

# Localisation sous-marine

## Système de logging pour déplacement de module sous-marin.

Ali Zoubir

Rapport de projet



Génie électrique  
École supérieure  
Suisse

14 juin 2023

## Table des matières

1	Cahier des charges	4
1.1	Description . . . . .	4
1.2	Aperçu . . . . .	4
1.3	Tâches à réaliser . . . . .	5
1.4	Description des blocs . . . . .	6
1.5	Jalons principaux . . . . .	7
1.6	Livrable . . . . .	7
2	Pré-étude	8
2.1	Fonctionnement du système . . . . .	8
2.1.1	Schéma bloc . . . . .	8
2.2	Choix des composants importants . . . . .	10
2.2.1	Senseur absolu . . . . .	10
2.2.2	Capteur de pression . . . . .	12
2.2.3	Affichage . . . . .	12
2.2.4	Carte SD . . . . .	13
2.2.5	Real Time Clock . . . . .	14
2.2.6	Microcontrôleur . . . . .	14
2.2.7	Batterie, charge et régulation . . . . .	15
2.3	Estimation des coûts . . . . .	16
2.4	Synthèse développement . . . . .	16
3	Développement schématique	17
3.1	Scéma bloc détaillé . . . . .	17
3.2	Choix des composants . . . . .	18
3.2.1	Microcontrôleur . . . . .	18
3.3	Dimensionnements . . . . .	19
3.3.1	Vue d'ensemble schématique . . . . .	19
3.3.2	Autonomie du système . . . . .	20
3.3.3	LED Interface . . . . .	22
3.3.4	Adaptation mécanique . . . . .	23
3.3.5	Bus de communications . . . . .	24
3.3.6	Périphériques . . . . .	26
3.3.7	Chargeur de batterie . . . . .	27
3.3.8	Synthèse et perspectives de l'étude . . . . .	28
4	Développement du PCB	28
4.1	Bill of materials . . . . .	28
4.2	Mécanique du projet . . . . .	29
4.2.1	Considérations mécaniques . . . . .	29
4.3	Placement des composants . . . . .	30
4.4	Mécanique du PCB . . . . .	32
4.5	Routage . . . . .	33

5 Développement firmware	34
5.1 Configuration des PINs dans Harmony . . . . .	34
5.2 Configuration des périphériques dans Harmony . . . . .	35
5.2.1 Timers . . . . .	35
5.2.2 USART . . . . .	36
5.2.3 Carte SD - SPI . . . . .	36
5.3 Code . . . . .	37
5.3.1 Callbacks . . . . .	38
5.3.2 Centrale inertuelle BNO055 . . . . .	39
5.3.3 Carte SD . . . . .	40



# Localisation sous-marine 2022, V0.0

## 1 Cahier des charges

### 1.1 Description

L'objectif de ce projet, et de stocker des données de mesures du déplacement d'un module sous-marin par une centrale inertuelle, dans le but de mathématiquement le localiser depuis son point de départ (référence). Ceci, car la localisation sous-marine n'est pas une tâche aisée due aux différentes contraintes de communication sous-marine notamment que les ondes électromagnétiques ne se propagent pas facilement.

### 1.2 Aperçu

- Sauvegarde d'un set de donnée chaque 100ms.
- Profondeur d'utilisation maximum, de 60m.
- 2 heure de logging dans carte SD.
- Sensing sur 9 axes :
  - Mesures [Il est souhaitable que les capteurs choisis aient une faible dérive] ;
  - Accéléromètre 3-axes.
  - Gyroscope 3-axes.
  - Magnétomètre 3-axes.
  - Senseur de température
  - Profondimètre [0->10bar] [Res 1/10]
  - 3 à 5 slots libres MikroE pour autres mesures.
- Possibilité de sauvegarder la localisation de points d'intérêts par :
  - Bouton de sauvegarde [A définir : Magnétique, Optique, Mécanique ou autre].
- Batterie, autonomie minimum de 2 heures [ $\sim 10^{\circ}\text{C}$ ].
- Charge de la batterie par connecteur USB.
- (Optionnel) Lecture des données par connecteur USB (Interfaçage électronique, software optionnel dans cette version).
- (Optionnel) Interface LED ou petit écran.

### 1.3 Tâches à réaliser

Développement et intégration d'un PCB avec capteurs et logging sur carte SD dans une lampe de plongée étanche.

- Développement schématique
  - Fonctionnement MCU.
  - Périphériques de mesures et de sauvegarde / Bus de communication.
  - Gestion batterie
- Routage pour intégration dans boîtier de lampe de plongée 200x45mm.
- Programmation mesure et sauvegarde chaque 100ms.
  - Configuration MCU.
  - Configuration des périphériques de mesure pour 9-DOF.
  - Configuration des périphériques de sauvegarde (Carte SD).
  - Configuration et communication avec l'interface.
  - Communication et traitement des données mesurées.

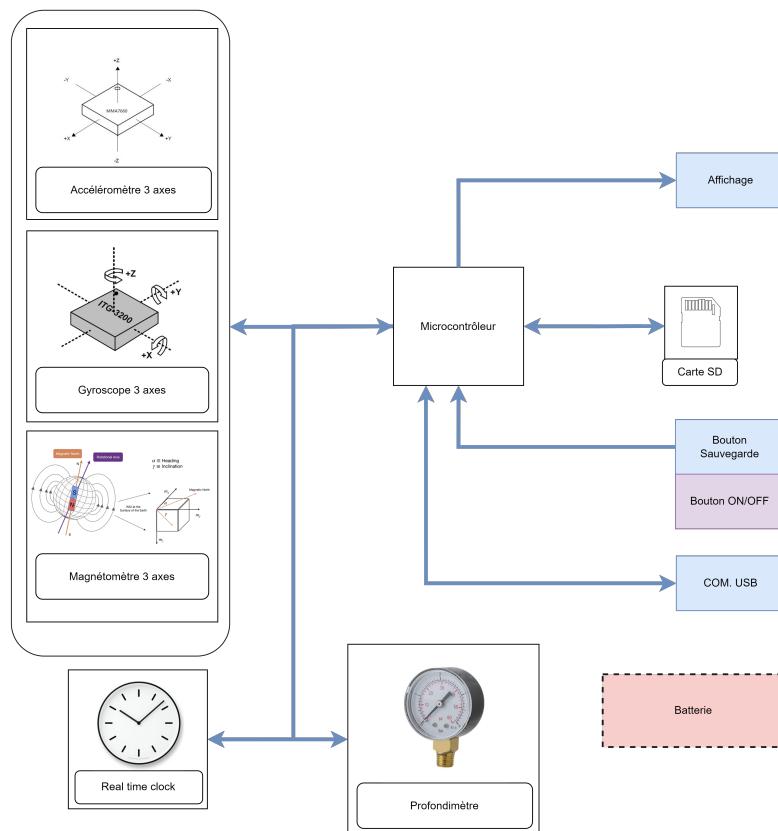


Figure 1 – Schéma de principe

Source : Auteur

## 1.4 Description des blocs

1. Carte SD :  
Stockage des données de mesures chaque 100ms, cœur du projet.
2. Accéléromètre-gyroscope-magnétomètre :  
Lecture des données individuelles brute ainsi que de fusion des capteurs, pour mesurer les déplacements sur 9 degrés de libertés.
3. Profondimètre :  
Mesure la pression pour déduire la profondeur, afin de corroborer les autres mesures des capteurs.
4. Real time clock :  
Permet de sauvegarder la temporalité du set de mesure dans la carte SD.
5. Affichage :  
Affichage LED ou écran, pour affichage pas encore définis (ex. Profondeur, état batterie...)
6. Bouton sauvegarde :  
Permet la mise en valeur d'un set de mesure. La forme de ce bouton n'est pas encore définie. Il sera peut-être fusionné avec le bouton ON/OFF.
7. Bouton ON/OFF :  
Permet d'allumer ou d'éteindre le système.
8. Batterie :  
Batterie du système, technologie à définir dans la pré-étude.
9. COM. USB :  
Permet de charger les batteries. Il faudra également prévoir dans cette version l'interface électronique pour la lecture de la carte SD directement par le port USB.
10. Microcontrôleur :  
Lis et traite les valeurs des capteurs, sauvegarde dans la carte SD...

## 1.5 Jalons principaux

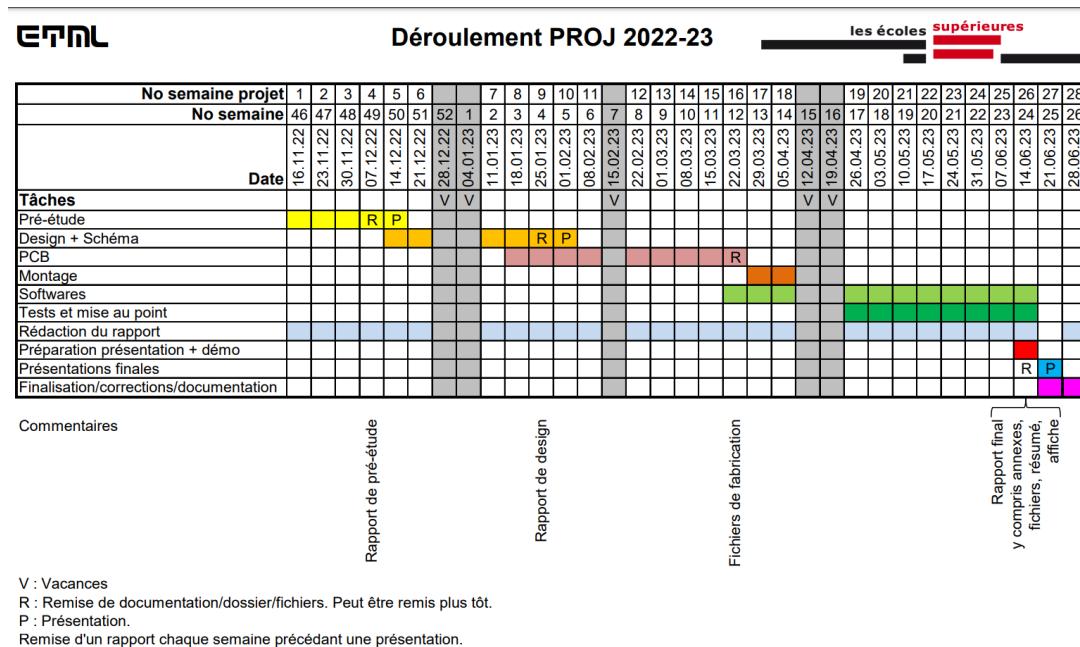


Figure 2 – Jalons principaux

## 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

L'objectif de cette pré-étude, est de se pencher sur le fonctionnement plus fondamental du système, faire des petits dimensionnements ainsi que de survoler différents aspects techniques liés au projet.

