21/06/2021

Rapport de stage

Conception de boîtiers connectés pour l'écologie

Lucas Bessiere
UNIVERSITÉ PARIS SACLAY – EGCE - IRD

Rapport de Stage :

Conception de boîtiers connectés pour l'écologie

Du 19 avril 2021 au 29 juin 2021

Tuteur de stage : François Rebaudo

Professeur référent : Xavier Mininger

Etablissement Scolaire : IUT de Cachan

Etablissement d'accueil : Institut de Recherche pour le Développement – laboratoire Evolution Génome Comportement Ecologie

Remerciements

Je souhaite avant tout remercier M.Rebaudo, mon tuteur de stage, pour sa bienveillance et sa disponibilité tout au long de ce projet. Il a toujours été présent pour m'aider et me transmettre ses connaissances ce qui m'a permis d'avancer à un bon rythme dans le projet.

Je souhaite remercier également M.Emond, informaticien au CNRS, qui m'a aidé et conseillé durant ce projet sur la partie hardware.

Enfin, je tiens à témoigner toute ma gratitude à tout le personnel de l'EGCE pour m'avoir aidé à m'intégrer au mieux et pour avoir su m'aider lorsque j'avais des questions.

Résumé

La lutte contre les insectes ravageurs des cultures est un enjeu important au niveau mondial du fait des pertes de récoltes, et plus particulièrement dans les pays en voie de développement. Les cultures y sont en effet essentielles pour nourrir la population. Etudier le cycle de vie de ces insectes peut permettre de mettre en place des stratégies de lutte plus économique et respectueuses de l'environnement.

Pour ce faire il est nécessaire de mesurer les variables climatiques et leurs variations dans le temps et dans l'espace. Des stations météorologiques sont déjà présentes dans de nombreux pays mais elles ne sont pas forcément situées au niveau des lieux d'études et ne permettent pas une mesure des variables ressenties par les insectes.

Ce stage a pour objectif de concevoir un dispositif simple qui mesure les valeurs des variables climatiques et les envoie à un serveur. Il doit pouvoir être déployé par des personnes non spécialisées dans l'électronique dans les pays d'intervention de projet.

Le dispositif développé est composé d'un microcontrôleur de type ESP32 relié à des capteurs (BME680 pour la température, l'humidité relative et la pression atmosphérique ; SI1145 pour le rayonnement lumineux ; DS18B20 pour une mesure complémentaire de la température). Les données sont envoyées à l'API du serveur via des requêtes POST en utilisant la carte wifi intégrée de l'ESP32. L'alimentation du dispositif se réalise grâce à un panneau solaire associé à une batterie. Un boîtier réalisé par impression 3D permet d'assurer la protection du dispositif. Suite à la mise en œuvre du système, les résultats obtenus sont concluants. Le système solaire permet de recharger la batterie entièrement durant les jours de beau temps. Ce projet est fonctionnel et il ouvre une multitude de perspectives d'optimisation.

Table des matières

1. Introduction

1.2. Contexte

Ce stage s'insère dans le cadre du projet de recherche « *Predicting Insect Pest Phenology* » (PI2P) financé par l'Agence Nationale de la Recherche et développé au sein de l'Unité Mixte de Recherche Évolution Génome Comportement et Écologie (EGCE http://www.egce.cnrs-gif.fr/ ; https://pi2p.ird.fr). Le projet PI2P a pour objectif de contribuer à une meilleure compréhension de la réponse des insectes à la température dans un contexte de changement climatique. Pour mener à bien cet objectif, une activité vise à modéliser le cycle de vie des insectes ravageurs des cultures en Afrique de l'Est et en Amérique du Sud (en particulier au Kenya et en Bolivie). Pour ce faire, il est notamment nécessaire de connaître l'environnement climatique au plus près des insectes. Pour cela il a été développé un premier prototype de capteurs connectés sur la base d'un ordinateur monocarte Raspberry Pi relié à internet et envoyant les données des capteurs à un serveur centralisant l'information.

L'objectif du stage est de réfléchir et de concevoir un dispositif qui soit plus optimisé et plus facile à déployer sur le terrain que celui existant. Dans ce stage, nous allons créer un dispositif qui permet de récolter des informations sur l'environnement (température, humidité relative, pression atmosphérique et un indicateur de l'intensité lumineuse). Un dispositif complémentaire a été développé en 2020 par un étudiant du parcours GEII qui repose sur un microcontrôleur de type Arduino alimenté par piles (2xAA) et permettant l'envoi des données par radio (Lora RFM9x sur la bande 868 MHz) entre le dispositif complémentaire et le dispositif principal (Titouan Soulard). Dans ce stage l'objectif est d'optimiser le dispositif principal.

Actuellement, les dispositifs sont constitués d'un Raspberry pi (modèles zeroW, 3B, 3B+ et 3A+) qui récupèrent les données d'un capteur de rayonnements lumineux (SI1145 ; https://www.adafruit.com/product/1777) et d'un capteur d'humidité relative, pression atmosphérique et température (BME680 ; https://www.adafruit.com/product/3660). Un capteur de type DS18B20 est aussi connecté afin de servir de contrôle pour la mesure de la température. Une fois collectées, les données sont transmises au serveur https://pi2p.ird.fr via une API REST au sein d'une base de données postGreSQL pour les centraliser et les rendre accessibles sur le site du projet.

Le principal inconvénient de ce dispositif est qu'il est impossible d'avoir des mesures dans un lieu sans habitation proche, car il est dépendant d'une alimentation électrique et d'une connexion wifi ou par câble à Internet. Par ailleurs le dispositif complémentaire permettant l'envoi des données reste de faible porté et ne peut être utilisé dans les zones reculées.

1.2. Structure d'accueil

Ce stage est réalisé au sein de l'UMR EGCE dirigée par Laure Kaiser-Arnauld, au sein du pôle écologie. L'UMR EGCE est sous la cotutelle de l'IRD, du CNRS et de l'Université Paris Saclay. Il comporte 3 pôles, le pôle comportement, le pôle génome et le pôle écologie. Ce stage se réalise sous la tutelle de l'IRD, au sein de l'UMR EGCE, dans le pôle écologie, et dans le cadre du projet de recherche PI2P (Figure 1).

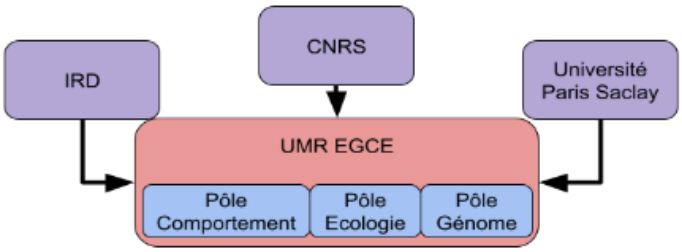

Figure 1. Organisation de l'entreprise. Organigramme représentant le fonctionnement de l'UMR EGCE. Le stage est réalisé au sein de pôle Ecologie.

2. État des lieux et proposition d'optimisation

2.1. Le système actuel

2.1.1. Description du système

Des systèmes ont déjà été développés et déployés sur le terrain pour répondre à la problématique du projet. Ils sont composés d'un Raspberry pi sur lequel sont connectés un BME680 et un SI1145. Le BME680 permet d'obtenir les informations de température, d'humidité relative et de pression atmosphérique, alors que le SI1145 permet d'obtenir des informations sur la lumière visible, la quantité d'UV et la quantité d'infrarouges. Afin de maintenir le Raspberry pi à une température acceptable, un ventilateur a été installé. Une fois

récoltées, les informations sont envoyées via internet (wifi ou filaire) au serveur du projet où elles sont accessibles pour tous, et en particulier pour les chercheurs associés au projet.

2.1.2 Avantages et inconvénients

Le premier avantage de ce système est qu'il est plus simple à débugger. En effet, la présence de la prise HDMI et des prises USB permet d'intervenir facilement pour résoudre les problèmes du système. Le second avantage est que le Raspberry pi peut stocker des données. Par conséquent, si la communication sans fil n'est plus disponible, il est toujours possible de les stocker sur le Raspberry pi et de venir les collecter via une clef USB.

Un inconvénient du système concerne la consommation du Raspberry pi. Celui-ci nécessite beaucoup d'énergie car il est conçu pour être utilisé comme un ordinateur monocarte. L'autre inconvénient est que si le boîtier se situe dans un espace sans accès à internet il ne peut pas transmettre ses données.

