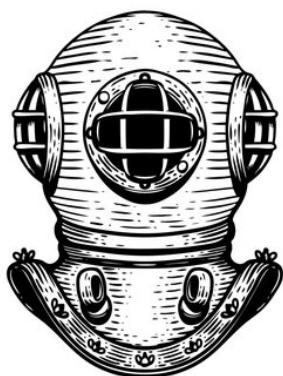


Localisation sous-marine

Système de logging pour déplacement de module
sous-marin.

Ali Zoubir



Rapport de projet



Génie électrique
École supérieure
Suisse

16 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 Capteur 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 inertielle BNO055	39
5.3.3 Carte SD	40
5.3.4 Application main	41
6 Validation du design	43
6.1 Liste de matériel	43
6.2 Contrôle des alimentations	43
6.2.1 Méthodologie	43
6.2.2 Mesures	44
6.3 Communication UART	44
6.3.1 Méthodologie	44
6.3.2 Mesures	45
6.4 Communication SPI, carte SD	46
6.4.1 Méthodologie	46
6.4.2 Mesures	46
7 Caractéristiques du produit fini	48
8 Conclusion	49
9 Bibliographie	50
10 Annexes	51
10.1 Fichier de modifs	52
10.2 Affiche du projet	53
10.3 Résumé	54
10.4 Mode d'emploi	55
10.5 Schéma électronique	56
10.6 Bill of materials	61
10.7 Implémentation	63
10.8 Code	65



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 dun 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 nest pas une tâche aisée due aux différentes contraintes de communication sous-marine notamment le fait que les ondes électromagnétiques ne sy propagent pas facilement.

1.2 Aperçu

- Sauvegarde dun set de donnée chaque 100ms.
- Profondeur 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 dintérêts 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 dun 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.

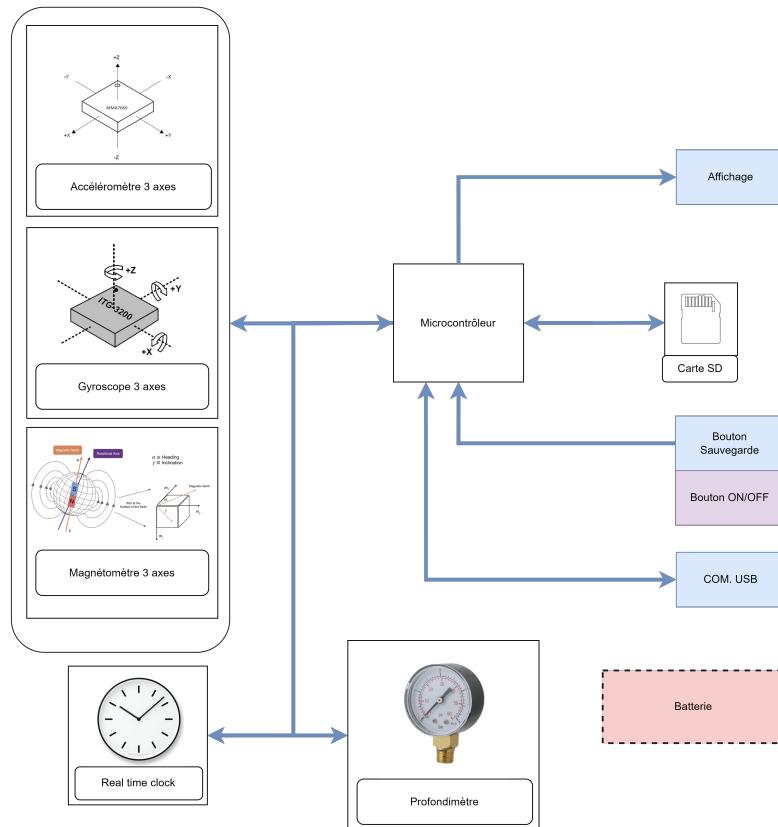


FIGURE 1 – Schéma de principe

Source : Auteur

1.4 Description des blocs

1. **Carte SD :**
Stockage des données de mesures chaque 100ms, cur 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 de rallumer ou 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 :**
Lit et traite les valeurs des capteurs, sauvegarde dans la carte SD...

1.5 Jalons principaux

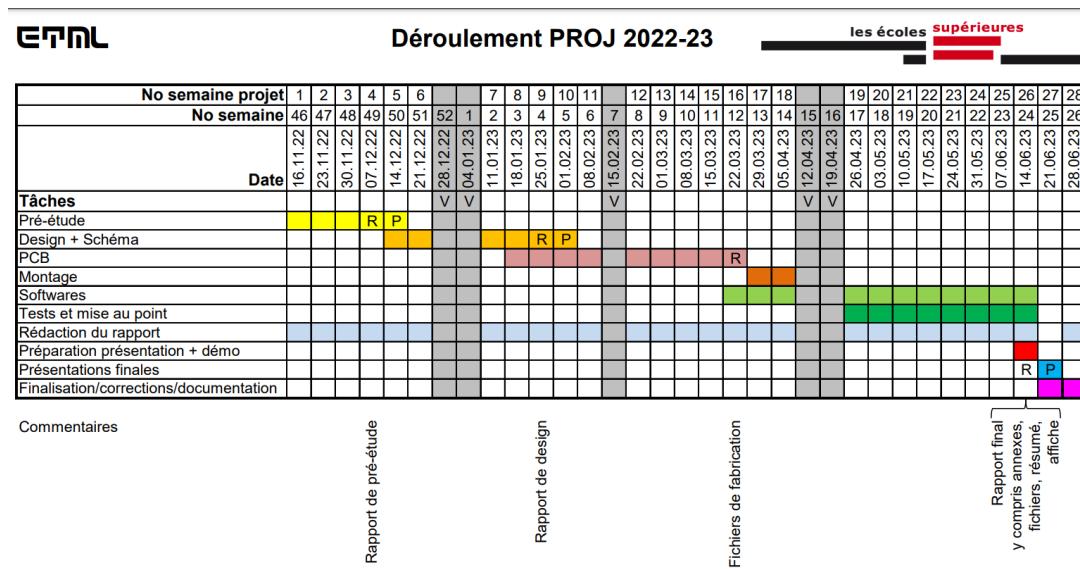


FIGURE 2 – Jalons principaux

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

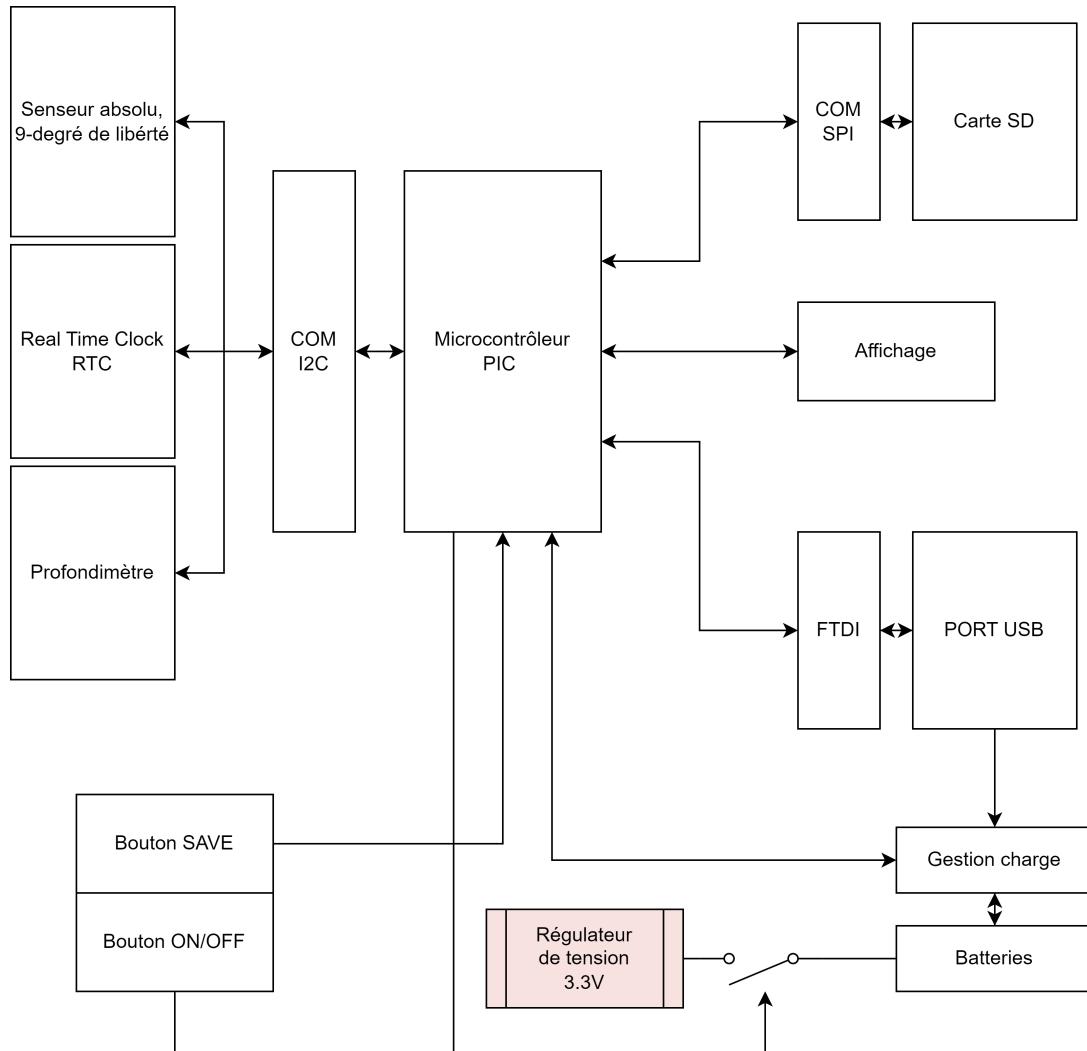


FIGURE 3 – Schéma bloc du module
Source : Auteur

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 : Un bouton étanche sera 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**¹.



FIGURE 4 – Schéma bloc du module

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

Sachant que la braise 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 Nº : **4646** qui ont des connexions bergs ainsi que tous les composants externes passifs déjà montés.

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]

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

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.

Changement : Le capteur de pression n'étant plus disponible, nous avons dû opter pour une alternative qui est :



FIGURE 7 – Capteur de pression alternatif

Source : Distrelec, MIPAM1XX010BAAAX

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 rond 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
$S_{timestamp}$	=	~ 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 **~11MB**.

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 LION 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.

changement : L'utilisation d'une RTC/RTCC a été remplacée par une simple mesure de différence de temps entre chaque mesures.

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.

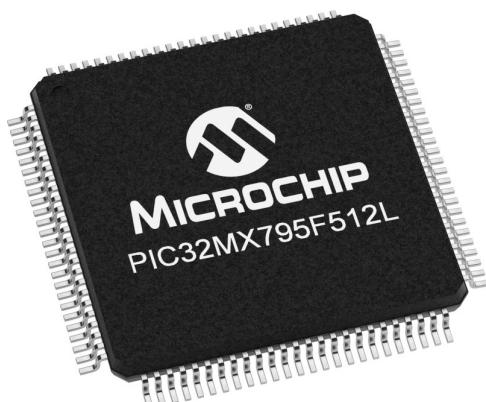


FIGURE 8 – Illustration du modèle MCU du kit ETML-ES
Source : <https://www.microchip.com/en-us/product/PIC32MX795F512L>

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 9 – 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 risques dûs au contact de l'eau (**Enflammement, éclatement...**) la technologie LI-ION est souvent utilisée pour les applications sous-marines dû à 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

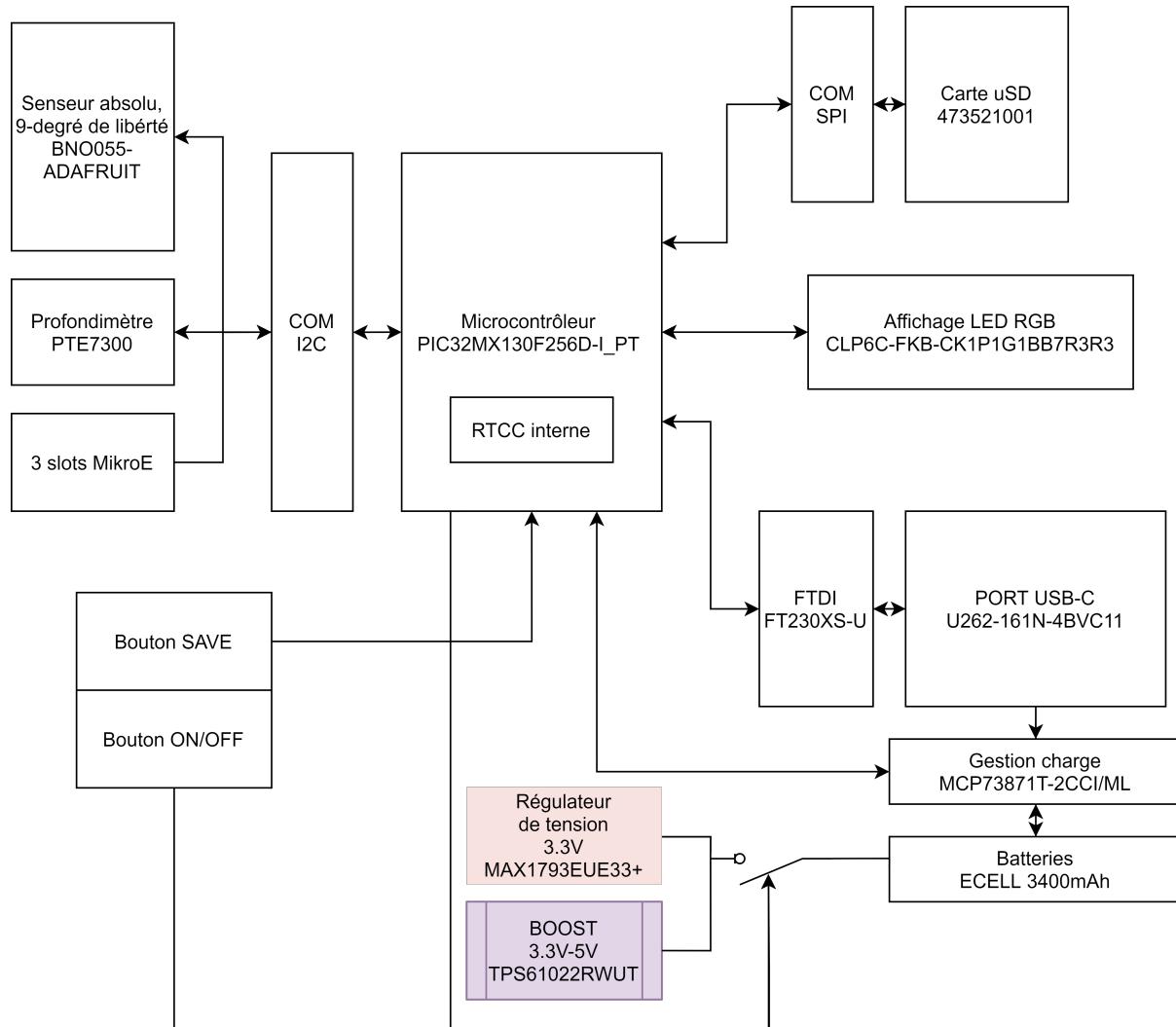


FIGURE 10 – Schéma bloc détaillé

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) ⁽¹⁾	Data Memory (KB)	Remappable Peripherals				Analog Comparators	USB On-The-Go (OTG)	I ² C	PMP	DMA Channels (Programmable/Dedicated)	CTMU	10-bit 1 Msps ADC (Channels)	RTCC	I/O Pins	JTAG	Packages	
				Remappable Pins	Timers ⁽²⁾ /Capture/Compare	UART	SPI/I ² S												
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 11 – Périphériques disponibles du PIC

Source : *PIC32MM0256GPM064 family datasheet*

Nous pouvons constater sur la figure 11 que les critères minimaux de mon projet sont respectés :

