

## PI\_MobilSens

Robust mobile sensing device

**Projet**

PI\_MobilSens

**Etudiants**

Adrien Rey, Robin Rittiner,  
Julien Fournier, Julien Michel,  
Mathieu Bourquenoud, Miguel Oliveira,  
John Isaac Wetenkamp,  
John Biselx, Sylvestre Van Kappel

**Professeurs**

Alexandra Andersson  
Medard Rieder

**Date**

20.06.2024

**Hes-SO**

Haute Ecole Spécialisée  
de Suisse occidentale  
Fachhochschule Westschweiz  
University of Applied Sciences and Arts  
Western Switzerland

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Contexte . . . . .	5
1.2	Objectifs . . . . .	5
1.2.1	Batterie . . . . .	5
1.2.2	Mécanique . . . . .	5
1.2.3	Électronique . . . . .	6
1.2.4	GNSS . . . . .	6
1.2.5	BLE Booster . . . . .	6
<b>2</b>	<b>Batterie</b>	
	<i>Adrien Rey</i>	<b>7</b>
2.1	Tadiran Lithium 2/3AA 3.6 V . . . . .	7
2.2	Tadiran Lithium 1/2AA 3.6 V . . . . .	8
2.3	Supercondensateur . . . . .	9
<b>3</b>	<b>Mécanique</b>	
	<i>Robin Rittiner, Julien Fournier</i>	<b>10</b>
3.1	Release système . . . . .	10
3.1.1	Variante 1 <i>Robin Rittiner</i> . . . . .	10
3.1.2	Variante 2 <i>Julien Fournier</i> . . . . .	12
3.1.3	Variante 3 <i>Julien Fournier</i> . . . . .	17
3.2	Boîtier . . . . .	18
3.2.1	Variante 1 <i>Robin Rittiner</i> . . . . .	18
3.2.2	Variante 2 <i>Julien Fournier</i> . . . . .	21
<b>4</b>	<b>Électronique</b>	
	<i>Julien Michel, Mathieu Bourquenoud, Miguel Oliveira</i>	<b>25</b>
4.1	Vibrations et chocs <i>Miguel Oliveira</i> . . . . .	25
4.1.1	Connecteur de la carte micro SD . . . . .	25
4.2	Design existant . . . . .	26
4.3	Variante électronique 1 <i>Julien Michel, Mathieu Bourquenoud</i> . . . . .	26
4.4	Variante électronique 2 <i>Miguel Oliveira</i> . . . . .	29

4.5	Évaluation d'un SoC LoRa <i>Julien Michel, Mathieu Bourquenoud</i>	32
4.5.1	Introduction	32
4.5.2	Contraintes	32
4.5.3	Microcontrôleur	32
4.5.4	Antenne	33
4.5.5	Conclusion	34
4.6	Évaluation de l'utilisation du eMMC <i>Julien Michel, Mathieu Bourquenoud</i>	34
4.6.1	Introduction	34
4.6.2	Contraintes	35
4.6.3	Composant	35
4.6.4	Conclusion	35
4.7	Interface GNSS	35

## 5 GNSS

	<i>John Isaac Wetenkamp, John Biselx</i>	<b>36</b>
5.1	Outils	36
5.1.1	Hardware	36
5.1.2	Software	38
5.2	Analyse de réception	39
5.2.1	M8	39
5.2.2	M10	40
5.3	Analyse de consommation	40
5.3.1	M8	40
5.3.2	M10	40
5.3.3	Choix module	41
5.3.4	Estimation de temps d'allumage du module GNSS	41
5.3.5	Protocoles de communication	42
5.4	Analyse de paramètres	42
5.4.1	Multiple GNSS assistance (MGA)	42
5.4.2	Mode basse-consommation	43
5.5	API UBX	43
5.5.1	Description	43
5.5.2	Problèmes rencontrés	43
5.5.3	Perspectives futures	44
5.6	Livrables	45

## 6 BLE Booster

	<i>Adrien Rey, Sylvestre van Kappel</i>	<b>46</b>
6.1	Introduction	46
6.2	Software	47
6.2.1	Architecture générale	47
6.2.2	Interface	48

6.2.3	Monkeylist :	52
6.2.4	Bluetooth Low Energy	53
6.2.5	Séquences	57
6.3	Hardware	59
6.4	Tests de portée	62
<b>7</b>	<b>Conclusion</b>	<b>64</b>
7.1	Batterie	64
7.2	Mécanique	64
7.3	Électronique	65
7.4	GNSS	65
7.5	BLE Booster	66
<b>8</b>	<b>Bibliographie</b>	<b>67</b>
<b>Annexes</b>		<b>70</b>
Batterie		70
Mécanique		70
Électronique		70
BLE Booster		70
GNSS		71

# 1 Introduction

## 1.1 Contexte

Ce projet vise à améliorer un dispositif de collier destiné à être porté par des singes pour enregistrer les sons qu'ils émettent.

L'objectif principal de ce projet est de réduire significativement le poids et la taille du collier. Cette réduction est cruciale pour minimiser l'impact sur les mouvements et le comportement naturel des singes, tout en maintenant, voire en améliorant, la capacité de l'appareil à enregistrer les sons de manière fiable et de maintenir le temps de fonctionnement du collier de 10 jours.

De plus, des problèmes sont apparus lors de l'utilisation de ces colliers sur le terrain. Leur fiabilité n'étant pas garantie, un objectif de ce projet est donc de régler ces problèmes pour rendre le système plus robuste.

En parallèle, le projet explore l'intégration d'un module GNSS pour permettre un suivi plus précis des déplacements des singes. Cette fonctionnalité supplémentaire offre la possibilité de corrélérer les données acoustiques avec les données de localisation, enrichissant ainsi la qualité et la profondeur des études comportementales.

Un autre aspect clé du projet est le développement d'un appareil de contrôle à distance. Le but principal de cet appareil est de pouvoir ouvrir le collier à distance, réduisant ainsi le stress pour les animaux et les risques de perturbation de leur comportement naturel. Ce contrôleur à distance exploitera la technologie Bluetooth Low Energy (BLE) pour assurer une communication efficace et économique en énergie entre le collier et le dispositif de contrôle.

## 1.2 Objectifs

### 1.2.1 Batterie

L'objectif principal de la section sur la batterie est de réduire la taille et le poids des colliers en remplaçant la batterie actuelle. En analysant la consommation d'énergie du système et en testant de nouvelles options de batteries, cette partie vise à identifier une solution plus compacte et légère sans compromettre la durée de fonctionnement des colliers.

### 1.2.2 Mécanique

Les objectifs de la partie mécanique consistent à proposer deux variantes de release system ainsi que deux variantes de boîtier capable de contenir toutes l'électronique nécessaire au bon fonctionnement du collier. Évidemment, le poids ainsi que l'encombrement sont deux éléments à réduire au maximum afin de garantir un confort pour les singes.

### 1.2.3 Électronique

L'objectif principal de la partie électronique est de concevoir les PCB contenant l'électronique embarquée du collier, en se basant sur le design existant de la première version du système. Le but étant de réduire la taille et le poids par rapport à la précédente version. Il est essentiel que la conception des PCB soit réalisée en étroite collaboration avec celle du boîtier. Comme pour la partie mécanique, deux variantes de design seront proposées, correspondant à chacun des boîtiers. En plus de la conception des nouveaux PCB, une solution doit être trouvée pour améliorer la robustesse du système de la carte SD face aux vibrations ainsi que l'ajout de pin de programmation.

### 1.2.4 GNSS

Les objectifs de la partie GNSS est d'explorer la possibilité d'ajouter un module GNSS au collier. Des tests de réception et de consommation de courant seront faits pour déterminer la faisabilité. De plus, l'efficacité énergétique de différents protocoles de communication est analysée et différents implémentations de code seront examinés.

### 1.2.5 BLE Booster

L'objectif principal de la section dédiée au BLE Booster est de développer et tester une solution permettant d'étendre la portée de communication des colliers. En utilisant la technologie Bluetooth Low Energy (BLE) et en intégrant une antenne directionnelle et un module Front-End (FEM), cette partie vise à surmonter les limitations de distance du smartphone actuellement utilisé pour cela. L'innovation apportée par le BLE Booster doit permettre d'assurer une communication plus fiable et efficace sur le terrain.

## 2 Batterie

Adrien Rey

Un des objectifs de ce projet est de diminuer la taille et le poids du collier actuelle. Avec les retours d'expérience, il est ressorti que la batterie utilisée était surdimensionnée par rapport à la consommation du système. L'utilisation de ce dernier étant limité par la taille de la carte SD à environ 12 jours.

Le système actuel utilisait une batterie *Tadiran Lithium AA 3.6 V (SL-360)*. Deux modèles ont été proposés pour la remplacer : la *Tadiran Lithium 1/2AA 3.6 V (SL-361)* ou *Tadiran Lithium 2/3AA 3.6 V (SL-761)* (les datasheets se trouvent en annexes). L'objectif étant de diminuer un maximum le poids et la taille du boîtier, il serait préférable d'utiliser la batterie 1/2AA (voir Figure 1). Toutefois, il a fallu tester les batteries pour s'assurer qu'elles étaient capables de fournir l'énergie nécessaire. Pour ce faire, deux colliers existants ont été adapté pour recevoir les nouvelles batteries. L'enregistrement à ensuite été lancé sur les deux systèmes.

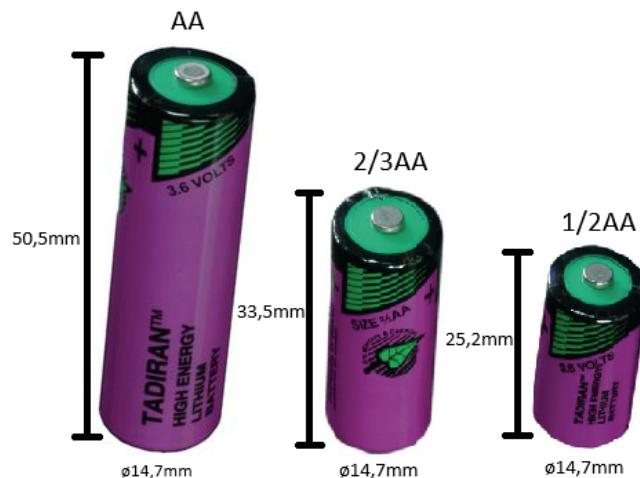


FIGURE 1 – Taille des batteries

### 2.1 Tadiran Lithium 2/3AA 3.6 V

Les système avec la batterie 2/3AA a tenu les 12 jours d'enregistrements. Le système c'est mis en power saving mode suite au remplissage de la carte SD (état *Disk Full*). La tension de la batterie à la fin du test était de 3.65 V. Ce modèle de batterie peut donc être utilisé pour le nouveau système. Cette solution nous permet de gagner 17mm sur la hauteur de la batterie et 6 grammes par rapport au modèle utiliser précédemment (voir figure 1).

## 2.2 Tadiran Lithium 1/2AA 3.6 V

Les système avec la batterie 1/2AA n'a jamais réussi à tenir les 12 jours d'enregistrements. L'enregistrement c'est toujours arrêté avant le remplissage de la carte SD. Le test a été relancé plusieurs fois et sur un collier différent pour s'assurer que le problème ne venait pas de l'électronique. Il est difficile de connaître la cause exacte. Le test se déroulant sur une longue période, il est difficile de le surveiller.

Une cause probable est le courant de décharge continue maximum de la batterie. Pour le modèle 1/2AA, le courant de décharge continue maximum n'est que de 6 mA alors que pour la batterie AA et 2/3AA il est respectivement de 20 mA et 75 mA. Le système nécessite un courant supérieur à ce que la batterie 1/2AA peut fournir en continu lors des phases d'écriture sur la carte SD. Cela conduit à des chutes de tension et donc des resets du système. La consommation du système en mode recording est montré dans la figure 2.

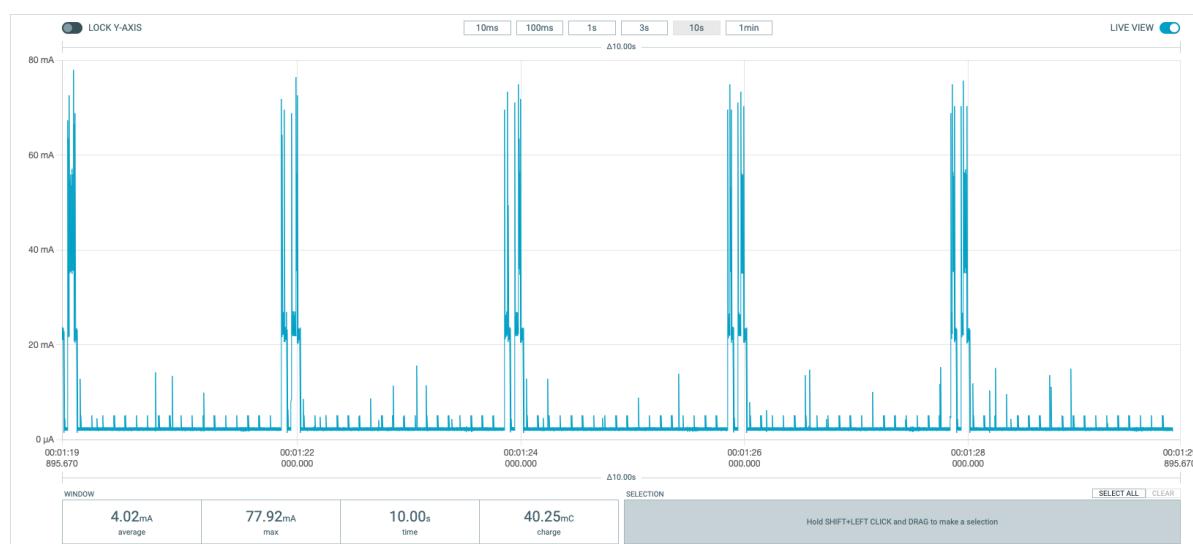


FIGURE 2 – Consommation en mode recording

Lors des tests, le système fonctionnait quand même pendant quelques jours (2-3 jours) avant de se remettre à zéro. Cela est dû à la mise en place d'un supercondensateur en parallèle de la batterie qui a justement pour but d'éviter les chutes de tension. Ce condensateur pouvait compenser ces chutes de tension un certain temps mais pas indéfiniment. Ce modèle de batterie peut donc pas être utilisé pour le nouveau système.

## 2.3 Supercondensateur

Lors de l'utilisation des colliers sur le terrain, beaucoup d'entre eux ont subit un reset qui pourrait être dû à un pic de consommation ou une chute de tension. Pour palier à ce problème, il a été décidé d'ajouter, en parallèle de la batterie, un autre supercondensateur hybrides LIC VMF en plus de celui déjà présent de 25 F. Ces condensateurs offrent une plus grande densité d'énergie, un faible taux d'auto-décharge et une charge ultra-rapide. Ils doivent permettre de :

- fournir des courants très élevés pendant de courtes périodes.
- stabiliser la tension lors de courants de pointe
- augmenter la durée de vie de la batterie en prenant en charge les demandes de courant de pointe

Le modèle VMF de 10 F de CDE a été choisis pour notre application. Les tests effectués sur la batterie 1/2AA et 2/3AA ont été réalisé avec le supercondensateur monté. Cela valide donc l'utilisation de ce dispositif dans le nouveaux système.

### 3 Mécanique

Robin Rittiner, Julien Fournier

Dans ce chapitre, il est question de toute la conception mécanique relative au collier de singe. Il est composé de 2 parties distinctes : le système de relâche du collier (Release System) et l'intégration de toute la partie électronique dans le boîtier. Il a été décidé de concevoir deux variantes de boîtier pour offrir un choix plus large et élargir les perspectives de solution pour le collier.

#### 3.1 Release système

Le release system doit permettre à l'utilisateur de détacher le collier du singe sans avoir besoin d'endormir le singe et de le faire manuellement. La commande est pilotée à distance grâce à un système Bluetooth. La difficulté est de permettre le relâchement du système avec le moins d'énergie possible, car l'énergie disponible. Voici les deux versions de release system :

##### 3.1.1 Variante 1

Robin Rittiner

La première variante est une version évolué de la première version du système de release (voir figure 3). Les deux lanières sont maintenues grâce à un fil en nylon. Lorsque la commande d'ouverture du collier intervient, le fil en cuivre qui est enroulé autour du fil en nylon est traversé par un fort courant, ce qui fait chauffer le fil et donc fondre le nylon. Un ressort permet de s'assurer que les deux parties se séparent.

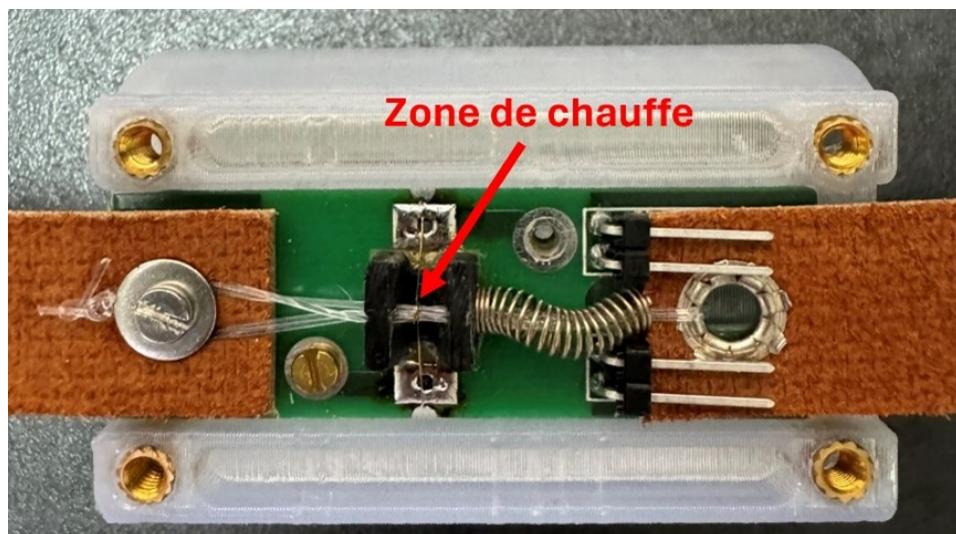


FIGURE 3 – Release system de la première version du collier

La nouvelle variante s'inspire du fil en nylon et du ressort pour sa conception. L'idée générale nous a été présentée par M. Medard Rieder. Il a fallu cependant apporter certaines modifications afin de le rendre plus compact (voir figure 4). Une goupille fait office d'ardillon (comme pour une ceinture). Elle est maintenue en positon fermé grâce au fil en nylon qui est tenu tendu grâce à une vis M2.5. Lorsque l'on veut ouvrir le système, le fil en cuivre chauffe et rompt le fil en nylon. Pour faire reculer l'ardillon et libérer la lanière, un ressort de compression est ajouté pour générer assez de force de recule. Le point fort de cette variante est de permettre de stocker beaucoup d'énergie dans le ressort indépendamment de l'énergie que l'on utilise pour libérer le système. Il est en revanche assez contraignant au montage.

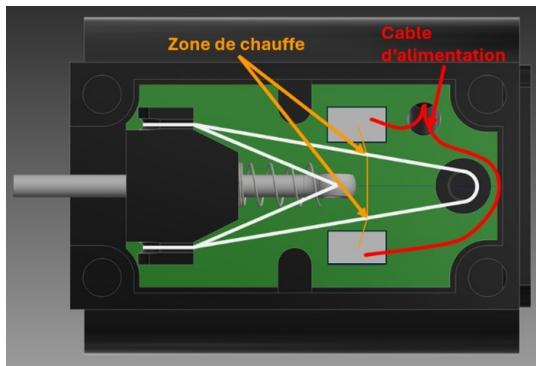


FIGURE 4 – Variante 1 du release system

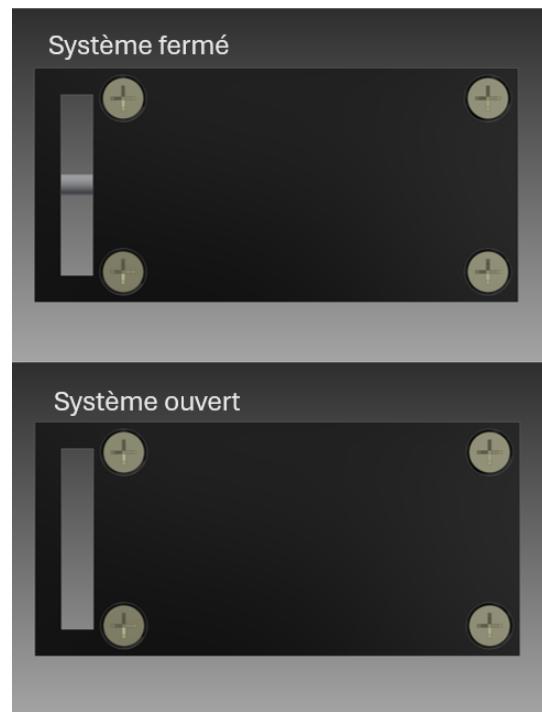


FIGURE 5 – Ouverture et fermeture de l'ardillon

### 3.1.2 Variante 2

*Julien Fournier*

Le système à fil chaud ayant comme grand désavantage une mise en place et un montage très laborieux, la recherche d'une idée se voulant facile à implémenter et réutilisable sans intervention supplémentaire est lancée. Il en ressort comme solution, à la base du système, un actuateur électromagnétique. En effet, pour pouvoir fournir une force suffisante dans un encombrement restreint, cela est un moyen optimal. Un mini moteur serait trop encombrant et trop coûteux, tandis qu'un piézoélectrique bien que très petit n'assurerait pas une force et une course suffisante. Une recherche d'un actuateur le moins encombrant possible fonctionnant à la tension disponible de 3 V CC a fait ressortir le composant existant suivant (voir figure 6).



FIGURE 6 – Solénoïde, Delta Electronics, DSOS-0416-03D, DigiKey

La conception du mécanisme complet s'est voulue la plus simple possible afin d'optimiser la production, le montage et la robustesse du système (voir figure 7).

