Rapport
Projet HuSki

INFO5 - S10

Par **Joffrey FERREIRA**, **Quentin FOMBARON**, **Titouan LARNICOL**,
Tim LEPAGE et **Thomas OZENDA**

Tuteur: **Bernard TOURANCHEAU**

Tuteur: **Bernard TOURANCHEAU**

I - Présentation du projet **2**
Le sujet 2
L'équipe 2
Le Budget prévisionnel 3

II - Travail réalisé **3**
Partie hardware 3
HuCard 3
HuConnect 4
Boitiers 4
Partie Software 5
HuCard 5
HuConnect 5
L'application 6
DevOps 8
Pipeline 8
Qualité de code 9

III -Tests **9**
1.Conditions réels 9

IV - Difficultés rencontrées **10**

V - Ouvertures et Conclusion **11**

I - Présentation du projet

1. Le sujet

La recherche d'un ski perdu est une expérience déplaisante de par l'effort que nécessite la recherche ainsi que de par l'idée de devoir avoir à acheter une nouvelle paire de ski. La présence de poudreuse rend d'autant plus difficile cette recherche car il est parfois impossible de retrouver des traces du ski perdu.
Le système que nous allons décrire ici a pour but de remédier à ce problème en utilisant les moyens modernes offerts par l'IoT. Nous avons décidé de nommer notre produit ***HuSki*** en références aux chiens de traineaux réputés pour leur ténacité et leur sens de l'odorat.

HuSki se décline en trois composants:

- La ***HuCard***, une carte dans un boîtier fixé sur les skis. Cette carte traque les skis et envoie leur position GPS par communication LoRa.
- La ***HuConnect***, une carte dans une coque de téléphone servant de gateway entre la *HuCard* et l'application mobile. La *HuConnect* reçoit les données LoRa et les transmet via Bluetooth.
- L'application ***HuSki***, une application mobile android pour géolocaliser les *HuCards* connectées.

Dans la suite de ce rapport nous allons détailler le fonctionnement de chacun de ces trois composants, les difficultés rencontrées lors du développement et nous finirons avec les ouvertures que ce projet apporte. Cependant, avant de poursuivre, il semble tout de même nécessaire de présenter l'équipe.

2. L'équipe

Nous étions pour ce projet une équipe de cinq étudiants, nous avons adopté la méthode SCRUM avec des sprints de une semaine. Nous disposions donc d'un scrum master: Joffrey chargé d'organiser le travail avec un kanban et de s'assurer du bon fonctionnement de l'équipe et d'un chef de projet: Quentin, responsable de la communication avec les professeurs. Nous nous sommes répartis en 3 équipes: hardware, software et DevOps.

- ➔ Quentin et Joffrey se sont occupés de la partie hardware, c'est à dire de l'utilisation des cartes électroniques, leur configuration et leur programmation.
- ➔ Tim et Titouan ont développé l'application native sous Android Studio.
- ➔ Thomas a mis en place l'organisation du développement logiciel via Jenkins, Gradle, Codacy et autres outils de génie logiciel.

Le projet s'étend sur une période de 3 mois pour un temps cumulé de 165 heures. Pour la communication entre nous nous utilisions Messenger et pour l'organisation Trello.

3. Le Budget prévisionnel

Le budget calculé de ce projet s'élève comme ci-après:

	Par heure (€)	Total (€)
Salaires	120.35	19 857.75
Bâtiment/matériel	5.45	900
Frais exceptionnels (total) Forfaits ski Achat matériel		170 70 100
Ordinateurs personnels	6.96	1 148.32
TOTAL		22 076.07