2.2. Proposition de solutions

L'objectif de mon stage est d'améliorer le système existant et de proposer des alternatives. Dans un premier temps, nous voulons optimiser la carte de commande. Nous avons donc cherché une carte moins imposante et moins coûteuse. Nous avons donc besoin d'un composant nous permettant de recevoir les données des capteurs et de les transférer directement au serveur. Il est donc important que le composant ait possibilité de se connecter au réseau internet et des GPIO pour recevoir les données. Le choix se porte entre un microcontrôleur (Arduino, esp32,) ou un ordinateur monocarte (Raspberry, dragon board…). La décision s'est portée sur un ESP32-DEVKITC-32UE de la marque Expressif Systems. Ce microcontrôleur est équipé du wifi et a un coût inférieur à une dizaine d'euros ce qui est moins onéreux qu'un Raspberry pi ou un modèle Arduino avec une carte wifi.

Après avoir décidé du microcontrôleur, nous avons étudié les différentes possibilités pour ce projet. La première possibilité consiste à réaliser un système similaire à celui existant en modifiant le microcontrôleur pour obtenir un système plus compact. La deuxième consiste à réaliser un système avec les mêmes capteurs que le projet existant mais en les connectant à un ESP32 qui serait alimenté par un module solaire avec une batterie. Cette solution est intéressante car dans les pays où seront installés les capteurs il est fréquent qu'il y ait des coupures de courant. La troisième consiste à prendre la deuxième solution et à lui ajouter un

module pour carte Sim et d'envoyer les données sous forme de SMS à un Raspberry pi connecté au même module et qui transmet les données dans les sms au serveur via le wifi. La quatrième est quasiment identique à la troisième à la différence que les données ne sont pas envoyées via le GSM mais via le GPRS qui permet de se passer du module de réception et donc d'envoyer les données directement au serveur.

Proposition :	ESP32 seul	ESP32 avec module solaire	ESP32 avec module solaire et Système GSM	ESP32 avec module solaire et Système GPRS
Avantage :	. Plus compacte que le système actuel	. Autonome énergétiquement	. Autonome énergétiquement. . Ne nécessite pas de wifi	. Autonome énergétiquement. . Ne nécessite pas de wifi . Pas besoin d'un module de relais
Inconvénient :	. Nécessite un réseau wifi. Pas autonome énergétiquement	. Nécessite un réseau wifi	. Achat d'une carte Sim et payer à chaque envoie de donnés . Le module a un prix non négligeable . Nécessite un module de relais	. Achat d'une carte Sim et payer à chaque envoie de donnés . Le module a un prix non négligeable

Table 1. Tableau comparatif des propositions. Ce tableau synthétise les avantages et inconvénients des différentes propositions de projet.

2.3. Choix du Système : ESP32 avec alimentation solaire

2.3.1 Intérêt

Notre choix s'est porté sur le système ESP32 alimenté par un système solaire car il répond à une première problématique qui est l'énergie. Nous pouvons choisir de l'utiliser avec le panneau solaire ou sur alimentation secteur. L'intérêt de ce système est qu'il est possible de choisir soit une alimentation solaire totalement autonome, soit une alimentation secteur permettant de recharger la batterie en continu et de faire fonctionner le système même en cas de coupure d'électricité. Il permet aussi de faire en sorte que le projet soit plus écologique.

2.3.2. Cahier des charges

Pour réaliser ce projet nous avons eu plusieurs contraintes. La première a été de réaliser un système avec des composants faciles à trouver et disponibles sur des sites spécialisés dans l'électronique afin d'assurer un approvisionnement possible partout et à long terme. Le prix du boîtier doit être le plus faible possible car il est destiné à des pays en voie de développement. Une autre contrainte est celle de l'énergie. En effet l'objectif est d'avoir un

boîtier fonctionnant avec l'énergie solaire et donc notre microcontrôleur doit être le moins énergivore possible. Enfin, il faut que le boîtier soit le plus compact possible.

2.3.3. Choix des composants

Pour le microcontrôleur, notre choix s'est porté sur un ESP32. Il est doté d'une puce wifi, de ports analogiques et d'un bus I^2C. Les avantages de l'ESP32 sont sa facilité de programmation, l'IDE Arduino est compatible tout comme la majorité des bibliothèques et son prix, moins de 9€.

Pour les capteurs, nous avons décidé de reprendre les mêmes capteurs que le projet initial, c'est à dire un BME680 et un SI1145. Ceux-ci sont fiables et permettent d'obtenir six informations différentes avec seulement deux cartes filles. Pour obtenir une température de contrôle, nous avons ajouté un DS18B20 qui permet d'indiquer une information de température supplémentaire.

Concernant la recharge solaire, nous avons choisi de prendre un kit qui permet de recharger un téléphone avec un panneau solaire et une batterie. L'intérêt est que les composants sont compatibles entre eux ce qui limite les risques de panne. Le deuxième intérêt est que la sortie est du 5V via USB et suffit de brancher l'ESP32 via son connecteur USB pour l'alimenter. L'autre intérêt est que ce système peut être réutilisé dans d'autres projets futurs car la majorité des microcontrôleurs ont une prise USB intégrée.

L'utilisation d'une imprimante 3D pour fabriquer le boîtier a été privilégiée. Ce choix est dû au fait que nous avons besoin d'un boîtier le plus compact possible et susceptible d'évoluer au fil du temps. Un autre avantage est qu'il est moins coûteux d'imprimer un boîtier que d'en acheter un.

3. Réalisation

3.1. Processus de conception

Pour réaliser ce projet, nous devons réaliser quatre parties différentes qui doivent s'assembler à la fin du projet. Nous allons, dans un premier temps, étudier les capteurs et trouver un moyen d'obtenir leurs valeurs. Puis, il s'agit de comprendre comment envoyer les données au serveur. Par la suite, il faudra fabriquer le système solaire qui emmagasinera

l’énergie la journée pour la restituer la nuit. Enfin il ne restera plus qu'à créer un boîtier et à l’imprimer. Concernant les relevés, nous avons décidé de mettre en sommeil l'ESP 32 pendant un certain temps, puis de le réveiller pour qu’il refasse un relevé.

3.2. Obtention des données

3.2.1. DS18B20

Pour le DS18B20 le schéma de câblage est simple, la masse du capteur est connecté à la masse du microcontrôleur et le +5V aux broches correspondantes et la broche des données à une entrée de l’ESP32. Ici nous avons choisi d’utiliser le GPIO 16 du microcontrôleur. Il convient de rajouter une résistance de 4.7 KΩ entre le +5V et les données.

La particularité du DS18B20 est qu’il est possible d’en mettre autant que nécessaire tant que le câblage est respecté. Une fois câblé, nous entrons un code dans l’ESP32 qui a pour but, cette fois, de récupérer la valeur de la température et non l’adresse du capteur.

3.2.2. BME680

Le BME680 est un capteur qui permet d’obtenir plusieurs valeurs différentes (humidité relative, température, pression, gaz). Pour transférer ces données il s’agit connecter le capteur à l’ESP32 via un bus I²C. Sur l’IDE Arduino il existe des librairies pour obtenir les différentes informations du capteur. Pour notre projet, nous choisissons d’utiliser uniquement les informations de température, de pression et d’humidité relative. Pour le connecter au microcontrôleur nous relions le port “data” (SDI) au GPIO 21 et le port “clock” (SCK) au GPIO 22.

3.2.3. SI1145

Comme le BME680 le SI1145 est un capteur qui transmet ses données via un bus I²C, nous pouvons connecter les deux capteurs sur les deux mêmes GPIO de l’ESP32. Sur l’IDE d’Arduino, il existe une librairie qui permet d’appeler les valeurs du capteur. Avec ce capteur nous cherchons à obtenir des informations concernant la lumière (UV, IR, Visible). Pour le connecter au microcontrôleur, comme précédemment nous relions le port “data” (SDA) du SI1145 au GPIO 21 et le port “clock” (SCL) du BME680 au GPIO 22.

3.2.4. Tension de la batterie

Comme nous utilisons une batterie au lithium, il faut faire attention à la tension de la batterie. D'après les données du constructeur, pour que la durée de vie de la batterie soit optimale, il ne faut pas que la tension passe en dessous de 2.75V. Si elle atteignait ce palier, il faudrait que quelqu'un la débranche. Pour savoir quand intervenir, nous allons utiliser un pont diviseur de tension qui va permettre de faire passer la tension en entrée du pont diviseur de 3.7V à 3.3V car les GPIO du microcontrôleur n'acceptent qu'une tension entre 0V et 3.3V.
Pour ce faire nous avons à choisir deux résistances. Pour les choisir, nous choisissons une valeur arbitraire pour la résistance R2, 1000 Ω.

