2024 – 2025

5A SAGI

POLYTECH ANGERS

Projet EDGAR

LE POT DE FLEURS CONNECTE

Hugo SAVOYE

Théo PERIDY

Hippolyte KUKLA

Tuteur : David LANGIN

université angers

Remerciements

Un grand merci à David Langin pour son accompagnement et son suivi tout au long de ce projet.

Merci également à Hassan Bouljroufi et Franck Mercier pour leur aide précieuse sur la partie électronique, à Boris Rayer pour son expertise sur la partie fabrication du prototype et à Alain Godon d'avoir accepté ce projet.

Nous remercions également toutes les personnes qui nous ont introduits au monde passionnant des plantes :

Rim Baccar, pour avoir patiemment répondu à toutes nos questions de novices.

Rose Hélène Morice, pour son aide précieuse dans la mise en relation.

Mathurin Prieur, pour la visite des serres et son expertise en botanique.

Romane Fabrés, pour son soutien dans la mise en relation avec l'Institut Agro.

Julie Tranchepain et Melody Gouarin, pour la visite enrichissante de leur club d'hydroponie.

Merci à tous !

Liste des sigles

Sigles	Définition
FDM/DFF	Dépôt de fil fondu
FDA	Food and Drug Administration
PWA	Progressive Web App
LED/DEL	Light Emitting Diode/Diode électroluminescente
API	Application Programming Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
JS	Javascript
UI	User Interface
UX	User Experience
PC	Personal Computer
SDA	Sélection directe à l'arrivée (spécifique réseau RNIS)
RNIS	Réseau numérique à intégration de services

Table des matières

Introduction

Au cours de notre dernière année à l'école d'ingénieurs Polytech à Angers en spécialité "Systèmes automatisés et génie informatique", nous avons été amenés à travailler autour d'un projet technique en groupe d'une durée de quatre semaines, afin de mettre en pratique nos compétences et méthodologies acquises durant notre cycle d'ingénieur. Nous étions trois étudiants à travailler sur ce projet : Hugo SAVOYE, Hippolyte KUKLA et Théo PERIDY.

Étant étudiants en école d'ingénieurs, nous avons peu de temps et de connaissances en matière de jardinage. Maintenir une plante en bonne santé nécessite une surveillance régulière de l'humidité du sol, de la lumière et de la température.

Nous avons donc décidé de développer un pot de fleurs connecté permettant de suivre en temps réel l'état d'une plante via une plateforme web. Ce système repose sur une carte Arduino équipée de capteurs, qui collecte et transmet les données grâce à un module Sigfox. L'utilisateur peut ainsi consulter ces informations à distance et être alerté en cas de besoin.

À travers ce projet, nous avons pu mettre en pratique nos compétences en conception 3D, électronique, programmation embarquée et développement web.

Ce rapport a pour objectif de présenter le déroulement du projet, les choix techniques opérés, les problèmes rencontrés, ainsi que les solutions mises en œuvre pour réaliser ce projet.

1. Présentation du projet

1.1. Contexte

Après plusieurs sessions de brainstorming, nous ne trouvions pas d'idée qui nous conviendrait à tous. Chacun avait ses conditions : Hippolyte souhaitait travailler sur du prototypage en impression 3D, Hugo était attiré par l'informatique, et Théo voulait de l'automatisme.

C'est alors que nous nous sommes rappelé le cours sur les objets connectés, qui regroupait justement les compétences qui nous intéressaient. Deux idées principales sont ressorties : une ruche connectée et un pot de fleurs connecté. Nous avons opté pour le pot connecté, notamment pour son lien avec la botanique et la présence de l'Institut Agro sur notre campus universitaire.

1.2. Cahier des charges

L'objectif de notre pot connecté était de faire que la plante soit aussi autonome que possible. Pour y parvenir, nous devions intégrer des capteurs capables de mesurer l'humidité, le niveau d'eau, la luminosité et la température.

Nous devions également concevoir un pot et/ou un boîtier intégrant ces capteurs, ainsi qu'un système de communication entre l'électronique et le serveur. Cela permettrait d'afficher les données collectées sur une interface accessible par web et téléphone mobile.

Pour mettre en place toutes nos idées, nous avons créé une carte mentale (voir Annexe) regroupant les besoins et les objectifs de départ. Cela nous a permis d'identifier toutes les ressources nécessaires pour la réalisation de ce projet.

1.3. Organisation du projet

1.3.1. Discord – suivi de projet

Pour assurer le suivi de notre projet, nous avons mis en place un serveur Discord. Notre organisation reposait sur deux aspects principaux : la centralisation des données et un système d'agenda avec un journal de bord.

Le serveur Discord regroupait tout : les choix technologiques, la sélection des capteurs, les liens vers les différentes documentations... Pour le suivi quotidien, nous faisions des réunions de cinq minutes en fin de journée afin de faire le point sur les tâches réalisées et de planifier celles du lendemain. Ce bilan quotidien nous permettait d'assurer une progression continue.

Les bilans hebdomadaires, quant à eux, étaient réalisés lors de visioconférences ou de visites avec M. Langin.

1.3.2. Gantt

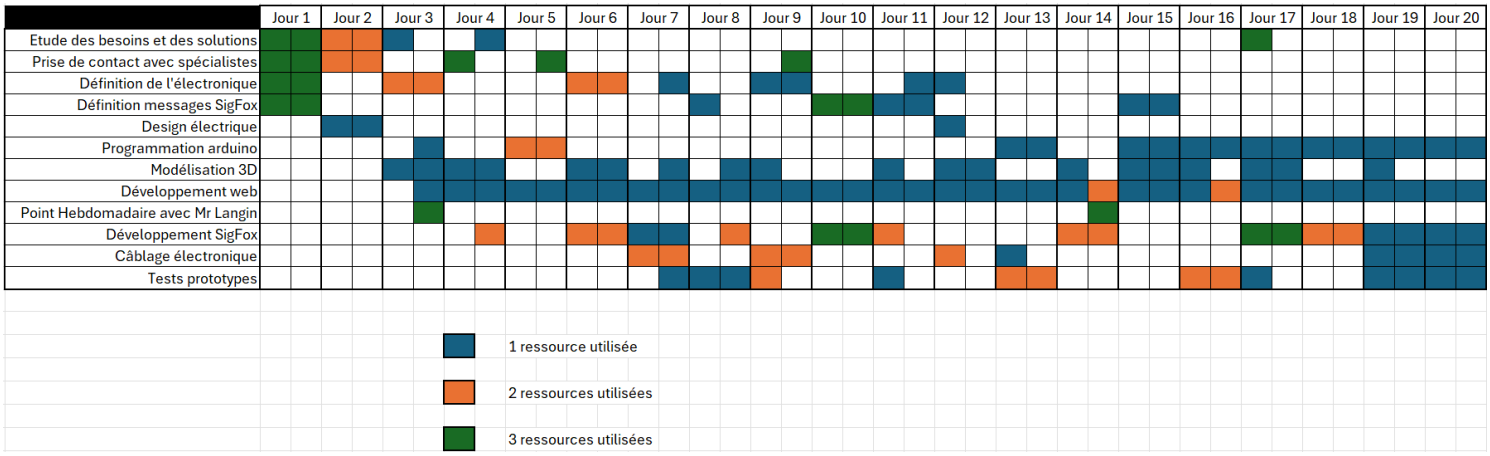


Figure 1. Diagramme Gantt du Projet

1.3.3. Graph camembert pour les tâches

En parallèle, une répartition des tâches en pourcentage a été établie pour quantifier l'effort nécessaire pour chaque phase du projet. Les pourcentages correspondent aux ressources utilisées pour chaque tâche.

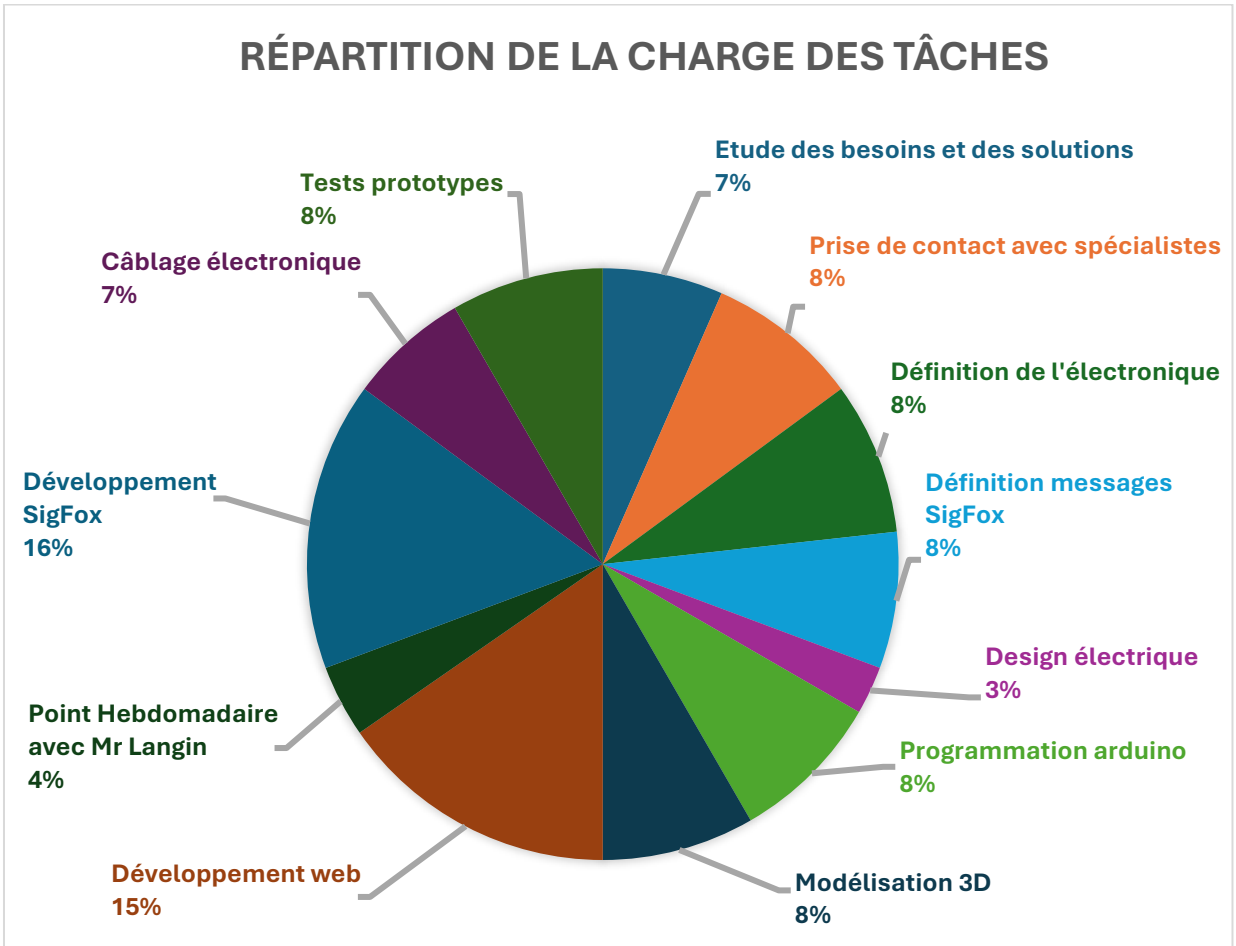


Figure 2. Diagramme camembert de la répartition de la charge des tâches

2.Sélection des technologies

Une fois notre cahier des charges défini, nous nous sommes intéressés aux différentes technologies présentes sur le marché pour concevoir notre pot connecté.

2.1. Besoins des plantes

Néanmoins, ayant peu de connaissances sur le besoin des plantes, nous nous sommes adressés à des professionnelles afin de pouvoir choisir, en conséquence, les différentes technologies.

Nous avons décidé de nous tourner vers trois possibilités pour obtenir des informations. La première est de s'informer via le moteur de recherche en naviguant de source en source. Ensuite, nous avons contacté des spécialistes via LinkedIn et en cherchant des documents de recherche nous avons pu trouver les noms de personnes compétentes sur le sujet. Enfin, nous sommes allés sur le site de l'institut agro Rennes Angers, aussi appelé Agrocampus, école d'ingénieur en agronomie, alimentation, horticulture et paysage. Nous avons été accueillis par des étudiants qui nous ont permis de lier des liens avec des étudiants passionnés qui nous ont accompagnés sur le projet.

À la suite de nos différents entretiens et recherches, voici un résumé de ce que l'on a pu trouver :

- Toutes les plantes sont différentes, les besoins sont propres à chacune des plantes et l'utilisateur devra être en mesure de paramétrer les critères en plus de ceux de base.
- La plante est influencée par la quantité de lumière qu'elle reçoit.
- La plante a besoin de cycle de journée, avoir une alternance jour/nuit.
- La plante est influencée par la quantité de nutriments présents dans le substrat.
- Certaines plantes ont besoin de geler pour pouvoir produire des fruits.
- L'arrosage est primordial à gérer correctement, certaines plantes ont besoin d'un niveau d'humidité constant quand d'autres ont besoin d'atteindre un niveau d'humidité proche de zéro.
- Le substrat doit rester aéré pour permettre aux racines de capter le CO_2 présent dans l'air.

2.2. Choix des composants

Pour mettre en œuvre notre solution de pot connecté, nous avons sélectionné des composants adaptés à notre projet, en privilégiant des outils simples et avec lesquels nous sommes familiers. Nos choix ont été guidés par des critères de compatibilité, de consommation énergétique et de fiabilité pour assurer le bon fonctionnement du système.

2.2.1. Microcontrôleur

Le microcontrôleur joue un rôle central dans notre système puisqu'il assure la lecture des capteurs, le traitement des données et la communication avec le serveur. Nous avons choisi d'utiliser une carte ***Arduino Uno***, notamment pour sa **facilité de programmation**, sa **large documentation** et sa **compatibilité** avec de nombreux capteurs et modules de communication.

Figure 3 Uno

2.2.2. Capteurs et actionneurs

Ensuite, pour permettre un suivi précis de l'état de la plante, nous avons choisi plusieurs capteurs à intégrer dans notre système :

- **Capteur de luminosité (TSL2561) :** Évalue l'exposition à la lumière pour ajuster l'éclairage si nécessaire.

