

Bachelor
Systèmes industriels
Option : Infotronics

Travail de bachelor

Diplôme 2023

Pablo Stoeri

*Edge-cloud instrumentation of irrigation wells
in frost control*

Professeur
Prof. Pierre-André Mudry

Expert
Alexandre Gillioz

Date de rendu du rapport
18.08.2023



Filière / Studiengang SYND	Année académique / <i>Studienjahr</i> 2022-23	No TB / Nr. BA IT/2023/90
Mandant / Auftraggeber	Etudiant / Student Pablo Stoeri	Lieu d'exécution / <i>Ausführungsart</i>
<input type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input checked="" type="checkbox"/> Etablissement partenaire CREALP		<input checked="" type="checkbox"/> HES—SO Valais <input type="checkbox"/> Industrie <input type="checkbox"/> Etablissement partenaire Partnerinstitution
Travail confidentiel / <i>vertrauliche Arbeit</i>	Expert / <i>Experte</i> (<i>données complètes</i>) <input type="checkbox"/> oui / ja <input checked="" type="checkbox"/> non / nein Alexandre Gillioz , alexandre.gillioz@altis.swiss Altis, Place de Curala 5, 1934 Le Châble	

Titre / *Titel***Edge-cloud instrumentation of irrigation wells in frost control**Description / *Beschreibung*

Le projet s'inscrit avec dans une collaboration avec le CREALP dans le cadre de la problématique de l'irrigation dans la lutte contre le gel. Durant les périodes de lutte, de nombreux puits d'irrigation sont utilisés en parallèle ce qui a pour effet de mettre fortement à contribution la nappe phréatique. Le canton souhaite, par le biais du CREALP, avoir une estimation des débits soutirés durant ces événements. L'objectif de ce projet est ainsi de proposer un prototype de solution de monitoring connecté à bas coût pour instrumenter les puits afin de détecter la mise en fonction du pompage.

Objectifs :

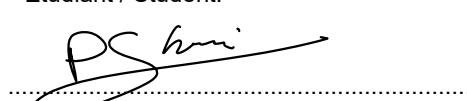
- Choisir et dimensionner un capteur permettant de détecter les enclenchements et déclenchements des cycles d'arrosage des puits agricoles
- Réaliser un prototype de système embarqué low-cost complet permettant de récupérer les données, estimer le débit soutiré par une modélisation des pompes et des conduites et transférer ces données en LoRa
- Tester le système en laboratoire et sur le terrain avec un système à bas prix et résistant aux contraintes de fonctionnement dans des conditions réelles.
- Réaliser un dashboard permettant d'extraire à partir des données mesurées des estimations des débits totaux soutirés à la nappe (dans un environnement contrôlé).

Signature ou visa / *Unterschrift oder Visum*

Responsable de l'orientation /
Leiter der Vertiefungsrichtung:



¹ Etudiant / Student:


Délais / *Termine*

Attribution du thème / *Ausgabe des Auftrags:*
15.05.2023

Présentation intermédiaire / *Zwischenpräsentation:*
19 - 20.06.2023

Remise du rapport final / *Abgabe des Schlussberichts:*
18.08.2023, 12:00

Expositions / *Ausstellungen der Diplomarbeiten:*
25.08.2023 – HEI
28.08.2023 – Monthei
31.08.2023 – Visp

Défense orale / *Mündliche Verfechtung:*
Semaine/Woche 36 (04-07.09.2023)

¹ Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.



Travail de diplôme | édition 2023 |

Filière
Systèmes industriels

Domaine d'application
Infotronics

Professeur responsable
Pierre-André Mudry
pierre-andre.mudry@hevs.ch

Partenaire
CREALP : Centre de Recherche
sur l'Environnement Alpin

Edge-cloud instrumentation of irrigation wells in frost control

Diplômant/e Pablo Stoeri

Objectif du projet

L'objectif de ce travail de bachelor est de concevoir un système capable de quantifier le volume d'eau prélevé à la nappe phréatique via un puits agricole. Par le biais de ce travail, le CREALP vise à mesurer l'impact des puits agricoles sur la nappe phréatique.

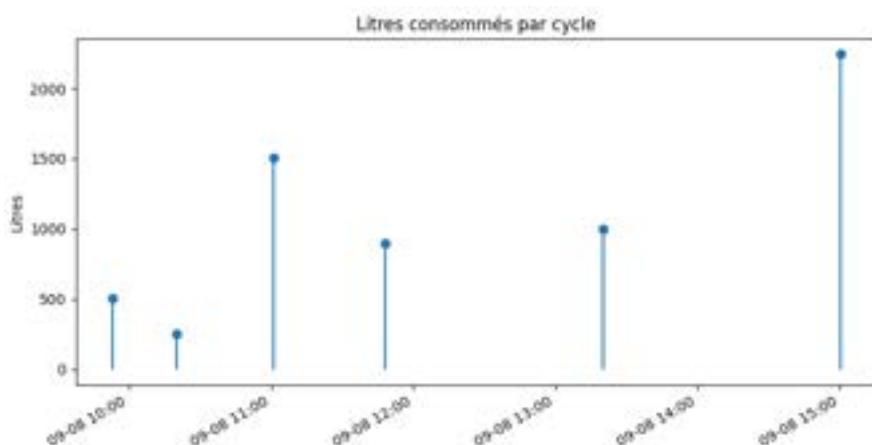
Méthodes | Expériences | Résultats

Pour déterminer le volume d'eau extrait de la nappe phréatique, nous mesurons la durée de passage de l'eau dans la conduite à la sortie du puits. En supposant un débit de pompe constant et préalablement connu, cette durée nous permet de calculer le volume d'eau correspondant. La détection du flux d'eau est réalisée en mesurant les vibrations de la conduite avec un accéléromètre.

Dans ce projet, un système alimenté par batteries, apte à détecter le passage de l'eau et à transmettre la durée de ce passage via le protocole de communication LoRa, a été implémenté.

Par la suite, des scripts ont été implémentés pour stocker les informations reçues via LoRa dans une base de données et afficher un tableau de bord sous forme graphique.

En laboratoire, le système est fonctionnel. La détection, la transmission, la base de données et les graphiques se font de la bonne manière. Cependant, afin qu'il soit utilisable sur le terrain, deux obstacles doivent être résolus : la consommation du système embarqué qui est bien trop élevée, et la transmission LoRa qui ne fonctionne actuellement pas lorsque le système est monté sur une conduite souterraine.



Le graphique ci-dessus est généré par un script python à partir des données enregistrées dans la base de données MySQL.

Information sur ce rapport

Information du contact

Auteur : PabloStoeri
Bachelor Student
HES-SO // Valais Wallis
Switzerland
Email : pablo.stoeri@gmail.com

Déclaration d'honneur

Je soussigné, Pablo Stoeri, déclare par la présente que le travail soumis est le résultat de mon travail personnel. Je certifie n'avoir eu recours ni au plagiat ni à d'autres formes de fraude. Toutes les sources d'information utilisées et les citations d'auteurs ont été clairement mentionnées.

Lieu, date : 1950 Sion, 17 août 2023

Signature : 

Remerciements

Cette partie du rapport est dédiée aux personnes qui m'ont apporté du soutien, de la motivation, du savoir ou des services. Je tiens donc à remercier les personnes suivantes :

- M. Pierre-André Mudry pour la supervision et le soutien durant l'intégralité du projet.
- M. Pascal Ornstein du CREALP, sans qui je n'aurais pas eu l'occasion de réaliser ce projet.
- M. Vlad Hasmatuchi, M. Charly-Loup Lecointre et M. Maxime Chiarelli pour l'accès au laboratoire d'hydraulique et pour les idées intéressantes concernant le design mécanique du système.
- M. Jérôme Darbelley pour les informations sur l'impression 3D et la petite formation qui va avec.
- M. Steeve Maillard et M. Laurent Maret pour le contact avec le service de l'agriculture du Valais.
- M. Jan Béguin, expert à l'OFAG, pour les informations à l'échelle du pays concernant ce genre de projet.
- M. Steve Gallay et tout le labo d'électronique pour les conseils et le routage du PCB.
- M. Romain Masserey pour la fabrication de mes pièces mécaniques et pour son savoir-faire dans le domaine.
- M. Laurent Clerc, chef des cultures de Châteauneuf, pour le temps accordé et les informations pratiques sur les puits agricoles.
- Mme. Cornelia Stoeri pour les relectures du rapport.
- Raphaël et Léa pour l'hébergement à Sion lors des périodes de grande fatigue.
- M. Loris Zufferey, pour son avis tout au long du projet, les discussions et les rires dans toutes les situations.
- Toute la classe I6 avec laquelle on ne s'ennuie jamais.

Liens avec la durabilité

En 2015, les États membres de l'ONU ont été appelés à adopter le programme "Agenda 2030" [1]. Ce dernier est une liste de 17 objectifs prenant en compte équitablement différents aspects :

- La dimension économique
- La dimension sociale
- La dimension environnementale du développement durable
- L'éradication de la pauvreté
- L'intégration du développement durable

Ces objectifs englobent un total de 169 cibles qui, d'ici 2030, devront être atteintes par tous les États membres de l'ONU. Bien que la Suisse ne soit pas membre de l'ONU, elle a aussi été appelée à réaliser ces objectifs [2].

Ces objectifs concernent tout le monde, et il est crucial de les intégrer dans chaque projet réalisé, quel que soit le domaine. La politique, l'ingénierie, l'agriculture ou encore l'éducation sont tous des secteurs ayant une influence sur le développement durable.

La durabilité est au cœur de ce projet. Si les ressources en eau étaient inépuisables, ce travail n'aurait pas lieu d'être. Comme l'indique clairement le 6^{ème} objectif de l'Agenda 2030, notamment le 6.5 [3], il est essentiel, d'ici à 2030, de "mettre en œuvre une gestion intégrée des ressources en eau à tous les niveaux". Ainsi, ce projet contribue à la réalisation de cet objectif n°6, car **le but est de mesurer combien d'eau est utilisée** pour, éventuellement, améliorer la gestion des puits agricoles.

Ce projet est également en lien avec le 12^{ème} objectif. La cible 12.2 met en lumière la nécessité de "parvenir à une gestion durable et à une utilisation rationnelle des ressources naturelles" [4]. Les mesures prises par le système développé pourraient révéler la quantité d'eau prélevée de la nappe phréatique. Ces données, comparées aux consommations théoriques souhaitées, pourraient permettre de quantifier le gaspillage d'eau dû à l'usage des puits agricoles.

De plus, l'un des thèmes abordés dans l'objectif n°2 est de promouvoir une agriculture durable [5]. Si l'on parvient à réduire la quantité d'eau gaspillée grâce aux données du système, nous ferons un pas de plus vers une agriculture plus durable.

En conclusion, comme dans tout projet d'ingénierie, des éléments non écologiques restent malheureusement inévitables. Nous ne connaissons pas, par exemple, tous les procédés de fabrication des divers composants. Que ce soit l'électronique, les matériaux utilisés pour étanchéifier le système, ou encore la production et la destruction des piles ou batteries utilisées, ces éléments ont un impact écologique difficilement quantifiable. Ainsi, ces éléments n'ont pas été pris en considération dans ce projet de bachelor.

Table des matières

Remerciements	vii
Liens avec la durabilité	ix
Table des matières	x
Acronymes	xiii
Glossaire	xv
1 Introduction	1
1.1 Contexte / Problématique	1
1.2 Objectifs	3
1.3 Structure de ce rapport	3
2 Analyse	5
2.1 Débit et pompage	6
2.2 Réalisations techniques antérieures	7
2.3 Faisabilité du projet	8
2.4 Conclusion	8
3 Design	9
3.1 Étapes clés	10
3.2 Planning	11
3.3 Conclusion	12
4 Implementation	13
4.1 Prototype n°1	14
4.2 Prototype pré-industriel	28
4.3 Dashboard	34
4.4 Conclusion	36
5 Tests et validation	39
5.1 Tests en laboratoire	40
5.2 Tests de consommation	46
5.3 Tests de portée	51
5.4 Tests sur le terrain	52
5.5 Résultats	56
5.6 Discussion	57

Table des matières

6 Conclusions	59
6.1 Résumé du projet	59
6.2 Comparaison avec les objectifs initiaux	60
6.3 Difficultés globales rencontrées	61
6.4 Améliorations et tâches futures	62
6.5 Perspectives futures	63
6.6 Conclusion personnelle	64
Bibliographie	65
A Mind Map	69
B Liste des coûts	71
C Code : UART vers graphiques	73
D Code : UART vers MySQL	77
E Code : TTN vers MySQL	79
F Code : MySQL vers graphique n°1	83
G Code : MySQL vers graphique n°2	87
H Graphiques de tests du prototype n°1	91
I Électronique	97
J Mécanique : Prototype n°1	101
K Mécanique : Prototype pré-industriel	103
Liste des figures	109
Liste des tableaux	110

Acronymes

- μC** Micro-contrôleur. [14](#), [15](#), [19](#), [20](#), [28](#), [29](#), [49](#)
- ADC** Analog to Digital Converter. [14](#), [15](#)
- ASA** Acrylonitrile Styrene Acrylate. [33](#)
- CREALP** Centre de Recherche sur l'Environnement Alpin. [1](#), [3](#), [7](#), [8](#), [15](#), [28](#), [63](#)
- FIFO** First In First Out. [17](#), [18](#), [20–22](#), [26](#)
- HAL** Hardware Abstraction Layer. [24](#), [25](#)
- I/O** Input / Output. [14](#), [28](#)
- I2C** Inter-Integrated Circuit. [14](#), [15](#), [18](#), [19](#), [23](#), [25](#)
- IOT** Internet Of Things. [51](#)
- LED** Light-Emitting Diode. [28](#), [29](#)
- Li-ion** Lithium-ion. [29](#), [30](#)
- LoRa** Long Range. [3](#), [9](#), [10](#), [12](#), [14](#), [16](#), [17](#), [22–26](#), [36](#), [39](#), [41](#), [44–46](#), [49–53](#), [59–61](#), [64](#)
- LSB** Least Significant Bit. [21](#), [22](#)
- MQTT** Message Queuing Telemetry Transport. [34](#)
- MSB** Most Significant Bit. [21](#), [22](#)
- OFAG** Office Fédéral de l'Agriculture. [1](#)
- PCB** Printed Circuit Board. [23](#), [28–30](#), [50](#), [51](#), [62](#)
- PLA** Polylactic Acid. [27](#)
- SDK** Software Development Kit. [24](#), [25](#)
- SLS** Selective Laser Sinterin. [33](#)
- SPI** Serial Peripheral Interface. [14](#), [15](#), [18](#)
- TTN** The Things Network. [23](#), [24](#), [34](#), [36](#), [51](#), [52](#), [59](#)
- UART** Universal Asynchronous Receiver Transmitter. [52](#), [53](#)
- UV** Ultraviolet. [27](#), [31](#), [33](#), [36](#)

Glossaire

Cube MX Cube MX est l'interface graphique de STM32 Cube IDE. Elle permet de configurer les clocks, les protocoles utilisés, les timers, et de nombreux autres éléments. Une fois la configuration terminée, Cube MX peut générer le code correspondant à ces paramètres et évite à l'utilisateur de tout devoir configurer à la main. [18](#), [19](#), [25](#)

Théorème d'échantillonnage Dans le cas général, le théorème d'échantillonnage [\[6\]](#) affirme que "l'échantillonnage d'un signal exige un nombre d'échantillons par unité de temps supérieur au double de l'écart entre les fréquences minimale et maximale qu'il contient". [15](#)

1 | Introduction

1.1 Contexte / Problématique

En Valais, il y a plus de 1600 puits agricoles répertoriés. La problématique actuelle est qu'il est impossible de savoir, ne serait-ce que grossièrement, combien d'eau est soutirée à la nappe phréatique par ces nombreux puits.

Dans ce contexte, ce travail traite la question suivante : **comment serait-il possible de quantifier le volume d'eau passant à travers les conduites de ces puits ?**

Ce projet a été proposé par le [CREALP](#), centre de recherche sur l'environnement alpin. L'une des préoccupations de cette fondation est la gestion de l'eau en Valais. Cette fondation calcule d'une part la quantité d'eau qui entre dans le territoire, par exemple par la pluie ou encore par la fonte des glaciers. D'autre part, elle calcule la quantité d'eau qui en sort, entre autres par le Rhône ou par l'utilisation d'eau dans l'agriculture. Ces calculs sont essentiels pour gérer cette ressource et pouvoir prévenir les concernés en cas de manque ou de danger, par exemple le manque d'eau dans les nappes phréatiques ou les crues en cas de fortes pluies.

Aujourd'hui, dans le milieu agricole, il est essentiel de comprendre qu'une partie de l'eau, utilisée pour l'arrosage ou la lutte contre le gel, provient des nappes phréatiques. Selon M. Jan Béguin, expert à l'[OFAG](#), un tiers de l'eau utilisée dans l'agriculture en suisse provient des nappes.

L'eau est soutirée aux nappes par le biais de puits qui sont utilisés par les agriculteurs. C'est-à-dire que ceux-ci couplent des pompes thermiques ou électriques aux puits, comme montré sur la figure 1.1, et peuvent à ce moment utiliser l'eau à la sortie de leur pompe selon leurs besoins.

Chapitre 1. Introduction



(a) Pompes thermiques sur tracteur



(b) Pompe thermique mobile

Figure 1.1 Puits agricoles en fonctions avec deux type de pompes différentes [7].

Pour des raisons de coûts, ces puits ne sont actuellement pas équipés de matériel de mesures permettant de quantifier l'eau qui est pompée hors des nappes phréatiques, et c'est pour cette raison que ce projet voit aujourd'hui le jour : **on aimerait connaître cette quantité d'eau soutirée aux nappes phréatiques par le milieu agricole.**

Sans contrainte particulière, cette problématique serait rapidement résolue en installant des débitmètres. Mais les contraintes qui empêchent la mise en pratique de cette solution sont bel et bien présentes. La principale contrainte est qu'il est impossible ou compliqué d'intervenir à l'intérieur du circuit hydraulique, c'est à dire qu'on ne peut pas placer de débitmètre à l'intérieur de ces conduites. Ceci est dû au fait que l'installation d'un tel appareil à l'intérieur d'une conduite ou entre deux conduites demanderait plusieurs heures de main d'oeuvre et des modifications des conduites qui dérangerait les agriculteurs. C'est aussi dû au fait que l'eau qui passe à travers ces conduites, directement extraite de la nappe phréatique, est remplie de résidus ce qui a pour conséquence d'endommager tous les éléments (comme des capteurs) sur son passage, surtout lorsque la pression est élevée. La deuxième contrainte importante est que ces débitmètres sont coûteux, d'autant plus s'ils sont connectés via un réseau quelconque.

Les contraintes du projet qui peuvent déjà être fixées à ce stade pour le système à développer sont donc les suivantes :

- Il doit être non-intrusif.
- Il doit être facile à installer
- Il doit être bon marché.
- Il doit permettre la télémesure

1.2 Objectifs

Selon le [CREALP](#), lorsque la pompe est allumée, le débit est maximal et il reste stable. Ceci veut dire que, lorsque de l'eau traverse la conduite, le débit est toujours le même. Nous utiliserons cette information comme hypothèse de travail.

Par conséquent, le premier objectif de ce projet est de réaliser un système permettant de déterminer **quand est-ce que de l'eau traverse les conduites, pendant combien de temps t, et d'en calculer le volume d'eau utilisé**. Ce calcul se réalise en intégrant le débit constant par rapport au temps t. Cependant, cela nécessite de mesurer de manière expérimentale, avant l'installation du système, le débit lorsque la pompe est allumée. Une manière expérimentale de mesurer le débit est expliquée dans le chapitre suivant. L'envoie des données via [LoRa](#) est également inclus dans cet objectif initial.

De même il est important pour le [CREALP](#) que le système qui va permettre de faire ces mesures soit résistant aux contraintes du terrain auxquelles il sera confronté et, comme dit précédemment, bon marché.

Le second objectif est d'implémenter un premier dashboard permettant de visualiser le volume d'eau soutiré à la nappe phréatique pour les différents puits. Cela permet de visualiser de manière localisée, l'impact sur la nappe.

Pour conclure, le troisième objectif de ce projet est d'étendre cette approche à plusieurs puits situés dans différentes régions. C'est à dire d'installer plusieurs systèmes, et de réaliser, par exemple sous forme de carte, un deuxième dashboard. Ce dashboard a pour but de montrer, de manière globale, l'utilisation des puits agricoles dans la plaine de Rhône. Ceci permettrait au [CREALP](#), mandant de ce projet, de visualiser ce qu'il se passe au niveau de l'utilisation de l'eau dans la nappe, plus spécifiquement durant les périodes de grande utilisation comme lors des arrosages intensifs (chaleur et sécheresse) ou lors des périodes de lutte contre le gel.

1.3 Structure de ce rapport

Ce rapport suit une organisation similaire à celle du projet dans son ensemble : premièrement la partie analyse présente de la faisabilité du projet et reprend certains principes de base. Ensuite, la partie centrale concerne le design, l'implémentation et les tests effectués avec leur analyse. Et pour finir, la conclusion dans laquelle se trouve un bilan du projet avec les résultats obtenus et les améliorations nécessaires et possibles.

2 | Analyse

Avant de se lancer pleinement dans du travail technique précédé par des choix importants, les questions suivantes ont été posées :

- Comment peut-on calculer le débit ?
- Qu'est-ce qui a déjà été réalisé ?
- Est-ce que le travail demandé est faisable ?

Sommaire

2.1	Débit et pompage	6
2.2	Réalisations techniques antérieures	7
2.3	Faisabilité du projet	8
2.4	Conclusion	8

2.1 Débit et pompage

Pour commencer l'analyse de ce travail de diplôme, il est fondamental de comprendre ce qu'est le débit. Dans notre cas nous parlons du débit volumique, c'est-à-dire le volume qui traverse une surface par unité de temps [8]. La formule ci-dessous, où D_V est le débit en m^3/s , v la vitesse du flux en m/s et S la surface traversée en m^2 , explique mathématiquement la notion de débit.

$$D_V = \iint_S \vec{v} \bullet \vec{d}s$$

Une manière simple de mesurer un débit est de faire couler de l'eau dans une bassine de volume connu, et de mesurer le temps nécessaire afin de la remplir. Le volume de la bassine divisé par le temps de remplissage donne comme résultat le débit.

Dans le cadre de ce projet, c'est une technique de ce type qui va être utilisée pour calibrer le système implémenté.

Le mandant de ce projet, le CREALP, a déterminé qu'une mesure tout ou rien était suffisante étant donné que le débit est estimé constant une fois que la pompe du système est enclenchée.

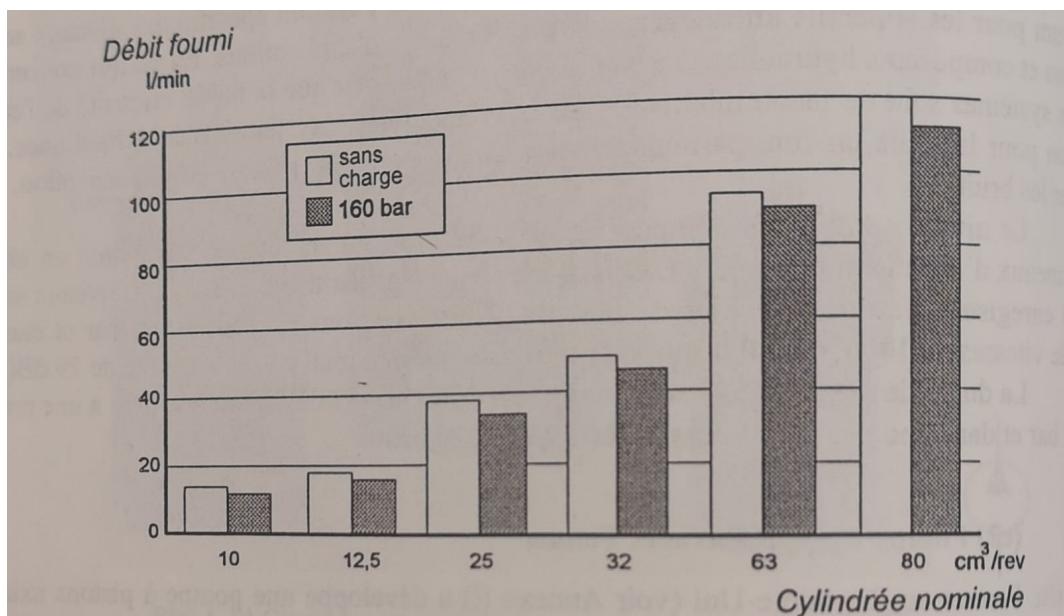


Figure 2.1 Débit d'une pompe thermique en fonction de la pression, graphique tirée de la page 78 du livre de Erik Trostmann [9].

La figure 2.1 ci-dessus montre que sur des pompes thermiques, qui sont souvent utilisées pour extraire l'eau des nappes phréatiques à travers les puits agricoles, le débit n'est influencé que d'environ 10% par la pression, donc par la charge. Cela veut dire, dans le cadre de ce projet, que les systèmes d'arrosage raccordés en amont de la pompe ont au maximum 10% d'influence sur le débit à vide.

2.2 Réalisations techniques antérieures

En revanche, les pompes électriques, elles, posent plus de soucis pour ce projet. En effet, ces pompes sont réglables et paramétrable par les agriculteurs et peuvent donc avoir un comportement variable (si les paramètre changent, le débit peut changer aussi). Ce qui est problématique dans ce projet où l'on veut partir du principe que le débit est constant. L'avantage est que ces pompes sont plus rarement utilisées, du fait que ce sont souvent des installations dans les champs où l'électricité n'est pas forcément raccordée. Lors des différentes journées de test sur le terrain, cet élément est à clarifier.

2.2 Réalisations techniques antérieures

Réalisé par le CREALP

Le [CREALP](#) a déjà mis en place un premier projet afin de trouver une solution à la problématique traitée dans ce travail. Le travail réalisé, non-concluant dans sa globalité, a quand même permis de définir la faisabilité de ce projet de bachelor. Sans entrer dans les détails de la figure 2.2, on peut en conclure qu'un accéléromètre suffit à détecter les cycles d'arrosage et que l'on pourrait, à partir de ces mesures et des paramètres physiques des puits, définir combien d'eau a été soutirée à la nappe phréatique.

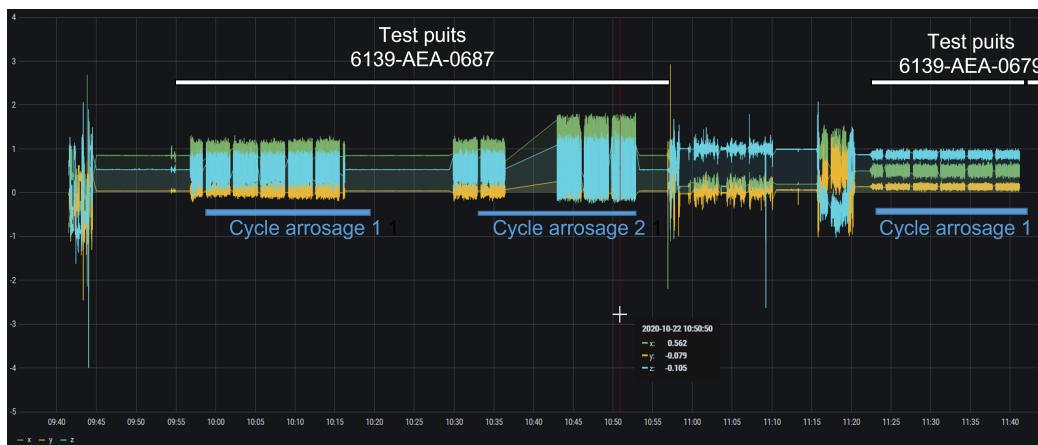


Figure 2.2 Tests effectués par le CREALP avec un accéléromètre [10].

Le prototype qui a été réalisé pour faire ces tests est une base technique intéressante. Du fait que les tests de la figure 2.2 montrent des variations d'accélération utilisables pour ce travail, l'accéléromètre utilisé est donc une première référence matérielle qui pourrait permettre au système réalisé durant ce projet de fonctionner. Les caractéristiques de ce capteur d'accélération vont donc être utilisées pour les choix futurs.

Réalisé par M. Solioz

Sous la direction du professeur Mudry, M. Solioz a implémenté un système pour son travail de master qui avait aussi pour but de calculer le débit d'eau traversant un tube ou une conduite [11]. Différentes méthodes ont été prises en compte, mais il a gardé une méthode qui combine des accélérateurs avec des capteurs piézoélectriques. Son travail apporte de la confiance en la réussite de ce projet car, avec un système non-intrusif, il est parvenu à une précision du calcul de débit qui se rapproche des valeurs réelles (environ 7% d'erreur). Cette précision a été atteinte en utilisant du Machine Learning.

Chapitre 2. Analyse

Étant donné que le [CREALP](#) part du principe que ces puits agricoles sont utilisés en mode "tout ou rien", la précision atteinte par M. Solioz n'est, dans le cadre de ce projet, pas nécessaire. Dans notre cas, lorsqu'on allume la pompe, le débit est au maximum, et lorsqu'on l'éteint, le débit est nul.

Réalisé par M. Maceiras Ferreiro

Un autre système qui va dans la même direction que celui implémenté durant ce projet est le travail de bachelor de M. Maceiras Ferreiro [12]. Lui aussi a réalisé un système permettant de faire une mesure de débit sur des tuyaux. Son travail utilise également un accéléromètre dont les valeurs sont traitées par du Machine Learning. Le projet n'a malheureusement pas donné les résultats escomptés sur le terrain. Les tests en laboratoire montrent cependant que la mesure de débit avec un accéléromètre est une piste qui mérite d'être explorée.

