrôle d'interface utilisateur (UI control)** : Offre des éléments interactifs personnalisables pour répondre aux besoins spécifiques de l'application.
- **Modèle (Template)** : Permet d'intégrer du contenu HTML personnalisé, offrant une flexibilité avancée pour inclure graphiques, widgets et plus encore.

Ces noeuds apportent un ensemble puissant d'outils pour la création d'interfaces visuelles riches, éliminant la nécessité d'une programmation manuelle pour chaque élément. Cela encourage la collaboration efficace entre les développeurs et les utilisateurs non techniques dans la conception d'interfaces utilisateur conviviales et informatives.

17.2. INA219

Dans cette section, nous explorons le composant **INA219**, un élément clé de notre projet. L'**INA219** est équipé de deux sorties qui fournissent des valeurs en millampères et en volts, offrant ainsi des informations cruciales sur la consommation.



Pour tirer le meilleur parti de l'INA219, j'ai mis en place une configuration sophistiquée. J'ai configuré des nœuds de fonctions spécifiques pour exclure les valeurs négatives. Ces valeurs négatives sont généralement des erreurs de lecture et doivent être traitées correctement pour garantir des données précises. Ensuite, j'ai élaboré une séquence de traitement pour afficher ces valeurs de manière compréhensible dans un libellé.

En plus de cela, j'ai mis en place un nœud "join" qui joue un rôle crucial. Ce nœud fusionne les deux valeurs obtenues à partir des sorties de l'INA219 en un seul message cohérent. Ce message est ensuite acheminé vers un autre nœud de fonction spécialisé. Ce nœud effectue des calculs complexes pour obtenir les données de consommation en watts. Ces données sont ensuite affichées à la fois dans un libellé, offrant une visualisation claire des résultats, et dans un graphique, permettant une compréhension visuelle de l'évolution de la consommation.

```

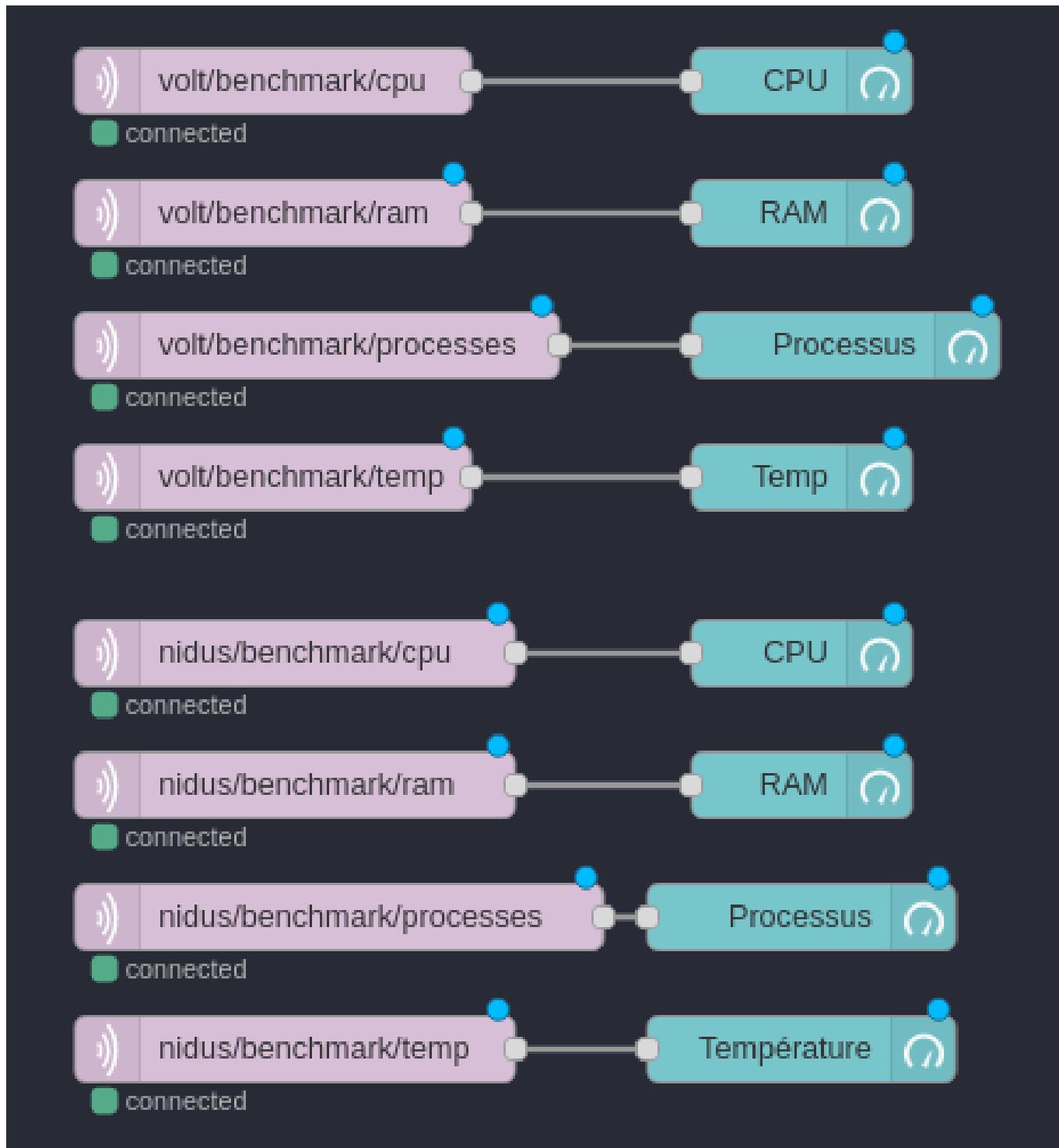
// Récupérer les valeurs de courant (mA) et de tension (V) depuis les propriétés
msg.payload
var current_mA = msg.payload.miliamps;
var voltage_V = msg.payload.voltage;

// Calculer la puissance en watts (W)
var power_W = (current_mA / 1000) * voltage_V; // Convertir le courant en ampères

// Vérifier si la tension est négative
if (voltage_V < 0.5) {
    // Si la tension est négative, ne rien faire et retourner le message inchangé
    return null;
}

// Créer un nouvel objet msg avec la puissance en watts comme payload
msg.payload = power_W;
msg.topic = "Watt";
// Renvoyer le message modifié
return msg;
  
```

17.3. Monitoring



Dans cette section, nous abordons le **Monitoring**, une étape cruciale de notre projet. Pour cette tâche, j'ai choisi d'utiliser le protocole **MQTT**, qui présente des avantages significatifs en termes de rapidité et de légèreté par rapport au **SSH**.

En commençant par la réception des données via le nœud MQTT, celles-ci sont dirigées vers un nœud de type "gauge" (*jauge*) qui affiche la valeur en temps réel. Cette représentation visuelle offre une vue instantanée de la consommation, permettant une surveillance efficace.

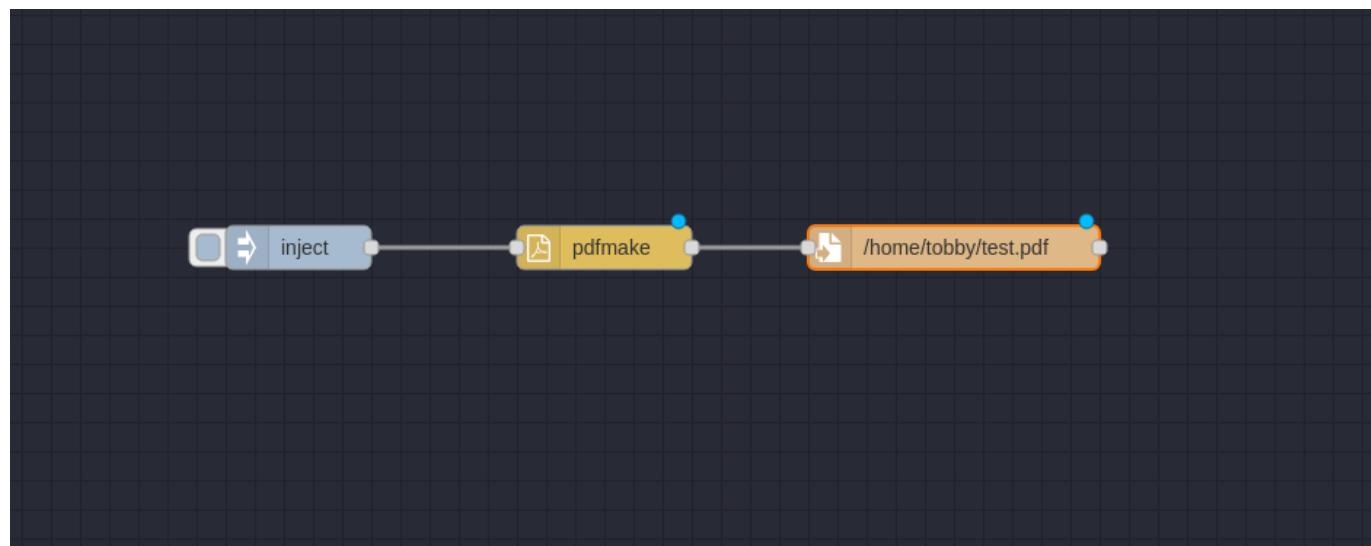
17.4. PDF

17.4.1. Base

Pour generer un PDF, il faut passer un Json dans le payload du message :

```
{  
  "_msgid": "b63574aa110e9d9b"  
  , "payload":  
    {  
      "content":  
        [  
          "First paragraph",  
          "Another paragraph, this time a little bit longer to make sure, this line  
          will be divided into at least two lines"  
        ]  
    },  
    "topic": ""  
}
```

Qui est reçu dans le noeud pdfmake qui le passe en Base64 qui est ensuite reçu dans le noeud write file qui l'écrit dans un fichier PDF.



17.5. Images de graphiques et de tableaux

Une fois que la génération de PDF est maîtrisée, il est temps de valoriser davantage les informations en y ajoutant des images.

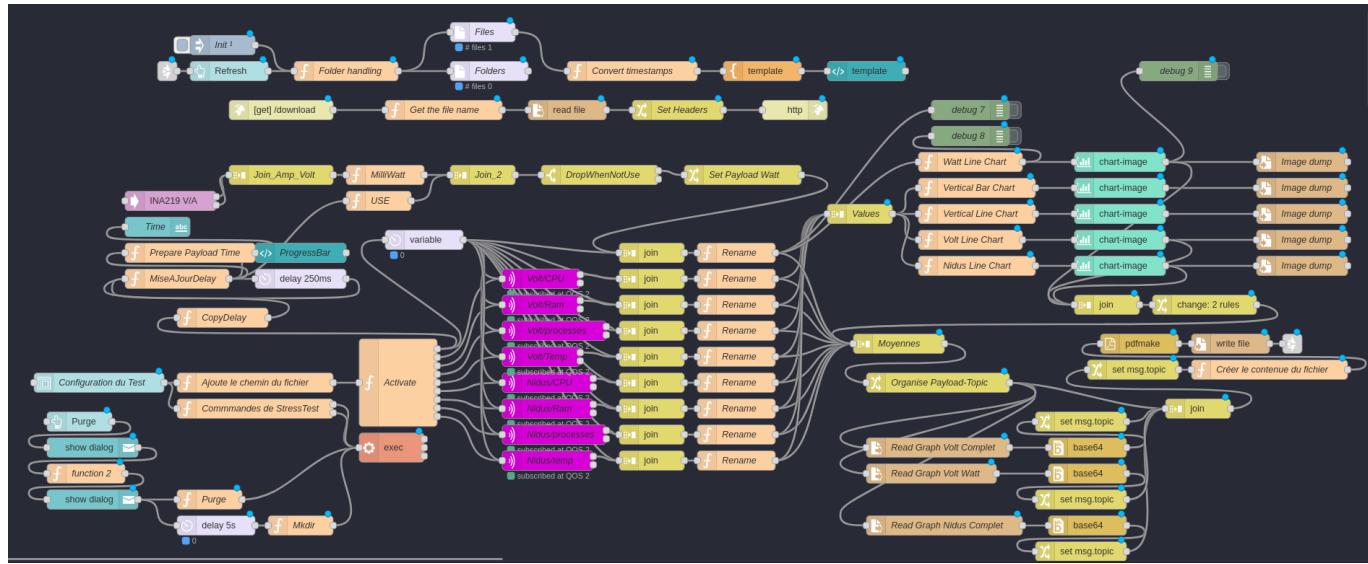
En effet, bien que disposer des valeurs à un instant donné soit utile, pouvoir visualiser ces valeurs sous forme de graphique est encore plus puissant. Pour réaliser cela, nous utiliserons le nœud **node-red-contrib-chart-image**, qui nous permettra de générer des graphiques. Ce nœud repose sur le module **Chart.js**, qui permet de créer des graphiques en utilisant du code JavaScript.