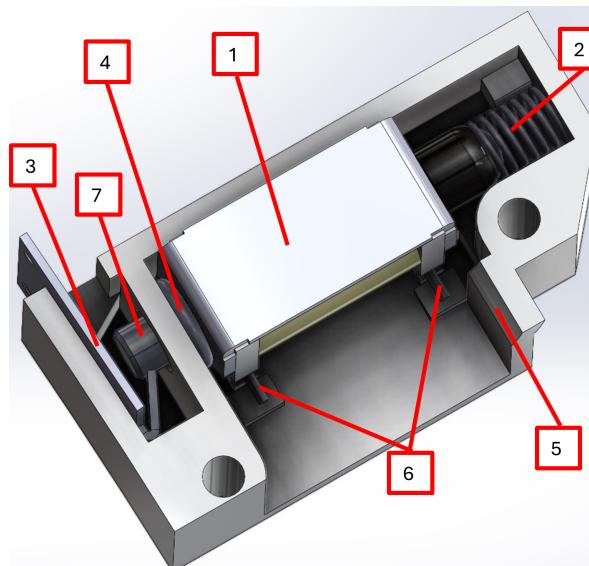


FIGURE 7 – Release system composé de l'actuateur existant

L'actuateur (1) prend place dans sa zone de boîtier dédiée (5) dans laquelle il y est calé. La tige mobile du solénoïde (7), lorsque celui-ci n'est pas alimenté, est en position sortie et retient une tôle pliée (3) sur laquelle sera fixé le collier. En amenant la tension électrique nécessaire aux bornes (6) la tige coulisse et laisse la plaque métallique librement glisser, guidée par le boîtier, de part le poids du dispositif et les mouvements du singe, et ainsi permet au collier de se libérer. Un trou dans la paroi étant obligatoire pour laisser passer la tige et assurer le rôle de palier, un joint torique (4) est monté pour assurer l'étanchéité. Il est choisi avec un diamètre intérieur de même dimension que l'arbre (4 mm) afin de venir juste effleurer la surface et minimiser le frottement. Un ressort de rappel (2) est ajouté pour aider le ressort intégré au composant à réussir à retourner la tige en position de blocage, opération plus difficile due à la force de frottement supplémentaire engendrée par le contact du joint. La rigidité du ressort a toutefois été choisie suffisamment faible pour que l'actuateur arrive toujours à s'ouvrir. La force maximale a été définie selon la figure 9 tiré de la fiche technique. A la force linéaire active à mi-course est soustraite la force du ressort interne et une marge estimée pour la force de frottement résistante, ce qui laisse la force calculée en figure 8.

$$F_{res} = \frac{50 \cdot 9.81}{1000} - \frac{7.5 \cdot 9.81}{1000} - 0.2 = 0.217 \text{ N}$$

FIGURE 8 – Force de ressort nécessaire estimée

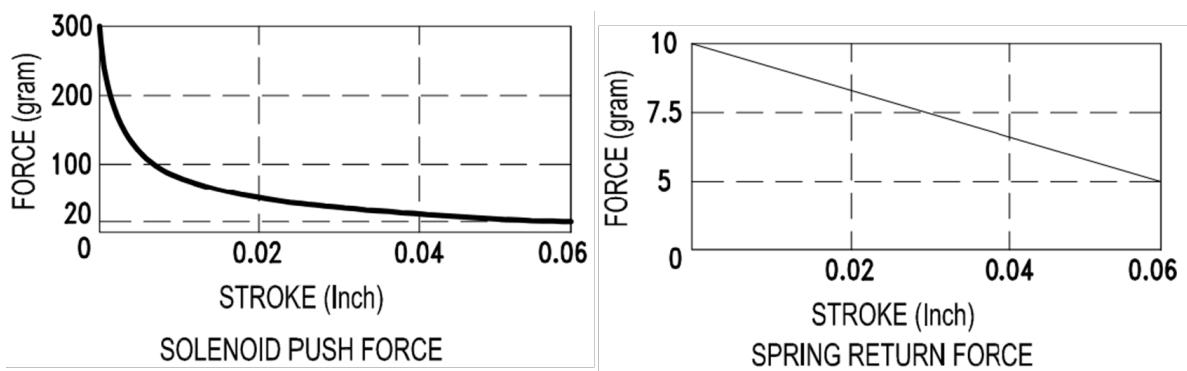


FIGURE 9 – Graphiques des forces de l'électro-aimant

Le ressort qui a été choisi est le WY4-5 fourni par Misumi avec une constante de ressort de 0.1 N/mm. Les efforts de frottements ne pouvant pas être bien estimés une série de prototypage a été effectuée afin de tester le dispositif avec différents joints et ressorts. Il en est sorti un résultat non satisfaisant. Bien que le verrouillage s'est avéré performant et robuste, ce n'était pas le cas pour le déverrouillage. Même sans ressort de rappel et sans joint, la résistance due au serrage du collier provoquant une pression de la tôle pliée sur la tige suffisait tout juste à empêcher l'ouverture. Il était alors évident que le solénoïde n'était pas assez puissant. Malheureusement aucun composant possédant une force légèrement plus grande dans les dimensions et voltages souhaités n'a été trouvé.

Une considération pour un actuateur fait maison a toutefois été émise afin que l'idée puisse éventuellement être reprise dans une étude de développement ultérieur du produit. Le principe du dispositif ainsi que sa mise-en-œuvre sont très simples c'est pourquoi une adaptation sur mesure peut s'avérer intéressante. Ce système est fait d'un fil de bobinage enroulé un certainement nombre de tours autour d'une tige mobile en matériau ferromagnétique doux (pas d'aimantation rémanente après désactivation du champs) comme de la ferrite. Lorsque qu'un courant électrique continu traverse le fil un champs magnétique va être créé, ce qui va magnétiser la tige. Les deux champs magnétiques vont se repousser et ainsi provoquer une force de répulsion qui va entraîner le coulissemement. La force sera annulée dès lors que le courant sera stoppé. L'ampleur de cette force est proportionnelle au produit au carré du courant et du nombre de tours. Les formules qui suivent définissent cela, avec  $N[-]$  le nombre de tours de la bobine ;  $i[A]$  le courant,  $\mu_0$  la perméabilité magnétique du vide,  $B_0[T]$  l'induction magnétique dans l'axe,  $l[m]$  la longueur sur laquelle est répartie la bobine,  $D[m]$  le diamètre moyen de la bobine,  $F[N]$  la force maximale exercée, et  $A[m^2]$  la section du noyau ferromagnétique (voir figure 10).

$$F = \frac{B_0^2 \cdot A}{2 \cdot \mu_0} [N], \quad B_0 = \frac{\mu_0 \cdot N \cdot i}{\sqrt{l^2 + D^2}} [T]$$

FIGURE 10 – Formule de la force d'un électro-aimant

Après démontage de l'électroaimant existant il ressort les éléments suivants dans la figure 11. En (1) le noyau mobile, en (2) le ressort intégré, en (3) la structure du composant, en (4) le noyau fixe en plastique, et en (5) le fil de bobinage.

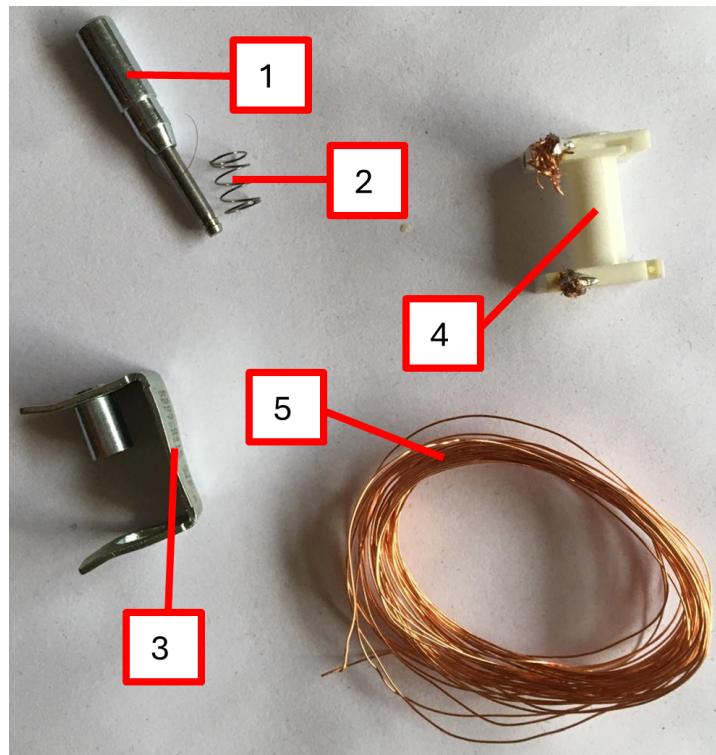


FIGURE 11 – Actuateur démonté

La première chose qui ressort est le fait que la partie réellement ferromagnétique du noyau mobile paraît être la tige de plus petite section (2 mm). Cela est déduit de part les couleurs des matériaux, mais reste à être vérifié. Dans cette hypothèse cela signifie qu'il n'y a pas toute la section disponible (4 mm) potentiellement active qui est utilisée. Un premier changement consisterait simplement à conserver uniquement le noyau fixe bobiné et d'y intégrer une tige totalement ferromagnétique de section 4 mm sur toute sa longueur. Le tout serait placé dans le boîtier dans lequel le palier serait directement imprimé pour remplacer celui qui se trouvait de base sur la carcasse de l'actuateur. La force maximale étant directement proportionnelle à la section du noyau et donc au carré du rayon, cela aurait déjà comme effet de **quadrupler la force**. Au cas où ce ne serait pas suffisant un agrandissement du bobinage, autour d'un noyau fixe dessiné sur mesure, pourrait être fait en plus. Le modèle de base possède une bobine de fil 0.2 mm de diamètre (0.25 mm en comptant la couche isolante), un diamètre moyen de 6.175 mm, répartie sur une longueur de 10 mm, avec 188 tours d'enroulement, et une résistance de 2.3 Ω. La force maximale théorique, avec son noyau de 2 mm de diamètre, est de **1.04 N** (**4.16 N avec tige de 4 mm**). Dans la fiche technique elle est annoncée à **2.943 N**, ce qui signifie que certaines plus grandes sections du noyau sont ferromagnétiques également. En essayant d'aller à la limite du raisonnable en terme d'encombrement, et en gardant la même résistance électrique pour travailler avec le même courant, voici les caractéristiques de la bobine sur mesure : un fil de 0.3 mm de diamètre (0.35 en comptant la couche isolante), un diamètre moyen de 8 mm, répartie sur une longueur de

15 mm, avec 367 tours d'enroulement. La force maximale théorique qui en découle, avec un noyau de 4 mm de diamètre, est de **7.6 N**. En théorie la force maximale pourrait être **2.6 fois** plus élevée (**7.3 fois** plus élevée si la tige d'origine est considérée à 2 mm). Tout cela au prix d'un encombrement de 5 mm de plus en longueur et 4 mm de plus en largeur (voir figure 12).

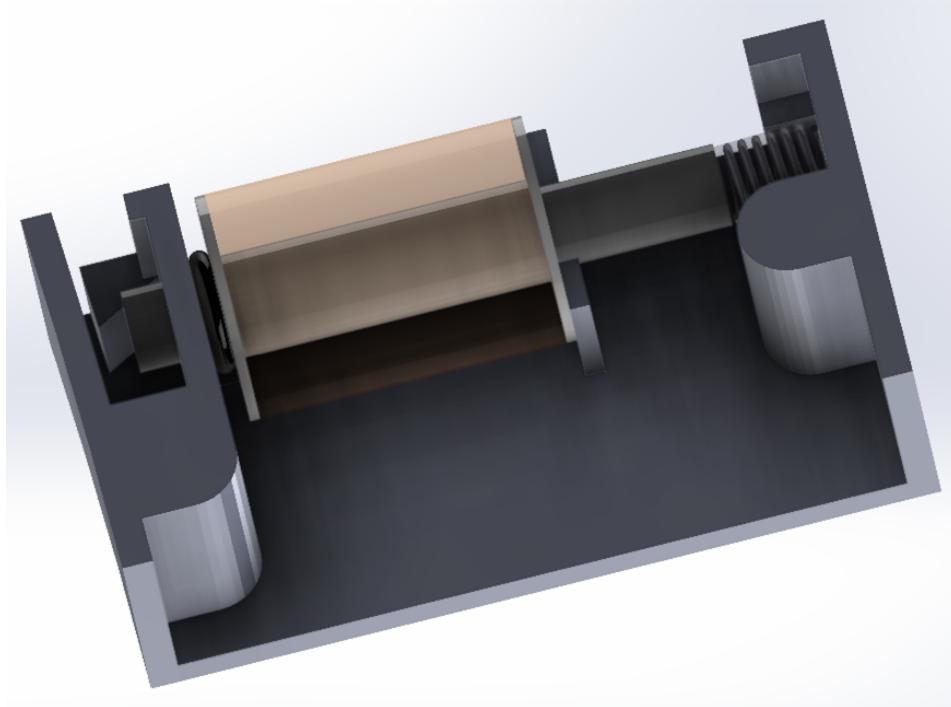


FIGURE 12 – Actuateur fait maison

Le potentiel de ce système est donc bien présent. Il faudrait effectuer plusieurs tests avec d'abord une simple modification du noyau, ce qui serait le plus pratique en terme de fabrication, puis si cela n'est pas suffisant un ajustement du bobinage.

### 3.1.3 Variante 3

*Julien Fournier*

Etant donné le temps à disposition et du fait qu'il fallait une version fonctionnelle en fin de travail, la solution du fil chaud a été choisie afin d'assurer le coup et de pouvoir se concentrer sur le design du boîtier. De légères modifications ont été apportées pour s'adapter à la version de boîtier prévue à la base pour l'actuateur. La taille a encore été miniaturisée à l'extrême, la goupille et le ressort ont encore été raccourcis, la base du socle sur laquelle viennent se fixer les composants est cette fois-ci intégrée directement à la paroi du boîtier afin de rendre moins laborieux le montage, et des évidements ont été fait sur le tendeur pour s'adapter à la taille figure 13).

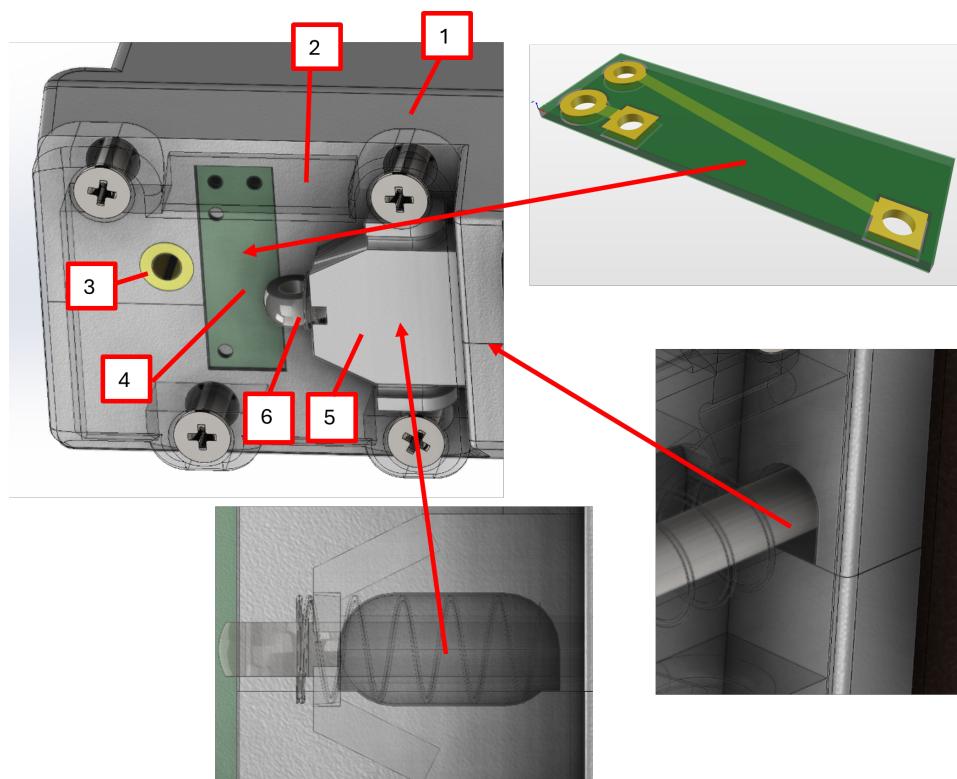


FIGURE 13 – Troisième variante du release system

Sur l'image ci-dessus, on y trouve en (1) le corps du boîtier, en (2) le couvercle du release system, en (3) l'insert pour la vis de fixation du fil nylon, en (4) le PCB miniaturisé accueillant les deux fils d'alimentation provenant de l'intérieur du boîtier et le composant résistif chauffant, en (5) le tendeur tenu en place par l'évidement oblong, et en (6) l'ensemble goupille et ressort qui traverse le boîtier et collier via une ouverture dédiée. Le oblong ainsi que l'ouverture pour la goupille sont séparés en deux, ce qui devrait permettre un montage facilité. Il est sûr qu'une telle miniaturisation est risquée, il faudra donc encore effectuer une phase de test pour valider son efficacité de fonctionnement.

### 3.2 Boîtier

Le boîtier doit permettre d'y ranger toute la partie électronique ce qui correspond aux différents PCB, la batterie, le câblage, etc. Il doit être résistant aux chocs et ne doit pas se dégrader durant toute la durée du test. Le boîtier doit aussi garantir une étanchéité afin d'éviter tout contact entre de l'eau et les composants électroniques. Un accès à la carte SD doit être prévu afin d'extraire les données et la batterie doit aussi rester accessible afin de permettre son changement.

#### 3.2.1 Variante 1

*Robin Rittiner*

Afin de diminuer au maximum le volume du boîtier, les PCB sont disposés tout autour de la batterie car c'est l'élément le plus contraignant en termes de taille, ce qui détermine la taille générale du boîtier (voir figure 14). De plus, le boîtier doit être compatible avec le système de release présenté précédemment (variante 1).

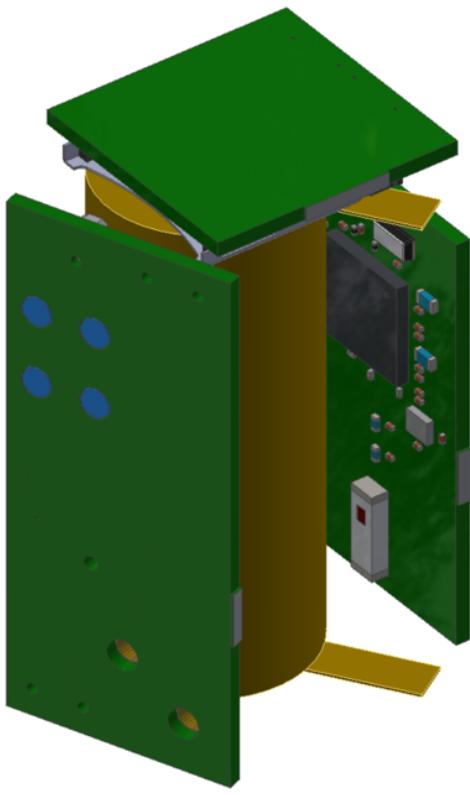


FIGURE 14 – Positionnement des PCB  
autour de la batterie

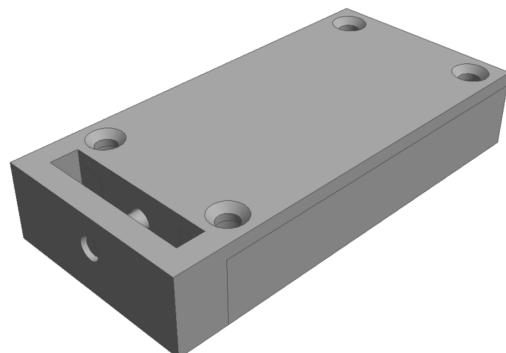


FIGURE 15 – Release system

Les deux PCB sur le côté (micro et communication) sont fixes et ne doivent pas nécessairement pouvoir bouger (voir figure 16). C'est pourquoi il est prévu de les coller. La liaison entre ces deux PCB se fait avec un PCB flexible visible en violet sur la figure 16. De la place a aussi été prévue pour les deux condensateurs. La batterie est maintenue latéralement grâce à un guide directement intégré dans le fond du boîtier (voir figure 17). Un point de colle est prévu pour la maintenir au fond du boîtier.

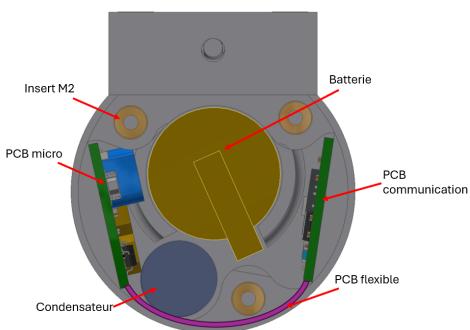


FIGURE 16 – Vue de dessus du boîtier avec composants

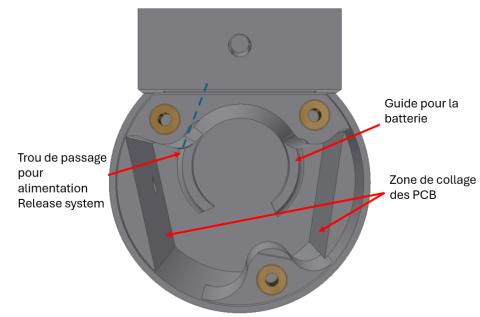


FIGURE 17 – Vue de dessus du boîtier sans composants

Afin de gagner de l'espace, le PCB de la carte SD est directement fixé sur le couvercle du boîtier. Cela permet aussi, lors de l'ouverture du couvercle, d'avoir accès au lecteur de carte SD tout en gagnant un maximum de place. La connexion avec les autres PCB se fait aussi à l'aide de PCB flexible (en vert sur la figure 19). Un trou est prévu pour permettre l'enregistrement audio avec le micro. L'étanchéité au niveau du trou et du couvercle se fait simplement à l'aide de colle.