I - I2C *I - SPI* *I - UART* *I - RTCC*

3.3 Dimensionnements

3.3.1 Vue d'ensemble schématique

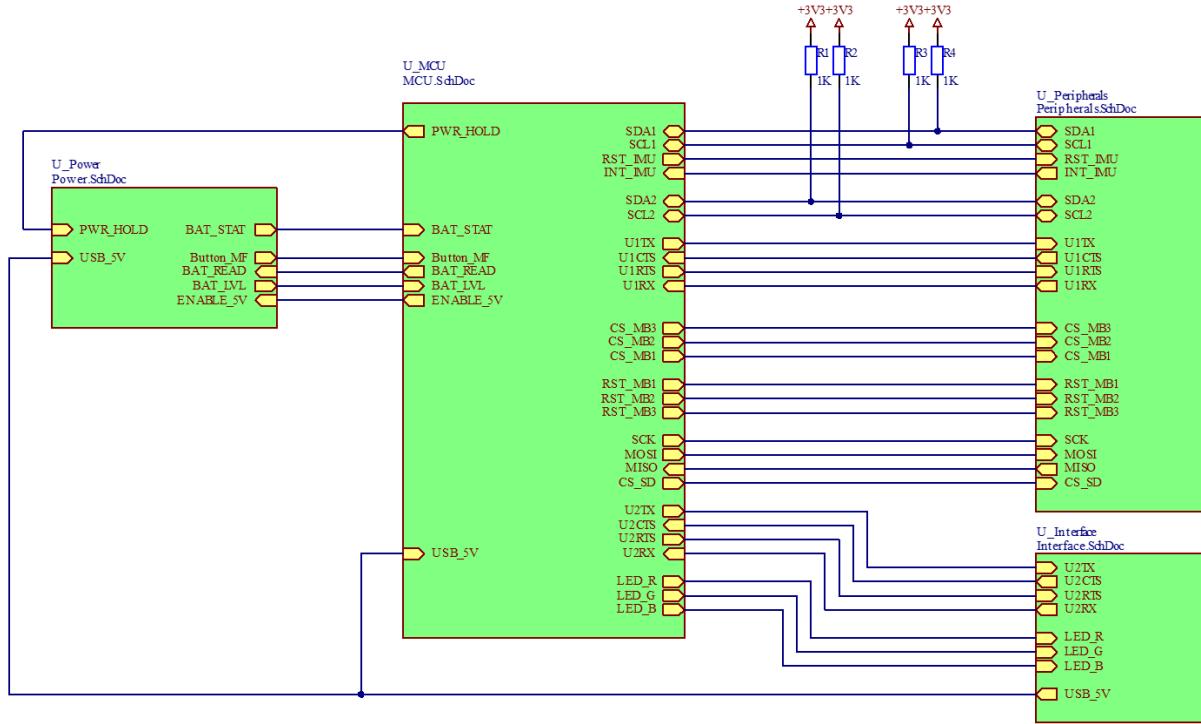


FIGURE 12 – Schéma bloc de la schématique

Source : Auteur

Nous pouvons constater sur la figure 12 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 :

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

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

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

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

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 ~ 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 pourrait requérir d'être alimentée en permanence, on pourrait donc imaginer 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 13.

2. Datasheet SanDisk

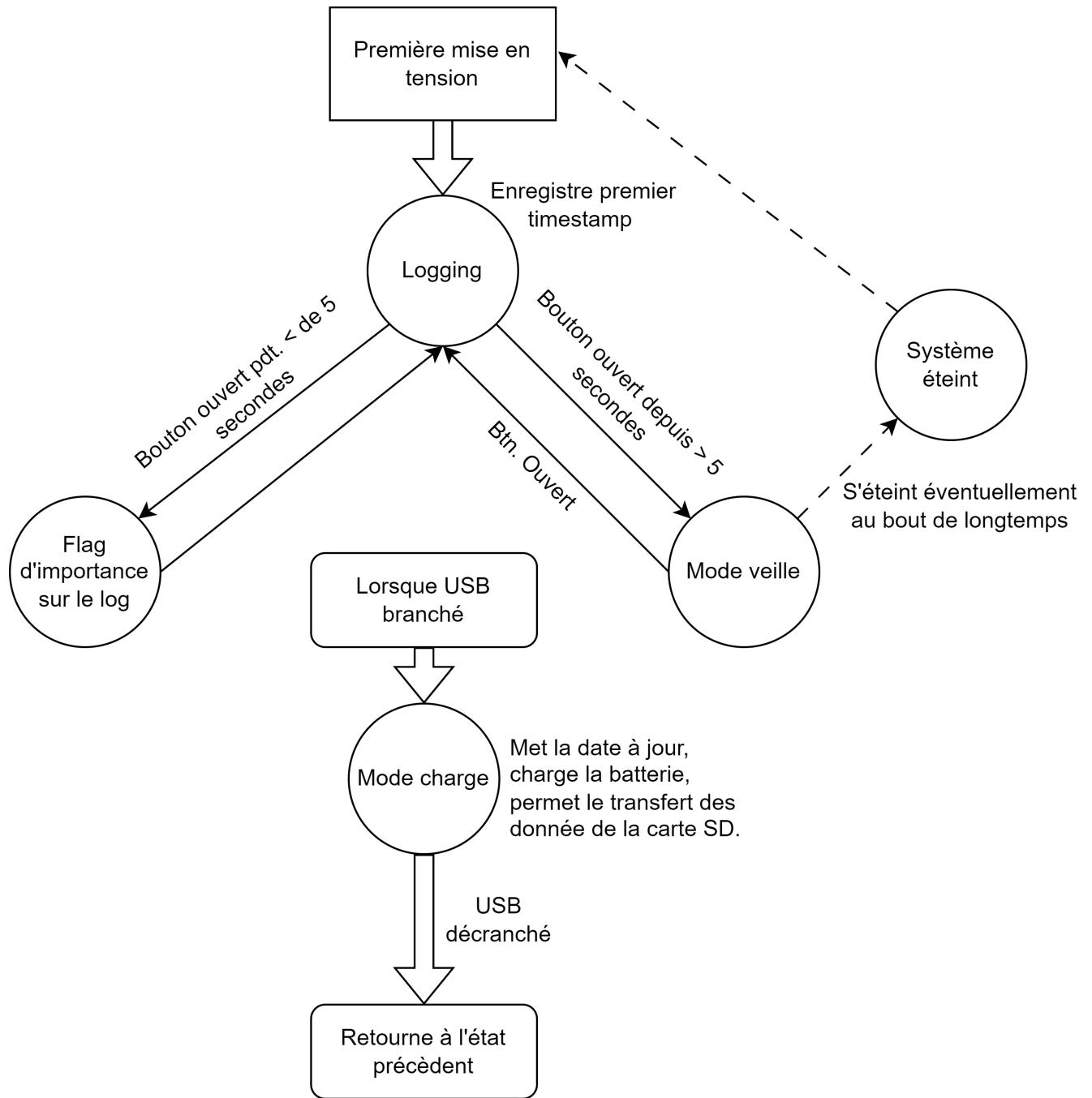


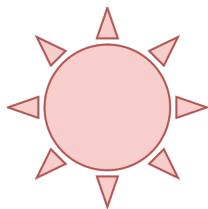
FIGURE 13 – Diagramme des états du système
Source : Auteur

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.

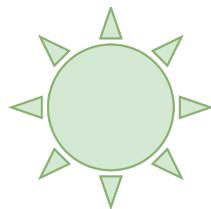
La consommation de la led RGB étant relativement importante, des mécanismes d'économie d'énergie seront mis en place dans le développement firmware.

Diagramme d'interface :



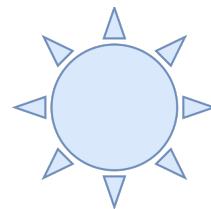
Rouge

- 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.



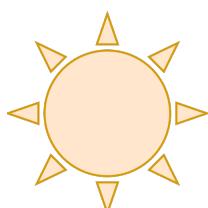
Vert

- 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.



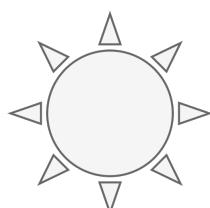
Bleu

- 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 14 – 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 :

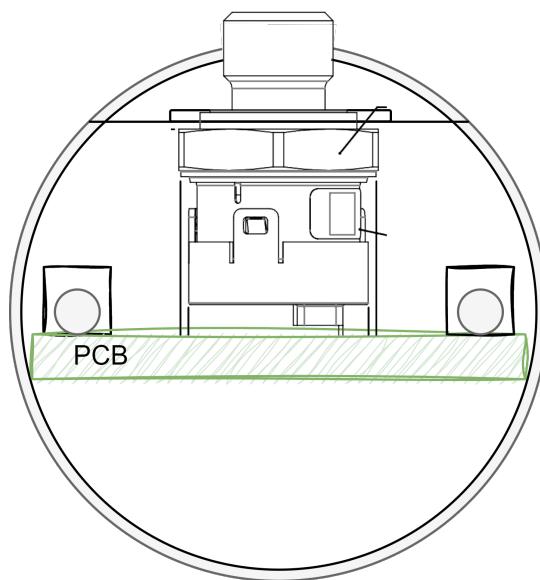


FIGURE 15 – Ébauche intérieur du cylindre

Source : Auteur

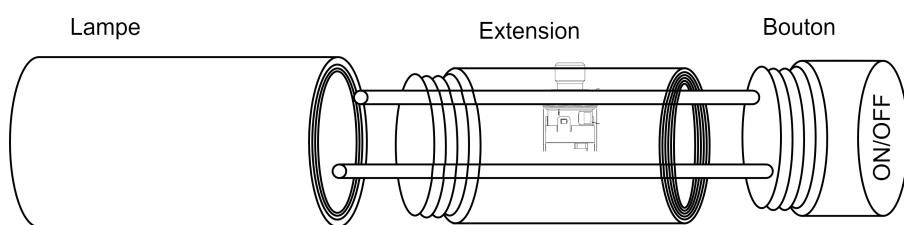


FIGURE 16 – Ébauche globale

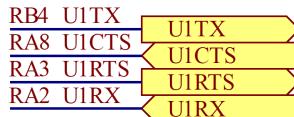
Source : Auteur

3.3.5 Bus de communications

UART (1) :

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

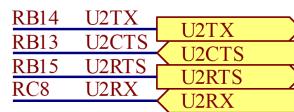
Pinning :



UART (2) :

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

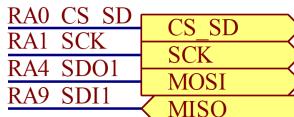
Pinning :



SPI :

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

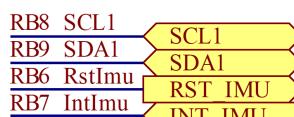
Pinning :



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 :

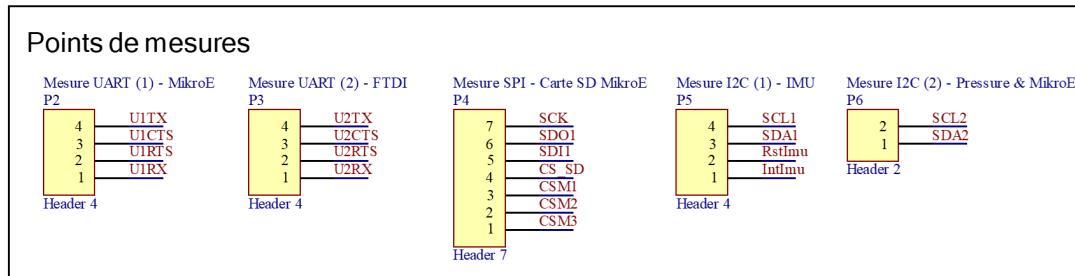


FIGURE 17 – Connecteurs pour analyseur logique

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 :

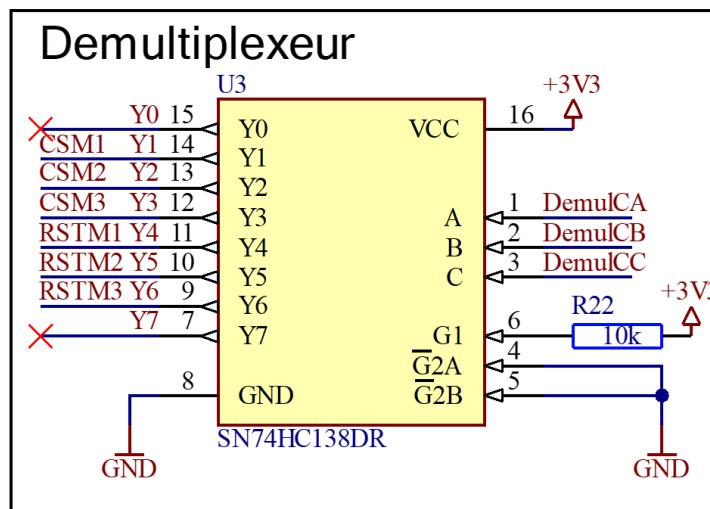


FIGURE 18 – Demultiplexeur

3.3.6 Péphériques

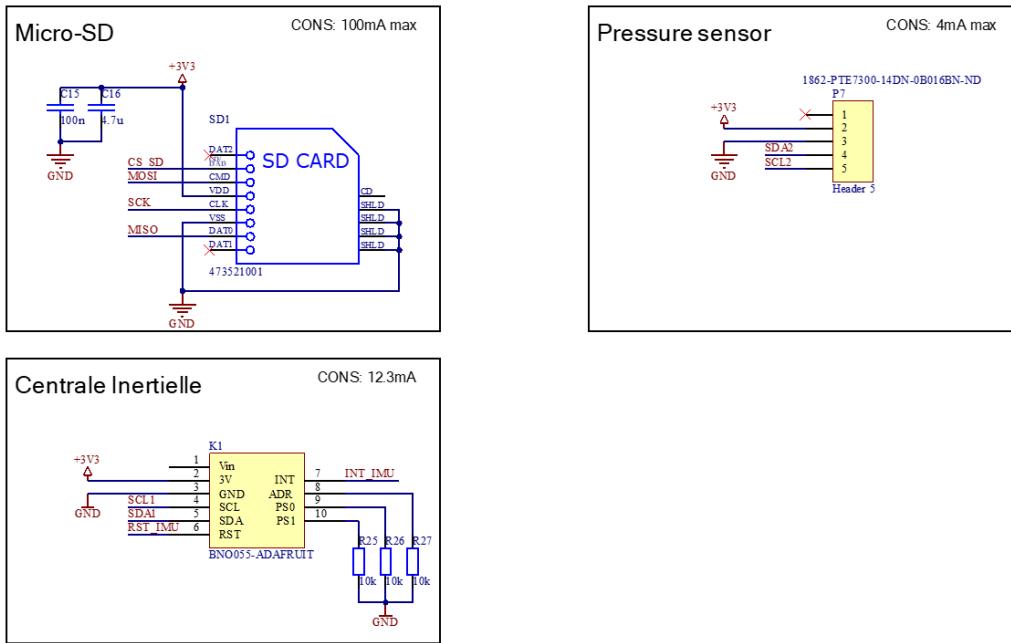


FIGURE 19 – Carte-SD, capteur de pression et centrale inertie.

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

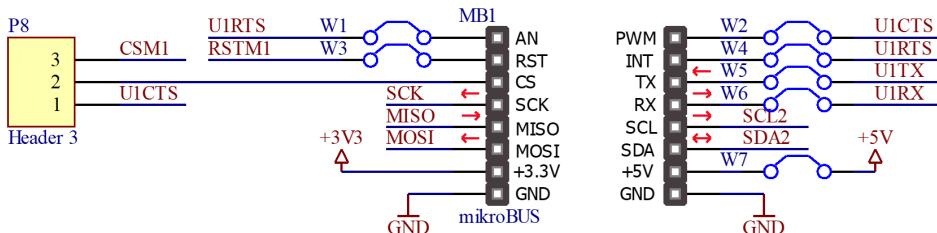


FIGURE 20

Les slots MikroE (3x figure 20) 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.