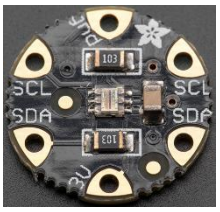


Figure 4 : Capteur de luminosité TSL2561

- **Capteur d'humidité du sol (SEN0114) :** Permets de mesurer la teneur en eau dans le sol afin de savoir s'il faut arroser la plante ou non.

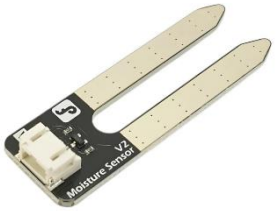


Figure 5 : Capteur d'humidité marque DFRobot

- **Capteur de température (LM35) :** Utile pour indiquer la température de la pièce et déplacer la plante en conséquence dans un endroit plus chaud ou plus froid.

Figure 6 : Capteur de température LM35

- **Capteur de niveau d'eau (SEN0205) :** Permets de connaître le niveau d'eau restant dans le réservoir d'eau.

Figure 7 : Capteur de niveau d'eau SEN0205

- **Mini-pompe 5V :** Cette mini-pompe contrôlée par un module relais permet d'alimenter en eau la plante.

Figure 8 : Mini-pompe 5v

- **LED 24v :** Enfin, la LED 24V commander également par un relais, apporte de la lumière à votre végétal lorsqu'il fait trop sombre en journée.

2.2.3. Communication

Pour transmettre toutes ces données collectées par les capteurs à un serveur distant, nous avons opté pour un module Sigfox, dont nous avons eu connaissance en cours d'objets connectés. Il possède une faible consommation énergétique et une longue portée permettant une transmission des données sans Wifi.

Figure 9 : Module Sigbee Sigfox

2.2.4. Alimentation

Le choix de l'alimentation a été difficile puisqu'au début, nous voulions partir sur une alimentation par batterie, cependant, nous avons remarqué qu'avec les nombreux capteurs, actionneurs, le microcontrôleur et l'éclairage, une batterie n'aurait pas suffi.

Composants	Alimentation
Module Sigfox, capteur luminosité	3,3v
LED	24v
Carte Arduino, Mini-pompe, Capteur d'humidité, de température, de niveau d'eau	5v

Nous avons donc choisi d'utiliser une alimentation TRACO POWER TXL 100-0524DI alimentant notre système en 24 V CC ou 5 V CC.

Figure 10 : Bloc d'alimentation

Nous avons également utilisé un régulateur 5v -> 3.3v pour alimenter le module Sigfox. Nous avons emprunté à Polytech ces 2 éléments, ce qui nous a évité d'effectuer une nouvelle commande.

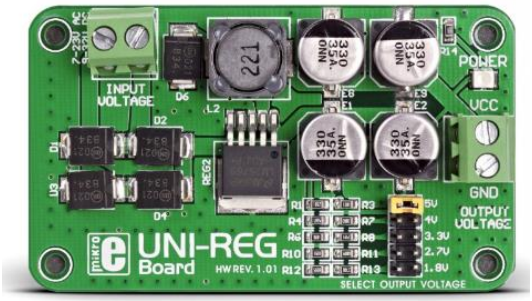


Figure 11 : Régulateur de tension

Autres que ces composants, nous avions également besoin de matériaux pour réaliser le pot avec une imprimante 3D. C'est pourquoi nous allons maintenant examiner leurs contraintes.

2.3. Contraintes des matériaux

Ce projet s'inscrit dans l'environnement du vivant, les plantes sont sensibles à la qualité du sol et de l'eau qui leur est apportée.

2.3.1. Prototypage par impression 3D

Dans le cadre de notre projet, l'impression 3D FDM représente plusieurs avantages, faible coût, rapidité et recyclage des prototypes.

Cependant, bien que les matières des filaments soient d'origine naturelle, l'utilisation de cette technologie représente des enjeux :

« Le PETG, et parfois le PLA, sont salués dans la communauté de l'impression 3D pour être chimiquement neutres et en tant que matériaux pratiquement adaptés au contact alimentaire. Vos bouteilles de soda en plastique, vos contenants alimentaires et vos boîtes à salade sont fabriqués à partir de PET, un proche parent du PETG.

Le problème, avec tous les objets imprimés en 3D, en général, est qu'ils sont fabriqués à partir de couches. Les rainures entre ces couches sont un lit de semence pour les bactéries, car elles sont presque impossibles à nettoyer correctement et contiennent des résidus.

Un autre problème est celui des pigments de couleur et des additifs inconnus. Ceux-ci pourraient ne pas être aussi neutres pour votre corps que le plastique pur. Il est également difficile de dire exactement combien il y a de ces substances. Le filament PETG naturel/pur sans aucune couleur serait votre meilleure solution, mais il est difficile de dire exactement ce qu'il contient. »

D'après un article de Prusa Research

Ces enjeux sont importants dans des applications alimentaires, mais est-ce qu'on peut l'appliquer pour l'utilisation botanique ?

D'après INERIS, l'Institut National de l'Environnement Industriel et des Risques, la plante est susceptible d'absorber les polluants présents dans le sol via la dilution dans l'eau qui alimente la plante.

C'est pourquoi si l'on utilise l'impression 3D pour la solution finale du produit, il serait primordial de s'orienter vers du filament approuvé par la FDA, acronyme de Food and Drug Administration. Comme précisé dans l'article de Prusa Research, même si le filament est lui-même compatible au contact alimentaire, la méthode d'impression FDM va créer de petits interstices qui seront un nid à bactérie, mais pour de l'application botanique cela ne pose pas de problème.

2.3.2. Est-ce que l'impression 3D est une solution finale ?

Possible, la solution de l'impression 3D permet à n'importe qui ayant une imprimante 3D de reproduire notre projet chez lui. Dans le cas où l'on souhaitera vendre notre produit, cela pourrait être adressé à des personnes aimant les tutos et notre solution serait vendue en kit. Cependant, l'impression 3D FDM est assez inconstante dans le comportement après impression. Comme des problèmes d'étanchéité, ce qui est très gênant dans notre projet.

La solution serait d'utiliser d'autres types de technologies comme des plastiques via injection de thermodurcissables. La plus grosse contrainte sera la buse d'arrosage que nous présenterons ensuite dans la partie sur la conception du pot qui a des contraintes assez spécifiques.

2.3.3. Réutilisation du surplus d'arrosage

Lors d'un arrosage d'une plante en pot, le pot laisse s'écouler l'eau en surplus par les trous en bas du pot, c'est ce qu'on appelle le **drainage**.

Figure 12. Schéma du drainage d'une plante en pot

Afin d'optimiser l'utilisation de l'eau, nous avons choisi de renvoyer l'eau drainée dans le réservoir. Cependant, l'eau qui est drainée peut être chargée du substrat (la terre), il faut donc la filtrer efficacement pour ne pas abimer ou boucher la pompe utilisée pour l'arrosage. Pour choisir le filtre, nous nous sommes tournés vers les élèves de l'Agrocampus qui nous ont conseillés un géotextile. Ce sont des tissus en polymères qui sont perméables aux fluides, qui sont faciles à découper et à exploiter.

2.4. Technologies pour application, api et serveur

2.4.1. Choix des technologies web / mobile

À l'origine, nous recherchions une technologie permettant de développer une application web pouvant également fonctionner comme une application mobile. Malheureusement, aucune solution parfaite n'existe. Une approche combinant un site web et une application mobile native offrirait de meilleures performances, mais le développement serait trop long.

Dans mon cas, deux options se démarquaient :

- Site web + application hybride.
- Site web avec une Progressive Web App (PWA).

L'avantage principal de l'application hybride est l'accès aux fonctionnalités natives du mobile, grâce à un code directement intégré au système de l'appareil. De son côté, la PWA offre une installation en un clic depuis le navigateur, bien que cette simplicité ne soit pas aussi « magique » qu'il n'y paraît. Son autre point fort est qu'elle repose uniquement sur du code web, simplifiant ainsi le développement.

Cependant, les deux solutions partagent un inconvénient majeur : les performances. Comme leur affichage dépend du navigateur et de la connexion, elles ne peuvent jamais rivaliser avec une application native en termes de fluidité et de réactivité.

Finalement, nous avons opté pour la PWA, car nous n'avions pas de besoin spécifique nécessitant les compatibilités natives offertes par une application hybride.

2.4.2. Choix des langages

Puisque nous avons choisi de développer une PWA, j'avais accès à l'ensemble des technologies du web. Et il y en a énormément, avec la multitude de frameworks existants.

N'ayant pas une grande expertise en développement web, nous avons décidé de rester sur les bases. Pour le front-end, nous avons opté pour une combinaison classique : HTML, CSS et JavaScript. Côté back-end, nous avons choisi Node.js et MongoDB.

Nous avons repris les technologies que nous avions utilisées dans le cadre des objets connectés, ce qui nous a permis de gagner du temps et de faciliter le développement.

2.4.3. Serveur / API

Pour le serveur, c'est essentiellement du bricolage, une technique que j'utilisais déjà. J'héberge le front-end sur GitHub Pages, ce qui nous a permis d'obtenir un nom de domaine et un hébergement gratuit. Cependant, GitHub Pages ne supporte que les sites statiques, c'est pourquoi seul le front y est déployé.

Le back-end, quant à lui, tourne en local sur mon PC. Pour permettre les requêtes entre le front et le back, j'utilise ngrok, qui nous génère un nom de domaine accessible depuis l'extérieur.

Concernant les API, la tâche s'est avérée compliquée. Une grande partie des connaissances en botanique est stockée dans des livres, et nous n'avions trouvé que très peu d'API adaptées. Celles que nous avons testées étaient souvent mal conçues, avec peu de données ou ne correspondant pas à nos besoins. Nous recherchions une API capable de nous fournir des informations précises sur les besoins en eau et lumière des plantes, sachant que nous étions trois novices dans ce domaine.

Finalement, nous avons trouvé l'API plant.id qui semblait convenir, mais elle était limitée à 100 requêtes par mois en version gratuite, ce qui est très peu. Malgré cette contrainte, c'était la seule option qui répondait à nos attentes.

3. Réalisation

3.1. Conception du pot connecté

3.1.4. Tests d'étanchéité et d'écoulement

Avant de pouvoir sélectionner l'impression 3D pour le prototypage, nous avons réalisé un essai pour évaluer l'étanchéité :

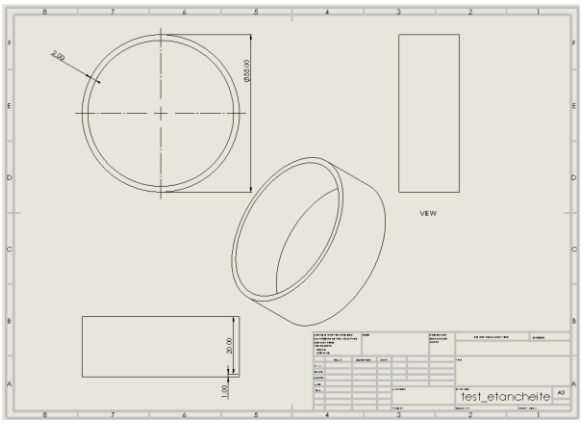


Figure 13. Prototype test étanchéité

Une fois la pièce imprimée, nous l'avons remplie d'eau et l'avons laissée sur une feuille d'essuis pendant quelques jours afin d'observer s'il y avait une présence de fuite ou un écoulement par capillarité, ce n'était pas le cas, nous avons donc conclu qu'on pouvait obtenir une solution étanche avec l'impression 3D FDM.

Dans le même temps, nous avons réalisé des tests pour l'écoulement du drainage. Nous avons exploré deux solutions qui à l'origine visaient à remplacer l'utilisation d'un filtre. La première reprend le principe d'un goulot d'étranglement, nous espérions que la terre ferait un blocage, laissant passer uniquement l'eau. Cependant cette solution ne fonctionnait pas, après différents tests, le substrat était emporté par le drainage. La deuxième solution explorée est de reproduire le principe d'un filtre en laissant uniquement le passage de l'eau via de petits trous dans un quadrillage. Finalement, soit les trous étaient trop petits, ne laissant pas le drainage s'évacuer assez rapidement, ce qui est mauvais pour la plante ; soit les trous étaient trop grands laissant fuir le substrat avec le drainage, comme pour la première solution.

De plus, en étudiant la solution avec les élèves de l'Agrocampus, ils nous ont expliqué que le plus optimal était d'utiliser les perçages classiques des pots de fleurs qui permettent un drainage efficace et d'utiliser un filtre afin d'assurer la qualité de l'eau du réservoir.

Figure 14. Premier et deuxième test de drainage

3.1.5. Adaptation aux capteurs et actionneurs

La première étape était de prendre des mesures des capteurs et actionneurs qui seront utilisés dans le projet. Vous pouvez trouver les dessins techniques en annexe. Voici un tableau résumant les contraintes des différents modules et les solutions apportées :

Capteur/Actionneur	Contrainte	Solution
Capteur d'humidité	Dois être en contact avec le substrat dans le pot	Faire passer le capteur par le fond du pot en chauffant la zone et ajout de colle autour des branches de la fourche pour éviter la fuite.
Capteur de présence d'eau	Dois pouvoir évaluer le niveau d'eau efficacement	Les capteurs sont chacun placés à un tiers du réservoir (respectivement 1/3, 2/3 et 3/3). Le premier capteur n'est pas placé à 0 afin que la pompe soit toujours sous le niveau de l'eau, car elle ne doit pas fonctionner à vide.
Capteur de température	Le capteur ne doit pas être perturbé par la chaleur générée par l'alimentation ou les différents composants électroniques du pot.	Le capteur est placé en surface du pot, le plus éloigné possible des autres composants.
Capteur de luminosité	Le capteur doit pouvoir mesurer la luminosité ambiante, mais ne doit pas être influencé par le système d'éclairage lorsque celui-ci est allumé.	Le capteur est placé sur la courbe bombée de la partie inférieure du pot, à un endroit où il peut capter efficacement la luminosité ambiante et qu'il soit le minimum atteint par l'éclairage d'appoint.
Pompe	La pompe doit accéder au réservoir, mais son contrôle est fait via un câble. Il faut gérer l'étanchéité du réservoir tout en assurant son bon fonctionnement.	La pompe est montée en force contre le réservoir avec un joint d'étanchéité.

Éclairage LED	L'éclairage doit être placé au-dessus de la plante et à hauteur réglable pour adapter à la plante.	La fixation de l'éclairage est modulaire via des pièces qui se vissent.