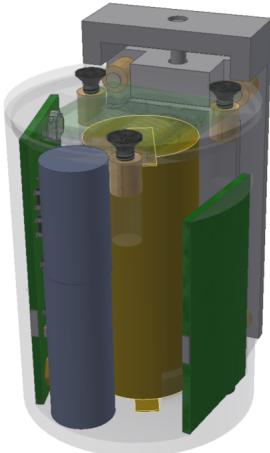


FIGURE 18 – Boîtier fermé en transparent

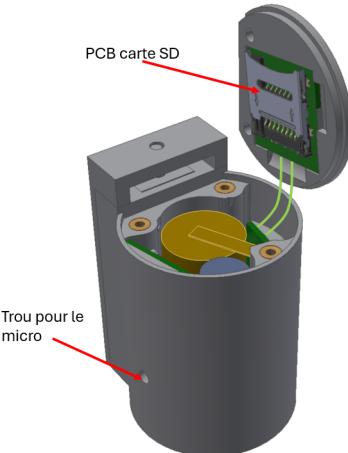


FIGURE 19 – Boîtier ouvert avec tous les composants

Afin d'assurer la fixation et le réglage de la lanière en cuir, une simple languette métallique prise entre deux vis permet de régler la taille du collier en fonction du singe (voir figure 20). La fixation de l'autre côté se fait comme présenté pour le release system, c'est-à-dire comme un système de boucle et de ceinture.

Pour finir, on peut voir en figure 21) la différence de dimension entre l'ancien design et la variante de boîtier n°1. Le gain en hauteur est de 25 mm et le gain en rayon est de 12 mm.

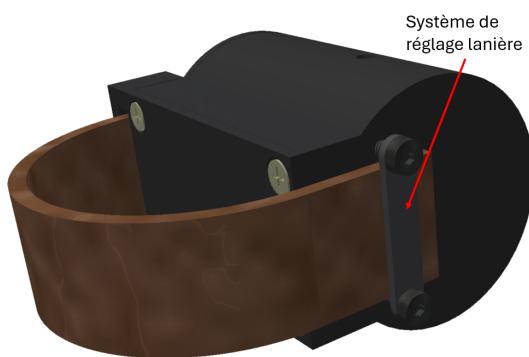


FIGURE 20 – Serrage et réglage de la lanière

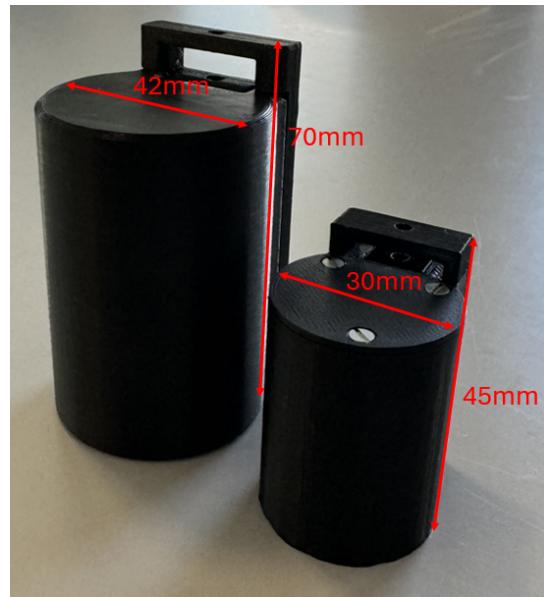


FIGURE 21 – Comparatif avec l'ancien design

### 3.2.2 Variante 2

*Julien Fournier*

Cette version de boîtier s'est basée sur le souhait d'optimiser la facilité d'accès aux composants, la facilité de montage, et la robustesse de l'assemblage, le tout dans un encombrement le plus faible possible. Afin d'assurer une bonne résistance mécanique et une bonne tenue aux UV le matériau choisi est l'ABS. Cependant pour des raisons de facilité d'impression les prototypes de développement sont imprimés en PLA. Le plan de travail a consisté à effectuer un premier design adapté à l'actuateur, ce qui constituait la première itération, puis à le modifier pour accueillir la solution du fil chaud, et enfin effectuer un certain nombre d'itérations successives synchronisées avec le développement électronique des PCB. La première itération, qui ne sera pas détaillée (voir figure 22).

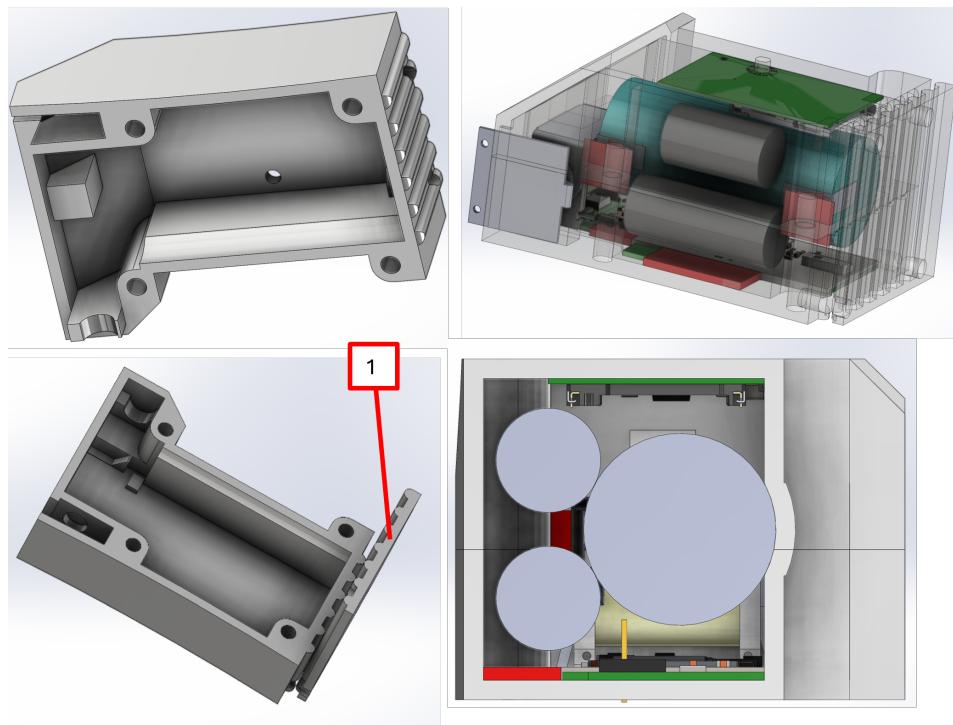


FIGURE 22 – Première itération du boîtier

A noter la présence d'un système de serrage de collier par une plaque ondulée (**1**) vissée sur le côté du boîtier qui peut être serrée lorsque le collier est ajusté à souhait. Ce système a été abandonné du fait du besoin de faire entrer à cet endroit le câble plat du GNSS intégré à la lanière. Cela nécessite que l'élément soit fixe et non ajustable. Le système de serrage restera donc comme celui existant (collier séparé en deux).

Après toutes les itérations le design final extérieur est représenté avec son encombrement général dans la figure 23.

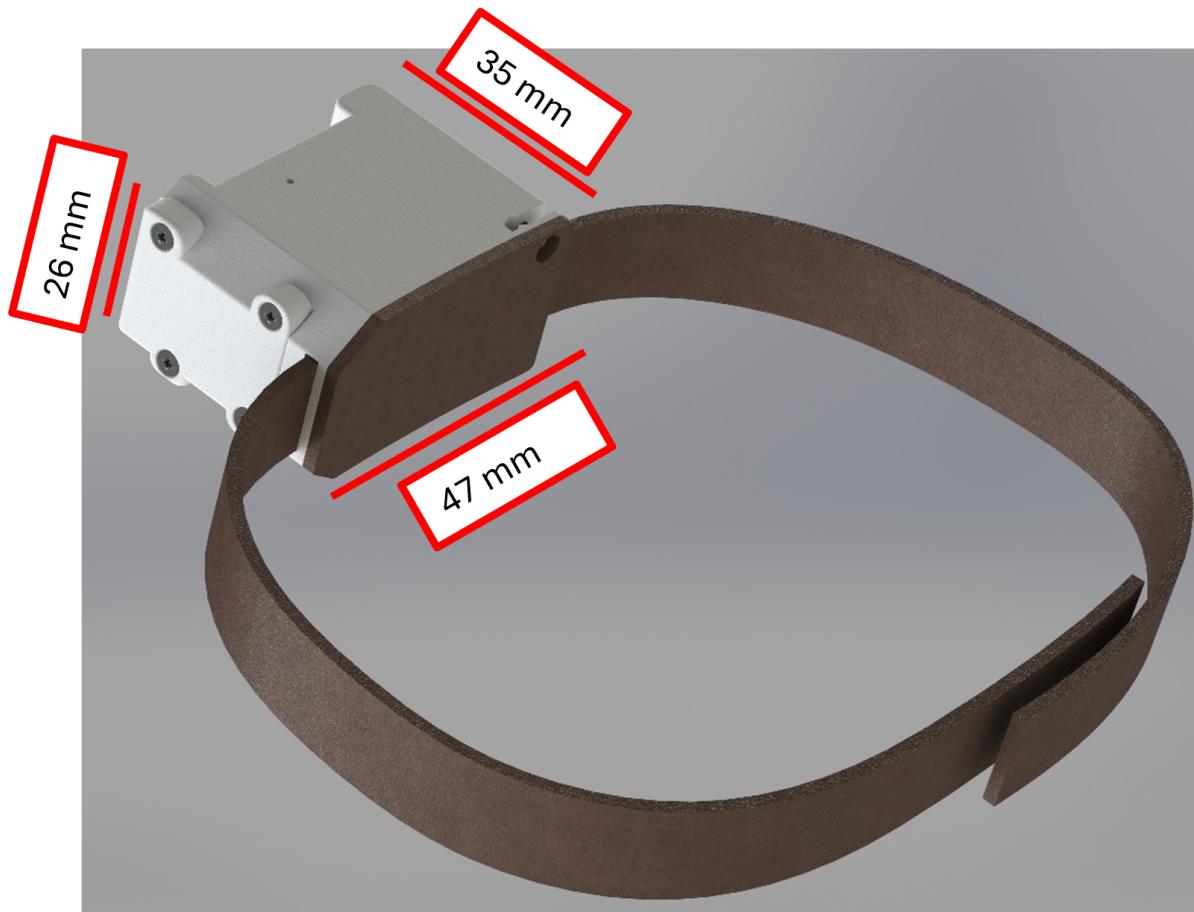


FIGURE 23 – Vue globale du système complet final

Dans la figure 24 ci dessous : (1) les protections en cuire pour le cou du singe, en (2) le PCB principal avec l'antenne et la grande super-cap, en (3) le joint torique d'étanchéité de diamètre extérieur 37.1 mm et d'épaisseur 1.6 mm, en (4) la pile, en (5) des ressorts de contact et de maintien de la pile sur lesquelles seront soudés les fils d'alimentation principaux du système, en (6) les vis à tête conique M2x16 pour la fermeture du boîtier, en (7) les inserts M2 en laiton à monter à chaud, en (8) le système de fixation du collier, en (9) les vis à tête conique M2x8 pour la fermeture du release system, en (10) le release system, en (11) l'ouverture pour le bouton reset dans laquelle sera collé une membrane en caoutchouc flexible pour assurer l'étanchéité, en (12) le PCB de la carte SD avec la petite super-cap, et en (13) une ouverture de passage pour le câble plat du GNSS.

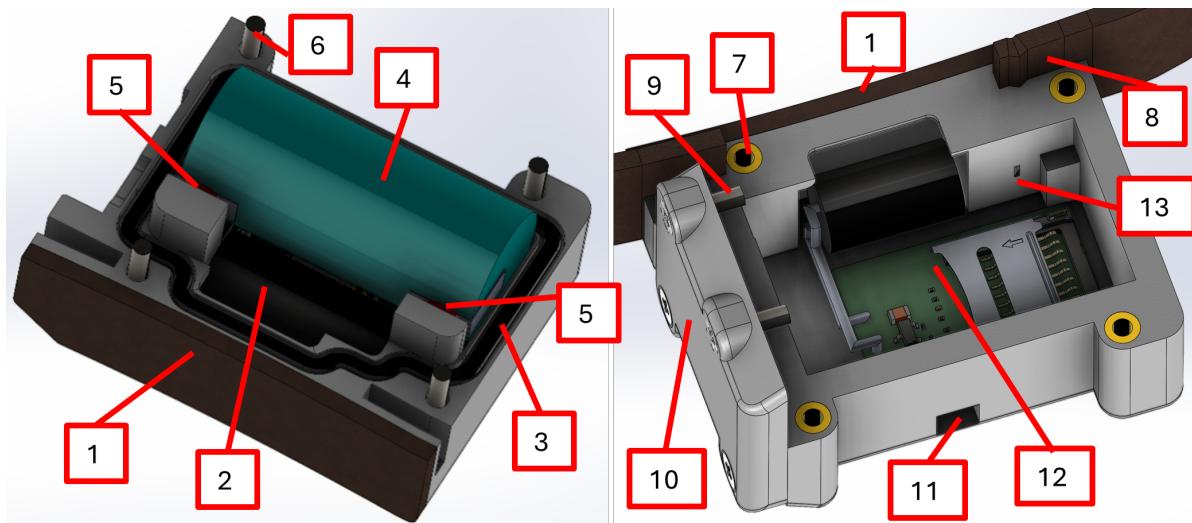


FIGURE 24 – Vue des deux parties du boîtier avec leurs composants

Le système de fixation du collier peut être visualisé plus en détail dans la figure 25 : avec en (1) l'écrou M2, en (2) la vis de serrage, et en (3) l'arrêté de blocage.

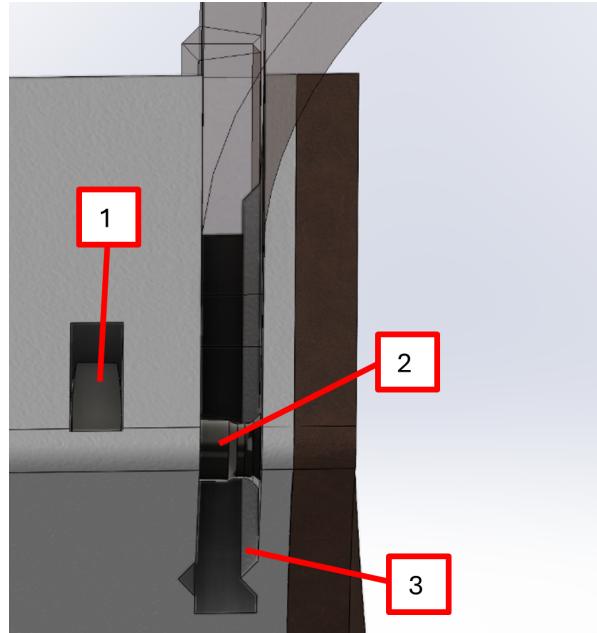


FIGURE 25 – Vue détaillée du système de fixation du collier

Enfin une dernière visualisation en coupe montre l'agencement intérieur du système lorsque le boîtier est fermé (voir figure 26). A noter la nappe flexible (1) de 50 mm de long assurant la liaison entre les deux PCB. Sa longueur permet d'assurer un jeu suffisant pour une ouverture confortable du boîtier.

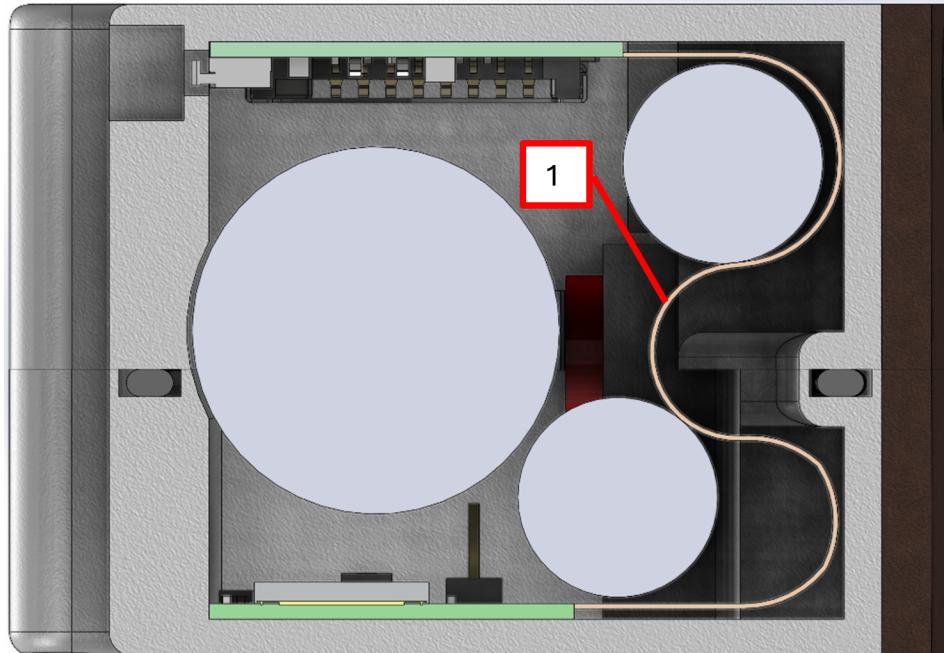


FIGURE 26 – Vue en coupe du système complet final

En terme d'encombrement l'objectif est rempli avec des dimensions finales étant très proches de la première variante et passablement plus faibles que le modèle existant. En terme de masse le système (sans le collier) arrive à une valeur estimée (les cartes électroniques ont été modélisées en matériau ABS, ce qui diffère quelque peu de la masse réelle, mais l'estimation reste bonne) de **43.47 g**. En prenant en compte le collier, toutefois en négligeant la masse du GNSS, avec un diamètre de tête de singe estimé à 10 cm, la masse monte à environ **52.87 g**. Cela est tout à fait satisfaisant sachant que la masse visée était de 40 g et que le modèle existant pèse 80 g.

## 4 Électronique

*Julien Michel, Mathieu Bourquenoud, Miguel Oliveira*

À l'instar des différentes variantes de conception mécanique, deux conceptions de cartes électroniques ont été développées. Chacune a été conçue spécifiquement pour correspondre à une variante mécanique particulière avec laquelle elle doit s'intégrer.

À noter que le design des PCB s'est fait, pour les deux variantes, en étroite collaboration avec la conception mécanique du boîtier correspondant, afin que les parties mécaniques et électroniques soient le plus adaptées possible dans les deux sens.

### 4.1 Vibrations et chocs

*Miguel Oliveira*

Le singe vervet est un animal actif qui saute et bouge considérablement. Par conséquent, celui-ci génère d'importants chocs et vibrations au sein du boîtier. Électriquement, le risque est que des déconnexions momentanées au niveau des connecteurs. Ceci concerne tous les composants interchangeables, comme la pile et la carte. Ces derniers étant critiques au bon fonctionnement de l'appareil, il est d'autant plus important de sécuriser les connexions.

#### 4.1.1 Connecteur de la carte micro SD

Dans le cas de l'enregistrement des données sur la carte SD, les chocs et vibrations peuvent dégrader considérablement ses performances et son bon fonctionnement. Le connecteur de carte SD utilisé actuellement a un mécanisme de chargement et d'éjection de la carte SD de type Push/Push. Bien que celui-ci soit intuitif pour l'utilisateur, il a comme désavantage de souffrir d'une faible résistance aux chocs et aux vibrations (voir figure 27).



FIGURE 27 – Connecteur de carte SD :

- (a) 47352-1001, Push / Push
- (b) MEM2067-02-180-00-A, articulated

Le connecteur de carte SD, MEM2067, est de style articulé. La charnière, bien que moins pratique que le système push-push, assure un contact sécurisé lors du verrouillage de la carte SD.

Sur les datasheets de deux supports, nous remarquons que le temps de discontinuité électrique lors de vibrations ou de chocs est garanti à maximum 1 ms pour le connecteur Molex, contre 1  $\mu$ s pour le connecteur MEM2067. Un compromis doit être fait sur l'encombrement de ce support. En effet, il faut laisser un espace libre au-dessus du composant pour ouvrir et fermer la charnière.

## 4.2 Design existant

Les deux variantes du design électronique sont basées sur la version existante du précédent design. Celui-ci comprenait deux PCB : le premier pour l'antenne, le micro-contrôleur, le micro et la carte SD, le second pour l'alimentation et l'électronique du “burn wire”, leurs dimensions étant 55 x 15 mm et 51 x 18 mm, respectivement.

Le redesign vise entre autres à réduire la place occupée par la partie électronique, permettant un boîtier plus compact et incidemment plus léger. La majorité des modifications apportées concernent le placement des composants et, de fait, le routage. Le schéma électronique ayant déjà été validé par le précédent design, seules de légères modifications y ont été apportées. Y figure, entre autres, le remplacement du support de la carte SD et l'ajout d'une super-capacité pour encaisser les pics de consommation lors des écritures sur la carte SD, comme décrit au point 2.

Pour les deux variantes, la taille de la plupart des composants passifs (résistances, condensateurs) a été réduite au format 0603 (0201-impérial) afin de gagner de la place. Certaines capacités de prévention non montées autour de l'antenne ont aussi pu être supprimées, celles-ci n'étant pas utilisées dans le précédent design.

## 4.3 Variante électronique 1

*Julien Michel, Mathieu Bourquenoud*

Cette première variante répartit l'électronique sur 3 PCB connectés par des parties flexibles, comme mentionné précédemment au point 3.2.1, en plus du PCB pour le système de release.

Le premier PCB contient la gestion de l'alimentation, le bouton reset (accessible en ouvrant le boîtier), le micro, l'électronique de la partie “burn wire”, ainsi que les pads auxquels les super-capacités seront connectées avec des fils. Les dimensions de ce PCB sont 36 x 15.5 mm.

