

ECOLE DES MÉTIERS DE LAUSANNE
ECOLE SUPÉRIEURE

GÉNIE ÉLECTRIQUE

SYSTÈME D'ENREGISTREMENT DE TRAJECTOIRES DE VOL

Boîte noire miniaturisée

Auteur

Ali ZOUBIR

Superviseur

Juan José MORENO

Mandant

ASSOCIATION POUR LE MAINTIEN
DU PATRIMOINE AÉRONAUTIQUE

10 septembre 2023

Table des matières

Acronymes	5
Glossaire	5
1 Cahier des charges	6
1.1 Introduction	6
1.2 Aperçu	6
1.3 Tâches à réaliser	6
1.4 Schéma de principe	7
1.5 Planification	8
1.6 Livrable	8
2 Pré-étude	9
2.1 Fonctionnement du système	9
2.2 Schéma bloc	9
2.3 Description des blocs	9
2.4 Choix des composants et technologies	10
2.4.1 Microcontrôleur	10
2.4.2 Centrale inertielle	11
2.4.3 GPS / GNSS	12
2.4.4 Carte SD	13
2.4.4.1 Estimation de la capacité	13
2.5 Batterie, autonomie, charge et régulation	14
2.5.1 Technologies	14
2.5.2 Choix de la batterie	14
2.5.3 Régulateur de charge	15
2.6 Systèmes d'économie d'énergies	16
2.7 Diagramme des états du système	16
2.8 Estimation du coût du prototype	17
2.9 Conclusion et perspectives de la Pré-étude	17
3 Développement du schéma électronique	18
3.1 Blocs développés	18
3.2 Microcontrôleur	19
3.2.1 Connexion	19
3.2.2 Programmateur et reset	19
3.2.3 LED de vie	20
3.3 Périphériques	21
3.3.1 Carte SD	21
3.3.2 Centrale inertielle	21
3.3.3 GNSS	22
3.3.4 USB - FTI	22
3.4 Alimentations	23
3.4.1 Chargeur de batterie	23
3.4.2 Système d'enclenchement	24
3.5 Conclusion et perspectives de l'étude	24

4 Développement du circuit-imprimé	24
4.1 Mécanique du projet	25
4.1.1 Choix du boîtier	25
4.1.2 Placement des composants	25
4.1.3 Assemblage	26
4.2 Bill of materials	27
4.3 Routage	28
4.4 Conclusion et perspectives de la conception	29
5 Développement du firmware	29
5.1 Protocoles du GNSS	29
5.1.1 Messages NMEA	30
5.1.2 Interprétation des données NMEA	30
5.1.3 Code décodeur de trame NMEA	31
5.2 Configuration des périphériques	32
5.2.1 Timers	33
5.2.2 USARTs	35
5.2.3 Carte SD	35
5.3 Application principale	36
5.3.1 Commandes USB-UART	36
5.3.2 État de logging du système	38
5.4 Carte SD	39
5.4.1 Carte SD : initialisation et configuration	39
5.4.2 Carte SD : écriture des données	40
5.5 Centrale inertielle	41
5.5.1 Initialisation	41
5.5.2 Lecture des données	41
5.5.3 Système de détection de mouvements	42
5.5.3.1 Gestion de l'interruption	42
5.5.3.2 Mode low power	42
5.5.3.3 Configuration enclenchement automatique	43
5.6 Format des données	44
5.6.1 Données de l'IMU	44
5.6.2 Données du GNSS	45
5.7 Fonctions des fichiers	46
5.7.1 Centrale inertielle	46
5.7.2 Carte SD	47
5.7.3 Communication série avec FTI	47
6 Développement application interface	48
6.1 Choix de l'environnement	48
7 Validation du design	50
7.1 Liste de matériel	50
7.2 Consommations	50
7.2.1 Méthode de mesure	50
7.2.2 Schéma de mesure	50
7.2.3 Mesures	50
7.2.3.1 Proposition de stratégies d'économie d'énergie	51

7.3 Bus de communications	51
7.3.1 Communication I2C	52
7.3.1.1 Méthode de mesure	52
7.3.1.2 Schéma de mesure	52
7.3.1.3 Mesures	52
7.3.2 Communication UART GNSS	54
7.3.2.1 Méthode de mesure	54
7.3.2.2 Schéma de mesure	54
7.3.2.3 Mesures	54
7.3.3 Communication SPI, carte SD	56
7.3.3.1 Méthode de mesure	56
7.3.3.2 Schéma de mesure	56
7.3.3.3 Mesures	56
8 Caractéristiques du produit fini	56
9 Conclusion	57
10 Bibliographie	58
11 Annexes	59

Acronymes

MCU microcontrôleur.

PCB circuit imprimé.

IMU centrale inertielle.

RF radio-fréquence.

GPS global Positioning System.

GNSS global navigation satellite systems.

Glossaire

Centrale inertielle Instrument utilisé en navigation, capable d'intégrer les mouvements d'un mobile (accélération et vitesse angulaire) pour estimer son orientation (angles de roulis, de tangage et de cap), sa vitesse linéaire et sa position.

timestamp Enregistrement de l'heure et/ou la date d'un événement.

GPS Le système de positionnement global (GPS) est un service public américain qui fournit aux utilisateurs des services de positionnement, de navigation et de synchronisation (PNT). Ce système se compose de trois segments : le segment spatial, le segment de contrôle et le segment utilisateur. L'U.S. Space Force développe, entretient et exploite les segments spatial et de contrôle.

GNSS Le système mondial de navigation par satellite (GNSS) est un terme général décrivant toute constellation de satellites qui fournit des services de positionnement, de navigation et de synchronisation (PNT) à l'échelle mondiale ou régionale.

PIC32 Famille de microcontrôleur 32-bits de Microchip.

FTDI Composant Future Technology Devices International. Sert de convertisseur USB-UART.

harmony Configurateur graphique, générateur de code pour les microcontrôleurs Microchip.

datasheet Document du fabricant fournissant les spécifications d'un produit.



Boîte noire miniaturisée 2023, 1942B

1 Cahier des charges

1.1 Introduction

Ce projet vise à stocker les données de mesures et de localisation d'un avion en utilisant une centrale inertuelle et un GPS/GNSS. En combinant ces technologies, nous pouvons enregistrer des informations précises sur les caractéristiques du vol et la trajectoire de l'avion. En cas d'accident, ces enregistrements permettent de déterminer les causes potentielles. En somme, ce système de collecte et de stockage de données fournit une compréhension approfondie des vols et des données essentielles. Le cahier des charges détaillé est disponible en annexe.

1.2 Aperçu

- Sauvegarde des données inertielles chaque 500ms par défaut.
- Sauvegarde des données de localisation chaque 5'000ms par défaut.
- Possibilité de configurer les temps de sauvegarde.
- Résistance aux chocs.
- Bonne autonomie / Low power.
- [Global Positioning System. \(GPS\)](#)
- [Global navigation satellite systems. \(GNSS\)](#).
- [Timestamp](#) par satellite.
- [Centrale inertuelle](#) :
 - Accéléromètre 3-axes.
 - Gyroscope 3-axes.
- Charge, lecture et config. par USB-C.

1.3 Tâches à réaliser

Développement et intégration d'un PCB avec capteurs et logging sur carte SD dans un boîtier compact.

- Développement schématique
 - Fonctionnement MCU.
 - Périphériques de mesures et de sauvegarde / Bus de communication.
 - Gestion batterie
- Routage pour intégration dans boîtier résistant aux chocs.
- Programmation mesure et sauvegarde des données.
 - Configuration MCU.
 - Configuration du périphérique de mesure pour [centrale inertuelle. \(IMU\)](#).
 - Configuration du périphérique de sauvegarde (Carte SD).
 - Configuration du périphérique de localisation [GPS/GNSS](#).
 - Configuration et communication avec l'interface.
 - Communication et traitement des données mesurées.

1.4 Schéma de principe

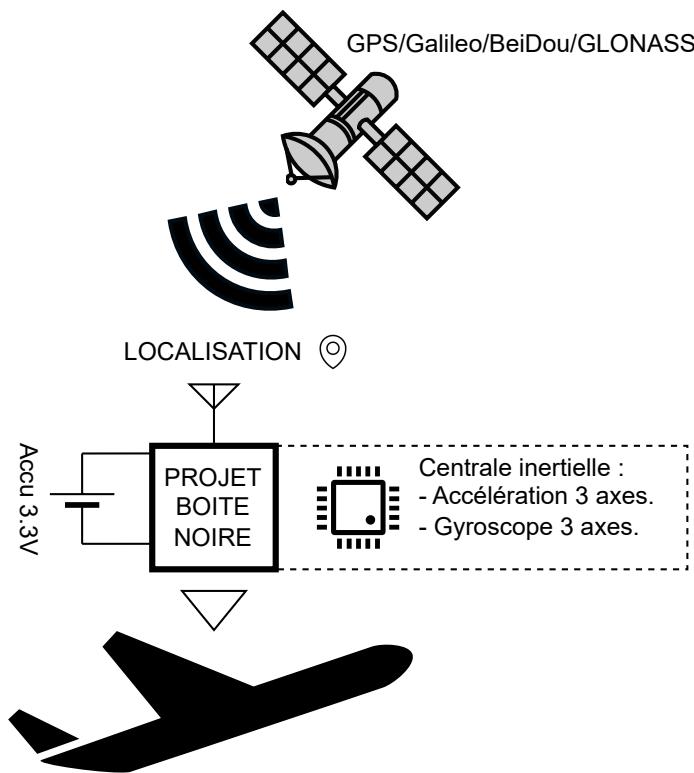


FIGURE 1 – Schéma de principe.

Source: Auteur

Ce système électronique de mini boîte noire pour avion, serait capable d'enregistrer des informations récentes sur les données inertielles et la position d'un vol, dans une mémoire non volatile (carte microSD). Le dispositif, abrité dans un boîtier plastique pour assurer une réception optimale des données **GPS** et une installation compacte. Celui-ci fournirait des données via un port USB pour l'extraction des mesures sur la carte microSD ou pour configurer des paramètres tels que les intervalles de mesure. Les mesures sur les trois axes d'accélération et de vitesse angulaire seraient recueillies par défaut toutes les 500 ms et les données de position **GPS** toutes les 5000 ms. Des intervalles d'enregistrement plus longs sont envisageables pour optimiser la durée de vie de la carte SD, selon la taille et l'organisation des données. Le dispositif sauvegarderait les données des 15 dernières minutes de vol (ou plus) dans un fichier CSV pour traitement ultérieur, selon le principe FIFO. L'objectif principal du prototype est de privilégier la compacité pour minimiser son encombrement à bord de l'avion.

1.5 Planification

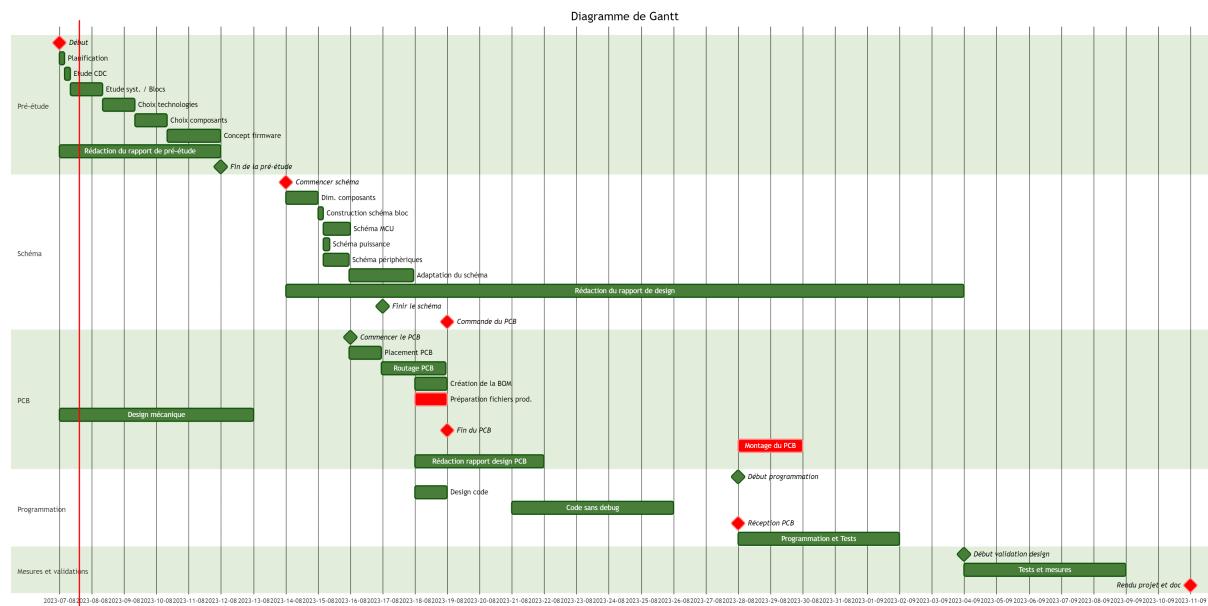


FIGURE 2 – Planification, Diagramme de gantt.

Source: Auteur

No jour de travail	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
Date	lundi 7 août 2023	mardi 8 août 2023	mercredi 9 août 2023	jeudi 10 août 2023	vendredi 11 août 2023	samedi 12 août 2023	dimanche 13 août 2023	lundi 14 août 2023	mardi 15 août 2023	mercredi 16 août 2023	jeudi 17 août 2023	vendredi 18 août 2023	samedi 19 août 2023	dimanche 20 août 2023	lundi 21 août 2023	mardi 22 août 2023	mercredi 23 août 2023	jeudi 24 août 2023	vendredi 25 août 2023	samedi 26 août 2023	dimanche 27 août 2023	lundi 28 août 2023	mardi 29 août 2023	mercredi 30 août 2023	jeudi 31 août 2023	vendredi 1 septembre 2023	samedi 2 septembre 2023	dimanche 3 septembre 2023	lundi 4 septembre 2023	mardi 5 septembre 2023	mercredi 6 septembre 2023	vendredi 8 septembre 2023	samedi 9 septembre 2023	lundi 11 septembre 2023		
Cahier des charges																																				
Pré-étude																																				
Dimensionnement + design + schéma																																				
Design de parties mécaniques																																				
Montage PCB																																				
Design software																																				
Réalisation des softwares																																				
Mise en service et tests																																				
Caractéristiques et mesures																																				
Préparation et présentation																																				
Rédaction rapport																																				
Finalisation/corrections/documentation																																				

FIGURE 3 – Planification théorique.

Source: Auteur

1.6 Livrable

- Les fichiers sources de CAO électronique des PCB réalisés
- Tout le nécessaire à fabriquer un exemplaire hardware de chaque :
- fichiers de fabrication (GERBER) / liste de pièces avec références pour commande / implantation
- Prototype fonctionnel
- Modifications / dessins mécaniques, etc
- Les fichiers sources de programmation microcontrôleur (.c / .h)
- Tout le nécessaire pour programmer les microcontrôleurs (logiciel ou fichier .hex)
- Un calcul / estimation des coûts
- Un rapport contenant les calculs - dimensionnement de composants - structogramme, etc.

2 Pré-étude

2.1 Fonctionnement du système

Le microcontrôleur interagît avec 4 périphériques principaux : Avec le **GNSS**, il partage une communication qui lui permet d'obtenir les informations de localisation par le biais de plusieurs systèmes de satellites. Il y a ensuite, la centrale inertuelle qui lui donne accès de une multitude de mesures sur 9 axes, or, ici les mesures gyroscopiques et d'accélération sont exploitées. La carte SD, permet quant-à-elle, de stocker toutes ces données pour avoir minimum les information des 15 dernières minutes de vol. Le dernier périphérique principal **FTDI**, permet d'avoir une interface avec un ordinateur via connexion USB-C.

2.2 Schéma bloc



FIGURE 4 – Schéma bloc.

Source: Auteur

2.3 Description des blocs

Bloc	Description
GNSS.	Récepteur Radio-fréquence. (RF) avec antenne interne/externe et communication UART.
Microcontrôleur. (MCU).	Microcontrôleur PIC32, intelligence du système, basse consommation.
IMU.	Centrale inertuelle, accélération, gyroscope...
Carte SD	Stockage des données de vol.
FTDI.	Convertit la communication USB en série.
Régulateurs.	Le régulateur de charge gère la charge de l'accu. et un régulateur 3.3V le suit.
Batterie.	La batterie est un accu que l'on peut charger par USB et permet une bonne autonomie.

2.4 Choix des composants et technologies

L'objectif de la pré-étude consiste en grande partie à sélectionner méthodiquement les technologies et les composants du projet. Cette partie du travail est essentielle et critique.

2.4.1 Microcontrôleur

Le microcontrôleur nécessite au minimum les périphériques suivants :

2 UART **1 SPI** **1 I2C**

Il est préférable que le **MCU** dispose de différentes configurations de gestion de puissance, notamment des modes d'économie d'énergie, afin d'avoir une maîtrise de la consommation et de permettre une meilleure autonomie. Enfin, le standard de l'école veut que les familles de microcontrôleurs PIC32 (Microchip) sont préférées.

	Memory Flash/SRAM	Automotive	Connectivity	Functional Safety	Graphics	Motor Control	Security	Ultra-Low Power	Touch
PIC32MZ EF FPU MIPS32® M-Class, 252 MHz	512-2048 KB/ 128-512 KB	●	●	●	●		●		●
PIC32MZ DA MIPS32 microAptiv™, 200 MHz	1024-2048 KB/ 256-640 KB		●	●	●		●		●
PIC32CX SG Arm® Cortex®-M4F, 120 MHz	1024 KB/ 256 KB	●	●	●	●	●	●		●
PIC32MK MIPS32 microAptiv, 120 MHz	256-1024 KB/ 128-256 KB	●	●	●	●	●			●
PIC32MX 3/4 MIPS32 M4K, 80-120 MHz	32-512 KB/ 8-128 KB		●	●					
PIC32MX 5/6/7 MIPS32 M4K, 60 MHz	64-512 KB/ 16-128 KB	●	●						
PIC32MX 1/2 XLP MIPS32 M4K, 72 MHz	128-256 KB/ 32-64 KB		●	●				●	
PIC32MX 1/2/5 MIPS32 M4K, 50 MHz	16-512 KB/ 4-64 KB		●	●					
PIC32CM JH Arm® Cortex®-M0+, 48 MHz	128-512 KB/ 16-64 KB	●	●	●		●	●		●
PIC32CM Lx Arm® Cortex®-M23, 48 MHz	256-512 KB/ 32-16 KB		●			●	●		●
PIC32CM MC Arm® Cortex®-M0+, 48 MHz	64-128 KB/ 8-16 KB			●		●			
PIC32MM MIPS32 microAptiv UC, 25 MHz	16-256 KB/ 4-32 KB	●					●		

FIGURE 5 – Familles PIC32.

Source: [Microchip, familles](#)

Sur la figure 5 le **MCU** sélectionné appartient à la gamme MX (Baseline performance-mémoire) et à la famille XLP qui offre notamment la fonctionnalité "Ultra low power" qui est celle qui nous intéresse.

PIC32MX274F256D

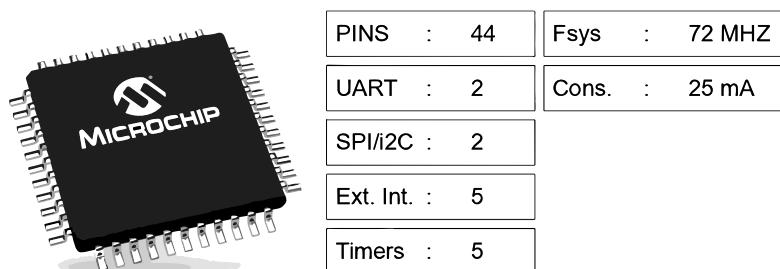


FIGURE 6 – Caractéristiques du PIC32 choisi.

Source: Auteur

2.4.2 Centrale inertielle

Pour la centrale inertielle, il existe un composant avec lequel j'ai déjà acquis une certaine expérience et eu l'occasion d'utiliser et de créer des librairies pour le firmware en C. Celui-ci est performant et très utilisé dans l'industrie. Il est hautement configurable et a le grand avantage de calculer déjà une fusion de capteurs ainsi qu'une compensation de la dérive par les mesures de température. Cela permet notamment d'accéder à des données plus poussées, telles que les quaternions et les angles d'Euler. Il s'agit du **BNO055** de BOSCH.

