



DIGICHRONE®
COLLABORATIVE WAY FOR INNOVATIVE PROJECTS



Réseaux
et Télécoms

iut Nord Franche-Comté

SYNCHRONISEUR SOLAIRE



Rapport de stage

9 janvier - 3 mars 2023

Présenté par

Yassine EL HAMIOUI

🌐 Stephane GIVRON - Responsable Pédagogique

🌐 Francois SPIES - Responsable Professionnel

📍 4 Place Tharradin 25200 MONTBELIARD

📍 21 rue marceau petit 21340 NOLAY

Remerciements

Je tiens à exprimer mes sincères remerciements à mon tuteur, M. François SPIES, pour son encadrement et son soutien tout au long de mon stage. Grâce à sa supervision et à ses conseils, j'ai pu réaliser mon travail dans des conditions optimales et acquérir des compétences précieuses.

Je souhaite également exprimer ma gratitude envers M. Michel STENTA, chef de l'entreprise, pour m'avoir offert cette opportunité de stage enrichissante.

Je remercie également chaleureusement M. Givron, mon tuteur académique, pour son aide et son soutien en cas de besoin, ainsi que pour le matériel fourni qui a été d'une grande utilité pour la réalisation de mon projet.

Enfin, je souhaite exprimer ma reconnaissance envers les responsables du département R&T pour m'avoir accordé une salle de réunion calme et un environnement propice pour effectuer mon stage dans les meilleures conditions.

Abstracts

The purpose of this internship report is to describe the tasks and results achieved during the period spent during the internship at the **DIGICHROME®** company.

This internship takes place during the period from **January 9 to March 3, 2023** and I was assigned to the project entitled "**Solar Synchronizer**". The aim of this project is to study and realize a current measurement solution for a solar synchronizer, based on a **Raspberry Pi** and **Home Assistant** . I worked completely **independently**, while benefiting from the advice of my tutor **Mr. Spies**, who provided me with the right leads to follow. I had the opportunity to put my theoretical knowledge into practice and develop new professional skills. The implementation steps will be described in detail, highlighting challenges and solutions. In short, this internship report reflects my career as an intern at DIGICHROME®, as well as the skills I acquired during that period.

Sommaire

1 Présentation de l'entreprise	01
1.2 Objectif du stage	01
1.3 Problématique	02
1.3.1 Présentation de la problématique	02
1.3.2 Justification de la problématique	02
2 Cadre théorique	03
2.1 État de l'art	03
2.1.1 Quelles sont les marques qui vendent des prises connectés ?	03
2.1.2 Qu'est ce qu'un synchroniseur solaire ?	03
2.1.3 Qu'est ce que Home Assistant ?	04
2.1.4 Quelles sont les normes de la domotique ?	04
3 Méthodologie et approche	05
3.1 Description de la méthodologie et des outils utilisés pour réaliser le projet	04
3.1.1 Choix de la méthodologie	05
3.1.2 Description des outils utilisés	05
3.2. Sources d'informations	06
3.2.1 Techniques et outils de collecte de données	06
3.3 Traitement de données	07
3.3.1 Méthodes et outils utilisés pour traiter les données collectées	07
4 Présentation du projet	08
4.1 Cahier des charges du projet de stage	08
4.2 Étapes de réalisation du projet	08
4.2.1 Home Assistant	09
4.2.1.1 Mise en place de Home Assistant	09-10-11
4.2.1.2 Accès en SSH aux fichiers du système	12
4.2.2 Bus MQTT	13-14
4.2.2.1 ESP8266	15-16
4.2.2.2 MQTT sur Home Assistant	17-18
4.2.3 Prise connecté	19
4.2.3.1 MSS310 sur Home Assistant	20
4.2.4 Capteur de mesure de courant	21-22-23-24
4.3 Sécurisation du système	25

Sommaire

5 Résultats et analyses.....	26
5.1 Présentation des résultats obtenus suite à la réalisation du projet de stage.....	
5.2 Analyse des résultats.....	
5.2.1 Interprétation et explication des résultats obtenus.....	
5.2.2 Comparaison avec les objectifs fixés initialement.....	
6 Conclusion et perspectives.....	27
6.1 Synthèse des résultats obtenus.....	
6.2 Conclusion générale du stage.....	
6.3 Perspectives d'avenir et recommandations.....	
7 Annexes.....	28-42
7.1 Liste des annexes : Figures, images, Screenshot, documents, etc.....	
9 Bibliographie.....	43-44
9.1 Liste des références bibliographiques utilisées pour le projet de stage.....	

1 Présentation de l'entreprise DIGICHRONE®

DIGICHRONE, est une société par actions simplifiée, elle est en activité depuis 7 ans. Située à NOLAY (21340), elle est spécialisée dans le secteur d'activité du conseil en systèmes et logiciels informatiques. M. **Michel STENTA** est président de l'entreprise DIGICHRONE.

En raison de contraintes de pratique, j'ai été placé dans un lieu éloigné de l'entreprise. De plus, mon tuteur était en activité à l'université pendant la durée de mon stage, ce qui aurait probablement causé des problèmes de communication s'il avait dû se déplacer. Cependant, j'ai pu surmonter ces obstacles grâce à une salle de réunion mise à ma disposition pendant les deux mois de mon stage. J'ai partagé cette salle avec deux camarades qui étaient également en stage (sur différents projets).

Cette salle était équipée d'un écran, de câbles et de **tout le matériel nécessaire** pour mener à bien mon stage. Grâce à cet équipement, j'ai pu accomplir les tâches ainsi que communiquer avec mon tuteur à distance, ce qui a grandement facilité le déroulement de mon stage.

En somme, la mise à disposition d'une salle équipée a été une solution pratique pour surmonter les contraintes de communication et garantir le bon déroulement de mon stage.

1.2 Objectif du stage

Ce stage est une étape primordiale pour ma formation en **Réseaux & Télécommunications** en deuxième année de Bachelor. En effet, il occupe un tiers du quatrième semestre et a pour objectif principal de me permettre d'acquérir une expérience pratique dans un domaine spécifique en travaillant au sein d'une entreprise ou d'une organisation.

Ce stage offre une occasion unique d'appliquer les connaissances théoriques et pratiques acquises à l'université dans un contexte réel, en développant des **compétences professionnelles** telles que la **recherche de solutions en autonomie, la résolution de problèmes** et la **gestion de projet**. Cette expérience m'a permis de mieux comprendre certains enjeux du monde professionnel et de développer une expertise qui sera utile pour la suite de mon parcours qui peut se compléter à la spécialité **cybersécurité** choisie à l'université. En somme, ce stage représente une étape clé dans mon parcours universitaire et professionnel

1.3 Problématique

1.3.1 Présentation de la problématique

Comment améliorer l'efficacité énergétique d'un système photovoltaïque en temps réel grâce à l'intégration d'une carte de mesure de courant et d'un bus MQTT, et comment faciliter le déploiement de cette solution en l'intégrant dans une distribution Raspberry Pi ?

1.3.2 Justification de la problématique

La surveillance de la consommation d'énergie revêt une importance primordiale dans notre société actuelle. Elle permet de **réduire** les coûts liés à l'énergie, d'**optimiser** les sources d'énergie renouvelable pour protéger l'environnement, et de répondre à la demande croissante pour les systèmes **domotiques intelligents** dans les foyers.

En effet, les foyers sont aujourd'hui de plus en plus équipés de nombreux appareils électroniques et électroménagers qui consomment de l'énergie. Il est donc nécessaire de **surveiller** cette consommation afin de pouvoir l'optimiser et réduire les coûts. De plus, avec la transition énergétique vers des sources d'**énergie renouvelable**, la surveillance de la consommation d'énergie est encore plus importante pour garantir une utilisation efficace et optimale de ces sources.

Enfin, la demande croissante pour les systèmes domotiques intelligents dans les foyers rend la surveillance de la consommation d'énergie encore plus cruciale. Les systèmes domotiques intelligents permettent une **gestion automatisée de l'énergie** dans les foyers, mais cela nécessite une surveillance continue de la consommation d'énergie pour pouvoir ajuster les paramètres en temps réel.

Dans le cas du projet, le synchroniseur doit **mesurer la consommation** générale d'une habitation et la production des **panneaux solaires**. Si on **produit plus** d'électricité que l'on en consomme, on peut activer une ou plusieurs prises pour consommer l'excédent. A l'inverse, si on **consomme plus** que l'on ne produit, il faut couper une ou plusieurs prises pour moins consommer.

Les **wattmètres connectés** (prise connectée intelligente) permettent de mesurer avec précision la consommation de chaque prise pour faciliter le choix des prises à activer ou désactiver.

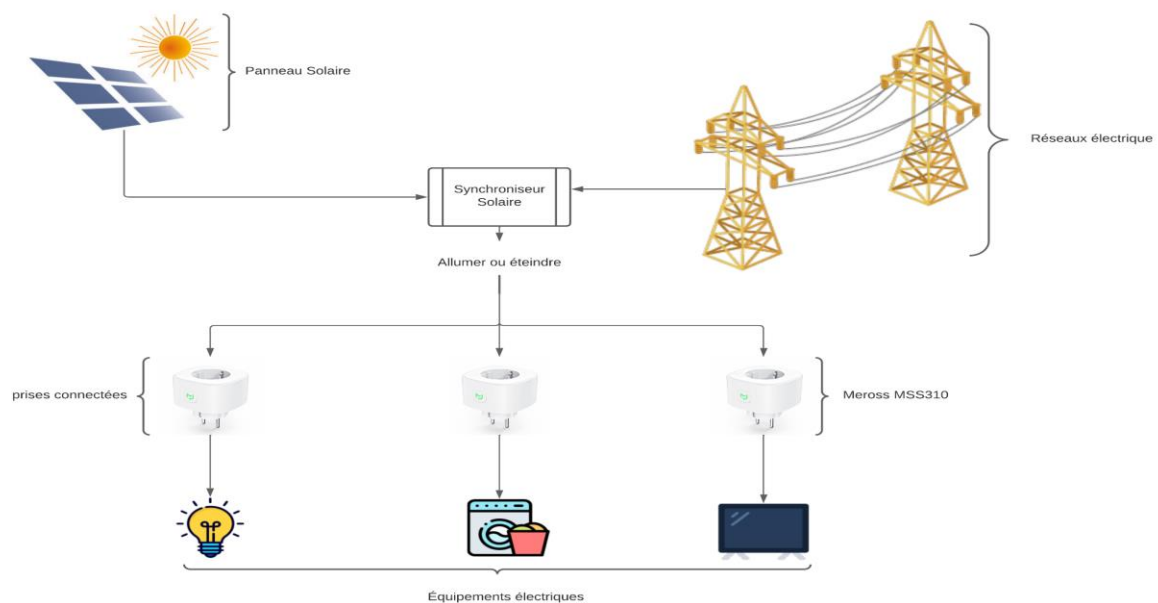


Figure 1 : Schéma visuel de l'utilité d'un synchroniseur solaire

2 Cadre théorique

2.1 État de l'art

L'**état de l'art** est une étape cruciale dans tout projet de recherche ou de développement. Il s'agit de la **collecte** et de la **synthèse d'informations** existantes sur un sujet donné, afin de comprendre les travaux antérieurs et d'établir un contexte pour le projet en cours. Pour le projet de développement d'un synchroniseur solaire basé sur un **Raspberry Pi** et **Home Assistant**, il est important de prendre en compte les travaux de recherche existants sur les systèmes de mesure de courant et de la consommation d'énergie. Cette étape permettra d'identifier les innovations récentes, les limitations techniques et les opportunités pour améliorer la performance et la fiabilité du système.

2.1.1 Quelles sont les marques qui vendent des prises connectées ?

Une étude du marché a été établie, voici le résultat des recherches :

- **Meross** : Smart Wi-Fi Plug MSS310
- **Konyks** : Priska Max 3
- **TP-Link** : Tapo P110
- **Philips** : Hue Smart Plug
- **Amazon** : Echo Flex
- **Bosch** : Smart Plug
- **Wiz** : Smart Plug Wi-Fi (avec mesure consommation)
- **Eve** : Eve Energy (Technologie Homekit Apple)
- **Awox** : Plug Plus (impossible d'accéder au paramétrage complet via WiFi)
- **Edimax** : SP-1101W (pas de suivi de consommation)
- **Trendnet** : THA-101 (avec mesure consommation)

2.1.2 Qu'est ce qu'un synchroniseur solaire ?

En effet, c'est le titre de mon projet, il est donc important de savoir sur quoi se base le projet. Les synchroniseurs solaires jouent un rôle crucial dans l'intégration de l'énergie solaire produite par les panneaux solaires au **réseau électrique**. Ils utilisent des techniques de régulation de la fréquence et de la tension pour **synchroniser la production d'énergie solaire** avec le réseau électrique. Les synchroniseurs solaires classiques utilisent des techniques telles que la commande de fréquence de source et la commande de tension de point de puissance pour synchroniser la production d'énergie solaire, mais ces techniques peuvent avoir des limitations en termes de flexibilité et de robustesse face aux perturbations du réseau électrique. Les avancées récentes dans la recherche sur les synchroniseurs solaires ont conduit à l'utilisation de techniques de commande avancées telles que la commande de synchronisation de phase et la commande de synchronisation de tenseur pour **améliorer la flexibilité et la robustesse des synchroniseurs solaires**. Il y a également des avancées en termes d'intégration des synchroniseurs solaires aux **systèmes de stockage d'énergie** pour améliorer la flexibilité et la fiabilité des systèmes photovoltaïques.

2.1.3 Qu'est ce que Home Assistant ?

L'utilisation de **Home assistant** est un sujet de recherche en plein essor, en particulier en ce qui concerne les **personnes âgées et leur soutien**. Une étude récente a montré que les perceptions des HA étaient positives, y compris leur potentiel pour **favoriser le vieillissement à domicile**. Cependant, il existe des défis liés à l'apprentissage de la technologie et au remplacement des anciennes habitudes par de nouvelles. Les participants ont également proposé des recommandations pour les futures compétences **VHA**¹ et pour plus d'éducation et de formation sur l'utilisation des HA. Il y a un besoin croissant de prendre en charge des langues autres que l'anglais pour les assistants à domicile virtuels. Il y a des recherches actives sur la compréhension du langage naturel pour les assistants virtuels, comme une architecture de réseau neuronal capsule personnalisée qui effectue la détection d'intention et le remplissage de créneaux de manière conjointe. L'**Internet des objets (IoT)** est également utilisé pour rendre les **maisons plus intelligentes**, en permettant la **connectivité à distance** et le suivi des objets du monde réel à travers le cyberspace. Les systèmes de domotique, comme les assistants à domicile virtuels, peuvent être utilisés pour **automatiser les tâches** ménagères et **surveiller la sécurité** de la maison. Il y a également des recherches sur la combinaison de l'IoT avec les assistants virtuels pour améliorer la communication et la connectivité dans la maison.

2.1.4 Quelles sont les normes de la domotique ?

Il existe **plusieurs normes** pour la domotique, qui définissent les caractéristiques techniques et les protocoles de communication pour les systèmes de contrôle de la maison. Les normes les plus courantes incluent :

- **KNX (anciennement EIB)** : une norme internationale pour les systèmes de contrôle de bâtiment qui permet la communication entre différents appareils et systèmes.
- **Zigbee** : une norme de communication sans fil pour les réseaux de capteurs et d'appareils de contrôle de la maison.
- **Z-Wave** : une norme de communication sans fil pour les réseaux de capteurs et d'appareils de contrôle de la maison.
- **EnOcean** : une norme pour les équipements énergétiquement autonomes utilisant l'énergie de l'environnement (lumière, chaleur, mouvement) pour alimenter leurs fonctions.
- **UPnP** : une norme de communication pour les réseaux domestiques qui permet la découverte et la configuration automatique des appareils connectés.
- **BACnet** : une norme de communication pour les systèmes de contrôle de bâtiment qui permet la communication entre différents appareils et systèmes.

Il existe d'autres normes, mais celles-ci sont les plus couramment utilisées. Il est important de savoir que la compatibilité entre les différentes normes n'est pas toujours possible, il faut donc s'assurer que les appareils et les systèmes choisis sont compatibles entre eux.

¹ **VHA** : Les aides auditives (VHA) sont des dispositifs électroniques portables qui aident les personnes malentendantes à **mieux entendre et comprendre la parole**. Les aides auditives modernes peuvent être équipées de fonctionnalités supplémentaires telles que la connectivité Bluetooth, la réduction du bruit et la suppression de larsen. Les aides auditives sont souvent prescrites et adaptées par des professionnels de la santé auditive tels que les audioprothésistes ou les médecins ORL.

3.2 Sources d'informations

La domotique est une technologie qui a connu un développement **important** ces dernières années, offrant ainsi une grande variété de services et de fonctionnalités. En effet, il est de plus en plus fréquent de voir des maisons intelligentes équipées de systèmes domotiques. Dans ce contexte, il était important pour moi, de me **familiariser** avec cette technologie et d'en apprendre davantage sur ses applications concrètes.

J'ai principalement utilisé des **forums en ligne** et des **articles spécialisés** pour m'informer sur le sujet. Les forums sont une excellente ressource pour obtenir des **conseils pratiques et des retours d'expérience** de la part de personnes ayant déjà mis en place des systèmes domotiques dans leur maison. Les articles spécialisés, quant à eux, m'ont permis de mieux comprendre les principes de fonctionnement de la domotique et les différentes technologies utilisées dans les systèmes domotiques.

3.2.1 Techniques et outils de collecte de données

Les techniques de collecte de données utilisées pour ce projet ont principalement été l'observation directe, l'analyse documentaire et les échanges avec mes tuteurs. L'observation directe a été l'une des techniques de collecte de données clés utilisées pour ce projet. J'ai pu observer directement les différentes tâches à accomplir et les **processus impliqués** dans la réalisation du projet. Cette technique a permis d'obtenir une compréhension approfondie de la situation et de l'environnement

L'analyse documentaire a également été utilisée pour collecter des données. J'ai effectué une recherche approfondie sur les technologies de domotique et les différents outils qui pourraient être utilisés pour réaliser le projet. J'ai également analysé des documents tels que les **manuels d'utilisation** des outils et des technologies pour m'assurer de comprendre leur fonctionnement et leur utilisation.

Les échanges avec mes tuteurs ont également été une source importante de données. Ces derniers ont permis d'obtenir des **commentaires** et des **suggestions** qui ont aidé à orienter le projet et à le mener à bien.