En plus du nœud de graphique, nous aurons besoin du nœud **node-red-node-base64**, qui facilitera la conversion d'images en base64 et vice versa. Cette conversion est essentielle pour intégrer les images dans le document PDF.

Cette combinaison de nœuds nous permettra de créer des représentations visuelles attrayantes et informatives des données, offrant ainsi une compréhension plus approfondie et une présentation visuellement engageante.

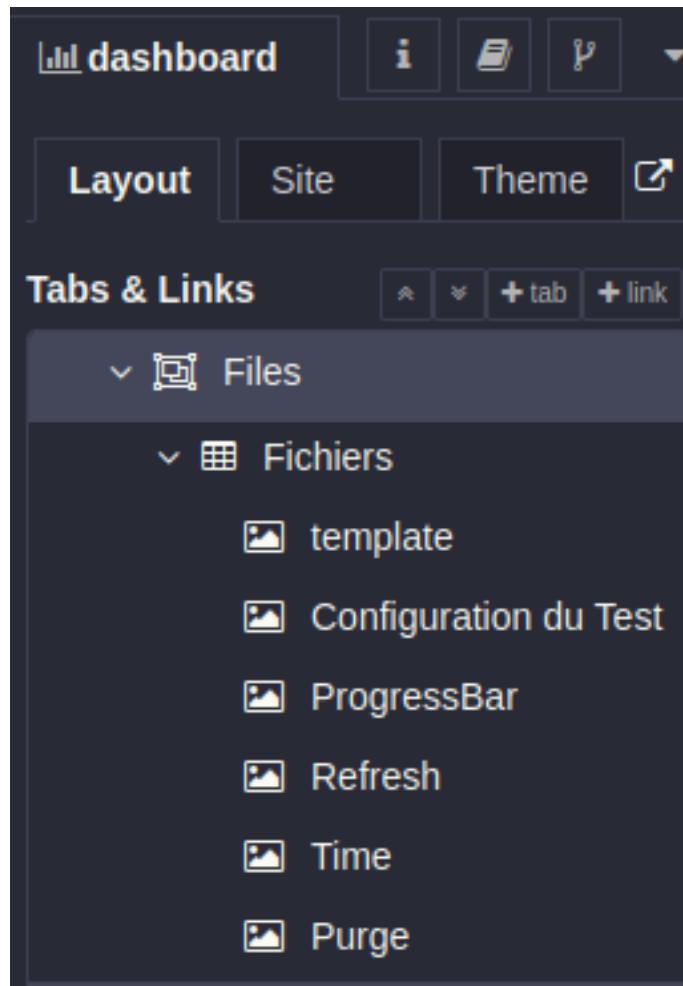
18. Stress Test V1.0

J'ai créé une page qui permet de générer un rapport en fonction de la durée et de l'exécution d'un stress test sur Nidus et/ou sur Volt. Voici le flux complet pour la génération du rapport:



Pour être honnête, il faut admettre que la lisibilité initiale n'est pas optimale. Par conséquent, j'ai décidé de décomposer le processus en plusieurs étapes afin d'obtenir une meilleure compréhension globale.

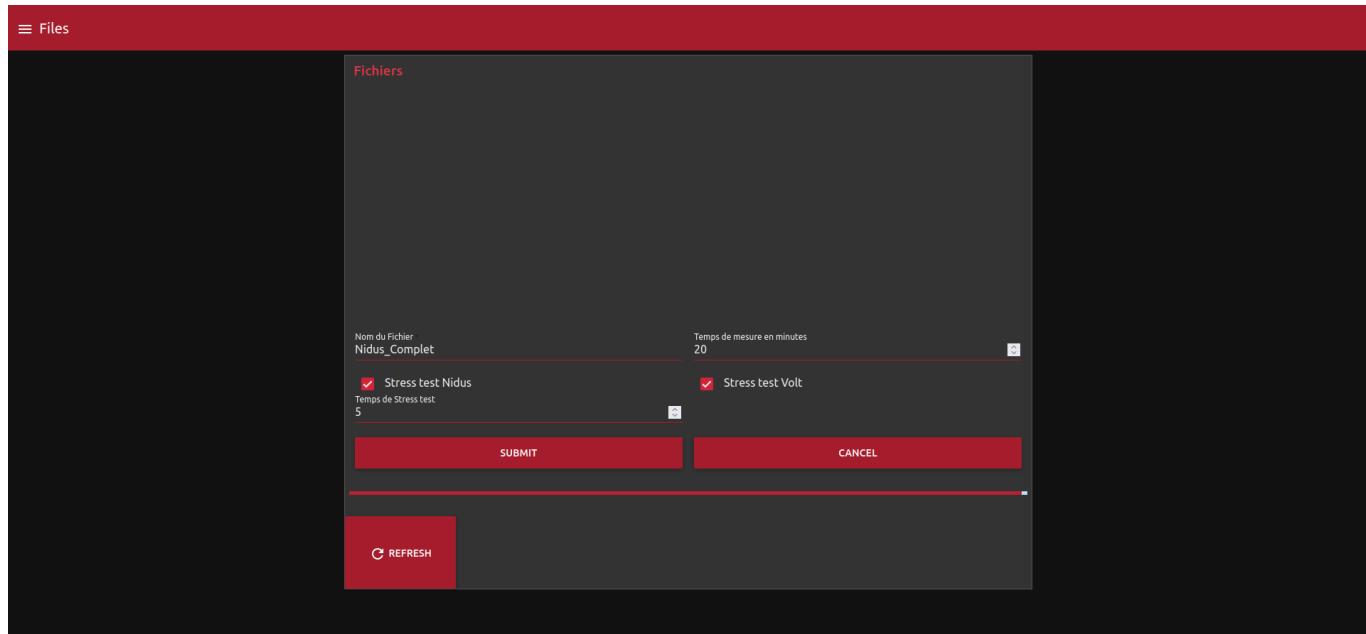
18.1. Écran d'Accueil



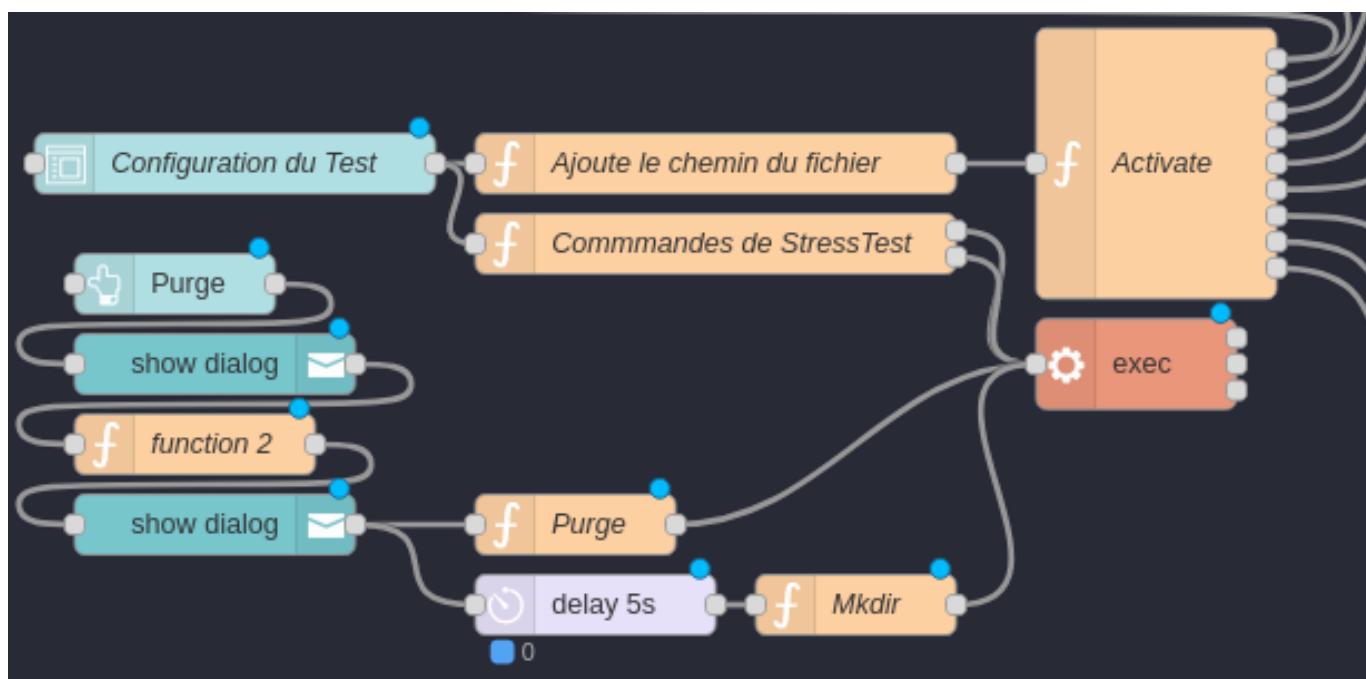
Au premier abord, vous serez accueilli par un navigateur de fichiers et un formulaire. Ce formulaire vous permet de spécifier la durée du test et de décider si vous souhaitez exécuter un test de stress sur Nidus et/ou sur Volt. Voici le contenu de la page "file" qui contient le formulaire:

A screenshot of a 'File' page with a red header bar containing the text 'Files'. Below the header, there's a section titled 'Fichiers'. The main content area contains a form for stress testing. The form includes fields for 'Nom du Fichier' (File name) and 'Temps de mesure en minutes' (Measurement time in minutes), both with dropdown menus. There are also checkboxes for 'Stress test Nidus' and 'Stress test Volt'. Below these are fields for 'Temps de Stress test' (Stress test time) and two large red buttons: 'SUBMIT' and 'CANCEL'. At the bottom of the form, there are three buttons: 'REFRESH' (red), a timer displaying '00:00' (white text on a black background), and 'PURGE' (red).