La détermination de la valeur de R1 se fait avec la formule du pont diviseur de tension : $V_{out} = V_{in} * (R2 / (R1 + R2))$.

Avec : $V_{out} = 3.3$; $V_{in} = 3.7$; R2 : 1000

On obtient donc l'équation suivante :

$3.3 = 3.7 * (1000 / (1000 + R1))$

$\Leftrightarrow 3.7 / 3.3 = 1000 / (1000 + R1)$

$\Leftrightarrow (3.7 / 3.3) *1000 = 1000 + R1$

$\Leftrightarrow (((3.7 / 3.3) *1000) – 1000) = R1$

$\Leftrightarrow 1121.2 – 1000 = R1$

$\Leftrightarrow 121.2 = R1$

Nous choisissons de prendre 120 Ω car c'est une valeur de référence proche de 121.2.
Une fois connecté au microcontrôleur nous écrivons le code suivant :

Initialisation des variables :
```
const float TMaxA0 = 3.3;
int analogInput = 33;
float R1 = 120.0;
float R2 = 1000;
int valeur = 0;
int Tension;
```

lancement du setup :
```
pinMode(analogInput, INPUT);
```

lancement de la boucle :
```
valeur = analogRead(analogInput);
Tension = ((valeur * ( TMaxA0 / 4095)) / (R2/(R1+R2)));
```

Pour effectuer ce calcul, nous devons prendre en compte le fait que l'entrée analogique de l'ESP32 est connectée à un Convertisseur analogique numérique (CAN) sur 12 Bits. La valeur de la tension en entrée est donc traduite par une valeur entre 0 et 4095. Pour obtenir la valeur de la tension à l'entrée du microcontrôleur, il faut donc effectuer le calcul suivant :

T_{micro} = tension en entrée du microcontrôleur

V_{can} = Valeur entre 0 et 4095 fournie par le CAN

T_{maxCan} = Tension maximale admissible pour le CAN

$T_{micro} = V_{can} * (T_{maxCan} / 4095)$

Avec ce calcul nous obtenons la valeur de la tension à l'entrée du microcontrôleur.

Pour la suite de l'équation, nous utilisons le calcul d'un pont diviseur de tension :

$V_{out} = V_{in} * (R2 / (R1 + R2))$

Sachant que nous avons utilisé un pont diviseur de tension pour abaisser la tension, nous inversons le calcul pour obtenir la valeur avant le pont diviseur.

$V_{in} = V_{out} * (R2 / R1 + R2)$

Avec ce code, nous obtenons la valeur de la tension. En envoyant cette valeur au serveur, nous pouvons surveiller l'état de charge de la batterie.

3.3. Restitution des données

Une fois les données récupérées sur l'ESP32, il faut les envoyer sur le serveur. Pour cela nous utilisons le wifi du microcontrôleur qui envoie les données via une API de type REST (requêtes POST pour envoyer les données avec identification par clef chiffrée).

3.4. Système solaire

Nous avons choisi d'utiliser un système d'énergie autonome afin que le système ne soit pas dépendant de l'électricité quel que soit le lieu où se situera le boîtier. Les pays d'expérimentation étant situés proche de l'équateur nous avons décidé d'utiliser un système avec un panneau solaire. Pour ce faire, il faut un assemblage capable de recharger une batterie la journée pour tenir toute la nuit. Après de nombreuses recherches nous avons

trouvé un projet créé par Adafruit qui permet de recharger un smartphone avec de l'énergie solaire.

Le système est composé de quatre parties : un panneau solaire, une batterie, un module qui contrôle la charge de la batterie et un module qui permet de brancher un appareil avec une prise USB. Ce projet est très intéressant car il peut être utilisé pour d'autres expérimentations à partir du moment où le microcontrôleur peut être connecté en USB.

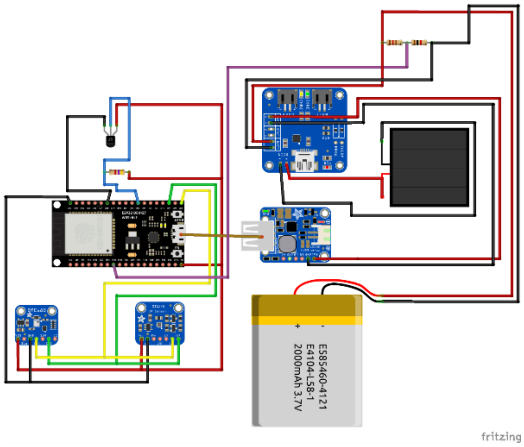

Figure 2 . Schéma du système complet. Schéma des connections à réaliser pour obtenir le système complet composé d'une batterie, d'un pont diviseur de tension, d'un module de charge, d'un panneau solaire, d'un USB booster, d'un ESP32, d'un SI1145, d'un BME680 et d'un DS19B20

3.4. Boîtier 3D

Une fois l'électronique terminée et fonctionnelle, nous avons réfléchi au boîtier qui contiendra les différents éléments. Pour le mettre au point nous avons utilisé le logiciel "openSCAD" qui est un logiciel de modélisation 3D paramétrique disponible sous licence GPL-2.

Dans un premier temps, nous avons pris les dimensions des composants puis nous avons imaginé leurs dispositions afin d'avoir un boîtier le plus compact possible tout en étant le plus rigide. Par la suite, nous avons commencé à créer le boîtier virtuellement avec les cotes choisies.

Cette première ébauche est assez satisfaisante mais comporte des défauts. Pour l'ébauche finale il faut prendre en compte plusieurs facteurs qui sont que le boîtier doit être très solide et le plus résistant aux intempéries. Pour ce faire, nous allons arrondir tous les bords pour augmenter la résistance. Sur ce modèle la quantité de plastique est importante, il faut donc enlever un maximum de plastique sans altérer la structure globale (figure 3). Une fois modélisé, le modèle a été imprimé avec l'imprimante 3D présente au Laboratoire (Figure 4).

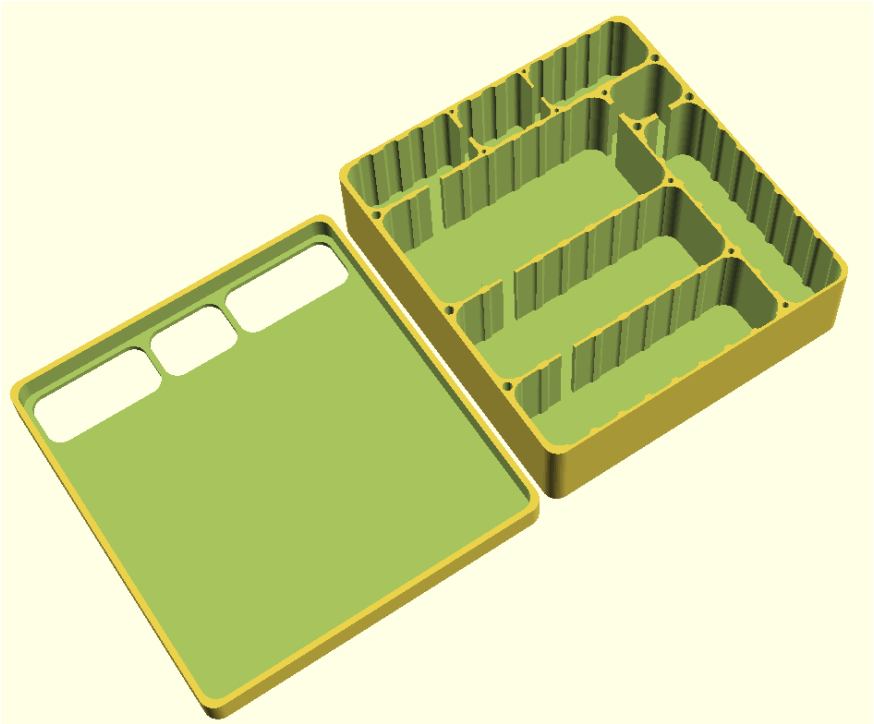

Figure 3. Image du boîtier 3D. Cette image est une capture d'écran du logiciel openSCAD, nous y voyons le boîtier optimisé pour consommer le moins de filament possible.