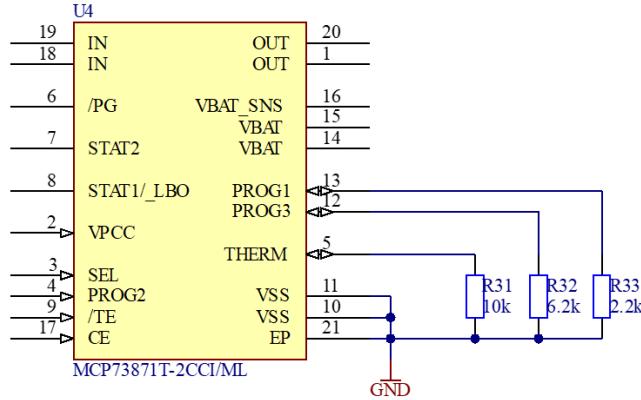


FIGURE 21 – IC régulation et gestion charge de l'accu

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

Où

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

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

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

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

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

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 global 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 22 – 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 bergs 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

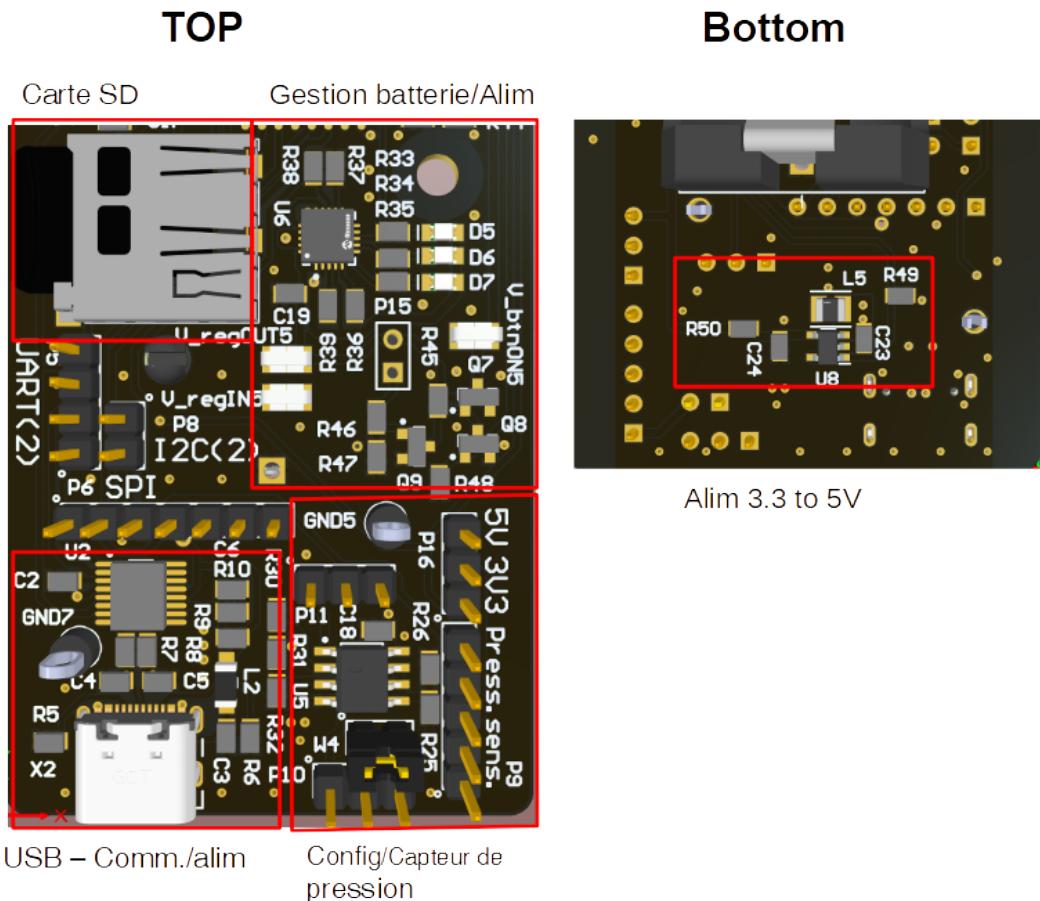


FIGURE 23 – Placement alimentations

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

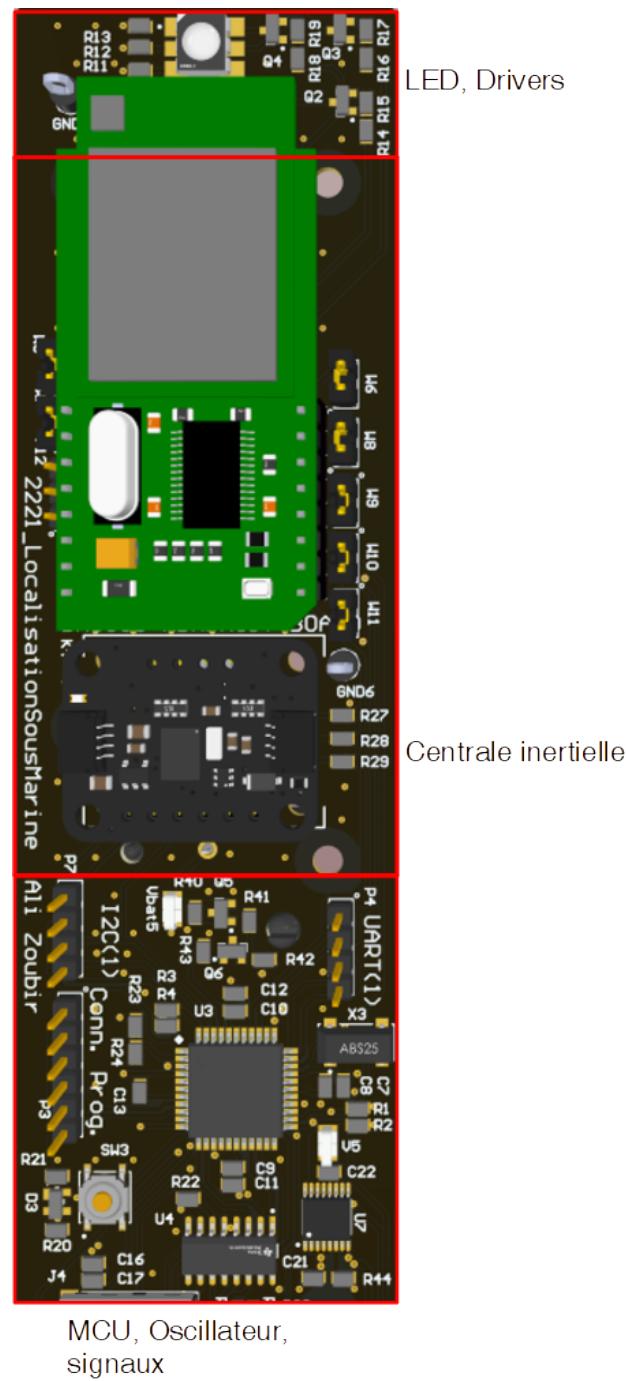


FIGURE 24 – Placement signaux

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

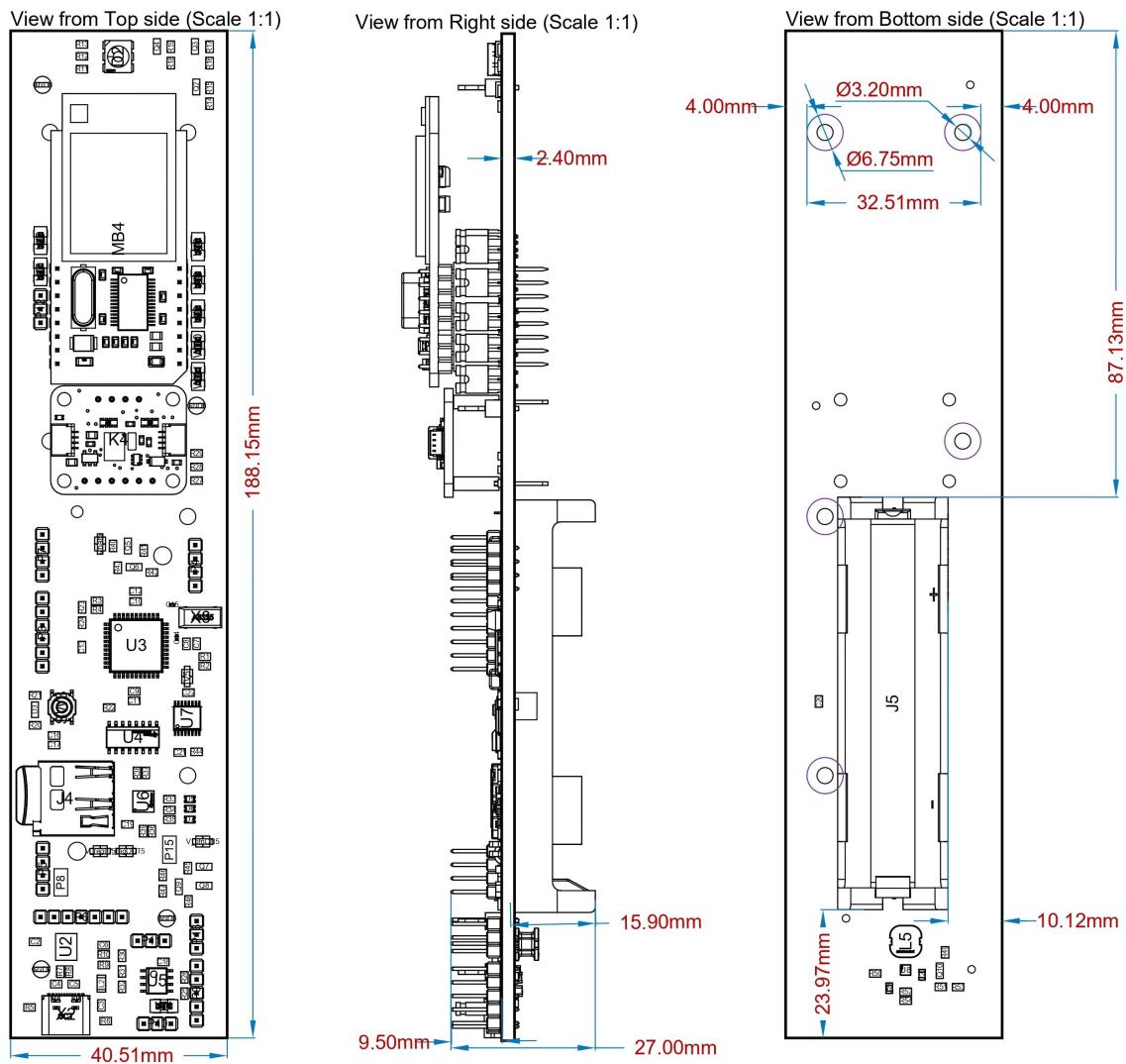


FIGURE 25 – Plan mécanique du PCB

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

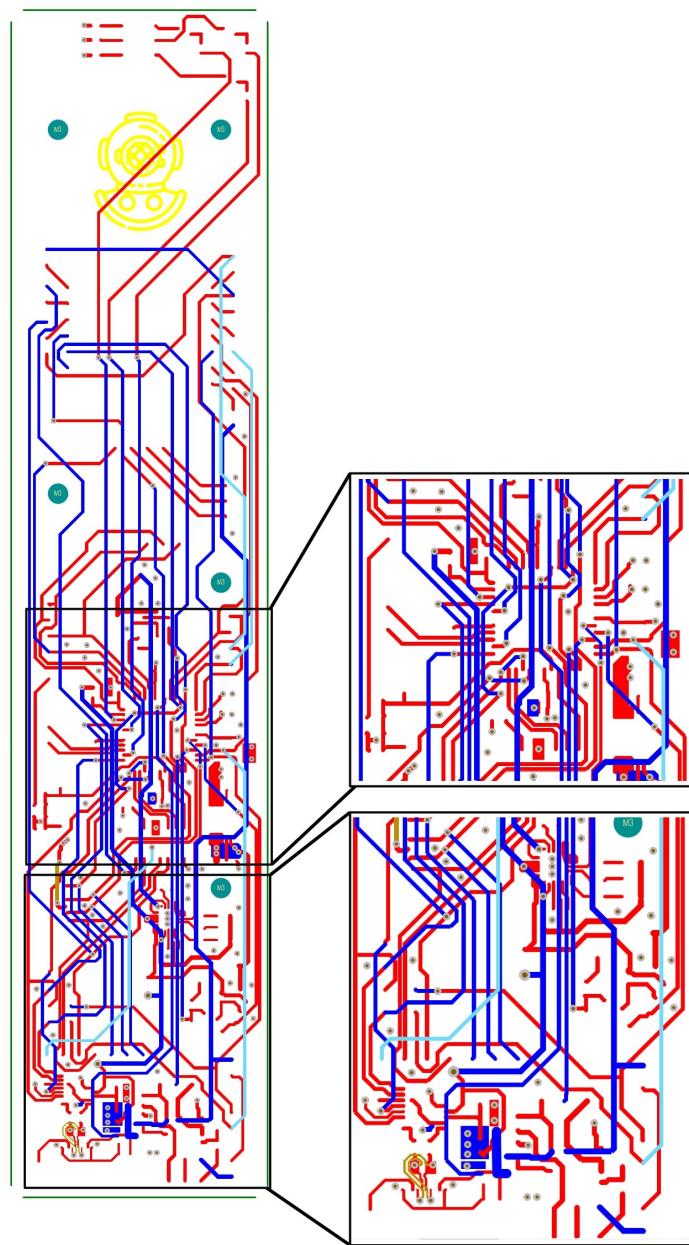


FIGURE 26 – Pistes du PCB

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 26 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 26 (Couche brune).

5 Développement firmware

Dans cette section, nous allons décrire et expliquer le processus de programmation du code qui a été implémenté dans le microcontrôleur PIC32MX130F256D. Le processus de programmation du code dans le PIC32MX130F256D comprend plusieurs étapes. Tout d'abord, il est nécessaire de disposer d'un environnement de développement intégré (IDE) adapté à ce microcontrôleur, tel que MPLAB X IDE. Cet IDE est équipé de l'environnement Harmony, qui permet l'utilisation d'un configurateur graphique pour les différentes bibliothèques du PIC.

5.1 Configuration des PINs dans Harmony

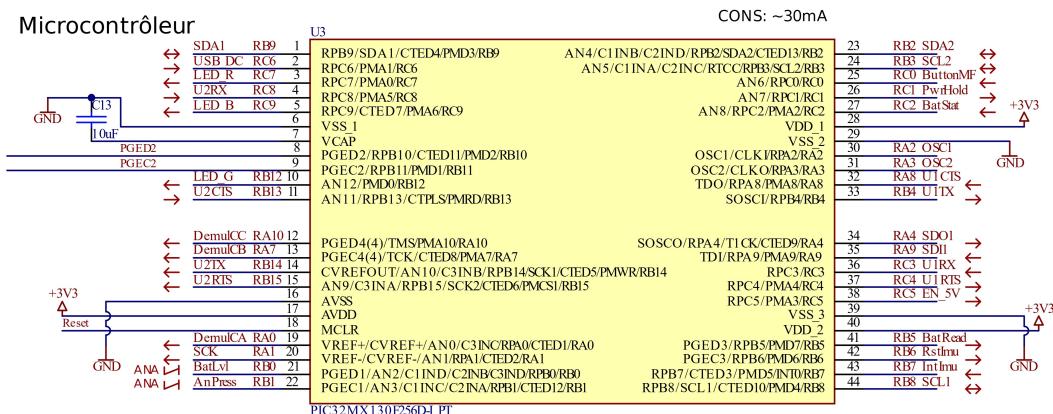


FIGURE 27 – Pinning réelles dans altium designer

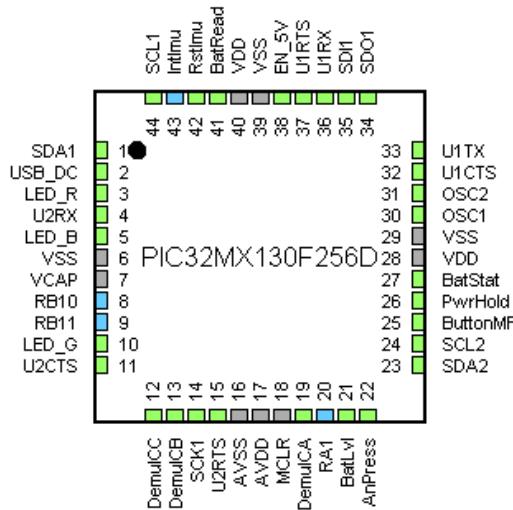


FIGURE 28 – Pinning dans Harmony

On peut constater que la PIN 20 (SCK) est en haute impédance dans Harmony, tandis que la PIN 14, qui était supposée être *U2TX*, est maintenant configurée en tant que SCK. Cela est dû à une erreur : SCK ne peut être assigné qu'à la PIN 14. Par conséquent, il a été nécessaire d'ajouter un fil externe pour relier la PIN 14 à la PIN 20, ce qui a entraîné la perte de la communication USB sur l'UART2. Cette modification sera expliquée en détail 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 fréquence de l'horloge système a été augmentée à 48 MHz dans le but d'accélérer l'ensemble du système, étant donné qu'il implique de nombreuses communications nécessitant des opérations rapides, que ce soit pour la préparation des tampons de données ou les divers calculs.