Caractéristiques importantes :

Résolution gyroscope	:	16	[bits]
Résolution accéléromètre	:	14	[bits]
Résolution magnétomètre	:	~0.3	[μ T]
I_{DD}	:	12.3	[mA]
Dérive de température	:	± 0.03	[%/K]
Dérive accéléromètre	:	0.2	[%/V]
Dérive gyroscope	:	<0.4	[%/V]

Afin de simplifier l'implémentation de ce composant dans le projet, sachant qu'il s'agit d'un boîtier 28-TFLGA difficile à souder ou à mettre au four, il est possible d'utiliser la carte miniature d'Adafruit, cette carte comprend tous les composants passifs requis. Cela facilite ainsi son montage sur le [circuit imprimé. \(PCB\)](#).



FIGURE 7 – Carte d'extension, centrale inertielle.

Source: [Digikey, 4646](#)

Données disponibles :

Température	
Vecteur gravité	XYZ
Orientation compensées, quaternion	WXYZ
Orientation compensée, angle de Euler	HPR
Données gyroscopiques	XYZ
Intensité du champ magnétique	XYZ
Accélération	XYZ

TABLE 1 – Liste des données accessibles.

2.4.3 GPS / GNSS

Pour le **GPS/GNSS**, différents critères entrent en jeu dans le cadre de ce projet : le prix, la facilité d'implémentation, la complexité (par complexité, nous entendons le nombre de fonctionnalités), la consommation et la performance.

Il existe un très grand nombre de récepteurs **RF** pour la navigation. Parmi les plus utilisés dans l'industrie, dont l'implémentation est la plus simple et la documentation la plus complète, il y a plusieurs gammes chez le fabricant **ublox**. Deux composants ont principalement été pris en considération : Le **CAM-M8C-0** (BeiDou, GLONASS, GNSS, GPS, QZSS) avec une antenne omnidirectionnelle interne au composant et différents modes de puissance, ainsi que le **MAX-M10M-00B** (BeiDou, Galileo, GLONASS, GNSS, GPS) sans antenne interne mais avec une consommation de base plus faible.

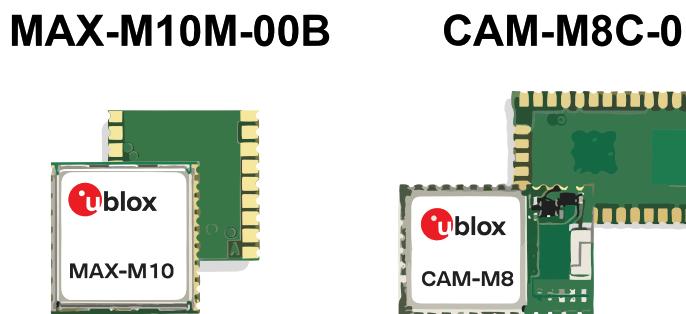


FIGURE 8 – Illustration des deux GNSS.

Source: Digikey, MAX-M10 et CAM-M8C

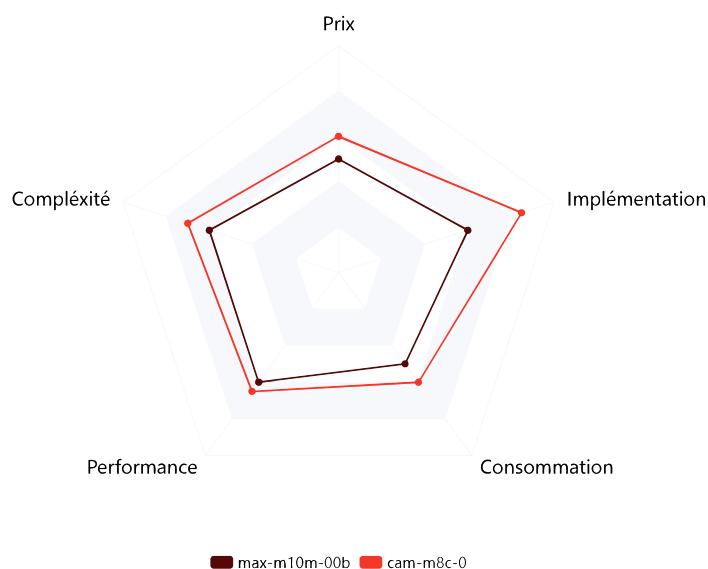


FIGURE 9 – Comparaison GNSS.

Source: Auteur

Malgré les différents avantages que présente le **MAX-M10M-00B** sur la figure 9, le choix s'est porté sur **CAM-M8C-0** grâce à sa facilité d'implémentation et à la garantie du fabricant sur son antenne omnidirectionnelle de qualité. Les détails concernant les protocoles du modules seront décrits lors de la section 5.1.

2.4.4 Carte SD

Les pilotes de carte SD disponibles dans [harmony](#) ne permettent pas une gestion des capacités de stockage trop importantes. Cela signifie, que nous ne pouvons pas avoir des cartes SD de trop grandes capacités, c'est pour cela que nous allons baser nos dimensionnements sur une carte de **256MB**.



FIGURE 10 – Illustration carte SD.

Source: <https://www.oempcworld.com/OEMPCworld-com/017045.html>

2.4.4.1 Estimation de la capacité Admettons les paramètres suivants où S représente une taille de données et T un temps :

S_{SD}	256	[MB]
S_{gyro}	16	[Bytes]
S_{accel}	16	[Bytes]
S_{gnss}	~ 100	[Bytes]
$T_{inertiel}$	0.5	[s]
T_{gnss}	5	[s]
T_{mesMin}	900	[s]

Nous pouvons déduire la taille mémoire que prendra 5 secondes d'enregistrement avec les paramètres par défaut du système :

$$S_{single} = \frac{T_{gnss}}{T_{inertiel}} S_{accel} + S_{gnss} = \frac{5}{0.5} 16 + 100 = 260 \text{ [Bytes]}$$

Nous pouvons enfin calculer à partir de cela, la taille mémoire que prendra 15 minutes d'enregistrement afin de valider le temps de logging minimum :

$$S_{mesures} = \frac{S_{single}}{T_{gnss}} * T_{mesMin} = \frac{260}{5} * 900 = 46'800 \text{ [Bytes]} = 49.8 \text{ [KB]}$$

Nous pouvons déduire avec cette estimation qu'une carte SD de 256MB est largement suffisante et permet de mesurer jusqu'à un temps calculable de cette façon :

$$T_{mesures} = \frac{S_{SD} * T_{gnss}}{S_{single}} = \frac{256 * 10^6 * 5}{260} = \sim 82'051 \text{ Minutes} = \sim 1368 \text{ Heures}$$

Nous estimons donc, que la capacité de stockage de la carte SD est largement suffisante pour le stockage de nombreuses données de vol.

2.5 Batterie, autonomie, charge et régulation

Afin de dimensionner une batterie pour le projet, il faut considérer les différentes consommation :

Liste des consommations principales

Microcontrôleur	24	[mA]	Typ.
Carte-SD	100	[mA]	Max.
Carte-SD	60	[mA]	Moyenne
IMU	12.3	[mA]	Typ.
GNSS	71	[mA]	Max.
GNSS	29	[mA]	Typ.
Totale max	<u>207.3</u>	[mA]	Max.
Totale moyennes	<u>125.3</u>	[mA]	Moyenne

TABLE 2 – Tableau des consommations de courant.

En examinant les consommations typiques et moyennes du tableau 2 à régime constant, et en visant une autonomie de **10 heures**, nous aurions besoin d'une batterie d'une capacité de **1253 mAh**.

2.5.1 Technologies

Concernant la technologie de la batterie, notre objectif, dans un souci d'ergonomie, est de permettre son chargement via un connecteur USB. La technologie offrant actuellement la meilleure densité d'énergie est le **Lithium Ion**¹. Elle présente cependant des inconvénients, énumérés dans le tableau 3.

Avantages	Inconvénient
Haute densité d'énergie.	Risque d'éclatement.
Poids léger.	Risque d'enflammement avec l'eau.
Haute durée de vie.	Sensible a la température.
Charge rapide.	Décharge complète altérante.

TABLE 3 – Tableau avantages/inconvénient LI-ION.

Les inconvénients listés dans le tableau 3 posent un risque pour l'intégrité des données de la carte-SD. C'est pourquoi il est essentiel de bien dimensionner le boîtier ainsi que l'intégration du circuit et de la batterie pour pallier ces dangers. Malgré les risques du Li-Ion, il s'agit d'une technologie très utilisée, y compris dans des domaines sensibles.

2.5.2 Choix de la batterie

La batterie doit avoir une capacité d'au moins **1260 mAh** et être suffisamment compacte pour être intégrée dans un boîtier de petite taille. Une batterie répond à ces critères : il s'agit de la **PICPAL36** de chez Farnell². Les dimensions et caractéristiques de cette batterie sont visibles sur les figures 11.

1. <https://www.epecetc.com/batteries/cell-comparison.html>

2. Distributeur de composants électroniques. [Lien de la batterie](#).

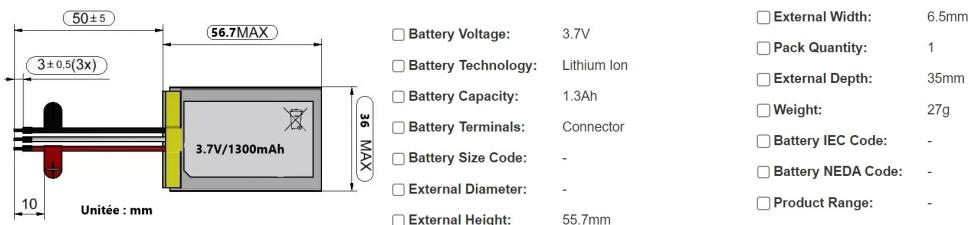


FIGURE 11 – Caractéristiques de la PICPAL36.

Source: Auteur

En raison de problèmes avec les commandes, une batterie alternative a dû être trouvée. Il s'agit de la XCell **B520003** Li-Ion, d'une capacité de **1600mAh**. Ses caractéristiques sont présentées dans la figure 12.

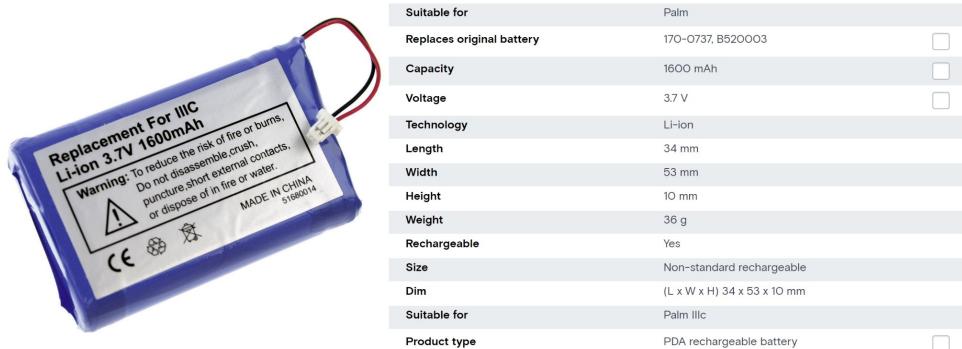


FIGURE 12 – Caractéristiques de la XCell B520003

2.5.3 Régulateur de charge

Le choix du régulateur de charge s'est orienté vers un composant déjà utilisé par l'école, dont les caractéristiques et le montage sont bien connus. C'est un composant performant et fréquemment employé. De plus, il est disponible dans le stock de composants de l'école. Il s'agit du **MCP73871T-2CCI/ML**. Il est configurable et permet la charge des batteries Li-ion et Li-po.

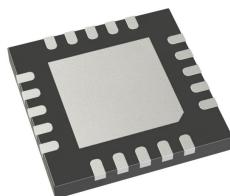


FIGURE 13 – MCP73871T-2CCI/ML.

Source: [Digikey IC BATT CHG LI-ION 1CELL 20QFN](#)

2.6 Systèmes d'économie d'énergies

Au-delà des 10 heures d'autonomie estimées, il est envisageable, pour maximiser le temps de logging, d'adopter des mécanismes d'économie d'énergie. Ces mécanismes pourraient être :

- **Modes de puissance du PIC32** Le microcontrôleur peut basculer entre différents modes de puissance où il réduit à la fois sa consommation et ses fonctionnalités. On pourrait envisager de passer en mode d'énergie restreinte entre les périodes d'enregistrement ou de limiter l'utilisation de certains périphériques.
- **Modes de puissance du GNSS** À l'instar du **MCU**, le composant de navigation dispose de plusieurs modes de fonctionnement qui permettent d'économiser de l'énergie. Un mode particulièrement utile permet de limiter sa cadence de données à 1Hz, étant donné qu'il n'est pas nécessaire qu'il fournisse des informations de localisation aussi fréquemment.
- **Alimentation du GNSS** On pourrait également envisager de ne pas alimenter le **GNSS** lorsqu'il n'est pas en service, afin de supprimer complètement sa consommation entre chaque mesure.
- **Modes de puissance IMU** La centrale inertuelle dispose également de différents modes. Ces modes permettent, par exemple, de limiter les mesures à certains axes ou de la mettre en veille. Elle se met d'ailleurs automatiquement en veille après un certain temps entre les mesures. Ces modes peuvent être utilisés efficacement.
 - **Alimentation de la carte SD** Il est envisageable d'éteindre entièrement la carte SD entre les écritures. Cependant, étant donné que les écritures seront fréquentes, l'intérêt de cette démarche est limité.
- **Mise en veille automatique du système** Si aucun mouvement n'est détecté après un certain délai, il est envisageable que le système s'éteigne entièrement.

2.7 Diagramme des états du système

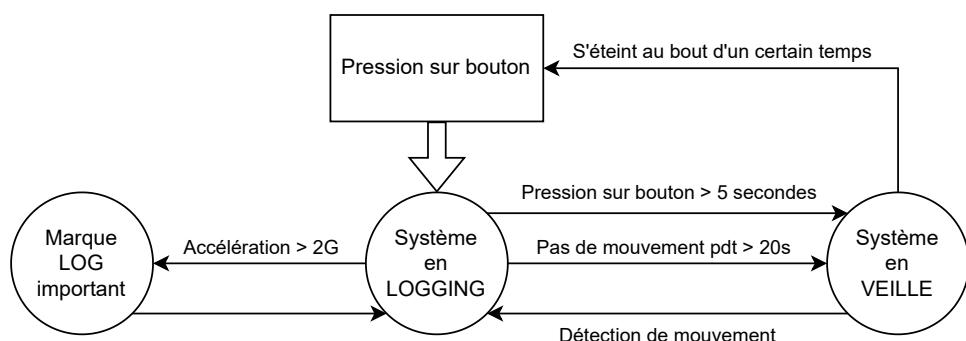


FIGURE 14 – Diagrammes des états (Sans USB).
Source: Auteur

Sur la figure 14 les états de communication par le port USB ne sont pas représentés, il s'agit des modes principaux.

2.8 Estimation du coût du prototype

À partir des composants et des technologies définis lors de la pré-étude, nous pouvons estimer le coût du projet. Cette estimation est présentée dans le tableau 4.

<u>Estimation des coût</u>	
Élément	Prix estimé
PCB	60.-
Batterie	36.-
IMU	26.-
GNSS	24.-
Boîtier	20.-
Composants divers	15.-
MCU	6.-
Régulateurs	6.-
USB & FTDI	3.-
Total	<u>196.-</u>

TABLE 4

Cette estimation est relativement élevée. Il est à noter que le coût du **PCB** pourrait être réduit en cas de commande groupée dans un panel³. Par ailleurs, les prix du boîtier et des composants semblent également être assez élevés.

2.9 Conclusion et perspectives de la Pré-étude

Fonctionnement du système : La conception du système semble possible, le microcontrôleur s'interface avec quatre périphériques essentiels : un **GNSS** pour la localisation, une centrale inertuelle pour des mesures sur 9 axes (priorisant les mesures gyroscopiques et d'accélération), une carte SD pour le stockage (conservant au moins 15 minutes de données de vol), et un périphérique **FTDI** pour l'interface avec un ordinateur via USB-C.

Choix des composants et technologies :

- *Microcontrôleur* : Requiert au moins 2 UART, 1 SPI, et 1 I2C. Le PIC32MX274F256D est privilégié car il offre divers modes d'économie d'énergie.
- *Centrale inertuelle* : Le BNO055 de BOSCH a été choisi, car il offre une variété de mesures avancées et une facilité d'implémentation grâce à la carte Adafruit.
- *GPS/GNSS* : Le CAM-M8C-0 d'ublox a été retenu pour sa facilité d'implémentation et son antenne interne omnidirectionnelle.
- *Carte SD* : Une capacité de 256MB a été choisie à cause des limites du pilote. Une telle carte est amplement suffisante pour enregistrer toutes les données de vol.
- *Batterie, charge, et régulation* : Une capacité de 1253 mAh est nécessaire pour atteindre l'autonomie souhaitée de 10 heures. Une batterie compact de 1600 mAh a été retenue.

Estimation des coûts : L'estimation totale pour le projet s'élève à environ 196 CHF. Néanmoins, il existe des possibilités d'économie, en particulier par une commande groupée du **PCB**.

En conclusion, cette pré-étude a décortiqué le projet de façon méthodique pour sélectionner les composants et technologies appropriés. Elle a également fourni des estimations de coûts et identifié des mécanismes d'économie d'énergie potentiels. La prochaine étape consistera à entamer la phase de conception en utilisant ces informations utiles.

3. Regroupement de plusieurs circuits imprimés en un seul, pour diminuer le coût.

3 Développement du schéma électronique

Dans cette section, nous décrirons la phase principale du développement ainsi que la démarche suivie pour élaborer le schéma électronique du projet.

3.1 Blocs développés

Pour faciliter le développement et la lecture du schéma, il est judicieux de diviser le système en plusieurs blocs. Une structure a ainsi été définie, divisant le circuit en trois blocs principaux : **Microcontrôleur 3.2** (Intelligence du système, connexion du programmeur et LED de vie.), **Périphériques 3.3** (GNSS, IMU, FTDI, connecteur USB, Carte SD.) et **Alimentations 3.4** (Connecteur batterie, gestion de charge, régulateurs de tension et système ON/OFF.).

Nous pouvons sur la figure 15 observer les différentes interactions entre les blocs, elles sont par la suite décrites dans le tableau 5.

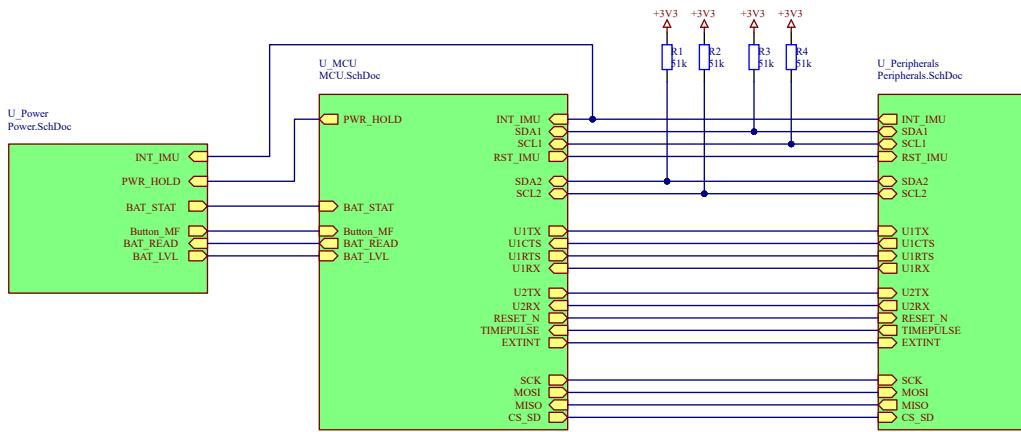


FIGURE 15 – Blocs du système.