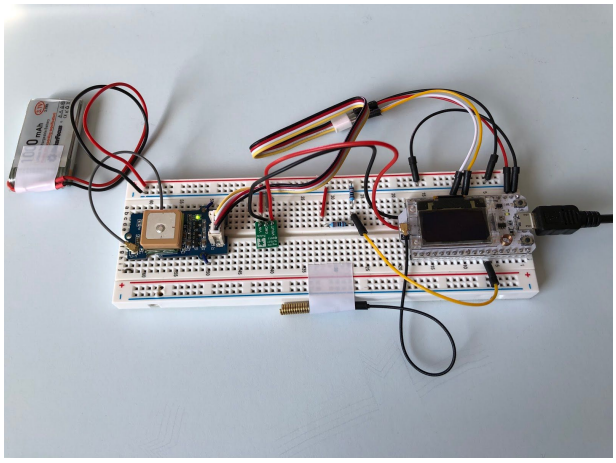
II - Travail réalisé

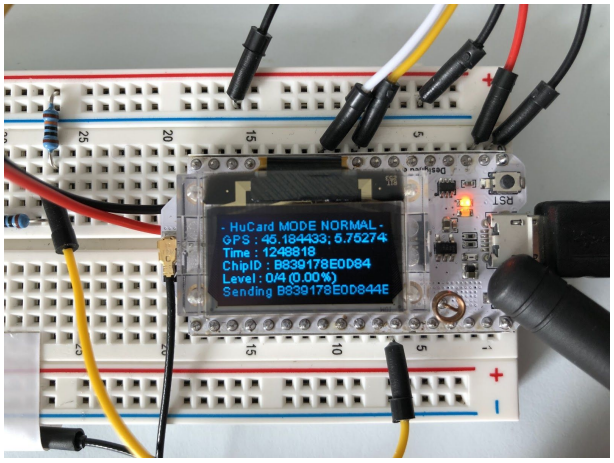
1. Partie hardware

a. HuCard

Le modèle de carte choisi est l'**Heltec WiFi LoRa 32 V2**, cette dernière permet une multitude de chose et est équipée d'un écran OLED aidant au debuggage. Notre version peut émettre en LoRa sur un plage de fréquence de 433 à 510 MHz, optimal pour espérer transmettre sous la neige. C'est à la *HuCard* que revient le travail de capter le signal GPS afin de récupérer ses coordonnées, c'est donc tout naturellement qu'une puce SIM28 et antenne GPS ont été intégrées au système. Un travail d'électronique a été nécessaire, un simple diviseur de tension avec 2 résistances de 100 kΩ afin de mesurer la tension de la batterie 3,7V en évitant de griller la carte dont les ports analogiques ne supportent que jusqu'à 3.3V. Un régulateur de tension a aussi été implémenté pour délivrer à la carte une tension constante de 5V dont a besoin la puce GPS pour fonctionner.

Les ports 22 et 23 servent respectivement à la réception (RX) et transmission (TX) des données à la puce GPS. Le port analogique 39 est lui chargé de mesurer la tension de la batterie, la tension est mesurée à la sortie du diviseur de tension.





b. HuConnect

Comme pour la HuCard une **Heltec Wifi LoRa 32** a été choisi pour les mêmes raisons que pour la HuCard ainsi que pour sa capacité de communiquer à la fois en bluetooth et en LoRa. Pour réaliser le prototype il faut donc également utiliser un diviseur de tension pour récupérer le niveau de batterie et l'envoyer à l'application.

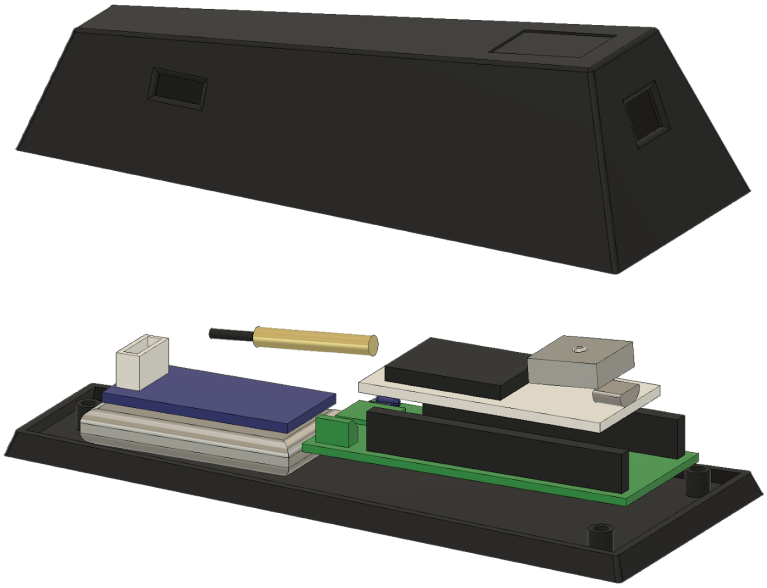
La batterie se vidant assez rapidement à cause de la lourde quantité de paquet transitant en LoRa et l'utilisation du module bluetooth de la carte, il a été nécessaire de réaliser des optimisations dont celles ci :

