الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Domaine : Sciences et Technologie
Filière : Electronique
Spécialité : Electronique Des Systèmes Embarqué - Master 2

Projet de fin d'étude

Réseau domotique local

Nebbak Mohamed El Mahdi

Promoteur : mr Kabir.Y

Année Universitaire 2023-2024

Remerciement

Aujourd'hui, je commence par exprimer ma profonde reconnaissance envers Dieu pour avoir facilité mon parcours jusqu'ici. C'est grâce à Sa bienveillance que mes parents et mes proches ont pu me soutenir tout au long de mes études.

Je souhaite exprimer ma sincère gratitude envers mon directeur de thèse, le Dr Kabir, pour ces précieux soutiens, leurs conseils avisés et leur patience infinie durant mon parcours deMaster. Leurs connaissances approfondies et leur expérience m'ont guidé et encouragé à chaque étape.

un immense merci à mes chers parents. Leur soutien inconditionnel, leur amour et leurs sacrifices ont été essentiels dans mon parcours. Leurs encouragements constants et leur confiance en moi m'ont donné la force nécessaire pour persévérer et réussir

Enfin, à mes chers amis, je vous suis reconnaissant pour votre amitié et votre soutien indéfectible. Votre présence a été une source de motivation et de réconfort tout au long de cette période intense. Sans vous, cette réussite aurait été plus difficile à atteindre.

Dédicaces

- À ma mère bien-aimée, qui est ma source de vie, d'amour et d'affection inépuisable,
- À toute ma famille, qui représente pour moi une source constante d'espoir et de motivation,

avisés et leur patience infinie,

- À tous mes amis, et particulièrement à mes cher amis bendada ahmed et imad lahachi,
- À tous ceux qui m'ont apporté leur aide, de près ou de loin, tout au long de ce parcours.

ص خلم

إنترنت الأشياء عبارة عن شبكة من الأجهزة الخاصة المترابطة، مثل الهواتف الذكية والمنازل الذكية، المجهزة بأجهزة ومعاينة البيانات المتقدمة ومشاركتها في الوقت الفعلي. تصبح هذه الأجهزة مستقلة، مما يتوقع من الأسباب والتكاليف مع راحة المستخدم. وهي مجهزة بالأجهزة والمحركات والقدرات الحسابية، المعتمدة في بيئات البرنامج، وتبديل البيانات الرقمية. الساعات الذكية لمراقبة الصحة، والمنازل الذكية، قادرة على التحكم بالأجهزة عن بعد، تحويل الكهرباء واحتمالات جديدة من خلال منصات وتطبيقات الاتصالات الخاصة بالمستخدمين. يلخص هذا الملخص التفاعل والتأثيرات المحتملة لتكنولوجيا إنترنت الأشياء
الهدف من هذا الإدراك هو إنشاء شبكة محلية لا تعتمد على خدمات غير مرتبطة أو غير ضرورية مما يجعلها أكثر ثقة للقيام بالمهام المهمة

الكلمات المفتاحية: إنترنت الأشياءالاتصال، الأجهزة المادية، الهواتف الذكية، المنازل، أجهزة الاستشعار المتقدمة، البيانات في الوقت الحقيقي، الحكم الذاتي، اتخاذ القرار، المحركات، القدرات الحسابية، تبادل البيانات الرقمية، الساعات الذكية، المنازل الذكية، التحكم عن بعد، الاقتصاديات، وفرص الخدمة، والمنصات المبتكرة، واحتياجات المستخدمين

Résumé

L'Internet des Objets (IoT) est un réseau de terminaux physiques connectés, comme smartphones et maisons, équipés de capteurs avancés pour collecter et partager des données en temps réel. Les objets deviennent autonomes, réduisent les erreurs et les coûts tout en améliorant le confort des utilisateurs. Intégrant capteurs, actionneurs et capacités de calcul, ils opèrent efficacement dans des environnements complexes et échangent des données numériques. Les montres intelligentes surveillent la santé, les maisons intelligentes permettent le contrôle à distance des appareils, transformant ainsi l'économie et créant de nouvelles opportunités à travers des plateformes et applications innovantes pour répondre aux besoins des utilisateurs.
L'objectif de cette réalisation est d'avoir un réseau localisé (IoT) qui ne dépend pas des services incompatibles ou inutiles, ce qui rend plus fiable l'exécution de tâches importantes.

Mots clés : IoT, smartphones, maisons intelligentes, capteurs avancés, données en temps réel, autonomie des objets, efficacité opérationnelle, santé connectée, contrôle à distance, transformation économique, nouvelles opportunités, plateformes et applications innovantes.

Abstract

The Internet of Things (IoT) is a network of interconnected physical devices, such as smartphones and smart homes, equipped with advanced sensors for real-time data collection and sharing. These devices become autonomous, reducing errors and costs while enhancing user comfort. Integrated with sensors, actuators, and computational capabilities, they operate efficiently in complex environments, exchanging digital data. Smart watches monitor health, smart homes enable remote device control, transforming the economy and creating new opportunities through innovative platforms and applications to meet user needs. This abstract summarizes the fundamental aspects and potential impacts of IoT technology.

The goal of this realization is to have a localized (IoT) network that does not depend on unrelatable or unnecessary services making it more trusted to do important tasks.

Keywords : Internet of Things (IoT), connectivity, physical devices, smartphones, homes, advanced sensors, real-time data, autonomous, decision-making, actuators, computational abilities, digital data exchange, smart watches, smart homes, remote control, economies, service opportunities, innovative platforms, user needs

Abréviation

LAN	Local Area Network
HDD	Hard Disk Drive
SSD	Solid State Drive
CPU	Central Processing Unit
PLA	Polylactic Acid
OS	Operating System
SSH	Secure Shell
IDE	Integrated Development Environment
OTA	Over-the-Air
MQTT	Message Queuing Telemetry Transport
API	Application Programming Interface
YAML	Yet Another Markup Language
SSID	Service Set Identifier
IP	Internet Protocol
DHCP	Dynamic Host Configuration Protocol
WIFI	Wireless Fidelity
Web UI	Web User Interface
GPIO	General-Purpose Input/Output
PCB	Printed Circuit Board
R,V,O	Rouge, Vert, Orange
I2c	Inter-Integrated Circuit
VHS	Video Home System
UPS	Uninterruptible power supply
AC	Alternative Current
DC	Direct Current

Table Des Matières

Table Des Figures

Introduction

L'Internet des Objets (IoT) est un réseau composé de plusieurs types de terminaux physiques, comme les smartphones, les voitures et même les maisons, qui sont connectés en permanence à un réseau. Ces appareils utilisent des capteurs très avancés, intégrés et une connectivité réseau excellente pour être capables d'analyser et de partager des données en temps réel. Grâce à cela, ils deviennent des objets intelligents qui peuvent prendre des décisions de manière autonome ou entrer en contact avec les autres appareils. Cela permet de faire moins d'erreurs et de réduire les coûts, tout en maximisant le confort du client.

L'IoT fait que nos objets physiques du quotidien sont pourvus de capteurs, d'actionneurs et d'unités de calcul, ce qui leur permet de se faire, par exemple, suivre de façon intelligente dans des environnements complexes. Les données collectées sont ensuite échangées numériquement. Les objets ne sont plus seulement des biens, mais aussi des services à part entière. Par exemple, les montres intelligentes surveillent la santé des utilisateurs, et les maisons intelligentes rendent possible la commande à distance de différents appareils.

L'IoT est au cœur d'un mouvement de changement qui peut véritablement transformer l'économie en offrant de nouvelles formes de services et en renouvelant les opportunités. L'IoT répond aux différents besoins des utilisateurs grâce à la création de nouvelles plateformes et applications pour l'exploiter.

Dans ce projet, notre objectif est d'améliorer cette technologie en répondant à des exigences spécifiques. Nous allons realiser une plateforme de gestion pour un réseau de capteurs intelligents, permettant aux utilisateurs de contrôler et de recevoir des données de leurs objets connectés.

Chapitre 1

Serveur de contrôle

1.1 Introduction

Pour gérer nos microcontrôleurs et les faire communiquer entre eux, nous avons besoin d'un centre de communication. Celui-ci se présente sous la forme d'un serveur qui gère le trafique entre ces derniers mais aussi une interface de contrôle pour l'utilisateur. le serveur doit être branché dans le même LAN assurant la confidentialité de nos informations et trafic mais aussi la basse latence entre les composants sans oublier l'inentamée d'une connexion externe.

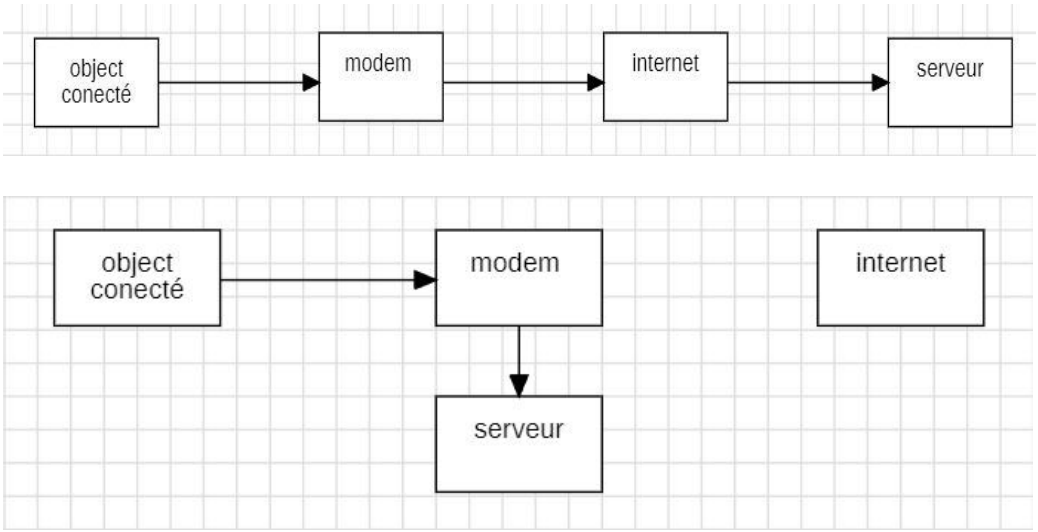


Figure 1.1: représentation de trafique entre serveur externe et local

1.2 Partie matérielle

nous avons choisi le Raspberry Pi 4 pour sa taille, sa versatilité, sa basse consommation d'énergie, mais aussi pour le support communautaire autour de ce SBC (voir Figure 1.2). Pour pouvoir interagir avec le Raspberry Pi, nous devons ajouter un périphérique

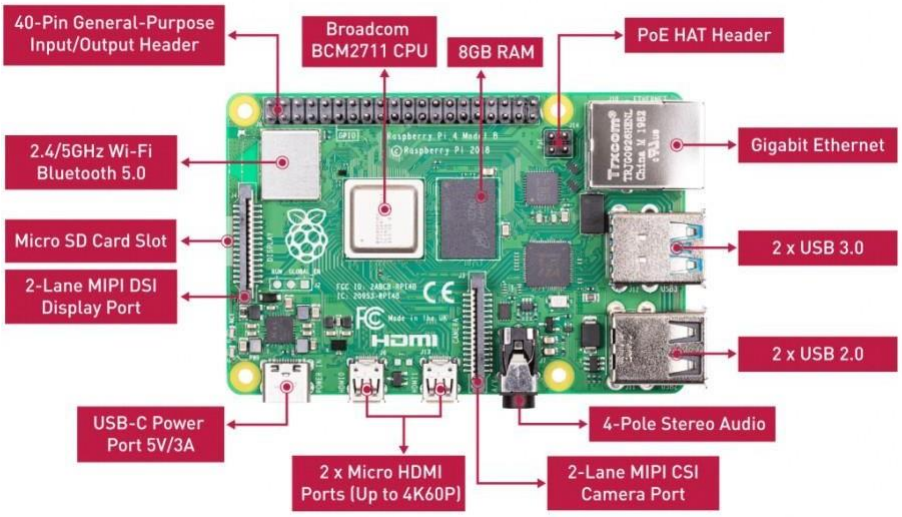


Figure 1.2: Photo du raspberry pi 4 avec ces specifcation [1]

d'entrée/sortie. Cela se présente sous la forme d'un écran tactile de 5 pouces alimenté par USB (voir Figure 1.3). Sa basse consommation et sa résolution permettent non seulement d'être alimenté directement par le Raspberry Pi mais aussi de réduire le coût et l'exigence graphique sur les performances.

Figure 1.3: Photo de l'écran en question [2]

Le Raspberry Pi doit être en marche tout le temps, ce qui
exclut l'utilisation d'une carte microSD comme stockage pour le système d'exploitation de ce dernier. En effet, ces cartes ont un nombre limité de cycles de lecture/écriture, ce qui finira par bloquer le stockage en mode lecture seule. Pour y remédier, le Raspberry Pi supporte le démarrage par USB et il possède quatre ports. Nous nous intéresserons au port USB 3.0 avec une vitesse de 5Gbps, ce qui donne à l'aide d'un adaptateur sata a usb la possibilité d'utiliser un HDD ou même un SSD en 2.5 pouces car ceux-ci demandent seulement du 5V et sont plus compacts (Figure 1.4.

[3]

[4]

[5]

Figure 1.4: Représentation des volumes de stockage

Vu la totalité des périphériques qui requièrent du 5V, cela résulte en basse tension et haut courant : 3.1A pour le Raspberry, 1A pour l'écran et 1.5A pour le disque. Pour cela, nous avons utilisé un module abaisseur de 12V à 5V 50W (Figure 1.5), rendant la tension requise plus standard et nécessitant moins de courant. dans le but d'assurer le fonctionnement stable du CPU bcm2711 du raspberry il est

Figure 1.5: Module abaisseur de 12V à 5V 50W [6]

Il est impératif d'utiliser un system de refroidissement actif qui va veiller a garder la température du CPU basse ainsi augmenter la cadence de ce dernier donnant un boot de performance.

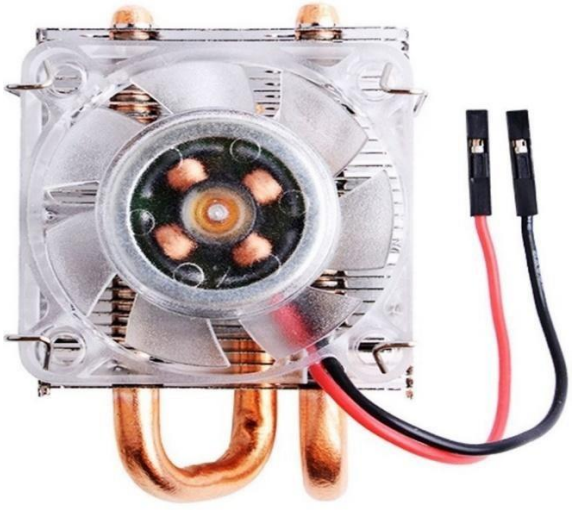


Figure 1.6: refroidissement actif dédié [7]

Afin de ranger tout cela dans un espace compact, nous avons modélisé en 3D sur Fusion 360 un boîtier pour contenir tous les composants précédemment cités, puis imprimé en 3D avec du filament PLA. Ce dernier n'est pas fait pour endurer les chaleurs émises par nos composants, mais la dissipation d'air par le refroidissement devrait y remédier. Le résultat se présente comme suit (Figure 1.7):

(a)

(b)

(c)

Figure 1.7: Photo du Raspberry Pi 4 avec boîtier customisé

1.3 Partie Logiciel

installation os Tout d'abord, nous devons choisir un OS pour le Raspberry Pi. Afin d'éviter les problèmes de compatibilité, nous utilisons celui fourni par la Raspberry Pi Fondation [8] Raspberry Pi OS, basé sur Debian. Tous les programmes compatibles avec Debian seront compatibles par défaut. Pour installer Raspberry Pi OS, il faut d'abord installer RaspberryPi Imager (Figure 1.8) depuis l'onglet Software du site officiel Raspberry Pi [1]. Après avoir

Figure 1.8: Photo de la page web en question

exécuté ce programme, on choisit le modèle de Raspberry Pi en cliquant sur "Choose OS" et en sélectionnant le modèle correspondant, dans notre cas le Raspberry Pi 4. Ensuite, on choisit la version de l'OS en 64 bits avec environnement graphique. Puis, on branche le volume de stockage (SSD) avant de lancer l'installation. Il est important de cliquer sur l'onglet paramètres et d'activer le SSH sur le Raspberry Pi afin d'accéder à distance au terminal (Figure 1.9).