### 2.1 Fonctionnement du système

#### 2.1.1 Schéma bloc

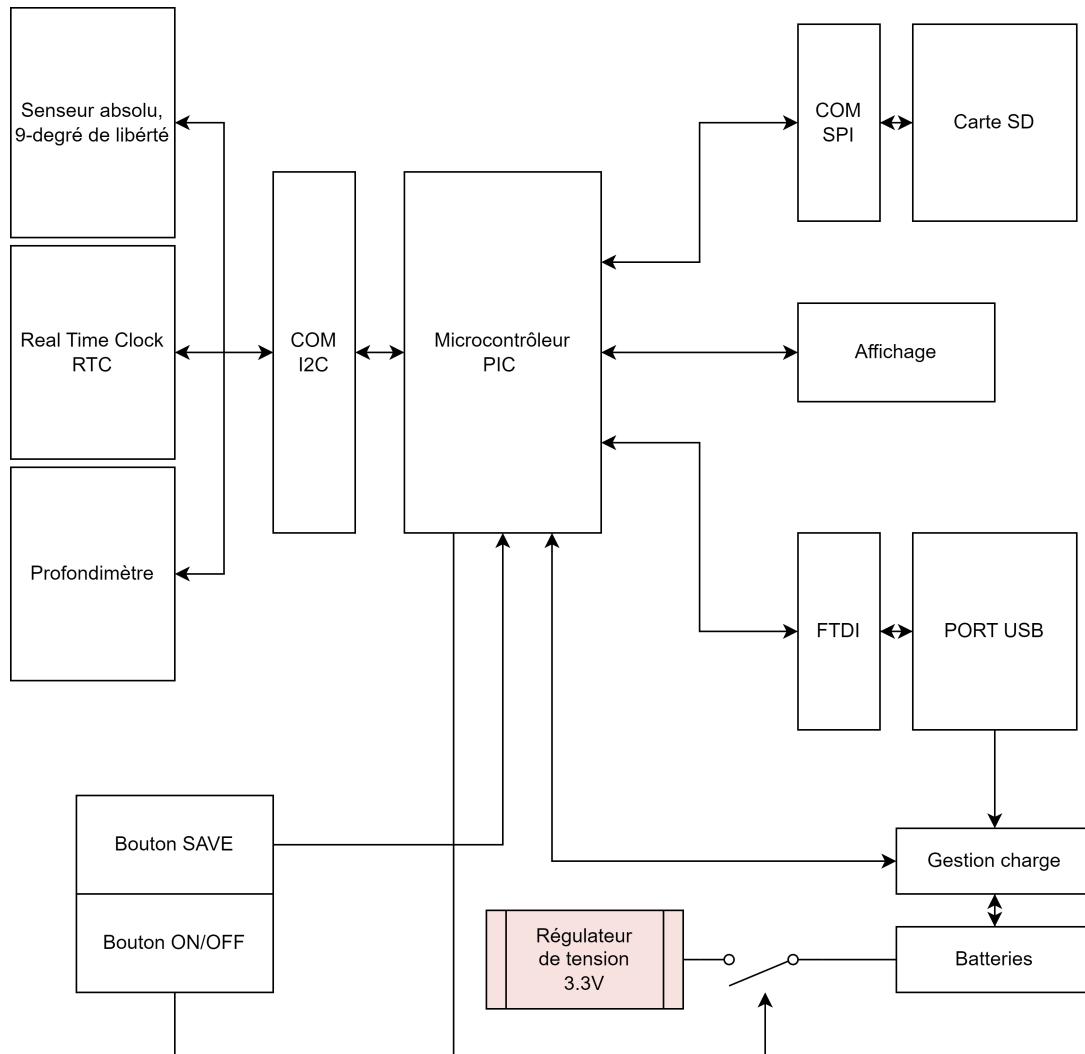


Figure 3 – Schéma bloc du module  
Source : Auteur

Capteurs : Les différents capteurs sont interfacés sur le même bus, et ont comme master le microcontrôleur en communication bidirectionnel, afin d'à la fois configurer les registres des périphériques et de lire leurs mesures.

Carte SD : La carte SD est interfacée en SPI et va contenir les données des différents capteurs ainsi que leurs éventuels flags d'importance (sauvegarde), sa taille sera dimensionnée ultérieurement.

Port USB & charge : Un port USB est présent, afin charger les batteries par un IC de gestion de charge connecté directement au 5V. De plus le port USB est communiquant avec le microcontrôleur par un driver FTDI, afin d'éventuellement ajouter un système de lecture de la carte SD, directement par USB. Ceci dans cette version ou une ultérieure. Le port USB pourrait aussi servir à fixer la référence de la RTC.

Bouton multifonction : Sachant qu'un bouton étanche est déjà présent sur le module, l'exploiter en tant que bouton multifonction est une solution ergonomique pour ne pas mettre en péril l'étanchéité globale. Ce bouton ferait office de ON/OFF et de "sauvegarde" de point d'intérêt. Pour se faire, le bouton contrôlerait par un transistor de commutation l'alimentation du système, puis lors de l'allumage du microcontrôleur, le MCU prendrait la relève en maintenant le système allumé à son tour, permettant ainsi de lire le bouton et de sur une pression longue déconnecter l'alimentation.

Affichage : L'affichage permettra de visualiser différentes données, dont les plus importantes tel que la pression ou le statut de la batterie.

La forme de l'affichage est encore à définir selon la mécanique du module, mais le plus élégant, serait l'utilisation d'un petit écran OLED.

Capteur de pression : Le capteur de pression devra avoir un contact direct avec l'eau, cela impliquera de la mécanique et de la gestion d'étanchéité. Une autre possibilité aurait été de mesurer optiquement la déformation du boîtier pour en déduire la pression, mais la complexité est trop importante.

## 2.2 Choix des composants importants

### 2.2.1 Senseur absolu

Pour le senseur absolu, il existe des IC permettant directement de faire la fusion des senseurs (Accéléromètre, gyroscope, magnétomètre et thermomètre), ce qui épargne toute une phase de calcul chronophage, en permettant directement de lire les quaternion, angles de Euler, vecteurs de rotations, cap de direction etc... directement sur le composant. Il existe différents IC dont deux ce sont montrés très intéressants, le BNO85 et le BNO55, les deux étant PIN-Compatible, j'ai décidé d'opter pour le BNO055<sup>1</sup>.



Figure 4 – Schéma bloc du module

Source : <https://www.mouser.ch/new/bosch/bosch-bno55-sensor/>

Sachant que la brazure de ce type de boîtier est compliquée et également dans un but de simplification du projet, j'ai décidé d'utiliser les cartes d'évaluation d'adafruit N° : 4646 qui ont des connections bergs ainsi que tous les composants externes passifs déjà montés.

#### Caractéristiques importantes :

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

Nous allons par la suite voir sur la figure 5, quelles données du BNO055 sont disponibles ainsi que leurs tailles mémoires.

---

1. K:/ES/PROJETS/SLO/2221\_LocalisationSousMarine/doc/composants/9DOF-BNO055

Table 3-36: Temperature Data

Parameter	Data type	bytes
TEMP	signed	1

Table 3-34: Gravity Vector Data

Parameter	Data type	bytes
GRV_Data_X	signed	2
GRV_Data_Y	signed	2
GRV_Data_Z	signed	2

Table 3-32: Linear Acceleration Data

Parameter	Data type	bytes
LIA_Data_X	signed	2
LIA_Data_Y	signed	2
LIA_Data_Z	signed	2

Table 3-30: Compensated orientation data in quaternion format

Parameter	Data type	bytes
QUA_Data_w	Signed	2
QUA_Data_x	Signed	2
QUA_Data_y	Signed	2
QUA_Data_z	Signed	2

Table 3-28: Compensated orientation data in Euler angles format

Parameter	Data type	bytes
EUL_Heading	Signed	2
EUL_Roll	Signed	2
EUL_Pitch	Signed	2

Table 3-27: Yaw rate data

Parameter	Data type	bytes
Gyr_Data_X	signed	2
Gyr_Data_Y	signed	2
Gyr_Data_Z	signed	2

Table 3-26: Magnetic field strength data

Parameter	Data type	bytes
Mag_Data_X	signed	2
Mag_Data_Y	signed	2
Mag_Data_Z	signed	2

Table 3-25: Acceleration data

Parameter	Data type	bytes
Accel_Data_X	signed	2
Accel_Data_Y	signed	2

Figure 5 – Donnée de sortie de l'IC (43 bytes)

Source : [https://cdn-shop.adafruit.com/datasheets/BST\\_BNO055\\_DS000\\_12.pdf](https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf)

## 2.2.2 Capteur de pression

Pour le capteur de pression, une modification mécanique du boîtier sera très probablement nécessaire. J'ai pu trouver un capteur correspondant aux caractéristiques demandées du projet, celui-ci est plutôt générique et peut communiquer en I2C :  
PTE7300-14AN-1B016BN



Figure 6 – Illustration capteur de pression  
Source : Distrelec, PTE7300-14AN-1B016BN

L'avantage avec le capteur ci-dessus est le système hermétique pour le trou, un autre capteur peut être utilisé lors de l'étude, néanmoins la modification mécanique étant probablement inévitable, le système de vissage de la figure 6 est intéressant.

## 2.2.3 Affichage

Pour l'affichage, je vais essayer d'opter pour un petit afficheur OLED, en gardant la possibilité en cas de complication lors de l'étude, l'utilisation de simples LEDS d'indications.

Il existe plusieurs affichages OLED ronds petits formats, sur lesquels je me pencherais plus en détail lors de l'étude.

### 2.2.4 Carte SD

Taille mémoire : Afin de dimensionner la taille de stockage de la carte SD, il faut utiliser les différentes caractéristiques du projet. Normalement la taille de la carte SD n'est clairement pas un problème, sachant que seulement du texte est enregistré et que les tailles mémoires disponibles peuvent être très élevées. Néanmoins il est intéressant de faire le dimensionnement pour connaître le minimum, et pour éventuellement adapter le projet avec d'autres systèmes de mémorisation.

Où :

$T_{rec}$	=	7200'000	[ms]	Temps à enregistrer
$T_{ech}$	=	100	[ms]	Temps d'un échantillon
$S_{mes}$	=	150	[bytes]	Taille de toutes les données de mesures
$S_{timestamp}$	=	$\sim 3$	[bytes]	Taille de l'information de temporalité
$S_{flag}$	=	1	[bytes]	Taille de l'indication d'importance

Nombre de mesure à effectuer :

$$Nb_{mesures} = \frac{T_{rec}}{T_{ech}} \quad (1)$$

D'après (1), nous avons un nombre de mesure de 72'000.

Taille minimum :

$$Taille_{min} = Nb_{mesures} * (S_{mes} + S_{timestamp} + S_{flag}) \quad (2)$$

D'après (2), la taille mémoire minimum doit être de  $\sim 11\text{MB}$ .

Nous pouvons donc constater que pour une utilisation standard de 2h, la mémoire occupée est très faible, d'où l'intérêt de sauvegarder dans la carte SD la date, afin de pouvoir faire plusieurs "expéditions" en "une fois", sans avoir à vider la carte.

## 2.2.5 Real Time Clock

L'objectif de la RTC, est de donner l'information de la temporalité de la mesure (timestamp), afin de lors du traitement des donnée avoir accès à ce paramètre.

Sachant que l'échantillonnage des mesures est de 100ms, la RTC devrait permettre cette résolution. Néanmoins une autre information importante, comme mentionnée lors de la section 2.2.4, est la date de la mesure, afin de permettre plusieurs expéditions par utilisation de la carte.

J'ai donc décidé d'utiliser une RTC pour l'heure grossière de départ (Année, date, heure, minute, seconde) et les compteur du MCU pour faire le delta entre chacune des mesures en ms.

La RTC devra pouvoir tenir le minimum de 2 heure d'utilisation, à cette fin, la batterie LI-ION déjà présente sera suffisante.

La RTC devra avoir une faible consommation, le calendrier ainsi qu'une bonne précision. A cette fin, la RTC S-35390A-T8T1G est assez générique et possède une bonne documentation.

## 2.2.6 Microcontrôleur

Le microcontrôleur devra avoir un nombre suffisant de communications, sachant que beaucoup sont présentes dans le projet (I2C, SPI, UART...), ce qui signifie un nombre de pattes élevées.

Des calculs peuvent aussi être nécessaire, si il s'avère qu'il faille faire une traitement des données préliminaire, il faudrait donc opter pour un MCU 32bits si possible.

La famille PIC est celle standardisée par l'école supérieure, c'est donc pour cette famille-ci que je vais opter.