- La carte ne fait rien tant qu'aucune connexion bluetooth est établie.
- Si une carte est mise en mode active, la HuConnect envoie un message aux autre HuCard pour couper leur transmission LoRa
- Une HuCard émet seulement si elle est appareillée par une HuConnect

Une optimisation a été imaginé mais trop tard : Le niveau de batterie des cartes est codé par un entier allant de 0 à 4. Ce dernier changeant peu rapidement il serait plus efficace de ne pas le transmettre à chaque fois mais plutôt que les cartes envoie un paquet différent pour le mettre à jour.

c. Boitiers

Afin de proposer un prototype complet et fonctionnel, nous avons pensé à modéliser un boîtier et l'imprimer en 3D. Ce dernier doit respecter des contraintes de taille, de poids, d'étanchéité et de praticité pour clipser/déclipser le boîtier au ski afin de recharger la batterie. Nous avons épuré un maximum le design pour l'intégrer au ski avec une forme favorisant l'écoulement de l'eau et évitant la neige stagnante. C'est à l'aide de *Fusion360* que le boîtier a été modélisé et imprimé avec une Creality CR-10.



2. Partie Software

a. HuCard

Lorsque la *HuCard* est allumée, cette dernière commence à se localiser mais attend d'être ajouté sur l'application avant de transmettre. Une fois les coordonnées GPS acquises, un paquet LoRa est transmis comportant :

- L'adresse MAC de la carte (ID unique)
- La latitude, longitude et altitude
- Le niveau de la batterie (entre 0 et 4)

La carte reçoit également des des paquets de l'application afin de modifier son mode de transmission qui sont composés comme ceci :

- L'adresse MAC de la carte
- Le mode de transmission (**0** : NORMAL, **1** : ACTIVE, **3** : INITIALIZE, **4** : STOP)

Pour changer de mode de transmission, la *HuCard* doit recevoir un paquet LoRa contenant son adresse MAC suivi du mode dans lequel elle doit être mise.

b. HuConnect

Cette carte pouvant communiquer en LoRa et en Bluetooth, elle sert de passerelle entre notre application et les *HuCards*. Au démarrage, une liste de d'adresse MAC (ChipID) vide est initialisée qui sera remplie et modifié par l'application. Quand un paquet LoRa est reçu, la *HuConnect* le déchiffre, vérifie si elle fait partie de sa table et le transmet en bluetooth à

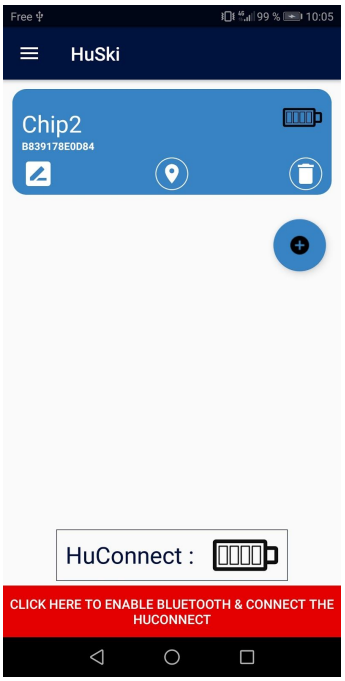
l'application si c'est le cas. Et inversement elle envoie en LoRa les messages reçus par bluetooth de la part de l'application.

c. L'application

L'application est composée de deux pages principales. La première représente la liste des cartes ***HuCard*** connectées et la deuxième est la page de géolocalisation de la ***HuCard*** sélectionnée. Nous allons dans la suite détailler les différentes fonctionnalités.

Architecture du projet Android Studio

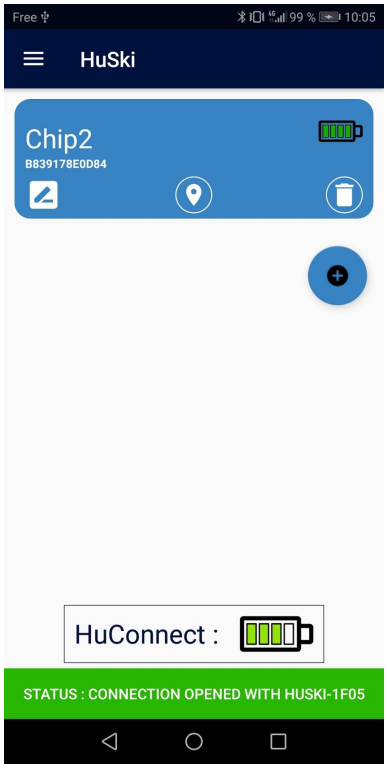
L'application utilise les fragments d'Android Studio, chacun représentant une page: *ListFragment* et *FindFragment*. La navigation entre les pages est implémentée dans *MainActivity*. Le répertoire *dataStructure* contient les classes java représentant les structures de données utilisées.