2.3 Faisabilité du projet

Les 3 projets mentionnés ci-dessus montrent deux éléments :

- Les objectifs de ce travail de bachelor sont réalisables. M. Solioz et M. Maceiras Ferreiro ont tous les deux travaillé sur une mesure la plus exacte possible, alors que dans ce projet, très différent dans son utilisation, l'objectif est de capter si de l'eau passe à travers la conduite.
- Des imprévus concernant la mise en pratique de ce travail sont à prévoir. Bien que M. Solioz a fourni un rendu avec des objectifs atteints. Cependant, les trois travaux mentionnés montrent que, sur ce genre de mesures, des détails comme le placement de l'accéléromètre peuvent entraver la réalisation des objectifs initialement fixés.

2.4 Conclusion

Le bilan de cette analyse est positif. Des projets antérieurs ont été réalisés dans des domaines semblables et montrent la faisabilité de ce travail de bachelor. Les éléments clés de cette analyse sont :

- L'utilisation d'un accéléromètre est une première solution à cette mesure de débit non-intrusive. Ce projet va donc être construit sur cet élément, mais il est important de se rappeler que d'autres moyens de mesure sont possibles, par exemple un microphone ou un capteur piézoélectrique. Il est donc judicieux de ne pas fermer la porte à ces autres solutions, en ayant un software et un hardware adaptable à d'autre capteurs.
- Les tests en laboratoire et sur le terrain sont très importants pour ce projet. Ils révéleront sûrement des détails importants et qui devront être pris en compte pour la suite.
- Il est important, afin de pouvoir atteindre les objectifs, de se rappeler tout au long du projet, que le but du système est de capter si de l'eau traverse la conduite, mais aussi de pouvoir quantifier ce volume d'eau.

3 | Design

Suite à l'analyse, la structure globale du projet est la suivante :

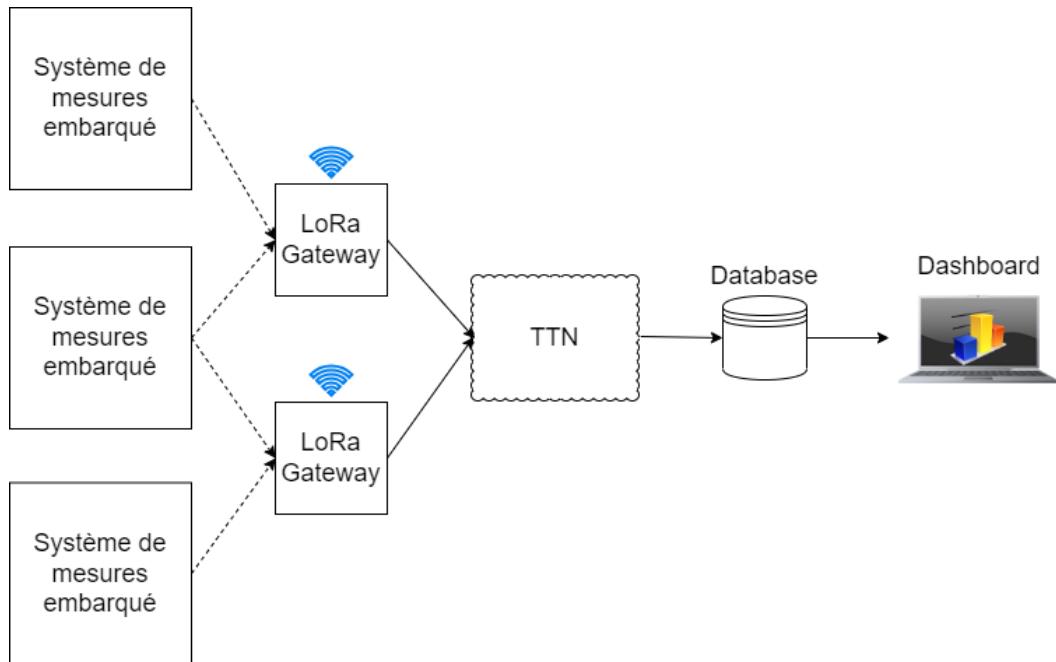
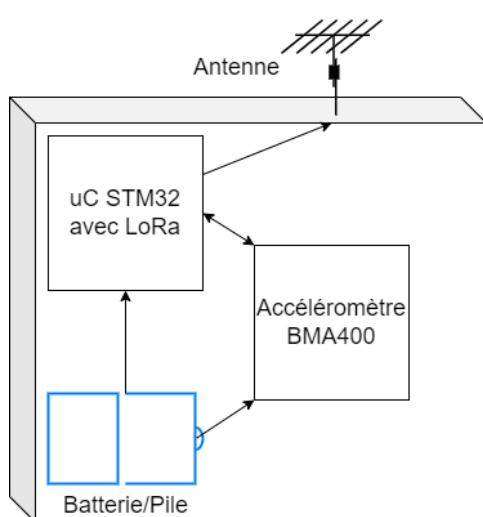


Figure 3.1 Structure globale du projet

Sur la figure 3.1 ci-dessus, l'unique partie qui est déjà implémentée en majorité est le lien entre les Gateways et TTN. Ce lien se fait en paramétrant le Gateway et l'application sur la console TTN [13].



Suite à l'analyse, le choix a été pris d'utiliser un accéléromètre, tout en prévoyant un changement de capteur par la suite, si nécessaire. Le système de mesure embarqué contient un micro-contrôleur afin de recevoir les données de l'accéléromètre et de les traiter. De même, il contient la partie transmission vers le réseau LoRa depuis lequel les données seront mises en forme (dashboard). La figure 3.2 représente ce qui est à implémenter à l'intérieur du système.

Figure 3.2 Structure globale du système

3.1 Étapes clés

Afin de définir les étapes clés du projet, il est intéressant de répondre à la question suivante : Qu'est-il nécessaire de faire afin d'arriver, à la fin du projet, à un prototype pré-industriel ? C'est-à-dire un prototype qui pourrait dans le futur être fabriqué en plus grande quantité sans changement majeur.

À première vue, il y a beaucoup d'inconnues avant d'arriver à ce genre de prototype. Les questions suivantes se posent :

- Quel type de design mécanique pourrait être efficace pour la mesure ?
- Quel matériau choisir pour le boîtier qui englobera le système et pour la partie qui sera en contact avec la conduite ?
- Quel composant faut-il choisir, que ce soit pour la partie LoRa la partie micro-contrôleur, ou la partie accéléromètre ?

Afin de pouvoir séparer les inconnues et d'avoir une liberté sur la suite du projet, la décision a été prise de procéder en deux étapes :

- **Étape clé n°1** : Réaliser un premier prototype à l'aide de kits de développements. Ces kits me fournissent une base électronique dont le fonctionnement est presque garanti, ce qui me permet de développer les différents éléments en sachant que les obstacles ne proviennent pas de l'électronique.
- **Étape clé n°2** : Réaliser le prototype pré-industriel sur lequel la partie électronique est refaite et contient uniquement les éléments nécessaires à ce projet. Théoriquement, ce qui a été programmé pour le premier prototype ne nécessite que très peu d'adaptation pour fonctionner sur le deuxième.

Cette manière de faire a aussi l'avantage de pouvoir faire des tests entre la première et la deuxième étape et de faire les adaptations nécessaires. Par exemple, avant ces tests intermédiaires, il est très difficile de savoir si l'accéléromètre va devoir être placé à l'intérieur du boîtier ou s'il est nécessaire de le placer directement sur la conduite du puits.

Étape clé n°3 : les tests. Ils permettent de voir si le système est utilisable en laboratoire comme en conditions réelles. Il sera important de préparer ces tests et de savoir avec précision ce qui est attendu, et de comparer avec ce qui se passe réellement.

L'**étape clé n°4** de ce projet est la partie dashboard. Elle apporte une plus-value du fait qu'elle est visuelle. Le but du dashboard est de montrer la réponse à la problématique initiale sous forme de graphiques ou de cartes. Ce qui veut dire que, si les valeurs récoltées par le système permettent de faire des graphiques sur lesquels on peut voir quel volume d'eau a été soutiré à la nappe phréatique, durant quelle période, et à quel endroit, le système pourra être décrit comme fonctionnel. L'utilisation d'une base de données dans laquelle on vient chercher les informations sera probablement nécessaire à cette étape.

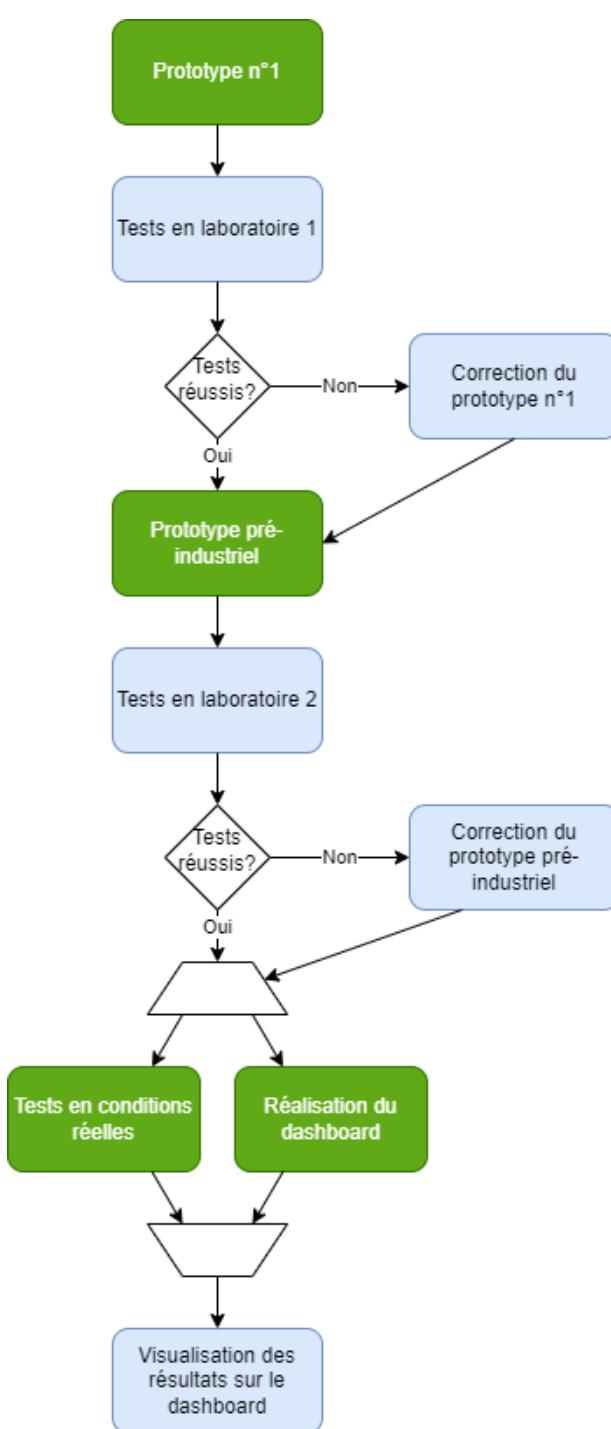
3.2 Planning

Maintenant que les étapes clés ont été déterminées, et après avoir fait un mindmap (Annexe A) avec toutes les tâches qui ont été pensées dans un premier temps, un planning a pu être établi. En commençant par placer en fin de projet les dates durant lesquelles il serait bien de tester le ou les systèmes, il a ensuite été possible d'établir un rétro-planning, pour en arriver au début du projet, c'est-à-dire le 15 mai 2023.

Du	15.mai	22.mai	29.mai	05.juin	12.juin	19.juin	26.juin	03.juil	10.juil	17.juil	24.juil	31.juil	07.août	14.août
Au	19.mai	26.mai	02.juin	09.juin	16.juin	23.juin	30.juin	07.juil	14.juil	21.juil	28.juil	04.août	11.août	18.août
Numéro de semaine	1	2	3	4	5	6	7	8	9	10	11	12	13	14
<i>Analyse générale du projet</i>														
<i>Prototype 1 : Choix des composants</i>														
<i>Prototype 1 : Design</i>														
<i>Prototype 1 : Programmation</i>														
<i>Prototype 2 : Choix des composants</i>														
<i>Prototype 2 : Design</i>														
<i>Prototype 2: Schéma électronique</i>														
<i>Prototype 2 : Montage</i>														
<i>Prototype 2 : Adaptation du code</i>														
<i>Tests en laboratoire</i>														
<i>Sur le terrain</i>														
<i>Analyse des tests</i>														
<i>Dashboard</i>														
<i>Reserve</i>														

Figure 3.3 Planing simplifié du projet

3.3 Conclusion



En résumé, les étapes clés sont montrées en vert, et dans l'ordre chronologique, sur la figure 3.4 ci-contre. À partir de ces étapes, des tâches précises en ont été déduites. Et finalement à partir de ces tâches, un planning (figure 3.3) a été implémenté.

On peut globalement distinguer que ce projet contient 3 parties bien distinctes qui nécessitent chacune des compétences différente : Premièrement, la partie back-end qui englobe la communication avec le capteur, le traitement des données, et l'envoie via LoRa. Deuxièmement, la partie mécanique qui consiste à concevoir et fabriquer ce qui entour le système qui doit venir se fixer sur une conduite d'un puits agricole. Et finalement la partie front-end, qui, elle, permet de visualiser graphiquement les données enregistrées par le système.

Figure 3.4 Diagrammes des étapes clés

4 | Implementation

Ce chapitre concerne la partie centrale de ce rapport : ce qui a été réalisé et comment.

Sommaire

4.1	Prototype n°1	14
4.1.1	Choix des kits de développement	14
4.1.2	Programmation	16
4.1.3	Mécanique	26
4.2	Prototype pré-industriel	28
4.2.1	Électronique	28
4.2.2	Adaptation du code	30
4.2.3	Mécanique	31
4.3	Dashboard	34
4.3.1	De TTN à la base de donnée	34
4.3.2	De la base de données à des graphiques	35
4.4	Conclusion	36

4.1 Prototype n°1

Ce prototype n°1 a pour premier but de démontrer qu'il est bel et bien possible de détecter le passage de l'eau dans une conduite de puits agricole en mesurant les vibrations. Le second but est de préparer la conception du second prototype, nommé pré-industriel.

4.1.1 Choix des kits de développement

Le choix des deux kits de développement doit être fait en pensant au fait que, dans un deuxième temps, un prototype pré-industriel va être réalisé. De même, la communication entre le kit comprenant le micro-contrôleur et celui avec l'accéléromètre doit être possible, et adaptable à d'autres types de capteurs. Finalement, il est important de penser, déjà durant cette étape, à la consommation des différents composants électroniques, le système final étant destiné à être alimenté par piles ou batteries.

Kit uC + Lora

Ce kit de développement doit contenir les éléments suivants :

- Un μ C capable de communiquer avec le capteur, que ce soit en **ADC**, en **I2C**, ou en **SPI** par exemple. Il est aussi important que ce soit un μ C qui puisse fonctionner dans différents mode afin de pouvoir économiser du courant lorsque toutes les fonctionnalités ne sont pas requises.
- Un récepteur/transmetteur **LoRa** afin d'établir une liaison à distance.
- Le kit doit pouvoir être alimenté par piles ou batteries afin de pouvoir faire des tests sans forcément avoir besoin d'une alimentation externe.

Le professeur Mudry, supervisant ce travail de bachelor, a mis à disposition du matériel qui pourrait être utilisé. Dans ce matériel, un kit contient tout le nécessaire cité ci-dessus : le B-072Z-LRWAN1 [14]. Celui-ci est constitué des éléments suivants :

- Un module CMWX1ZZABZ-091 de la marque Murata. Ce chip contient un μ C STM32L072CZ et un transmetteur/récepteur LoRa SX1276.
- Un STM32 servant de programmeur ST-Link.
- La possibilité d'alimenter cette carte avec 3 piles AAA.
- Deux connecteurs micro-USB.
- Des pin headers reliés à toutes les I/O.



Figure 4.1 Kit de développement sélectionné : B-072Z-LRWAN1

D'autres kits auraient pu être envisagés, mais étant donné que celui-ci, figure 4.1 contient tout le nécessaire et qu'il est à disposition, la décision a été prise de s'en servir.

Kit accéléromètre

Dans un premier temps, une solution simpliste a été évaluée : Un accéléromètre avec une sortie analogique. Ce type de capteur nécessite l'utilisation de l'[ADC](#) sur le micro-contrôleur.

Positif	Négatif
Facilité de prise en main	Consomme trop (plusieurs centaines de μA)
	Ne fournit pas d'interruption pour réveiller l' μC
	Il faut déterminer le timing de la prise de mesure dans le code de l' μC manuellement

Table 4.1 Accéléromètre avec sortie analogique, points positifs et négatifs.

L'accéléromètre ADXL335 [15] a été pris comme référence pour les informations de courants dans le tableau ci-dessus.

Le point déterminant est celui de la consommation. Dans ce projet, moins la consommation est élevée, plus le système fonctionnera longtemps sans remplacement de pile ou de batterie. Il est donc nécessaire d'optimiser la consommation au mieux.

C'est pour cela que, pour ce projet, il est plus intéressant de choisir un capteur, en l'occurrence un accéléromètre, avec lequel la communication se fait en [SPI](#) ou en [I2C](#). Il faut savoir que la plupart des capteurs avec lesquels il est possible de communiquer en [SPI](#), il est aussi possible de communiquer en [I2C](#).

Les caractéristiques recherchées pour cet accéléromètre sont les suivantes :

- Une plage de mesure paramétrable, au moins avec les plages $\pm 2g$ et $\pm 4g$, vu que les tests effectués par le [CREALP](#) montrent que les vibrations sont dans ces plages [10].
- Une mémoire qui peut être remplie automatiquement avec les données d'accélération à une fréquence paramétrable. Étant donné que nous ne savons pas encore quel méthode va être utiliser pour l'analyse des données, et selon le [Théorème d'échantillonnage](#), cette fréquence doit pouvoir être supérieure à 500 Hz afin d'éviter que la fréquence des vibrations soit plus grande que la moitié de la fréquence de mesure.
- Une consommation la plus basse possible.
- Une sortie qui fournit une interruption au [μC](#) pour lui éviter de devoir rester trop souvent éveillé.

Chapitre 4. Implementation

Avec ces caractéristiques, deux familles de composants s'offrent à nous : les centrales inertielles et les accéléromètres. Les centrales inertielles, telle que la BMI088 [16], contiennent les données concernant l'accélération. De plus, elles contiennent des données comme l'orientation du capteur (gyroscope). Ces dernières données n'apportent pas de plus-value aux mesures du système implémenté.

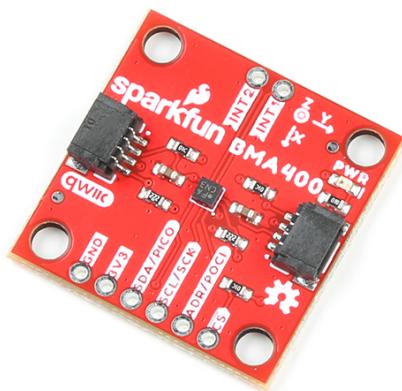


Figure 4.2 Kit SparkFun avec un accéléromètre BMA400.

Avec toutes les informations citées ci-dessus, la décision est d'utiliser l'accéléromètre BMA400 de Bosch Sensortec [17]. De plus, pour éviter de réaliser l'électronique qui va autour du capteur, un kit de développement de SparkFun [18] est utilisé pour ce premier prototype (figure 4.2). Celui-ci contient l'accéléromètre BMA400.

4.1.2 Programmation

Avant de commencer cette partie, comme pour beaucoup d'autres, il est fondamental de se remettre en contexte : quel est le but de la programmation du système à implémenter ? Récupérer les données brutes de l'accéléromètre, les traiter afin qu'elles soient utilisables pour un calcul de débit d'eau, et envoyer les données qui sont intéressantes pour le mandant, via le protocole de communication LoRa. Il y a donc trois parties distinctes à implémenter :

- La communication avec le capteur
- Le traitement des données du capteur
- La transmission des données via LoRa

L'ordre d'implémentation est logiquement dans l'ordre des points cité ci-dessus car sans communication avec le capteur, il n'y a pas de données à traiter. Et sans données traitées, rien d'utile peut être envoyé via LoRa.

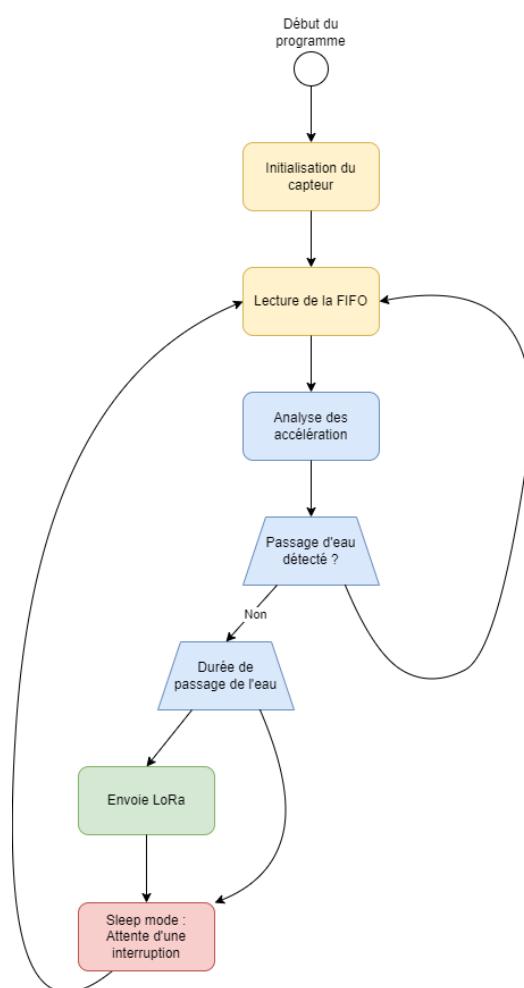
La décision a été prise de ne pas mettre le code du système embarqué en annexe. Cependant, les fichiers cités dans ce chapitre se retrouvent tous sur :

https://github.com/pablo-1424/TB_2023_Stoeri_Public_Final.

Structure du code

La réalisation d'une structure du code est une étape importante avant de se plonger entièrement dans les différentes parties concernée.

Premièrement, le langage utilisé est le C. Nous avons donc la possibilité de faire des fonctions, mais pas de classe. Pour ce genre de projet, dans lesquels la difficulté ne vient pas forcément de la structure mais plutôt des différents protocoles de communication à utiliser, la programmation orientée objet qu'aurait pu apporter le C++ n'est pas nécessaire.



Dans le code réalisé, 4 étapes clés sont à noter :

- L'initialisation du capteur.
- La prise de mesure et son traitement.
- L'envoi via LoRa.
- Le mode veille lorsque qu'on attend le prochain passage de l'eau.

Évidemment cette structure peut être améliorée, surtout au niveau de la prise de mesures. Celle-ci se fait actuellement en continu, mais on pourrait imaginer que lorsque l'optimisation de la consommation va devenir une priorité, la lecture de la FIFO n'a pas besoin d'être effectuée en continu.

Cependant, pour cette partie de programmation, l'objectif est d'abord d'implémenter un code fonctionnel et c'est pour cela que le choix a été pris de réaliser une structure simple.

Figure 4.3 Diagramme de structure du code

Communication avec le capteur

Pour commencer, un choix important a été pris : Entre les deux kits de développement, la communication s'effectue-t-elle via [SPI](#) ou [I2C](#) ? Ces deux protocoles sont largement utilisés dans le monde des systèmes embarqués. Il est donc intéressant de les comparer, afin de choisir celui qui est le plus optimal pour notre système :

- SPI a un plus grand débit de données.
- SPI peut envoyer et recevoir des données simultanément.
- I2C a besoin, sans l'alimentation, que de 2 fils pour la communication, SPI en a besoin de 4.
- I2C procure une confirmation sur la réception des données.
- I2C a un câblage moins complexe lorsqu'on utilise plusieurs slaves

Le protocole I2C a été choisi pour ces avantages cités ci-dessus qui correspondent bien aux besoins du système implémenté. La figure 4.4 représente le fonctionnement global du protocole.

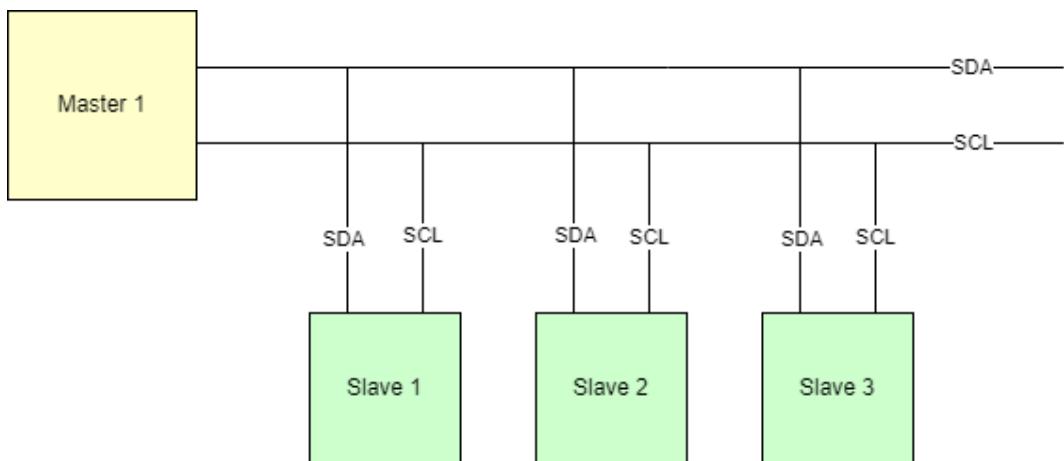


Figure 4.4 Diagramme global du protocole I2C

Pour l'implémentation de cette communication, 3 étapes ont été effectuées : Compréhension du protocole global et lecture des données dans les registres correspondants, compréhension du fonctionnement de la mémoire [FIFO](#) et utilisation de celle-ci, et finalement utilisation de la broche d'interruption à disposition sur le capteur, afin de pouvoir, en cas de début de cycle de pompage, réveiller le micro-contrôleur. La première étape citée ne se retrouve plus dans l'implémentation finale étant donné que la récupération des données d'accélérations se fait à travers la mémoire [FIFO](#).

La première étape consiste donc à établir une communication avec le capteur. Pour ce faire, il a tout d'abord été nécessaire de configurer le protocole I2C dans le [Cube MX](#). Les éléments qui ont été paramétrés sont les suivants :

- Mode : I2C
- Interruptions : Désactivées
- Pin utilisées : /I2C1_SCL sur la pin PB8 et /I2C1_SDA sur la pin PB9

Tous les autres paramètres contiennent les valeurs par défaut de l'interface [Cube MX](#). Les éléments concernant le paramétrage de l'[I2C](#) mentionnés ci-dessus restent inchangés pour toutes les étapes à venir.

Ensuite, l'objectif a été de tester la communication entre le STM32 et l'accéléromètre. Pour ce faire, il faut transmettre, à l'adresse correspondante à l'accéléromètre, la valeur du registre nommé CHIPID. Cela informe l'accéléromètre que la communication est testée et que pour confirmer qu'elle est fonctionnelle, il faut transmettre au micro-contrôleur la valeur de 0x90. Finalement, si l'[μC](#) reçoit cette valeur, la communication est donc établie correctement.

Du point de vue du micro-contrôleur, on utilise donc les fonctions "transmit" et "receive" mises à disposition par le HAL, figure 4.5, pour ce protocole I2C.

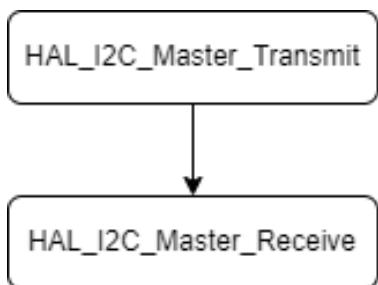


Figure 4.5 Fonctions I2C : Premières variantes

Cette méthode a aussi été utilisée afin de **lire les valeurs actuelles d'accélération**. Lorsqu'on veut connaître la valeur de différents registres avec cette méthode, le seul paramètre qui varie dans la fonction "transmit" est justement celui du registre dont on veut la valeur.

Par défaut, cet accéléromètre contient des valeurs d'accélération sur 12 bits par axe. Pour connaître les valeurs actuelles d'accélération, pour tous les axes, 6 registres sont à disposition, soit 2 par axe. C'est à dire que les 8 bits de poids faibles concernant un axes se trouvent dans un registre, et les 4 bits de poids fort se trouvent dans un autre registre. Les registres concerné dans ce cas sont les registres 0x04 à 0x09.

Une fois que cette étape a été implémentée correctement, les premiers tests en laboratoire, mentionnés plus tard dans ce rapport, ont pu être effectués.