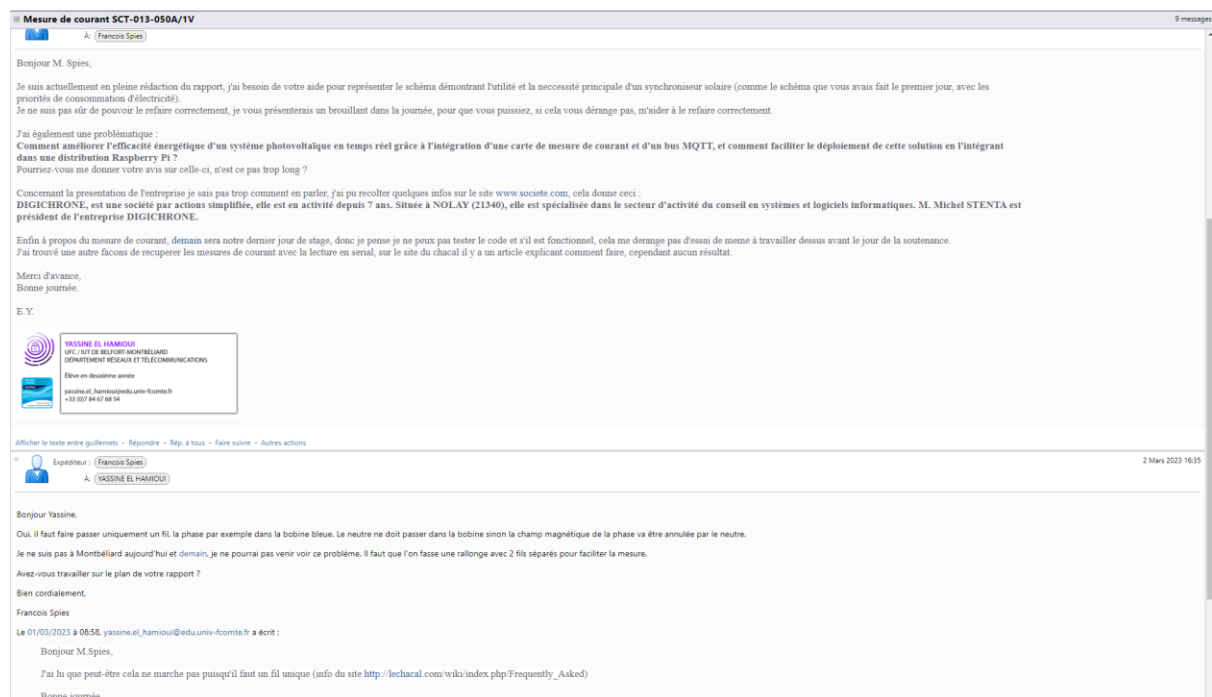


Figure 3 : Exemple d'échange avec mon tuteur (voir img-3.1 pour une meilleure visibilité)

3.3 Traitement de données

Dans le cadre de ce projet, j'ai appliqué la même technique lorsque j'effectue une tâche : dans un premier temps **j'analyse la tâche, je m'informe sur les technologie utilisé**, puis je prends des notes ou je garde ouvert la fenêtre jusqu'au terme de la tâche, puis les données collectées ont été adapté à la tâche à effectuer sous forme de brouillons. Je m'explique, si on prend par exemple le **développement d'un programme** qui connecte deux objet Iot (Internet Of Things) avec un **microcontrôleur**, je procédé de la manière suivante: je m'informe, j'adapte les informations récoltées sous forme de schéma, puis dans le cas de la programmation, je développe chaque partie du code avec un **langage compréhensible par un candide** (par exemple au lieu du code, j'écris en **langage naturel** ce que chaque partie doit faire) avant de le transformer/traduire en **code informatique**. Grace à cette méthode je peux me **concentrer sur le problème à résoudre** au lieu de m'attarder sur le fonctionnement informatique du script. Cela me permet également de **vérifier** que chaque étape est **cohérente** et **logique** avant de commencer à coder.

3.3.1 Méthodes et outils utilisés pour traiter les données collectées

Les méthodes et outils utilisés pour traiter les données collectées ont inclus des technologies Iot, la domotique utilise principalement **trois technologies** : la technologie par **réseau câblé**, la technologie par **réseau sans fil** et la technologie **courant porteur en ligne** ou **CPD** ². Les logiciels utilisés pour le traitement de données comprennent des langages tels que **Arduino en C++** et **Python**. Voici un schéma montrant la structure générale des tâches effectués :

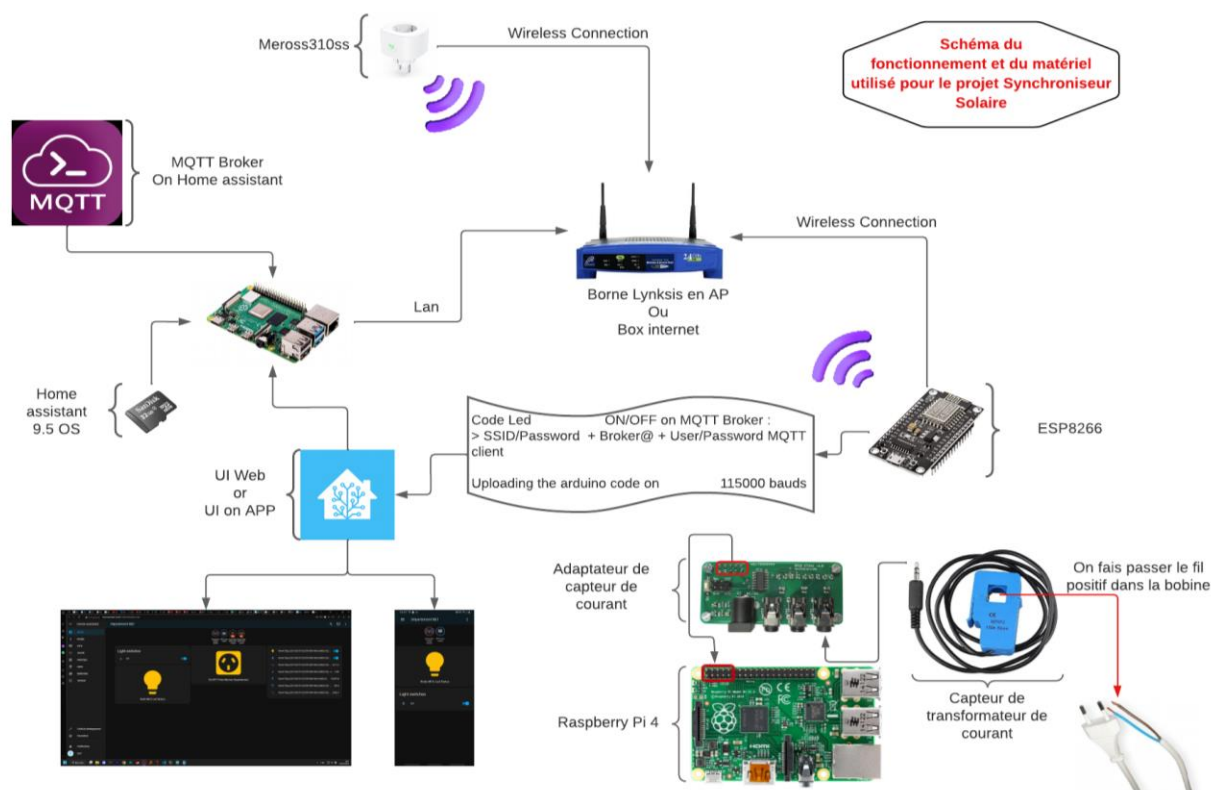


Figure 1 : Schéma visuel de l'utilité d'un synchroniseur solaire

² **CPD** : La technologie du courant porteur en ligne (CPD) utilise les lignes électriques existantes pour transmettre des données numériques. Elle est souvent utilisée pour connecter différents équipements ou pour permettre une surveillance à distance de certains paramètres. La CPD peut cependant avoir des limites en termes de vitesse de transmission des données et de fiabilité.

4 Présentation du projet

4.1 Cahier des charges du projet de stage

Les tâches principales consistent à ajouter un pilote de carte de mesure de courant sur un Raspberry Pi, à intégrer ce pilote au synchroniseur solaire existant, à ajouter un **bus MQTT** et une **extension home assistant** et enfin à intégrer l'ensemble dans une distribution **Raspberry Pi 4**³ pour faciliter le déploiement. En réalité, la thématique principale du sujet m'a été fourni avant le début du stage, j'ai donc pu commencer ma préparation, en effet je suis en parcours **cybersécurité**, j'ai pu participer à une seule journée aux cours sur l'Internet des objets, je possédais donc des bases très fragiles. Broker MQTT, Qos, publication/souscription, EPS8266, n'était pas des termes nouveaux, cependant celle-ci n'était pas maîtrisée et claire, j'ai donc décidé de consacrer ma préparation et les premiers jours du stage dans l'acquisition du plus de savoir et de compétence dans le domaine du projet

4.2 Étapes de réalisation du projet

La réalisation du projet de stage s'est déroulée en **plusieurs étapes**. Tout d'abord, il a fallu effectuer une **recherche bibliographique** approfondie sur les différentes technologies de domotique et les services disponibles sur le marché. Cette étape a permis de se **familiariser** avec les **concepts clés** de la domotique et d'**identifier** les services qui pourraient répondre aux besoins du projet. Ensuite, il a fallu procéder à une analyse détaillée des besoins du cahier des charges, pour définir les fonctionnalités requises pour le projet. Cette étape a été cruciale pour **orienter la réalisation du projet** et garantir sa pertinence par rapport aux attentes cahiers des charges. La phase de conception a ensuite commencé, où plusieurs solutions ont été proposées pour aboutir au fonctionnement des tâches à effectuer. Enfin, la phase de développement a commencé, où les fonctionnalités ont été implémentées et testées.

³ **Raspberry Pi 4** : Le Raspberry Pi 4 est un ordinateur monocarte (ou carte mère) de petite taille conçu par la Fondation Raspberry Pi. Il est conçu pour être abordable et facilement accessible à tous, il est souvent utilisé comme une plate-forme de développement pour des projets électroniques et informatiques, tels que la robotique, l'Internet des objets, les serveurs de médias, etc.

4.2.1 Home Assistant

Home Assistant est un système **open-source** de domotique qui permet de **gérer** et de **contrôler divers appareils connectés** dans une maison ou un bâtiment. Il permet de créer des **automatisations** et de **centraliser la gestion** de ces appareils à travers une interface web ou une application mobile. Il est compatible avec une grande variété de protocoles de communication tels que **MQTT**, **Zigbee**, **Z-Wave**, et bien d'autres. Il est possible d'installer Home Assistant (HA) depuis le site officiel www.home-assistant.io sur son propre ordinateur **windows** ou **linux**, cependant il faudrait laisser l'ordinateur allumer tout le temps sinon le serveur ne serait plus disponible, ce qui n'est pas du tout pratique. Home Assistant est un système assez petit, quoi de mieux qu'un Raspberry Pi 4 (Rpi) pour faire tourner HA. Sur Raspberry Pi 4, HA se présente sous forme de système d'exploitation basé sur une distribution Linux, nommé **HassOS**.

4.2.1.1 Mise en place de Home Assistant

Dans un premier temps j'ai décidé de découvrir Home Assistant sur une **machine virtuelle** (VM) linux, l'interface est accessible à l'**adresse IP** de home assistant sur le **port 8123** depuis un navigateur. Bien évidemment il faut être **dans le même réseau** afin de se connecter à la page, dans le cas où l'on arrive pas à récupérer l'adresse IP de HA il suffit de se connecter par le **nom de domaine** et le **port**, qui se présente sous cette forme : "homeassistant.local.8123". Afin de voir ce qui se passe sur le Rpi, ce dernier a été branché sur un écran par HDMI, j'ai pu trouver un terminale de lancement, sous forme de log (journal informatique), lorsque Home Assistant est prêt, l'affichage est mis à jour, une **bannière** est affichée (voir img-5.1 et 5.2). On peut également accéder à **Home Assistant Observer**, sur le **port 4357** (donc "homeassistant.local.4357" voir img-4.1), ce dernier nous fournit des informations cruciales du **statut** de HA. Cette page nous informe de l'**état du superviseur**, de la prise en charge ou encore l'état de "santé".

Qu'est ce que donc le "superviseur" ?

Dans Home Assistant le superviseur c'est un composant logiciel qui permet **la gestion et la surveillance** des applications de home assistant, il permet également d'installer des **modules complémentaires**, que l'on appelle "**add-ons**". Ces derniers ajoutent des **fonctionnalités** à HA, tels que des **serveurs DNS** (Domain Name System), des **bases de données** ou encore **MQTT**. Le superviseur surveille surtout **l'état de la configuration** de Home Assistant et gère les **mise à jour** nécessaires à l'ensemble du système.

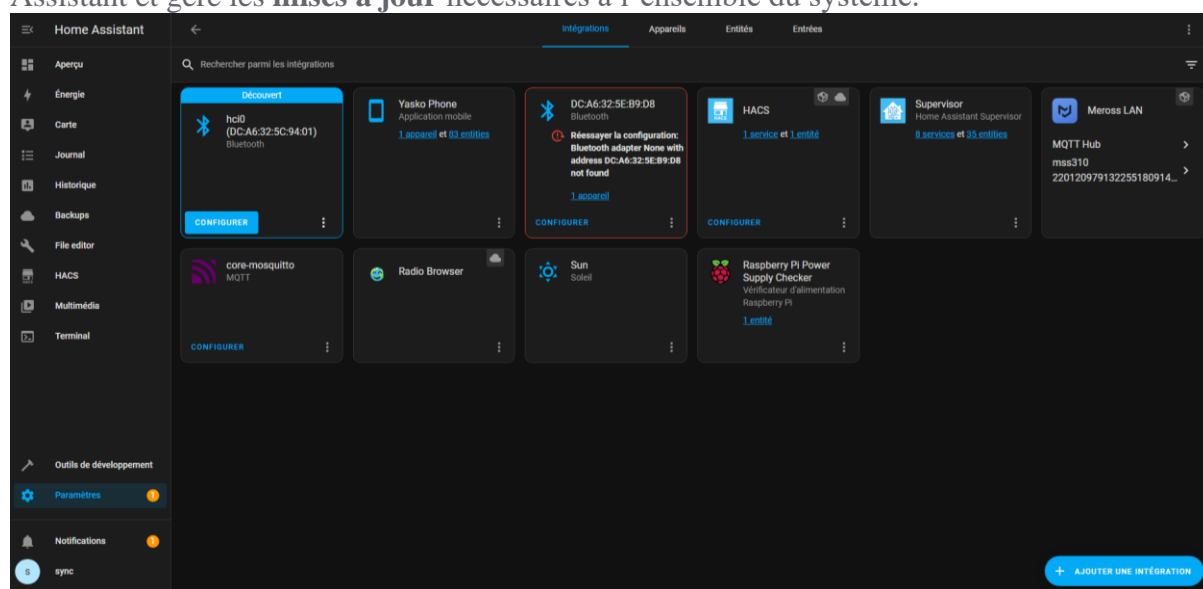


Figure 5 : Page d'accueil des intégrations et des modules complémentaires

L'installation de Ha sur machine virtuelle, à été assez rapide, le superviseur s'est chargé en moins de 10 minutes. Lors de mon passage sur Rpi 4 j'ai remarqué une **énorme différence** au niveau des **performances**, j'ai donc essayé plusieurs fois la réinstallation de Home Assistant avec des paramètres différents tels que le nom utilisateur et mot de passe ou bien encore l'activation ou la désactivation du protocole SSH ⁴ en pensant que la configuration était mauvaise. J'ai donc hypothésé que c'était uniquement un **manque de performances**, pour vérifier mon hypothèse j'ai réinstallé Ha sur la VM Linux mais avant cela j'ai **limité les performances** de la VM afin d'**imiter** celle d'un Rpi 4. J'ai donc configuré des paramètres à **4Go de mémoire vive** et j'ai limité le **processeur à 4 cœurs**. Ce test à **valider mon hypothèse**, en effet avec ces paramètres le lancement de Ha à pris presque 1 heure, contrairement au premier test avec des performances de 8Go de mémoire vive et 6 cœurs pour le processeur. En réalité sur Rpi 4 cela m'as pris plus de temps, j'ai pu vérifier avec **Wire Shark** (outil de capture et d'analyse de paquets réseau) afin d'**analyser le flux**, et j'ai pu remarquer que des paquets était cheminé vers HA, ce qui rend une explication logique du pourquoi sur Rpi 4, le temps de préparation du superviseur et de l'interface à pris presque 2 heure. J'ai rencontré des soucis avec les superviseurs lorsque je devais faire des redémarrages suite à une mise à jour ou une configuration significative, ces problèmes ont été causés principalement par le superviseur qui plantait. Le superviseur pouvait planter pour plusieurs raisons, tel que des **soucis de compatibilité** des modules complémentaires, par exemple la mise à jour d'un add-ons sans que le superviseur ne soit mis à jour ou encore une mauvais configuration des intégrations, parfois c'était un problème de ressources dû au manque de performances, qui peuvent perturber le superviseur. La majorité des fois, le superviseur ne démarrait pas du tout, je devais donc réinstaller HA depuis le début, bien heureusement, j'ai entrepris des **backups manuellement** avant chaque mise à jour, cependant le lancement de HA reste très long (presque 2 heures même si HA indique 20 minutes...)

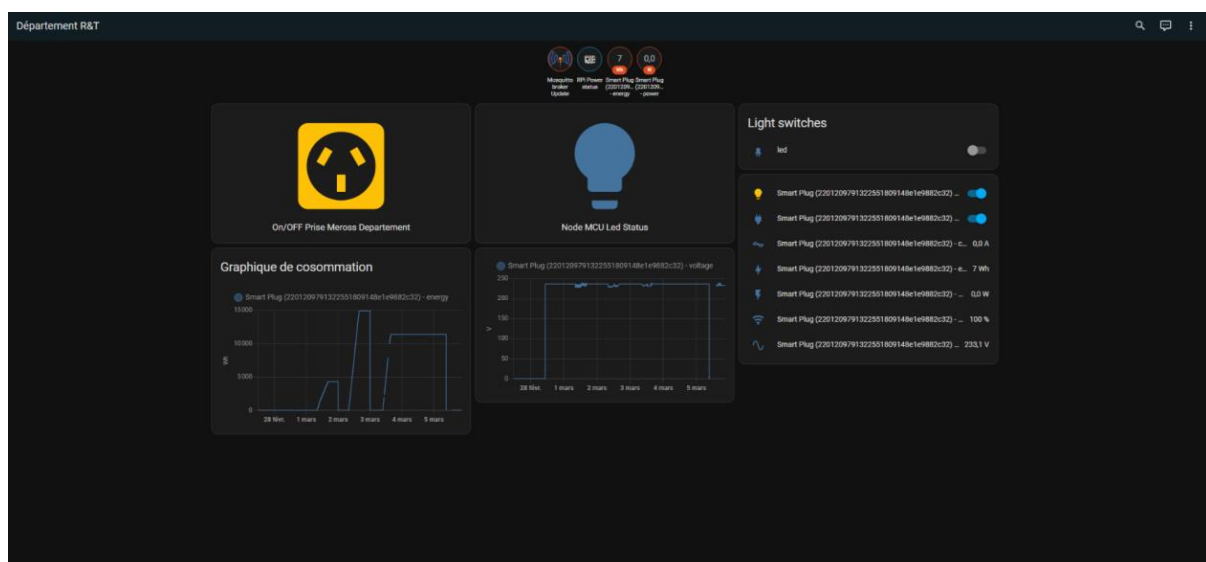


Figure 6 : Rendu finale du tableau de bord

⁴ SSH : SSH (Secure Shell) est un protocole de communication sécurisé qui permet à un utilisateur d'accéder à distance à un ordinateur ou un serveur via une connexion réseau.

Il est important de **garder à jour** et de **configurer** avec soin régulièrement l'état du système pour contrer tout type de soucis et d'anomalie. Le superviseur de Home Assistant est généralement **très fiable et stable**, mais il est toujours possible de rencontrer des soucis. Dans ces cas, pour savoir si HA est prêt on doit tout simplement rafraîchir la page, mais cela reste assez perturbant de devoir vérifier manuellement pendant presque 2 heures, j'ai donc opté pour une solution facilitant la récolte d'informations : j'ai développé un programme python qui envoie une requête à l'adresse de HA "<http://homeassistant.local:8123>" et qui a pour objectif d'attendre que la configuration et la page soit disponible. Ce script python est disponible en annexe (voir img-6.1 et img-6.2) et à ce lien : https://github.com/Yasko0x000/Projet_de_Stage_Synchroniseur_Solaire/blob/master/Code/check_ha_page.py

Pour ce faire j'ai déclaré 2 variables, une **url** qui renvoie vers **l'adresse de HA** et une deuxième qui renvoie à la variable précédente plus le paramètre **"/config"**.

```
url = "http://homeassistant.local:8123"
config_url = url + "/config"
```

J'ai ensuite utilisé une boucle **"while True"** pour envoyer continuellement des **requêtes** tant que les conditions d'arrêt ne sont pas remplies. Ensuite j'envoie une requête **GET** à l'adresse de ha et je vérifie si le code de réponse est 200 (c'est-à-dire, 200 équivaut au statut de réponse HTTP 200 ok qui indique la réussite de la requête).

```
while True:
    try:
        response = requests.get(url)
        if response.status_code == 200:
            config_response = requests.get(config_url)
```

Donc si le code de réponse est 200, alors la page est disponible. Il est important de noter que HA peut être démarré mais sans que le superviseur ait préparé et lancé les intégrations, les modules complémentaires ainsi que les services paramétrés. J'ai donc ajouté une **deuxième requête GET** à l'adresse "<http://homeassistant.local:8123/config>" pour vérifier si ha est réellement prêt.

```
config_response = requests.get(config_url)
if config_response.status_code == 200:
    print("La page est prête !")
    break
else:
    print("La page est disponible, mais la configuration n'est pas encore prête...")
```

Lorsque la page de configuration est prête, le script affiche "La page est prête !" et sort de la boucle while True. Sinon, le script affiche "La page est disponible, mais la configuration n'est pas encore prête...!" et continue d'attendre dans la même boucle. Dans le cas contraire, si la requête GET ne renvoie pas 200, alors le script affiche "La page n'est pas encore disponible, réessayez plus tard..." et comme le cas précédent, le programme continue à tourner en boucle jusqu'à ce que la configuration soit prête. Enfin j'ai ajouté une instruction "time.sleep(10)" pour faire une pause de 10 secondes avant de renvoyer une autre requête, ce qui permet de ne pas surcharger le serveur avec des requêtes inutiles.