Le premier fragment (ou page), *ListFragment*, est le fragment d'accueil. Il contient la liste des *HuCards* enregistrées, offre la possibilité de les renommer, les supprimer ou bien d'en enregistrer d'autres. Il n'y a pas de limite au nombre de *HuCards* enregistrables, une application *HuSki* peut donc suffire à gérer les skis de toute une famille. Ce fragment indique aussi le niveau de batterie de chacune des *HuCards* ainsi que celui de la *HuConnect* qui doit obligatoirement être connectée en bluetooth au smartphone pour fonctionner. Si l'appareil n'est pas connecté (comme l'image ci contre), on a la possibilité de se rendre directement dans les menu Paramètre du téléphone pour l'activer. Ceci est géré par la classe nommée "*Peripherique*".

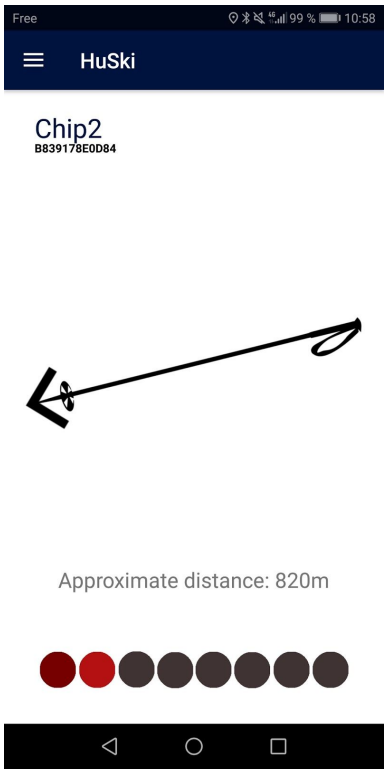
L'ajout de nouvelles *HuCard* se fait par scan de QR code via le bouton '+' (bas droite). En effet à chaque *HuCard* est associé un QR code correspondant à son adresse Mac. Ceci sert à savoir de quelles cartes récupérer les informations et quelles cartes ignorer.

Une fois la connexion bluetooth établie entre le smartphone et la *HuConnect* on obtient l'affichage suivant:



Une fois les composants connectés, on peut appuyer sur le bouton de localisation (droite) pour accéder à la seconde page majeure de notre système: la page de localisation du ski perdu.

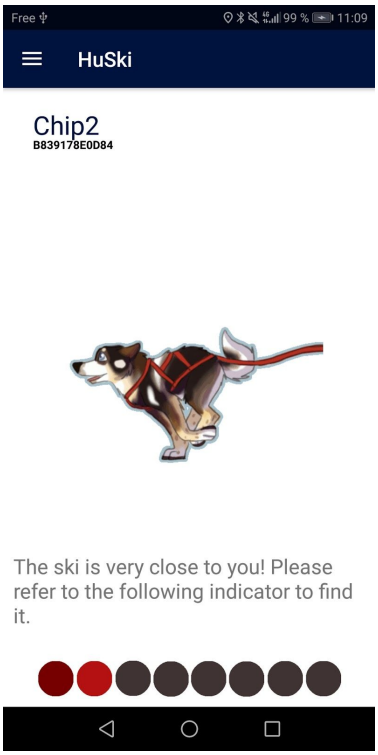
L'appui sur ce bouton envoi un signal à la *HuConnect* qui se charge de passer la *HuCard* concernée en mode actif (cette carte enverra donc des données toutes les secondes au lieu de toutes les 2 minutes). Ce mode sera désactivé lorsque l'utilisateur quittera la page de localisation ou fermera l'application.



Si le ski perdu est à une distance supérieure à 20 mètres (précision relative du GPS) on est guidés par une boussole (qu'il faut au préalable calibrer pendant cinq secondes avec une simple manipulation du téléphone). Le bâton de ski sur l'image ci-contre pointe vers la direction générale du ski perdu. Pour ce faire nous récupérons les coordonnées GPS de la *HuCard* concernée et les coordonnées GPS du smartphone de l'utilisateur. Grâce aux Sensor Events du smartphone nous sommes en mesure de déterminer son orientation et donc de toujours pointer le bon angle en direction du ski. Grâce à ces deux positions GPS il est également aisé de calculer une distance approximative entre les deux positions.