A Installation de Home Assistant

Ensuite, il faut installer Home Assistant, un logiciel domotique open source qui permet de communiquer avec une majorité d'appareils connectés, y compris ceux créés par nous-mêmes sous ESPHome.
Il existe plusieurs versions de Home Assistant :

- **Home Assistant OS:** Un système d'exploitation dédié à Home Assistant.
- **Home Assistant Core:** Une application Home Assistant pouvant être installée sous Linux.

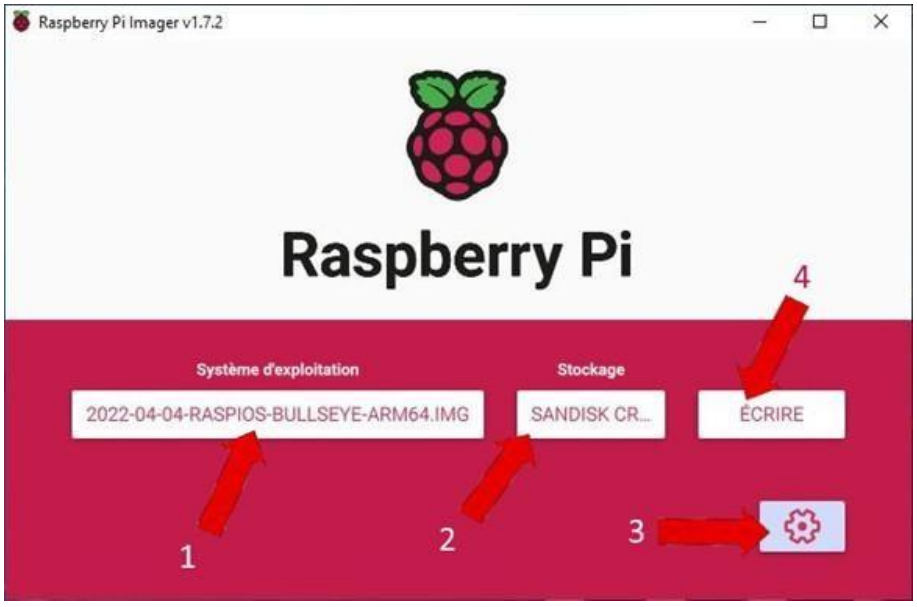


Figure 1.9: Photo de chaque étape d'installation du raspberry pi os sur un ssd

Figure 1.10: Photo de chaque étape d'installation du raspberry pi os sur un ssd (suite)

- **Home Assistant Docker:** Un conteneur permettant d'exécuter Home Assistant dans un environnement isolé et peu demandant en termes de puissance.

Bien que Home Assistant OS soit performant, il ne permet pas l'utilisation d'autres programmes en dehors des extensions de Home Assistant. La version Docker est moins exigeante en termes de puissance, mais elle manque d'options telles que les add-ons et les restaurations de sauvegarde, rendant cette version peu convenable pour notre utilisation. Nous nous tournons donc vers la version Core, qui offre à la fois les options et fonctionnalités de la version OS et l'indépendance de la version Docker, tout en permettant l'installation d'autres programmes.

En utilisant cette vidéo [9] comme base, ainsi que le supervised installer de Home Assistant [10] ,nous pouvons procéder à l'installation.

B Installation de Home Assistant Supervised:

Pour installer Home Assistant Supervised sur un Raspberry Pi 4, suivez ces étapes :

1. **Installation des dépendances:** Depuis le lien github car ces dernier change avec le temps.

2. **Installation de Docker:** Pour installer Docker, utilisez la commande suivante. Si vous recevez une erreur disant que Docker est introuvable, redémarrez simplement votre Raspberry Pi.

```
curl -fsSL get.docker.com | sh
```

3. **Installation de OS-Agent:**

   (a) Visitez le lien suivant pour obtenir la dernière version de OS-Agent : OS-Agent Releases[4].

   (b) Déterminez l'architecture de votre machine avec la commande suivante :

```
uname -a
```

   Pour un Raspberry Pi, vous verrez quelque chose comme ceci :

```
Linux server-pi 6.1.0-rpi7-rpi-v8 #1 SMP PREEMPT
Debian 1:6.1.63-1+rpt1 (2023-11-24) aarch64
GNU/Linux
```

   Cela indique que la version aarch64 est celle qui convient.

   (c) Téléchargez et installez OS-Agent:

```
wget https://github.com/home-assistant/os-
agent/releases/download/1.6.0/os-
agent_1.6.0_linux_aarch64.deb sudo dpkg -i os-
agent_1.6.0_linux_aarch64.deb
```

[4] https://github.com/home-assistant/os-agent/releases/tag/1.6.0

(d) Testez l'installation :

```
gdbus introspect --system --dest io.hass.os
--object-path /io/hass/os
```

4. Installation de Home Assistant Supervised :

(a) Téléchargez le package d'installation de Home Assistant Supervised :

```
wget -O homeassistant-supervised.deb
https://github.com/home-assistant/supervised-
installer/releases/latest/download/homeassistant-
supervised.deb
```

(b) Installez Home Assistant Supervised :

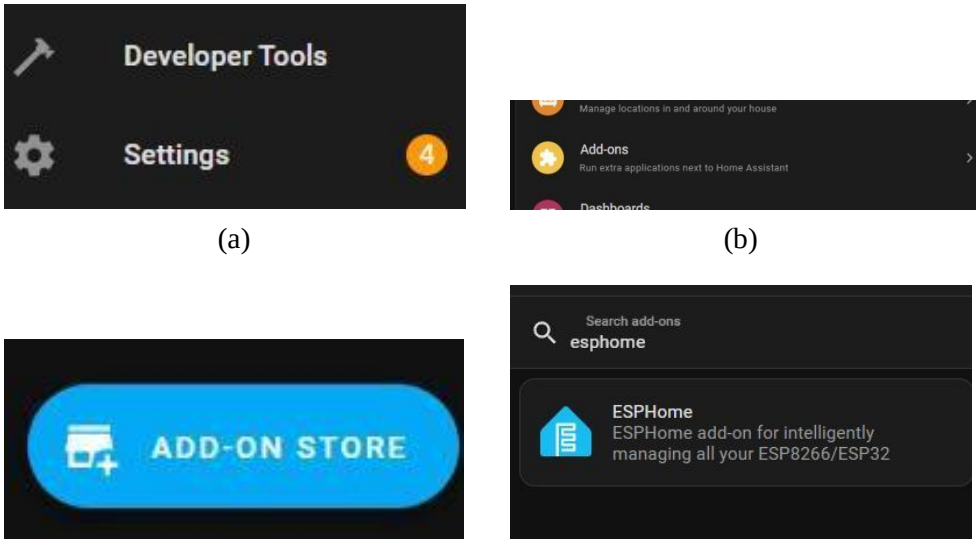
```
sudo apt install./homeassistant-supervised.deb
```

Durant l'installation, il nous sera demandé de choisir le type de machine ; dans ce cas, il s'agit du Raspberry Pi 4. L'installation se termine lorsque le terminal nous indique le port sur lequel la page web est accessible. Par défaut, c'est le 8123. Après redémarrage, nous serons accueillis par la page de configuration et le choix de realiser une nouvelle configurationou de charger une sauvegarde.

C Langage de programmation et IDE

Afin de choisir notre IDE et langage, ces derniers doivent nous permettre de relier nos objets connectés entre eux ainsi qu'avec Home Assistant sur le Raspberry Pi, le tout sans compter sur des services hors réseau local. Pour cela, nous trouvons dans l'onglet add-on de Home Assistant l'intégration ESPHome (Figure 1.11).

Cette dernière permet à la fois de flasher les microcontrôleurs des objets connectés, les mettre à jour via OTA (firmware et code), suivre leur état de connexion ainsi que les lier à Home Assistant sans avoir à mettre en place un serveur MQTT, mais en utilisant l'API de Home Assistant déjà présente (figure 1.12).



(a) (b)

(c) (d)

Figure 1.11: Etapes d'installation ESPhome

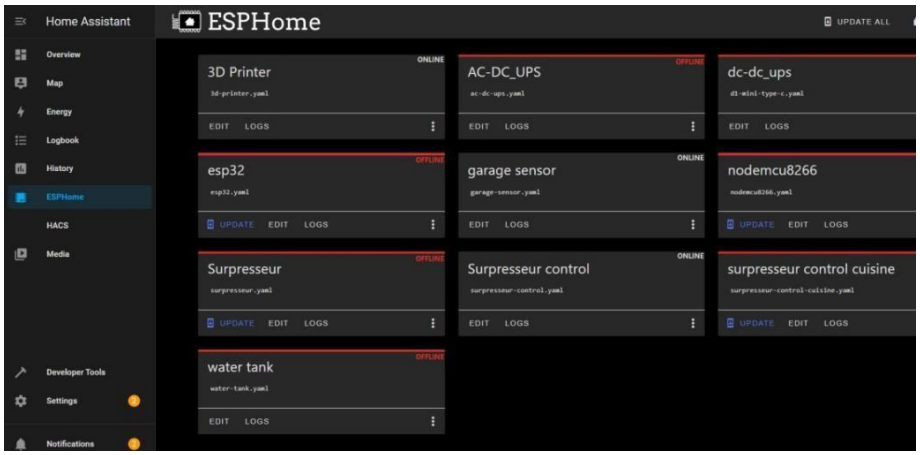


Figure 1.12: Liste des appareils actif

ESP Home utilise le langage de programmation YAML, étant à la fois simple à apprendre et comprendre, mais aussi ne requérant aucune librairie à être installée, voire même pas d'IDE. On peut programmer, modifier et compiler le code du microcontrôleur directement à partir de l'interface web de ESP Home, sans oublier la riche documentation disponible sur le site https://esphome.io [5], comprenant des exemples clairs et simples à suivre pour les microcontrôleurs ainsi que la variété de capteurs supportés [11].

[5] https://esphome.io

Figure 1.13: ESPHome.io

Chapitre 2

Communication entre les composants

2.1 Introduciton

La communication entre les composants est une partie essentielle pour dans un écosystème, cette dernière dois être à la fois rapide et stable pour assurer la fiabilité du system.
ainsi des paramètres tel que la distance vitesse et la tance sont a prendre en considération

2.2 Choix du protocole

Pour assurer la communication entre nos objets connectés, il nous faut un protocole de communication fiable, rapide et simple à mettre en place tel qu'un protocole sans fil. En cherchant selon ces critères, on trouve le Bluetooth, le WiFi, Zigbee et Z-Wave (Figure **??**). Pour ces deux derniers, ils requièrent des émetteurs/récepteurs propriétaires additionnels qui vont ajouter au coût mais aussi à la complexité du développement. D'autre part, le Bluetooth est limité par sa courte portée ainsi que son absence dans l'ESP8266 et le RP2040. Par ailleurs, la norme WiFi se présente comme le meilleur candidat grâce à sa longue portée, sa disponibilité dans tous les microcontrôleurs, comme dans ESPHome, et son faible coût, facilitant sa mise en place.

[12]

[13]

[14]

[15]

Figure 2.1: Logos de Zigbee, Z-Wave, Bluetooth, et Wi-Fi

2.1 Choix des composants

Dans le but de couvrir complètement notre environnement domotique, nous avons décidé de mettre en place des points d'accès WiFi tous reliés au même routeur, ainsi qu'un Raspberry Pi. Les microcontrôleurs utilisent la norme WiFi 4 à 2,4 GHz. Les routeurs WiFi 4 sont peu coûteux en raison de leur ancienneté. De plus, pour économiser davantage en termes de prix, nous utilisons d'anciens modems. Malgré leurs ancienté ils suppotent le wifi4

2.2 Mise en place :

Pour une couverture excellente, nous allons mettre en place un point d'accès à chaque étage de notre espace de travail, chacun relié directement au routeur par un câble Ethernet pour assurer la stabilité et minimiser les points de défaillance dans le réseau. Ces derniers sont positionnés de manière à ne pas être trop proches ni trop éloignés afin que, si l'un d'eux tombe en panne, les points d'accès adjacents puissent prendre le relais sans trop de perte (Figure 2.2).

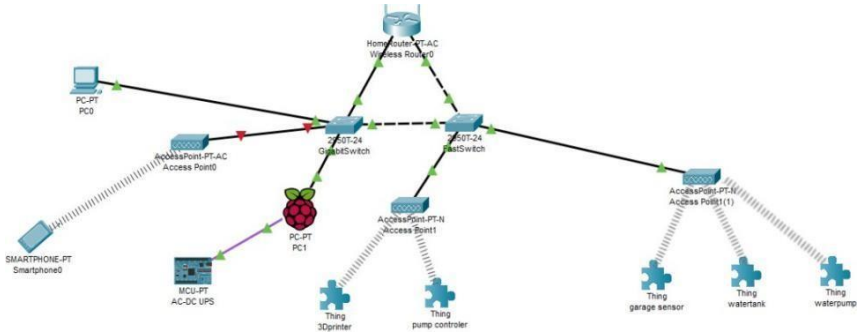


Figure 2.2: Schéma du réseau sous cisco packet tracer

Nous avons aussi décidé de leur donner le même SSID et mot de passe pour permettre un transfert souple et sans faille des objets connectés en cas de problème. Il est aussi important de s'assurer que chacun des points d'accès utilise une sous-bande de fréquence différente des autres pour éviter les interférences entre eux (Figure 2.3).

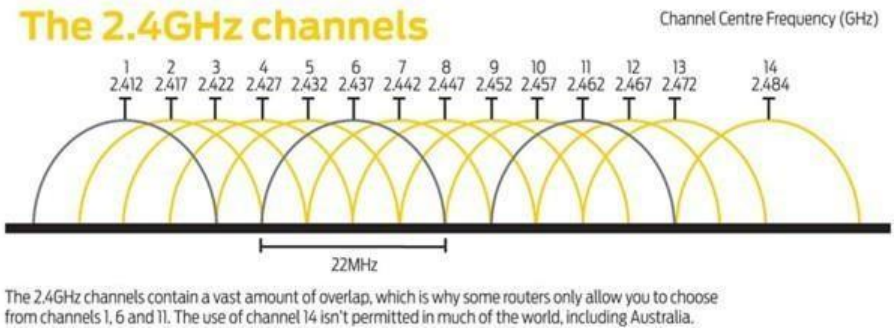


Figure 2.3: Bande de fréquence [16]

2.3 Optimisation

Pour éviter les problèmes d'attribution d'adresse, il est préférable de donner une adresse IP statique à chaque appareil connecté dans le modem principal à l'onglet DHCP. Les points d'accès doivent acquérir les adresses à partir de ce dernier (Figure 2.4).