Source: Auteur

Connexion/s	Description
INT_IMU	Interruption de la centrale inertuelle, informe le MCU et peut allumer le système.
RST_IMU	Permet de réinitialiser la IMU .
PWR_HOLD	Le MCU peut se maintenir alimenté par cette connexion.
BAT_STAT	Fournit le statut de la batterie au MCU .
Button_MF	Fournit le niveau logique du bouton au MCU .
BAT_READ	Le MCU peut activer la lecture de la tension de la batterie.
BAT_LVL	Lecture analogique de la tension de batterie.
SDA1, SCL1	Communication I2C avec l' IMU .
SDA2, SCL2	Communication I2C optionnelle avec le GNSS .
U1TX/RX...	Communication UART avec le FTDI pour l'USB.
U2TX/RX	Communication UART avec le GNSS .
RESET_N	Permet de réinitialiser le GNSS .
TIMEPULSE	Permet de mesurer le temps par un signal pulsé.
EXTINT	Permet de gérer le mode d'alimentation du GNSS .
SCK, MOSI...	Communication SPI avec la carte SD.
R1,2,3,4	Résistances de PULL-UP pour communication I2C.

TABLE 5 – Description des connexions

3.2 Microcontrôleur

3.2.1 Connexion

Pour utiliser le microcontrôleur, il est nécessaire de définir ses entrées/sorties en se référant à son [datasheet](#). Ce dernier permet de connaître les connexions dédiées à certains bus de communication ou à des entrées analogiques.

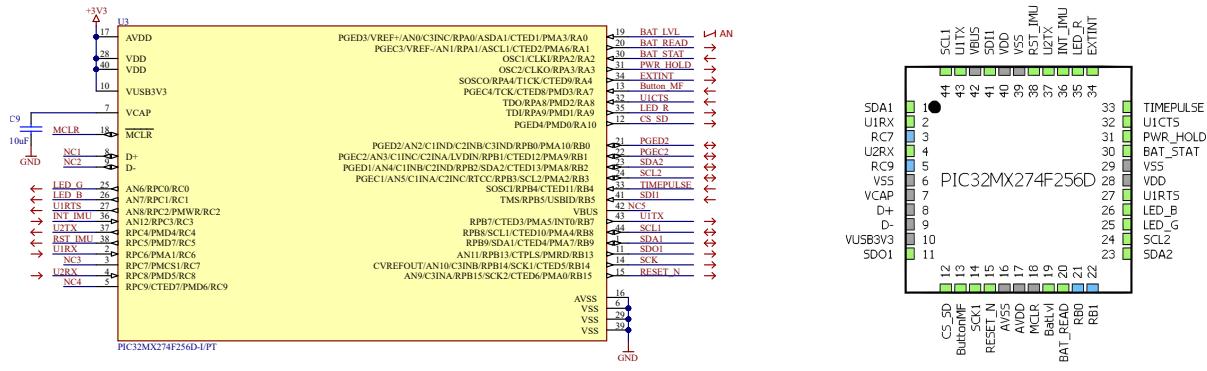


FIGURE 16 – Configuration des PINs du microcontrôleur.

Source: Auteur

Pour valider les connexions de la figure 16a, une vérification a été réalisée à l'aide du configurateur graphique [harmony](#), comme le montre la figure 16b. Cette étape a confirmé la possibilité d'attribuer certaines fonctions à des PINs spécifiques. Les PINs non utilisés du microcontrôleur sont redirigées vers un connecteur. Dans le contexte du prototype, cela permet de les exploiter ailleurs sur la carte.

3.2.2 Programmateur et reset

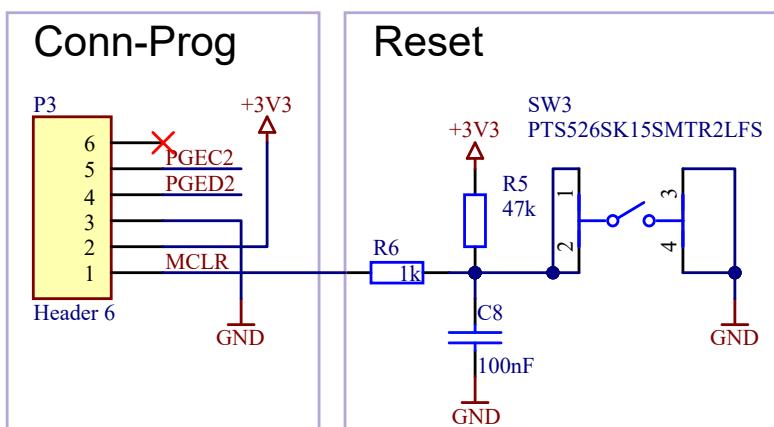


FIGURE 17 – Schéma programmateur et reset.

Source: Auteur

Sur la figure 17, nous pouvons observer le connecteur de programmation *P3*. La connexion **MCLR** (Master Clear), qui permet de réinitialiser le **MCU** lors de sa programmation, est suivie d'un bouton pour permettre une réinitialisation manuelle.

3.2.3 LED de vie

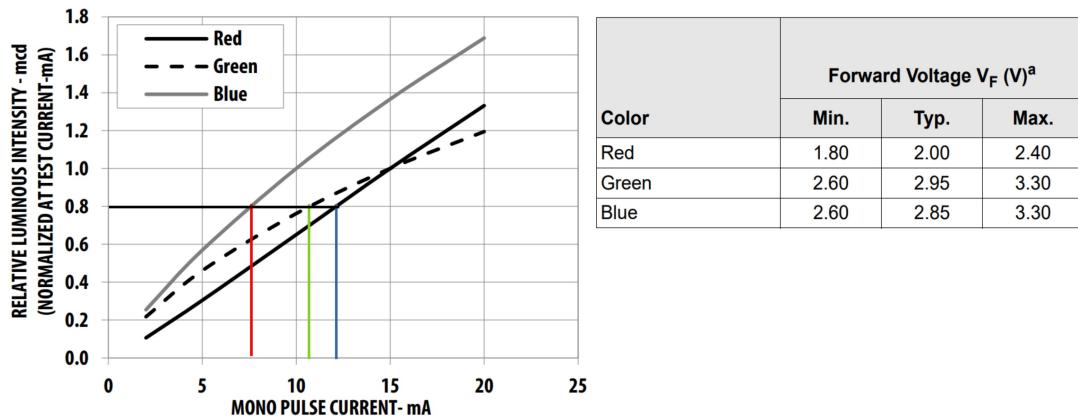


FIGURE 18 – Données de la LED
Source: [datasheet ASMB-KTF0-0A306](#)

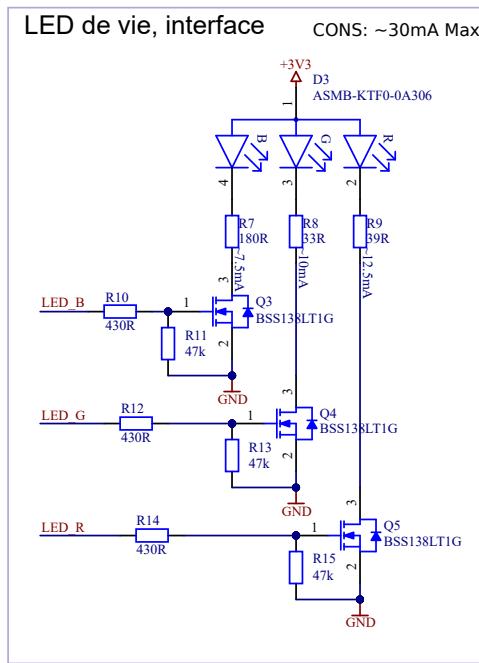


FIGURE 19 – Schéma de la LED de vie.
Source: Auteur

Les résistances de la figure 19 ont été dimensionnées en respectant les caractéristiques des LEDs pour chacune des couleurs (voir figure 18). Ces dernières éclairent à 80% de leur luminosité nominale dans un souci d'économie d'énergie, leur utilité étant réduite à ce stade de prototypage.

Exemple dimensionnement résistance LED bleue

$$R_{blue} = \frac{(V_{cc} - V_{blue})}{I_b} = \frac{(3.3 - 2.85)}{12 * 10^{-3}} = 37.5\Omega$$

3.3 Périphériques

Dans cette section, nous allons décrire les schématiques des périphériques du système.

3.3.1 Carte SD

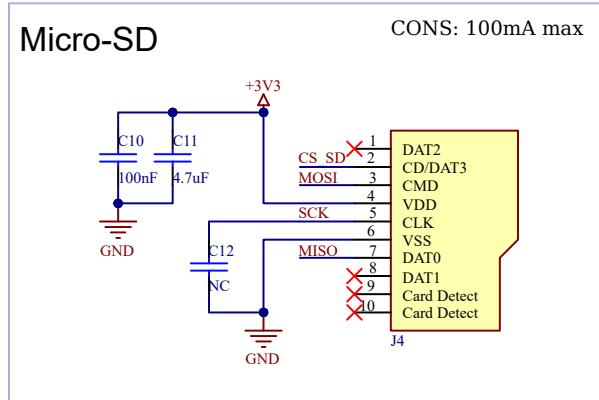


FIGURE 20 – Branchement carte SD.

Source: Auteur

La carte SD de la figure 20 est branchée en respectant la connectique pour une communication SPI simple. De plus, un condensateur est connecté entre **SCK** et **GND**. Toutefois, ce condensateur ne sera pas monté sauf si un filtrage sur la ligne de clock s'avère nécessaire.

3.3.2 Centrale inertie

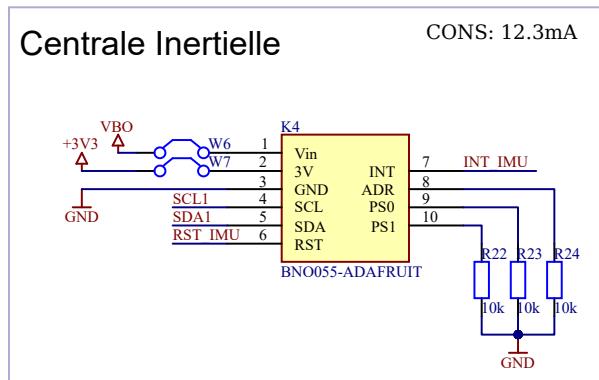


FIGURE 21 – Schéma centrale inertie.

Source: Auteur

La centrale inertie, étant déjà montée sur une carte d'extension, ne nécessite pas de composants passifs additionnels. Trois résistances de PULL-DOWN sont présentes, respectivement sur **ADR**, **PS0** et **PS1**. Elles permettent de configurer l'adresse et l'interface de communication de l'**IMU**. De plus, un jumper⁴ est présent sur les alimentations. Ceci permet de choisir si la centrale inertie est alimentée indépendamment du reste du système (pour pouvoir l'allumer séparément) ou avec la même alimentation que le reste du système.

4. Connecteur permettant de faire ou de rompre une connexion.

3.3.3 GNSS

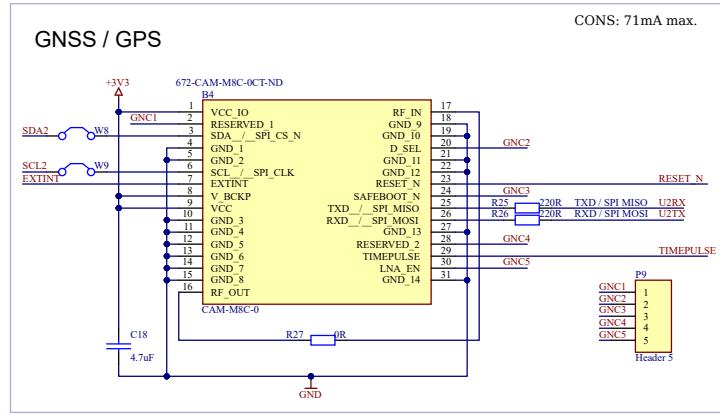


FIGURE 22 – Schéma du GNSS.

Source: Auteur

Le **GNSS** est connecté selon un schéma présenté dans son [Hardware Integration Manual](#), à la section 2.2. Ce schéma représente un design minimal pour recevoir des données via l'interface UART. Par rapport à la figure 22, les différences sont les suivantes : l'interface I_C est également disponible, la PIN **RESET_N** est connectée au **MCU** et les PINS non-utilisées sont reliées à un connecteur afin de permettre leur utilisation éventuelle dans le cadre du prototype.

3.3.4 USB - FTDI

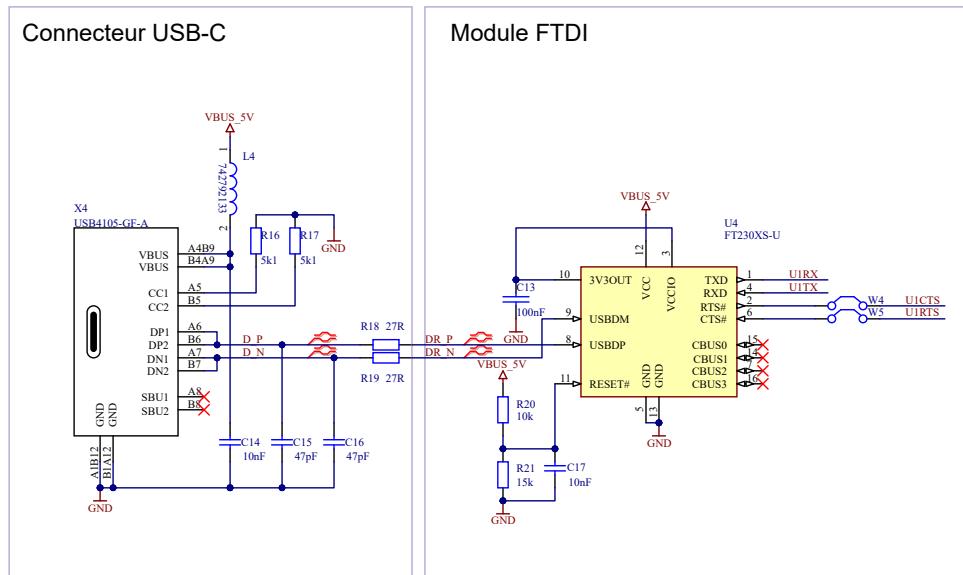


FIGURE 23 – Schéma connecteur USB et FTDI.

Source: Auteur

Le schéma du connecteur USB-C est conçu pour présenter une résistance aux interférences. Son signal 5V est filtré par une *ferrite bead*. Les pistes différentielles de l'USB sont ensuite reliées au module **FTDI**, qui est interfacé en UART et est alimenté par l'USB.

3.4 Alimentations

Dans cette section, nous allons décrire les fonctionnement et dimensionnements principaux du bloc alimentation comprenant les différents régulateurs et la batterie.

3.4.1 Chargeur de batterie

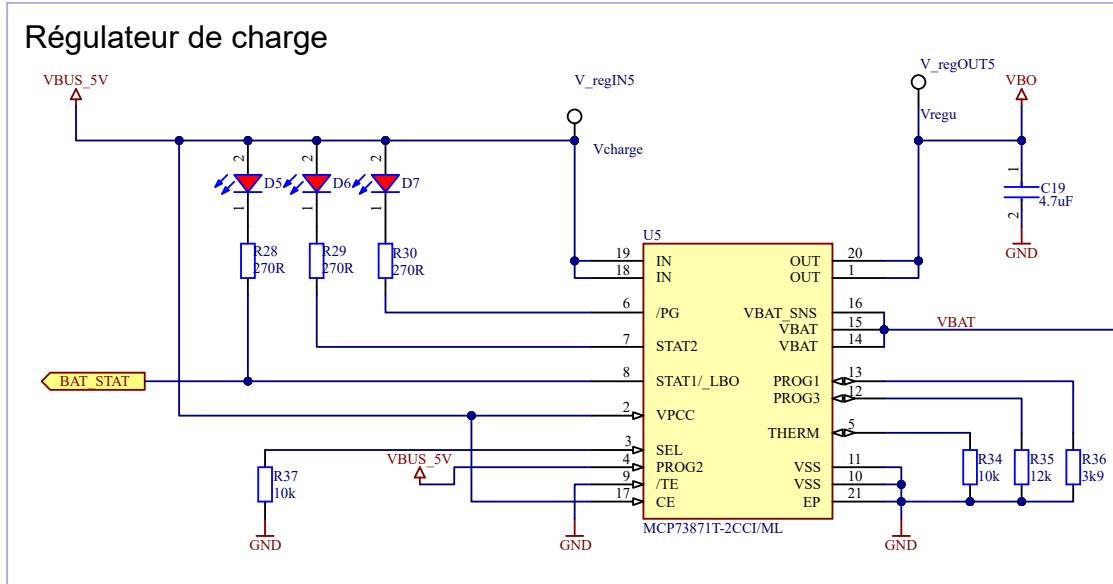


FIGURE 24 – Schéma chargeur de batterie.

Source: Auteur

Le composant **MCP73871T-2CCI/ML** est configurable pour délivrer un courant de charge programmable à la batterie. Ce courant se sépare en deux phases : la charge normale et la charge de terminaison. Cette dernière est activée lorsque la batterie approche de sa capacité maximale. Pour déterminer ces courants, il convient de se baser sur la capacité de la batterie et de certains ratios standards associés.

Où :

$C = 1600mAh$ est la capacité de la batterie.

$k_{term} = 0.05$ est le ratio de fin de charge.

$k_{chg} = 0.15$ est le ratio de charge.

Le courant de terminaison peut être calculé ainsi :

$$I_{term} = C * k_{term} = 1600 * 0.05 = 80 mA$$

Par conséquent, d'après la [datasheet](#) ([Equation 4-2](#)) la résistance de programmation de fin de charge vaudrait :

$$R_{prog3} = \frac{1000V}{I_{term}} = \frac{1000}{65 * 10^{-3}} = 12.5 k\Omega \Rightarrow 12 k\Omega$$

Nous pouvons de la même façon cacluler la résistance de charge **Rprog1** :

$$R_{prog1} = \frac{1000V}{C * k_{chg}} = \frac{1000}{1600 * 10^{-3} * 0.15} = 4.166 k\Omega \Rightarrow 3.9 k\Omega$$

3.4.2 Système d'enclenchement

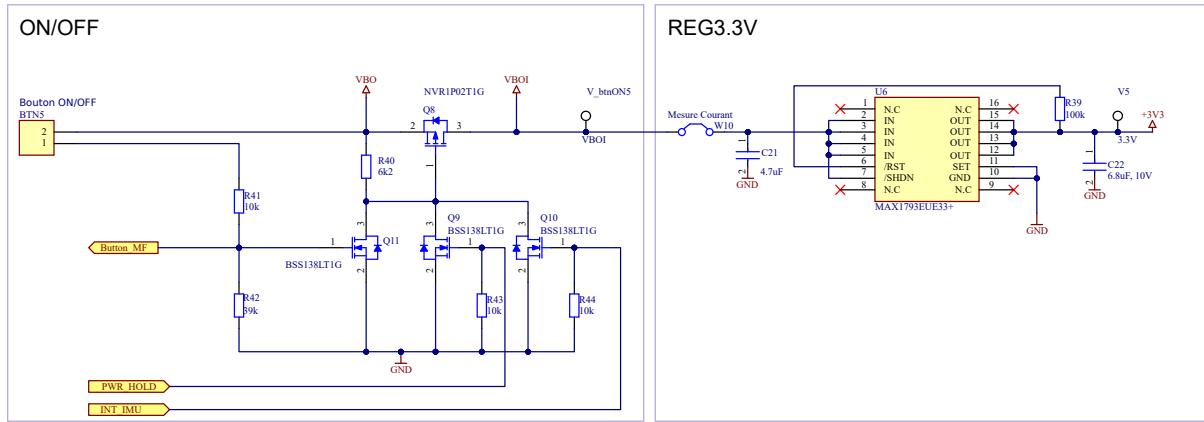


FIGURE 25 – Schéma allumage du système.

Source: Auteur

Sur la figure 25, nous observons les différentes manières de connecter la tension régulée de la batterie au régulateur linéaire 3.3V. Quand cette connexion n'est pas réalisée, le système n'est pas alimenté, à l'exception de l'**IMU** qui peut être alimenté de manière autonome. Pour mettre en marche le système, trois options sont possibles :

- Le microcontrôleur peut maintenir le système alimenté avec **PWR_HOLD**.
- L'**IMU** peut 'allumer le système avec son interruption **INT_IMU**.
- Un bouton peut être pressé pour enclencher la carte **Button_MF**.