Si la distance est inférieure à 20 mètres nous ne pouvons plus utiliser les positions GPS comme repère, nous passons donc en mode RSSI. Ici c'est la jauge en bas de l'écran qui va vous guider pour retrouver votre ski.

Nous récupérons la force du signal RSSI correspondant à l'intensité du signal reçu par la gateway. Plus le signal est fort plus la *HuCard* et la *HuConnect* sont proches. C'est donc un système de chaud/froid.



En mode RSSI nous avons fait le choix de désactiver la boussole pour ne pas induire l'utilisateur en erreur. A la place un huski animé vous aidera à retrouver vos skis !
Une fois le ski retrouvé il suffit de quitter l'application ou la page de localisation pour envoyer un signal à la *Hucard* pour la remettre en mode normal (un envoi de paquet toutes les 2 minutes).

3. DevOps

Dans une optique d'amélioration de qualité de code, nous avons souhaité adopter une démarche orientée DevOps. Ceci se traduit par la mise en place d'une intégration continue, de tests automatisés, ainsi que la mise à disposition de l'apk signée

a. Pipeline

Après quelques tentatives infructueuses, nous nous sommes tournés vers la solution open source Jenkins pour notre intégration continue. Cela se présente sous la forme d'un serveur sur lequel nous allons définir une série d'actions à effectuer pour passer du code pure à une application signée.

La pipeline se définie comme suit :

- Une première étape permet de produire l'apk non signée en mode débug (cela nous servira pour les tests "Instrumented Tests", série de test propre à Android.
- Ensuite, nous lançons tous les tests.

- Nous assemblons une première version en mode debug, ce qui nous permet de la transmettre à l'outil FireBase Test Lab avec l'outil gcloud. Ceci nous permet de réaliser tous les InstrumentedTest (des tests qui tournent sur des devices physiques ou émulées et qui permettent de tester le bon fonctionnement de notre application avec les API)
- Une fois ces tests réalisés, cela nous permet de générer l'APK ainsi que de la signer

b. Qualité de code

En parallèle, nous avons décidé de se doter d'un outil nous permettant d'avoir des métriques de qualité de code. Nous avons d'abord décidé de mettre en place un serveur SonarQube. Devant le manque de place sur notre serveur, nous nous sommes donc tournés vers SonarCloud, avant de finalement opter pour l'outil Codacy. Cet outil nous permet de mettre en évidence toutes les mauvaises pratiques, éléments susceptibles de générer des erreurs, les problèmes de sécurité ainsi que de surveiller l'aspect visuel du code (dans le but de l'unifier).

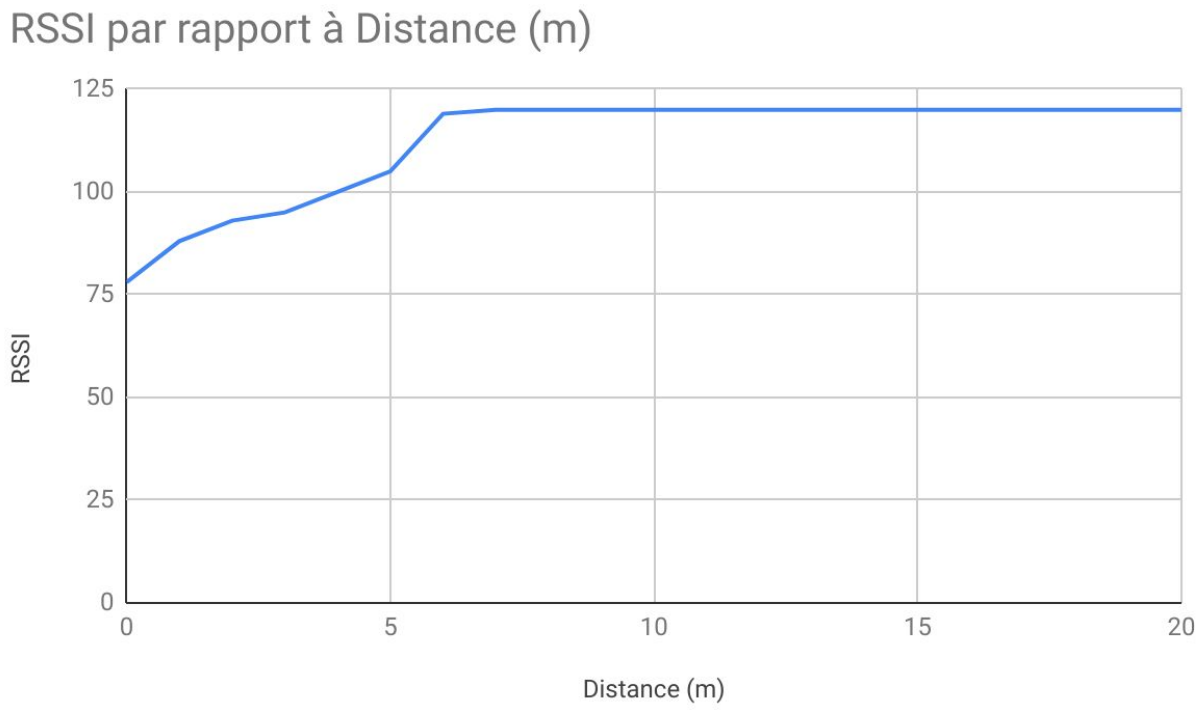
Actuellement, notre projet est classé dans la catégorie "B" et en dessous des 20% d'erreur, seuil auquel est fixé le "quality gate".