3.1.6. Prototype

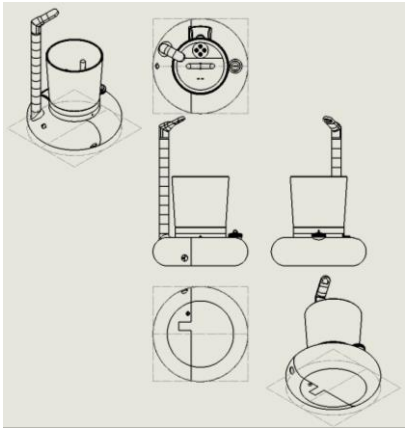


Figure 15. Plan du prototype final

Vous pouvez trouver en annexe le dessin technique de chacune des pièces.

La buse d'arrosage :

Notre solution apporte l'eau directement au substrat via le dessous, ce qui évite l'évaporation donc optimise l'arrosage. On pourrait se demander si la terre sur le dessus du pot ne serait pas sèche, via la capillarité, l'hydratation va se répartir dans le pot. De plus, la plante va récupérer l'eau via ses racines, donc si dans le substrat l'apport en eau est bien réparti, cela ne posera pas de problème.

La première solution proposée est la suivante, un seul canal qui dessert les deux sorties. L'entrée de la buse est basée sur le diamètre du tuyau qui est contraint par la sortie de la pompe. Le conduit d'arrosage est percé sur toute la longueur pour répartir au mieux l'apport en eau dans le substrat.

Cependant, n'ayant qu'un seul conduit commun, la pression n'était pas constante sur chacun des trous. C'est pourquoi nous avons réfléchi au prototype suivant.

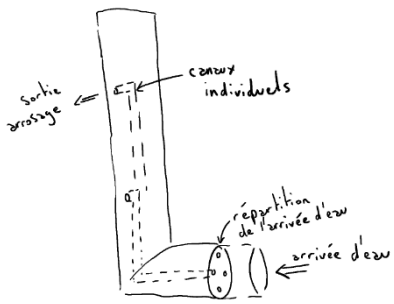


Figure 16. Photo et dessin explicatif du prototype de buse d'arrosage

Ce prototype solutionne la problématique de la pression irrégulière, l'arrivée d'eau se divise en différents canaux, un par sortie. De ce fait, les canaux étant fins, la pression est constante sur toutes les sorties, on obtient un arrosage en goutte à goutte ce qui favorise la répartition efficace du liquide. Nous avons donc choisi cette solution technique pour la version finale du prototype.

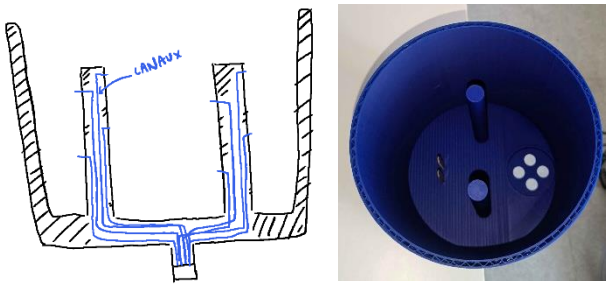


Figure 17. Schéma en vue en coupe et photo du premier prototype de pot

Afin d'intégrer l'arrosage au pot, nous avons choisi d'implanter complètement les buses dans le corps du pot, faisant traverser les canaux d'arrosage dans le fond du pot. Cependant, nous avons observé des défauts d'étanchéité, lors du cycle d'arrosage, au lieu de sortir par les buses, une grande partie de l'eau allait se loger dans le fond du pot, un défaut dû à des erreurs d'impression. Afin d'éviter ce genre de problème, nous avons décidé d'imprimer la buse à part du pot et via un joint assurer l'étanchéité du système. Voir Figure 18 qui suit.

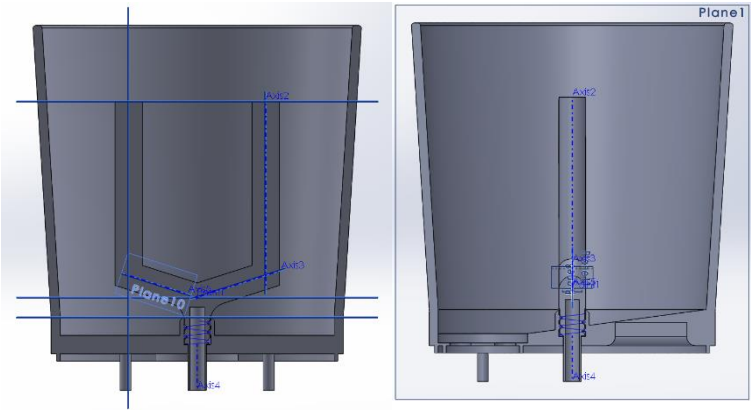


Figure 18. Vue en coupe de la version finale du pot

On observe sur cette figure que la buse est accouplée via un pas de vis. Il y a deux cylindres qui font office de détrompeur pour placer le pot aligné avec la sortie de drainage et la position du filtre. Sur la vue de droite, on voit que le fond du pot est une pente, ce qui facilite le drainage et le guide aux trous d'évacuation. Sur cette même coupe, on voit à droite la zone permettant d'insérer le capteur d'humidité dans le fond du pot.

La Base du pot :

Le pot a une base en forme de bouée, ce qui assure la stabilité, mais aussi une optimisation de la taille du réservoir.

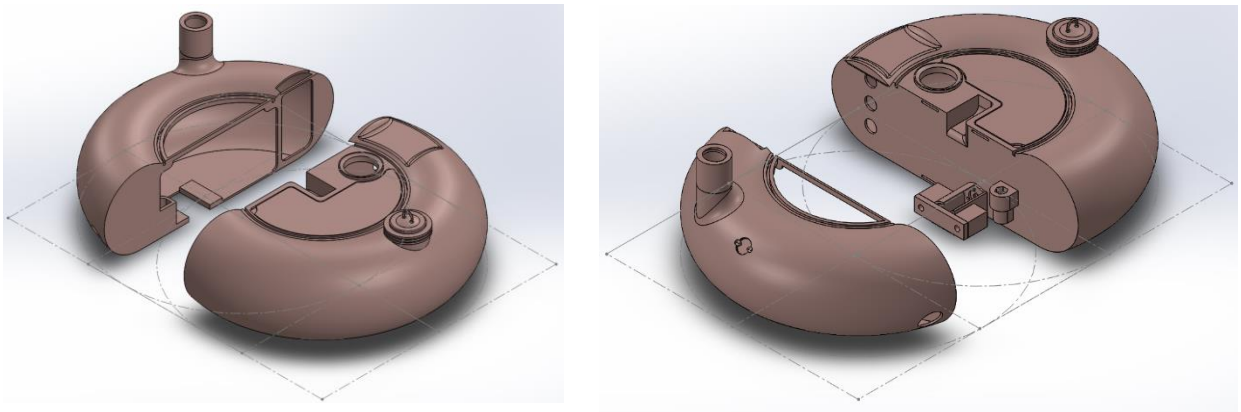


Figure 19. Vue du modèle du réservoir et de la partie stockage électronique

Ces pièces sont contraintes en taille par la surface d'impression des machines disponibles à Polytech, c'est pourquoi nous avons adapté les mesures verticales suivant notre besoin, assurant un volume du réservoir d'environ 2 litres, ce qui permet de nombreux cycles d'arrosages (dépendant de la plante) entre deux remplissages. Sur l'image de droite, on peut observer la partie de fixation de la pompe. Elle reprend la forme de la pompe et assure la pression contre le joint via un jeu de vis-écrou. Sur le réservoir, on observe aussi un bouchon, celui-ci sert à protéger l'eau de la lumière ce qui peut provoquer l'apparition d'algues néfastes pour la pompe. Vous pouvez retrouver en annexe les images avec détails des différents éléments composant les décisions de modélisation.

3.2. Électronique

Après avoir détaillé la conception 3D du pot, cette partie se concentre sur l'intégration électronique du système. Nous y présenterons le câblage des composants ainsi que la gestion de l'alimentation.

3.2.7. Câblage général du système

Par manque de temps, nous n'avons pas réalisé de carte électronique pour embarquer tous les composants dans le pot. Néanmoins, Hassan Bouljroufi nous a aidés à trouver des solutions pour alimenter notre système, et tester certains composants.

Concentrons-nous sur le câblage des composants :

- Les capteurs d'humidité et de température sont connectés sur les broches analogiques de l'Arduino.
- Les capteurs de niveau d'eau et le buzzer sont connectés sur les broches digitales de l'Arduino.
- Pour la lampe et la mini-pompe, ces 2 composants sont connectés à des broches des relais sur le shield. Ainsi, nous pouvons contrôler leurs états à partir d'une broche numérique de l'Arduino.
- Le capteur de luminosité possède une interface I2C pour échanger ses données. Il faut donc le connecter aux broches SDA et SCL de l'Arduino pour récupérer ces informations.
- Pour terminer, la carte Sigfox vient se connecter sur 2 broches digitales de l'Arduino. Nous utilisons une librairie (SoftwareSerial) pour lier le Rx de Sigfox au Tx de l'Arduino et inversement pour le Tx de Sigfox.

3.2.8. Gestion de l'alimentation

Pour assurer le bon fonctionnement du pot connecté, nous devions nous assurer que tous les éléments du système soient correctement alimentés en tension et courant.

Voici un schéma très grossier qui montre comment chaque composant est alimenté en tension :

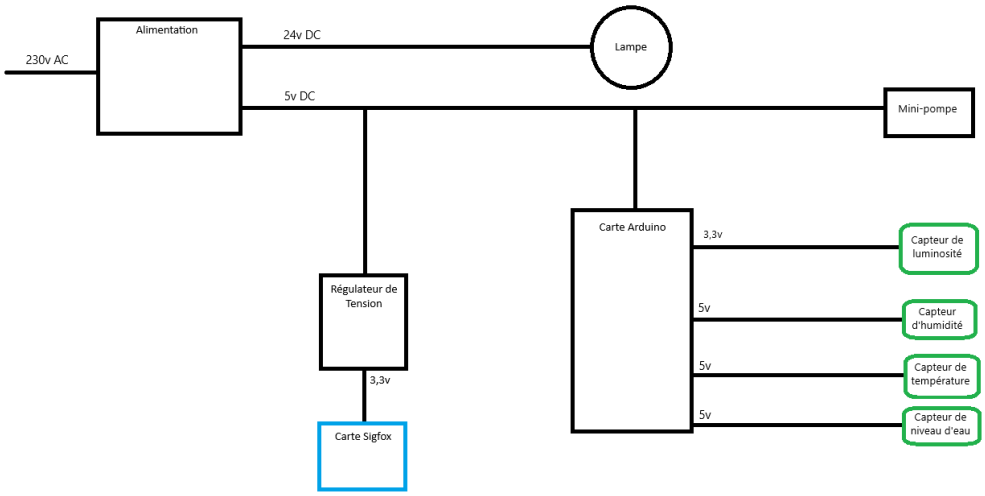


Figure 20 : Schéma d'alimentation

Comme expliqué dans la partie « choix des composants », nous avons choisi d'utiliser une alimentation par secteur et non sur batterie. Les capteurs et la carte Sigfox consomment très peu de courant, cependant, ce n'est pas le cas pour la mini-pompe (150mA), l'Arduino ne fournit pas assez de courant, et la lampe a besoin de 24v pour fonctionner. On ne pouvait pas se permettre d'utiliser une batterie pour notre projet.

3.3. Récupération des données

Une fois l'électronique reçue et testée, nous pouvions commencer la partie programmation de l'Arduino pour récupérer des données et les envoyer grâce à Sigfox à un serveur distant.

3.3.1. Code Arduino

La première étape de cette partie a été de réaliser un code pour récupérer l'ensemble des données provenant des capteurs connectés à la carte Arduino Uno.

Récupération des données des capteurs :

Comme indiqué précédemment, il y a un total de 6 capteurs connectés à l'Arduino (1 capteur d'humidité, 1 capteur de température, 1 capteur de luminosité et 3 capteurs de niveau d'eau). L'objectif a donc été de créer des codes séparément pour faire fonctionner chaque capteur indépendamment, puis de les rassembler dans un programme pour les faire fonctionner « en même temps ». En réalité, les capteurs n'ont pas besoin de fonctionner chaque seconde, car la température, l'humidité ou la luminosité ne vont pas chuter/augmenter d'un seul coup. On pouvait donc se permettre de vérifier chaque donnée l'une après l'autre sans se soucier d'être très précis.

Gestion de l'arrosage et de l'éclairage :

Autre que les capteurs, notre système est muni de 2 actionneurs : une mini-pompe et une lampe à LED.

En raison de la limitation en courant et en tension de l'Arduino, nous avons utilisé un shield relais (Cf. Figure 21) qui permet de contrôler des appareils.

Figure 21 : Shield relais Arduino

Ces 2 éléments fonctionnent par cycle. La mini-pompe est activée selon les paramètres donnés par l'utilisateur. Par exemple, si une plante a souvent besoin d'eau, l'utilisateur va renseigner sur son application qu'il faut l'arroser 2 à 3 fois par semaine. La carte Arduino va prendre en compte ce paramètre et activera la pompe tous les 2/3 jours.

Concernant la lampe, elle s'activera sous certaines conditions. La lampe possède un cycle d'activation (elle aussi indiqué par l'utilisateur), par exemple de 8h à 18h en hiver. À l'aide d'un composant horloge (que nous n'avons malheureusement pas pu commander), la lampe s'activera si le temps actuel se trouve entre 8h et 18h et si la luminosité est inférieure à un certain seuil. Et inversement, si la luminosité dans la pièce est trop forte, la lampe sera désactivée.

Également, une fonction a été créer pour gérer le remplissage du réservoir. Si l'utilisateur rempli le réservoir d'eau, un buzzer émet un son pour l'avertir de plein de celui-ci.

3.3.2. Communication Sigfox

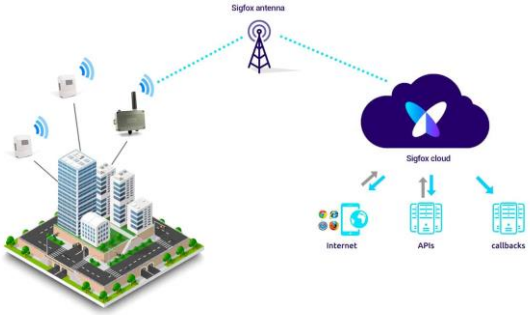
Une fois la partie « récupération des données » terminée, nous nous sommes intéressés à la partie « envoi des données vers le serveur distant ». Pour cela, nous avons décidé d'utiliser un protocole de communication nommé « **Sigfox** ».