Après avoir rempli le formulaire:



Les nœuds responsables de cette section sont les suivants:



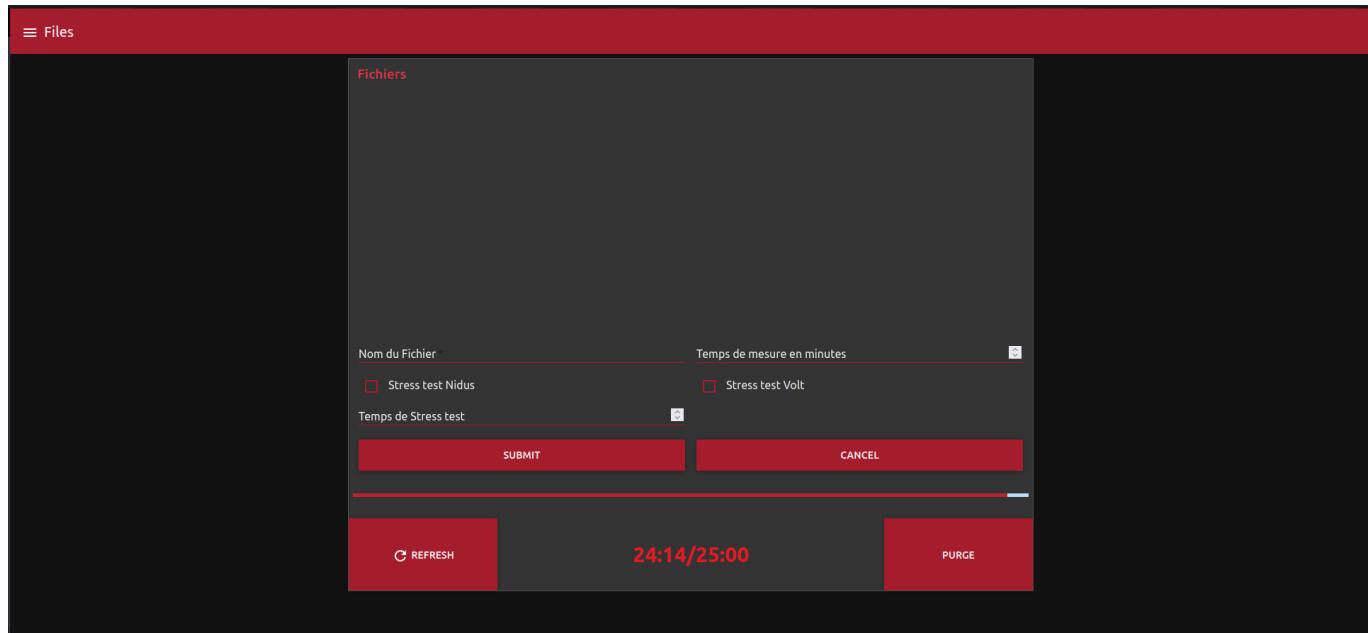
Deux éléments se distinguent ici:

- Un formulaire de "Configuration du Test"
- Un bouton "Purge" dont nous discuterons ultérieurement

Le formulaire recueille les données saisies par l'utilisateur. Ensuite, il transmet ces données en sortie. Deux fonctions sont connectées à cette sortie. La première fonction ajoute les chemins des fichiers, tels que "chart.png" et "report.pdf", à un tableau. La seconde fonction gère l'exécution des tests de stress en fonction des entrées de l'utilisateur, et les envoie ensuite à un nœud "exec" qui exécute les commandes sur Nidus et/ou Volt.

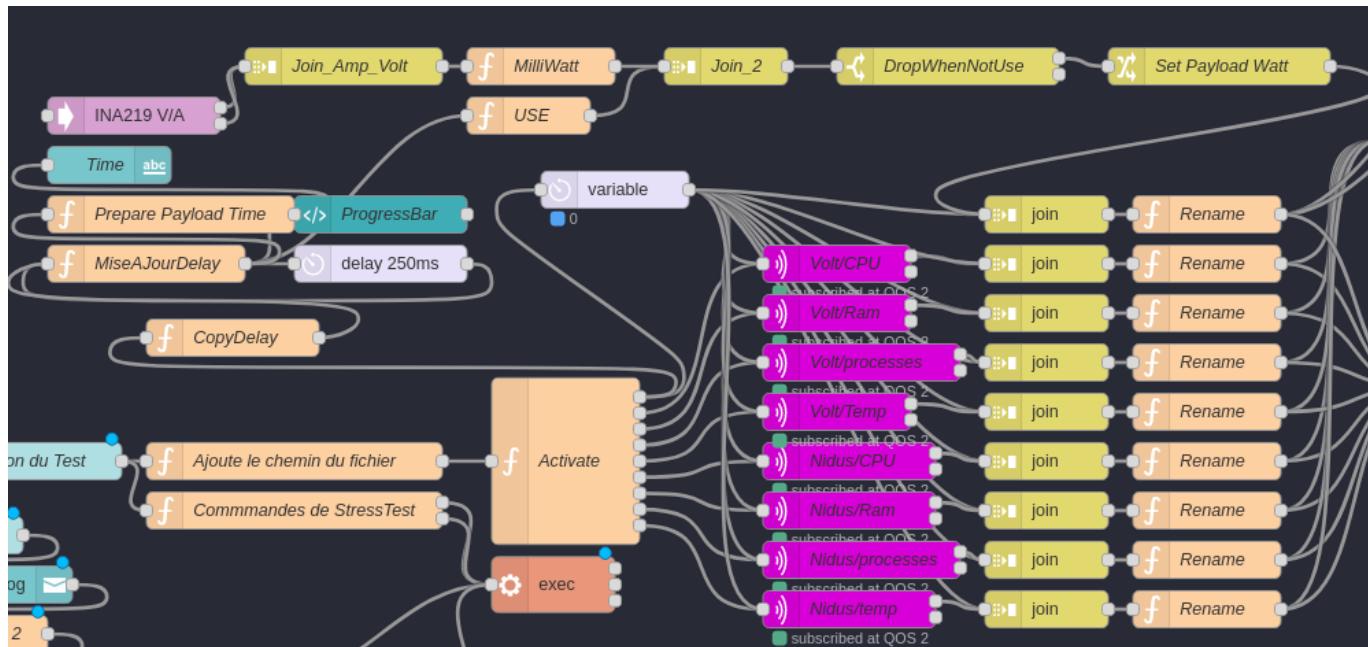
La première fonction transmet ensuite les données à une fonction à sorties multiples, ce qui permet d'envoyer différents messages distincts.

18.2. En Exécution



Pendant l'exécution, une **barre de progression** est affichée pour montrer l'avancement du test, accompagnée d'une **étiquette** en dessous pour indiquer le pourcentage d'avancement. Cela permet d'obtenir une meilleure visualisation de l'état d'avancement.

En arrière-plan, un certain nombre de tâches se déroulent :



Pour en donner plus de détails :

- La première sortie du nœud **Activate** est connectée à un nœud **delay** qui ajuste la durée du test, ainsi qu'à une série d'autres nœuds qui gèrent la barre de progression.
- La deuxième sortie du nœud **Activate** est reliée à un nœud MQTT amélioré. Celui-ci permet de souscrire aux **topics** appropriés. Au début du test, il souscrit au topic `#/benchmark/#`, puis à la fin du test, il reçoit le topic `/` pour se désinscrire. Cela permet de filtrer uniquement les informations nécessaires et d'éviter d'être submergé par les messages superflus envoyés sur le broker MQTT.
- Les six autres sorties du nœud **Activate** ont la même fonction. Elles envoient toutes des messages pour modifier le topic MQTT.

Ensuite, ces messages sont acheminés vers un nœud **join**, qui les combine en un tableau de messages. Ce tableau est ensuite transmis à un nœud **function** chargé de traiter les données. Parmi les tâches effectuées par ce nœud figurent la personnalisation des **topics** pour chaque ensemble de données et le calcul de la moyenne des valeurs reçues :

```
// Définir le sujet du message
msg.topic = "volt/benchmark/cpu";

// Vérifier si le tableau payload existe et n'est pas vide
if (msg.payload && Array.isArray(msg.payload) && msg.payload.length > 0) {
    // Convertir les valeurs en chaînes de caractères en nombres entiers
    var numericValues = msg.payload.map(function (value) {
        return parseInt(value, 10); // 10 indique la base décimale
    }).filter(function (value) {
        return !isNaN(value); // Filtrer les valeurs non numériques
    });

    // Vérifier si des valeurs numériques ont été trouvées
    if (numericValues.length > 0) {
        // Calculer la somme des valeurs numériques dans le tableau
        var sum = numericValues.reduce(function (acc, value) {
            return acc + value;
        }, 0);

        // Calculer la moyenne en divisant la somme par le nombre d'éléments
        var moyenne = sum / numericValues.length;

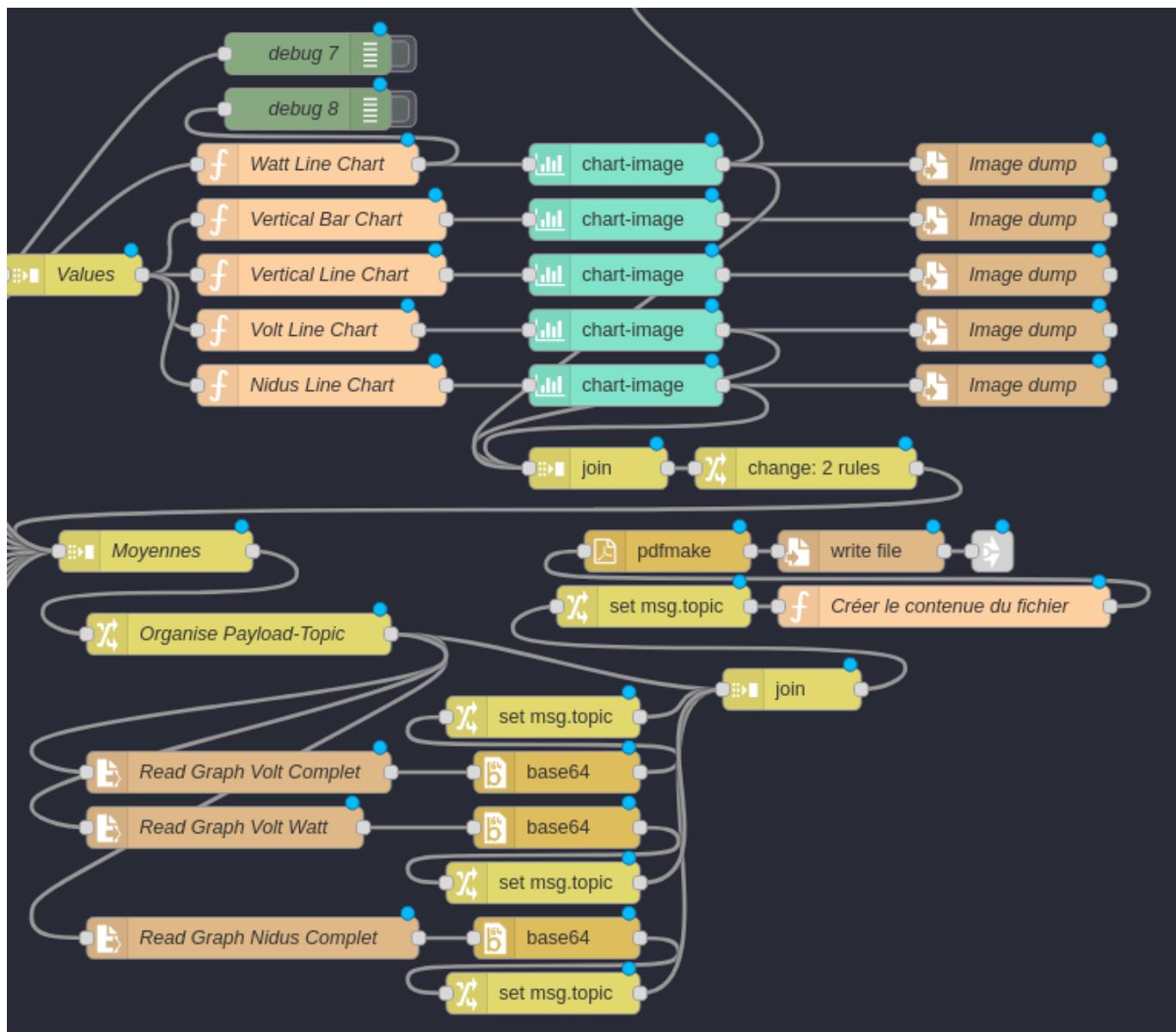
        // Arrondir la moyenne à deux chiffres après la virgule et au multiple de
        // 0.05 le plus proche
        moyenne = Math.round(moyenne * 20) / 20;

        // Ajouter la moyenne au message
        msg.moyenne = moyenne.toFixed(2);
    } else {
        // Si aucune valeur numérique n'a été trouvée, définir la moyenne à 0
        msg.moyenne = "0.00";
    }
} else {
    // Si le tableau est vide ou n'existe pas, définir la moyenne à 0
    msg.moyenne.volt.benchmark.cpu = "0.00";
}

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

La partie supérieure permet d'atteindre le même résultat à l'aide de l'INA219. Cependant, puisque je ne peux pas choisir le moment où je veux récupérer les valeurs et qu'elles sont envoyées de manière continue, j'ai utilisé une astuce consistant à détourner les messages de mise à jour de la **barre de progression**. Je les ai synchronisés avec les messages de l'INA219, puis les ai dirigés vers un nœud **join** qui les regroupe. Ensuite, ces messages sont envoyés dans un nœud **switch** qui rejette les messages ne provenant pas de la barre de progression. Cela a pour effet de ne conserver que les messages de l'INA219 pendant le test.

Une fois les ensembles de données collectés, il est temps de les utiliser :



Après l'application des fonctions **rename**, deux nœuds **join** sont utilisés pour regrouper les données. L'un regroupe les tableaux de données, tandis que l'autre regroupe les moyennes calculées.

Intéressons-nous d'abord au nœud **Values**, car c'est le premier à être utilisé. Il permet de créer un tableau de données qui est ensuite transmis à une série de nœuds de fonctions. Ces nœuds de fonctions traitent les données et les formatent pour créer des **graphiques linéaires** sous forme d'images PNG :