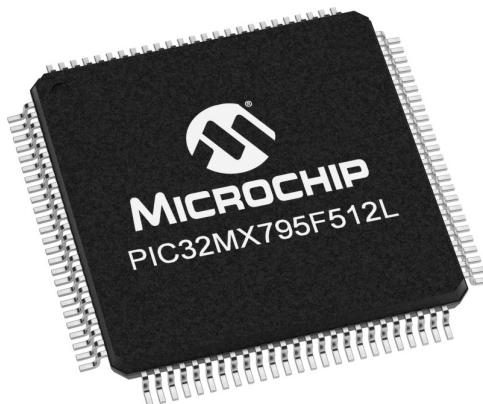


Figure 7 – Illustration du modèle MCU du kit ETML-ES

Source : <https://www.microchip.com/en-us/product/PIC32MX795F512L>

### 2.2.7 Batterie, charge et régulation

Pour la technologie de batterie, en utilisation sous-marine, j'ai trouvé ce tableau de comparaison :

Chemistry	Energy Density (Whr/kg)	Pressure Compensatable (Whr/kg)	Outgassing	Cycles	Comments
Alkaline	140	No	Possible, at higher temperatures	1	Inexpensive, easy to work with
Li Primary	375	No		1	Very high energy density
Lead Acid	31.5	Yes (46)	Yes, even with sealed cells	~100	Well established, easy to work with technology
Ni Cad	33	No	If overcharged	~100	Very flat discharge curves
Ni Zn	58.5	Possibly (160)	None	~500	Emerging Technology
Li Ion	144	No	None	~500	In wide use in small packs
Li Polymer	193	Possibly	None	~500	Only "credit card" form factor currently available
Silver zinc	100	No	Yes	-30	Can handle very high power spikes

Figure 8 – Comparaison des technologies de batteries  
Source : Power Systems for Autonomous Underwater Vehicles[1]

Pour des raisons de praticité et étant-donné la documentation plus importante, j'ai décidé d'utiliser la technologie LI-ION :

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

Table 1 – Tableau avantages/inconvénient LI-ION)

Malgré les risque dûs au contact de l'eau (Enflammement, éclatement...) la technologie LI-ION est souvent utilisée pour les application sous-marines dû a ses différents avantages, c'est pour cela que j'opterais pour cette technologie.

## 2.3 Estimation des coûts

Ici je vais me baser sur les composants que j'ai pu trouver et estimer le coût moyen de ceux-ci, c'est à titre purement indicatif, (les prix sont généralement estimés à la hausse).

Composant	Estimation
Profondimètre	70.-
Centrale inertielle	35.-
RTC	5.-
Microcontrôleur	5.-
Carte SD	20.-
Affichage OLED	45.-
FTDI	4.-
Batterie LI-ION	20.-
IC chargeur	4.-
Traco-power 3.3V	10.-
PCB	40.-
Total	258.-

L'estimation des prix étant plutôt élevée, des économies peuvent être très facilement réalisées, en changeant l'affichage OLED pour des LEDS ou en modifiant le PCB (Le simplifier ou changer de fournisseur (eurocircuit)).

## 2.4 Synthèse développement

J'ai pu lors de cette pré-étude, établir le fonctionnement global du système, choisir certaines technologies et composants importants, ainsi que pu procéder à certains dimensionnements utiles quant au futur développement. Par la suite, je vais affiner les différents éléments abordés lors de la pré-étude, effectuer le développement plus détaillé de chacun des blocs et réaliser la schématique du projet. Lors de la pré-étude, je n'ai pas eu accès au boîtier mécanique du projet, ce qui a restreint mon champs d'action lors de certains dimensionnement, tandis que pendant l'étude j'aurais accès à celui-ci, ce qui risque d'impacter/modifier certains aspect fixés lors des sections antérieures. Je suis très intéressé par le projet et me réjouis grandement de poursuivre son développement.

### 3 Développement schématique

#### 3.1 Scéma bloc détaillé

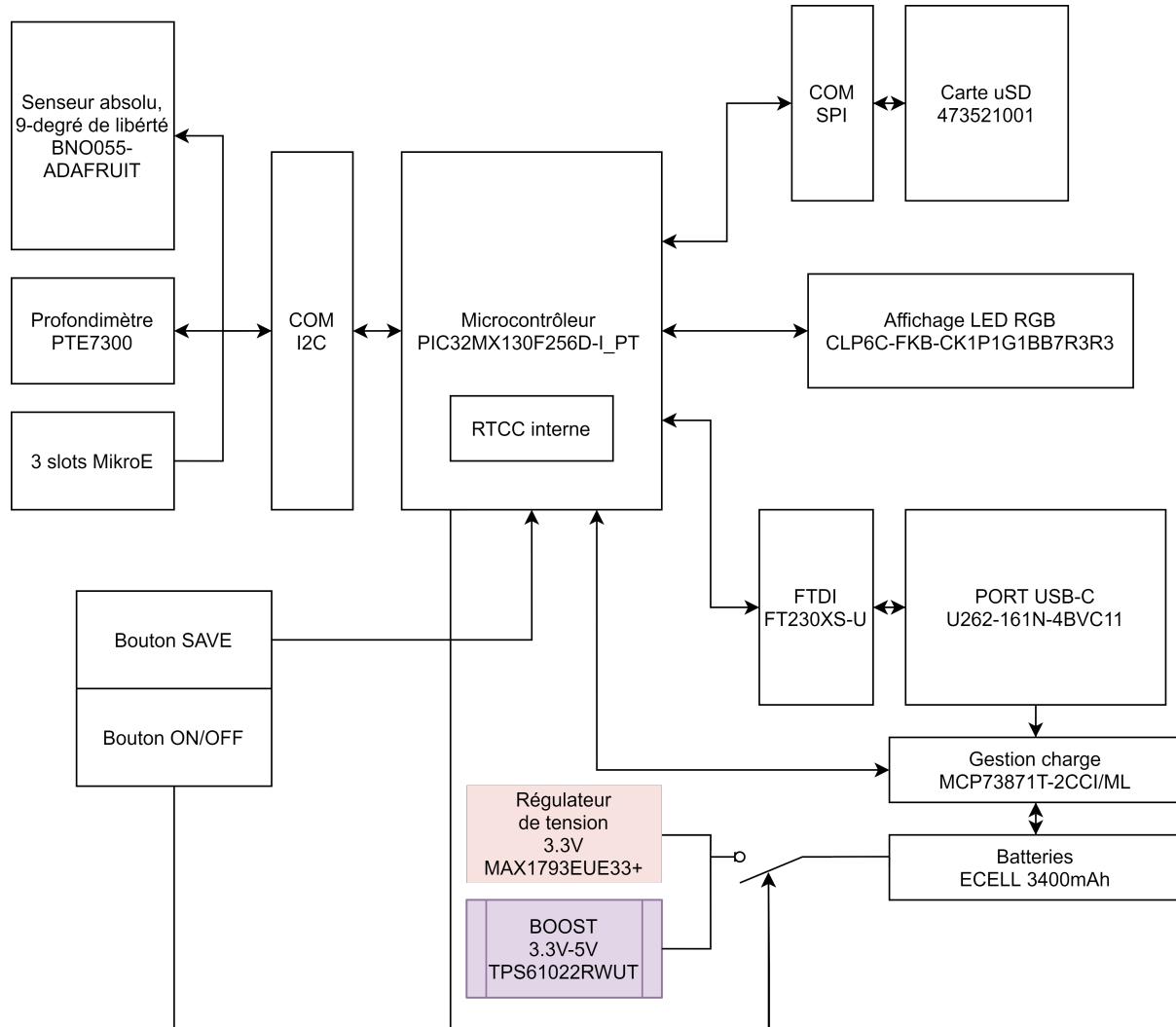


Figure 9 – Schéma bloc détaillé

Source : Auteur

## 3.2 Choix des composants

### 3.2.1 Microcontrôleur

Lors de la recherche de composants, j'ai décidé d'utiliser l'un des PIC32 standards de l'ES : PIC32MX130F064D-I/PT.

Device	Pins	Program Memory (KB) <sup>(1)</sup>	Data Memory (KB)	Remappable Peripherals					Analog Comparators	USB On-The-Go (OTG)	I <sup>2</sup> C	PMP	DMA Channels (Programmable/Dedicated)	CTMU	10-bit 1 Msps ADC (Channels)	RTCC	I/O Pins	JTAG	Packages
				Remappable Pins	Timers <sup>(2)</sup> /Capture/Compare	UART	SPI/I <sup>2</sup> S	External Interrupts <sup>(3)</sup>											
PIC32MX130F064D	44	64+3	16	32	5/5/5	2	2	5	3	N	2	Y	4/0	Y	13	Y	35	Y	VTLA, TQFP, QFN

Figure 10 – Périphériques disponibles du PIC

Source : PIC32MM0256GPM064 family datasheet

Nous pouvons constater sur la figure 10 que les critères minimaux de mon projet sont respectés :

1 - I2C

1 - SPI

1 - UART

1 - RTCC

### 3.3 Dimensionnements

#### 3.3.1 Vue d'ensemble schématique

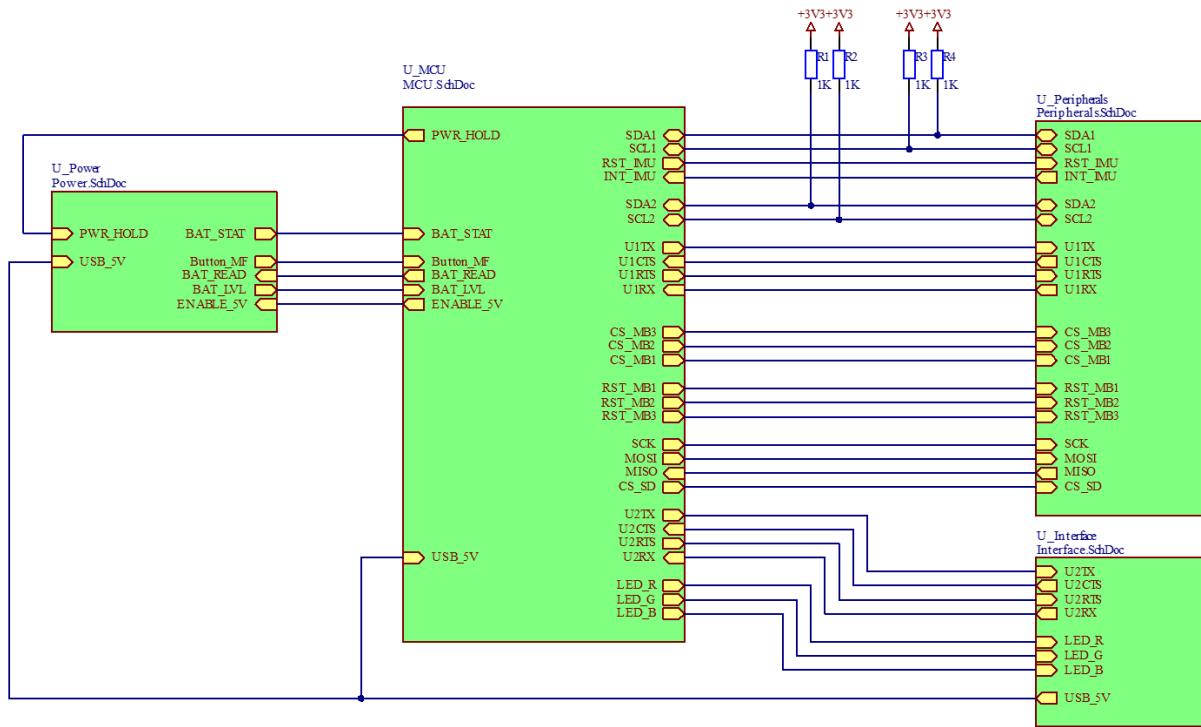


Figure 11 – Schéma bloc de la schématique

Source : Auteur

Nous pouvons constater sur la figure 11 la structure des différents blocs du schéma :

Bloc	Description
Power	Contient les différents régulateurs du système, ainsi que la gestion de charge de la batterie.
MCU	Contient l'intelligence du système, avec le microcontrôleur ainsi que tous ses composants passifs associés.
Peripherals	Périphériques du système : Carte-SD, Centrale inertielle, Capteur de pression, Slots MikroE.
Interface	Connecteur USB avec convertisseur serial (FTDI) et tous les composants passifs de sécurité. Interface LED RGB pour le statut.