Figure 2.4: Page d'adresse ip statique pour les composants

Sécurité

Afin d'assurer la sécurité du réseau et l'intégrité des composants dans notre système, nous avons désactivé l'option de diffusion du SSID. Ainsi, le réseau Wi-Fi ne s'affichera pas lors d'une recherche et sera moins vulnérable en cas d'attaque.

Conclusion

Après avoir réalisé les étapes redécampent cité, notre réseau est d'un part suffisamment stable et sécurisé afin de relier nos composants entre eux, mais flexible pour de futures modifications.

Chapitre 3

Réalisation d'appareils

3.1 Initialisation du microcontrôleur

Introduction

dans ce chapitre nous allons realiser nos propre objets connecté les relier avec le réseau communiqué avec le serveur et entre eux ainsi de lire les données depuis ces derniers.

Pour realiser nos objets connectés, nous devons d'abord nous rendre à l'adresse du RaspberryPi avec le port 8123 pour accéder à Home Assistant (ex. 192.168.1.104:8123). Ensuite, entrez respectivement dans Paramètres, Add-ons, ESPHome, puis sur Open Web UI. Une fois sur ESPHome, cliquez sur "secrets" en haut à droite et écrivez la commande suivante :

```
# Your Wi-Fi SSID and password

wifi_ssid: "nom_du_reseau_wifi"

wifi_password: "motdepasse"
```

Une fois sauvegardé, cela permet de définir le nom du réseau et le mot de passe sans qu'ils soient écrits dans le code. Ainsi, ils seront assignés directement à la création de nouveaux objets. Retournez à la page principale d'ESPHome et cliquez en bas à droite sur "New Device" (Figure 3.1). continuez et nommez votre objet. Une fois sur "Next", choisissez le type de microcontrôleur que vous comptez utiliser (Figure 3.2). Une configuration primaire sera créée ainsi qu'une clé de communication qui permettra à Home Assistant de communiquer avec l'appareil (Figure 3.3).

Ensuite, appuyez sur "Install" et il vous sera demandé de quelle manière installer le programme sur votre microcontrôleur. La manière la plus fiable serait de le brancher sur le serveur tournant ESPHome. Branchez le microcontrôleur dessus via USB et une option telle que /dev/ttyUSB0 apparaîtra (Figure 3.4).

Il est possible d'avoir plus d'options si plusieurs adaptateurs série sont branchés sur le serveur. Dans ce cas, trouvez celui qui vous concerne et poursuivez l'installation. Une fois

Figure 3.1: La page principale d'ESPHome

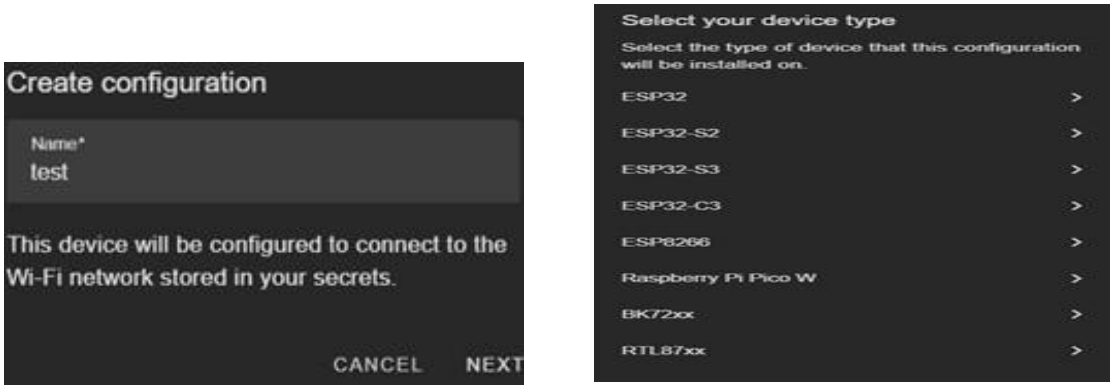


Figure 3.2: Etapes de configuration

Figure 3.3: Configuration primaire

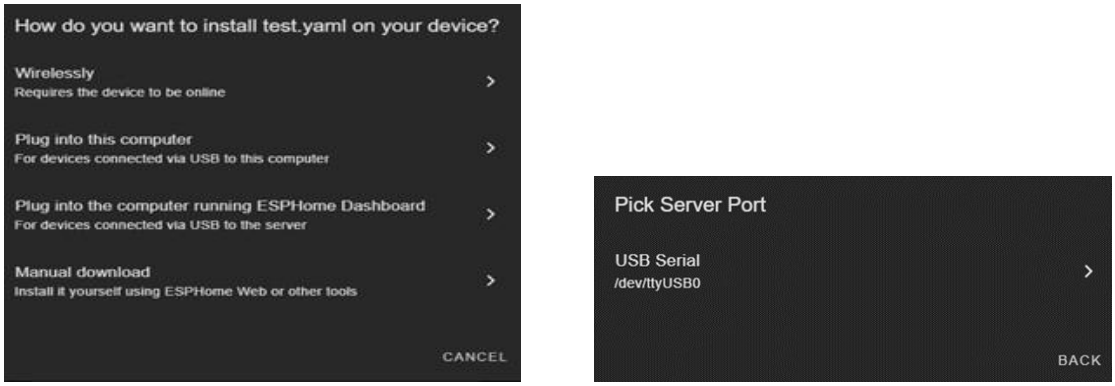


Figure 3.4: Etapes d'installation du microcontoleur

terminée, débranchez le microcontrôleur. Les prochaines mises à jour seront sans fil, mais si le microcontrôleur ne peut se connecter au réseau, il se met en mode fallback où il diffuse son propre réseau pour accéder manuellement via une page web et assigner un nouveau réseau si souhaité (Figure 3.5).

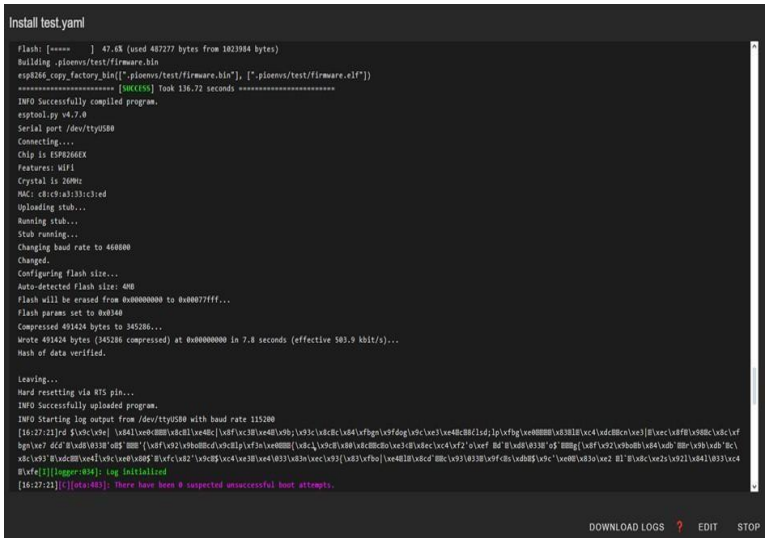


Figure 3.5: install test YAM

3.2 Prise connecté :

Introduction

Comme premier objet, nous avons créé une prise qui consiste à ouvrir ou fermer un relais à notre commande, permettant ainsi d'allumer ou d'éteindre l'imprimante 3D. Nous avons réalisé le schéma suivant (Figure 3.6) :

Ainsi que le code qui convient. Le choix du pin D8 (GPIO15) pour le relais est obligatoire car ce dernier est en pulldown. Autrement, si on utilise un pin en pullup, sela va activer le relais brièvement durant le démarrage du microcontrôleur. De l'autre côté, le

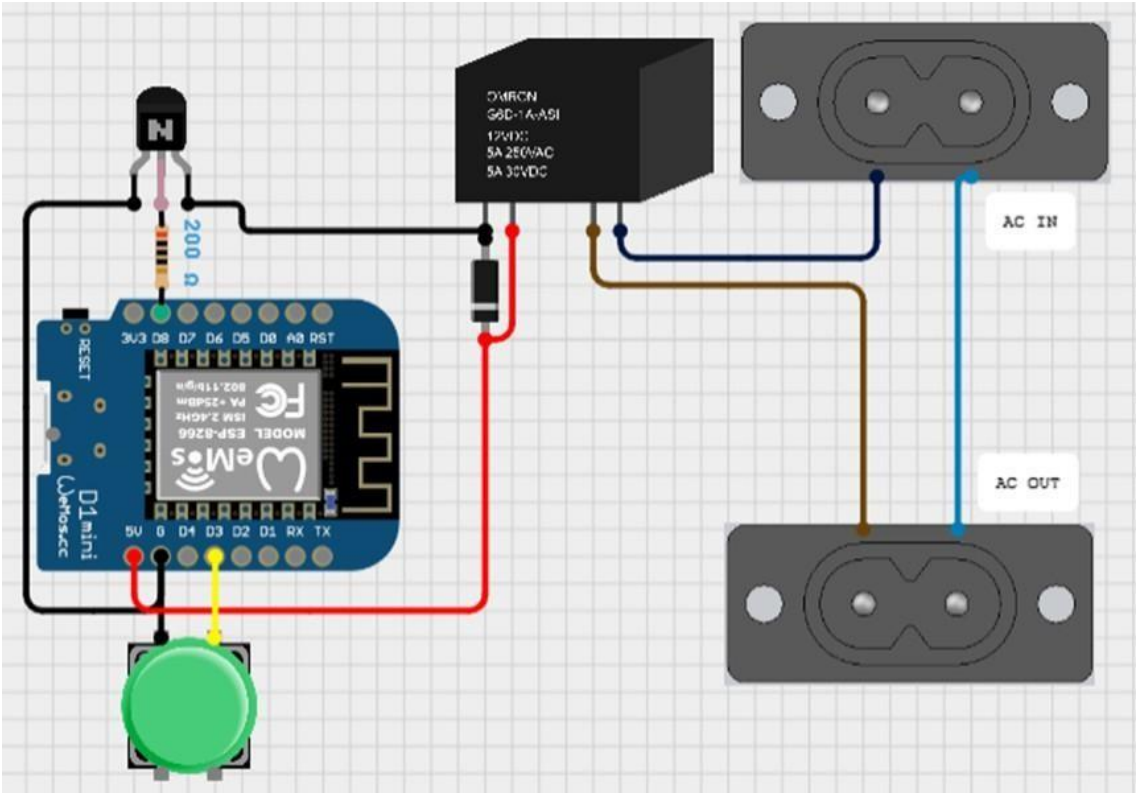


Figure 3.6: schéma de la prise connecté

pin D3 (GPIO0) est en pullup, donc le bouton fonctionnera correctement. Aussi, le pin D4 (GPIO2) contrôle la LED intégrée dans l'ESP8266 afin de donner l'état du relais.

```
esphome:
  name: 3d-printer
  friendly_name: 3D Printer

esp8266:
  board: esp01_1m

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "fo8twiOqsmYxnacmhUEjVmyX+v/TV3hxpTYSxSpcR0M="

ota:
  password: "38a11a11da7f87f0b8ae76b7671fd252"
```

```
wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

  # Enable fallback hotspot (captive portal) in case wifi
     connection fails
  ap:
    ssid: "3D-Printer Fallback Hotspot"
    password: "ZC4vPQW5XBw9"

captive_portal:

# activation button
binary_sensor:
  - platform: gpio
    pin:
      number: 0
      inverted: yes
      mode: INPUT_PULLUP
    name: "button"
    on_press:
      then:
        - switch.toggle: relay

# relay
switch:
  - platform: gpio
    name: "Relay"
    pin:
      number: 15
      inverted: false
    id: relay
    on_turn_on:
      then:
        - switch.turn_on: led
        - logger.log: output.turn_on
```

```
  on_turn_off:
    then:
      - switch.turn_off: led
      - logger.log: output.turn_off

- platform: gpio
  name: "LED"
  pin:
    number: 2
    inverted: yes
  id: led
  restore_mode: ALWAYS_OFF
```

Après avoir branché la prise à notre imprimante, nous devons nous rendre à Home Assistant afin de l'intégrer dans l'écosystème. Tout d'abord, récupérez la clé du microcontrôleur qui contrôle la prise depuis le code dans la page d'ESPHome (Figure 3.7).

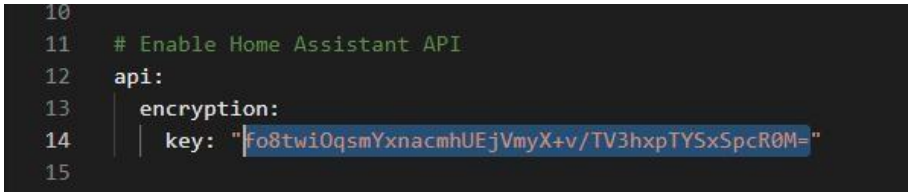


Figure 3.7: Photo du code avec la clef selectionné

Ensuite, allez à Home Assistant dans Settings, Devices & Services. Si l'intégration ESPHome est déjà ajoutée, l'appareil sera dans le menu Discovered en haut. Sinon, il faut aller à Add Integration et ajouter ESPHome. Une fois fait, allez dans l'intégration ESPHome, cliquez sur "Devices" et vous verrez la liste des objets. Cliquez sur celui qui vous concerne. D'ici, vous avez accès à l'historique, les données enregistrées tout en bas appelées logbook si vous voulez monitorer vos capteurs, et vous pourrez les ajouter à la page d'accueil. Vous pourrez aussi voir et contrôler toutes les entrées et sorties de votre objet connecté, mais il est bien plus facile de le faire depuis la page d'accueille d'accueil de Home Assistant (Figure 3.8).

(a) (b) (c)

Figure 3.8: Page d'accueil de Home Assistant

(a) (b) (c)

Figure 3.9: Matériel

3.1 Control directe

Nous avons créé la même prise connectée comme avec l'imprimante 3D mais dans un cas de figure différent consistant à contrôler un surpresseur d'eau cette fois si sans avoir à utiliser une application ni d'interface web mais directement à l'aide d'un bouton physique placé à distance.

Pour cela nous avons créé une sorte d'interface de control avec un simple bouton est une led d'état avec le boitier qui convient sous fusion 360 (Figure 3.10).