Le second PCB est composé de l'antenne et du micro-contrôleur avec ses deux quartz et les composants passifs qui y sont liés. Ses dimensions sont 28.5 x 15.5 mm. Le fait

qu'il soit plus court permet de laisser de la place pour la partie flexible qui le connecte avec le PCB de la carte SD, sans que celle-ci ne subisse de contraintes mécaniques trop importantes dû à sa courbure.

Le dernier PCB est donc celui de la carte SD, qui est intégré dans le couvercle du boîtier. Celui-ci contient uniquement le support de la carte SD et quelques composants passifs. Ses dimensions sont 17.2 x 16.7 mm. Une seconde partie flexible a été ajoutée à ce PCB pour la programmation. Cette dernière partie flexible a des pistes à nu à son extrémité afin d'en faire un connecteur FFC. Cette partie flexible est laissée libre dans le boîtier. Il sera ensuite possible de programmer le micro-contrôleur en ouvrant le couvercle et avec l'aide d'un connecteur FFC.

Une variante pour la programmation, si le connecteur FFC flexible intégré n'est pas idéal, serait d'ajouter les pads nécessaires sur la partie bottom, pour pouvoir programmer la micro-contrôleur avec des pogo pins. Il serait cependant nécessaire de trouver une méthode de fixation du PCB de la carte SD dans le couvercle. L'utilisation de la colle empêchant l'accès aux pads exposé à l'arrière du PCB. Utiliser, par exemple, de la bande adhésive sur le couvercle pour éviter qu'il ne tombe à l'intérieur du boîtier résoudrait le problème. Fixer le PCB de manière permanente par l'utilisation de colle apparaît comme une solution plus durable et résistante, raison de notre choix d'implémenter un connecteur FFC.

Le layout des 3 parties qui composent le PCB semi-flex avec leurs dimensions est visible ci-dessous dans la figure 28.

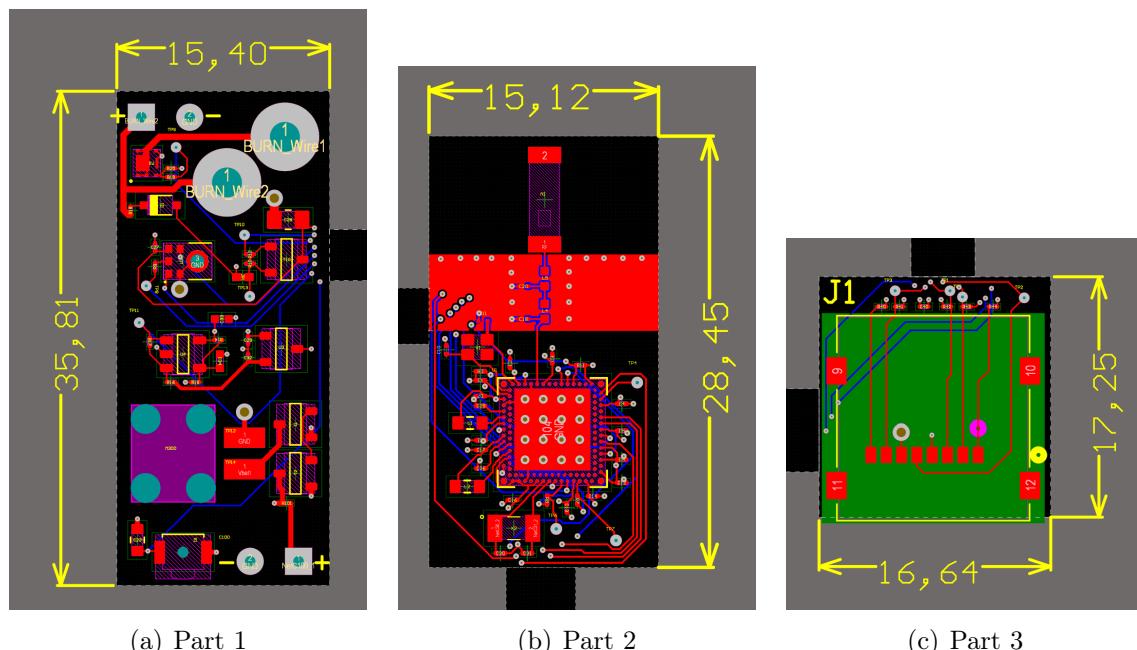


FIGURE 28 – Layout des 3 parties du PCB semi-flex et dimensions en mm

Une représentation 3D du PCB final ci-dessous dans la figure 29.

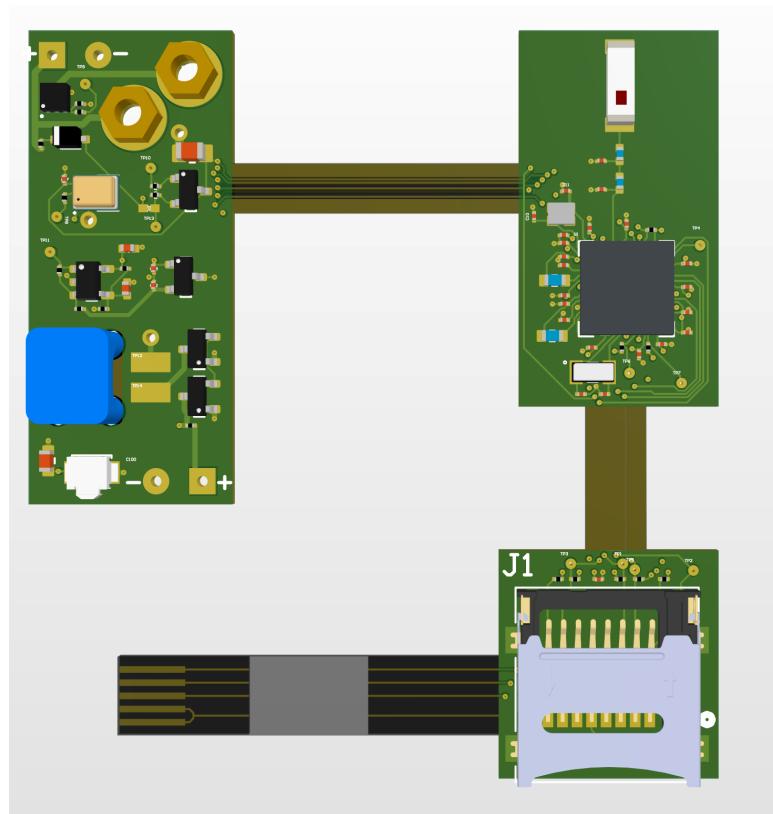


FIGURE 29 – PCB SemiFlex final design

A noter encore que le composant bleu sur le premier PCB est facultatif et ne devra être monté que si cela permet toujours de faire passer les câbles d'alimentation provenant de la pile. La connexion au système de release se fait comme lors de la précédente version, décrite au point 3.2.1. Le PCB a été refait selon la forme nécessaire pour le boîtier (voir figure 30).

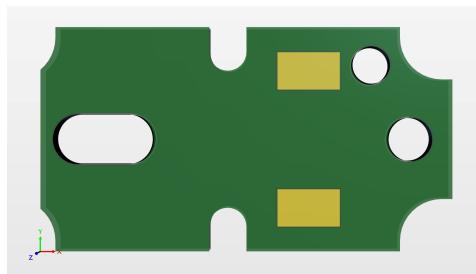


FIGURE 30 – PCB Système de release

## 4.4 Variante électronique 2

Miguel Oliveira

Les composants électroniques ont été réarrangés pour que les dimensions des cartes électroniques minimisent le dépassement de ses dimensions avec la longueur et la largeur de la pile.

Afin de respecter ces contraintes dimensionnelles, le circuit est réparti sur deux PCB disposés autour de la pile. Le premier qui sera présenté sera désigné comme étant le PCB du bas. La deuxième carte à être présentée, est désignée comme le PCB du haut.

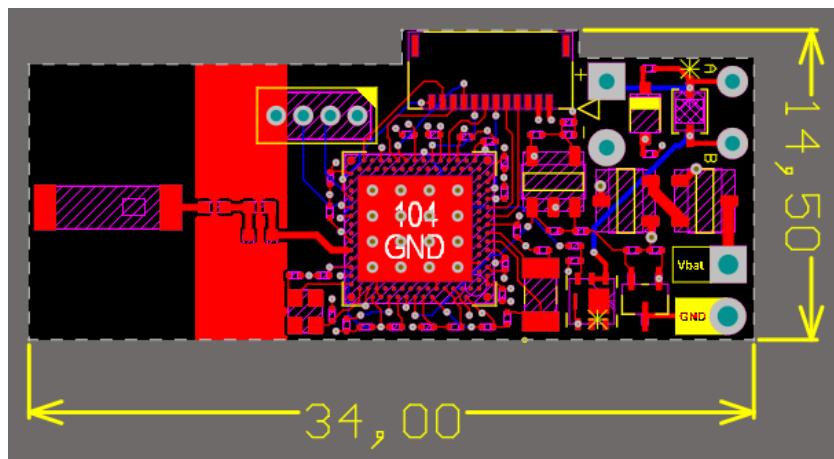


FIGURE 31 – PCB du bas avec ses dimensions en mm

Le PCB du bas, qui est le plus long, reçoit le microcontrôleur, l'antenne Bluetooth, le circuit d'alimentation et la partie contrôlant le release système qui pour rappel est adapté à la solution comportant un électro-aimant (voir figure 31). Un tel actionneur nécessiterait un courant de l'ordre du circuit pour faire brûler le fil de nylon, à savoir 2 A. C'est pourquoi, le même circuit du système "burn wire" est utilisé. À l'exception d'une diode de roue libre a été ajoutée aux bornes de la sorties prévu pour alimenter cet actionneur.

Au lieu d'utiliser des connecteurs volumineux et souvent sensibles aux vibrations, les fils de l'actionneur et de la pile seront simplement insérés et soudés dans des through-hole pads de 1.02 mm de diamètre. Ces trous peuvent accueillir des fils de calibre 18 à 22 AWG. Un calibre de 23 AWG est déjà suffisant, car il supporte un courant de 3 A (pour une tension de 3.3 V et 20 cm de fil), alors que notre application nécessite un maximum de 2 A. La polarité des pads d'alimentation est indiquée par une sérigraphie explicite ajoutée sur le PCB.

Comme aucun composant n'est soudé sur le Bottom Layer, ce PCB est collé au fond du boîtier sur la partie du bas.

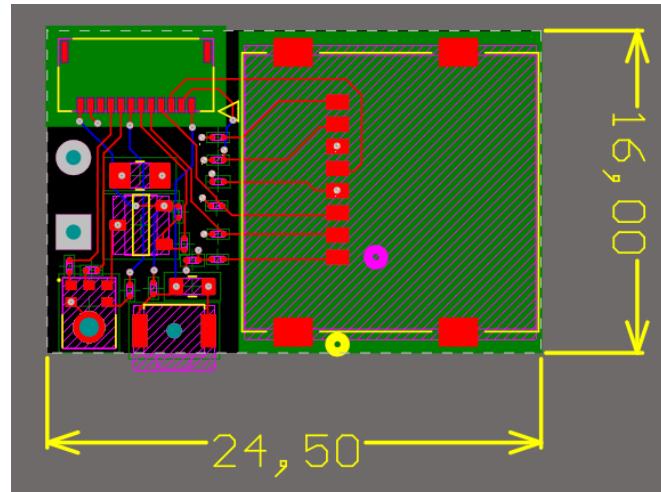


FIGURE 32 – PCB du haut avec ses dimensions en mm

Sur le deuxième PCB, nous retrouvons le connecteur de la carte SD, le micro et le bouton reset (voir figure 32). Ce dernier est bord de la carte ce qui permet d'appuyer dessus à travers une membrane étanche lorsque le boîtier est fermé. Cette carte est un peu plus large que la précédente. Ceci s'explique par la largeur imposante du connecteur de la carte SD. Le PCB du haut est collé sur le couvercle du boîtier sur la face du Bottom layer.

Les deux cartes sont reliées par le biais d'un câble volant FPC d'une longueur de 51 mm. Sur chaque PCB, il y a un connecteur FPC avec 12 contacts d'un pitch de 0.50 mm (voir figure 33).

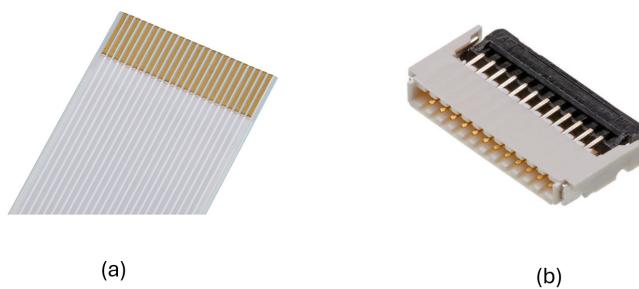


FIGURE 33 – (a) câble FPC - 15020-0117  
(b) 503480-1200 - connecteur FPC

Il a été envisagé, au cours de la conception, de n'avoir qu'un seul PCB avec une partie « pont » en flex PCB. Il est prévu de faire produire le PCB chez le fabricant Eurocircuit qui propose une solution SEMI-FLEX. La flexibilité de ce type de PCB est limitée à l'installation. Il n'est donc pas adapté à flétrir plusieurs fois. Vu qu'à chaque fois que l'on ouvre le boîtier, le câble se déplie, l'utilisateur risque de rapidement endommager le PCB.

Le câble FPC choisi se révèle être constitué de Premo-Flex ce qui lui confère un cycle de vie de flexion atteignant les 50 000 flexions.

Les deux super condensateurs étant considérablement encombrants, il est prévu de tirer parti de la longueur de leurs pattes pour les tordre à 90° et « coucher » le composant en face de leurs PCB respectifs. À titre préventif, il est conseillé de mettre de la gaine thermorétractable autour des pattes pour éviter les court-circuits.

Dans la figure 34 ci-dessous, le rendu final des deux PCB :

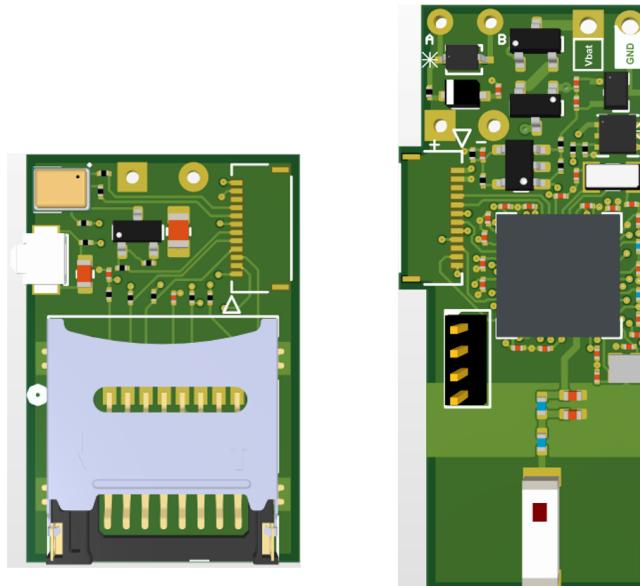


FIGURE 34 – PCB du bas et du haut en vue 3D

## 4.5 Évaluation d'un SoC LoRa

*Julien Michel, Mathieu Bourquenoud*

### 4.5.1 Introduction

Un problème soulevé avec le design actuel est la portée du Bluetooth. Celle-ci est relativement restreinte et utilise une fréquence qui possède une faible capacité de pénétration. En zone dense et humide, comme sera le lieu d'utilisation de l'appareil, la réduction de la portée devient significative. Le débit de données nécessaire étant faible, l'utilisation d'un système à longue portée LoRa permettrait de pallier ce problème. Celui-ci possède par ailleurs l'avantage de consommer très peu d'énergie, ainsi que de permettre une triangulation pour une localisation sans GNSS. La section suivante évalue la possibilité d'utiliser un système LoRa.

Du à la quantité importante de modifications nécessaires, la section suivante se concentrera uniquement sur l'évaluation et la recherche de composants. Aucune implémentation ne sera conduite.

### 4.5.2 Contraintes

Le design actuel utilise un microcontrôleur intégrant une chaîne de transmission RF Bluetooth. L'utilisation d'un circuit auxiliaire externe au microcontrôleur augmenterait de manière significative la surface nécessaire, déjà fortement restreinte. Il est donc nécessaire d'utiliser un microcontrôleur intégrant directement la chaîne RF.

Pour éviter une réécriture complète du logiciel, il est nécessaire d'utiliser un microcontrôleur compatible avec Zephyr, de préférence avec une architecture ARM.

Il y a, de plus, une contrainte liée à la taille de l'antenne. Si celle-ci est intégrée au PCB, elle doit rester suffisamment petite pour ne pas augmenter la taille de la carte. Les bandes LoRa étant de plus faible longueur d'onde, ceci peut poser un challenge significatif.

### 4.5.3 Microcontrôleur

Après une recherche incluant les divers paramètres et contraintes, une seule série de microcontrôleurs apparaît comme viable. Série relativement récente des bureaux de ST-Microelectronics, la série STM32WLE est équipée d'un module radio certifié comme compatible avec LoRa. Ces microcontrôleurs utilisent un cœur ARM M0+ ou M4, similaire à celui utilisé dans le design du circuit original, assurant un support par Zephyr.

Une rapide recherche dans les divers fournisseurs usuels indique une disponibilité importante, critère souvent mis de côté mais critique à tout nouveau design. Par exemple, chez le fournisseur Mouser, le stock varie entre 2k et 6k unités. Chez Farnell, le stock est similaire.

Comparé au nRF5340 actuellement utilisé, les caractéristiques sont proches, comme visible dans la table 1.

Spécification	nRF5340	STM32WLE5JCI6
Prix pour 1k	4.75	4.93
Cœur	M33	M4
Fréquence [MHz]	64	48
Flash [kB]	1024	256
RAM [kB]	64	64
I/O	48	43
Tension [V]	1.7-5.5	1.8-3.6

TABLE 1 – Comparaison nRF5340 et STM32WLE5JCI6

La série des microcontrôleurs STM32WLE semble donc compatible avec les contraintes de l’application.

#### 4.5.4 Antenne

Comme cité précédemment, l’antenne pose un problème significatif au niveau du volume nécessaire. L’utilisation d’un monopole monté en surface comme dans le design actuel requiert un plan de masse d’une surface minimale. Le protocole LoRa supporte plusieurs bandes de fréquences, l’utilisation de la fréquence la plus élevée permet une réduction des dimensions de l’antenne. L’évaluation se fait donc sur une antenne en 868 MHz.

Les contraintes de dimensions étant principalement dictées par des phénomènes physiques hors de contrôle du fabricant, les divers modèles d’antenne requièrent des surfaces similaires. Pour évaluer la faisabilité, nous utilisons l’antenne ANT-868-USP-T.

Comme visible sur la figure 35, la surface demandée pour le plan de masse est bien supérieure à ce qui est faisable. La solution utilisant une antenne montée en surface n’est donc pas adaptée.

Une autre proposition serait l’utilisation d’une antenne externe fixée au collier. En collant une antenne dipôle en PCB flexible, la surface requise serait largement suffisante. Placer l’antenne sur la surface intérieure permettrait de la protéger de l’environnement.

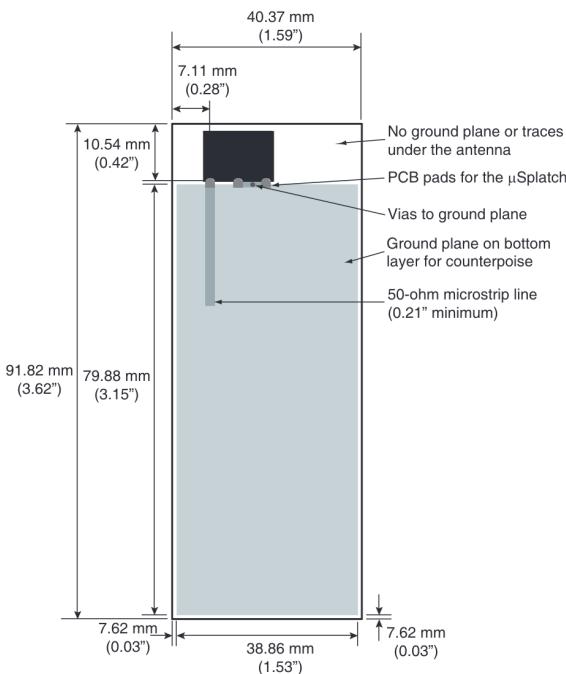


FIGURE 35 – Dimensions pour l'antenne

#### 4.5.5 Conclusion

L'utilisation du LoRa permettrait une connexion à des distances très supérieures à ce que permet le Bluetooth. De plus, en plaçant des bornes à des positions connues, il est aussi possible de trianguler la position des unités sans l'aide de GNSS.

L'avantage est donc clair, mais la contrainte de l'antenne reste importante et demande une évaluation approfondie pour trouver une solution adaptée. De plus, plusieurs modifications du logiciel sont nécessaires, ce qui augmente considérablement le temps de développement.

## 4.6 Évaluation de l'utilisation du eMMC

*Julien Michel, Mathieu Bourquenoud*

#### 4.6.1 Introduction

La carte SD utilisée dans le système actuel pose plusieurs problèmes. Premièrement, la surface occupée reste relativement importante. De plus, n'étant pas soudée directement, les chocs et vibrations peuvent provoquer une déconnexion momentanée qui peut corrompre les fichiers. Et finalement, la consommation de ce type de mémoire est en général plus élevée que d'autres types de mémoire embarqués.

Pour pallier ces problèmes, la section qui suit évalue la possibilité d'intégrer une mémoire eMMC. Celle-ci utilise un protocole identique à celui des cartes SD, permettant ainsi de conserver exactement le même code.

#### 4.6.2 Contraintes

La quantité de données à stocker est importante, et une capacité d'au moins 32 GB est demandée. Le courant d'écriture devra être le plus faible possible, préféablement inférieur à 100 mA.