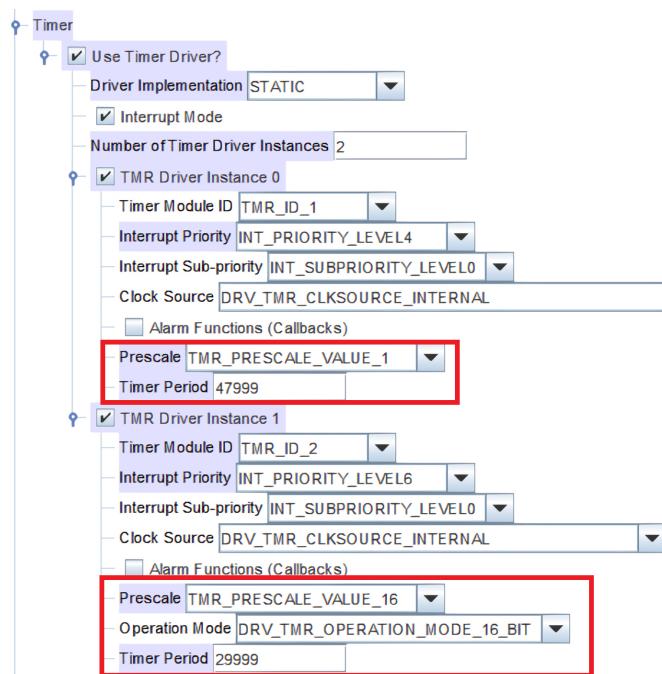


FIGURE 29 – Configuration dans harmony

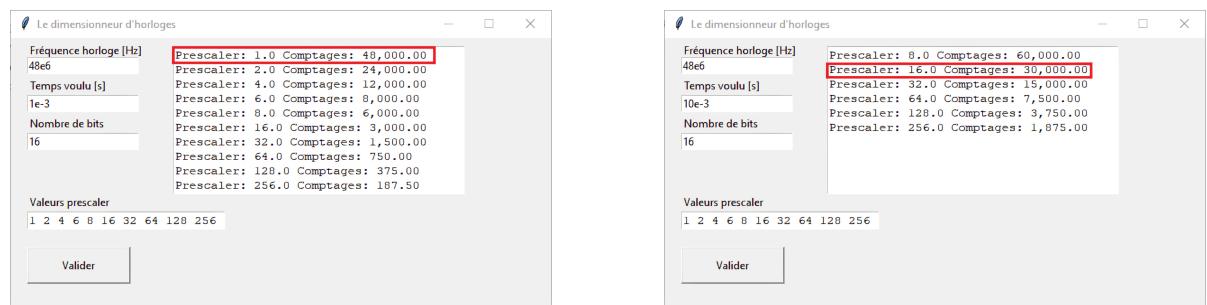


FIGURE 30 – Application timer développée par l'auteur

5.2.2 USART

Dans le but de vérifier les données de mesure en temps réel et de faciliter le débogage, une communication sérielle USART a été mise en place. Pour cela, le périphérique UART1, initialement prévu pour le slot mikroe, a été utilisé. Un module USB-to-TTL externe sera utilisé pour lire les données via Putty sur un ordinateur.

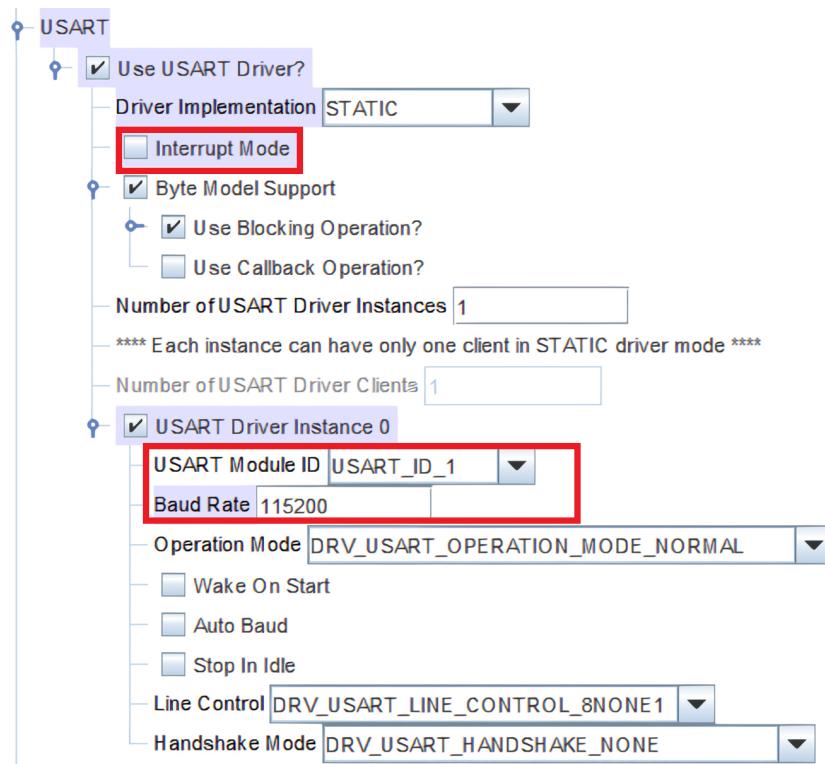


FIGURE 31 – Configuration USART

Nous pouvons constater sur la figure 31 que l'USART est configuré sans interruption à un bauderate de 115200.

5.2.3 Carte SD - SPI

L'utilisation du SPI a été optimisée en choisissant une fréquence de 5 MHz afin de réduire le temps d'exécution sur le microcontrôleur, compte tenu du grand nombre de trames nécessaires pour les opérations FAT. Cependant, j'ai rencontré un problème lié à la clock du SPI. En raison de la vitesse élevée et des modifications que j'ai dû apporter, la clock interfère inutilement avec le FTDI, et un fil relie SCK et U2TX, créant ainsi une inductance parasite. Pour résoudre ce problème, j'ai ajouté un condensateur de **33pF** entre SCK et GND pour stabiliser la communication. Vous pouvez consulter la configuration Harmony de la carte SD sur la figure 32.

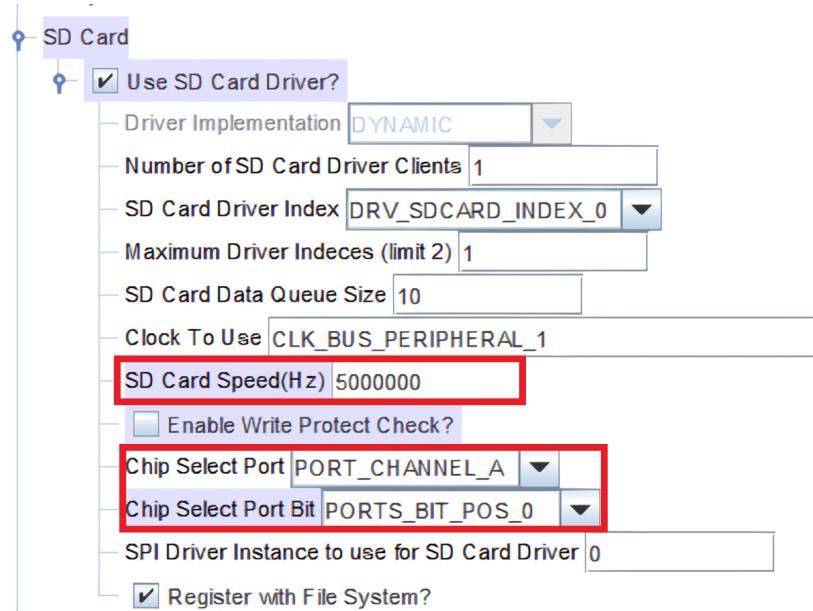


FIGURE 32 – Configuration du SPI

5.3 Code

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

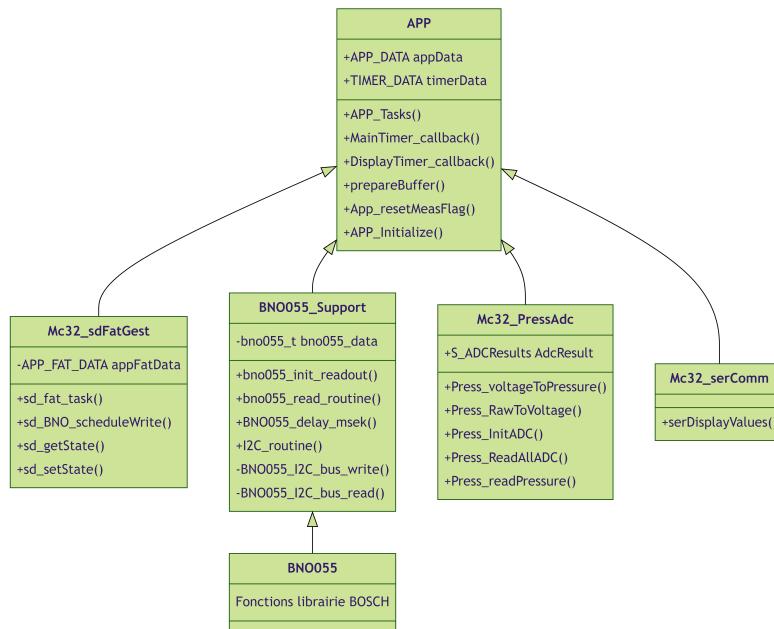


FIGURE 33 – Hiérarchie des fichiers du projet

5.3.1 Callbacks

Chacun des timers appelle dans leur interruption une fonction appartenant au fichier *app.c*, qui contient les actions définies pour chaque intervalle de temps spécifique. On peut visualiser cela sur le diagramme présenté dans la figure 34.

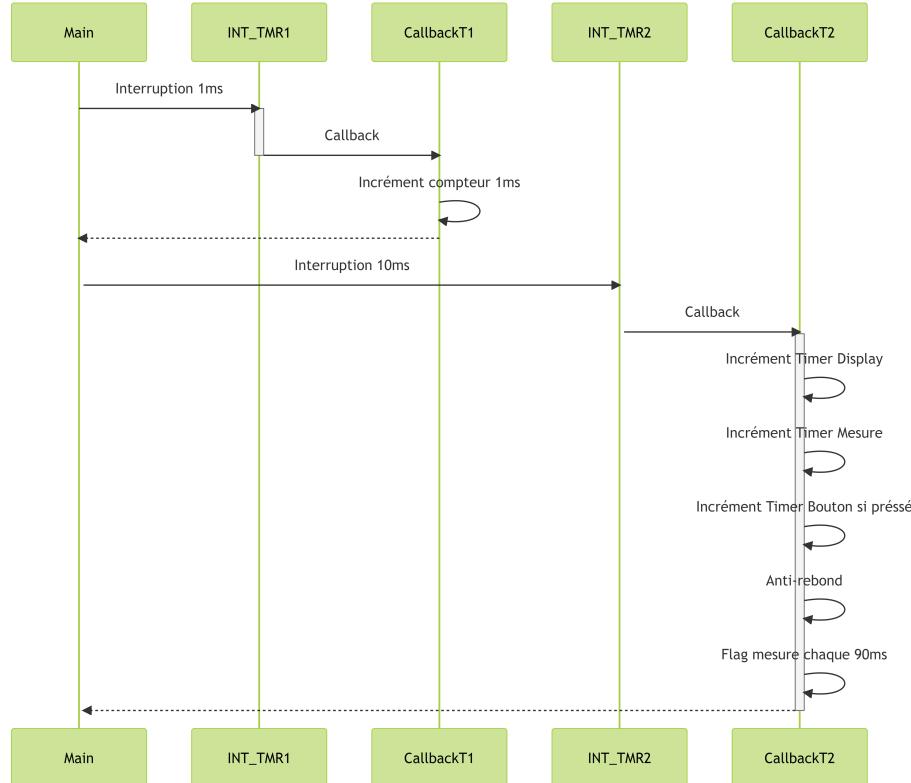


FIGURE 34 – Interactions des interruptions et des callbacks

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 bibliothèque de BOSCH³. Je l'ai configurée en 32 bits et j'ai créé les fonctions de bas niveau (i2c) en établissant le lien avec la bibliothèque BOSCH. Tout cela a été fait dans le fichier BNO055_support.c.

Pour faire le lien entre la bibliothèque de haut niveau et la bibliothèque de bas niveau, j'ai utilisé un pointeur de fonction présent dans la structure de données du BNO, comme illustré dans le listing 1.

```
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

```
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 i2c au BNO055

3. Bibliothèque 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 communication de la carte SD 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.

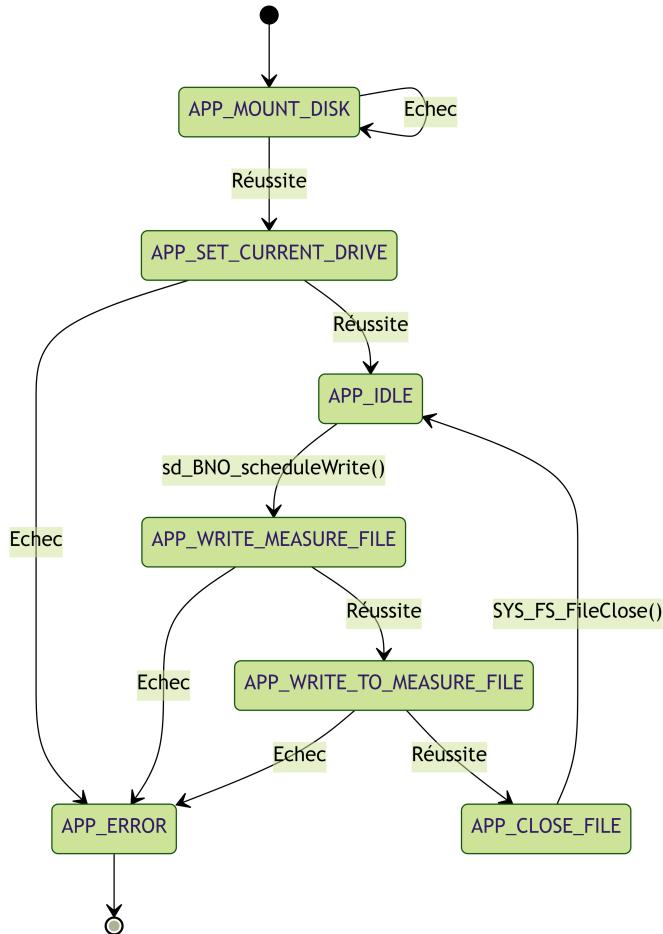


FIGURE 35 – Machine d'état de la carte SD

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

Les données sont écrite dans un fichier .CSV nommé "MESURES". La forme de la trame est la suivante :

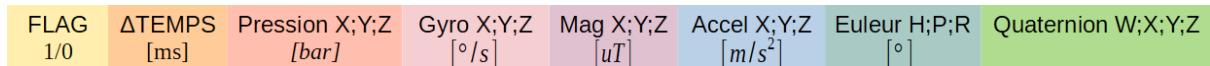


FIGURE 36 – Format de la trame

Listing 5 – Ecriture du buffer

Exemple trame CSV : 0;372;1.025;-0.2500;0.3800;9.7900;-0.1250;0.1250;-0.0625;-44.8750;35.0625;-7.2500;0.0000;-0.0100;-0.2700;0.0000;-2.1875;-1.5000;16379;320;216;-1;

5.3.4 Application main

On peut visualiser la machine d'état de l'application principale sur la figure 37.