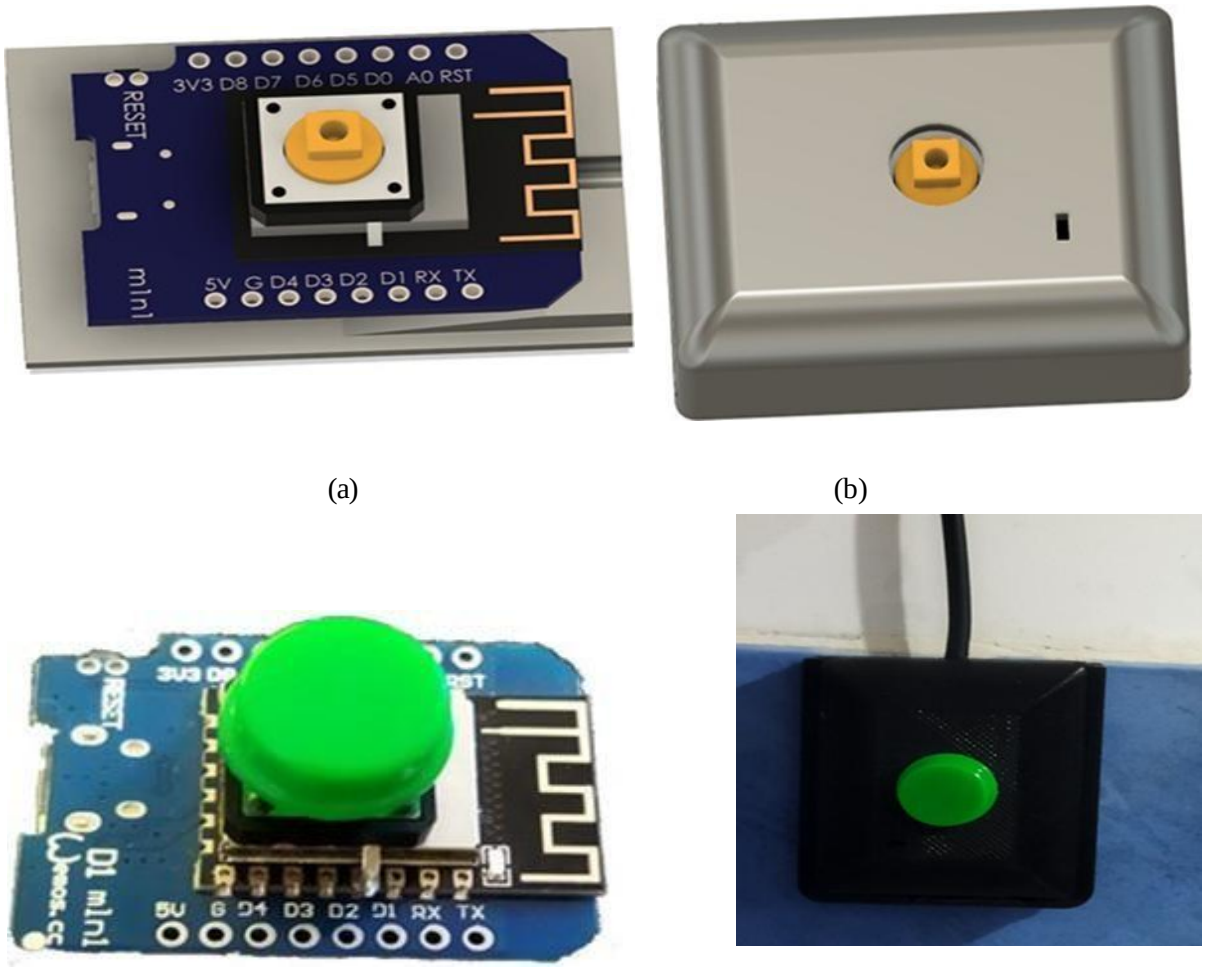


Figure 3.10: Interface de control

Ensuite nous avons ajouté le code qui convient sous esp home ainsi qu'intégrer le microcontrôleur a home assistant a home assistant.

```
esphome:
  name: surpresseur-control
  friendly_name: Surpresseur control

esp8266:
  board: esp01_1m

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "Rx+8PPIuaUddH80InjiSA25etL0GPY39CefQ7vjHrtI="

ota:
  password: "a8a3f4fe80909d4ede23ddce76f4a6ac"

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

  # Enable fallback hotspot (captive portal) in case wifi
      connection fails
  ap:
    ssid: "Surpresseur-Control"
    password: "ke7q1jIhLhk1"

captive_portal:

# activation button
```

```
binary_sensor:
  - platform: gpio
    pin:
      number: 4
      inverted: yes
      mode: INPUT_PULLUP
    name: "boutton"
# led state

output:
 - platform: gpio
   pin: GPIO2
   inverted: yes
   id: led_output

light:
 - platform: binary
   name: "LED Light"
   output: led_output
```

Afin de communiqué les microcontrolleurs entre eux le plus efficace est d'utiliser espnow un protocol de comunication realiser par espressif similaire au protocol wifi permetant do comuniquer les microcontroleurs entre eux directemant mais vue la distance entre le relay (RDC) et les interfaces (2eme, 3eme) nous avont utiliser la fonctionnalité des scènes de home assistant qui serte nessesite de passé par le serveur raspberry mais permet d'utiliser les point d'acces wifi qui couvre tout l'espace de travail.

Tout d'abord il faut s'y rendre au menu settings , automatisation & scenes (Figure 3.11).

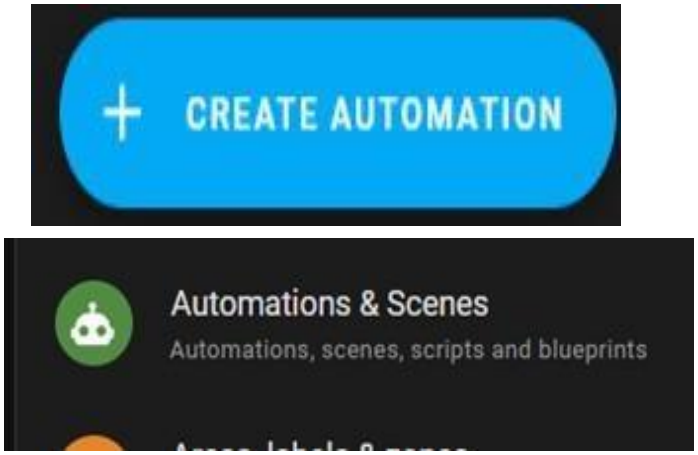


Figure 3.11: Automatisation & scenes

Puis creer une nouvelle automatisation (Figure 3.12).

Dans notre cas de figure on realiser trois automatisations (Figure **??**):

- Changer l'etat du relay si le boutton d'une des interfaces est appuier.
- Allumer la led des interfaces si le relay est activé.
- Etiende la led des interfaces si le relay est desactivé.

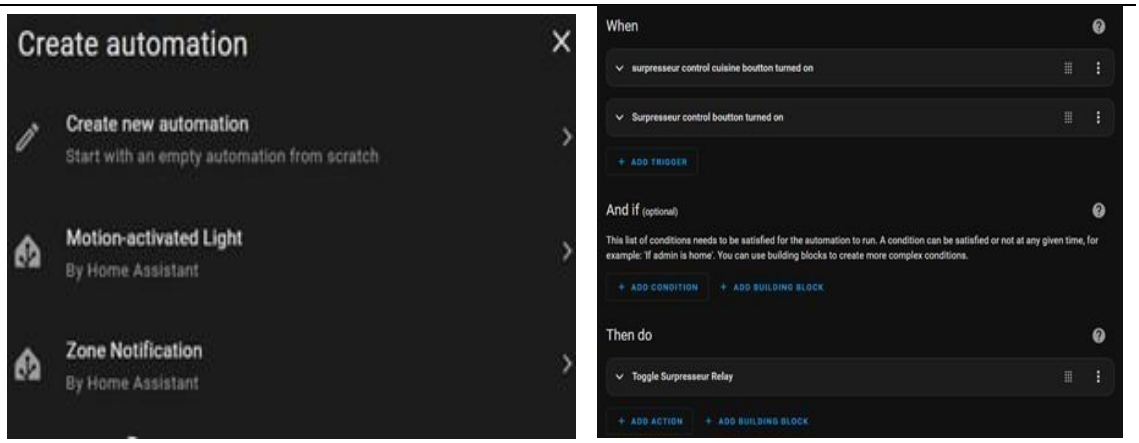


Figure 3.12: Automatisation & scenes

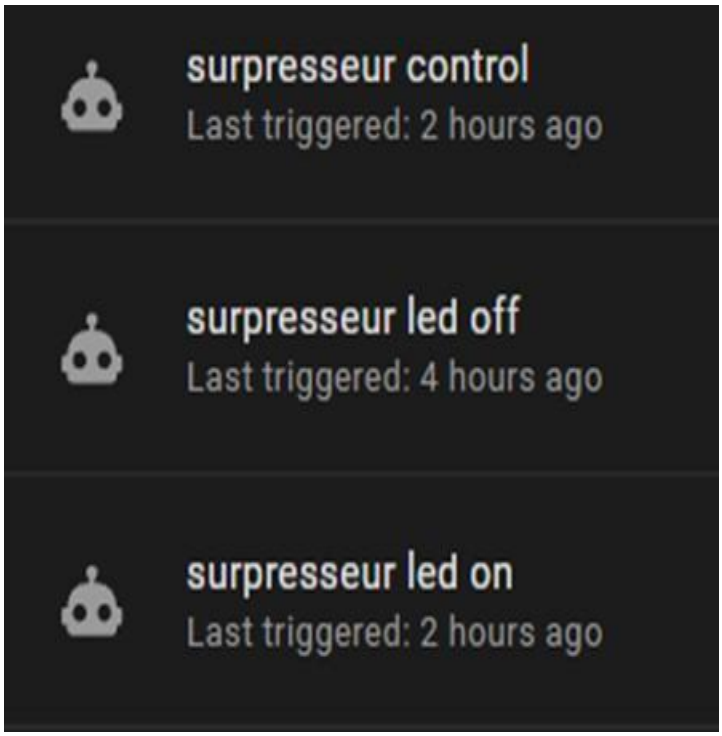


Figure 3.13: Automatisation

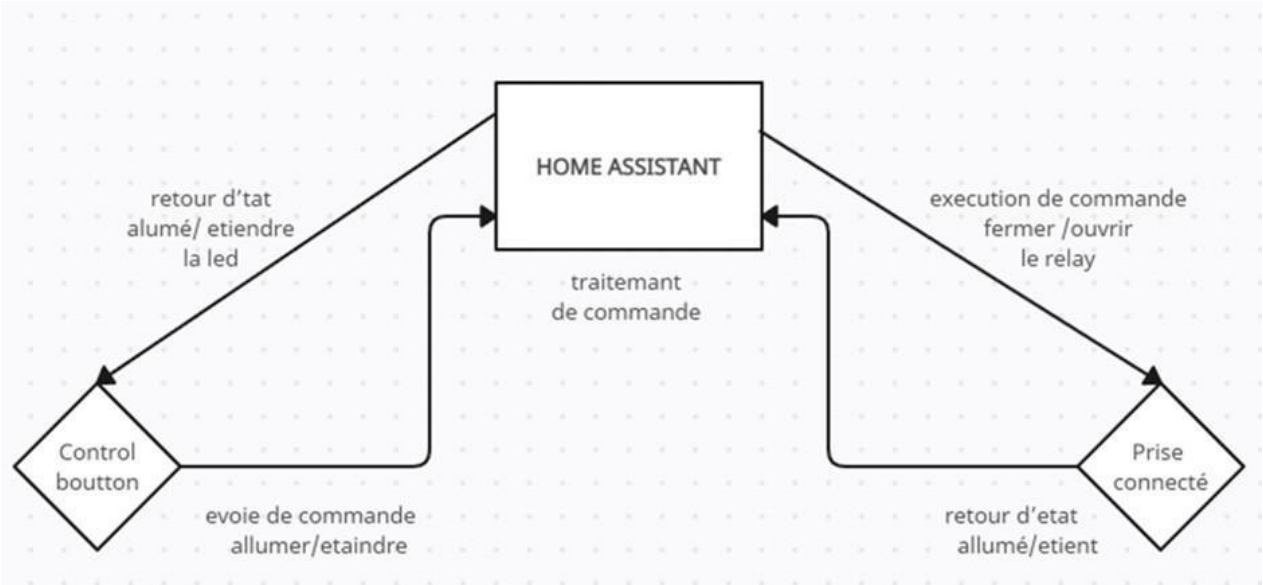


Figure 3.14: organigramme de comunication

Difficulté

Selon l'emplacement du bouton de contrôle humide ou sec il faudrait utiliser une peinture fine isolant le surcuit de l'humidité et corrosion un processus appelé tropicalisation.

Conclusion

Au final nous avons realiser une interaction sans files entre plusieurs microcontrôleurs ainsi que defaciliter le control de nos objets connectés et ouvrir plus de possibilité.

3.2 Onduleur AC-DC

le but de cet onduleur est de gardé le system en marche même en cas de coupure de façon efficace par rapport au onduleur traditionnelles.

le principe de ce dernier est d'éviter les pertes fer des onduleur classique en replaçant la conversion dc/ac du transformateur en une conversion dc/dc avec un buck/boost (régulateur) pour une sortie 12Vdc au lieu du 220Vac

Ainsi cela allongera l'autonomie de batterie et son temps de vie.

la tension standard 12Vdc permettra d'alimenter une large panoplie d'appareils tel que les routeurs et modem mais aussi les caméras de surveillance.

A Composants

- Régulateur de tension 12V 10A.
- Alimentation 12V 10A compacte.
- (02) batteries 12v 9Ah.
- Module anti décharge.
- Relay 20A 5Vdc .
- Module buck 5V a entré variable.
- Connecteur xt60 male et femelle.
- adc ADS 1115 16bit .
- (02) capteur de courant acs712 20A .
- écran oled 0.96' i2c .
- (03) led (rouge vert orange) .
- (03) pcb .
- (06) bornier 16A.
- Transistor bjt npn.
- ESP 8266.
- Cable.

- Connecteur jack 12V femelle.
- Couvert fusible.
- Fusible 0.5A.
- Connecteur jst 8pin.

B Conception

Nous avons réalisé le schéma ci-dessous puis séparer en parties afin de simplifier le concept de l'onduleur (Figure 3.15).

