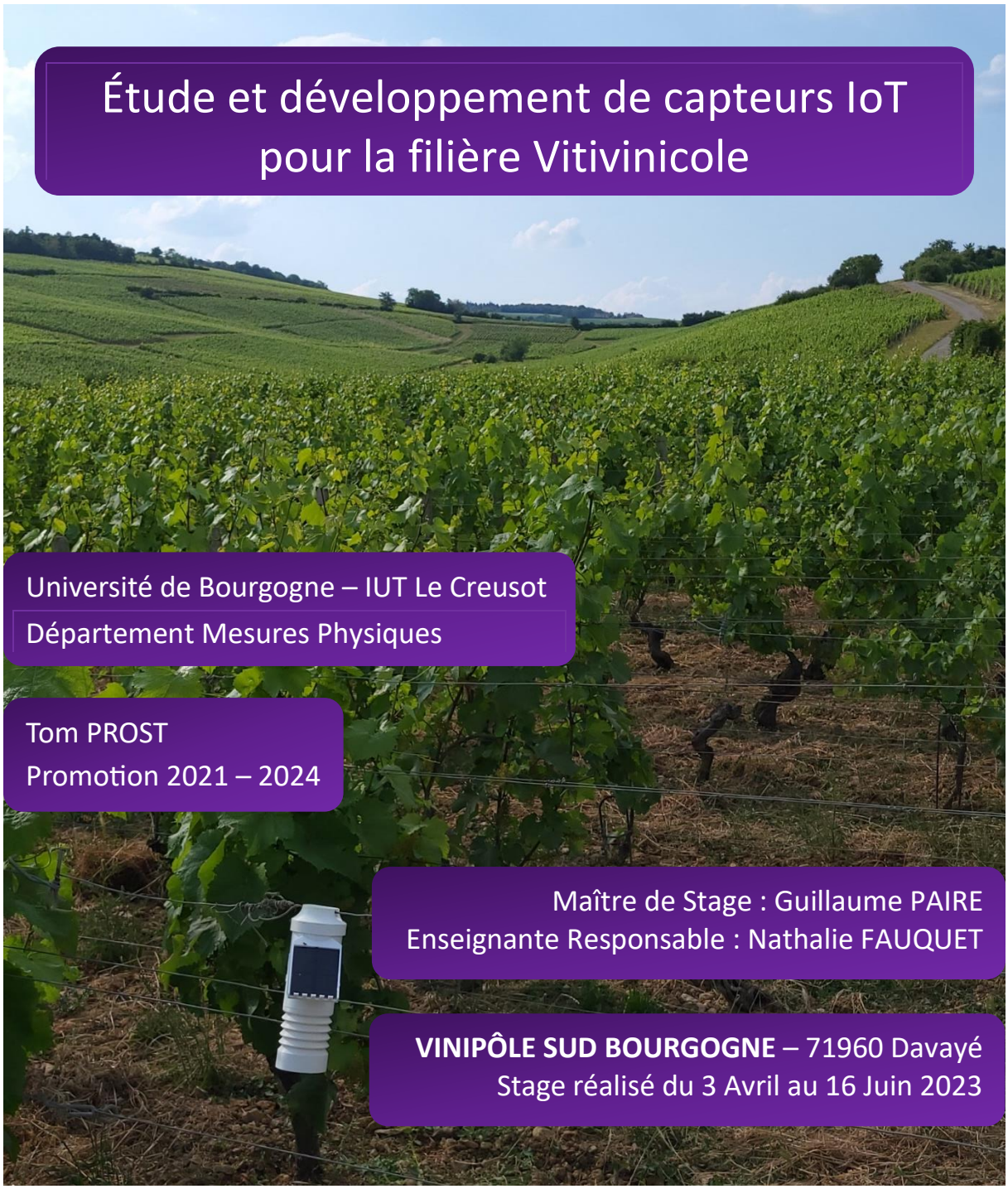
Étude et développement de capteurs IoT pour la filière Vitivinicole



Université de Bourgogne – IUT Le Creusot
Département Mesures Physiques

Tom PROST
Promotion 2021 – 2024

Maître de Stage : Guillaume PAIRE
Enseignante Responsable : Nathalie FAUQUET

VINIPÔLE SUD BOURGOGNE – 71960 Davayé

Stage réalisé du 3 Avril au 16 Juin 2023

Remerciements

Je tiens à remercier toutes les personnes qui ont contribuées à la réflexion autour de ce stage et qui m'ont aidé pour toutes les étapes de ce projet jusqu'à la rédaction de ce rapport.

Aussi, l'Université de Bourgogne, l'IUT du Creusot ainsi que le département Mesures Physiques pour m'avoir aidé dans la recherche et l'organisation du stage et bien entendu le Vinipôle Sud Bourgogne de m'avoir permis de réaliser cette période en entreprise.

Tout d'abord, j'adresse mes remerciements à mon maître de stage, **Mr Guillaume PAIRE**, Responsable du VITILAB, pour l'aide qu'il m'a apporté par ses connaissances, sa disponibilité ainsi que pour les échanges constructifs qui ont été d'une aide précieuse tout au long de ce stage.

Je remercie également **Mr Benjamin ALBAN**, Directeur du Vinipôle Sud Bourgogne, qui a cru aux finalités qui débouchaient de ce sujet de stage et qui a accepté ma candidature.

Je remercie par ailleurs toute l'équipe du Vinipôle Sud Bourgogne pour leur accueil et leur bonne humeur au sein de leur équipe durant ces 11 semaines, et particulièrement **Mme Olivia TROLY**, secrétaire du service, et **Mr Pierrick SAILLANT**, développeur, pour m'avoir aidé à rédiger du code informatique.

Pour finir, je remercie **Mme Laurence CORON**, secrétaire du Département Mesures Physiques, **Mme Nathalie Fauquet**, mon enseignante responsable et **Mme Catherine VIGIANNO**, enseignante qui m'a aidé à trouver mon stage, ainsi que tout le corps enseignant du Département Mesures Physiques de l'IUT du Creusot pour tout ce qu'ils m'ont apportés durant ces deux années comme connaissances techniques et savoir-vivre qui me sont indispensables aujourd'hui en entreprise.

Sommaire

I. Contexte du stage

I.1 Objectifs personnels

L'IUT nous fournit une liste des entreprises ayant déjà accueilli un stagiaire en Mesures Physiques (MP), soit plus de 360 contacts dans tous les domaines de l'industrie.

J'ai choisi ce stage car un sujet plutôt orienté en électronique, informatique était la suite logique de l'option Techniques d'Instrumentation que j'avais choisi pour ma deuxième année de BUT MP. Le milieu viticole me permettait d'appliquer les connaissances acquises durant mes deux années d'IUT à un secteur proche de mes racines à savoir les gens qui travaillent la Terre.

I.2 Présentation de l'entreprise

I.2.1 Vinipôle Sud Bourgogne

Le Vinipôle Sud Bourgogne est une association basée sur Davayé (71960) constituée de :

- 3 membres fondateurs : le Conseil Général de Saône-et-Loire, la Chambre d'Agriculture de Saône-et-Loire et le Bureau Interprofessionnel des Vins de Bourgogne (BIVB).
- 3 membres associés : Le Lycée viticole "Lucie Aubrac" et le CFPPA de Davayé, Le Centre Œnologique de Bourgogne et L'institut Français de la vigne et du Vin.

Ce lieu (*Figure 1*) ouvert aux professionnels réunit sur le site du Lycée Viticole de Davayé : conseils, analyses œnologiques, formations et expérimentations. Cet ensemble de compétences est au service de la filière et l'accompagne dans la mise en place d'une vitiviniculture durable en Bourgogne du Sud. (1. *Vinipôle Sud Bourgogne | Pôle de compétences pour la viticulture*.)

Figure 1 : Vue aérienne du VitiLab et du Vinipôle

Les missions du Vinipôle Sud Bourgogne sont de répondre aux enjeux de demain et parmi son programme de recherche, 5 grandes thématiques sont développées :

- Changement climatique : atténuation (bilan carbone), adaptation (maintien d'une production en quantité et qualité) et gestion des accidents climatiques (gel, grêle, canicule…).
- Matériel végétal : conserver, sélectionner, évaluer et produire des vignes adaptées aux exigences sanitaires, environnementales et qualitatives des prochaines décennies (porte-greffe, variétés, cépages…).
- Agriculture de précision : évaluation, utilisation et réglages de nouveaux outils (épandeurs engrais, pulvérisateurs, Outils d'Aide à la Décision…) et de nouvelles méthodes culturales (densité, mode de conduite, intrants…).
- Agroécologie : évaluation et mise en place d'infrastructures agroécologiques (arbres, haies…) pour favoriser la biodiversité et encourager les démarches de certification environnementales (Agriculture Biologique…).
- Numérique et Robotique :

Dans le cadre du développement de cette dernière thématique, a vu le jour en 2021 le VITILAB, pôle de Recherche & Développement (R&D) du Vinipôle Sud Bourgogne.

I.2.2 VITILAB

Le VITILAB c'est :

- Un centre de ressources : il a pour vocation de diffuser les usages et techniques numériques en viticulture qui répondent aux enjeux contemporains du terrain.
- Un lieu d'expérimentations : il évalue et teste en grandeur nature des services, des outils ou des usages nouveaux destinés à la filière vitivinicole du type exosquelette, applications mobiles, robotique…
- Un espace de coworking : travailler dans un cadre stimulant avec bornes interactives, réalité virtuelle et un FabLab à disposition avec imprimante 3D, découpe laser, scanner 3D et fraiseuse numérique.

C'est dans ce contexte de l'expansion de la thématique nouvelles technologies que mon stage intervient.

I.3 Objet de l'étude

Les enquêtes de terrain montrent que les exploitants actuels et la nouvelle génération souhaitent avoir à leur disposition des données plus précises afin de se rapprocher du pilotage en temps réel, dans cette optique, les Technologies de l'Information et de la Communication (TIC) trouvent de plus en plus leurs places.

L'agriculture et plus précisément ici la filière vitivinicole cherche donc à récupérer des données, pour mener des expérimentations et piloter les cultures. Aujourd'hui, on a des capteurs vendus en solutions commerciales ou bien des capteurs où il faut récupérer les données sur place. Dans le premier cas, ces méthodes sont onéreuses et la plupart du temps,

nous sommes non-propriétaires de données qui sont pourtant censés nous appartenir. Et dans le deuxième cas, cela va à l'encontre de la philosophie du capteur connecté qui devrait être là pour simplifier la vie de l'utilisateur, donc si l'exploitant perd du temps à relever ses données, la méthode sera abandonnée.

Il faudra donc trouver des solutions pour remédier à ces problèmes.

I.3.1 Problématique

C'est donc la création de solutions peu coûteuses pour l'utilisateur et autonome en énergie afin de récupérer des données en direct gratuitement qui a fait l'objet de mon stage au Vinipôle.

Mes missions seront de développer une solution pour chacune des requêtes suivantes :

- Développer un modèle de station météo existante.
- Monitorer le niveau d'eau dans une cuve enterrée.
- Monitorer en direct la consommation électrique de chaque appareil dans un chai.

Chacun de ces projets a été émis par une entité différente.

I.3.2 Cahier des charges

Les objectifs à atteindre sont de démontrer la faisabilité d'objets connectés, de la preuve de concept à la réalisation des prototypes. Ces objets connectés devront pouvoir être réalisés en DIY (à faire soi-même), communiquer sur un réseau libre d'accès et de droit tout en gardant les données confidentielles, de rapatrier les données et de pouvoir les traiter avec la base de code informatique existante au sein du service ou non si une meilleure solution était trouvée. Une condition essentielle est que cette approche doit se faire sans travail supplémentaire pour le producteur.

Les enjeux de la mission pour le Vinipôle sont que ces réalisations soient faciles à prendre en main afin d'être vulgarisées au plus grand nombre pour rendre accessible à tous ces nouveaux moyens de travailler.

Le principe du VITILAB n'est pas de créer des besoins futiles mais de répondre à une demande des producteurs sur des problématiques de terrain afin de solutionner des besoins identifiés. Le VITILAB est donc un pôle de ressources où les personnes souhaitant évoluer avec les nouvelles technologies peuvent se rendre. C'est aussi un lieu où l'on peut faire fabriquer des pièces, ou bien venir avec des commandes de solutions afin qu'elles soient réalisées sur mesure.

Le précepte du VITILAB est de ne pas réinventer ce qui existe déjà mais de mener une recherche bibliographique pour comparer l'existant et choisir la technologie la plus adaptée au cas concret qu'il faut résoudre.

II. Planification du travail

J'ai réalisé un diagramme de GANTT prévisionnel (*Figure 2*) au début du projet, puis je l'ai actualisé au fur et à mesure afin d'obtenir le diagramme de GANTT réel (*Figure 2*). En effet, il est nécessaire de structurer l'avancée du projet afin de voir si je prenais du retard. La date de fin du stage étant fixée au 16 juin, j'ai donc organisé mon projet de façon à finir le plus de tâches possibles sur les trois capteurs dans les onze semaines qui m'étaient imparties.

Le stage s'est scindé en deux parties avec la première partie théorique : études bibliographiques sur l'actualité des capteurs connectés, les différentes méthodes utilisées, les logiciels de développement…

La deuxième a été consacrée à la sélection des technologies les mieux adaptées aux usages qui étaient définis, dans le but de réaliser des prototypes fonctionnels pour les tester en grandeur nature chez les commanditaires.

Figure 2 : Diagrammes de GANTT prévisionnel et réel

À voir sur *Figure 2*, la programmation et le câblage ont duré toute la fin de mon stage après la recherche bibliographique et l'attente des composants. Car le but étant de livrer un produit le plus facile d'utilisation, le plus complet et fiable possible, dès que je faisais un test et que je constatais une amélioration possible je faisais un nouveau prototype. J'ai cependant pu mener à bien et réaliser la majeure partie des missions qui m'étaient incombées malgré les aléas du développement de prototypes.

III. État de la technique

III.1 IoT

III.1.1 Généralités

III.1.1.1 *Définition*

Depuis quelques années, un nouveau domaine s'ouvre à nous : l'IoT. Il existe plusieurs définitions de l'IoT (Internet of Things), peut-être devrait-on plutôt dire l'Internet des Choses (objets physiques, matériels, logiciels, objets virtuels, services, personnes, animaux, etc.). Les différentes définitions font penser qu'il s'agit finalement d'un Internet of Everything. À terme, il s'agira tout simplement « d'Internet » dans lequel tout sera hyperconnecté. La définition IERC (Cluster des projets de recherche IoT Européens) de 2010 était que : « l'Internet des Objets fait partie de l'Internet du futur et consiste en une infrastructure globale dynamique ayant des identifiants, des attributs physiques et des entités virtuelles utilisant des interfaces intelligentes, qui sont parties intégrantes du système d'information ». L'IERC donne maintenant une définition qui s'approche de l'Internet of Everything : « L'Internet des Objets est une infrastructure de réseaux mondiale dynamique avec des capacités d'autoconfiguration basées sur des protocoles de communications standards et interopérables où les « choses » physiques et virtuelles ont des identités, des attributs physiques et des personnalités virtuelles utilisant des interfaces intelligentes et sont intégrées de manière transparente au réseau d'information ». (2. Guillemin, P. (2018).)

III.1.1.2 *En* viticulture

Dans le cadre de l'IOT en viticulture, divers objets sont connectés au réseau Internet, pouvant inclure des capteurs de sol, des stations météorologiques, des drones, des équipements de pulvérisation automatisés, des systèmes d'irrigation intelligents... Les capteurs de sol sont utilisés pour mesurer les caractéristiques du sol telles que l'humidité, la température et la composition chimique, ce qui permet aux viticulteurs de mieux comprendre les besoins en eau et en nutriments des vignes. Les stations météorologiques fournissent des données en temps réel sur les conditions météorologiques, telles que la température, l'humidité, la vitesse du vent et l'ensoleillement, ce qui aide les viticulteurs à prendre des décisions concernant les opérations agricoles. Les drones et les caméras de surveillance sont utilisés pour surveiller l'état des vignes, détecter les maladies, les ravageurs ou les mauvaises herbes, et évaluer la maturité des raisins grâce à des algorithmes de reconnaissance entrainés par l'IA (Intelligence Artificielle). Ces informations permettent aux viticulteurs d'intervenir rapidement et de prendre des mesures préventives pour maintenir la santé des vignes et améliorer la qualité des récoltes.

En résumé, l'IOT en viticulture permet aux viticulteurs d'obtenir des données en temps réel sur les conditions du vignoble, d'automatiser certains processus agricoles, d'améliorer la précision des décisions prises et d'optimiser les rendements et la qualité des raisins. Cela contribue à une gestion plus efficace et durable des vignobles.

L'objectif ici n'est pas que les capteurs communiquent entre eux mais juste d'avoir un réseau d'objets connectés à disposition pour que l'opérateur puisse faire des choix en fonction des données relevées.

III.1.2 Support matériel

Un système informatique embarqué est un ensemble de composants (cartes programmables, capteurs et actionneurs) intégrés à un objet (maison, avion, robot…). Il sert à piloter cet objet à distance ou de manière autonome. Lorsqu'un système informatique embarqué échange des informations avec un ordinateur, une tablette ou un smartphone, par le biais de protocoles de communication (Wifi, Bluetooth, réseau filaire…), on parle d'objet connecté. (3. JACQUOT, D)

Pour créer ses propres objets connectés, il faut donc :

III.1.2.1 *La carte programmable*

Une carte programmable (ou microcontrôleur) intègre un microprocesseur qui effectue tous les traitements et qui stocke le code du programme. Il en existe des dizaines mais la plus connu reste Arduino car tout est disponible en open source (accès libre à l'intérieur du programme et documentation gratuite complète de tous les composants), et depuis quelques années, un autre microcontrôleur prend de la place : l'ESP32. L'ESP32 est le microprocesseur et de nombreuses marques vendent des cartes programmables intégrant un ESP32. J'utiliserai maintenant l'analogie ESP32 pour désigner la carte programmable.

III.1.2.2 *Les capteurs*

Les capteurs sont des composants qui envoient des informations (entrées) au programme du système embarqué. Ils convertissent des grandeurs physiques ou mesures comme l'appui sur un bouton, la distance, la température, la luminosité, les mouvements, l'altitude… en une donnée exploitable dans un programme.

III.1.2.3 *Les actionneurs*

Les actionneurs sont des composants qui agissent sur un système pour en modifier son comportement (sorties). Les actionneurs transforment les informations reçues du programme pour activer un moteur, un buzzer, une LED (Diode Électro Luminescente)…

III.1.3 Éditeur de code

Pour commander le support matériel, il faut écrire du code informatique. Il existe divers éditeurs de code, le plus répandu chez les débutants étant Arduino IDE car la firme Arduino offre une suite complète comprenant les cartes programmables, le langage informatique et l'éditeur. Une alternative pour les projets plus poussés est VSCode (Visual Studio Code), propriété de Microsoft. Dans les deux cas le langage informatique est le C++, mais Arduino en possède une version simplifiée.

III.1.4 Technologies de communication

Pour communiquer, les objets peuvent s'appuyer sur des réseaux locaux (wifi, privé, Ethernet, Bluetooth…), on ne les détaillera pas ici car du fait de leurs faibles portées, ils ne sont pas intéressants pour des applications en extérieur, ou bien sur des réseaux publics opérés à grande échelle. Les réseaux opérés se divisent en deux catégories :

- Les réseaux LPWAN (réseau étendu à basse consommation) sur les bandes de fréquences ISM (non licenciées) basés essentiellement sur les technologies Sigfox et LoRaWAN (centrés sur une fréquence de 868Mhz en Europe).
- Les réseaux basés sur des bandes de fréquences licenciées (2/3/4/5G).

Les réseaux LPWAN basés sur les fréquences ISM permettent des consommations énergétiques très basses, offrant ainsi des autonomies de plusieurs années sur piles ou batteries, comparé aux réseaux de téléphonie mobile qui permettent des débits de données bien plus élevées mais avec des consommations énergétiques bien plus élevées.

Pour des raisons d'autonomie, la technologie cellulaire est abandonnée au profit des LPWAN. SigFox est délaissé pour laisser la place au LoRaWAN car la technologie LoRa est en open-source contrairement à Sigfox, ce qui ne pose pas de problème sur la pérennité de l'entreprise.

III.1.4.1 *LoRa*

LoRa est l'acronyme de Long Range (Grande portée). C'est une technologie radio (sans fil) dans laquelle un émetteur à très faible consommation envoie de petits paquets de données à faible vitesse (0.3kb/s à 5,5kb/s) à un récepteur situé à grande distance. Suivant la qualité de l'antenne et de l'environnement, la portée peut aller de 5 à 20 kms.

III.1.4.2 *LoRaWAN*

LoRaWAN est l'acronyme de long-range-wide-area network (réseau étendu à longue portée). C'est un protocole de communication radio fondé sur la technologie LoRa (Long Range). Dans le cadre de l'Internet des objets, il permet de structurer un réseau LPWAN (*Figure 3*), intégrant des équipements terminaux (End-device) à faible consommation électrique par l'intermédiaire de passerelles (Gateway) pour récupérer les données collectées sur des serveurs. (4. (2023). LoRaWAN. fr.wikipedia.org)

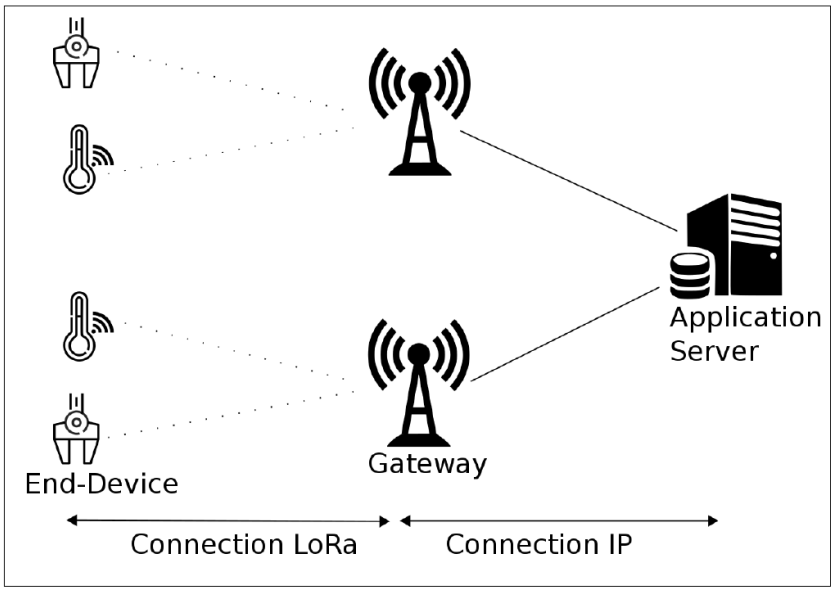


Figure 3 : Architecture LoRaWAN

Il y a deux y deux types de réseaux LoRaWAN : les nationaux et locaux. Dans le premier cas, celui-ci est géré par un opérateur auquel vous louez un abonnement pour faire transiter vos données sur leurs réseaux. Deuxièmement, réseau aussi dit privé, signifie que vous installez votre propre Gateway qui fera le lien entre vos appareils LoRa et Internet. (5. Mocq, F. (2022).)

En règle générale, les passerelles et les serveurs sont connectés via une liaison historique (Wi-Fi, Ethernet). Contrairement aux communications cellulaires dans lesquelles des périphériques mobiles sont associés aux relais transmetteurs, les nœuds LoRa ne sont pas associés à une passerelle spécifique. Toutes les données transmises par le nœud sont envoyées à toutes les passerelles et chaque passerelle qui reçoit un signal le transmet à un serveur relais du réseau. LoRa fonctionne sur des bandes de fréquences radio sans licence (ISM), donc l'utilisation d'une passerelle personnelle est parfaitement légale dans la plupart des pays. Il existe différentes fréquences ISM, selon l'emplacement géographique. Par exemple, aux états unis, la fréquence la plus usitée est la 915 Mégahertz (MHz), alors qu'en Europe, le standard est la 868 MHz. Il existe un cryptage de bout en bout AES-128, ce qui augmente la sécurité des données.
L'un des points forts de l'ESP32 choisi dans un projet antérieur du VitiLab est qu'il dispose d'une antenne LoRa, il peut ainsi être converti en Gateway (*Figure 4*). Les tests avaient montré qu'il valait mieux investir dans une vraie Gateway.

Figure 4 : Passerelle créée grâce à un ESP32

Le temps maximum d'occupation du canal radio (duty-cycle) imposé par l'utilisation de fréquences libres est un facteur limitant le nombre de paquets pouvant être émis par un équipement. Par exemple, le résultat de la limitation à 1 % sur la bande 868 MHz est un temps de transmission de 36 secondes par heure et par canal pour chaque terminal. Dans notre cas, on peut approximer que l'on ne peut pas envoyer de données plus rapidement que toutes les 10 minutes par appareil si l'on veut respecter la réglementation.

III.1.4.3 *TheThingsNetwork*

The Things Network (TTN) est un réseau basé sur la technologie LoRaWAN. Il s'agit d'un projet communautaire et open-source (*Figure 5*), autrement dit un projet de crowdsourcing mondial, ouvert, gratuit et décentralisé. C'est une communauté internationale qui regroupe plus de

170000 contributeurs dans environ 151 pays dans lesquels sont déployés plus de 20000 passerelles. Les volontaires prennent en charge la fourniture, la construction et la maintenance des passerelles LoRa. Ceux-ci transmettent les signaux radio longue portée des capteurs à un centre de contrôle via Internet. Là, les signaux sont traités et transmis à des destinataires définis. C'est ce site qui permet de faire le lien entre la passerelle qui reçoit les données des nœuds (Endnode) et qui envoie ces données via internet à une base de données. (6. The Things Network. .)

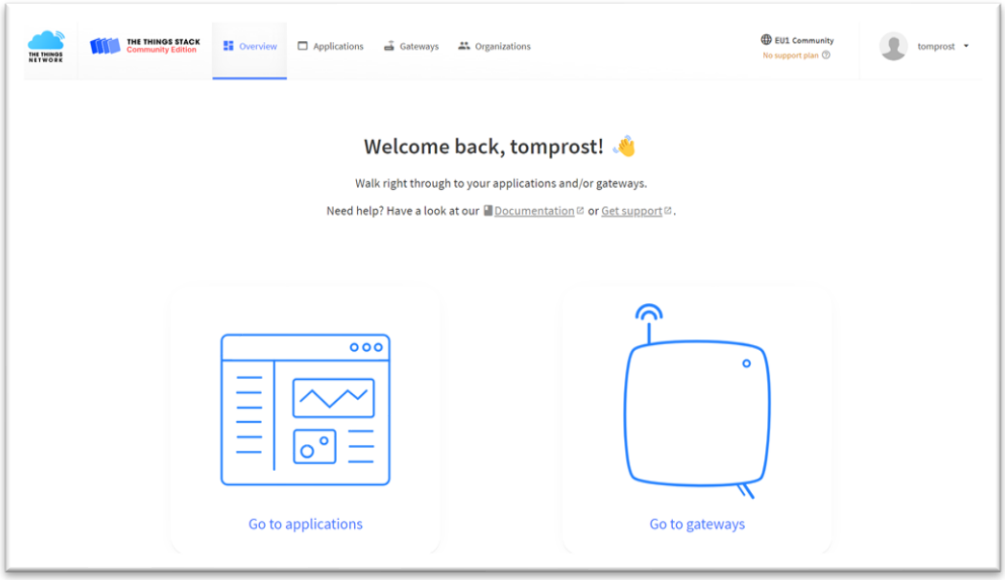


Figure 5 : Page d'accueil du compte TTN

III.1.5 Webhook

Un webhook est une fonction de rappel basée sur le protocole HTTP. Il permet à 2 API (Interface de programmation d'application) de communiquer entre elles. Ainsi, dans notre cas, le webhook permet la communication entre TTN et le serveur d'application (la base de données).

III.1.6 API et base de données

API est une abréviation qui signifie Application Programming Interface. C'est un moyen de communication entre deux logiciels, que ce soit entre différents composants d'une application ou entre deux applications différentes. Dans notre cas précis, cette API fait la jonction entre le Gateway qui envoie de l'information en boucle et l'ordinateur qui fait une requête pour recevoir de l'information (*Figure 6*). Par un réseau Wifi interne auquel est connecté le Gateway, il peut envoyer les informations à l'API, ceci est orchestré par un programme informatique dans TTN permettant le bon déroulé des opérations de transit d'information.

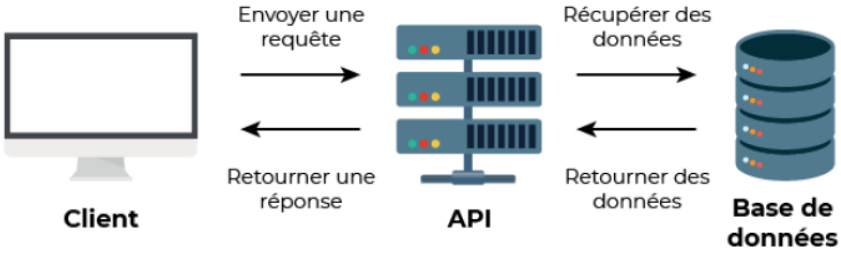


Figure 6 : Fonctionnement d'une API

III.1.7 Résumé

La *Figure 7* reprend le processus d'acheminement et de stockage de la donnée grâce au réseau LoRa.

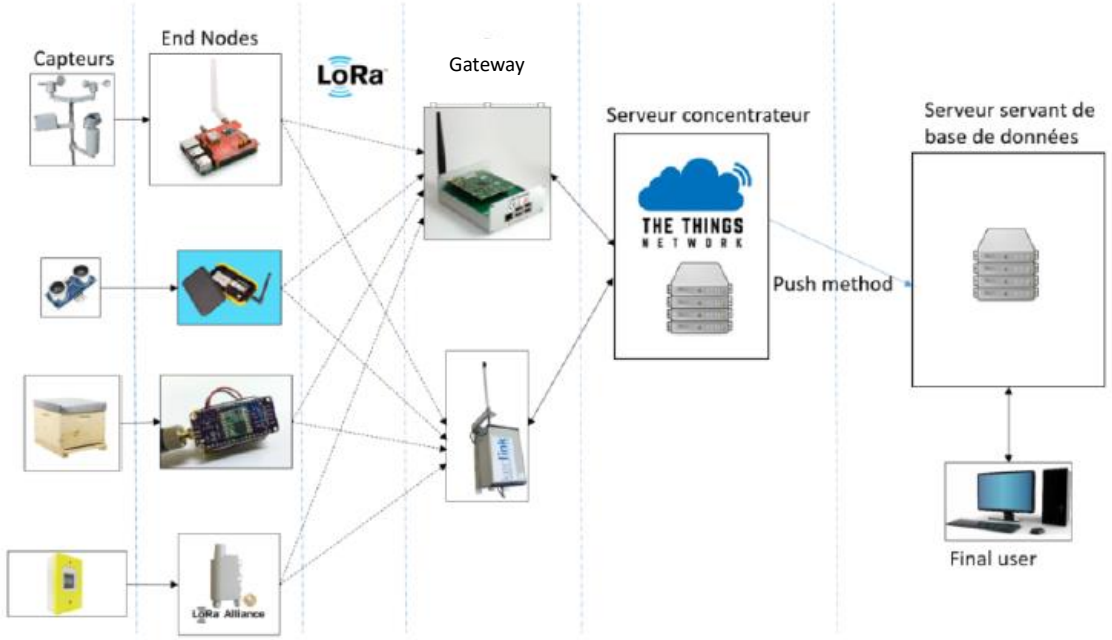


Figure 7 : Architecture globale du rapatriement des données par le réseau LoRa

Les capteurs mesurent des variables qui sont transmises à l'EndNodes LoRa, qui sont les dispositifs permettant de transmettre les données des capteurs. Ces nœuds sont souvent placés à distance ou dans des lieux isolés. Dans notre cas précis nous utilisons l'ESP32.
Il envoie des paquets d'informations en LoRa qui sont reçus par les passerelles dans son rayon de portée, qui elles, étant connectées à internet, vont renvoyés ces données sur TTN.
Via le webhook, les données vont transiter dans internet jusqu'au serveur servant de base de données. Et grâce à une API, on peut accéder sur la base de données en ligne aux datas en tant qu'utilisateur via un appareil connecté à internet (smartphone, ordinateur…).

De manière encore plus synthétique et pour le capteur qui fera l'objet du prochain paragraphe, le trajet du signal est le suivant :

Station météo (end device) -----LoRa-----> Gateway -----Wifi-----> Internet | TheThingsNetwork | iot.protechebdo.fr | Utilisateur final qui exploite les données

III.2 Base de travail : le VITINODE

Depuis la création du VITILAB en 2021, plusieurs projets ont été montés dont le projet « VITINODE ». Cette petite station météo connectée permet de suivre en direct la température et l'hygrométrie de l'air en inter-ceps (au cœur de la vigne). La problématique qui avait menée à son développement est que les stations météos dont disposaient les viticulteurs étaient hors parcelles, ce qui n'est pas assez précis lors des périodes de risques comme la survenue du gel ou l'apparition de maladie qui sont fonction de l'hygrométrie telle le mildiou. De plus, ces stations sont chères car il faut louer le matériel et un abonnement pour avoir accès aux données. Le but de VITINODE (*Figure 8*) est de fournir des données précises au cœur de la parcelle et d'être peu onéreuse. Le projet est déjà bien abouti car le design 3D, les composants choisis et le code sont bien développés et les retours de terrain sont plutôt encourageants. Seuls bémols, la station ne capte que peu de données différentes, et l'autonomie peut laisser à désirer, de l'ordre de 1 à 3 mois, ce qui retarde le vigneron de devoir aller la chercher pour la mettre à charger puis de retourner la mettre en place. (7. Acetayls. . GitHub - acetayls/Vitinode.)

Figure 8 : VITINODE en inter ceps

III.2.1 Besoins matériels

III.2.1.1 *Antenne LoRa*

Le Gateway LoRa qui a été retenu est de la marque GemTek (*Figure 9*). Il se branche sur le secteur, se connecte en WiFi pour faire le lien entre le réseau LoRa et Internet, et peut supporter jusqu'à un millier d'objets connectés. (*Annexe 1*)

Figure 9 : Gateway LoRa de chez GemTek

III.2.1.2 *Carte programmable*

L'équipe ayant travaillé sur ce projet a donc choisi d'utiliser comme carte programmable un ESP32 de chez LiLyGo (*Figure 10*) : le LiLyGo TTGO LoRa32 V1.0. Ce microcontrôleur a été choisi pour sa bonne connectivité car il dispose du Wifi, du Bluetooth, et d'un module LoRa avec une bonne sensibilité d'émission et de réception des données et qui est compatible avec les fréquences européennes. Il possède également un régulateur de charge pour les batteries au lithium, ce qui signifie que l'on peut brancher l'ESP32 via son port micro-USB à un chargeur et il va réguler la tension et le courant pour l'adapter de façon optimale à la batterie qui servait à l'alimenter. Il supporte parfaitement l'environnement de développement Arduino, et peut être utilisé pour la vérification des programmes de manière très facile et rapide.

Figure 10 : LiLyGo TTGO LoRa32 V1.0

III.2.1.3 *Coque de protection*

Les modèles (*Figure 11*) ont été intégralement conçus par notre équipe du VitiLab et sont ensuite imprimés en ABS par l'imprimante 3D Ultimaker S3 du laboratoire. Il est également possible et conseillé d'imprimer les stations en ASA qui est un dérivé de l'ABS mais résistant aux UV, ce qui est intéressant dans notre cas. Il faut aussi noter que le plastique d'impression doit être blanc car une coque imprimée en noir retiendra beaucoup plus la chaleur ce qui réduira la durée de vie des composants électroniques.

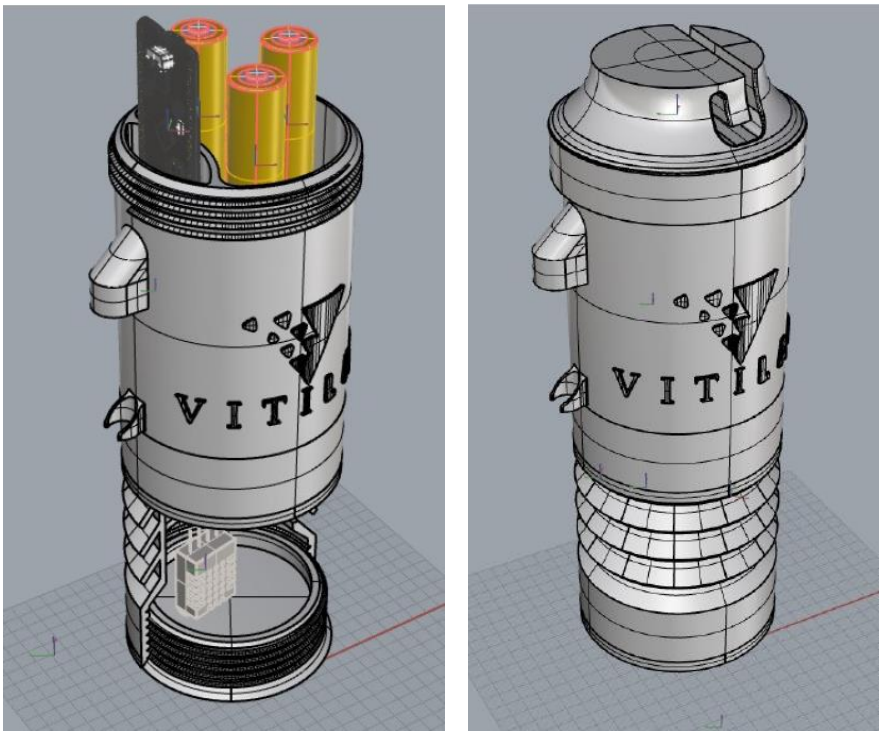


Figure 11 : Vue sous Rhino7 de la station météo

III.2.1.4 *Capteur*

Leur choix s'est porté sur un DHT22. Le capteur de température et d'humidité DHT22 communique avec un microcontrôleur via un port série. Le capteur est calibré et ne nécessite pas de composants supplémentaires pour pouvoir être utilisé. Il est utile de choisir un DHT22 à 3 broches au lieu d'un 4 broches car cela encombrera moins l'intérieur de la coque. Le DHT22 sera relié à l'ESP32 grâce à des fils jumper (*Annexe 2*).

Figure 12 : Modèle de DHT22 à 3 broches

III.2.1.5 *Batterie et câble*

Les batteries sont de petits accumulateurs LiPo (Lithium Polymère) de tension nominale 3,7 V (*Figure 13*) car l'ESP32 devant être alimenté entre 3.3 et 7V, le 3,7V est un bon compromis parmi les différents standards. (*Annexe 3*) La capacité choisie était de 2500 mAh (milli Ampère heure).

Figure 13 : Batterie LiPo de 3.7V

Pour recharger la batterie via l'ESP32, ou bien pour téléverser le programme de l'ordinateur à l'ESP32, il faudra un câble micro-USB *(Figure 14*).

Figure 14 : Câble micro-USB

III.2.1.6 *Résistances*

Il faudra aussi deux résistances (*Figure 15*) de 3.3 kΩ (kOhms) afin de créer un pont diviseur de tension dans le but de lire le niveau de batterie restante.

Figure 15 : Résistance de 3.3kΩ

III.2.1.7 *Câblage*

Dans cette étape, il faudra souder les différentes broches du DHT22 et les fils de la batterie sur les GPIO (ports d'entrée-Sortie) du microcontrôleur ESP32. Il faudra donc avoir un disposition un fer à souder et un peu d'étain pour réaliser cette opération.

Toutefois, il est possible également d'utiliser une "breadboard" (Carte prototype ou PCB) pour éviter d'avoir à faire les soudures et ainsi pouvoir faire des tests. Cependant cette solution ne peut être retenue si on veut mettre le matériel dans le boîtier du Vitilab afin de le placer en condition réelle sur le terrain.

Quoiqu'il en soit, les différentes broches et fils doivent être reliés aux ports (GPIO) du microcontrôleur suivant (*Figure 16*) :

DHT 22 :

- La broche DHT (aussi appelée Data, Out…) sur le port 16 (ou sur un autre port GPIO disponible en *Annexe 4*.)
- La broche + (Vcc) sur le port 3,3V
- La broche – (GND) sur le port Masse (GND)

Pour la batterie et les résistances, respecter le schéma suivant :

- Masse (GND) <- 3.3 kOhm -> PIN 36 <- 3.3kOhms -> +5V

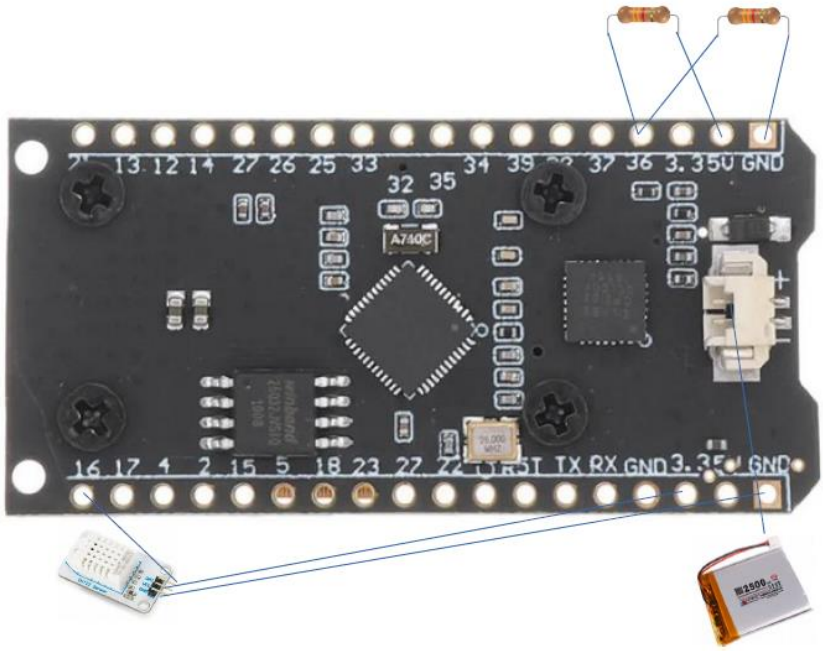


Figure 16 : Illustration du câblage des composants sur l'ESP32

III.2.2 Besoins logiciels

Le Vitilab a fait intervenir un collaborateur interne développeur WEB car la partie élaboration d'une API, création d'une base de données, et écriture du code nécessitait une intervention externe pour l'avancée du projet.

III.2.2.1 *TheThingsNetwork*

Tout le détail est sur le GitHub, mais les étapes principales se résument à créer un compte TTN, puis il faut s'identifier dans la zone Europe et rejoindre l'organisation du VITILAB pour avoir accès à l'application vinimétéo. Si la Gateway est déjà connectée, il ne reste plus qu'à créer des Devices pour associer un code d'authentification à chaque objet connecté.

Pour communiquer avec le serveur https://iot.protechebdo.fr/devices, les data des stations doivent être mise en forme. On utilise un uplink payload formatter dans TTN. Il est programmé par défaut dans l'application TTN.

III.2.2.2 *Visual Studio Code (VS Code)*

Pour saisir les informations spécifiques à nos besoins, on aura besoin de modifier le code du programme que vous pourrez télécharger sur le dépôt git dont l'adresse est fournie en *III.2.2.4.*

Visual Studio Code est un éditeur de code développé par Microsoft. Il s'agit d'un logiciel gratuit qui est installable sur Windows mais aussi Linux ou MacOS.

III.2.2.3 *PlatformIO IDE*

Platformio est un IDE (Integrated Development Environment). Il propose un ensemble d'outils de développement professionnel. Cet IDE est une extension que l'on rajoute dans VS Code et il est très répandu chez ses utilisateurs, et le projet VITINODE utilise les fonctions de PlatformIO dans son code.
L'utilité d'utiliser PlatformIO est qu'on peut créer un projet et que n'importe qui téléchargeant ce projet pourra directement l'utiliser dans PlatformIO contrairement à Arduino IDE.
Un projet se compose de plusieurs types de fichiers :

- READ ME où l'auteur peut décrire brièvement le projet
- PlatformIO.ini où sont définis la carte utilisée, le langage, la vitesse de communication et les librairies appelées dans le programme
- Les librairies (lib_deps) sont des packages téléchargeables dans PlatformIO, elles ont été écrites en open source et sont utiles car elles résument tout un programme en quelques lignes, elles doivent être appelées dans le code avec #include <XXX.h>
- Le fichier .cpp où est écrit le code principal, l'appel des librairies se fait au début et ensuite on peut appeler des fonctions incluses dans la librairie du type XXX.read(); ce qui en une ligne lit les données, fait les calculs…
- Le fichier .h où sont décrites toutes les variables modifiables afin qu'un utilisateur ne modifie pas le code principal

J'assimilerai maintenant PlatformIO à VS Code.

III.2.2.4 *Programme du VITINODE*

Pour que l'objet connecté puisse fonctionner il y aura besoin du programme développé par le VITILAB qui se nomme VITINODE - Main.
On peut le trouver dans le dépôt GitHub du projet ou dans la bibliographie de ce rapport. Le READ ME (texte explicatif) du projet explique également toute la construction du VITINODE, plus en détail que l'état de l'art de ce rapport.

Comme c'est un projet, dans le dossier src, il y a le fichier main.cpp qui contient le corps du code, et le fichier configuration.h où sont regroupés toutes les variables modifiables. On peut y changer la fréquence à laquelle on envoie des données, quel type de capteur est branché, et bien penser d'adapter les codes d'authentification qui ont été générés sur TTN pour le device en question. Il faudra ensuite compiler (build) et téléverser (upload) avec un câble µUSB le programme depuis l'ordinateur afin que l'ESP32 soit encodé et en état de marche.

Le programme principal est différent de la structure normale à savoir déclaration des variables globales et appel de librairies puis la partie Setup (initialisation) et enfin la partie Loop (répétition infinie du programme) car on ne rentre jamais dans le Loop. Toutes les mesures sont effectuées dans la partie déclaration et l'envoi des données sur TTN se fait dans le Setup

puis l'ESP32 se met en Deepsleep et ne se réveille qu'au bout du temps choisi et recommence les mesures.

III.2.2.5 *IoT.ProtecHebdo*

C'est sur ce site aussi appelé IoT VSB, qu'une fois tous les composants sont branchés, que l'ESP32 encodé émet, que les données sont reçues sur TTN, que vont les données. Ce site a été développé par un développeur WEB pour le VITILAB, il constitue la base de données pour le stockage. C'est elle qui joue l'interface homme-machine. Cette base permet de gérer, transformer ou éditer les données, de les exporter sous Excel ou en CSV, afin de pouvoir les conserver dans le temps, sans surcharger l'hébergement sur les serveurs. Il faudra se créer un compte pour accéder aux données.

III.1 Méthodologie bibliographique

Pour mener mes recherches bibliographiques sur les meilleures technologies de capteurs, j'ai utilisé plusieurs moyens de recherches. Si une recherche en français est infructueuse, le réflexe à avoir est de taper la recherche en anglais dans le moteur de recherche quel qu'il soit car l'offre de document technique accessible est bien plus étoffée en anglais qu'en français. Si le moteur de recherche anglais ne donnait rien de convaincant non plus, je me documentais sur différents sites que voici.

III.1.1 GitHub

GitHub est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels, qui a été lancée en 2008 et été rachetée par Microsoft en 2018. Elle repose sur Git, un système de gestion de code open source créé par Linus Torvalds, le créateur de Linux, dans le but d'accélérer le développement logiciel. Git permet de stocker le code source d'un projet et de suivre l'historique complet de toutes les modifications apportées à ce code. Grâce aux outils qu'elle fournit pour gérer les conflits éventuels résultant des changements apportés par plusieurs développeurs, il est possible de collaborer efficacement sur un même projet.

Je m'en servais en outil de recherche car les développeurs qui le souhaitent peuvent laisser leurs projets de code en open-source, on peut donc les consulter et les télécharger si l'on souhaite s'en servir de base.

III.1.2 Forum Arduino

Autre outil pratique car la communauté Arduino est très active sur les projets de capteurs DIY, ce qui signifie que quand l'on fait une recherche sur un capteur avec « Arduino » en mot clé, on trouve des sujets de forum où les gens postent des questions et d'autres répondent sur un principe collaboratif avec très souvent une réponse à notre question dans les retours d'expériences publiés dans les commentaires.

III.1.3 RitonDuino

En parcourant le forum Arduino, je suis tombé sur un membre extrêmement présent sur beaucoup de sujets et dans son profil, un lien vers un blog personnel. Il y écrit beaucoup d'articles sur toutes les réalisations qu'il fait, très centré sur des capteurs DIY économes en énergie.

III.2 Mes projets

Pour rappel, la problématique du stage est de concevoir des objets connectés en DIY afin qu'ils soient peu coûteux par leur conception et de pouvoir disposer des données en libre accès. J'ai ainsi mené trois projets en parallèle pour développer des solutions IoT utiles en vitiviniculture.

III.2.1 Améliorer l'autonomie du VITINODE

La principale tâche noire du VITINODE est son autonomie. Malgré le mode DeepSleep qui met en veille profonde l'ESP32 le temps qu'il n'envoie pas de données afin de réduire fortement la consommation électrique, la batterie ou les batteries mises en parallèle ne duraient que 1 à 3 mois. Ceci n'est pas suffisant car il doit être installé sur des câbles de vignes idéalement de janvier à septembre, car depuis le début de la période de taille jusqu'à la fin des vendanges, donc il doit être autonome en énergie sur une longue durée.
Je me suis basé ici sur un projet traitant de l'alimentation par panneaux solaires. (8.(BACHETTI, Alimentation par batterie + panneaux solaires, 2020))

III.2.1.1 *Régulateur de charge*

Le paragraphe 3.4 de cet article décrit le TP4056. Il est équipé de deux LEDs : une rouge quand la charge est en cours et une bleue quand la charge est terminée. Quand le panneau ne fournit plus de courant, les deux LEDs sont éteintes. On peut brancher un chargeur USB, il existe des modèles mini, micro-USB ou type C, on choisira le modèle µUSB par souci de rétrocompatibilité avec l'ESP32 qui charge en µUSB. Ou alors utiliser les deux pastilles marquées IN+ et IN- pour y brancher un panneau solaire 5V. La carte TP4056 (*Figure 17*) a une consommation très faible (typiquement 2.5µA) sur la batterie lorsque son entrée est débranchée ou que le panneau ne fournit pas d'énergie. Il faut aussi bien choisir le modèle avec circuit intégré de protection de la batterie contre les surcharges et décharges excessives pour qu'il charge la batterie en utilisant la méthode de charge à courant constant / tension constante jusqu'à 4,2 V et se bloque en dessous de 2,4V. Le TP4056 admet jusqu'à 8,2V en entrée avant destruction, il faut donc choisir un panneau solaire dont la tension à vide en plein soleil ne dépasse pas 8,2V.
Les avantage de ce petit module régulateur de charge est qu'il est très efficient en énergie, qu'une résistance permet de diminuer le courant de charge mais les batteries utilisées supportent 1A qu'il délivre en version d'usine, 1A étant également le courant maximum d'alimentation d'un ESP32. (*Annexe 5*)

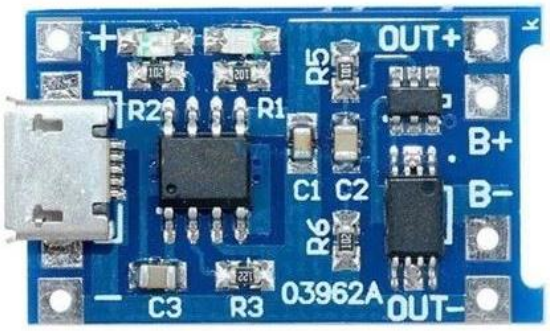


Figure 17 : TP4056 µUSB avec protection

III.2.1.2 *Panneau solaire*

Il faut donc un panneau solaire qui délivre 5V sans dépasser 1A et de tension à vide inférieure à 8,2V. Mon choix s'est porté sur le panneau solaire Seeed Studio, de puissance 0.5W car il délivre en moyenne 5V à 100mA (P = U*I soit 0,5W = 5V * 1.10^{-2}A). Sa tension à vide est de 8V, ce qui est le maximum autorisé sans risque de dommages. Autres avantages, il est en silicium monocristallin ce qui signifie qu'il a un meilleur rendement qu'un équivalent en silicium polycristallin, et il est de faibles dimensions de 70*55*3 mm, ce qui facilitera l'intégration au VITINODE sans être trop volumineux.

III.2.1.3 *Câblage*

Le TP4056 charge la batterie grâce au courant délivré par un panneau solaire, qui à son tour alimente le système par le port où se fixe normalement la batterie (*Figure 18*). Par mesure de précaution, on peut rajouter une diode entre le panneau solaire et le TP4056 afin d'être sûr qu'il n'y ait pas de retour de courant dans le panneau solaire. (*Annexe 6*)

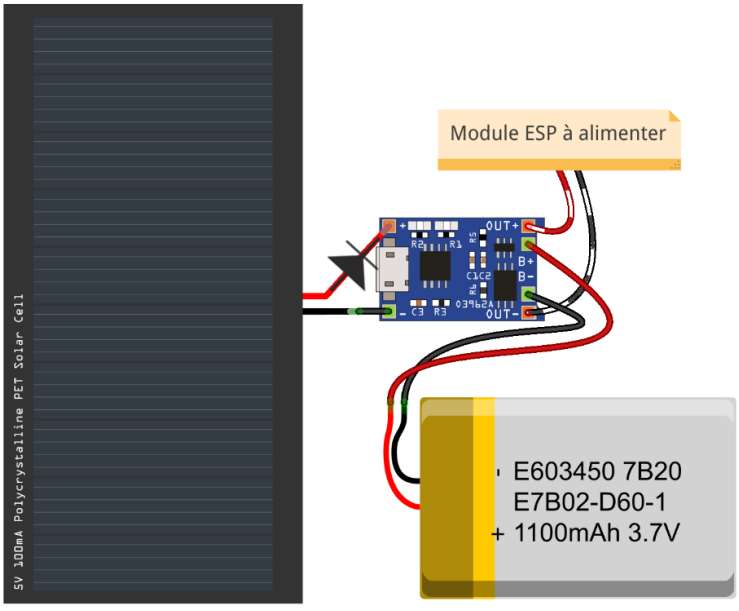


Figure 18 : Câblage du TP4056 avec un panneau solaire

III.2.1.4 *Design 3D*

Avant de commencer à dessiner en 3D une pièce, il est intéressant de regarder sur Thingiverse où les gens déposent des fichiers STL, qui est le standard de fichier prêt à imprimer, le problème étant qu'ils ne sont pas modifiables ou très peu, on ne peut jouer que sur l'échelle du modèle.

Les prochaines personnes qui prendront le projet et sachant se servir de Rhino7 pourront remodeler le fichier original afin d'imprimer des corps de VITINODE directement avec le support panneau intégré. Afin d'intégrer le panneau solaire aux VITINODES existants, il faudra faire un support à imprimer en 3D afin de le fixer sur la coque. Pour cela, je dois choisir entre les différents logiciels car rien n'existe sur Thingiverse du fait que ce soit un projet interne.

On a appris à se servir de FreeCAD à l'IUT mais n'étant pas des plus ergonomique et Rhino7 étant très compliqué à prendre en main pour les personnes n'ayant pas été formées, j'ai cherché quel logiciel serait adapté à un semi débutant en CAO (Conception Assistée par Ordinateur), et Fusion360 remplissait ces critères. Un gros avantage de ce logiciel est l'historique de conception, ce qui permet de revenir en arrière et que les modifications s'appliquent à la suite.

III.2.2 Niveau d'eau connecté

La ferme expérimentale de Jalogny, qui est une composante de la Chambre d'Agriculture de Saône-et-Loire, avait fait une requête quant à une solution pour monitorer le niveau d'eau d'une cuve enterrée. Actuellement, un technicien doit aller voir en soulevant le couvercle de la cuve, ce qui est en plus d'être peu pratique n'est pas très précis car il est difficile d'estimer ce qu'il reste sur une cuve de 4m de haut. (9. Bachetti, H. . *Contrôle de Niveau d'Eau Connecté*.)

III.2.2.1 *HC-SR04*

La solution la plus usitée dans la communauté geek est un capteur à ultrason, baptisé le HC-SR04. Il permet de monitorer une distance allant de 2 à 450 cm, précis à 0,5 cm, tout en étant bon marché. En exploitant le principe d'un sonar, il mesure donc le temps que met à revenir l'onde ultrasonore réfléchie et renvoie un signal carré dont la longueur de l'impulsion est proportionnelle au temps de réverbération. Il a besoin d'une alimentation en 5V, et possède deux broches (pins) TRIG et ECHO, TRIG étant celle qui reçoit l'ordre de faire une mesure par l'ESP32 et ECHO celle où l'on va lire la durée de l'impulsion. (*Annexe 7*)

Connaissant la hauteur de la cuve et la distance mesurée entre le capteur et l'obstacle, ici la surface de l'eau, on peut en déduire la hauteur d'eau et par un produit en croix le remplissage (*Figure 19*).

Maintenant, ce projet sera appelé AGRISONAR afin de le dénommer plus facilement.

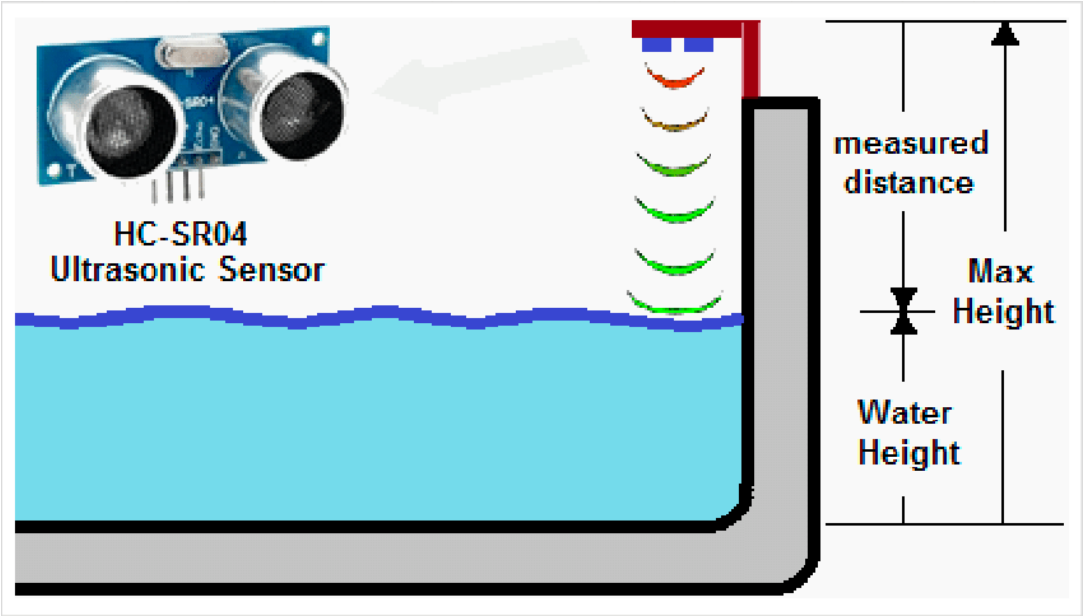


Figure 19 : Capteur HC-SR04 et principe de mesure

Un inconvénient du HC-SR04 est qu'il n'est pas étanche et sera donc sensible à la condensation sous le couvercle de la cuve (BACHETTI, Contrôle de Niveau d'Eau Connecté, 2022).

III.2.2.1 *DYP-A02YY*

Ce capteur (*Figure 20*) est du même type, même résolution et plage de mesure que le HC-SR04 à la différence d'être étanche. Attention toutefois à bien choisir l'option de transmission de données en PWM (Modulation de largeur d'impulsion) par rapport à un protocole RS485 ou UART. Sinon le protocole de lecture des données ne sera pas du tout le même. (*Annexe 8*)

Figure 20 : DYP-A02YY

III.2.3 Consommation d'énergie en temps réel

Ce projet s'inscrit dans la suite de ce qu'avait commencé Melvin CUZIN et Yassine NOUR-CHAFI, les précédents stagiaires du Vinipôle venant de Mesures Physiques Creusot. L'objectif est d'avoir des outils pour quantifier la consommation d'énergie, applicable en théorie à tous types de vinifications, qui fournirait un suivi précis des consommations énergétiques de la parcelle à la bouteille. La mise en place d'outils connectés facilitant le remplissage de ce calculateur créé par mes précédents collègues permettrait de générer une prise de conscience concernant la consommation d'énergie et d'établir des plans d'actions pour approcher au mieux la neutralité carbone.

La partie de la parcelle à la cuverie ayant déjà été abordée par d'autres avant moi, le but maintenant sera donc de compléter la gamme de mesure en pouvant installer des mouchards sur les machines de la cuverie pour quantifier l'énergie électrique nécessaire pour vinifier un vin. En plus d'avoir la consommation générale au compteur EDF, il s'agit de pouvoir monitorer la consommation de chaque machine sachant qu'elles sont alimentées en triphasé. L'appareil de mesure doit pouvoir être mobile dans la cuverie. Le but est donc de mesurer la puissance active consommée par un appareil branché sur une prise triphasée.

III.2.3.1 *Triphasé*

L'amplitude Vpp d'un signal électrique alternatif pur est la valeur entre le 0V et le maximum, la valeur efficace $Veff = Vpp/\sqrt{2}$. Le déphasage est le retard que les phases ont entre elles.

Dans une alimentation domestique, l'électricité est dite monophasée car il n'y arrive dans la prise que la terre, un neutre et une phase, que l'on nomme 230V car il y a une différence de potentiel de 230Veff entre la phase et le neutre, la phase étant à 230 et le neutre à 0V, et la terre étant là en sécurité.

Dans une installation industrielle, l'électricité est triphasée, selon la nouvelle norme il y a une terre, un neutre et trois phases. Il y a une tension de 230Veff à chaque fois entre le neutre et une phase (tensions simples) mais 400Veff entre les phases (tensions composées).

La nouvelle norme européenne associe la terre a un fil de couleur verte et jaune, le neutre en bleu et toutes les autres couleurs pour les phases avec une préférence pour le noir, le marron et le gris (*Figure 21*). L'ancienne norme ne rendait pas le neutre obligatoire ce qui donnait des prises triphasées à 4 fiches.

Un système de tension triphasé est constitué de trois tensions sinusoïdales de même fréquence et de même amplitude qui sont déphasés entre eux de 120 degrés dans le cas idéal. Lorsque les trois conducteurs sont parcourus par des courants de même valeur efficace et sont déphasés de 2π⁄3, le système est dit équilibré. Idéalement la tension des trois phases est constante et indépendante de la charge, seul le courant de chaque phase devant être dépendant de la puissance de sortie.
Du fait du déphasage de 120°, un réseau dont la tension efficace entre phase et neutre est de 230 V aura une tension composée de $\sqrt{3} * 230\ Veff = 400\ Veff$ entre les phases.

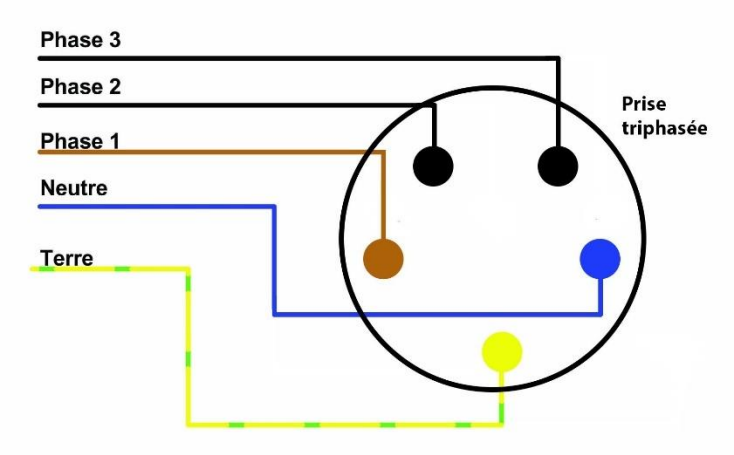


Figure 21 : Nouvelle norme de prise industrielle en triphasé

L'intérêt est un meilleur rendement des machines, plus de puissance et moins de pertes lors du transport. Le problème est que le 400V (aussi appelé 380V) est plus dangereux et il faut donc réfléchir à la sécurité électrique de l'opérateur.

Plusieurs principes physiques peuvent permettre de calculer la puissance.

III.2.3.2 *Capteur à effet Hall*

Un capteur à effet Hall permet de mesurer une variation de champ magnétique. Dans la mesure où un courant électrique engendre un champ magnétique lorsqu'il parcourt un matériau conducteur, on se sert aussi d'un capteur à effet Hall particulier pour mesurer l'intensité d'un courant électrique.

III.2.3.3 *Bobine de Rogowski*

La boucle de Rogowski ou boucle de courant est un instrument dédié à la mesure des courants alternatifs de faible intensité.

III.2.3.4 *Pince ampèremétrique*

La pince ampèremétrique, aussi appelée capteur de courant sans contact, est un type d'ampèremètre permettant de mesurer l'intensité du courant électrique circulant dans un fil conducteur sans avoir à ouvrir le circuit pour y placer un ampèremètre classique. Le fonctionnement de la pince ampèremétrique se base sur la mesure indirecte du courant circulant dans un conducteur à partir du champ magnétique ou du champ électrique qu'engendre cette circulation de courant.

Elle permet une sécurité pour l'opérateur qui réalise la mesure puisqu'aucun contact électrique avec le circuit n'est effectué. Il n'est donc pas nécessaire d'enlever l'isolant électrique entourant le fil lors de la mesure.

III.2.3.5 *Compteur électrique triphasé*

Toutefois, ces différentes méthodes de mesures de courants nécessitent une mesure de tension simultanée dans le cas où le système viendrait à se déséquilibrer et alors les tensions

ne seraient plus constantes. Pour des raisons de sécurité, j'ai choisi de ne pas utiliser de capteurs séparés pour chaque phase pour faire diminuer le prix mais de choisir un compteur où l'opérateur a juste à brancher les entrées et les sorties des phases et du neutre dans un bornier à vis (*Figure 22*), ce qui évite la manipulation de trop de fils.

Maintenant, ce projet sera appelé VINIPOWER afin de le dénommer plus facilement.

Afin de monitorer la sortie du compteur, il existait des versions en protocole RS485, mais une liaison filaire vers une centrale n'est pas possible du fait du réarrangement régulier du matériel dans la cuverie.

L'option retenue est donc un compteur avec une sortie impulsionnelle, le SDM530D de chez EASTRON. Il génère des impulsions proportionnelles à l'énergie mesurée et il délivre 800 impulsions/kWh. La sortie d'impulsion est une sortie de transistor passive, dépendante de la polarité, qui nécessite une source de tension externe pour fonctionner correctement. Pour cette source de tension externe, la tension doit être comprise entre 5 et 27 V CC, et le courant d'entrée maximum doit être de 27mA DC. Pour utiliser la sortie d'impulsion, connecter le 5V DC au connecteur 9 (anode), et le fil de signal (S) au connecteur 8 (cathode). Les impulsions du compteur sont indiquées sur une LED du panneau avant. (*Annexe 9*)

Figure 22 : Compteur de puissance active en triphasé EASTRON SDM530D

Afin de s'assurer que la sortie à impulsions soit entièrement séparée du circuit interne et de l'ESP32, on place un optocoupleur (*Figure 23*) entre les deux. Un optocoupleur est un composant électronique capable de transmettre un signal d'un circuit électrique à un autre, sans qu'il y ait de contact galvanique entre eux, car la transmission est reconstituée par un émetteur de lumière et un capteur de lumière. (*Annexe 10*)

Figure 23 : Module d'isolation basé sur un optocoupleur

On alimente le module entre 5V et GND depuis l'ESP32, l'entrée + est la cathode de la sortie à impulsion et l'entrée – est reliée à la masse. La sortie OUT se lit directement sur un port GPIO de l'ESP32.

III.2.3.6 *Alimentation*

Une option aurait été d'alimenter le montage par batterie mais la question ne s'est pas posé puisque nous avons une source électrique à disposition, certes en triphasé, mais qu'il fallait exploiter. La solution la plus adaptée et la plus sécuritaire sera d'utiliser une alimentation à découpage. J'ai choisi comme modèle (*Figure 24*) une MeanWell RS-15-5 3A. On branche le neutre et une phase une fois qu'ils sont sortis du compteur sur N et L AC, la terre à la masse et l'on obtient du 5V DC entre V+ et V-. Il y a un potentiomètre d'ajustement (ADJ). L'alimentation de l'ESP32 sera réalisée donc réalisé par le 5V délivré par le MeanWell. Il peut délivrer jusqu'à 3A d'intensité mais comme l'ESP32 n'a besoin que d'un ampère maximum pour s'alimenter, l'alimentation s'adaptera pour délivrer le courant de façon optimale. (*Annexe 11*)

Figure 24 : Alimentation à découpage MeanWell RS-15-5 3A

III.2.3.7 *Draw.io*

Pour mieux me repérer dans les montages que j'aurai à câbler, j'ai dessiné sous un logiciel nommé Draw.io des schémas électriques afin d'être sûr de ne pas me tromper lors du prototypage.

IV. Prototypes, tests & mesures

IV.1 Acquisition des données

Il existe beaucoup de manières pour récupérer les données, en Wifi, avec des protocoles RS485, MQTT, JavaScript, des plateformes tout en un de développement IoT comme Blynk…

Mais pour répondre aux différents cahiers des charges et afin de limiter le nombre de compétences à maitriser sur des technologies différentes, j'ai fait le choix de rapatrier les données de chaque projet en me basant sur le VITINODE. Chaque objet connecté sera donc équipé d'un ESP32 qui conditionnera la mesure avant de l'envoyer en LoRa.

Le code sera modifié de manière à pouvoir acquérir les données de chaque projet avec le même code afin de faire un projet tout en un à télécharger puis après choisir le type de capteur dans le code afin d'en activer les bonnes parties.

IV.2 Prototypage

IV.2.1 SOLAR VITINODE

À la différence des deux autres projets qui nécessitent de coder, j'ai dû ici seulement brancher les câbles et réaliser l'intégration du panneau solaire au VITINODE.

IV.2.1.1 *Support 3D*

J'ai ainsi modélisé un support pour le panneau solaire (*Figure 25*) afin qu'il soit adaptable sur les anciens VITINODES. Les fils sortent sous la plaque VITILAB et en perçant la coque du VITINODE, on peut les faire rejoindre le TP4056. Le support se fixe à la colle chaude à l'opposé de l'antenne LoRa sur la coque du VITINODE.

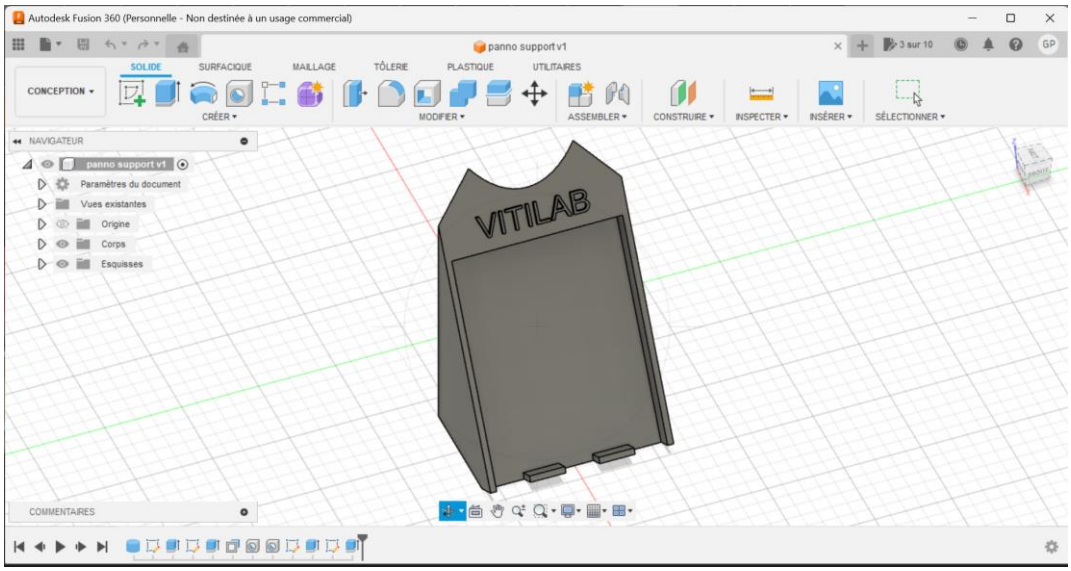


Figure 25 : Vue sous Fusion360 du support pour panneau solaire

IV.2.1.2 *Câblage*

Pour câbler comme sur la *Figure 26* , j'ai suivi le câblage énoncé sur la *Figure 18* .

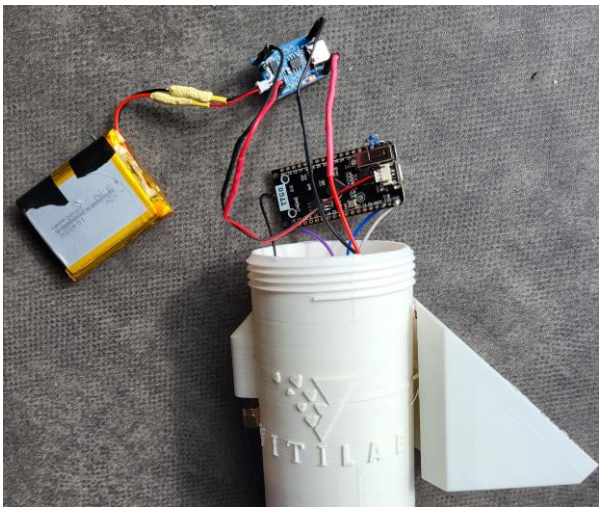


Figure 26 : Montage du panneau solaire avec le régulateur de charge

IV.2.2 AGRISONAR

Tous les tests jusqu'à la version définitive seront faits sur le HC-SR04 car si un capteur devait être endommagé, il vaut mieux le faire sur un HC-SR04 à 2€ que sur un DYP-A02YY à 20€. Afin de tester le fonctionnement du capteur, je l'ai branché comme sur la *Figure 27*. L'ESP est alimenté via un port USB et le capteur est branché sur les ports 5V et GND pour fonctionner, et les pins TRIG et ECHO sur les GPIO 16 et 17.

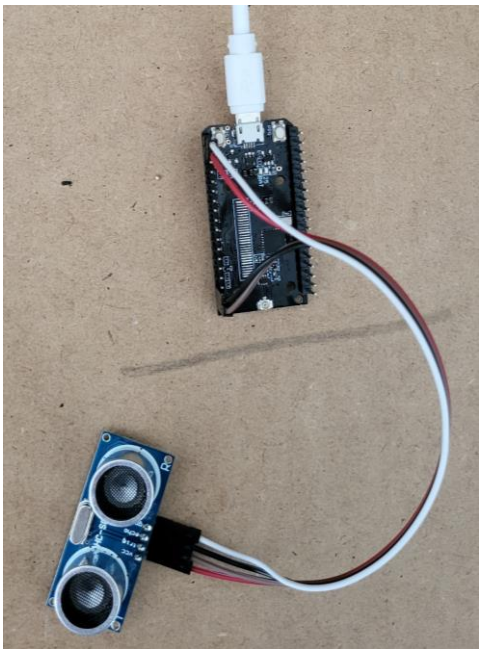


Figure 27 : Branchement du capteur de niveau

IV.2.3 VINIPOWER

IV.2.3.1 *Simulation d'impulsion*

N'ayant pas d'habilitation électrique, je créé une maquette de simulation pour tester mon programme avant les tests grandeur nature. Le but étant de simuler une impulsion qui serait émise par le compteur lorsque de l'énergie est consommée, je me suis servi d'un ESP32, d'un optocoupleur et d'un bouton poussoir que j'ai monté comme sur la *Figure 28*. Le schéma électrique a été réalisé sur Draw.io .

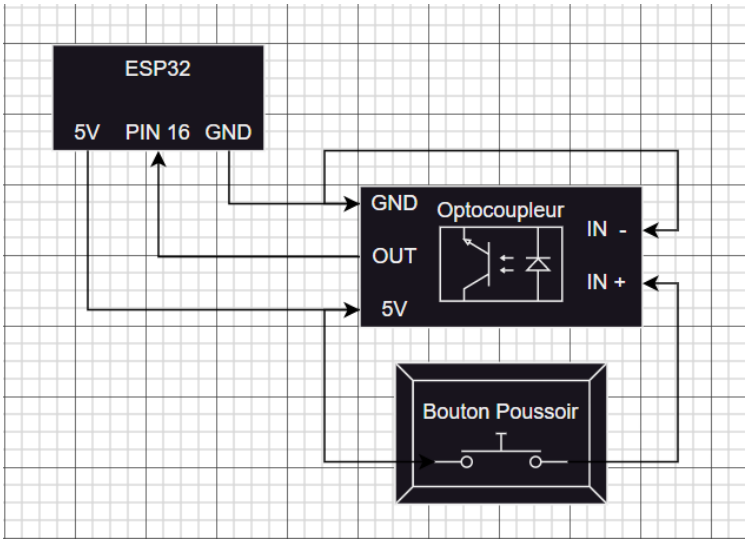


Figure 28 : Schéma de câblage de la maquette de simulation d'impulsion

Le montage est fonctionnel, car lorsque je clique sur le bouton poussoir (BP), la LED s'allume (*Figure 29*) signifiant qu'une impulsion a bien été émise par l'optocoupleur.

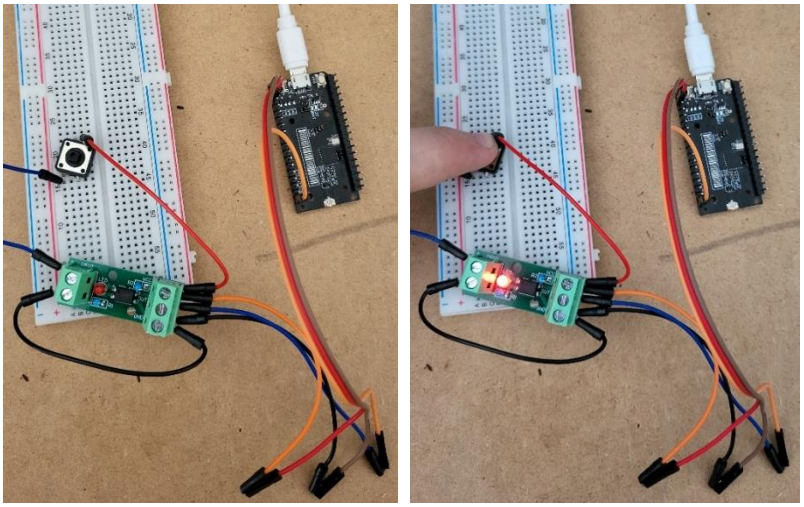
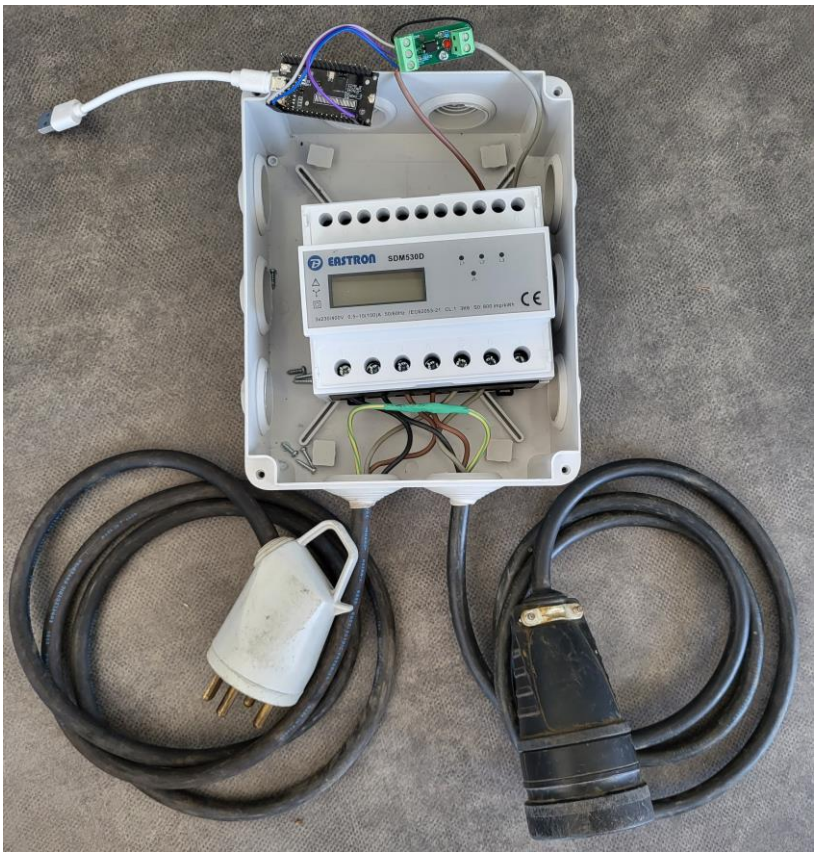


Figure 29 : Maquette de simulation d'impulsion en fonctionnement

IV.2.3.2 *Compteur électrique triphasé*

J'ai donc en parralèle monté un protoype (*Figure 30*) du VINIPOWER dans une boîte de dérivation. Je me suis basé sur le schéma électrique dessiné sur Draw.io de la *Figure 31*. À ce

stade du prototype, la seule différence est que je n'ai pas reçu l'alimentation à découpage donc j'alimente l'ESP32 avec un port USB de l'ordinateur.



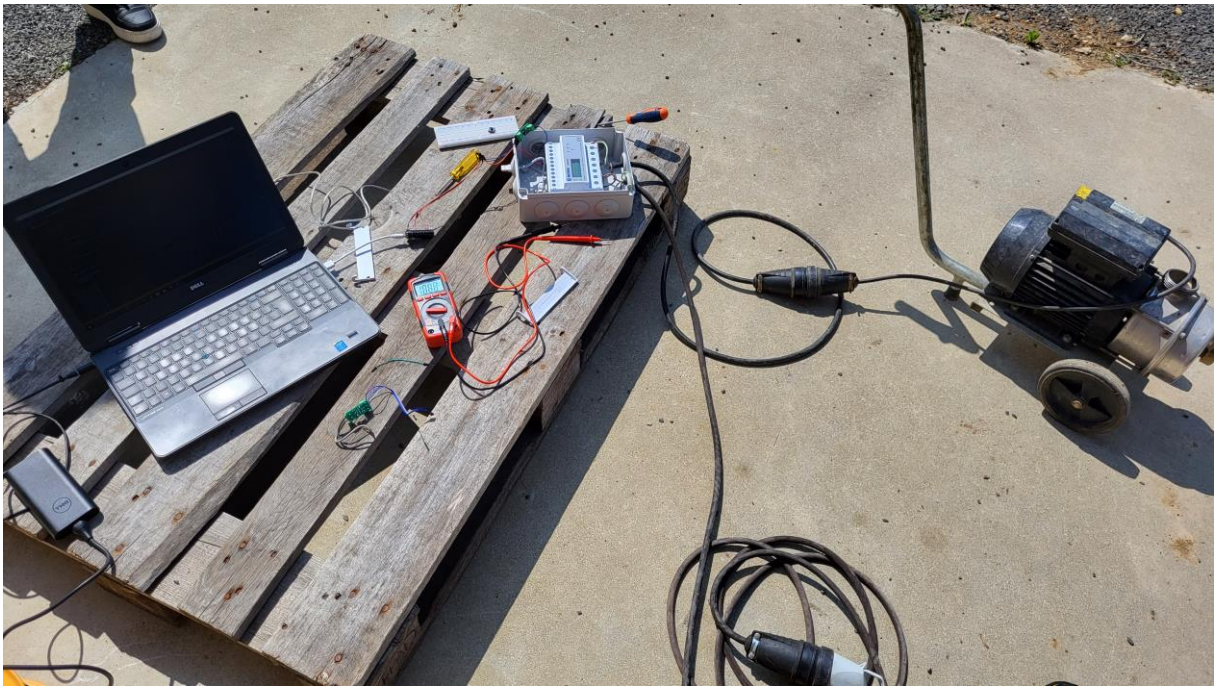


Figure 30 : Prototype du VINIPOWER sans l'alimentation à découpage

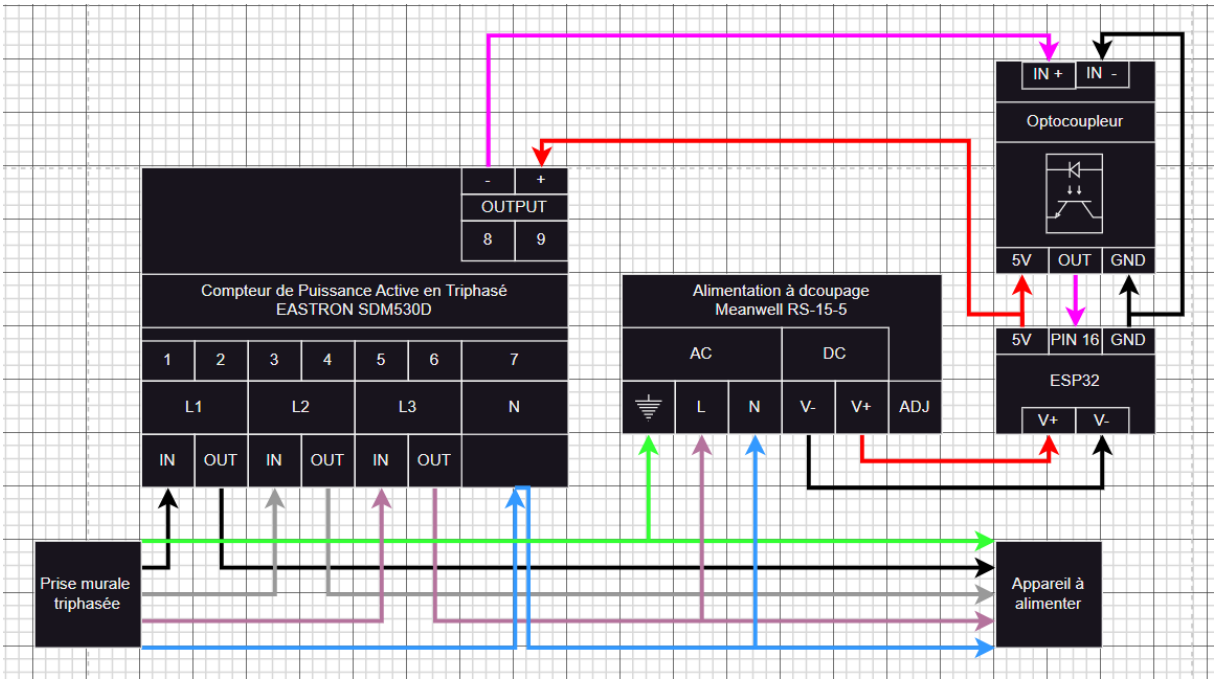


Figure 31 : Schéma électrique sous Draw.io du VINIPOWER

IV.3 Test des prototypes

Sachant que je veux utiliser un ESP32 communiquant en LoRa en intégrant les projets dans un même programme, pour gagner du temps, je procédais en deux étapes avec d'abord un développement sous Arduino puis le transfert sous VSCode.

IV.3.1 Moniteur Série Arduino

Le langage utilisé ici est du C++ simplifié. Arduino possède de base certaines librairies pour faciliter la programmation. Cependant nous utiliserons d'autres librairies qui devront être installées afin de permettre à la carte de comprendre notre programme.
Les programmes sont divisés en 3 grandes parties :

- Déclaration des variables globales et appel de librairies
- La partie Setup (initialisation)
- La partie Loop (répétition infinie du programme)

Il existe de très nombreux programmes déjà écrits en open-source et qui fonctionne très bien. Ils sont très utiles pour essayer le branchement des capteurs afin de voir dans un premier temps si les données mesurées sont fiables. L'ESP32 est branché sur un port USB de l'ordinateur ce qui permet d'utiliser le Moniteur Série qui renvoie sur l'écran ce que calcule l'ESP32.

Afin de pouvoir monitorer en série un ESP32 avec Arduino IDE, il suffit de paramétrer correctement (Figure 32) la carte (choisir le bon port USB où celle-ci est branchée et choisir la bonne carte dans les paramètres). Il faut donc d'abord installer le package ESP32 téléchargeable sur Internet. Contrairement à la configuration d'un Arduino, il faut aussi bien

modifier le BaudRate dans le moniteur série car il est défini sur 9600 alors qu'un ESP32 communique à la vitesse de 115200 bits/s (=115200 Bauds). On peut ouvrir le moniteur série depuis l'onglet Tools ou avec Ctrl+Maj+M.

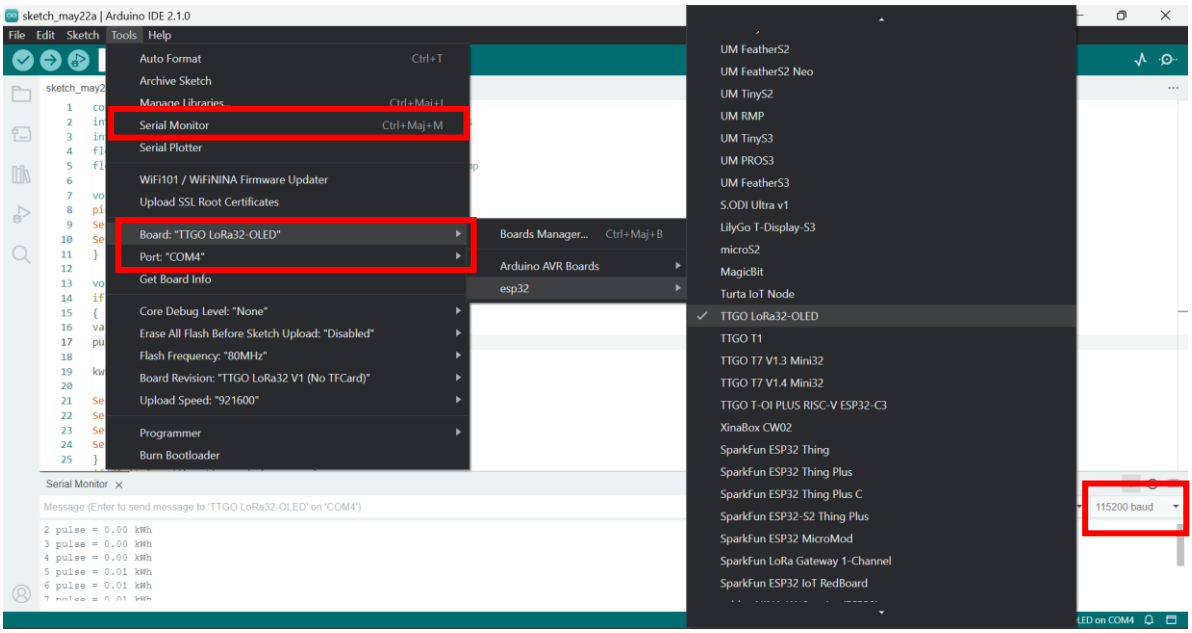


Figure 32 : Paramétrage du moniteur série pour un ESP32 dans Arduino IDE

La preuve de concept est validée lorsque l'on obtient des valeurs cohérentes avec ce que l'on doit mesurer (*Figure 33*). Pour AGRISONAR, je bougeais ma main au dessus du capteur, il répond bien au changement de distance et pour VINIPOWER, chaque pression du bouton poussoir est comptabilisée donc une impulsion le serait aussi. Les codes des programmes de test prototypes sous Arduino sont en *Annexe 12 et Annexe 13.*

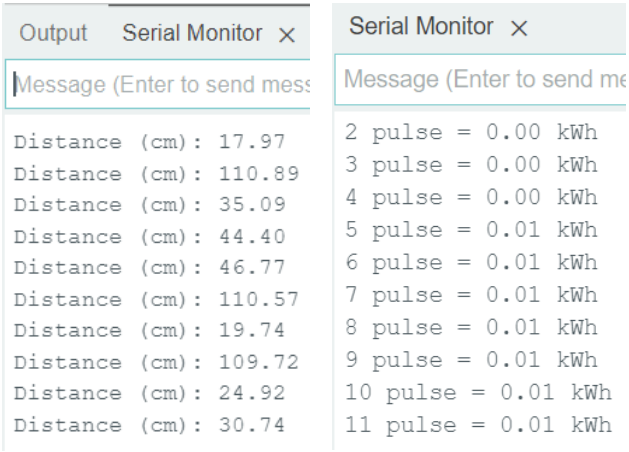


Figure 33 : Mesures cohérentes sur Arduino IDE pour AGRISONAR et VINIPOWER

L'inconvénient de l'Arduino IDE est que l'on doit installer des packages de cartes et de librairies pour chaque appareil où l'on souhaite faire tourner le programme.

C’est donc pour cela que la suite des projets se développe sous VS Code car Platform Io gère toute l’installation des différentes bibliothèques intégrées aux projets et il présente d’autres avantages.

IV.3.2 Moniteur Série VS Code

Un problème lors du passage d’Arduino IDE à VS Code (*Figure 34*) est que certaines fonctions sont spécifiques à Arduino. Il faut donc se débrouiller pour les recréer pour les intégrer dans le code du VITINODE. Le code principal est disponible est *Annexe 14* et le fichier à configurer est en *Annexe 15*, ce qui est non surligné est le code original, ce qui est surligné en vert, je l’ai rajouté pour AGRISONAR et ce qui est surligné en jaune a été rajouté pour VINIPOWER. Pour rappel, je travaille dans PlatformIO IDE qui est intégré à VS Code.

IV.3.2.1 *AGRISONAR*

Comme dit précédemment, une facilité sous VSCode est que l’on peut ajouter des librairies au projet. J’en ai trouvé une minimaliste pour module d'échographie sur Arduino et compatible avec les modules HC-SR04. C’est l’équivalent de mon programme Arduino (*Annexe 12*) mais en moitié moins de ligne. Pour l’intégrer au code, je dois l’ajouter au projet, l’appeler dans main.cpp puis j’ai copié le style d’écriture du code existant pour mettre en forme les données lues par Ultrasonic.h. Cela marche car je lisais la distance dans le moniteur série.

IV.3.2.2 *VINIPOWER*

J’étais bloqué sur ce projet car du fait que la consommation d’énergie est continue, il fallait ici monitorer continuellement tant que l’appareil triphasé est branché et consomme, contrairement à VITINODE et AGRISONAR où les mesures sont ponctuelles. Ce mode de mesure n’est donc pas compatible avec le DeepSleep qui ne relève pas de données durant son sommeil pour économiser de la batterie. Or comme l’alimentation sera prise sur le secteur, il n’y a pas d’autonomie énergétique à avoir ce qui fait que le deepsleep n’est plus nécessaire. J’ai alors repris la logique du code Arduino (*Annexe 13*) à savoir incrémenter une variable lorsqu’une impulsion est détectée. Cependant, dans la logique d’avoir un seul programme pour tous les capteurs, j’ai été bloqué pour désactiver le deepsleep uniquement pour VINIPOWER (VITI_TYPE 5 dans le code) mais pas pour les autres capteurs (VITI_TYPE 1,2,3,4). J’ai donc contacté Mr Pierrick SAILLANT, le développeur du code qui avait travaillé pour le Vitilab, qui est aussi le propriétaire du dépôt GitHub du projet. Il a fait des commits (modifications du code en ligne) selon mon cahier des charges et maintenant le deepsleep est bien désactivé uniquement pour VINIPOWER tout en monitorant en continu les impulsions reçues.

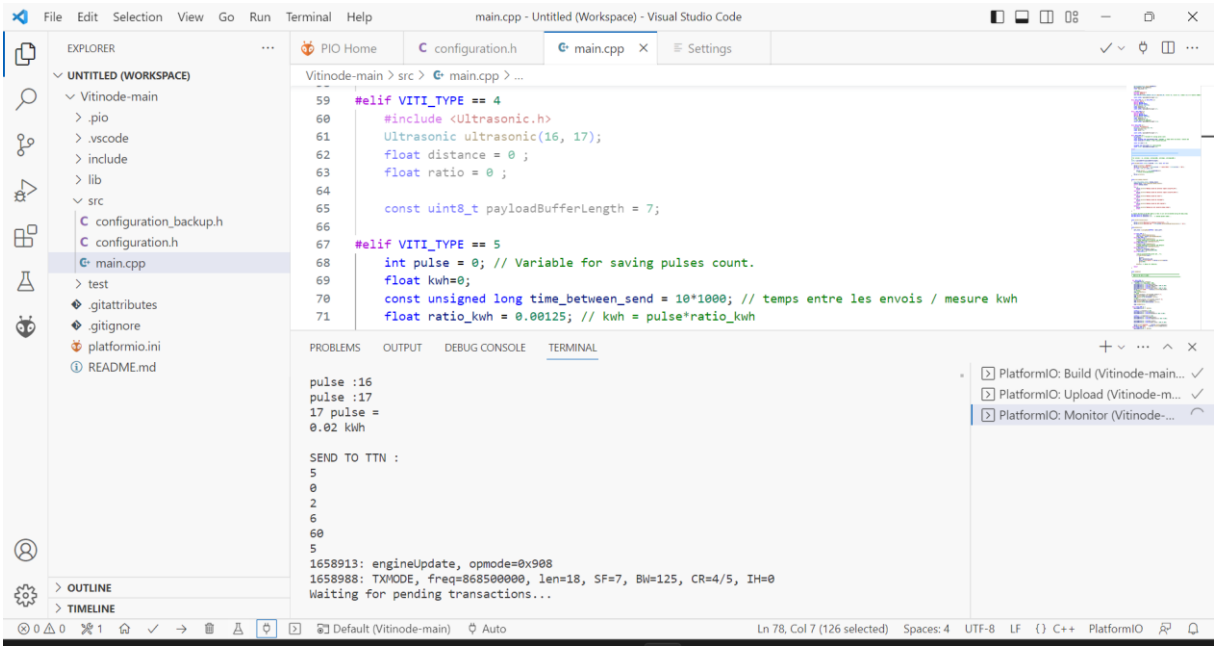


Figure 34 : Vue du projet sous PlatformIO

IV.3.3 TheThingsNetwork

La dernière étape du test des prototypes avant de passer à la version finale est que les données soient bien reçues sur TTN afin qu'elles correctement transmise sur la base de données. En même temps qu'il a modifié le deepsleep, Mr SAILLANT a modifié pour que les données soit envoyées à intervalle régulier modifiable pourr VINIPOWER car pour le reste, elles sont envoyées juste avant que l'ESP32 ne se mette en deepsleep.

Lorsque l'on crée un nœud sur TTN, il est rattaché à une application où l'on peut regrouper tous les capteurs du même type. Pour ne pas saboter l'application vini-météo où sont enregistrés tous les VITINODES des viticulteurs, j'ai créé une nouvelle application.

IV.3.3.1 *Uplink payload formatter*

Dans le code, les données sont mises sous forme de payload buffer (mémoire tampon des données utiles) qui est un paquet contenant une partie des informations. Il peut y en avoir un certain nombre (payload_buffer_length) suivant les données à envoyer. Le minimum ici sera 5 car il y en a un pour le type de capteur, deux pour la mesure car un pour la partie entière et un pour la partie décimale et encore deux pour le niveau de batterie, et chaque mesure en rajoutera deux. L'ESP32 met donc sous forme de payload buffer les données puis les envoie en LoRa, elles sont décryptées par la gateway et envoyés sur Internet et arrive « devant » TTN.

Pour que TTN puisse les reconnaitre, elles doivent être décodées par un uplink payload formatter, qui fonctionne comme un videur de boîte de nuit pour imager mes propos : si ce n'est pas bien présenté on ne rentre pas. C'est un formatter de type JavaScript, un code qui va comparer ce qu'on lui a autorisé avec les datas. S'il reconnaît, il exécute la suite de son code (*Annexe 16*) sinon il ne se passe rien. Les données bien formatées sont ensuite reconstruites,

la partie décimale retrouvant sa partie entière et prenant un nom compréhensible. Dans la même logique que les annexes 4 et 5, dans l'*Annexe 16,* ce qui est non surligné est le code original, en vert rajouté pour AGRISONAR et en jaune pour VINIPOWER.

IV.3.3.2 *Webhook*

Les données étant maintenant interprétables par un utilisateur, on veut maintenant pouvoir les stocker pour les traiter ultérieurement car les serveurs de TTN ne font que faire transiter l'information. J'ai donc créé une intégration Webhook au format JSON, qui va permettre de faire le lien entre l'API de TTN et celle de protechebdo. Pour se faire, je me suis référé au tutoriel disponible dans la rubrique Application du site protechebdo. Les données sont à présent acheminées du nœud jusque dans la base de données. (10. *IOTVSB*.)

V. Résultats

V.1 SOLAR VITINODE

Les tests de terrain sont plutôt concluants. Le panneau solaire n'arrive pas à recharger la batterie du VITINODE si elle est trop basse mais il subvient à la consommation journalière. Si on place le VITINODE avec une batterie chargée, le panneau fournit quasiment l'énergie en journée qui est consommée la nuit. Les tests montrent aussi que même en inter ceps, au milieu des feuilles, tant qu'il n'est pas totalement obstrué il arrive à produire de l'énergie. La *Figure 36* est une capture d'écran d'IOT VSB (l'autre nom de protechebdo), d'un SOLAR VITINODE. Ce qui est d'autant plus probant pour la validation du montage est qu'on demande ici un envoi de données toutes les 15 minutes, alors qu'il est conseillé aux utilisateurs de récupérer les datas une fois par heure. Les unités : la température est donnée en °C, l'humidité relative en pourcentage et la batterie en V. Ce qui nous intéresse ici est le voltage de la batterie car sa tension nominale étant de 3,7V, le fait qu'elle se maintienne à environ 3,6V est convaincant.

Figure 35 : SOLAR VITINODE dans la vigne

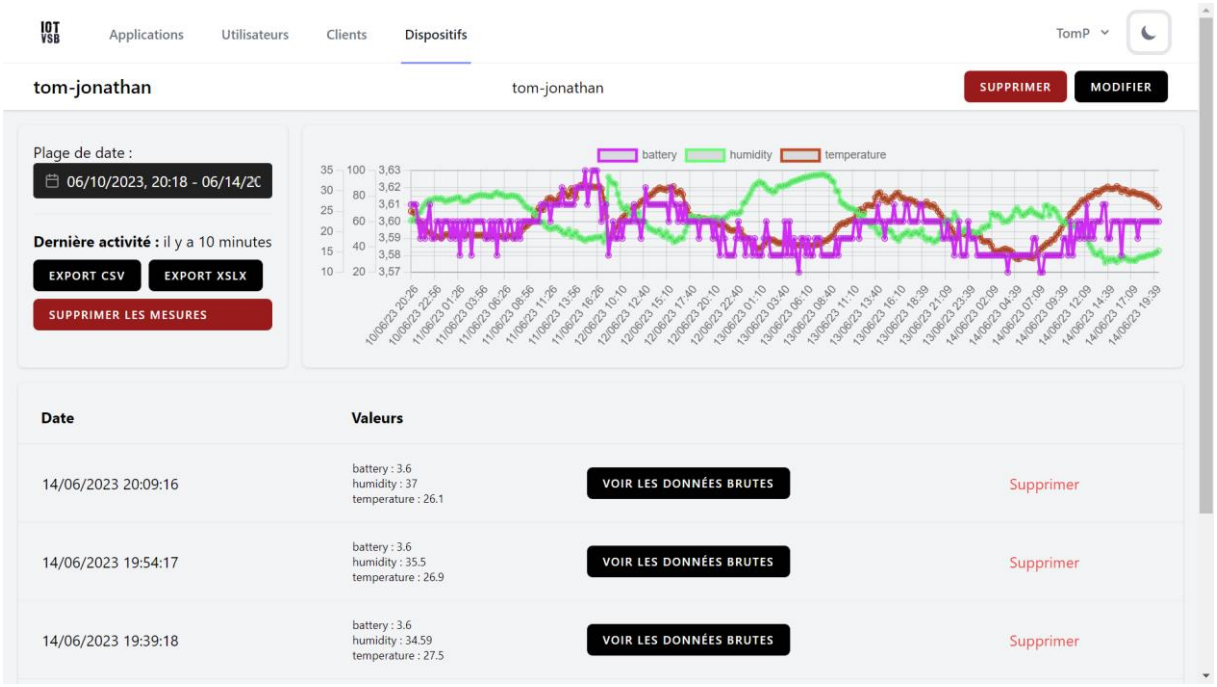


Figure 36 : Vue sur IoT VSB des données d'un SOLAR VITINODE

V.2 AGRISONAR

Je n'ai pas encore pu tester l'AGRISONAR dans les conditions réelles de son utilisation à savoir dans une cuve enterrée dans la ferme expérimentale de Jalogny, mais il n'y a pas de raison que les tests soient moins concluants que dans le laboratoire. Comparé au prototype, j'ai donc opéré quelques opérations afin de remplir le cahier de charges. Comme évoqué précédemment, le problème des objets connectés alimentés avec les petites batteries LiPo 3,7V est leur autonomie. Pour cela j'ai donc choisi une grosse pile 3,6V (non-rechargeable contrairement à une batterie), mais comme on ne demandera qu'une donnée par jour, on devrait approcher une année d'autonomie, ce qui est acceptable dans ce contexte. L'autre point important était l'étanchéité du dispositif, j'ai alors remplacé le HC-SR04 par le DYP-A02YY, qui est étanche jusqu'à la sortie des fils. Pour garantir un bon fonctionnement, il fallait donc protéger l'ensemble de l'objet car il aurait été trop compliqué de n'avoir que le capteur sous le couvercle et l'ESP32 au-dessus. J'ai pour cela mis le tout dans une petite boite hermétique , jointée entre le bouchon, et les sorties extérieures (*Figure 37*). AGRISONAR est ainsi autonome en énergie, en transmission des données et imperméable.

J'ai caché le niveau de batterie sur la *Figure 38* pour voir que le remplissage en % évolue inversement à la distance en cm, ce qui est normal car il est définit comme 100 - (distance*100)/400, 400 étant la hauteur de la cuve en cm.

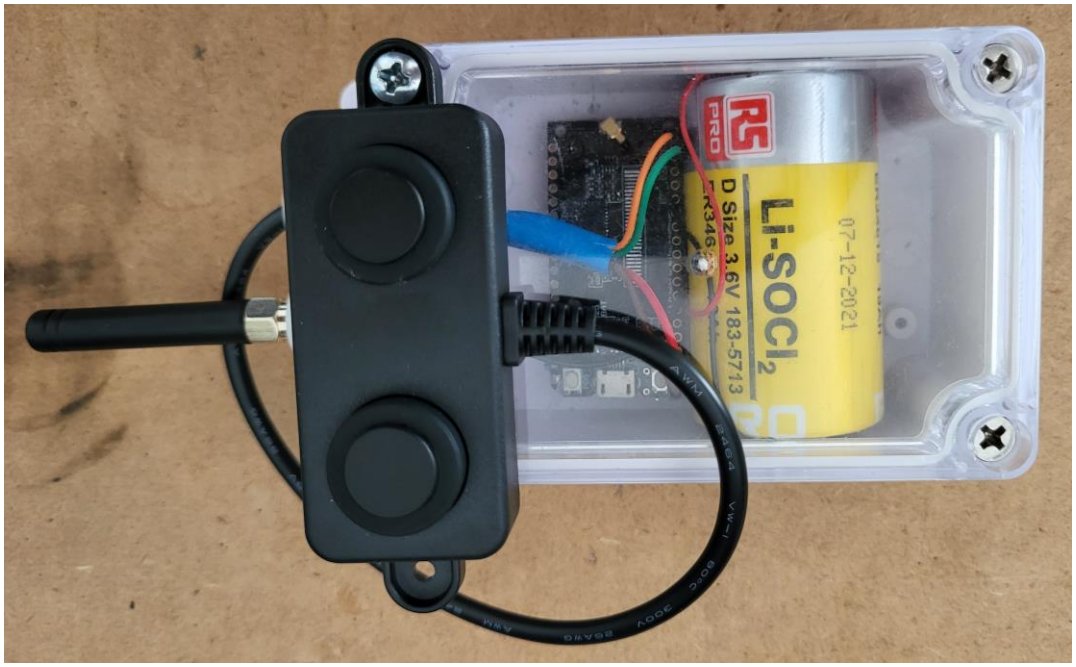


Figure 37 : AGRISONAR prêt à être installé

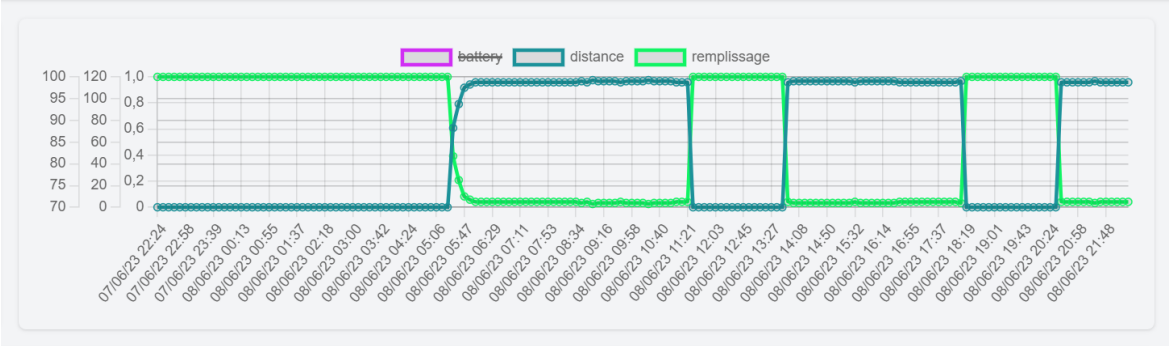


Figure 38 : Vue sur IoT VSB des données d'un AGRISONAR

V.3 VINIPOWER

Ce projet restera le moins achevé des trois car je n'ai pas eu le temps d'aller plus loin que le stade de prototype (*Figure 30*). En effet, je n'ai pas reçu les prises mâle et femelle en triphasé à 5 broches. Les tests ont donc été effectués avec des prises triphasée 4 broches, et donc sans neutre ce qui fait que l'alimentation à découpage ne marchait pas. Cependant, je continue à croire en la faisabilité du projet. Le graphique d'évolution de la consommation devrait ressembler à la *Figure 39* car la variable où sont comptées les impulsions est remise à 0 après chaque envoi sur TTN, afin de voir la consommation sur la période déterminée. La batterie est à un peu plus de 6V car c'est la tension du port USB.

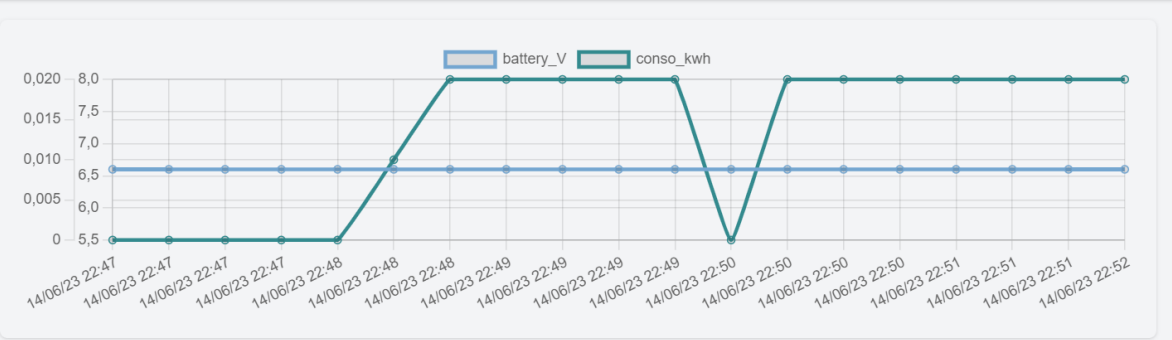


Figure 39 : Vue sur IoT VSB des données d'un VINIPOWER

VI. Conclusions et perspectives

VI.1 Techniques

Les capteurs développés sont susceptibles d'évoluer à la fois sur le plan esthétique et pratique car ils devront être testés sur plusieurs exploitations agricoles pour adapter les outils aux réalités du terrain, en prenant en compte des aspects tels que les capteurs, l'ergonomie et l'environnement technique. Des améliorations pourraient ainsi être apportées afin de les rendre plus simples d'utilisation.

Malgré cela, les prototypes actuels sont fonctionnels et remplissent leur rôle en collectant des données sur une interface avec une fréquence d'une donnée par heure. Seul le VINIPOWER ne sera pas opérationnel le jour de la soutenance, mais cela ne représente qu'une question de temps. En effet, j'ai réussi à négocier un contrat à durée déterminée d'un mois pour continuer à travailler sur ce type de missions.

Ces dispositifs permettent aux producteurs de prendre conscience des facteurs influents sur leur activité, sans nécessiter de travail supplémentaire. Cela leur permettra de mieux prendre en compte ces aléas et de les gérer de manière plus efficace.

VI.2 Personnelles

Au cours de ma première expérience professionnelle, j'ai pu réellement comprendre l'importance de la communication et du travail d'équipe dans la réalisation d'un projet commun. En combinant mes connaissances en électronique et en programmation avec le secteur viticole, qui m'était totalement étranger, j'ai pu constater l'impact de mon travail à l'interface de ces disciplines. Ce stage m'a également permis de développer mes compétences en gestion de projet en adoptant la méthode agile. J'ai appris à travailler rapidement sur un projet, à évaluer et à résoudre les problèmes rencontrés, tout en gérant plusieurs projets simultanément. J'ai eu la chance de bénéficier d'une certaine liberté dans ma façon de travailler, ce qui m'a été bénéfique pour progresser et atteindre les résultats finaux.
Je conclus ces 11 semaines de stage avec une grande satisfaction, ayant obtenu des résultats concrets dans un domaine qui correspond à mes intérêts. Cela renforce mon engagement dans

cette voie, et dès septembre 2023, je poursuivrai une formation ingénieur dans la filière en alternance Énergie et Conception des Installations de l'École Centrale de Lyon.

VII. Références bibliographiques

1. *Vinipôle Sud Bourgogne | Pôle de compétences pour la viticulture*. https://www.vinipole-sud-bourgogne.fr/ (Consulté le 01/06/2023)

2. Guillemin, P. (2018). État de l'art sur l'Internet des Objets en Europe - L'IdO (IoT) en Europe. *Techniques de l'Ingénieur*, paragraphe 2.2. https://doi.org/10.51257/a-v1-te8001 (Consulté le 01/06/2023)

3. JACQUOT, D. & Lycée Lucie AUBRAC. (2023). *Cours IOT*.

4. Contributeurs aux projets Wikimedia. (2023). LoRaWAN. fr.wikipedia.org. https://fr.wikipedia.org/wiki/LoRaWAN (Consulté le 03/06/2023)

5. Mocq, F. (2022). Connecter une Gateway LoRa RAK833 à TTN v3. *Framboise 314, le Raspberry Pi à la sauce française. . ..* https://www.framboise314.fr/connecter-une-gateway-lora-rak833-a-ttn-v3/ (Consulté le 05/06/2023)

6. The Things Network. . The Things Network. https://www.thethingsnetwork.org/ (Consulté le 07/06/2023)

7. Acetayls. . GitHub - acetayls/Vitinode. GitHub. https://github.com/acetayls/Vitinode (Consulté le 10/06/2023)

8. Bachetti, H. . *Alimentation par batterie + panneaux solaires*. https://riton-duino.blogspot.com/2020/12/alimentation-par-batterie-panneaux.html (Consulté le 11/06/2023)

9. Bachetti, H. . *Contrôle de Niveau d'Eau Connecté*. https://riton-duino.blogspot.com/2022/06/controle-de-niveau-deau-connecte.html (Consulté le 12/06/2023)

10. *IOTVSB*. . https://iot.protechebdo.fr/applications (Consulté le 14/06/2023)

VIII. Table des illustrations

IX. Table des Annexes

X. Annexes

Manufacturer Name	Gemtek
Manufacturer Part Number	TBMH100-868 / TBMH100-915
Seller SKU	GWGEM1010EU /GWGEM1010US
Product Attributes	Gateway
Gateway Type	Indoor Gateway
Width (cm)	8
Height (cm)	8
Depth (cm)	4
Lead-time if in stock (in working days)	5
Lead-time if backordered (in working days)	15
Power Supply	Electrical power
Sensor sensibility	-140/-135 dBm (EU/US)
Antenna	Integrated
LoRa Class	Class A, Class C
Operating Temperature (min to max)	0-40°C

Annexe 1 : Datasheet du Gateway LoRa GemTek

Caractéristiques du produit
Tout d'abord, la dimension longue: 28mm X largeur 12mm X hauteur 10mm
Deuxièmement, la puce principale: capteurs de température et d'humidité Aosun DHT22
Troisièmement, la tension de fonctionnement: DC 3.3-5.5 V
Quatrièmement, caractéristiques:
1, la plage de mesure d'humidité: 0 --- 100% RH
2, précision de mesure d'humidité: ± 2% hr
3, la plage de mesure de température:-40 --- 80 °
4, la précision de mesure de la température: ± 0.5 °
5, la tension de fonctionnement: DC5V couramment utilisé
6, sortie de signal numérique à bus unique, port de données série;

Annexe 2 : Datasheet du DHT22

Type: li-polymère

Capacité standard: 5800mAh

Tension standard: 3.7V

Courant de charge: 5800mAh à 4.2V

Tension de charge: 4.20 +/-0.03V

Temps de charge: environ 3 heures

Tension de coupure de décharge: 3.0V

Épaisseur: 10.0 +/-0.5mm

Largeur: 55.0 +/-1.0mm

Longueur: 75.0 +/-1.0mm

PCM: Protégez la carte de module de circuit (PCM) à l'intérieur, avec fil rouge (+) et noir (-).

Temps de recharge: >500Composition: Lithium polymère

Utilisation pour: GPS PSP Recorder Digital Camera MP3 MP4

Méthode de charge standard: 0,5C CC (courant constant) charge à 4.25V, puis CV (tension constante 4.2V) charge jusqu'à ce que le courant de charge soit inférieur ou égal à 0,05CLe courant de charge maximum: 1C

Courant de décharge maximum: 1C

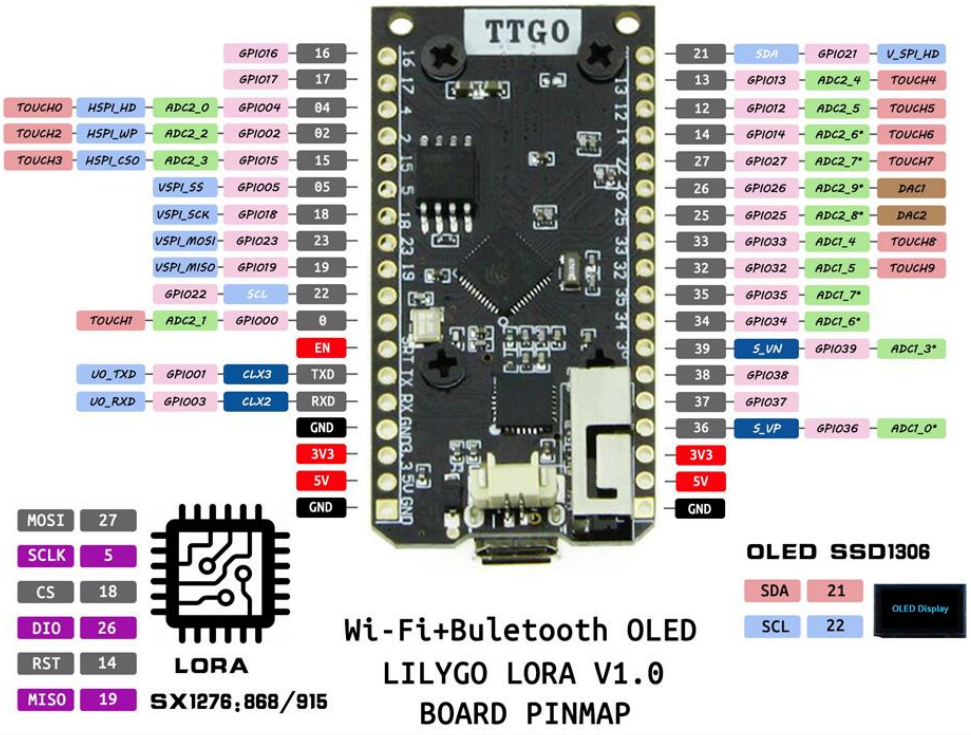
Tension de coupure de décharge: 2.75V

La température de décharge: -20 degrés C ~ + 60 degrés C

Temps de charge: 2-4hrs (standard); 2hrs (rapide)

Fonction de protection: surcharge, décharge excessive, surintensité, protection contre les courts-circuits, protection contre les surtempératures

Annexe 3 : Datasheet de la batterie



Annexe 4 : Câblage des GPIO de l'ESP32

Module de charge: Charge linéaire.
Courant: 1A réglable.
Précision de charge: 1.5%.
Tension d'entrée: 4.5V-5.5V.
Tension de charge complète: 4.2V ± 1%
Indicateur LED: le rouge est en charge; le vert est complètement chargé.
Interface d'entrée: micro USB/mini USB/type-c USB
Température de travail:-10 ° à 85 °.
Polarité inversée: NON

Annexe 5 : Datasheet du TP4056

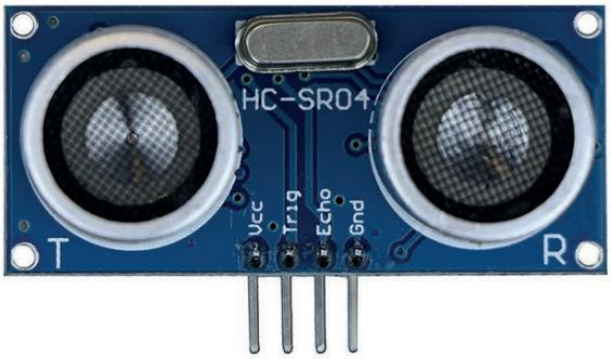
- Tension de charge maximale de 6,4 V.
- Tension de circuit ouvert 8,2 V
- Courant typique 100 mA

Tension typique 5,5 V.

Annexe 6 : Datasheet du panneau solaire

Kitronik

okdo
DESIGN THE WORLD



Ultrasonic Ranging Module

HC-SR04

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

- Using IO trigger for at least 10us high level signal
- The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

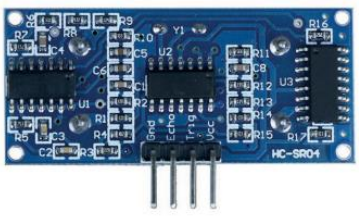
Test distance = (high level time×velocity of sound (340M/S) / 2

ELECTRIC PARAMETERS

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground



T: +44 (0) 115 970 4243 W: www.kitronik.co.uk E: support@kitronik.co.uk

Annexe 7 : Datasheet du HC-SR04

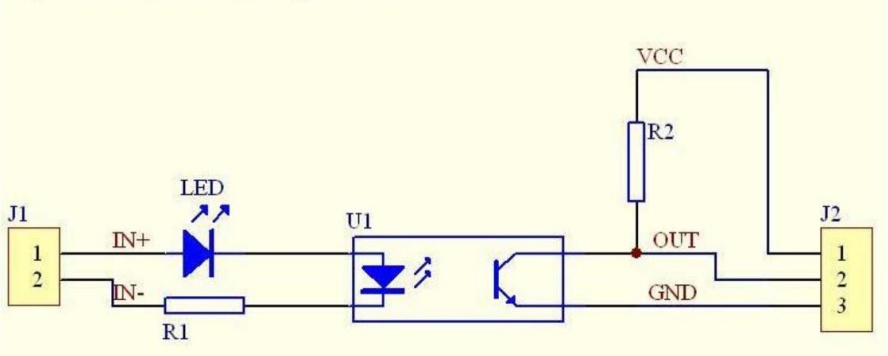
- Courant moyen: <8mA
- Tension de fonctionnement: 3.3V ~ 5V 、 5V-24V (Sortie RS485)
- Temps de réponse: 100ms
- Portée de détection (objet plat):**3-450cm**
- Sortie: URAT / PWM /RS485/Quantité de commutateur
- Température de fonctionnement: -15 ~ 60℃
- Angle de référence: 60 °
- Qualité étanche: IP67
- Précision de mesure: ±(1 + S * 0.3%)

Article de paramètre	Sortie PWM
Tension d'entrée	5
Moyenne de travail Courant	≤ 15
Valeur de crête-crête Courant	60
Distance de la zone aveugle	0 à 3
Gamme d'objets plats	3-420
Cycle de travail	Contrôlé
Mode de sortie	PWM
Mesure de la température ambiante Précision	±(1 + S * 0.5%)
La fréquence centrale De la sonde	40K ± 1.0K
ESD	± 4/± 8
Température Indemnisation	Non
Température de travail	-15 ~ 60
Température de stockage	-25 ~ 80
Humidité de travail	≤ 80%
Humidité de stockage	≤ 90%

Annexe 8 : Datasheet du DYP-A02YY

Annexe 9 : Datasheet du SDM530D

Tension du signal d'entrée: DC 3V-5V/12V/24V (en option)
Tension du signal de sortie: tension large adaptée aux DC1.8V-24V
Caractéristiques EL817:
La tension d'isolement est supérieure à 3000V
Courant d'entrée max 50MA
Courant de travail> 1MA
Courant de sortie maximum 50MA (le courant d'entrée doit être 15-20MA)
Température de fonctionnement-55 ~ 110 degrés.
Fréquence maximale du signal d'application: 5KHz



Annexe 10 : Datasheet du module d'isolation à optocoupleur

MEAN WELL 15W Single Output Switching Power Supply **RS-15 series**

■ Features :
* Universal AC input / Full range
* Protections: Short circuit / Overload / Over voltage / Over temperature
* Cooling by free air convection
* LED indicator for power on
* 100% full load burn-in test
* No load power consumption<0.5W
* All using 105°C long life electrolytic capacitors
* Withstand 300VAC surge input for 5 second
* High operating temperature up to 70°C
* Withstand 5G vibration test
* High efficiency, long life and high reliability
* 3 years warranty

SPECIFICATION

MODEL		RS-15-3.3	RS-15-5	RS-15-12	RS-15-15	RS-15-24	RS-15-48
OUTPUT	DC VOLTAGE	3.3V	5V	12V	15V	24V	48V
	RATED CURRENT	3A	3A	1.3A	1A	0.625A	0.313A
	CURRENT RANGE	0 ~ 3A	0 ~ 3A	0 ~ 1.3A	0 ~ 1A	0 ~ 0.625A	0 ~ 0.313A
	RATED POWER	9.9W	15W	15.6W	15W	15W	15.024W
	RIPPLE & NOISE (max.) Note.2	80mVp-p	80mVp-p	120mVp-p	120mVp-p	200mVp-p	200mVp-p
	VOLTAGE ADJ. RANGE	2.9 ~ 3.6V	4.75 ~ 5.5V	10.8 ~ 13.2V	13.5 ~ 16.5V	22 ~ 27.6V	43.2 ~ 52.8V
	VOLTAGE TOLERANCE Note.3	±3.0%	±2.0%	±1.0%	±1.0%	±1.0%	±1.0%
	LINE REGULATION Note.4	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%	±0.5%
	LOAD REGULATION Note.5	±2.0%	±1.5%	±0.5%	±0.5%	±0.5%	±0.5%
	SETUP, RISE TIME	1000ms, 30ms/230VAC 1000ms, 30ms/115VAC at full load					
	HOLD UP TIME (Typ.)	70ms/230VAC 12ms/115VAC at full load					
INPUT	VOLTAGE RANGE	85 ~ 264VAC 120 ~ 370VDC					
	FREQUENCY RANGE	47 ~ 63Hz					
	EFFICIENCY (Typ.)	72%	77%	81%	81%	82%	82%
	AC CURRENT (Typ.)	0.35A/115VAC 0.25A/230VAC					
	INRUSH CURRENT (Typ.)	COLD START 65A / 230VAC					
	LEAKAGE CURRENT	<2mA / 240VAC					
PROTECTION	OVERLOAD	Above 105% rated output power Protection type : Hiccup mode, recovers automatically after fault condition is removed					
	OVER VOLTAGE	3.8 ~ 4.45V	5.75 ~ 6.75V	13.8 ~ 16.2V	17.25 ~ 20.25V	28.4 ~ 32.4V	55.2 ~ 64.8V
		Protection type : Shut off o/p voltage, clamping by zener diode					
	OVER TEMPERATURE	Shut down o/p voltage, recovers automatically after temperature goes down					
ENVIRONMENT	WORKING TEMP.	-20 ~ +70°C (Refer to "Derating Curve")					
	WORKING HUMIDITY	20 ~ 90% RH non-condensing					
	STORAGE TEMP., HUMIDITY	-40 ~ +85°C, 10 ~ 95% RH					
	TEMP. COEFFICIENT	±0.03%/°C (0 ~ 50°C)					
	VIBRATION	10 ~ 500Hz, 5G 10min./1cycle, period for 60min. each along X, Y, Z axes					
SAFETY & EMC (Note 6)	SAFETY STANDARDS	UL62368-1, TUV EN62368-1, EAC TP TC 004, CCC GB4943.1, BSMI CNS14336-1 approved					
	WITHSTAND VOLTAGE	I/P-O/P:3KVAC I/P-FG:2KVAC O/P-FG:0.5KVAC					
	ISOLATION RESISTANCE	I/P-O/P, I/P-FG, O/P-FG:100M Ohms / 500VDC / 25°C/ 70% RH					
	EMC EMISSION	Compliance to EN55032 (CISPR32) Class B, EN61000-3-2,-3,GB9254 class B,GB17625.1, EAC TP TC 020, CNS13438 Class B					
	EMC IMMUNITY	Compliance to EN61000-4-2, 3, 4, 5, 6, 8,11, EN55024, EN61000-6-1, light industry level, criteriaA, EAC TP TC 020					
OTHERS	MTBF	1608.8Khrs min. MIL-HDBK-217F (25°C)					
	DIMENSION	62.5*51*28mm (L*W*H)					
	PACKING	0.13Kg; 108pcs/15Kg/0.71CUFT					
NOTE	1. All parameters NOT specially mentioned are measured at 230VAC input, rated load and 25°C of ambient temperature. 2. Ripple & noise are measured at 20MHz of bandwidth by using a 12" twisted pair-wire terminated with a 0.1uf & 47uf parallel capacitor. 3. Tolerance : includes set up tolerance, line regulation and load regulation. 4. Line regulation is measured from low line to high line at rated load. 5. Load regulation is measured from 0% to 100% rated load. 6. The power supply is considered a component which will be installed into a final equipment. All the EMC tests are been executed by mounting the unit on a 230mm*230mm metal plate with 1mm of thickness. The final equipment must be re-confirmed that it still meets EMC directives. For guidance on how to perform these EMC tests, please refer to "EMI testing of component power supplies." (as available on http://www.meanwell.com) 7. The ambient temperature derating of 3.5°C/1000m with fanless models and of 5°C/1000m with fan models for operating altitude higher than 2000m(6500ft).						

File Name:RS-15-SPEC 2019-07-03

Annexe 11 : Datasheet du MeanWell RS-15-5

```
#include <Arduino.h>  // permet d'utiliser la bibliothèque Arduino
const int trigPin = 16; //définir les pin de lecture
const int echoPin = 17;
#define SOUND_SPEED 0.034 //définir la vitesse du son en cm/µS

long duration;  //définir les variables
float distanceCm;

void setup() {
  Serial.begin(115200); // début communication série
  pinMode(trigPin, OUTPUT); // mets la broche TRIG en GPIO sortie
  pinMode(echoPin, INPUT); // mets la broche ECHO en GPIO entrée
}

void loop() {
  digitalWrite(trigPin, LOW);  // nettoie la broche
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);  // mets la broche TRIG à l'état haut pendant 10µs
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);  // lit la broche écho et retourne durée de
l'impulsion
  distanceCm = duration * SOUND_SPEED/2;  // Calcule la distance

  Serial.print("Distance (cm): ");  // écrire dans le moniteur série
  Serial.println(distanceCm);
  delay(1000);
}
```

Annexe 12 : Code Arduino pour AGRISONAR

```
const int input = 16; // Broche où est lue l'impulsion.
int pulse = 0; // Variable où est stockée le nombre d'impulsions
int var = 0;
float kwh=0;
float pulse_rate=0.00125; //car 800 imp/kwh soit 1/800kwh par imp

void setup(){
pinMode(input, INPUT);  //définir broche comme GPIO d'entrée
Serial.begin(115200); // vitesse de communication en bauds
Serial.println(F("No pulses yet...")); // Message à envoyer au début
}

void loop(){
if(digitalRead(input) > var)  // si on détecte une impulsion
{
var = 1;
pulse++;  //on incrémente la variable

kwh = pulse*pulse_rate; // on calcule l'équivalent en puissance

Serial.print(pulse);  //écrire dans le moniteur série
Serial.print(" pulse = ");
Serial.print(kwh);
Serial.print(" kWh \n");
}
if(digitalRead(input) == 0) {var = 0;}
delay(1); // Délai de stabilité
}
```

Annexe 13 : Code Arduino pour VINIPOWER

```
#include <Arduino.h>
#include <TTN_esp32.h>

#include "TTN_CayenneLPP.h"
#include "configuration.h"

TTN_esp32 ttn ;
TTN_CayenneLPP lpp;
const uint8_t fport = 10;

/******
*
*  SENSOR DECLARATION
*
 ******/
#include <Adafruit_Sensor.h>
#include <Adafruit_I2CDevice.h>

#define BATPIN 36
float batt_level = 0;


#if VITI_TYPE == 1
    #include <OneWire.h>
    #include <DallasTemperature.h>
    #define ONE_WIRE_BUS 17
    OneWire oneWire(ONE_WIRE_BUS);
    DallasTemperature sensors(&oneWire);
    DeviceAddress deviceAddress;
    float temp_sonde = 0;

    //Display
    #include <U8g2lib.h>
    #include <Wire.h>
    U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /*rst*/ 16, /*scl*/ 22,
/*sda*/ 21);///* reset=*/ U8X8_PIN_NONE, /* clock=*/ 16, /* data=*/ 17);

    const uint8_t payloadBufferLength = 5;

#elif VITI_TYPE == 2 || VITI_TYPE == 3
    #include <DHT.h>
    #define DHTPIN 16
    #define DHTTYPE DHT22
    DHT dht(DHTPIN, DHTTYPE);
    float humidity = 0;
    float temperature = 0;
```

```
    const uint8_t payloadBufferLength = 7;

#elif  VITI_TYPE == 3
    #include <DHT.h>
    #define DHTPIN 16
    #define DHTTYPE DHT22
    DHT dht(DHTPIN, DHTTYPE);
    float humidity = 0;
    float temperature = 0;
    // avec pluvio et capacitif
    const uint8_t payloadBufferLength = 11;


#elif VITI_TYPE == 4
    #include <Ultrasonic.h>
    Ultrasonic ultrasonic(16, 17);
    float distance = 0 ;
    float ratio = 0 ;

    const uint8_t payloadBufferLength = 7;

#elif VITI_TYPE == 5
    int pulse = 0; // Variable for saving pulses count.
    float kwh=0;
    const unsigned long time_between_send = 10*1000; // temps entre les envois
/ mesure kwh
    float ratio_kwh = 0.00125; // kwh = pulse*ratio_kwh

    const int input = 16;

    unsigned long last_send = 0; //milliseconds
    const uint8_t payloadBufferLength = 5;

#endif


/* Payload buffer is bytes send to ttn and manage by ttn payload formater.
*
* For vitibox : [1, int(sonde), sonde/100]
*
* For vititeo : [2, int(hum), int(hum/100), int(temp), int(temp/100) ]
*/
uint8_t payloadBuffer[payloadBufferLength];

void message(const uint8_t* payload, size_t size, int rssi)
{
```

```
    Serial.println("-- MESSAGE");
    Serial.print("Received " + String(size) + " bytes RSSI=" + String(rssi) +
"db");
    for (int i = 0; i < size; i++)
    {
        Serial.print(" " + String(payload[i]));
        // Serial.write(payload[i]);
    }
    Serial.println();
}


void print_wakeup_reason()
{
    esp_sleep_wakeup_cause_t wakeup_reason;
    wakeup_reason = esp_sleep_get_wakeup_cause();
    switch (wakeup_reason)
    {
    case 1:
        Serial.println("Wakeup caused by external signal using RTC_IO");
        break;
    case 2:
        Serial.println("Wakeup caused by external signal using RTC_CNTL");
        break;
    case 3:
        Serial.println("Wakeup caused by timer");
        break;
    case 4:
        Serial.println("Wakeup caused by touchpad");
        break;
    case 5:
        Serial.println("Wakeup caused by ULP program");
        break;
    default:
        Serial.println("Wakeup was not caused by deep sleep");
        break;
    }
}

// stores the data on the RTC memory so that it will not be deleted during the
deep sleep
RTC_DATA_ATTR int bootCount = 0;
RTC_DATA_ATTR int pckCounter = 0;   // sending packet number...


void waitForTransactions()
{
```

```
    Serial.println("Waiting for pending transactions... ");
    Serial.println("Waiting took " + String(ttn.waitForPendingTransactions())
+ "ms");
}

void getSensors()
{
    batt_level = analogRead(BATPIN) * batt_coeff;


    #if VITI_TYPE == 1
        sensors.requestTemperatures();
        temp_sonde = sensors.getTempCByIndex(0);
    #elif VITI_TYPE == 2
        humidity = dht.readHumidity();
        // Read temperature as Celsius (the default)
        temperature = dht.readTemperature();
    #elif VITI_TYPE == 3
        humidity = dht.readHumidity();
        // Read temperature as Celsius (the default)
        temperature = dht.readTemperature();
    #elif VITI_TYPE == 4
        distance = ultrasonic.read();
        ratio = 100 - (distance*100)/400;
    #elif VITI_TYPE == 5

        //Serial.println(F("No pulses yet..."));
        if(!digitalRead(input))
            {
            pulse++;
            kwh = pulse*ratio_kwh;
            Serial.print("pulse :");Serial.println(pulse);
            delay(500);
            }
        delay(1); // Delay for stability.

    #endif
}


void sendData()
{
/**********************************************************************
*  HERE GO THE DATA TO SEND !
 **********************************************************************/
```

```
  #if VITI_TYPE == 1
    payloadBuffer[0] = byte(1);
//    uint32_t s = temp_sonde * 100;
    payloadBuffer[1] = int(temp_sonde);
    payloadBuffer[2] = int(int(temp_sonde * 100) % 100);
    payloadBuffer[3] = int(batt_level);
    payloadBuffer[4] = int(int(batt_level * 100) % 100);
    Serial.print("sonde :  ");Serial.println(temp_sonde);
    // display
    u8g2.clearBuffer();
    // sonde
    String to_display = String(temp_sonde)+" C";
    u8g2.setFont(u8g2_font_ncenB24_tr);
    u8g2.drawStr(5,48,to_display.c_str());
    // batt
    String batt_display = String(batt_level)+" V";
    u8g2.setFont(u8g2_font_ncenB10_tr);
    u8g2.drawStr(60,64,batt_display.c_str());

    u8g2.sendBuffer();

  #elif VITI_TYPE == 2
    payloadBuffer[0] =  byte(2);

    uint32_t h = humidity * 100;
    payloadBuffer[1] = int(humidity);
    payloadBuffer[2] = int(int(humidity * 100) % 100);

    uint32_t t = temperature * 100;
    payloadBuffer[3] = int(temperature);
    payloadBuffer[4] = int(int(temperature * 100) % 100);

    payloadBuffer[5] = int(batt_level);
    payloadBuffer[6] = int(int(batt_level * 100) % 100);

    Serial.print("humidity : ");Serial.println(humidity);
    Serial.print("temp :  ");Serial.println(temperature);
  #elif VITI_TYPE == 3
    payloadBuffer[0] =  byte(3);

    uint32_t h = humidity * 100;
    payloadBuffer[1] = int(humidity);
    payloadBuffer[2] = int(int(humidity * 100) % 100);

    uint32_t t = temperature * 100;
```

```
    payloadBuffer[3] = int(temperature);
    payloadBuffer[4] = int(int(temperature * 100) % 100);

    payloadBuffer[5] = int(batt_level);
    payloadBuffer[6] = int(int(batt_level * 100) % 100);

    Serial.print("humidity : ");Serial.println(humidity);
    Serial.print("temp :  ");Serial.println(temperature);

  #elif VITI_TYPE == 4
    payloadBuffer[0] =  byte(4);

    uint32_t d = distance * 100;
    payloadBuffer[1] = int(distance);
    payloadBuffer[2] = int(int(distance * 100) % 100);

    payloadBuffer[3] = int(ratio);
    payloadBuffer[4] = int(int(ratio * 100) % 100);

    payloadBuffer[5] = int(batt_level);
    payloadBuffer[6] = int(int(batt_level * 100) % 100);

    Serial.print("Distance in CM: "); Serial.println(distance);
    Serial.print("Remplissage en % : "); Serial.println(ratio);

  #elif VITI_TYPE == 5
    payloadBuffer[0] =  byte(5);

    payloadBuffer[1] = int(kwh);
    payloadBuffer[2] = int(int(kwh * 100) % 100);

    payloadBuffer[3] = int(batt_level);
    payloadBuffer[4] = int(int(batt_level * 100) % 100);

    Serial.print(pulse);Serial.println(" pulse = ");
    Serial.print(kwh);Serial.println(" kWh \n");
  #endif

  Serial.println("SEND TO TTN : ");
  for (int i = 0; i < sizeof(payloadBuffer); i++) {
    Serial.println(payloadBuffer[i]);
  }
  Serial.println(payloadBufferLength);
  ttn.sendBytes(payloadBuffer, payloadBufferLength, fport);
}
```

```
void setup()
{
    Serial.begin(115200);
    delay(1000);
    Serial.println("Starting");

    #if VITI_TYPE == 1
        sensors.begin();

        Wire.begin(21, 22);
        u8g2.begin();
        u8g2.clearBuffer();
        u8g2.setFont(u8g2_font_ncenB10_tr);
        u8g2.drawStr(5,16,"Starting ...");
        u8g2.sendBuffer();
//        u8g2.drawStr(5,32,"Not Connected yet");
    #elif VITI_TYPE == 2 || VITI_TYPE == 3
        dht.begin();
    #elif VITI_TYPE == 5
        pinMode(input, INPUT_PULLDOWN);
    #endif



    // Print the wakeup reason for ESP32
    //print_wakeup_reason();
    ttn.begin();
    // Declare callback function for handling downlink messages from server
    ttn.onMessage(message);
    // Join the network
    ttn.join(devEui, appEui, appKey);
    Serial.print("Joining TTN ");
    while (!ttn.isJoined())
    {
        Serial.print(".");
        delay(500);
    }
    Serial.println("\njoined!");


    #if VITI_TYPE != 5 // Les autres que 5 utilisent le deep sleep, pour 5,
voir dans loop()
        // Make sure any pending transactions are handled first
        waitForTransactions();
        //
```

```
        getSensors();
        // Send our data
        sendData();
        // Make sure our transactions is handled before going to sleep
        waitForTransactions();


        // Sleep time in micro seconds so multiply by 1000000
        esp_sleep_enable_timer_wakeup(secs_between_send * 1000000);

        // Go to sleep now
        Serial.println("Going to sleep!");
        esp_deep_sleep_start();
        // Everything beyond this point will never be called
    #endif
}



void loop()
{
    #if VITI_TYPE == 5
        // on regarde en permanence si on à une impulsion.
        getSensors();

        // tout les time_between_send Milliseconds, on envoi à TTN (et on
remet à zero de compteur d'impulsion)
        if ((millis() - last_send) > time_between_send) {
            // Send our data
            sendData();
            // Make sure our transactions is handled before going to sleep
            waitForTransactions();
            // Remise à Zéro du compteur d'impulsion
            pulse = 0;

            last_send = millis();
        }
    #endif
    // for other, go to deep sleep --> Never called
}
```

Annexe 14 : Programme principal (main.cpp) sous VSCODE de l'ensemble des projets

```
/***************************************
*
* CHANGER LES VALEURS SUIVANTE selon la configuration !!!!
*
***************************************/

// identifiant du node TTN
const char* devEui = "70B3D57ED0056DF3"; // Change to TTN Device EUI
const char* appEui = "0000000000000000"; // Change to TTN Application EUI
const char* appKey = "A175BEBB1955490D7E41FB3976966FFC"; // Change to TTN
Application Key

// selection station meteo (uncomment le type en supprimant // et commenter
les autres avec //)
//#define VITI_TYPE 1 //si vitbox : Sonde DS18B20 --> 1;
//#define VITI_TYPE 2 // si station météo : sonde DHT22 --> 2 ;
//#define VITI_TYPE 3 // si station météo : sonde capa + DHT22 + pluvio--> 3 ;
//#define VITI_TYPE 4 // si niveau eau : sonde HCSR04 --> 4
#define VITI_TYPE 2 // si compteur énergie -> 5

// Temps entre les mesures (en secondes)
const unsigned long secs_between_send = 60*60;

// Coefficient de la batterie (en fonction des valeurs des résistances, mesure
sur le terrain)
const double batt_coeff = 0.001611721612; // (3.3V * R1/(R1+R2) / 4095 ) // si
les résistances sont identiques : 0.001074481

/***************************************
*
* FIN des valeurs à modifier !
*
***************************************/
```

Annexe 15 : Programme à modifier (configuration.h) sous VSCODE de l'ensemble des projets

```
function decodeUplink(input) {
   var data = {};
   var warnings = [];

   if (input.fPort == 10) {
      if (input.bytes[0] == 1) {
        data.sonde = input.bytes[1] + input.bytes[2]/100;
        data.battery_V = input.bytes[3] + input.bytes[4]/100;
      }
      else if (input.bytes[0] == 2) {
        //data.counter = (input.bytes[0] << 8) + input.bytes[1];
        data.temperature_Celsius = input.bytes[3] + input.bytes[4]/100;
        data.humidity_pourcent = input.bytes[1] + input.bytes[2]/100;
        data.battery_V = input.bytes[5] + input.bytes[6]/100;
      }
      else if (input.bytes[0] == 4) {
        data.distance_cm = input.bytes[1] + input.bytes[2]/100;
        data.remplissage_pourcent = input.bytes[3] + input.bytes[4]/100;
        data.battery_V = input.bytes[5] + input.bytes[6]/100;
      }
      else if (input.bytes[0] == 5) {
        data.conso_kwh = input.bytes[1] + input.bytes[2]/100;
        data.battery_V = input.bytes[3] + input.bytes[4]/100;
      }
   }
   else {
      warnings.push("Unsupported fPort");
   }
   return {
      data: data,
      warnings: warnings
   };
}
```

Annexe 16 : code du Uplink Payload Formatter dans TTN