Mais qu'est-ce que Sigfox ?

Il s'agit d'une technologie de communication dédiée aux objets connectés. Elle utilise un réseau bas débit pour transmettre de petites quantités de données à faible consommation énergétique et sur une longue portée.

Son fonctionnement est plutôt simple :

- Un microcontrôleur récupère les données de capteurs (température, humidité du sol…).
- Il traite ces données et les envoie au module Sigfox.
- Ce module transmet les données via ondes radio à l'antenne Sigfox la plus proche, qui relaie ces données aux serveurs Sigfox.
- Les serveurs Sigfox stockent ces données et les rendent accessibles sur une plateforme web où l'utilisateur peut les consulter.



La plateforme web Sigfox (Cf. Annexe 8) est utile pour gérer et surveiller les données envoyées par les objets connectés via le réseau Sigfox. Chaque module Sigfox a besoin d'une licence Sigfox pour se connecter à son Cloud. Pour notre projet, nous avons donc associé une licence à notre module sur la plateforme Sigfox, ce qui nous a permis de le configurer comme nous le souhaitions.

Les messages peuvent être différenciés de 2 façons.

Tout d'abord, nous avons les messages **UPLINK** :

Cela signifie qu'un appareil envoie des données vers un serveur. Dans notre cas, l'uplink correspond à l'envoi d'un message depuis le module Sigfox connecté à l'Arduino, vers les serveurs Sigfox.

Néanmoins, nous sommes limités : nous ne pouvons envoyer que 140 messages UPLINK par jour, de taille 12 octets.

Puis nous avons les messages **DOWNLINK** :

Le downlink permet au serveur d'envoyer des données vers un appareil. Ici, le serveur peut envoyer une commande à l'Arduino (comme le temps de cycle d'arrosage).

Ce type d'envoi est aussi limité : seulement 4 messages downlink par jour, avec 8 octets max par message.

Et selon le type de message (uplink ou downlink), nous avons conçu un système d'encodage (expliqué après) permettant d'envoyer les données de la carte Arduino vers le serveur.

3.3.3.Envoie des données via Sigfox

Nous allons donc maintenant vous présenter comment les données ont été formatées avant envoi par le module Sigfox.

Encodage :

La taille des messages étant limitée, nous avons décidé d'encoder les messages en hexadécimal. Arduino peut envoyer différents messages selon les besoins :

➔ Remontée d'informations de la carte Arduino vers le serveur.

Ce message contient des informations collectées par les capteurs à remonter au serveur pour suivre l'évolution des conditions de la plante.

Voici, comment celui-ci est construit :

	Type message	**Température**	**Humidité**	**Luminosité**	**Réservoir**	Total
Nbr de bits	3	10	10	16	2	41
Nbr d'octets						6

Le nombre de bits de chaque donnée correspond à la taille maximale que peut prendre une donnée lorsqu'elle est envoyée. Pour la température, cela signifie qu'elle est codée sur 10 chiffres binaires, ce qui permet de représenter plus de 800 valeurs (de 0.0 à 80.0). Et le nombre d'octets correspond à la taille du message en octets.

Vous trouverez en annexe 9 les différentes valeurs possibles pour chaque variable et à quoi elles correspondent.

➔ Remontée des alertes de la carte Arduino vers le serveur.

Pour ce 2e type de message, il consiste à remonter les alertes lorsqu'elles surviennent.

Les alertes ont été définies en fonction de données bien précises envoyées par le serveur à l'initialisation de l'Arduino. Voici les types d'alarmes :

TempMax	**TempMin**	**HumiditéMin**	**LuminositéMax**	**LuminositéMin**	**RéservoirMin**
La température est trop haute, déplacez la plante dans un endroit plus frais.	La température est trop basse, déplacez la plante dans un endroit plus chaud.	Pour certaines plantes, si ce seuil est dépassé, il faut arroser la plante (activation de la mini-pompe).	Il y a trop de luminosité, déplacer la plante vers un endroit un peu moins éclairé ou Arduino éteint la lampe.	La plante a besoin de luminosité (Allumage de la lampe en conséquence).	Il faut ajouter de l'eau dans le réservoir

Chaque alarme se déclenche donc si un certain seuil est dépassé. Par exemple, pour « températureMax », si la température actuelle est plus grande que la valeur de « températureMax », une alarme se déclenche.

➔ Demande d'initialisation de la carte Arduino vers le serveur.

Enfin, ce 3e message est envoyé au serveur uniquement lors du démarrage de l'Arduino pour récupérer certaines infos importantes comme :

- Les seuils pour les alarmes (températureMin, luminositéMax…)
- Le nombre d'arrosages

- Le temps d'arrosage en minute
- La durée et le début d'éclairage
- Et la date

Chaque message possède en premier la valeur du « type message ». Cela permet au serveur de différencier si le message qu'il reçoit est une remontée d'informations ou une alerte.

- Si « type message » = 1, il s'agit d'une remontée d'infos vers le serveur.
- Si « type message » = 2, il s'agit d'une remontée d'alertes vers le serveur.
- Si « type message » = 3, il s'agit d'une demande d'initialisation.

Explication encodage du 1er type de message (Remontée d'informations de la carte Arduino vers le serveur) :

1) Données d'entrée :

Voici un exemple de données d'entrée :

Type de message	Température	Humidité	Luminosité	Réservoir
1	27,32	625	30 000	1

2) Préparation des valeurs pour l'encodage :

Dans cette partie, nous avions juste besoin de convertir en un nombre entier la température et de ne garder que 3 chiffres significatifs, car nous n'avons pas besoin d'autant de précision.

	Type de message	Température	Humidité	Luminosité	Réservoir
Valeur initiale	1	27,32	625	30 000	1
Valeur finale	1	273	625	30 000	1

3) Conversion en binaire :

Chaque valeur est convertie en binaire :

	Type de message	Température	Humidité	Luminosité	Réservoir
Valeur initiale	1	27,32	625	30 000	1
Valeur en binaire	001	0100010001	1001110001	0111010100110000	01

4) Concaténation des bits :

On concatène toutes les valeurs binaires obtenues dans l'ordre :

001 0100010001 1001110001 0111010100110000 01

5) Conversion en hexadécimal :

Enfin, on découpe le message binaire en groupes de 4 bits en commençant par la gauche et on les convertit en hexadécimal :

0010	0100	0100	0110	1001	1100	0111	0001	0111	0101	0011	0000	01
2	4	4	6	9	C	7	1	7	5	3	0	2

Soit comme message final :

2 44 69 C 71 75 30 2

Vous pouvez voir la liste de tous les messages en Annexe.

Cette partie a été réalisée pour chaque type de message et une fois mise au point, le serveur pouvait s'appuyer dessus pour créer le décodage.

3.3.4. Récupération des données par le serveur

(1) Décodage

Pour décoder les messages côté serveur, Sigfox appelle l'API /message, qui constitue un point d'entrée HTTP GET. Cette API prend en entrée plusieurs paramètres : le device, le time, la data, et la station. L'élément clé de ces paramètres est la data, car elle contient la chaîne hexadécimale représentant les valeurs des capteurs et est limitée à un maximum de 12 octets.

Ensuite, le message est décodé. En premier lieu, on décode l'en-tête du message pour déterminer son type. Si le message est de type 1, cela signifie qu'il s'agit d'un envoi des valeurs des capteurs du pot. Si le message est de type 2, il s'agit d'une alerte indiquant que les valeurs des capteurs ont dépassé un seuil minimal ou maximal.

Selon le type de message, la fonction sizes ajuste les octets alloués pour permettre le décodage du message. Le décodage est effectué par la fonction decode. Celle-ci prend en entrée le code hexadécimal et les tailles définies par sizes. Le processus de décodage consiste à convertir l'hexadécimal en binaire, puis à diviser cette chaîne binaire en segments selon les tailles spécifiées. Ces segments binaires sont ensuite convertis en valeurs décimales.

Les valeurs ainsi extraites sont renvoyées à notre API, qui les enregistre dans la base de données MongoDB.

3.4. Interface utilisateur/serveur

3.4.1. UI/UX

N'ayant pas de compétences en UI/UX, nous ne pouvions pas réinventer la roue pour l'interface. Nous avons donc choisi de m'inspirer de solutions existantes. Dans un premier temps, nous avons analysé plusieurs applications mobiles dédiées aux plantes, puis nous nous sommes intéressés aux avancées en IA. Récemment, certains modèles permettent de générer rapidement une interface et d'itérer dessus. Nous avons ainsi découvert les solutions Lovable et Bolt.

Après les avoir testées, nous avons trouvé Lovable plus performant. Nous avons donc généré un cahier des charges à l'aide de ChatGPT, définissant nos besoins, et nous l'avons soumis à Lovable. Le résultat obtenu était satisfaisant, mais ne correspondait pas aux stacks techniques requises, notamment en HTML, CSS et JavaScript. Nous avons alors repris le design généré et l'ai reproduit manuellement. Cette approche nous a permis de concevoir la page vitrine, qui est ensuite devenue la base des autres pages du projet.

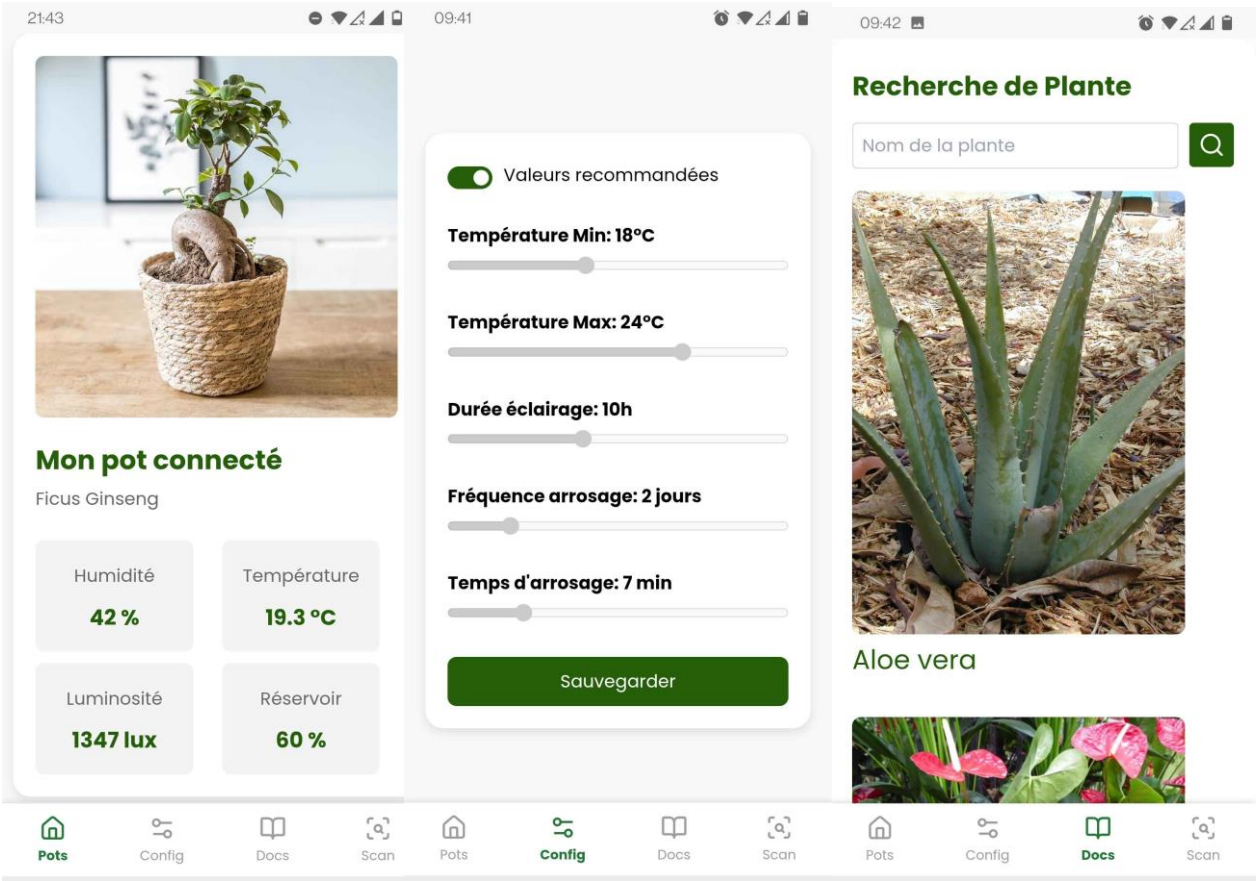


Figure 22 - Interface mobile de l'application

3.4.2. Fonctionnalités principales

Nous avons ensuite développé pot.html, qui interagit avec l'API /last_recherche2 pour récupérer et afficher les dernières valeurs enregistrées en base de données. La page settings.html, quant à elle, permet aux utilisateurs de définir les seuils minimums et maximums pour chaque capteur. Toutefois, bien que l'envoi des valeurs fonctionne, nous avons rencontré des problèmes avec les messages downlink de Sigfox, empêchant la récupération des données envoyées par le serveur. Cette page mériterait d'être améliorée, mais faute de pouvoir la finaliser, nous avons décidé d'explorer d'autres fonctionnalités.

3.4.3. Fonctionnalités secondaires

Nous avons ainsi implémenté la base de données plant.id, qui nous a permis de développer les pages docs.html et scan.html. Ces pages restent incomplètes, mais fonctionnelles. docs.html permet actuellement de rechercher des plantes par leur nom et, à terme, d'afficher leurs besoins spécifiques. scan.html, de son côté, permet de prendre en photo une plante et d'identifier son espèce via l'API, avec leur taux de correspondance. Nous avons également commencé à implémenter une fonctionnalité permettant d'ajouter des photos depuis la galerie, mais nous n'avions pas eu le temps de la finaliser.

3.4.4. Restriction ngrok

Enfin, il est important de noter que l'utilisation de ngrok impose l'ajout d'un header spécifique à chaque requête pour contourner une restriction destinée aux développeurs :

"ngrok-skip-browser-warning":"1"

Cet header permet de désactiver la sécurité et d'éviter les avertissements du navigateur.