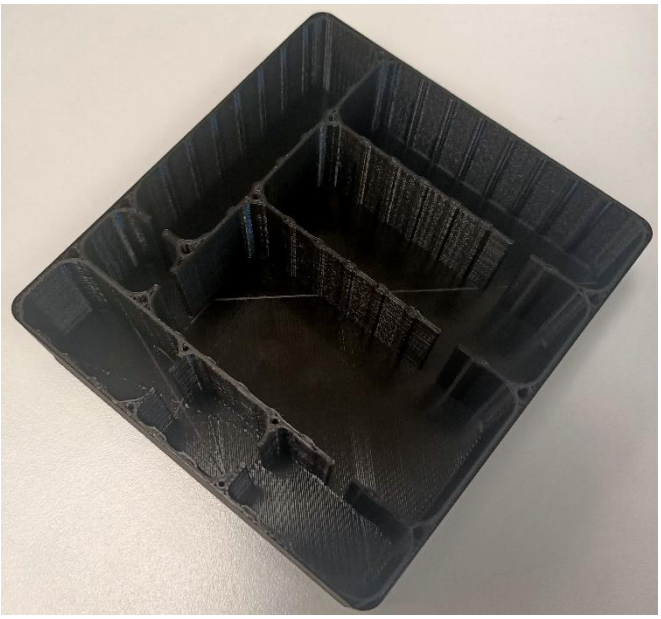

Figure 4. Image du boîtier imprimé en 3D. Ce boîtier a été imprimé en 12h avec une buse de 0.6 mm sur une imprimante de type Ultimaker 2+ et du filament PLA noir Ultimaker

4. Résultats et discussion

Une fois l'électronique terminée et le système fonctionnel, le dispositif a été mis en fonctionnement pour réaliser des tests. Pour ce test, nous envoyons les informations de tension de la batterie, de température, de pression atmosphérique, d'humidité relative et des informations concernant les rayons lumineux sur le serveur du projet. Ce test a pour but d'observer comment se comporte le système final.

4.1. Résultats

4.1.1. Système solaire

Intéressons-nous à la tension de la batterie. Contrairement aux capteurs qui ont déjà été utilisés dans le projet initial et que nous avons pu tester pendant mon stage, nous n'avions pas de référence sur le comportement de la batterie à long terme (Figure 5).

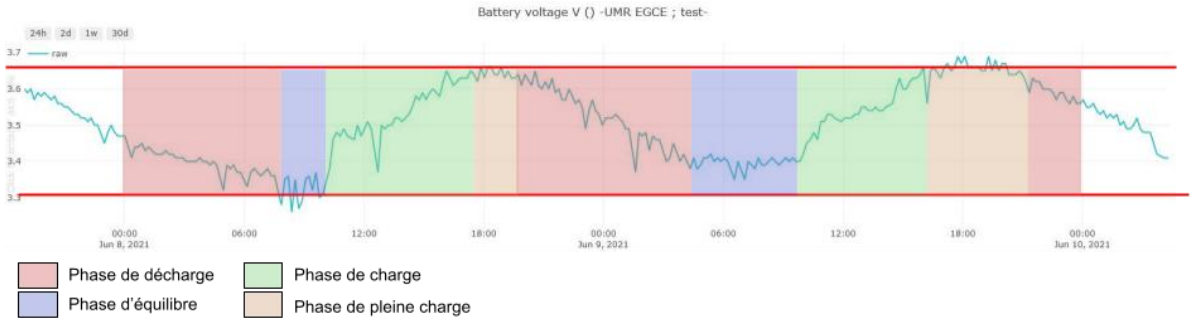

Figure 5. Graphique de la tension de la batterie sur deux jours. Ce graphique montre les phases de charges, de décharge, d'équilibre et de pleine charge.

Avec le graphique qui récapitule les valeurs de la tension par rapport au temps, nous pouvons observer de la tension de la batterie sur deux jours. La tension varie entre 3.7V et 3.3V ce qui est satisfaisant car nous ne devons pas passer en dessous des 2.75V. Les phases de charge et de décharge sont cohérentes, entre le début de soirée (environ 19h) et le milieu de matinée (environ 10h) la batterie se décharge plus qu'elle ne se recharge, c'est un comportement normal car c'est entre ces heures qu'il y a moins de soleil. En revanche, entre le milieu de matinée et le début de soirée, la tension de la batterie augmente, ce qui signifie que la batterie se recharge plus qu'elle ne se décharge. Un autre point intéressant est que la batterie remonte quasiment tous les jours à 3.7V. Cette valeur est intéressante car, une fois installé, s'il n'y a pas de soleil pendant un certain temps, le système peut continuer de fonctionner.

4.1.2. Les capteurs

Pour le SI1145, nous pouvons observer que les courbes sont cohérentes. En effet les pics de luminosité visible, infrarouge et ultraviolet correspondent aux moments où la batterie se recharge et donc au moment où la luminosité est la plus forte (Figure 6).

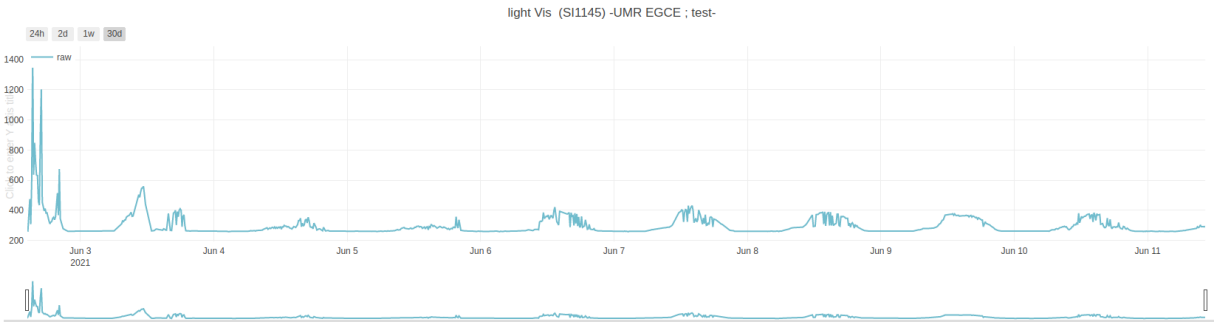

Figure 6. Graphique des variations de valeur du SI1145. Ce graphique expose les différentes variations de lumière visible pendant une semaine.

Pour le BME680 c'est comme le SI1145, les courbes sont cohérentes. Nous voyons que la température augmente en fin de matinée tout comme la luminosité. Pour l'humidité, elle augmente quand les rayonnements solaires diminuent (Figure 7). C'est dû au principe que l'air est plus sec la journée à cause des rayonnements qui l'assèchent.

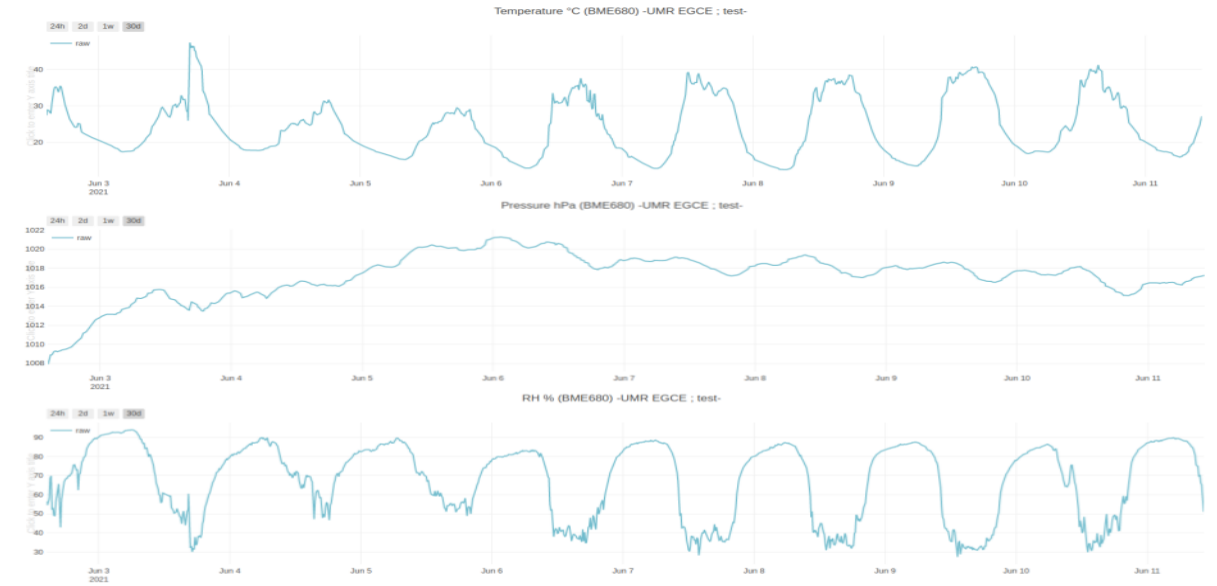

Figure 7. Graphiques des variations de valeurs du BME680. Ce graphique montre les variations de température, de pression et d'humidité relative pendant une semaine.