4.2.1.2 Accès en SSH aux fichiers du système

Afin d'éviter de devoir brancher un écran et clavier pour chaque besoin d'opérer sur le système, j'ai installé très simplement un **terminal SSH** sous forme d'add-ons, la connexion se fait automatiquement et cela affiche directement le terminal ha (voir Figure 7).

Comme pour chaque système ouvert au grand public, on peut trouver une **interface graphique** facilitant la configuration du système mais ce dernier peut parfois être **limité niveau fonctionnalité**. Pour cela j'ai préféré paramétrer l'interface d'overview, les automatisations, les intégrations ainsi que la majorité de mon travail sur HA avec le **format de sérialisation de données** par HA :

“**YAML Ain't Markup Language**”

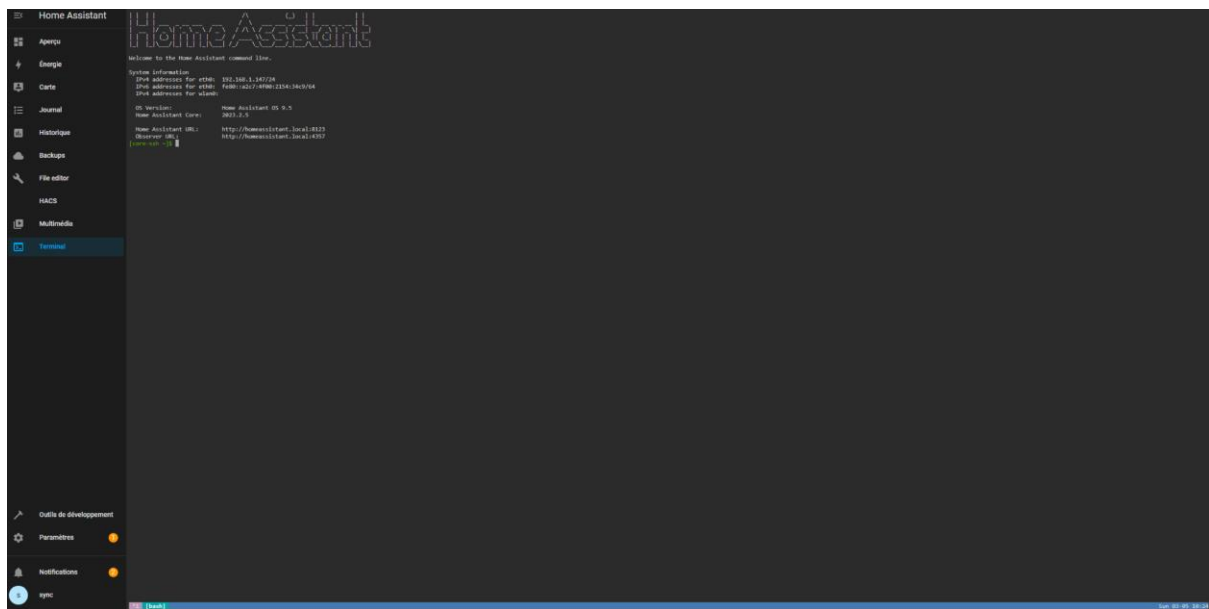


Figure 7 : Bannière de HA sur le terminale depuis ssh

Pour accéder aux fichiers **yaml** on peut soit y accéder depuis le terminal au chemin “/home/homeassistant/.homeassistant/configuration.yaml” ou bien, HA propose en parallèle un module complémentaire appelé **File Editor** qui nous permet d’avoir une interface graphique des fichiers. Après quelques jours j’ai opté pour File Editor (voir img-7.1), au départ le fichier qui m’intéressais principalement était configuration.yaml, dans ce derniers on trouvait les paramètres de configurations généraux, sur lequel on peut configurer presque tous les services et les intégrations disponible. Cela jusqu’à l’arrivé d’une mise à jour qui à modifier cela. Suite à la mise à jour, l’automatisation et la configuration du tableau de bord reste tout de même disponible depuis les fichier yaml et depuis l’interface graphique (voir img-7.2).

Yaml est parfois considéré comme un **langage plus lisible** et plus simple que des formats de sérialisation de données tels que **XML** et **JSON**, ce retour des utilisateur est vrai, en effet il m’as suffi d’utiliser l’interface graphique pour simuler une tâche et d’analyser le code généré par l’interface, le site officiel de Home Assistant à également était utile, cette page explique le fonctionnement : <https://www.home-assistant.io/docs/configuration/yaml/>.

4.2.2 Bus MQTT

MQTT, est un **protocole de communication** de messagerie **légère** qui permet à des dispositifs de communiquer entre eux en utilisant un **modèle de publication et de souscription**. Les messages envoyés via MQTT sont souvent très courts et utilisent **peu de bande passante**, ce qui le rend idéale pour des dispositifs avec des **performances parfois limitées**, typiquement en domotique.

J'ai commencé par faire quelques manipulation avec MQTT, pour cela, j'ai **flashé** une image **Raspbian OS** à l'aide de **Raspberry Pi Imager** (voir img-8.1 et 8.2) afin de faire fonctionner le Raspberry Pi 4. Suite à la configuration du Rpi, je l'ai connecté à mon réseau local et mis en marche SSH. Depuis mon pc, je me suis connecté et j'ai démarré les manipulations avec MQTT entre Rpi et Kali linux (sous WSL2 ⁵) : avec **mosquitto** j'ai pu faire des tests de souscription et publication, sous des topics. J'ai pris comme exemple l'iut, j'ai donc structuré le topic sous cette forme :

IutNFC	DépartementRT	amphi2
		salle104
		salle203
IutNFC	DépartementGaco	amphi2
		salleTD

Si on prend par exemple un client qui s'abonne au topic "iutnfc/departementrt/#", il recevra tous les messages publiés sur les topics qui commencent par "iutnfc/departementrt/". Le caractère "#" est utilisé comme référence pour indiquer que l'on a accès à tous les niveaux de profondeur. Nous avons un autre caractère "+" est utilisé comme un parametre pour remplacer un seul niveau de profondeur dans un topic. Par exemple, si un client s'abonne au sujet "iutnfc/+amphi2", il recevra tous les messages publiés sur des sujets qui commencent par "iutnfc/", suivi d'un niveau quelconque, puis de "amphi2".

⁵ **WSL2** : Windows Subsystem for Linux est une **couche de compatibilité** permettant d'exécuter des exécutables binaires **Linux** de manière native sur **Windows** (Version 10, 11 ou Server 2019) .

Il pourrait recevoir des messages publiés sur des sujets tels que "iutnfc/departementrt/amphi2" ou "iutnfc/departementgaco/amphi2", mais pas "iutnfc/amphi2", car ils ne correspondent pas à la structure de topic spécifiée. Le caractère "+" est utile pour **s'abonner** à une série de **sujets similaires**, mais avec une partie de ces sujets qui varie. Il permet de s'abonner à des sujets qui correspondent à une structure générale spécifique, tout en excluant les sujets qui ne correspondent pas à cette structure, comme pour le cas de l'amphi utilisé par le département R&T et Gaco.

Voici quelques exemple pratique avec mosquitto :

- ❖ Pour s'abonner au topic "iutnfc/departementrt/" et afficher les messages reçus de toutes les salles du département R&T :

```
mosquitto_sub -t "iutnfc/departementrt/#"
```

- ❖ Pour publier un message "maintenance" sur le topic "iutnfc/departementrt/salle203" :

```
mosquitto_pub -t "iutnfc/departementrt/salle203" -m "maintenance"
```

- ❖ Pour s'abonner au topic "iutnfc/departementrt/amphi2/" ou "iutnfc/departementgaco/amphi2/" et afficher les messages reçus sur amphi2 :

```
mosquitto_sub -t "iutnfc/+/amphi2/#"
```

- ❖ Pour publier un message "reservé GPO" sur le topic "iutnfc/departementrt/salle203" :

```
mosquitto_pub -t "iutnfc/departementrt/amphi2/" -m "reservé GPO"  
mosquitto_pub -t "iutnfc/departementgaco/amphi2/" -m "reservé GPO"
```

ou plus simplement en une ligne :

```
mosquitto_pub -t "iutnfc/+/amphi2" -m "reservé GPO"
```

4.2.2.1 ESP8266

Afin de manipuler MQTT en liaison avec le projet, j'ai pu utiliser un **Esp8266**⁶. Ce dernier a été utilisé comme une **Led**, mais bien évidemment avec une autre nécessité tel qu'un avertissement lors du dépassement d'un certain seuil, défini de consommation d'énergie, celui-ci pourrait être paramétré par exemple pour le **faire clignoter au dépassement** de la limite définie précédemment. Dans mon cas, j'ai programmé le microcontrôleur en **Arduino en C++**, pour tout simplement **allumer** et **éteindre** une led avec un programme qui exécute une ligne de commande, que l'on appelle **Payload** avec un argument sous forme de bouton **OFF** ou **ON** depuis **Home assistant**.

Ce code utilise les bibliothèques **ESP8266WiFi.h** et **PubSubClient.h** pour connecter un ESP8266 à un réseau WiFi et à un serveur MQTT (sur Home Assistant). Il définit les variables de connexion WiFi, et la variable de connexion MQTT, **mqtt_server**, donc son adresse. Depuis Home assistant on doit configurer un **login**, afin d'avoir une **meilleure sécurisation**. Il utilise un objet **WiFiClient** pour créer une connexion réseau et un objet **PubSubClient** pour **communiquer** avec le **serveur MQTT**.

```
6  const char* ssid = "FuzeTea";
7  const char* password = "guadeloupe";
8  const char* mqtt_server = "192.168.1.147";
9  const char* mqtt_username = "mqttbroker";
10 const char* mqtt_password = "rahXuchooy0ahh4Sheet8aegaileifae2Aephoh3aiyahgo6uaziojeoMah5zei7";
11
12 WiFiClient espClient;
13 PubSubClient client(espClient);
14 unsigned long lastMsg = 0;
15 #define MSG_BUFFER_SIZE (50)
16 char msg[MSG_BUFFER_SIZE];
17 int value = 0;
```

La fonction **setup_wifi()** se connecte au réseau **WiFi** en utilisant les variables **ssid** et **password**, et affiche l'**adresse IP locale** obtenue. Il est essentiel d'avoir une connexion wifi afin d'être dans la portée du serveur MQTT.

La fonction **callback()** est utilisée pour traiter les messages MQTT reçus. Elle affiche le sujet et le message reçus, puis vérifie si le message est **"ON"** pour allumer la LED **intégrée** de l'ESP8266 ou **"OFF"** pour l'éteindre. Sinon, il affiche "Je ne comprend pas". Si on analyse la ligne 54, on peut se demander pourquoi ne pas remplacer cela tout simplement par "if ((char)payload == ON { ...", l'explication est tout simplement que la variable payload est de **type tableau**, et qu'il faut donc accéder à chaque élément **individuellement** pour vérifier s'il correspond à la condition. C'est pourquoi nous avons besoin d'utiliser **payload[0]** et **payload[1]** pour accéder aux deux premiers caractères de la variable payload.

```
43 void callback(char* topic, byte* payload, unsigned int length) {
44     Serial.print("Message arrived [");
45     Serial.print(topic);
46     Serial.print("] ");
47     for (int i = 0; i < length; i++) {
48         Serial.print((char)payload[i]);
49     }
50     Serial.println();
51
52     // Switch on the LED if an 1 was received as first character
53     Serial.print((char)payload[0]);
54     if ((char)payload[0] == 'O' && (char)payload[1] == 'N') {
55         digitalWrite(BUILTIN_LED, LOW); // Turn the LED on (Note that LOW is the voltage level
56         // but actually the LED is on; this is because
57         // it is active low on the ESP-01)
58     } else if ((char)payload[0] == 'O' && (char)payload[1] == 'F' && (char)payload[2] == 'F') {
59         digitalWrite(BUILTIN_LED, HIGH); // Turn the LED off by making the voltage HIGH
60     }
61     else{
62         Serial.println("Je ne comprend pas");
63     }
```

La fonction `reconnect()` tente de se reconnecter au serveur MQTT si la connexion a été perdue. Il utilise également un **ID client aléatoire** pour se connecter. Lorsque le serveur MQTT reçoit un message “ON” ou “OFF” sur le topic “**home/fuzetea/led**”, l’esp8266 lit le message et **allume** ou **éteint** la led.

```
68 void reconnect() {
69     // Loop until we're reconnected
70     while (!client.connected()) {
71         Serial.print("Attempting MQTT connection...");
72         // Create a random client ID
73         String clientId = "ESP8266Client-";
74         clientId += String(random(0xffff), HEX);
75         // Attempt to connect
76         if (client.connect(clientId.c_str(), mqtt_username, mqtt_password)) {
77             Serial.println("connected");
78             // Once connected, publish an announcement...
79             client.publish("lol", "connection etablie");
80             // ... and resubscribe
81             client.subscribe("home/fuzetea/led");
82         } else {
83             Serial.print("failed, rc=");
84             Serial.print(client.state());
85             Serial.println(" try again in 5 seconds");
86             // Wait 5 seconds before retrying
87             delay(5000);
88         }
89     }
}
```

La fonction `setup()` configure le pin **BUILTIN_LED** de l'ESP8266 comme sortie, initialise la communication série, appelle la fonction `setup_wifi()` pour se connecter au réseau WiFi, configure le serveur MQTT sur le **port 1883** et la fonction de rappel.

```
92 void setup() {
93     pinMode(BUILTIN_LED, OUTPUT);    // Initialize the BUILTIN_LED pin as an output
94     Serial.begin(115200);
95     setup_wifi();
96     client.setServer(mqtt_server, 1883);
97     client.setCallback(callback);
98 }
```

La fonction `loop()` vérifie si la connexion MQTT est toujours active et si ce n'est pas le cas, appelle la fonction `reconnect()`. Il gère également la publication périodique de messages "hello world #" suivi d'un compteur incrémentant sur le sujet "lol".

```
100 void loop() {
101
102     if (!client.connected()) {
103         reconnect();
104     }
105     client.loop();
106
107     unsigned long now = millis();
108     if (now - lastMsg > 2000) {
109         lastMsg = now;
110         ++value;
111         snprintf (msg, MSG_BUFFER_SIZE, "hello world #%ld", value);
112         Serial.print("Publish message: ");
113         Serial.println(msg);
114         client.publish("lol", msg);
115         delay(3000);
116     }
117 }
```

⁶ **ESP8266** : L'ESP8266 est un **module de communication sans fil** populaire utilisé dans les projets IoT (Internet des objets). Il est souvent utilisé pour connecter des appareils à un réseau WiFi et à Internet, et peut également être utilisé pour créer des réseaux locaux ou pour communiquer avec d'autres appareils sans fil. Il est également très populaire pour les **projets d'électronique basée sur Arduino**.

4.2.2.2 MQTT sur Home Assistant

Dans le précédent chapitre “4.2.2.1 ESP8266”, j’ai programmé un microcontrôleur afin de manipuler MQTT, l’Esp8266 doit communiquer avec le serveur de MQTT **hébergé** sur HA. J’ai donc ajouté l’intégration MQTT. Pour la configuration cela est assez simple, il est impératif de créer un profil qui va me servir de phase d’authentification pour MQTT, j’ai donc indiqué ceci :

- **Login** : “mqttbroker”
- **Mot de passe** : “rahXuchooy0ahh4Sheet8aegaileifae2Aephoh3aiyahgo6uaziojeoMah5zei7”

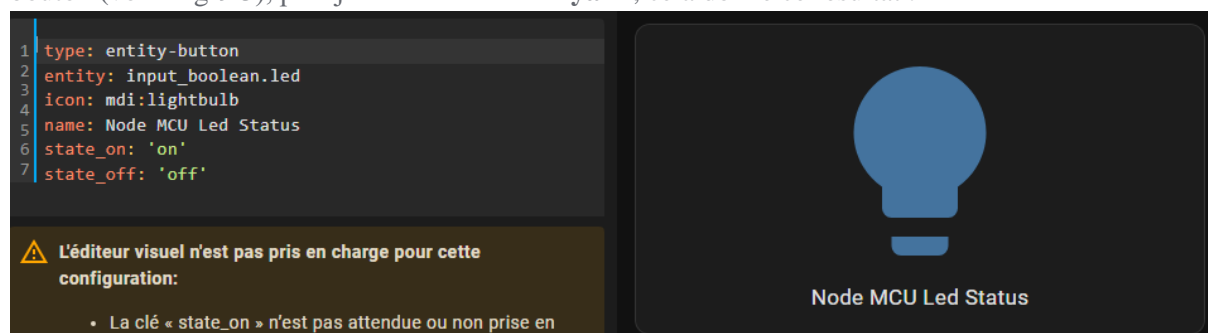
Une fois le profil créé, j’ai configuré cela dans le broker MQTT (mosquitto-core voir img-9.1).

Par suite de la connexion du microcontrôleur à MQTT, il est nécessaire d’allumer et d’éteindre la led **manuellement**, pour cela on se rend sur la page de configuration de MQTT à travers les intégrations et on doit indiquer les informations nécessaires à la publication d’un message. On indique le **topic**, la **qualité de service** ainsi que le message à envoyer, soit “ON” ou “OFF” (voir img-9.2). On peut également faire cela par ligne de commande avec l’installation préalable de mosquitto et mosquitto-client sur linux, la façon de configurations comprends les mêmes paramètres cités précédemment :

```
mosquitto_sub -h 192.168.1.147 -t home/fuzetea/led -u mqttbroker -P rahXuchooy0ahh4Sheet8aegaileifae2Aephoh3aiyahgo6uaziojeoMah5zei7
```

Cette commande me permet de m’abonner au topic de la led, **-h** indique l’adresse du **host MQTT**, **-t** est le **topic**, **-u** est l’**username**, donc le login et **-P** est le **mot de passe**. Si on veut publier un message, on remplace uniquement **sub** par **pub** et on ajoute le paramètre **-m** pour indiquer le message (soit ON, soit OFF). Cette manipulation n’est pas du tout pratique si l’on doit effectuer une simple tâche d’allumer ou d’éteindre une led, pour cela, j’ai **automatisé** cette manipulation avec un simple bouton (qui **switch** de la phase allumé à éteint, ou l’inverse), ce bouton est donc accessible depuis le **tableau de bord**.

En première étape je me suis rendu sur la page du tableau de bord et j’ai ajouté une carte appelé bouton (voir img-9.3), puis j’ai modifié cela en **yaml**, cela donne ce résultat :



J’ai configuré le type de la carte en bouton, j’ai indiqué quel **entité**⁷ est ciblé de l’action, puis j’ai créé une icône d’une ampoule à l’aide de www.flaticon.com/ et j’ai enfin défini l’état du bouton, lorsque le bouton est allumé alors cela déclare un état nommé “on”, à l’inverse lorsque le bouton est éteint alors l’état déclaré est “off”. Cependant ce bouton reste inutile sans une action consécutive, passons maintenant à l’automatisation de celui-ci.

⁷ **Entité** : Une entité représente une **partie d’un système** ou d’un appareil physique qui peut être surveillée ou contrôlée par Home Assistant. Cela peut inclure des capteurs de température, des interrupteurs, des lumières, des caméras, des thermostats et bien plus encore.

Il est possible de créer une automatisation à partir de l'interface graphique (voir img-10), comme cité précédemment j'ai préféré manipuler yaml, pour éviter tout type de limites. Je me suis donc rendu sur le fichier **automatisations.yaml** (voir <https://pastebin.com/aWvEpKN6> ou img-7.1). Dans ce code j'ai paramétré les 2 automatisations qui reflètent le switch d'un état allumé vers un état éteint ou encore l'inverse :

- La première automatisation (alias: "Switch LED with Entity Button ON") est déclenchée lorsqu'un bouton (input_boolean.led) **passse de "off" à "on"**. L'action associée consiste à publier un message MQTT sur le topic "home/fuzetea/led" avec la payload **"ON"**.
- La seconde automatisation (alias: "Switch LED with Entity Button OFF") est déclenchée lorsqu'un bouton (input_boolean.led) **passse de "on" à "off"**. L'action associée consiste à publier un message MQTT sur le topic "home/fuzetea/led" avec la payload **"OFF"**.