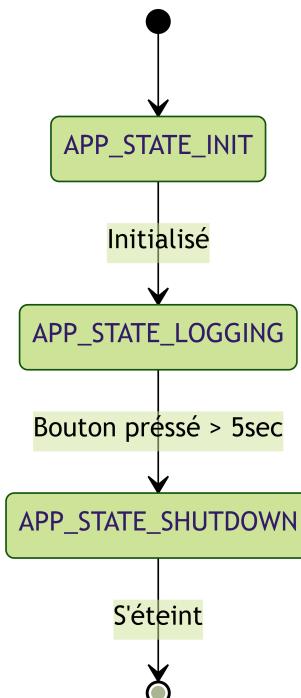


FIGURE 37 – Machine d'état application

Fonctionnement APP_STATE_LOGGING : Une fois que l'application est en mode logging, le fonctionnement est décrit sur la figure 38 sous forme de flowchart.

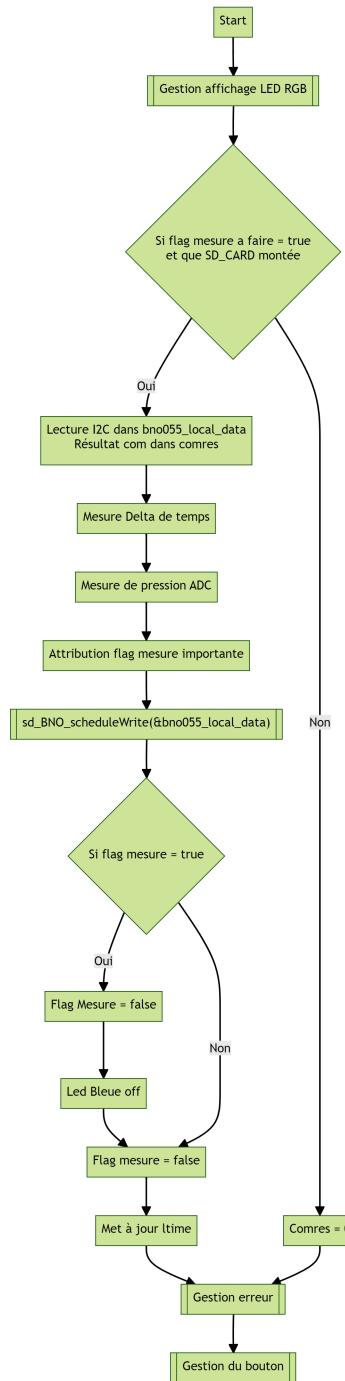


FIGURE 38 – Fowchart state logging

6 Validation du design

Lors de cette section, sera décrite la procédure de vérification des caractéristiques du projet ainsi que sa validation.

6.1 Liste de matériel

1. Oscilloscope Tektronix RTB2004 ES.SLO2.05.01.16
2. Analyseur logique USB, 8 canaux, 24 MHz
3. USB vers TTL HW-597
4. Carte Localisation-Sous-Marine V0.0

6.2 Contrôle des alimentations

En premier lieu, une vérification des tensions d'alimentation permet de valider un aspect critique et fondamental de la carte.

6.2.1 Méthodologie

Mesure du 3.3V : Alimentation de la carte par une connexion brève entre les pins du connecteur du bouton **P15**. Mesure sur le testpoint *V_regOUT5*, voir figure 39.

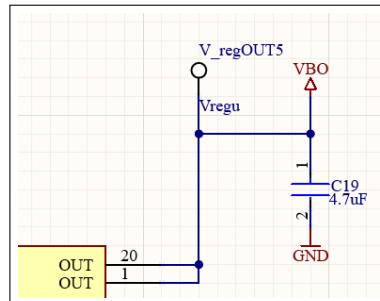


FIGURE 39 – Testpoint mesure 3.3V

Mesure du 5V : Pour mesurer le 5V, il faut ponter par une résistance de 0Ω la résistance R50, ainsi que activer la Pin RC5 / EN_5V. Ensuite, la mesure a été prise sur le connecteur P16, pin-3, avec l'oscilloscope (Numéro 1 de la liste de matériel 6.1) :

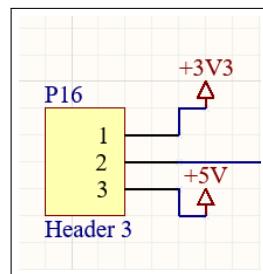
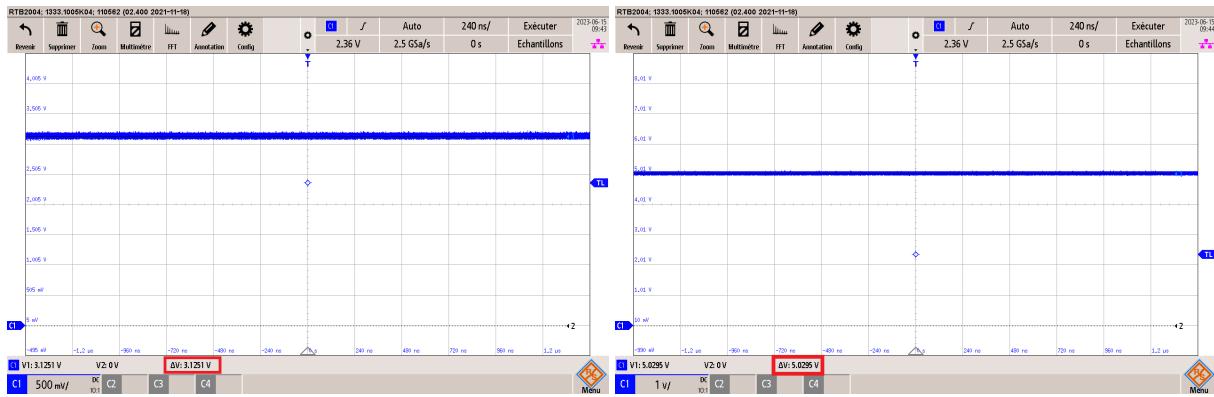


FIGURE 40 – Schéma de mesure 5V



(a) Mesure 3.3V

(b) Mesure 5V

FIGURE 41 – Mesures des tensions d’alimentation

6.2.2 Mesures

Analyse : Nous pouvons observer les valeurs respectives des tensions sur les figures 41a et 41b. La tension d’alimentation du microcontrôleur est mesurée à 3.125V, ce qui peut être expliqué par une chute de tension aux bornes de la batterie LI-ION qui nécessite donc d’être chargée. La tension 3.3V oscille légèrement, ce qui peut s’expliquer par le fait qu’il s’agit de l’alimentation principale de la carte avec le plus de consommateurs. Mais également par le fait que des signaux rapides sont générés par le microcontrôleur et ses différents périphériques.

La tension d’alimentation du capteur de pression, dont la tension dimensionnée est de 5V, est quant à elle mesurée à 5.0295V, ce qui signifie une bonne précision de la part du circuit de boost. On peut également voir que la tension n’oscille pas et ne va donc pas perturber le capteur de pression.

Nous pouvons donc confirmer le fonctionnement des blocs d’alimentation du projet.

6.3 Communication UART

Comme décrit lors de la sous-section 5.2.2, une communication série est implémentée dans le projet. Cette communication n’est pas critique pour le projet, mais il est important de vérifier son fonctionnement pour d’éventuelles versions ultérieures.

6.3.1 Méthodologie

Pour la mesure, l’analyseur logique (Numéro 2 de la liste de matériel 6.1) a été utilisé. Les trames U1TX et U1RX ont été mesurées sur le connecteur P4 :

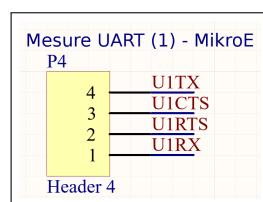


FIGURE 42 – Schéma de mesure UART1

6.3.2 Mesures

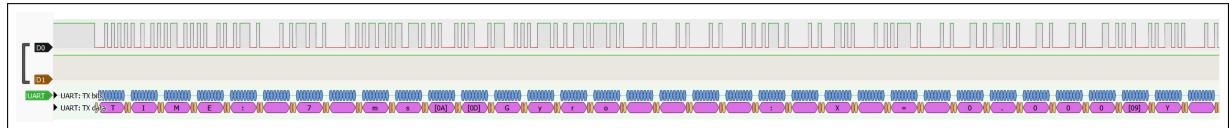


FIGURE 43 – Mesures trames UART

Nous pouvons lire des caractères ASCII sur la trame de la figure 43, qui reflètent le message envoyé contenant les mesures.

En connectant le module USB-TO-TTL (Numéro 3 de la liste de matériel 6.1) aux broches Tx et Rx de la figure 43, puis en ouvrant une communication série via PuTTY à une baudrate de 115200, nous obtenons la communication de la figure 44 dans la console :

```

Quater. : W = -13504     X = 00709      Y = -1819      Z = -9071
DT: 60 ms      PRESSURE = -1.236375  Gravity : X = -2.260     Y = 0.500      Z = 9.520
Gyro : X = 0.125      Y = -0.313    Z = 0.000
Mag  : X = 41.375      Y = 1.375    Z = -24.687
Accel : X = 0.060      Y = 0.040    Z = -0.010
Euler : H = 292.187    P = -3.000    R = -13.313
Quater. : W = -13504     X = 00709      Y = -1819      Z = -9071

DT: 60 ms      PRESSURE = -1.236375  Gravity : X = -2.260     Y = 0.500      Z = 9.520
Gyro : X = 0.188      Y = -0.188    Z = 0.000
Mag  : X = 42.063      Y = 1.062    Z = -25.187
Accel : X = 0.040      Y = 0.040    Z = -0.040
Euler : H = 292.187    P = -3.000    R = -13.313
Quater. : W = -13504     X = 00709      Y = -1819      Z = -9071

DT: 60 ms      PRESSURE = -1.236375  Gravity : X = -2.260     Y = 0.500      Z = 9.520
Gyro : X = -0.063      Y = 0.000    Z = 0.125
Mag  : X = 42.500      Y = 1.750    Z = -24.687
Accel : X = 0.030      Y = 0.030    Z = -0.030
Euler : H = 292.187    P = -3.000    R = -13.313
Quater. : W = -13504     X = 00709      Y = -1819      Z = -9071

DT: 60 ms      PRESSURE = -1.236375  Gravity : X = -2.260     Y = 0.500      Z = 9.520
Gyro : X = 0.000      Y = 0.188    Z = 0.063
Mag  : X = 41.000      Y = 1.375    Z = -25.562
Accel : X = 0.050      Y = 0.040    Z = -0.010
Euler : H = 292.187    P = -3.000    R = -13.313
Quater. : W = -13504     X = 00709      Y = -1819      Z = -9071

DT: 60 ms      PRESSURE = -1.236375  Gravity : X = -2.260     Y = 0.500      Z = 9.520
Gyro : X = -0.125      Y = 0.188    Z = 0.125
Mag  : X = 42.063      Y = 1.375    Z = -25.562
Accel : X = 0.050      Y = 0.040    Z = -0.010
Euler : H = 292.187    P = -3.000    R = -13.313
Quater. : W = -13504     X = 00709      Y = -1819      Z = -9071

```

FIGURE 44 – Réception UART sur PuTTY

Analyse de la réception : Sur la figure 44, on peut visualiser les données de la centrale inertuelle, le temps écoulé entre chaque mesure, ainsi que les données du capteur de pression. Nous pouvons observer qu'entre chaque mesure, il s'écoule un laps de temps de $60ms$, ce qui est plus rapide que les spécifications demandées par le cahier des charges. La mesure du capteur de pression à -1.236 bar n'est pas cohérente, car le capteur de pression n'était pas connecté au moment des mesures. Nous pouvons également constater que les mesures de la centrale inertuelle sont cohérentes selon les données typiques d'une centrale inertielles⁴.

4. Exemple de données de fusion d'une centrale inertielles [4]

6.4 Communication SPI, carte SD

Il reste maintenant un élément critique à valider : la communication SPI avec la carte SD. Cette communication permet le logging des valeurs de mesure et est donc un aspect essentiel du projet.

6.4.1 Méthodologie

Pour cette mesure j'ai mesuré les signaux (SCK, SDO1, SDI1, CS_SD) du SPI avec l'analyseur logique (Numéro 2 de la liste de matériel 6.1).

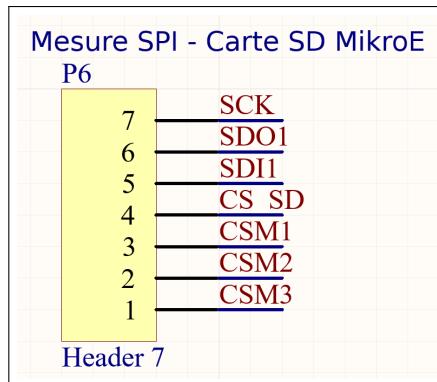


FIGURE 45 – Schéma de mesure carte SD

6.4.2 Mesures

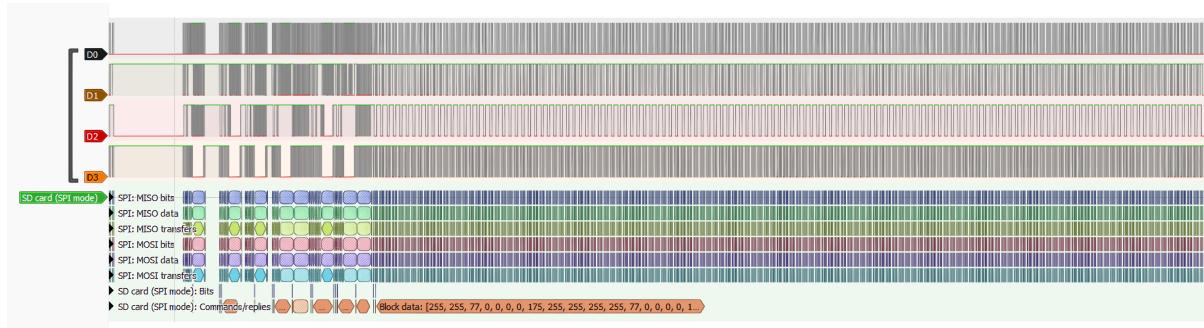


FIGURE 46 – Vue d'ensemble de la communication SPI

Comme le montre la figure 46, on peut constater que les trames SPI utilisées pour la gestion de la FAT sont nombreuses. C'est pourquoi il est important d'avoir une fréquence élevée afin d'optimiser le temps des échanges de données.

Nous allons maintenant mesurer le temps entre deux écritures sur la carte SD afin de vérifier le temps écoulé entre les mesures. et l'écriture.

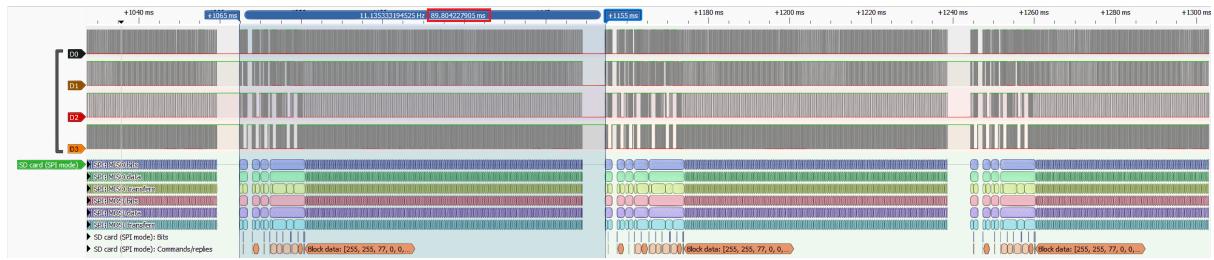


FIGURE 47 – Temps entre deux écritures sur la carte SD

Nous pouvons observer sur la figure 47 que le temps mesuré entre une première écriture et le début de la suivante est de $89.804ms$. Le temps disponible entre deux écritures est d'environ $\sim 6ms$. Nous pouvons donc constater l'importance d'une vitesse élevée sur la communication SPI et sur le microcontrôleur.

Avec une fréquence d'horloge de $48MHz$, on peut donc déduire :

$$N_{op} = T_{dispo} * F_{sys} = 6 * 10^{-3} * 48 * 10^6 = 288'000 \quad (7)$$

D'après le calcul 7 nous savons que le MCU peut effectuer 288'000 cycles machines durant ce temps disponible. Nous considérons ce nombre d'opérations suffisants au système, sachant qu'il n'y a pas beaucoup d'autre opérations. **Cet élément est à prendre en compte en cas d'ajout de fonctionnalité dans le système.**

Mesure début de trame : Sur la figure 48 nous pouvons observer à quoi ressemble un début de trame sur la carte SD pour système FAT.