```
// Données reçues du flux précédent
var rawData = msg.payload;
var delayInSeconds = msg.delay / 1000; // Conversion en secondes

// Extraction des données nécessaires
var voltWatt = rawData;
// Création du graphique
var chartData = {
  type: 'line', // Changement du type de graphique en "line"
  options: {
    title: {
      display: true,
      text: 'Comparaison des performances'
    },
    legend: {
      display: true
    }
  }
}
```

```

    },
    chartArea: {
        backgroundColor: '#d3d7dd'
    },
    plugins: {
        datalabels: {
            display: false // Désactiver l'affichage des étiquettes de
données
        }
    },
    data: {
        labels: Array.from({ length: voltWatt.length }, (_, i) => (i *
delayInSeconds).toFixed(1)), // Temps en secondes
        datasets: [
            {
                label: "Volt Watt",
                borderColor: 'rgba(0, 255, 255, 1)',
                fill: false,
                data: voltWatt,
                pointRadius: 0,
            },
        ],
    },
};

msg.payload = chartData;

return msg;

```

L'exemple ci-dessus est volontairement plus simple, car il ne contient qu'un seul ensemble de données, à savoir les watts de Volt.

À la sortie de cette fonction, un nœud utilise ce qui a été créé pour générer un **tampon PNG**. Ce tampon est ensuite transmis à un nœud **write file** qui écrit le fichier dans le dossier spécifié par le nœud **Ajoute le nom du fichier**, situé après le formulaire. Simultanément, le nœud envoie également le tampon à un nœud **join** qui attend que tous les graphiques soient créés pour qu'ils puissent être réutilisés.

Une fois que le signal indiquant que les fichiers ont été créés est reçu, le nœud **join** appelé **Moyenne** peut transmettre ses données. Ces données sont réorganisées par un nœud **change**, puis envoyées à plusieurs autres nœuds pour récupérer les images en base64 des graphiques. Ces images sont envoyées en même temps que les moyennes à la fonction **Créer le contenu du fichier** :

```

// Cette section crée un objet payload qui sera utilisé pour générer un rapport
PDF.

msg.payload = {
    // Header du rapport PDF
    header: function (currentPage, pageCount, pageSize) {
        return [
            {
                text: "Tobler Cyril", // Nom de l'auteur du rapport
                alignment: "left", // Alignement du texte à gauche
                fontSize: 10, // Taille de la police 10
                margin: [15, 10, 0, 0] // Marge (haut, droite, bas, gauche)
            },
            {
                text: "Nom du projet : Confuse T-Rex", // Nom du projet
                alignment: "center", // Alignement du texte au centre
                fontSize: 10, // Taille de la police 10
                margin: [0, 0, 0, 0] // Pas de marge
            }
        ];
    }
};

```

```
        }
    ],
// Footer du rapport PDF
footer: function (currentPage, pageCount) {
    return {
        columns: [
            {
                text: currentPage.toString() + " / " + pageCount, // Numéro de page actuel / Nombre total de pages
                alignment: "left",
                fontSize: 10,
                margin: [15, 0, 0, 10]
            },
            {
                text: new Date().toLocaleDateString("fr-FR"), // Date actuelle au format français
                alignment: "right",
                fontSize: 10,
                margin: [0, 0, 15, 10]
            }
        ],
        margin: [0, 0, 0, 10] // Marge du footer
    };
},
// Contenu du rapport PDF
content: [
    {
        text: "Rapport d'utilisation", // Titre du rapport
        style: "header", // Style de texte "header" défini ci-dessous
        margin: [0, 10, 0, 0] // Marge du titre
    },
    {
        text: "Les valeurs sont des moyennes sur les " + Math.floor(msg.delay / (1000 * 60)) + " dernières minutes" // Informations sur les valeurs
    },
    {
        text: "Nidus :",
        style: "header2"
    },
    {
        text: "CPU : " + msg.payload.moyenne["nidus/benchmark/cpu"]
    },
    {
        text: "RAM : " + msg.payload.moyenne["nidus/benchmark/ram"]
    },
    {
        text: "Nombre de processus : " + msg.payload.moyenne["nidus/benchmark/processes"]
    },
    {
        text: "Température CPU : " + msg.payload.moyenne["nidus/benchmark/temp"]
    },
    {
        image: 'nidusImage',
        width: 500,
        pageBreak: 'after'
    },
    {
        text: "Volt",
        style: "header2"
    },
    {
        text: "CPU : " + msg.payload.moyenne["volt/benchmark/cpu"]
```

```

    },
    {
        text: "RAM : " +
msg.payload.moyenne["volt/benchmark/ram"]
    },
    {
        text: "Nombre de processus : " +
msg.payload.moyenne["volt/benchmark/processes"]
    },
    {
        text: "Température CPU : " +
msg.payload.moyenne["volt/benchmark/temp"]
    },
    {
        text: "MilliWatt : " +
msg.payload.moyenne["volt/benchmark/watt"]
    },
    {
        image: 'voltImage',
        width: 500
    },
    {
        image: 'wattImage',
        width: 500
    },
],
// Images à inclure dans le rapport
images: {
    voltImage: 'data:image/png;base64,' +
msg.payload.voltGraph.toString('base64'), // Image Volt au format base64
    wattImage: 'data:image/png;base64,' +
msg.payload.wattGraph.toString('base64'), // Image Watt de Volt au format base64
    nidusImage: 'data:image/png;base64,' +
msg.payload.nidusGraph.toString('base64'), // Image Nidus au format base64
},
// Styles de texte personnalisés
styles: {
    header: {
        fontSize: 22, // Taille de la police 22
        bold: true, // Texte en gras
        margin: [0, 30, 0, 0] // Marge du titre
    },
    header2: {
        fontSize: 18, // Taille de la police 18
        bold: true, // Texte en gras
        margin: [0, 20, 0, 0] // Marge des sous-titres
    }
}
};