#### 4.6.3 Composant

De nombreuses eMMC sont disponibles sur le marché. Pour explorer la faisabilité, notre choix s'est porté sur la série EMMCxG-TY29 de Kingston Digital. Celles-ci proposent des versions 64 GB, 128 GB et 256 GB. La consommation de courant se situe entre 80 mA et 90 mA, avec un débit de 220 MB/s. La puissance consommée est environ deux fois inférieure à celle d'une carte SD, et le débit environ deux fois supérieur. À volume de données équivalent, la consommation énergétique est donc réduite par un facteur quatre.

Le boîtier est un FBGA153, en 11.5 x 13.0 mm, significativement plus petit qu'une carte SD.

#### 4.6.4 Conclusion

L'utilisation d'une eMMC permettrait de réduire la surface ainsi que la consommation, augmentant ainsi l'autonomie. La stabilité du système serait aussi fortement augmentée. Accessoirement, la mémoire devant obligatoirement être lue par le microcontrôleur, il serait possible d'éviter l'utilisation d'un système de fichiers et d'enregistrer directement les données en brut, évitant ainsi la corruption des données, celles-ci n'ayant plus d'interdépendances.

### 4.7 Interface GNSS

Bien que la partie GNSS soit encore en phase d'étude de faisabilité, il est important de prévoir une interface avec les PCB, en allouant des pins du microcontrôleur à la communication avec le module GNSS. De plus, il est nécessaire de penser à la connectique appropriée. L'emplacement du module sur le collier doit également être déterminé.

## 5 GNSS

*John Isaac Wetenkamp, John Biselx*

Il a été demandé par les mandants du projet d'explorer la possibilité d'ajouter un GNSS au système. Cette section s'agit donc d'une étude de faisabilité. Afin de faire une analyse complète, tous les facteurs contribuant à la consommation du module GNSS sont étudiés et présentés ici. Cela comprend : des mesures concrètes de consommation de deux modules GNSS, deux protocoles de communication ainsi que leur efficacité énergétique et différents paramètres GNSS permettant de réduire la consommation. L'objectif à la fin de cette étude est de déterminer un budget d'utilisation pour le GNSS ainsi qu'un chemin viable à suivre pour l'implémentation complète de celui-ci.

### 5.1 Outils

Les outils ont été imposés soit selon les choix faits dans les étapes précédentes de ce projet ou bien par le hardware choisi. Tous les outils utilisés sont résumés ici afin de faciliter les analyses qui suivront.

#### 5.1.1 Hardware

##### 5.1.1.1 Kit de développement

Un certain nombre de tests hardware ont été faits avec l'objectif d'avancer cette étude de faisabilité. Dans le cadre de ces tests, il a été nécessaire de se rapprocher le plus possible à la réalité hardware du système existant. Un kit de développement a donc été choisi pour pouvoir utiliser le même processeur que celui déjà présent sur le prototype. Ce kit fournit également des périphériques utiles pour la mise en marche rapide du GNSS. Il s'agit donc du nRF5340 DK fabriqué par Nordic Semiconductor (Nordic Semiconductor, [s. d.(d)]) (voir figure 36).



FIGURE 36 – nRF5340 DK

### 5.1.1.2 Power Profiler Kit II

Nordic Semiconductor fabrique également un instrument qui permet de faire des mesures de courant entre 200 nA jusqu'à 1 A. Cet outil a été utilisé pour faire les mesures de courant des modules GNSS (Nordic Semiconductor, [s. d.(e)]) (voir figure 37).

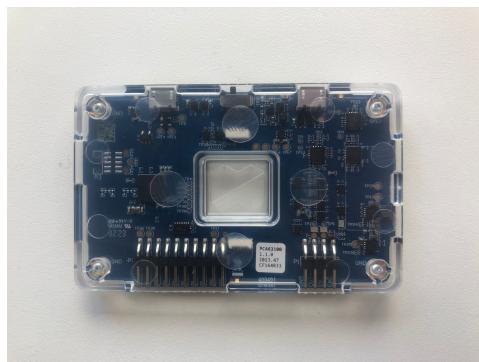


FIGURE 37 – Power Profiler Kit II

### 5.1.1.3 Modules GNSS

Pour faire des analyses, on nous a fourni deux modules GNSS fabriqués par u-blox, modèles CAM-M8C (u-blox, [s. d.(a)]) et RCB-F9T timing board (u-blox, [s. d.(b)]). Ces modèles ne sont pas les plus récents, mais ils étaient disponibles sur place. En faisant des recherches pour un module plus économique nous avons trouvé le module M10, modèle SAM-M10Q (u-blox, [s. d.(c)]) que nous nous sommes procurés.

La façon de communiquer avec les modules GNSS ne change pas entre les différents modules. Les modules CAM-M8C et SAM-M10Q intègrent une antenne embarquée afin de faciliter la conception du PCB, tandis que RCB-F9T nécessite une antenne externe. Ce dernier était seulement utilisé pour tester la communication avec le module.

Afin de paramétriser les messages que le module envoie, u-center (voir section 5.1.2.3) a été utilisé pour mettre les paramètres en flash. Cependant, cette méthode fonctionne uniquement avec le RCB-F9T. Après de maintes recherches, il a été découvert seulement vers la fin du projet que les modules CAM-M8C et SAM-M10Q ne comportent pas de flash programmable intégrée. En consultant les forums u-blox en ligne, nous ne sommes visiblement pas les seuls à avoir cru que ces modules comportaient une mémoire flash (cupbus105113, [s. d.]) (Alpenhunde, [s. d.]). Cette confusion est due au fait que les modules qui ont une mémoire flash ont une indication claire sur l'inclusion de la flash sur leur site internet tandis que les modules qui n'en ont pas, n'ont aucune mention de mémoire (u-blox, [s. d.(f)]).

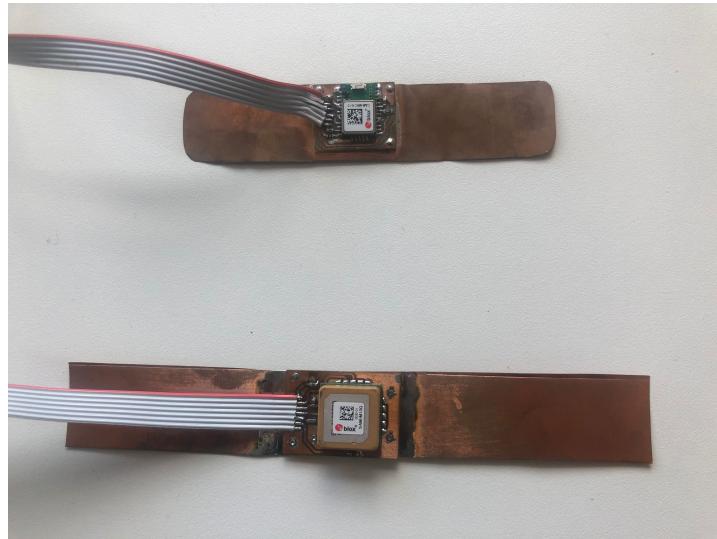


FIGURE 38 – Modules GNSS (M8 en haut, M10 en bas)

Nous avons cherché si le module M10 disposait d'un élément qui remplace la mémoire flash programmable. Il y a une mémoire OTP (one-time programmable), mais elle est très limitée (69 Bytes) et donc nous ne l'utiliserons pas. Cela veut dire qu'il faudra programmer la configuration sur la board et l'envoyer par UART au module GNSS à chaque démarrage. Les paramètres peuvent être sauvegardés dans la battery-backed RAM (BBR), qui normalement devrait restaurer les paramètres sauvegardés, mais nous n'avons pas pu faire fonctionner cet élément. Ceci reste une perspective future intéressante à explorer.

### 5.1.2 Software

#### 5.1.2.1 nRF Connect SDK

En suivant les instructions laissées sur le dépôt git par nos prédecesseurs (HEVS, [s. d.]), le choix d'un environnement de développement a été également facilité. Le SDK nRFConnect comporte une extension VS Code qui a été mise en action (Nordic Semiconductor, [s. d.(c)]). Le SDK inclut Zephyr OS. La version 2.4.0 du SDK a été sélectionné afin de suivre la sélection précédente.

#### 5.1.2.2 nRF Connect for Desktop

nRF Connect for Desktop (Nordic Semiconductor, [s. d.(b)]) est un ensemble d'outils qui contient entre autres Power Profiler, l'outil qui permet de visualiser les mesures prises par le Power Profiler Kit II.

### 5.1.2.3 Outils u-blox

Un certain nombre d'outils sont intégré par défaut à cet environnement de travail telles que des librairies du UART qui permettront de communiquer avec le GNSS. D'autres outils qui ne sont pas inclus dans cet environnement ont également été utilisés pour faciliter certains aspects du travail réalisé. u-center par exemple (u-blox, [s. d.(e)]), est un outil qui permet de configurer facilement le module GNSS (pour changer par exemple le protocole utilisé ou bien différents paramètres pour réduire la consommation). L'utilité de cet outil provient partiellement du fait qu'il est possible de stocker ces paramètres dans la mémoire flash du module GNSS. En ce faisant, cela permet de libérer le processeur sur le système complet du travail et de l'énergie nécessaire pour configurer le GNSS au boot. u-center est également utile pour analyser l'efficacité énergétique de certains paramètres avant d'avoir un code complètement fonctionnel sur le kit de développement. L'application permet de programmer le GNSS directement par le biais d'un câble USB-UART.

Dans le cadre de cette étude un API mis à disposition par u-blox a également été testé (u-blox, [s. d.(h)]). L'objectif de cet API, appelé ubxlib est de rendre l'initialisation du module GNSS plus simple en passant par des couches d'abstraction plus élevées. Il permet également de configurer dynamiquement le module GNSS, cela enlève donc le besoin d'avoir une mémoire flash programmable sur le module GNSS (permettant ainsi d'économiser du budget et de la place sur le PCB). Un retour sur l'utilisation de cet API sera donné à la fin dans la section 5.5.

## 5.2 Analyse de réception

### 5.2.1 M8

Dans un premier temps, nous avons analysé si le module M8 était capable recevoir des signaux GNSS. Le plan de masse est plus grand (20 x 100 mm) que le minimum recommandé, mais il y a deux déviations d'une utilisation normale. Le plan de masse doit être intégré dans le collier, comme il est trop grand pour être placé autre part. Cela veut dire que le plan sera potentiellement couvert par du cuir et plié autour du cou du singe, ce qui peut poser un autre problème : l'atténuation des signaux GNSS par le corps du singe. Un autre point est que u-blox recommande d'éloigner le module GNSS d'au moins 10 mm de la chair, comme ça atténue la réception.

Pour qu'un module puisse recevoir un signal GNSS utilisable, ce signal doit être supérieur à 32 dB. L'outil u-center affiche la réception de chaque signal reçu d'un satellite, le maximum observé était 45 dB. La pose de couches de cuir qui s'accumulent à 10 mm n'atténue pas les signaux d'une façon mesurable. Avec le module orienté vers le ciel et le plan de masse mis sur ces couches de cuir et le tout placé autour de la poignée, une atténuation maximale de 5 dB est possible. En retournant la poignée de 180° - donc le module ne voit pas le ciel - les signaux sont atténuer de 15-20 dB, ce qui rend souvent

les signaux inutilisables. Lorsque les couches de cuir sont enlevées et nous plaçons le plan de masse autour de la poignée, le module orienté vers le ciel, nous observons une atténuation de 12 dB.

La distance entre plan de masse et singe est donc important, mais nous pouvons gérer cela dans la conception. Cependant la posture du singe n'est pas maîtrisable et modifie l'atténuation plus que la distance, ce qui pose un problème non-négligeable avec le module M8.

### 5.2.2 M10

Le module M10 est en général moins sensible que le M8. Nous avons pu garder son plan de masse plus grand (22 x 151 mm) que celui du module M8. Après avoir mis le plan de masse plaqué contre et autour de la poignée, sans cuir, avec le module orienté vers le ciel, l'atténuation est de 5 dB. Si la poignée est retournée, l'atténuation maximale est de 10 dB - pour certains satellites il n'y a pas de changements. Le M10 est donc plus performant.

## 5.3 Analyse de consommation

Pour cette section, les modules étaient paramétrés selon leurs paramètres d'usine, pour comparer les performances entre les modules.

### 5.3.1 M8

Avec le Power Profiler Kit II nous avons mesuré le courant dont le module a besoin lorsqu'il démarre. Pour cela, la broche de reset a été connectée à la masse pour activer le reset, ensuite nous l'avons déconnecté pour le faire démarrer.

Pendant la période où le reset était activé un courant moyen de 14.59 mA a été mesuré. Après cela, il y a 2 phases qui s'alternent : Une phase de « basse » consommation et une de « haute » consommation. La phase « haute » consomme en moyenne 30.56 mA, avec un maximum de 39.94 mA. Sur les deux minutes mesurées, la phase haute était active environ 80 % du temps.

### 5.3.2 M10

Pendant la période où le reset était activé nous avons mesuré un courant moyen de 3.10 mA, avec des pics de 3.92 mA. Après cela, il y a plusieurs niveaux de consommation, la moyenne étant 12.20 mA, avec un maximum de 17.10 mA. La BBR du module M10 consomme très peu, au maximum 1.32 mA et en moyenne 0.21 mA.

### 5.3.3 Choix module

Les résultats des sections précédentes signifient qu'il vaut la peine d'éteindre entièrement le module GNSS, au lieu de le garder allumé ou de le mettre en mode reset. Réallumer le module M10 après avoir coupé l'alimentation pendant 3 min est semblable à un hot start, il retrouve sa position après quelques secondes. En attendant 10 min au lieu de 3 min, il perd sa position et c'est considéré comme un cold start.

Les tests suivants ont été effectué dehors avec un orage s'approchant :

Module	Type	Moyenne [s]	Type de fixation
M8	cold start	160.09	3D
M8	warm start	-	N'a pas eu de fixe
M10	cold start	38.19	3D
M10	hot start	3.67	3D
M10	hot start	2.40	2D

TABLE 2 – Temps de fixation pour modules M8 et M10

Sur un balcon ayant 2 balcons au dessus et le toit, le temps de démarrage au froid pour le module M10 était d'environ 1.5 min avec un ciel couvert. Le temps de fixation est donc très variable et dépendant beaucoup des conditions environnementales.

A partir de maintenant nous ne considérons plus le module M8, nous considérons uniquement le module M10 car il est supérieur en tout.

### 5.3.4 Estimation de temps d'allumage du module GNSS

La pile que nous avons choisie est la *Tadiran Lithium 2/3AA 3.6 V (SL-761)*, qui a une capacité nominale de 1.5 Ah. La pile a tenu 12 jours (288 h) et la consommation moyenne du système de base était 4.02 mA (voir figure 2).

La pile serait entièrement plate après  $\frac{1.5\text{Ah}}{4.02\text{mA}} = 373.13$  h, soit 15.54 jours, sans module GNSS. Si on suppose que la fixation de position dure 2 min et que nous prenons la position chaque heure, nous consommons en moyenne  $\frac{2}{60} \cdot 12.20$  mA = 0.41 mA de plus pour le module GNSS.

La durée avec le module GNSS serait donc  $\frac{1.5\text{Ah}}{4.02\text{mA}+0.41\text{mA}} = 338.86$  h, soit 14.11 jours. Si on double le temps alloué à la fixation, donc 4 min, la durée serait 310.34 h, soit 12.93 jours, ce qui nous donne une petite marge.

Pendant les tests, la durée pour fixer la position valait 38.19 s selon la table 2, mais ce temps peut varier fortement selon les conditions environnementales.

### 5.3.5 Protocoles de communication

Pour la famille de module GNSS choisie, il existe deux protocoles de communication qui peuvent être utilisées indépendamment l'un de l'autre ou bien ensemble (voir la fiche technique pour le module M8 (u-blox, [s. d.(a)])). Ces protocoles sont NMEA et UBX. NMEA est un protocole ASCII avec un certain nombre de messages disponibles pour indiquer tous les nombreux paramètres du GNSS. UBX est un protocole créé par u-blox, le fabricant de cette famille de GNSS. Ce dernier est un protocole binaire.

Nous avons essayé de mesurer le courant partant dans la ligne TX du module GNSS, mais la résistance interne du Power Profiler Kit II doit être trop élevée car la communication cesse lorsque l'appareil de mesure est branché.

Pour tout de même estimer la consommation énergétique des protocoles de communication, nous pouvons comparer les tailles des messages. Nous utilisons pour cet exemple le message GLL du protocole NMEA et le message UBX-NAV-PVT du protocole UBX (u-blox, 2023b, chap. 2.7.6 et 3.15.11). Selon les tests réalisés de notre part, le message GLL peut contenir jusqu'à 52 Bytes tandis que le message UBX-NAV-PVT a une taille fixe de 100 Bytes selon u-center. Le message GLL contient la position avec le status de la fixation et un peu plus, lorsque UBX-NAV-PVT inclut tout ce que GLL a et encore beaucoup plus : vitesse, date, heure, etc. En revanche, il est probable que le décodage des messages binaires soit plus efficace qu'un parsing ASCII, mais nous n'avions pas les moyens pour vérifier l'impact que cela avait.

## 5.4 Analyse de paramètres

### 5.4.1 Multiple GNSS assistance (MGA)

u-blox propose diverses méthodes pour rendre la fixation de position plus rapide. Il existe 3 services distincts : AssistNow Online, AssistNow Offline et AssistNow Autonomous (u-blox, 2023a, chap. 3.11). Le service Online, comme son nom l'indique, a besoin d'une connexion Internet à tout temps, ce qui le rend inutilisable dans le cadre de ce projet. Les services Offline et Autonomous sont mutuellement exclusifs, le service Offline prenant la priorité si les deux sont enclenchés en même temps. Les services Online et Offline ont besoin d'un token d'authentification que u-blox fournit. Il y a une période d'essai de 30 jours, après cela devient payant (u-blox, [s. d.(d)]).

Le service Offline n'a pas été testé car il a été jugé que rajouter ce service prendrait plus d'effort que de valeur rapportée. Il faudrait mettre en place un système de stockage des coordonnées des satellites sur la board, étant donné que le M10 n'a pas de flash et que sa BBR (battery-backed RAM) est potentiellement insuffisante, cette solution ne semble pas viable. Il serait cependant envisageable de télécharger les coordonnées sur un ordinateur et de les transférer sur la carte SD, mais cela veut dire qu'il faudrait dédier une zone de la carte SD pour les coordonnées. De plus, il faudrait ajouter de la logique

pour gérer la communication du service Offline.

Le service Autonomous n'a pas besoin de connexion Internet ni de stockage basé sur la board. Cependant, ce service envoie son status à la board, donc il faudrait gérer cette logique pour ne pas éteindre le module lorsqu'il est en train de faire des calculs. D'ailleurs, pour pouvoir utiliser ces données, le module doit maintenir et calculer des données, ce qui coûte cher en terme de consommation. Nous jugeons donc qu'il serait mieux de laisser ce service de côté pour l'instant, mais cela serait une piste à poursuivre si la réception sur le terrain était sous-optimale.

### 5.4.2 Mode basse-consommation

Les modules M8 et M10 ont des paramètres pour configurer des modes de consommation. Celles-ci ne sont pas adaptées pour ce cas d'utilisation, car le mode de basse consommation est prévu pour l'utilisation constante du module GNSS, lorsque nous allumons notre module jusqu'à ce qu'il a pu fixer sa position ou un temps défini est dépassé.

## 5.5 API UBX

### 5.5.1 Description

u-blox fournit un API afin de pouvoir utiliser le protocole UBX avec facilité, comme décrit dans la section 5.1.2.3. Cet API permettrait de configurer dynamiquement les paramètres du module GNSS si besoin ainsi que lire la position.

Au début de ce projet nous avions prévu d'utiliser le message UBX-NAV-SOL qui a la même taille que le message NMEA GLL (52 Bytes), mais comme c'était en binaire pas de parsing ASCII serait nécessaire. Le message UBX-NAV-SOL est disponible pour le module M8, mais nous avons su qu'il n'était pas disponible pour le M10 seulement vers la fin du projet. C'est pourquoi que cette analyse a été faite.

### 5.5.2 Problèmes rencontrés

La seule documentation de cette API se trouve dans des README dispersés sur le dépôt GitHub de celui-ci. Le README approprié qui indique la démarche à suivre pour intégrer l'API à Zephyr (u-blox, [s. d.(j)]) indique un certain nombre de lignes à rajouter au fichier appelé west.yml, le fichier qui indique à west (le manager de packages pour zéphyr) comment récupérer les packages demandés. Cependant, ce README est ambigu concernant le west.yml à modifier : est-ce qu'il s'agit d'un nouveau west.yml pour le projet sur lequel nous travaillons ou bien du west.yml inclu dans le SDK fourni par Nordic ? Nous n'avons pas pu trouver de réponse claire à ce sujet.

Dans un premier temps, nous avons essayé d'ajouter les lignes spécifiées à un nouveau west.yml. Après avoir créé un nouveau fichier west.yml avec le contenu nécessaire, VS-Code indique à travers le plugin nRFConnect que west doit être mis à jour. La mise à jour

est automatique, mais malheureusement il lance un nouveau téléchargement de Zephyr et du SDK nRF Connect. Cette méthode est donc redondante étant donné l'impossibilité d'indiquer un SDK existant (au moins à notre connaissance). De plus, le toolchain (compilateur, etc.) n'est pas téléchargé et VSCode continue à utiliser le toolchain téléchargé à travers nRFConnect For Desktop. Utiliser un toolchain qui n'est pas au même emplacement que le SDK créé des conflits et même un blinky tout simple ne compile plus.

La seule manière trouvée pour intégrer l'API à VSCode consistait donc en modifier le west.yml du SDK. La démarche à suivre pour ce faire est inclue en annexe dans 8. Cette méthode permet de compiler et de faire fonctionner un blinky. Néanmoins, il n'est toujours pas possible avec cette méthode d'utiliser l'API correctement.