3.5 Conclusion et perspectives de l'étude

Tout au long de cette étude, nous avons pu décomposer et analyser le processus de développement du schéma électronique. La démarche a priorisé la modularité, avec une division du système en trois blocs : **Microcontrôleur 3.2**, **Périphériques 3.3** et **Alimentations 3.4**. Cette approche a permis une meilleure lisibilité et une facilité d'adaptation quant au prototypage. Les différentes interactions et connexions entre les composants ont été détaillées.

Par la suite, le circuit imprimé sera développé, les composants placés et routés le tout dans un design petit et compact. La modularité du projet, permettra des tests empiriques sur les différents blocs du système, lors de la mise en service.

4 Développement du circuit-imprimé

Dans cette section, nous nous attarderons sur le processus de conception et de développement du **PCB**. Nous aborderons également les différents aspects mécanique d'intégration du projet, en essayant d'analyser les différentes contraintes des composants et la structure globale. Cette analyse combinée des aspects électroniques et mécaniques est cruciale pour assurer la réussite du projet.

4.1 Mécanique du projet

L'un des objectifs majeurs du projet est de concevoir un produit qui, par sa petite taille et son design compact, se prête à une installation qui se veut à la fois discrète et non encombrante.

4.1.1 Choix du boitier

Pour réduire la taille au maximum, un processus itératif a eu lieu, consistant à trouver un boîtier plastique le plus petit possible et d'essayer en respectant les contraintes de taille, de placer les composants, puis, d'observer et déduire si la taille est suffisante. Grâce à cette procédure itérative, un boîtier a été déterminé : le **SIC5-9-3B** de chez **TAKACHI**.

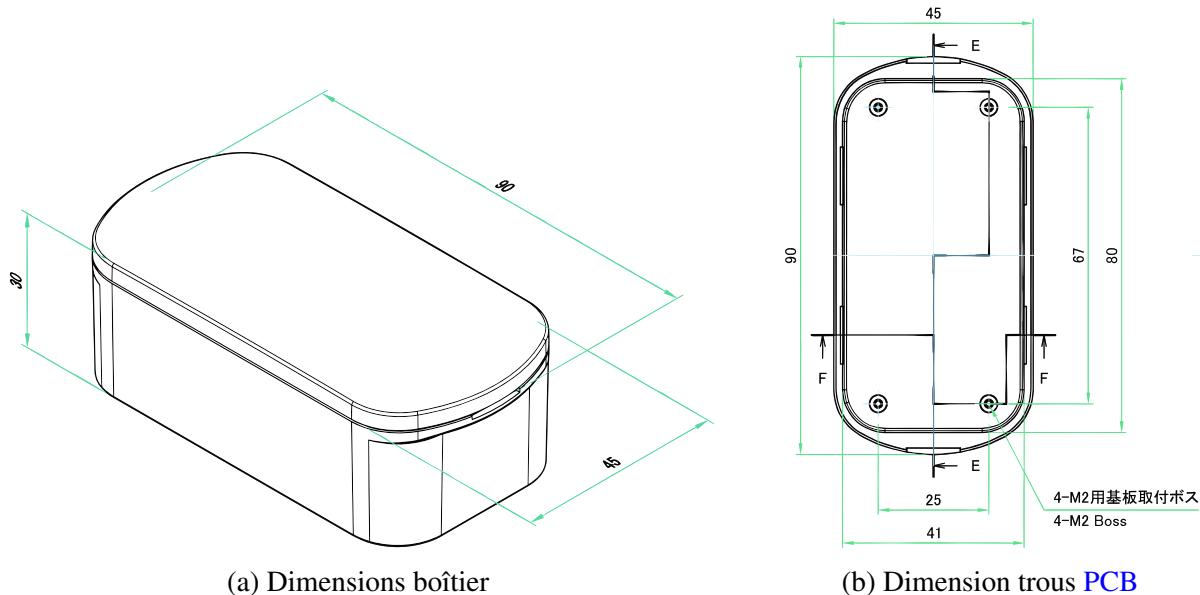


FIGURE 26 – Dimensions du SIC5-9-3B.

Source: [Datasheet du SIC5-9-3B, TAKACHI](#)

4.1.2 Placement des composants

Désormais, sachant que nous connaissons les dimensions du **PCB** grâce à la figure 26b, nous pouvons positionner les composants de manière optimisée pour le routage et la mécanique. En prenant différents éléments en compte :

Élément	Contraintes, règles et dispositions particulières
> Connecteur USB	: Placer proche du bord, usinage du boîtier pour passer chargeur.
> Connecteur µSD	: Proche du bord, facile d'accès.
> GNSS	: Isolé des autres composants, faciliter réception.
> Carte BNO055	: Espace requis, permettre de le braser sur le PCB .
> LED de vie	: Dégagée, peut être déportée avec un guide-lumière.
> Connecteur batterie	: Accessible et permettre de passer les fils de l'autre côté du PCB .
> Connecteur bouton ON/OFF	: Bouton déporté sur le boîtier.
> Bouton reset	: Accessible pendant le développement et en cas de bug.

TABLE 6 – Tableau des contraintes de placement.

Source: Auteur

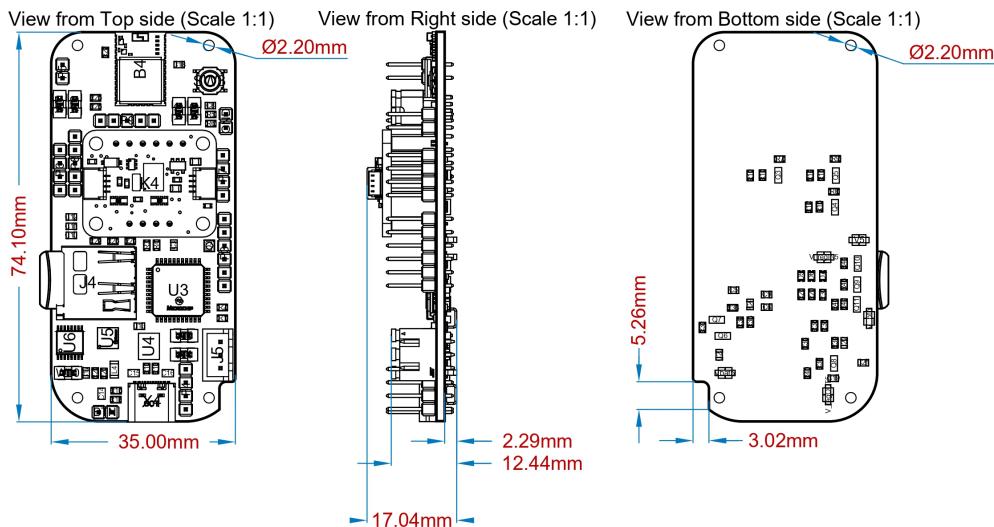


FIGURE 27 – Placement des composants et dimensions de la carte.

Source: Auteur

En considérant les contraintes des différents éléments de la table 6, nous obtenons un design tel que celui de la figure 27. Nous y observons plusieurs solutions apportées telles que : un trou (En bas à droite du **PCB**) permettant de passer les fils de la batterie, une carte SD et un connecteur USB placés proche du bord, un bouton de reset accessible et un **GNSS** isolé du reste pour garantir l'intégrité de l'antenne.

4.1.3 Assemblage

Lors de cette section, nous allons assembler le circuit imprimé avec ses composants et la batterie, dans le modèle 3D du boîtier **SIC5-9-3B**.

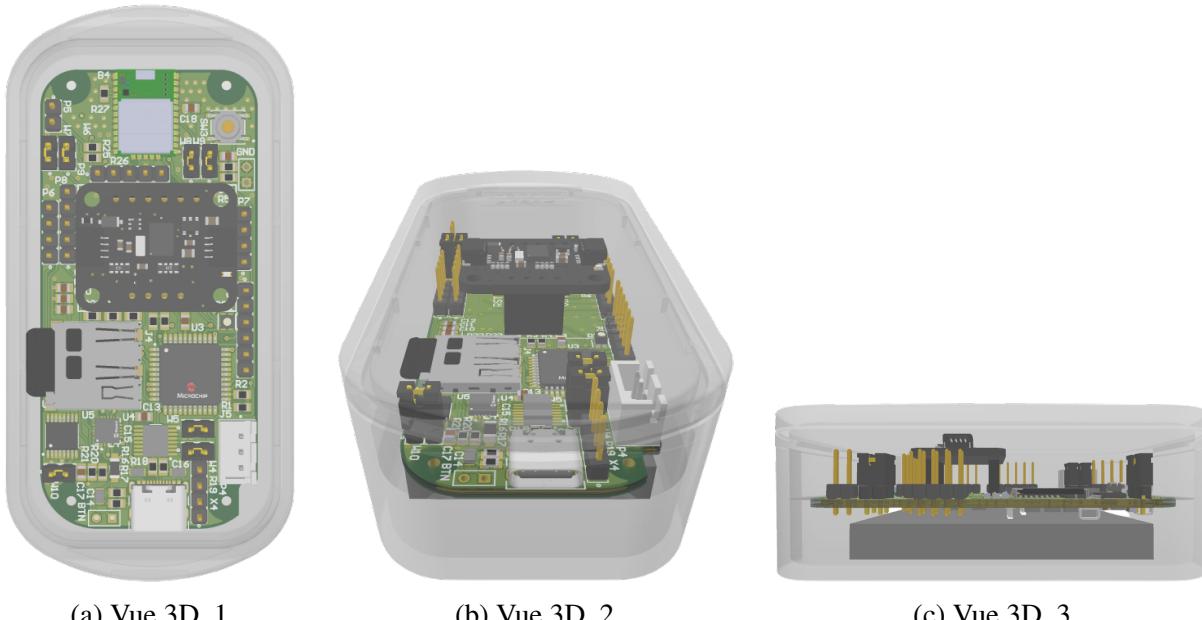


FIGURE 28 – Vues 3d de l'assemblage.

Source: Auteur

Comme nous pouvons observer sur la figure 28, l'ensemble, s'assemble correctement.

4.2 Bill of materials

Avec le schéma électronique et la mécanique désormais finalisés, nous pouvons dresser une liste des composants nécessaires. La table 7 répertorie les composants **hors-stock** qui ont été commandés auprès de différents fournisseurs. La table 8, quant à elle, liste les composants **en stock** disponibles dans les locaux de l'école supérieure, avec leurs coûts respectifs. Il est à noter que ces derniers n'ont pas été commandés. Les tableaux 9 et 10 détaillent respectivement les valeurs des résistances et des condensateurs utilisés pour le projet.

Composant	Fournisseur	Référence	Quantité	Prix [CHF]
Batterie PICPAL36	Farnell	PICPAL36	1	36,54
BNO055-Adafruit	Digikey	1528-4646-ND	1	26,06
GNSS	Digikey	672-CAM-M8C-0CT-ND	1	23,49
Boitier boîte noire	Farnell	CHH9840BK	1	6,84
Guide lumière	Digikey	LFB075CTP-ND 492-1980-ND	4	6,13
PIC32MX274F256D	Digikey	PIC32MX274F256DT-I/PTCT-ND	1	6,07
Bouton poussoir ext.	Digikey	EG5949-ND	1	4,54
Connecteur uSD	Digikey	732-3819-1-ND	1	2,71
LEDS	Digikey	516-3906-1-ND 732-4985-1-ND	4	2,05
Connecteur USB	Digikey	2073-USB4105-GF-ACT-ND	1	1,97
Mosfet P	Digikey	NVR1P02T1GOSCT-ND	2	0,7
Bouton tactile reset	Digikey	CKN12221-1-ND	1	0,35
Ferrite Bead 600 Ohm	Digikey	732-4655-1-ND	1	0,22
Connecteur batterie	Farnell	B3B-XH-A (LF)(SN)	1	0,11
TOTAL		117.78		

TABLE 7 – Liste des composants hors-stocks.

Composant	Fournisseur	Référence	Quantité	Prix [CHF]
Régulateur linéaire 3.3V	Digikey	MAX1793EUE33+-ND	1	3.85
IC chargeur de batterie lithium	Digikey	MCP73871T-2CCI/MLCT-ND	1	2.1
Mosfet Canal N	Digikey	BSS138LCT-ND	7	2.1
FTDI, USB-UART	Digikey	768-1154-5-ND	1	1.97
TOTAL		10.02		

TABLE 8 – Liste des composants en stock.

Résistances														
Valeur	27R	220R	270R	430R	1k	5k1	6k2	8k2	10k	15k	39k	47k	51k	100k
Quantité	2	3	4	6	1	2	1	1	12	2	4	1	4	1

TABLE 9 – Liste des valeurs des résistances.

Condensateurs						
Valeur	47pF	10nF	100nF	4.7uF	6.8uF	10uF
Quantité	2	2	7	4	1	2

TABLE 10 – Liste des valeurs des condensateurs.

4.3 Routage

Une fois les composants placés, les pistes et les plan ont pu être

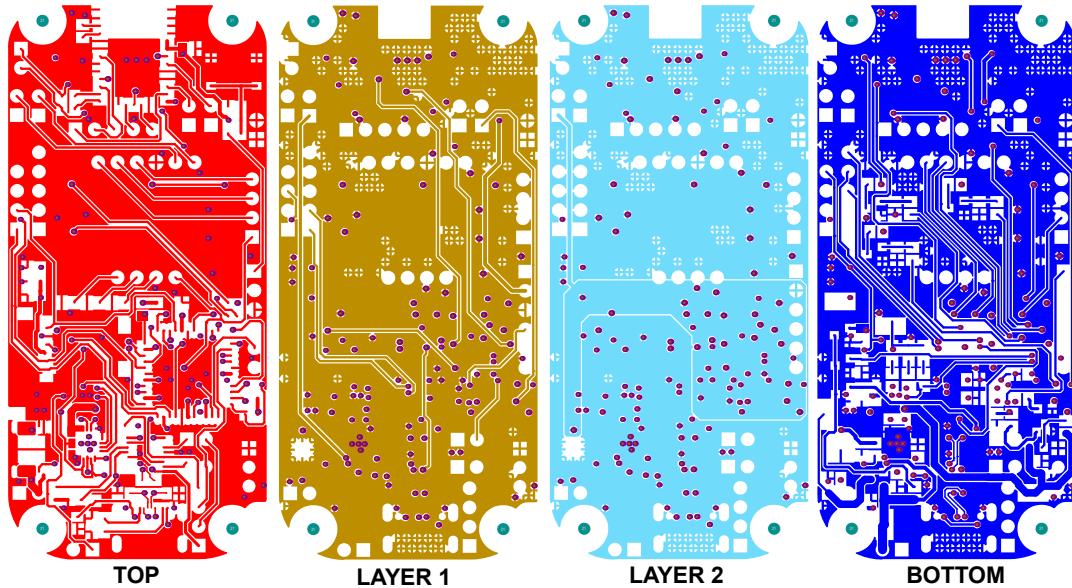


FIGURE 29 – Routage des différentes couches.

Source: Auteur

Les justifications du routage des différentes couche sont présentées dans la table 11.

TOP	<ul style="list-style-type: none"> ➢ Composants à braser au four. ➢ Signaux proches du MCU. ➢ Signaux sortant du GNSS. ➢ Pistes horizontales. ➢ Plan de masse. ➢ Connecteurs pour boutons et batterie.
LAYER 1	<ul style="list-style-type: none"> ➢ Pistes de signaux verticaux. ➢ Plan de masse.
LAYER 2	<ul style="list-style-type: none"> ➢ Plan de masse homogène. ➢ Plan de 3.3V.
BOTTOM	<ul style="list-style-type: none"> ➢ Composants des régulateurs. ➢ Mosfets et composants passifs. ➢ Pistes de signaux verticaux. ➢ Plan de masse.

TABLE 11 – Description des éléments du routage.

Le routage étant complété et contrôlé, il a pu être commandé sous forme de panel avec un pochoir chez [Eurocircuit](#). Celui-ci a eu des retards de production et de livraison par UPS.

4.4 Conclusion et perspectives de la conception

Durant cette section, nous avons conçu le circuit imprimé de la boîte noire, défini son implémentation dans un petit boîtier plastique, vérifié l'intégration du circuit et de la batterie à l'intérieur, et avons décrit les différentes justifications de développement. Par la suite, la carte sera montée, brasée, et les phases de programmation sur la carte auront lieu.

5 Développement du firmware

Lors de cette section, nous décrirons le processus de développement du firmware du PIC32. Les décisions prises seront expliquées et les différents algorithmes seront illustrés et décrits.

En premier lieu nous allons analyser lors de la section 5.1 comment traiter les données du **GNSS**, il s'agit d'un élément critique.

5.1 Protocoles du GNSS

Il existe différents protocoles pour le format de données de localisation. Le **CAM-M8C-0** supporte plusieurs protocoles, visibles sur la figure 30 :

1.15 Protocols and interfaces

Protocol	Type
NMEA	Input/output, ASCII, O183, version 4.0 (Configurable to V 2.1, V 2.3 or V4.1)
UBX	Input/output, binary, u-blox proprietary
RTCM	Input, message 1, 2, 3, 9

FIGURE 30 – Protocoles disponibles.

Source: [Datasheet du CAM-M8C-0](#)

NMEA : Norme établie par la *National Marine Electronics Association*. Elle suit un format **ASCII**.

UBX : Format propriétaire de u-blox avec des données **binaires**. Il permet d'envoyer des trames de configuration.

RTCM : Protocole pour des données GPS différentielles. Établi par la *Radio Technical Commission for Maritime Service*.

Le **CAM-M8C-0** est configuré, par défaut, pour envoyer des messages au format NMEA, comme illustré sur la figure 31.

8 Default messages

Interface	Settings
UART Output	9600 Baud, 8 bits, no parity bit, 1 stop bit Configured to transmit both NMEA and UBX protocols, but no UBX messages and only the following NMEA have been activated at start-up: GGA, GLL, GSA, GSV, RMC, VTG, TXT
UART Input	9600 Baud, 8 bits, no parity bit, 1 stop bit, Autobauding disabled Automatically accepts following protocols without need of explicit configuration: UBX, NMEA, RTCM The GPS receiver supports interleaved UBX and NMEA messages.
DDC	Fully compatible with the I ² C industry standard, available for communication with an external host CPU or u-blox cellular modules, operated in slave mode only. NMEA and UBX are enabled as input messages, only NMEA as output messages Maximum bit rate 400 kb/s.
TIMEPULSE (1 Hz Nav)	1 pulse per second, synchronized at rising edge, pulse length 100 ms

FIGURE 31 – Protocole par défaut.

Source: [Datasheet du CAM-M8C-0](#)

5.1.1 Messages NMEA

Comme nous avons pu constater sur la figure 31, il y a plusieurs messages **NMEA** :

GGA, GLL, GSA, GSV, RMC, VTG, TXT

Ceux-ci sont décrits dans le [manuel de référence du protocole NMEA](#) sur la figure 32

1. Output Messages.....	1-1
GGA —Global Positioning System Fixed Data.....	1-2
GLL—Geographic Position - Latitude/Longitude.....	1-3
GSA—GNSS DOP and Active Satellites.....	1-3
GSV—GNSS Satellites in View	1-4
MSS—MSK Receiver Signal.....	1-4
RMC—Recommended Minimum Specific GNSS Data.....	1-5
VTG—Course Over Ground and Ground Speed	1-5
ZDA—SiRF Timing Message	1-6
150—OkToSend	1-6

FIGURE 32 – Messages NMEA.

Source: [Manuel du protocole NMEA](#)

Les différents messages de la figure 32 présentent des différentes données sous divers formats. Les messages peuvent être activés ou désactivés en configurant le module u-blox.

5.1.2 Interprétation des données NMEA

Avant d'avoir le **PCB** monté et exploitable, sachant que nous connaissons le protocole par défaut du module, nous pouvons analyser des données **NMEA** pour mieux les comprendre et les traiter par la suite. Pour ce faire, le site <https://www.nmeagen.org/> permet de générer des coordonnées avec le protocole **NMEA**.



FIGURE 33 – Application d'une localisation NMEA.

Source: nmeagen.org, aéroport de La Blécherette, Lausanne

Sur la figure 34, les messages de la figure 33 sont décodés via un [analyseur NMEA en ligne](#).