### 3.3.2 Autonomie du système

Afin de proportionner la batterie du circuit, il a fallut dimensionner les différentes consommations des composants, ceci par le biais de leurs documentations :

MCU - 30mA	BNO055 - 12.3mA	Capt. Pression - 4mA	Carte-SD - 100mA
MikroE - ??mA	Régulateurs - 40uA	LED RGB - 25mA	

Nous pouvons constater que la plus grande consommation vient de la carte micro-SD, qui au maximum peut induire 100mA.<sup>2</sup>

Afin d'obtenir une autonomie d'au moins 2h (selon CDC), il faudrait une capacité de :

$$Capacite = Consommation_{tot} * Temps \quad (3)$$

Ce qui nous fait une capacité de  $\sim 342.68\text{mAh}$ , valeur facilement atteignable par les batteries li-ion du marché. Étant-donné que différents projets utilisaient des batteries 3400mAh, dans un objectif de conformité et de simplification des commandes, j'ai choisis cette même valeur. Ce qui signifie une autonomie de  $\sim 20$  heures, sans compter les différents mécanismes d'économie d'énergie.

C'est un temps largement suffisant pour la durée de plusieurs expéditions, néanmoins la RTCC du microcontrôleur requiert d'être alimentée en permanence, j'ai donc décidé de déterminer un fonctionnement, où lorsque l'on charge la batterie la date se mettrait à jour et le mode "éteint" serait juste un mode de veille qui attendrait un niveau positif sur le switch avant de commencer le logging avec un timestamp principale contenant la date, puis, seulement des deltas entre les mesures. Un diagramme des états est présents à la figure 12.

---

2. Datasheet SanDisk

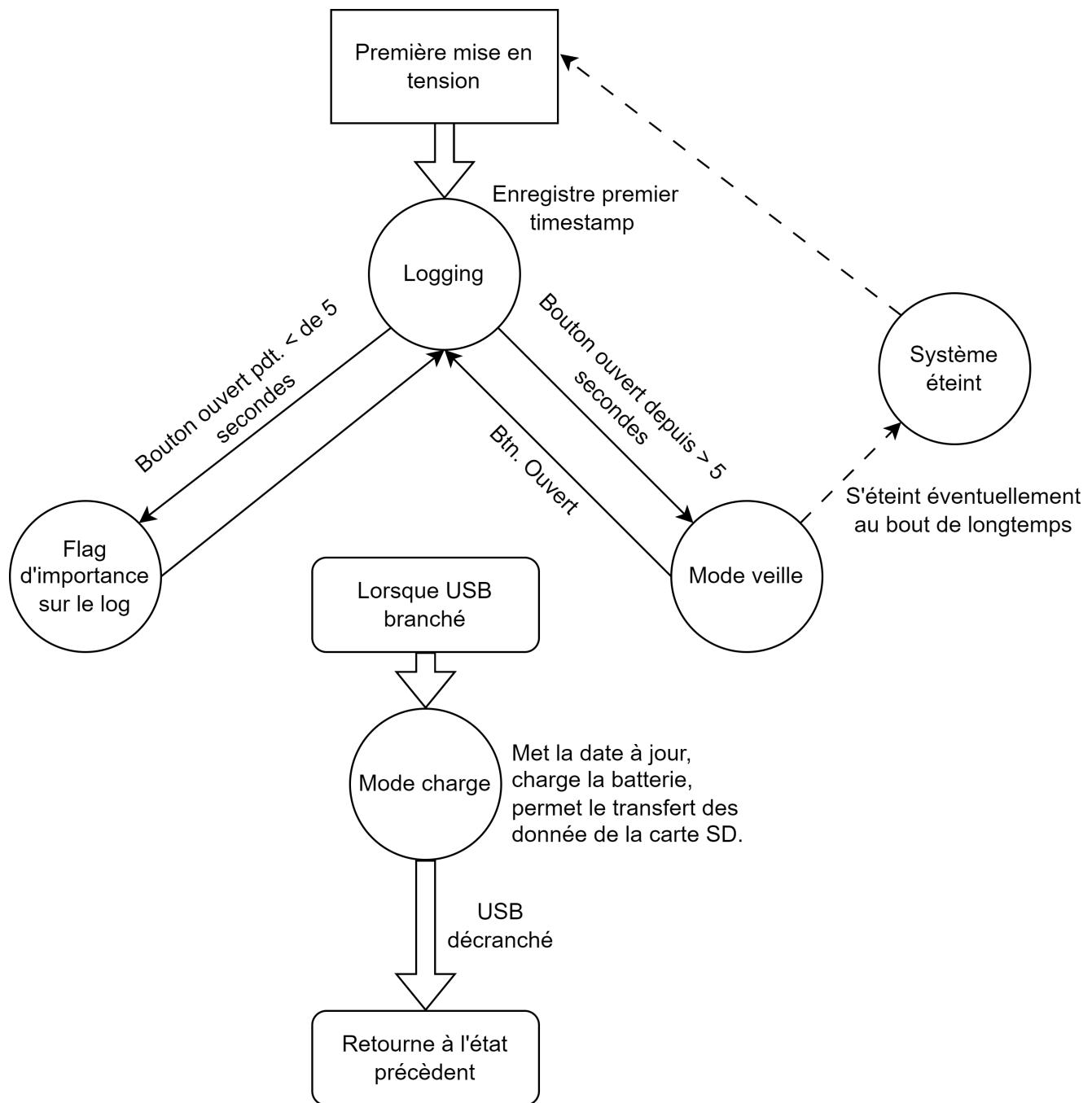


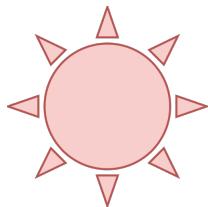
Figure 12 – Diagramme des états du système  
Source : Auteur

### 3.3.3 LED Interface

Afin d'informer l'utilisateur de ce qu'il se passe dans le système, j'ai décidé d'implémenter en tant qu'interface, une led RGB. Celle-ci sera un minimum puissante, afin de pouvoir être lisible lors de l'utilisation sous-marine du module.

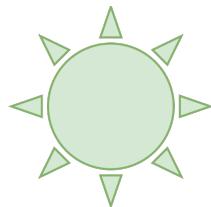
La consommation de la led RGB étant relativement importante, des mécanismes d'économie d'énergie seront mis en place dans le développement firmware.

Diagramme d'interface :



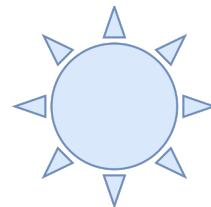
#### Rouge

- Constamment allumé : Indique que la batterie doit être chargée.
- Clignote rapidement ; Indique le passage en mode veille ou le passage en mode éteint.
- Clignote lentement: Indique que la carte SD est pleine.



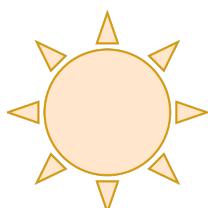
#### Vert

- Clignote lentement : Indique que le logging est en cours.
- Clignote rapidement : Indique que la charge est en cours
- Allumé et carte branché : Indique que la charge est complète.



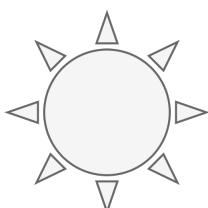
#### Bleu

- S'allume : Indique que le flag d'importance a été enregistré.
- Clignote rapidement : Indique qu'un transfert de fichier est en cours.



#### Orange

- Indique une erreur autre.



#### Eteint

- Constamment éteint : Indique que l'appareil est en veille ou entièrement éteint.

Figure 13 – Définitions des états de la LED RGB

Source : Auteur

### 3.3.4 Adaptation mécanique

L'idée étant d'obtenir une mesure de pression sans modification mécaniques sur le boîtier originale, plusieurs idées ont émergées :

- 1) Mesurer une déformation mécanique a-même le module, dans le but de déduire la pression (Développement d'un capteur).
- 2) Ajout d'une rallonge cylindrique au module, afin de fixer un capteur de pression à plat sur celui-ci, tout en permettant les modifications mécaniques sans altération du boîtier originale.

Par sa complexité moins importante et due aux contraintes de temps, la seconde option sera celle développée lors de cette version du projet. Voici des ébauches (Pas à l'échelle) du concept :

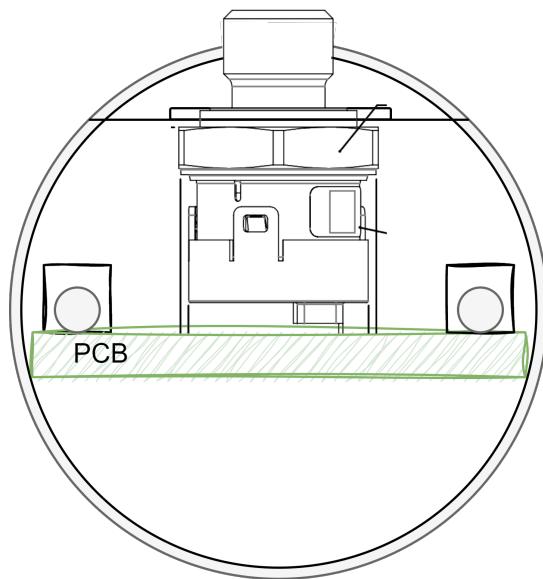


Figure 14 – Ébauche intérieur du cylindre

Source : Auteur

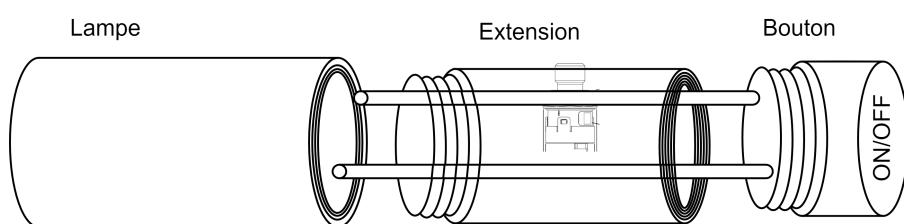


Figure 15 – Ébauche globale

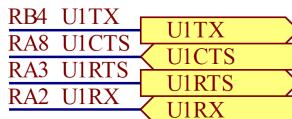
Source : Auteur

### 3.3.5 Bus de communications

UART (1) :

Utilisation : Communication avec les boards MikroE, pour les clicks-boards utilisant la comm. série.

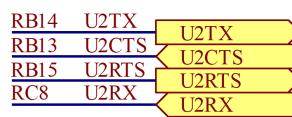
Pinning :



UART (2) :

Utilisation : Communication avec FTDI conversion USB-Serial. Transfert des fichiers de la carte-SD et mise-à-jour de la RTCC.

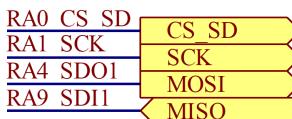
Pinning :



SPI :

Utilisation : Communication avec la carte micro-SD, écriture des mesures, timestamps et flag d'importance.

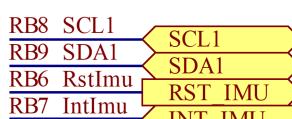
Pinning :



I2C (1) :

Utilisation : Lecture des mesure de la centrale inertuelle BNO055 et paramétrage des registres de celui-ci.

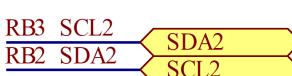
Pinning :



I2C (2) :

Utilisation : Lecture des données du capteur de pression et est également connecté aux slots MikroE, pour permettre à ceux-ci de communiquer via I2C.