4. Bilan

Pour terminer ce rapport, nous allons dans cette dernière partie nous intéresser aux problèmes rencontrés lors de la réalisation de ce projet. Puis nous aborderons dans un bilan technique, les fonctionnalités validées et les améliorations possibles.

4.5. Les problèmes rencontrés

Le projet a rencontré plusieurs problèmes, notamment un manque de compétences en UI/UX, ce qui a conduit à s'inspirer d'applications existantes et à utiliser Lovable pour générer un design. Les fonctionnalités principales, comme pot.html et settings.html nécessitent des améliorations avec l'ajout de l'API plant.id. De plus, les pages secondaires docs.html et scan.html sont incomplètes.

Pour la partie communication Sigfox, nous avons perdu beaucoup de temps à comprendre comment fonctionnait ce protocole. Malheureusement, le module que nous utilisions ne fonctionnait qu'avec une librairie spécifique, ce qui nous a causé des problèmes pour mettre en place le « downlink », permettant au serveur d'envoyer des informations à la carte Arduino.

4.6. Bilan technique

Nous allons maintenant aborder les tâches réalisées et validées.

4.6.5. Ce qui est validé

- Récupération des données des capteurs et fonctionnement de la mini-pompe et de la lampe.
- Encodage des messages à envoyer de l'Arduino vers le serveur grâce à Sigfox.
- Forme actuelle du prototype.
- Sélection des actionneurs et capteurs.
- PWA fonctionnelle
- Interface de visualisation
- Serveur fonctionnel

4.6.6. Améliorations possibles

Malgré les mesures prises pour éviter les fuites, les défauts d'impression et les contraintes de l'impression 3D FDM ont provoqué la présence de fuites sur le réservoir. Une solution est de se tourner vers une autre technologie de fabrication pour assurer l'étanchéité du réservoir et du pot.

N'ayant pas eu le temps d'imprimer de carte électronique pour la gestion de la puissance et des commandes, la partie stockage électronique du prototype n'est pas finie. Si l'on avait la possibilité de pousser le projet plus loin, il faudrait prévoir des éléments de fixations permettant d'optimiser le fonctionnement et l'encombrement de la solution.

Concernant la programmation de l'Arduino, par manque de compréhension et de problème avec le module Sigfox, nous n'avons pas réussi à réaliser la fonctionnalité « downlink ».

Si ce projet était à refaire, on aurait peut-être choisi un autre protocole de communication plus simple et on aurait perdu moins de temps.

Pour le site web et la PWA, il serait essentiel d'améliorer les fonctionnalités principales en les liant avec la base de données. L'objectif serait notamment de permettre l'ajout de son pot connecté en précisant le type de plantes et d'afficher automatiquement leurs besoins spécifiques.

Une autre amélioration importante consisterait à intégrer un système de notifications web push, afin que l'utilisateur puisse recevoir des alertes même lorsqu'il n'est pas sur le site ou l'application.

Enfin, il faudrait finaliser les pages dédiées à l'identification des plantes ainsi que celles répertoriant leurs besoins, afin d'offrir une expérience plus complète et intuitive.

4.6.7. Entrepreneuriat

Récemment, on nous a dit qu'il n'existait ni bonne ni mauvaise idée pour entreprendre, mais que tout dépendait de la manière dont on s'y prend.

C'est en partie vrai, mais ça ne fait pas tout. Nous avons analysé le marché des pots de fleurs et des pots connectés, il apparaît clairement qu'il n'est pas très porteur. À l'époque où les objets connectés étaient à la mode, plusieurs entreprises se sont lancées dans le domaine des pots connectés. Pourtant, la plupart n'ont survécu qu'un ou deux ans. Pourquoi ? Il est difficile de le savoir avec certitude, n'ayant pas fait partie de ces entreprises. Cependant, si autant d'entre elles ont fini en liquidation, c'est probablement parce que la demande n'était pas au rendez-vous.

Pour vérifier cette hypothèse, nous avons contacté un entrepreneur ayant lancé une entreprise de pots de fleurs low-tech. Il nous a confirmé que le marché des pots connectés était compliqué : la plupart des consommateurs préfèrent acheter des plantes bon marché et, une fois celles-ci mortes, en racheter de nouvelles plutôt que d'investir dans des solutions plus chères et complexes pour les entretenir. D'après lui, l'opportunité d'intervention se situe plutôt au moment où la plante commence à mourir, en proposant une expertise et des solutions pour aider les personnes à la sauver et à mieux s'en occuper.

Par ailleurs, créer une entreprise de vente de produits physiques représente un défi majeur, notamment en raison des investissements initiaux importants que cela exige. Avec l'avancement du projet et nos connaissances en botanique, il reste difficile d'évaluer si notre idée pourrait réellement trouver sa place sur le marché. Nous avons mené une partie des recherches nécessaires à la conception du produit, mais pour valider sa viabilité, il faudrait identifier un besoin client précis pour y répondre au mieux.

Conclusion

Travailler sur un projet concret, visant à réaliser un pot de fleurs connecté, nous a permis de mettre en pratique nos connaissances acquises au cours de notre formation à Polytech Angers, mais aussi à découvrir de nouvelles technologies (que ce soit avec la communication Sigfox, mais aussi en termes de programmation web).

Ce projet a été une expérience très enrichissante, bien que très succincte. Nous avons acquis des compétences dans les domaines de l'électronique, les systèmes embarqués, modélisation et impression 3D et développement web.

De la conception du système basé sur l'Arduino à l'intégration des capteurs et actionneurs, en passant par le développement d'une application web et l'impression 3D du pot, chaque étape nous a permis de renforcer notre maitrise sur ces technologies.

L'organisation et l'analyse du besoin nous ont également permis d'adopter une approche professionnelle dans la gestion de projet. Et, au-delà des aspects techniques, cette expérience nous a offert une première immersion dans le domaine de la botanique.

Collectivement, nous sommes fiers des réalisations accomplies, tout en identifiant les pistes d'améliorations possibles. Nous sommes satisfaits du résultat obtenu et d'avoir pu réaliser concrètement ce projet.

Bibliographie

INERIS. (s.d.). *Modèle de transfert sol-plante des polluants organiques*. Récupéré sur https://www.ineris.fr/sites/ineris.fr/files/contribution/Documents/sol_pante.pdf

MongoDB. (s.d.). Récupéré sur https://www.mongodb.com/docs/

Node.js. (s.d.). Récupéré sur https://nodejs.org/docs/latest/api/

PWA. (s.d.). Récupéré sur https://developer.mozilla.org/fr/docs/Web/Progressive_web_apps

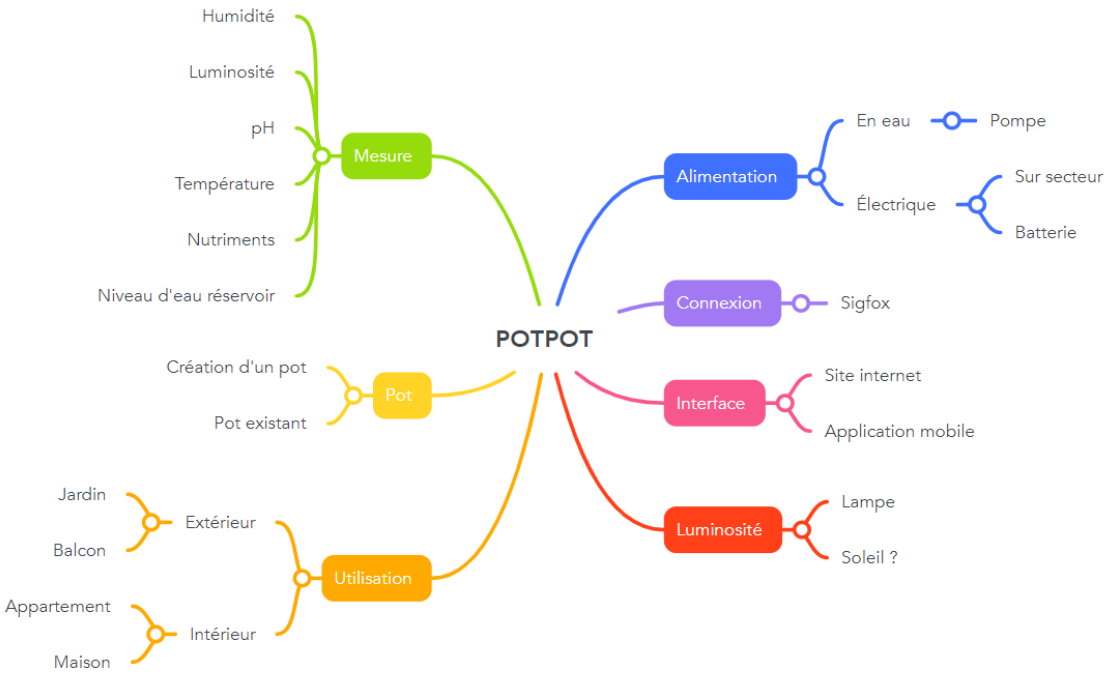
Research, P. (s.d.). *Impression FDM pour le contact alimentaire*. Récupéré sur https://help.prusa3d.com/fr/article/impression-fdm-pour-le-contact-alimentaire_112313

Wikipédia. (s.d.). *Géotextile*. Récupéré sur https://fr.wikipedia.org/wiki/Géotextile