III -Tests

1.Conditions réels

Nous avons essayé le module GPS en extérieur et nous avons déterminé grâce à google maps une incertitude de 20m. Quant au RSSI, nous avons réalisé un test en intérieur, dans le bâtiment de Polytech.

Relevé des valeurs du RSSI par rapport à la distance:



On peut voir qu'après 6 ou 7 mètres, on atteint le niveau maximum du RSSI. Pour un produit final il faudrait avoir du matériel nous permettant d'avoir un RSSI maximum à plus ou moins 20 mètres de la HuCard puisque le GPS n'est pas assez précis pour nous permettre d'atteindre à 100% un périmètre de 5m autour de la carte.

IV - Difficultés rencontrées

Tout au long de ce projet, nous avons dû faire face à plusieurs problèmes techniques, que ce soit au niveau hardware ou software. Ces difficultés ont été, dans l'ensemble, surmontées assez rapidement sans nous faire perdre trop de temps.

Premièrement, nous avons dû faire face à quelques inconvénients électrique pour la carte : Le module GPS fonctionne avec une tension de 5V, les pins de la carte supportent 3.3 V et nos batteries délivrent 3.7V. Nous avons donc réalisé électroniquement un diviseur de tension et utiliser un régulateur de tension. Ensuite, nous avons eu des problèmes pour faire fonctionner le GPS au début du projet car nous avions peu de documentation sur comment l'utiliser. De l'aide sur des forums en ligne a été requis pour résoudre nos problèmes et nous avons été retardé de quelques jours. Nous avons également voulu mettre en place un chiffrement des communications car nous travaillons en LoRa point à point mais il a été difficile de trouver une librairie compatible avec notre carte esp32 et fonctionnelle avec notre protocole de communication.

Concernant le développement de l'application, une décision complexe et importante a dû être prise quant au choix de la technologie. Après avoir commencé à développer une application hybride sous Ionic, nous nous sommes rapidement rendu compte que l'implémentation de certaines fonctionnalités reposant sur l'hardware du téléphone allait être plus compliqué que prévu (boussole, bluetooth…). Il a donc été préférable de recommencer le développement sous Android Studio qui permet de créer des applications android natives et d'accéder plus simplement aux services du téléphones.

Un autre problème est survenu lorsque nous avons implémenté la partie connectivité Bluetooth de l'application. Le temps alloué à cette partie a été plus long que prévu (quelques jours). Au final cela n'a eu que très peu d'impact sur l'avancée du projet.

V - Ouvertures et Conclusion

Toute l'équipe est contente d'avoir collaborer sur ce projet et de l'avoir mené de la conception à la réalisation complète d'un prototype avec un boîtier. Cela nous a permis de travailler des compétences très variés qui vont du design au devops en passant par de la programmation pur.

Nous avons essayé le plus possible d'avoir une démarche incrémentale, un produit fiable et suffisamment documentés et évolutif pour les futurs étudiants qui reprendront éventuellement le projet.