Chapitre 4. Implementation

Ensuite, après avoir réalisé ce premier code et ces premiers tests en utilisant les fonction Transmit et Receive, il s'est agit de faire la configuration finale des registres de l'accéléromètre. Cette configuration contient différents points importants :

- Nous voulons lire la [FIFO](#), donc celle-ci doit se remplir à la fréquence voulue. Le choix a été pris de remplir la fifo à une fréquence de 800 Hz, la plus grande valeur autorisée par l'accéléromètre BMA400. Cela veut dire que 800 fois par secondes, l'accéléromètre va automatiquement mettre les valeurs d'accélérations dans la fifo, et nous pouvons venir lire toutes ces valeurs quand nous en avons besoin.
- Nous voulons que les accélérations des 3 axes soient enregistrées dans la [FIFO](#).
- La gamme d'accélération que nous voulons est de $\pm 2g$, la plus précise. Cette gamme a été choisie suite aux premiers tests en laboratoire où l'on peut voir qu'une gamme d'accélération plus large risque de ne pas fonctionner pour des raisons de précision.
- Une interruption doit être générée par l'accéléromètre lorsqu'il y a un changement d'activité. Cela veut dire que lorsque les valeurs lues par l'accéléromètre dépasse un seuil paramétré, l'accéléromètre envoie une interruptions à l'[μC](#).

Concernant ce dernier point, le seuil reste toujours le même une fois configuré. Par contre, il se fait par rapport à la valeur des x dernières valeurs (x étant aussi un paramètre des configurations). C'est à dire que dans notre cas, l'accélération au repos va être de 1g. L'accéléromètre va générer une interruption lorsqu'on dépasse le seuil paramétré par rapport à ces 1g de moyenne au repos. Des informations importantes sur le paramétrage des registres de cette détections d'activité ont été trouvée sur le site de la communauté BOSCH [\[19\]](#).

Finalement, la configuration complète de l'accéléromètre est représentée dans le tableau 4.2. La datasheet du BMA400 est la source de toutes les informations concernant ses registres [20].

Nom de registre	N° de registre	Fonctionnalité	Valeur
ACC_CONFIG0	0x19	Sleep mode -> normal mode	0x02
ACC_CONFIG1	0x1A	Range = 2g, f = 800 Hz	0x3B
FIFO_CONFIG0	0x26	Mesure de X, Y, Z en 12 bits	0xE0
INT1_MAP	0x21	Interruption générique 1 sur pin int1	0x04
INT_CONFIG1	0x20	"non-latched" mode	0x00
INT12_IO_CTRL	0x24	Pin int1 en mode "high-active"	0x02
GEN1INT_CONFIG0	0x3F	Détection de changement d'activité sur X, Y, et Z.	0xFB
GEN1INT_CONFIG1	0x40	Détection au dessus du seuil.	0x02
GEN1INT_CONFIG2	0x41	Seuil de détection de 24 mg.	0x03
GEN1INT_CONFIG3	0x42	Durée de dépassement MSB.	0x00
GEN1INT_CONFIG31	0x43	Durée de dépassement LSB.	0x01
INT_CONFIG0	0X1F	Interruption 1 générique ajoutée au contrôle d'interruption	0x04

Table 4.2 Configuration des registres de l'accéléromètre.

Toute cette configuration se fait à l'appel de la fonction BMA400_InitConfig(). La prochaine étape de ce chapitre est la **lecture de la FIFO**. Tout d'abord, la lecture se fait à l'aide de la fonction HAL_I2C_Mem_Read(...). C'est donc une méthode différente de celle qui avait été utilisée pour tester la communication entre le micro-contrôleur et l'accéléromètre.

Avant d'aller lire le registre contenant les bytes d'accélération de la **FIFO**, il est nécessaire de lire les deux registres indiquant combien de valeurs sont actuellement stockées dans la mémoire de l'accéléromètre. Cela permet de par la suite, lors de la lecture des valeurs, indiquer combien de bytes doivent être lu.

Dans un premier temps, une erreur a été commise pour cette lecture. Nous savons que pour la valeur de l'accélération d'un axe, 2 bytes doivent être lus : le byte de **MSB** et le byte de **LSB**. Par déduction, nous avons donc lu l'intégralité de la **FIFO** et séparé le nombre de bytes en bloc de 6 bytes (2 bytes par axe). Le résultat de cette lecture n'a pas été concluant. Le byte de séparation en est la raison. C'est-à-dire qu'à chaque fois que l'accéléromètre exécute une lecture des 3 axes et en met les valeurs dans la mémoire, il ajoute un byte de séparation.

Une fois que la séparation des bytes lu dans la mémoire a été fait par bloc de 7 bytes, le problème a été résolu et le traitement de ces données peut réellement commencer. La première étape préliminaire au traitement est de mettre les **MSB** et leur **LSB** correspondant ensemble.

Traitement des données du capteur

Tout d'abord, il est important de comprendre les données reçues afin de savoir exactement à quoi elles correspondent. Idéalement, une accélération se mesure en [g] ou en [m/s²]. Nous savons, grâce à la datasheet du capteur BMA400 que lorsque la valeur reçue avec la configuration citée ci-dessus est 1024, l'accélération mesurée est de 1g = 9.81 m/s. La décision a été prise de ne pas transformer ces valeurs reçues en g ou en m/s. La raison de cette décision est que, même si ces deux unités de mesure sont plus intuitives, il n'y a pas d'utilité à ajouter des nombres à virgules dans le code pour ce travail. Nous restons donc avec une plage de -2 à 2g qui correspond à -2048 à 2048 dans le code.

Ensuite, étant donné que les valeurs d'accélérations des 3 axes sont lues dans la [FIFO](#), il est intéressant d'en faire la norme N.

$$N = \sqrt{x^2 + y^2 + z^2}$$

Ce calcul nous permet de connaître l'accélération globale du système. Les axes séparés ne nous apportent que très peu. Il aurait été possible de travailler sur les 3 axes séparés mais une calibration aurait été nécessaire une fois le système installé. Cela est dû au fait que qu'il faut savoir la valeur de base des accélérations des différents axes pour pouvoir ensuite en déduire des vibrations, et donc du passage de l'eau.

Avec la norme, la calibration n'est pas nécessaire étant donné que la norme des 3 axes est toujours égale à 1g, soit 1024 dans le code, lorsque le système est au repos. Et cela quelque soit la position du système.

Maintenant que la valeur normée est connue, la décision a été d'utiliser la **méthode de seuillage**. C'est à dire qu'une zone est déterminée autour de la valeur de repos, et on part du principe qu'une fois qu'on sort de cette zone, de l'eau passe à travers la conduite. Par contre, un simple seuillage ne suffit pas car il est possible que certaines valeurs lues dans la [FIFO](#) soient erronées ou perturbées.

La solution a été de faire une filtre par comptage des valeurs dépassant le seuil. Dans le cas où ce compte est supérieur à la valeur fixée, 14 en l'occurrence, on peut cette fois dire avec certitude qu'un flux d'eau traverse la conduite. La valeur fixée, nommée "MIN_COUNT", a été déterminée en faisant des tests en laboratoire. La valeur du seuil, nommée "THRESHOLD", fixée à la valeur de 30, a été déterminée de la même manière.

La dernière étape restante avant de pouvoir transmettre les données via [LoRa](#) est de compter le nombre de secondes durant lesquelles un flux d'eau a été capté. Cette tâche se fait avec un compteur hardware (TIM6). Ce timer est configuré afin que le callback soit appelé toutes les secondes. À l'intérieur de ce callback nous implémentons la variable contenant le nombre de secondes d'allumage du puits. Cette variable, nommée "seconds", est un tableau de deux bytes, un [LSB](#) et un [MSB](#), ce qui nous permet de compter jusqu'à 65'535 secondes, soit 18 heures. Étant donné que Stéeve Maillard nous a transmis l'information que les puits ne sont et seront jamais en activité pendant autant de temps en une seule fois, 2 bytes suffisent largement.

Le choix de travailler avec deux byte et non pas d'utiliser un `uint16_t` a été pris car de cette manière, aucune action spécifique est nécessaire avant de pouvoir remplir le tableau de byte qui va être envoyé via [LoRa](#).

Transmission LoRa

La transmission LoRa a été le plus grand obstacle rencontré dans ce projet. Cette étape a été la raison d'un certain retard sur le planning.

Pour commencer, il faut comprendre que dans ce travail, le but n'est pas de coder la couche physique de la communication LoRa. Le but est comme pour l'[I2C](#), d'utiliser des fonctions déjà implémentées afin de pouvoir transmettre les données désirées.

Malheureusement, le chip contenant le micro-contrôleur et le transmetteur LoRa ne peut pas être choisi tel quel dans les configurations Cube MX. Ceci a pour conséquence qu'on ne peut pas configurer la transmission [LoRa](#) et générer le code qui va avec. La configuration dans Cube MX est donc uniquement pour le micro-contrôleur du chip et pas pour le chip en entier.

La conclusion de ce fait est la suivante : Une partie de code contenant la communication en LoRa va devoir être importée.

Le premier code qui a été utilisé est l'expansion LoRa qui est implementé par STMicroelectronics : I-CUBE-LRWAN [21]. Une vidéo [21] expliquant l'utilisation basique de cette expansion a permis de rapidement recevoir des données via LoRa sur [TTN](#). Les éléments importants qui ont été réalisés pour faire cette première communication sont les suivants :

- Résoudre les "include path" dans les propriétés du projet afin de ne plus avoir d'erreur de compilation.
- Paramétriser les valeurs liées au kit et à TTN afin de pouvoir les connecter

Cette expansion a été utile car elle a aussi permis de voir rapidement si le [PCB](#) du prototype pré-industriel est fonctionnel. Cependant, là où cette expansion est limitée, c'est qu'il est extrêmement complexe de l'intégrer à un projet existant ou de le modifier pour faire autre chose qu'uniquement envoyer des données via LoRa. Ceci est dû au fait que, malheureusement, cette expansion n'a pas été réalisée à l'aide de Cube MX, ou que les fonctions du Cube MX ont été modifiées. Des tentatives d'utilisation de cette expansion avec un code contenant la communication avec le capteur, les erreurs de compilations ont été si nombreuses et difficilement résolvables que la décision a été prise d'abandonner son utilisation.

Chapitre 4. Implementation

De ce fait, une autre solution a dû être trouvée. Celle qui a été la plus convaincante est l'utilisation du code de Paul Pinault nommé "IT-SDK" [22].

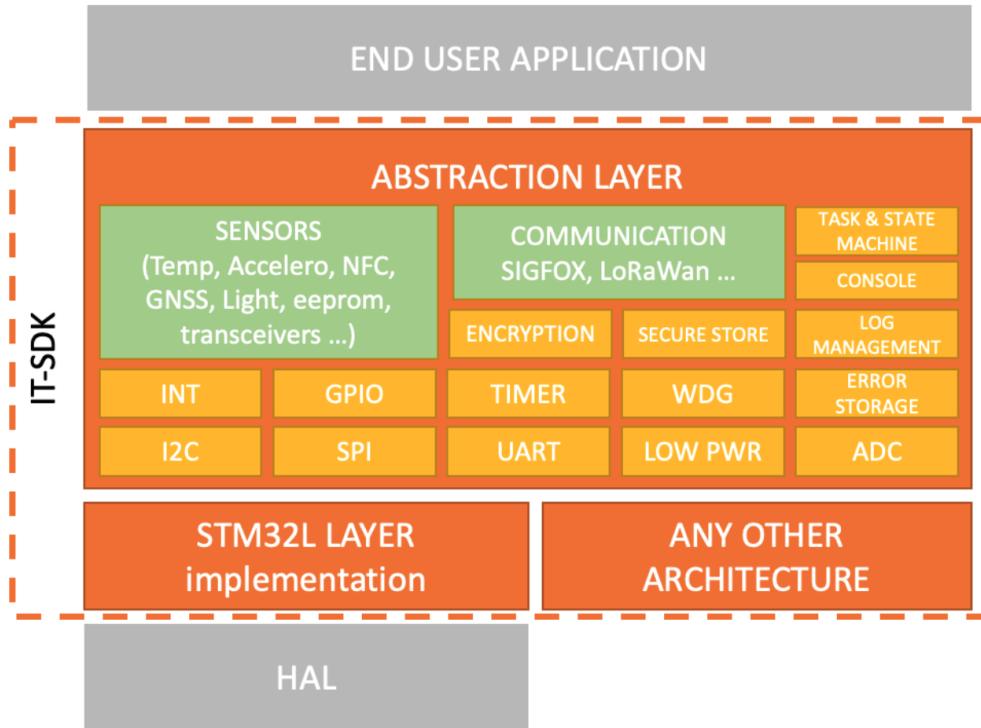


Figure 4.6 Structure de l'IT-SDK

Un [SDK](#) procure un certain nombre de fichiers et de fonctions qui aident au développement software d'un système. En l'occurrence, cet "IT-SDK" choisi fournit une couche d'abstraction en plus contenant entre autres toutes les fonctions du [HAL](#) modifiées. Ces fonctions sont modifiées dans l'optique qu'il serait possible d'utiliser ce [SDK](#) avec d'autre micro-processeurs. Ce kit de développement software permet aussi d'implémenter des machines d'états, et une gestion des tâches et des timers. Mais le plus important pour ce projet est qu'il fourni des fonctions claire concernant la connexion et les envois [LoRa](#).

Pour commencer l'utilisation de ce [SDK](#), un exemple, également fournit par Paul Pinault, a été utilisé [23]. Les étapes à suivre pour la mise en place de cet exemple sont présents sur la page GitHub citée ci-dessus. Une première compilation de cet exemple a été exécutée. Celle-ci a générer plusieurs erreurs concernant les "includes". Ce problème a été résolu en modifiant les "include paths" dans les propriétés du projet.

Une fois que la compilation a été accomplie dans erreurs, les paramètres concernant la partie [LoRa](#) on pu être ajustés dans le fichier configLoRaWan.h. Cette étape étant terminée, le code a pu être chargé dans le kit de développement et des données ont été reçues sur [TTN](#).

Il est important de comprendre à ce niveau-là pourquoi la solution "IT-SDK" et plus avantageuse que la solution I-CUBE-LRWAN. Le [SDK](#) est construit sur un projet [Cube MX](#) ce qui permet de modifier les paramètres du projet facilement. En plus de cela, l'envoie [LoRa](#) sur la solution de STMicroelectronics est très complexe et peu adaptable à d'autres situations. En résumé, le SDK de Paul Pinault est un support à ce projet, tandis que l'expansion I-CUBE-LRWAN est une base compliquée sur laquelle il faudrait venir greffer des parties de ce projet.

Ensuite, le choix a été fait d'utiliser uniquement la partie concernant [LoRa](#) de cet SDK et d'utiliser directement les fonctions du [HAL](#) pour le reste (I2C, timers, interruption, et stop mode). Cet décision a été prise pour deux raisons. La première est qu'il est possible que cet solution contenant le [SDK](#) ne soit pas la bonne. Dans ce cas il est intéressant d'avoir un code qui fonctionne sans le SDK, excepté pour la partie [LoRa](#). La deuxième raison est qu'il aurait été nécessaire de comprendre chaque partie du développement de Paul Pinault, comme la gestion des timers, la gestion des interruptions, la gestion des tâches, et bien d'autres encore.

Les deux seuls éléments qui ont posé problème suite à cet décision est l'utilisation des callbacks de timers et d'interruptions. C'est à dire que le SDK contient déjà ces callbacks, et qu'il a donc été nécessaire d'aller modifier les fichiers dans lesquels ils sont présents afin de traiter les élément nécessaires.

Code final

Pour information, le code complet utilisé sur le prototype pré-industriel est dans le dossier "TB_Stoeri_FinalCode". Le fichier "README.txt" dans ce même dossier donne des informations sur les différents fichiers.

La fusion entre la partie [LoRa](#) et la partie mesures/traitement des données a été une étape clé. Celle-ci a notamment été possible grâce à l'[SDK](#). Il y a tout de même certains points qui on dû être clarifiés. Premièrement, étant donné que nous utilisons les fonctions du HAL et non pas celles de l'SDK pour certaines actions mentionnées ci-dessus, la partie watchdog de l'IT-SDK a dû être désactivée dans l'interface graphique CubeMX et dans le fichier de configuration du SDK. Cette modification n'avait pas été faite dans un premier temps, ce qui avait pour conséquence que le programme se redémarrait à chaque fois après 27 secondes. C'est en fait grâce à ce temps constant entre le début du programme et le redémarrage que nous avons pu trouver que l'erreur venait du watchdog. Celui-ci n'a pas d'utilité particulière dans notre code actuel, mais il est intéressant de voir que ce genre de configuration peut aussi être faite avec le [SDK](#).

Ensuite, comme mentionné ci-dessus, nous avons décidé d'utiliser uniquement le SDK pour l'envoie [LoRa](#). Cependant, afin d'avoir une cohérence entre la configuration du SDK et la configuration que nous générions sur [Cube MX](#), il est nécessaire de modifier certains éléments dans le fichier de configuration général du SDK nommé config.h. Globalement, les éléments importants de cette configuration sont les suivants :

- Activation de l'utilisation l'[I2C](#)
- Activation de l'utilisation de timers hardware
- Activation de la librairie LoRa

Chapitre 4. Implementation

Le mode stop est une des dernières étapes qui a été implémentée durant ce projet. Les deux étapes exécutées afin d'entrer dans ce mode sont l'appel des fonctions HAL_SuspendTick et HAL_PWR_EnterSTOPMode.

Afin de réveiller le micro-contrôleur de ce mode, une interruption sur la pin PB12 est nécessaire. La gestion du Callback de cette interruption est implémentée dans le fichier gpio_wrapper.c de l'SDK. Les deux fonctions appelée au réveil sont SystemClock_Config et HAL_ResumeTick. Celles-ci permettent de reprendre le déroulement du code là où il avait été suspendu.

De même, la gestion du callback du timer TIM6 qui permet de compter le nombre de secondes d'allumage de la pompe est dans le fichier timer.c du SDK.

Finalement, le code implémenté pour ce projet, selon la structure mentionnée précédemment, permet de lire les valeurs d'accélération dans la [FIFO](#), compter le nombre de secondes pendant lesquelles de l'eau traverse la conduite en question, et d'envoyer ce nombre de secondes via [LoRa](#) sur l'application TTN. Et lorsqu'aucune de ces actions est nécessaire, le système est conçu pour se mettre en mode STOP.

La décision d'envoyer un nombre de secondes et non pas un nombre de litres ou de m^3 d'eau consommé est simplement due au fait qu'il est plus facile pour l'utilisateur d'installer le système et de pouvoir modifier dans le script python les données spécifiques au puits, c'est-à-dire le débit moyen. Le script python en question, en annexe [E](#), est expliqué dans la partie Dashboard de ce rapport.

4.1.3 Mécanique

Le but de cette partie mécanique est de faire une pièce permettant de fixer les deux kit de développements sur une conduite et de pouvoir ainsi faire des premiers tests. Les points clés de ce premier développement mécanique du projet sont les suivants :

1. La forme doit épouser au mieux différents type de conduites.
2. Le système doit pouvoir se fixer solidement sur une conduite
3. La fixation du kit contenant l'accélémètre doit être réfléchie pour être le plus rigide possible.
4. Le procédé de fabrication
5. La fixation du kit de développement programmable est facultative et n'ajoute pas de contraintes particulière.

Après avoir été discuter des quatre points ci-dessus et du projet dans sa globalité avec les personnes compétentes pour cette partie mécanique, c'est à dire à l'atelier d'hydraulique, à l'atelier de mécanique et à l'impression 3D, de nombreux éléments en sont ressortis.

Premièrement pour épouser la forme de la conduite le mieux possible, tout en sachant que la taille de la conduite n'est pas fixe, le choix a été de faire une forme triangulaire. C'est en fait la seul option qui a été trouvée. En effet, toute autre forme avait soit un nombre de points de contacts plus faible, ce qui fait que les forces des vibrations seraient moins bien transmises, soit une forme qui ne permet pas de s'adapter à plusieurs diamètres de tube. En second lieu, au niveau de la fixation sur la conduite, les personnes

4.1 Prototype n°1

à l'atelier hydraulique ont proposé différentes sortes d'attaches, essentiellement des brides en métal et en plastique. Les brides en métal sont celles qui sont le plus solides, qui transmettent le mieux les forces, et qui, si elles sont en acier inoxydable, ne sont pas affaiblies par la pluie ou l'humidité. C'est donc le choix qui a été fait. Ensuite, il a été recommandé de fixer l'accéléromètre directement sur la conduite. Il a tout de même été décidé que, dans un premier temps, l'accéléromètre soit à l'intérieur de la pièce. Cela permet d'analyser si les accélérations se transmettent de manière correcte à travers la pièce. Si ce n'est pas le cas, dans un second temps, le système sera revu pour le prototype final afin de placer l'accéléromètre directement sur la conduite. L'avantage de le mettre à l'intérieur de la pièce est que, pour le prototype pré-industriel, l'étanchéité est plus facilement garantie.

Concernant le procédé de fabrication de cette première pièce, il a été choisi de la réaliser à l'impression 3D au sein de la HES à Sion. Cela a permis d'avoir une pièce rapidement fabriquée, et surtout de pouvoir analyser si ce genre d'impression 3D est envisageable pour la version finale. Et pour conclure, il a été décidé que le kit de développement qui contient le micro-contrôleur sera aussi directement vissé sur cette pièce pour des raisons pratiques, et car cela ne demande pas d'effort particulière au niveau du design.

À l'aide de tous ces choix et de toutes ces informations, le premier dessin 3D, figure 2.2, a pu être réalisé sur Inventor Professional 2024.

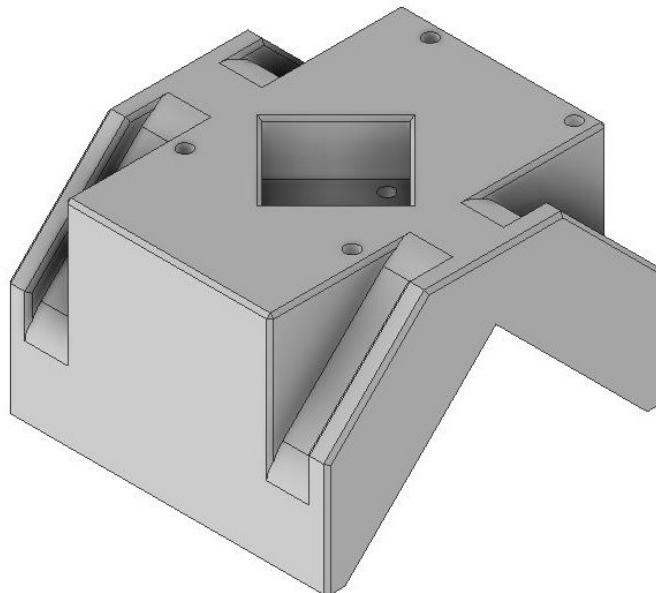


Figure 4.7 3D de la pièce de fixation pour les kit de développement.

Étant donné que cette impression n'a ni besoin d'être étanche, ni besoin d'être résistante aux UV, le fil PLA a été choisi et l'impression a été réalisée dans une imprimante PRUSA i3 MK3S+ [24].

Chapitre 4. Implementation

La procédure pour imprimer des pièces en 3D avec ce type d'imprimante est la suivante :

1. Réaliser la pièce sur un logiciel 3D, Inventor Professionnal 2024 en l'occurrence.
2. Exporter le fichier en un ".stl".
3. Configurer l'impression sur le logiciel "PrusaSlicer"
4. Une fois que le G-Code correspondant a été chargé sur la carte SD, insérer la carte dans l'imprimante et lancer l'impression.

Le dessin mécanique de ce prototype n°1 se trouve en annexe [J](#)

4.2 Prototype pré-industriel

L'objectif du prototype pré-industriel dans ce projet est de fournir un système complet, avec les mêmes fonctionnalités que le premier prototype, mais en ajoutant les contraintes suivantes :

- La résistance aux conditions extérieures
- Le prix qui doit être aussi bon marché que possible.
- Les tailles des conduites variables.

4.2.1 Électronique

La partie électronique de ce projet est encore une fois un point clé. Elle permet non-seulement d'avoir un PCB qui convient parfaitement au travail à réaliser, mais c'est aussi intéressant pour le [CREALP](#) de voir ce qui peut être réalisé dans le cas où le système implémenté durant ce projet serait produit en quantité (plusieurs centaines). Le prix d'un kit de développement, comme celui sur lequel la première partie du travail a été réalisée, est un frein à la production à bas coût.

Par contre, la décision a été prise de garder le kit de développement contenant l'accéléromètre. Ce choix est dû au fait qu'il ne contient pas plus que ce qui est nécessaire à ce projet, excepté deux connecteurs et une [LED](#), dont la piste d'alimentation a été sectionnée pour des raisons de consommation. Ce sectionnement est conseillé sur le schéma du kit [\[25\]](#).

La méthode utilisée pour ce développement électronique est d'utiliser le micro-contrôleur présent sur le kit de développement concerné et de s'inspirer du schéma de ce kit en n'utilisant que les composants nécessaires au prototype. Les deux plus grands changements qui ont été effectués entre le kit de développement et le circuit imprimé réalisé pour ce prototype sont les suivants :

- Le kit de développement contient un [µC](#) qui sert de ST Link pour la programmation. Ce micro-contrôleur n'a aucun besoin d'être remis sur le circuit imprimé de ce prototype car il est possible, et même nécessaire, de le remplacer par un ST Link externe qui nous permet de programmer notre STM32.
- Sur le kit, toutes les entrées/sorties du micro-contrôleur sont reliées à des connecteurs sur le [PCB](#). Mais cela n'est pas nécessaire dans le cadre de ce projet, étant donné que nous savons que certaines pin resterons inutilisées. En ne reliant pas ces [I/O](#) à des connecteurs, on économise de la place sur notre circuit imprimé revu.

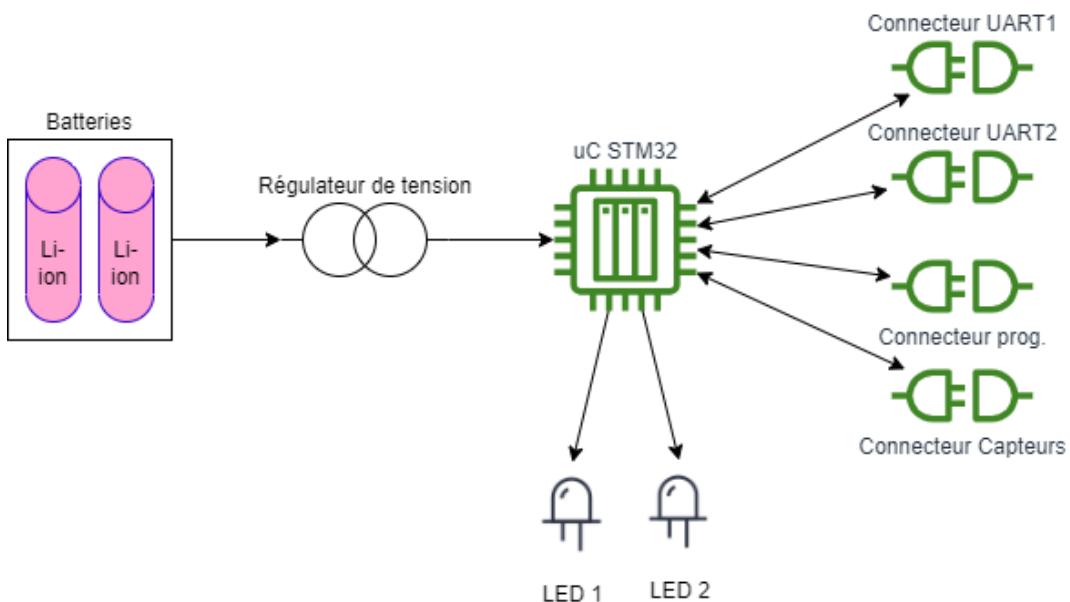


Figure 4.8 Schéma électronique simplifié.

Les schéma électronique simplifié 4.8, inspiré du schéma complet, en annexe I, montre tous les composants présents sur le [PCB](#) de ce prototype pré-industriel pour lesquels plusieurs choix ont été pris. Tout d'abord, comme dit précédemment, le microcontrôleur est exactement le même que sur le kit de développement. Théoriquement, cela permet de ne rien avoir à changer dans le code réalisé pour le kit. Ensuite, le régulateur de tension est aussi identique à celui présent sur le kit. Du fait que le kit est, comme le [PCB](#) réalisé, alimenté par des piles ou par du 5V provenant des connecteurs micro-USB, l'utilisation du même régulateur de tension est une solution.

De même, afin de pouvoir faire des tests, et même du débogage, deux [LED](#) ont été placée sur le circuit imprimé.

Au niveau de l'antenne, il aurait été possible de designer une antenne directement sur le [PCB](#). L'étanchéité aurait été plus facilement réalisée et de la place aurait pu être économisée. Par contre, l'antenne aurait été extrêmement proche des batteries/piles, ce qui aurait probablement engendré plus de problèmes de portée. La décision a donc été prise de faire mettre un connecteur pour la même antenne utilisée sur le kit de développement. La seule contrainte est de faire une ligne de 50 Ohms entre l'[μC](#) et le connecteur.