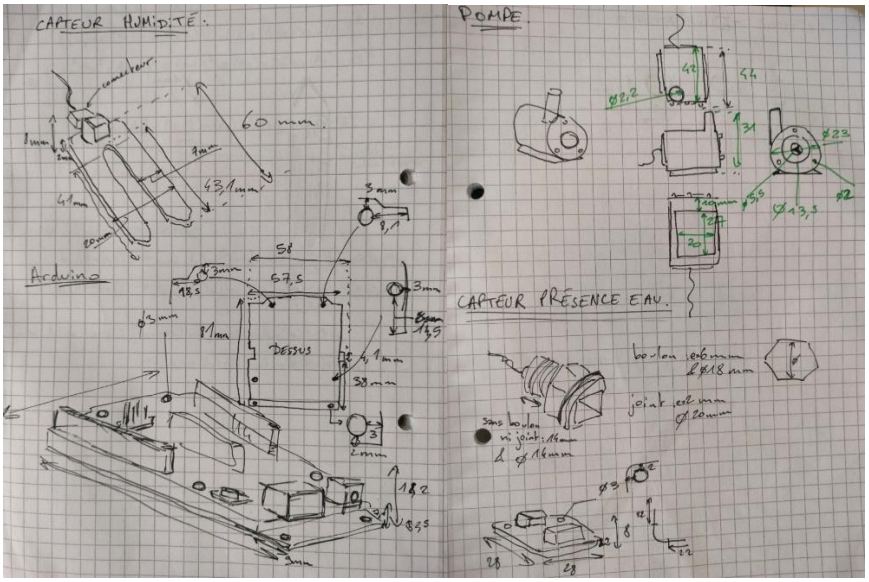
Tables des Illustrations

Table des Annexes

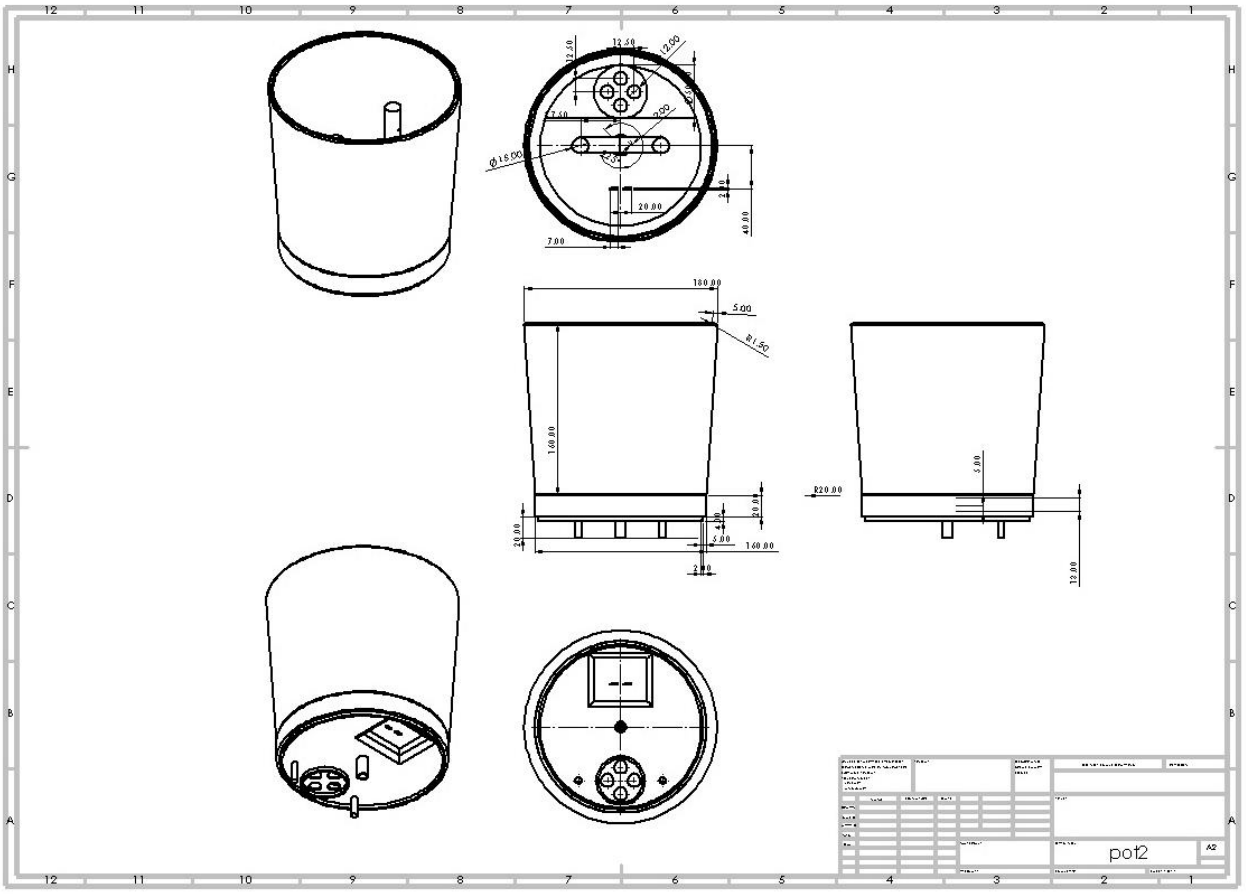
Annexes



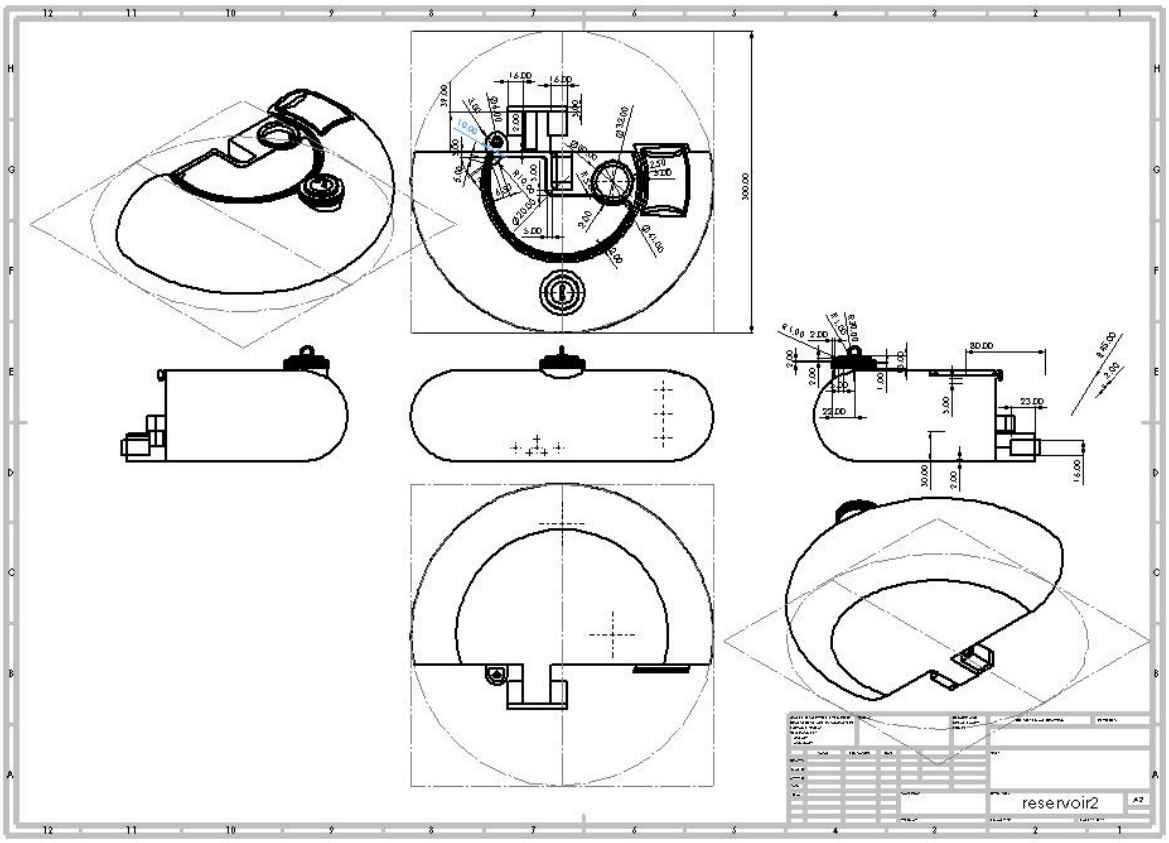
Annexe 1. Carte mentale du projet



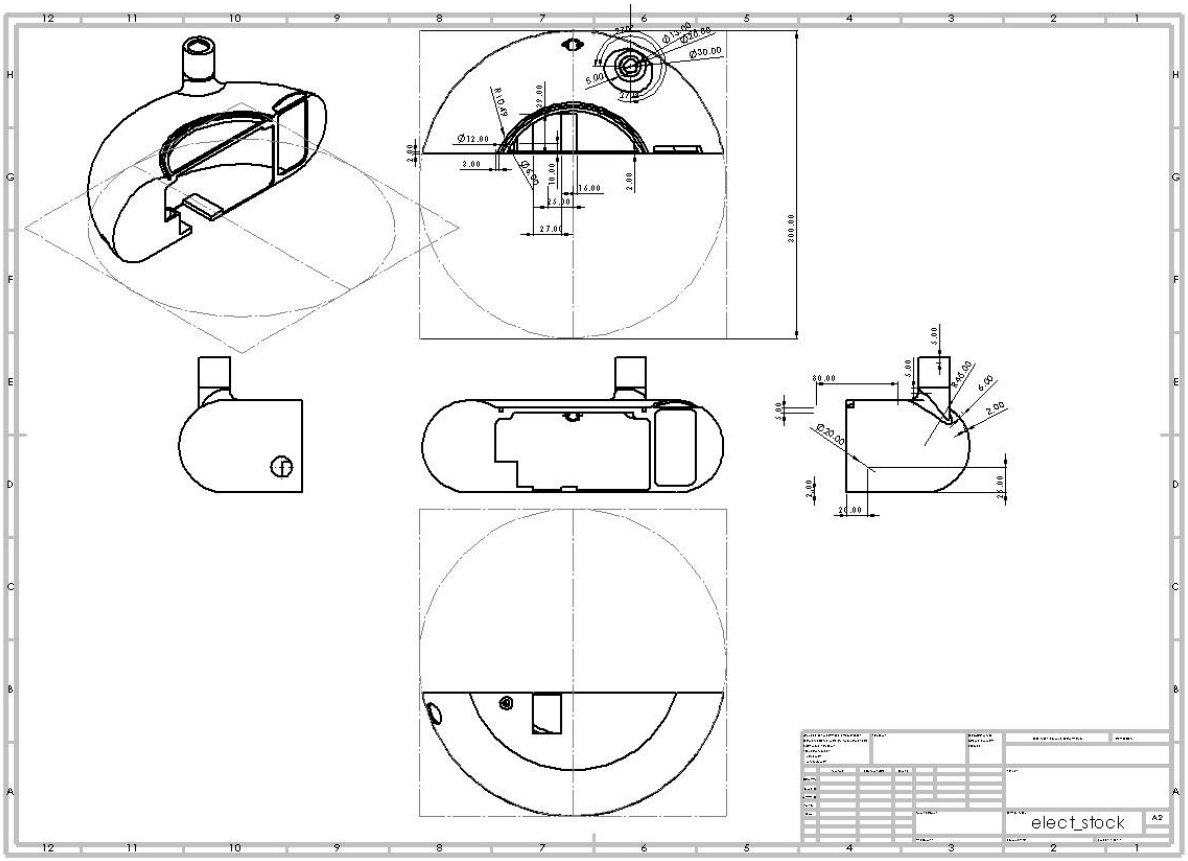
Annexe 2. Dessin technique des capteurs



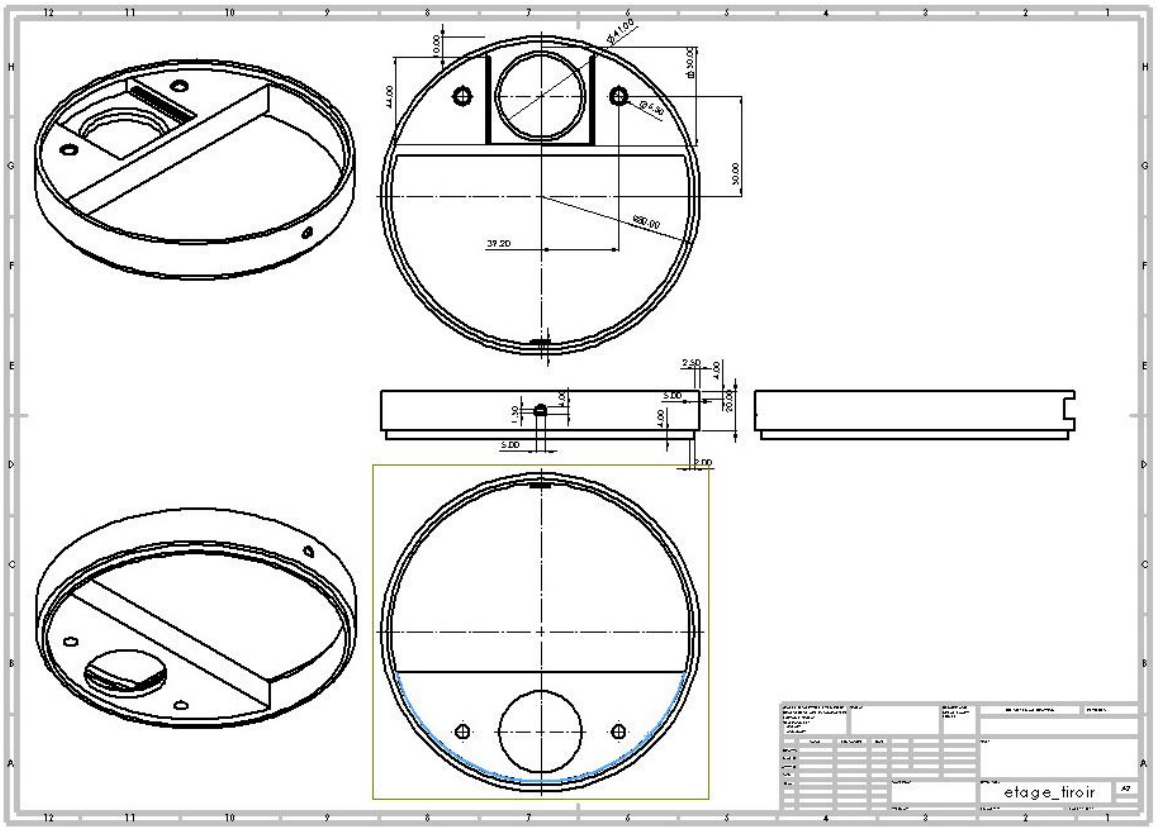
Annexe 3. Pot



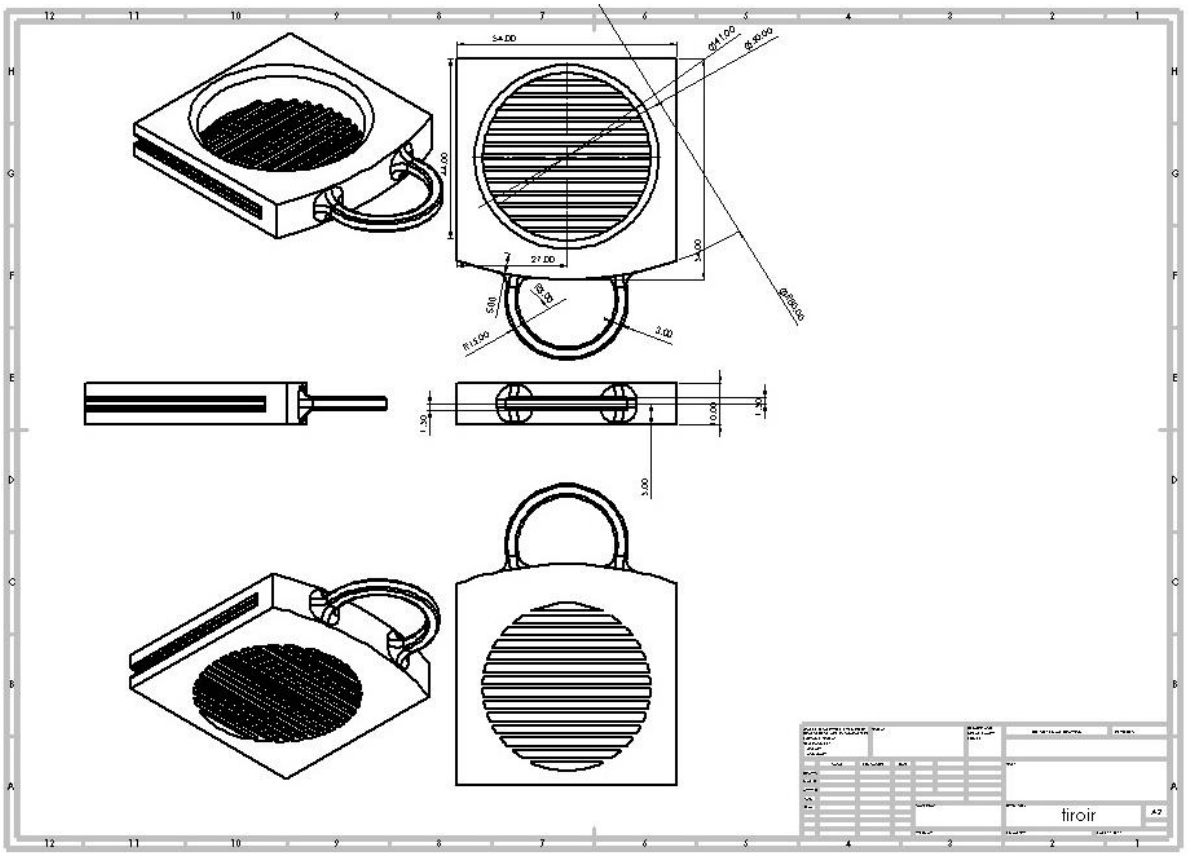
Annexe 4. Réservoir