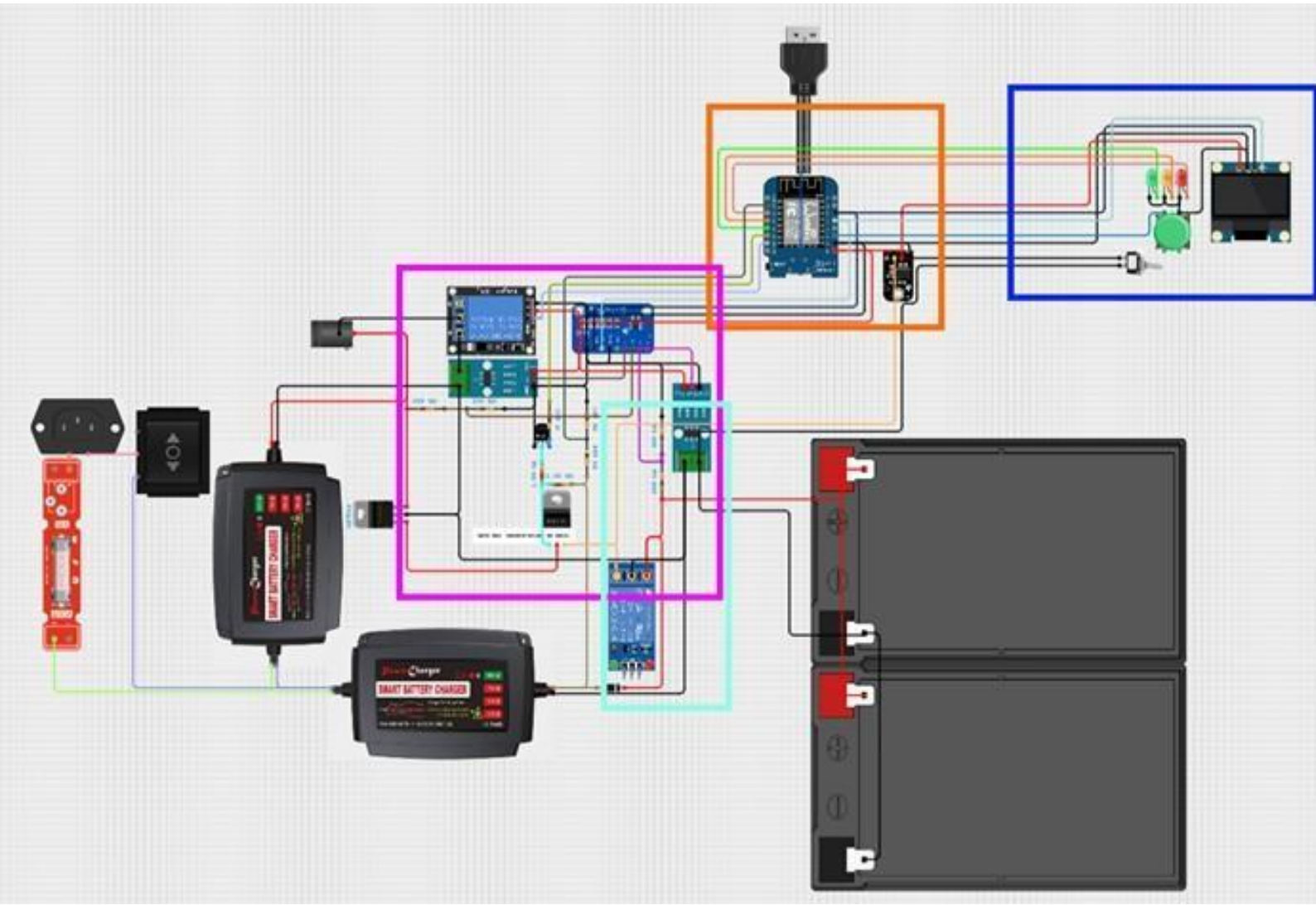


Figure 3.15: concept de l'onduleurs sous cirkit

Partie bms

Tout d'abord nous avons relier nos batteries (en parallèle pour plus d'autonomie) avec le chargeur comme présent (en respectant la tension de charge de ces dernière) avec un capteur de courant a effet hall (capable de capter le courant positif et négatif pour la décharge et charge de la batterie respectivement) entre le chargeur et la batterie

Puis a un capteur de courant, enfin un module anti décharge afin de protéger la batterie en faible tension (Figure 3.16).

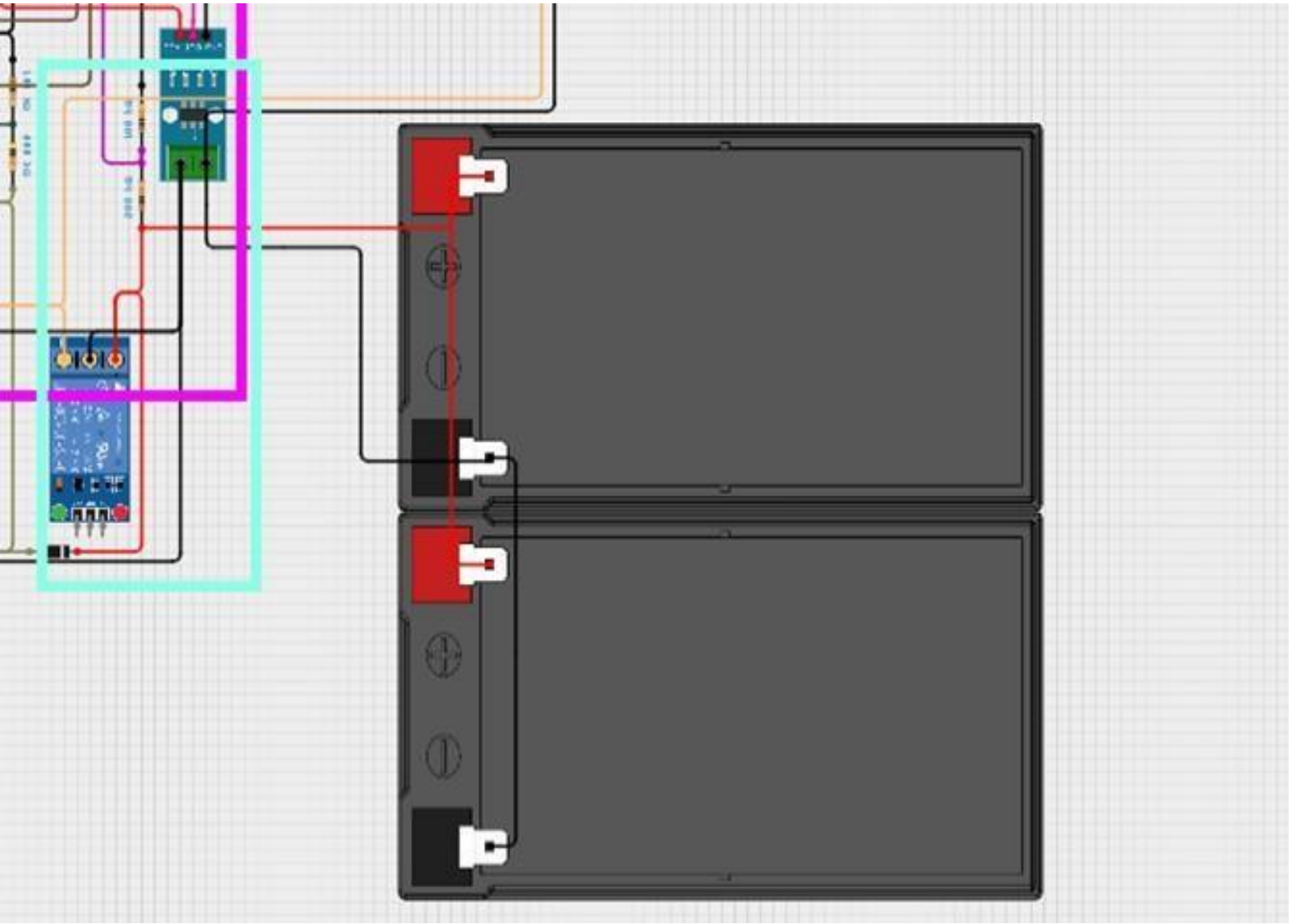


Figure 3.16: Partie bms

Partie logique

Cette partie rassemble les composants entré sortie ainsi que le traitement des données d'abord on a un régulateur abaisseur buck qui permet de convertir la tension de la batterie a 5V afin d'alimenter les capteurs et actionneurs ,ce dernier est contrôlé par un bouton interrupteur qui en gros permet d'allumer et éteindre microcontrôleur (esp8266) qui par la suite alimente notre l'onduleur et ses composants, l'écran est relier a l'esp via i2c et permet d'afficher les information tension/courant de la batterie mais aussi courant/tension de sortie principale les led R,V,O permettent d'indiqué respectivement si y a une coupure ou pas et si la batterie se charge, le bouton poussoir différemment du bouton du premier est utilisé pour le débogage mais peut être utiliser pour afficher la puissance et pourcentage utilisé, enfin comme notre microcontrôleur possède une seul entré analogique a 10 bit nous avons utilisé ADS1115 un adc de 16 bit a 4 entré, ce dernier permet non seulement d'avoir plus de port a plus haute résolution mais aussi moins susceptible au bruit parasite car il interface avec l'esp via i2c (Figure 3.17).

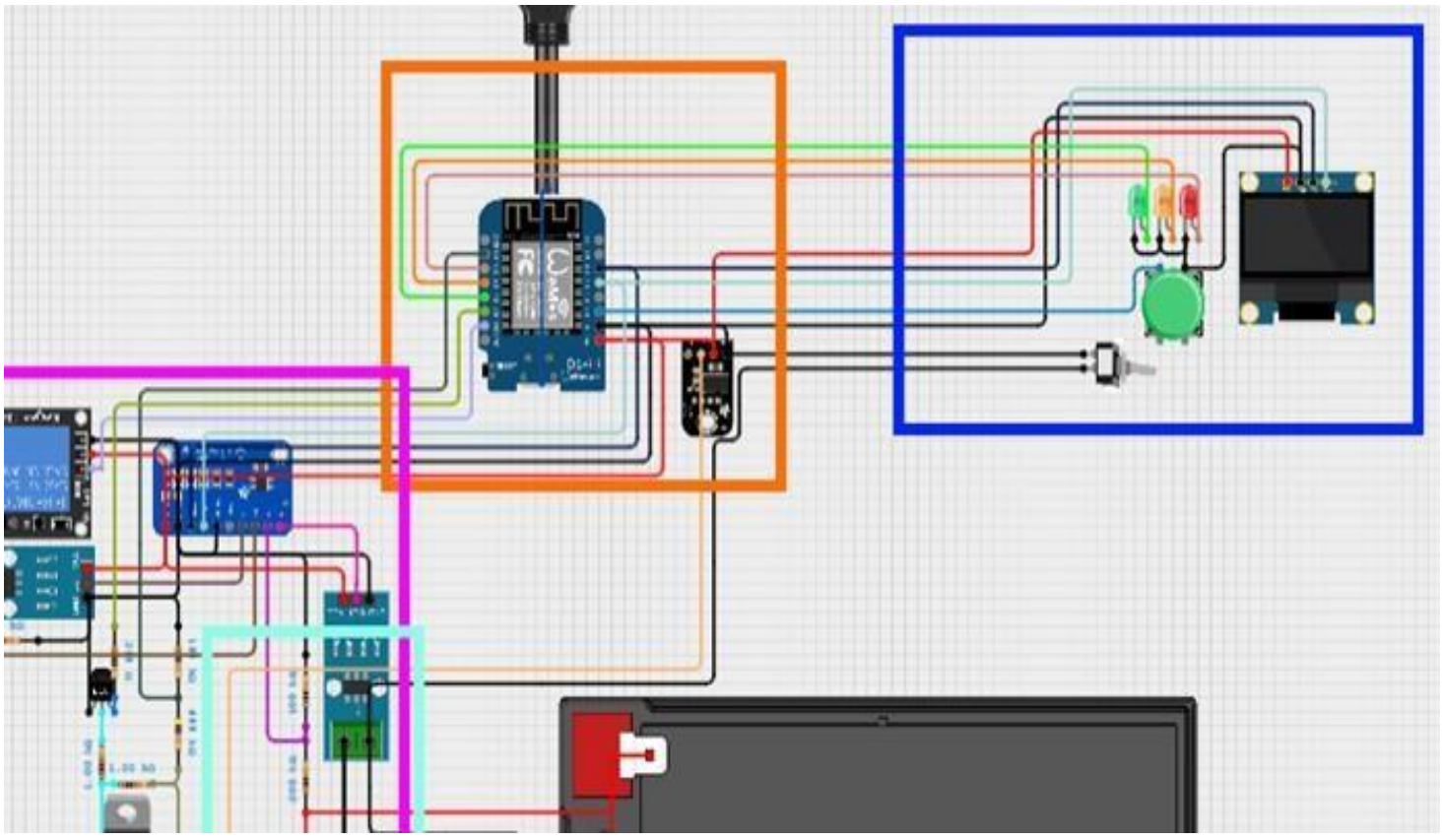


Figure 3.17: Partie logique

Partie puissance

Elle consiste à assembler les composants d'alimentation, régulation de tension et de protection

Pour commencer on a un transistor MOSFET qui gère le changement entre batterie et alimentation con cas de coupure

Ensuite le buck/boost permet de sortir une tension 12v stable peu importe la tension de la batterie et cela en parallèle avec l'alimentation de charge afin d'alimenter directement en cas de coupure, enfin nous avons un capteur de courant et un autre de tension ainsi qu'un Relay afin de couper le courant en cas de sous/sur tension pour protéger le system (Figure 3.18).

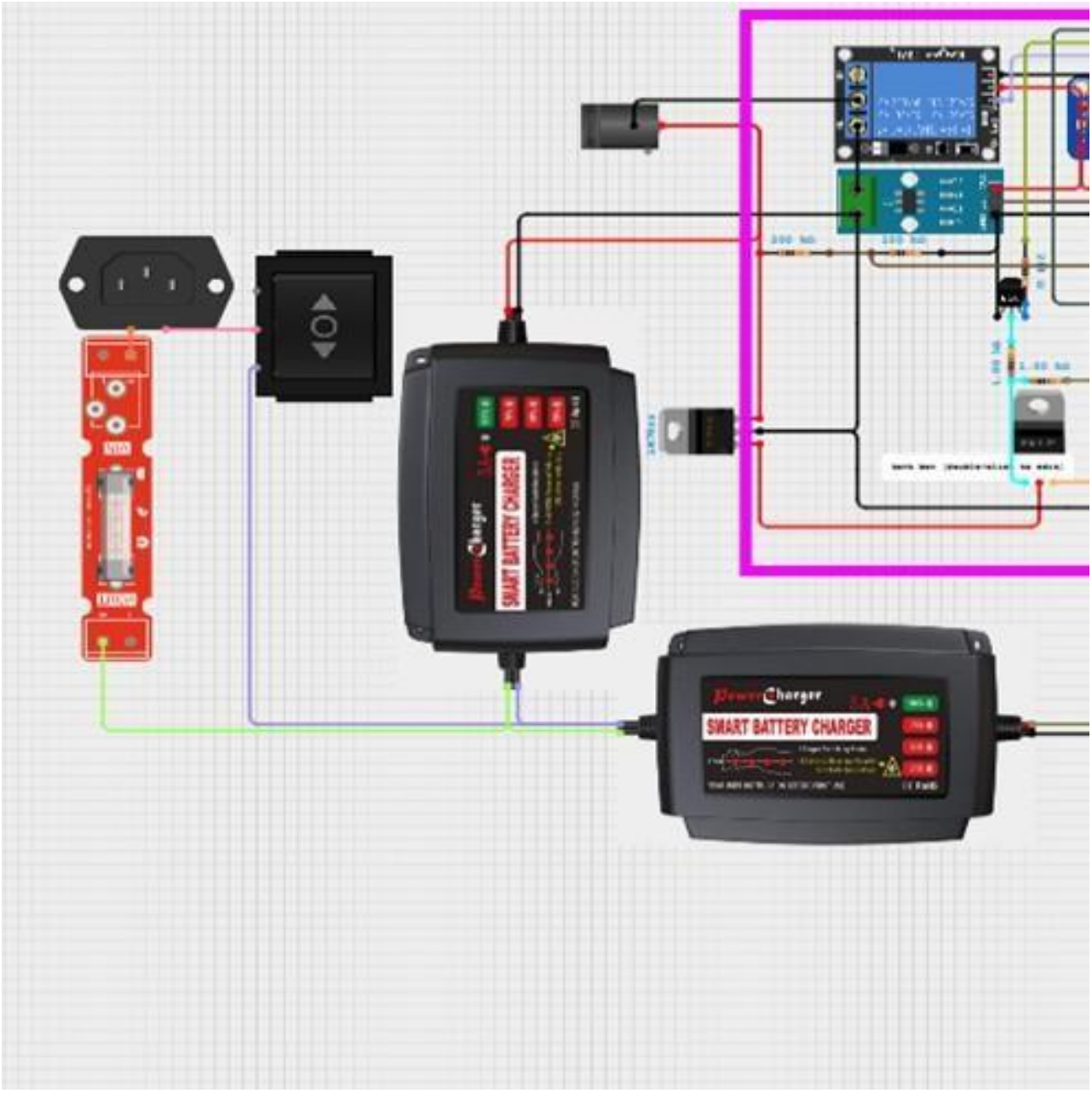


Figure 3.18: Partie puissance

l'utilisation de deux alimentations evite la complexité de realiser une alimentaion pour gerer a la fois la charge des batteries et alimenter les appareiles. la tension de l'alimentation de charge est indiqué sur la batterie entre 13.6V 13.8V en normal et de14.5V a 14.9V en charge