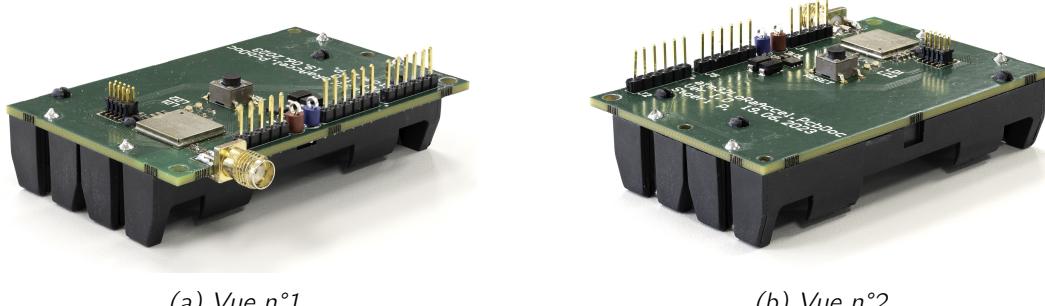
Et pour conclure ces choix concernant les composants électroniques, le choix a été pris d'utiliser des batteries [Lithium-ion](#). Ces batteries fournissent une tension nominale de 3,7 Volts et une **capacité d'environ 2600 mAh**. Cette capacité a été le point clé pour le choix de l'alimentation. La première solution a été de mettre 3 pile AAA, comme sur le kit de développement utilisé. Celles-ci ont une capacité d'environ 1250 mAh, ce qui correspond à la moitié d'une batterie Li-ion. Les pile AA sont la dernière solution qui concurrence avec les batteries choisies. Celle-ci ont une capacité semblable aux batteries Li-ion, mais lorsqu'on choisit d'en prendre des rechargeables, les pile AA ont une capacité légèrement inférieure aux batteries Li-ion.

Chapitre 4. Implementation

Il est intéressant de noter aussi la courbe de décharge de ces piles [Li-ion](#). Cette courbe montre une excellente stabilité de la tension durant la majeure partie de la décharge, surtout avec les faibles courants qui vont être utilisés par le système.

Par contre, le prix des batteries 18650 [Li-ion](#) est plus élevé, 71.- pour 8 pièces sur [digitec.ch](#).

Pour finir ce chapitre sur l'électronique du système, le routage du [PCB](#) en deux couches, en annexe I, et le montage des composants a principalement été réalisé par les spécialistes du laboratoire d'électronique au sein de la HES. Cela a été un énorme gain de temps et une garantie en plus concernant le fonctionnement de l'électronique.



(a) Vue n°1.

(b) Vue n°2

Figure 4.9 Résultat final de la partie électronique.

Une fois que le [PCB](#) a été routé, commandé, et monté, la première vérification a été de voir si tous les GND se connectaient entre eux et si tous les 3V3 se connectaient aussi entre eux (test au multimètre). Et finalement, le PCB a pu être alimenté en 5V et un contrôle de tension sur la broche de test rouge "3V3" a été effectué. La tension étant bien à 3.3V, les premiers codes ont pu être testés.

4.2.2 Adaptation du code

L'adaptation du code afin qu'il puisse fonctionner sur le nouveau [PCB](#) a été moins complexe que prévu.

Le seul changement qui a été réalisé est le suivant : Connaissant la schématique du [PCB](#), les sorties correspondantes aux LEDs doivent être mises à 3.3V lorsqu'on veut les éteindre.

Par contre, le chargement du code sur le micro-contrôleur ne se fait plus comme avec le kit de développement. Vu que le nouveau [PCB](#) ne contient pas de programmeur ST-LINK intégré, il a fallu en utiliser un externe au système. Ensuite, le chargement du code s'est fait à l'aide de l'application STM32CubeProgrammer.

4.2.3 Mécanique

Le but de ce deuxième prototype est d'implémenter un système qui pourrait être fait en plus grande quantité. Le choix des composants et de la matière est très important en vue de l'utilisation en extérieur du système.

Le premier développement mécanique qui a été réalisé pour les premiers tests avec les kits de développement a été un succès, étant donné qu'il a rempli ses deux objectifs : Permettre de fixer le système sur une conduite et transmettre assez de vibration pour qu'elles soient détectables.

Pour le développement mécanique final, des contraintes se rajoutent :

- Le boîtier doit être étanche et résistant aux [UV](#), afin de résister aux conditions extérieures.
- La mécanique doit aussi être bas coût.

Une dernière contrainte se rajoute. Suite à une première prise de mesure sur le site de Châteauneuf, il a été observé que les conduites principales des puits agricoles sont de tailles variables. Elles ont des diamètres entre 80mm et 200mm. La dernière contrainte qui se rajoute donc pour cet aspect mécanique est que le système doit être adaptable à ces différents diamètres, sans avoir à refaire un design complet.

Concept

Pour le concept général de ce design mécanique, deux solutions ont été prises en compte :

1. Reprendre le concept d'une impression 3D comme pour le prototype n°1 montré sur la figure [4.7](#), et ajouter de la matière afin d'étanchéifier la partie électronique du système.
2. Séparer le design en deux parties : L'attache sur la conduite, et le boîtier étanche contenant l'électronique.

Après discussion avec les personnes qualifiées du laboratoire de mécanique et celui d'impression 3D, tous les deux au sein de la HES, la deuxième solution est plus intéressante. Séparer le système en deux parties a un avantage conséquent, il est beaucoup plus flexible. La taille des conduites est variable ; donc l'attache sur ces conduites va aussi être de taille variable. Avec cette séparation du design en deux parties, uniquement l'attache sur la conduite doit être adaptée.

Cette solution est aussi avantageuse car dans ce projet, deux boîtiers différents vont être réalisés : Celui pour le kit de développement et celui pour le prototype pré-industriel. Il est donc possible, avec cette solution, de créer différents boîtiers qui viennent sur les mêmes attaches.

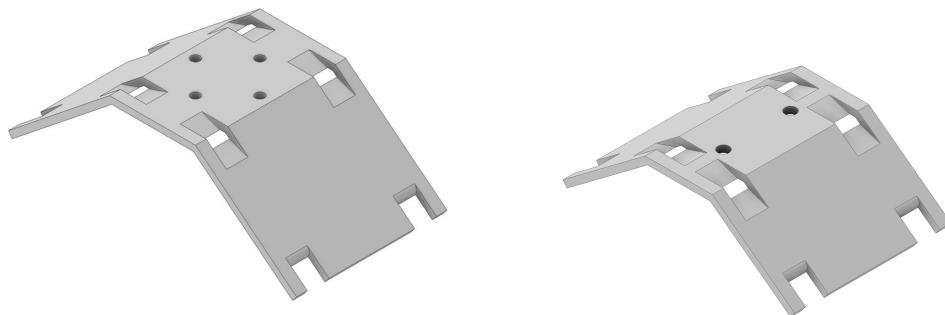
Attaches

Tout d'abord, l'objectif de ces fixations est de s'attacher solidement sur les différentes conduites et de permettre une installation sûre des différents boîtiers, afin de faciliter la transmission des vibrations provenant des conduites.

Nous savons, suite au tests en laboratoire effectués avec le prototype n°1, que les vibrations se transmettent suffisamment à travers l'impression 3D. Cela veut dire que pour ces attaches, le matériau utilisé doit être d'une rigidité égale ou plus élevée que celui de l'impression 3D.

Par contre, pour que les attaches en impression 3D soient assez solides, il serait nécessaire de faire une pièce volumineuse, semblable à celle qui a été faite pour le prototype n°1 (figure 4.7).

Finalement, la solution choisie pour ces pièces de fixation est de faire ces attaches en tôle d'acier inoxydable. Le design simple, montré sur la figure 4.10, est idéal pour ces attaches. Percée, pliée, et fraisée, la tôle de 3mm d'épaisseur est largement assez solide et peu volumineuse. L'avantage est aussi que ce design nécessite peu de matière et d'usinage. De même, la différence entre une attache pour une conduite de 80mm de diamètre et celle pour une conduite de 200mm de diamètre est très faible. Le design reste intacte, excepté les angles des pliages et les dimensions de la tôle qui doivent être adaptés.



(a) Tôle de fixation pour conduite de taille 1. (b) Tôle de fixation pour conduite de taille 2.

Figure 4.10 Design des tôles de fixation

Ce design avec des tôles de 3mm d'épaisseur possède un dernier avantage pour ce projet : Le boîtier va venir se fixer très proche de la conduite (< 10mm) ce qui a pour conséquence que la distance entre l'accéléromètre et la conduite est la plus petite possible. L'avantage de cette distance réduite est que les vibrations seront moins atténuerées par la mécanique.

Les dessins mécaniques de ces deux tôles se trouvent en annexe K

Boîtier

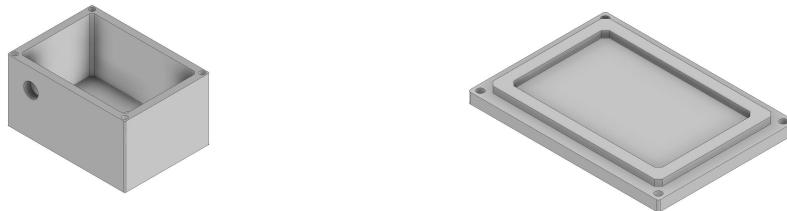
Le boîtier du prototype pré-industriel doit respecter les contraintes dues à l'utilisation à l'extérieur.

Étant donné que l'impression 3D est une solution rapide et peu coûteuse, elle a été la première solution envisagée. Le seul obstacle est le fait que certains filaments ne résistent pas aux conditions extérieures, c'est-à-dire aux [UV](#) et au projections d'eau.

À chaque obstacle, sa solution. En l'occurrence, l'idéal aurait été de faire une impression [SLS](#). Ce type d'impression est résistante aux [UV](#), et, du fait que le plastique est déposé en fusion, ce type d'impression est également étanche. Suite à un problème technique sur cette imprimante au sein de la HES, cette solution a dû être mise de côté.

Finalement le choix a été le suivant : Faire un impression comme pour le prototype n°1 mais cette fois-ci avec un filament [ASA](#). L'avantage de celui-ci est qu'il résiste mieux aux [UV](#) ce qui résout déjà une partie de l'obstacle cité ci-dessus.

Le design de ce boîtier final, figure 4.11, est réfléchi pour qu'il soit le plus simple possible. La simplicité permet souvent de réaliser des meilleures impressions 3D. Les parois sont aussi fines que possible afin d'économiser du poids. Le couvercle possède des détachements afin de pouvoir venir le caler idéalement contre la partie inférieure.



(a) Partie inférieure du boîtier.

(b) Couvercle.

Figure 4.11 Design du boîtier final.

Afin de rendre ce boîtier étanche, un traitement de surface a été appliqué sur ses faces externes. Le fluide XTC-3D [26] permet de remplir les fines rainures créées par l'impression 3D.

Étant donné qu'à ce niveau-là du projet, nous ne savions pas encore si la partie électronique de ce prototype allait être fonctionnelle, un second boîtier a été créé afin d'y mettre les deux kits de développement comme pour le premier prototype. Ce boîtier pouvait également venir se fixer sur les tôles. Cependant, étant donné que l'électronique a rapidement été fonctionnelle, ce montage n'a jamais été utilisé.

Le dessin mécanique de ce boîtier du prototype pré-industriel se trouve en annexe K

4.3 Dashboard

Cette partie Frontend a pour but de pouvoir visualiser les données sous forme de graphiques ou de cartes. Cette étape contient deux tâches séparées :

- Récupérer les données de TTN et les enregistrer dans une base de données.
- Utiliser les données enregistrées afin de pouvoir les visualiser graphiquement.

Pour chacune des tâches ci-dessus, un script a été réalisé. Le langage python a été utilisé pour réaliser ces deux objectifs. Ce langage de programmation a de nombreux avantages. Sans notre cas le plus grand avantage est qu'il est extrêmement bien documenté et permet d'importer de nombreuses librairies.

4.3.1 De TTN à la base de donnée

Afin de pouvoir récupérer les messages reçus sur l'application [TTN](#) sur une base de donnée, sont les pré-requis :

- Créer une nouvelle "API key" dans l'application TTN ce qui permet d'y avoir accès.
- Créer une base de données permettant d'enregistrer toutes les données reçues

La base de donnée choisie est mySQL. Ce n'est peut-être pas la meilleure solution pour des données de type timeseries, mais elle est gratuite, facile à mettre en place, et la documentation fournie par les utilisateurs concernant les différentes actions possibles avec mySQL est excellente. Cette base de données sert, dans notre cas, uniquement à sauvegarder des données et de montrer qu'il est possible de créer facilement un dashboard à partir des données de TTN.

Nous utilisons, dans le script concerné en annexe E, la librairie "paho" qui nous permet de récupérer les données reçues sur [TTN](#) via le protocole de messagerie publish-subscribe [MQTT](#). Celui-ci est basé sur le protocole TCP-IP. Un exemple fournit sur le site officiel de TTN [27] a permis de réaliser cette étape sans problème.

Ensuite, lorsqu'un nouveau message arrive sur TTN, et donc est reçu via MQTT dans notre script, le nombre de secondes reçues est multiplié par le débit en litres par seconde, ce qui permet d'obtenir le nombre de litres consommés.

Finalement, nous utilisons la librairie mysql [27] afin d'accéder à la base de données mySQL créée en local. Afin de pouvoir transmettre les données sur mySQL, la première étape est de se connecter à la base de données. On peut ensuite effectuer la commande voulue afin d'insérer nos valeurs dans cette même base de données.

4.3.2 De la base de données à des graphiques

Cette partie aurait pu être effectuée à l'aide d'autres outils, comme Excel par exemple. Cependant il est intéressant pour l'utilisateur de simplement pouvoir lancer un script et c'est pour cette raison qu'il a été décidé d'en implémenter un.

Avec la même librairie MySQL, citée précédemment, il est donc possible de se connecter à la base de donnée. Mais cette fois-ci la commande change, nous ne voulons pas insérer des données mais les lire.

Deux variantes de requêtes SQL sont présentées dans les scripts. L'une d'entre elle permet de faire un graphique sur toutes les valeurs présentes dans la base de données et l'autre de faire un graphique sur les 6 dernières valeurs mesurées.

Les deux scripts sont identiques, excepté pour la forme du graphique : Le premier [F](#) indique combien d'eau a été consommé à chaque cycle d'utilisation individuellement. Le second [G](#) montre le total de l'eau consommée dans le temps.

Les résultats de ces deux graphiques se retrouvent respectivement sur les figure [5.5](#) et [H.8](#).

4.4 Conclusion

Pour conclure cette partie centrale, le bilan est le suivant :

- Un **premier prototype**, figure 4.12, **contenant deux kits de développement** a été réalisé. Celui-ci permet de faire des tests en laboratoire, mais pas sur le terrain, étant donné qu'il n'est ni étanche, ni résistants au **UVs**.
- Un **prototype pré-industriel**, figure 4.13, a été implémenté. Celui-ci contient une électronique revue et une mécanique adaptable à différentes conduites. De même, il a été conçu de manière à ce qu'il puisse résister aux **UV** et qu'il soit étanche. Cependant, ces deux caractéristiques n'ont pas été testées durant ce projet.

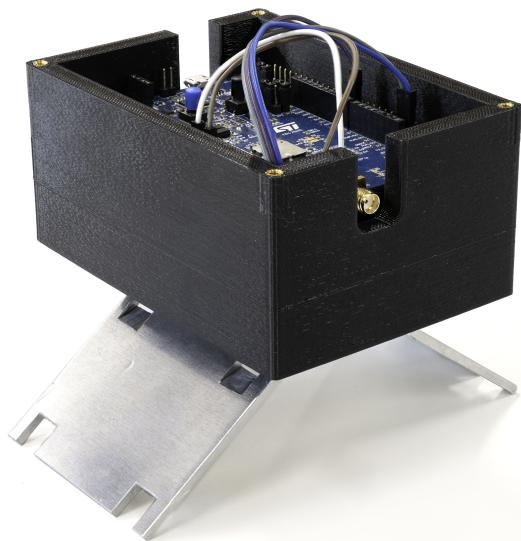


Figure 4.12 Montage du prototype n°1 (sans couvercle).

Au niveau de la **programmation**, le code implémenté permet de détecter si de l'eau passe à travers la conduite, de calculer pendant combien de temps, et d'envoyer cette information via **LoRa**.

Et finalement, un **script python** a été implémenté afin de pouvoir transférer les données depuis l'application **TTN** vers une base de donnée MySQL. Depuis cette base de données, deux autres scripts implémenté durant ce projet permette de transformer ces données en graphique afin d'avoir un rendu visuelle des différents cycle du puits sur lequel notre prototype est installé.

4.4 Conclusion

Concernant les **coûts du prototype pré-industriel**, le total pour une fabrication est d'environ 122 CHF. Si ce système est produit en plus grande quantité, par exemple 100 pièces, le coût par unité serait de 58 CHF. Ce prix est acceptable, mais il serait tout à fait possible d'économiser 20 francs en mettant des piles rechargeables AA et une antenne différente.

La source des coûts mentionnés ci-dessus se trouve en annexe [B](#).

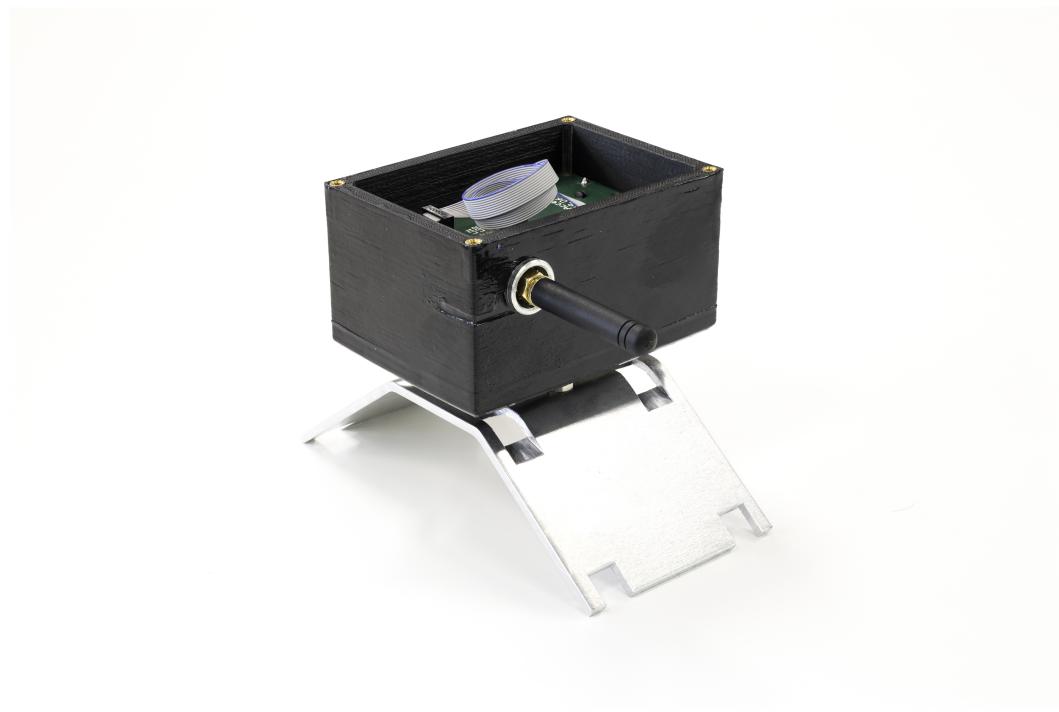


Figure 4.13 Montage final avec le nouveau PCB (sans couvercle).

5 | Tests et validation

Les tests et les mesures sont, comme mentionné dans le design, une étape clé du projet. Comme vu dans le chapitre 2, d'autres prototypes ont été réalisés dans des domaines similaires. La question à laquelle ce chapitre répond n'est donc pas si on arrive à implémenter un système capable de mesurer le volume d'eau soutiré à la nappe phréatique, mais bien si les deux systèmes conçus sont fonctionnels en laboratoire comme sur le terrain.

Pour rappel, le système embarqué final doit :

- être autonome.
- mesurer le temps durant lequel un flux d'eau passe à travers une conduite.
- transmettre les informations nécessaires en utilisant le protocole LoRa.
- résister aux conditions extérieures.

Excepté la résistance aux conditions extérieures, ces points vont être testés dans ce chapitre.

Sommaire

5.1	Tests en laboratoire	40
5.1.1	Avec le prototype n°1	41
5.1.2	Avec le prototype pré-industriel	45
5.2	Tests de consommation	46
5.3	Tests de portée	51
5.4	Tests sur le terrain	52
5.5	Résultats	56
5.6	Discussion	57

5.1 Tests en laboratoire

L'objectif des tests en laboratoire d'hydraulique, qui se situe au sein de la HEI à Sion, est de voir si, dans des conditions idéales, le prototype est fonctionnel. Les avantages du laboratoire sont les suivants :

- Le débit d'eau dans les conduites est modifiable, les tests peuvent donc se faire avec des débits différents.
- Des conduites de tailles différentes sont à disposition, il est donc possible de voir si le système fonctionne sur plusieurs tailles de conduites.
- Les pompes sont électriques, ce qui fait que les vibrations générées par celles-ci sont plus faibles que celles générées par une pompe thermique. L'avantage est donc que si on détecte le passage de l'eau en laboratoire, dans lequel les vibrations sont plus faibles, on devrait aussi pouvoir le détecter sur le terrain.
- On peut facilement enclencher et déclencher les pompes. On peut donc faire des cycles assez courts afin de voir rapidement si le système est fonctionnel.

Le banc de test utilisé, figure 5.1, comprends donc une pompe électrique (rouge), deux vannes pour régler le débit (orange), deux débitmètres (bleu) raccordés sur les conduites qui sortent directement de la pompe, et finalement, une conduite opposée à la pompe (rose) sur laquelle les vibrations dues au moteur devraient être plus faibles.



Figure 5.1 Installation complète utilisée pour les tests en laboratoire.

5.1.1 Avec le prototype n°1

Pour ces tests en laboratoire, effectués le 14 juin 2023, la programmation de la communication LoRa n'ayant pas encore été implémentée. Dans tous les cas, afin d'être sûr des valeurs détectées par le système, celles-ci sont transmises par câble en UART (115200 Bauds). Un programme en Python, en annexe C, a été réalisé, afin de pouvoir récupérer les valeurs sur le port Serial d'un PC. Ce programme affiche également deux graphiques une fois que le nombre de valeurs voulues a été lue. Du côté du système réalisé à l'aide des kits de développement, deux valeurs sont envoyées sur l'UART : La norme des accélérations et une valeur qui indique si de l'eau passe à travers la conduite ou non (0 = pas d'eau, 1 = eau).

Les deux objectifs principaux de ce test sont de **voir si on capte des vibrations utilisables** selon le passage ou non de l'eau dans la conduite et de **voir si la détection des cycles d'allumages se fait correctement**. Si le deuxième objectif est atteint, cela veut dire que le premier est atteint aussi. Ensuite ce test devrait aussi permettre de répondre aux deux questions suivantes :

- Quel est l'impact de la pompe sur les vibrations captées ?
- Est-ce que le système arrive à détecter le passage de l'eau, même quand les vibrations générées par la pompes sont très faibles ?

8 tests ont été effectués :

1. Pompe éteinte, vanne fermée, côté pompe
2. Pompe allumée, vanne fermée, côté pompe
3. Pompe allumée, vanne ouverte, côté pompe
4. Vanne fermée, 3 cycles d'allumage de la pompe, côté pompe
5. Vanne ouverte, 3 cycles d'allumage de la pompe, côté pompe
6. Pompe allumée, vanne fermée, côté opposé à la pompe
7. Pompe allumée, vanne ouverte, côté opposé à la pompe
8. Vanne ouverte, 4 cycles d'allumage de la pompe, côté opposé à la pompe

Tous les tests mentionnés ci-dessus sont réussis. Les valeurs trouvées correspondent toujours aux attentes. Les tests 6, 7, et 8 représentent parfaitement les capacités du prototype n°1.

Chapitre 5. Tests et validation

La figure 5.2 montre les résultats du test 7 dans lequel on a simplement ouvert la vanne pour obtenir un débit de 4000 litres/heures et allumé la pompe. La mesure a été prise du côté opposé à la pompe. Le résultat de cette expérience est que le système détecte bel et bien le fait que l'eau passe à travers la conduite (et que la pompe est allumée). Cette détection se fait avec un seuillage sur les valeurs d'accélération.

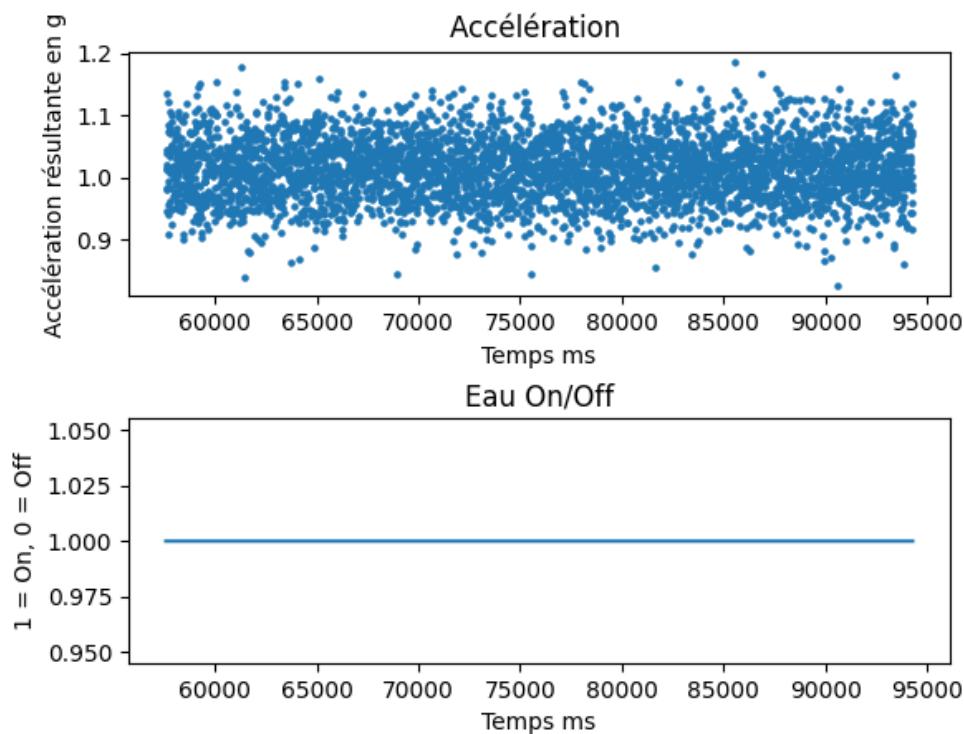


Figure 5.2 Graphique du test 7

5.1 Tests en laboratoire

En dehors du fait que ces premiers tests en laboratoire sont globalement réussis, la figure 5.3, montre les capacités du système. C'est-à-dire que lors de ce test durant lequel on a allumé la pompe mais fermé la vanne, la mesure qui a été prise du côté opposé de la pompe est instable. Ce résultat peut paraître négatif, mais si les tests 6 et 7 ,qui ont pour seule différence d'avoir ouvert la vanne, sont comparés, on voit que l'on détecte vraiment l'eau et pas seulement les vibrations dues à la pompe. Cette conclusion peut être faite car on voit sur la figure 5.3 (vanne fermée) que le système capte des vibrations, mais pas assez pour être stable. Par contre, avec la vanne ouverte, figure 5.2, on capte toujours que de l'eau passe à travers la conduite.

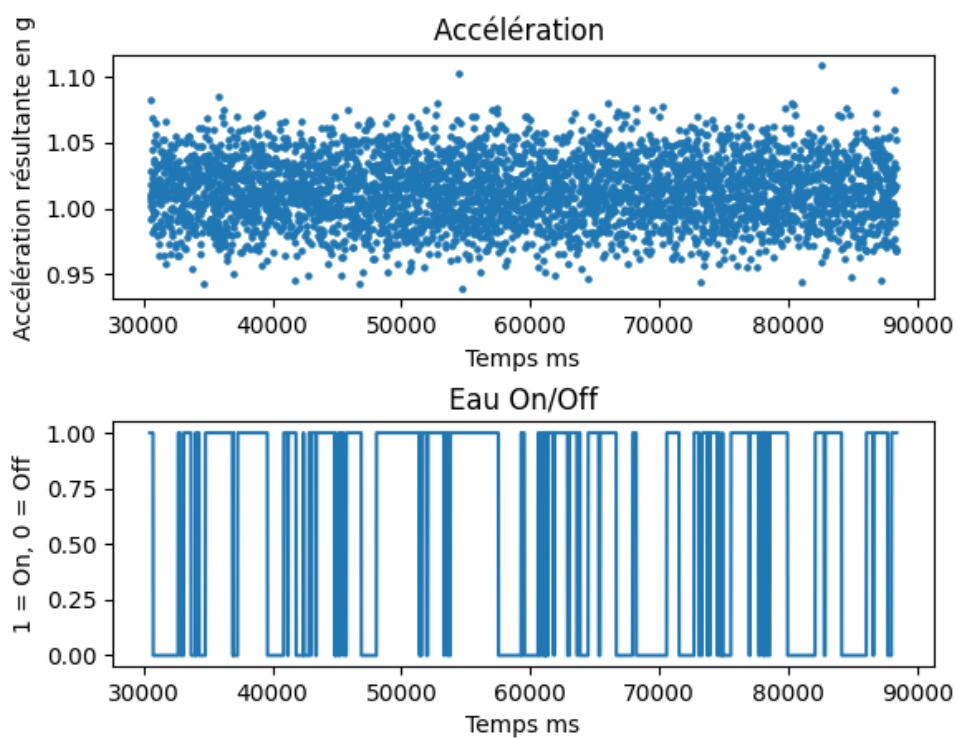


Figure 5.3 Graphique du test 6

Chapitre 5. Tests et validation

Finalement, le test n°8, figure 5.4, résume parfaitement cette série de test. On peut voir, sur le graphique des accélérations, 4 cycles d'enclenchement de la pompe avec les vannes ouvertes. Le débit est réglé à 4000 litres/heure. Ces valeurs ont, comme dit précédemment, été prises sur la conduite du côté opposé à la pompe. Le graphique qui montre si de l'eau traverse la conduite est parfaitement en phase avec celui des accélérations, et donc avec les cycles d'allumages qui durent environ 10 secondes (10 secondes allumé, 10 secondes déclenché).