Un exemple est donné sur le dépôt git ubxlib sur comment initialiser et utiliser le module GNSS pour lire la position (u-blox, [s. d.(i)]). Cet exemple n'est cependant pas fonctionnel avec la méthode d'intégration précisée précédemment. Le résultat est un code qui compile mais un code d'erreur *U\_ERROR\_COMMON\_PLATFORM* (-8). Quelques références à cette erreur ont été trouvées sur la section issues sur le dépôt git de l'API mais leur situation était souvent différente de la nôtre (KjartanOli, [s. d.]) (mos216, [s. d.]). Lorsque nous avons commencé cette démarche, ces issues étaient encore ouverts mais ils ont été fermés tout récemment. Il est possible que ces discussions apportent des éléments de réponse pour le futur du projet. Ces problèmes viennent certainement de l'intégration de l'API au sein du SDK nRF Connect.

Étant donné un grand nombre de difficultés, il a été décidé que pour le cadre de ce projet d'abandonner l'utilisation de l'API UBX. La documentation est trop dispersée (uniquement sur GitHub sous forme de README et de header files), l'intégration au SDK nRFConnect est trop onéreuse (même s'il s'agit de la plateforme recommandée pour cet API), et il reste tout de même possible de communiquer avec le GNSS sans l'API. Pour les futurs travaux de ce projet cet API pourrait être repris si nécessaire car il a pour objectif d'apporter une certaine facilité pour la configuration dynamique du GNSS.

### 5.5.3 Perspectives futures

Le problème de manque de flash programmable réduit le nombre de choix pour les travaux futurs du projet. Il ne reste plus que trois possibilités :

- Implémenter sa propre mémoire flash sur le PCB
- Configurer le module GNSS au démarrage (cela utilise néanmoins du temps de CPU à chaque démarrage)
- Trouver un module GNSS comportant à la fois une mémoire flash programmable et une antenne intégrée

Implémenter sa propre mémoire flash sur le PCB impliquerait beaucoup de travail et n'est pas le choix le plus adapté. Après de maintes recherches, il ne semble malheureusement pas y avoir de module M10 comportant à la fois une mémoire flash et une antenne intégrée (c'est soit l'un soit l'autre). Il ne reste plus que l'option de configurer le module au démarrage. Pour ce faire, des paquets doivent être envoyés au module au démarrage à travers le protocole UBX. Dans des travaux futurs il sera nécessaire de mesurer et de minimiser la consommation énergétique de cette étape supplémentaire à prendre.

L'analyse du protocole UBX était tout de même nécessaire, car UBX est le seul protocole qui permet de configurer le module M10. À ce moment nous recommandons le code qui peut encoder et décoder des messages UBX sans utiliser l'API de u-blox (u-blox, [s. d.(g)]). En faisant des recherches nous avons trouvé que Zephyr supportera partiellement le protocole UBX, ce qui pourrait être intéressant quand Zephyr 3.7.1 sera publié (mayankmahajan-nxp, [s. d.]).

Un autre travail futur serait d'analyser la consommation de courant de plus près, ce que nous n'avions pas pu faire à cause d'un manque de temps. L'idée serait de prendre des codes qui permettent de décoder des messages UBX (soit le code de (u-blox, [s. d.(g)]), soit Zephyr 3.7.1) et NMEA (Zephyr le supporte) et de les faire tourner sur un kit de développement, tout en prenant des mesures de courant.

## 5.6 Livrables

Plusieurs livrables ont été réalisés pour la partie GNSS du projet :

1. Un code exemple sur GitHub qui démontre comment décoder des messages UBX envoyés au CPU par le module GNSS (u-blox, [s. d.(g)]).
2. Un code exemple sur GitHub qui démontre comment recevoir des messages du module GNSS via un pont UART (John Biselx, [s. d.(b)]).
3. Un code python sur GitHub qui permet de lire les valeurs envoyées par les modules GNSS M8 et M10 et de les stockés dans un fichier .csv (John Biselx, [s. d.(a)]).

Les deux exemples en haut n'ont pas été combinés en un étant donné les difficultés rencontrées avec la mémoire flash programmable. Cependant, la combinaison des deux est aisée étant donné des fonctions bien définies qu'il suffit de copier et de coller. Le premier exemple a été tiré de l'API UBX. Il s'agit des fonctions qui permettent de décoder le buffer qui contient un code binaire selon le protocole UBX. Ce code fonctionne indépendamment de la plateforme et a été testé sur un ordinateur portable avec un vrai message UBX sauvegardé dans un fichier. u-blox fournit également des fonctions pour encoder des messages, chose qui sera certainement utile pour la configuration dynamique des paramètres. Le deuxième exemple est configuré pour fonctionner avec NMEA mais afin de l'adapter pour UBX il suffit d'imprimer les messages en utilisant l'option pour afficher des caractères hexadécimal plutôt que des caractères de type string.

## 6 BLE Booster

Adrien Rey, Sylvestre van Kappel

### 6.1 Introduction

Le collier est contrôlé par une application sur un téléphone via une connexion Bluetooth Low Energy (BLE). L'application permet de détecter les colliers à proximité et de s'y connecter. Une fois connecté, il est possible d'activer ou de désactiver l'enregistrement, de réinitialiser le collier et de l'ouvrir.

Durant les tests sur le terrain, l'ouverture du collier via l'application a posé problème en raison de la portée limitée, la distance entre le téléphone et le collier étant trop grande. L'objectif de cette partie du projet est de développer un appareil qui remplace le téléphone, en utilisant une antenne directionnelle et un Front-End Module (FEM) pour amplifier le signal Bluetooth. Ce nouveau système doit permettre d'augmenter la distance à laquelle le collier peut être contrôlé. Il reprend les fonctionnalités de l'application mobile existante. Cependant, cet appareil ne remplacera pas complètement le smartphone. Ce dernier permet aussi, lors de la connexion à un collier, de mettre à jour sa date et son heure. Cette fonction ne sera pas implémentée dans le BLE Booster par mesure de simplicité.

L'appareil se base sur le kit de développement nRF21540-DB, qui contient un processeur nRF52840 et un Front-End Module (FEM) nRF21540. Il faudra y ajouter l'antenne directionnelle, une interface utilisateur et une batterie pour que l'appareil soit utilisable sur le terrain. Le processeur est programmé avec nRF Connect et le programme utilise le RTOS Zephyr (voir figure 39).

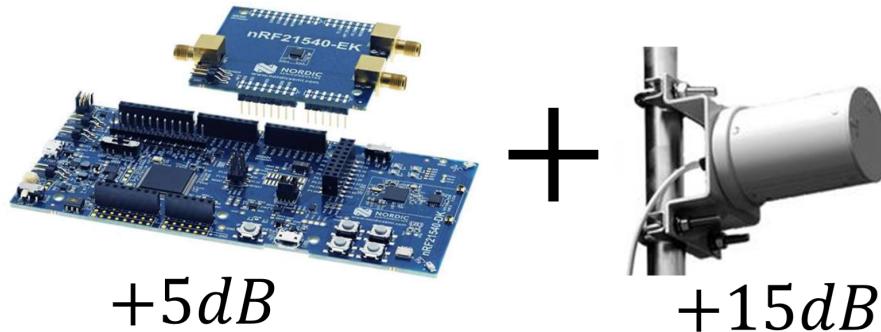


FIGURE 39 – nRF21540-DB

## 6.2 Software

### 6.2.1 Architecture générale

Le programme est séparé en deux tâches principales. La première est la gestion de l'interface et la deuxième est la gestion du Bluetooth. Les données sont partagées entre les deux tâches par la structure de données "monkeylist" et elles communiquent via des fonctions de callback. Les callbacks sont assignés dans "connection". L'interface inclut le contrôle de l'écran dans "display\_controller" et "display", ainsi que la gestion des boutons dans "button\_manager". La partie Bluetooth permet de scanner les différents colliers, de s'y connecter et de les contrôler. Le code est réparti dans "ble" et "snes\_client". La figure 40 montre les interactions entre les différents composants du système.

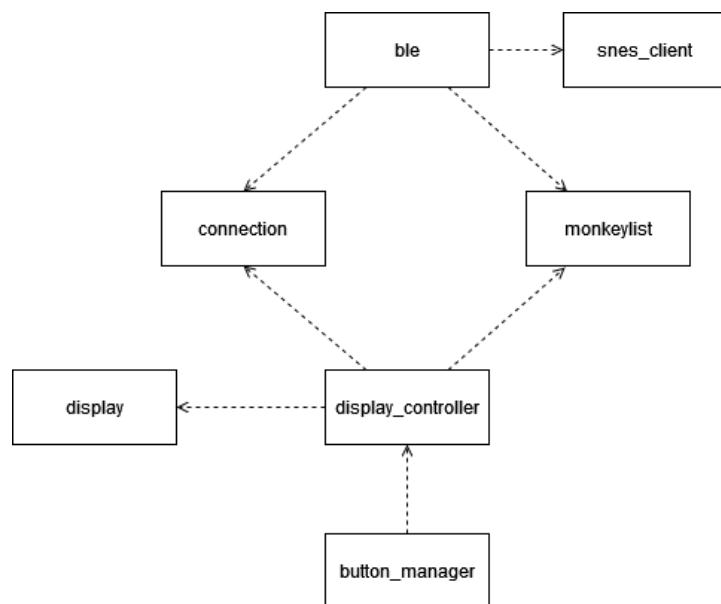


FIGURE 40 – Interface - Page principale

### 6.2.2 Interface

L'interface utilisateur du système comprend un écran et 4 boutons. L'écran utilisé est un écran Nextion. L'affichage est programmé dans le logiciel Nextion Editor, où la disposition et le style des différents éléments sont configurés. L'écran est ensuite contrôlé par le processeur via UART pour afficher le contenu.

Voici la page principale de l'interface. Elle contient une liste des différents devices scannés. Les boutons "up" et "down" permettent de monter et descendre dans la liste. Les trois petits points en bas indiquent si il y a encore des devices non affichés. Le device actuellement sélectionné est grisé (voir figure 41).

Devices Detected :	
<b>Device 100</b>	RSSI : -65
Device Status :	Recording
Nbr of days recorded :	10
<b>Device 23</b>	RSSI : -30
Device Status :	Recording
Nbr of days recorded :	10
<b>Device 231</b>	RSSI : -55
Device Status :	Recording
Nbr of days recorded :	10
...	

FIGURE 41 – Interface - Page principale

Le bouton "select" permet de se connecter au device sélectionné. Lorsque ce bouton est appuyé la page suivante est affichée (voir figure 42).



FIGURE 42 – Interface - Chargement

Ensuite lorsque le device est connecté la page suivante est affichée. Avec les boutons "up" et "down" les options "Open Collar", "Reset & Start Recording", "Toggle Recording" et "Exit" peuvent être choisies (voir figure 43).

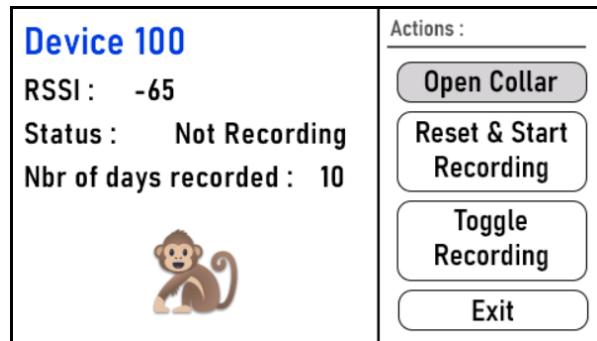


FIGURE 43 – Interface - Page Device

Pour exécuter les différentes actions, il faut appuyer sur le bouton "select". Une confirmation est alors demandée (voir figure 44).

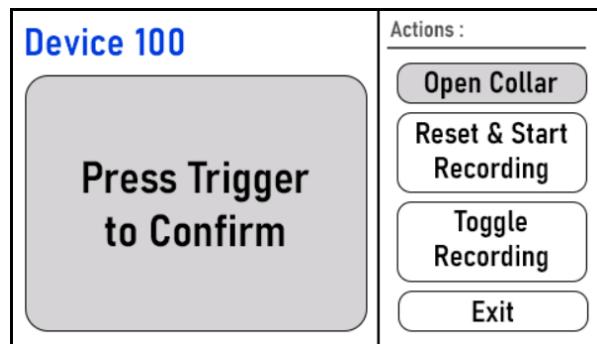


FIGURE 44 – Interface - Popup

Le popup reste actif pendant 5 secondes. Pour confirmer, il faut appuyer sur le bouton "trigger". L'affichage devient alors vert (voir figure 45).

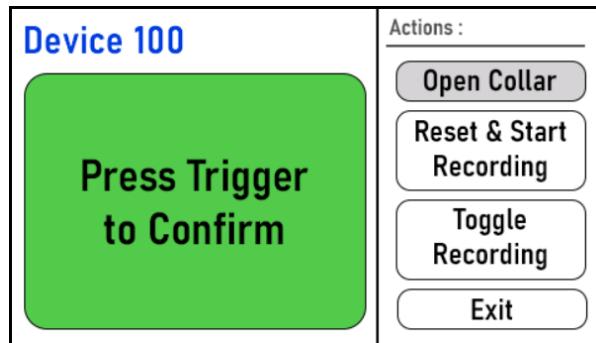


FIGURE 45 – Interface - Confirmation

Le code de contrôle de l'interface est répartit dans 3 composants : "Button Manager", "Display Controller" et "Display".

#### *Display :*

Le code de *display.h* et *display.c* sert à contrôler l'écran. Il utilise un périphérique UART avec l'API polling de Zephyr. Plusieurs fonction sont implémentées. Elles permettent d'afficher les différentes informations sur l'écran. Comme le programme utilise plusieurs threads, les fonctions de display sont protégées par un mutex pour s'assurer que les messages UART soient envoyés sans se couper entre eux. Le diagramme de classe suivant montre les différentes fonctions implémentées par display (voir figure 46).

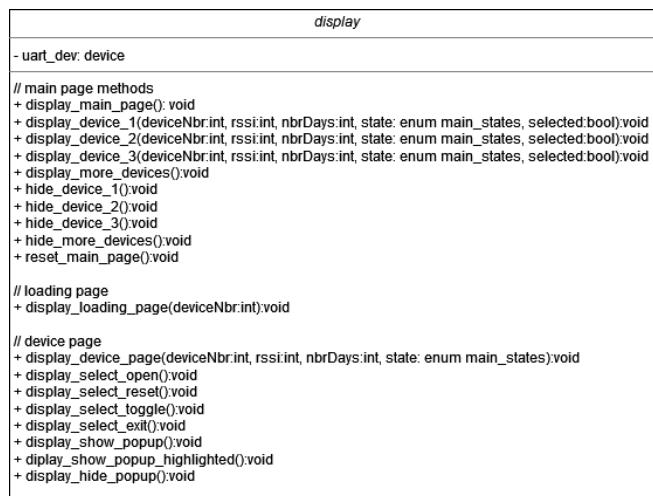


FIGURE 46 – Class Diagram - display

*Display Controller :*

Le code de *display\_controller.h* et *display\_controller.c* sert à contrôler l'interface en générale. La figure 47 montre les différentes fonction implémentées.

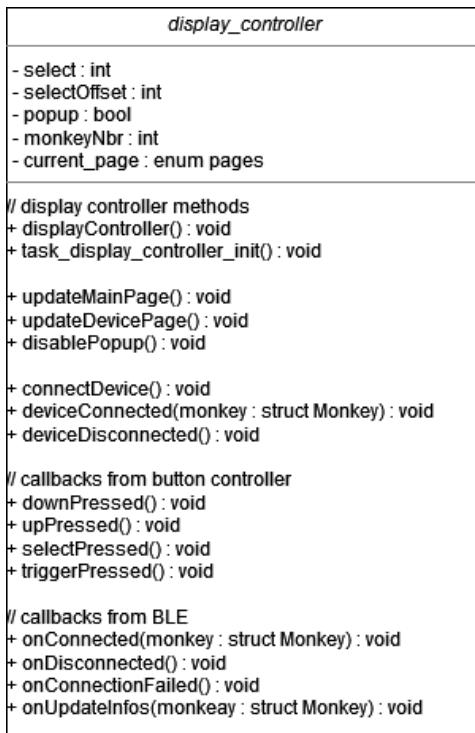


FIGURE 47 – Class Diagram - display controller

La fonction *displayController* est appelée dans un thread. Elle contient la boucle infinie. Dans cette fonction, les fonction *update MainPage* et *update Device Page* sont appelées en fonction de la page actuelle de l'affichage. Quand la page principale est affichée, la liste de devices découverts est affichée. Elle est prise dans Monkeylist. Il y a 4 fonctions qui sont appelées par "button\_manager". Dans la page principale, les boutons "up" et "down" permettent de faire défiler la liste. Le bouton "select" permet de se connecter à un device. Lorsqu'il est pressé, la fonction "connect" de "connection" est appelée, et la page de chargement est affichée. Il y a 4 callback qui sont appelé par "connection" lorsqu'il y a des événements qui proviennent de la partie bluetooth. Lorsque le système s'est connecté, la page device est affichée. Dans cette page, il est possible d'ouvrir le collier, de réinitialiser le système et de permuter l'enregistrement. Lorsqu'une action est sélectionnée, une confirmation est demandée et une fonction de "connection" est appelée.

***Button Manager :***

Le button manager est relativement simple. Lorsqu'un bouton est pressé, une interruption est déclenchée. Dans l'interruption, un work est reschedulé de 15 ms. Ceci permet tout d'abord de faire un anti-rebond, en reprogrammant le work à chaque rebond et deuxièmement de passer un minimum de temps dans l'interruption en utilisant un work, ce qui permet d'éviter de bloquer un autre processus. Le work est inséré dans le scheduling de Zephyr comme les autres tâches du programme.

**6.2.3 Monkeylist :**

La structure de donnée "Monkeylist" contient une liste chaînée de "Monkey". Monkey est un struct qui contient le numéro du device, le rssi, le nombre de jours d'enregistrement, l'état de device, l'adresse bluetooth du device et un pointeur sur le prochain Monkey. Monkeylist contient aussi des fonctions pour agir sur la liste. Dans les fonctions, les accès à la liste sont protégés par un mutex car plusieurs threads peuvent accéder à cette liste. La fonction *appendOrModifyMonkey* contrôle si un "Monkey" avec le même ID (numéro du device) existe déjà. Si c'est le cas, il est modifié avec les nouvelles valeurs, sinon un nouveau est ajouté à la liste. Il y a des fonctions pour accéder à un monkey par ID (numéro du device) et également par index (ordre d'arrivée). La fonction *getThreeMonkeys* permet d'accéder à trois "Monkey" à partir d'un index particulier sans avoir à appeler trois fois la méthode d'accès normale. Elle a été implémentée pour l'affichage, qui affiche trois devices en même temps. La figure 48 montre les différentes fonction implémentées.

<i>monkeylist</i>
- <b>headref</b> : struct Monkey*
+ <b>initMonkeyList()</b> : void
+ <b>appendOrModifyMonkey(num:int, rssi:int, record_time:int, state : enum main_states, address:bt_addr_le_t)</b> : void
+ <b>removeMonkey(num:int)</b> : void
+ <b>getAllMonkeys(monkeysArray : struct Monkey*)</b> : void
+ <b>getMonkeyAtIndex(monkey : struct Monkey*, index : int)</b> : bool
+ <b>getMonkeyByID(monkey : struct Monkey*, id : int)</b> : bool
+ <b>getThreeMonkeys(monkeys : struct Monkey*, startIndex : int)</b> : bool
+ <b>getNumMonkeys()</b> : int

FIGURE 48 – Class Diagram - Monkeylist

### 6.2.4 Bluetooth Low Energy

Dans la communication BLE, le BLE Booster a le rôle de centrale. Il scanne les différents périphériques (colliers), initie la connections et consomme les données fournis par les périphériques. Pour y parvenir le BLE est géré dans un thread dédié : *bleThread*, cela permet à la partie Bluetooth et interface d'être indépendante l'une de l'autre. Les fichiers *ble.h*, *snes\_client* et *snes* permettent d'implémenter la communication BLE.

*BLE* :

Le code de *ble.h* et *ble.c* sert à contrôler la communication Bluetooth Low Energy. La figure 49 montre les différentes fonction implémentées.



FIGURE 49 – Class Diagram - BLE

Une fois les initialisations faites, l'appareil scanne en continue. Lorsqu'un device est détecté, la méthode *ble\_device\_found\_cb* est appelé. Ne s'intéressant que au collier Speak No Evil, seul les périphériques avec l'identifiant de fabriquant correspondant à

0x5A02 (University of Applied Sciences Valais / Haute Ecole Valaisanne) sont ajoutés à la monkeylist. La structure des données fabriquant reçu lors dans un paquet d'advertising sont visibles ci-dessous. Les données du device découvert sont donc accessibles à l'interface (voir figure 50). Si le collier se trouve déjà dans la liste, ses données sont alors mises à jour (rssi, statut et nombre de jours d'enregistrement). Si un collier venait à ne plus être scanné pendant plus de 4 secondes, il est retiré de la liste par la méthode *ble\_remove\_device* qui est appelée en continu dans la fonction d'entrée du thread.

```
1 static uint8_t manufacturer_data[] = {  
2     0x5A, 0x02,          // HEI Company Id  
3     0x00, 0x01, 0x00,    // Firmware revision v0.1.0  
4     0x00,                // Number of days of recording  
5     0x00,                // Status : 0x01 -> waiting for SD Card,  
6                           0x02 -> IDLE, 0x03 -> Recording,  
7                           0x04 -> Low Batt, 0xff -> Error  
8 };
```

FIGURE 50 – Manufacturer data pour le device

Lorsqu'un collier a été scanné, la connexion peut être initiée depuis l'interface. Dans un premier temps, pour la gestion de la connexion et des échanges de données, les exemples suivants ont été suivis : central bluetooth (Zephyr, [s. d.(a)]) et smart button central (Michael angerer, [s. d.]). La connexion était possible mais il était ensuite impossible de découvrir les services du périphérique et donc de l'échanger des données avec ce dernier. Le collier se déconnectait après un timeout (raison 8). Pour corriger cette erreur, il a fallu changer la manière de gérer la communication. Pour se faire, un service BLE GATT avec un client SNES (Speak No Evil Service) a été utilisé. A l'aide du client SNES, la découverte des services GATT peut être effectuée lorsque la connexion est établie en appelant la fonction *bt\_gatt\_dm\_start*. Le central peut donc s'abonner aux différentes notifications et envoyer des commandes au collier. Des callbacks permettent de gérer les données reçues des notifications SNES et les désabonnements des notifications.