Pinning :



Pour ce qui est des mesures sur ces différents bus de communications, des connecteurs bergs ont été prévus, afin de pouvoir connecter facilement un analyseur logique sur les différentes trames :

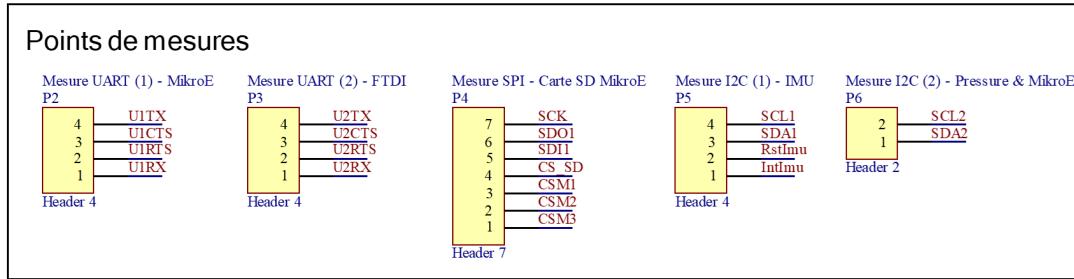


Figure 16 – Connecteurs pour analyseur logique

Une contrainte s'est créée, lorsque toutes les pins du microcontrôleur ont été allouée alors qu'il restait des connexion pour les "Chip select" et "Reset" des carte click-board Mikroe. Afin de remédier à ce problème, j'ai décidé d'utiliser un démultiplexeur, sachant que chacune des ces PINS, peuvent être activée seulement une à une :

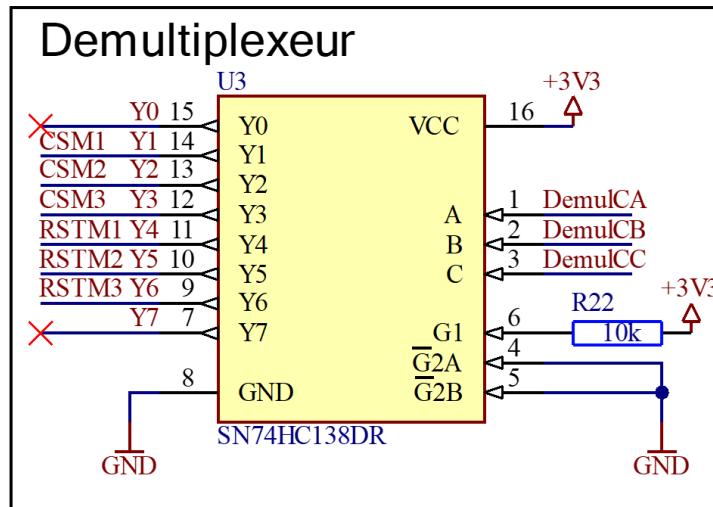


Figure 17 – Demultiplexeur

### 3.3.6 Pérophériques

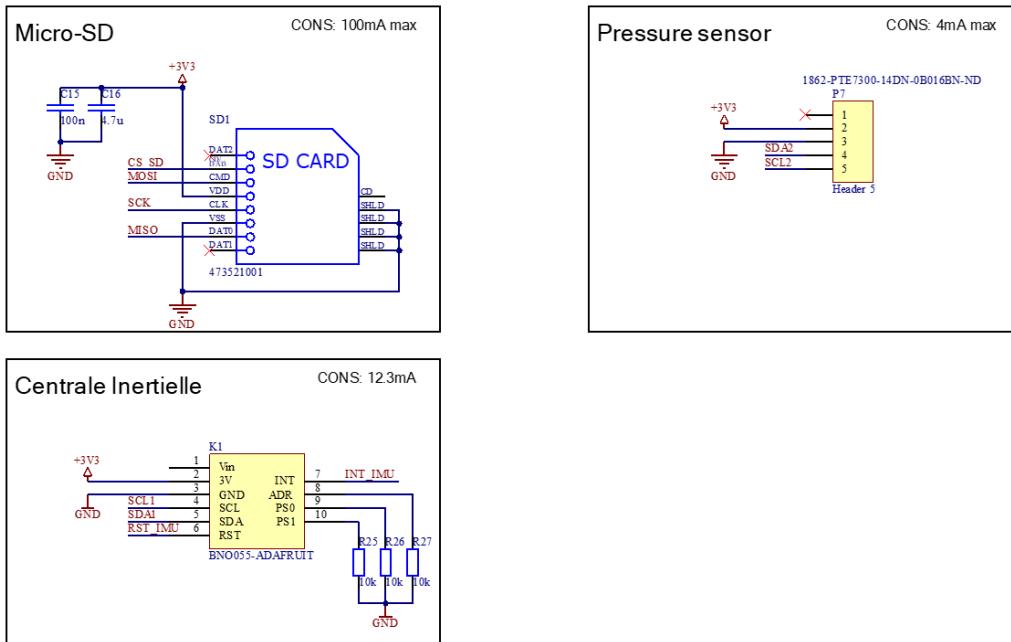


Figure 18 – Carte-SD, capteur de pression et centrale inertie.

Pour la carte-SD, certaines pins ne sont pas utilisée car pas utile dans le cadre du projet (Ex : Pin CD (Card Detect)). Pour le capteur de pression, il s'agit d'un simple connecteur (AMTEK 5H2001N0-105PXBL00A01) qui vas venir connecter le senseur rattaché à l'extension mécanique. Quant à la centrale inertie (BNO055 adafruit-board) le bit d'adresse supplémentaire est mis à 0.

### Slots MikroE

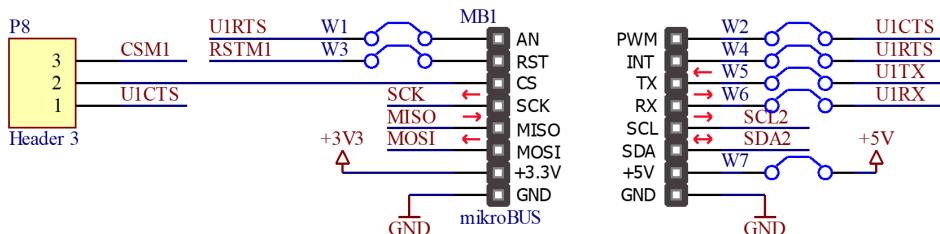


Figure 19

Les slots MikroE (3x figure 19) sont prévus pour pouvoir utiliser le plus de bus de communication possibles. Les jumpers sont prévus pour éviter les collisions sur les lignes.

### 3.3.7 Chargeur de batterie

Ici je vais me pencher sur les dimensionnement des 3 résistances PROG du composant de régulation et de charge d'accu, les autres composants passifs n'ont pas requis de dimensionnements particuliers.

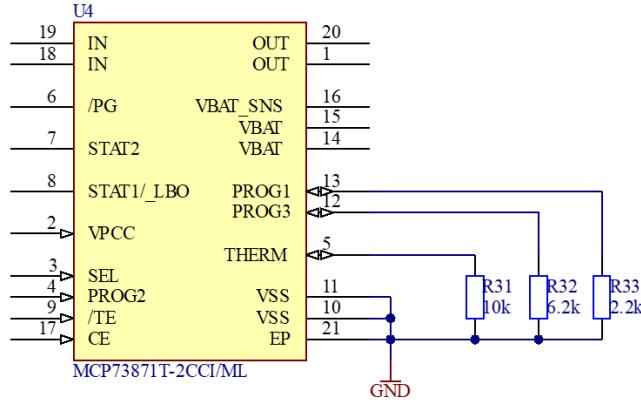


Figure 20 – IC régulation et gestion charge de l'accu

Afin de déterminer le comportement de la charge via les différentes étapes de courants, quelques équations ont été utiles :

Où

$$C = 3400 \text{ mAh}$$

$$ratio_{term} = 0.05 \quad ratio_{chrg} = 0.1$$

$$I_{term} = C * ratio_{chrg} \quad (4)$$

D'après 4,  $I_{term} = 170 \text{ mA}$ .

$$R_{prog3} = \frac{1000V}{I_{term}} \quad (5)$$

D'après 6,  $R_{prog3} = 5k88\Omega$  E12  $\Rightarrow 6k2\Omega$  (Arrondis au supérieur pour limiter courant de terminaison).

$$R_{prog1} = \frac{1000V}{C * ratio_{chrg}} \quad (6)$$

D'après 6,  $R_{prog1} = 2k94\Omega$  E12  $\Rightarrow 2k2\Omega$  (Arrondis à l'inférieur pour limiter courant de charge).

### 3.3.8 Synthèse et perspectives de l'étude

Lors du développement de la schématique, je n'ai pas eu de grands dimensionnements à faire mais plutôt dû mettre en place des mécanismes permettant la communication avec tous les senseurs et périphériques du système. J'ai essayé d'être le plus explicite possible lors de la création des différents blocs du schéma électrique, pour permettre une compréhension aisée de celui-ci. J'ai pu faire un contrôle mutuel de la schématique avec mon collègue M. Meven Ricchieri. Je n'ai pas rencontré de problèmes particuliers, j'ai pu compléter la schématique, avancer sur le concept globale et je suis très enthousiaste de continuer ce projet. Désormais il va falloir préparer la création du PCB, en contrôlant les footprints du circuit et développer davantage l'aspect mécanique du projet. La schématique issue de cette partie développement, sera disponible en annexe de ce rapport.

## 4 Développement du PCB

### 4.1 Bill of materials

La BOM complète est disponible dans les répertoires du projet, voici un extrait des prix des composants importants :

Composant	Prix/unité
C0805C106K8PACT	0,61
BAS40	0,14
150080SS75000	0,19
BNO055-ADAFRUIT	27,5
742792133 (ferrite bead)	0,24
SRN6045-1R0Y (Inducteur de puissance)	0,57
BSS138LT1G (Power Mosfet)	0,41
NVR1P02T1G (Power Mosfet)	0,43
473521001 (Molex connector)	4,34
FT230XS-U (FTDI)	2,08
PIC32MX130F256DI_PT	4,11
SN74HC138DR (demultiplexeur)	0,42
MCP73871T-2CCI/ML (Chargeur)	2,23
MAX1793EUE33+ (Regulateur 3,3V)	4,02
TPS61022RWUT (Boost)	2,07
U262-161N-4BVC11 (USB-C)	0,39
<u>Total</u>	49,75

Figure 21 – Prix des composants

## 4.2 Mécanique du projet

Afin d'installer le PCB dans le boîtier de la lampe, les deux tiges internes du boîtier peuvent avec des attaches servir de support. Pour se faire, imprimer une pièce à visser en 3d ou en utilisant simplement des brides, pourrait permettre de maintenir la carte dans le boîtier. Il faudra donc mettre des trous dans le PCB pour permettre des visées ou des brides.

### 4.2.1 Considérations mécaniques

Tige conductrice : Sachant que les tiges de support de la lampe sont conductrices, il faut donc prévoir une zone sans composant, sans cuivre apparent et si possible sans pistes sur les bords de la couche bottom.

Carte SD : La carte SD requiert un support relativement grand et un espace doit être prévu pour pouvoir insérer/retirer la carte facilement et sans qu'elle dépasse trop du PCB.

Centrale inertie : La centrale inertie, se connecte via des berges femelles et va par conséquent prendre de la place en hauteur, ce qui doit être considéré.

Slot MIKROE : Un slot mikroE est présent dans le projet et pour être implémenté, va requérir un allongement mécanique du bouton de la lampe, pour gagner de la place. Cette pièce doit être produite et usinée, car elle requiert d'être étanche.

LED RGB : La LED en bout du PCB peut exploiter le réfracteur déjà présent de la lampe.

Connecteur USB : Pour charger l'appareil, un port USB devrait être disponible au bord du PCB.