Date	Time UTC	Latitude	Longitude	Altitude	Fix	Fix quality	PDOP	HDOP	VDOP	Inview Sats	Active sats	Active GPS
N/A	08:06:14	46.54516667	6.6169	0.0	N/A	GPS	N/A	1.0	N/A	N/A	N/A	N/A
N/A	08:06:14	46.54516667	6.6169	0.0	F	GPS	1.0	1.0	1.0	N/A	12	12
21/08/2023	08:06:14	46.54516667	6.6169	0.0	F	GPS	1.0	1.0	1.0	N/A	12	12

FIGURE 34 – Messages NMEA décodés.

Source: [NMEA online analyser](https://nmeagen.org/)

5.1.3 Code décodeur de trame NMEA

Afin de développer le code du décodeur **NMEA**, la librairie **minmea** de licence libre, a pu être utilisée, modifiée et adaptée. Un algorithme a ensuite été mis en place :

- 1 Lecture du type de message (**GBS**, **GGA**, **GLL**, **GSA**, **GST**, **GSV**, **RMC**, **VTG**, **ZDA**) ;
- 2 Parsing des valeurs correspondantes à l'ID du message ;
- 3 Sauvegarde des valeurs dans une structure.

Code Source 1 – gnss_posGet_nmea(...), décodage **NMEA**.

```
// Get message ID, strict
*msg_id = minmea_sentence_id(message, true);
// Parse message depending on ID
if (*msg_id == MINMEA_SENTENCE_GBS)
    minmea_parse_gbs(&sentences->gbs, message);
else if (*msg_id == MINMEA_SENTENCE_GGA)
    minmea_parse_gga(&sentences->gga, message);
else if ...
```

Dans le code 1, on fait appel à une fonction qui va masquer les caractères du type du message. Ensuite, selon le message dont il s'agit, on le décode de différentes façons.

Code Source 2 – minmea_parse_gbs(...), décodage message **GBS**.

```
bool minmea_parse_gbs(struct minmea_sentence_gbs *frame, const char
    ↵ *sentence)
{
    // $GNGBS,170556.00,3.0,2.9,8.3,,,,*5C
    char type[6];
    if (!minmea_scan(sentence, "tTffff", ,
                      type,
                      &frame->time,
                      &frame->err_latitude,
                      &frame->err_longitude,
                      &frame->err_altitude,
                      &frame->svid,
                      &frame->prob,
                      &frame->bias,
                      &frame->stddev
                      ))
        return false;
    if (strcmp(type+2, "GBS"))
        return false;
    return true;
}
```

Dans le code 2, on masque les éléments du message selon son format, d'une manière similaire à la fonction standard **scanf(...)**.

Les données spécifiques au message **GBS** sont ensuite stockées dans une structure, que l'on peut observer dans le code 3.

Code Source 3 – Structure **GBS**.

```
struct minmea_sentence_gbs {
    struct minmea_time time;
    struct minmea_float err_latitude;
    struct minmea_float err_longitude;
    struct minmea_float err_altitude;
    int svid;
    struct minmea_float prob;
    struct minmea_float bias;
    struct minmea_float stddev;
};
```

Sachant que plusieurs types de messages peuvent être interceptés, chacune des structures des différents formats de messages est elle-même stockée dans une structure que l'on peut observer dans le code 4.

Code Source 4 – Structures messages.

```
typedef struct minmea_messages{
    struct minmea_sentence_gbs gbs;
    struct minmea_sentence_rmc rmc;
    struct minmea_sentence_gga gga;
    struct minmea_sentence_gll gll;
    struct minmea_sentence_gst gst;
    struct minmea_sentence_gsa gsa;
    struct minmea_sentence_gsv gsv;
    struct minmea_sentence_vtg vtg;
    struct minmea_sentence_zda zda;
}minmea_messages;
```

Les données sont ensuite traitées dans l'application. Pour paramétriser le **CAM-M8C-0**, des messages **UBX** peuvent être envoyés. Pour ce faire, des éléments de la librairie *ubxlib* peuvent être repris, notamment les fonctions d'encodage et de décodage des messages **UBX**.

5.2 Configuration des périphériques

Les périphériques utilisés par le microcontrôleur doivent être paramétrés et initialisés. Le configurateur [harmony](#) permet de le faire simplement.



5.2.1 Timers

Lors de cette section, 2 timers vont être configurés, ceux-ci ont des utilités différentes :

Timer 1 sert pour les attentes bloquantes précises. 1 [ms]

Timer 2 gère les délais entre les mesures et l'affichage des LEDs. 10 [ms]

Dimensionnement timer 1

$$f_{sys} = 72 \text{ MHz}$$

$$f_{per} = 1 \text{ kHz}$$

$$presc = 8 [-]$$

$$C_{T1} = \frac{f_{sys}}{f_{per} * presc} - 1 = \frac{72 * 10^6}{1 * 10^3 * 8} - 1 = 8'999 [-]$$

Dimensionnement timer 2

$$f_{sys} = 72 \text{ MHz}$$

$$f_{per} = 100 \text{ Hz}$$

$$presc = 16 [-]$$

$$C_{T1} = \frac{f_{sys}}{f_{per} * presc} - 1 = \frac{72 * 10^6}{100 * 16} - 1 = 44'999 [-] \quad (1)$$

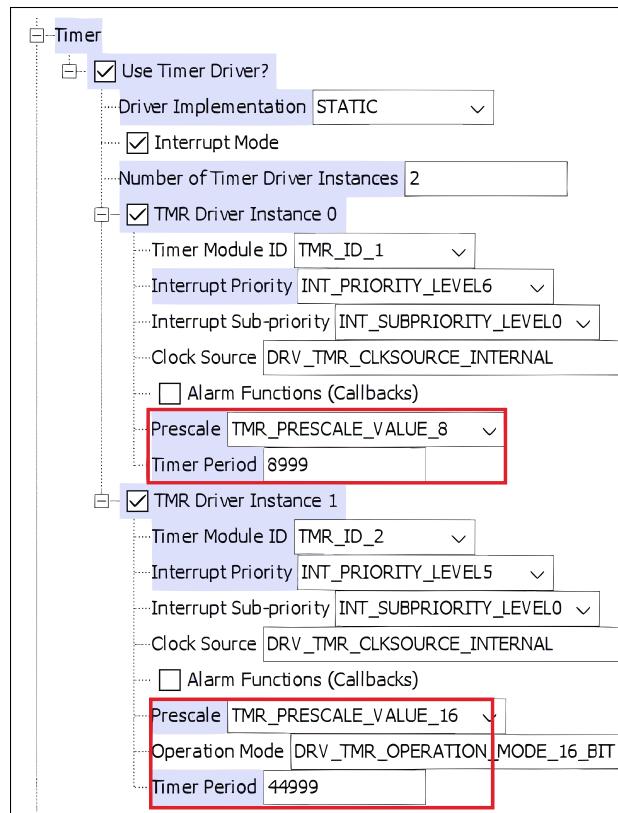


FIGURE 35 – Configuration des timers.

Source: Auteur

Code Source 5 – Interruption **timer 1**, 1 [ms]

```
void delayTimer_callback(){
    /* Increment delay timer */
    timeData.delayCnt++;
}
```

Code Source 6 – Interruption **timer 2**, 10 [ms]

```
void stateTimer_callback()
{
    /* Increment all counters */
    timeData.ledCnt++;
    timeData.measCnt[BN0055_idx]++;
    timeData.measCnt[GNSS_idx]++;
    timeData.tmrTickFlag = true;
    /* When the button is pressed, the hold time is counted. */
    if(timeData.flagCntBtnPressed){
        timeData.cntBtnPressed++;
    }
    /* Do debounce on button every 10 ms */
    DoDebounce(&switchDescr, ButtonMFStateGet());
    /* Start a measure set each IMU period */
    if( ( timeData.measCnt[BN0055_idx] %
        → (timeData.measPeriod[BN0055_idx]/10) ) == 0)
        timeData.measTodo[BN0055_idx] = true;

    /* Start a measure set each GNSS period */
    if( ( timeData.measCnt[GNSS_idx] %
        → (timeData.measPeriod[GNSS_idx]/10) ) == 0)
        timeData.measTodo[GNSS_idx] = true;
    /* Manage LED if enabled */
    if((timeData.ledCnt >= LED_PERIOD) && (appData.ledState ==
        → true))
        LED_B0ff();
}
```

Dans le code 6, nous pouvons voir le code de la routine d'interruption du timer 2. Lors de cette interruption, les différents compteurs de mesures sont incrémentés et différentes conditions vont tester si la période de mesure est atteinte. Une gestion du bouton a également lieu, notamment une mesure du temps d'appui, et une limitation des effets de rebonds mécaniques sur les signaux électriques observés par le MCU par un échantillonnage diminué. Enfin, la LED de vie est gérée, notamment son temps allumé.

5.2.2 USARTs

Deux bus de communication UART ont dû être configurés dans le cadre de ce projet ; ceux-ci ont des utilités différentes.

Configuration des USARTs

ID du BUS	Utilité	Baudrate	Interruption	Trame	Parité
UART ID1	Réceptions commandes USB	9600	Priorité 1	8bits + 1stop	Non
UART ID2	Communication avec le GNSS	9600	Non	8bits + 1stop	Non

TABLE 12

L'USART ID1, mentionné dans le tableau 12, est configuré avec une interruption. Celle-ci a été mise en place afin d'implémenter un mécanisme de FIFO associé à un tampon de réception. Ceci facilite le traitement des données et évite leur perte lors de réceptions consécutives.

5.2.3 Carte SD

Harmony permet de configurer directement un périphérique carte-SD avec le bus SPI.

(a) Configuration Carte SD. This screenshot shows the configuration for an SD card driver. It includes fields for Driver Implementation (set to DYNAMIC), Number of SD Card Driver Clients (1), SD Card Driver Index (DRV_SDCARD_INDEX_0), Maximum Driver Indices (limit 2) (1), SD Card Data Queue Size (10), Clock To Use (CLK_BUS_PERIPHERAL_1), SD Card Speed(Hz) (10000000), and a checkbox for Enable Write Protect Check? (unchecked). It also shows Chip Select Port (PORT_CHANNEL_A) and Chip Select Port Bit (PORTS_BIT_POS_10). A checkbox for Register with File System? is checked. A note at the bottom says "SPI Driver Instance to use for SD Card Driver 0".

(b) Configuration SPI. This screenshot shows the configuration for a SPI driver instance. It includes fields for SPI Module ID (SPI_ID_1), SPI Interrupt Priority (INT_PRIORITY_LEVEL1), SPI Interrupt Sub-priority (INT_SUBPRIORITY_LEVEL0), Master\Slave Mode, Data Width, Buffer Mode (Allow Idle Run unchecked), Protocol Type (DRV_SPI_PROTOCOL_TYPE_STANDARD), Baud Clock Source (SPI_BAUD_RATE_PBCLK_CLOCK), Clock\Baud Rate - Hz (1000000), Clock Mode (DRV_SPI_CLOCK_MODE_IDLE_LOW_EDGE_FALL), Input Phase (SPI_INPUT_SAMPLING_PHASE_AT_END), Dummy Byte Value (0xFF), Max Jobs In Queue (10), and Minimum Number Of Job Queue Reserved For Instance (1).

FIGURE 36 – Harmony, driver carte-SD.

Source: Auteur

Sur la figure 36a, on peut observer la configuration de la carte SD. La broche de *Chip Select* est attribuée à **RA10**, qui correspond à **CS_SD** sur le schéma présenté à la figure 16a. La connexion est établie avec une fréquence de **10MHz**.

Concernant la figure 36b, les spécificités du bus SPI y sont configurées, incluant le mode de l'horloge et la phase de l'échantillonnage. Ces configurations ont été adaptées en adéquation avec les protocoles des cartes SD⁵.

5. SD Memory Card Interface Using SPI.

5.3 Application principale

Lors de cette section nous allons décrire le fonctionnement principale du système sous la forme d'un diagramme d'état sur la figure 37.

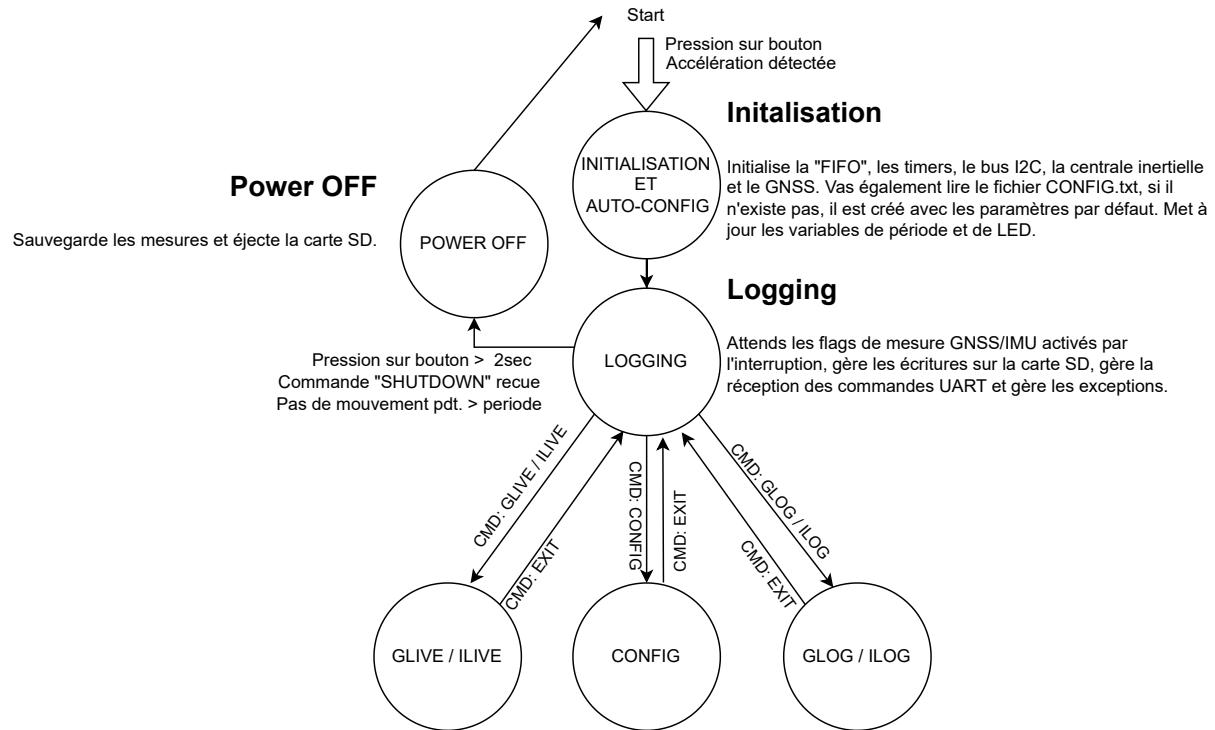


FIGURE 37 – Diagramme d'état principal.

Source: Auteur

5.3.1 Commandes USB-UART

Les commandes permettent d'interagir avec le système et de manipuler les différentes données de la boîte noire.

Liste des commandes

GLIVE	-LVG	: Permet d'afficher les données du GNSS en live sur le terminal sans les enregistrer.
ILIVE	-LVI	: Lance l'envoi des données live de l' IMU sans les enregistrer.
GLOG	-GL	: Envoie les données de localisation de la carte-SD au terminal.
ILOG	-IL	: Envoie les données inertie de la carte-SD au terminal.
GCLR	-GC	: Supprime les logs du GNSS .
ICLR	-IC	: Supprime les logs de l' IMU .
EXIT	X	: Quitte le mode en cours.
SHUTDOWN	-OFF	: Sauvegarde et éteint le système.
CONFIG	-CFG	: Démarrer le mode de configuration du système.
>	INTG :XXXX	: Dans mode config. permet de régler la période de mesure du GNSS .
>	INTI :XXXX	: Dans mode config. permet de régler la période de mesure de l' IMU .
>	LEDV :X	: Dans mode config. permet d'activer ou non la LED de vie.
>	TOFF :XX	: Dans mode config. permet de fixer le temps d'inactivité (éteint le système).

TABLE 13 – Commandes disponibles

Les commandes de la table 13 peuvent également être envoyées en caractères minuscules.

Sur la figure 38, la séquence de la boucle principale de l'application est représentée. Celle-ci interagit avec les différents périphériques en utilisant diverses bibliothèques C.

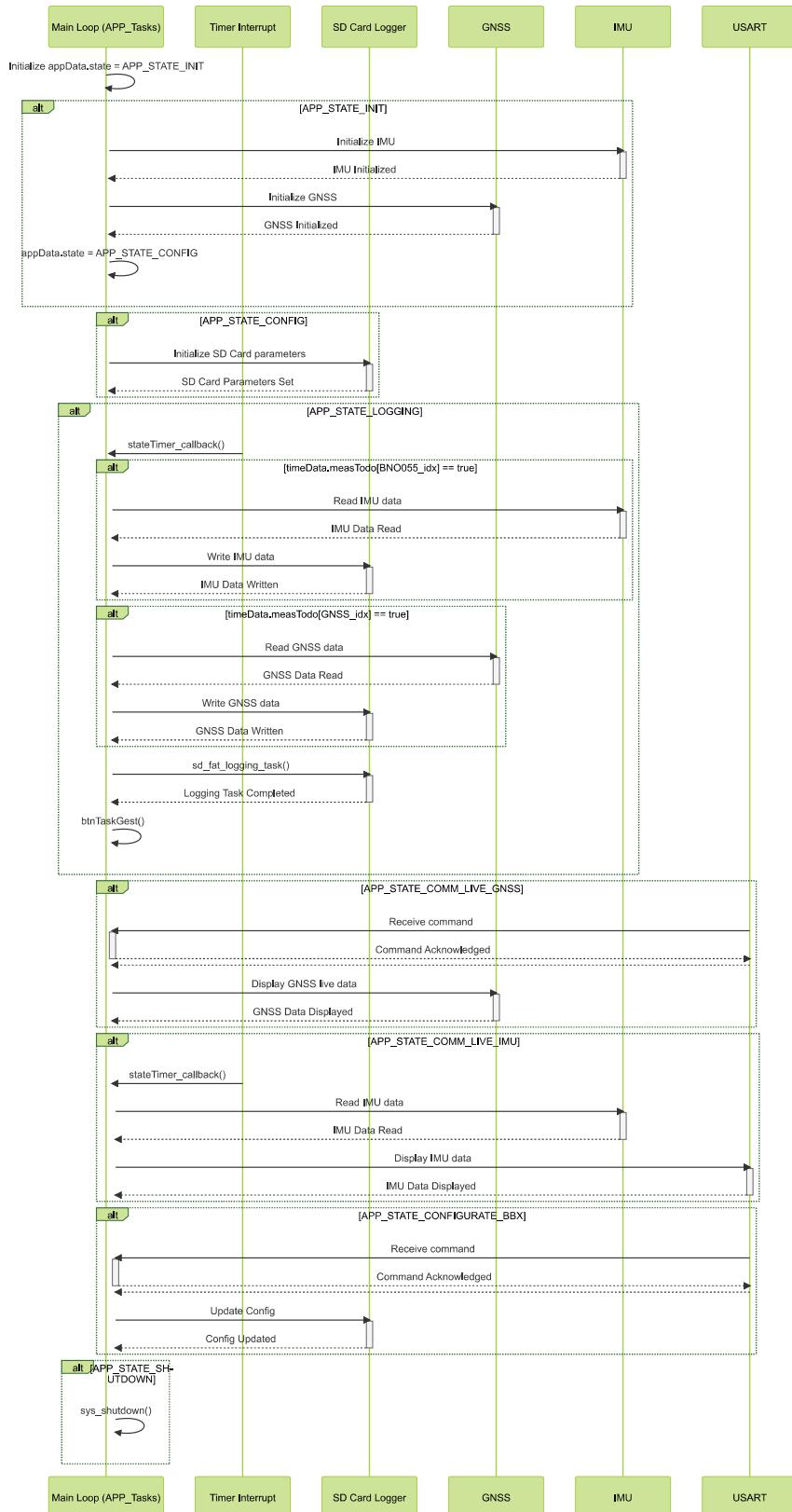


FIGURE 38 – Diagramme de séquence principal.

Source: Auteur

5.3.2 État de logging du système

L'état de logging est représenté par un flowchart sur la figure 39.