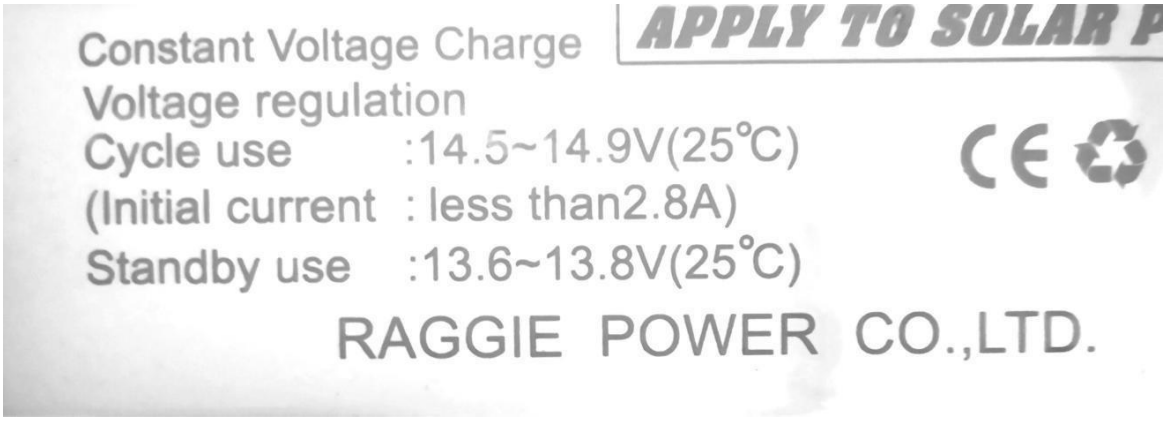


Figure 3.19: tension de charge donné par le fabriquant

il est peu commun de trouver des alimentations d'une telle tension voire même avec un courant élevé ,alors l'utilisation d'un module boost permet de remédier.

B Réalisation

Partie materiel

Pour commencer nous devant trouver un boitier pouvant contenir le volume des composants en question, un lecteur de cassette VHS par exemple possédant les dimensions idéales surtout pour l'épaisseur des batteries.

Figure 3.20: un lecteur de cassette VHS [17]

Après avoir vider le boitier de ces anciens composants il est important d'avoir une bonne destitution de poids dans ce dernier pour éviter de les endommager.

Figure 3.21: Photo des composant placé dans les boitiers.

Nous remarquons qu'il reste peu d'espace pour mettre en place un PCB contenant tout le schéma expliqué dans le partie conception afin d'y remédier nous avons opté pour une méthode appelé "stacked board design" ou "multi-board assembly" qui consiste a séparé notre carte en plusieurs avec chaque une d'entre elle une fonction précise

Cela permettra non seulement de gagner de l'espace mais aussi faciliter les modifications et isoler la résolution des problèmes sur chaque partie (Figure 3.22).

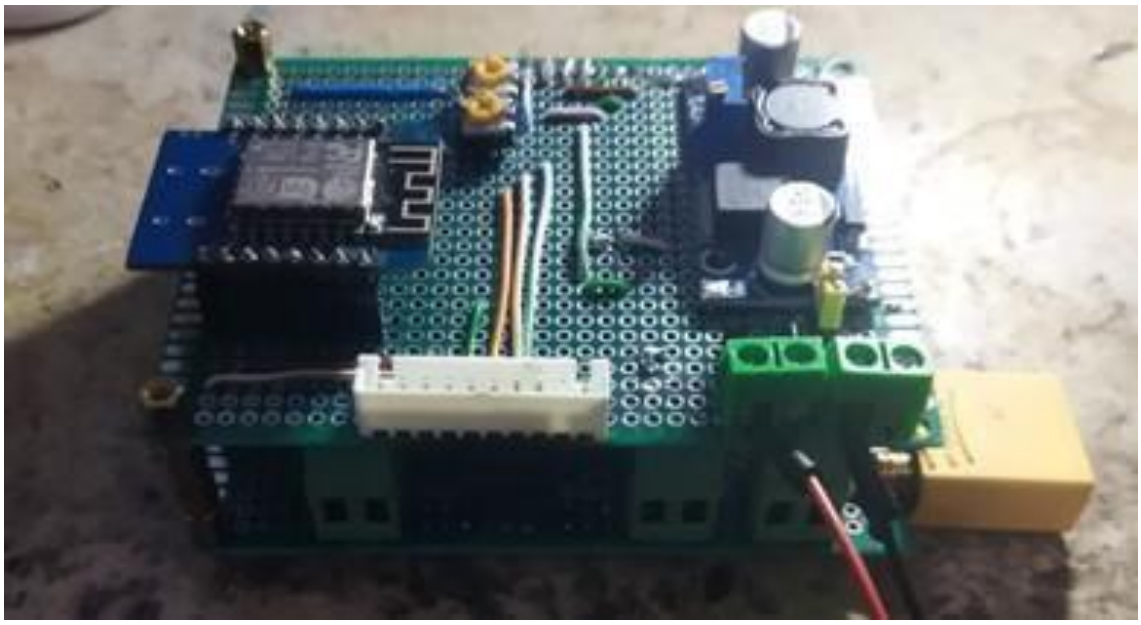


Figure 3.22: Réalisation hardware

- **Carte centrale** Responsable de connecter le reste des cartes au microcontrôleur ainsi que de les alimenter (Figure **??**).

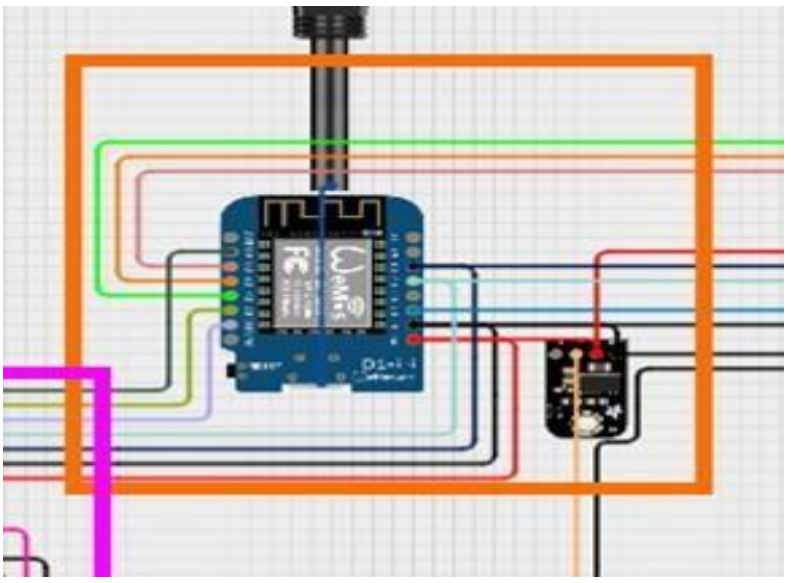


Figure 3.23: carte centrale avec schéma

.

- **Carte de puissance** Gère les capteurs et actionneurs ainsi que le courant de l'alimentation.

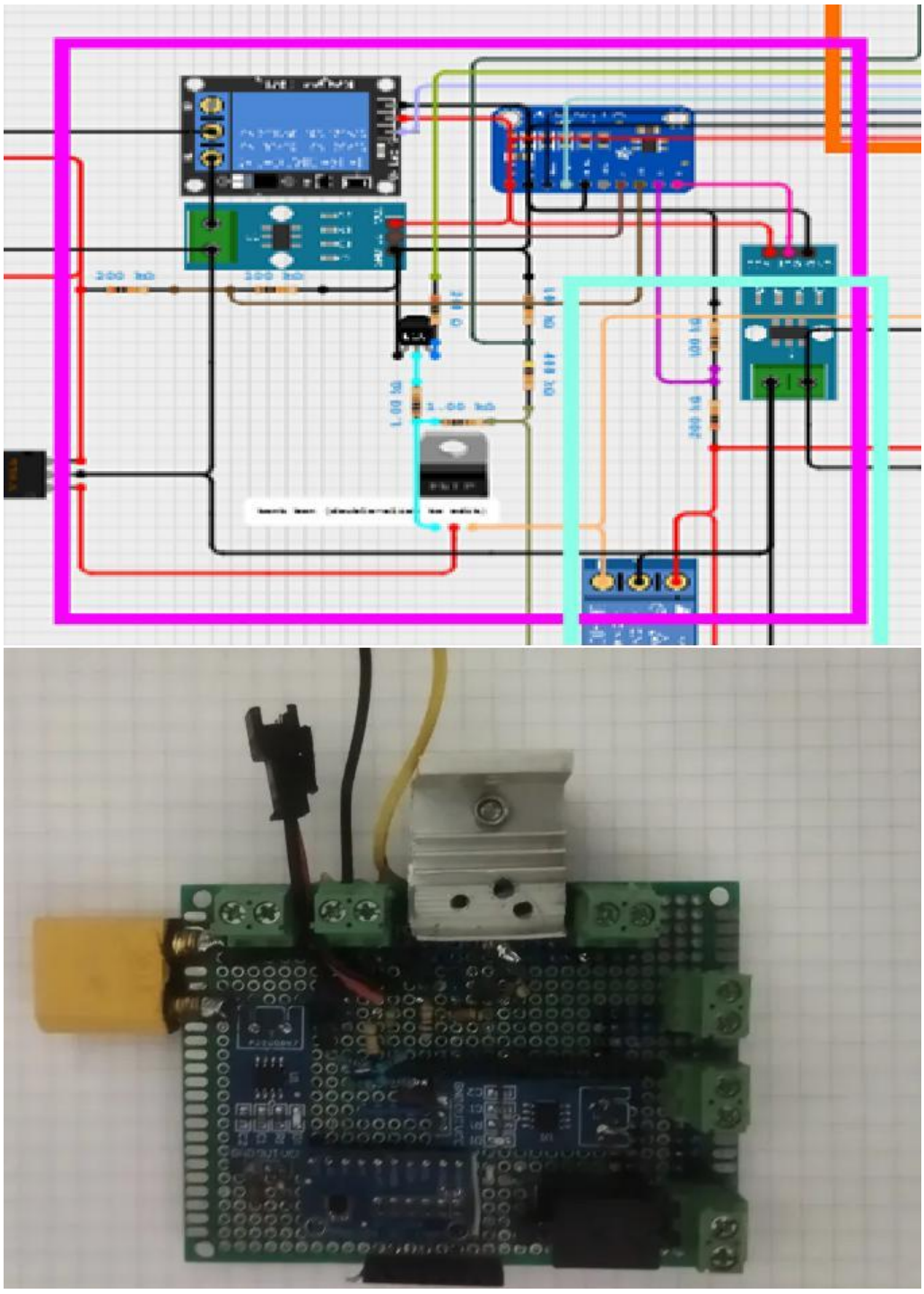


Figure 3.24: carte de puissance avec schéma

comme cette dernière possède beaucoup de composants il est important de choisir l'enlacement de ce dernier avant de les souder nous avons mis les composants à signal analogique comme priorité pour avoir un chemin court ainsi évitée davantage les bruits, puis tracer le chemin des circuits.

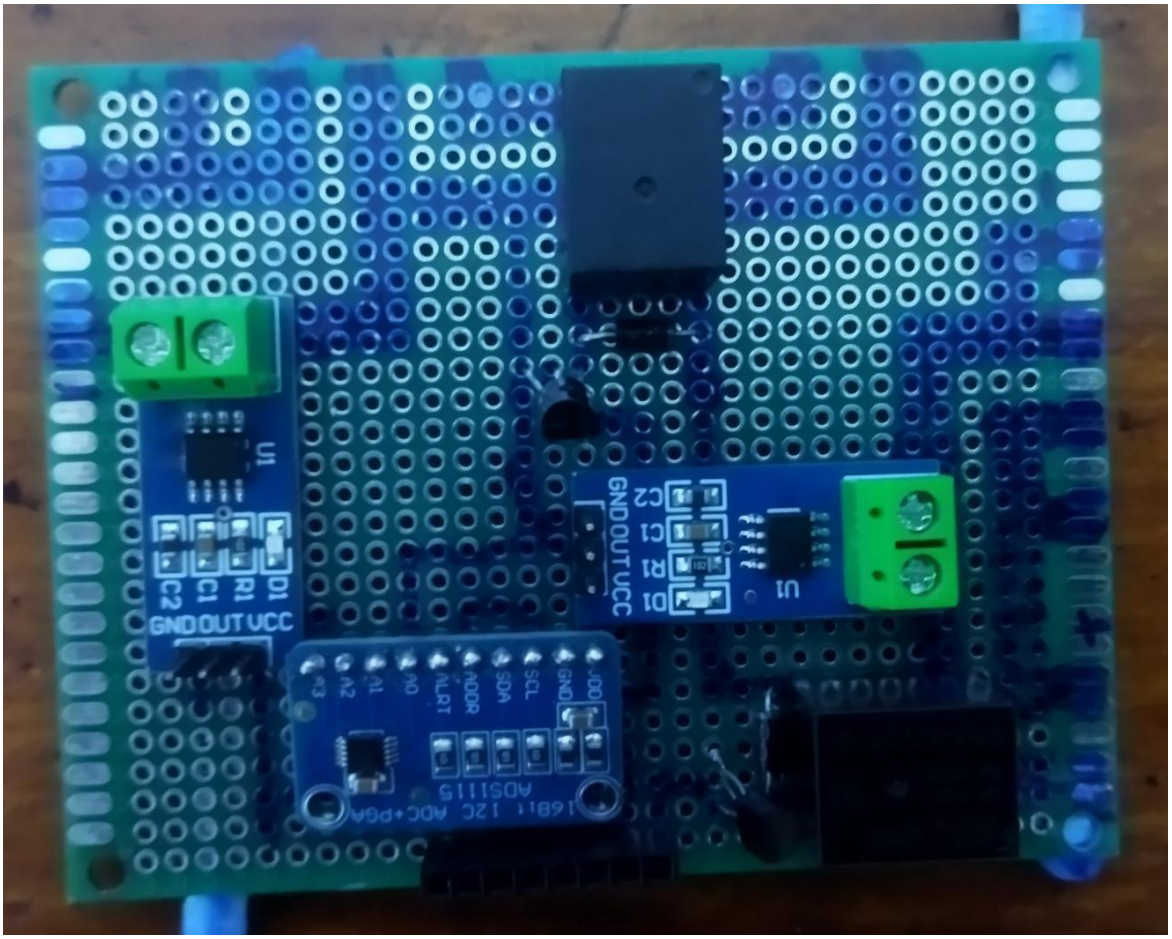


Figure 3.25: creation de cercuit

un connecteur xt60 a été nécessaire pour connecter et supporter le haut corrant venant des batteries Cependant la soudure de ce dernier peu s'avérer difficile vue l'épaisseur du métal.