En ce qui concerne le DS18B20 les résultats sont loin d'être satisfaisants. Il manque énormément de valeurs. Le manque de valeurs est imputable au code qui n'envoie pas les valeurs d'erreur (-127 qui signifie que le capteur n'est pas détecté et +85 qui signifie que la valeur n'a pas pu être lue). Cependant, la cause de ces erreurs n'a pas pu être à ce jour identifiée.

4.2. Discussion

Ce projet s'inscrit dans un but de recherche dans lequel nous cherchons à étudier des insectes ravageurs par rapport aux conditions climatiques. Nous avons donc réfléchi au projet pour obtenir des informations les plus précises tout en essayant d'être en phase avec les contraintes des pays dans lesquels les dispositifs vont être déployés.

La production d'électricité est une activité qui peut être très polluante dans certains pays et donc utiliser de l'énergie renouvelable, en utilisant l'énergie solaire, pour des capteurs permet de rendre ce projet plus vert. Toutefois, nous utilisons des composants comportant du plastique et autres matériaux non naturels qui ne sont pas écologiques et qui, de plus, ont été transportés par bateau ou par camion ce qui contribue à augmenter l'empreinte carbone du projet. De même, l'utilisation d'une batterie au lithium, faiblement recyclable actuellement, est un facteur de pollution. Au regard de ce constat, nous pouvons dire que nous essayons de réduire notre impact environnemental mais le projet n'est pas totalement écologique.

Le système d'envoi des données via le wifi est très utile car il permet de ne pas engendrer des frais supplémentaires. En revanche le problème de ce système est qu'il est impossible de le poser loin d'une habitation. Une des améliorations possibles serait d'utiliser un module GSM pour envoyer les données via SMS ou GPRS au serveur. Ces options n'ont pas pu être développées dans le cadre de ce stage faute de temps.

Le boîtier imprimé en 3D est une bonne alternative par rapport à un boîtier acheté dans le commerce. En effet, cela permet d'avoir un boîtier sur mesure en parfaite adéquation avec les besoins.

Bien que le dispositif soit prometteur, il reste à déterminer l'origine du problème lié au DS18B20 et optimiser le code pour réduire la consommation électrique. Celle-ci pourrait être plus faible si nous enlevons le module USB qui fait passer la tension de 3.7V à 5V. Après plusieurs tests, il est remonté que ce module consomme en permanence 30mA. C'est problématique car l'ESP32 se met en veille entre chaque mesure et donc il ne consomme quasiment rien. Une autre amélioration pourrait être d'élaborer un module qui coupe le système si la tension de la batterie est trop faible.

5. Conclusion

Pour conclure, ce stage m'a permis de mettre mes compétences au service d'un projet ayant pour objectif de comprendre et de lutter contre les insectes ravageurs des cultures. J'ai apporté des connaissances en électronique pour répondre à une problématique qui était d'améliorer le système actuel de récolte de données climatique.

J'ai eu un vaste choix concernant les options d'optimisation et j'ai choisi de réaliser une amélioration sur la taille et le choix des composants et sur l'alimentation électrique du projet. J'ai choisi une alimentation via panneau solaire car les pays étudiés sont proches de l'équateur donc il n'est pas difficile de trouver un lieu exposé aux rayons du soleil. Un autre intérêt du système solaire est que j'ai choisi un système qui contient une prise USB Femelle comme sortie de puissance. L'avantage est qu'il peut être facile de l'utiliser dans un autre projet nécessitant une alimentation autre qu'une prise électrique.

Le système avec panneau solaire à un fonctionnement assez simple, un panneau solaire recharge une batterie qui donne l'énergie au microcontrôleur. La batterie est utile quand le soleil est moins important (les jours de pluie ou la nuit). Pour avoir en temps réel la tension de la batterie, nous avons choisi de l'envoyer comme pour la valeur des capteurs pour pouvoir suivre cette tension depuis le site du projet.

Le choix de changer de composant de traitement pour passer d'un Raspberry pi à un ESP32 a été réalisé car un Raspberry pi consomme bien trop d'énergie pour pouvoir l'alimenter avec une petite batterie et un seul panneau solaire. Il aurait fallu plusieurs panneaux et plusieurs batteries. Le changement pour un ESP32 a nécessité un recodage complet du système, En effet le Raspberry pi n'utilise pas le même langage que l'ESP32. Les

avantages de ce composant sont qu'il est peu cher, qu'il est équipé d'une puce WIFI, qu'il est programmable avec l'IDE Arduino et compatible avec la majorité des librairies et enfin qu'il est petit.

Ce projet est loin d'être terminé et les possibilités d'amélioration sont importantes, il peut être agrémenté d'un système autonome d'envoi de données (GPRS, GSM, …). Le code peut être optimisé pour consommer encore moins d'électricité. Il reste donc une multitude de possibilités pour améliorer le projet au niveau hardware comme software.

Durant ce projet j'ai acquis des compétences dans la modélisation 3D ainsi que dans la compréhension d'un système autonome énergétiquement et dans la communication serveur client.

6. Table des figures

7. Sources

- https://learn.adafruit.com/adafruit-fona-mini-gsm-gprs-cellular-phone-module/assembly
- https://www.robotshop.com/media/files/content/w/wav/pdf/4g-3g2ggsmgprs-gnss-hat-raspberry-pi-datasheet.pdf
- https://randomnerdtutorials.com/bme280-sensor-arduino-pressure-temperature-humidity/
- https://randomnerdtutorials.com/esp8266-bme280-arduino-ide/
- https://github.com/MHEtLive/arduino-esp32
- https://randomnerdtutorials.com/esp32-ds18b20-temperature-arduino-ide/
- https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf
- http://emery.claude.free.fr/esp32-wifi.html
- http://electroniqueamateur.blogspot.com/2020/05/envoi-de-donnees-de-lesp32-esp8266-vers.html

- https://www.az-delivery.de/fr/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/esp32-wifi-sensor-projekt
- http://www.boichat.ch/joomla/index.php?option=com_content&view=article&id=132:un-tutoriel-sur-l-esp32-un-thermometre-digital&catid=92&Itemid=503
- https://www.thingsmobile.com/fr/business
- https://www.watteo.fr/connaitre-lautonmie-dune-batterie-decharge-profonde/
- https://letmeknow.fr/shop/fr/blog/142-tutoriel-les-sleep-modes-de-lesp32
- https://randomnerdtutorials.com/esp32-web-server-arduino-ide/
- https://rntlab.com/question/import-serial-error-with-esp32-on-doit-esp32-revkit-vi-board/
- https://randomnerdtutorials.com/getting-started-with-esp32/
- https://www.teachmemicro.com/send-data-sim800-gprs-thingspeak/
- https://how2electronics.com/send-gsm-sim800-900-gprs-data-thingspeak-arduino/
- https://how2electronics.com/
- https://randomnerdtutorials.com/esp32-http-get-post-arduino/
- https://www.insunwetrust.solar/blog/le-solaire-et-vous/orientation-inclinaison-photovoltaique/
- https://www.hindawi.com/journals/jre/2020/8893891/
- https://news.dualsun.com/co-en/12/2014/what-is-the-optimal-orientation-and-tilt-angle-for-solar-panels/#:~:text=In%20this%20case%2C%20for%20the,low%20tilt%20at%2020%C2%B0!