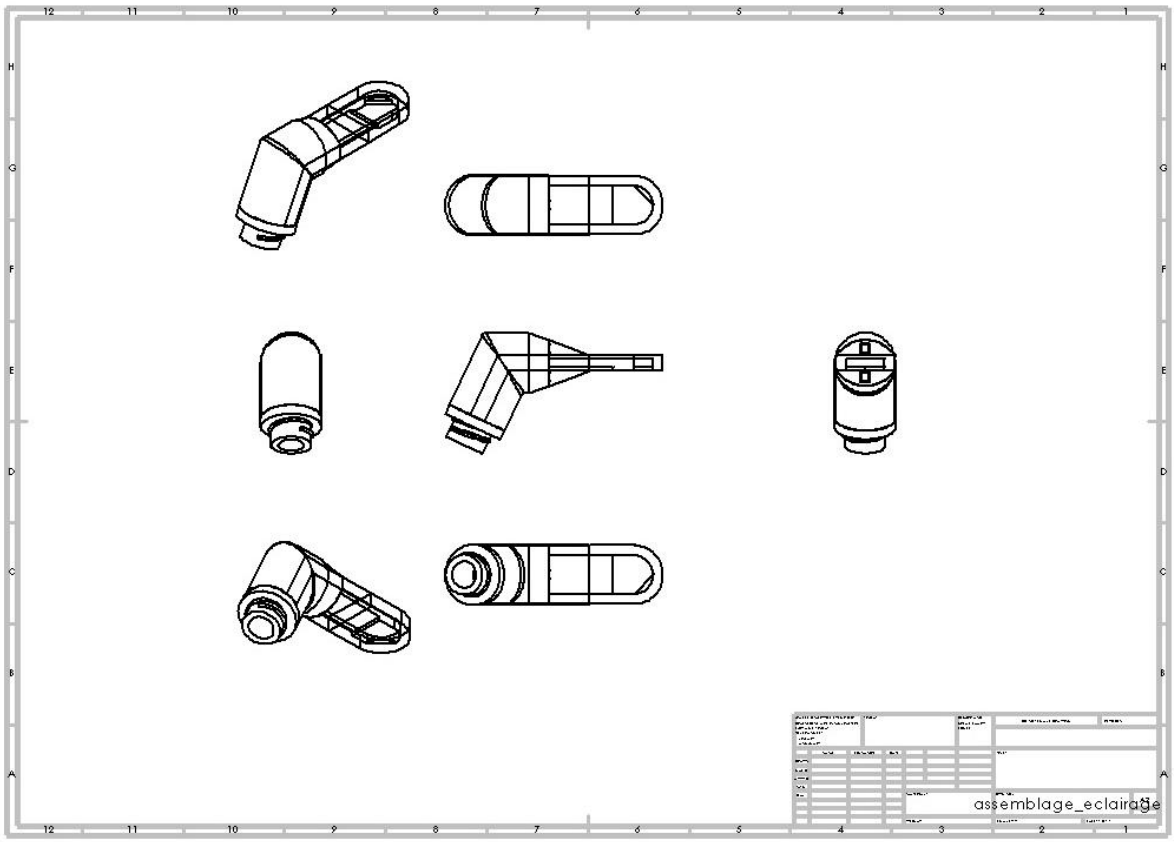
Annexe 5. Stockage électronique



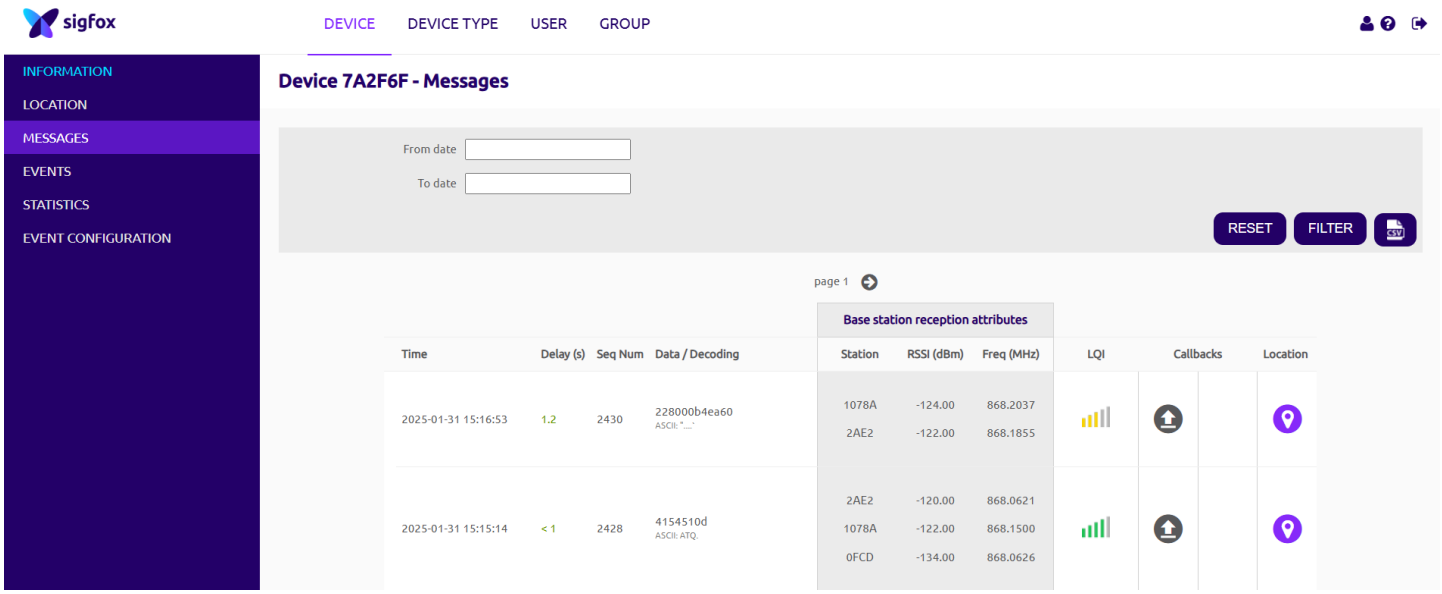
Annexe 6. Etage Filtre



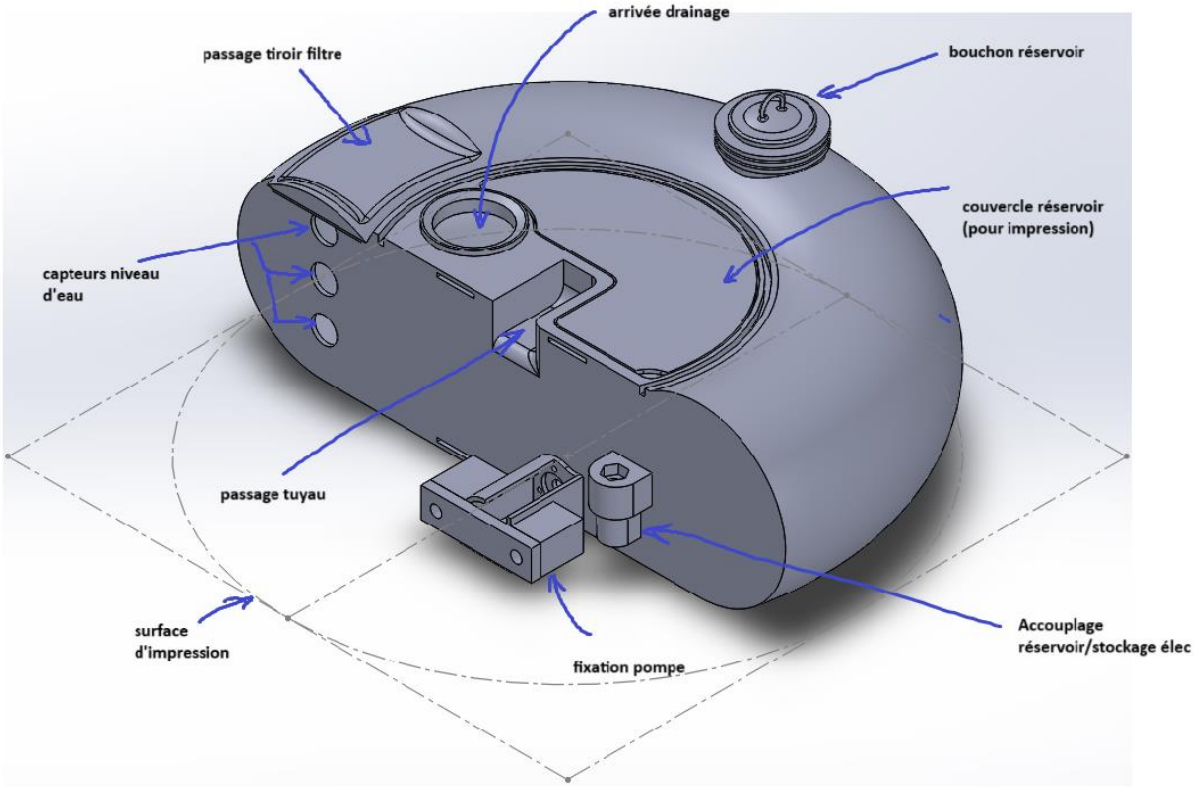
Annexe 7. Tiroir filtre



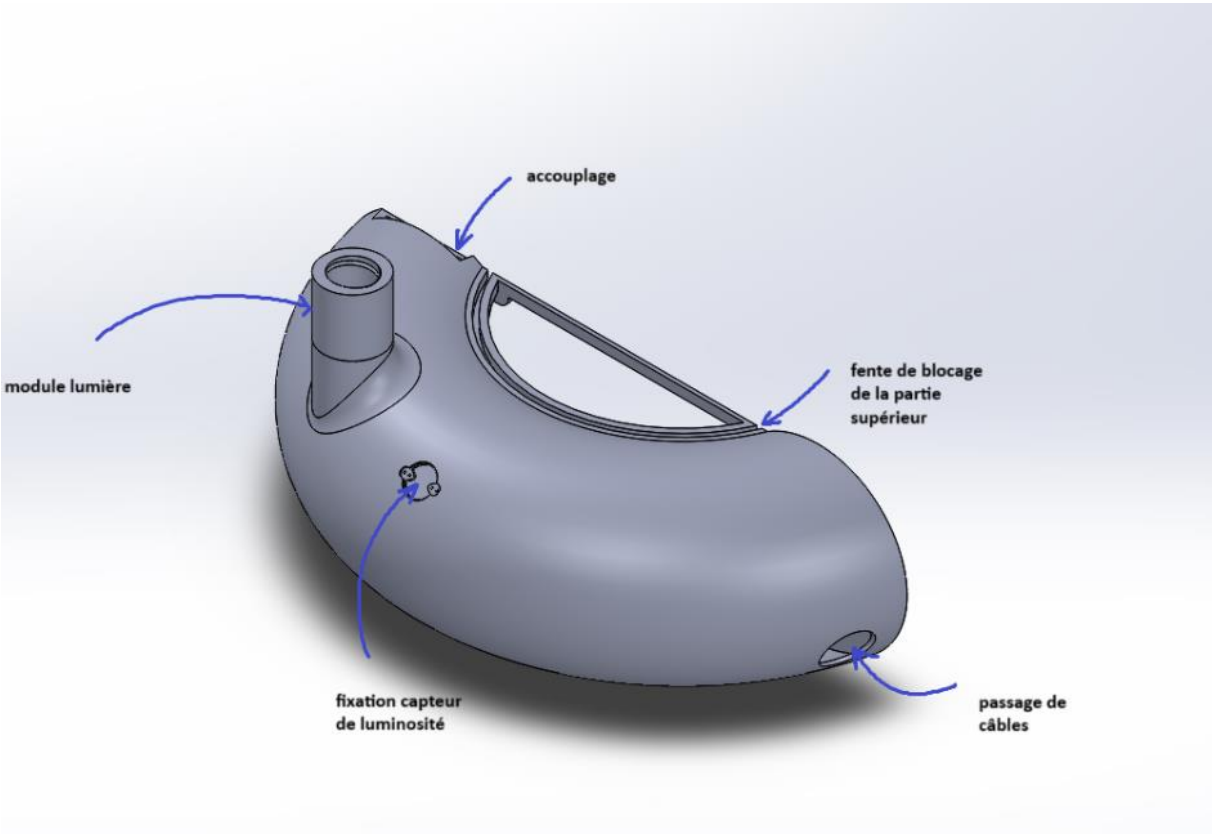
Annexe 8. Assemblage des 2 parties dédiées à l'éclairage



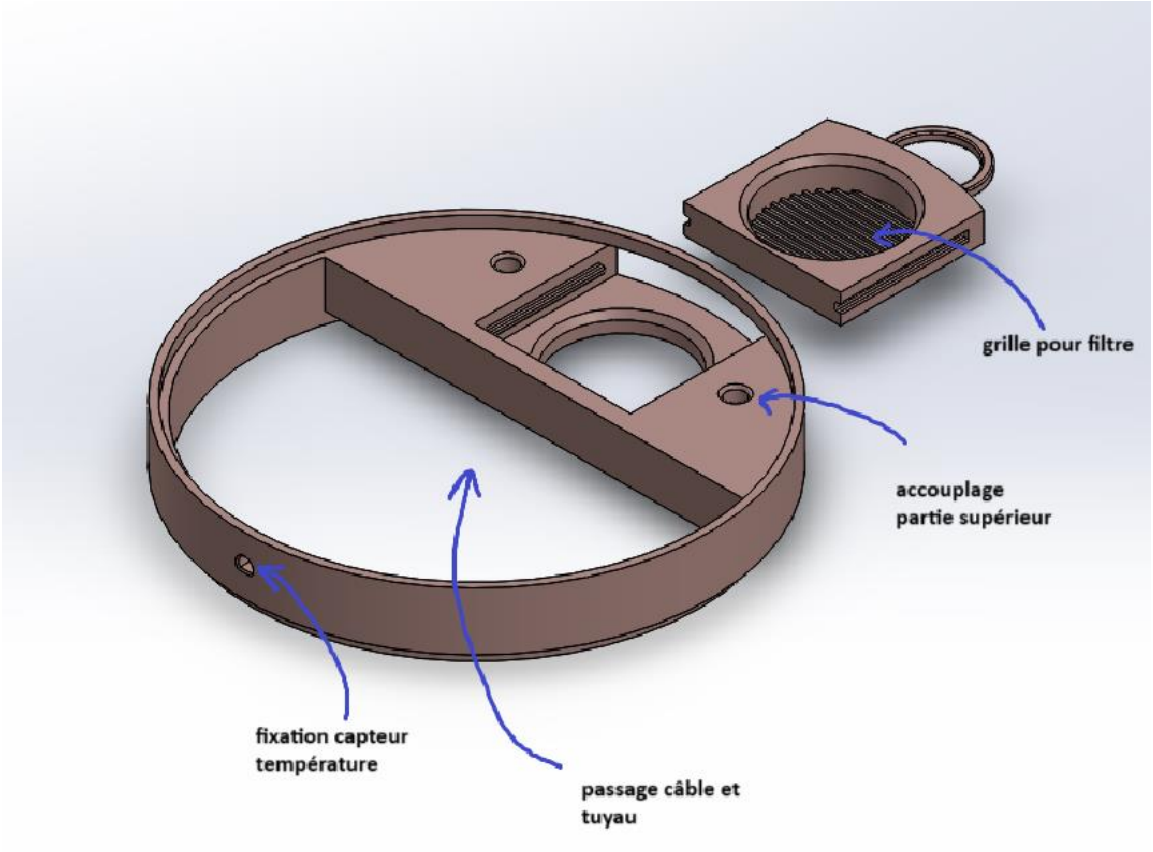
Annexe 9. Affichage des messages reçus par le cloud Sigfox