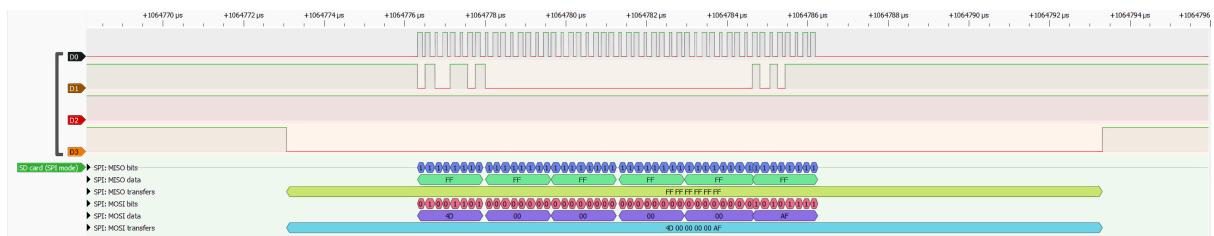


FIGURE 48 – Mesure début de trame

7 Caractéristiques du produit fini

Caractéristique	Attribut	Valeur alternative
Axes de mesures	9	-
Mesures	[ms][bar][°/s][uT][Euler][Quaternion][°C]	
Temps entre mesures	90 [ms]	11.11 Hz
Nombre de mesures max	1.641 M	-
Capacité carte SD	256 [MB]	-
Pression maximum	10 [bars]	145 PSI
Autonomie	~ 20 [h]	72'000 [s]
Batterie	3400 [mAh]	11.22 [Wh]
Profondeur	101.97 [mH2O]	-
Précision pression	0.15 [%]	-
Slot Mikroe	1 [-]	-
Vitesse MCU	48 [MHz]	-
Interface	LED RGB	-
Communications	I2C, SPI, UART, USB	-
Vitesse SPI	5 [MHz]	-
Vitesse UART	115200 [Bd]	-
Mise en évidence mesure	Oui	-
Compensation température	Oui	-

TABLE 3 – Caractéristiques du projet V0.0

On peut voir sur le tableau 3 les caractéristiques issues du développement final du projet dans sa version 0.0. Un test de logging de 5 heures a été effectué sans difficulté et a récolté 30 MB de données dans le fichier CSV, ce qui correspond effectivement aux caractéristiques du tableau 3.

8 Conclusion

Synthèse Le présent rapport a décrit le développement et la validation d'un projet de localisation sous-marine. L'objectif principal du projet était de concevoir et de mettre en oeuvre un système capable de collecter et de stocker des données de déplacement, de temps et de pression lors de plongées. Après avoir réalisé les différentes étapes de développement, nous pouvons conclure que le projet a été mené à une version finie.

Développement Au cours de la phase de développement, plusieurs étapes clés ont été franchies. Tout d'abord, une analyse approfondie des besoins et des contraintes a été réalisée, ce qui a permis de définir les spécifications du système. Ensuite, un processus itératif de conception a été suivi, comprenant la sélection des composants appropriés, la création des schémas électroniques, et la fabrication du prototype.

Design L'évaluation du design du projet a été effectuée en suivant une méthodologie rigoureuse de vérification et de validation. Les principales caractéristiques du projet, telles que les tensions d'alimentation, la communication UART et la communication SPI avec la carte SD, ont été vérifiées avec succès. Les mesures effectuées ont montré que le système fonctionnait correctement et répondait aux spécifications requises.

Test Le projet a été testé avec succès lors d'un enregistrement de données de déplacement pendant une durée de 5 heures, ce qui a permis de collecter 30 MB de données. Ces résultats confirment que le système est capable de fonctionner de manière fiable et de fournir les fonctionnalités attendues.

Apports Ce projet a permis d'acquérir une expérience précieuse dans le domaine de la conception et du développement de systèmes électroniques. Il a également mis en évidence l'importance de l'organisation, de la structure et de la vérification étape par étape des éléments du design.

Correctifs Afin de simplifier la mise en place du système, des correctifs mentionnés dans le fichier MODIF (Section 10.1) permettrait de palier à certaines erreurs non-critiques de développements.

Améliorations Des améliorations et des développements futurs peuvent être envisagés, tels que l'ajout de fonctionnalités supplémentaires, l'optimisation de la communication, l'extension des capacités de stockage ainsi que la mise en place d'une communication USB directement par le FTDI en corrigeant le pinning de SCK. Ces évolutions permettraient d'explorer de nouvelles possibilités d'application de ce système de localisation sous-marine.

Remerciements Je tiens à remercier sincèrement M. Juan José Moreno, mon responsable de projet, Dr. Gaston Baudat pour avoir mandaté le projet et pour sa contribution algorithmique sur la localisation, et M. Frédéric Mueller pour son aide mécanique et pour la conception de la rallonge du module. Leur soutien et leur expertise ont été essentiels pour mener à bien ce projet. Je suis reconnaissant de leur précieuse collaboration et de leurs conseils tout au long du processus.

9 Bibliographie

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.”
- [4] M. Rozbicka-Goodheart, “Estimation of object orientation using sensor fusion and vhdl,” 2021.

10 Annexes



FIGURE 49 – Illustration annexes

Source : <https://www.alamyimages.fr/photo-image-vecteur-icone-annexe-175637003.html>

10.1 Fichier de modifs



ELCO – SLO

Projet ETML-ES – Modification

PROJET:	2221 Localisation sous-marine		
Entreprise/Client:	Gaston Baudat	Département:	SLO
Demandé par (Prénom, Nom):	Juan José Moreno	Date:	17.06.2023
Objet (No ou réf, pièce, PCB...)	2221		
Version à modifier:	V0.0		

Auteur (ETML-ES):	Ali Zoubir	Filière:	SLO
Nouvelle version:	V0.0	Date:	05.12.2018

1 Description ou justification

Des correctifs sont à faire pour perfectionner le fonctionnement du projet, notamment permettre une communication USB directe et simplifier certains éléments.

2 Référence conception

K:\ES\PROJETS\SLO\2221_LocalisationSousMarine

https://github.com/Ali-Z0/2221_LocalisationSousMarine.git

3 Détail des modifications

#	Description	Fait	Approuvé
1	Changer les pins U2TX et SCK (SCK valable que sur la pin 14 au-lieu de 29)	Non	
2	Oscillateur externe doit être connecté comme oscillateur secondaire et non principale	Non	
3	Revoir système multiplexeur pour chips select (polarité cs de la carte sd différent)	Non	

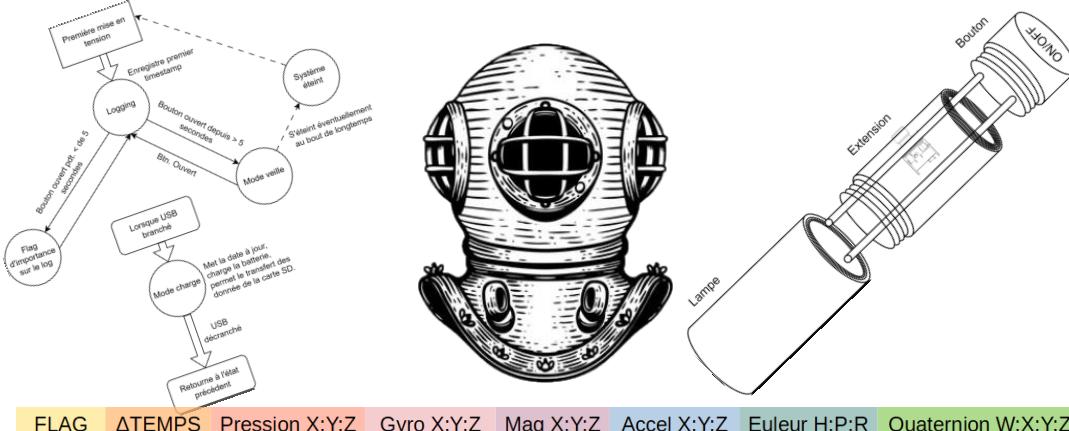
4 Remarques

Les modifications ci-dessus sont à faire pour une version ultérieure du projet.

10.2 Affiche du projet

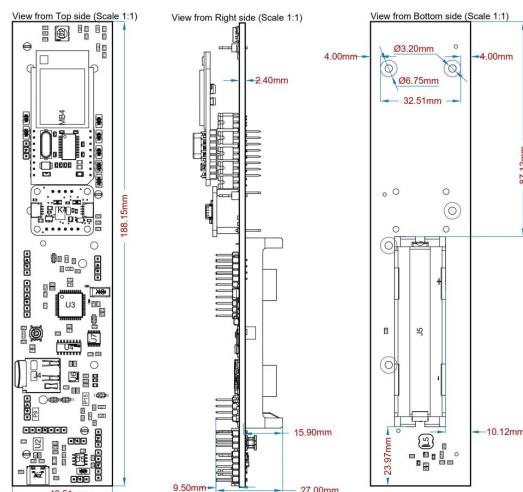


L'objectif de ce projet, et de stocker des données de mesures du déplacement d'un **module sous-marin** par une **centrale inertielle**, dans le but de mathématiquement le **localiser** depuis son point de départ (référence). Ceci, car la localisation sous-marine n'est pas une tâche aisée due aux différentes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



FLAG	ΔTEMPS	Pression X;Y;Z	Gyro X;Y;Z	Mag X;Y;Z	Accel X;Y;Z	Euler H;P;R	Quaternion W;X;Y;Z
1/0	[ms]	[bar]	[°/s]	[uT]	[m/s²]	[°]	

Caractéristique	Attribut	Valeur alternative
Axes de mesures	9	-
Mesures	[ms] [bar] [°/s] [uT] [Euler] [Quaternion] [°C]	
Temps entre mesures	90 [ms]	11.11 Hz
Nombre de mesures max	1.641 M	-
Capacité carte SD	256 [MB]	-
Pression maximum	10 [bars]	145 PSI
Autonomie	~ 20 [h]	72'000 [s]
Batterie	3400 [mAh]	11.22 [Wh]
Profondeur	101.97 [mH2O]	-
Précision pression	0.15 [%]	-
Slot Mikroe	1 [-]	-
Vitesse MCU	48 [MHz]	-
Interface	LED RGB	-
Communications	I2C, SPI, UART, USB	-
Vitesse SPI	5 [MHz]	-
Vitesse UART	115200 [Bd]	-
Mise en évidence mesure	Oui	-
Compensation température	Oui	-



Un processus itératif de **conception** a été suivi, comprenant la sélection des composants appropriés, la **création** des schémas électroniques, et la **fabrication** du prototype.

L'évaluation du design du projet a été effectuée en suivant une méthodologie rigoureuse de vérification et de **validation**. Les principales caractéristiques du projet, ont été vérifiées avec **succès**. Les mesures effectuées ont montré que le système fonctionnait correctement et répondait aux spécifications requises.

Des **améliorations** et des développements futurs peuvent être envisagés, tels que l'ajout de fonctionnalités supplémentaires, l'optimisation de la **communication**, l'extension des capacités de **stockage** ainsi que la mise en place d'une communication **USB** directement par le FTDI implémenté. Ces évolutions permettraient d'explorer de nouvelles possibilités d'application de ce système de localisation sous-marine.

10.3 Résumé



ETML-ES

RESUMÉ – Projet

Ali Zoubir
SLO 2
2022-2023

Titre :

2221 Localisation Sous-Marine

Contexte et objectifs :

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 le fait que les ondes électromagnétiques ne s'y propagent pas facilement.

Résultats obtenus et conclusion :

Objectif : Concevoir et mettre en œuvre un système de collecte et de stockage de données de déplacement, de temps et de pression lors de plongées.

Développement : Analyse approfondie des besoins et des contraintes, définition des spécifications du système, conception itérative, sélection des composants appropriés, création des schémas électroniques et fabrication du prototype.

Design : Évaluation rigoureuse du design avec succès, vérification des principales caractéristiques telles que les tensions d'alimentation et les communications UART et SPI avec la carte SD.

Test : Enregistrement réussi de données de déplacement pendant 5 heures, collecte de 30 Mo de données, confirmant la fiabilité et les fonctionnalités du système.

Apports : Acquisition d'une précieuse expérience en conception et développement de systèmes électroniques, soulignant l'importance de l'organisation et de la vérification étape par étape.

Correctifs : Mise en place de correctifs pour simplifier l'installation du système et résoudre certaines erreurs non critiques de développement.

Améliorations futures : Ajout de fonctionnalités supplémentaires, optimisation de la communication, extension des capacités de stockage et mise en place d'une communication USB directe par le FTDI en corrigeant le pinning de SCK pour explorer de nouvelles possibilités d'application.

Maître(s) de projet :
Entreprise mandataire :

Juan José Moreno
Gaston Baudat



Ecole supérieure, école des métiers Lausanne
Rue de Sébeillon 12
CH-1004 Lausanne
www.etml-es.ch

| 1 / 1 |

10.4 Mode d'emploi

1. Insérer la carte SD format FAT.
2. Allumer le système avec une impulsion sur le bouton.
 - Si la LED clignote en vert, le logging est en cours.
 - Si la LED s'allume en rouge, il y a une erreur (Vérifier carte SD ou relancer système).
3. Presser brièvement sur le bouton pour mettre la prochaine mesure en importance haute (1 dans CSV).
4. Une fois le logging terminé, maintenir le bouton plus de 5 secondes pour éteindre le système.

Notes Il est conseillé de bien éteindre le système par le bouton, pour ne pas mettre en danger l'intégrité de la carte SD.

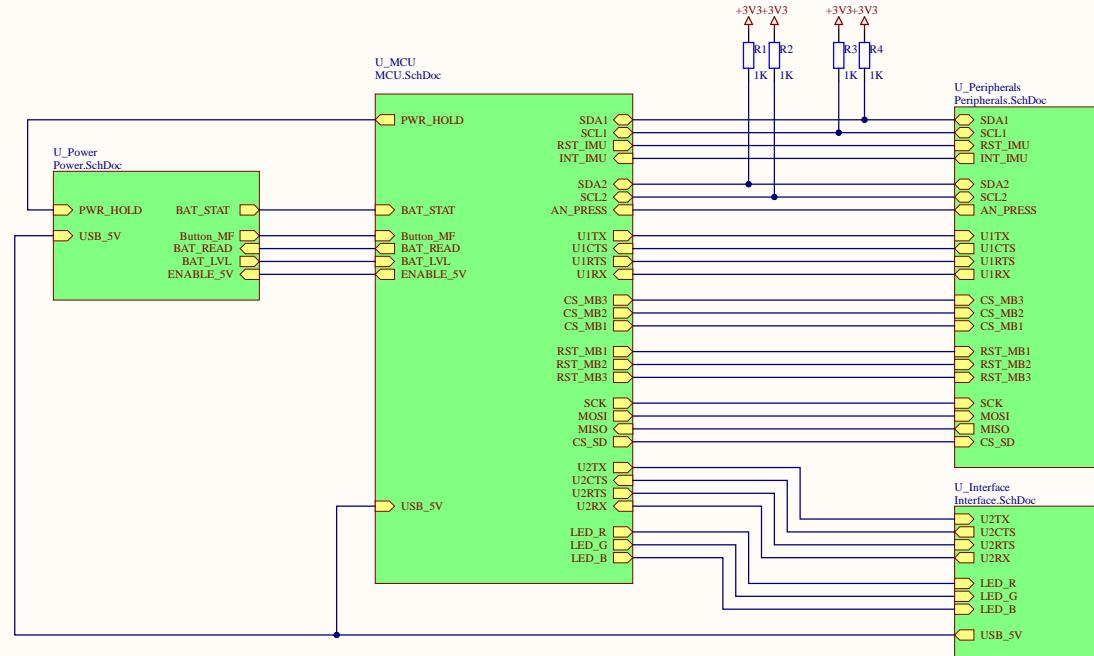
5. Retirer la carte SD.
6. Insérer la carte SD sur un ordinateur avec un adaptateur.
7. Récupérer le fichier *MESURES.CSV*.
8. Recommencer le cycle.