Prenons le cas de la première automatisation, le paramètre **"trigger"** indique quand l'automatisation doit être déclenchée.

```
3 trigger:
4   - platform: state
5     entity_id:
6       - input_boolean.led
7     from: "off"
8     to: "on"
```

Il est donc déclenché lorsque l'état du bouton sur le tableau de bord **passse d'une valeur à une autre** (c'est-à-dire de "on" à "OFF" ou de "OFF" à "ON").

Le paramètre **"action"** indique l'action à faire quand l'activation de l'automatisation est effectuée. La commande **"mqtt.publish"** me permet de publier un message MQTT avec la valeur "ON" ou "OFF" sur le topic "home/fuzetea/led".

```
10 action:
11   - service: mqtt.publish
12     data:
13       topic: home/fuzetea/led
14       payload: "ON"
15       qos: 0
16       retain: false
```

J'ai défini le niveau de **Qos** sur 0, ce qui signifie que le message est publié au moins une fois **sans confirmations**, étant donné que ce sont uniquement des tests, et que le projet se place sur un réseau assez petit, je me suis permis de **réduire** la qualité de service. En conditions réelles je conseille le niveau de **Qos 1** sur un réseau domotique (publication, plus une confirmation de réception).

Le champ **"retain"** je l'ai laissé par défaut, puisqu'il est inutile pour ma configuration. Si ce champ est défini sur "true", alors le Payload est **stocké** par le broker MQTT et il renvoie ce message à tous les nouveaux clients qui s'abonnent au topic.

4.2.3 Prise connecté

Les prises connectées sont un équipement essentiel au fonctionnement du projet, le synchroniseur solaire a besoin de connaître des **mesures de consommations précises** afin de déterminer l'action à entreprendre. Comme le nom l'indique, une prise connectée est tout simplement un dispositif domotique qui permet de **contrôler à distance** l'alimentation électrique d'un équipement branché dessus. Dans le cadre de ce projet, une étude approfondie des différents types de prises connectées disponibles sur le marché a été menée pour comparer le modèle **Meross MSS310** (voir chapitre 2.2.1 et Figure 8) aux autres, afin de déterminer le **plus approprié** aux besoins spécifiques de la solution de domotique.

Les prises connectées Meross utilisent MQTT pour permettre aux utilisateurs de contrôler à distance leurs appareils connectés via l'application mobile Meross ou d'autres plates-formes de domotique compatibles, Home Assistant en fait partie. La prise Meross MSS310 est conforme aux exigences du projet, car elle comprend les caractéristique nécessaire, tels que la connexion au réseau internet (par wifi avec une fréquence de 2.4Ghz), mais ce qui nous intéresse est sa fonctionnalité de mesure de la consommation électrique.

Afin d'accéder aux fonctionnalités de la prise, Meross ont mis en place une **application** qui permet de **configurer** et de **monitorer** leurs prises connectées. Il suffit de suivre les instructions d'installation et de connexion indiqué par l'application, ensuite il faut ajouter les informations de connexion au réseau (donc nom SSID et mot de passe). Généralement Meross utilise les **API*** Cloud de Meross pour communiquer avec les appareils, plutôt que le protocole MQTT. Cependant cela reste possible d'utiliser le protocole MQTT, dans le cas de l'intégration **Meross LAN** sur Home Assistant.



Figure 8 : Prise connecté Meross MSS310

* **API**: API signifie "Application Programming Interface" en anglais, une API définit un **ensemble de règles**, de **protocoles** et de **standards** qui permettent à une application de **demandeur** ou de **fournir des données** à une autre application. Les développeurs peuvent utiliser une API pour intégrer les fonctionnalités d'une application dans une autre application ou pour automatiser des tâches.

4.2.3.1 MSS310 sur Home Assistant

En lisant la documentation de Meross pour l'intégrer sur HA, ils indiquent de l'ajouter tels que les autres intégrations, cependant elle m'était introuvable, j'ai donc trouvé une solution alternative. La communauté de HA a développé un Store nommé **HACS** (Home Assistant Community Store) qui se présente sous forme de **plateforme communautaires** qui permet d'installer et de gérer des **intégrations** pour Home Assistant. HACS utilise **Github** pour stocker et gérer les dépôts de logiciels tiers. Je tiens à rappeler que Github est une plateforme de développement basée sur le cloud. Une fois l'installation de HACS, j'ai dû me connecter avec mon compte Github pour profiter des intégrations de HACS, j'ai rapidement trouvé l'intégration Meross LAN, étant donné que la prise est sous le même réseau, ce derniers détecte automatiquement la prise MSS310.

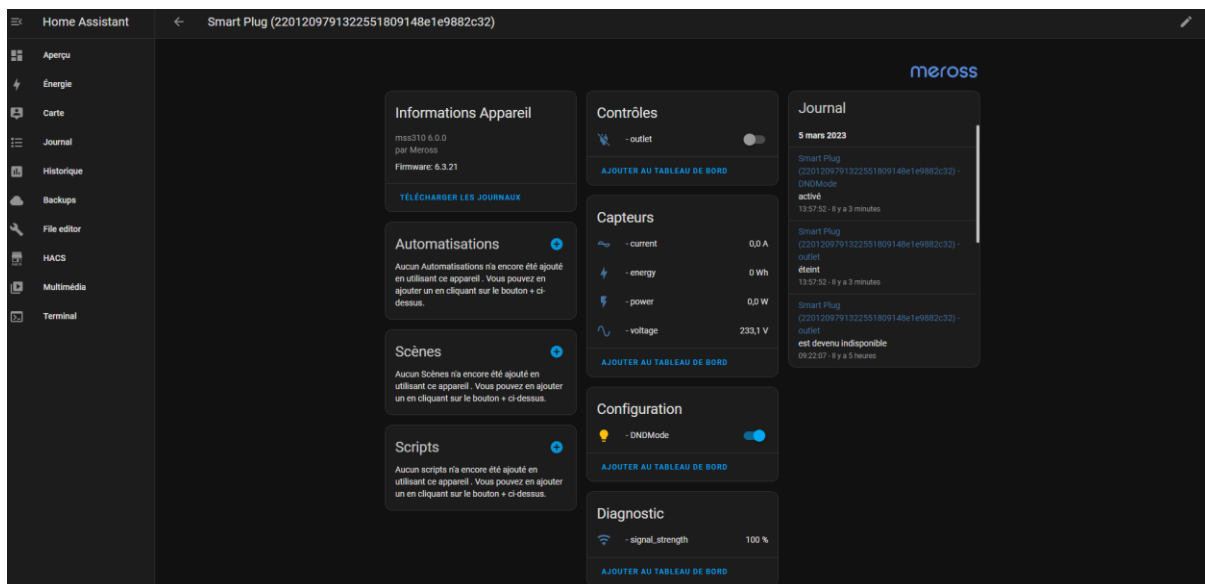


Figure 9 : Page de configuration de Meross LAN

J'ai décidé d'ajouter le bouton pour allumer/éteindre la prise (voir img-11.1) avec l'interface graphique au vu de la rapidité de ce dernier. Cependant l'intégration des **graphiques de consommation** et la **liste informative de chaque entité** de la prise, je les ai configurés en code yaml.

Pour ce faire, j'ai spécifié le type de graphique à créer, qui est un graphique d'historique, puis j'ai spécifié le nombre d'heures à afficher dans le graphique. Dans ce cas, le graphique affichera les données des 168 dernières heures, **soit une semaine**. (voir img-11.2)

Pour ajouter la liste des informations générales de la prise, j'ai tout simplement déclaré les entités à afficher. (voir img-11.3)

4.2.4 Capteur de mesure de courant

Dans ce chapitre, j'ai abordé la mise en place d'un **capteur de mesure de courant**. Celui-ci devrait être intégré au système existant grâce à l'utilisation d'un pilote sur Raspberry Pi. Ces mesures de courant vont permettre une **surveillance de la consommation énergétique** du système solaire synchronisé. Cette fonctionnalité est essentielle pour optimiser les performances et limiter la consommation excessive ou la sous-consommation d'énergie. Les résultats de ces mesures devront être transmis à l'aide du protocole MQTT, afin qu'elle soit également affichée sur l'extension Home Assistant.

Comme cité précédemment, pour mesurer le courant, j'ai utilisé un Raspberry Pi 4 avec une extension **SCT-013-050A/1V** pour profiter des performances de celui-ci, par la suite, un **RPIZ_CT3V1** (voir img-12.1), sera utilisé.

Voici le lien du RPIZ_CT3V1 : http://lechacal.com/wiki/index.php?title=RPIZ_CT3V1.

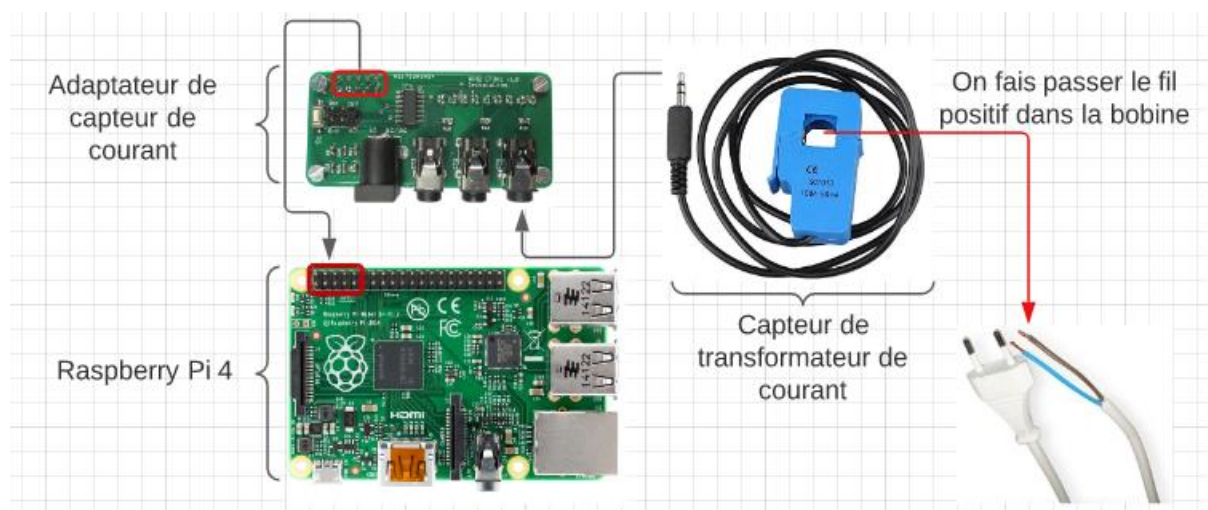


Figure 10 : Plan du capteur de mesure de courant

Je n'ai pas réussi à venir à bout de cette tâche, mais j'ai tout de même effectué énormément de tentatives, j'ai alors programmé deux code python différents dans l'espoir de réussir à récupérer des mesures précises, qui suivant la théorie, devraient marcher sans soucis.

Étant donné qu'il suffit d'exécuter un programme python, j'ai simplement installé Raspbian OS à l'aide de Raspberry Pi Imager et je me suis connecté en SSH depuis mon pc.

En réalité ces codes je n'ai pas pu les tester, puisque j'ai **identifié** un peu tardivement la raison pour laquelle la valeur retournée était toujours négative et fixe (voir img-12.2), la raison était simplement dû à la **façon de comment je mesurais le courant passant dans le fil**. Il fallait tout d'abord comprendre comment fonctionne le capteur de courant en ma possession :

Ce capteur de courant est un **transformateur** de courant à **noyau ouvert**, ce qui signifie que son noyau est fendu et qu'il peut être placé autour d'un **seul fil conducteur** sans avoir besoin de couper le fil ou d'établir une connexion électrique.

En outre, il faut faire passer un seul fil puisque **le neutre ne doit pas passer dans la bobine**, sinon le champ magnétique va être annulé par le neutre.



Figure 11 : Image montrant la bonne façon de mesurer issue du site ["http://lechacal.com/wiki/index.php/Frequently_Asked"](http://lechacal.com/wiki/index.php/Frequently_Asked)

Le premier code (voir <https://pastebin.com/kGcMFVtA>), utilise la connexion du capteur de courant “SCT-013-000 de 50A/1V” (voir figure 10) sur la broche **GPIO 17** du Raspberry Pi 4 pour la lecture du signal du capteur de courant. Voici une explication plus détaillée :

Dans ce code, j’ai commencé par importer deux bibliothèques, la bibliothèque “**pigpio**” pour l’interface avec les **GPIO** du Raspberry Pi et la bibliothèque “**time**” pour la gestion du temps.

```
C: > Users > rayan > Dropbox > PC > Documents > Synch_Solar > Code > current_sensor_GPIO.py > ...
1  import time
2  import pigpio
```

Ensuite, j’ai défini la fonction “**measure_current()**” qui permet de lire la tension en entrée du GPIO, de la convertir en tension réelle, puis en courant. Cette dernière va retourner la valeur du courant mesuré.

```
14 # Fonction pour mesurer le courant
15 def measure_current():
16     value = pi.read(GPIO)
17     voltage = value * 3.3 / 4095 # Conversion de la valeur lue en tension
18     current = voltage / COEFF    # Conversion de la tension en courant
19     return current
```

Pour convertir la tension mesurée en une valeur de courant j’ai utilisé un coefficient de calibration “**COEFF**”. La calibration de ce coefficient dépend de la plage de mesure du capteur et de la résistance de charge du circuit. J’ai défini la calibration à 50A/V, pour que chaque volt de courant mesurée, corresponde à 50 ampères.

```
21 # Boucle principale
22 try:
23     while True:
24         current = measure_current()
25         print(f"Courant: {current:.2f} A")
26         time.sleep(1)
```

J’ai utilisé cette boucle pour mesurer le courant à intervalles réguliers, ensuite la valeur mesurée est affichée à l’écran avec une précision à **2 chiffres après la virgule**.

Enfin lorsqu’on appuie sur “Ctrl+C” alors le programme s’arrête et le GPIO est remis à zéro.

Après plusieurs tentatives j'avais considéré ce code tel qu'un échec, (alors qu'il est bien fonctionnel) et j'ai programmé un deuxième script python (voir <https://pastebin.com/k3jccZUd>) :

Pour ce code j'ai utilisé la bibliothèque "**spidev**" pour communiquer avec le convertisseur analogique-numérique (CAN) via le bus **SPI**⁹ du Raspberry Pi et comme pour le précédent code la bibliothèque "**time**" pour gérer les délais d'attente.

J'ai ensuite créé un objet **SpiDev** et je l'ai configuré pour qu'il utilise le bus **SPI 0** et le **device 0**, puis j'ai défini la vitesse de transfert de données maximale à **1 MHz**.

```
4 spi = spidev.SpiDev()
5 spi.open(0, 0)
6 spi.max_speed_hz = 1000000
```

J'ai également défini la fonction "**readadc()**" qui prend en entrée le numéro de l'entrée analogique à lire, donc compris entre **0 et 7** pour mon cas et renvoie la valeur numérique de la tension par le CAN.

```
8 def readadc(adcnun):
9     if ((adcnun > 7) or (adcnun < 0)):
10         return -1
11     r = spi.xfer2([1, (8 + adcnun) << 4, 0])
12     adcout = ((r[1] & 3) << 8) + r[2]
13     return adcout
```

La première instruction permet de vérifier si le canal est valide sinon le code renvoi -1. Ensuite l'instruction suivante envoie les commandes nécessaires au CAN pour lire la valeur sur le canal spécifié, enfin la dernière instruction rassemble la valeur lue en combinant les octets renvoyés par le CAN.

En dernière étape, j'ai fait une boucle qui lit périodiquement la tension sur le canal 0 en faisant appelle à la fonction "**readadc**" puis la convertie en tension en **multipliant par 3.3** et en **divisant par 1024** la résolution du CAN.

```
15 while True:
16     value = readadc(0)
17     voltage = value * 3.3 / 1024
18     current = (voltage - 1.65) / 0.066
19     print("Courant : {:.2f} A".format(current))
20     time.sleep(1)
```

A l'aide d'un facteur de conversion spécifique à SCT-013 50A/1V, on peut convertir la tension mesurée en courant (0.066 dans mon cas). Enfin la mesure récupérée est affichée avec une précision de 2 chiffres après la virgule grâce à la variable "**format()**".

⁹ **SPI**: Une liaison SPI (pour Serial Peripheral Interface) est un bus de données série synchrone baptisé ainsi par Motorola.

Sur le site officiel **lechacal**, ils présentent une façon de lire les mesure du capteur de courant en utilisant des requêtes **JSON**¹⁰ sur **HTTP**¹¹ :

http://lechacal.com/wiki/index.php/Raspberrypi_Current_and_Temperature_Sensor_Adaptor

Cette manipulation n'est pas vérifiée par mes soins, mais dans le cadre théorique semble correcte. Ils indiquent qu'il suffit d'installer leurs fichiers de préconfiguration en ligne de commande sur le terminale :

```
sudo apt-get install lighttpd
sudo wget lechacal.com/repo/emonwrt3/lighttpd.conf -O /etc/lighttpd/lighttpd.conf
sudo /etc/init.d/lighttpd redémarrage
wget lechacal.com/repo/emonwrt3/emonwrt3_rpi_armhf_v1.0.1.deb
sudo dpkg -i emonwrt3_rpi_armhf_v1.0.1.deb
```

Une fois le téléchargement terminé, on peut accéder au lien par le navigateur :

<http://raspberrypi/cgi-bin/emonwrt3-json?last=1>

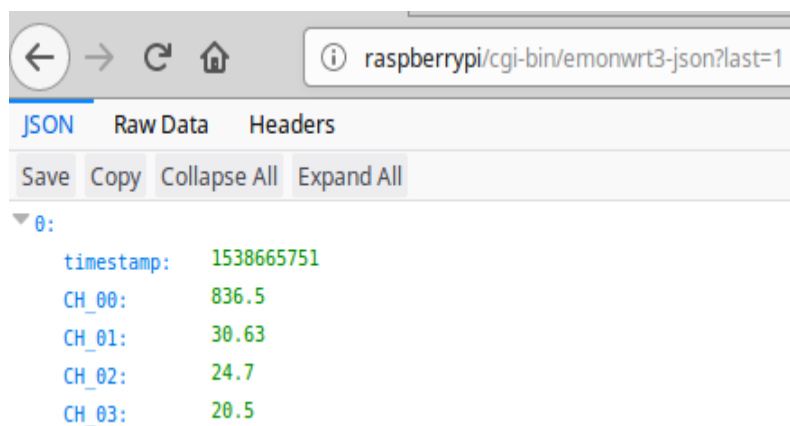


Figure 12 : Image de l'affichage JSON des mesures issue du site www.lechacal.com

Il faut modifier last = 1 au nombre d'enregistrements que l'on veut **acquérir**. Ou sinon il faut **supprimer** complètement la variable pour obtenir toutes les données (la limite par défaut est de 128 enregistrements).

¹⁰ **JSON**: JSON (JavaScript Object Notation – Notation Objet issue de JavaScript) est un **format léger d'échange de données**. Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou générable par des machines.

¹¹ **HTTP**: L'Hypertext Transfer Protocol, littéralement « protocole de transfert hypertexte », est un protocole de **communication client-serveur** développé pour le World Wide Web.

4.3 Sécurisation du système

La **sécurité des systèmes** des maisons connectées est une préoccupation majeure pour les propriétaires et les fabricants. Les systèmes de domotique sont souvent vulnérables aux attaques malveillantes, telles que les **attaques par force brute** et les **attaques par déni de service (DDOS ¹²)**. Par conséquent, il est important d'avoir une stratégie de sécurité solide pour protéger les données personnelles et la vie privée des utilisateurs. Les mesures de sécurité comprennent l'utilisation de **mots de passe forts**, la **mise à jour régulière** du firmware est une fonctionnalité disponible depuis Home Assistant et l'utilisation de protocoles de sécurité tels que le **chiffrement de bout en bout**. J'ai sécurisé MQTT avec une authentification, mais à cela on pourrait définir une qualité de service (Qos) plus élevée. Le niveau Qos 1 serait le plus optimal, pour ne pas trop charger et ralentir les performances globales, donc le Qos 1 consiste dans l'envoi du message **au moins une fois** par le broker MQTT, et le client doit **confirmer la réception**. Le broker MQTT réessayera jusqu'à ce que le client vérifie la réception si le client n'est pas accessible.