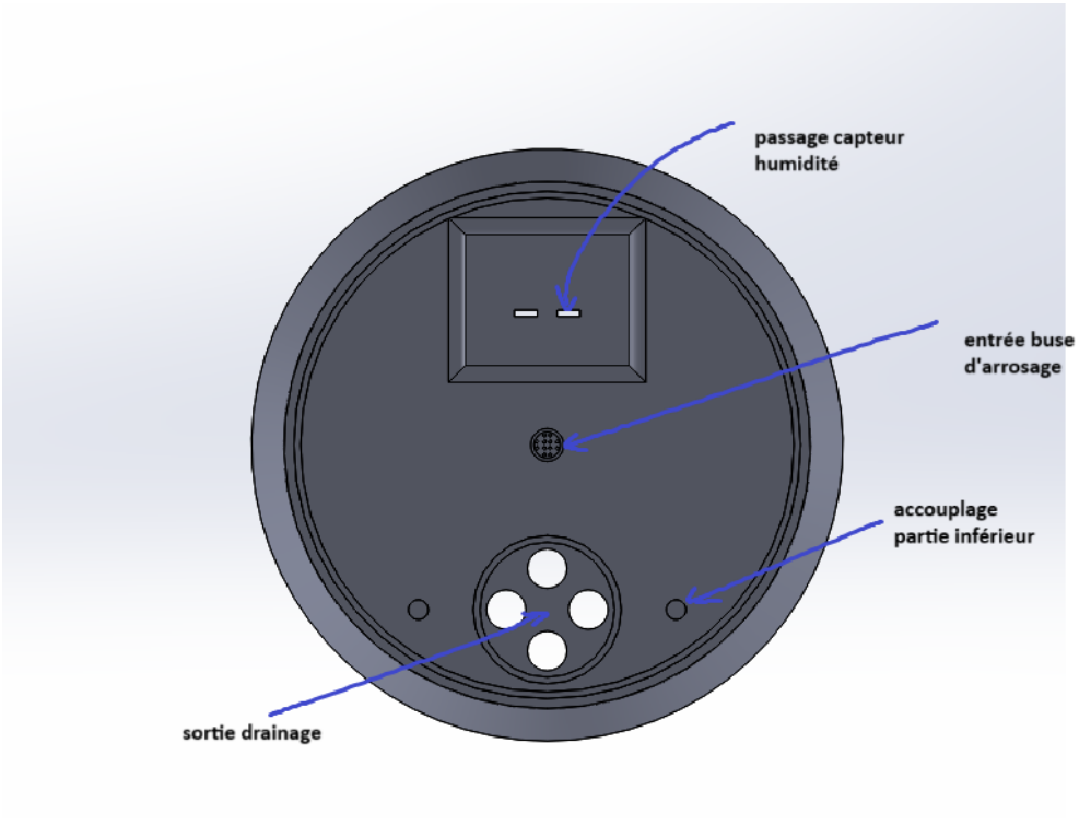
Annexe 10. Explication Réservoir



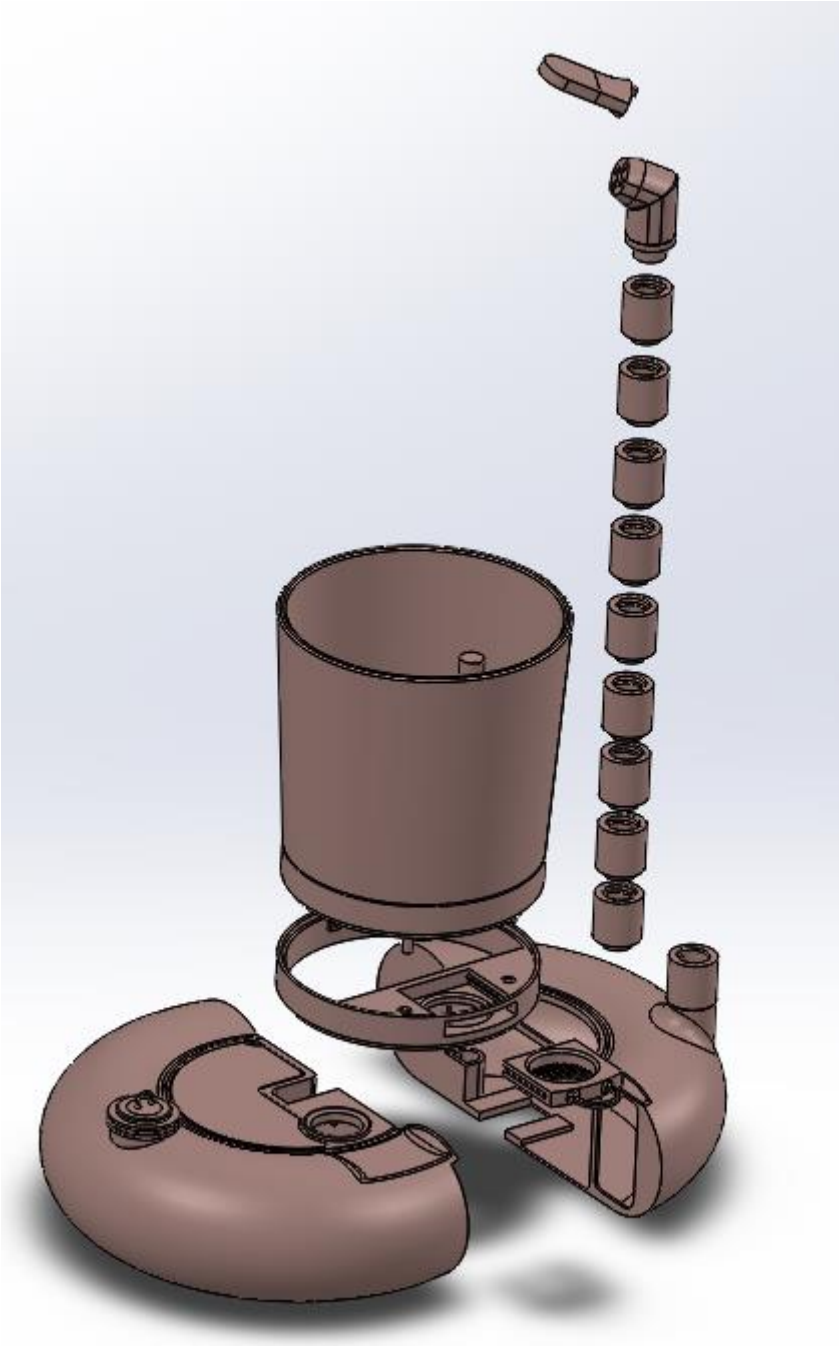
Annexe 11. Explication stockage électronique



Annexe 12. Explication tiroir et étage tiroir du filtre



Annexe 13. Explication vue inférieure du pot



Annexe 14. Vue explosée du pot

Liste des messages SigFox

Liste des messages envoyés de l'Arduino vers le serveur par Sigfox

- Température / Humidité / pH / Nutriments / Luminosité / LVL Batterie — MAX 10 fois/J
- Message d'alerte : Réservoir vide / Température extrême / Conso élec faible — MAX 24 fois/J

Liste des messages envoyés par l'utilisateur vers le serveur puis Arduino

- Fréquence d'arrosage — MAX 10 fois/J
- Demande de remontée d'infos -> pas possible — MAX 20 fois/J
- Gestion luminosité — MAX 5 fois/J
- Initialisation des règles (dépendant des plantes) — MAX 5 fois /J
- Test d'arrosage — MAX 5 fois/J
- Température — MAX 5 fois/J

Liste des messages envoyés par le serveur vers l'Arduino

- Heure — MAX 1 fois/J

Message Sigfox Initialisation :

Type de message : 0 à 7 -> 8 valeurs -> 2^3 = 8 valeurs

Température : 0.0 à 80.0 -> 800 valeurs -> 2^10=1024 valeurs

Humidité nécessaire : 0 à 950 -> 951 valeurs -> 2^10=1024 valeurs

3. Si humidité entre 0 et 300 -> le sol est sec
4. Si humidité entre 300 et 700 -> le sol est humide
5. Si humidité entre 700 et 950 -> le capteur est dans l'eau

Luminosité : 0 à 40 000 -> 40 000 valeurs ->2^16= 65 536 valeurs

Lumière faible – Moyenne – Forte – Intense – Pleine Lumière

soit 0 à 4 -> 5 valeurs -> 2^3 = 8 valeurs

Réservoir vide : 0 à 2 -> 3 valeurs -> 2^2 = 4 valeurs

3 capteurs : LVL Haut – LVL Moyen – LVL Bas

3. Si capteurs LVL Haut = 1 et LVL Moyen = 1 et LVL Bas = 1 -> le réservoir est plein -> « Réservoir vide » est donc = 2
4. Si capteurs LVL Haut = 0 et LVL Moyen = 1 et LVL Bas = 1 -> le réservoir est moitié plein -> « Réservoir vide » est donc = 1
5. Si capteurs LVL Haut = 0 et LVL Moyen = 0 et LVL Bas = 0 -> le réservoir est vide -> « Réservoir vide » est donc = 0

Définition du cycle d'arrosage : 1 à 10 -> 10 valeurs -> 2^4=16 valeurs

Time : 0000 à 2460 -> 2461 valeurs -> 2^12 = 4 096 valeurs

3+10+10+16+2+4= 45 bits = octets

Codage sur 12 octets = 96 bits

Annexe 15. Listes des messages Sigfox

Exemple Encodage :

TypeMessage Temp Humidité Luminosité Réservoir cycle

2 27.3 625 30000 1 2

2 273 625 30 000 1 2

0010 | 01 0001 0001| 10 0111 0001 | 0111 0101 0011 0000 | 01 | 0010

0010 0100 | 0100 0110 | 0111 0001 | 0111 0101 | 0011 0000 | 0100 1000

24 46 71 75 30 48

24 46 71 75 30 48

Serveur vers Arduino :

TypeMessage Time cycle TempMax TempMin Humidité Luminosité Réservoir

Encapsulation par Sigfox et envoie au serveur :

device={device}&time={time}&data={data}&station={station}

data = Message selon Type Message

Type Message :

= 1 -> Remontée d'infos Arduino vers Serveur

TypeMessage Temp Humidité Luminosité Réservoir

3+10+10+16+2= 41 bits = 5,125 octets -> 6 octets

= 2 -> Remontées d'infos avec alertes Arduino vers Serveur

TypeMessage [TempMax TempMin HumiditéMin LuminositéMax LuminositéMin RéservoirMin] TempMax XOR TempMin | HumiditéMin | LuminositéMax XOR LuminositéMin | RéservoirMin

4bits 6bits 10 +10 +16 + 2(38 bits max) = 48bits -> 6 octets

0010 100000 0000101001 0000000000 0000000000000000 00

= 3 -> Demande initialisation Arduino vers Serveur

TypeMessage

4 bits

Électronique et systèmes embarqués :

- Conception d'un système embarqué basé sur Arduino Uno
- Sélection et intégration de capteurs (humidité, température, luminosité, niveau d'eau)
- Utilisation d'actionneurs (mini-pompe, éclairage LED)
- Schéma électronique et câblage des composants
- Gestion de l'alimentation et optimisation énergétique
- Communication avec Sigfox
- Programmation Arduino en C++ pour la récupération et traitement des données

Développement web

- Développement d'un serveur backend en Node.js
- Création d'API pour la communication entre back et front
- Gestion des bases de données avec MongoDB
- Encodage et décodage des messages pour Sigfox
- Développement d'une Progressive Web App (PWA)
- Développement front avec HTML, CSS et JS
- Hébergement du front-end sur GitHub Pages
- Utilisation de ngrok pour exposer une API locale
- UI/UX

Modélisation 3D & prototypage

- Conception et impression 3D FDM du pot et des supports
- Tests d'étanchéité et d'écoulement pour optimiser le design
- Prototypage de buses d'arrosage et tests d'efficacité
- Étude des matériaux pour éviter la toxicité des plastiques en contact avec la plante

Gestion de projet

- Organisation et suivi du projet via Discord
- Analyse de la viabilité commerciale du produit
- Réalisation de tests fonctionnels et analyse des résultats

Et enfin, ce projet nous a permis de faire un premier pas dans le domaine de la botanique.

Annexe 16. Compétences acquises

Résumé :

EDGAR : Le Pot de Fleur Connecté

Dans un monde où la technologie s'invite dans chaque aspect de notre quotidien, pourquoi ne pas l'utiliser pour mieux prendre soin de nos plantes ? C'est le défi relevé par trois étudiants en ingénierie de Polytech Angers à travers EDGAR, un pot de fleur connecté révolutionnaire.

Pensé pour les amateurs de plantes en quête de simplicité, EDGAR surveille en temps réel l'humidité du sol, la luminosité et la température grâce à une carte Arduino et des capteurs intelligents. Via une plateforme web, l'utilisateur peut suivre l'état de sa plante et recevoir des alertes en cas de besoin.

Ce projet mêle modélisation 3D, électronique, programmation embarquée et développement web, tout en explorant les défis du prototypage et des matériaux adaptés à l'environnement végétal. Entre innovation et contraintes techniques, ce rapport dévoile les choix technologiques, les obstacles rencontrés et les perspectives d'amélioration d'un produit alliant high-tech et nature.

Mots Clés : génie informatique, système embarqué, développement web, modélisation 3D, prototypage, programmation objets connectés.

Summary :

EDGAR: The Smart Flowerpot

In a world where technology is becoming part of every aspect of our daily lives, why not use it to take better care of our plants? This is the challenge taken on by three engineering students from Polytech Angers with EDGAR, a revolutionary smart flowerpot.

Designed for plant lovers seeking simplicity, EDGAR monitors soil moisture, light levels, and temperature in real time using an Arduino board and smart sensors. Through a web platform, users can track their plant's status and receive alerts when necessary.

This project combines 3D modeling, electronics, embedded programming, and web development while tackling the challenges of prototyping and selecting materials suited to plant environments. Balancing innovation and technical constraints, this report unveils the technological choices, obstacles encountered, and potential improvements for a product that merges high-tech with nature.