8. Annexes

8.1 Guide de montage

Liste du matériel :

Composants :

- 1 : SI1145

Figure 8. Image d'un SI1145

- 1 : BME680

Figure 9. Image d'un BME 680

- 1 : DS18B20

Figure 10. Image d'un DS18B20

- 1 : ESP32-DEVKITC-32UE

Figure 11. Image d'un ESP32 Devkitc 32UE

- 1 : Panneau solaire Adafruit 200

Figure 12. Image d'un panneau solaire Adafruit 200

- 1 : Adafruit MCP73871 (interface et chargeur)

Figure 13. Image d'un Adafruit MPC73871

- 1 : Adafruit TPS61030-32 (USB Boost)

Figure 14. Image d'un Adafruit TPS61030-32

- 1 : Support de batterie

Figure 15. Image d'un support de batterie

- 1 : batterie

Figure 16. Image d'une batterie LIPO de 2600 mAh

- 1 : résistance 4.7 KΩ
- 1 : résistance 1000 Ω
- 1 : résistance 120 Ω
- Du fil de différentes couleur en multibrin

Outils :

- 1 : Fer à souder
- 1 : pince à dénuder
- 1 : pince coupante

Hardware :

Étape 1 : Préparation du panneau solaire

Coupez le connecteur du panneau et soudez un fil rouge sur le fil rouge initial et soudez un fil noir sur le fil noir initial. Installez un bouton sur le fil rouge du panneau solaire (figure 17).

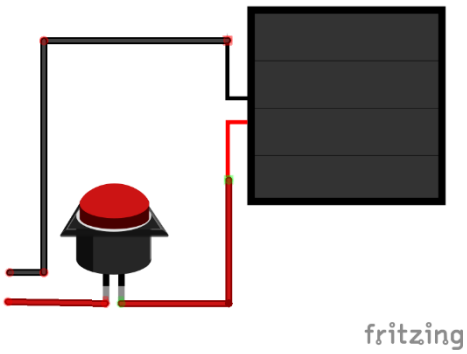

Figure 17. Schéma du câblage d'un bouton sur le panneau solaire

Étape 2 : préparation de logement de la batterie

Soudez dans un premier temps des fils rouges et noir sur le support de la batterie. ATTENTION : Il ne faut pas souder les fils sur la batterie mais bien sur le support. Soudez une patte de la résistance de 120 Ω sur le fil rouge (+3.7V), puis soudez une patte de la résistance de 1000 Ω sur le fil noire (masse). Soudez les deux pattes libres des résistances ensemble et soudez un fil à la jonction. Enfin soudez un bouton après les résistances sur le fil rouge (figure 18).

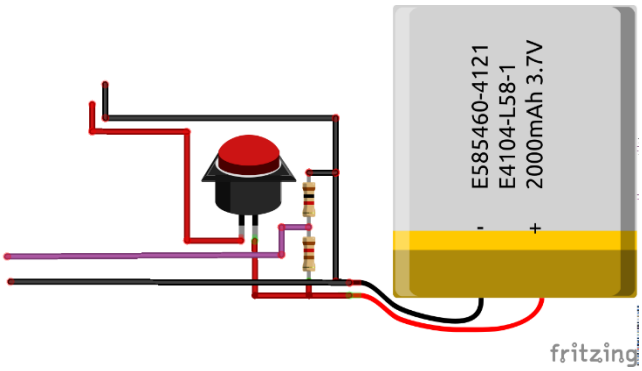

Figure 18. Schéma du câblage d'un bouton et d'un pont diviseur de tension sur le support de la batterie

Étape 3 : Préparation de l'USB Booster

Soudez un fil rouge sur la borne "BAT" de l'USB Booster et un fil noir sur la borne "GND" du module (figure 19).

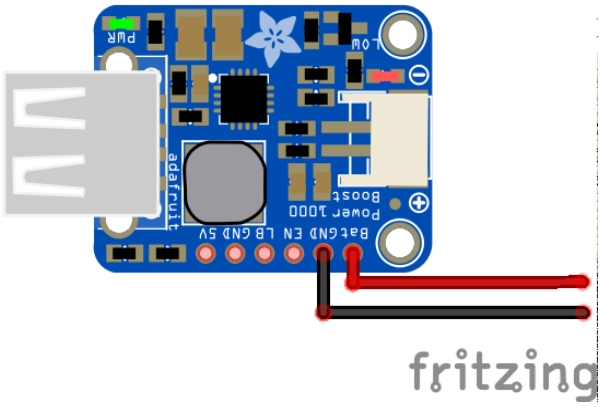

Figure 19. Schéma du câblage de deux fils sur l'USB Booster

Étape 4 : Assemblage du module Solaire

Soudez le fil rouge de l'USB Booster à la borne load "L+" du MCP73871 et le fil noir à la borne Load "GND" du MCP7387. Soudez le fil rouge en sortie du bouton de la batterie à la borne BATT "B+" du MCP7387 et le fil noir à la borne BATT "GND" du MCP7387. Soudez le fil rouge en sortie du bouton du panneau solaire à la borne DCIN " + " du MCP7387 et le fil noir à la borne DCIN " - " du MCP7387 (figure 20).

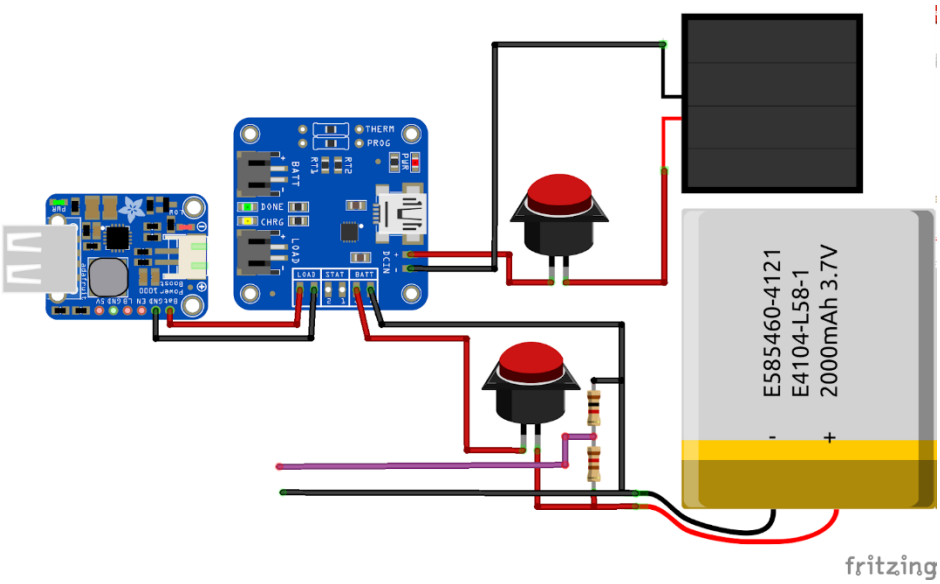

Figure 20. Schéma de câblage de l'USB Booster, de la batterie et du panneau solaire sur le MCP73871

Étape 5 : Câblage du DS18B20

Soudez deux fils noirs au pin “GND” du DS18B20, un petit fil rouge au pin “+5V” du capteur et un petit fil d’une autre couleur au pin “data” du capteur. Soudez une résistance de 4.7 KΩ entre le petit fil rouge et le petit fil des datas. Soudez deux longs fils rouges sur le pin de la résistance où est soudé le petit fil rouge et un long fil sur l'autre pin de la résistance (figure 21).

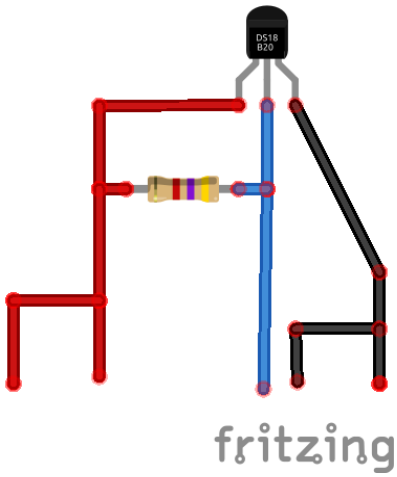

Figure 21. Schéma de câblage du DS18B20

Étape 6 : Câblage du BME680

Soudez deux fils rouges sur la borne “VIN”, deux fils noirs sur la borne “GND”, deux fils jaunes sur la borne “SCK” et deux fils verts sur la borne “SDI” (figure 22).

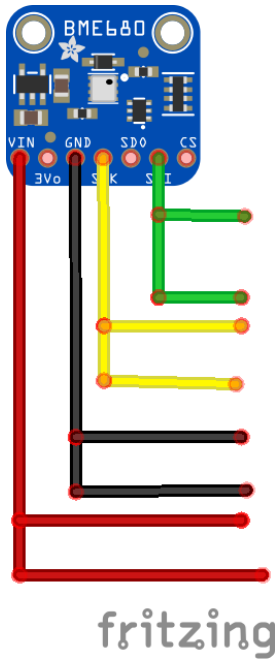

Figure 22. Schéma de câblage des fils sur le BME680

Étape 7 : Câblage du SI1145

Soudez un fil rouge sur la borne “VIN”, un fil noir sur la borne “GND”, un fil jaune sur la bornes “SCL” et un fil verts sur la borne “SDA” (figure 23).