En outre, la mise en place d'un système de **backup automatique** peut aider à **prévenir la perte de données** en cas d'attaque ou de défaillance du système. Il faut également prendre en compte qu'un Raspberry Pi 4 est **assez fragile** et sa mémoire est une carte sd, malgré la fiabilité de la majorité des cartes sd, ce n'est pas rare qu'une carte sd soit endommagée après une longue utilisation. J'ai donc paramétré des backups réguliers à 16 heure (voir img-13), ces derniers sont **stockés** sur le **cloud google drive** pour une redondance maximale. Le cloud de Google Drive est tout de même assez limité en termes d'espace de stockage (15 Go, ce qui a été totalement suffisant pour l'entière durée du stage). Pour une utilisation en conditions réelles, il serait plus propice d'utiliser un service de cloud avec un plus grand espace de stockage.

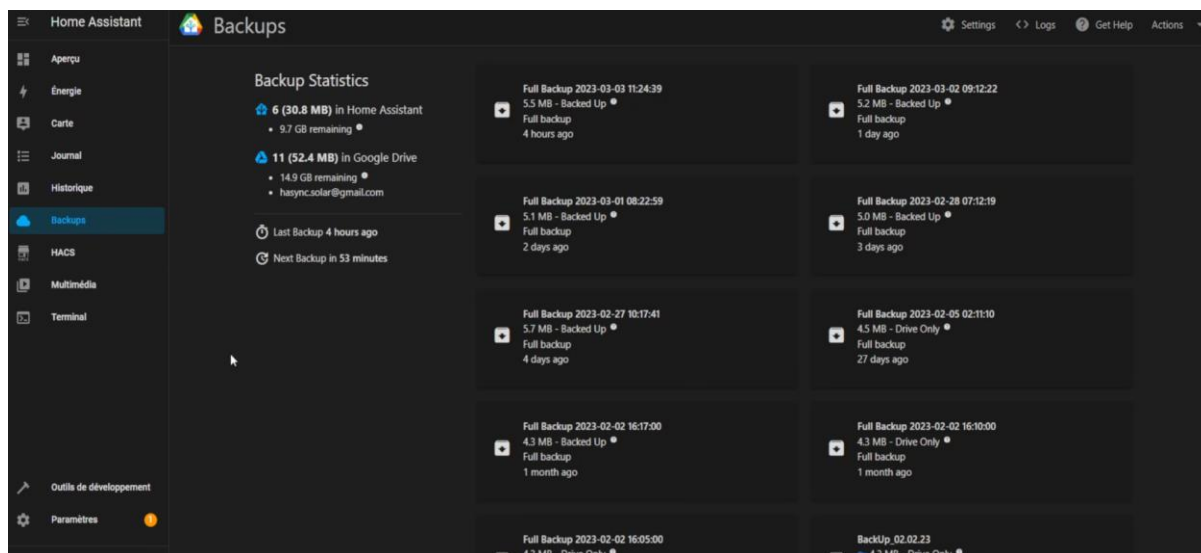


Figure 12 : Backups automatique de la configuration et des données entières de HA

¹² **DDOS** : DDoS signifie Distributed Denial of Service, ce qui se traduit littéralement par "Service de Déni Distribué". C'est une attaque informatique qui vise à rendre un site web, un serveur ou un réseau **indisponible en inondant la cible** avec un **grand nombre de requêtes** ou de connexions. Ces requêtes sont envoyées simultanément depuis un grand nombre de sources, ce qui rend la défense contre l'attaque difficile.

5 Résultats et analyses

5.1 Présentation des résultats obtenus suite à la réalisation du projet de stage

Au cours de mon stage, j'ai travaillé sur la mise en place d'un système de surveillance de la consommation d'énergie dans la domotique. La **théorie** du synchroniseur solaire obtenus montre que le système permettrait de **réduire les coûts** énergétiques de manière significative, en **réduisant les pertes d'énergie**. De plus, le système a été facilement intégré aux processus de la maison connecté avec home assistant qui devrais permettre une meilleure gestion de l'utilisation de l'énergie. En effet, ce dernier permet de fructifier pleinement l'énergie produite par les panneaux photovoltaïques sans perte conséquentes d'énergie.

5.2 Analyse des résultats

Après avoir présenté les résultats obtenus, on peut constater que la réalisation du projet n'a pas été complètement abouti, suite à une **contrainte** de "temps", je n'ai pas pu atteindre les objectifs fixés initialement, notamment en termes d'applications pratique du **synchroniseur solaire**. De plus, les résultats ont été comparés avec les normes et standards en vigueur dans le domaine, ce qui a permis de constater que le projet était conforme aux exigences (du moins, en ce qui concerne les taches effectuées).

5.2.1 Interprétation et explication des résultats obtenus

L'interprétation des résultats obtenus a permis de mettre en évidence les **points forts** et les **points faibles** du projet, ainsi que les facteurs qui ont influencé ces résultats. Les limites du projet ont également été identifiées, notamment en termes de **performances** du Raspberry Pi 4, que je rappelle, c'est le hardware sur lequel j'ai le plus travailler. Les racines du projet sont installées, tel que Home Assistant, où tout le travail se doit d'être intégré dans celui-ci, afin de faciliter la manipulation par les utilisateurs.

5.2.2 Comparaison avec les objectifs fixés initialement

La comparaison des résultats obtenus avec les objectifs fixés initialement a permis de constater que le projet a été mené à bien, quelques fois des voies sans issue ont été très retardataires, mais cela reste également un objectif principale d'améliorer mes compétences en recherche et résolutions des problèmes, une fois de plus suite à une contrainte le projet n'est pas totalement fini, le côté théorique est correctement réalisé. Les performances du projet ont été mesurées et comparées avec les normes et standards en vigueur, ce qui a permis de constater que le projet était conforme aux exigences.

6 Conclusion et perspectives

6.1 Synthèse des résultats obtenus

La synthèse des résultats obtenus a permis de mettre en évidence les principaux résultats du projet, notamment en termes de performances, d'efficacité et de délais de réalisation. Les résultats ont été présentés de manière claire et synthétique, ce qui a permis de constater que le projet a été mené à bien, bien que les objectifs fixés non pas tous été atteints.

6.2 Conclusion générale du stage

En conclusion, ce stage a été une expérience très bénéfique pour moi. J'ai eu l'opportunité de développer mes compétences techniques et professionnelles en travaillant sur un projet concret. La réalisation du projet m'a permis de mettre en pratique les connaissances théoriques acquises à l'université et de les appliquer à un environnement réel. De plus, j'ai pu acquérir des compétences non techniques telles la gestion de projet, la résolution de problèmes, etc. qui sont très importantes dans le monde professionnel.

Les résultats obtenus par suite de la réalisation du projet ont été satisfaisants, et ont confirmé que le projet a été mené à bien dans les délais impartis. L'analyse des résultats a permis de comprendre les performances du projet, de les interpréter et d'expliquer leur impact sur le projet. La comparaison des résultats avec les objectifs fixés initialement a également permis d'évaluer la réussite partielle du projet. En synthèse, ce stage a été une expérience très bénéfique pour moi et m'a permis d'acquérir des compétences précieuses pour ma carrière professionnelle future. Les résultats obtenus ont été satisfaisants et ont confirmé l'efficacité du projet. Enfin, les perspectives d'avenir et les recommandations que je propose pourraient aider à améliorer la mise en œuvre du projet dans le futur.

6.3 Perspectives d'avenir et recommandations

Dans ce dernier paragraphe, je vais aborder les perspectives d'avenir du projet et des recommandations pour améliorer la qualité et l'efficacité du projet, ainsi que pour réduire les coûts et les délais de réalisation. Tout d'abord, il est important de souligner que le projet peut énormément évoluer et il est totalement possible d'en venir à bout, il peut être étendu pour répondre à des besoins spécifiques en matière de domotique. Cela inclut l'expérimentation avec différents protocoles de communication, l'ajout de fonctionnalité supplémentaire qui contribue à l'efficacité du synchroniseur solaire, telles que la détection de mouvement (pas forcément utile pour les chambres, mais plus pour les couloirs et les bâtiments).

Je conseille de mettre l'accent sur la précision de la mesure de courant, peut-être qu'il faudrait également explorer des alternatives à Home Assistant sur Raspberry Pi 4, tel que l'installation de ce dernier sur un serveur dédiés en cloud (par exemple chez ovhcloud.com). On peut profiter de plus de performances, ce qui pourrait revenir moins chère et plus pratique qu'un.

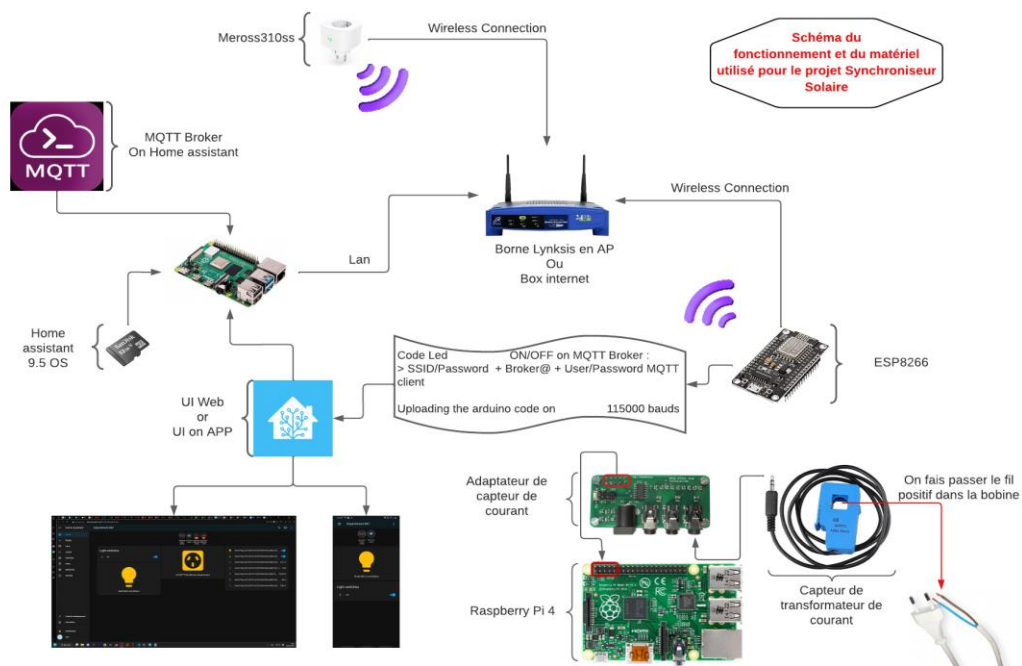
En conclusion, je lègue ce document aux futurs développeurs travaillant sur ce projet, ce qui permettra de réduire les délais de réalisation.

7 Annexes

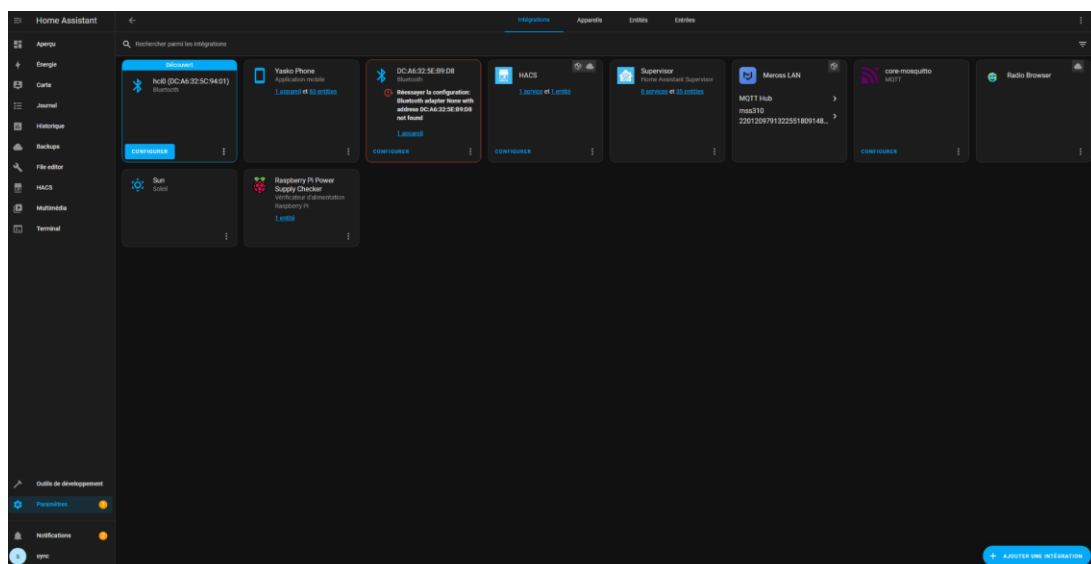
7.1 Liste des annexes : Figures, images, Screenshot, documents, etc...

Les images sont disponibles également sur mon GitHub :
https://github.com/Yasko0x000/Projet_de_Stage_Synchroniseur_Solaire/tree/master/img

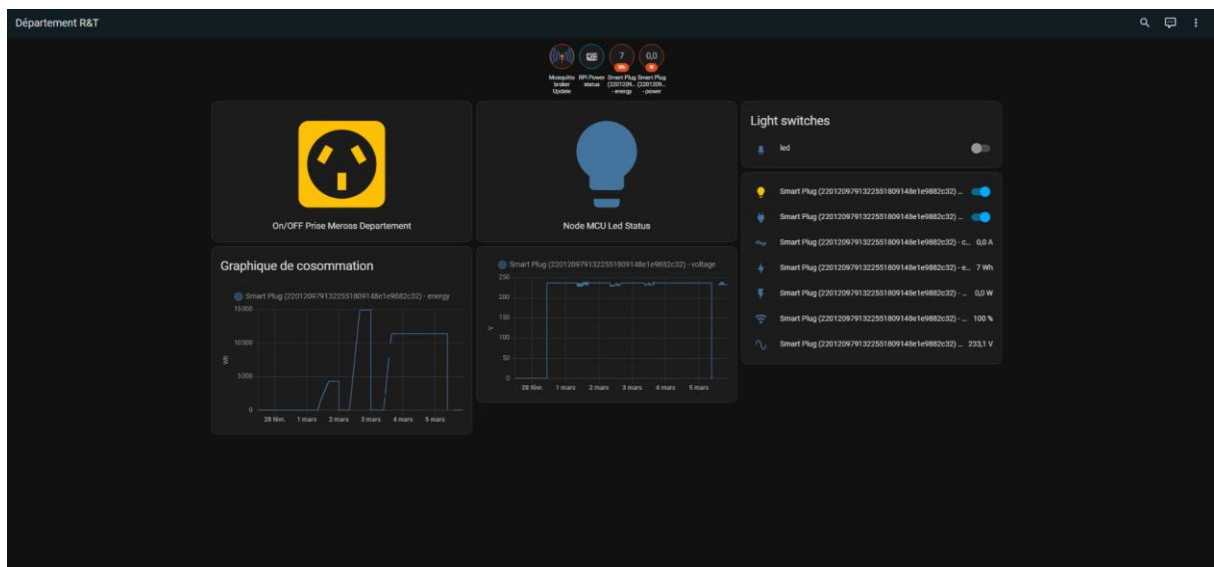
- **Figure 1 : Schéma visuel de l'utilité d'un synchroniseur solaire**
 - <https://lucid.app/lucidchart>
- **Figure 2 : Monday.com**
 - <https://yasko0x000.monday.com/>
- **Figure 4 : Schéma du fonctionnement et du matériel utilisé pour le projet Synchroniseur Solaire**



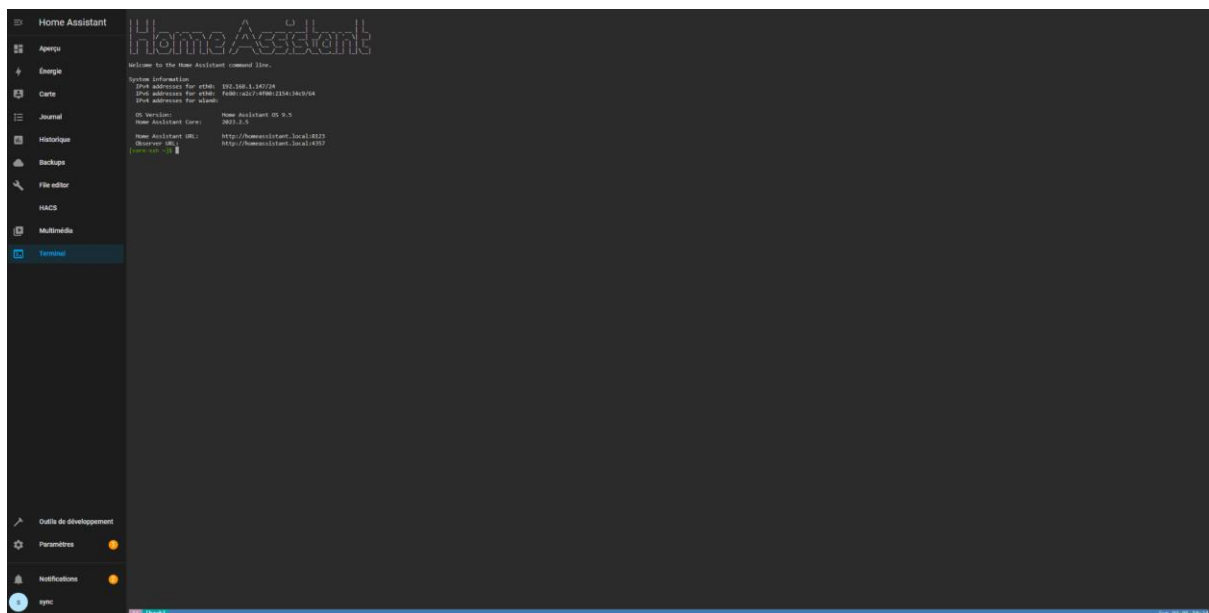
- **Figure 5 : Page d'accueil des intégrations et des modules complémentaires**



- **Figure 6 : Rendu finale du tableau de bord**



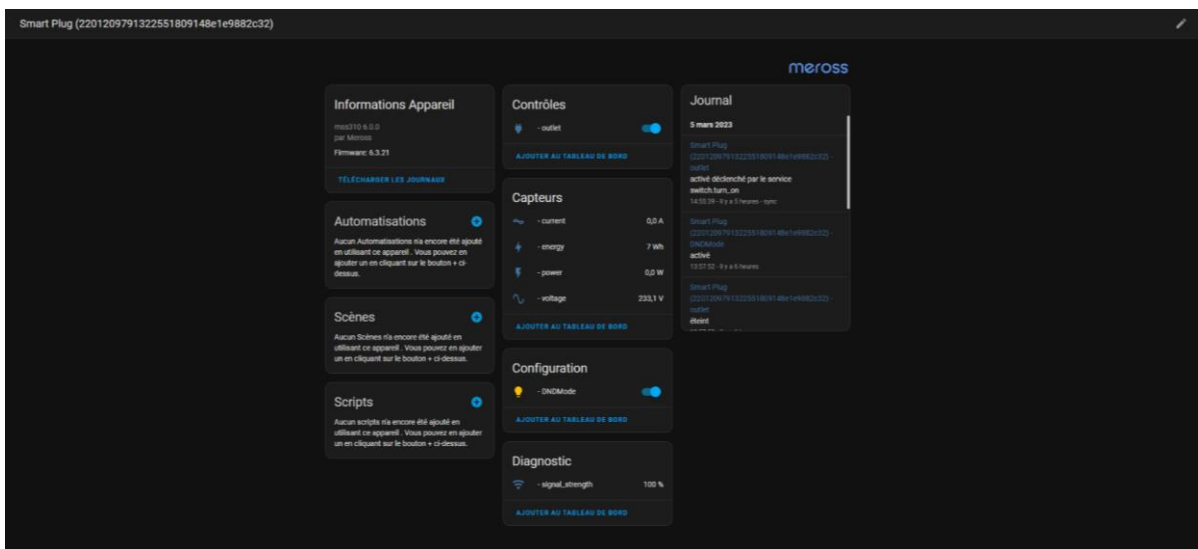
- **Figure 7 : Bannière de HA sur le terminale depuis ssh**