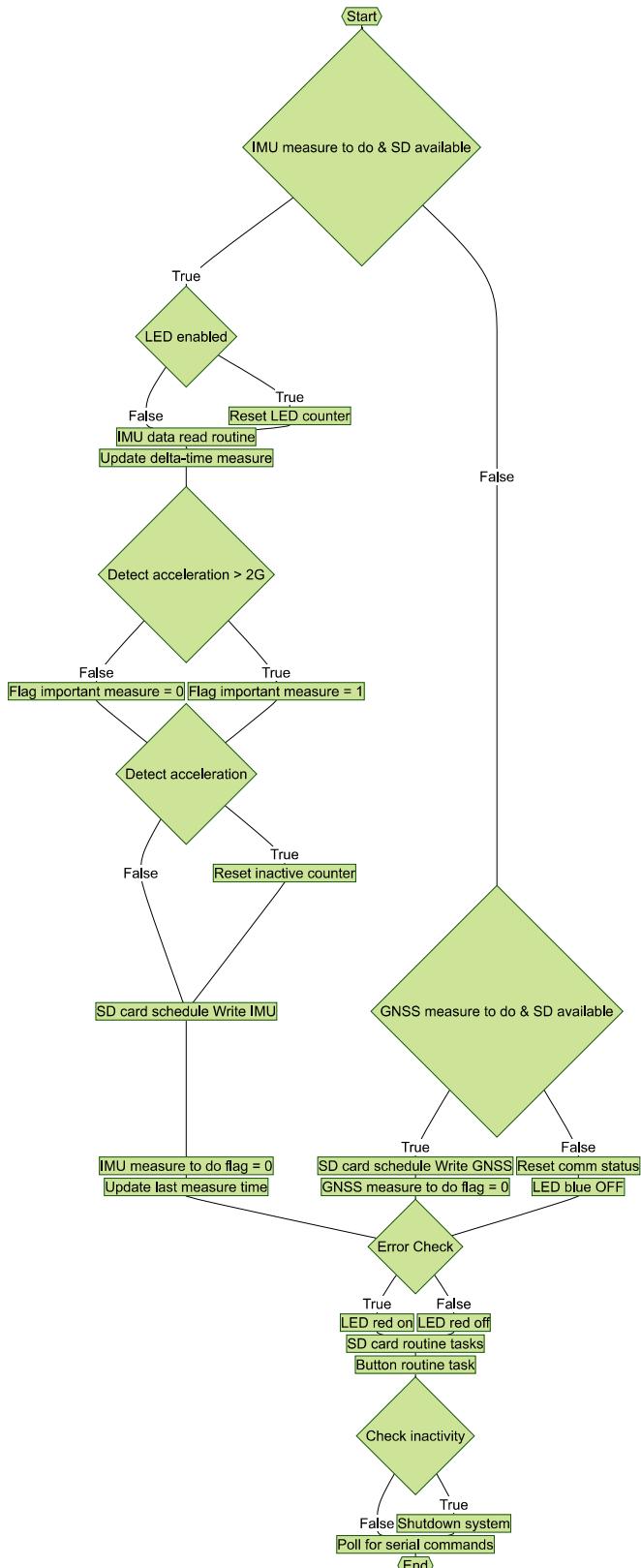


FIGURE 39 – Flowchart de l'état de logging.

Source: Auteur

5.4 Carte SD

Le code de gestion de fichier de la carte SD est divisé en deux machines d'états. La première est dédiée à l'initialisation, à la lecture et à la modification du fichier de configuration. La seconde s'occupe du logging et de l'enregistrement des mesures dans des fichiers distincts. Ces machines d'états sont non-bloquantes et doivent être appelées périodiquement pour permettre une routine de gestion de la carte SD.

5.4.1 Carte SD : initialisation et configuration

La machine d'état initiale est appelée lorsqu'elle se trouve dans l'état **APP_STATE_CONFIG**, comme illustré sur la figure 38. Elle continuera d'être appelée jusqu'à ce qu'elle atteigne l'état **IDLE**. Notons également que cette machine d'état est sollicitée en mode configuration (*init = false*) durant l'état **APP_STATE_CONFIGURATE_BBX**, visible sur la même figure 38. Ce processus est représenté sous forme de diagramme d'état à la figure 40

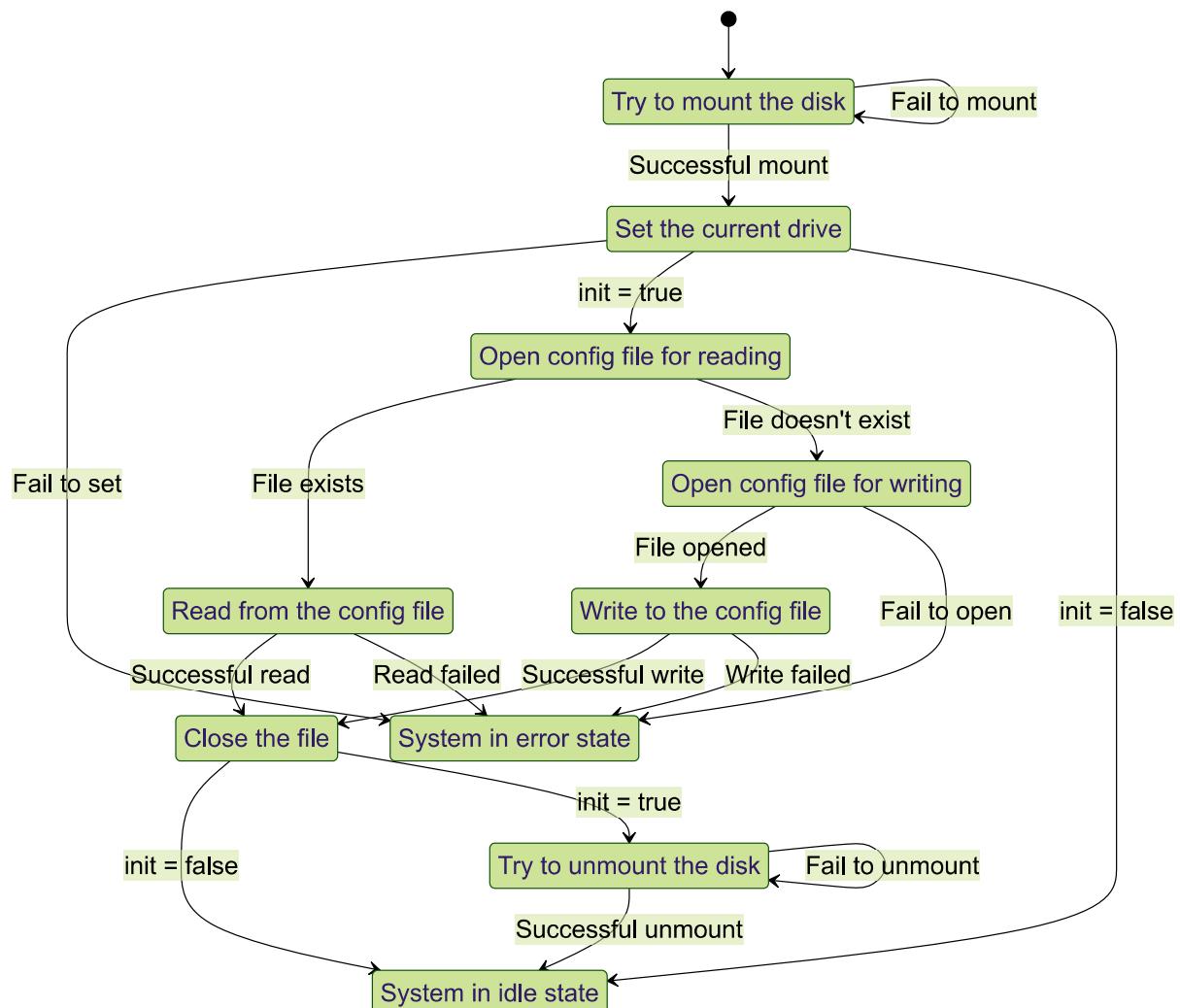


FIGURE 40 – Application carte SD initialisation/config.

Source: Auteur

5.4.2 Carte SD : écriture des données

Cette machine d'état est systématiquement appelée lorsque le système se trouve dans l'état **APP_STATE_LOGGING**, comme illustré sur la figure 38. Ceci est réalisé via la fonction `sd_fat_logging_task()`. Pour programmer des écritures et interagir avec cette machine d'état, deux fonctions d'interface sont utilisées : `sd_IMU_scheduleWrite()` qui permet de planifier l'écriture de données de l'**IMU**, et `sd_GNSS_scheduleWrite()` pour l'écriture des données du **GNSS**.

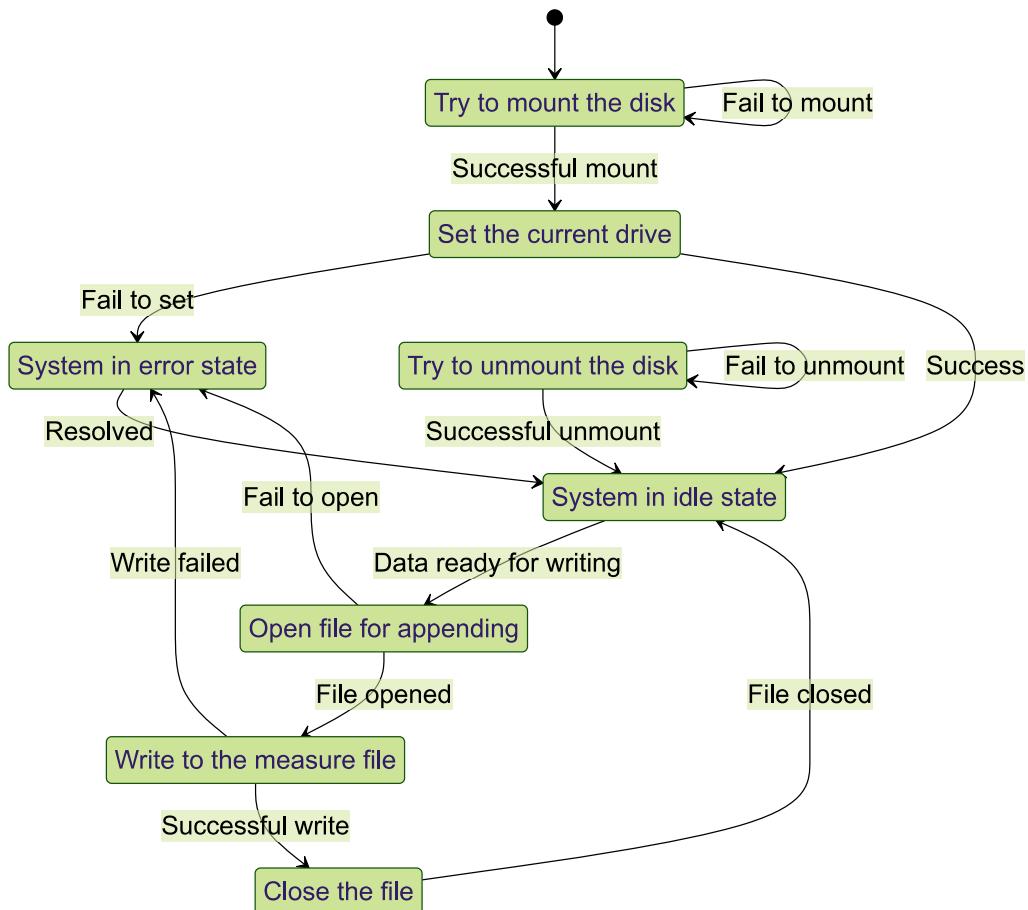


FIGURE 41 – Machine d'état logging.

Source: Auteur

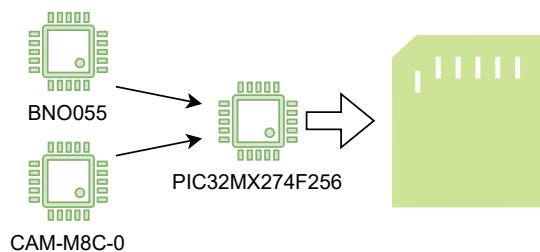


FIGURE 42 – Illustration des interactions.

Source: Auteur

5.5 Centrale inertie

Lors de cette section, nous allons décrire les différents aspects de la mise en place de la centrale inertie **BNO055**.

Interfaçage de l'IMU Bosch propose une librairie⁶ d'APIs pour communiquer avec la centrale inertie. Cette librairie est accompagnée d'un fichier *support* qui sert à lier les fonctions de protocoles aux fonctions bas niveau du bus de communication I2C.

5.5.1 Initialisation

Procédure d'initialisation

- * 1 Réinitialisation hardware par la pin RESET pendant 100ms.
- * 2 Désactivation de l'interruption⁷ de l'**IMU**.
- * 3 Attribution des fonctions bas niveau dans les pointeurs de la librairie.
- * 4 Obtention des informations du BNO055 (IDs, Version...).
- * 5 Attribution du mode d'alimentation "Normal".
- * 6 Choix du mode d'opération "AMG" (pour lecture simple Accel. magnitude et gyro.).
- * 7 Lecture initiale à vide, d'initialisation (Accel. magnitude gyroscope.).
- * 6 Choix du mode d'opération "NDOF" (pour lecture des données de fusions complexes.).
- * 7 Lecture initiale à vide, d'initialisation (Euler, Quaternion, gravité et accel. linéaire).
- * 8 Programmation de l'interruption du BNO055 pour la détection d'une certaine accélération.

5.5.2 Lecture des données

La phase cruciale concernait l'initialisation du BNO055. La lecture des diverses valeurs est assurée par une fonction spécifiquement créée, nommée `bno055_read_routine()`. Cette fonction récupère les données suivantes : Quaternion WXYZ, magnitude XYZ, gyroscope XYZ, angles d'Euler HPR, accélération linéaire XYZ, et gravité XYZ. Ces données sont stockées dans une structure locale au fichier *app.c* et possède la structure suivante :

```
typedef struct {
    s32 comres;
    bool flagMeasReady;
    uint8_t flagImportantMeas;
    struct bno055_gravity_double_t gravity;
    struct bno055_linear_accel_double_t linear_accel;
    struct bno055_euler_double_t euler;
    struct bno055_gyro_double_t gyro;
    struct bno055_mag_double_t mag;
    struct bno055_quaternion_t quaternion;
    unsigned long time;
    unsigned long l_time;
    uint16_t d_time;
}s_bno055_data;
```

6. https://github.com/BoschSensortec/BNO055_driver

7. Reset toutes les sources d'interruptions de l'**IMU** dans le registre 0x3F bit 6.

5.5.3 Système de détection de mouvements

La centrale inertuelle *BNO055* permet de configurer une interruption pour un certain événement et une certaine durée. Dans cette section, nous décrirons la configuration de cette interruption pour l'enclenchement du système ainsi que la possibilité d'activer/désactiver cette option.

Code Source 7 – Configuration de l'interruption

```
/* BNO055 motion interrupt mode */
1) bno055_set_accel_any_motion_no_motion_axis_enable
   ← (BNO055_ACCEL_ANY_MOTION_NO_MOTION_X_AXIS, 1);
2) bno055_set_accel_any_motion_no_motion_axis_enable
   ← (BNO055_ACCEL_ANY_MOTION_NO_MOTION_Y_AXIS, 1);
3) bno055_set_accel_any_motion_no_motion_axis_enable
   ← (BNO055_ACCEL_ANY_MOTION_NO_MOTION_Z_AXIS, 1);

4) bno055_set_accel_any_motion_thres(10);
5) bno055_set_accel_any_motion_durn(20);
6) bno055_set_intr_accel_any_motion(1);
7) bno055_set_intr_mask_accel_any_motion(1);
```

Les valeurs de configuration peuvent être définies en tant que constantes pour être plus facilement configurable lors de mise-à-jour firmware.

Description des lignes du code [7](#)

- * 1 Activation de l'axe X pour le mode *anymotion*.
- * 2 Activation de l'axe Y pour le mode *anymotion*.
- * 3 Activation de l'axe Z pour le mode *anymotion*.
- * 4 Attribution de la valeur de déclenchement à $78.3mg = 0.768m/s^2$.
- * 5 Configuration de la durée à 20ms.
- * 6 Active l'interruption, accélération.
- * 7 Masque l'interruption.

5.5.3.1 Gestion de l'interruption L'interruption est réinitialisée au démarrage du système ainsi que lors de sa mise en veille.

5.5.3.2 Mode low power lorsque le système se met en veille, la *BNO055* est configurée pour fonctionner uniquement avec son accéléromètre et son mode de puissance passe en *low power*.

Code Source 8 – Configuration de l'interruption

```
/* Set acceleration only operation, power saving */
bno055_set_operation_mode(BNO055_OPERATION_MODE_ACCONLY);
/* set the power mode to LOW POWER */
bno055_set_power_mode(BNO055_POWER_MODE_LOWPOWER);
/* Reset interrupt pin */
bno055_set_intr_RST(1);
```

5.5.3.3 Configuration enclenchement automatique Le système propose deux modes d'opération : le premier permet un enclenchement automatique lors de la détection d'un mouvement, tandis que le second active le système suite à une impulsion sur le bouton. Il est à noter que pour activer le mode automatique, une première impulsion sur le bouton est nécessaire et le système doit avoir transité au moins une fois par l'état "shutdown" du code. Dans les deux modes, le système peut entrer en veille si aucun mouvement n'est détecté sur une durée configurée.

Lorsque le système est en mode d'enclenchement automatique, le circuit reste partiellement actif, ce qui engendre une consommation d'énergie détaillée ultérieurement dans la section "Validation du design". Même si la batterie est en mesure de supporter cette consommation, il reste possible de désactiver ce mode pour activer le système uniquement par le biais du bouton.

Configuration La centrale inertuelle peut être alimentée soit de manière indépendante, soit avec le reste du système, influençant ainsi les modes décrits précédemment. Pour configurer ces modes, deux options sont disponibles :

W6 connecté	IMU alimentation autonome.	Mode accélération auto-enclenchement.
W7 connecté	Alimentation dépendant du système.	Mode enclenchement bouton uniquement.
W6 & W7	Connexion à éviter !	

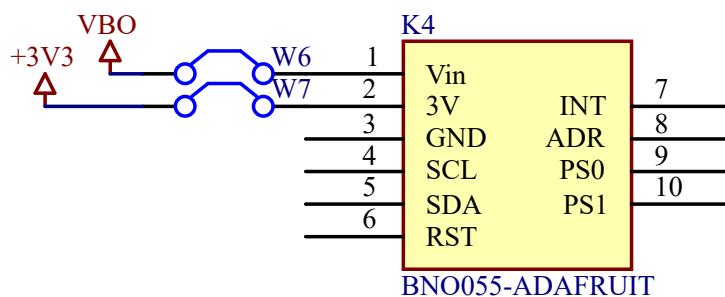


FIGURE 43 – Connexions des jumber du BNO055.

Source: Auteur

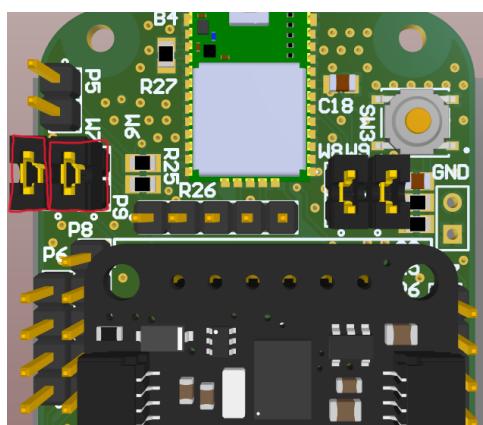


FIGURE 44 – Emplacement des jumpers sur le **PCB**.

Source: Auteur

Sur la figure 44, les jumpers sont entourés en **rouge**. Le jumper **W6** se situe à droite tandis que **W7** se situe à gauche.

5.6 Format des données

Dans cette section, nous décrirons les formats des données enregistrées sur la carte SD.