Le kit de développement utilise un module front-end embarqué qui permet d'amplifier le signal RF, pour augmenter la distance de portée, la force et la robustesse d'une connexion de liaison. Pour activer le FEM (front-end module), il faut le configurer dans le device tree ainsi que modifier les configurations du projet (Nordic Semiconductor, [s. d.(f)]). Lorsqu'il était activé, on ne voyait pas de différence au niveau des RSSI des devices scannés. Mais lorsqu'on essayait de se connecter à un device, la connexion était à chaque fois interrompue. Il était difficile de déboguer cette partie car il n'y a pas de moyen de savoir si le FEM est actif mis à part les performances du système. Il fallait donc comparer le comportement du BLE Booster avec et sans le FEM. Comme il restait peu de temps pour le projet, il a été décidé de mettre le temps restant sur les tests et la fiabilisation du système au détriment du FEM. Le problème de connexion n'a donc pas encore été réglé. Et le système actuel fonctionne sans le FEM.

**SNES :**

Ce fichier d'en-tête définit les UUID du service ainsi que les caractéristiques associées, telles que la commande, le statut, le nombre de jours d'enregistrement, l'identifiant de l'appareil et le gain d'entrée du microphone (voir figure 51). De plus, le fichier contient des structures et des fonctions pour gérer les connexions Bluetooth, envoyer des commandes, mettre à jour les valeurs des caractéristiques et informer les appareils connectés des changements d'état. Il définit aussi une structure de callbacks qui sont implémentés dans le client SNES.

snes
+ bt_snes_init(callbacks: bt_snes_cb*) int + bt_snes_cmd_send(conn: bt_conn*, data: uint8_t*, len: uint16): int + bt_snes_update_status_cb(status: uint8_t): int + bt_snes_update_days_of_records_cb(dor: uint8_t): int + bt_snes_update_device_identifier_cb(device_id: uint8_t): int + bt_snes_update_mic_input_gain_cb(input_gain: uint8_t): int + bt_snes_get_mtu(conn: bt_conn*: uint32_t + on_snes_connected(conn: bt_conn*: void + on_snes_disconnected(): void

FIGURE 51 – Class Diagram - SNES

**Client SNES :**

L'implémentation de ce client se base sur l'exemple central uart (Nordic Semiconductor, [s. d.(a)]). Il a été adapté au besoin du projet par Patrice Rudaz. Cette solution permet de simplifier le code contenu dans le fichier ble comme le client est adapté au spécifications du projet.

La classe *snes\_client* implémente un client Bluetooth GATT (Generic Attribute Profile) pour interagir avec un service SNES (voir figure 52). Elle gère les abonnements aux notifications pour les caractéristiques du service SNES suivants :

- Status : status
- Jour d'enregistrement : dor
- Identifiant du device : device\_id
- Gain du microphone : mic\_gain

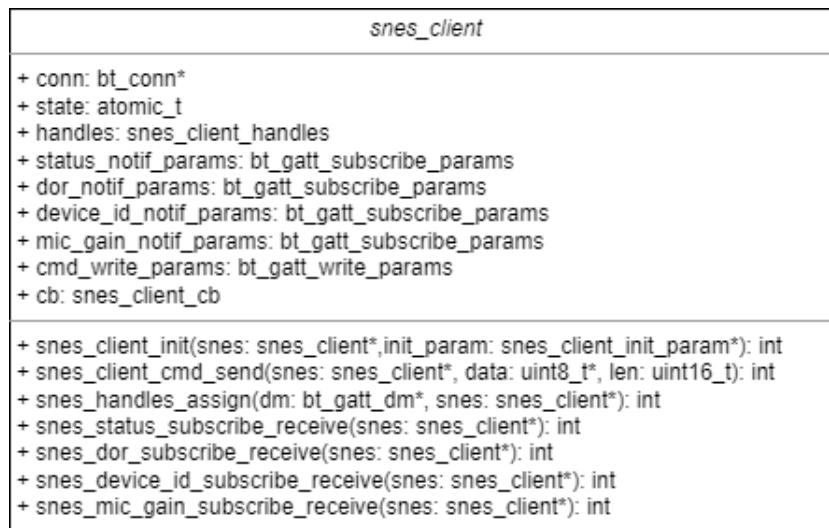


FIGURE 52 – Class Diagram - Client SNES

Les fonctions de rappel (callbacks) sont définies pour traiter les notifications reçues et les désabonnements éventuels. Lorsqu'une donnée est reçue via une notification, le module vérifie si une fonction de rappel correspondante est définie et l'appelle avec les données reçues. Les fonctions de la classe incluent également l'initialisation du client, l'envoi de commandes vers le serveur SNES, et l'affectation des handles des caractéristiques et des CCC (Client Characteristic Configuration) lors de la découverte du service. Les commandes présentées dans la figure 53 sont possible. Un message est toujours composé de deux bytes : le header et la commande. Le BLE booster peut seulement envoyer une commande pour ouvrir le collier (*BLE\_MSG\_OPEN\_COLLAR*), remettre à zéro le device (*BLE\_MSG\_RESET\_DEVICE*) et enclencher/déclencher l'enregistrement (*BLE\_MSG\_TOGGLE\_RECORDING*). Les autres commandes sont utilisés par l'application mobile.

```

1 #define BLE_MSG_HEADER 0xa5
2 #define BLE_MSG_OPEN_COLLAR 0x01
3 #define BLE_MSG_RESET_DEVICE 0x02
4 #define BLE_MSG_TOGGLE_RECORDING 0x03
5 #define BLE_MSG_SHUTDOWN_HARDWARE 0x04
6 #define BLE_MSG_START_HARDWARE 0x05
7 #define BLE_MSG_SEND_CURRENT_TIME 0x06
8 #define BLE_MSG_SEND_NEW_DEVICE_IDENTIFIER 0x07
9 #define BLE_MSG_SEND_NEW_MIC_INPUT_GAIN 0x08
10 #define BLE_MSG_CONFIG_BT_CTS_CLIENT 0xff

```

FIGURE 53 – Commandes

### 6.2.5 Séquences

La figure 54 montre la séquence de connections à un collier :

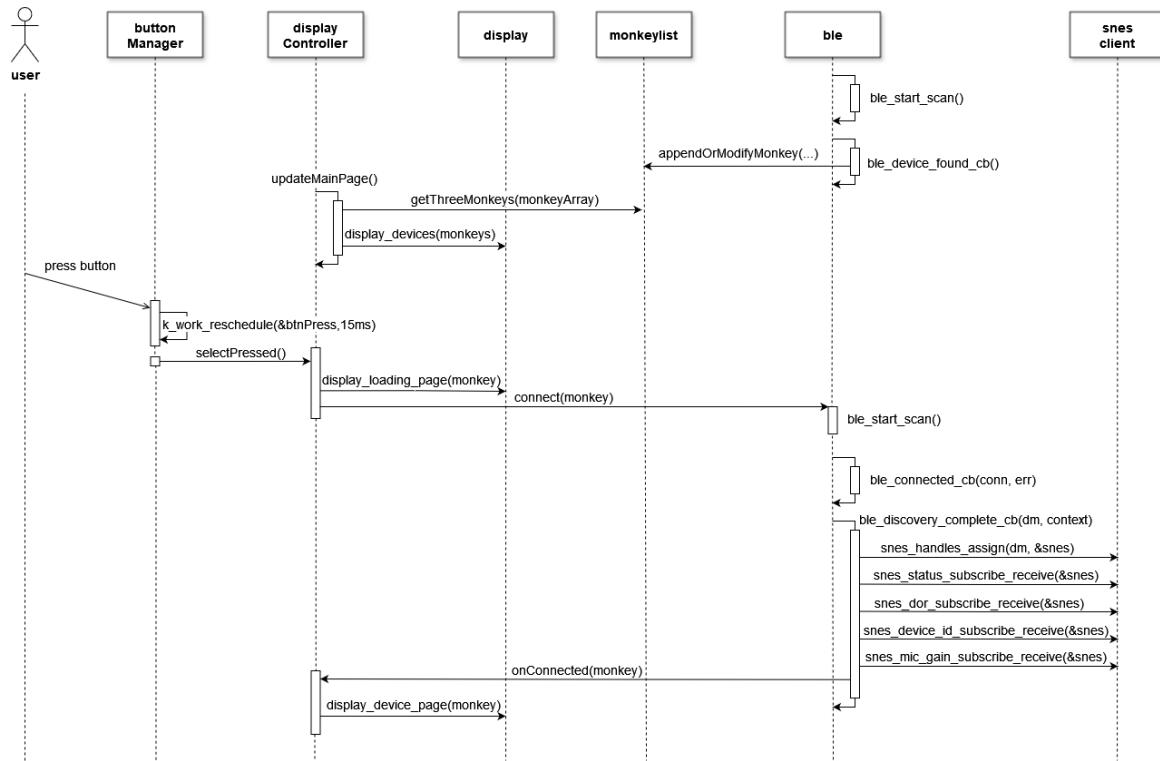


FIGURE 54 – Séquence de connexion

Dans un premier temps, les colliers disponibles sont ajouté dans "monkeylist" par "ble". En parallèle, les colliers enregistré dans "monkeylist" sont affichés à l'écran par "display\_controller". Lorsque l'utilisateur appuie sur le bouton pour se connecter à un collier, la connection est initiée par "ble". Lorsque la connexion est réussie, le callback "onConnect" de "display\_controller" est appelé. Si la connexion échoue, le callback "onConnectionFailed" sera appelé à la place.

La figure 55 montre la séquence de l'envoi de la commande d'ouverture du collier :

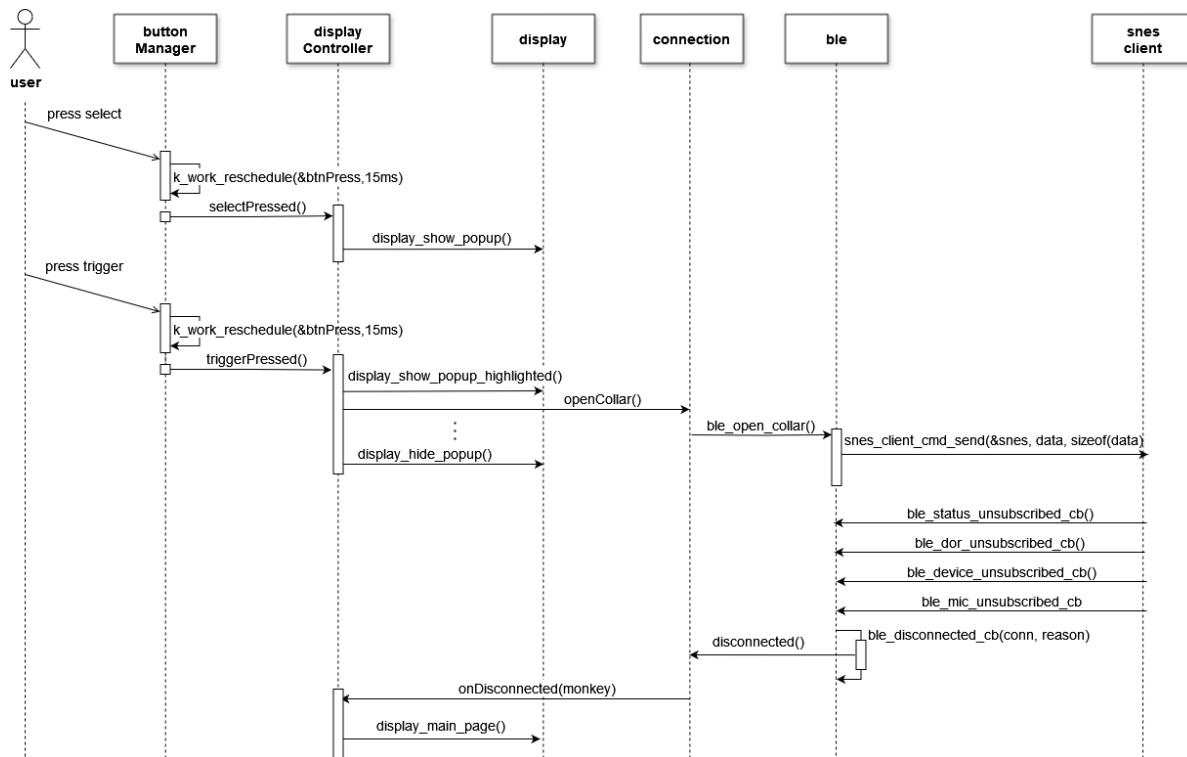


FIGURE 55 – Séquence d'envoi d'une commande

Lorsque l'utilisateur souhaite ouvrir le collier, il doit dans un premier temps sélectionner la commande. Ensuite, un popup apparaît sur l'écran et l'utilisateur peut appuyer sur le déclencheur pour confirmer la commande. A ce moment, le popup devient vert, pour indiquer que la commande a bien été prise en compte et disparaît après quelques instants. Pendant ce temps, la commande est envoyée à "ble" et le message est transmit au collier. Une fois que le collier a été ouvert, il est déconnecté et "ble" appelle la méthode de callback "onDisconnected" de "display\_controller", qui va ré-afficher la page principale sur l'interface graphique.

## 6.3 Hardware

Le hardware se base sur le kit nRF51240-DB. Une petite carte d'extension a été ajoutée sur les gpios du kit. Elle contient un fil qui se branche sur l'entrée VIN du kit pour l'alimentation du système, un connecteur d'alimentation d'entrée, quatre connecteurs pour les boutons et un connecteur pour l'écran. Comme le connecteur de l'antenne ne tient pas bien, une petite pièce imprimée en 3D a été dessinée pour tenir le connecteur.

Le système inclut également une batterie. Un module qui contient 4 liion 18650 a été utilisé pour cela (voir figure 56). Il est rechargeable par USB et fournit une alimentation 5 V 3 A. Le port USB a été déporté sur une paroi du boîtier et le bouton pour allumer et éteindre la batterie a été déporté sur le dessus. Pour allumer le système il faut appuyer sur le bouton, et pour l'éteindre, il faut faire un double appui sur le bouton.



FIGURE 56 – BLE Booster - batterie

L'écran et les boutons "up", "down" et "select" sont fixés sur le couvercle transparent du boîtier. Le bouton "trigger" est placé sur le dessus du boîtier. La batterie et le kit nRF51240-DB sont fixés sur une plaque, elle même fixée au fond de boîtier. L'antenne est fixée à l'arrière du boîtier et est orientée à 45° vers le haut grâce à une pièce d'adaptation. La pièce d'adaptation inclue aussi une fixation pour poignée.

Les figures suivantes montrent l'appareil sous plusieurs angles :

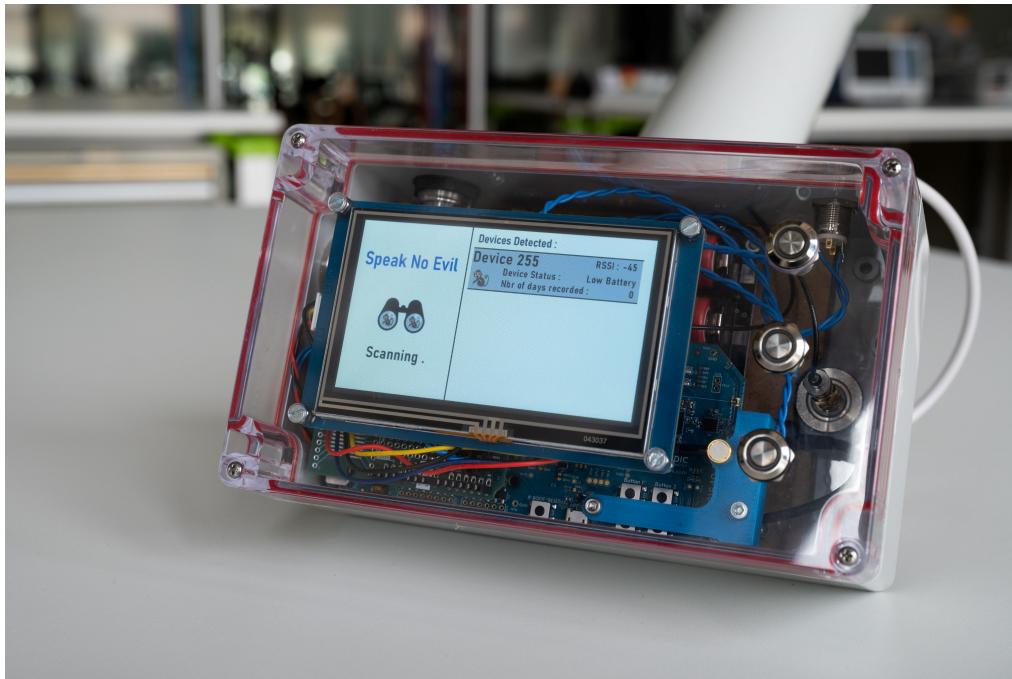


FIGURE 57 – BLE Booster - photo



FIGURE 58 – BLE Booster - photo - antenne

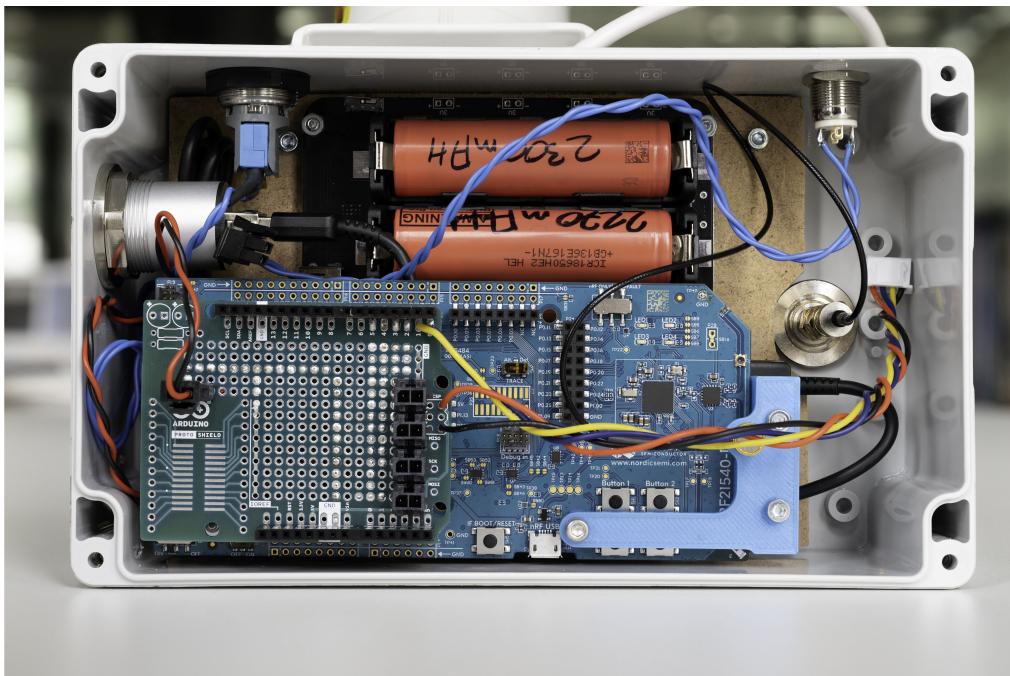


FIGURE 59 – BLE Booster - photo - intérieur

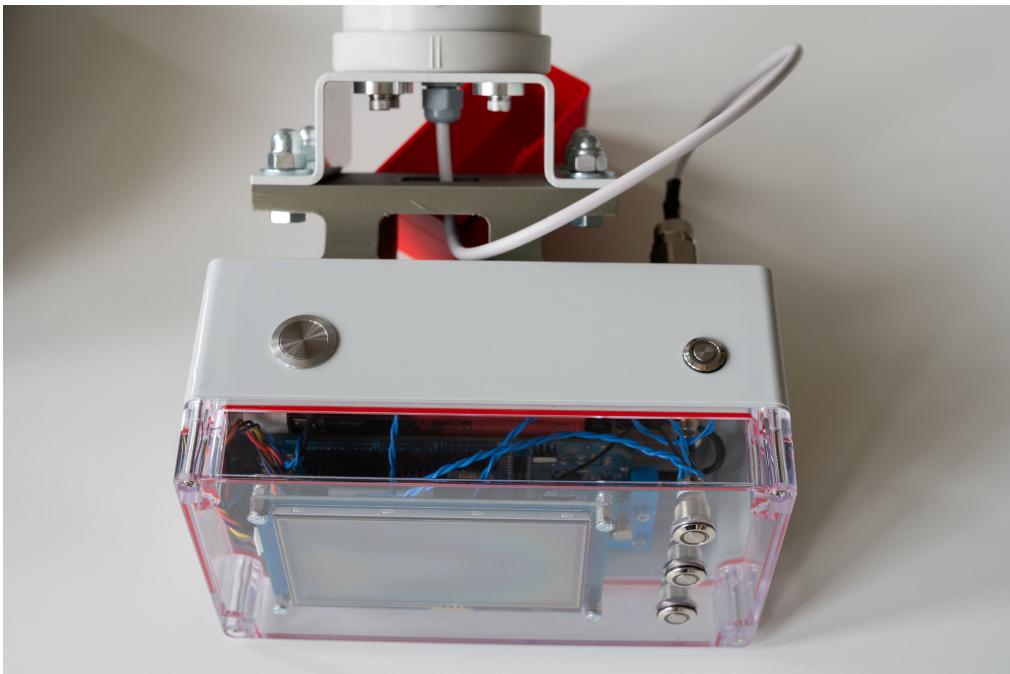


FIGURE 60 – BLE Booster - dessus



FIGURE 61 – BLE Booster - photo - port de recharge

## 6.4 Tests de portée

Des tests de portées ont été effectués dans le but de comparer le fonctionnement du BLE Booster et du smartphone. Ces tests sont indicatifs et ne garantissent en aucun cas un fonctionnement au distance testées dans toutes les situations. La portée dépend forttement des conditions météorologiques, de l'environnement et des obstacles. Deux situations ont été testés : en ligne droite sans obstacles et en ligne droite avec des arbres entre le collier et l'appareil. Trois devices ont servis pour ce test :

- Speak No Evil avec une batterie 1/2AA montés dans son boîtier (N°110)
- Speak No Evil avec une batterie AA sans boîtier (N°54)
- Kit nRF5340 branché sur un laptop qui permet d'avoir les log du collier (N°255)

*Tests portée sans obstacles :*

	N° device	40m en intérieur			80m en extérieur			150m en extérieur		
		Scan	Connection	Ecriture	Scan	Connection	Ecriture	Scan	Connection	Ecriture
BLE Booster sans FEM	110	✓	✓	✓	✓	✓	✓	✓	✓	✗
	54	✓	✓	✓	✓	✓	✓	✓	✓	✗
	255	✓	✓	✓	✓	✓	✓	✓	✓	✓
Smartphone	110	✓	✓	✓	✓	✗	✗	✗	✗	✗
	54	✓	✓	✓	✗	✗	✗	✗	✗	✗
	255	✓	✓	✓	✓	✗	✗	✗	✗	✗

FIGURE 62 – Ligne droite sans obstacles

Tests portée avec obstacles :

	N° device	40m en extérieur			50m en extérieur		
		Scan	Connection	Ecriture	Scan	Connection	Ecriture
BLE Booster sans FEM	110	✓	✓	✓	✓	✓	✓
	54	✓	✓	✓	✓	✓	✓
Smartphone	110	✓	✓	✓	✗	✗	✗
	54	✓	✓	✓	✓	✗	✗

FIGURE 63 – Ligne droite avec obstacles

Pour ces tests les obstacles étaient des arbres. Le but était d'essayer de se rapprocher d'un cas réel pouvant survenir lors de l'utilisation du BLE Booster. La figure 64 montre l'environnement du test. Le cercle rouge indique la position approximative des colliers lors des tests.