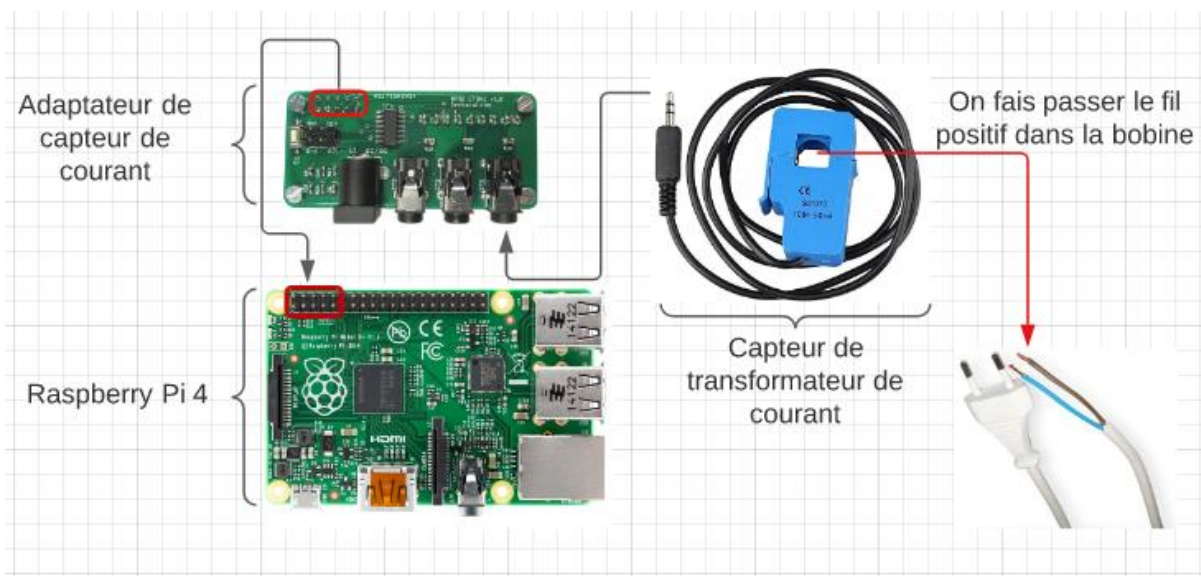
- **Figure 8 : Prise connecté Meross MSS310**
 - <https://www.amazon.fr/Intelligente-Compatible-SmartThings-Programmable-Consommation/dp/B08L3C49LJ>



- Figure 9 : Page de configuration de Meross LAN



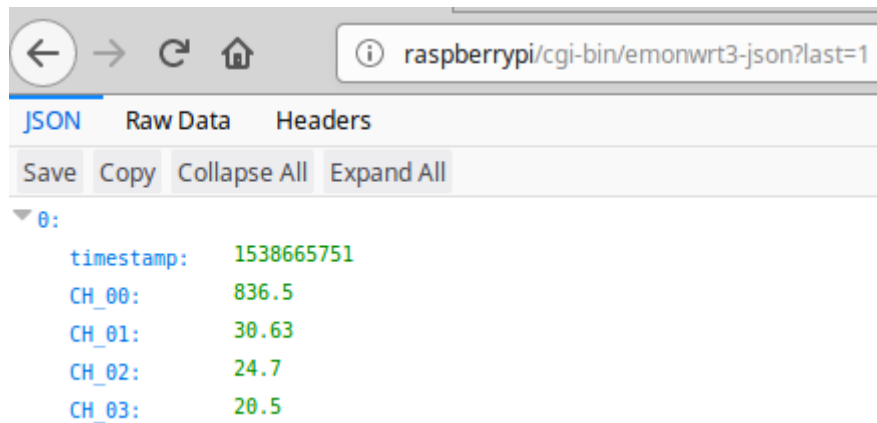
- Figure 10 : Plan du capteur de mesure de courant



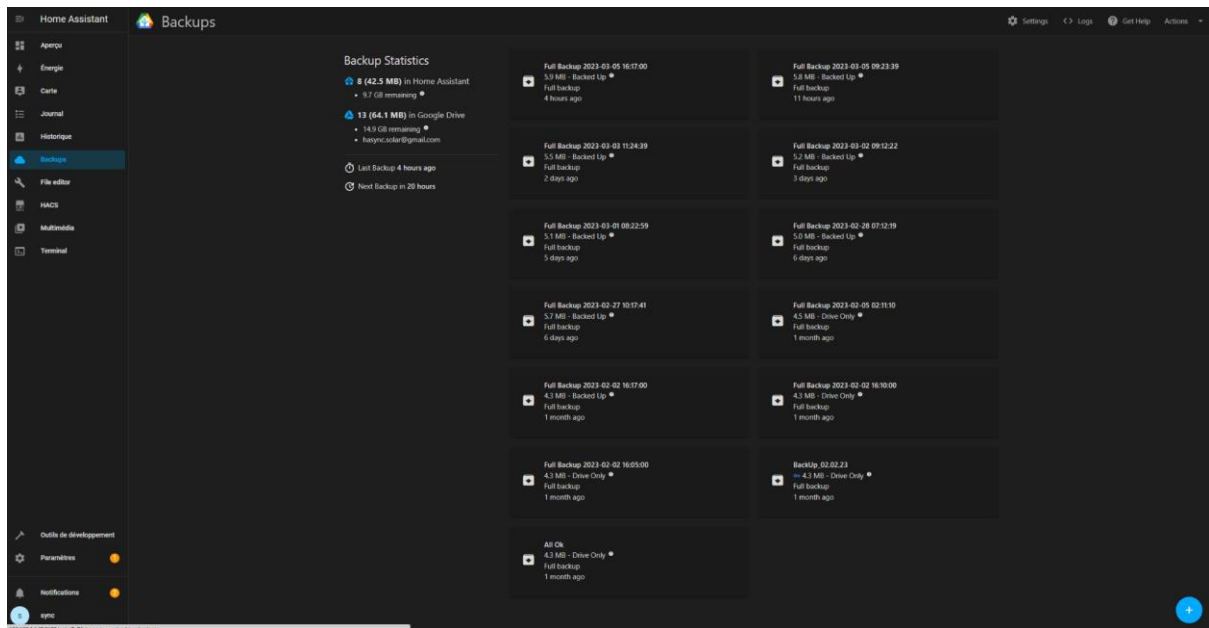
- Figure 11 : Image montrant la bonne façon de mesurer
 - http://lechacal.com/wiki/index.php/Frequently_Asked



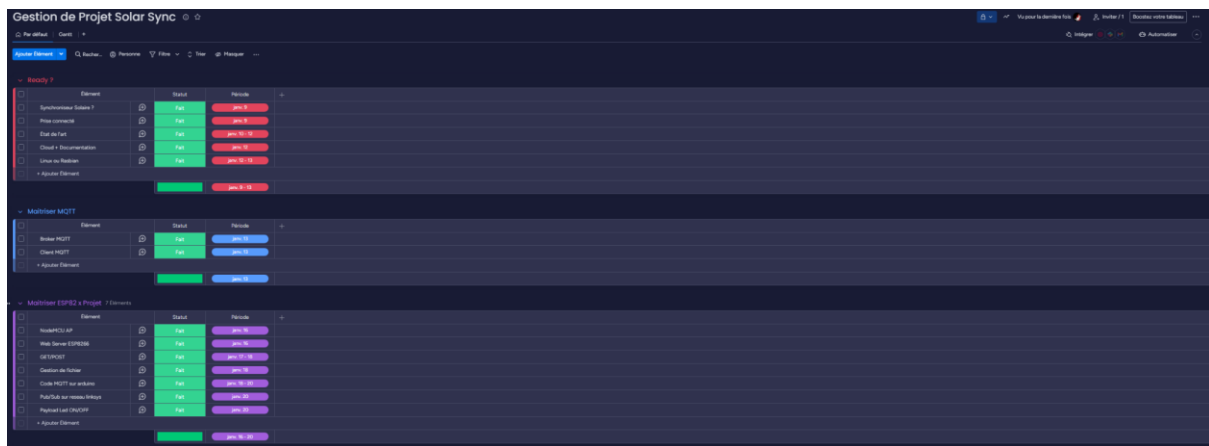
- **Figure 12 : Image de l'affichage JSON des mesures**
 - http://lechacal.com/wiki/index.php/Raspberrypi_Current_and_Temperature_Sensor_Adaptor



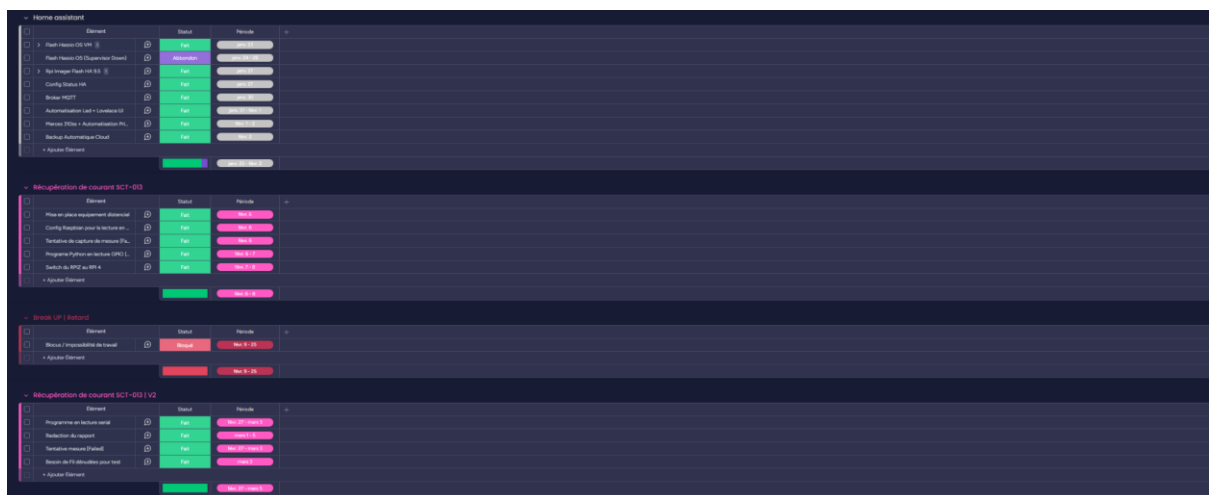
- **Figure 12 : Backups automatique de la configuration et des données entières de HA**



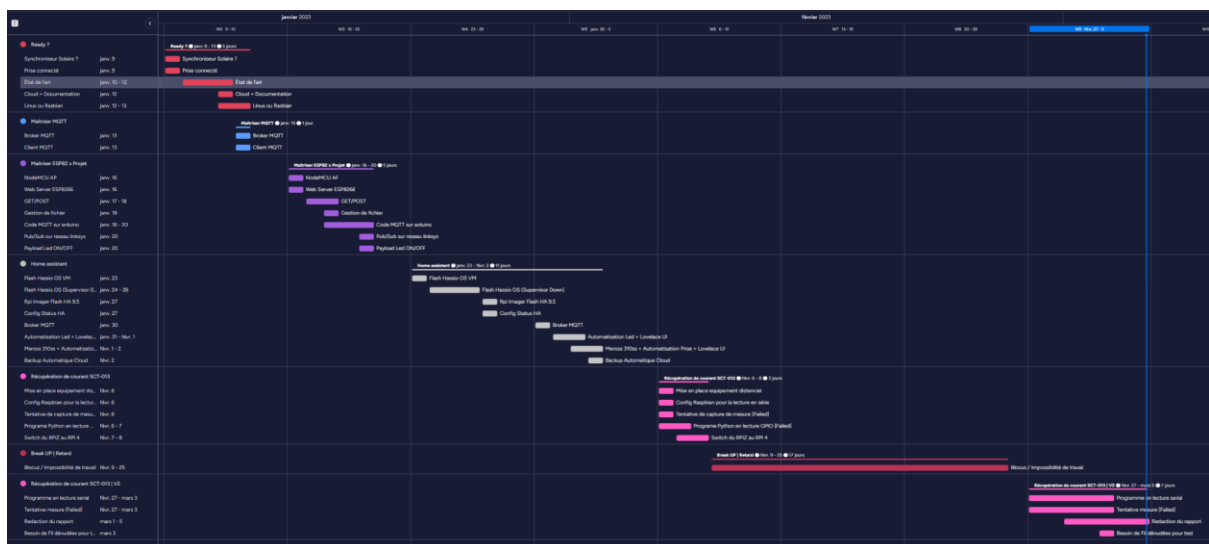
- **Img-1.1 : Monday.com | Tableau partie 1**



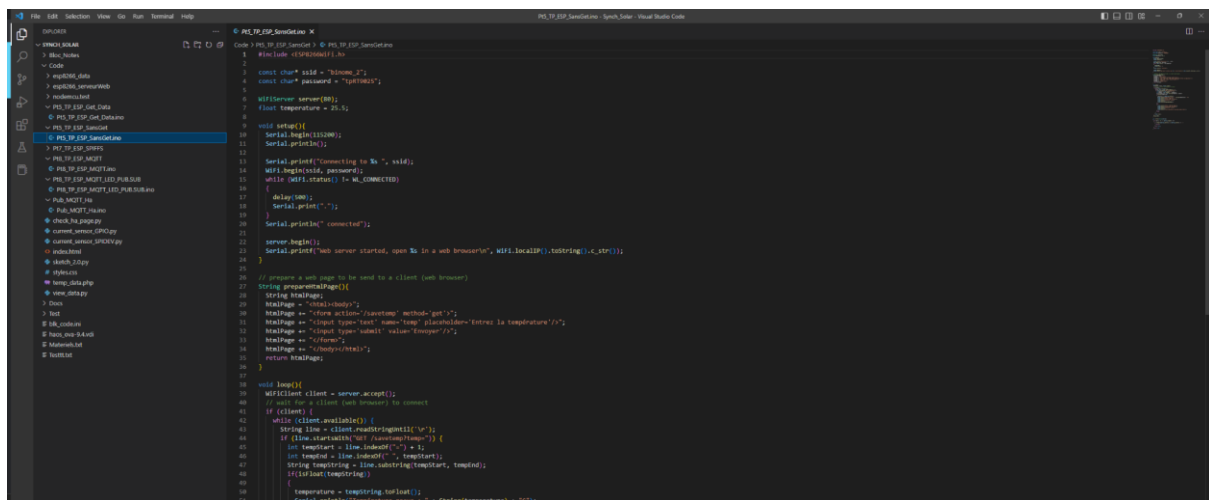
- **Img-1.2 : Monday.com | Tableau partie 2**



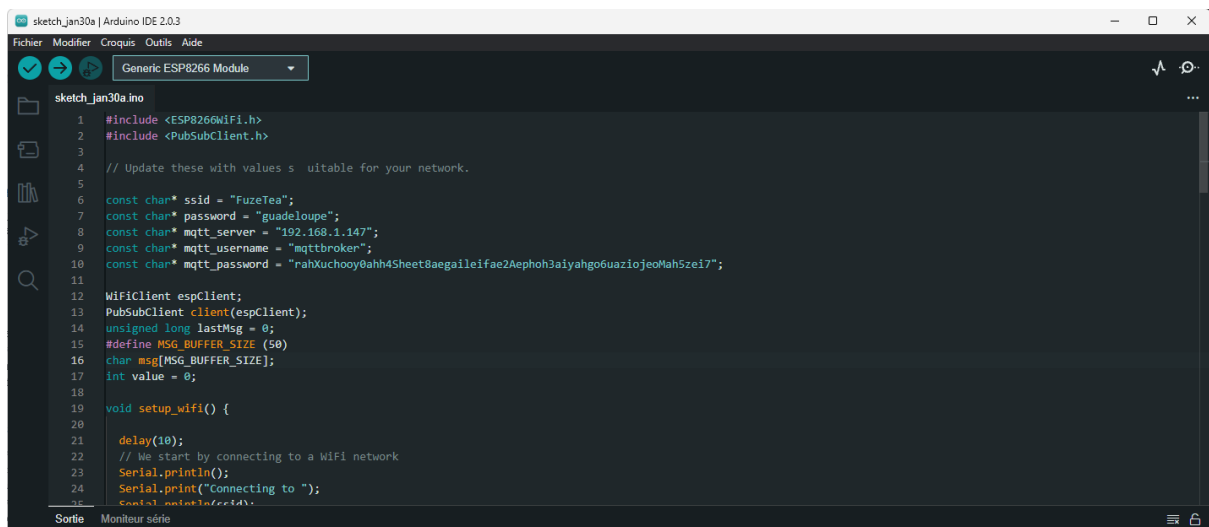
- **Img-1.3 : Monday.com | Gant sur la durée du stage**



- **Img-2.1 : Visual Studio Code**

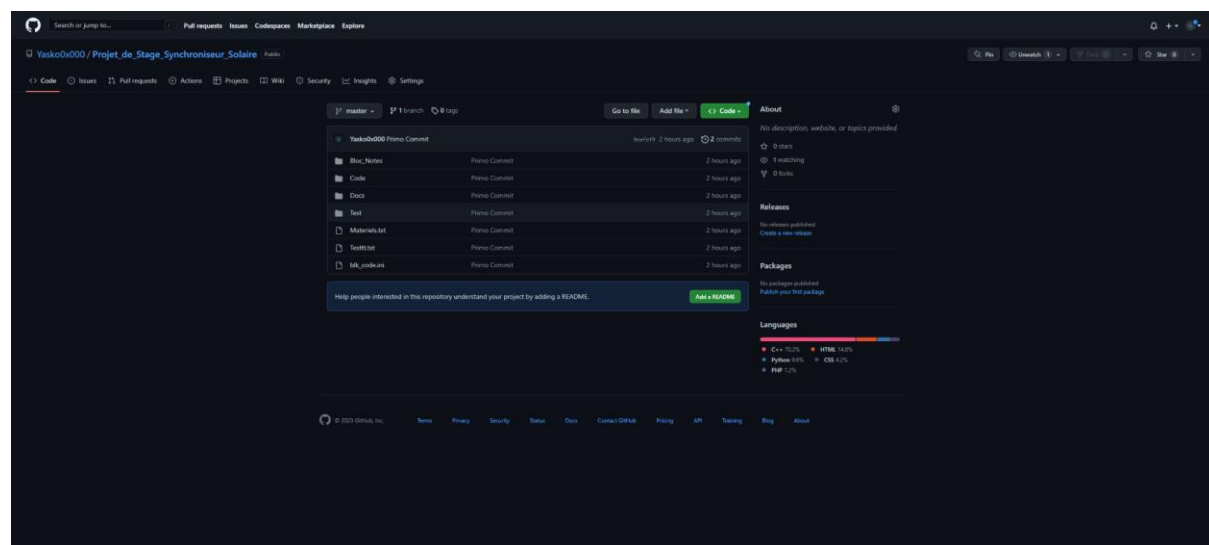


- **Img-2.2 : Arduino IDE**



- **Img-2.3 : Github**

- https://github.com/Yasko0x000/Projet_de_Stage_Synchroniseur_Solaire



- **Img-3.1 : Échange avec mon tuteur**

▣ Mesure de courant SCT-013-050A/1V

Bonjour M. Spies,

Je suis actuellement en pleine rédaction du rapport, j'ai besoin de votre aide pour représenter le schéma démontrant l'utilité et la nécessité principale d'un synchroniseur solaire (comme le schéma que vous avais fait le premier jour, avec les priorités de consommation d'électricité).

Je ne suis pas sûr de pouvoir le refaire correctement, je vous présenterais un brouillant dans la journée, pour que vous puissiez, si cela vous dérange pas, m'aider à le refaire correctement.

J'ai également une problématique :

Comment améliorer l'efficacité énergétique d'un système photovoltaïque en temps réel grâce à l'intégration d'une carte de mesure de courant et d'un bus MQTT, et comment faciliter le déploiement de cette solution en l'intégrant dans une distribution Raspberry Pi ?

Pourriez-vous me donner votre avis sur celle-ci, n'est ce pas trop long ?