5.6.1 Données de l'IMU

Les données de la centrale inertielles sont enregistrer dans un fichier CSV et suivent un format fixe :

Flag accélération;Delta time;Gravity X;Y;Z;Gyroscope X;Y;Z;Magnitude X;Y;ZAccél. linéaire X;Y;Z;Angle Euler H;P;R,Quaternion W;X;Y;Z

```
0;500;-3.5900;-3.9900;8.2000;17.3750;141.6875;20.0625;-0.8750;28.3750;-17.8750;0.9700;-0.0300;-0.4700;19.4375;25.9375;-21.4375;15475;-2902;3674;-2655;
0;500;-9.1700;-3.4500;-0.2200;6.0625;184.1250;-153.9375;21.0000;26.7500;-9.3750;0.0200;-1.0300;-0.3300;35.1875;93.6875;-69.3125;11140;-521;11538;-3308;
0;500;-0.7300;-2.2000;-9.5200;-26.1875;26.6250;1.5625;12.2500;26.7500;19.5000;0.3500;0.5500;0.0000;21.0625;166.9375;4.3125;272.2977;15992;-1936;
0;500;-9.1600;-2.9400;1.8500;21.7500;-80.5000;-52.6875;22.0000;24.1875;-14.5625;1.6800;-0.9300;-0.2700;45.2500;57.7500;-69.1875;11613;936;10478;-4787;
0;500;2.4100;-2.6900;9.1100;-32.1250;-58.6875;-29.3750;-14.3750;28.2500;-19.5625;1.2500;0.6500;-1.1100;24.1250;16.1875;13.6250;15752;-2619;-1450;-3367;
0;500;9.6700;-0.1800;-1.5900;18.1875;-87.3125;-61.8750;-26.8750;26.1875;6.5000;-0.5400;0.2900;-0.3300;10.6250;173.3125;80.5000;10648;-1472;-12328;-958;
0;500;5.2100;-2.8900;7.7800;-2.2500;129.8125;-21.5625;-24.0625;25.2500;-7.7500;0.5000;0.5100;-0.8700;26.8125;20.3750;32.0625;15093;-3554;-3876;-3605;
```

FIGURE 45 – Données CSV.

Source: Auteur

Exemple d'un set de données CSV Ayant précédemment développé un script Python, nous pouvons visualiser simplement et rapidement les données de logs inertielles sous forme de courbes, comme sur la figure 46.

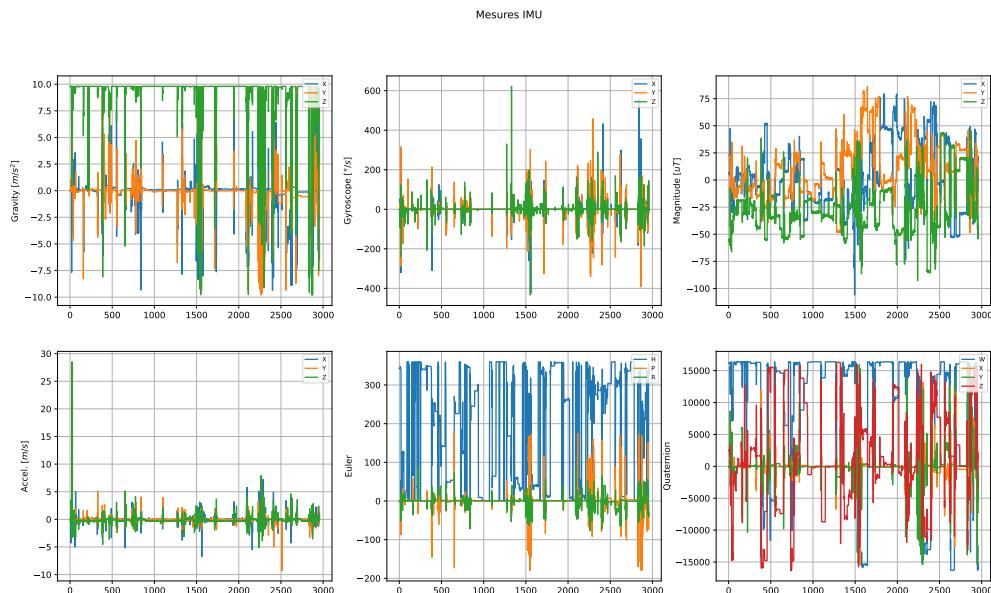


FIGURE 46 – Visualisation des données inertielles.

Source: Auteur

Il est important de souligner que, lors de la prise des mesures présentées à la figure 46, le module était en déplacement. Nous constatons que 3000 mesures ont été effectuées et sur la figure 45 nous pouvons observer que l'intervalle entre les mesures était de 500ms, ce qui indique un enregistrement d'une durée de 25 minutes. Le prototype en l'état actuel ne permet pas de localiser suffisamment de sa

5.6.2 Données du GNSS

Les données du **GNSS** ont été sauvegardées en format NMEA dans un fichier texte sans modifications. Cela a été fait afin de permettre des analyses plus approfondies lors du développement en important les données sur ordinateur, plutôt que de les analyser directement sur le microcontrôleur. Toutefois, il serait tout à fait possible d'analyser les messages NMEA sur le microcontrôleur et d'enregistrer uniquement les données pertinentes sur la carte SD. Une telle approche permettrait d'économiser de l'espace de stockage sur la carte SD. Néanmoins, comme nous l'avons observé dans la section [2.4.4](#), une telle économie ne serait pas nécessaire étant donné l'importante marge de capacité.

Exemple de logs NMEA Les données de la figures [47](#) contiennent différents messages NMEA (colorés en bleus) loggés lors d'un essai.

```
$GNGSA,A,1,,,,,,,,,,99.99,99.99,99.99*2E
$GPGSV,1,1,00*79
$GLGSV,1,1,00*65
$GNGLL,,,,,,V,N*7A
$GNRMC,,V,,,,,,,N*4D
$GNVTG,,,,,,N*2E
$GNGGA,,,,,0,00,99.99,,,,,*56
```

FIGURE 47 – Données NMEA loggées.

Source: Auteur

Sur la figure [47](#), nous constatons qu'au moment de la prise de ces logs, aucun satellite n'a été détecté (le message GSV indique **00**) et que les valeurs reçues n'étaient pas valides (comme le montrent les **V** colorés en rouge). Il est probable que ce problème de réception soit dû à un plan de masse autour de l'antenne insuffisamment étendu, empêchant un blindage adéquat pour la réception du signal. Pour remédier à ce problème, des tests avec des antennes externes de différents types sont actuellement en cours. Toutefois, dans son état actuel, le prototype est capable de recevoir des données d'heure et de date, et peut se connecter à jusqu'à 4 satellites (maximum observé lors de tests extérieurs. Le prototype pourrait être sujet à de meilleures conditions dans des zones à signaux forts).

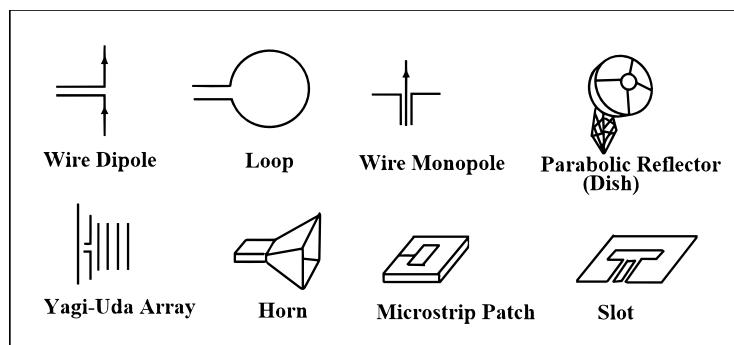


FIGURE 48 – Types d'antennes.

Source: <https://www.fiberglassantennas.com/antenna-types.html>

5.7 Fonctions des fichiers

Dans cette section, nous examinerons les différentes fonctions et variables mises à disposition et utilisées par les différents fichiers des bibliothèques C créées. Pour illustrer cela de manière pratique, nous utiliserons une représentation sous forme de classes.

5.7.1 Centrale inertie

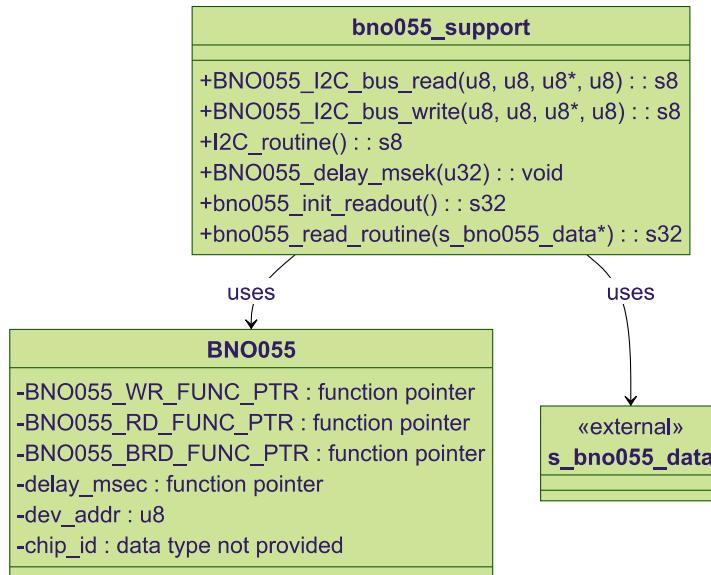


FIGURE 49 – Classe de la librairie de l’IMU.

Source: Auteur

5.7.2 Carte SD

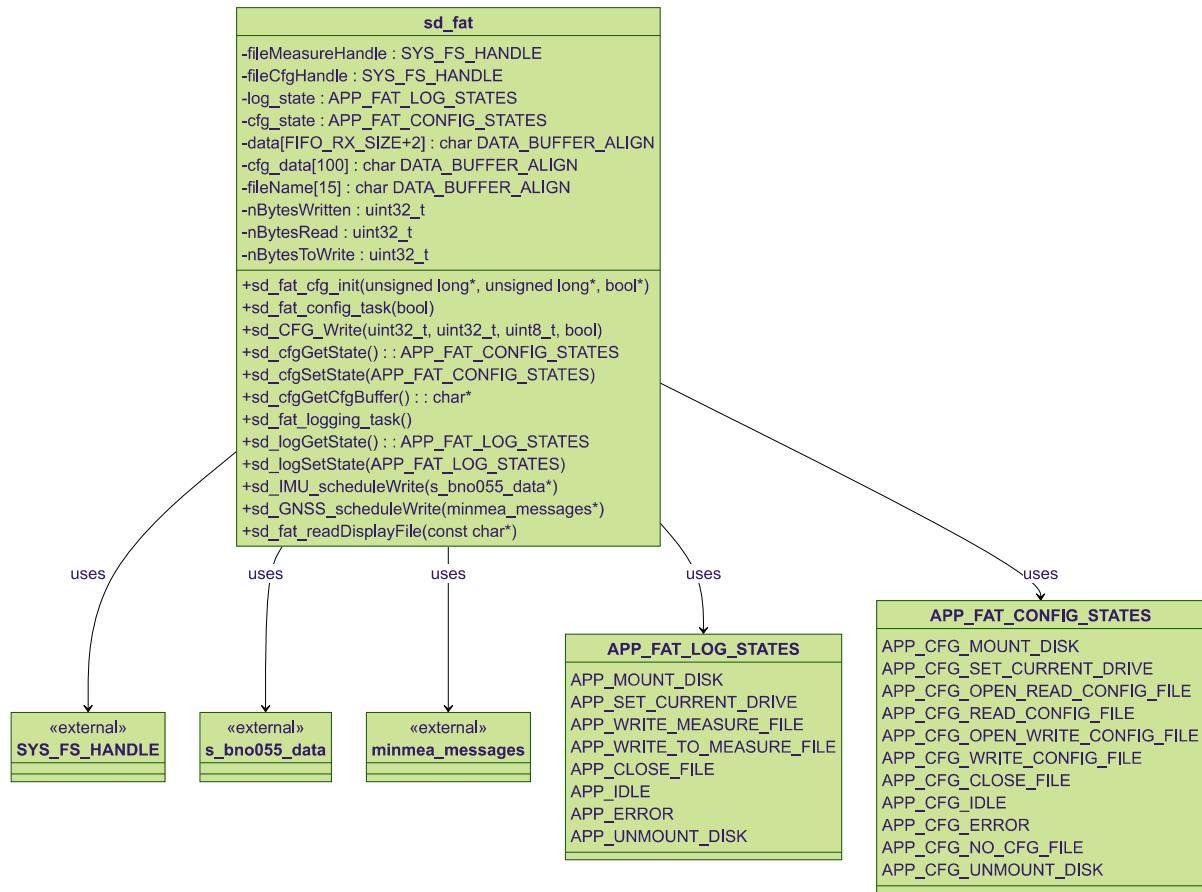


FIGURE 50 – Classe de la librairie de la carte SD.

Source: Auteur

5.7.3 Communication série avec FTDI

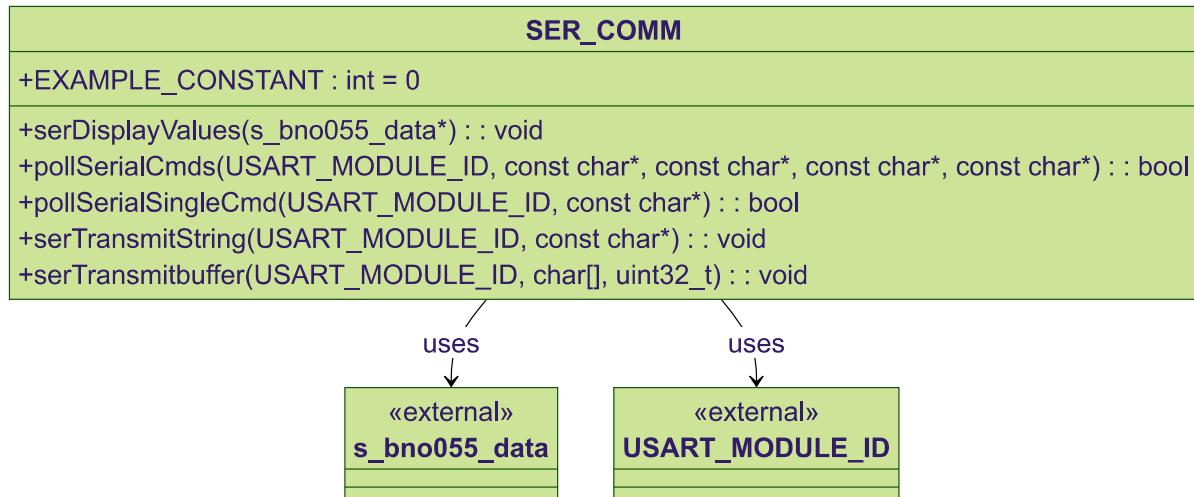


FIGURE 51 – Classe de la librairie communication USART.

Source: Auteur

6 Développement application interface

Étant donné que la boîte noire développée offre plusieurs commandes et permet une communication USB-C avec divers appareils, la création d'une interface graphique est particulièrement adaptée. Cette application facilite l'accès aux données en temps réel et aux logs, et permet également de configurer aisément la boîte noire. Dans cette section, nous décrirons la démarche suivie pour créer cette application.

6.1 Choix de l'environnement

Pour développer cette application, le langage de programmation Python a été choisis. Ce langage est particulièrement adapté pour concevoir des applications graphiques grâce à la bibliothèque Tkinter. Il s'agit d'une bibliothèque graphique libre pour Python qui facilite la création d'applications compatibles avec les systèmes d'exploitation les plus populaires auprès du grand public.

Tkinter permet d'utiliser des objet/widgets pour facilement mettre en place une interface graphique de façon très intuitive.

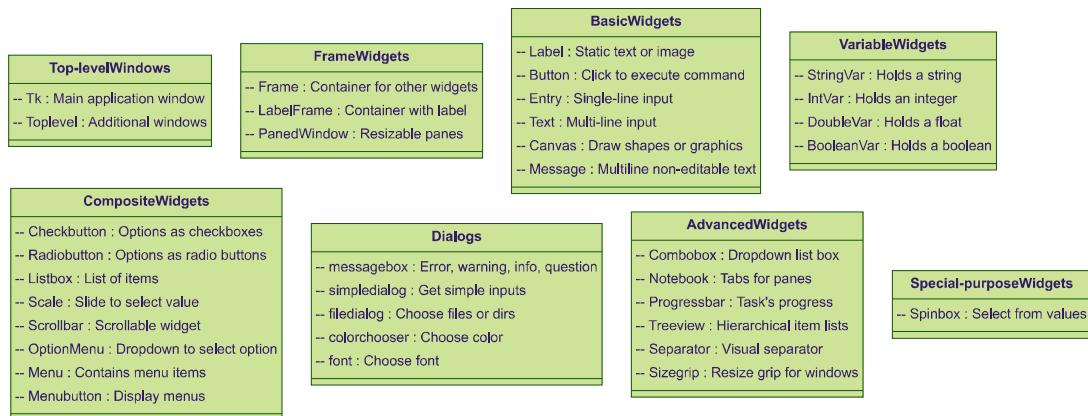


FIGURE 52 – Widgets Tkinter.

Source: Auteur

Le code python complet de l'application est disponible en annexe de ce rapport.

Code Source 9 – Extrait de code, création de la forme.

```
# Setup main window
gRoot = Tk()
gRoot.config(bg="white")
gRoot.geometry("1080x640")
gRoot.title("Black Box Connect")
# Style configuration
sty = ThemedStyle(gRoot)
sty.set_theme('radiance')
```

Comme illustré sur la figure 53, différentes fonctions de l'application sont visibles. Le port de communication série ainsi que le baud rate sont configurables. À la fermeture de l'application, le port se ferme automatiquement. L'application permet d'afficher les données des périphériques en temps réel *LIVE* ou sous forme de *LOGS*. Elle offre également la possibilité de supprimer les fichiers de logs stockés sur la carte SD et de sauvegarder les données reçues dans divers formats tels que "CSV, TXT...", comme le démontre la figure 55. Enfin, la configuration de la boîte noire est accessible via le menu présenté sur la figure 54.

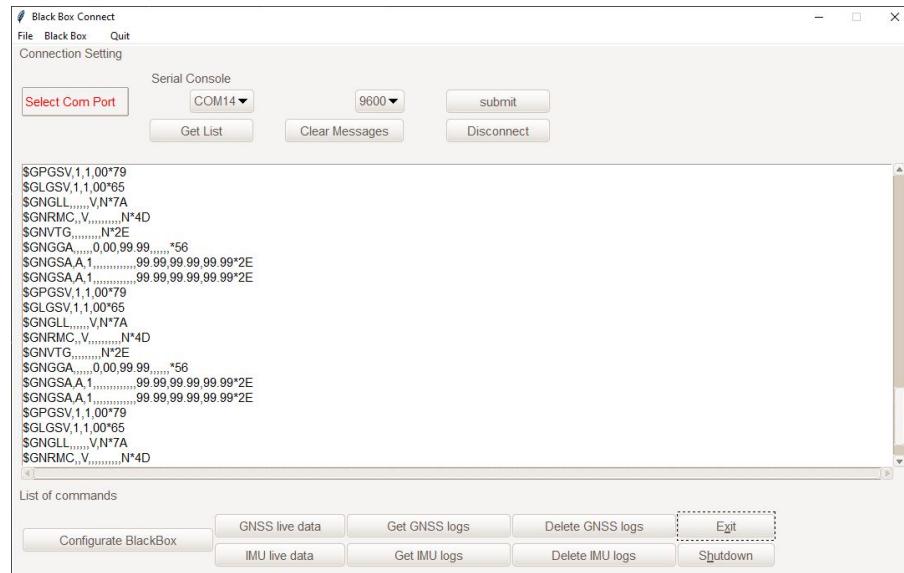


FIGURE 53 – Application, logs GNSS obtenus.

Source: Auteur



FIGURE 54 – Menu configuration.

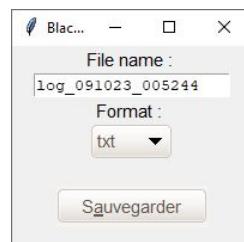


FIGURE 55 – Menu de sauvegarde.

Source: Auteur

7 Validation du design

Dans cette section, nous décrirons la procédure de vérification des caractéristiques du projet ainsi que sa validation.