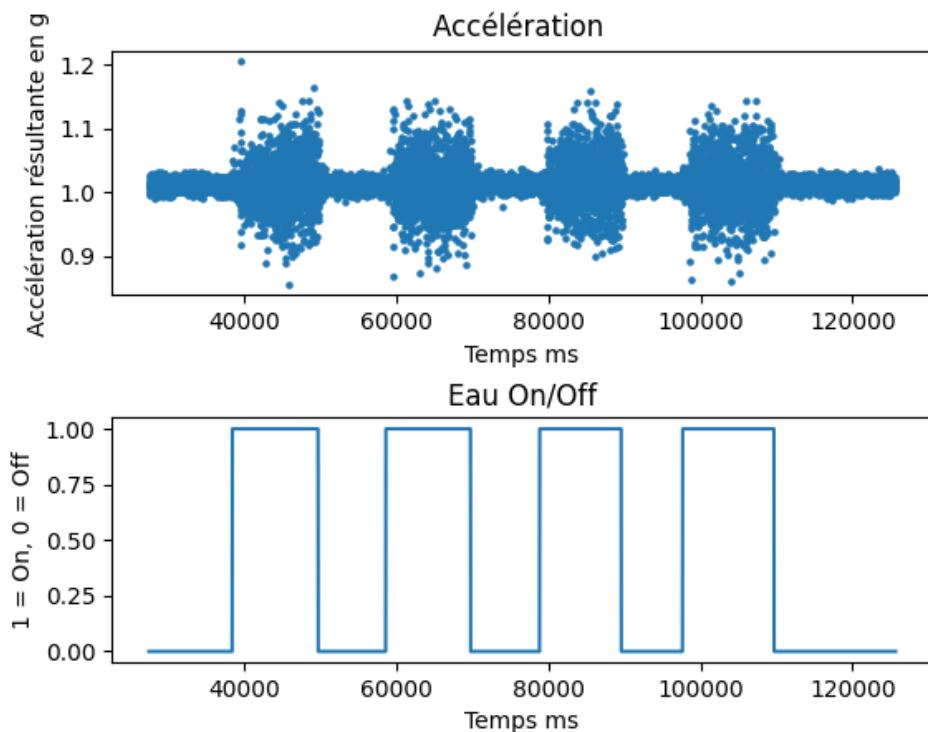


Figure 5.4 Graphique du test 8

Suite à ces tests, la conclusion est la suivante : le prototype n°1 fonctionne parfaitement en laboratoire, excepté que pour ces tests la partie communication LoRa n'a pas encore été implémentée. Tous les graphiques concernant ces tests, y compris ceux qui sont montrés ci-dessus, se trouvent dans l'annexe H.

5.1.2 Avec le prototype pré-industriel

Ces tests en laboratoires avec le prototype final ont été réalisé après les tests sur le terrain présentés plus tard dans ce chapitre. Ceci est dû au fait que le prototype pré-industriel n'a pas pu être installé pendant plusieurs jours, et qu'il n'a donc pas pris énormément de mesures.

Le système a été installé exactement au même endroit que pour les tests avec le prototype n°1.

L'antenne relais a été installée à environ 25 mètres du banc de tests et la communication [LoRa](#) a été testée avant de commencer les différents cycles. Cela a permis de pouvoir faire ensuite ces tests sans interruptions, sur batteries, non-câblé, et pendant plusieurs heures.

L'objectif de ce test est de répondre à la question suivante : Est-ce qu'en laboratoire, le système est totalement fonctionnel ? Pour y répondre, 6 cycles différents ont été effectués avec la pompe.

Cycles	Heure de début	Heure de fin	Temps écoulé
Cycle 1	09 :40	09 :50	10'
Cycle 2	10 :15	10 :20	5'
Cycle 3	10 :30	11 :00	30'
Cycle 4	11 :30	11 :48	18'
Cycle 5	13 :00	13 :20	20'
Cycle 6	14 :15	15 :00	45'

Table 5.1 Cycles effectués par la pompe du banc de test.

Le banc de test a été réglé sur un débit de 3000 litres par heures lorsque la pompe est allumée. Le calcul qui a donc été fait pour obtenir les valeurs de la figure 5.5 ci-dessous est donc $L_c = t \bullet 3000/3600$ avec t en secondes et L_c en litres.

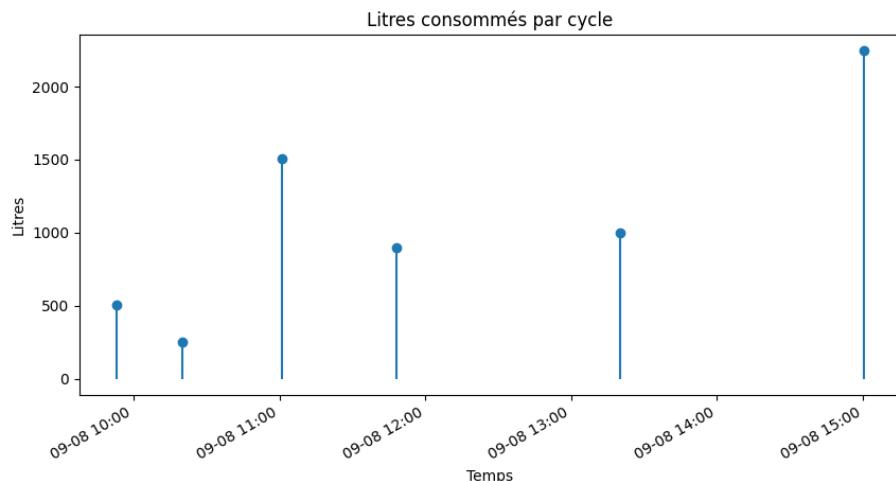


Figure 5.5 Litres consommés à chaque cycle d'utilisation.

La figure 5.5 montre que le système a parfaitement fonctionné. Il a capté les passages de l'eau, transmis le nombre de secondes via LoRa. Pendant ce temps, le script python E a permis de transférer les valeurs sur la base de données mxSQL. Et finalement, à l'aide des deux scripts python, F et G, ces graphiques on pu être réalisés.

Finalement, la figure H.8 montre la consommation totale d'eau qui aurait été utilisée si ce banc de tests n'avait pas été en boucle fermée¹.

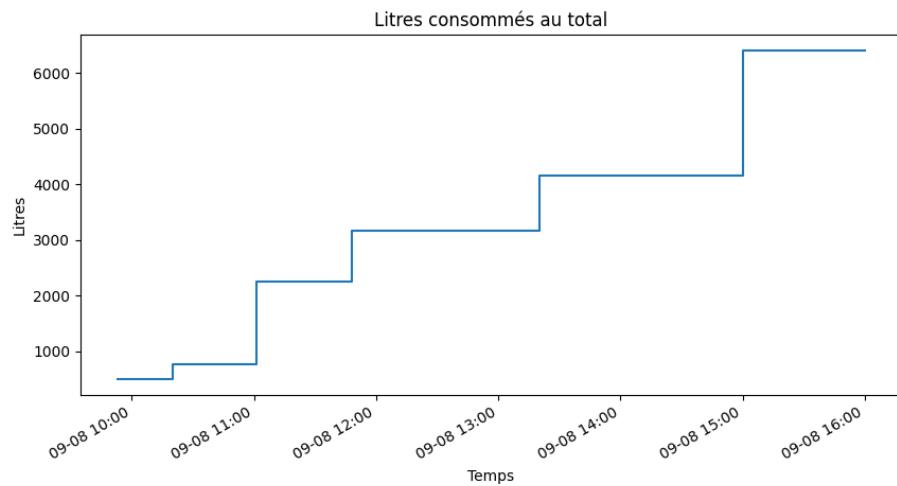


Figure 5.6 Total des litres consommés en fonction du temps.

5.2 Tests de consommation

Un des éléments importants pour le prototype pré-industriel est qu'il soit autonome pendant plusieurs mois. Il est donc fondamental de savoir combien le système consomme. À partir de cette information, on peut en déduire le temps que vont mettre les batteries à se décharger.

Les premiers tests de consommation ont indiqué 25 mA, mais ceux-ci étaient non-significatifs car une des LEDs était encore allumée, ce qui n'as pas besoin d'être le cas lorsque le système est en fonction. Les LEDs sur le nouveau PCB ont juste pu être désactivées dans le code. Celle sur le kit contenant l'accéléromètre a dû être mise hors fonction en coupant la piste, comme indiqué sur la datasheet du devKit [25].

Une fois cette étape a été réalisée, une mesure de courant a été réalisée.

Sur la figure 5.7, une mesure de courant a été effectuée sur 8 secondes. Durant cette période, le système a été au repos au début et à la fin, c'est à dire en mode sleep. Ensuite le système a été réveillé par l'accéléromètre et les mesures d'accélération ont été effectuées. Et pour finir, à partir du premier pic de courant à 50 mA jusqu'au retour au repos, il s'agit de la transmission LoRa.

1. Dans ce cas, un système en boucle fermée veut dire que l'eau d'un réservoir est pompée à travers des conduites et est renvoyée dans le même réservoir

5.2 Tests de consommation

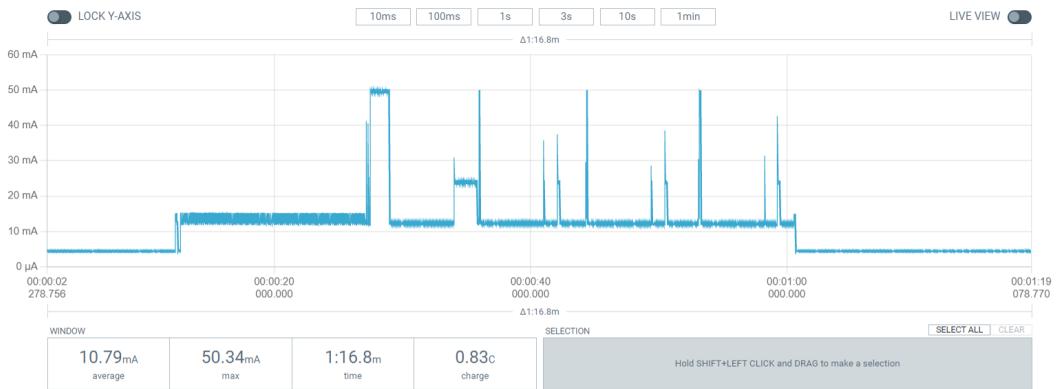


Figure 5.7 Mesure de courant sleep-I2C-LoRa.

En sachant que le système va être réveillé entre 2 et 12h par jour, nous savons qu'il sera au repos pendant 75% du temps en moyenne. C'est donc sur le système au repos que les recherches ont continué.

Chapitre 5. Tests et validation



Figure 5.8 Mesure de courant au repos.

La mesure de courant, figure 5.8, au repos, plus exactement en STOP MODE, montre une consommation moyenne de 2.29 mA.

Le total théorique devrait pourtant être le suivant :

Composant	Condition	Consommation
Régulateur de tension	$0 \text{ mA} < I_{Out} < 500 \text{ mA}$	$200 \mu\text{A}$ max.
Accéléromètre	Mode normal	$14.5 \mu\text{A}$
STM32L072	Stop mode	$0.43 \mu\text{A}$ min.
SX1276	Mode Idle	$1.5 \mu\text{A}$
Total		$217 \mu\text{A}$ max.

Table 5.2 Consommation théorique des composants

On sait donc qu'on a une marge de progression d'un facteur 10 au minimum. Il faut donc trouver d'où vient cette surconsommation de courant.

La première hypothèse a été que le régulateur de tension consomme trop pour une quelconque raison. Le système a donc été alimenté directement en 3.3V sur la broche de test. Cette broche se situe donc entre le régulateur et les composants alimentés en 3.3V. Cette alimentation différente n'a provoqué aucun changement significatif : On se trouve toujours entre 2.2 et 2.4 mA.

5.2 Tests de consommation

Ensuite une mesure de courant a été effectuée, toujours avec l' μ C en stop mode, avec l'accéléromètre débranché. Cette mesure permet d'isoler un des deux composants : L'accéléromètre ou l' μ C avec la transmission LoRa. La valeur moyenne de cette mesure est de 2.18 mA. On peut affirmer avec certitude que la surconsommation vient du micro-contrôleur. La figure 5.9 montre que le développement jusqu'ici est correct, vu que le plus grand dissipateur de chaleur est bien le processeur Murata.

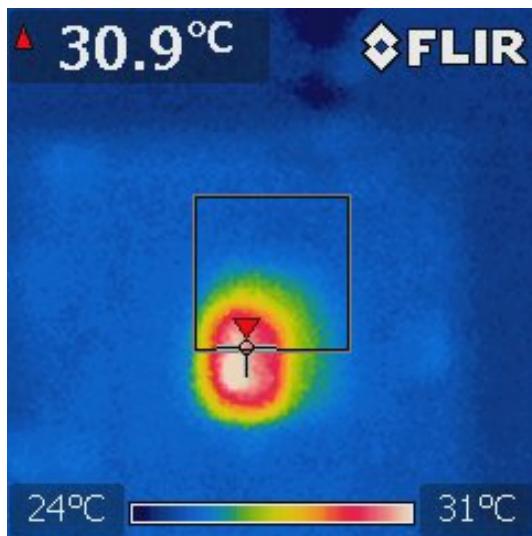


Figure 5.9 Image thermique du PCB.

Le but, à ce niveau de la recherche, est d'abord de savoir si la surconsommation vient de la partie transmission du chip ou du STM32. Mais d'abord, une autre mesure a été effectuée en chargeant un code vide sur le chip. Désactiver les LEDs et passer en stop mode sont les seules actions réalisées dans ce code. Le résultat, sur la figure 5.10, montre une moyenne à 2.81 mA, ce qui est une augmentation par rapport au code utilisé de base. Cette augmentation peut être due à l'attribution des pin, ou au code IT-SDK qui englobe le projet de base et qui est fait pour consommer le moins de courant possible.

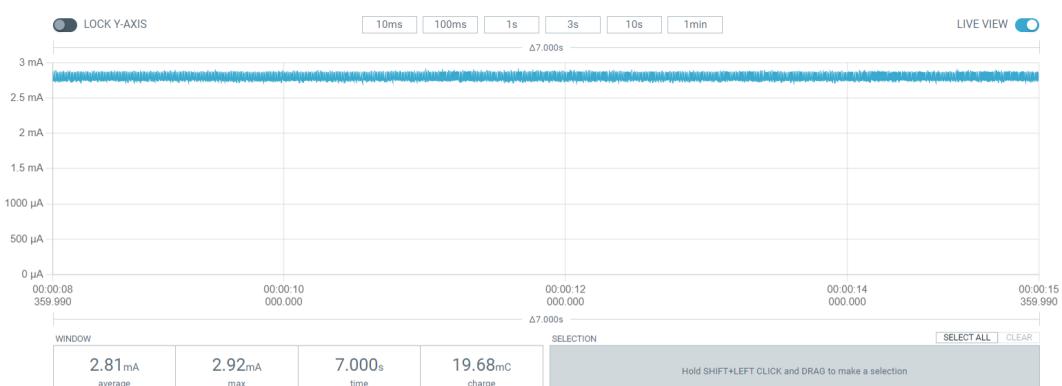


Figure 5.10 Image thermique du PCB.

Chapitre 5. Tests et validation

En repassant à travers les datasheet des composants présents à l'intérieur du chip Murata, c'est à dire le STM32 et le SX1276, un élément est frappant : Lorsque le module LoRa est en "Idle mode", il doit consommer $1.5 \mu A$, mais lorsqu'il est en "Standby Mode" il consomme **1.8 mA**. Le module SX1276 a donc été mis en "Idle mode" manuellement à l'aide du code fourni par le SDK utilisé : pas de changement au niveau de la consommation.

Afin d'écartez cette option, une lecture du mode du module de transmission a été implémentée dans le code, mais celle-ci indique bien que nous nous trouvons en "Idle mode". SX1276GetStatus() est la fonction utilisée pour cette vérification.

Finalement le dernier essai a été de charger le même code vierge, qui a été utilisé précédemment, dans le kit de développement. Nous obtenons avec cela une consommation de 3.3 mA . Ce qui représente 0.5 mA de plus que sur la même mesure, avec les mêmes conditions, sur le nouveau PCB.

La conclusion de cette dernière mesure indique au moins deux sources plausibles à cette surconsommation :

1. Le code contient un élément qui consomme énormément. Par exemple des pin, des clocks, ou des fonctionnalités qui seraient mal configurées.
2. Sur le schéma électronique du kit de développement, différents ponts sont installés ou non, des diodes et des résistances peuvent être enlevées ou ajoutées. Il est donc possible que ces différents éléments ne soient pas configurés de la bonne manière. Ce qui impact directement le schéma du PCB du prototype pré-industriel étant donné qu'il en a largement été inspiré.

Il est important de se rappeler de la consommation actuelle du système : **2.29 mA**, soit au minimum **10 fois plus**. Concernant les deux pistes mentionnées ci-dessus, des questions, actuellement en suspens, se posent :

- Si le code est la source du problème, qu'est-ce qui pourrait provoquer une telle consommation en sachant qu'en "Stop mode", la consommation devrait être 10 à 100 fois plus basse ?
- Si le **PCB** ou/et le schéma électronique contient une erreur, qu'est-ce qui pourrait provoquer une consommation de 2.29 mA ? Car dans le cas où une pin serait par exemple connectée au $0V$, le courant serait plus proche de 16 mA qui est le courant maximal selon la datasheet du STM32L072 [28].

Pour conclure ce chapitre, la source de la surconsommation la plus probable est tout de même qu'il y ait un problème électronique. Étant donné que plusieurs codes différents ont été chargés, que ce soit le code final ou le code vierge par exemple, et que la consommation ne variait pas énormément pour ces différents codes, on peut donc estimer que le problème est plus probablement externe au code.

Cependant, même si cette consommation est trop élevée, en prenant en compte que nous faisons des mesures pendant 12h maximum par jour et que le reste du temps nous sommes en mode stop, le système consomme 7.6 mA en moyenne. Ce qui veut dire que le système pourrait tout de même être autonome pendant environ 28 jours.

5.3 Tests de portée

Dès le début, l'objectif a été de transmettre les données vers [TTN](#) par le système de communication nommé [LoRa](#). La question de quel moyen de communication utiliser ne s'est donc jamais posée durant ce projet. Mais est-ce que [LoRa](#) est la bonne solution ?

Tout d'abord, le but de cette communication sans fil est de transmettre entre une et cinq fois une dizaine de bytes sur un réseau. [LoRa](#) est idéal pour ces faibles débits et pour l'[IOT](#) en général.

Nous savons que, théoriquement, la portée de la transmission [LoRa](#) se trouve entre 5 et 20km, selon les obstacles qui se trouvent entre l'objet connecté et l'antenne relais [29].

Quelle portée atteignons nous donc avec notre prototype pré-industriel ? Pour répondre à cette question, 3 tests ont été effectués : À l'intérieur du bâtiment 23 de la HES-SO à Sion, entre l'intérieur et l'extérieur du bâtiment (le gateway à l'intérieur), et à l'extérieur sur le domaine agricole de Châteauneuf.

Les résultats sont les suivants :

1. À l'intérieur, la portée est très faible, entre 20 et 30m.
2. À travers une vitre et en s'éloignant à l'extérieur en ligne directe (sans obstacle), la portée est de 80 à 120m.
3. À l'extérieur uniquement, en ligne directe, la portée est de 500m maximum.

Ces tests montrent clairement les portées prédites par les utilisateurs du système de communication [LoRa](#) ne sont pas atteintes.

Ceci peut être dû à une mauvaise configuration de l'envoie [LoRa](#), un problème électro-nique sur le [PCB](#), ou un problème de l'autre côté de la transmission, c'est-à-dire côté gateway.

Le chapitre suivant aborde entres autres cette dernière option.

5.4 Tests sur le terrain

Les essais en conditions réelles comprennent toujours des surprises. Un des objectifs est donc d'anticiper au mieux les différents facteurs liés à l'environnement, ou au réseau par exemple.

Dans notre cas, étant donné les premiers tests de portée durant lesquels les objectifs n'ont pas été atteints, une solution de secours a été implémentée. C'est-à-dire qu'à la place de récupérer la durée d'enclenchement par LoRa et TTN, on les récupère directement en câblé via l'UART. Une fois les données récupérées, on les envoie également sur la base de données MySQL. Le code en Python de cette solution de secours se trouve dans l'annexe D.

Pour commencer, ces tests se sont déroulés à Châteauneuf, plus exactement dans les cultures en bout de piste de l'aérodrome de Sion. Ce domaine agricole possède 4 puits différents, et toutes les conduites et les pompes sont souterraines. Ce qui pose malheureusement des problèmes en plus au niveau de la transmission LoRa. Selon le chef des cultures, Laurent Clerc, de nombreux puits sont souterrains en Valais. Cette information est à été un peu surprenante, car la différence est importante par rapport aux images présentées dans l'introduction (figures 1.1). Cependant, c'est un excellent test pour le système, pour deux raisons :

1. Les vibrations sont plus faibles lorsque les conduites sont enterrées.
2. La transmission doit être parfaite afin de pouvoir envoyer quelque chose à travers le sol.

Étant donné les problèmes de portées présent avant ces tests, la décision a été prise de ne pas connecter l'antenne relais LoRa. En sachant que le local dans lequel l'antenne aurait pu être installée se trouve à 800m du puits en question, il paraissait impossible qu'une communication s'établisse.

Étonnement, une fois que le capteur a été mis sous tension, en dehors du puits, des données ont été reçues sur TTN. Ce qui veut dire que le système a réussi à ce connecter à l'application via une autre antenne relais de la région. Aucune conclusion ne peut être tirée par rapport à la portée du système, mais selon la carte des antennes relais², l'antenne relais la plus proche se trouve à 700m du point de test. Par contre, cet information remet en doute l'efficacité de l'antenne relais utilisée pour ce projet.

2. TTN Mapper : <https://ttnmapper.org/heatmap/>

Après la mise sous tension, le système a été installé sur la conduite dans le puits d'accès, comme montré sur la figure 5.11. À partir de ce moment-là, la transmission LoRa n'a plus été possible. Le système a donc, comme prévu, été câblé pour récupérer les données par [UART](#).



Figure 5.11 Prototype pré-industriel installer sur une conduite souterraine.

En installant le système, la pompe était en fonctionnement. Ensuite, la pompe a suivi les phases suivantes :

- 11h00 : extinction.
- 12h00 : allumage.
- 12h30 : extinction.

Les phases d'allumage ont été parfaitement détectées par le système. En sachant, suite au informations de Steeve Maillard, que le débit nominal de ce puits est de 2000 litres par minutes, un graphique de consommation 5.12 a pu être réalisé. Pour une question d'esthétique, ces graphiques ont été faits sur Excel avec les données de MySQL.

Sur le graphique ci-dessus, on observe trois colonnes différentes. La première et la troisième sont la preuve que le système fonctionne, étant donné qu'il a capté le passage de l'eau, envoyé le nombre de secondes d'enclenchement via l'UART et le nombre de seconde a pu être enregistré dans la base de données. Cependant, à 11h35, un passage de l'eau a été détecté. En effet, à 11h35 deux avions de chasse de l'armée suisse ont effectué un touch-and-go ce qui a généré énormément de volume sonore, et donc des vibrations dans le sol.

C'est une information amusante, mais pas uniquement. Cela montre que le seuillage effectué par le système est trop sensible.

Chapitre 5. Tests et validation

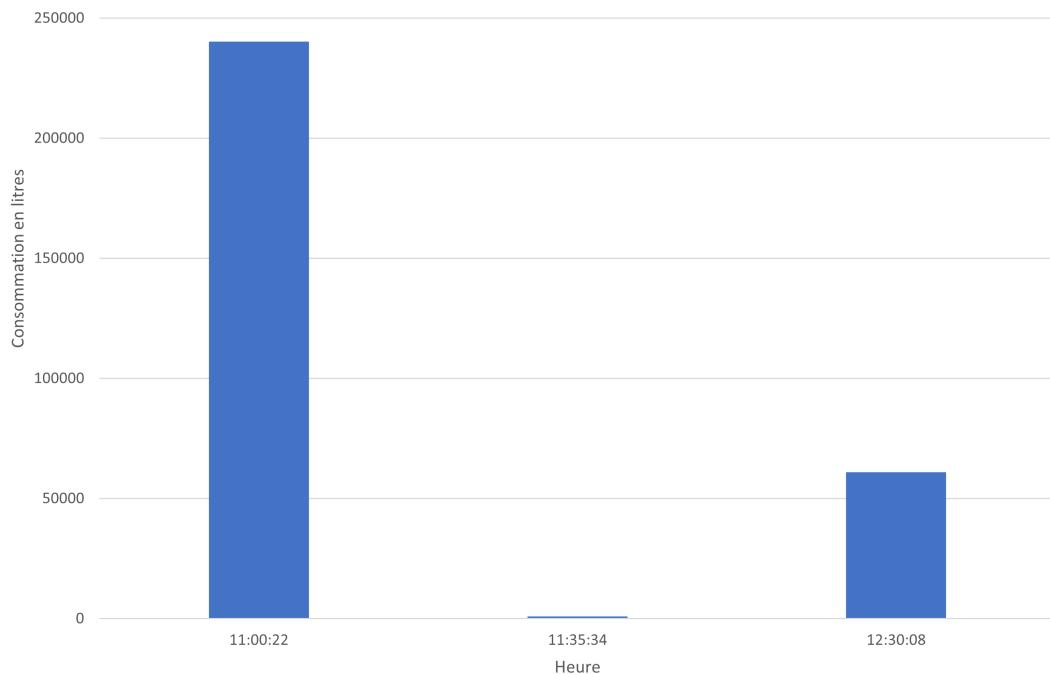


Figure 5.12 Mesures prises sur le terrain : litres consommés par cycles.

Dans le cas, où ce système serait installé pendant une plus longue durée, il sera donc nécessaire d'affiner le seuillage afin de détecter le passage de l'eau uniquement, et non pas les perturbations qu'il peut y avoir. Évidemment, les passages d'avions de ce type ne sont pas significatifs vu qu'il sont très courts, mais il est possible que le système capte aussi la pluie sur la conduite par exemple. Ceci engendrerait une plus grande erreur sur les résultats de mesures.

Par contre, suite à ces tests sur pompes électriques, nous pouvons partir du principe que la pluie ne va pas générer plus de vibrations que l'allumage d'une pompe, quelle qu'elle soit. Ceci est dû au fait que ces pompes génèrent en fait énormément de vibrations dans les conduites.

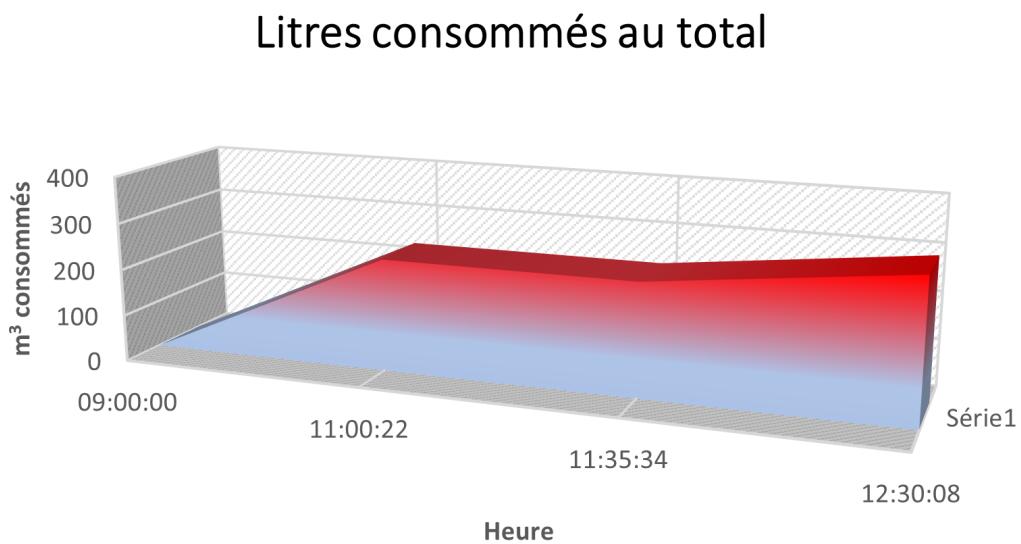


Figure 5.13 Mesures prises sur le terrain : litres consommés au total.

Pour conclure ces tests, le graphique de la figure 5.13 montre un aperçu de ce qui a été consommé au total pendant le temps des tests sur le terrain. On retrouve cette fois-ci des m^3 et non pas des litres ($1000l = 1m^3$).

5.5 Résultats

Les résultats de ces différents tests sont positifs. Le système fonctionne parfaitement en laboratoire. Cependant, certains éléments sont à améliorer ou clarifier concernant le terrain.

Premièrement, la portée du système n'est actuellement pas exactement déterminée. Mais elle devient surtout insuffisante lorsqu'on met le système sur des conduites souterraines.

Ensuite, la consommation du système est trop élevée. Après avoir fait les différents tests de consommation, seul le chip Murata peut être la source de cette surconsommation. Le code ou l'électronique autour du chip sont les deux variantes possibles à ce courant trop élevé.