Figure 3.26: connecteur xt60

- **Carte d'interface** Permet d'indiquer les donner ainsi que d'interagir avec écran bouttons led (Figure 3.27):

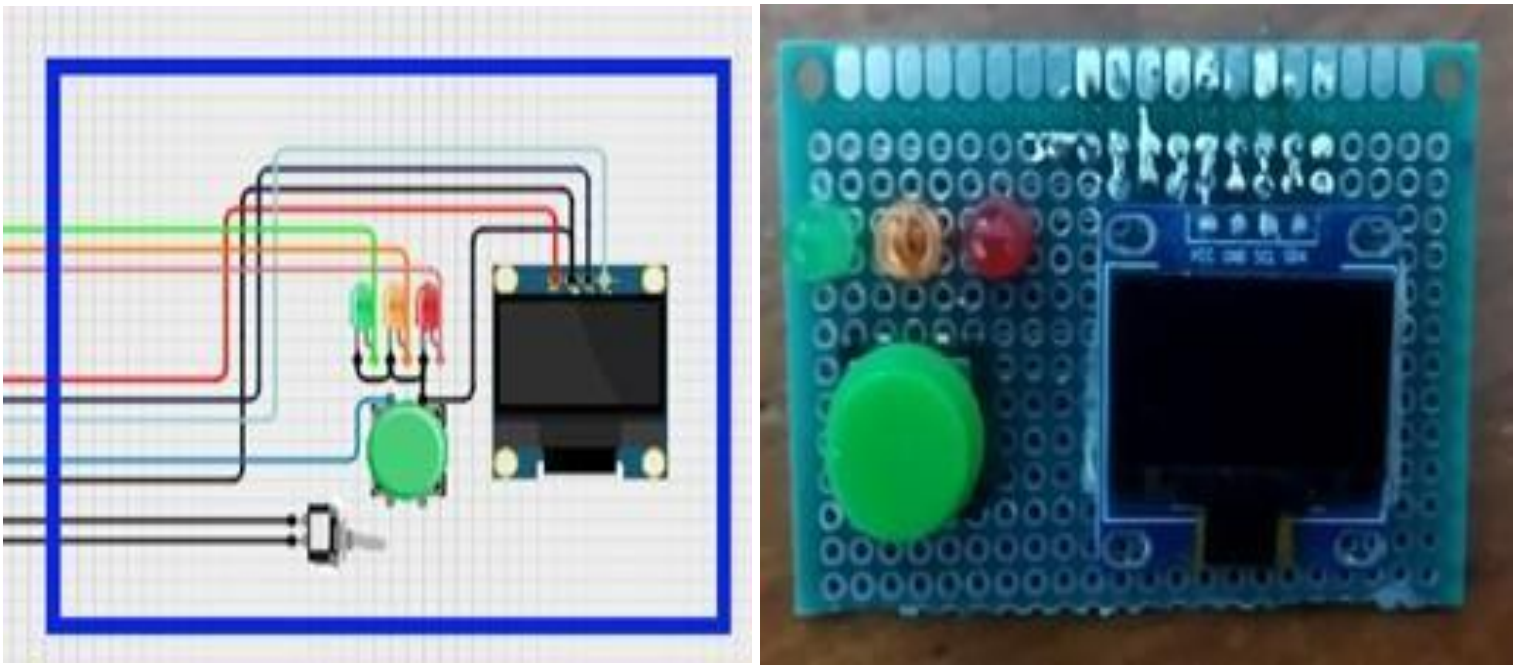
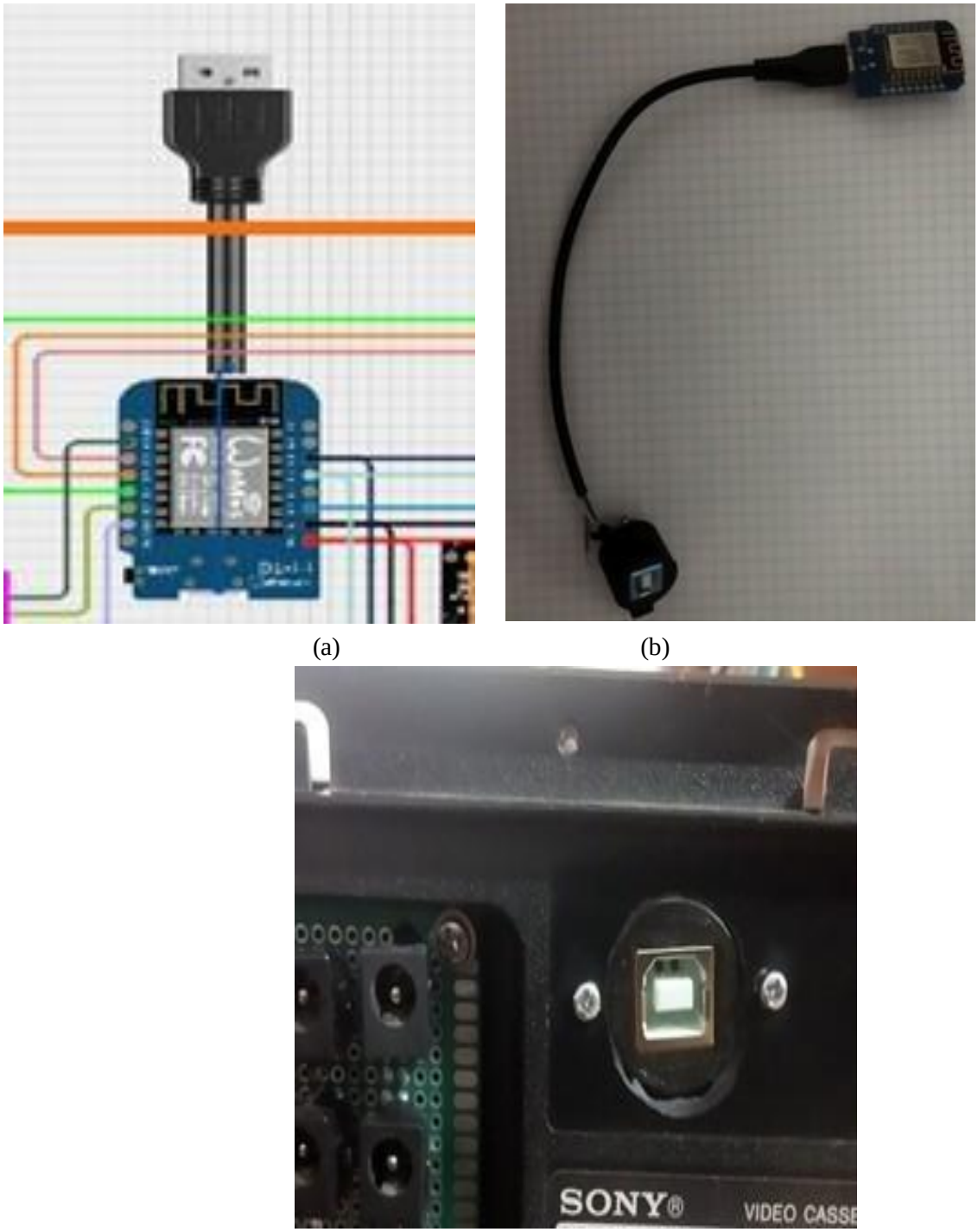


Figure 3.27: Carte d'interface avec schéma

Pour protéger le sécurit nous avons créé le couvert suivant sous fusion 360 (Figure 3.28)

Figure 3.28: couvert

Ainsi que cette adaptateur micro usb male vers usb type b femelle pour communiquer avec le port serie du micro controlleurs sans l'interface esp permettant de lire les informations du port (Figure 3.29).



(a) (b)

(c)

Figure 3.29: Adaptateur micro usb

Nous pouvons constater que le fils rouge n'est pas connecté ceci est fait exprès afin d'éviter les conflits d'alimentations et n'affecte en aucun cas le fonctionnement du câble usb.

partie logiciel

l'organigramme suivant represante le fonctionnemant general du code necessaire au fonctioemment de l'enduleur.