FIGURE 64 – Environnement tests portée avec obstacles

Ces tests permettent de se rendre compte de l'utilité du BLE Booster. La portée du système est meilleure que celle du téléphone. Il faudra tout de fois le tester dans des conditions réelles pour connaître son vrai comportement. Les tests ont permis de relever un point important. L'antenne directionnelle augmente la portée du système mais elle oblige aussi à l'utilisateur à toujours la pointé en direction du singe. A une grande distance, il devient difficile de pointer correctement le collier et de maintenir la position suffisamment longtemps pour permettre la connection et l'envoie de commande. De plus si les singes sont en mouvement, la tâche devient encore plus ardu.

## 7 Conclusion

### 7.1 Batterie

Les tests effectués sur les deux modèles de batterie proposés ont permis de sélectionner un nouveaux modèle : *Tadiran Lithium 2/3AA 3.6 V (SL-761)*. Cela permet de gagner 17 mm et 6 g sur le modèle précédemment utilisé. Ce gain de taille et de poids est considérable et a permis de nettement diminuer la taille du boîtier. L'ajout du supercondensateur en parallèle de la batterie devra permettre d'éviter les problèmes de sur-courants et de chutes de tensions. Cela permet de rendre le système global plus fiable.

### 7.2 Mécanique

En ce qui concerne le release system le choix pour le prototype final à l'issue de ce travail s'est porté sur la solution du fil chaud, car s'est une valeur sûre en terme de robustesse de verrouillage et d'efficacité de libération du collier. De plus, les deux variantes associées à ce système ont permis une relativement bonne miniaturisation et une potentielle facilitation de mise en oeuvre. La version la plus petite reste toutefois à tester avant d'être validée. Selon le résultat une version optimale de taille intermédiaire pourrait être étudiée à posteriori.

Un potentiel certain existe pour l'électro-aimant modifié sur mesure, avec une augmentation de force calculée prometteuse, dans un encombrement demeurant raisonnable. En effet, en terme de longueur le dispositif reste très proche de la version avec fil chaud, pour une largeur légèrement supérieure de quelques petits millimètres. Mais cela serait un prix à payer tout à fait acceptable étant donné l'énorme gain qui serait apporté en terme de facilité de mise en oeuvre et d'utilisation. Encore une fois une phase de test itérative devra être mise en place pour une version future afin de trouver la composition idéale du solénoïde.

Pour ce qui est du boîtier, les deux versions répondent de manière satisfaisante aux objectifs dimensionnels et massiques (pour la variante 2 qui est plus lourde : **52.87 g** y compris le collier sans GNSS). La deuxième variante, dont l'encombrement, qui, bien qu'il soit très légèrement supérieur (**47 x 35 x 26 mm**), reste très proche de la première variante (**45 x 30 x 30 mm**), tout en offrant une excellente robustesse d'assemblage, une bonne étanchéité sans collage, et une certaine facilité d'accès aux composants et de montage. Cependant l'objectif premier étant un poids et un encombrement minimal, la version cylindrique est le meilleur choix grâce à sa géométrie qui permet un placement optimal des composants dans l'espace. Le joint d'étanchéité par simple collage au silicone évite une épaisseur plus grande des parois nécessaire pour accueillir une rainure et un joint torique. Une taille encore plus petite pourrait être visée dans le futur en remplaçant la variante 1 du release system par la version encore plus miniaturisée (variante 3).

## 7.3 Électronique

L'objectif de la partie électronique était principalement de réduire la taille de l'électronique embarquée du collier, en collaboration avec la mécanique pour le boîtier.

Deux variantes de design des PCB ont été proposées, chacune adaptée à une version de conception mécanique. Le choix est laissé pour décider de la variante la mieux adaptée. La première variante contient des parties de PCB flexible, et nécessite donc la possibilité de faire du semi-flex pour la fabrication.

En outre, le support de la carte SD a été remplacé par un nouveau, plus résistant aux vibrations et aux chocs, qui avaient posé quelques problèmes avec la première version du système.

Nous avons aussi évalué l'utilisation d'un SoC LoRa, ainsi que celle d'une mémoire eMMC. Les deux possibilités affichent des avantages notables, mais d'importantes modifications sont requises. De ce fait, elles n'ont pas été implémentées, mais laissées en tant que propositions pour d'éventuelles futures versions.

## 7.4 GNSS

L'objectif de la partie GNSS était de voir s'il était possible d'intégrer un module GNSS au collier.

Le module GNSS choisi est le M10 fait par u-blox, car le modèle M8 est moins performant dû à son âge.

Dans nos tests, la réception était impactée par les signaux devant traverser la chair, ce qui se traduit à la posture du singe, et la disposition de l'environnement. Nous ne savons pas comment la réception sera affecté par les conditions de la jungle.

La consommation de courant du module GNSS est élevée, mais si nous restreignons l'usage du GNSS à une fois par heure, avec une limite de 2 min pour fixer sa position, l'impacte sur la durée de vie de la pile sera gérable. Selon les besoins, la limite pourrait être augmentée à 4 min, la pile tenant presque 13 jours dans ces conditions.

Le choix de communication se détermine par l'effort que nous souhaitons y investir. L'API faciliterait l'implémentation du code, mais faire fonctionner l'API avec le SDK Nordic pourrait prendre du temps. Nous recommandons de ne pas passer par l'API et d'utiliser directement la fonction qui décode UBX. Le protocole UBX est nécessaire pour configurer le module GNSS, mais envoie des messages de position de plus grande taille que le protocole NMEA. Cependant NMEA aurait besoin de code supplémentaire à celui de la fonction qui décode UBX.

Nous pensons que l'intégration d'un module GNSS est possible. Cependant, il faudra aller sur le terrain et faire des tests de réception et de consommation, ce qui permettrait de déterminer un budget plus précis pour le temps d'utilisation du module GNSS.

## 7.5 BLE Booster

Le développement du BLE Booster avait pour but d'améliorer la portée du système de contrôle des colliers via Bluetooth Low Energy (BLE), en remplaçant le smartphone par un appareil dédié qui intègre une antenne directionnelle et un kit de développement nRF21540-DK.

Les tests effectués ont révélé que le BLE Booster offre une portée supérieure à celle du smartphone. Cependant, il est crucial de noter que la performance de l'appareil peut varier en fonction des conditions environnementales et des obstacles présents. L'antenne directionnelle, bien qu'elle améliore la portée, impose une contrainte supplémentaire à l'utilisateur qui doit constamment orienter l'antenne vers le collier, ce qui peut s'avérer difficile à longue distance ou lorsque les singes sont en mouvement.

L'appareil développé reprend la majorité des fonctionnalités de l'application mobile existante, à l'exception de la mise à jour de la date et de l'heure des colliers, le smartphone devra donc toujours être utilisé lors de la configuration des colliers.

En conclusion, le BLE Booster représente une solution prometteuse pour améliorer le contrôle des colliers dans des environnements variés. Sa portée pourrait encore être augmenté par l'utilisation du module FEM. Des tests en conditions réelles seront nécessaires pour évaluer pleinement ses performances et identifier d'éventuelles améliorations à apporter.

## 8 Bibliographie

### Références

- ALPENHUNDE, [s. d.]. *How do I store configuration item in the flash memory of an M10S. I only get the red triangle in the U-center software.* [en ligne]. Alpenhunde. [visité le 2024-06-17]. Disp. à l'adr. : <https://portal.u-blox.com/s/question/0D52p0000DgwUF0CQ2/how-do-i-store-configuration-item-in-the-flash-memory-of-an-m10s-i-only-get-the-red-triangle-in-the-ucenter-software>.
- CUPBUS105113, [s. d.]. *MAX-M10S not saving to flash in U-Center2* [en ligne]. cupbus105113. [visité le 2024-06-17]. Disp. à l'adr. : <https://portal.u-blox.com/s/question/0D52p0000D5eZkoCQE/maxm10s-not-saving-to-flash-in-ucenter2>.
- HEVS, [s. d.]. *nRFMonkey gitlab project* [en ligne]. HEVS. [visité le 2024-06-17]. Disp. à l'adr. : <https://gitlab.hevs.ch/patrice.rudaz/nrf-monkey>.
- JOHN BISELX, [s. d.(a)]. *SerialReaderGNSS GitHub project* [en ligne]. HEVS. [visité le 2024-06-19]. Disp. à l'adr. : <https://github.com/PI-MobileSens/SerialReaderGNSS>.
- JOHN BISELX, [s. d.(b)]. *UARTpassthroughGNSS GitHub project* [en ligne]. HEVS. [visité le 2024-06-19]. Disp. à l'adr. : <https://github.com/PI-MobileSens/UARTpassthroughGNSS>.
- KJARTANOLI, [s. d.]. *uGnssPosGet sometimes fails with U ERROR COMMON NOT INITIALISED* [en ligne]. KjartanOli. [visité le 2024-06-17]. Disp. à l'adr. : <https://github.com/u-blox/ubxlib/issues/237>.
- MAYANKMAHAJAN-NXP, [s. d.]. *Zephyr UBX support* [en ligne]. Zephyr. [visité le 2024-06-19]. Disp. à l'adr. : <https://github.com/zephyrproject-rtos/zephyr/pull/68350>.
- MICHAEL ANGERER, [s. d.]. *Smart button central example* [en ligne]. Michael angerer. [visité le 2024-06-12]. Disp. à l'adr. : [https://github.com/nrfconnect/sdk-nrf/tree/main/samples/bluetooth/central\\_uart](https://github.com/nrfconnect/sdk-nrf/tree/main/samples/bluetooth/central_uart).
- MOS216, [s. d.]. *Unable to bring up GNSS!* [en ligne]. mos216. [visité le 2024-06-17]. Disp. à l'adr. : <https://github.com/u-blox/ubxlib/issues/241>.
- NORDIC SEMICONDUCTOR, [s. d.(a)]. *Central uart example* [en ligne]. Nordic Semiconductor. [visité le 2024-06-12]. Disp. à l'adr. : [https://github.com/nrfconnect/sdk-nrf/tree/main/samples/bluetooth/central\\_uart](https://github.com/nrfconnect/sdk-nrf/tree/main/samples/bluetooth/central_uart).
- NORDIC SEMICONDUCTOR, [s. d.(b)]. *nRF Connect for Desktop* [en ligne]. Nordic Semiconductor. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop>.
- NORDIC SEMICONDUCTOR, [s. d.(c)]. *nRF Connect SDK* [en ligne]. Nordic Semiconductor. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.nordicsemi.com/Products/Development-software/nRF-Connect-SDK>.

NORDIC SEMICONDUCTOR, [s. d.(d)]. *nRF5340 DK Product Page* [en ligne]. Nordic Semiconductor. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.nordicsemi.com/Products/Development-hardware/nRF5340-DK>.

NORDIC SEMICONDUCTOR, [s. d.(e)]. *Power Profiler Kit II* [en ligne]. Nordic Semiconductor. [visité le 2024-06-15]. Disp. à l'adr. : <https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2>.

NORDIC SEMICONDUCTOR, [s. d.(f)]. *Working with RF front-end modules* [en ligne]. Nordic Semiconductor. [visité le 2024-06-13]. Disp. à l'adr. : [https://docs.nordicsemi.com/bundle/ncs-2.5.3/page/nrf/device\\_guides/working\\_with\\_fem.html](https://docs.nordicsemi.com/bundle/ncs-2.5.3/page/nrf/device_guides/working_with_fem.html).

U-BLOX, [s. d.(a)]. *CAM-M8 series* [en ligne]. u-blox. [visité le 2024-06-12]. Disp. à l'adr. : <https://www.u-blox.com/en/product/cam-m8-series>.

U-BLOX, 2023a. *IoT Location-as-a-Service* [en ligne]. u-blox. [visité le 2024-06-12]. Disp. à l'adr. : [https://content.u-blox.com/sites/default/files/documents/SAM-M10Q\\_IntegrationManual\\_UBX-22020019.pdf](https://content.u-blox.com/sites/default/files/documents/SAM-M10Q_IntegrationManual_UBX-22020019.pdf).

U-BLOX, [s. d.(b)]. *RCB-F9T timing board Product Page* [en ligne]. u-blox. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.u-blox.com/en/product/rcb-f9t-timing-board>.

U-BLOX, [s. d.(c)]. *SAM-10Q module* [en ligne]. u-blox. [visité le 2024-06-12]. Disp. à l'adr. : <https://www.u-blox.com/en/product/sam-m10q-module>.

U-BLOX, [s. d.(d)]. *SAM-M10Q - Integration manual* [en ligne]. u-blox. [visité le 2024-06-12]. Disp. à l'adr. : <https://portal.thingstream.io/pricing>.

U-BLOX, 2023b. *u-blox M10 SPG 5.10 - Interface description* [en ligne]. u-blox. [visité le 2024-06-15]. Disp. à l'adr. : [https://content.u-blox.com/sites/default/files/u-blox-M10-SPG-5.10\\_InterfaceDescription\\_UBX-21035062.pdf](https://content.u-blox.com/sites/default/files/u-blox-M10-SPG-5.10_InterfaceDescription_UBX-21035062.pdf).

U-BLOX, [s. d.(e)]. *u-center Product Description* [en ligne]. u-blox. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.u-blox.com/en/product/u-center>.

U-BLOX, [s. d.(f)]. *UBX-F10 series Product Page* [en ligne]. u-blox. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.u-blox.com/en/product/ubx-f10-series>.

U-BLOX, [s. d.(g)]. *ubxlib GitHub* [en ligne]. u-blox. [visité le 2024-06-20]. Disp. à l'adr. : [https://github.com/u-blox/ubxlib/blob/master/common/ubx\\_protocol/src/u\\_ubx\\_protocol.c](https://github.com/u-blox/ubxlib/blob/master/common/ubx_protocol/src/u_ubx_protocol.c).

U-BLOX, [s. d.(h)]. *ubxlib : u-blox host library* [en ligne]. u-blox. [visité le 2024-06-17]. Disp. à l'adr. : <https://www.u-blox.com/en/product/ubxlib>.

U-BLOX, [s. d.(i)]. *ubxlib/example/gnss/pos\_main.c* [en ligne]. u-blox. [visité le 2024-06-17]. Disp. à l'adr. : [https://github.com/u-blox/ubxlib/blob/master/example/gnss/pos\\_main.c](https://github.com/u-blox/ubxlib/blob/master/example/gnss/pos_main.c).

U-BLOX, [s. d.(j)]. *ubxlib/port/platform/zephyr/README* [en ligne]. u-blox. [visité le 2024-06-17]. Disp. à l'adr. : <https://github.com/u-blox/ubxlib/tree/master/port/platform/zephyr>.

ZEPHYR, [s. d.(a)]. *Central example* [en ligne]. Zephyr. [visité le 2024-06-12]. Disp. à l'adr. : [https://github.com/nrfconnect/sdk-nrf/tree/main/samples/bluetooth/central\\_uart](https://github.com/nrfconnect/sdk-nrf/tree/main/samples/bluetooth/central_uart).

ZEPHYR, [s. d.(b)]. *Modules (External Projects)* [en ligne]. Zephyr. [visité le 2024-06-19]. Disp. à l'adr. : <https://docs.zephyrproject.org/latest/develop/modules.html#integrate-modules-in-zephyr-build-system>.

ZEPHYR, [s. d.(c)]. *ubxlib/port/platform/zephyr/default.conf* [en ligne]. Zephyr. [visité le 2024-06-19]. Disp. à l'adr. : <https://github.com/u-blox/ubxlib/blob/master/port/platform/zephyr/default.conf>.

ZEPHYR, [s. d.(d)]. *West Manifests – Zephyr* [en ligne]. Zephyr. [visité le 2024-06-19]. Disp. à l'adr. : <https://docs.zephyrproject.org/latest/develop/west/manifest.html#manifest-imports>.

ZEPHYR, [s. d.(e)]. *zephyr/submanifests/example.yaml.sample* [en ligne]. Zephyr. [visité le 2024-06-19]. Disp. à l'adr. : <https://github.com/zephyrproject-rtos/zephyr/blob/main/submanifests/example.yaml.sample>.

## Annexes

### Batterie

- Datasheet batterie SL360
- Datasheet batterie SL361
- Datasheet batterie SL761

### Mécanique

- Fichier STEP (Variante 1 Rittiner.stp) de la variante 1 (Boitier et Release System)
- Dossier zip (Boitier Idee1.zip) comprenant les fichiers Solidworks de la variante 2 du Release System intégré dans la première itération de la deuxième variante du boîtier, et son fichier STEP correspondant (Boitier Idee1.STEP)
- Dossier zip (Boitier Idee2.zip) comprenant les fichiers Solidworks de la variante 3 du Release System intégré dans l'itération finale de la deuxième variante du boîtier, et son fichier STEP correspondant (Boitier Idee2.STEP)
- Dossier zip (ReleaseSystem Idee3.zip) comprenant les fichiers Solidworks de la variante 2 home made du Release System, et son fichier STEP correspondant (ReleaseSystem Idee3.STEP)

### Électronique

- Fichiers Altium pour la variante 1 de l'électronique
- Fichiers Altium pour la variante 2 de l'électronique
- Fichiers Altium pour la variante 3 du release system (mécanique variante 2)

### BLE Booster

- Code complet nRF
- Configuration écran Nextion
- Diagramme UML de classe et de séquence

## GNSS

Pour plus informations, voir section 5.6 :

- Un code exemple (DecodeGNSS.zip) qui démontre comment décoder des messages UBX envoyés au CPU par le module GNSS
- Un code exemple (UARTpassthroughGNSS.zip) qui démontre comment recevoir des messages du module GNSS via un pont UART
- Un code python (SerialReaderGNSS.zip) qui permet de lire les valeurs envoyées par les modules GNSS M8 et M10 et de les stockés dans un fichier .csv

Ce qui suit est une démarche à suivre pour intégrer la librairie ubxlib au nRF Connect SDK. Le manque de documentation claire à ce sujet est étonnant mais ces étapes semblent fonctionner plus que les autres manières étudiées.

1. Toutes les commandes west utilisées dans la console se basent sur un seul fichier qui se trouve dans le répertoire suivant à l'installation locale de nRF Connect SDK : ncs/nrf/west.yml
2. Une fois que ce fichier a été localisé ; il suffit de rajouter les lignes suivantes :

```
1 path: nrf
```

Et juste en dessous :

```
1 import: "../submanifests"
```

Cela indique à west qu'il faut chercher un dossier appelé submanifests dans le dossier du SDK ((Zephyr, [s. d.(e)]), (Zephyr, [s. d.(d)]), (Zephyr, [s. d.(b)])).

3. Créer le dossier appelé submanifests dans votre installation du SDK 2.4.0 locale et y rajouter un fichier .yml avec le contenu suivant :

```
1 manifest:
2   remotes:
3     - name: u-blox
4       url-base: https://github.com/u-blox
5
6   projects:
7     - name: ubxlib
8       remote: u-blox
9       path: ubxlib
10      import: true
11      clone-depth: 1
```

Ce fichier est basé sur celui-ci (u-blox, [s. d.(j)]).

4. Retournez maintenant à VSCode et cliquez sur Manage Toolchains dans l'extension nRFConnect. Cliquez sur Open Terminal Profile. Cela donne accès à la ligne de commande de l'environnement virtuel python utilisé par west.

5. Lancer 'west update' devrait télécharger ubxlib de façon automatique (vous pouvez vérifier qu'il existe maintenant un dossier appelé ubxlib dans l'installation locale du SDK v2.4.0).
6. Le build de l'application avec ubxlib devrait maintenant fonctionner. Il suffit de rajouter les bons paramètres dans prj.CONF (voir (Zephyr, [s. d.(c)])).