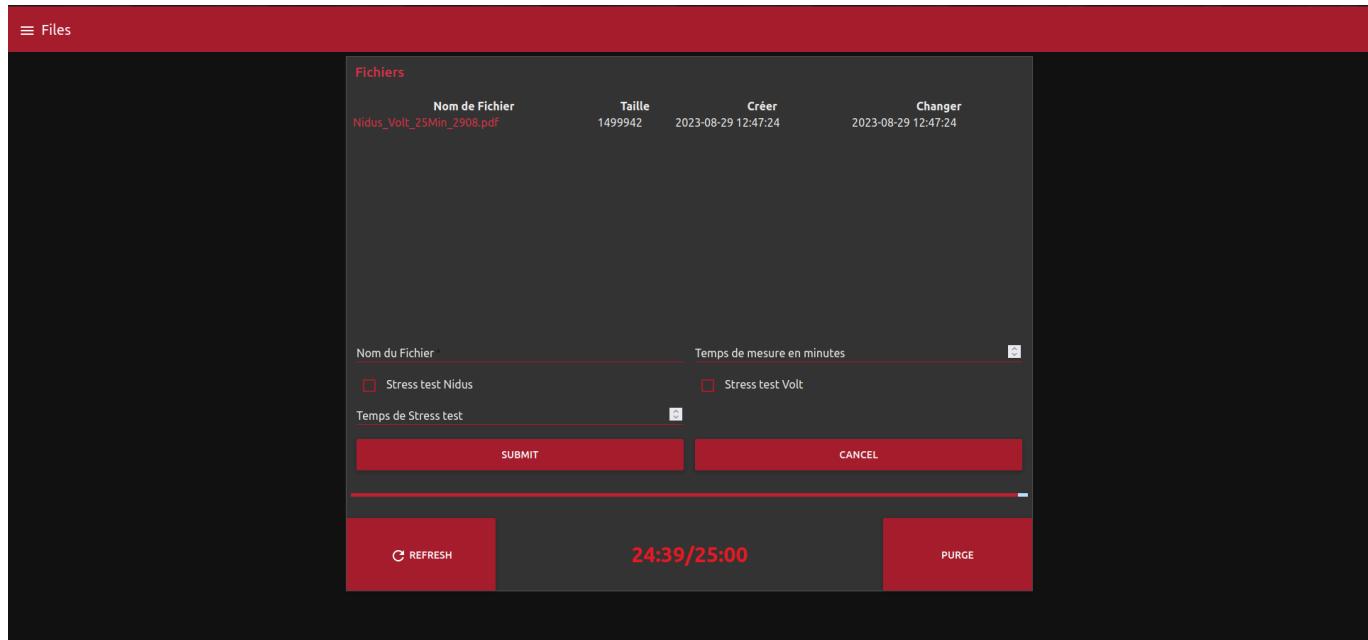
// Renvoie le message avec le payload généré
return msg;

```

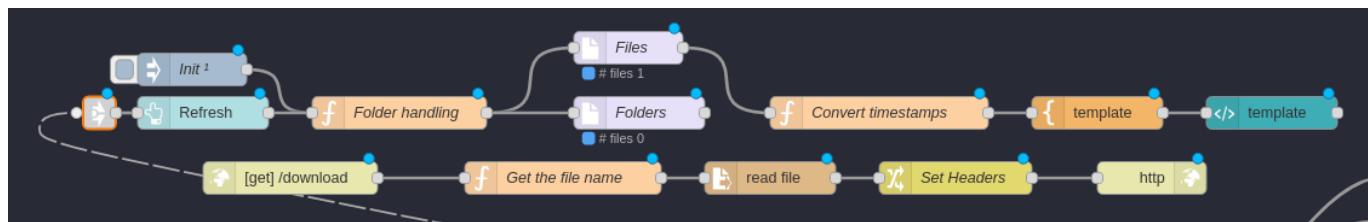
Cette fonction va créer, de manière similaire aux graphiques, une structure utilisée par PDFMake pour générer un fichier PDF. Cette structure est ensuite transmise à un nœud **pdfmake**, qui la convertit en base64 et l'envoie à un nœud **write file**. Ce dernier écrit le fichier PDF dans le dossier spécifié par le nœud **Ajoute le nom du fichier**, situé après le formulaire.

Le nœud final permet de mettre à jour le modèle HTML qui répertorie les fichiers PDF et PNG dans le dossier défini par le nœud **Ajoute le nom du fichier**. Ce modèle HTML permet de les télécharger en un seul clic.

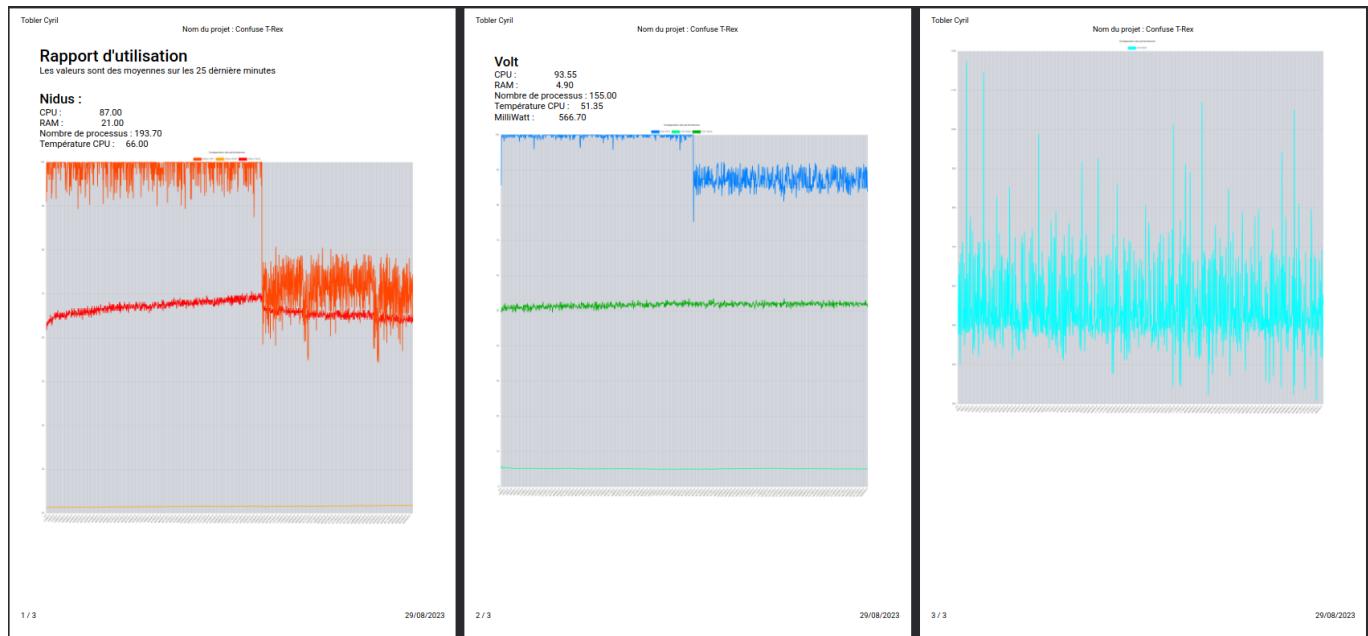
18.3. Résultat



Pour obtenir les résultats, il suffit de cliquer sur le nom du fichier, qui sera automatiquement téléchargé. Ce processus est géré par ces nœuds :

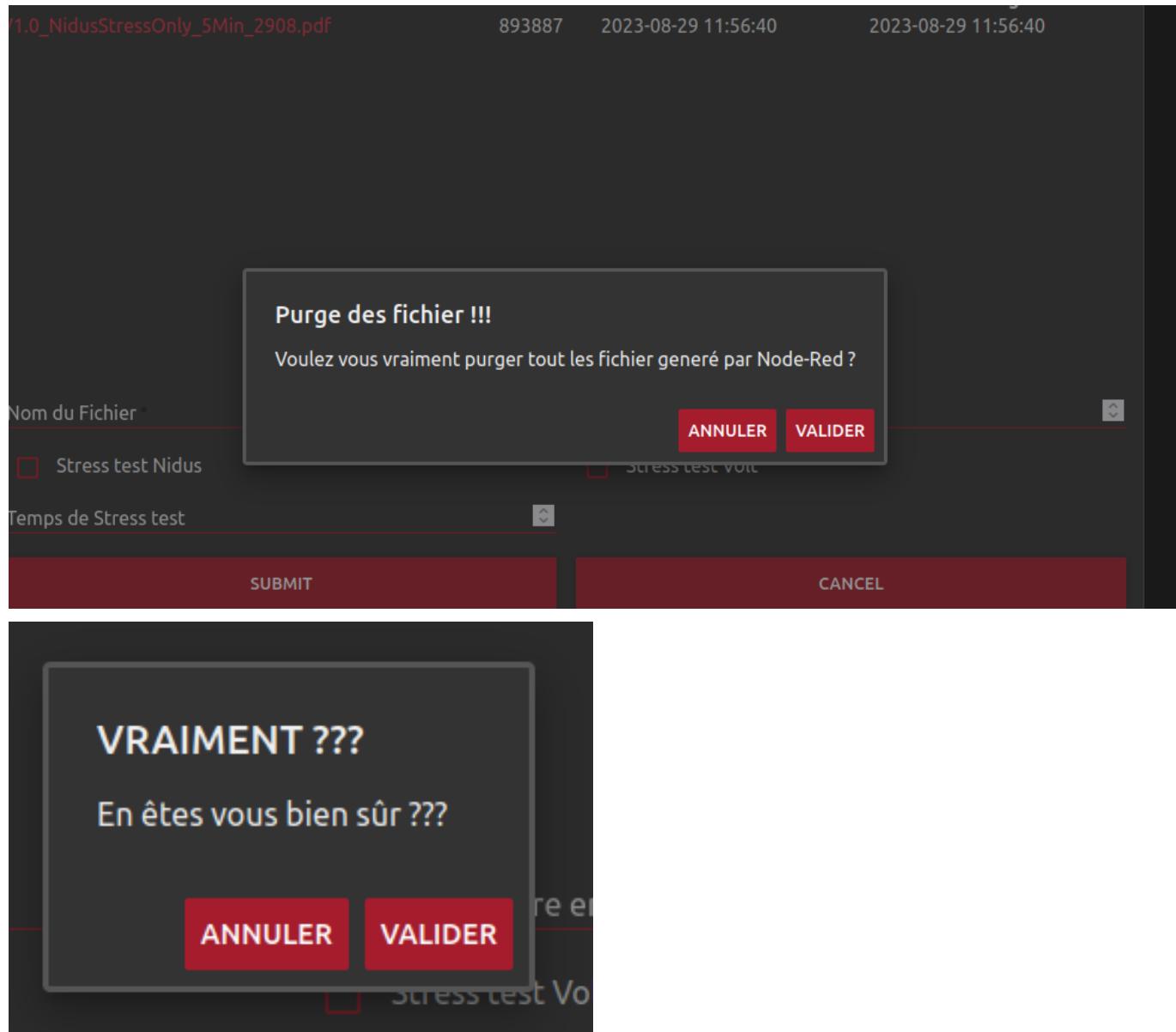


La partie supérieure gère l'affichage des fichiers dans un modèle et ajoute aux noms de fichier des requêtes GET qui permettent de télécharger les fichiers en un seul clic. La partie inférieure gère la réception des requêtes GET et envoie le fichier demandé à un nœud **read file**, qui le lit et l'envoie ensuite à un nœud **http response**. Ce dernier envoie le fichier au client ayant effectué la requête.

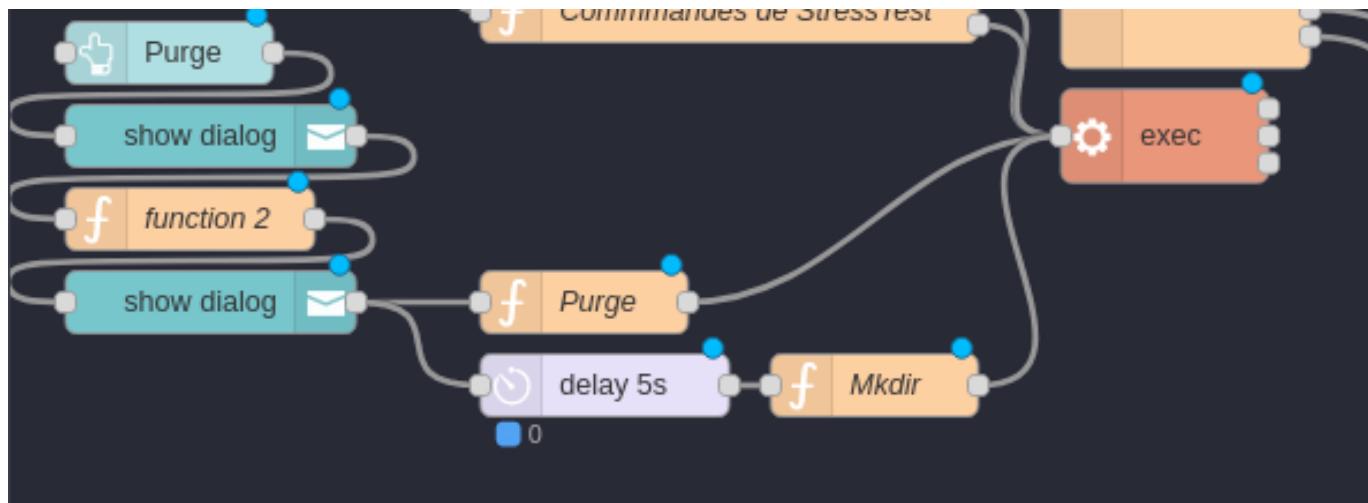


18.4. Purge

Au cours de mes tests, j'ai réalisé qu'un problème survient lorsque l'on génère un certain nombre de rapports, le dossier devient rapidement surchargé. Par conséquent, j'ai décidé de mettre en place un bouton permettant de purger le dossier de tous les fichiers .pdf et .png qui s'y trouvent. Cependant, pour éviter toute suppression accidentelle de fichiers importants, j'ai mis en place un système de confirmation demandant à l'utilisateur s'il est sûr de vouloir supprimer les fichiers.



Voici les nœuds qui gèrent cette partie :



Ce que l'on peut observer, c'est qu'après avoir appuyé sur le bouton de purge, un message est envoyé dans un nœud `show dialog` qui affiche une fenêtre de confirmation. Si l'utilisateur appuie sur le bouton "Oui", un message est transmis à un nœud de fonction qui vérifie le contenu du message et redemande une confirmation s'il est à nouveau validé. À ce stade, deux flux sont créés :

- Le premier effectue la purge totale de tous les fichiers dans `/home/NodeRed/`.
- Le second commence par un délai de quelques secondes avant de recréer les dossiers de structuration.

19. Gatling Test V2.0

19.1. But

L'objectif de cette étape est d'intégrer Gatling aux tests de Node-Red, offrant ainsi la possibilité de réaliser à la fois des tests de charge et des tests de stress sur la même infrastructure.

19.2. Étapes à Atteindre

1. **Exécution d'un Test Préétabli sur Gatling depuis Node-Red** : La première étape consiste à configurer et à exécuter un test préétabli à l'aide de Gatling directement depuis l'environnement Node-Red. Cela permettra de lancer les scénarios de test sur l'application ou le système cible.
2. **Récupération des Résultats de Gatling et Création de Graphiques pour l'Incorporation au PDF** : Une fois le test Gatling terminé, nous devrons récupérer les résultats générés par Gatling. Ces résultats seront ensuite transformés en graphiques informatifs pour être intégrés dans le rapport PDF. Cette étape vise à rendre les données de performance facilement compréhensibles.
3. **Définition de la Durée du Test Gatling depuis Node-Red** : Pour chaque test Gatling, il sera nécessaire de définir la durée de l'exécution du test directement depuis Node-Red. Cela permettra de personnaliser la durée des tests en fonction des exigences du projet.

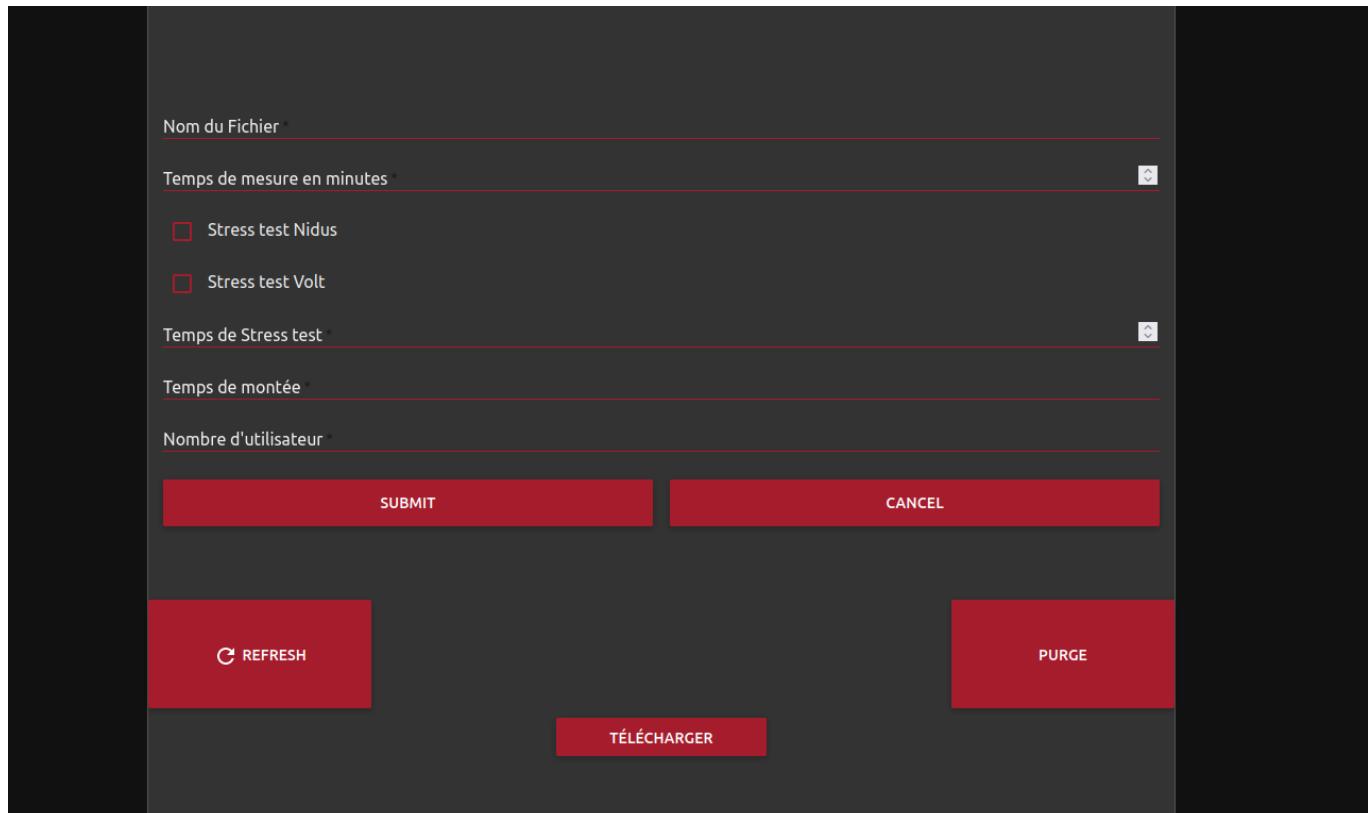
Cette intégration de Gatling aux tests Node-Red offre un moyen puissant d'évaluer les performances de l'infrastructure tout en maintenant un contrôle complet sur les scénarios de test et les paramètres de durée.

19.3. Exécution d'un Test Préétabli sur Gatling depuis Node-Red

L'exécution d'un test préétabli sur Gatling depuis Node-Red revêt une importance cruciale pour la suite du projet. Cela permettra de lancer les scénarios de test sur l'application ou le système cible. Pour réaliser cette étape, nous allons utiliser le nœud **exec** de Node-Red, qui nous permet d'exécuter des commandes sur le système d'exploitation. Ce nœud sera utilisé pour lancer les commandes Gatling nécessaires afin de lancer les tests.

Je lui transmets la commande suivante en entrée :