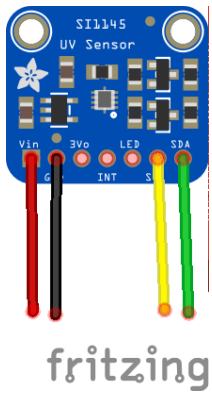

Figure 23. Schéma de câblage des fils sur le SI1145

Étape 7 : Assemblage des capteurs

Soudez un des fils verts du BME680 au fil vert du SI1145. Soudez un des fils jaunes du BME680 au fil jaune du SI1145. Soudez un des fils rouges du BME680 à un des fils rouges du DS18B20, puis soudez l'autre fil rouge du DS18B20 au fil rouge du SI1145. Soudez un des fils noirs du BME680 à un des fils noirs du DS18B20, puis soudez l'autre fil noire du DS18B20 au fil noir du SI1145 (figure 24).

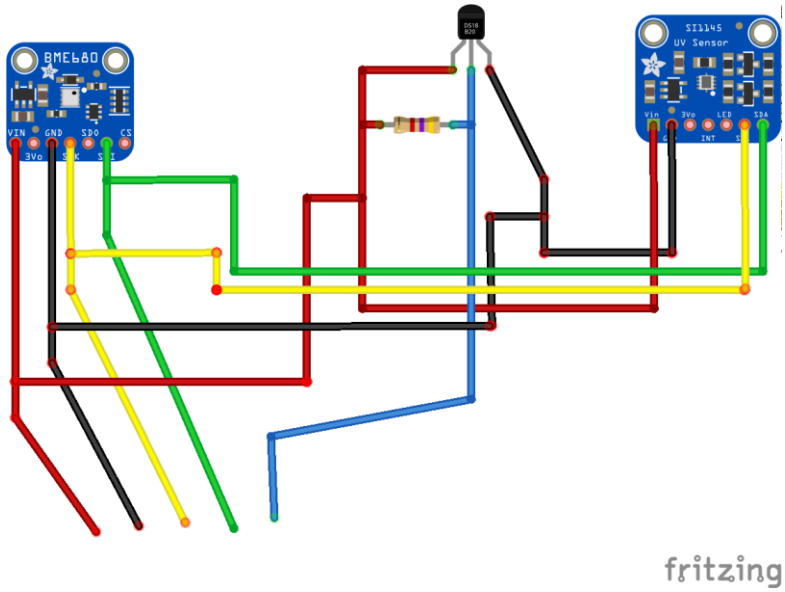

Figure 24. Schéma de câblage des capteurs entre eux

Étape 7 : assemblage à l'ESP32

Soudez le fil rouge à la broche "+5V" de l'ESP32, les fils noirs aux broches "GND" de l'ESP32, le fil jaune à la broche "GPIO 22" de l'ESP32, le fil vert à la broche "GPIO 21" de l'ESP32, le fil des datas du DS18B20 à la broche "GPIO16" de l'ESP32, le fil sortant des résistances de la batterie à la broche "GPIO 33" et enfin connectez le câble USB entre l'USB booster et l'ESP32 (figure 25).

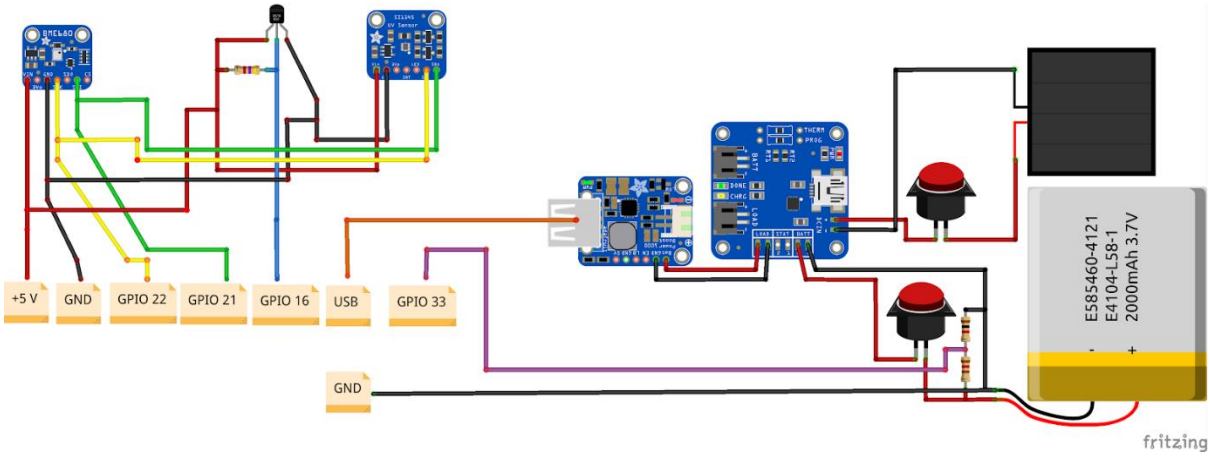

Figure 25. Schéma de câblage global du système

Vérifications :

Pour vérifier si le système fonctionne, allumez tous les interrupteurs et observez les led.

Si la led “PWR” du MCP7387 s’allume ou clignote c’est que le panneau fournit de l’énergie. Si elle ne clignote pas, vérifiez que le panneau est au soleil ou sous une source de lumière très forte, que le bouton est bien sur “on” et que les soudures sont correctes. Si après ces vérifications la led ne s’allume toujours pas, dessoudez les fils du panneau et branchez le MCP7387 sur secteur. Si la led s’allume c’est que le problème vient du panneau et/ou du bouton. Si elle ne s’allume pas, c’est qu’il y a un problème sur le MCP7387.

Si la led “PWR” de l’USB Boost s’allume alors que le bouton du panneau est sur “OFF”, c’est que le câblage avec la batterie est bon, si elle ne s’allume pas, vérifiez que la batterie est chargée et que les soudures sont correctes.

Position des éléments dans le boîtier :

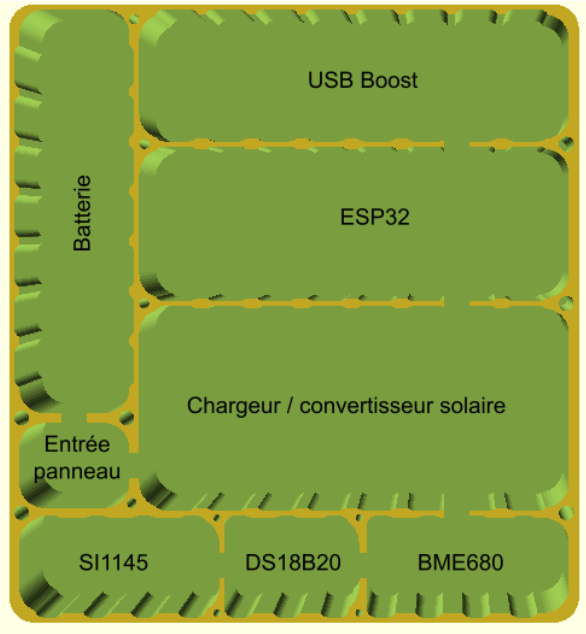

Figure 26. Disposition des éléments dans le boîtier

Software :

Une fois l'électronique fonctionnelle, il ne reste plus qu'à téléverser le code dans l'ESP32. Pour cela, téléchargez l'application Arduino sur votre ordinateur et collez le code.