Le résultat de cette consommation est que le système peut être autonome pendant environ 28 jours. Ce qui n'est effectivement pas suffisant pour ce système qui devrait plutôt pouvoir être autonome pendant 6 mois à une année. Cette durée de vie lui permettrait de rester sur le terrain pendant toute la période d'été ou toute la période d'hiver sans qu'il n'y ait à se préoccuper de ces batteries.

Le système a donc besoin de certaines améliorations. Cependant, au niveau de la prise de mesure, nous pouvons dire avec certitude que la mesure d'accélération convient parfaitement à la détection du passage de l'eau. Même sur des systèmes générants peu de vibrations, comme en laboratoire, les deux prototypes sont capables de détecter les passages de l'eau dans les conduites. La méthode utilisée, le seuillage, permet cette détection.

Le seul élément à configurer selon les différentes conduites est la valeur du seuil. Cette valeur doit être réglée afin d'éviter la détection de perturbations qui peuvent être présentes sur le terrain.

Nous allons conclure ce chapitre sur un autre point positif : le design mécanique. Celui-ci permet de transmettre les vibrations, peut s'adapter à différentes conduites, et le boîtier a été réalisé afin qu'il puisse être étanche.

5.6 Discussion

Les résultats mentionnés ci-dessus montrent que le système peut être utilisé pour la détection de l'eau dans les conduites de puits agricoles. Les éléments qui sont encore à améliorer sont faisables.

Nous pouvons donc dire que les différents choix tout du long du projet n'ont peut-être pas été parfaits, mais ils ont permis d'implémenter un système prouvant qu'il est possible de répondre à l'objectif du projet : **réaliser un système permettant de déterminer quand est-ce que de l'eau traverse une conduite, pendant combien de temps, et d'en calculer le volume d'eau soutiré à la nappe phréatique.**

D'autres méthodes auraient pu être utilisées, comme l'utilisation d'un microphone ou d'un capteur piézo-électrique. Cependant, l'accéléromètre paraît être moins sensible aux perturbations extérieures qu'un microphone par exemple. Dans un milieu comme l'agriculture, dans lequel il y a beaucoup de bruit lié aux différents véhicules, les valeurs mesurées par un microphone par exemple auraient nécessité plus de traitement afin d'arriver à un résultat aussi stable que celui qui a été réalisé dans ce projet.

Il a également été prouvé que la réalisation d'un dashboard est totalement accessible. Un exemple en a été fait dans ce projet, dans lequel nous mettons les données dans une base de données depuis laquelle des graphiques peuvent être réalisés.

6 | Conclusions

6.1 Résumé du projet

Le mot-clé de ce résumé est **polyvalence**. Le système réalisé a demandé des connaissances en programmation, en électronique, et en mécanique.

En laboratoire, le prototype pré-industriel répond aux attentes. Premièrement, il mesure le nombre de secondes pendant lesquelles de l'eau passe à travers la conduite. Ensuite il envoie ce nombre de secondes via le protocole [LoRa](#). Une fois que les données sont reçues dans l'application [TTN](#), elles sont traitées par un script python qui les enregistre dans une base de donnée mySQL. Et pour finir, deux scripts python permettent d'afficher ces données sous forme de graphique.

Sur le terrain, l'élément qui a posé problème est la transmission [LoRa](#). Étant donné que les tests sur le terrain ont été réalisés sur des puits en dessous du niveau du sol, la transmission a été bloquée. Cependant, en dehors du puits, la transmission fonctionnait. Ce qui veut dire que le test est à refaire en installant ce système sur un puits semblable à ceux de la figure 1.1 mentionnés dans l'introduction.

En dehors de ce problème de communication, un autre point est à résoudre pour que le système soit fonctionnel sur le terrain : la consommation.

La consommation actuelle étant trop élevée, une solution doit être trouvée. Nous savons actuellement que cette surconsommation provient du chip Murata, mais il est difficile de dire si l'erreur vient de l'électronique ou du code.

Finalement, le point important est que, même sur le terrain, la prise de mesure avec l'accéléromètre fonctionne. Ce qui veut dire que ce système est une solution à la problématique qui était : **comment serait-il possible de quantifier le volume d'eau passant à travers les conduites de ces puits agricoles ?**

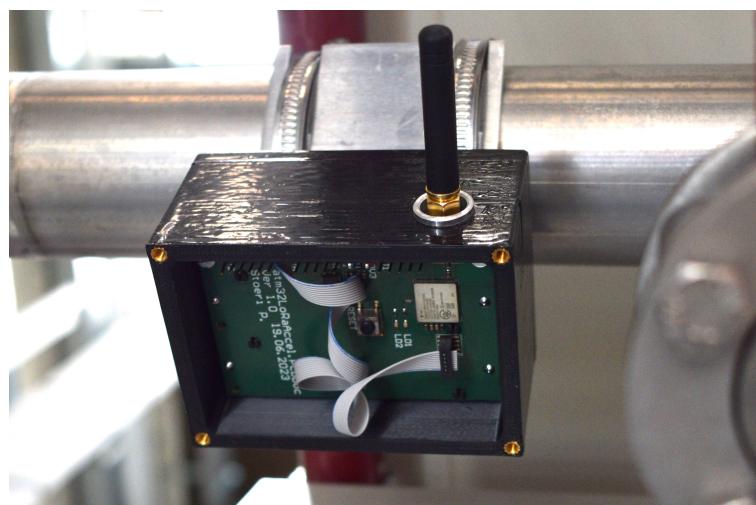


Figure 6.1 Prototype final en laboratoire.

6.2 Comparaison avec les objectifs initiaux

Les objectifs initiaux du système implémenté durant ce projet étaient :

- Le système doit être autonome, donc sur piles ou batteries.
- Il doit être capable de détecter le passage de l'eau dans une conduite de puits agricole et de mesurer la durée du passage de l'eau.
- Il doit pouvoir envoyer les données nécessaires via le protocole de communication [LoRa](#).
- Son design mécanique doit être conçu pour être résistant aux conditions du terrain et il doit permettre au système d'être fixé sur différentes tailles de conduites.
- Il doit être bon marché.
- Un dashboard doit être implémenté afin de mettre les données sous forme de graphique ou de carte.

Sur ces six objectifs, trois sont parfaitement atteints. La détection est fonctionnelle, le design mécanique remplit les conditions fixées, et le dashboard montre qu'il est possible de récupérer les valeurs et de les mettre sous forme de graphique.

Les trois autres objectifs sont partiellement atteints. Premièrement, le système est autonome, mais pas assez longtemps. Actuellement, le prototype final a une autonomie d'environ 28 jours, alors que nous voudrions une autonomie de six mois jusqu'à une année. Cependant, bien que le problème de surconsommation ne soit pas encore résolu, il paraît totalement possible de diminuer la consommation de ce système d'un facteur 10.

Deuxièmement, la transmission [LoRa](#) ne fonctionne pas encore lorsque le système est placé sur une conduite en souterrain. Il serait intéressant de connaître la réelle portée du système : la raison pour laquelle les tests de portée n'ont pas été concluants est peut-être que le gateway utilisé soit abîmé.

Finalement, le prix par système, dans le cas où nous en produirions 100, est de 58 CHF. Ce prix est trop élevé pour répondre entièrement à l'objectif "bon marché". Cependant, environ 20 CHF peuvent être économisé en mettant des piles rechargeables AA et en changeant l'antenne.

6.3 Difficultés globales rencontrées

La plus grande **difficulté technique** rencontrée durant ce projet a été de trouver une solution viable afin de pouvoir transmettre les informations en [LoRa](#).

Au **niveau organisationnel**, deux difficultés sont ressorties tout au long du projet. Premièrement, l'évaluation du temps consacré aux différentes tâches a été une difficulté, surtout au début du projet. Finalement, tout a pu être réalisé comme sur le planning, excepté pour les tests de longue durée sur le terrain. Cela est principalement dû au fait que l'implémentation de la transmission [LoRa](#) a pris plus de temps que ce qui avait été prévu.

Et deuxièmement, la combinaison de l'électronique, de la mécanique, et de la programmation a aussi été une complexité globale de ce projet. Mais cette combinaison a aussi permis de rester productif la quasi totalité du projet, car lorsqu'une des parties mentionnées était bloquée, il était possible d'avancer sur une tâche totalement différente.

6.4 Améliorations et tâches futures

Pour le prototype final, deux catégories d'améliorations sont envisagées pour le futur : les essentielles et les optionnelles. Car le système est fonctionnel, mais certains points nécessitent des modifications. Le système requiert également des tests supplémentaires qui sont aussi cités dans ce chapitre. Voici les éléments à modifier dans un premier temps :

- La consommation du système doit être réduite. Certaines pistes ont déjà été explorées, mais il y a probablement une erreur dans le système qui empêche de faire une réelle optimisation du système et qui doit être résolue.
- Un test d'étanchéité doit être effectué. La seule tâche à réaliser avant ce test est d'utiliser de la pâte d'étanchéité entre le couvercle et la partie inférieure du boîtier, et d'appliquer de la Loctite 542 sur les filetages.
- Il est aussi nécessaire de refaire des tests de portées et d'essayer de faire fonctionner l'envoie des données également dans les puits souterrains.

Ensuite, les améliorations citées ci-dessous sont optionnelles. Elles permettent d'optimiser au mieux le système actuel afin de faire diminuer le prix ou la consommation de courant par exemple. Évidemment, tout système contient toujours des améliorations possibles, mais celles qui sont citées ci-dessous apporteraient une réelle plus-value à ce travail :

- Au niveau de l'optimisation de la consommation, il serait bien de ne plus mesurer les accélération en continu à l'avenir. Cette manière de faire a été choisie pour l'instant, mais consomme trop de courant pour que la batterie puisse tenir sur le long terme. On pourrait imaginer de se mettre en mode sleep ou en mode stop et de se réveiller toutes les 5 secondes afin de prendre une mesure.
- Le kit de développement contenant l'accéléromètre pourrait être remplacé par un [PCB](#), ou même inclus sur le [PCB](#) contenant le reste des composants. Cette dernière solution devrait être testée afin d'être sûr que les vibrations soient bien transmises. Mais le fait de ne plus avoir un kit pourrait être rentable dans le cas où l'on produirait plus de 50 pièces.
- Les batteries actuelles pourraient être remplacées par des batteries ou des piles moins chères, par exemple des piles rechargeables AA.

6.5 Perspectives futures

La principale perspective pour le future du système implémenté pendant ce projet est d'installer ce système sur plusieurs puits, typiquement durant des périodes de lutte contre le gel ou de sécheresse. On pourrait alors répondre à la question que se pose le CREALP : Quel est l'impact de l'utilisation des puits agricole sur la nappe phréatique dans la plaine du Rhône.

Un graphique comme celui en figure 6.2, dans lequel on montre ce qu'ont consommé 5 puits au total pendant un mois, est un exemple de ce qui pourrait être réalisé.

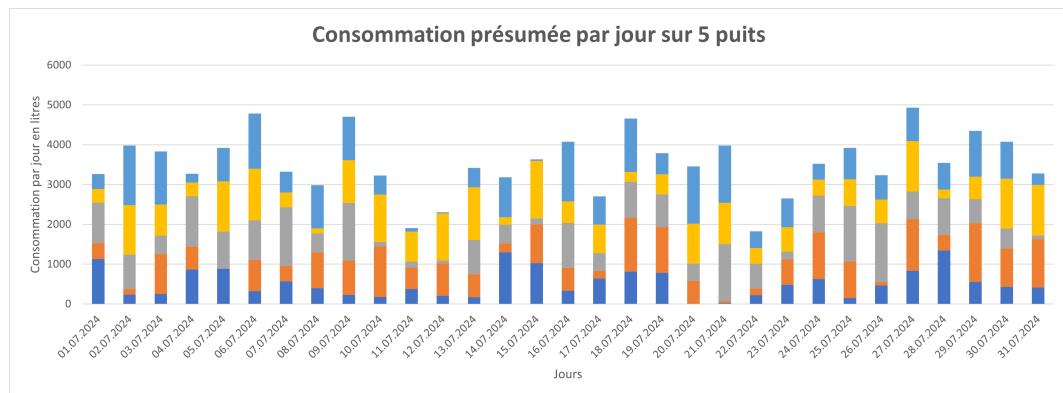


Figure 6.2 Consommation présumée de 5 puits.

Les données pourraient ensuite être mises sous forme de carte où l'on montre la consommation de différents puits dans une région. Un exemple est montré sur la figure 6.3.

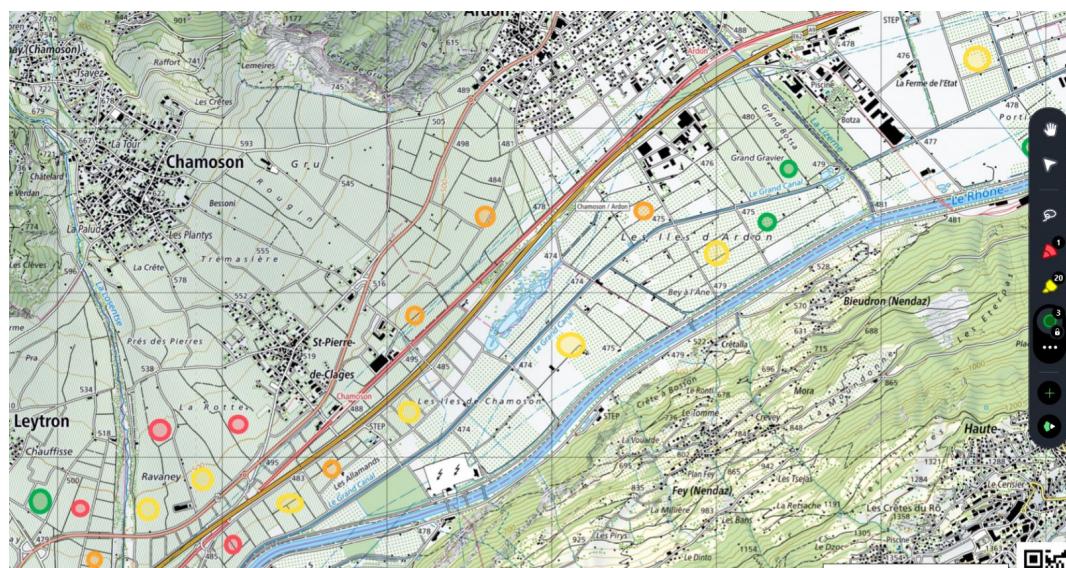


Figure 6.3 Consommation en couleur de différents puits dans une région.

Les valeurs et les couleurs des deux figures ci-dessus sont aléatoires, mais permettent de visualiser les perspectives futures de ce projet.

Chapitre 6. Conclusions

Finalement, le système pourrait être appliqué à d'autres puits agricoles en Suisse, ou dans d'autres pays où l'on retrouve le même type de puits. En fait, un système de ce type peut être appliqué à de nombreux cas dans lesquels le but est de détecter si de l'eau traverse une conduite.

6.6 Conclusion personnelle

Le principal challenge pour moi a été de travailler de manière autonome. Durant nos études, les projets sont principalement en groupe, et les responsabilités sont donc réparties. Expérimenter le fait de porter seul toute la responsabilité était source de grande satisfaction, mais aussi d'incertitudes. Même si j'apprécie cette approche, je n'y étais pas habitué. Finalement, ce que j'ai appris de cette manière de travailler est qu'il est pertinent de solliciter l'avis ou même l'aide de différentes personnes, mais les décisions importantes reviennent à la personne réalisant le projet. Je me suis parfois laissé trop influencé par les différents spécialistes et n'ai pas suffisamment remis en question leurs avis, ce qui m'a conduit à prendre certaines décisions de manière précipitée.

Au niveau technique, j'ai été à l'aise durant la réalisation de la mécanique et lors de la programmation. La partie électronique a été plus ardue pour moi, étant donné que je n'avais jamais véritablement utilisé des outils tels qu'Altium Designer.

Ce qui m'a demandé le plus de temps et de patience durant ce projet est la transmission LoRa. De nombreuses recherches et essais ont été nécessaires.

En fin de compte, ce projet représente, pour moi comme pour beaucoup d'autres, le plus grand accomplissement de mes études (501 heures de travail). Par moments, il a été une source de stress. Mais, dans l'ensemble, ce furent les 14 plus belles semaines de mes études. Les personnes avec qui j'ai pu discuter et collaborer ont été bienveillantes, et la majorité des tâches de ce projet m'ont passionné.

Sion, 18th August 2023
Pablo Stoeri

Bibliographie

- [1] *Transforming our world : the 2030 Agenda for Sustainable Development / Department of Economic and Social Affairs.* URL : <https://sdgs.un.org/2030agenda> (visité le 13/06/2023).
- [2] *17 objectifs de développement durable.* URL : <https://www.eda.admin.ch/agenda2030/fr/home/agenda-2030/die-17-ziele-fuer-eine-nachhaltige-entwicklung.html> (visité le 13/06/2023).
- [3] *Objectif 6 : Garantir l'accès de tous à l'eau et à l'assainissement et assurer une gestion durable des ressources en eau.* URL : <https://www.eda.admin.ch/agenda2030/fr/home/agenda-2030/die-17-ziele-fuer-eine-nachhaltige-entwicklung/ziel-6-verfuegbarkeit-und-nachhaltige-bewirtschaftung-von-wasser.html> (visité le 13/06/2023).
- [4] *Objectif 12 : Établir des modes de consommation et de production durables.* URL : <https://www.eda.admin.ch/agenda2030/fr/home/agenda-2030/die-17-ziele-fuer-eine-nachhaltige-entwicklung/ziel-12-fuer-nachhaltige-konsum-und-produktionsmuster-sorgen.html> (visité le 13/06/2023).
- [5] *Objectif 2 : Éliminer la faim, assurer la sécurité alimentaire, améliorer la nutrition et promouvoir l'agriculture durable.* URL : <https://www.eda.admin.ch/agenda2030/fr/home/agenda-2030/die-17-ziele-fuer-eine-nachhaltige-entwicklung/ziel-2-den-hunger-beenden-ernaehrungssicherheit-und-eine-bessere.html> (visité le 13/06/2023).
- [6] *Théorème d'échantillonnage.* In : Wikipédia. Page Version ID : 205241297. 17 juin 2023. URL : https://fr.wikipedia.org/w/index.php?title=Th%C3%A9or%C3%A8me_d%27chantillonnage&oldid=205241297 (visité le 11/08/2023).
- [7] Pascal Ornstein et CREALP. *Nappe phréatique du Rhône et irrigation. Vers une conciliation des usages...* 15 oct. 2020. (Visité le 16/05/2023).
- [8] *Débit (physique).* In : Wikipédia. Page Version ID : 198625228. 13 nov. 2022. url : [https://fr.wikipedia.org/w/index.php?title=D%C3%A9bit_\(physique\)&oldid=198625228](https://fr.wikipedia.org/w/index.php?title=D%C3%A9bit_(physique)&oldid=198625228) (visité le 13/06/2023).
- [9] Erik Trostmann. *Technologie des transmissions à eau.* CRC Press, 13 mars 2019. 192 p. isbn : 0-8247-9784-1. (Visité le 14/06/2023).
- [10] CREALP. *Graphique de test avec un accéléromètre.* 22 oct. 2020. (Visité le 17/05/2023).
- [11] Baptiste Solioz. *Low-Cost sensor for flow rate measurements.* 3 mars 2021. (Visité le 16/05/2023).
- [12] Jérémie Maceiras Ferreiro. *I.A. pour mesure non-intrusive de débit.* 20 sept. 2018. url : <https://sonar.ch/hesso/documents/317467> (visité le 23/05/2023).

Bibliographie

- [13] *Management platform for The Things Network*. The Things Network Console. 30 mai 2023. url : <https://eu1.cloud.thethings.network/console> (visité le 10/07/2023).
- [14] *B-L072Z-LRWAN1 - STM32L0 Discovery kit LoRa, Sigfox, low-power wireless - STMicroelectronics*. url : <https://www.st.com/en/embedded-software/i-cube-lrwan.html> (visité le 30/07/2023).
- [15] ANALOG DEVICES. *Small, Low Power, 3-Axis +-3g, Accelerometer ADXL335*. 2010. url : <https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL335.pdf> (visité le 22/05/2023).
- [16] *Inertial Measurement Unit BMI088*. Bosch Sensortec. url : <https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi088/> (visité le 16/06/2023).
- [17] *Accelerometer BMA400*. Bosch Sensortec. url : <https://www.bosch-sensortec.com/products/motion-sensors/accelerometers/bma400/> (visité le 16/06/2023).
- [18] *SparkFun Triple Axis Accelerometer Breakout - BMA400 (Qwiic) - SEN-21208* - SparkFun Electronics. url : <https://www.sparkfun.com/products/21208> (visité le 16/06/2023).
- [19] <https://community.bosch-sensortec.com/t5/user/viewprofilepage/user-id/11110>. *BMA400 activity detection*. Section : MEMS sensors forum. 24 fév. 2021. url : <https://community.bosch-sensortec.com/t5/MEMS-sensors-forum/BMA400-activity-detection/td-p/21382> (visité le 05/07/2023).
- [20] Bosch Sensortec GmbH. *Datasheet BMA400*. Fév. 2023. url : <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bma400-ds000.pdf>.
- [21] *I-CUBE-LRWAN - LoRaWAN software expansion for STM32Cube (UM2073)* - STMicroelectronics. url : <https://www.st.com/en/embedded-software/i-cube-lrwan.html> (visité le 29/07/2023).
- [22] Paul Pinault. *Disk91 IoT_SDK not only for STM32*. original-date : 2018-11-03T17:43:36Z. 17 mai 2023. url : <https://github.com/disk91/stm32-it-sdk> (visité le 01/08/2023).
- [23] Paul Pinault. *LoRaWan example for MURATA CMWX1ZZABZ using Itsdk*. original-date : 2019-02-05T21:08:38Z. 3 août 2023. url : <https://github.com/disk91/itsdk-example-murata-lora> (visité le 13/08/2023).
- [24] *Kit de l'imprimante 3D Original Prusa i3 MK3S+ / Imprimantes 3D Original Prusa par Joseph Prusa directement*. Prusa3D by Josef Prusa. url : <https://www.prusa3d.com/fr/produit/kit-de-l-imprimante-3d-original-prusa-i3-mk3s/> (visité le 23/07/2023).
- [25] Dryw Wade. *Schématique BMA400 Sparkfun*. 28 oct. 2022. url : https://cdn.sparkfun.com/assets/0/2/5/5/b/SparkFun_Triple_Axis_Accelerometer_Breakout_-_BMA400__Qwiic_.pdf?_gl=1*seqwdm*_ga*MzQwOTE10TQzLjE20TEwNDI4Njc.*_ga_T369JS7J9N*MTY5MTI0MDA4Ny4zLjAuMTY5MTI0MDA4Ny42MC4wLjA. (visité le 02/08/2023).

- [26] *3DP XTC-3D Epoxy 181gr. small, 29.50 CHF.* 3DP XTC-3D Epoxy 181gr. small, 29.50 CHF. url : https://www.3d-printerstore.ch/3DP-XTC-3D-Epoxy-181gr-small_1 (visité le 23/07/2023).
- [27] *Python MySQL.* url : https://www.w3schools.com/python/python_mysql_getstarted.asp (visité le 14/08/2023).
- [28] ST. *STM32L072 Datasheet.* Sept. 2019. url : <https://www.st.com/resource/en/datasheet/stm32l072v8.pdf>.
- [29] *What is the realistic range of LoRa ? What is the actual range that can be achieved ? | LPWA FAQ.* Murata Manufacturing Co., Ltd. url : <https://www.murata.com/en-sg/support/faqs/lpwa/lora/hardware/0008> (visité le 04/08/2023).

A | Mind Map

Annexe A. Mind Map

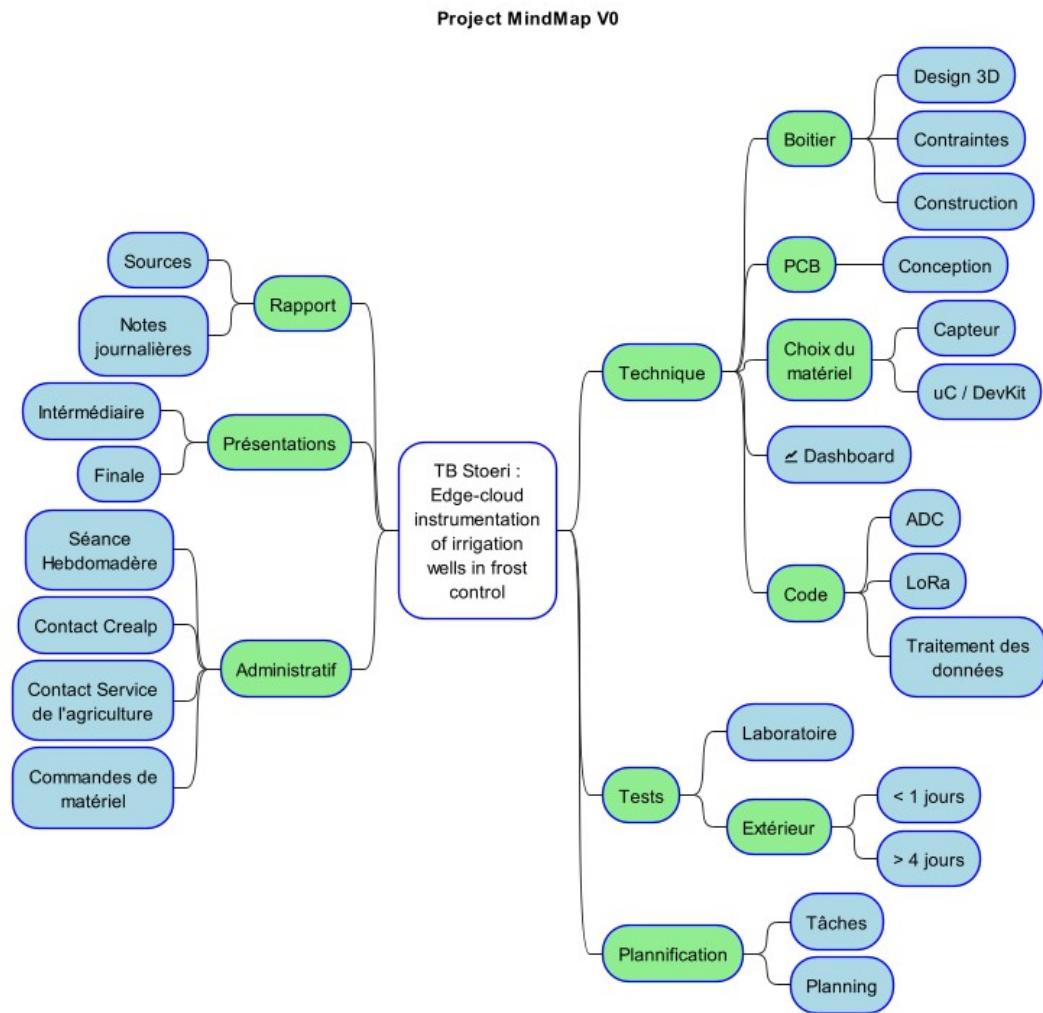


Figure A.1 Mind Map réalisé en début de projet.