```
tobby@Nidus:~ $ /home/tobby/.gatling/gatling-charts-highcharts-bundle-3.9.5/bin/gatling.sh -s CuriusTREx -bm --run-mode local -erjo " -Dusers=5 -Dramp=5"
```



Nom du Fichier *

Temps de mesure en minutes *

Stress test Nidus
 Stress test Volt

Temps de Stress test *

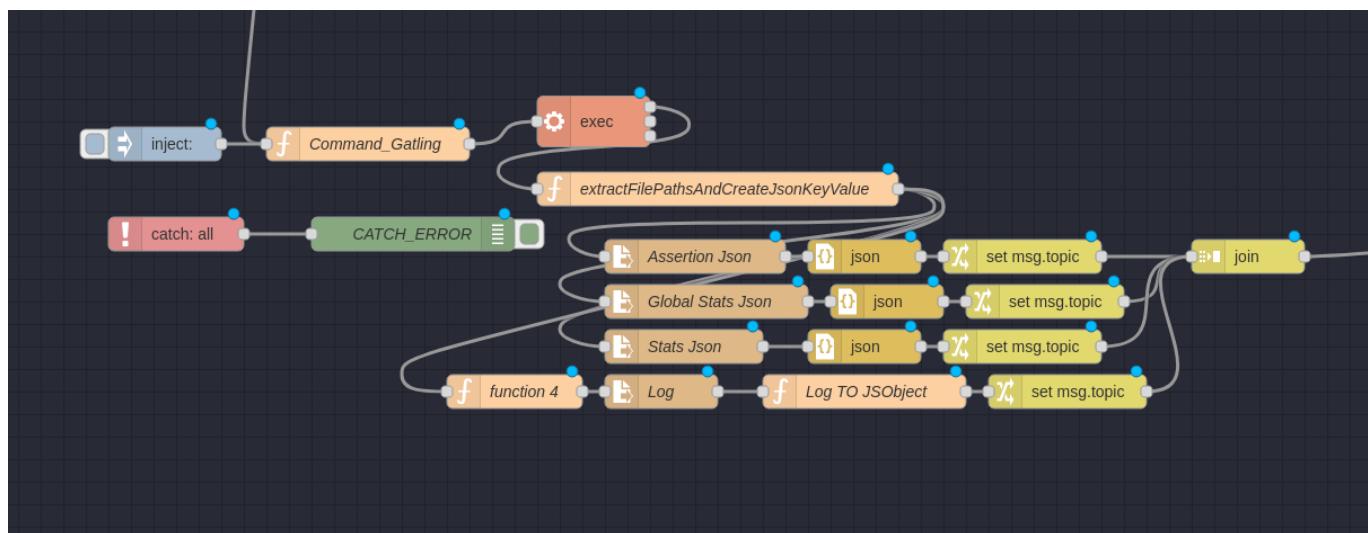
Temps de montée *

Nombre d'utilisateur *

SUBMIT CANCEL

REFRESH PURGE

TELECHARGER



19.4. Envoi de Commande avec une Durée

L'une des premières étapes essentielles est de pouvoir définir une durée pour le test. Pour accomplir cette tâche, j'ai employé un nœud **function** qui permet de spécifier la durée du test en fonction de la valeur saisie par l'utilisateur. Cette valeur est ensuite transmise à un nœud **exec** qui se charge d'exécuter la commande Gatling nécessaire. Voici le code de la fonction :

```
msg.original = msg.payload;
if (msg.topic !== "inject" && msg.payload.time_gatling !== undefined &&
msg.payload.user_gatling !== undefined) {
    // Crée la commande en utilisant les valeurs spécifiées
    msg.payload.time = msg.payload.time_gatling * 60;
    msg.payload = `./home/tobby/.gatling/gatling-charts-highcharts-bundle-
3.9.5/bin/gatling.sh -s CuriusTREx -bm --run-mode local -erjo "-
Dusers=${msg.payload.user_gatling} -Dramp=${msg.payload.time_gatling}"`;
} else {
    // Utilise la commande d'origine car il ne s'agit pas d'un noeud inject ou
    // les valeurs ne sont pas fournies
    // Par défaut, utilisez la commande actuelle
    msg.payload = "/home/tobby/.gatling/gatling-charts-highcharts-bundle-
3.9.5/bin/gatling.sh -s CuriusTREx -bm --run-mode local -erjo \" -Dusers=5 -
Dramp=5\"";
}

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

19.5. Récupération des Informations

Pour obtenir les informations d'un rapport Gatling, j'ai dû entreprendre une rétro-ingénierie significative. En effet, Gatling ne propose pas de mécanisme direct pour extraire des informations depuis la ligne de commande. Il est nécessaire de collecter les données directement à partir des fichiers générés par Gatling. Voici un exemple de fichier généré par Gatling :

```
RUN CuriusTREx_Bash curiustrex-bash 1693898386165      3.9.5
USER   CuriusTREx START  1693898386953
REQUEST request_0  1693898386949  1693898386985  OK
REQUEST request_1  1693898387009  1693898387014  OK
USER   CuriusTREx START  1693898387108
REQUEST request_0  1693898387107  1693898387128  OK
REQUEST request_1  1693898387132  1693898387136  OK
REQUEST styles.css 1693898387162  1693898387167  OK
REQUEST styles.css 1693898387162  1693898387167  OK
REQUEST request_2  1693898387184  1693898387189  OK
REQUEST request_2  1693898387184  1693898387189  OK
REQUEST styles.css 1693898387211  1693898387222  OK
REQUEST Home.jpg   1693898387212  1693898387228  OK
REQUEST styles.css 1693898387223  1693898387233  OK
REQUEST Données.jpg 1693898387213  1693898387234  OK
REQUEST Home.jpg   1693898387224  1693898387239  OK
REQUEST Test_Complet.jpg 1693898387215  1693898387241  OK
```

On peut donc voir qu'il y a **beaucoup d'information à traiter dans ce cas particulier**, le test dure 30 secondes avec 5 utilisateurs, et le fichier **simulation.log** fait **3114 lignes**.

Étant donné que les informations sont sauvegardées à chaque fois dans un fichier différent, j'ai mis en place une fonction dans un nœud qui me permet de récupérer le nom du dossier dans lequel le rapport et les logs sont sauvegardés.