10.5 Schéma électrique

ETML-ES

16 juin 2023

Localisation sous-marine V0.0



Fichier : SchemaBloc.SchDoc

Date : 08.02.2023

Heure : 14:08:45

Auteur : Ali Zoubir



Projet : 2221_LocalisationSousMarine.PrjPcb

No : 2221

Nb feuilles : 6

Feuille n° : 1

Chemin : C:\Users\alizoubir\Documents\ETML-ES-2eme\PROJ\2221_LocalisationSousMarine\hard\2221_LocalisationSousMarine\Schematic.Sch

56

A

B

C

D

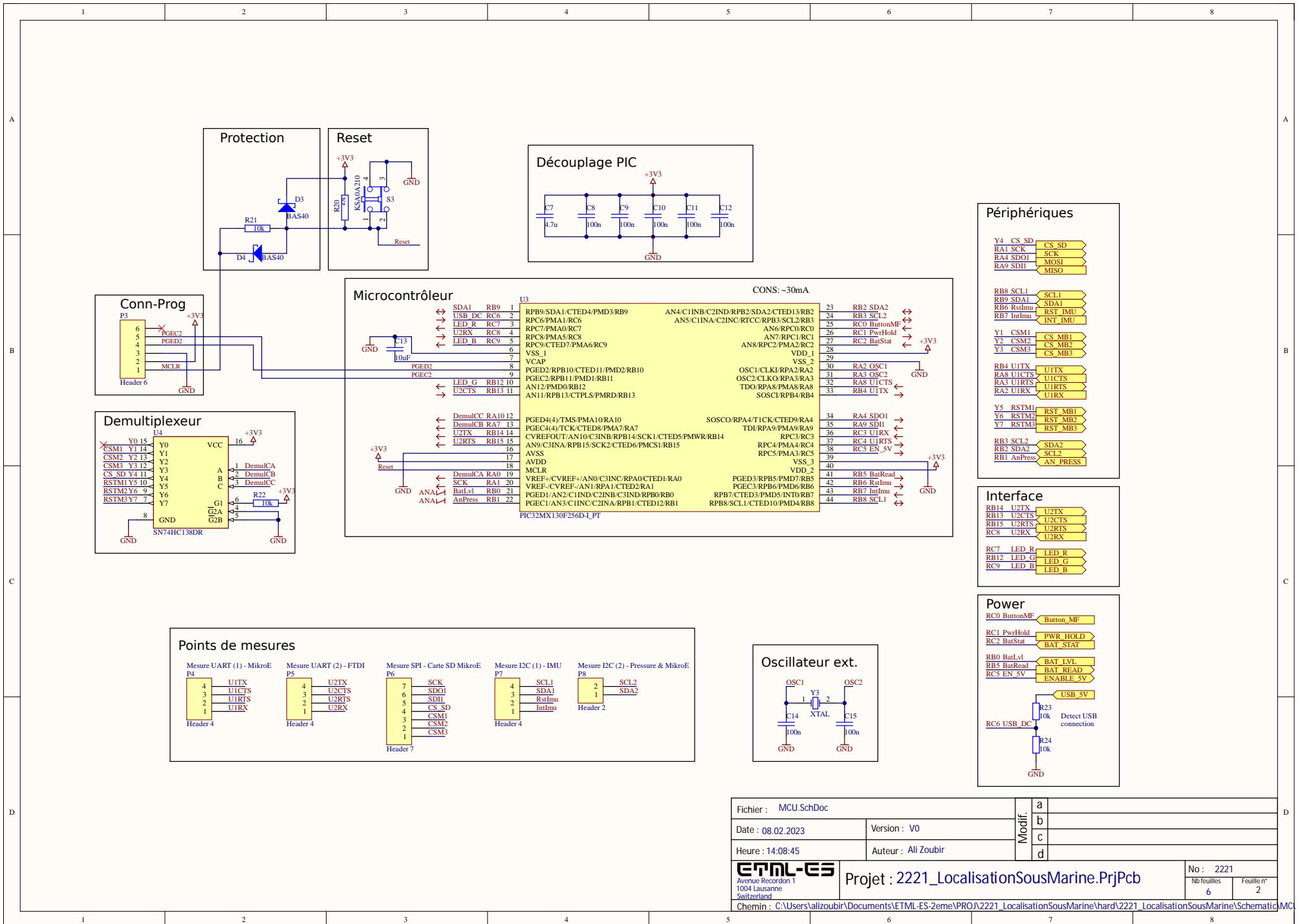
16 juin 2023

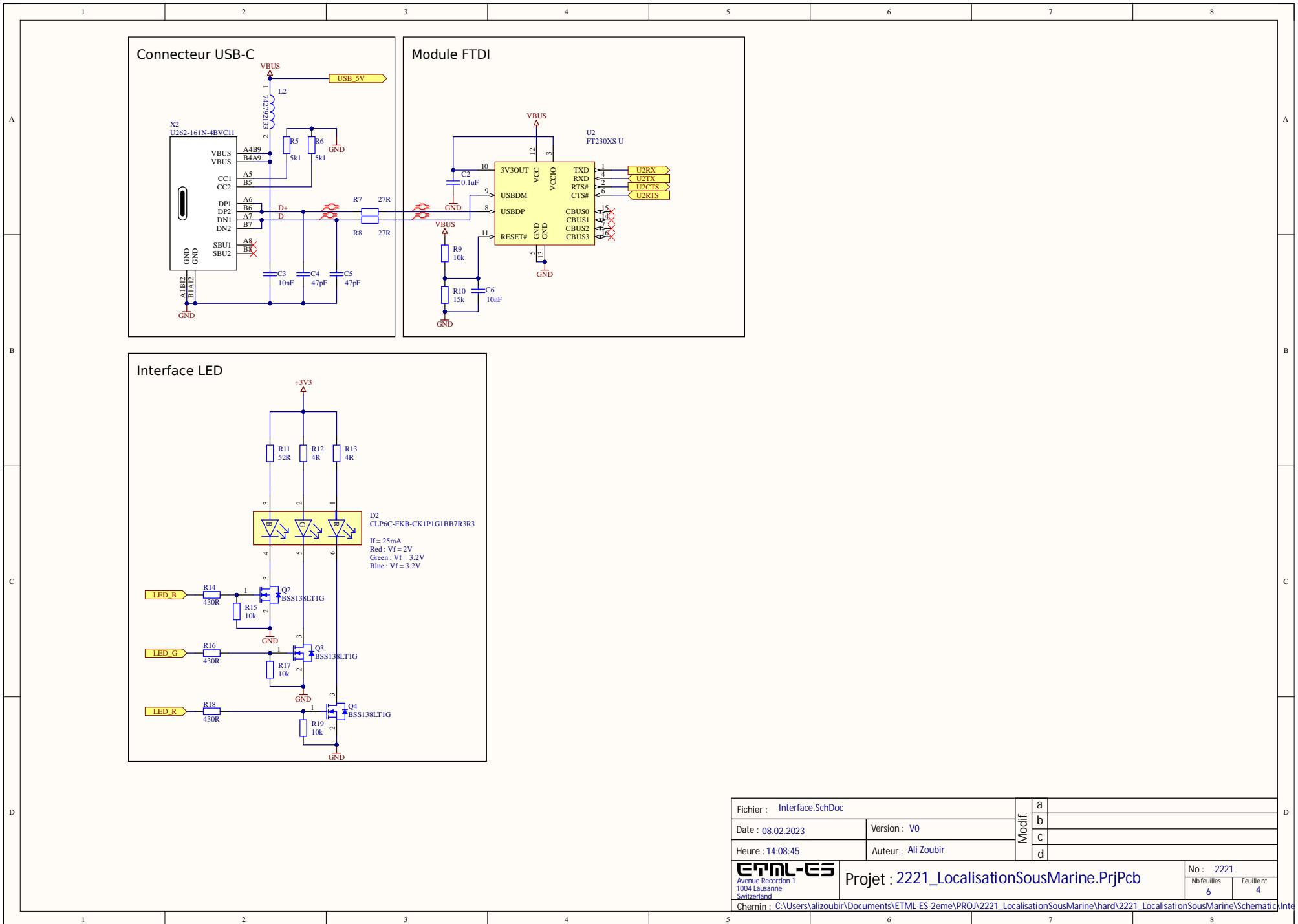
Localisation sous-marine V0.0

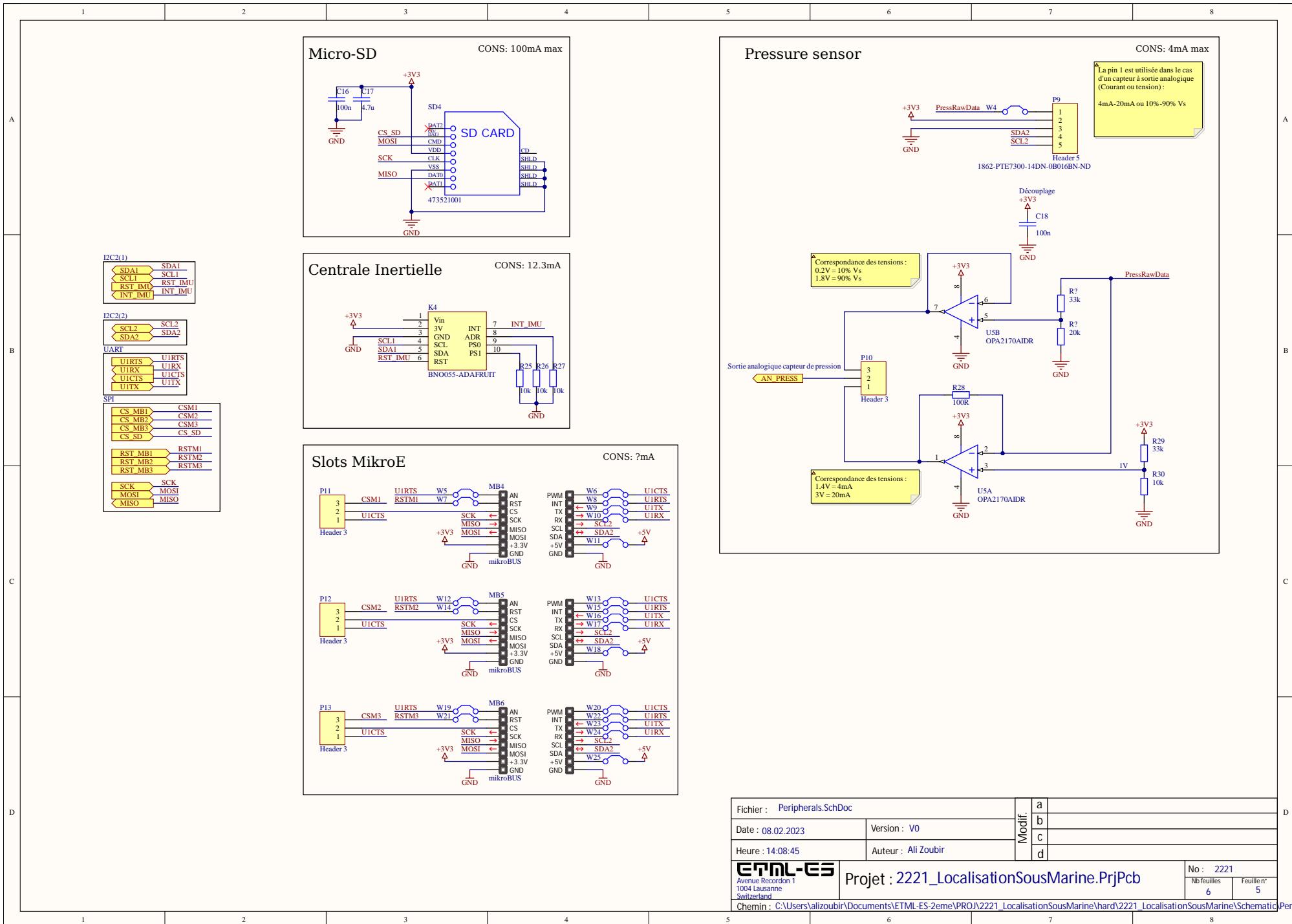
1 2 3 4 5 6 7 8

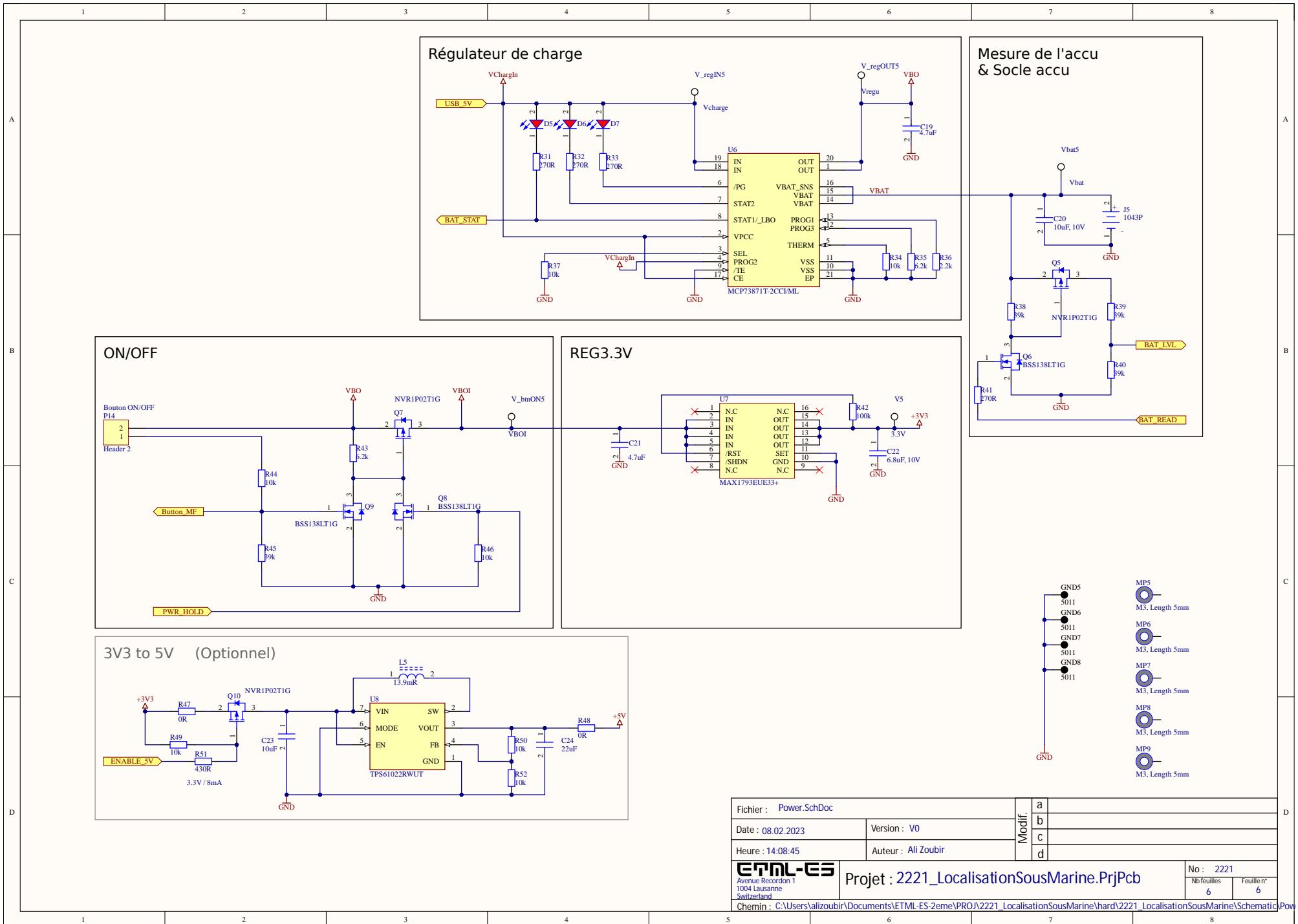
ETML-ES

1 2 3 4 5 6 7 8







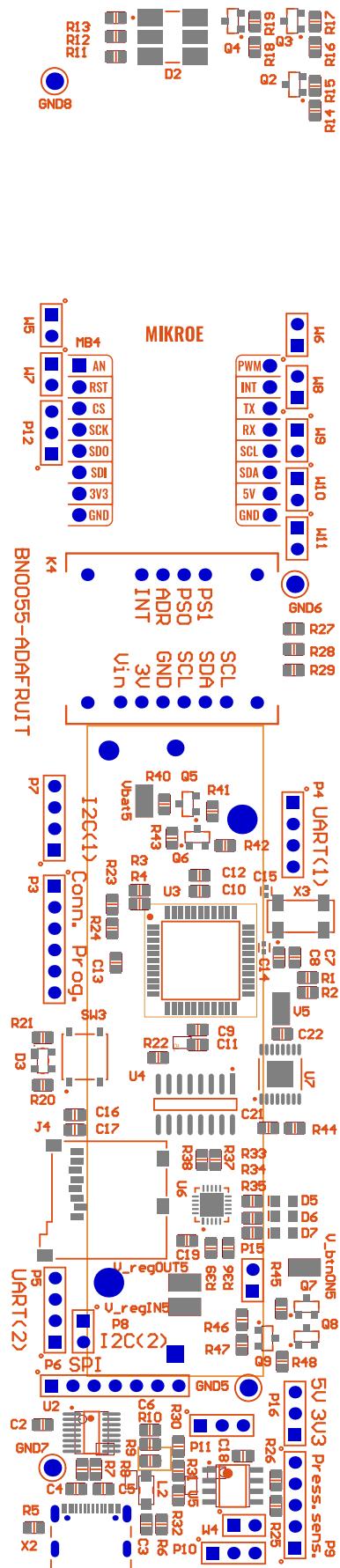


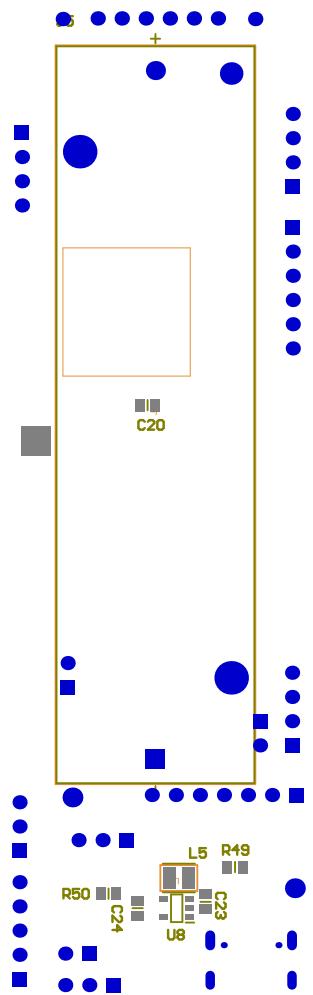
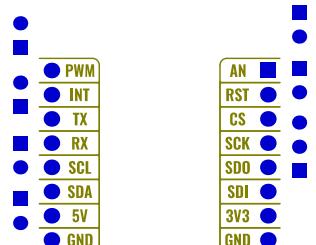
10.6 Bill of materials

Comment	Description	Designator	Footprint	LibRef	Quantity
0.1uF	Capacitor	C1	6-0805_L	Cap	1
10nF	Capacitor	C2, C5	6-0805_L	Cap	2
47pF	Capacitor	C3, C4	6-0805_L	Cap	2
4.7u		C6, C16	6-0805_L	C	2
100n		C7, C8, C9, C10, C11, C13, C14, C15	6-0805_L	C	8
10uF		C12	6-0805_L	C	1
4.7uF, 35V	Multilayer Ceramic Capacitors 4.7uF ±10% 35V X7R SMD 0805	C17, C19	6-0805_L	CMP-08246-002735-1	2
10uF, 10V	CAP CER 10UF 10V X5R 0805	C18	6-0805_L	CMP-2007-03243-2	1
6.8uF, 10V	Cap Ceramic 6.8uF 10V X5R 20% Pad SMD 0805 +85°C Automotive T/R	C20	6-0805_L	CMP-08246-000118-1	1
C0805C106K8PACT_U	CAP CER 10UF 10V X5R 0805	C21	FP-C0805C-DG-MFG	CMP-2007-03243-2	1
GRM21BR61A226 ME44L	Chip Multilayer Ceramic Capacitors for General Purpose, 0805, 22uF, X5R, 15%, 20%, 10V	C22	FP-GRM21B-0_2-e0_2_0-7-IPC_C	CMP-06035-039918-1	1
CLP6C-FKB-CK1P1G1BB7R3R3	LED RGB DIFFUSED 6PLCC SMD	D1	FP-CLP6C-FKB-CK1P1G1BB7R3-MFG	CMP-13977-000020-2	1
BAS40		D2, D3	BAS40	BAS40_0	2
Super Red, 1.9 V	SMT Mono-color Chip LED Waterclear WL-SMCW, size 0805, Super Red, 1.9 V, 140 deg, 60 mcd	D4, D5, D6	WL-SMCW_0805	150080SS75000	3
5011	Test Point, Black, Through Hole, RoHS, Bulk	GND1, GND2, GND3, GND4	KSTN-5011_V	CMP-1672-00003-4	4
1043P	BATT HOLDER 18650 1 CELL PC PIN	J1	FP-1043P-MFG	CMP-19636-000034-1	1
BNO055-ADAFRUIT		K1		BNO055-ADAFRUIT	1
742792133	SMD EMI Suppression Ferrite Bead WE-CBF, Z = 600 Ohm	L1	SMD-1206	CMP-0220-00010-1	1
SRN6045-1ROY	FIXED IND 1UH 4.2A 13.9 MOHM SMD	L2	FP-SRN6045-MFG	CMP-07248-000038-1	1
M3, Length 5mm	WA-SMSI SMT Steel Spacer, M3 Thread Internal, Length 5mm	MP1, MP2, MP3, MP4, MP5	troueAbraser	9774050360	5
Header 6	Header, 6-Pin	P1	HDR1X6	Header 6	1
Header 4	Header, 4-Pin	P2, P3, P5	HDR1X4	Header 4	3
Header 7	Header, 7-Pin	P4	HDR1X7	Header 7	1
Header 2	Header, 2-Pin	P6, P11	HDR1X2	Header 2	2
Header 5	Header, 5-Pin	P7	HDR1X5	Header 5	1
Header 3	Header, 3-Pin	P8, P9, P10	HDR1X3	Header 3	3
BSS138LT1G	Power MOSFET, -200 mA, 50 V, N-Channel, 3-Pin SOT-23, Pb-Free, Tape and Reel	Q1, Q2, Q3, Q5, Q7, Q8	ONSC-SOT-23-3-318-08_V	CMP-1058-00767-1	6
NVR1P02T1G	Power MOSFET, -20 V, -1 A, P-Channel, 3-Pin SOT-23, Pb-Free, Tape and Reel	Q4, Q6, Q9	ONSC-SOT-23-3-318-08_V	CMP-1058-00745-1	3
Resistor		R1, R2, R3, R4	AXIAL-0.4	Res2	4
Res2	Resistor	R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15, R16, R17, R18, R19, R25, R26, R27, R44	6-0805_L	Res2	19
47k		R20	6-0805_L	R	1
10k		R21, R22, R23, R24, R31, R34, R41, R43, R46, R47, R49	6-0805_L	R, CMP-2001-04376-1, CMP-2001-00410-1	11
270R		R28, R29, R30, R38	6-0805_L	CMP-2001-00528-1	4
6.2k		R32, R40	6-0805_L	CMP-2001-04031-1, CMP-2001-00562-1	2

2.2k		R33	6-0805_L	CMP-1013-00510-2	1
39k		R35, R36, R37, R42	6-0805_L	CMP-2001-00410-1	4
100k		R39	6-0805_L	CMP-2001-00479-1	1
Res2	Resistor	R45	AXIAL-0.4	Res2	1
430R		R48	6-0805_L	CMP-2001-00410-1	1
KSA0A210		S1	6x6_mm_SMD_W S-TASV	KSA0A210	1
473521001	MicroSD(TM), Pitch 1.1 mm, 8 Position, Height 1.88 mm, -20 to 85 degC, RoHS, Tape and Reel	SD1	MOLX- 473521001_V	CMP-2000-05848-1	1
FT230XS-U	USB to Basic UART Interface Chip, UHCI/OHCI/EHCI Compatible, USB 2.0 Compatible, - 40 to +85 degC, 16- Pin SSOP, Pb- Free, Tube	U1	SSOP-16_N	CMP-0248-00025-2	1
PIC32MX130F256D I_PT	Microchip PIC32MX130F256D I/PT, 32bit PIC Microcontroller, 40MHz, 256 kB Flash, 44-Pin TQFP	U2	QFP80P1200X1200 X120-44N	PIC32MX130F256D I_PT	1
SN74HC138DR	IC 3-8 LINE DECODE/DEMUX 16 SOIC	U3	FP-D0016A-IPC_A	CMP-1633-00238-3	1
MCP73871T- 2CCI/ML	Charger IC Lithium Ion/Polymer 20- QFN _4x4_	U4	Microchip_C04- 126_IPC_B	MCP73871T- 2CCI/ML	1
MAX1793EUE33+		U5	TSOP65P640X110- HS-17N	MAX1793EUE33+	1
TPS61022RWUT	POWER MANAGEMENT	U6	FP-RWU0007A- MFG	CMP-04918- 000300-1	1
3.3V	Test Point, 1 Position SMD, RoHS, Tape and Reel	V1	KSTN-5019_V	CMP-1672-00008-1	1
VBOI	Test Point, 1 Position SMD, RoHS, Tape and Reel	V_btnON1	KSTN-5019_V	CMP-1672-00008-1	1
Vcharge	Test Point, 1 Position SMD, RoHS, Tape and Reel	V_regIN1	KSTN-5019_V	CMP-1672-00008-1	1
Vregu	Test Point, 1 Position SMD, RoHS, Tape and Reel	V_regOUT1	KSTN-5019_V	CMP-1672-00008-1	1
Vbat	Test Point, 1 Position SMD, RoHS, Tape and Reel	Vbat1	KSTN-5019_V	CMP-1672-00008-1	1
Jumper	Jumper Wire	W1, W2, W3, W4, W5, W6, W7, W8, W9, W10, W11, W12, W13, W14, W15, W16, W17, W18, W19, W20, W21	RAD-0.2	Jumper	21
U262-161N- 4BVC11	U262-161N- 4BVC11 XKB Connectivity	X1	GCT_USB4105-GF- A	USB-C Type 2.0	1