7.1 Liste de matériel

- **P1** : Oscilloscope Tektronix RTB2004 ES.SLO2.05.01.11
- **P2** : Multimètre GwInsteek GDM-396 ES.SLO2.00.00.94
- Carte Mini-Boite-Noire 1924B

7.2 Consommations

Dans cette section, nous mesurerons les différentes consommations du système. Cette étape est importante pour caractériser le système et déterminer son autonomie.

7.2.1 Méthode de mesure

L'objectif est de basculer entre les différents modes (veille, logging...) de la carte et d'en mesurer la consommation. À cet effet, un ampèremètre a été placé en série avec la batterie, comme illustré à la figure 56.

7.2.2 Schéma de mesure

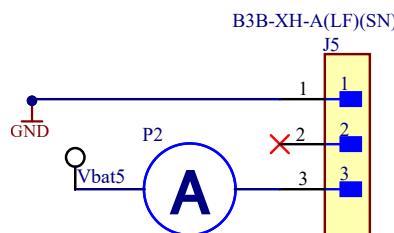


FIGURE 56 – Schéma de mesure, courant
Source: Auteur

7.2.3 Mesures

Index	Etat du système	Condition	Courant [mA]	Symbol
[1]	Eteint.	Mode auto-enclenchement OFF.	0.02	I_{off}
[2]	Veille.	Est passé par l'état SHUTDOWN.	4.4	I_{sleep}
[3]	Veille brutale. ⁸	Pas passé par l'état SHUTDOWN.	9.56	I_{ws}
[4]	Initialisation.	-	90	I_{init}
[5]	Logging.	-	100	I_{log}
[6]	Shutdown.	-	83	I_{sh}
[7]	Communication.	USB branché.	0	I_{usb}

TABLE 14 – Mesure des consommations
Source: Auteur

8. Lorsque le système a été éteint de façon non-contrôlée (Batterie débranchée).

Nous pouvons par les mesure de la table 14 déduire les éléments suivants :

Où :

Capacité de la batterie $C = 1600 \text{ mAh}$

- Temps de logging (Table 14-[5]) : $T_l = \frac{C}{I_{log}} = 16h.$
- Temps épuisement batterie en veille (Table 14-[2]) : $T_l = \frac{C}{I_{sleep}} = 364h = 15J.$
- Temps épuisement en veille brutale (Table 14-[3]) : $T_l = \frac{C}{I_{ws}} = 167h = 7J.$

Par conséquent, les caractéristiques d'autonomie de la batterie sont suffisantes pour notre application.

7.2.3.1 Proposition de stratégies d'économie d'énergie Afin de diminuer la consommation lors du mode veille il existe différentes possibilités :

Débraser la LED de la carte BNO0555 d'adafruit

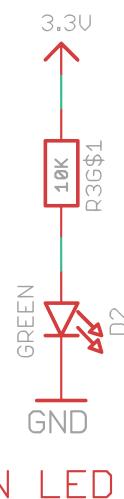


FIGURE 57 – LED de vie de la carte d'adafruit (2.2V).

Source: Schéma de la [carte BNO055 d'adafruit](#)

Comme le montre la figure 57, une LED reste constamment allumée lors de la mise sous tension de la carte de l'**IMU**. Elle présente une différence de potentiel de 2.2V à ses bornes, entraînant une consommation constante de 0.11mA. Bien que cette consommation soit faible, elle est inutile. Dans le cadre de ce prototype, la LED n'a pas été débrasée, mais un adhésif a été collé dessus afin de limiter la pollution lumineuse sur la LED d'état qui passe par un guide-lumière.

Permettre de changer de mode plus facilement Permettre à l'utilisateur de facilement basculer entre le mode "auto-enclenchement" et "enclenchement manuel" permettrait d'éviter des consommations inutiles.

7.3 Bus de communications

Avec le code implémenté dans le microcontrôleur, les différents périphériques fonctionnent correctement et la communication avec les différents bus est opérationnelle. C'est pourquoi, dans cette section, l'objectif principal est de mesurer la qualité et l'intégrité des signaux plutôt que d'analyser les différents protocoles.

7.3.1 Communication I2C

La communication I2C s'effectue entre la centrale inertuelle BNO055 et le microcontrôleur. Par ce bus, les configurations du registre de l'**IMU** ainsi que les données inertielles mesurées, en fonction de ces configurations, sont transmises. Comme nous avons pu le voir précédemment, cette communication est effective et les données sont cohérentes. Dans ce contexte, nous examinerons si l'intervalle entre les mesures est respecté, la durée de ces mesures, ainsi que la qualité des signaux I2C.

7.3.1.1 Méthode de mesure La boîte noire a été configurée pour transmettre et enregistrer des données inertielles à intervalles de 2 secondes. De plus, les mesures ont été prises pendant la transmission des données inertielles, et non lors de la configuration. Enfin, les trames ont été décodées automatiquement par l'oscilloscope grâce à sa fonction "Protocol".

7.3.1.2 Schéma de mesure Le schéma de mesure est présenté sur la figure 58.

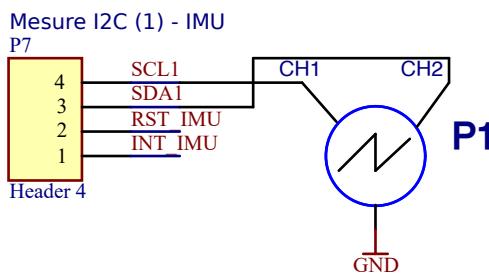


FIGURE 58 – Schéma de mesure, mesures I2C.

Source: Auteur

7.3.1.3 Mesures Comme le montre la figure 59, il y a un délai de 1.98 secondes entre l'envoi de deux sets de données de la centrale inertuelle.

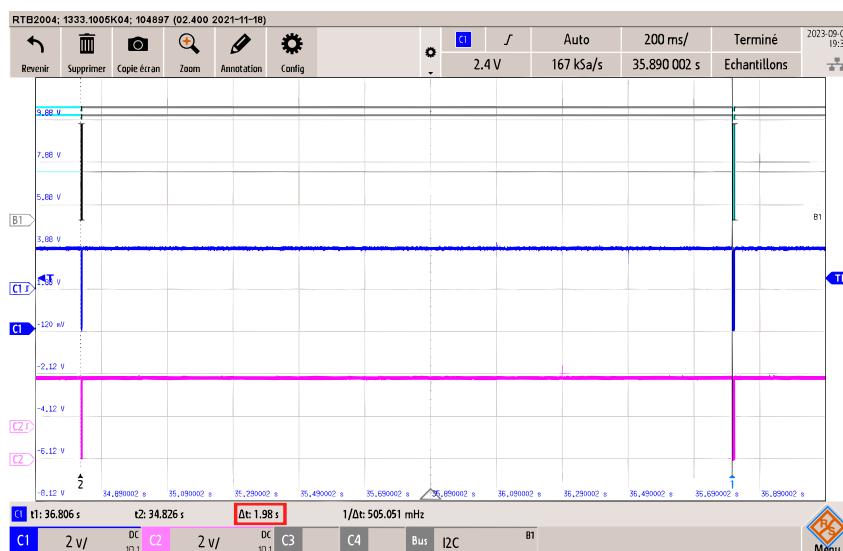


FIGURE 59 – Intervalle entre deux mesures, config. 2 secondes.

Source: Auteur

Malgré la légère différence sur la figure 59 de $20ms$, probablement due à une imprécision de mesure, nous pouvons conclure que l'intervalle entre les mesures respecte bien la configuration établie, même après modifications de celle-ci.

Sur la figure 60, nous observons la durée d'une trame de mesures qui englobe toutes les données mentionnées dans la section 5.6.1. La durée de transmission d'un set de mesure est de $2.274ms$. Compte tenu que le bus est configuré en mode "fast" à $400kbit/s$, cela suggère qu'environ ~ 910 bits ont été transmis, ce qui explique la densité des données rendant la figure peu lisible.

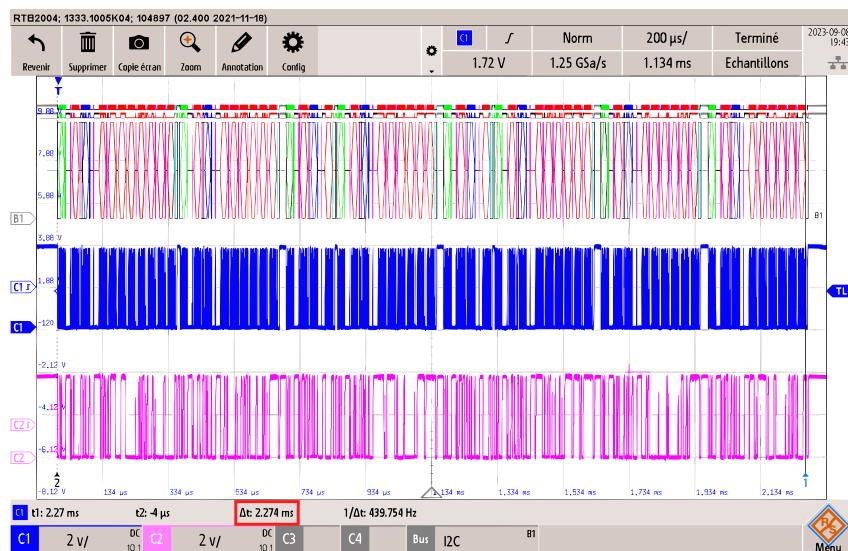


FIGURE 60 – Durée d'une mesure.

Source: Auteur

Sur la figure 61, nous observons le début d'une communication avec l'**IMU**. Le premier byte (**0x28**) correspond à l'adresse du BNO055. Ensuite, le registre **0x28** est adressé : il s'agit du registre contenant les données **LSB** de l'accélération linéaire. Les flancs des bits transmis semblent suffisamment nettes et peu parasités pour une bonne communication.

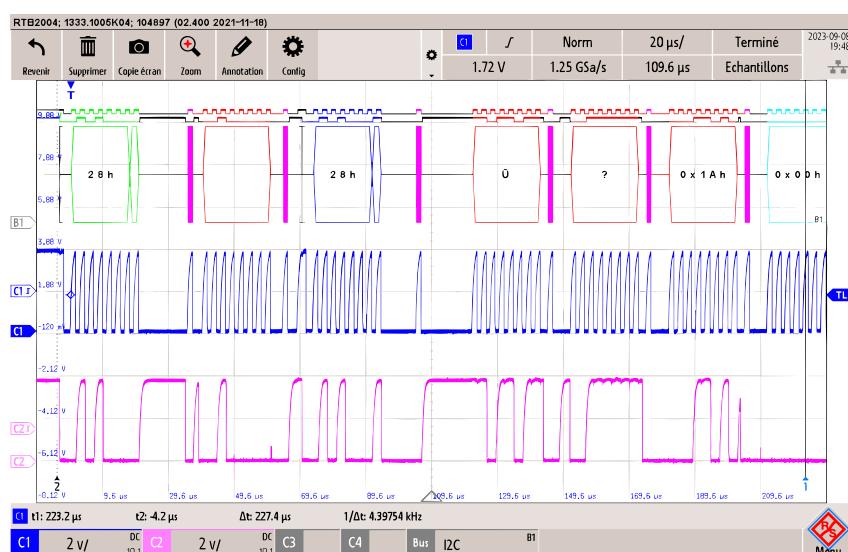


FIGURE 61 – Début de trame d'une mesure.

Source: Auteur

7.3.2 Communication UART GNSS

Dans cette section, nous analyserons les différentes caractéristiques de la communication UART entre le microcontrôleur et le module **GNSS**.

7.3.2.1 Méthode de mesure Pour réaliser ces mesures, le système a été activé et le **GNSS** a été configuré par défaut à une fréquence de **1Hz**.

7.3.2.2 Schéma de mesure Le schéma de mesure est présenté sur la figure 62, nous nous intéressons ici, qu'au message envoyés par le GNSS dans sa configuration par défaut.

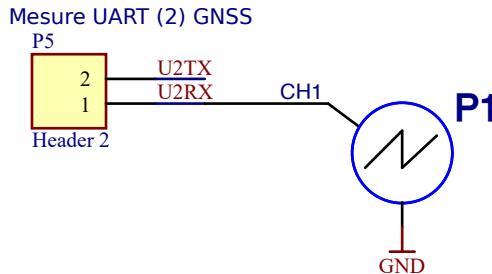


FIGURE 62 – Schéma de mesure **GNSS**.

Source: Auteur

7.3.2.3 Mesures Sur la figure 63, nous observons l'intervalle entre les messages GNSS. Effectivement, cet intervalle est de 1s, offrant ainsi suffisamment de temps au **MCU** pour traiter ces données, que ce soit pour les afficher sur un terminal ou les enregistrer sur la carte SD.

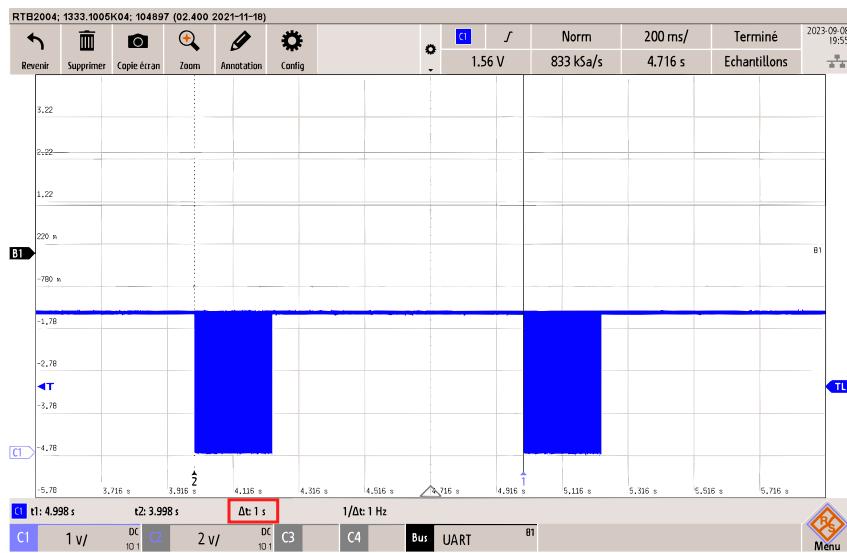
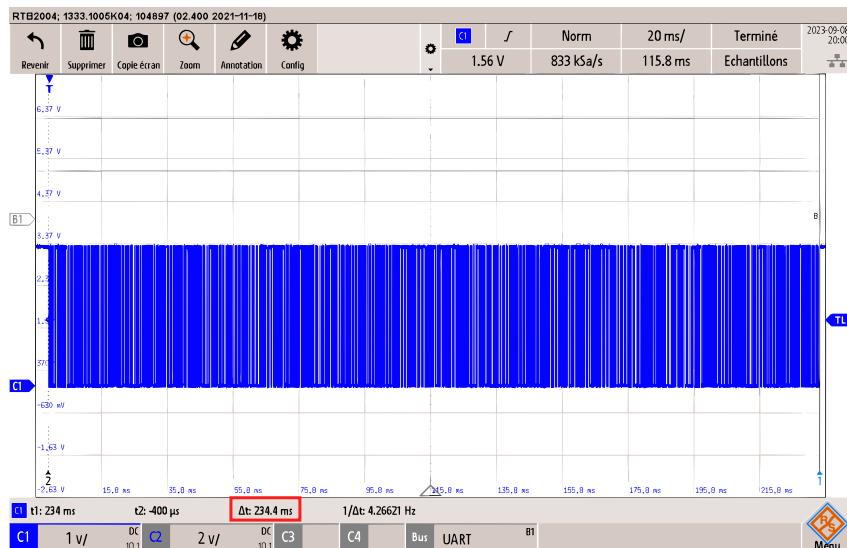


FIGURE 63 – Intervalle entre les données NMEA.

Source: Auteur

Sur la figure 64, nous constatons que la durée d'envoi des messages **NMEA** est de **234.4ms**. Sachant que l'intervalle entre les transmissions est de **1s**, cela laisse une fenêtre de **756ms** au **MCU** pour traiter ces données. Cette période est largement suffisante pour la transmission des données via le second UART (opérant au même baud rate) ou vers la carte SD, dont la fréquence

est nettement supérieure ($10MHz$). Si nous considérons la durée des transmissions de l'**IMU**, qui est de $2.274ms$, nous estimons avec une marge qu'environ $\sim 400ms$ des $756ms$ disponibles sont utilisés pour traiter les données. Ce temps de traitement pourrait être encore diminué si la fréquence d'enregistrement du **GNSS** était baissée. Ainsi, nous pourrions envisager d'obtenir jusqu'à ~ 155 mesures d'**IMU** dans l'intervalle donné.



7.3.3 Communication SPI, carte SD

7.3.3.1 Méthode de mesure

7.3.3.2 Schéma de mesure Le schéma de mesure est présenté sur la figure

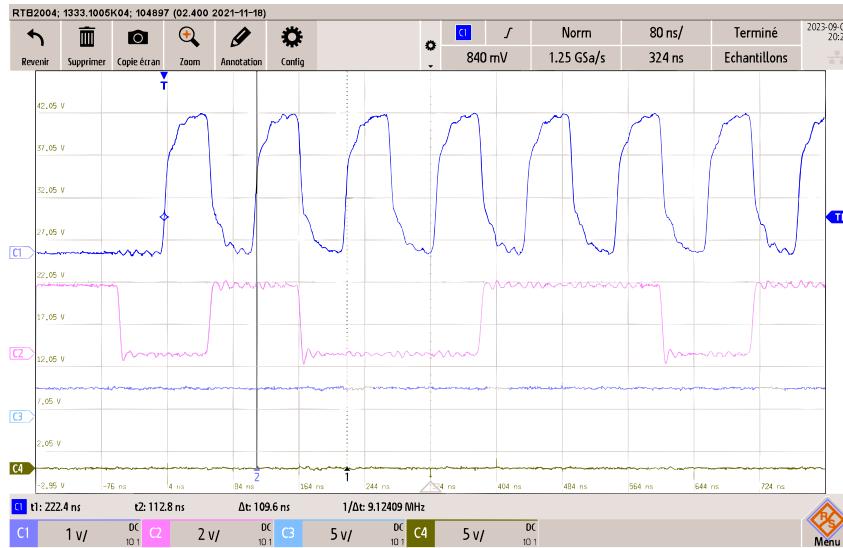


FIGURE 66 – Mesure fréquence clock SPI.

Source: Auteur

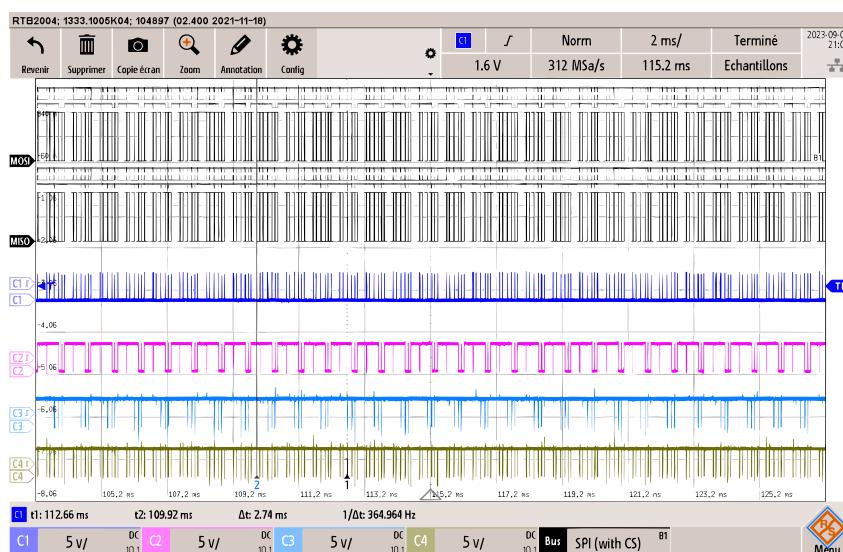


FIGURE 67 – Densité de la communication SPI.

Source: Auteur

7.3.3.3 Mesures

8 Caractéristiques du produit fini

9 Conclusion

10 Bibliographie

Références

- [1] P. Kordowski, Z. Jakielaszek, M. Nowakowski, and A. Panas, “Miniaturized flight data recorder for unmanned aerial vehicles and ultralight aircrafts,” in *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pp. 484–488, 2018.

11 Annexes