Ensuite, j'en fais un tableau de chemins de fichiers :

```
// Fonction pour extraire le chemin après "file://"
var inputPayload = msg.payload;

// Recherche de l'index de "file://"
var fileIndex = inputPayload.indexOf("file://");

// Si "file://" est trouvé, extrayez le chemin après
if (fileIndex !== -1) {
    // Extraire le chemin après "file://"
    var filePath = inputPayload.substring(fileIndex + 7); // +7 pour sauter
    "file://"

    // Supprimer le fichier index.html du chemin, s'il est présent
    filePath = filePath.substring(0, filePath.length - 12); // -11 pour
    enlever "/index.html"

    // Mettre le chemin extrait dans le message de sortie
    msg.payload = filePath;

    // Tableau de noms de fichiers .json
    var jsonFilesList = [
        "assertions.json",
        "global_stats.json",
        "stats.json"
    ];

    // Créer un tableau de clé-valeurs pour les fichiers .json
    var jsonFilesKeyValue = {};

    // Parcourir la liste des noms de fichiers et construire les chemins complets
    for (var i = 0; i < jsonFilesList.length; i++) {
        var jsonFileName = jsonFilesList[i];
        var jsonFilePath = filePath + '/js/' + jsonFileName;
        jsonFilesKeyValue[jsonFileName] = jsonFilePath;
    }

    // Ajouter le tableau de clé-valeurs au message
    msg.jsonFiles = jsonFilesKeyValue;

    // Renvoyer le message modifié
    return msg;
} else {
    // Aucun "file://" trouvé, renvoyer le message d'origine
    return msg;
}
```

Tout cela se fait à la sortie du nœud exec, lequel renvoie les informations suivantes :

```
[...]
=====
2023-09-05 11:05:03                                     29s elapsed
---- Requests -----
[...]
---- CuriusTREx -----
[#####] 100%
    waiting: 0      / active: 0      / done: 150
=====
Simulation CuriusTREX_Bash completed in 29 seconds
Parsing log file(s)...
Parsing log file(s) done
Generating reports...

=====
---- Global Information -----
[...]
---- Response Time Distribution -----
> t < 800 ms                                         2816 (100%)
> 800 ms <= t < 1200 ms                           0 (  0%)
> t >= 1200 ms                                      0 (  0%)
> failed                                            0 (  0%)
=====
Reports generated in 0s.
Please open the following file:
file:///home/toblerc/T%C3%A9l%C3%A9chargements/gatling-charts-highcharts-bundle-3.9.5/results/curiustrex-bash-20230905090433239/index.html
```

19.6. Traitement des données

Une fois les chemins des fichiers définis, je lis les fichiers JSON et les transmets à un nœud **JSON** chargé de les traiter et de les renvoyer dans un format exploitable par Node-Red.

En sortie, j'assemble les quatre fichiers en un seul message que j'envoie ensuite aux fonctions de création de graphiques.

19.6.1. Création des graphiques

Je crée deux graphiques, l'un utilisant le pourcentage de réussite et d'échec pour générer un bar chart :

```
// Données reçues du flux précédent
var rawData = msg.payload.gatling.Global; // Utilisez les données de
msg.payload.gatling.Global
msg.topic = "VoltBarChart"; // Définir le sujet du message

// Extraction des données nécessaires
var groupData = [
  rawData.group1,
  rawData.group2,
  rawData.group3,
  rawData.group4
];

var labels = [];
var percentages = [];

// Parcourir les données des groupes et extraire les informations
for (var i = 1; i <= groupData.length; i++) {
  var groupName = "group" + i;
  labels.push(rawData[groupName].htmlName);
  percentages.push(rawData[groupName].percentage);
}

// Crédit au graphique
var chartData = {
  type: 'bar', // Utiliser un graphique de type "bar"
  options: {
    title: {
      display: true,
      text: 'Répartition des performances par groupe' // Titre du graphique
    },
    chartArea: {
      backgroundColor: '#d3d7dd' // Couleur de fond du graphique
    },
    scales: {
      x: {
        beginAtZero: true,
        title: {
          display: true,
          text: 'Groupes' // Titre de l'axe des X
        }
      },
      y: {
        beginAtZero: true,
        title: {
          display: true,
          text: 'Pourcentage' // Titre de l'axe des Y
        }
      }
    }
},
```

```

plugins: {
    datalabels: {
        anchor: 'end',
        align: 'top',
        formatter: function (value) {
            return value + '%'; // Ajouter le symbole de pourcentage à
l'étiquette
        }
    }
},
data: {
    labels: labels,
    datasets: [
        {
            label: "Pourcentage", // Légende de l'ensemble de données
            backgroundColor: 'rgba(0, 128, 255, 0.7)', // Couleur du
remplissage des barres
            data: percentages
        }
    ]
}
};

msg.payload = chartData; // Attribuer les données du graphique au message
return msg; // Renvoyer le message

```

Le deuxième calcule le temps de réponse de chaque requête afin de créer un graphique linéaire :

```

// Assurez-vous que msg.payload.gatling.Log est un tableau
var logEntries = Array.isArray(msg.payload.gatling.Log) ? msg.payload.gatling.Log
: [];

// Création du graphique
var chartData = {
    type: 'line', // Utilisez un graphique de type "line"
    options: {
        title: {
            display: true,
            text: 'Comparaison des performances' // Titre du graphique
        },
        legend: {
            display: true // Affichez la légende du graphique
        },
        chartArea: {
            backgroundColor: '#d3d7dd' // Couleur de fond du graphique
        },
        plugins: {
            datalabels: {
                display: false // Désactivez l'affichage des étiquettes de
données
            }
        }
    },
    data: {
        labels: logEntries.map(function (entry) {
            if (entry.type === 'RUN') {
                return 'Début de la Simulation'; // Label pour les opérations de
type "RUN"
            } else if (entry.type === 'REQUEST') {
                // Calcul de la différence en millisecondes entre le début de la
simulation et le début de la requête
                var simulationStart = logEntries.find(function (item) {
                    return item.type === 'RUN';

```

```

    });
    if (simulationStart) {
        var startTime = simulationStart.start;
        var requestTime = entry.start;
        var timeDiff = requestTime - startTime;
        return (timeDiff / 1000).toFixed(0) + ' s'; // Temps en
secondes
    } else {
        return 'N/A'; // Si aucune opération de type "RUN" n'est
trouvée
    }
} else {
    return ''; // Autres types d'opérations (USER, etc.) sans label
}
},
datasets: [
{
    label: "Temps de Réponse", // Légende de l'ensemble de données
    borderColor: 'rgba(0, 128, 255, 1)', // Couleur de la ligne du
graphique
    fill: false, // Désactivez le remplissage sous la ligne
    data: logEntries.map(function (entry) {
        if (entry.type === 'REQUEST') {
            // Calcul de la différence en millisecondes entre le
début et la fin de la requête
            var requestStart = entry.start;
            var requestEnd = entry.end;
            var requestTime = requestEnd - requestStart;
            return requestTime;
        } else {
            return 0; // Valeurs pour les opérations de type "RUN" ou
autres
        }
    }),
    pointRadius: 0, // Diamètre des points sur le graphique
}
]
},
];
msg.payload = chartData; // Attribuez les données du graphique au message
return msg; // Renvoyez le message

```

19.6.2. PDF

Une fois le chart créé, je le réutilise dans le noeud de création de PDF pour les ajouter au rapport :

```

/**
 *
 * Vu précédemment
 *
 */
{
    text: "Gatling", // Section Gatling
    style: "header2"
},
{
    text: "Nom de la simulation : " +
msg.payload.gatling.Assertion.simulation // Nom de la simulation Gatling
},
{
    text: "Commencée : " +
msg.payload.gatling.Assertion.start // Date de début de la simulation Gatling (à
formater)

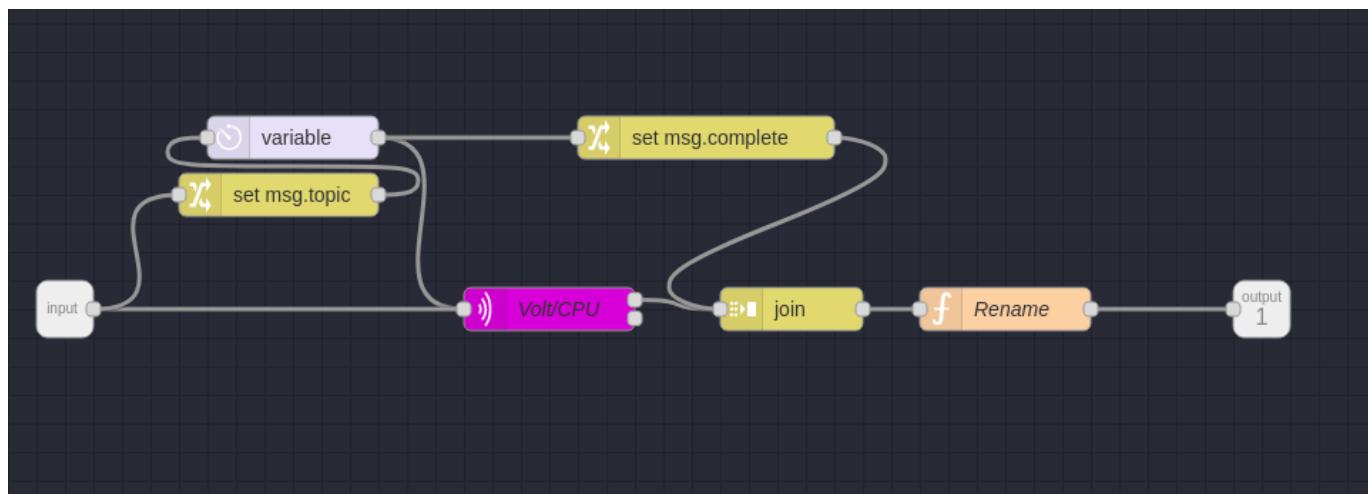
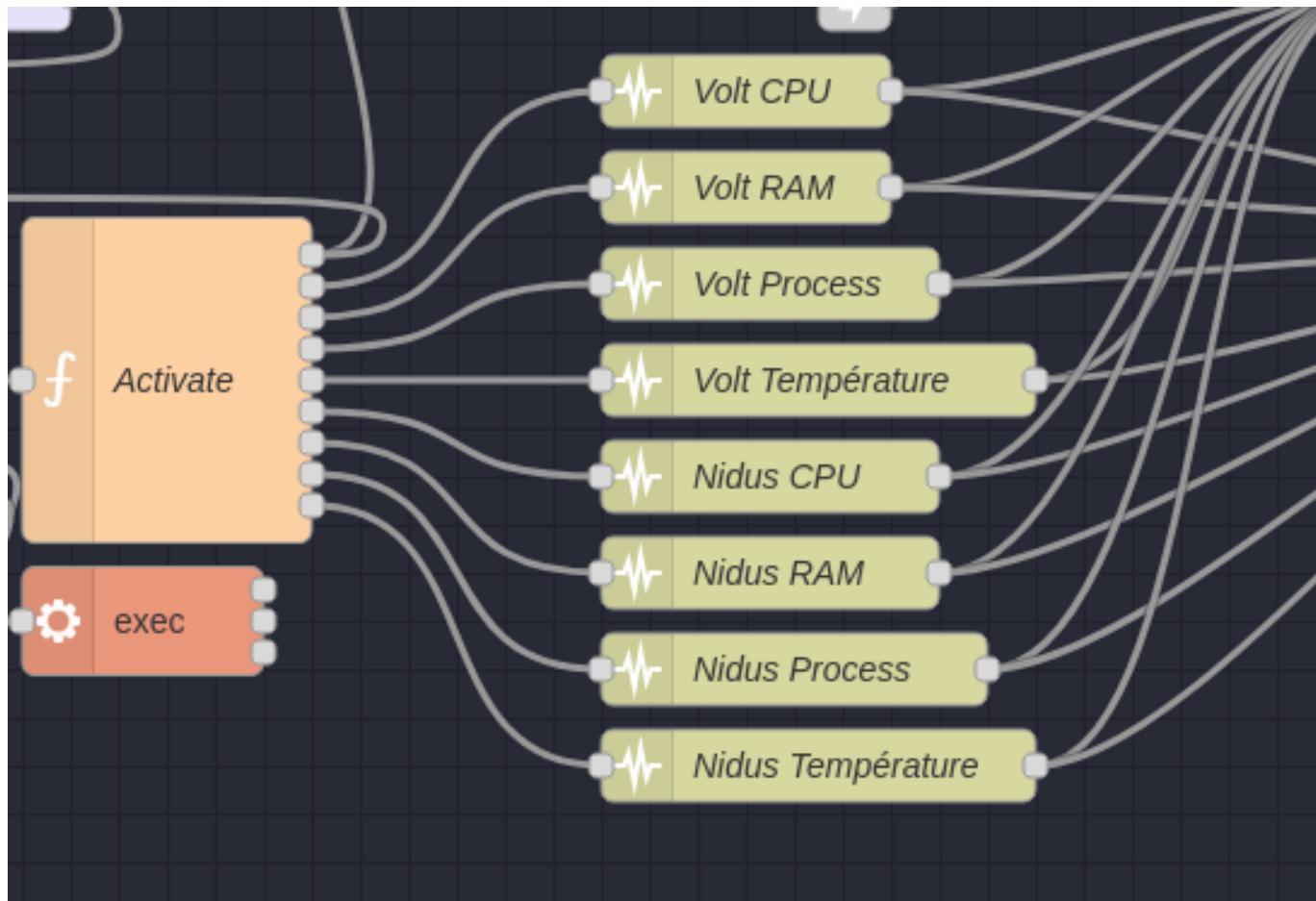
```

```
},
{
    image: 'gatlingImage', // Image Gatling
    width: 500,
},
{
    image: 'gatlingGraphComplete', // Image graphique complet Gatling
    width: 500,
},
],
images: {
    voltImage: 'data:image/png;base64,' +
msg.payload.voltGraph.toString('base64'), // Utilisation du buffer pour l'image Volt
    wattImage: 'data:image/png;base64,' +
msg.payload.wattGraph.toString('base64'), // Utilisation du buffer pour l'image Watt de Volt
    nidusImage: 'data:image/png;base64,' +
msg.payload.nidusGraph.toString('base64'), // Utilisation du buffer pour l'image Nidus
    gatlingImage: 'data:image/png;base64,' +
msg.payload.gatlingGraph.toString('base64'), // Utilisation du buffer pour l'image Gatling
    gatlingGraphComplete: 'data:image/png;base64,' +
msg.payload.gatlingGraphComplete.toString('base64'), // Utilisation du buffer pour l'image Gatling
},
/**
 *
 * Vu precedemment
 */