### 4.3 Placement des composants

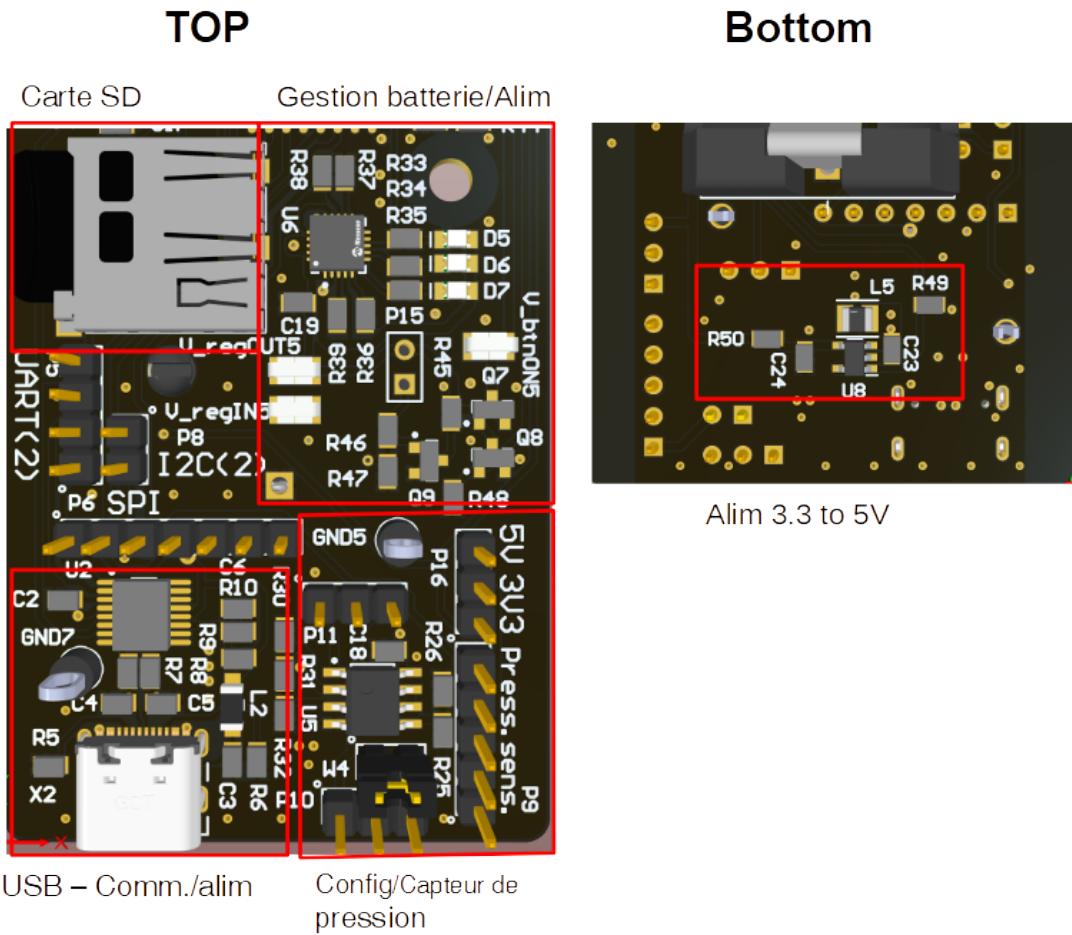


Figure 22 – Placement alimentations

Alimentation : J'ai décidé de placer en bas du PCB les parties d'alimentations du projet. Cette zone comporte :

Composant	Détail	Justification
USB	FTDI, piste 5V, pistes différentielles	Bord du PCB pour cablage ergonomique
Capteur de pression	Connecteur, Jumper choix alim, Choix mode (courant,tension,i2c)	Capteur de pression proche du bord branchement plus simple
Carte SD	Condensateurs de découplages, pistes SPI	Plus grand consommateur
Gestion batterie	Régulateur de charge, bouton ON/OFF, régulateur 3.3V	Zone dédiée, proche de la batterie
Boost 5V (bottom)	Circuit de boost 3.3-5V	Isolé, loin des petits signaux (bruit). Proche du capteur de pression (5V)

Table 2 – Composants de la zone et justification

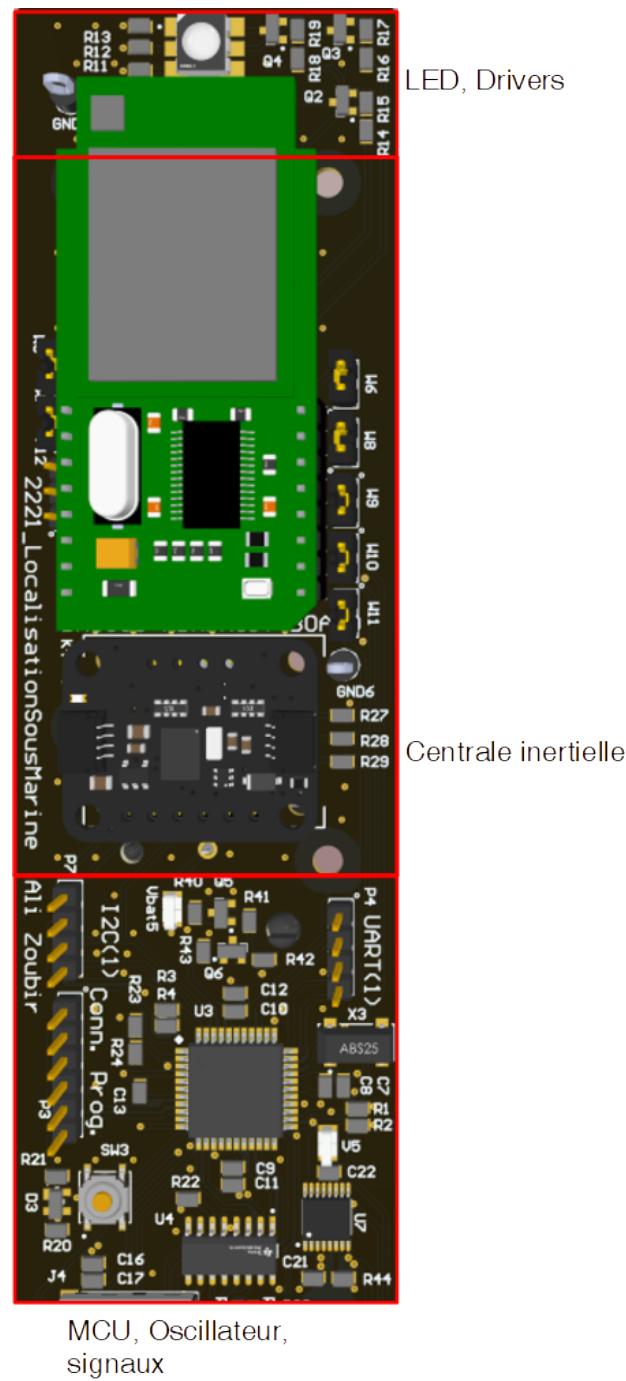


Figure 23 – Placement signaux

Le microcontrôleur est plus ou moins centrée par rapport aux différents périphériques, afin de minimiser la longueur des pistes des petits signaux. Il y a un bouton de reset, un multiplexeur et un oscillateur externe, au plus proche du MCU.

#### 4.4 Mécanique du PCB

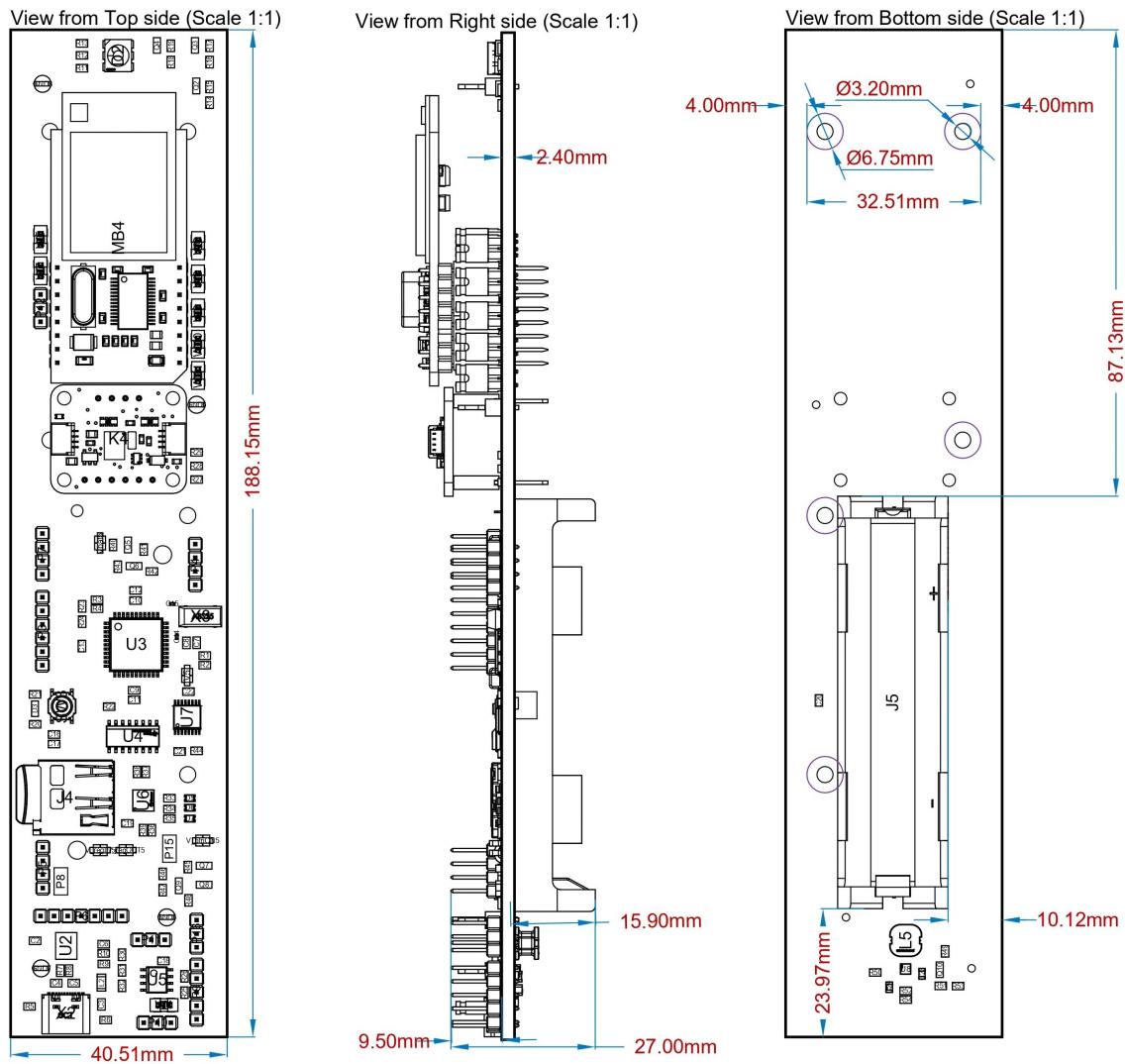


Figure 24 – Plan mécanique du PCB

La conception du PCB comprend des dimensions spécifiques, avec une longueur de 188.15 mm, une largeur de 40.51 mm et une hauteur de 27 mm. Pour faciliter l'installation et la fixation, cinq trous M3 sont répartis le long du PCB. Cependant, la présence du socle de la pile, qui a une hauteur de 15.9 mm, peut poser un problème, car il peut entraver le placement du PCB. Afin de résoudre cette contrainte, une décision a été prise de positionner le socle de la pile au centre du PCB. Cette disposition permet de mieux répartir l'espace disponible et d'éviter que le socle de la pile ne perturbe les autres contraintes mécaniques. En examinant une représentation en 3D de la carte, on peut constater que la carte SD simulée s'intègre parfaitement à la surface du PCB et ne dépasse pas ses dimensions.