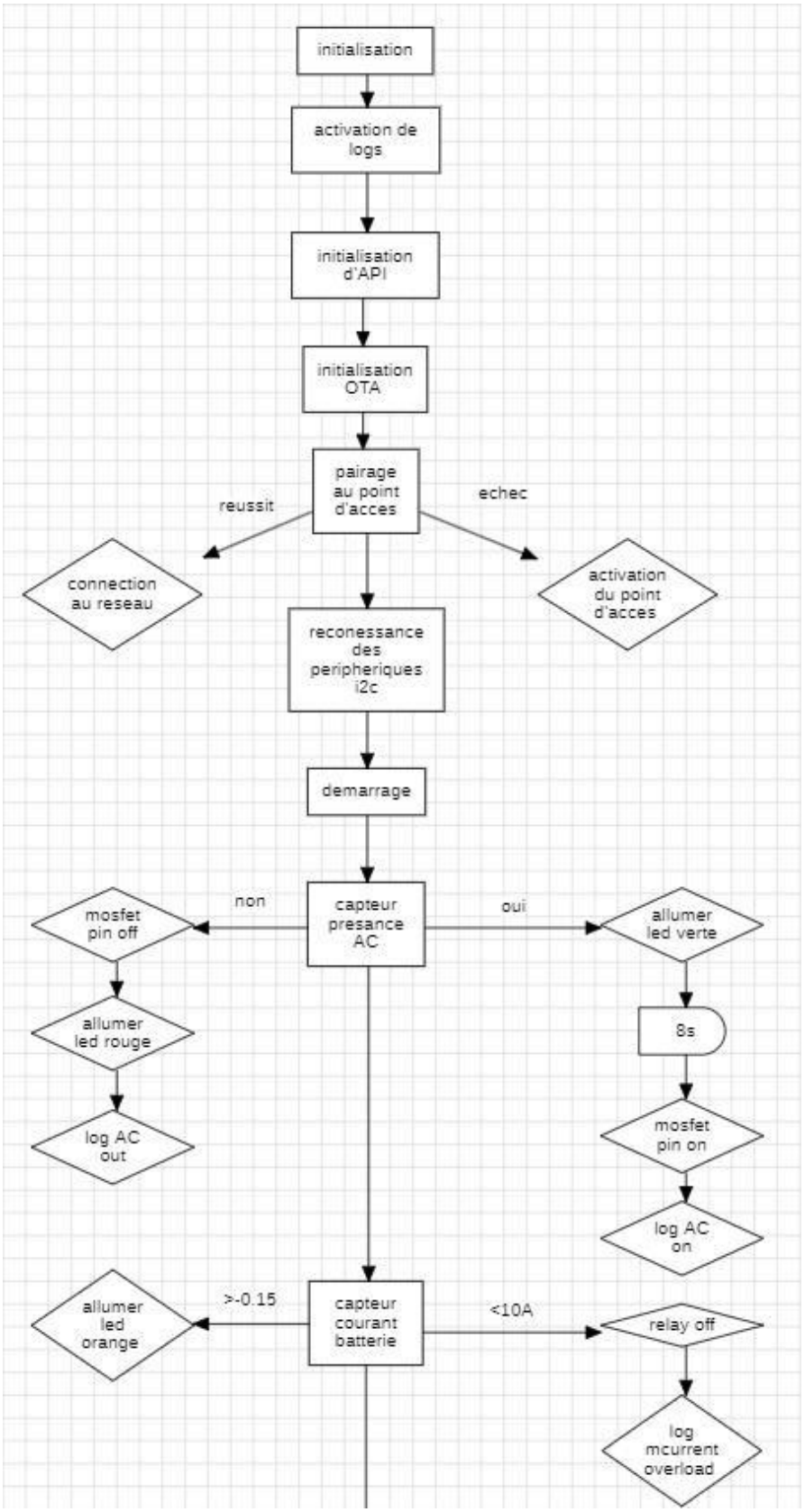


Figure 3.30: organigramme onduleur 1

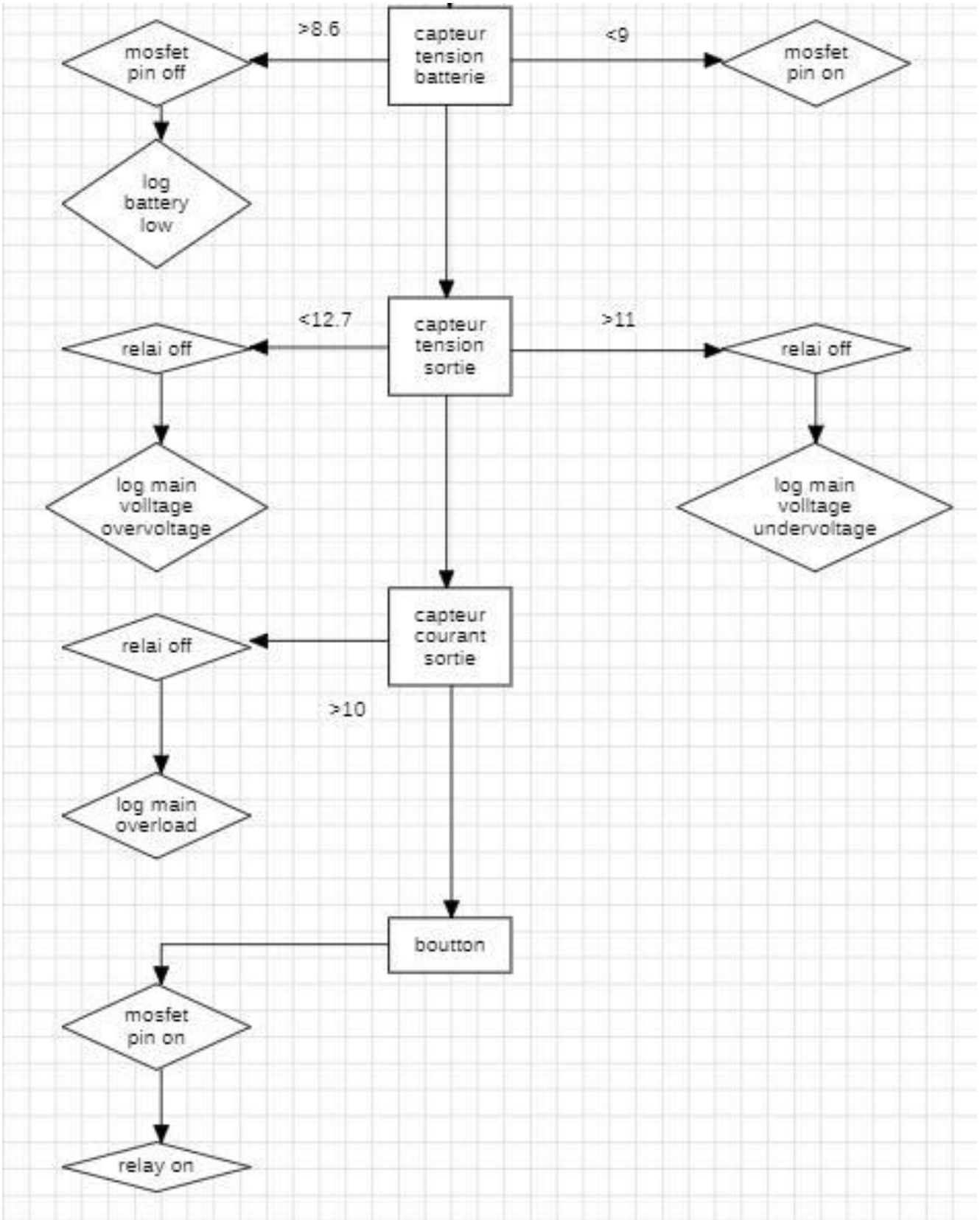


Figure 3.31: organigramme onduleur 2

le code suivant gère le fonctionnement du microcontrôleur est ces composants dans notre onduleur en respectant l'organigramme precedant .

```
esphome:
  name: ac-dc-ups
  friendly_name: AC-DC_UPS

esp8266:
  board: esp01_1m

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "kcMgik30NaNX9BAGRNl9QHNwGFPrLLbKJfDFOwaWbTM="

ota:
  password: "e2a9d6ff512c745d7ba21714d33f8231"

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

  # Enable fallback hotspot (captive portal) in case wifi
     connection fails
  ap:
    ssid: "D1-Mini-Type-C Fallback Hotspot"
    password: "hLYovQxrSaIZ"

captive_portal:
i2c:
  sda: GPIO4
  scl: GPIO5
  scan: True
```

```
# external ADC
ads1115:
  - address: 0x48

sensor:
#AC sensor
  - platform: adc
    pin: A0
    name: "ac prensance sensor"
    update_interval: 1ms
    id: "ac"
    filters:
      - sliding_window_moving_average:
          window_size: 10
          send_every: 10
    on_value_range:
        #trun on/off battery power
      - above: 0.5
        then:
          - switch.turn_on: led_green
          - switch.turn_off: led_red
          - logger.log: "AC out"
          - delay: 6s
          - switch.turn_off: relay2
      - below: 0.49
        then:
          - switch.turn_on: relay2
          - switch.turn_on: led_red
          - switch.turn_off: led_green
          - logger.log: "AC on"

#battery current
  - platform: ads1115
    multiplexer: 'A3_GND'
    resolution: 16_BITS
    gain: 6.144
    name: "batterry Current Sensor"
```

```
    id: "bcurrent"
    update_interval: 10ms
    filters:
      - sliding_window_moving_average:
          window_size: 100
          send_every: 100
      - lambda: return (round((x*65636/5)/1)*1-32768)/1311;
          #return value in A
      - offset: -0.03
    unit_of_measurement: "A"
    accuracy_decimals: 2
    on_value_range:
        #trun on/off led o for charging state
      - above: -0.10
        then:
          - switch.turn_off: led_orange
      - below: -0.15
        then:
          - switch.turn_on: led_orange
        #current protection above 10A
      - above: 9.9
        then:
          - switch.turn_off: relay2
          - logger.log: "mcurrent overload"

#battery voltage
  - platform: ads1115
    multiplexer: 'A2_GND'
    resolution: 16_BITS
    gain: 6.144
    name: "batterry voltage Sensor"
    id: "bvoltage" update_interval:
    10ms
    filters:
      - sliding_window_moving_average:
          window_size: 100
          send_every: 100
```

```
    - lambda: return x*3; #return value in V
    - offset: -0.0
  unit_of_measurement: "V"
  accuracy_decimals: 2
  on_value_range:
       #trun on/off battery for low voltage protection
     - above: 9.0
       then:
         - switch.turn_on: relay
         - logger.log: "main voltage restored"
     - below: 8.6
       then:
         - switch.turn_off: relay
         - logger.log: "battery low"

#main current
 - platform: ads1115
  multiplexer: 'A0_GND'
  resolution: 16_BITS
  gain: 6.144
  name: "main Current Sensor"
  id: "mcurrent"
  update_interval: 10ms
  filters:
    - sliding_window_moving_average:
        window_size: 100
        send_every: 100
    - lambda: return (round((x*65636/5)/1)*1-32768)/1311;
        #return value in A
    - offset: -0.03
  unit_of_measurement: "A"
  accuracy_decimals: 2


#main voltage
 - platform: ads1115
  multiplexer: 'A1_GND'
```

```
    resolution: 16_BITS
    gain: 6.144
    name: "main voltage Sensor"
    id: "mvoltage"
    update_interval: 10ms
    filters:
      - sliding_window_moving_average:
          window_size: 100
          send_every: 100
      - lambda: return x*3; #return value in V
      - offset: -0.0
    unit_of_measurement: "V"
    accuracy_decimals: 2
    on_value_range:
        #trun on/off battery for low voltage protection
       - above: 12.7
         then:
           - switch.turn_on: relay2
           - logger.log: "main overvoltage"
       - below: 11
         then:
           - switch.turn_off: relay2
           - logger.log: "main undervoltage"

#I2C display
display:
  - platform: ssd1306_i2c
    model: "SSD1306 128x64"
    reset_pin: GPIO0
    address: 0x3C
    lambda: |-
      it.printf(0, 0, id(fonts),"batC: %.2f A"
          ,id(bcurrent).state);
      it.printf(0, 15, id(fonts),"batV: %.2f V"
          ,id(bvoltage).state);
      it.printf(0, 30, id(fonts),"mainC: %.2f A"
          ,id(mcurrent).state);
```

```
    it.printf(0, 45, id(fonts),"mainV: %.2f V"
       ,id(mvoltage).state);
  update_interval: 1s

font:
 - file: "Montserrat-Medium.ttf" # path font file
   id: fonts
   size: 15

# control button
binary_sensor:
  - platform: gpio
    pin:
      number: 2
      inverted: yes
      mode: INPUT_PULLUP
    name: "button"
    on_press:
     then:
       - switch.turn_on: relay

switch:
# main relay
 - platform: gpio
   name: "Relay"
   pin:
     number: 15
   inverted: false
   id: relay
   restore_mode: ALWAYS_ON
   on_turn_on:
   - if:
      condition:
        sensor.in_range:
          id: bcurrent
          above: 10.0
      then:
```

```
      - switch.turn_off: relay
      - logger.log: "mcurrent overload"

# battery relay/mosfet
  - platform: gpio
    name: "Relay2"
    pin:
      number: 13
    inverted: false
    id: relay2
    restore_mode: ALWAYS_ON
    on_turn_on:
    - if:
        condition:
          sensor.in_range:
            id: mcurrent
            above: 10.0
        then:
        - switch.turn_off: relay2
        - logger.log: "mcurrent overload"

# led red power outage
  - platform: gpio
    name: "LED_R"
    pin:
      number: 16
      inverted: no
    id: led_red
    restore_mode: ALWAYS_OFF

# led orange charging
  - platform: gpio
    name: "LED_O"
    pin:
      number: 14
      inverted: no
    id: led_orange
```

```
    restore_mode: ALWAYS_OFF

# led green on grid
  - platform: gpio
    name: "LED_G"
    pin:
      number: 12
      inverted: no
    id: led_green
    restore_mode: ALWAYS_ON
```

conclusion :

Après avoir intégré l'onduleur à home assistant nous avons réussi à garder notre system en marche en cas de coupure de façon efficace mais aussi à lire les données du capteur en direct ainsi que de Contrôler les sorties de ce dernier.

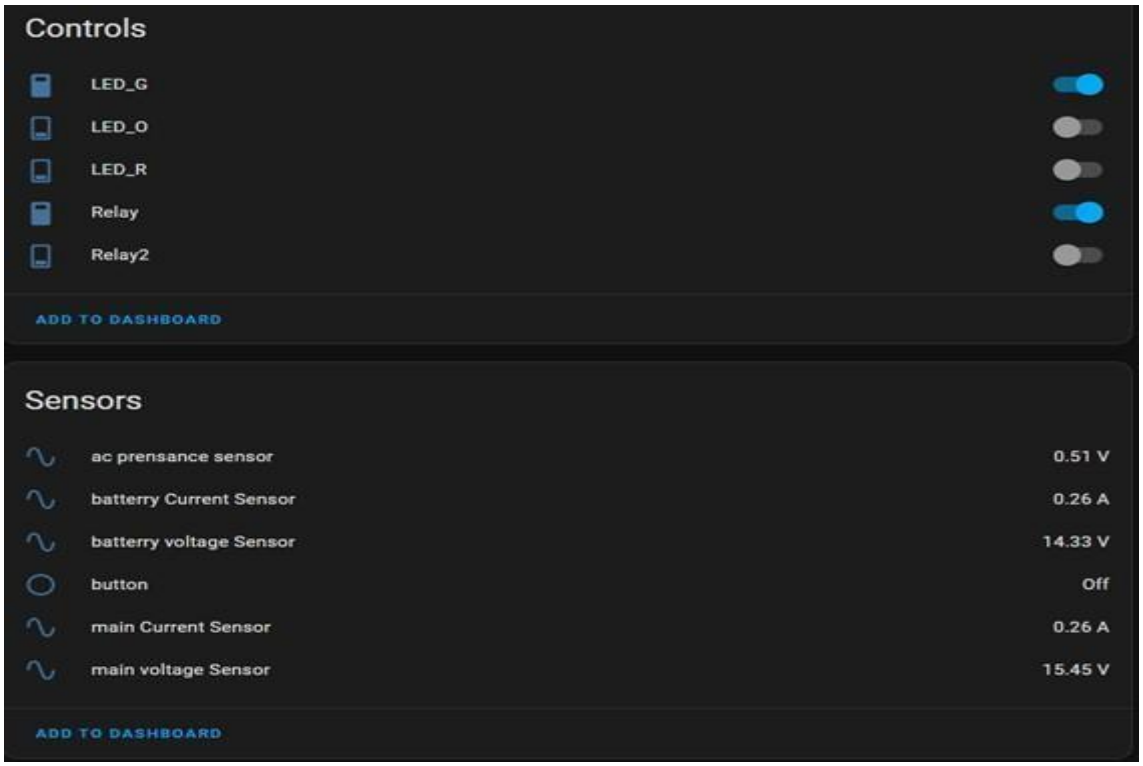


Figure 3.32: Controller les sorties

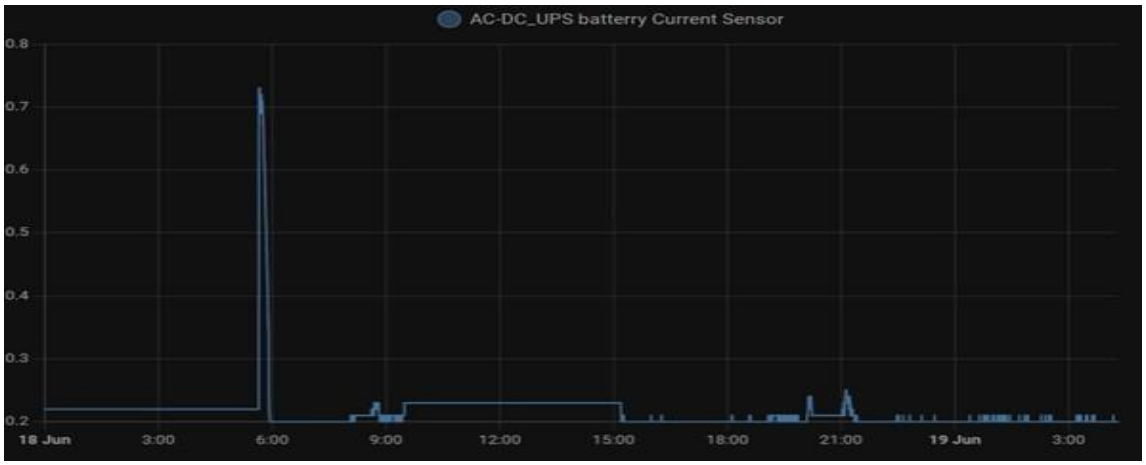


Figure 3.33: Controller les sorties

defficuté

il est imperatif d'utilisé un mosfet p-channel pour switch et munis d'un dessipateur thermique car contrairement a un relay sa

Conclusion Générale

Après la mise en place de notre écosystème domotique, il est évident que les avantages offerts par notre solution dépassent celles proposées par les systèmes grand public. Voici les principaux points à considérer :

Amélioration de la fiabilité

Contrairement aux solutions grand public, notre système domotique se distingue par sa fiabilité, sa stabilité et son indépendance vis-à-vis des accès extérieurs. Ainsi, les utilisateurs bénéficient d'une expérience plus sécurisée, essentielle pour maintenir un fonctionnement optimal au quotidien.

Automatisation Avancée et Accessible

Le système permet une programmation simple et a un potentiel illimité, similaire à celui des automates programmables industriels, mais il n'est pas réservé au domaine industriel. Il est également accessible aux résidences et aux petites entreprises sans nécessiter d'investissements coûteux. Cela permet d'améliorer considérablement l'efficacité opérationnelle et de libérer du temps pour les utilisateurs.

En substance, notre écosystème domotique représente une avancée significative vers une gestion intelligente et efficace pour répondre aux besoins des utilisateurs. Grâce à sa fiabilité renforcée et à son automatisation avancée, notre solution promet de transformer positivement la façon dont les utilisateurs interagissent avec leur environnement domestique. Nous pensons que cette approche est l'avenir de la technologie résidentielle, offrant des solutions adaptées aux besoins modernes.

Bibliographie

[1] https://www.seeedstudio.com/blog/2020/05/28/meet-the-brand-new-raspberry-pi-4-8gb-ram/

[2] https://www.melopero.com/en/shop/raspberry-pi/5inch-capacitive-touch-screen-lcd-h-slimmed-down-version-800x480-hdmi-toughened-glass-panel-low-power/

[3] https://www.prince-tech.com.tw/products_detail/55.htm

[4] https://www.westerndigital.com/products/memory-cards/sandisk-ultra-uhs-i-chromebook-microsd?sku=SDSQUAB-064G-GN6FA

[5] https://a.aliexpress.com/_EHrKPcj

[6] https://www.amazon.com/Stayhome-Converter-Voltage-Vehicle-Display/dp/B07STWJ9ZK

[7]https://fr.aliexpress.com/item/4000812444223.html?spm=a2g0o.order_list.order_list_main.247.4c125e5bKA7FdF&gatewayAdapt=glo2fra

[8] https://www.raspberrypi.com/software/

[9] https://www.youtube.com/watch?v=EWObIl7SCwY

[10] https://github.com/home-assistant/supervised-installer

[11] https://esphome.io/index.html

[12] https://www.isoled.shop/en/led-technik/led-steuerungen/zigbee-controller.html

[13] https://www.securitysystemsnews.com/tag/z-wave-alliance

[14] https://commons.wikimedia.org/wiki/File:Bluetooth_FM_Color.png

[15] https://www.flickr.com/photos/43132845@N03/3972857586/

[16] https://www.quora.com/Would-choosing-channels-other-than-1-6-or-11-have-any-effect-on-my-WiFi-connection

[17] https://www.algeria.ubuy.com/en/product/BTR00ESXA-sony-model-slv- 688hf-vcr-plus-vhs-player-recorder-remote-works