```

19.7. Refactoring

Pour finir, j'ai créer des subflow pour le noeud MQTT afin de simplifier la visualisation et le management des données :



20. Gatling V3.0

Maintenant que l'étape de "PoC" a été atteinte, il est temps de passer à la version utilisable du projet. À partir de Node-Red, vous pouvez désormais réaliser les actions suivantes :

- Lancer des StressTests
- Lancer des tests Gatling
- Récupérer les informations de monitoring (CPU, RAM, Température, etc.)
- Récupérer les informations de Gatling (Temps de réponse, pourcentage de réussite, etc.)
- Joindre les informations pour créer des graphiques
- Regrouper tout pour créer un rapport PDF

Les objectifs à atteindre sont donc les suivants :

- Passer des paramètres de test à Gatling depuis Node-Red
- Intégrer le CSV de Gatling avec Node-Red
- Améliorer l'interface utilisateur de Node-Red
- Ajouter les valeurs souhaitées dans le rapport PDF
- Modifier les graphiques pour une meilleure lisibilité
- Ajouter la possibilité d'exporter les données au format JSON
- Ajouter la possibilité d'importer les données au format JSON et de générer un PDF

20.1. UI

Pour l'interface utilisateur (UI), j'ai choisi de limiter le temps sélectionné à la durée du test. Ce choix s'explique par le fait qu'en principe, la durée des mesures devrait toujours correspondre à la durée de Gatling. Cependant, je laisse toujours la possibilité de définir la durée du StressTest indépendamment de Gatling.

J'ai ajouté une entrée pour définir si un ventilateur est activé. Actuellement, avec des températures proches de 30 degrés à l'extérieur, le ventilateur est nécessaire pour éviter la surchauffe du Raspberry Pi. Sachant cela, je voulais avoir la possibilité de le notifier dans le rapport.

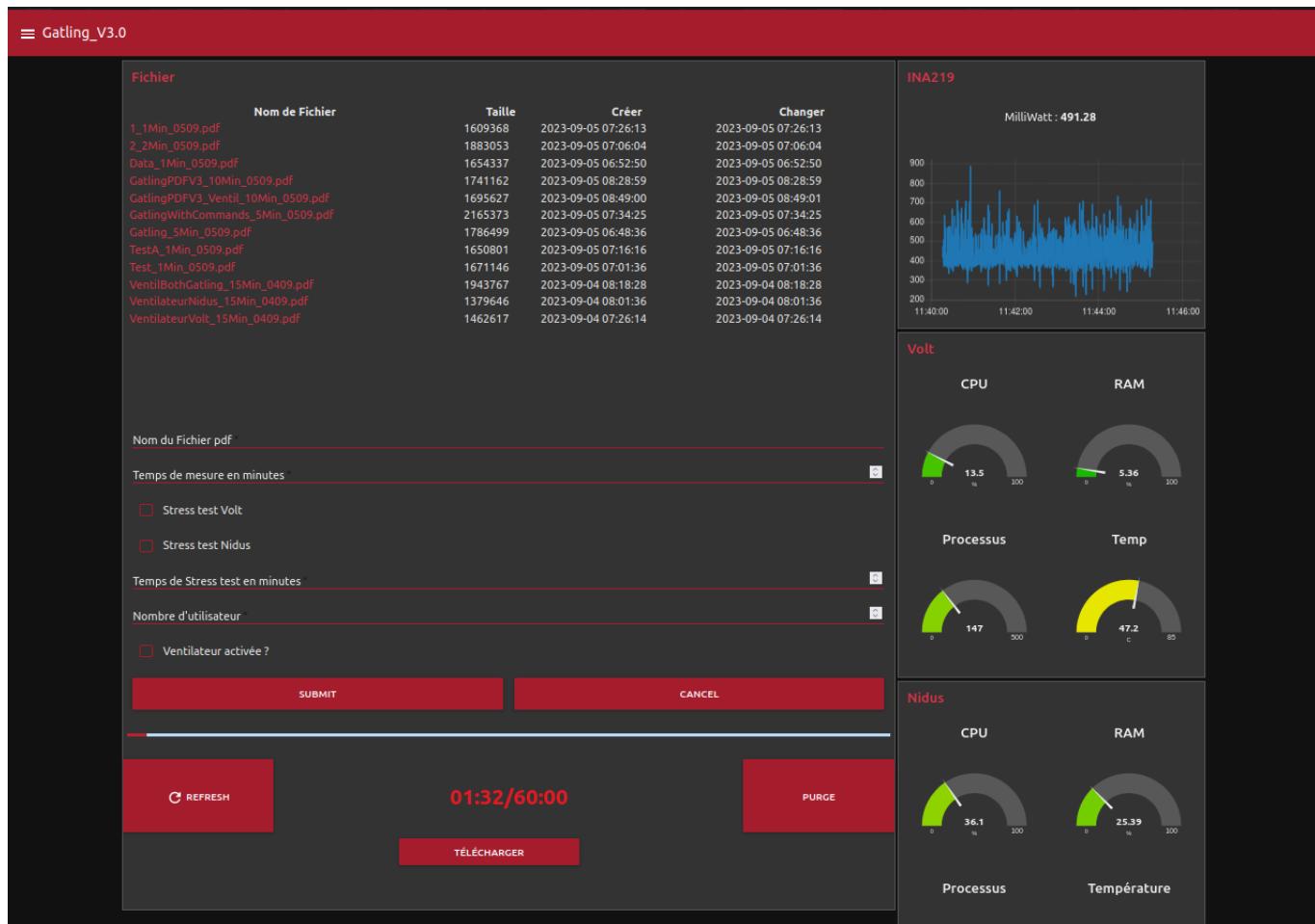
À l'heure actuelle, le ventilateur est un ventilateur de bureau pris dans mon stock personnel, actionné par un bouton. Il n'est donc pas possible de le contrôler depuis Node-Red. Cependant, il est possible qu'à l'avenir, un ventilateur contrôlable soit utilisé, auquel cas il sera possible de l'activer ou de le désactiver depuis Node-Red.

Pour permettre de garder un certain contrôle, j'ai ajouté les informations de monitoring directement à côté du formulaire de lancement du test. Ainsi, il est possible de voir les valeurs de monitoring en temps réel et de surveiller le déroulement du test.

20.1.1. Ventilateur



20.1.2. UI



21. Remerciement

Je tiens à exprimer ma profonde gratitude envers les personnes qui ont joué des rôles essentiels dans la réalisation de ce projet. Avant tout, je souhaite exprimer ma sincère reconnaissance à M. Benoit Vianin, dont la proposition du projet, le matériel fourni et les conseils avisés ont été cruciaux pour sa mise en place. Sa précieuse assistance technique a été d'une grande importance.

Je tiens également à adresser mes remerciements à M. Fabien Maire, Directeur du service Informatique du SIS2 (Service Informatique du Secondaire 2), pour son accompagnement et ses conseils tout au long de ce travail. Sa vision éclairée et son expertise ont été des facteurs clés dans la réussite de ce projet.

Un remerciement spécial s'adresse à M. Christophe Singele, enseignant en microtechnique au CPNE-TI, pour son soutien inestimable dans la compréhension des schémas électriques et dans la résolution des problèmes de câblage et de connectique. Sa patience et ses connaissances ont été une source inestimable d'aide.

Je saisirai également cette occasion pour exprimer ma gratitude envers l'équipe du SIS2 pour son accueil chaleureux dans leurs locaux et pour le soutien constant qu'ils m'ont apporté tout au long de ce travail. Leur environnement de travail a été propice à la réalisation de ce projet.

Je remercie sincèrement M. Patrice Lüthi, enseignant en informatique au CPNE-TI, pour sa contribution essentielle dans la mise en place de la communication entre l'INA219 et les Raspberry Pi. Son expertise technique a joué un rôle clé dans la résolution des défis techniques.

Enfin, Un remerciement tout particulier à ma mère et à ma copine pour leur soutien infaillible durant ma formation, qui m'a conduit à accomplir ce travail. Leur encouragement constant et leur confiance en moi ont été des sources d'inspiration essentielles.

Ces individus exceptionnels ont joué un rôle capital dans la réalisation de ce projet. Leur soutien, leur expertise et leur encouragement ont été essentiels, et je leur suis profondément reconnaissant.

22. Sources

1. **Guide d'Installation Node-Red**
Installer Node-Red
2. **Guide de Sécurisation de Node-Red**
Sécurisation de Node-Red
3. **Tutoriel de Base Rototron**
Tutoriel Rototron
4. **Documentation Technique de l'INA219**
Documentation INA219
5. **Recherche de M. Lamber**
Profil de Consommation par M. Lamber
6. **Recherche de M. Pol J. Planas Pulido**
Profil de Consommation par M. Pol J. Planas Pulido
7. **Bibliothèque Python pour l'INA219**
Bibliothèque pi-ina219
8. **Forum Problème de Détection I2C**
Forum Raspberry Pi
9. **Tutoriel Mise en Place INA219**
Tutoriel INA219
10. **Tutoriel Crédit d'un Enregistreur de Consommation**
Tutoriel Enregistreur de Consommation
11. **Tutoriel Complet avec Arduino**
Tutoriel Complet avec Arduino
12. **Téléchargement Gatling**
Téléchargement Gatling
13. **Tutoriel Avancé Gatling**
Tutoriel Avancé Gatling
14. **Tutoriel de Démarrage Rapide Gatling**
Tutoriel de Démarrage Rapide Gatling
15. **Tutoriel sur l'utilisation de S1seven**
Tutoriel S1seven