## 4.5 Routage

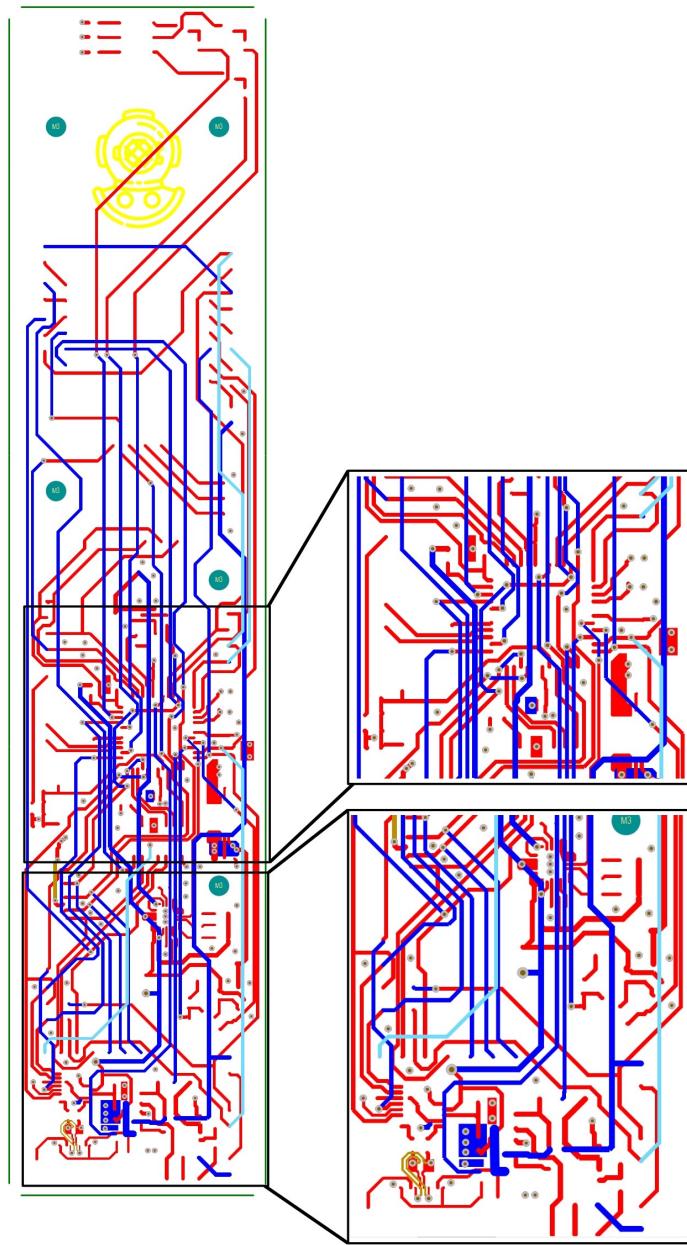


Figure 25 – Pistes du PCB

Routage sur 4 couches, chacune possède une orientation de piste : Top-Horizontal, Bottom-Vertical, Layer1-Vertical, Layer2-Vertical. La zone la plus denses est celle du microcontrôleur, c'est aussi où il y a le plus de vias. On peut voir que les pistes du bas sur la figure 25 sont plus épaisses, c'est parce que ce sont les pistes d'alimentations principales. Le 3.3V a une arborescence en arbre, avec le tronc qui traverse le PCB et les branches qui vont alimenter les différents systèmes, en passant bien d'abord par les condensateur de découplages de chacun.

Piste différentiel : L'USB possède une piste différentiel Data+/Data-, on peut la voir en bas de la figure 25 (Couche brune).

## 5 Développement firmware

Dans cette section, nous allons décrire et expliquer le procédé de programmation du code qui a été implémenté dans le microcontrôleur PIC32MX130F256D. Le processus de programmation du code dans le PIC32MX130F256D implique plusieurs étapes. Tout d'abord, il est nécessaire de disposer d'un environnement de développement intégré (IDE) adapté à ce microcontrôleur, ici, MPLAB X IDE, avec l'environnement Harmony permettant l'utilisation d'un configurateur graphiques pour les différentes libraires du PIC.

### 5.1 Configuration des PINs dans Harmony

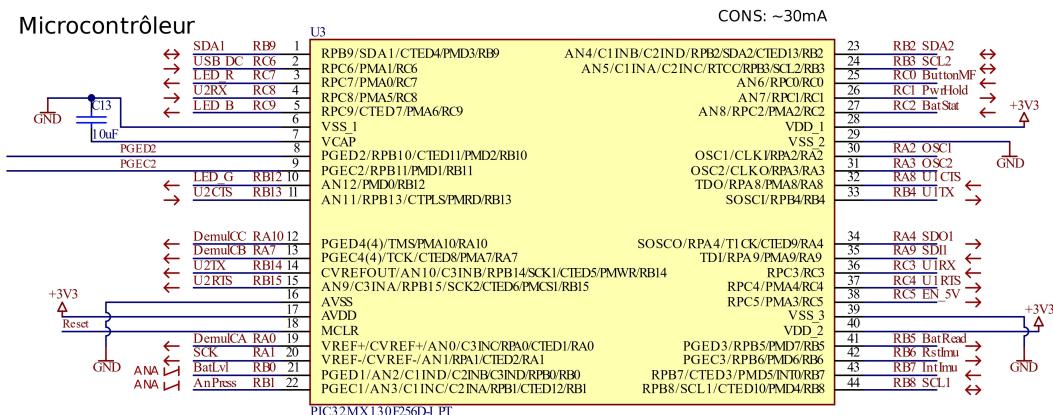


Figure 26 – Pinning réelles dans Altium designer

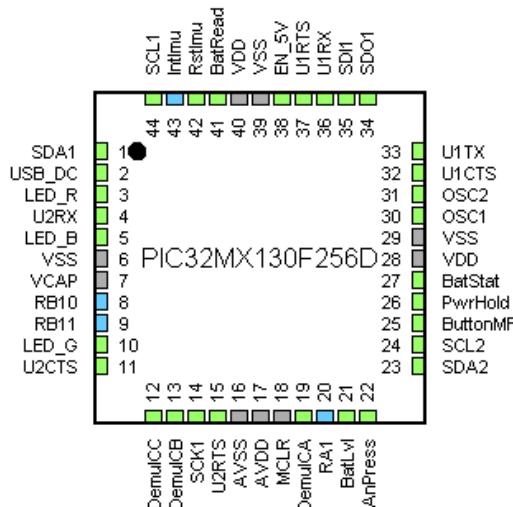


Figure 27 – Pinning dans Harmony

On peut voir que la PIN 20 (SCK) est en haute impédance dans harmony, tandis que la PIN14 qui était supposée être U2TX est devenue SCK, ceci étant dû à une erreur : SCK n'est que valable sur la PIN14, il a donc fallut ajouter un fil extérieur pour router Pin14 à Pin20, sacrifiant ainsi la communication USB sur l'UART2, cette modification sera décrite ultérieurement.

## 5.2 Configuration des périphériques dans Harmony

### 5.2.1 Timers

Deux timers seront utilisés, l'un pour mesurer des attentes en ms et l'autre moins rapide, pour les diverses actions du programmes, avec une interruptions chaque 10ms.

Timer	Temps voulu
Timer 1	1ms
Timer 2	10ms

La clock du système a été accélérée à 48MHz, cela dans le but d'accélérer le système dans sa globalité, sachant qu'il y a beaucoup de communication qui nécessite des opérations rapides, que ce soit dans la préparation des buffers ou les différents calculs.

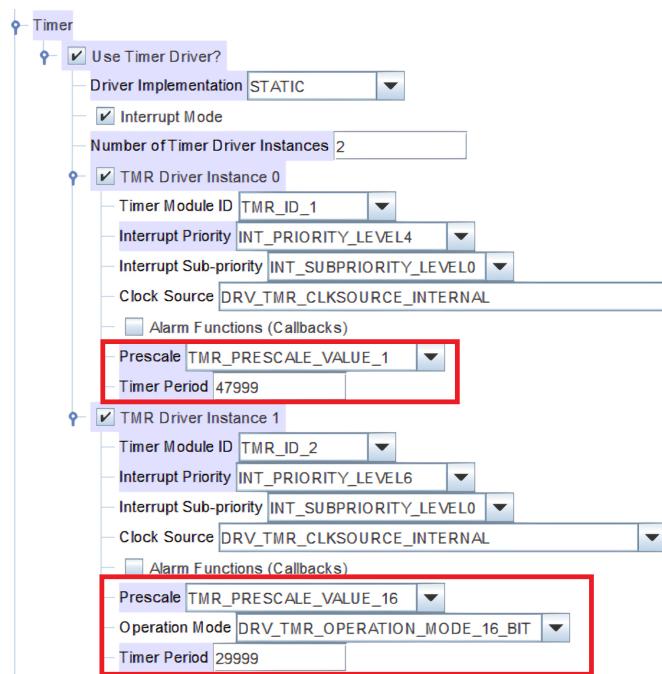


Figure 28 – Configuration dans harmony

Fréquence horloge [Hz]	Prescaler	Comptages
48e6	1.0	48,000.00
	2.0	24,000.00
	4.0	12,000.00
	8.0	6,000.00
	16.0	3,000.00
	32.0	1,500.00
	64.0	750.00
	128.0	375.00
	256.0	187.50

Fréquence horloge [Hz]	Prescaler	Comptages
48e6	8.0	60,000.00
	16.0	30,000.00
	32.0	15,000.00
	64.0	7,500.00
	128.0	3,750.00
	256.0	1,875.00

(a) Timer 1, Dimensionnement pour 1ms
(b) Timer 2, Dimensionnement pour 10ms

Figure 29 – Application timer développée par l'auteur

### 5.2.2 USART

J'ai décidé de configurer une communication série dans le but de vérifier mes données de mesures en temps réelles, simplifiant ainsi le debugage. Pour se faire, j'ai décidé d'utiliser le périphériques UART1 prévu à la base pour le slot mikroe. Je viendrais ensuite me connecter avec un module USB-to-TTL externe pour lire les données via Putty sur mon ordinateur portable.

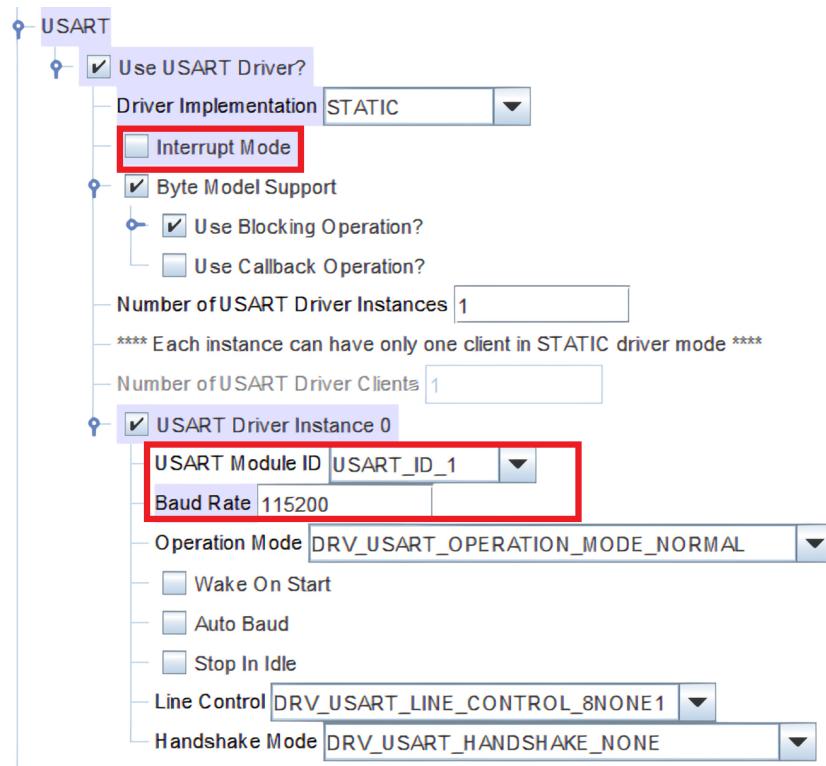


Figure 30 – Configuration UART

On peut constater sur la figure 30 que l'USART est configuré sans interruption a un bauderate de 115200.

### 5.2.3 Carte SD - SPI

J'ai optimisé l'utilisation du SPI en choisissant une fréquence de 5 MHz afin de minimiser le temps d'exécution sur le microcontrôleur, sachant que les opérations FAT nécessitent de nombreuses trames. Cependant, j'ai rencontré un problème lié à la clock du SPI. Étant donné la vitesse élevée et les modifications que j'ai dû effectuer, la clock interfère avec le FTDI inutilement, et un fil relie SCK et U2TX, créant ainsi une inductance parasite. Pour résoudre ce problème, j'ai dû ajouter un condensateur de 33pF entre SCK et GND, pour stabiliser la communication. Voir la configuration harmony de la carte SD sur la figure 31.