B | Liste des coûts

Composant/pièce	Quantité par système	CHF/pièce pour 1 système	CHF/pièce pour 100 systèmes	Source
Micro-contrôleur	1	15.50	11.80	digikey.ch
Kit Accéléromètre	1	7.79	7.79	digikey.ch
Régulateur de tension	1	3.10	2.10	distrelec.ch
Connecteur SMA	1	1.27	1.10	mouser.ch
Antennes	1	4.57	4.57	distrelec.ch
PCB	1	55.40	3.39	eurocircuits.com
Support batteries	1	7.93	5.29	digikey.ch
Batteries Li-Ion	2	8.30	7.72	conrad.ch
Boîtier	1	4	4	Jérôme Darbelley
Tôlerie	1	4	2	Romain Masserey

Table B.1 Listes de coûts pour les pièces coûtant plus de 0.30 .-

C | Code : UART vers graphiques

Annexe C. Code : UART vers graphiques

```
import time
import serial
import matplotlib.pyplot as plt

# Serial configuration, normally this is enough information
ser = serial.Serial('COM5', 115200)

# Number of serial value reads
testDuration = 1000

# Arrays for the serial Values
xpoints = []
ypoints = []
yOnPoints = []
yTimPoints = []

for x in range(testDuration):
    serVal = ser.readline().decode().strip()
    print(serVal)
    vals = serVal.split('\x00')
    xy = vals.pop()
    xy12Sep = xy.split(",")
    if len(xy12Sep) == 4:
        xpoints.append(int(xy12Sep[3]) / 800 * 1000)
        yTimPoints.append(int(xy12Sep[2]))
        ypoints.append(int(xy12Sep[1]) / 1024)
        if len(xy12Sep[0]) != 1:
            yOnPoints.append(int(xy12Sep[0]))
        else:
            yOnPoints.append(int(xy12Sep[0]))

    # Copy values in a file
    # To be safe and maybe re-use them if necessary
    file1 = open('testLab.txt', 'a')
    localtime = time.asctime(time.localtime(time.time()))
    file1.write("Test plot_AlternateWithWater_1 : \n" + localtime + "\n\n")
    file1.write("x points \n")
    file1.write(str(xpoints) + "\n\n")
    file1.write("Accel points \n")
    file1.write(str(ypoints) + "\n\n")
    file1.write("On/Off points \n")
    file1.write(str(yOnPoints) + "\n\n\n")
    file1.close()

# Adjusting for not superposing titles and graphs
fig = plt.figure()
fig.subplots_adjust(hspace=0.5, wspace=0.5)

# Plot the acceleration datas
plt.subplot(2, 1, 1)
plt.scatter(xpoints, ypoints, s=5)
plt.title("Accélération ")
plt.xlabel("Temps ms")
plt.ylabel("Accélération résultante en g")

# Plot the On/Off datas
plt.subplot(2, 1, 2)
plt.plot(xpoints, yOnPoints)
plt.title("Eau On/Off")
plt.xlabel("Temps ms")
plt.ylabel("1 = On, 0 = Off")
```

```
#These five lines where to test a timer
#plt.subplot(2, 1, 2)
#plt.plot(xpoints, yTimPoints)
#plt.title("Timer")
#plt.xlabel("Temps ms")
#plt.ylabel("Timer Value")

# Save the plot
plt.savefig('plot_AlternateWithWater_1.png')

# Show the wonderful plot
plt.show()
```


D | Code : UART vers MySQL

Annexe D. Code : UART vers MySQL

```
import serial
import mysql.connector

# Not used for this test
literPerHour = 0
# According to the document from Steve Maillard
literPerMinute = 2000
# Conversion
literPerSecond = literPerMinute/60

# Connect to mySQL function
def connectMySQL():
    try:
        db = mysql.connector.connect(
            host='localhost',
            port=3306,
            user='root',
            passwd='Past-1234',
            database='tb_stoeri',
            auth_plugin='mysql_native_password',
            connection_timeout=10)
    return db
    except mysql.connector.Error as e:
        print("ERROR %d: %s" % (e.args[0], e.args[1]))

# Open Serial port COM8 with baudrate = 9600
# No other parameters needed
ser = serial.Serial('COM8', 9600)

#Forever loop
while 1 :
    serVal = ser.readline().decode().strip()           #Read Line from Serial
    print(serVal)
    if "Seconds : " in serVal:                      #Does the line have "Seconds
        → : " in it
        serVal = serVal.replace('Seconds : ', '')
        print(serVal)
        valToSQL = int(serVal) * literPerSecond       # Convert in liters
        db = connectMySQL()                          # Call connect function
        curs = db.cursor()

        # Insert datas in database
        curs.execute("INSERT INTO `tb_stoeri`.`myDataForWater` (`waterQuantity`)
        → VALUES ('%d');" % (valToSQL))
        db.commit()
```

E | Code : TTN vers MySQL

Annexe E. Code : TTN vers MySQL

```
import paho.mqtt.client as mqtt
import matplotlib.pyplot as plt
import mysql.connector

# 3000 is the value from the lab test bench
literPerHour = 3000
literPerMinute = literPerHour/60
literPerSecond = literPerMinute/60

# Called when connect to ttn with threw MQTT
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("v3/stm32-sdk@ttn/devices/eui-70b3d57ed005fdb3/up")

# Connect to MySQL function
def connectMySQL():
    try:
        db = mysql.connector.connect(
            host='localhost',      #local host ou ip de l'ordi
            port=3306,             # normalemen c'est bon
            user='root',           # Mot de passe database
            passwd='...',          # Mot de passe database
            database='tb_stoeri',
            auth_plugin='mysql_native_password',
            connection_timeout=10)
        return db
    except mysql.connector.Error as e:
        print("ERROR %d: %s" % (e.args[0], e.args[1]))
```

```

# If a message arrives on TTN, that function is called
def on_message(client, userdata, msg):
    # Print the topic and the payload to see all the info
    print(msg.topic+" "+str(msg.payload))

    # Take out of the payload only the seconds value
    pos = str(msg.payload).find('bytes')
    realPayload = ''
    for i in range(20) :
        realPayload = realPayload + str(msg.payload)[i+pos+8]

    secondsOnly = ''
    pos = str(realPayload).find(',0,255,0,136')
    for i in range(pos) :
        secondsOnly = secondsOnly + realPayload[i]

    # Separate both bytes from each other
    separate = secondsOnly.split(',')
    print(separate)
    # Put the two bytes together with there right multiple
    secondsData = int(separate[0])*16*16+int(separate[1])
    literData = secondsData * literPerSecond
    pos = str(msg.payload).find("time:")
    time = ''
    for i in range(20) :
        time = time + str(msg.payload)[i+pos+8]
    print(time)

#Connect to the database
db = connectMySQL()
curs = db.cursor()

# Insert into the data base
curs.execute("INSERT INTO `tb_stoeri`.`myDataForWater` (`waterQuantity`)
    ↳ VALUES ('%d');"%literData))
db.commit()

# Créer le client
client = mqtt.Client()

# Attribution of callbacks
client.on_connect = on_connect
client.on_message = on_message

# Connect the MQTT client
client.username_pw_set("stm32-sdk@ttn", password="...") # Mot de passe TTN
client.connect("eu1.cloud.thethings.network", 1883, 60)

# Loop loop forever
client.loop_forever()

```


F | Code : MySQL vers graphique n°1

Annexe F. Code : MySQL vers graphique n°1

```
import mysql.connector
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.dates as mdates

# Function to connect to mySQL database
def connectMySQL():
    try:
        db = mysql.connector.connect(
            host='localhost',
            port=3306,
            user='root',
            passwd='Past-1234',
            database='tb_stoeri',
            auth_plugin='mysql_native_password',
            connection_timeout=10)
        return db
    except mysql.connector.Error as e:
        print("ERROR %d: %s" % (e.args[0], e.args[1]))

# Connection to the database
db = connectMySQL()
cursor = db.cursor()

# Data recuperation from mySQL :
# First line : All datas
# Second line : 6 last datas

# cursor.execute("SELECT * FROM myDataForWater")
cursor.execute("SELECT * FROM myDataForWater ORDER BY idmyDataForWater DESC
    ↳ LIMIT 6")

data = cursor.fetchall()

# Close connection
cursor.close()
db.close()
```

```
# If there are some datas
if data:
    # Select what I want from the datas
    x_data = [item[1] for item in data] # ChatGPT helped
    y_data = [item[2] for item in data] # ChatGPT helped
    print(x_data)
    print(y_data)

    # invert order
    x_data.reverse()
    y_data.reverse()

    # Create the graph
    plt.stem(x_data, y_data, basefmt="lightblue")

    # Formtating to good date format
    plt.gcf().autofmt_xdate() # ChatGPT helped
    date_format = mdates.DateFormatter('%d-%m %H:%M') # ChatGPT helped
    plt.gca().xaxis.set_major_formatter(date_format) # ChatGPT helped

    # Name the axes and show plot
    plt.title("Litres consommés par cycle")
    plt.xlabel("Temps")
    plt.ylabel("Litres")
    plt.show()
```


G | Code : MySQL vers graphique n°2

Annexe G. Code : MySQL vers graphique n°2

```
import datetime
import mysql.connector
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Function to connect to mySQL database
def connectMySQL():
    try:
        db = mysql.connector.connect(
            host='localhost',
            port=3306,
            user='root',
            passwd='Past-1234',
            database='tb_stoeri',
            auth_plugin='mysql_native_password',
            connection_timeout=10)
        return db
    except mysql.connector.Error as e:
        print("ERROR %d: %s" % (e.args[0], e.args[1]))

# Connection to the database
db = connectMySQL()
cursor = db.cursor()

# Data recuperation from mySQL :
# First line : All datas
# Second line : 6 last datas

# cursor.execute("SELECT * FROM myDataForWater")
cursor.execute("SELECT * FROM myDataForWater ORDER BY idmyDataForWater DESC
    ↳ LIMIT 6")

data = cursor.fetchall()

# Close connection
cursor.close()
db.close()
```

```

# If there are some datas
if data:
    # Select what I want from the datas
    x_data = [item[1] for item in data] # ChatGPT helped
    y_data = [item[2] for item in data] # ChatGPT helped
    print(x_data)
    print(y_data)

    # invert order
    x_data.reverse()
    y_data.reverse()

    for i in range(1, len(y_data)):
        y_data[i] = y_data[i] + y_data[i - 1]
    print(y_data)

    # These two lines are for the case that is shown in the report
    # They can be removed but it's not beautiful
    x_data.append(datetime.datetime(2023, 8, 9, 16, 0, 0))
    y_data.insert(0, y_data[0])

    # Create the graph
    plt.figure(figsize=(10, 5))
    plt.step(x_data, y_data)

    # Formatting to good date format
    plt.gcf().autofmt_xdate() # ChatGPT helped
    date_format = mdates.DateFormatter('%d-%m %H:%M') # ChatGPT helped
    plt.gca().xaxis.set_major_formatter(date_format) # ChatGPT helped

    # Name the axes and show plot
    plt.title("Litres consommés au total")
    plt.xlabel("Temps")
    plt.ylabel("Litres")
    plt.show()

```


H | Graphiques de tests du prototype n°1

Annexe H. Graphiques de tests du prototype n°1

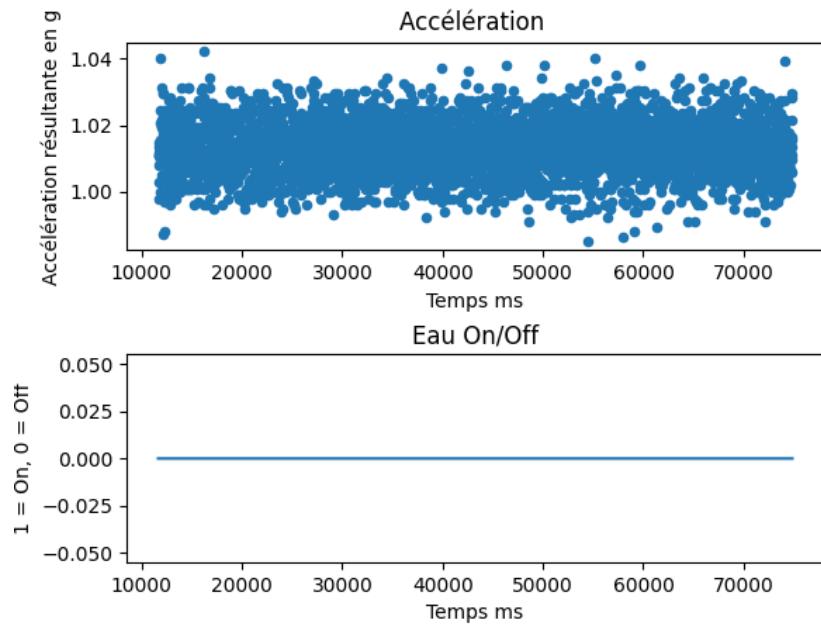


Figure H.1 test 1 : système fixé côté pompe, banc de test éteint.

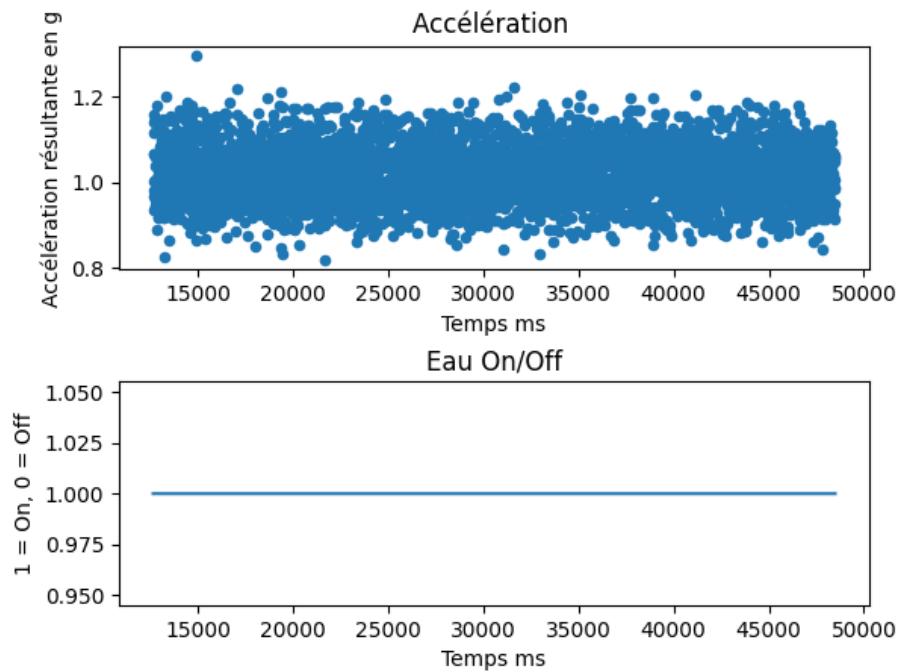


Figure H.2 Test 2 : système fixé côté pompe, pompe allumée mais vanne fermée

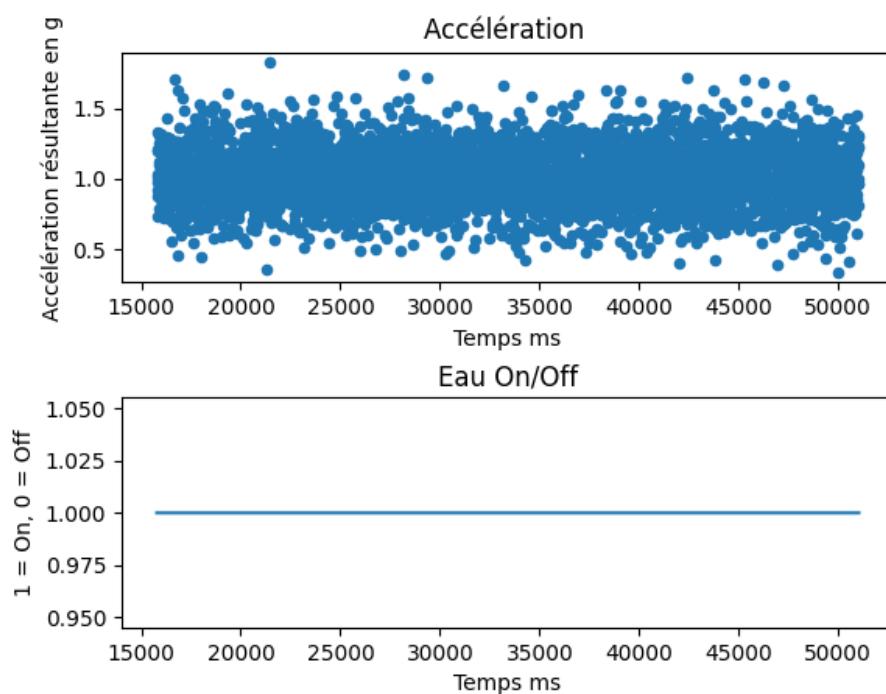


Figure H.3 Test 3 : système fixé côté pompe, pompe allumée et vanne ouverte

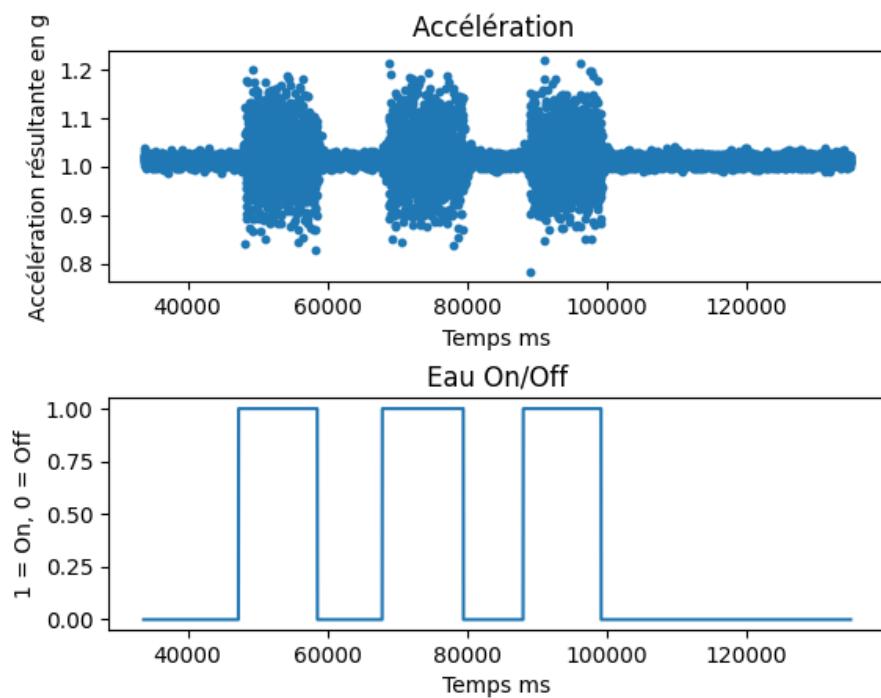


Figure H.4 Test 4 : système fixé côté pompe, 3 cycles d'allumage avec vanne fermée.

Annexe H. Graphiques de tests du prototype n°1

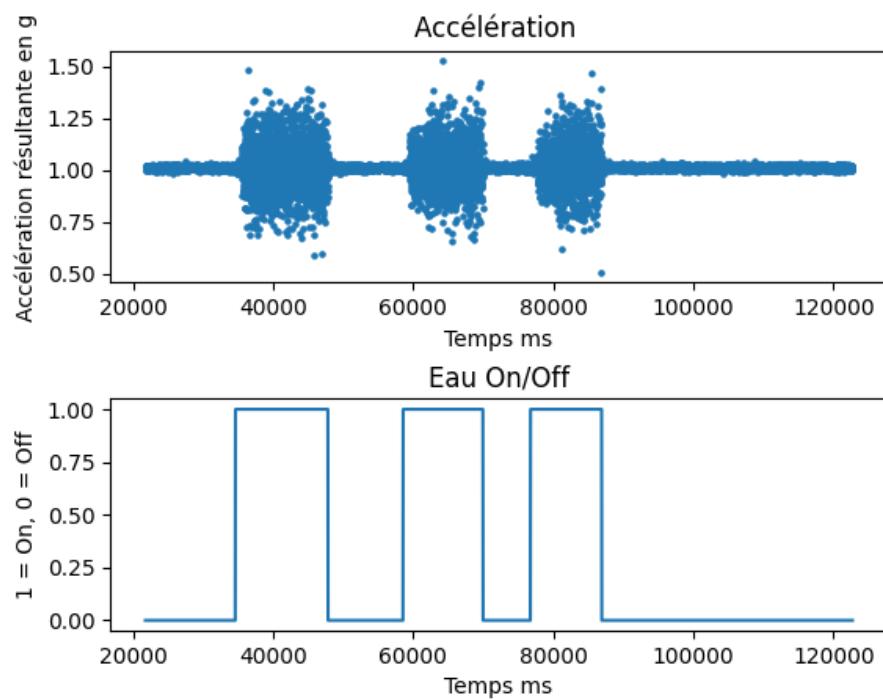


Figure H.5 Test 5 : système fixé côté pompe, 3 cycles d'allumage avec vanne ouverte.

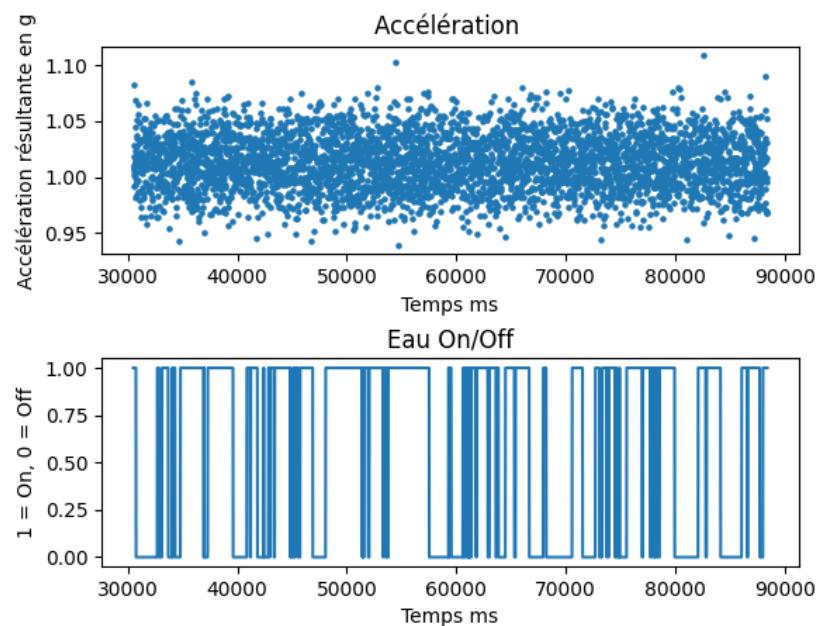


Figure H.6 Test 6 : système fixé côté opposé à la pompe, pompe allumée et vanne fermée.

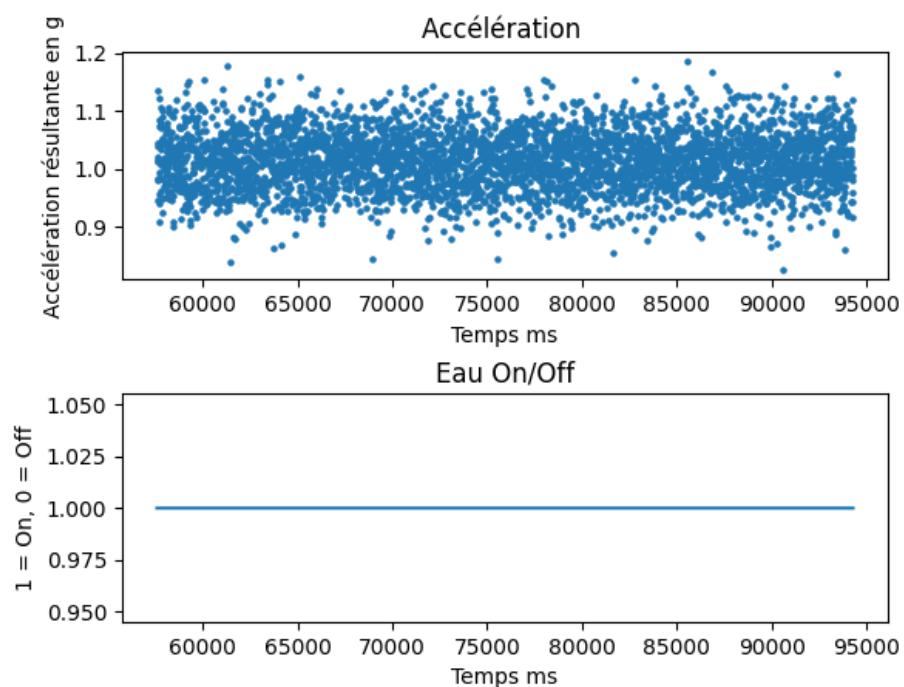


Figure H.7 Test 7 : système fixé côté opposé à la pompe, pompe allumée et vanne ouverte.

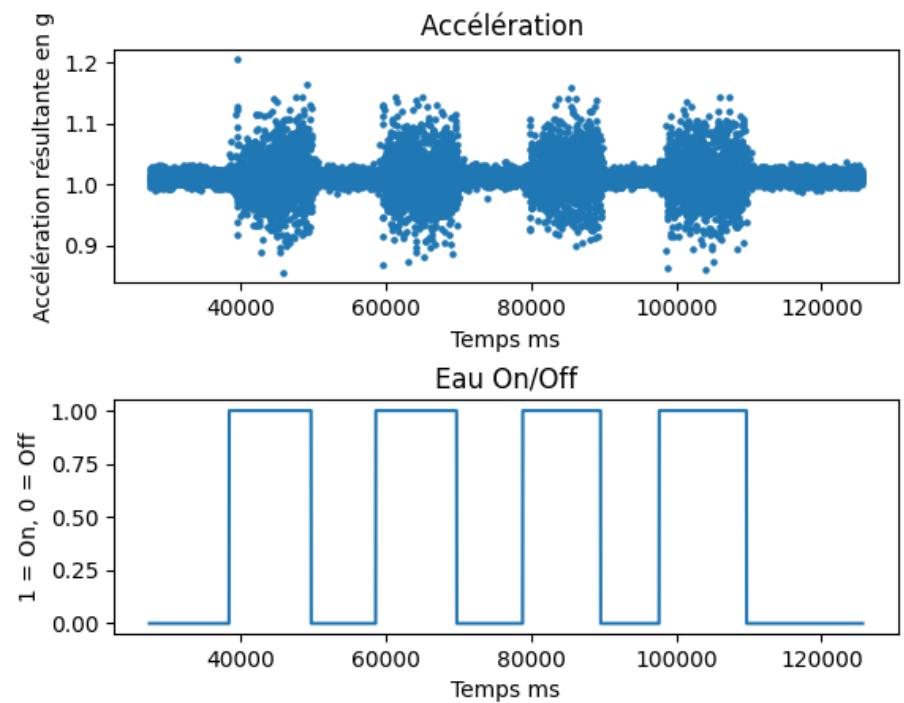
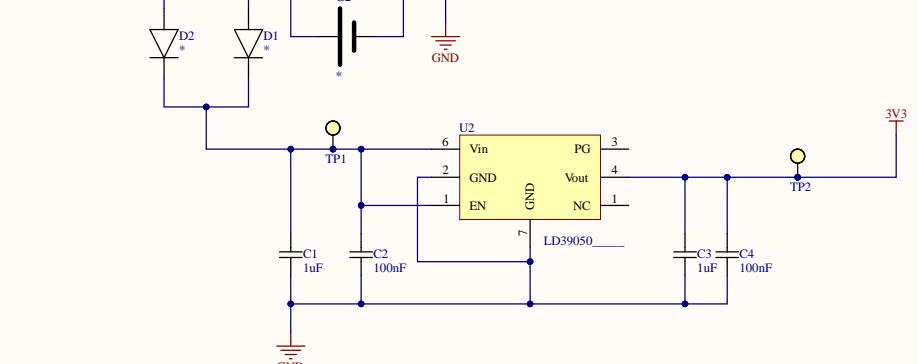


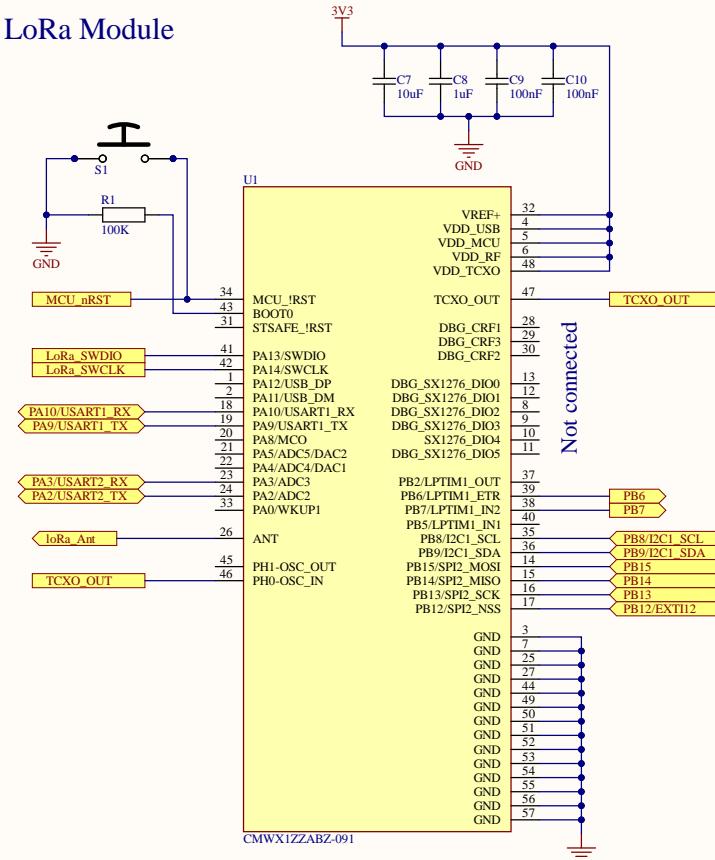
Figure H.8 Test 8 : système fixé côté opposé à la pompe, 4 cycle d'allumage avec vanne ouverte.