Pour téléverser le code, vous devrez installer les librairies suivantes :

- OneWire by JIM Studt et al. Version 2.3.5
- DallasTempérature by Miles Burton et al. Version 3.9.0
- Adafruit_SI1145 by Adafruit Version 1.1.1
- Adafruit_BME680 by Adafruit Version 2.0.0

Enfin, pour que le système fonctionne, modifiez l'URL du serveur, la clé d'entrée, les informations du wifi et l'ID des capteurs. Les paramètres à modifier sont dans la partie "settings" (figure 26).

```
/**********   Settings   **********/

#define ONE_WIRE_BUS 16                             //Pin for DS18B20
#define TEMPERATURE_PRECISION 12                    //Precision for DS18B20 in bits
#define SEALEVELPRESSURE_HPA (1013.25)              //Pressure at sea level
#define TIME_TO_SLEEP  1                            //Duration of sleep mode in secondes
#define BASE_URL "https://XXXX/" //Server URL
#define KEY "KEY"                  //API KEY for the server
#define BME680_T_ID "1"                             //ID of the corresponding sensor on the server
#define BME680_H_ID "2"                             //ID of the corresponding sensor on the server
#define BME680_P_ID "3"                             //ID of the corresponding sensor on the server
#define SI1145_VI_ID "4"                            //ID of the corresponding sensor on the server
#define SI1145_IR_ID "5"                            //ID of the corresponding sensor on the server
#define SI1145_UV_ID "6"                            //ID of the corresponding sensor on the server
#define DS18B20_T_ID "7"                            //ID of the corresponding sensor on the server


const char* ssid = "Your_SSID";                      //SSID WIFI
const char* password = "Your_Password";              //Password WIFI
```

Figure 27. Partie settings du code

8.2. Code de l'ESP32

```
/*  ------------------------------
 *  Arduino code for ESP32:
 *   - powered by solar panel
 *   - get data from BME680, SI1145, DS18B20
 *   - send data to a server API using wifi
 *
 *  Lucas Bessière and François Rebaudo
 *  (c) IRD UMR EGCE, Lic. GLP v3
 *
 *  last modif 2021-06-03
 *  ------------------------------
 *
 *  ------------------------------
 *  WIRING
 *  ------------------------------
 *
 *  BME680
 *  ---------------
 *  - VIN to 5V
 *  - GND to GND
 *  - SCK to 22
 *  - SD1 to 21
 *
 *  SI1145
 *  ---------------
 *  - VIN to 5V
 *  - GND to GND
 *  - SCL to 22
 *  - SDA to 21
 *
 *  DS18B20
 *  ---------------
 *  - VIN to 5V
 *  - GND to GND
 *  - data to 16
 *
 *  solar panel, power boost, MCP73871 USB/solar, ...
 *  ---------------
```

```
 *  to do
 *
 *
 *  ------------------------------
 *  INSTALLATION INSTRUCTIONS
 *  ------------------------------
 *
 *  File > Preferences > Additional Boards Manager URLs >
 *  https://dl.espressif.com/dl/package_esp32_index.json
 *
 *  Tools > Board > ESP32 Arduino > DOIT ESP32 DEVKIT V1
 *
 *  Tools > Manage libraries... >
 *  - OneWire
 *  - DallasTemperature
 *  - Wire
 *  - SPI
 *  - Adafruit_SI1145
 *  - Adafruit_Sensor
 *  - Adafruit_BME680
 */

/***     Libraries     ***/
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiMulti.h>
#include <HTTPClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_SI1145.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME680.h>

/**********   Settings   **********/
#define TIME_TO_SLEEP  600             //Duration of sleep mode in secondes

#define ONE_WIRE_BUS 16                //Pin for DS18B20
#define TEMPERATURE_PRECISION 12       //Precision for DS18B20 in bits
#define SEALEVELPRESSURE_HPA (1013.25) //Pressure at sea level
#define BASE_URL "https://XX"          //Server URL
#define KEY "Bearer xxxxxxx"
#define BME680_T_ID "114"              //ID of the corresponding sensor on the
server
#define BME680_H_ID "115"          //ID of the corresponding sensor on the server
#define BME680_P_ID "142"          //ID of the corresponding sensor on the server
#define SI1145_VI_ID "143"         //ID of the corresponding sensor on the server
#define SI1145_IR_ID "144"         //ID of the corresponding sensor on the server
#define SI1145_UV_ID "145"         //ID of the corresponding sensor on the server
#define DS18B20_T_ID "113"         //ID of the corresponding sensor on the server
#define TENSION_V_ID "146"         //ID of the corresponding sensor on the server

const char* ssid = "xxxxxxx";
const char* password = "xxxxxxx";

// battery voltage monitoring
const float TMaxA0 = 3.3;     // Tension max Arduino A1 admissible
int analogInput = 33;  //Entrée analogique
float R1 = 121.0;    // Résistance R1
float R2 = 989.0;    //Résistance R2
int valeurTension = 0;

WiFiMulti wifiMulti;

/**********   END Settings   **********/
```

```
#define uS_TO_S_FACTOR 1000000  /* Conversion factor for micro seconds to seconds
*/
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
Adafruit_SI1145 uv = Adafruit_SI1145();
Adafruit_BME680 bme;

void setup() {
  Serial.begin(115200);
  Serial.println("Initializing...");
  //  pinMode(Led1, OUTPUT);
  //  pinMode(Led2, OUTPUT);
  //  digitalWrite(Led1, LOW);
  //  digitalWrite(Led2, LOW);
  /***     battery voltage monitoring     ***/
  pinMode(analogInput, INPUT);
  /***     start sensors :     ***/
  sensors.begin();
  bme.setTemperatureOversampling(BME680_OS_8X);
  bme.setHumidityOversampling(BME680_OS_2X);
  bme.setPressureOversampling(BME680_OS_4X);
  bme.setIIRFilterSize(BME680_FILTER_SIZE_3);
  bme.setGasHeater(320, 150); // 320*C for 150 ms
  if (! uv.begin()) {
    Serial.println("SI1145 problem");
    while (1);
  }
  if (! bme.begin()) {
    Serial.println("BME680 problem");
    while (1);
  }
  /***      http and wifi :     ***/
  wifiMulti.addAP(ssid, password);
  Serial.println("Connecting");
  while(wifiMulti.run() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  /***     battery voltage monitoring     ***/
  valeurTension = analogRead(analogInput);
  float Tension = ((valeurTension * ( TMaxA0 / 4095)) / (R2/(R1+R2)));
  Serial.print(Tension);
  Serial.println(" V");
  delay(1000);
  /***     sensors :     ***/
  sensors.requestTemperatures();
  delay(1000); // delay for correct measurement of DS18B20 ???
  float temp_Capt_DS18B20_I = sensors.getTempCByIndex(0);
  Serial.print("DS18B20: ");
  Serial.println(temp_Capt_DS18B20_I);
  float temp_Capt_DS18B20;
  if((temp_Capt_DS18B20_I>=-40) && (temp_Capt_DS18B20_I<=70)){
    temp_Capt_DS18B20 = temp_Capt_DS18B20_I;}
  else{temp_Capt_DS18B20 = 85;}
  float UVindex = uv.readUV();
  UVindex /= 100.0;
  float DS18B20_T = temp_Capt_DS18B20;
  float SI1145_VIS = uv.readVisible();
```

```
  float SI1145_UV = UVindex;
  float SI1145_IR = uv.readIR();
  if(! bme.performReading()){
    Serial.println("Failed to perform reading :(");
    return;
  }
  float BME680_T = bme.temperature;
  float BME680_P = bme.pressure / 100.0;
  float BME680_H = bme.humidity;

  /***     http :      ***/
    //Check WiFi connection status
  if(wifiMulti.run()== WL_CONNECTED){
    sendData(BME680_T_ID, BME680_T);
    sendData(BME680_H_ID, BME680_H);
    sendData(BME680_P_ID, BME680_P);
    sendData(SI1145_VI_ID, SI1145_VIS);
    sendData(SI1145_IR_ID, SI1145_IR);
    sendData(SI1145_UV_ID, SI1145_UV);
    if(DS18B20_T != 85){
      sendData(DS18B20_T_ID, DS18B20_T);
    }
    sendData(TENSION_V_ID, Tension);
    Serial.println("...");
  }
  else {
    Serial.println("WiFi Disconnected");
    ESP.restart();          // restart in case of wifi deconnexion
  }
  delay(300);
  esp_light_sleep_start();
}

void sendData (String sensorId, float valueSensor){
  HTTPClient http;

  String buf = BASE_URL + sensorId + "/datas";
  http.begin(buf);
  Serial.println(buf);
  http.addHeader("accept", "application/json");
  http.addHeader("Authorization", KEY);
  http.addHeader("Content-Type", "application/json");

  String bufJSON = "{\"value\":" + String(valueSensor) + "}";
  Serial.println(bufJSON);
  int rStatusCode = http.POST(bufJSON);
  Serial.print(valueSensor);
  Serial.print(" ; ");
  Serial.println(rStatusCode);
  Serial.println(http.getString());
  http.end();
}
```