Concernant la presentation de l'entreprise je sais pas trop comment en parler, j'ai pu recolter quelques infos sur le site www.societe.com, cela donne ceci :

DIGICHRONE, est une société par actions simplifiée, elle est en activité depuis 7 ans. Située à NOLAY (21340), elle est spécialisée dans le secteur d'activité du conseil en systèmes et logiciels informatiques. M. Michel STENTA est président de l'entreprise DIGICHRONE.

Enfin à propos du mesure de courant, demain sera notre dernier jour de stage, donc je pense je ne peux pas tester le code et s'il est fonctionnel, cela me derange pas d'essai de meme à travailler dessus avant le jour de la soutenance.


J'ai trouvé une autre facons de recuperer les mesures de courant avec la lecture en serial, sur le site du chacal il y a un article explicant comment faire, cependant aucun résultat.

Merci d'avance,
Bonne journée.

E. Y.



Afficher le texte entre guillemets - Répondre - Rép. à tous - Faire suivre - Autres actions

●  Expéditeur :

Bonjour Yassine,

Oui, il faut faire passer uniquement un fil, la phase par exemple dans la bobine bleue. Le neutre ne doit passer dans la bobine sinon la champ magnétique de la phase va être annulée par le neutre.

Je ne suis pas à Montbéliard aujourd'hui et demain, je ne pourrai pas venir voir ce problème. Il faut que l'on fasse une rallonge avec 2 fils séparés pour faciliter la mesure.

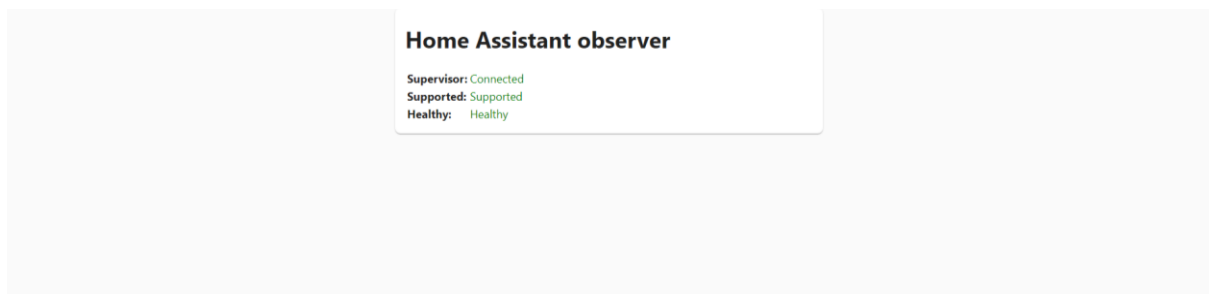
Avez-vous travailler sur le plan de votre rapport ?

Bien cordialement,

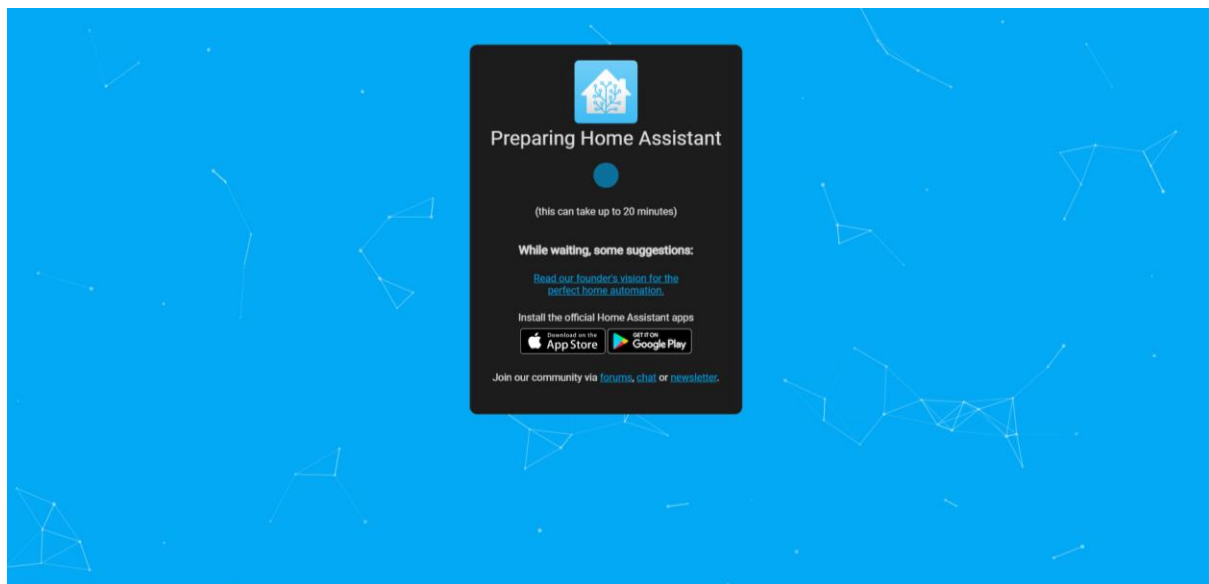
Francois Spies

Le 01/03/2023 à 08:58, yassine.el_hamioui@edu.univ-fcomte.fr a écrit :

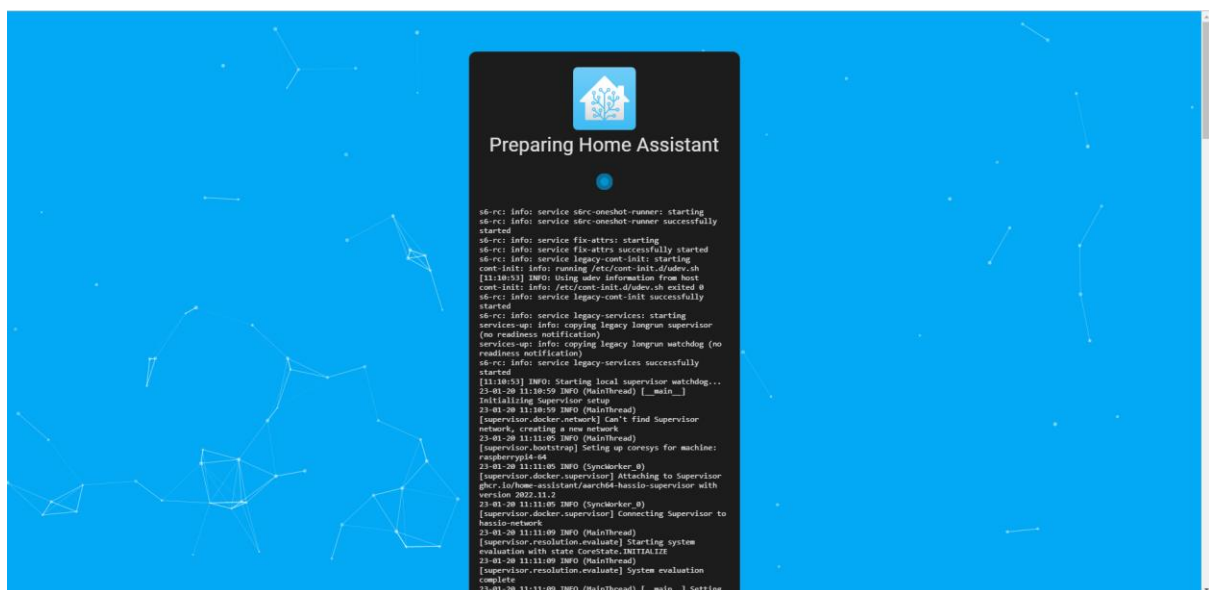
- **Img-4.1 : Home assistant Observer**



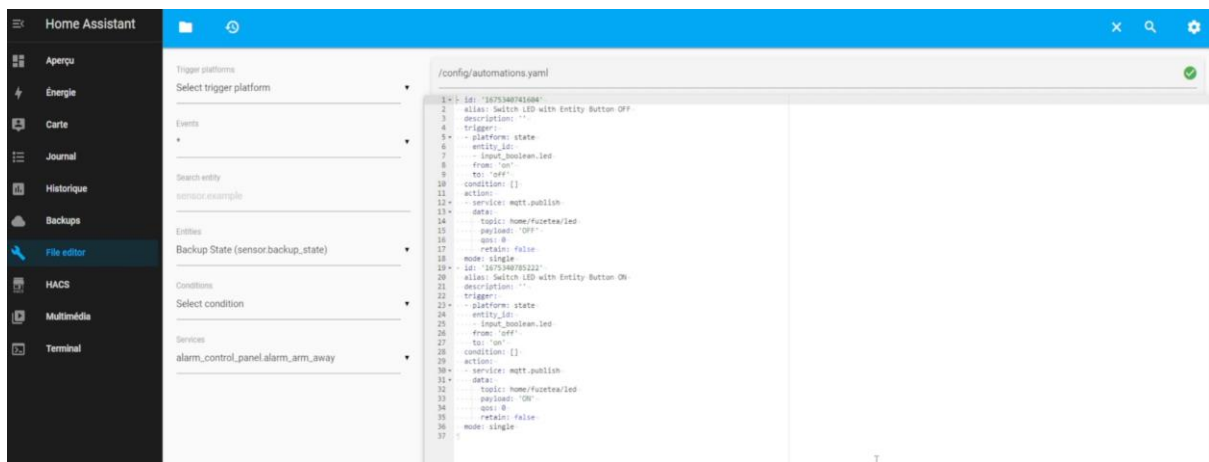
- **Img-5.1 : Preparing Home Assistant | Page de chargement**



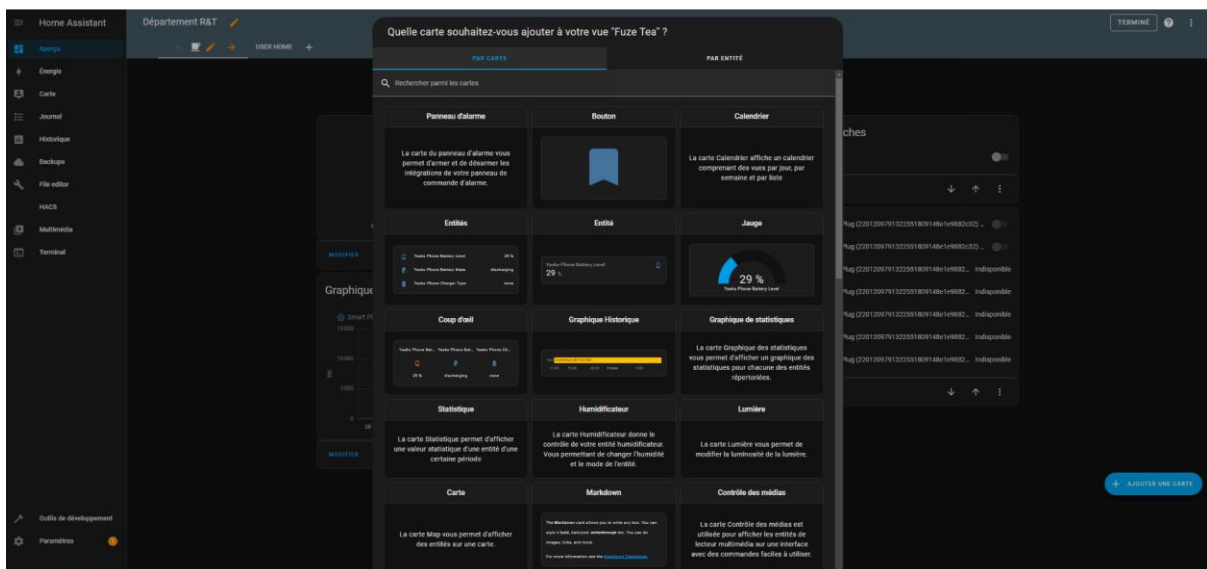
- **Img-5.2 : Preparing Home Assistant | Log**



- **Img-7.1 : File Editor YAML**



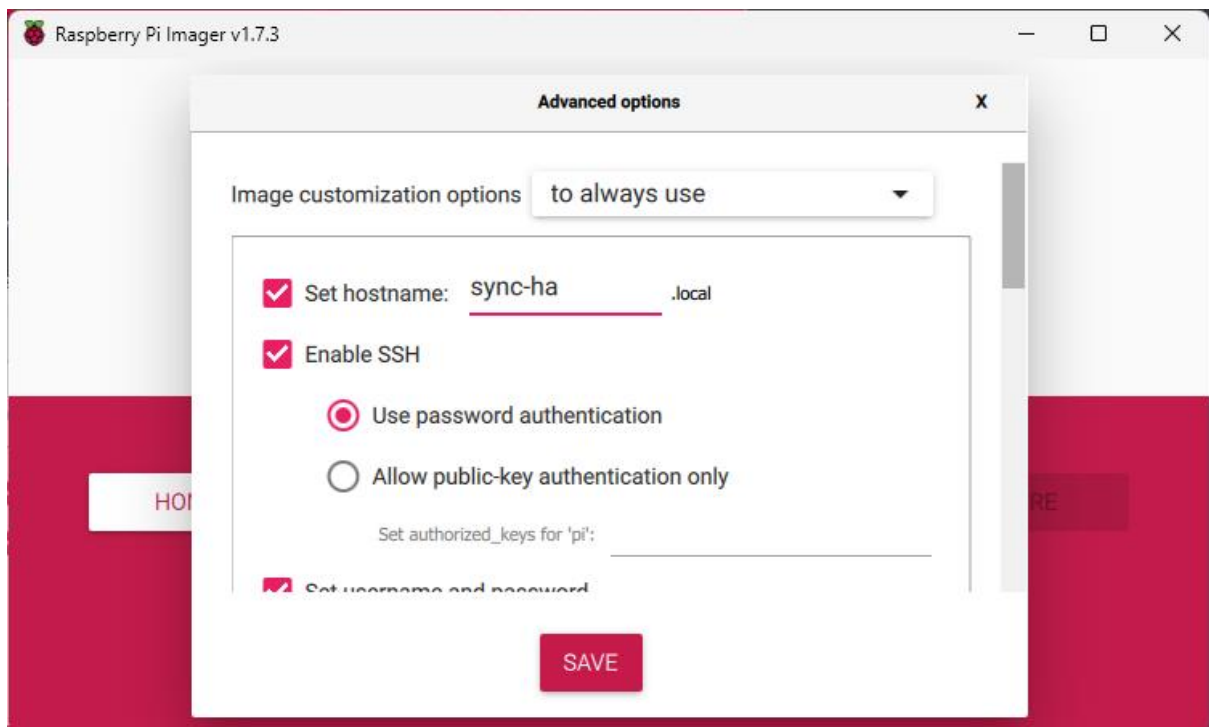
- **Img-7.2 : Interface Graphique carte du Tableau de Bord**



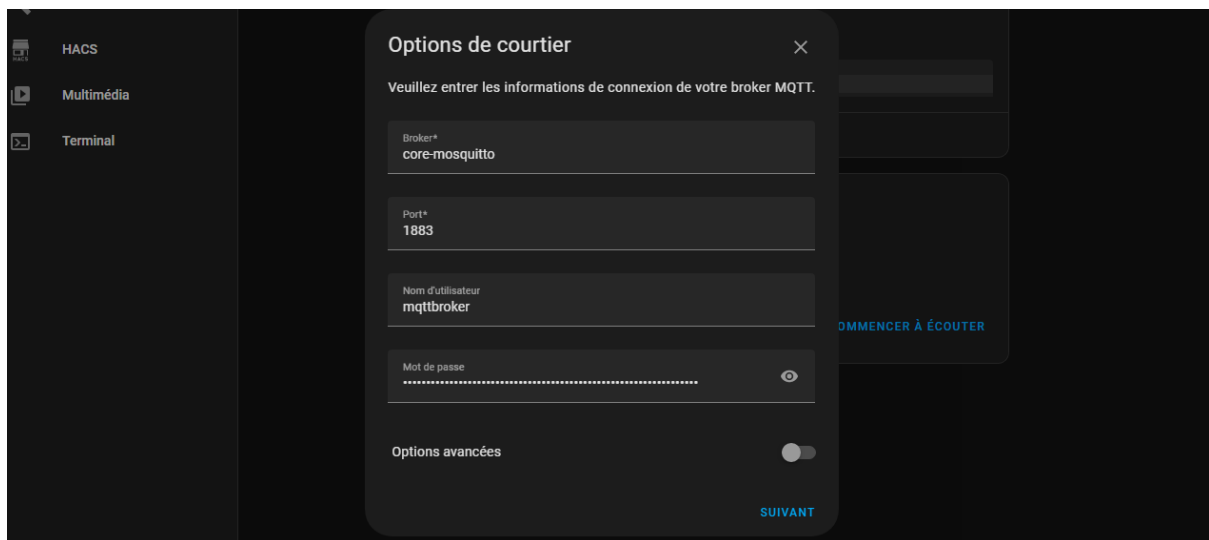
- **Img-8.1 : Raspberry Pi Imager | Écriture de HA**



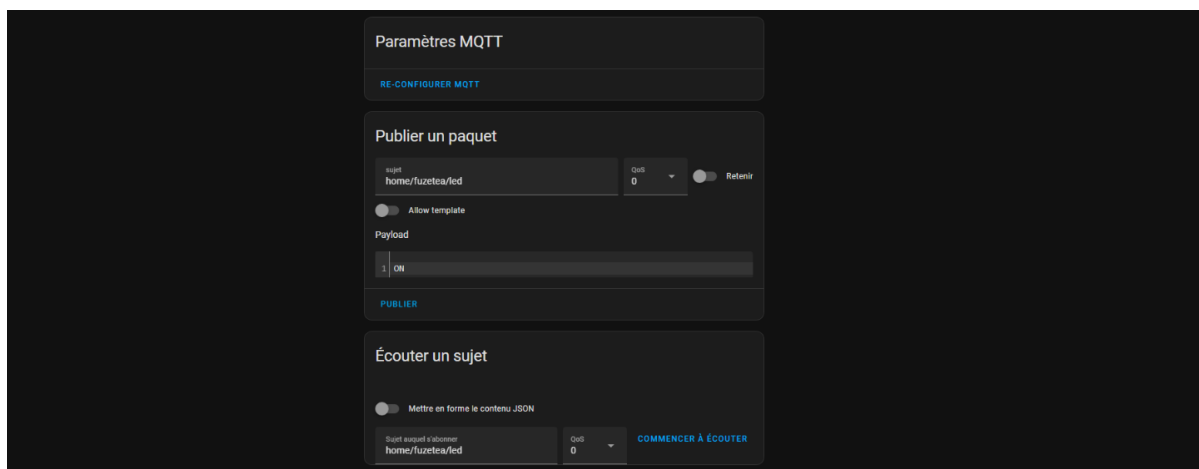
- **Img-8.2 : Raspberry Pi Imager | Options avancé**



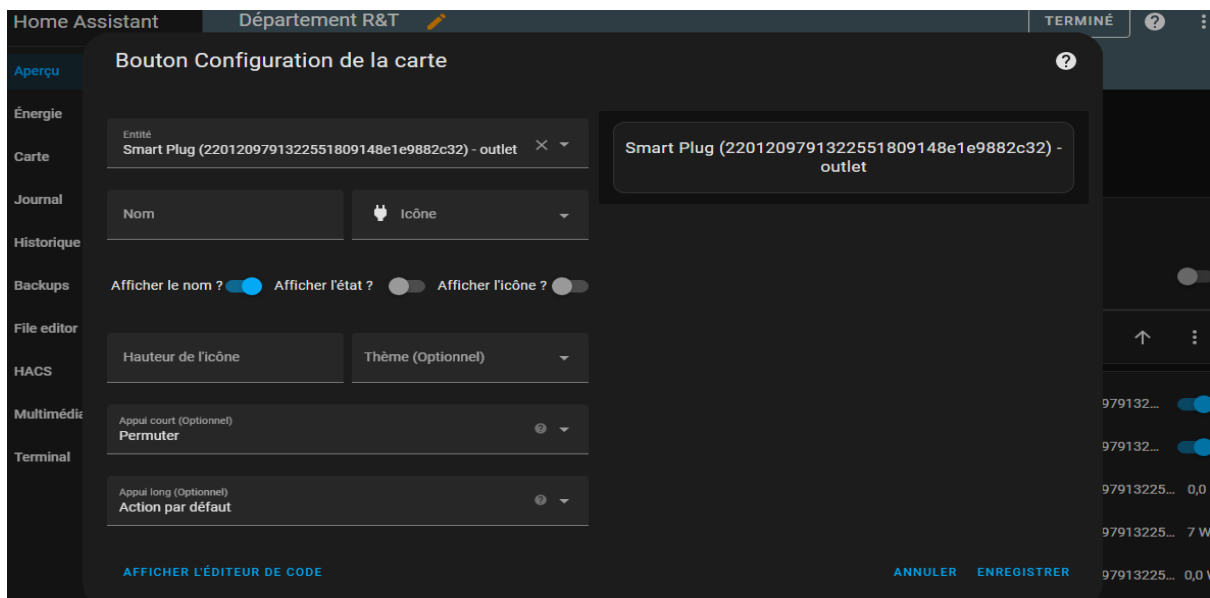
- **Img-9.1 : Mosquitto-core | Page de configuration**