I | Électronique

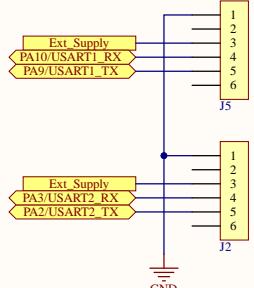
Power



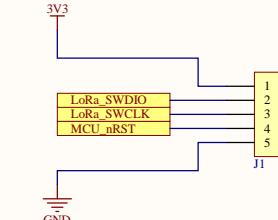
LoRa Module



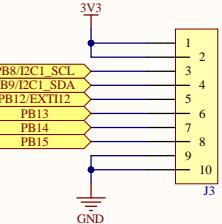
UART connector



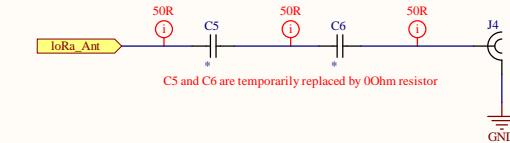
Programmation connector



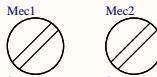
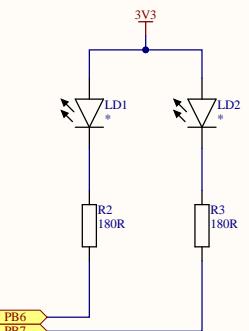
Sensors Connector



Antenna



LEDs



stm32LoRaAccel.PrjPcb

stm32LoRaAccel.SchDoc

Hes-SO // VALAIS WALLIS

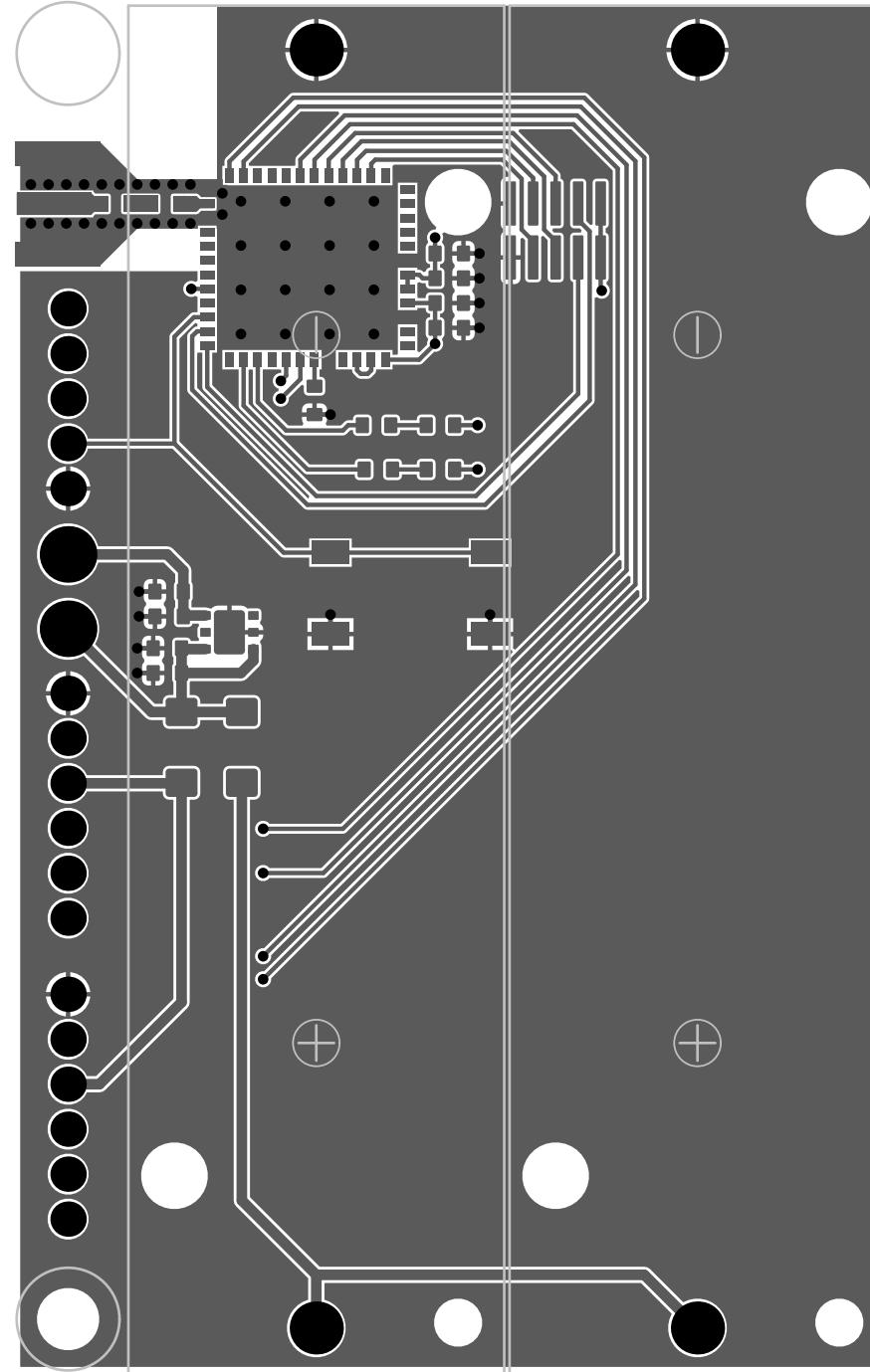
Date : 19.06.2023



Revision : 1.0

Sheet 1 of 1

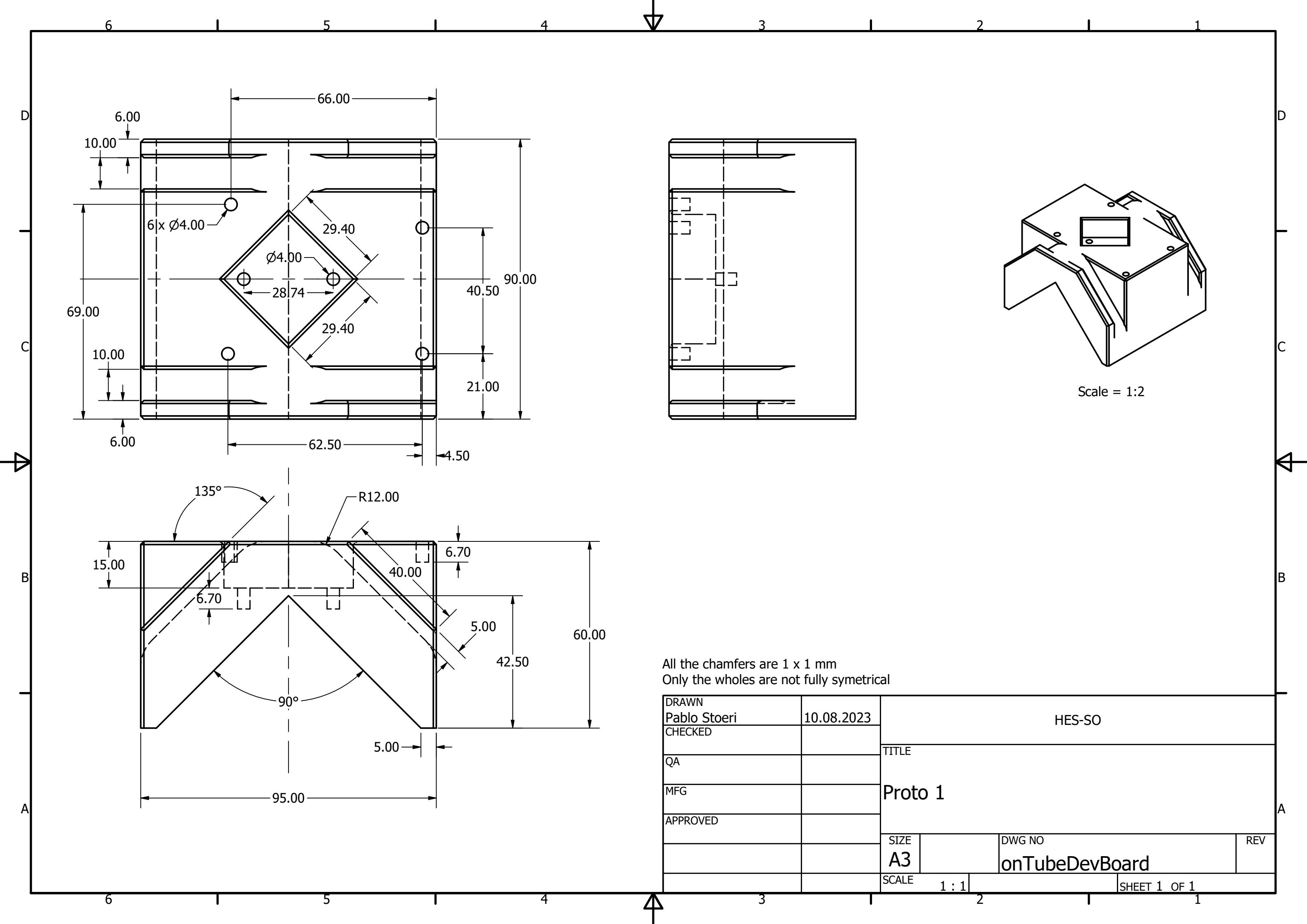
Design by : Stoeri P.



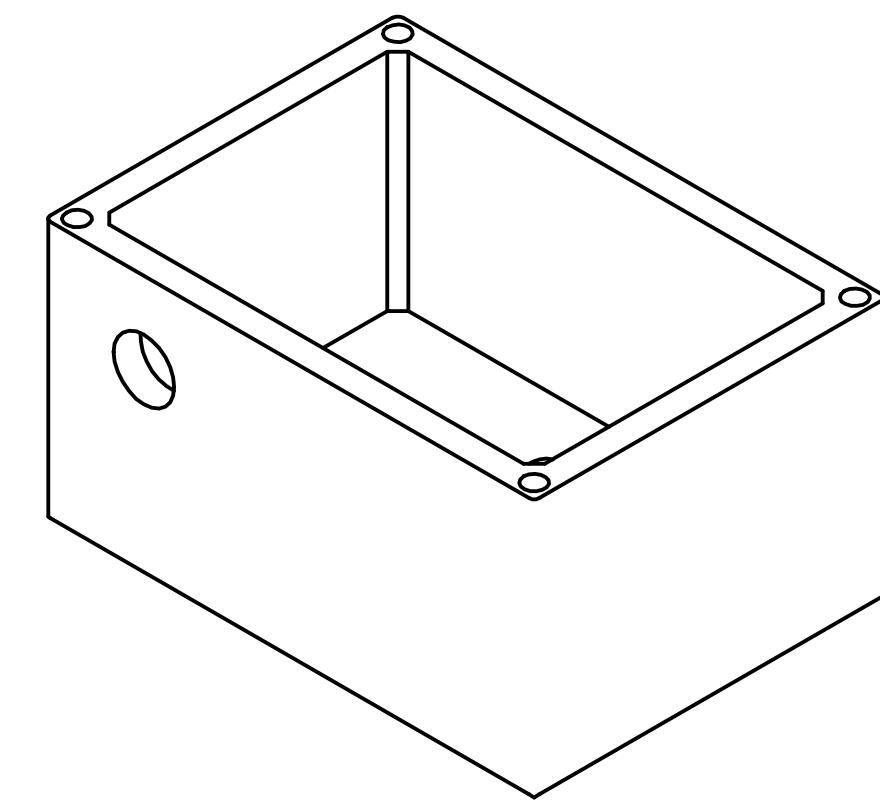
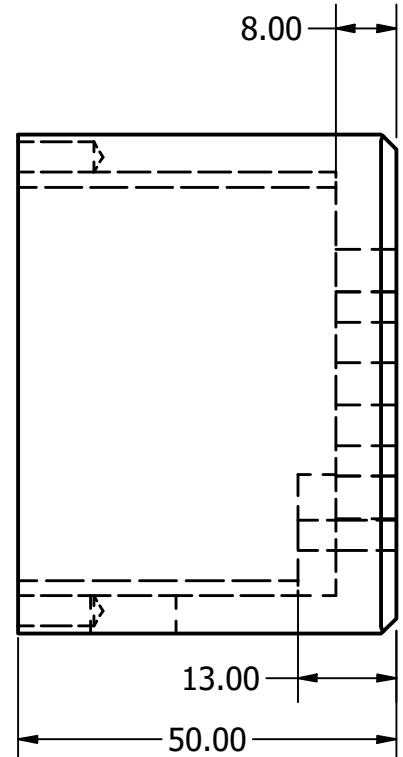
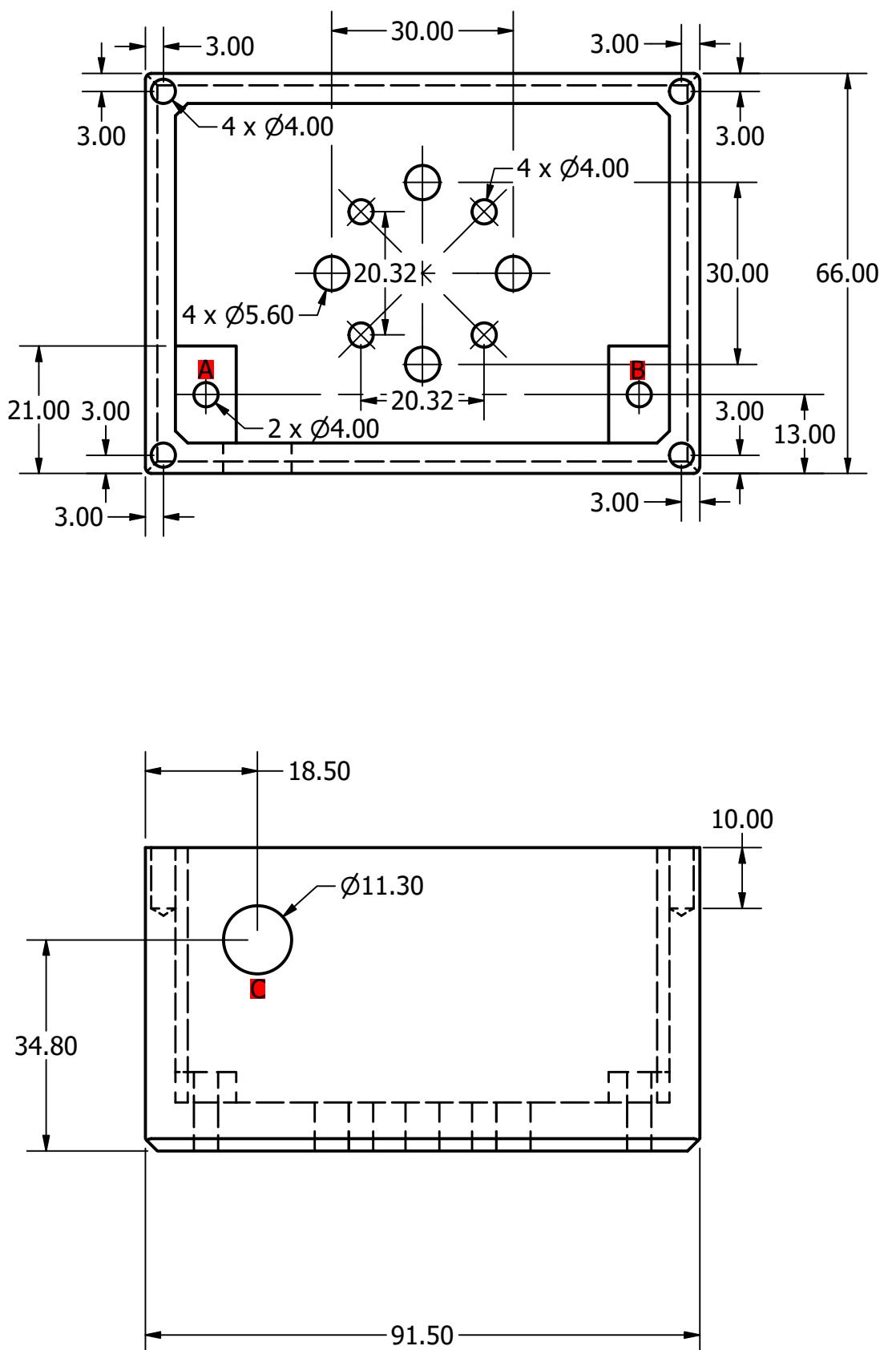
+

+

J | Mécanique : Prototype n°1

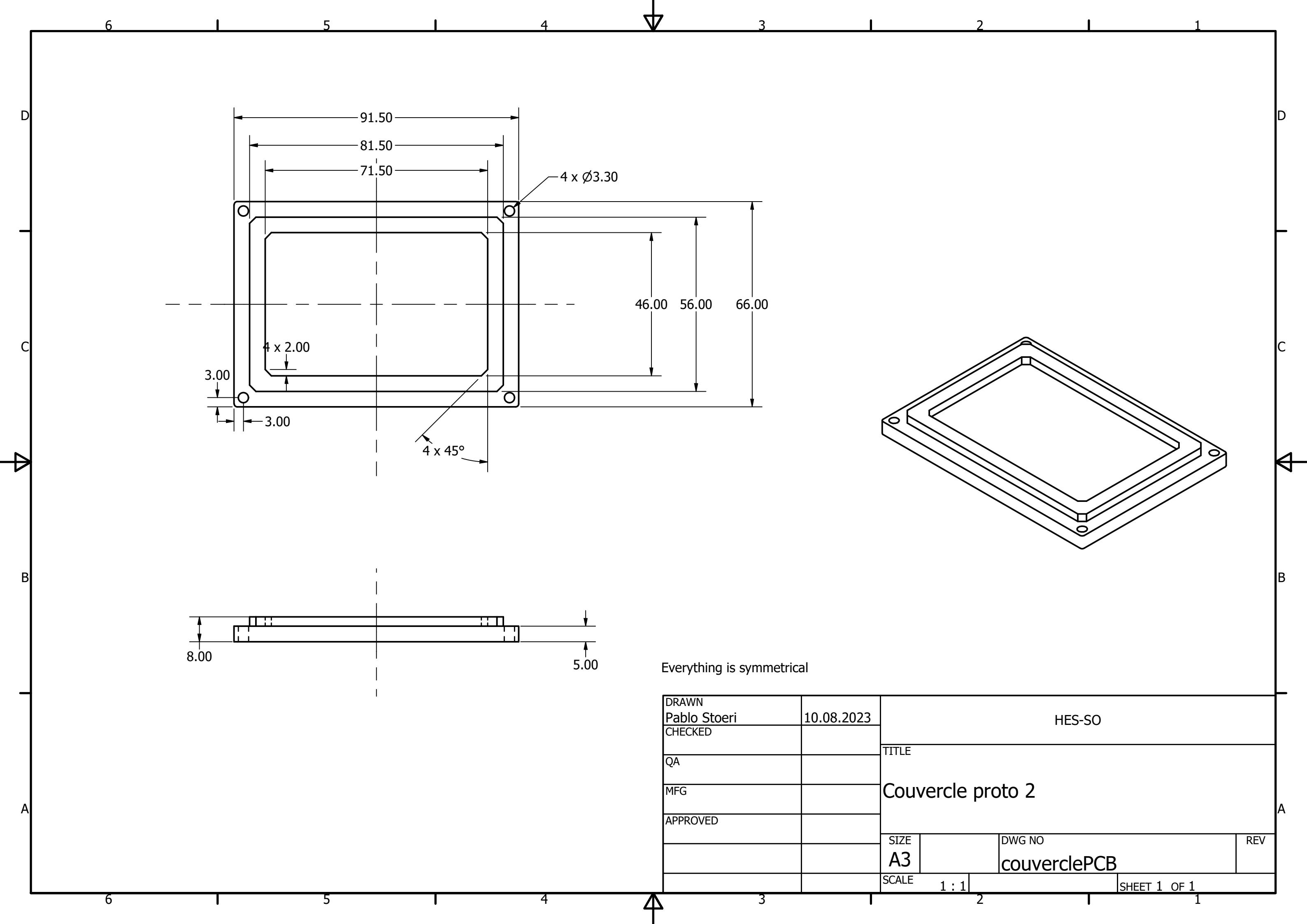


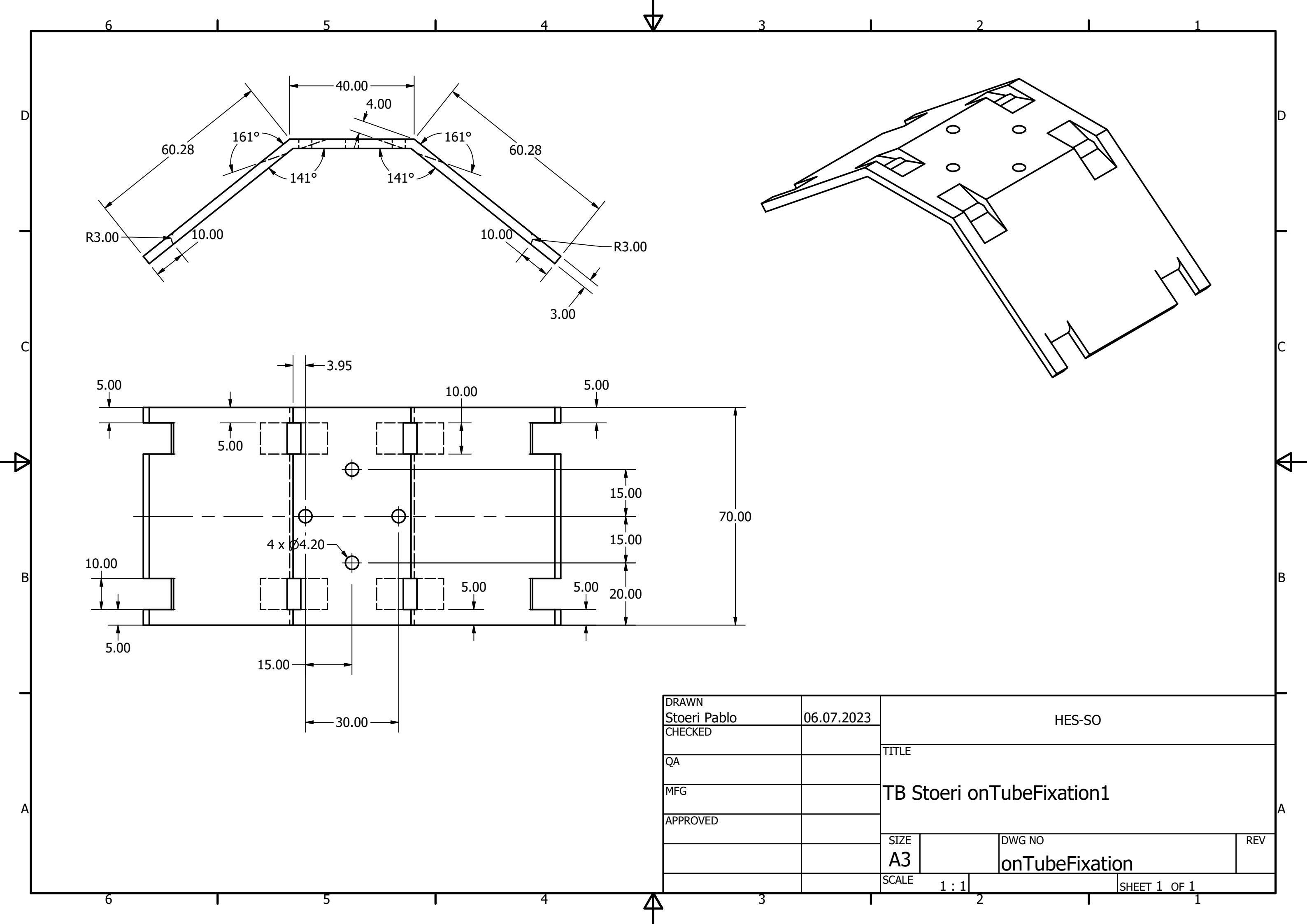
K | Mécanique : Prototype pré-industriel



Except A, B and C, everything is symmetrical from the top view

DRAWN Pablo Stoeri	10.08.2023	HES-SO		
CHECKED				
QA		TITLE		
MFG		Boîtier proto 2		
APPROVED				
SIZE A3		DWG NO boitierPCB	REV	
SCALE 1 : 1				SHEET 1 OF 1





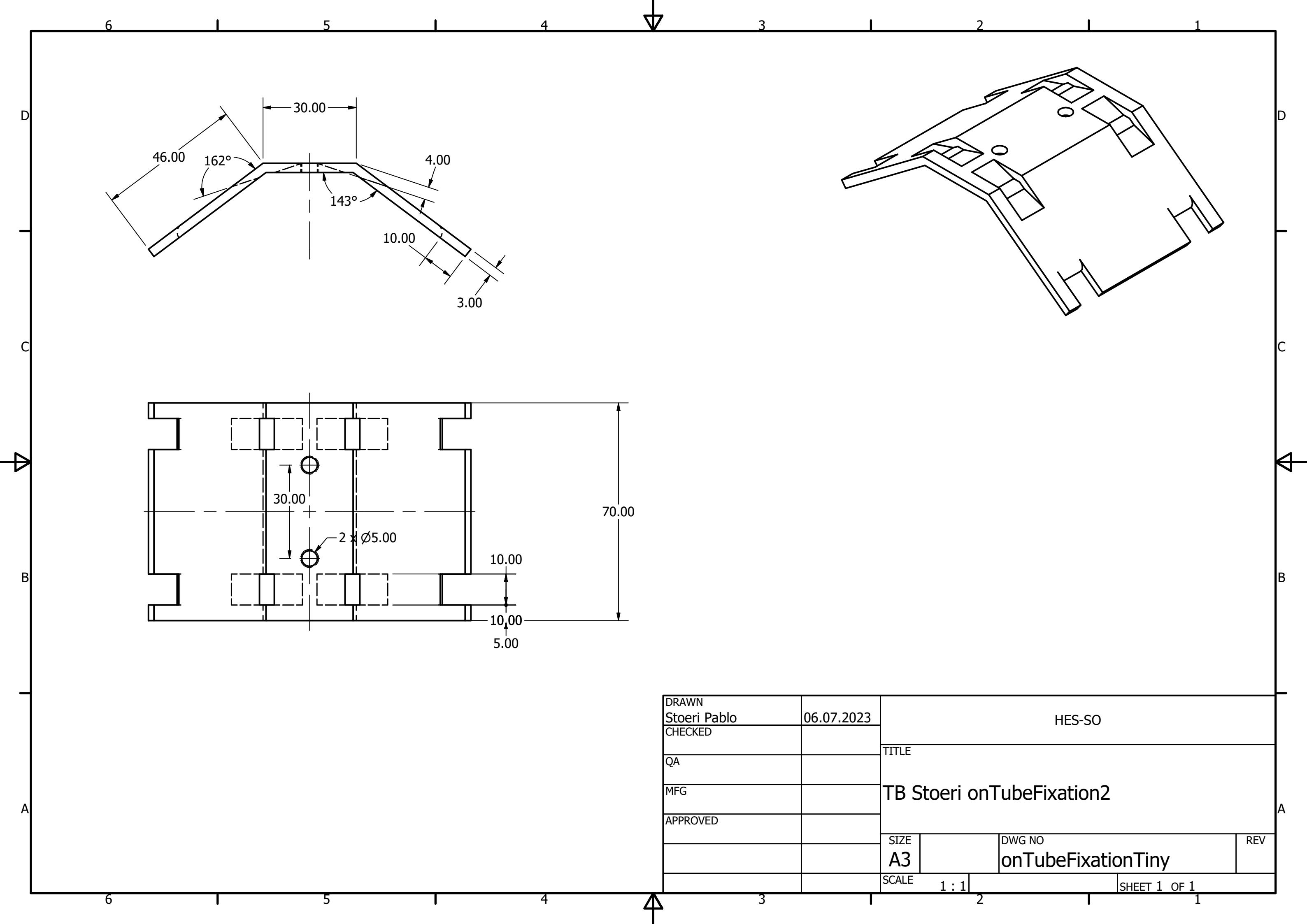


Table des figures

1.1	Puits agricoles en fonctions avec deux type de pompes différentes [7].	2
2.1	Débit d'une pompe thermique en fonction de la pression, graphique tirée de la page 78 du livre de Erik Trostmann [9].	6
2.2	Tests effectué par le CREALP avec un accéléromètre [10].	7
3.1	Structure globale du projet	9
3.2	Structure globale du système	9
3.3	Planing simplifié du projet	11
3.4	Diagrammes des étapes clés	12
4.1	Kit de développement sélectionné : B-072Z-LRWAN1	14
4.2	Kit SparkFun avec un accéléromètre BMA400.	16
4.3	Diagramme de structure du code	17
4.4	Diagramme global du protocole I2C	18
4.5	Fonctions I2C : Premières variantes	19
4.6	Structure de l'IT-SDK	24
4.7	3D de la pièce de fixation pour les kit de développement.	27
4.8	Schéma électronique simplifié.	29
4.9	Résultat final de la partie électronique.	30
4.10	Design des tôles de fixation	32
4.11	Design du boîtier final.	33
4.12	Montage du prototype n°1 (sans couvercle).	36
4.13	Montage final avec le nouveau PCB (sans couvercle).	37
5.1	Installation complète utilisée pour les tests en laboratoire.	40
5.2	Graphique du test 7	42
5.3	Graphique du test 6	43
5.4	Graphique du test 8	44
5.5	Litres consommés à chaque cycle d'utilisation.	45
5.6	Total des litres consommés en fonction du temps.	46
5.7	Mesure de courant sleep-I2C-LoRa.	47
5.8	Mesure de courant au repos.	48
5.9	Image thermique du PCB.	49
5.10	Image thermique du PCB.	49
5.11	Prototype pré-industriel installer sur une conduite souterraine.	53
5.12	Mesures prises sur le terrain : litres consommés par cycles.	54
5.13	Mesures prises sur le terrain : litres consommés au total.	55
6.1	Prototype final en laboratoire.	59

6.2	Consommation présumée de 5 puits.	63
6.3	Consommation en couleur de différents puis dans une région.	63
A.1	Mind Map réalisé en début de projet.	70
H.1	test 1 : système fixé côté pompe, banc de test éteint.	92
H.2	Test 2 : système fixé côté pompe, pompe allumée mais vanne fermée . .	92
H.3	Test 3 : système fixé côté pompe, pompe allumée et vanne ouverte . .	93
H.4	Test 4 : système fixé côté pompe, 3 cycles d'allumage avec vanne fermée.	93
H.5	Test 5 : système fixé côté pompe, 3 cycles d'allumage avec vanne ouverte.	94
H.6	Test 6 : système fixé côté opposé à la pompe, pompe allumée et vanne fermée.	94
H.7	Test 7 : système fixé côté opposé à la pompe, pompe allumée et vanne ouverte.	95
H.8	Test 8 : système fixé côté opposé à la pompe, 4 cycle d'allumage avec vanne ouverte.	95

Liste des tableaux

4.1	Accéléromètre avec sortie analogique, points positifs et négatifs.	15
4.2	Configuration des registres de l'accéléromètre.	21
5.1	Cycles effectués par la pompe du banc de test.	45
5.2	Consommation théorique des composants	48
B.1	Listes de coûts pour les pièces coûtant plus de 0.30 .-	72