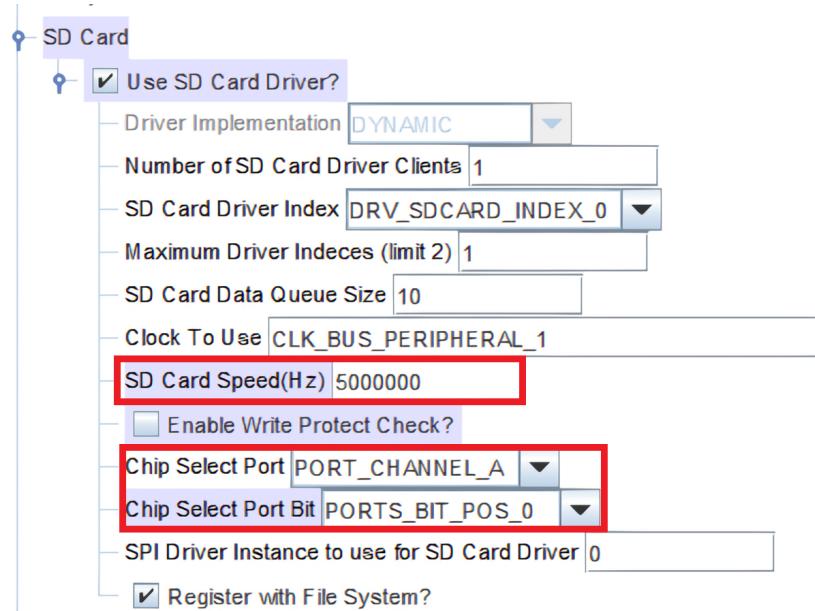


Figure 31 – Configuration du SPI

### 5.3 Code

Je vais dans cette section décrire le code du projet. Voici la hiérarchie des fichiers du projet :

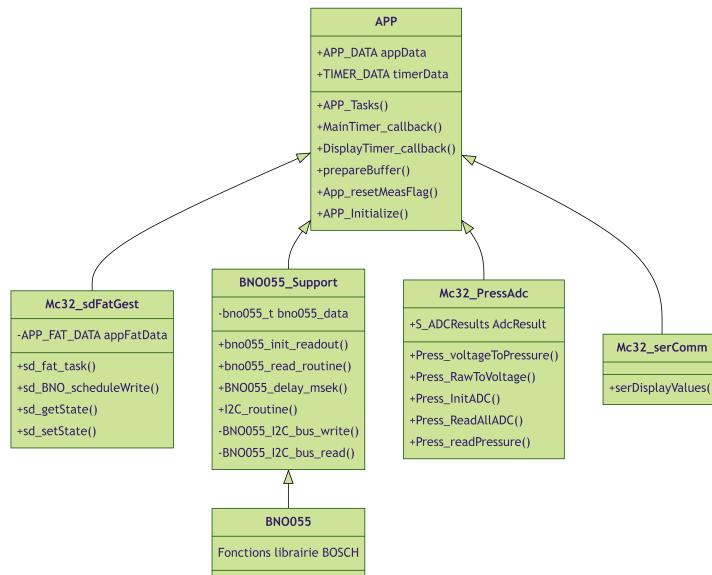


Figure 32 – Hiérarchie des fichiers du projet

### 5.3.1 Callbacks

Chacun des timers appellent dans leur interruption une fonction appartenante au fichier app.c qui contient les actions définies pour chaque lapse de temps fixés. On peut visualiser cela sur le diagramme de la figure 33.

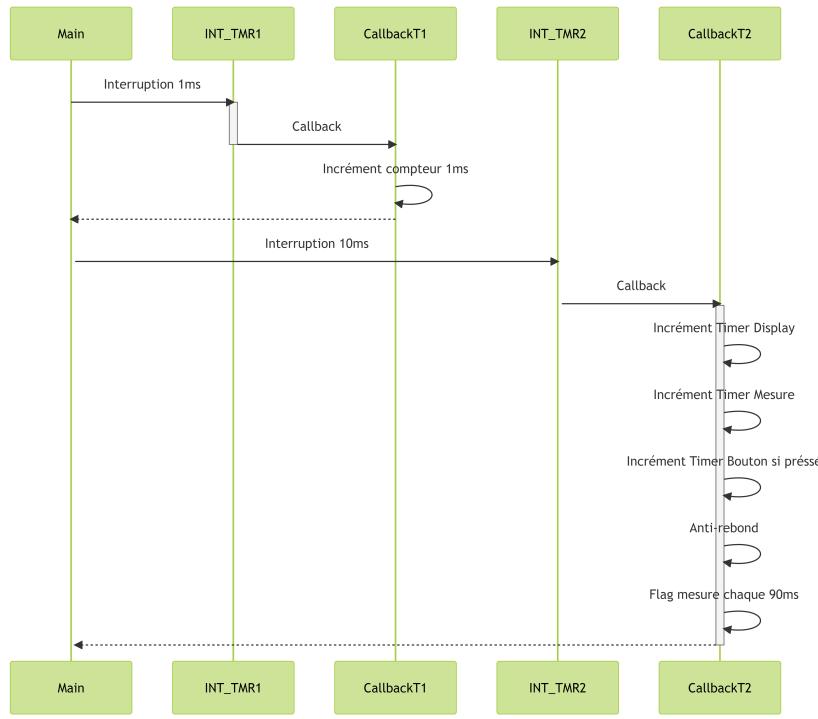


Figure 33 – Interactions des interruptions et des callbacks

Les callbacks en C offrent flexibilité, extensibilité et réutilisabilité. Ils permettent d'ajuster dynamiquement le comportement du programme, d'étendre les fonctionnalités et de réutiliser le code. Les callbacks favorisent également l'encapsulation et la personnalisation, améliorant ainsi la modularité et la maintenance du code.

### 5.3.2 Centrale inertielle BNO055

Pour ce qui est de la centrale inertielle BNO055, j'ai utilisé la librairie de BOSCH<sup>3</sup> ; La configurer pour du 32bits, créer les fonctions bas-niveau (i2c) et faire le liens avec la librairie BOSCH. Le tout dans le fichier BNO055\_support.c.

Pour faire le lien entre la librairie haut-niveau et bas-niveau, j'ai utilisé un pointeur de fonction présent dans la structure de donnée du BNO :

```
s8 I2C_routine( void )
{
    bno055.bus_write = BNO055_I2C_bus_write;
    bno055.bus_read = BNO055_I2C_bus_read;
    bno055.delay_msec = BNO055_delay_msek;
    bno055.dev_addr = BNO055_I2C_ADDR1;
    return BNO055_INIT_VALUE;
}
```

Listing 1 – Code lien pointeur de fonction

Voici le code une écriture sur le BNO055 par I2C :

```
s8 BNO055_I2C_bus_write( u8 dev_addr , u8 reg_addr , u8 *reg_data
, u8 cnt )
{
    s8 BNO055_iERROR = BNO055_INIT_VALUE;
    u8 array [I2C_BUFFER_LEN];
    u8 stringpos = BNO055_INIT_VALUE;
    array [BNO055_INIT_VALUE] = reg_addr;

    i2c_start();
    BNO055_iERROR = i2c_write( dev_addr<<1);

    for ( stringpos = BNO055_INIT_VALUE; stringpos < (cnt+
        BNO055_I2C_BUS_WRITE_ARRAY_INDEX); stringpos++)
    {
        BNO055_iERROR = i2c_write( array [stringpos] );
        array [ stringpos +
            BNO055_I2C_BUS_WRITE_ARRAY_INDEX] = *( reg_data + stringpos );
    }

    i2c_stop();
    if (BNO055_iERROR-1 != 0)
        BNO055_iERROR = -1;
    else
        BNO055_iERROR = 0;
    return (s8)(BNO055_iERROR);
}
```

Listing 2 – Code écriture au BNO055

---

3. Librairie du fabricant

Pour ce qui est de l'utilisation de la librairie haut-niveau bno055\_support, voici la préparation et la lecture des données :

```
/* BNO055 Read all important info routine */
bno055_local_data.comres = bno055_read_routine(&
    bno055_local_data);
/* Delta time */
bno055_local_data.d_time = timerData.TmrMeas - timerData.ltime
    ;
/* Pressure measure */
bno055_local_data.pressure = Press_readPressure();
/* Flag measure value */
bno055_local_data.flagImportantMeas = flagMeas;
```

Listing 3 – Code lecture des données par la librairie

### 5.3.3 Carte SD

La carte SD sa communication fonctionne sous forme d'une machine d'état non-bloquante, permettant ainsi de s'adapter aux situations de la carte sans bloquer le système pour autant.

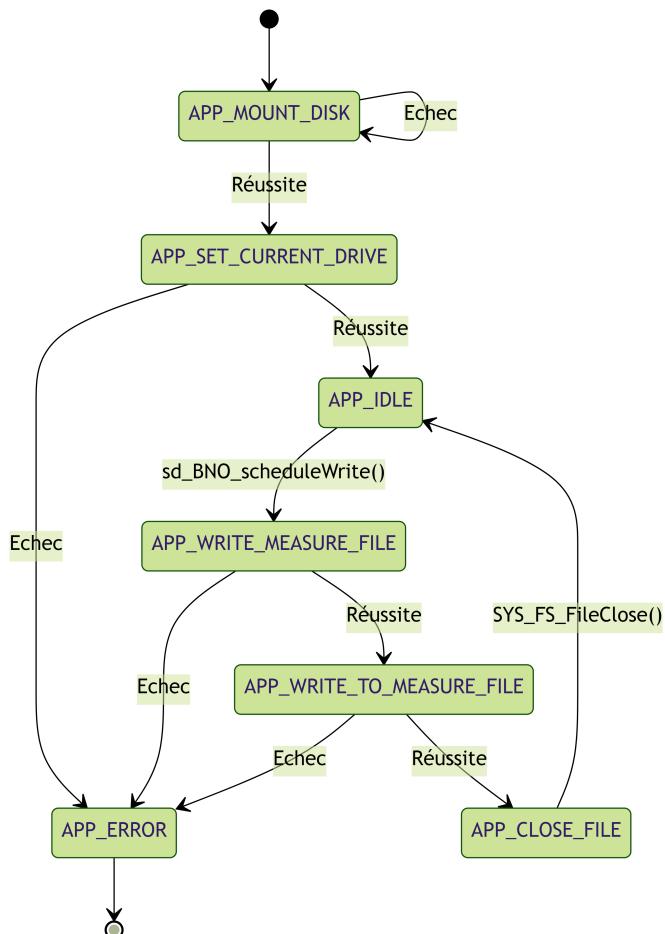


Figure 34 – Machine d'état de la carte SD

Planification d'une écriture Afin de lancer une écriture d'un set de mesure sur la carte SD, il faut utiliser la fonction sd\_BNO\_scheduleWrite() qui vas préparer le buffer d'écriture et modifier l'état de la carte SD :

```
/* Write measures to sdCard */  
sd_BNO_scheduleWrite(&bno055_local_data);
```

Listing 4 – Lancement d'une écriture sur la carte SD

## Références

- [1] A. Bradley, M. Feezor, H. Singh, and F. Yates Sorrell, “Power systems for autonomous underwater vehicles,” vol. 26, no. 4, pp. 526–538. Conference Name : IEEE Journal of Oceanic Engineering.
- [2] N. Shaukat, A. Ali, M. Javed Iqbal, M. Moinuddin, and P. Otero, “Multi-sensor fusion for underwater vehicle localization by augmentation of RBF neural network and error-state kalman filter,” vol. 21, no. 4, p. 1149. Number : 4 Publisher : Multidisciplinary Digital Publishing Institute.
- [3] A. S. Zaki, T. B. Straw, M. J. Obara, and P. A. Child, “High accuracy heading sensor for an underwater towed array.”