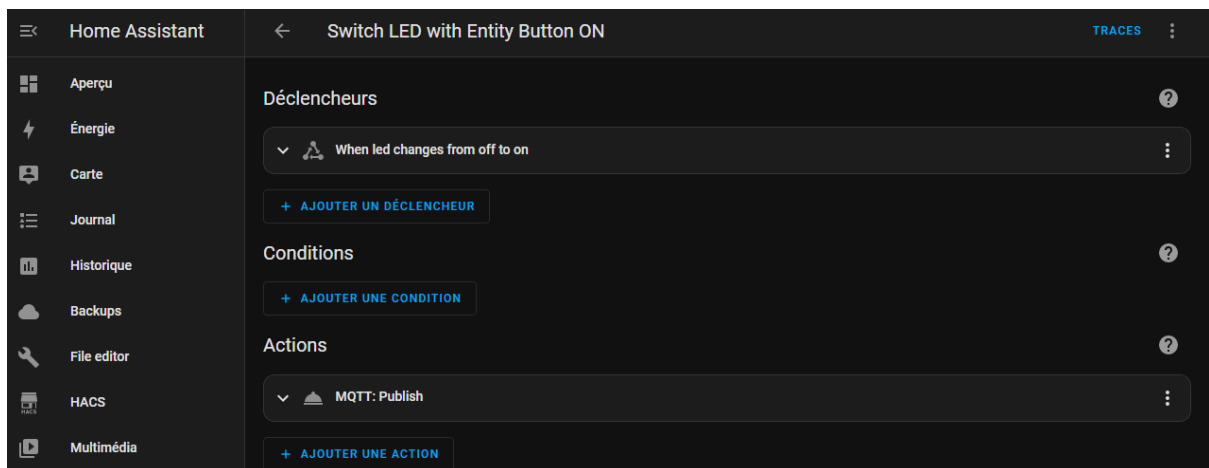
- **Img-9.2 : Mosquitto-core | Payload manuellement**



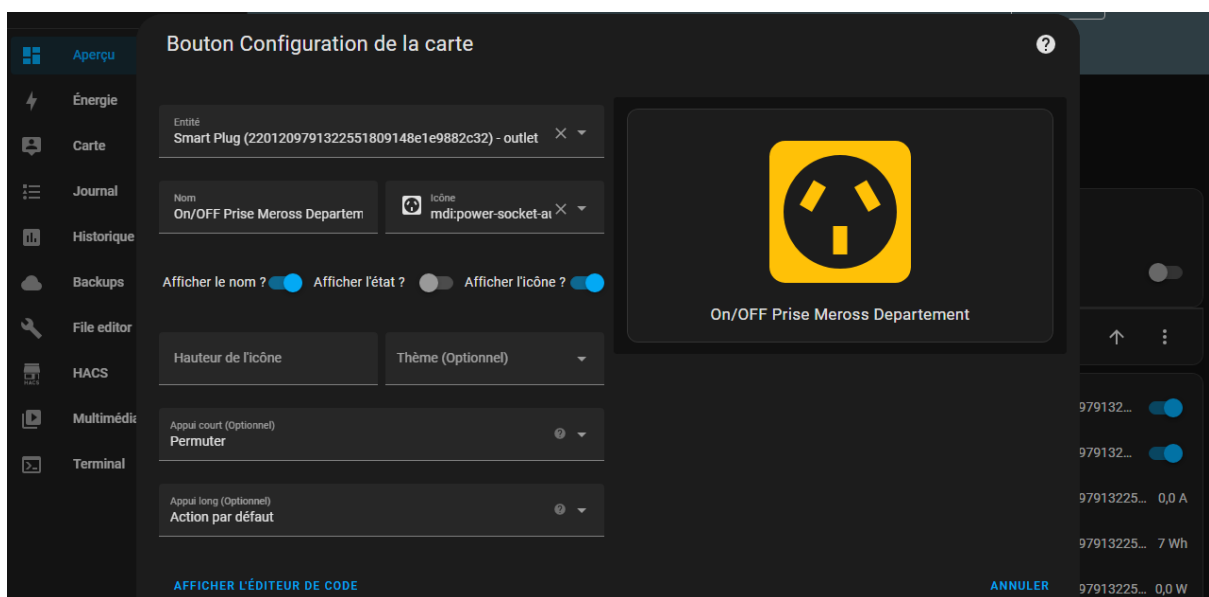
- **Img-9.3 : Interface graphique du tableau de bord | Ajout de la LED**



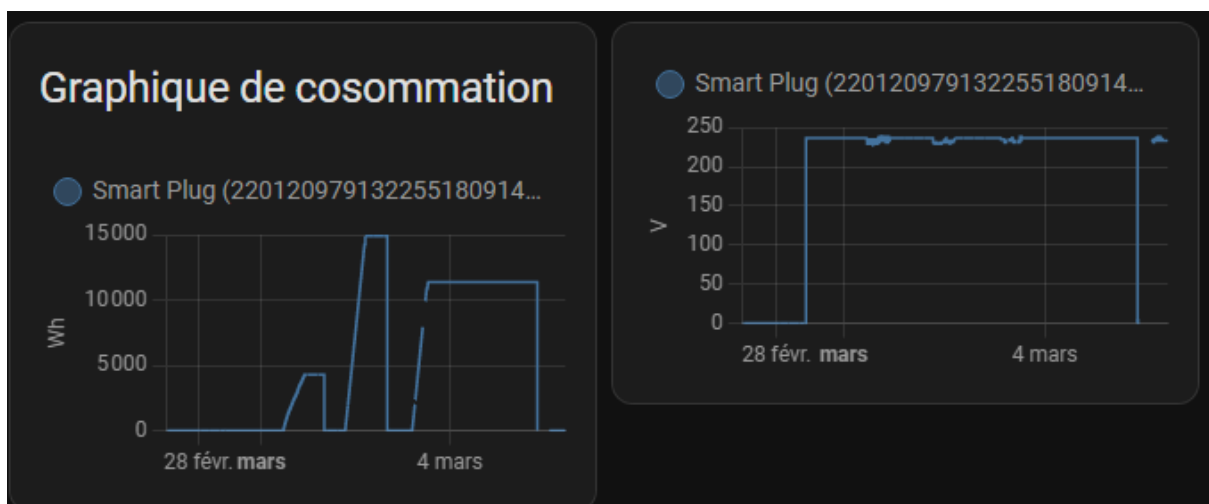
- **Img-10 : Interface graphique des automatisations**



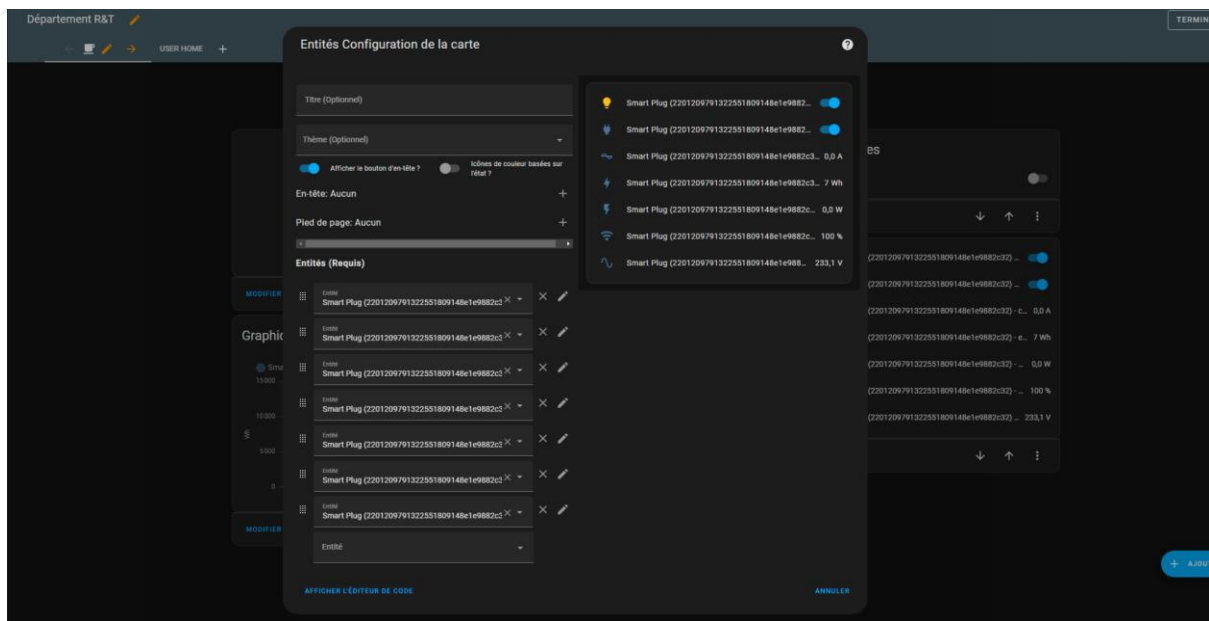
- **Img-11.1 : Interface graphique du tableau de bord | Ajout du bouton de la Prise MSS310**



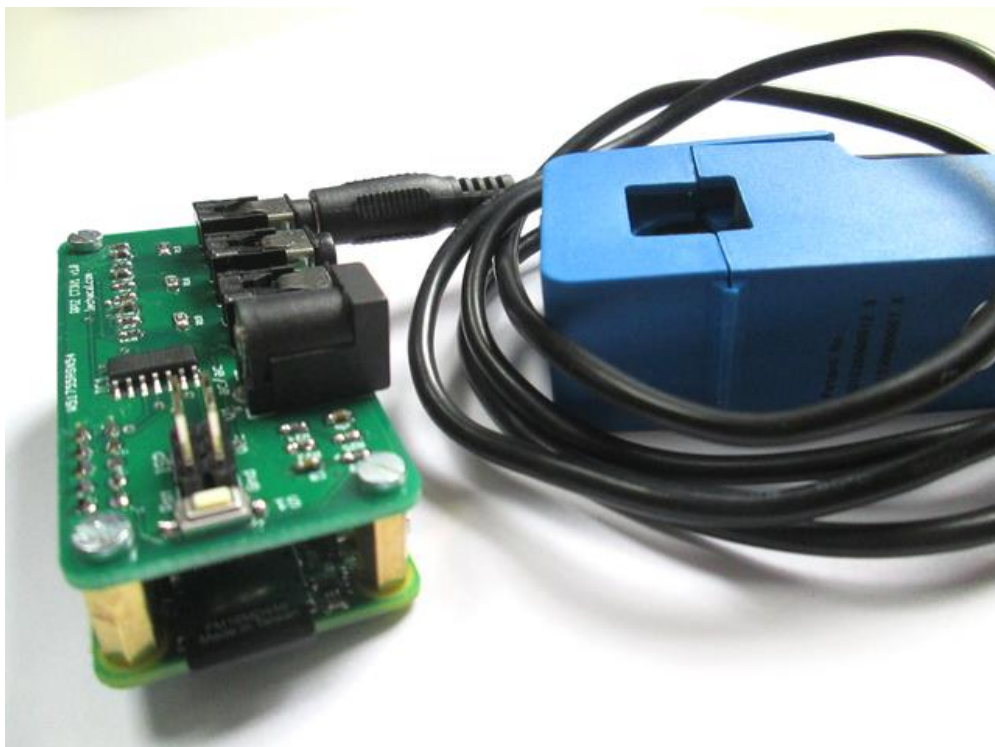
- **Img-11.2 : Graphique de consommation de la prise connecté**



- **Img-11.3 : liste des informations générales de la prise**



- **Img-12.1 : RPIZ_CT3V1**
 - http://lechacal.com/wiki/index.php?title=File:IMG_1854_small.png



- **Img-12.2 : Exécution du script de mesure de courant**

```

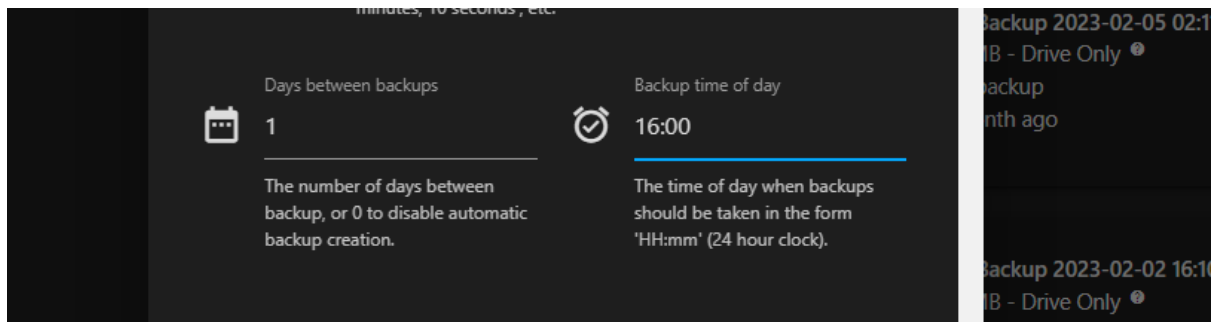
pi@raspberrypi: ~
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
^CTraceback (most recent call last):
  File "/home/pi/current_sensor.py", line 27, in <module>
    time.sleep(1)
KeyboardInterrupt

root@raspberrypi:/home/pi# nano current_sensor.py
root@raspberrypi:/home/pi# python3 current_sensor.py
/home/pi/current_sensor.py:7: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(F
also) to disable warnings.
  GPIO.setup(18, GPIO.IN)
Current: -23.04A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
Current: -22.994999999999997A
^X^CTraceback (most recent call last):
  File "/home/pi/current_sensor.py", line 27, in <module>
    time.sleep(1)
KeyboardInterrupt

root@raspberrypi:/home/pi# |

```

- **Img-13 : Heure de backup**



9 Bibliographie

9.1 Liste des références bibliographiques utilisées pour le projet de stage

- **Les systèmes domotiques intelligents :**
 - <https://www.libow.fr/blog-avis/blog/a-quoi-sert-un-systeme-domotique/>
- **Quelles sont les marques qui vendent des prises connectées ? :**
 - <https://www.lesnumeriques.com/prise-connectee/quelle-prise-connectee-choisir-a172141.html>
- **Meross MSS310 :**
 - <https://fccid.io/2AMUU-MSS310/User-Manual/User-Manual-3781399>
 - <https://www.meross.com/smart-plug/smart-plug-google-home/6>
- **Home assistant :**
 - <https://ieeexplore.ieee.org/abstract/document/5587059>
 - <https://www.sciencedirect.com/science/article/pii/S1877050917317118>
 - <https://devotics.fr/tag/home-assistant/>
 - <https://www.ca-sert-a-quoi.com/articles/domotique/presentation-de-la-solution-domotique-home-assistant/>
 - <https://www.dusuniot.com/fr/blog/what-is-an-iot-gateway/>
- **Les onduleurs réseau :**
 - <https://www.lepanneausolaire.net/les-onduleurs-reseau.php>
- **Normes de la domotique :**
 - <http://biblionumeric.epac-uac.org:8080/jspui/handle/123456789/3266>
 - <https://www.g-n-i.ch/data/13173662741108GNI-Bulletin.pdf>
 - <https://www.maison-et-domotique.com/11-choix-de-la-norme-domotique/>
- **Méthode Scrum :**
 - https://books.google.fr/books?hl=fr&lr=&id=c-eq_HR6Rh0C&oi=fnd&pg=PR5&dq=m%C3%A9thode+Scrum&ots=NMCQMihUiS&sig=4ymRhN8XGuV4XNX1j77naUWR2YE#v=onepage&q=m%C3%A9thode%20Scrum&f=false
 - <https://www.cairn.info/revue-i2d-information-donnees-et-documents-2016-2-page-12.htm?platform=hootsuite, https://hal.science/hal-03089815v3/preview/Scrum21f2.pdf>
- **Monday :**
 - <https://monday.com/lang/fr/>
- **Dispositif de Mesure de courant sans contact :**
 - [https://www.semanticscholar.org/paper/Dispositif-de-mesure-de-courant-sans-cont-act-](https://www.semanticscholar.org/paper/Dispositif-de-mesure-de-courant-sans-contact-act-)
- **Dispositif de magnétomètres pour la mesure de courant :**
 - <https://www.semanticscholar.org/paper/Dispositif-de-magnétomètres-pour-la-mesure-de-en-%3A-Wilsch/21afdc314c9c4aa13abc21304c536b94e82ed8c2>
- **Circuit LC :**
 - https://fr.wikipedia.org/wiki/Circuit_LC
- **Toroïde de Rogowski :**
 - https://fr.wikipedia.org/wiki/Enroulement_de_Rogowski
-
- **Bus MQTT :**
 - <https://ieeexplore.ieee.org/abstract/document/7977019>
 - <https://aws.amazon.com/fr/what-is/mqtt/#:~:text=Un%20client%20MQTT%20%C3%A9tablit%20une,aux%20abonn%C3%A9s%20qui%20sont%20int%C3%A9ress%C3%A9s>

- **ESP8266 :**
 - <https://www.planete-domotique.com/blog/2021/10/01/esp8266-microcontrôleur-rapport-qualite-prix-imbattable/>
 - <https://projetsdiy.fr/esp22-lire-ecrire-modifier-fichiers-spiffs-librairie/>
- **Yaml :**
 - <https://www.earthdata.nasa.gov/s3fs-public/imported/YAML%201.2%20Spec.pdf>,
 - <https://yaml.org/>, <https://docs.fileformat.com/fr/programming/yaml/>
 - <https://www.redhat.com/fr/topics/automation/what-is-yaml>
- **Capteur de mesure de courant :**
 - https://books.google.fr/books?id=b3-SFGPYJs4C&lpg=PA1&ots=Ra3j6_raR-&dq=Capteur%20de%20mesure%20de%20courant&lr&hl=fr&pg=PA1#v=onepage&q=Capteur%20de%20mesure%20de%20courant&f=false
 - https://d1wqtxts1xzle7.cloudfront.net/51842859/Capteurs4_GSI-libre.pdf?1487369397=&response-content-disposition=inline%3B+filename%3DPierre_Bonnet_Master_GSI_Capteurs_Chaine.pdf&Expires=1678049646&Signature=Cf4QBAq6io7AtQNFWFNyt8AjjRtMVooz~ODlvXRq~sbo6d2mUj8OfBECyB2U1lYv7-cla2t3biS~VSqAP4Sud-o2evNEJTGaysqMyb3M5~m00oQzWiNsHTsUwPzLPIhyGOLVI4xXIh6LN5ddB00m45xE D7vCzHFZdsjudszFUEETm1hP02lYwr7msJ2hVWwNnwxshtXXNZunTyYWIIRgLOEYqHJLZW-4sPyRXkneiE-riRowDnHmvYJK7XOMAKVR-4VoFxdxkUvDjpmP177g48qksPwXLveG7zua~DT88L2ARAOUhihvBG28kp1p2lp~GZv4dxjYoG2qsZHYDD3FvQ_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
 - <https://hal.science/hal-02981830/document>
- **L'impact des attaques par déni de service à l'heure de l'Internet des Objets :**
 - <https://www.globalsign.com/fr/blog/limpact-des-attaques-par-deni-de-service-a-iot>
- **Qos :**
 - <https://www.sciencedirect.com/science/article/abs/pii/S1084804516303186>
 - <https://ieeexplore.ieee.org/abstract/document/779924>
- **Prise connecté:**
 - <https://pomme-kit.fr/installer-une-prise-connectee-wifi-meross-compatible-homekit/#:~:text=Apr%C3%A8s%20avoir%20branch%C3%A9%20la%20prise,mod%C3%A8le%2C%20MSS210%20dans%20notre%20cas>
- **JSON :**
 - <https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON>