Smart Greenhouse

Projet d'ingénierie

2021 - 2022

Guillaume AUZELOUX

Lella POURCEL

Jérémy DJIAN

Victor LAVAUD

Mateo BORGOTTI

Théo FERRER BRUNET

Remerciements

Avant de débuter ce rapport, nous tenons à remercier :

Monsieur Jérôme Ferrari, pour son aide en tant que porteur de projet, qui nous a accompagné tout au long du projet et qui nous a aidé à surmonter nos problèmes.

Monsieur Alexis Derbey, notre référent hygiène, sécurité et technique, pour sa disponibilité pour la signature des annexes.

Aux personnels du FabLab pour leur aide précieuse, et leur temps accordé pour nous former de manière pédagogue et amicale.

Monsieur Vincent Imard, ingénieur au sein du G2Elab, pour son expertise apportée lors de l'étude et la conception du programme utilisant l'apprentissage par renforcement, soit le Q-learning.

Monsieur Christophe Rousseau, ingénieur d'études responsable technique des plateformes Observation de l'Eau dans le milieu naturel (Obs-Eau) et Géomécanique, Structures, Ouvrages, pour son aide concernant le prêt de capteurs de qualité de l'eau.

Sommaire

I. Présentation du Projet

I.1.Principe

Le projet Smart Greenhouse a pour objectif d'améliorer l'autonomie et la gestion d'une serre installée sur le toit de l'ENSE3. La serre, présente depuis 2016, a été fournie par l'entreprise MyFood. Cette entreprise propose de nombreux types de serre urbaine, qui peuvent donc s'implanter sur les toits, terrasses, balcons… La serre est initialement très peu automatisée. Ainsi, le laboratoire G2ELab, laboratoire de recherche en Génie Électrique situé au sein de l'école ENSE3, est le porteur de ce projet depuis l'implantation de la serre. Il missionne donc les groupes d'étudiants d'optimiser la gestion de la serre en utilisant des systèmes automatiques comme nous le verrons à la suite de ce rapport.

Le principe de fonctionnement de la serre est basé sur l'aquaponie. Ce principe vise à allier l'hydroponie, consistant à faire pousser des plantes hors sols en leur donnant les nutriments nécessaires grâce à l'eau riche en ces derniers, et l'aquaculture, l'élevage de poissons. Il suffit de nourrir les poissons qui vont alors produire des déjections dans l'eau de leur bassin, et cette eau sera alors purifiée par les plantes qui se nourrissent des nitrates présents dans les déjections des poissons. Tout ceci fonctionne en circuit fermé comme le montre la figure 1.

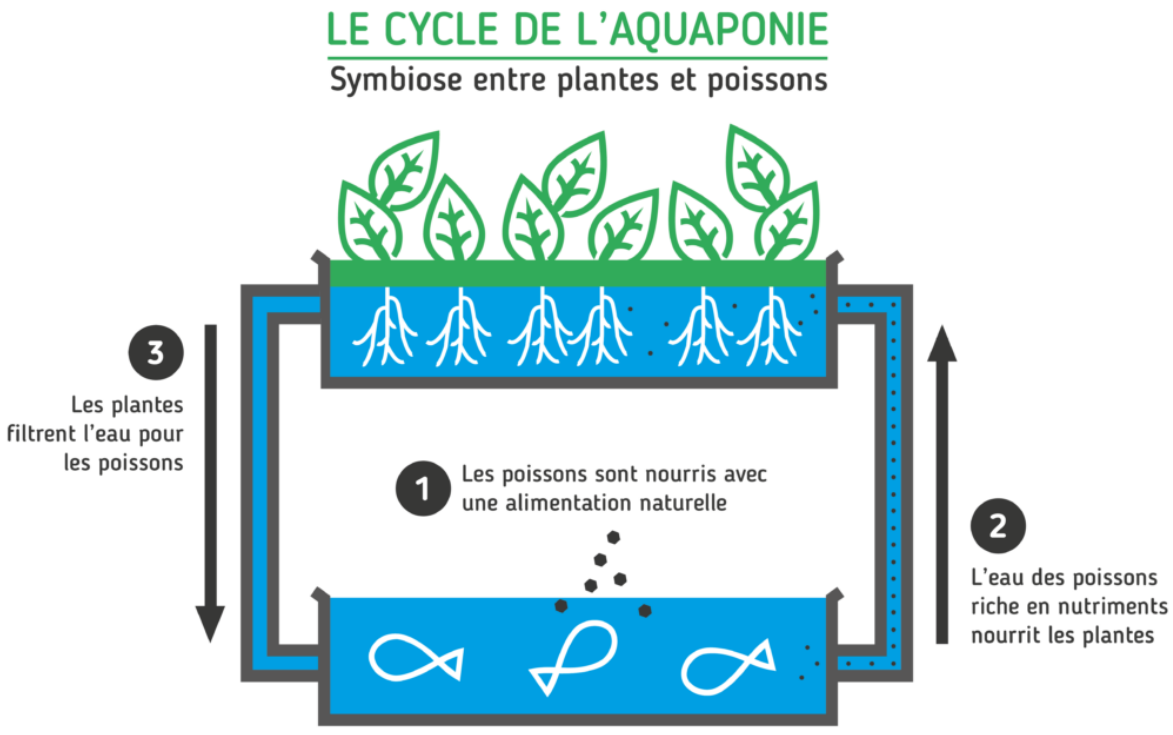


Figure 1 - Le cycle de l'aquaponie.

La serre est composée de différents éléments :

- Tout d'abord, à l'intérieur, se trouve un bassin de 2 m² et 50 cm de haut où vivent deux poissons rouges. Une pompe DC Jecod DCS-4000 permet de transporter l'eau du bassin vers les supports verticaux où grandissent les plantes.

- La serre est alimentée en énergie par des panneaux solaires présents sur son toit, des batteries ainsi que le réseau électrique. Un onduleur hybride présent dans l'armoire électrique choisit qui du réseau, des panneaux solaires ou de la batterie alimente les charges présentes dans la serre. Cet onduleur hybride suit les paramètres que nous lui imposons et que nous évoquerons par la suite.

- La serre est équipée de capteurs de production électrique pour les panneaux solaires, la batterie ou le réseau électrique. Elle compte aussi des capteurs de qualité de vie dans la serre (température de l'eau et de l'air, taux d'humidité, pH, etc.) ainsi qu'une fenêtre équipée d'un bras mécanique. Ces données sont accessibles sur le site Grafana (lien [1]) réalisé par le G2E Lab et l'ENSE3.

- La serre est également équipée de gouttières et d'un bac de récupération de pluie, nous détaillerons son intérêt plus tard.

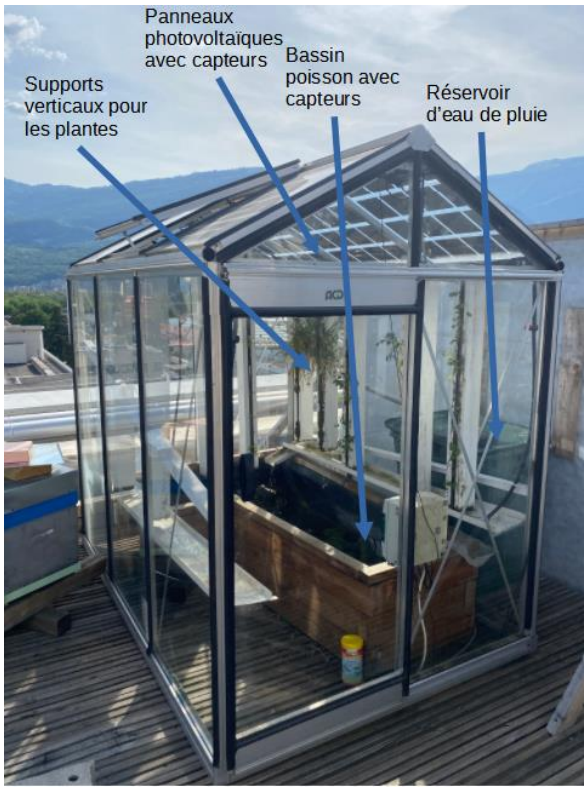


Figure 2 - Différents éléments de la serre.

I.2 Cahier des charges

Le projet s'articule sur deux axes bien précis :

Monitoring et entretien du vivant

Pour cette partie, le premier travail est d'assurer la pérennité des êtres vivants : ainsi, il nous est nécessaire d'entretenir la serre. Nous devons en effet assurer le nettoyage du bassin, du filtre et des supports verticaux pour les plantes, mais aussi maintenir le niveau d'eau du bassin. Ensuite, on se propose de planter de nouveaux légumes dans la serre et de tout mettre en œuvre pour assurer leur croissance. Si possible, nous pouvons également ajouter un ou des autres poissons dans le bassin.

Enfin, pour encore améliorer la vie des poissons du bassin, nous sommes missionnés d'ajouter un système de remplissage automatique d'eau car le niveau d'eau diminue du fait de la consommation des plantes, mais aussi fortement à cause de l'évaporation. Si nous avons le temps, nous pouvons également créer une application pour téléphone qui affichera en temps réel les données de la serre et indiquera à son utilisateur si ces données sont conformes ou si un problème doit être réglé.

Optimisation de l'énergie

Pour cette partie, nous allons devoir étudier le fonctionnement des panneaux solaires, des batteries et de l'onduleur.

Premièrement, nous devons d'abord réaliser un inventaire des capteurs présents dans le bassin à l'intérieur de la serre. Dans le cas où les capteurs seraient déficients, nous devons également les remplacer ou alors les nettoyer et refaire leur étalonnage.

Dans un second temps, on nous propose de réaliser des tests sur l'onduleur : plusieurs modes et paramètres peuvent être choisis et nous devons les tester et justifier les choix de ces derniers, dans le but d'optimiser un maximum la consommation des charges et la production des panneaux solaires.

Pour finir, on cherche à savoir si la source de puissance fournie par les panneaux solaires est suffisante pour alimenter la serre et être totalement indépendante du réseau électrique.

II. Travail réalisé

II.1 État des lieux des capteurs

II.1.a Capteurs pour le Vivant

Les capteurs pour le vivant de la serre sont les suivants :

- pH,
- Potentiel d'oxydo-réduction, (ORP)
- Conductivité, (noté EC sur Grafana)
- Oxygénation de l'eau,
- Température de l'eau,
- Température de l'air,
- Taux d'humidité de la serre.

Ceux-ci nous permettent de connaître en temps réel la qualité de l'eau des poissons, ainsi que l'environnement au sein même de la serre.

Lors de la découverte du projet, nous avons observé que le capteur de température de la serre, qui captait aussi son taux d'humidité, était défectueux. On remarque sur la figure 3, que les mesures du taux d'humidité dans la serre étaient régulièrement absentes. On retrouvait le même comportement sur la mesure de température de l'air dans la serre.

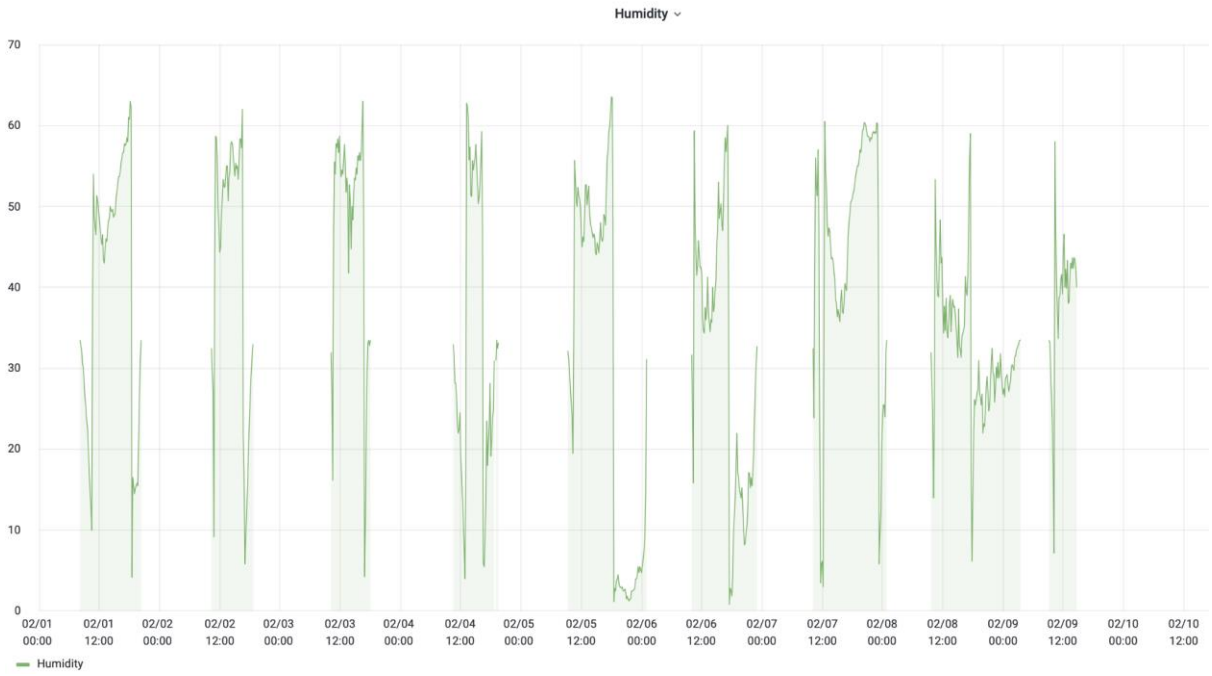


Figure 3 : Mesure du taux d'humidité dans la serre avant le changement du capteur.

Nous l'avons alors remplacé et il fonctionne à présent très bien. Ce capteur se doit être constamment fonctionnel, car il sert également au pilotage du moteur de la fenêtre : si le

capteur de température mesure 27°C ou plus, le programme ouvre la fenêtre présente sur le toit de la serre ; en revanche, si le capteur mesure une température inférieure ou égale à 23°C, la fenêtre est fermée. Nous avons, par la suite, observé que la donnée de conductivité n'était pas affichée sur Grafana (lien [1]), ce qui était un simple oubli qui a été corrigé. Pour les capteurs présents dans l'eau, nous avons effectué des mesures avec d'autres capteurs, emprunté à Christophe Rousseau, afin de vérifier si les premiers fonctionnaient de manière adéquate. Mis à part le capteur ORP que l'école n'avait pas en second exemplaire, les autres capteurs ont été vérifiés et fonctionnent. Lors des dernières séances de projet, nous avons constaté que le capteur de pH était défectueux, il affichait des valeurs de pH acide (aussi acide qu'un citron). Nous savons que ce capteur dysfonctionne : sinon nous aurions perdu nos poissons et nos plantes.

II.1.b Partie électrique

Il y a des capteurs de tension et d'intensité présents sur les différentes parties du système électrique. Il y a donc des capteurs pour obtenir les puissances :

- des batteries,
- des panneaux solaires,
- de la consommation électrique de la serre,
- du réseau.

Pour pouvoir obtenir les valeurs de tension et d'intensité avec un multimètre, nous avons conçu des boîtiers de dérivation afin de manipuler sans avoir à toucher directement le circuit, mais seulement en branchant des câbles (figure 4).

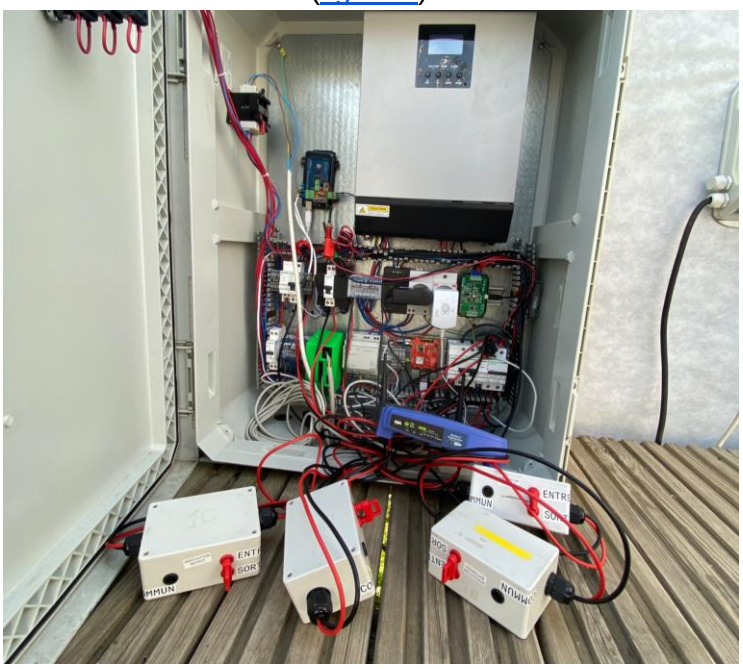


Figure 4 : Au premier plan : les quatres boîtes de dérivation que nous avons conçues.

Nous avons pu alors observer que tous les capteurs fonctionnent bien, excepté celui de la partie PV, qui est défectueux depuis le 1er avril. Comme on le remarque sur la figure 5, aucune donnée ne remonte sur Grafana (lien [1]) concernant la puissance produite par les

panneaux solaires. Nous n'avons pas changé le capteur car nous nous sommes rendu compte du dysfonctionnement tardivement.

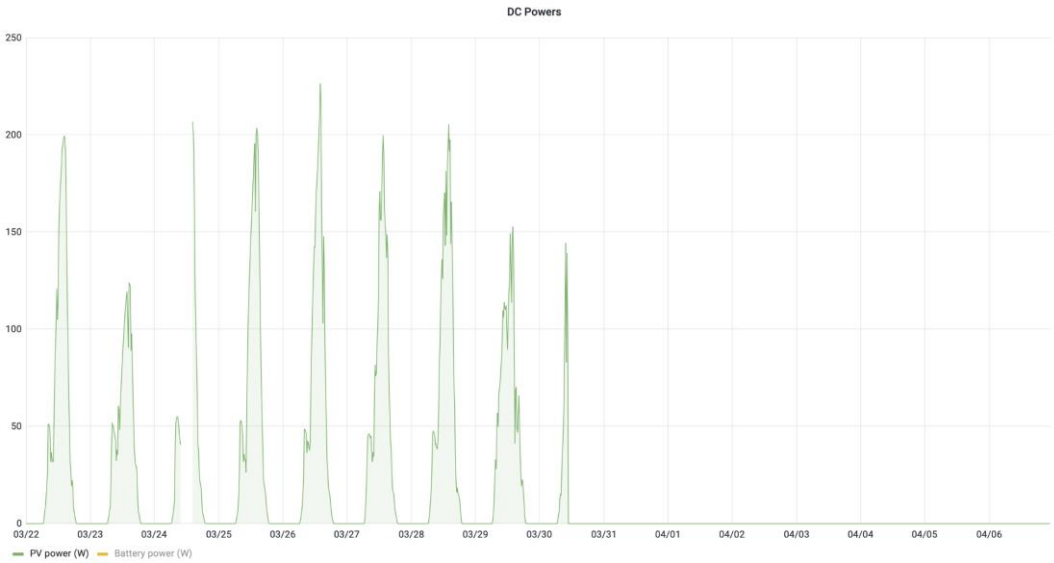


Figure 5 : Mesure de la puissance en sortie des panneaux solaires.

II.2 Optimisation de l'énergie

II.2.a Paramétrage de l'onduleur

L'objectif de cette partie est d'optimiser la gestion de notre micro-réseau par l'onduleur hybride. En effet, celui-ci a également pour rôle de gérer le basculement des puissances en alimentant les actionneurs et capteurs avec soit l'énergie des panneaux photovoltaïques, soit l'énergie stockée dans la batterie, soit l'énergie du réseau de distribution. C'est également lui qui décide quand recharger la batterie et avec quelle énergie. C'est sur ce point que nous nous sommes concentré. En observant les cycles de charge et décharge de la batterie, nous nous sommes rendu compte que la batterie est rechargée régulièrement par le réseau. Or, l'intérêt de cette batterie est uniquement de stocker le surplus d'énergie produite par les panneaux photovoltaïques la journée pour la redistribuer la nuit. De plus, si la batterie se retrouve chargée au maximum pendant la nuit par le réseau, on ne pourra plus stocker le surplus d'énergie solaire au lever du soleil le lendemain.

Sur la figure 6, on observe la production photovoltaïque (en vert) et la puissance entrante (partie négative) et sortante (partie positive) de la batterie (en jaune) sur trois jours. La journée, on voit nettement que la batterie est rechargée par le photovoltaïque (symétrie entre la courbe jaune et verte). La nuit, on observe de multiples pics de recharge de la batterie et une décharge constante. On en déduit rapidement que ces pics correspondent à des cycles de recharge de la batterie par le réseau suivi de cycle de décharge pour alimenter les appareils électriques de la serre.

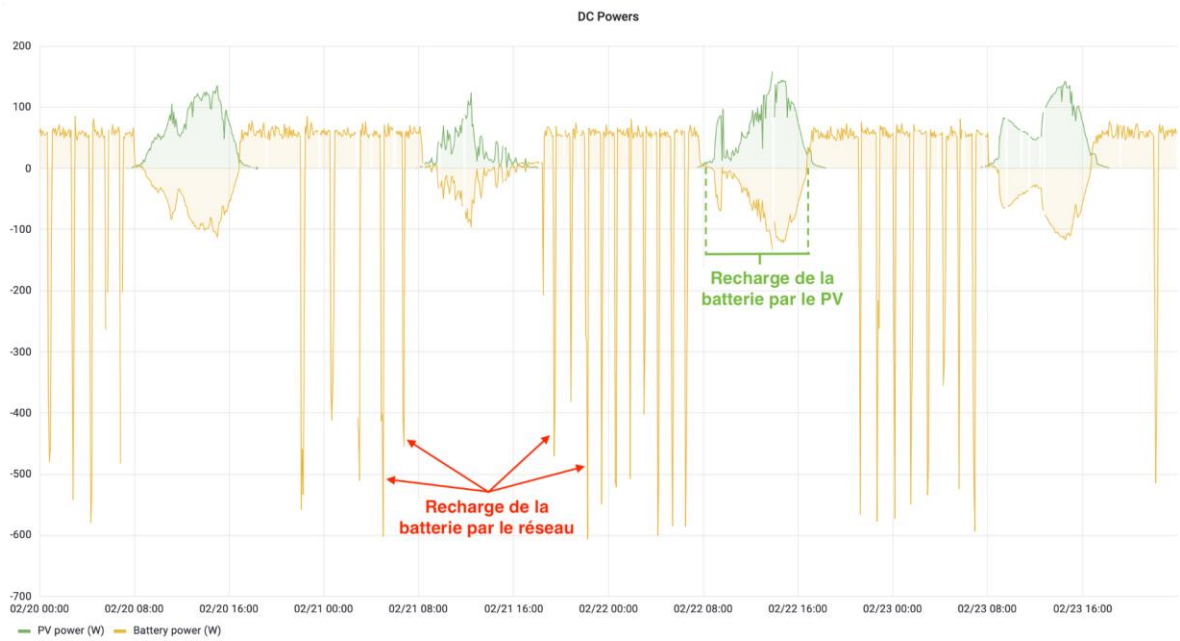


Figure 6 : Puissances des sources continues sur 3 jours avant ajustement de l'onduleur.

Après avoir fait ce constat, il est nécessaire de s'intéresser à l'onduleur pour modifier ses priorités et ses modes de fonctionnement. L'onduleur offre une vingtaine de paramètres ajustables. Parmi eux, on retrouve principalement deux paramètres essentiels correspondant à la hiérarchie des sources d'alimentation :

- La priorité de la source d'alimentation : on choisit ici que le solaire doit être la principale source d'alimentation de notre serre, si l'énergie produite par ceux-ci est nécessaire à l'alimentation de toutes les charges.
- La configuration de la source de charge batterie prioritaire : on choisit ici que l'énergie solaire est la seule source d'énergie pour charger la batterie, si l'énergie produite par celle-ci est nécessaire à l'alimentation de toutes les charges.

Une fois tous les paramètres de l'onduleur actualisés (cf livrable : Paramétrage de l'onduleur). On observe le nouveau comportement de notre système électrique en figure 7.

Figure 7 : Puissances des sources continues sur 3 jours après ajustement de l'onduleur.

Le nouveau comportement de notre système électrique correspond exactement à ce que nous voulions. En effet, on observe bien que la journée, le solaire alimente la serre et recharge la batterie. Lorsque le soleil se couche, la batterie prend le relais pendant environ 8h. Et lorsque la batterie est vide, le réseau alimente les charges de la serre jusqu'à ce que l'énergie en sortie des panneaux solaires soit suffisante.

II.2.b Bilan de production d'énergie solaire

On cherche à savoir si les 2,31m² de panneaux photovoltaïque recouvrant notre serre sont suffisants pour notre consommation. Pour cela, on dispose de multiples capteurs électriques disposés aux endroits stratégiques de notre réseau. Parmi les données qu'ils nous fournissent, deux nous intéressent : la puissance instantanée produite aux bornes des panneaux photovoltaïques et la puissance instantanée consommée par la serre. Avec cette seconde mesure, on remarque qu'elle est constante et qu'elle correspond à environ 50W, soit 1200Wh/jour.

Pour la puissance produite par le solaire, on observe de grandes variations liées à la météo : plus il fait beau, plus on produit. Ces variations sont également dépendantes de la saison de l'année : les journées d'été étant plus longues, on produit plus qu'en hiver. Pour déterminer l'énergie produite chaque jour, on intègre la puissance instantanée sur chaque journée. Ne disposant pas des données pour la totalité de l'année, nous sommes contraints de réaliser ces calculs sur des périodes distinctes.

Pour la période du 20 janvier 2022 au 8 mars 2022, on obtient par exemple ceci :

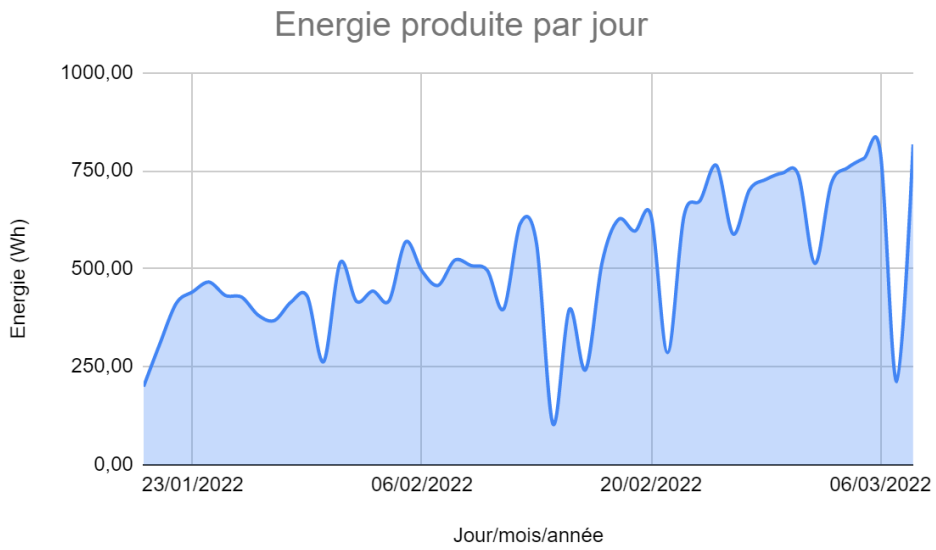


Figure 8 : Énergie électrique produite par les panneaux solaires du 20/01/2022 au 08/03/2022.

On peut nettement observer la tendance montante sur les mois mais également les variations liées à la météo.

Sur l'ensemble des données de 2021 et 2022 disponibles, on a ainsi pu établir que notre record de production journalière a été établi le 22/07/2021 avec une production de 1010,45 Wh produite sur la journée. Notre record de puissance instantané a quant à lui été établi le 01/06/2021 à 14:40 avec un pic à 211 W.

Ces records sont encore bien loin de l'énergie consommée par jour et montre bien que nos panneaux photovoltaïques sont insuffisants pour alimenter pendant un jour les charges de notre serre : surtout pendant la nuit où la batterie se déchargera totalement avant le lever du soleil. Pour augmenter notre part de solaire, il faudrait mieux orienter les panneaux afin qu'ils produisent les 140W/m² promis par cette technologie (lien [2]) et enfin, pour être totalement autonome, au moins doubler leur surface.

II.3 Nivellement automatique du bassin

II.3.a Etude hydraulique de l'écoulement

Nous avons réalisé un programme Matlab qui nous donne la hauteur du réservoir et du bassin en fonction du temps avec une simple vanne. Pour cela, nous avons seulement appliqué la relation de Bernoulli sur la ligne de courant entre un point de la surface du réservoir où la pression vaut P_{atm} et la sortie du tuyau où la pression vaut également P_{atm}.

Nous affichons les résultats pour un réservoir rempli en entier et une hauteur d'eau dans le bassin de 40 cm :

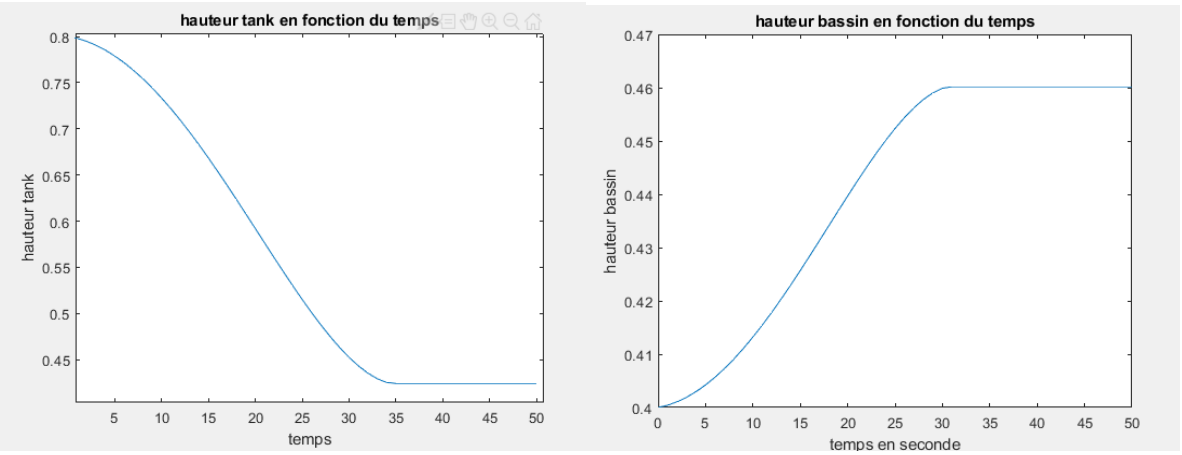


Figure 9 : Simulation de la hauteur du bassin et du réservoir en fonction du temps avec une simple vanne.

Nous pouvons faire de même avec une hauteur de réservoir différente, par exemple 50 cm. Dans ce cas-là, il n'y a pas d'écoulement dans le bassin. C'est pourquoi nous avons besoin d'une pompe qui viendra apporter la puissance nécessaire pour remplir le bassin même lorsque le niveau du réservoir est égal ou inférieur à la position du tuyau du bassin.

II.3.b Programme simple et direct

Notre première idée, pour permettre le nivellement automatique, était de mettre en place un programme assez simple se basant seulement sur la hauteur d'eau dans le bassin. Le but étant de séparer en plusieurs paliers la hauteur d'eau et de remplir le bassin lorsque cette hauteur d'eau atteint une valeur que l'on considère comme limite. La création de ce programme était plutôt simple car une ébauche avait déjà été effectuée l'année précédente, il fallait seulement le mettre à jour. Dans le cas pratique, le programme n'active pas directement la pompe mais d'abord des relais magnétiques, alimenté en 5V par la carte Raspberry Pi, qui eux permettent de faire le lien avec la pompe, elle alimentée en 24V par les batteries.

Le code python implémenté dans la Raspberry Pi se trouve sur la plateforme Alfresco.

II.3.c Développement de la méthode de Q-learning

Face à la solution rapide et efficace que nous avions, et avant tout pour nous permettre d'avoir de nouvelles connaissances de programmation, il nous a été proposé de développer une intelligence artificielle basée sur la technologie de Q-learning pour automatiser le nivellement du bassin. Quelques séances ont dû être nécessaires pour pouvoir prendre en main cette nouvelle technologie de programmation.

Le Q-learning est une méthode de programmation utilisant l'apprentissage par renforcement. Cette méthode vise à optimiser les actions d'un agent sur son environnement. Dans notre cas, l'environnement de notre programme est constitué de la serre connectée ainsi

que de quelques capteurs permettant d'avoir des informations plus précises sur l'évolution de la vie au sein de la serre. Cela est un des avantages de ce programme par Q-learning, pouvoir prendre en compte beaucoup plus de données en temps réel et donc d'avoir une action encore plus ciblée.

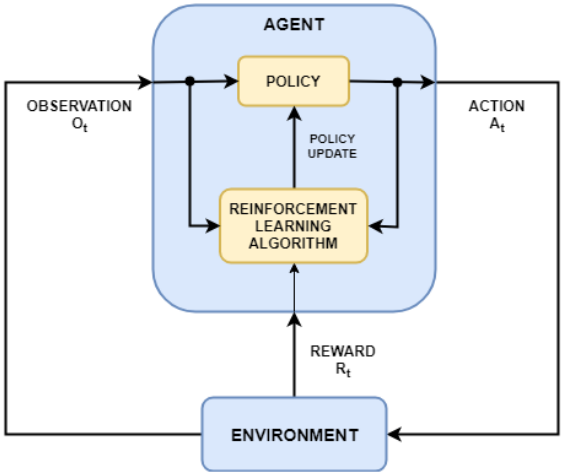


Figure 10 : Explication schématique du Q-learning

La figure 10 ci-dessus permet de comprendre le fonctionnement de la méthode Q-learning. Tout d'abord, l'agent récolte les données issues de l'environnement en temps réel (Observation O_t en reprenant l'écriture du schéma). Ensuite, le but de cette méthode est de récompenser positivement ou négativement (Reward R_t) l'agent en fonction de l'action (Action A_t) qu'il a effectué sur l'environnement en réaction aux données qu'il a obtenues, c'est ainsi que l'intelligence artificielle apprend en temps réel. Enfin, l'agent voulant de son côté maximiser sa récompense moyenne au cours du temps, il va petit à petit ne plus faire les actions que l'on considère comme mauvaises et qui lui valent une récompense négative.

Une fois que les bases de compréhension du Q-learning ont été vues, notamment grâce à Vincent Imard du G2Elab, il a fallu passer du cas théorique au cas pratique sur la serre. Dans notre cas, ce que l'on appelle l'action de l'agent représente l'activation de la pompe pour pouvoir remonter le niveau d'eau dans le bassin à partir du bassin de récupération d'eau de pluie. De plus, l'environnement représente la serre dans sa globalité et nous observons son évolution à l'aide des différents capteurs. Nous avons pu avoir accès à un programme Python contenant l'environnement représentant la serre. Ainsi, de notre côté, nous avons eu à mettre en place deux fonctions (Annexe 4): la première visant à définir la récompense que l'on veut donner à l'agent en regardant si l'action qu'il a effectué est en accord avec les données issues de la serre, la seconde pour modifier notre environnement suivant l'action que l'agent a effectué.

Dans l'objectif de récompenser correctement notre agent, nous avons accès à cinq types de données issues de capteurs de la serre. Ce sont les niveaux d'eau dans le bassin ainsi que dans le réservoir d'eau de pluie, la production des panneaux photovoltaïques, la prévision de précipitation et l'état de la pompe (activée ou non). Ainsi, il faut que l'action de l'agent soit en accord avec la valeur de ces observations. Un exemple à titre indicatif, si l'agent

remplit le bassin alors que ce dernier est déjà presque plein, il sera puni et le sera encore plus s'il ne fait pas soleil car cela implique que l'eau ne va pas s'évaporer rapidement donc il est inutile de le remplir pour le moment. Néanmoins, il est de notre ressort d'exploiter de la bonne manière ces données au sein de notre fonction et notamment de leur donner un ordre d'importance. Il paraît normal que la valeur du niveau d'eau dans le bassin soit une donnée qui prévaut sur les autres car elle est celle sur laquelle nous voulons et pouvons agir. Ainsi, en prenant tout cela en compte, nous avons programmé la fonction **reward** de l'Annexe 4.

Ensuite, nous devions programmer la fonction permettant d'appliquer l'action de notre agent à l'environnement. En effet, le but d'une telle méthode d'intelligence artificielle est de pouvoir l'entraîner au préalable pour que le code que l'on implémente dans la serre soit déjà efficace. Pour cela, il faut prendre en compte que la pompe délivre 18L d'eau en une minute d'activation. De plus, pour permettre une simulation crédible, il faut prendre en compte une évaporation dans le bassin ainsi que dans le réservoir d'eau de pluie. Enfin, le réservoir d'eau de pluie doit pouvoir être rempli de manière aléatoire pour pouvoir simuler les précipitations. L'ensemble de ses modifications sur l'environnement peut se retrouver au sein de la fonction **step** de l'Annexe 4.

Finalement, nous devions simuler l'implémentation de notre intelligence artificielle dans la serre. Pour cela, nous avons utilisé un fichier de données fourni par le G2Elab et prélevé sur une grande période de temps, représentant ainsi l'évolution des valeurs issues des capteurs de la serre au cours du temps. Le but de ces manipulations est de faire au préalable une boucle de 10000 entraînements puis de faire notre simulation sur l'environnement fictif dans lequel peut agir notre agent maintenant entraîné. Nous nous sommes rapidement rendus compte de certaines limites de nos premières fonctions de récompense. Dans un premier temps, l'action de l'agent n'évoluait jamais tout au long de la simulation. Dans le cas où l'agent activerait la pompe en continu, la récompense pour cette action est trop élevée en comparaison des autres. Sinon, dans le cas où la pompe ne s'active jamais, les récompenses négatives sont trop importantes donc l'agent considère que faire une quelconque action lui sera forcément défavorable ainsi, il n'agit pas car cela lui semble plus rentable. Après avoir arrangé les récompenses afin que l'agent effectue plusieurs actions différentes au cours de la simulation, un nouveau problème est apparu : la pompe était activée et désactivée sans cesse pour remplir seulement une petite quantité d'eau à chaque fois. Cette situation ne nous convient pas car nous voulons limiter la consommation électrique et donc faire fonctionner la pompe moins souvent mais sur des durées plus longues. Pour résoudre ce problème, nous avons décidé de récompenser positivement l'agent si l'action précédente était la même que celle qu'il allait prendre. Après de nombreux entraînements et simulations pour équilibrer les coefficients de récompenses, nous avons obtenu une solution convenable où le niveau du bassin arrive à se stabiliser autour d'une valeur moyenne satisfaisante (Annexe 4), tout en ne vidant pas complètement le réservoir d'eau de pluie. Enfin, nous voyons sur la Figure 11 que la pompe ne s'active pas tout le temps et évite les activations/désactivations sans réel intérêt.

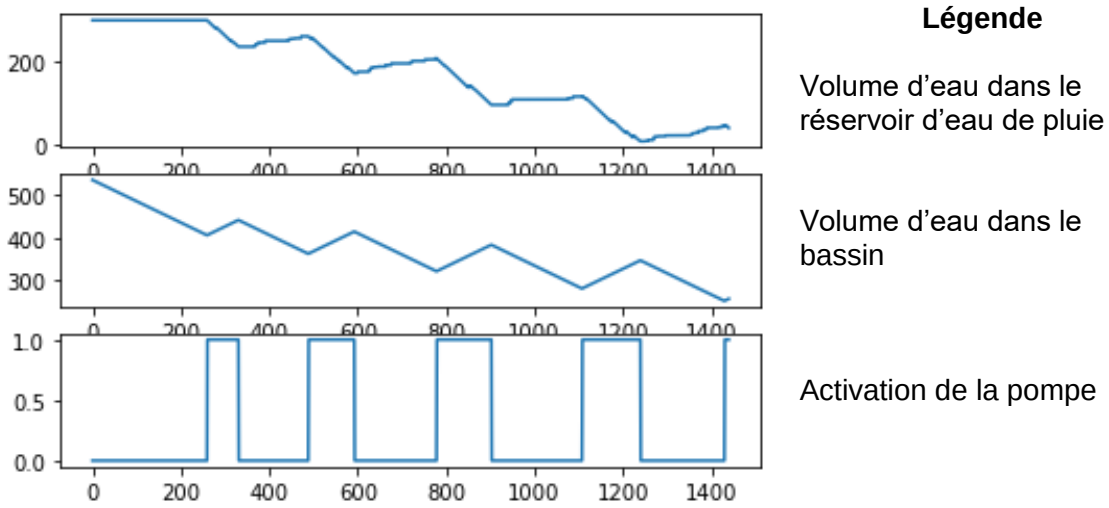


Figure 11 : Simulation finale de la méthode de Q-Learning

II-3.d Déploiement dans la serre et câblage

Dernière étape pour le nivellement du bassin, la mise en place finale du programme auquel nous devons ajouter plusieurs éléments afin de connecter correctement la carte Raspberry Pi aux capteurs et à la pompe. En effet, nous avons voulu dans un premier temps nous concentrer sur la mise en place du premier programme avec deux capteurs de distance, l'un dans la serre et l'autre dans le réservoir (lien [3]), pour avoir les niveaux d'eau. On peut voir sur la figure 12 que nous utilisons la valeur de ces deux capteurs pour contrôler un relais magnétique activant la pompe. Malheureusement, la mise en place d'une électrovanne, comme cela avait été proposé les années précédentes, pour limiter la consommation d'électricité lorsque l'eau dans le réservoir est suffisamment haute a été abandonnée pour le moment car la position du réservoir au même niveau que le bassin rend ce principe inintéressant pour le moment. De plus, la mise en place de notre programme de Q-learning nécessite plus de temps et cela n'a pas été possible cette année.

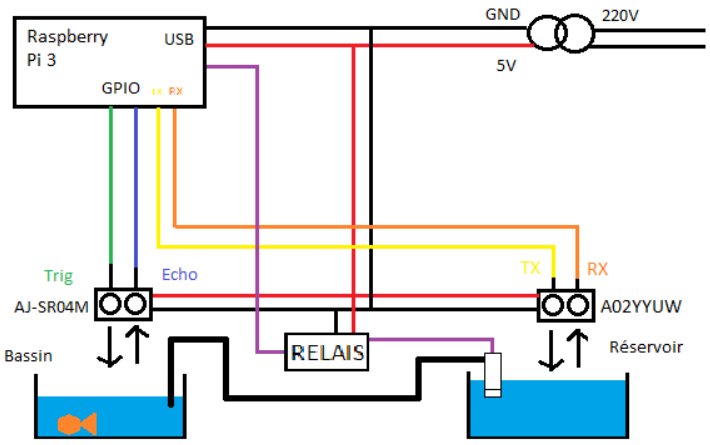


Figure 12 : Installation du nivellement automatique

II-3.e Transfert des données via LoRa

Après avoir effectué les mesures avec les différents capteurs, les données sont traitées par la Raspberry Pi et envoyées via le protocole de communication LoraWan. Pour pouvoir réaliser cette partie, nous nous sommes beaucoup appuyés sur le mini projet de Jérôme Ferrari “Monitorer la température et le PH d’un bassin avec une connexion Lora et remonter l’information dans Jeedom”.

L’avantage d’utiliser le protocole de communication LoRaWan est qu’il a une consommation électrique très faible tout en transférant les données sur une assez grande distance. Cet avantage se fait au détriment du flux de données. Ce protocole ne permet pas d'envoyer des fichiers lourds. Dans notre cas, on envoie des données de hauteur d’eau, c’est donc suffisant.

On a dans un premier temps branché la carte Dragino à notre raspberry et téléchargé différents outils (lmic-lora-gps-hat), une bibliothèque dédiée à l’envoi de données par Lora. La carte dragino permet d’ajouter une antenne pour émettre et recevoir un réseau sans-fil, elle se plug sur la carte du raspberry.

Les données sont ensuite envoyées à l’application depuis l’antenne à The Thing Network. Les données reçues doivent être déchiffrées avant de pouvoir être exploitées. Nous n’avons pas pu aller plus loin cette année. Il nous restait encore l’affichage sous forme de graphique des valeurs de niveaux d’eau sur Grafana.

II.4 Moteur de la fenêtre

L’année dernière, les étudiants avaient mis en place un moteur ouvrant la fenêtre dès que la température dépassait 27°C et la fermant dès qu’elle était inférieure à 23°C. Nous nous sommes rapidement rendus compte que le programme ne fonctionnait pas. En effet, sur Grafana (lien [1]) nous avons observé que le capteur virtuel de l’ouverture de la fenêtre (0 ou 1 donné par la carte Raspberry et transmis via LoRa) n’était pas en adéquation avec la réelle ouverture ou fermeture de la fenêtre. Il y avait en effet une erreur dans le code, juste un commentaire mal placé qui empêchait le programme de changer d’état la fenêtre et donc d’activer ou non son moteur.

Ensuite, nous nous sommes rendus compte que les branchements aussi étaient défectueux. Nous avons refait les branchements au niveau des relais. Il s’est avéré que les branchements étaient bien défectueux entre la carte raspberry et les relais.

Nous en avons profité pour refaire toutes les soudures, ainsi qu’imprimé en 3D et installé une fixation pour la carte Raspberry Pi (figure 13). Une fixation était déjà présente sur la carte mais avait fondu et s’était déformé. Nous l’avons réimprimé en ABS, un plastique qui supporte les hautes températures (80°C) que peut générer la carte Raspberry Pi.

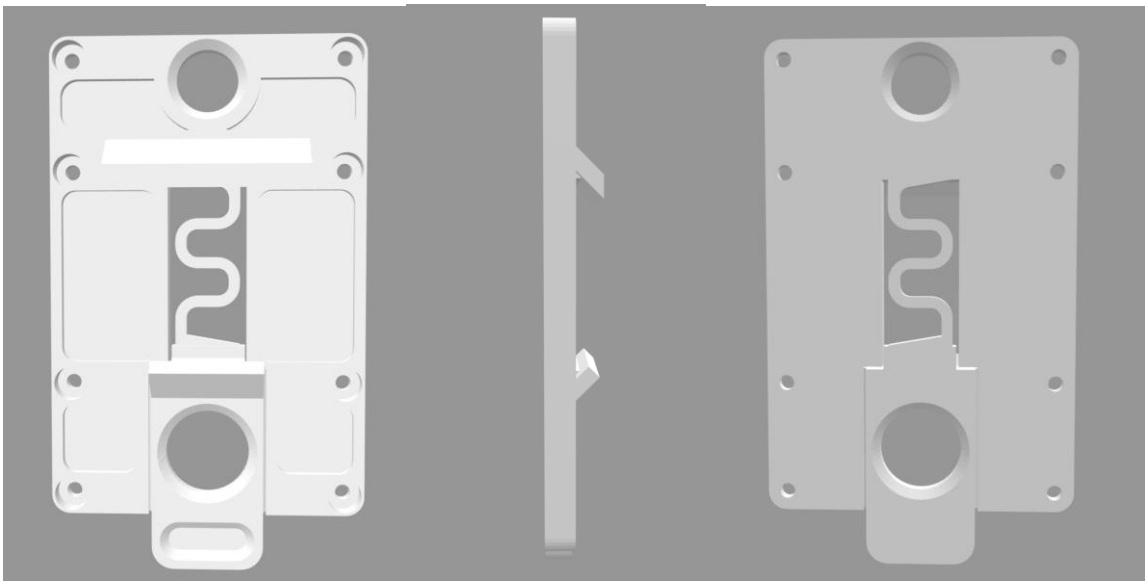


Figure 13 : vue de face, de droite et d'arrière du modèle 3D de la fixation de la carte

Actuellement, la fenêtre s'est bien initialisée lorsque nous avons rebranché la carte. Cependant, nous ne sommes pas sûrs que la fenêtre fonctionne réellement, car au vu de la température dans la serre, la fenêtre n'est pas amenée à fermer lors des heures où nous sommes à l'école.

II.5 Conception de différentes pièces mécaniques

Pour la protection des capteurs et des cartes Raspberry Pi, nous avons dû concevoir des boîtiers.

Boitier de protection du capteur de distance:

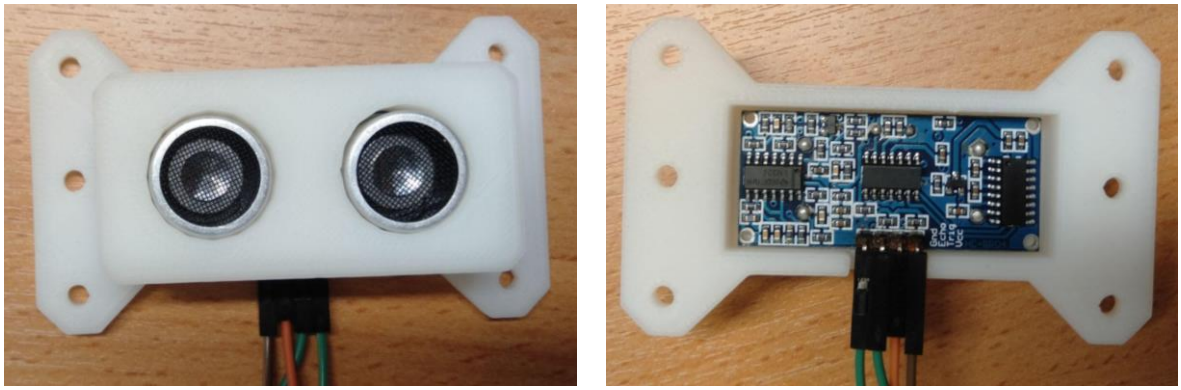


Figure 14 : Boîtier de protection du capteur de distance

Il a été conçu sous Catia et imprimé en 3D. Le socle vient se visser en dessous.

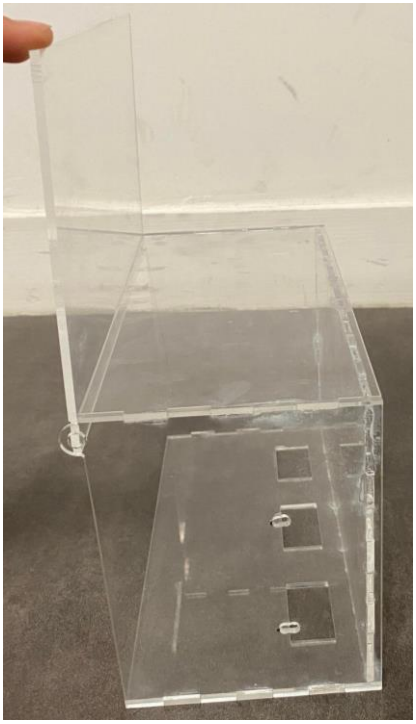


Figure 15 : Boîte de protection de cartes

Boite pour les cartes:

Nous avons utilisé la découpe laser pour concevoir cette boite. Nous avons placé des trous carrés pour les câbles, un couvercle qui permet de l'ouvrir à 180°, et deux trous pour la positionner sur les rails dans la serre grâce à des équerres en métal. Ces équerres ont également été confectionnées au Fablab grâce à la découpe à eau.

La boite est assez grande pour permettre aux années suivantes de pouvoir y mettre d'autres composants électriques. Elle est en plexiglas de 3 mm d'épaisseur.

II.6 Application pour les données

II.6.a Cahier des charges et moyens utilisés

Pour offrir un accès simple, rapide et visuel aux données pour le propriétaire de la serre, il a été décidé de développer une application mobile. Celle-ci doit être capable d'afficher l'état de santé actuel de la serre, sa production d'électricité, son taux d'utilisation du réseau des 10 derniers jours et signaler toute nécessité d'intervention humaine.

L'application sera développée pour les téléphones Android pour être accessible au maximum de smartphone possible. Son développement se fera dans Android Studio pour avoir un IDE totalement dédié aux développements Android et maximiser la compatibilité avec les différents téléphones et les différentes versions du système d'exploitation.

Le principal enjeu de ce projet sera le lien entre le téléphone et la base de données Influxdb où sont stockées les données de la serre. Le développement Android se faisant en Java, il est nécessaire de d'abord maîtriser l'envoie des requêtes à la base de données en ce langage avant de l'implémenter dans l'application.

II.6.b Récupérer les données de influxdb en Python

La première étape de ce projet est donc de savoir quelle requête envoyer à quel serveur et comment le faire en JAVA.

Pour commencer, nous disposions d'exemple de code Python permettant de récupérer des données stockées sur Influxdb. Nous les avons adaptés pour récupérer les données correspondant à notre projet. Pour cela, il faut

- **1) Importer la librairie adaptée**

  from influxdb import InfluxDBClient

- **2) Déclarer les paramètres et créer l'objet de connexion client**

Pour cela, on utilise la fonction InfluxDBClient() qui va prendre en entrée l'adresse IP du serveur, le port de connexion, le chemin, l'identifiant, le mot de passe et le nom de la base de données.

client = InfluxDBClient(host = INLUXDB_SERVER_IP, port = INLUXDB_SERVER_PORT, username = INFLUXDB_USERNAME, password = INFLUXDB_PASSWORD, path = INFLUXDB_PATH, database = INFLUXDB_DATABASE, timeout = 3, retries=2,ssl=True,verify_ssl=False)

- **3) Envoyer la requête adaptée en InfluxQL pour récupérer les données désirées.**

datasheet=client.query('SELECT mean(vBattery) FROM SensorDataElectric WHERE time >= now() - 1h and time <= now() GROUP BY time(60m) fill(null)')

II.6.c Récupérer les données de influxdb en JAVA

On suivra les mêmes étapes qu'en Python pour envoyer des requêtes en Java. Il faut donc d'abord trouver la bonne librairie permettant de le faire. Or, notre base de données Influxdb est en version 1.7. Elle est très différente des nouvelles versions 1.8+ et 2.x pour lesquelles on trouve de la documentation récente. Il faut être particulièrement attentif à la bibliothèque choisie. Nous utiliserons donc cette bibliothèque compatible à la version 1.7 : https://github.com/influxdata/influxdb-java

On suit donc les mêmes étapes qu'en python pour créer notre premier script JAVA.

- **1) Importer la librairie adaptée (importer précédemment les fichiers correspondants au projet).**

```
import org.influxdb.InfluxDB;
import org.influxdb.InfluxDBFactory;
import org.influxdb.dto.Query;
import org.influxdb.dto.QueryResult;
```

- **2) Déclarer les paramètres et créer l'objet de connexion client**.

Pour l'objet de connexion client, on utilise la méthode *InfluxDBFactory.connect(serverURL, username, password)* qui renvoie un objet de type InfluxDB à partir de l'URL du serveur, l'identifiant et le mot de passe.

- **3) Envoyer la requête adaptée en InfluxQL pour récupérer les données désirées.**

Une fois l'objet de type InfluxDB créé, on peut exploiter sa méthode *.query(requête, nom_de_la_base_de_donnée)* en créant une requête avec le constructeur Query() et une requête en langage InfluxQL.

```
QueryResult data = influxDB.query(new Query("SELECT mean(vBattery) FROM SensorDataElectric WHERE time >= now() - 1d and time <= now() GROUP BY time(60m) fill(null)", database));
```

II.6.d Design de l'interface

Le design de l'application est réalisé dans Android Studio en XML et se base sur le cours de openclassrooms intitulé "Développer votre première application android" (lien [4]).

Ayant un temps limité et un niveau débutant en développement Android, il a été décidé de concevoir une interface simple où les objets sont disposés les uns à la suite des autres verticalement. Pour cela, on utilise un LinearLayout. On place dans ce Layout 7 blocs TextView : un pour le texte de bienvenue et 6 pour les données pertinentes à afficher, un bouton pour actualiser les données affichées et une image pour rendre la page d'accueil plus esthétique et agréable.

Figure 16 : Visuel de l'interface de l'application

II.6.e Organisation du script

Pour faciliter la lecture et la gestion du code Java, il sera divisé en deux classes : la classe principale qui gère les interactions avec les éléments graphiques de la page d'accueil, et une classe "DataInfluxdb" qui gère les intéractions avec les serveurs extérieurs et devra se charger de récupérer les données et les traiter.

La classe "DataInfluxdb" propose donc deux constructeurs. Un par défaut sans paramètres qui va utiliser l'URL du serveur du G2ELAB, notre identifiant, notre mot de passe pour la base de données Greenhouse_monitor pour créer un objet de connexion. Et un second, qui demande en entrée les paramètres nécessaires pour créer l'objet de connexion.

Une fois les constructeurs définis, on ajoute à la classe des méthodes. La méthode la plus importante va être celle que l'on appellera à chaque actualisation des données affichées. En effet, celle-ci doit alors envoyer les requêtes correspondantes aux données que l'on veut afficher, récupérer ces données, les traiter, et renvoyer un objet de type String pour le concaténer au texte de l'objet graphique correspondant.

Cette partie n'a pas été terminée car un problème amont a été détecté et non résolu. Nous aborderons ce problème dans la partie qui suit.

II.6.f Implémentation du script Java dans Android Studio

Après avoir testé le script java avec succès dans l'environnement de développement NetBeans et établi une connexion entre le script et la base de données, il faut implémenter ce script dans Android Studio pour le faire fonctionner sur un smartphone.

On commence par ajouter au projet les fichiers .jar correspondant à la librairie Influxdb que l'on veut utiliser.

Ensuite, on crée la classe DataInfluxdb que l'on a développée et testée précédemment dans NetBeans. On vérifie que les méthodes provenant de la librairie sont bien détectées et n'affichent pas d'erreur.

Dans la classe principale, gérant les interactions avec l'interface graphique, on ajoute la méthode .setOnClickListener() pour appeler la méthode d'actualisation des données lorsque l'on appuie sur le bouton de l'interface.

Une fois tout implémenté, on peut commencer les premiers tests sur smartphone. L'interface s'affiche correctement. Cependant, après l'isolation de différentes sources potentielles d'erreur, on peut établir que la création de l'objet de connexion ne fonctionne pas et fait crasher l'application. On peut éviter le crash en isolant l'erreur à l'aide des lignes de code suivantes.

```
try{
   this.influxDB=     InfluxDBFactory.connect(serverURL,     username,
password);
} catch (Exception e) {
e.printStackTrace();
```

Cela ne fait pas fonctionner l'application pour autant. Le script Java fonctionnant correctement sur netBeans, on peut supposer deux sources potentielles de ce problème. La première est que les fichiers .jar utilisés sont trop vieux et pas adaptés à la version d'Android utilisée. La seconde est qu'il est nécessaire de passer par une autre méthode pour envoyer des requêtes à des serveurs depuis une application malgré le fait que la documentation de la librairie assure sa compatibilité avec le développement Android. La difficulté dans la résolution de cette erreur est qu'Android Studio n'affiche aucune erreur ou aucun avertissement lors de la compilation du projet. L'erreur apparaît uniquement lors des essais.

II.7 Etude sur le vivant

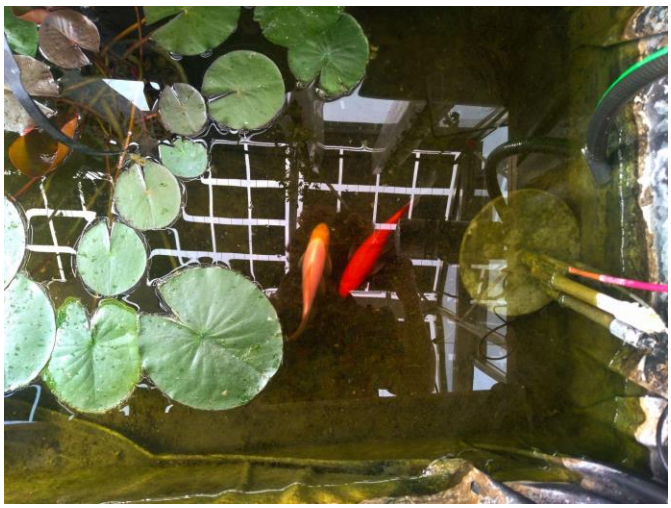


Figure 17 : Poissons et nénuphar dans le bassin

Nous avons effectué quelques recherches sur les conditions de vie des poissons rouges. Ces poissons sont plutôt résistants, ils peuvent en effet vivre dans une eau dont la température peut être comprise entre 2°C et 29 °C. Nous avons observé des températures du même ordre tout au long de l'année grâce à Grafana (lien [1]). Ainsi, d'autres espèces de poissons ne pourraient pas vivre dans des conditions pareilles.

Le pH doit être compris entre 6 et 7.5.

De plus, nous avons lu dans le "Guide pratique de l'aquaponie" par Gregory Biton que le rapport entre le niveau d'eau et la superficie du bassin devait être d'environ ⅓. Nous devrions avoir un niveau de bassin de 65 cm ce qui n'est pas possible puisque la hauteur est de 50 cm. Attention à ne pas mettre trop de poissons pour éviter les risques d'étouffement.

Pour ce qui est des plantes, nous avons effectué nous même des plantations de différentes graines que nous avons fait grandir suffisamment pour que les tiges soient résistantes et puissent survivre dans la serre. Nous avons donc aujourd'hui dans la serre des plants de tomates, de la ciboulette, des plants de persil plat mais aussi de la salade. Mais nous avons pu observer d'une semaine à l'autre que les tuyaux se bouchent et les plantes ne sont donc plus arrosées. Ainsi, nous avons perdu beaucoup de salades. Il faut donc penser à

mettre des plantes assez résistantes, qui poussent vite et peuvent survivre quelques jours sans eau (tomates, betteraves, le poireau, la ciboulette, le persil…).

Figure 18 : De haut en bas, de gauche à droite: persil plat, salade, ciboulette, tomate.

III. Conclusion

En conclusion, nous avons pu faire une grosse partie des objectifs à réaliser.

Nous avons pu contrôler l'intégralité des capteurs de la serre (capteurs du vivant et capteurs de la partie électrique), nous avons fixé les paramètres de l'onduleur hybride pour utiliser un maximum l'énergie photovoltaïque, nous nous sommes chargé de régler le problème lié à l'ouverture et la fermeture de la fenêtre programmé l'année dernière, nous avons développé le programme Q-Learning permettant de remplir le bassin des poissons, nous avons commencé le développement de l'application servant d'interface homme-serre. Toutes ces tâches s'ajoutent au maintien et à la croissance de la partie vivante de la serre.

Chaque membre du groupe à le sentiment d'avoir apporté quelque chose à ce projet. Ce projet nous a permis d'apprendre à gérer un projet sur le long terme, d'apprendre à travailler en groupe et également de s'autonomiser sur des projets techniques.

IV. Gestion de projet

IV.1 Présentation de l'équipe

Le projet requiert des compétences diverses telles que l'étude énergétique, la programmation ou encore l'hydraulique. L'équipe se compose de trois étudiants en filière Ingénierie de l'Énergie Électrique (Jérémy, Théo et Victor), deux en filière Automatique et Systèmes Intelligents (Guillaume et Mateo) et d'une étudiante en filière Mécanique et Énergétique (Lella).

Le responsable de projet est Guillaume Auzeloux, il coordonne la communication avec les porteurs de projets et participe à chaque revue de projet.

La responsable financière est Lella Pourcel, elle a effectué les demandes de subventions, supervisé et effectué les achats faits à Botanic.

Le responsable de la sécurité est Théo Ferrer Brunet, il s'est assuré que les conditions dans lesquelles nous travaillions soient bonnes et sécurisées notamment sur le toit où sont présentes des ruches et abeilles appartenant à l'association BeeGreen.

IV.2 Organisation

Toutes les semaines, Jérôme Ferrari voulait qu'on lui envoie un état des lieux de l'avancée du projet. Nous avons cessé après deux mois, car nous lui expliquions directement

de vive voix. Cela nous a permis au début d'avancer rapidement sur les problèmes rencontrés, car nous n'avions pas encore totalement pris en main la serre et son environnement.

Concernant notre gestion du temps, nous avons rapidement effectué un diagramme de Gantt afin de définir nos tâches et leur temps alloué. Après la deuxième revue de projet, il nous a été conseillé de le refaire en abaissant le nombre de livrables et de prendre conscience que nos travaux étaient chronophages. Nous avons alors refait pour nous permettre de voir à long terme, et d'observer si tous nos objectifs étaient réalisables à temps (Annexe 1).

Fin mars, après un entretien avec les porteurs de projet, nous avons mis à plat nos idées et objectifs en comptant le nombre d'heures nécessaires pour chacun. Il en est alors ressorti que nous devions sûrement abandonner l'idée d'un programme permettant d'observer si les poissons vont bien, par manque de temps et aussi d'idées pour le réaliser.

IV.3 Motivation le long du projet

L'équipe a globalement toujours été très motivée malgré des problèmes récurrents concernant notamment la mise en service de la fenêtre de la serre, ou encore le développement du programme de Q-Learning. Cette motivation est justifiée par la présence constante de tous les étudiants sauf en cas de contamination au Covid, où certains sont restés à distance pour le bien de tous (à retrouver en Annexe 3).

Le démarrage du projet était très excitant. L'idée de travailler sur une serre située sur le toit de l'école nous enchantait tous. Nous avions tous envie de mener ce projet à bien car il nous passionne. L'idée d'être sur le toit et pas constamment derrière un ordinateur nous a permis de ne pas ressentir de longueurs pendant les séances de projet.

Nous nous sommes également retrouvés autour de l'intérêt commun de faire pousser nos propres légumes, d'être responsable d'un système entier. La croissance des salades et du plant de tomates à la fin nous a rendu fortement enthousiastes.

Dès le début du projet, nous avions commencé à faire pousser plusieurs légumes et herbes aromatiques en semis dans le but de faire des récoltes quelques semaines plus tard. Cependant, nous les avons intégrés trop tôt dans la serre et ces dernières ont brûlé : cet événement à un peu fait tomber le moral de tout le monde. En revanche, la part vivante de ce projet est aussi importante que la part technique de ce dernier, ce qui apporte de la "légèreté" au projet.

IV.4 Ambiance dans le groupe

L'ambiance au sein du groupe a toujours été très bonne : chacun a su s'adapter aux autres et a su faire le travail demandé en binôme ou seul.

La cohésion s'est développée au fil du projet même si nous ne nous connaissions pas forcément tous. Cependant, tous les membres du groupe ont pu travailler avec tout le monde et aucun problème n'est apparu.

IV.5 Budget des dépenses faites et celle de l'année prochaine

De nombreux éléments et équipements étaient déjà en notre possession au commencement du projet comme des cartes raspberry, un tuyau et une pompe pour raccorder le réservoir d'eau et le bassin, des capteurs à ultrasons ainsi que plusieurs fichiers contenant un support de carte.

Dès le début du projet, Jérôme FERRARI nous a partagé sa volonté d'ajouter des poissons dans le bassin. Ces poissons ont besoin de coins d'ombre donc nous avons tout de suite pensé à acheter un nénuphar à mettre dans le bassin, ce que nous avons fait.

Après la perte des plantes que nous avons fait pousser en semis, nous avons songé à acheter des plants de légumes suffisamment grands et développés pour limiter le risque de les perdre dans la serre. Nous avons donc choisi un plant de tomates, des pousses de salade et des pousses de persil plat.

De semaines en semaines, nous devions vérifier le niveau d'eau du bassin des poissons : pour remplir ce bassin, nous utilisions au début une poubelle que nous remplissons dans un local sur le toit, mais nous avons pris la décision d'acheter un seau gradué avec anse pour faciliter cette tâche.

Pour l'année prochaine, nous conseillons d'acheter un capteur de pH car celui présent est défectueux : les mesures à la fin du projet sont totalement fausses et un recalibrage du programme python ne ferait que déplacer le problème dans le temps. Pour rappel, acquérir une bonne valeur de pH permet d'être sûr que l'eau n'est pas nocive pour les poissons. Nous recommandons également l'achat d'un capteur de potentiel d'oxydo-réduction : celui-ci servira à vérifier la validité de celui déjà présent dans la serre. Nous conseillons également d'acquérir d'autres plantes si les nôtres ne survivent pas pour l'année prochaine ainsi que du substrats spéciales plantes aquatiques servant au bon développement du nénuphar protégeant les poissons du soleil.

Le détail des achats est développé dans l'Annexe 2.

IV.6 Remarques pour les années à venir.

Pour permettre une meilleure compréhension du sujet aux futurs étudiants, nous avons réalisé une fiche d'aide. Cette fiche regroupe des informations concernant la procédure à suivre pour déconnecter l'électricité de la serre, faire de la maintenance (nettoyer la serre, le filtre, les racks à plante…) sur la serre.

Concernant l'onduleur, nous avons produit un document reprenant les paramètres modifiables dans son menu ainsi que les paramètres que nous avons choisis.

IV.6.a Travail à compléter

Tout d'abord, nous n'avons pas pu finaliser tous nos projets. Nous tenons à faire une liste de ce qu'il reste à compléter.

Premièrement, pour les capteurs, ceux pour le panneau photovoltaïque et le celui du pH sont à remplacer. Nous avions voulu acheter un capteur ORP pour pouvoir valider les données de celui présent dans la serre, mais par manque de temps nous ne l'avons pas fait. Nous conseillons donc d'en acheter un, nous avons mis une référence dans la fiche.

Nous suggérons de vérifier le bon fonctionnement de la fenêtre : si celle-ci ne fonctionne pas, il se peut que le câble reliant les cartes principales aux relais soit débranché, il suffit alors de le rebrancher (Figure 19).

Pour l'automatisation utilisant le machine learning, nous avons bien mis en place l'IA mais nous n'avons pas encore fait en sorte que l'IA reçoive des prévisions météo.

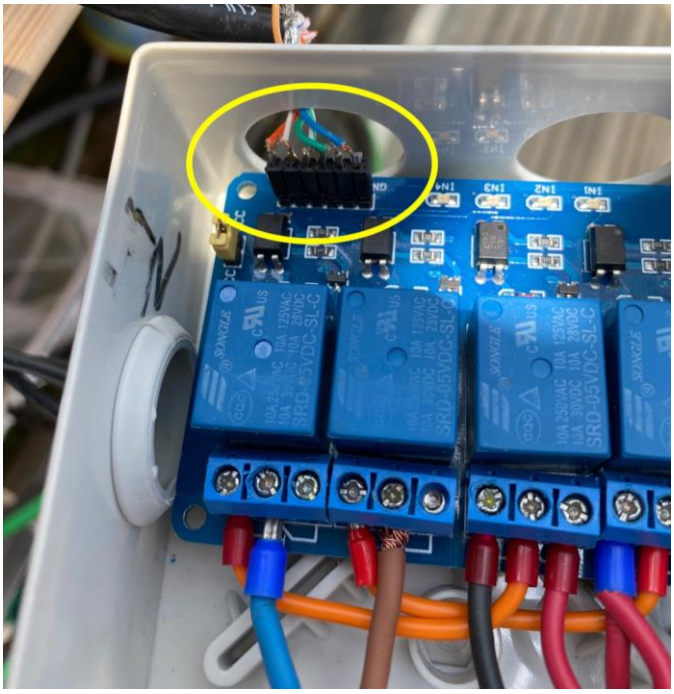


Figure 19 : Lieu de la possible déconnexion du câble servant à piloter la fenêtre (à rebrancher dans le bon sens)

IV.6.b Idées non réalisées

Le pilotage des batteries en prenant compte de la météo et de son état de charge n'a pas été réalisé par manque de temps. Pour l'instant c'est l'onduleur qui décide quand la serre utilise le réseau, les batteries ou les panneaux.

Il était proposé de surveiller l'état du vivant via une caméra présente dans la serre, pour filmer les poissons ou les plantes. La première idée était de prendre des photos à intervalles réguliers dans la journée, et d'ensuite par un algorithme de traitement d'images, surveiller si les poissons vont bien, ainsi que de suivre la pousse des plantes. Les problèmes instantanément identifiés sont le fait que nous ne voyons pas trop ce que signifiait "surveiller les poissons". Mis à part dire s'ils sont vivants ou morts, nous ne pouvons rien observer de plus. De plus, la transmission LoRA ne permet pas d'un envoi conséquent de données, ce qui pourrait être limitant.

La deuxième idée était de mettre en place une vidéo en live pour pouvoir observer les poissons et les plantes tous les jours, et ainsi mettre en avant au sein de l'école ce projet de serre connecté. Nous avons alors toujours le même problème, l'internet n'étant pas accessible sur le toit de l'école pour des raisons de sécurité, la transmission LoRa ne permettrait pas non plus d'envoyer un flux vidéo. Nous avons réfléchi à l'envoi d'une photo par jour qui pourrait être partagée sur Grafana et sur l'application, qui pourrait rendre plus concrètes toutes les données accumulées.

Nous recommandons évidemment aux futurs étudiants de nous contacter en cas de questionnement sur le travail effectué, nous avons essayé d'être le plus clair possible, mais il peut néanmoins rester quelques zones d'ombres dans le projet.

Référence bibliographique

[1] : Tableau de bord Grafana :

https://mhi-srv.g2elab.grenoble-inp.fr/grafana/d/PqWFDhNmk/serre-connectee-green-house?orgId=3&from=now-90d&to=now

[2] : Rendement d'un panneau solaire polycristallin :

Panneaux polycristallins : prix, rendement et fonctionnement

[3] : Fiche technique nouveau capteur distance ultrason waterproof :

A02YYUW Waterproof Ultrasonic Sensor Wiki - DFRobot

[4] : Cours pour développer une application :

Développez votre première application Android - OpenClassrooms

[5] : Livre sur l'aquaponie :

Guide pratique de l'aquaponie: produire ensemble légumes et poissons - construire sa propre installation - Grégory BITTON - 2017 - Editeur : Terran.

[6] : Ferrari, J. (2021, 6 janvier). Monitorer la température et le PH d'un bassin avec une connexion Lora et remonter l'information dans Jeedom. MiniProjets.net.

http://miniprojets.net/index.php/2021/01/06/monitorer-la-temperature-et-le-ph-dun-bassin-avec-une-connexion-lora/

[7] : Configuration de la Raspberry Pi

https://raspberry-pi.fr/installer-raspbian-premier-demarrage-configuration/

ANNEXE 1

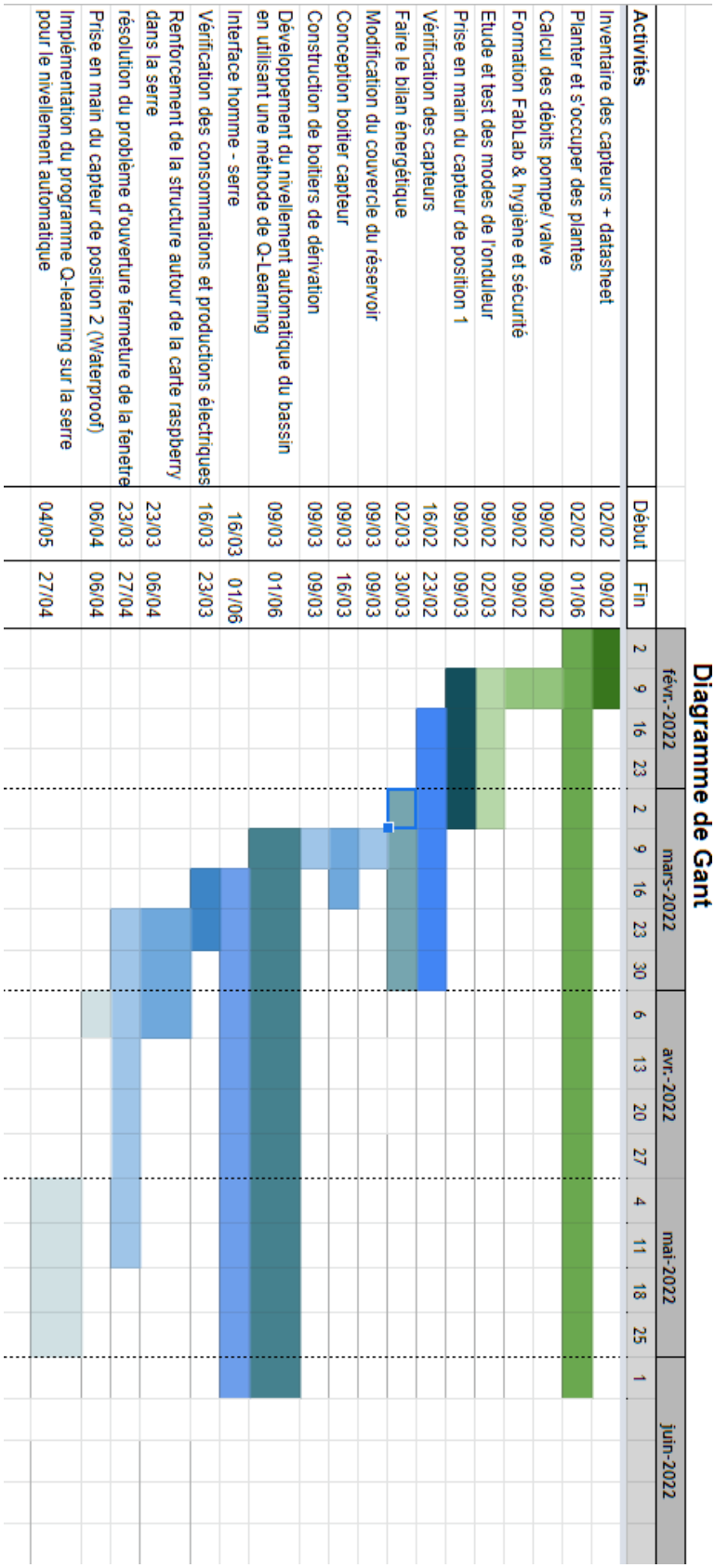


Diagramme de Gant
Activités
Début
Fin
Inventaire des capteurs + datasheet 02/02 09/02
Planter et s'occuper des plantes 02/02 01/06
Calcul des débits pompe/ valve 09/02 09/02
Formation FabLab & hygiène et sécurité 09/02 09/02
Etude et test des modes de l'onduleur 09/02 02/03
Prise en main du capteur de position 1 09/02 09/03
Vérification des capteurs 16/02 23/02
Faire le bilan énergétique 02/03 30/03
Modification du couvercle du réservoir 09/03 09/03
Conception boitier capteur 09/03 16/03
Construction de boitiers de dérivation 09/03 09/03
Développement du nivellement automatique du bassin en utilisant une méthode de Q-Learning 09/03 01/06
Interface homme - serre 16/03 01/06
Vérification des consommations et productions électriques 16/03 23/03
Renforcement de la structure autour de la carte raspberry dans la serre 23/03 06/04
résolution du problème d'ouverture fermeture de la fenetre 23/03 27/04
Prise en main du capteur de position 2 (Waterproof) 06/04 06/04
Implémentation du programme Q-learning sur la serre pour le nivellement automatique 04/05 27/04
févr.-2022
mars-2022
avr.-2022
mai-2022
juin-2022
2 9 16 23 2 9 16 23 30 6 13 20 27 4 11 18 25 1

ANNEXE 2

Coût de nos achats

Achats	Prix
Seau avec anse	8,50 €
1 plant de tomate	2,50 €
12 plants de persil plat	2,00 €
12 plants de salade	2,00 €
1 nénuphar	24,95 €
TOTAL	**39,95 €**

Coût des achats suivants

Achats	Prix
Capteur pH	20€ - 30€
Capteur ORP	70€ - 100€
Substrat aquatique	≈ 10€
TOTAL	100€ - 140€

ANNEXE 3

Fiche de présence des membres du groupe pendant la période du projet

Présence et horaire	févr.-2022				mars-2022					avr.-2022				mai-2022		
	2	9	16	23	2	9	16	23	30	6	13	20	27	4	11	18
Guillaume	8h15 - 18h	8h15 - 16h30	9h - 14h	Vacance	8h15 - 17h	9h30 - 16h	distanciel	9h - 16h	9h - 16h	9h - 16h	9h - 16h	Vacance	9h - 16h	9h - 16h	9h - 18h	8h30h - 18h
Lella	8h15 - 18h	8h15 - 16h30	9h - 14h	Vacance	8h15 - 17h	9h30 - 17h	08h30 - 17h	9h - 16h	9h - 16h	9h - 16h	9h - 16h	Vacance	9h - 16h	9h - 16h	9h - 18h	8h30h - 18h
Matéo	8h15 - 18h	abs	9h - 14h	Vacance	8h15 - 17h	distanciel	08h30 - 17h	9h - 16h	9h - 16h	9h - 16h	9h - 16h	Vacance	9h - 16h	9h - 16h	9h - 18h	8h30h - 18h
Victor	8h15 - 18h	8h15 - 16h30	9h - 14h	Vacance	8h15 - 17h	9h30 - 18h	08h30 - 17h	9h - 16h	9h - 16h	9h - 16h	9h - 16h	Vacance	9h - 16h	9h - 16h	9h - 18h	8h30h - 18h
Jérémy	8h15 - 18h	8h15 - 16h30	9h - 14h	Vacance	8h15 - 17h	9h30 - 18h	08h30 - 17h	9h - 16h	9h - 16h	9h - 16h	9h - 16h	Vacance	9h - 16h	9h - 16h	9h - 18h	8h30h - 18h
Théo	8h15 - 18h	8h15 - 16h30	9h - 14h	Vacance	8h15 - 17h	9h30 - 18h	08h30 - 17h	9h - 16h	9h - 16h	9h - 16h	9h - 16h	Vacance	9h - 16h	9h - 16h	9h - 18h	8h30h - 18h

ANNEXE 4

```python
def reward(self, action) -> float:

    # self.state = [Volume réservoir, prévision pluie, prévision production pv, pompe on-off]
    # Pompe active: action == 1, sinon action == 0

    global action_avant

    if action_avant == 1 and action_avant == action: #récompence si il continue de remplir le bassin
        return 10*(self.state[0])/(V_bassin) # on multiplie la récompense 10 par une valeur entre 0 et 1

    action_avant = action # pour avoir la valeur précédente de l'action

    if (self.state[1] <= 0.25*V_tank) and (self.state[3] <= 1) and (action==1):
        # si le niveau du bassin est bas + il pleut pas
        return -500*((V_tank-self.state[1])/V_tank)
    else:
        if self.state[0] >= 0.95*V_bassin: # si le niveau du bassin est haut
            if action==0:
                return 10*(self.state[0])/(V_bassin)
            else:
                return -100*(self.state[0]/V_bassin)
        if self.state[0] <= 0.25*V_bassin: # si le niveau du bassin est trop bas
            if self.state[3]>10: # si il fait soleil => évaporation plus forte
                if action==0:
                    return -100*((V_bassin-self.state[0])/V_bassin)
                else:
                    return 50*(self.state[0]/V_bassin)
            else: # si il ne fait pas soleil
                if action==0:
                    return -500*((V_bassin-self.state[0])/V_bassin)
                else:
                    return 100*(self.state[0]/V_bassin)
        else: # si le niveau du bassin est "au milieu"
            if action==0:
                return -10*((V_bassin-self.state[0])/V_bassin)
            else:
                return 100*((V_bassin-self.state[0])/V_bassin)
```

```python
def step(self, action):
    reward = 0 if action is None else self.reward(action)
    rain_prevision = 0
    pv_production_prevision = 0

    self.incr_date()

    self.get_pond().evaporation(0.5) # évaporation du bassin

    #Partie : Simulation de l'action sur env
    if self.__current_date.hour == 0 and self.__current_date.minute == 0:
        rain_prevision = self.get_rain_prevision()
        pv_production_prevision = self.get_production_prevision()

    if action == 1: # si pompe active
        self.get_pond().fill(18) # bassin rempli de 18 L
        self.get_tank().use(18) # réservoir vidé de 18 L

    r=0
    if np.random.uniform()>0.95: # création de la pluie aléatoire
        r = 5*np.random.uniform()
    self.get_tank().fill(r)

    self.state[0] = self.get_pond().get_water_level() # mise à jour du volume dans le bassin
    self.state[1] = self.get_tank().get_water_level() # mise à jour du volume dans le réservoir
    self.state[2]=True if action==1 else False

    self.get_exterior().set_sunshine(pv_production_prevision)
    self.state[4]=self.get_exterior().get_sunshine()

    self.total_reward += reward # ajout des récompenses pour avoir la moyenne finale
    done = None

    return self.state, reward, done, {}
```

ANNEXE 5

Code de l'Application

I. Code Java classe DataInfluxdb (développé sur NetBeans)

```
import org.influxdb.InfluxDB;
import org.influxdb.InfluxDBFactory;
import org.influxdb.dto.Query;
import org.influxdb.dto.QueryResult;

public class DataInfluxdb {
   /*
   final   String   serverURL   =   "http://mhi-srv.g2elab.grenoble-
inp.fr/influx/",
                username = "MHIGreenHouse",
                password = "V%5Z &wk77",
                database = "Greenhouse_monitor";
       final InfluxDB influxDB = InfluxDBFactory.connect(serverURL,
username, password);
        System.out.println(influxDB.query(new          Query("SELECT
mean(vBattery) FROM SensorDataElectric WHERE time >= now() - 1d and
time <= now() GROUP BY time(60m) fill(null)", database)));
    */
   //Variables globales
   String serverURL, username, password, database;
   InfluxDB influxDB;
   //CONSTRUCTEURS
   //Si nouvelle base de donnée
   public  DataInfluxdb(String  serverURL,  String  username,  String
password, String database){
       this.serverURL = serverURL;
       this.username=username;
       this.password=password;
       this.database=database;
       this.influxDB=  InfluxDBFactory.connect(serverURL,  username,
password);
   }
   //Par défaut : celle de la serre
   public DataInfluxdb(){
       this.serverURL       =       "http://mhi-srv.g2elab.grenoble-
inp.fr/influx/";
       this.username="MHIGreenHouse";
       this.password="V%5Z &wk77";
       this.database="Greenhouse_monitor";
       try{
```

```
            this.influxDB=          InfluxDBFactory.connect(serverURL,
username, password);
        } catch (Exception e) {
        e.printStackTrace();
    }

    }

    //METHODES
    public String query(){
        //A completer pour récupérer les données désirées
        QueryResult    data    =    influxDB.query(new    Query("SELECT
mean(vBattery) FROM SensorDataElectric WHERE time >= now() - 1d and
time <= now() GROUP BY time(60m) fill(null)", database));
        //Nettoyage de la donnée
        String  textrecup = data.toString();
        int index = textrecup.indexOf("mean", 43);
        String   valeur   =   (String)   textrecup.subSequence(index+7,
index+10);
        return valeur;
    }
}
```

II. Code Java page d'accueil (développé sur Android Studio)

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    private TextView mGreetingTextView;
    private TextView mtext_prod_pan;
    private TextView mtext_taux_res;
    private TextView mtext_tempe_air;
    private TextView mtext_tempe_water;
    private Button mPlayButton;
    DataInfluxdb madatabase;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //Récupération  des  objets  et  associations  aux  variables
correspondantes
```

```
        mGreetingTextView                                                =
findViewById(R.id.main_textview_greeting);
        mtext_prod_pan = findViewById(R.id.main_textview_prod_pan);
        mtext_taux_res = findViewById(R.id.main_textview_taux_res);
        mtext_tempe_air = findViewById(R.id.main_textview_tempe_air);
        mtext_tempe_water                                                =
findViewById(R.id.main_textview_tempe_water);
        mPlayButton = findViewById(R.id.main_button_play);
        madatabase = new DataInfluxdb();
        //Quand on va cliquer sur le bouton
        mPlayButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Appeler une fonction d'actualisation
                String valeur = madatabase.query();
                //Modification de ces variables
                mtext_prod_pan.append(valeur);
            }
        });
    }
}
```

III. Code xml (développé sur Android Studio)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical">

    <TextView
        android:id="@+id/main_textview_greeting"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:padding="8dp"
        android:background="#1D8348"
        android:textColor="@color/white"
        android:textStyle="bold"
        android:text="Bienvenue dans l'application de votre serre" />

    <TextView
        android:id="@+id/main_textview_prod_pan"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
```

```
        android:layout_marginEnd="16dp"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:hint="@string/nom_donnee1" />
    <TextView
        android:id="@+id/main_textview_taux_res"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:hint="@string/nom_donnee2" />
    <TextView
        android:id="@+id/main_textview_tempe_air"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:hint="@string/nom_donnee3" />
    <TextView
        android:id="@+id/main_textview_tempe_water"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:hint="@string/nom_donnee4" />
    <TextView
        android:id="@+id/main_textview_water_level"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:hint="@string/nom_donnee5" />
    <TextView
        android:id="@+id/main_textview_water_ph"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:padding="8dp"
        android:textColor="@color/black"
        android:textStyle="bold"
        android:hint="@string/nom_donnee6" />
    <Button
        android:id="@+id/main_button_play"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_margin="16dp"
        android:text="Actualiser les données" />
    <ImageView
        android:id="@+id/simpleImageView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/greenhouse" />
</LinearLayout>
```

ANNEXE 6

Protocole de connexion à une Raspberry Pi

Pour se connecter à la Raspberry Pi (RP) et pouvoir implémenter des programmes dessus, il a plusieurs étapes à suivre qui sont :

→ Autoriser la connexion à distance
→ Connecter la RP à un réseau wifi
→ Avoir l'adresse IP de la RP
→ Se connecter à la RP directement depuis son propre ordinateur.

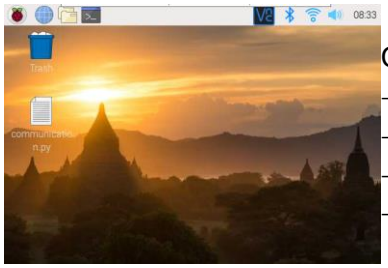
Pour réaliser la connexion, il est nécessaire d'avoir :

→ un clavier
→ une souris
→ un écran
→ un câble HDMI pour relier l'écran et la RP

Ce matériel est nécessaire seulement pour les 2 premières étapes. Donc si vous avez déjà l'adresse IP et que la RP est déjà connectée au réseau wifi il faut juste votre ordinateur.

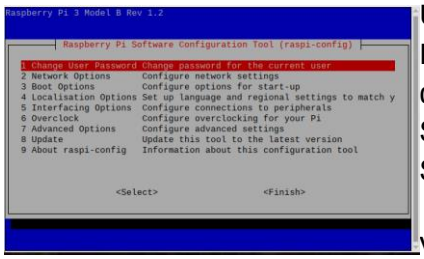
AUTORISER LA CONNEXION A DISTANCE

Branchez le clavier, la souris et l'écran (en utilisant le câble HDMI) et enfin branchez y l'alimentation. La RP est alors allumée et vous devriez voir le bureau de la RP qui ressemble un peu à celui de n'importe quel ordinateur.



On y trouve dans le bandeau en haut en partant de la gauche :
→ le menu d'application
→ le navigateur internet
→ l'explorateur de fichier
→ le terminal de commande

C'est ce dernier que l'on va ouvrir pour autoriser la connexion a distance, taper la commande *sudo raspi-config*



Un menu comme ci dessous devrait apparaître :
Désormais vous allez vous déplacer avec les flèches directionnels du PC :
Selectionner "Interfacing Options">"P3 VNC">"Enable VNC Server"

Voilà votre RP autorise l'accès à VNC Viewer
Dans ce menu vous pouvez configurer le mot de passe (ce qui pourrait être une bonne chose à faire pour se connecter après sur la RP depuis son ordi via VNC Viewer).

CONNECTER LA RP A UN RESEAU WIFI

Cette partie se déroule toujours en ayant le clavier, la souris et l'écran connectés à la RP.

La connexion à distance de la RP depuis l'ordinateur se fait lorsque les deux appareils sont sur le même réseau wifi. Pour cela vous devez cliquer sur l'icône wifi en haut à droite et entrer votre mot de passe.

Une autre manière de se connecter en wifi et d'ouvrir le menu de configuration de la RP avec la commande *sudo raspi-config* dans le terminal. Puis “Network Options”>”Wifi” et il vous faudra entrer le SSID et le mot de passe du réseau wifi.

Attention : On vous conseille de vous connecter à la RP en utilisant un partage de connexion depuis votre téléphone (et donc d’y connecter votre ordi aussi). Le réseau eduroam a tendance a beugué et donc à déconnecter la RP. Si vous voulez travaillez de chez vous sur la RP et bien… vous le pouvez !

OBTENIR L’ADRESSE IP DE LA RP

Cette partie se déroule toujours en ayant le clavier, souris et écran connecter à la RP.

Dans le terminal, taper la commande *ifconfig* `pi@raspberrypi:~ $ ifconfig`
l’adresse ip de la RP se trouve alors dans l’encadré en rouge

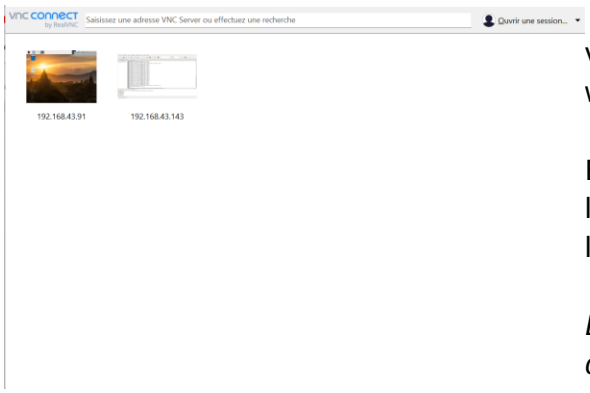
```
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:ba:8d:49  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 17  bytes 1004 (1004.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 17  bytes 1004 (1004.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.91  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 [illegible]:7bab:6426  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:ef:d8:1c  txqueuelen 1000  (Ethernet)
        RX packets 4508  bytes 306118 (298.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5536  bytes 3667391 (3.4 MiB)
```

CONNEXION DE LA RP A DISTANCE

Pour vous connecter à la RasberryPy du haut vous devez d’abord installez le logiciel VNC Viewer :



Avant de continuer, vérifiez bien que la RP et votre ordinateur sont bien sur le même réseau wifi.

Ensuite dans la barre de recherche entrez l'adresse ip de votre RP, il vous sera demandé l'identifiant et le mot de passe.

L'identifiant est bien souvent pi mais il peut être configuré avant si vous voulez le changer via le menu de configuration de la RP

Et voilà c'est fini, vous pouvez désormais vous connecter à distance à votre RP en utilisant VNC Viewer.