XTAL	Crystal Oscillator	Y1	R38	XTAL	1

10.7 Implémentation





10.8 Code

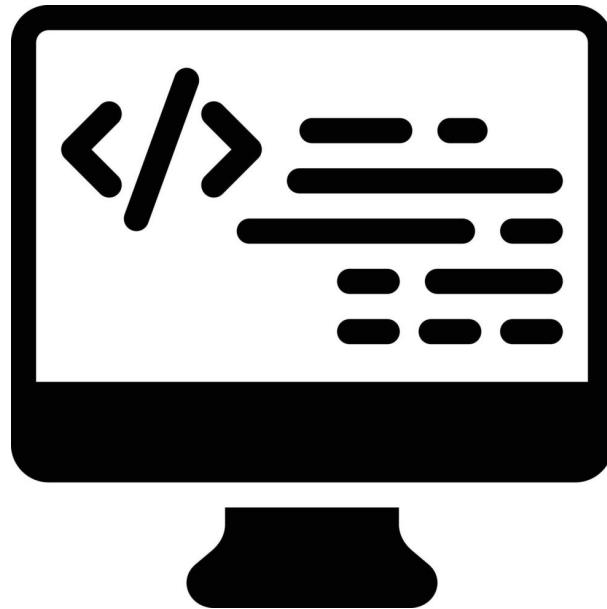


FIGURE 50 – Illustration code

Source : <https://www.vecteezy.com/vector-art/21464998-screen-coding-vector-illustration-on-a-background-premium-quality-symbols-vector-icons-for-concept-and-graphic-design>

```

1 ****
2 MPLAB Harmony Application Source File
3
4 Company:
5 Microchip Technology Inc.
6
7 File Name:
8 app.c
9
10 Summary:
11 This file contains the source code for the MPLAB Harmony application.
12
13 Description:
14 This file contains the source code for the MPLAB Harmony application. It
15 implements the logic of the application's state machine and it may call
16 API routines of other MPLAB Harmony modules in the system, such as drivers,
17 system services, and middleware. However, it does not call any of the
18 system interfaces (such as the "Initialize" and "Tasks" functions) of any of
19 the modules in the system or make any assumptions about when those functions
20 are called. That is the responsibility of the configuration-specific system
21 files.
22 ****/
23
24 // DOM-IGNORE-BEGIN
25 ****
26 Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.
27
28 Microchip licenses to you the right to use, modify, copy and distribute
29 Software only when embedded on a Microchip microcontroller or digital signal
30 controller that is integrated into your product or third party product
31 (pursuant to the sublicense terms in the accompanying license agreement).
32
33 You should refer to the license agreement accompanying this Software for
34 additional information regarding your rights and obligations.
35
36 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
37 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
38 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
39 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
40 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
41 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
42 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
43 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
44 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
45 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
46 ****/
47 // DOM-IGNORE-END
48
49
50 // ****
51 // ****
52 // Section: Included Files
53 // ****
54 // ****
55
56 #include "app.h"
57 #include "bno055.h"
58 #include "bno055_support.h"
59 #include "Mc32_I2cUtilCCS.h"
60 #include "Mc32_serComm.h"
61 #include "Mc32_sdFatGest.h"
62 #include "Mc32_PressAdc.h"
63 #include "Mc32Debounce.h"
64 #include <stdio.h>
65
66 // ****
67 // ****
68 // Section: Global Data Definitions
69 // ****
70 // ****
71 /* Switch descriptor */
72 S_SwitchDescriptor switchDescr;

```

```

73 // ****
74 /* Application Data
75
76 Summary:
77   Holds application data
78
79 Description:
80   This structure holds the application's data.
81
82 Remarks:
83   This structure should be initialized by the APP_Initialize function.
84
85 Application strings and buffers are be defined outside this structure.
86 */
87
88 APP_DATA appData;
89 TIMER_DATA timerData;
90
91 // ****
92 // ****
93 // Section: Application Callback Functions
94 // ****
95 // ****
96 void MainTimer_callback(){
97   /* Increment delay timer */
98   timerData.TmrCnt++;
99 }
100
101 void DisplayTimer_callback()
102 {
103   /* Increment utility timers */
104   timerData.TmrDisplay++;
105   timerData.TmrMeas++;
106   timerData.TmrTickFlag = true;
107   /* If button is pressed, count pressed time */
108   if(timerData.flagCountBtnPressed){
109     timerData.TmrBtnPressed++;
110   }
111   /* Do debounce every 10 ms */
112   DoDebounce(&switchDescr, ButtonMFStateGet());
113   /* Start a measure set each 90ms */
114   if( ( timerData.TmrMeas % 9 ) == 0 )
115     timerData.measTodoFlag = true;
116 }
117 /* TODO: Add any necessary callback functions.
118 */
119
120 // ****
121 // ****
122 // Section: Application Local Functions
123 // ****
124 // ****
125
126
127 /* TODO: Add any necessary local functions.
128 */
129
130
131 // ****
132 // ****
133 // Section: Application Initialization and State Machine Functions
134 // ****
135 // ****
136
137 // ****
138 Function:
139   void APP_Initialize ( void )
140
141 Remarks:
142   See prototype in app.h.
143 */
144
145 void APP_Initialize ( void )
146 {

```

```

147 /* Place the App state machine in its initial state.*/
148 appData.state = APP_STATE_INIT;
149 /* Init all counters and flags */
150 timerData.mainTmrCnt = 0;
151 timerData.TmrCnt = 0;
152 timerData.TmrTickFlag = false;
153 timerData.TmrDisplay = 0;
154 timerData.measTodoFlag = false;
155 timerData.flagCountBtnPressed = false;
156 timerData.TmrBtnPressed = 0;
157
158 /* Hold the device on */
159 PwrHoldOn();
160 /* Peripherals init */
161 DRV_TMR0_Start();
162 DRV_TMR1_Start();
163 i2c_init(1);
164 Press_InitADC();
165
166 /* System ON display */
167 LED_BOn();
168 BNO055_delay_msek(500);
169 LED_BOff();
170
171 /* Reset IMU */
172 RstImuOff();
173 BNO055_delay_msek(100);
174 RstImuOn();
175 BNO055_delay_msek(100);
176
177 /* Demultiplexer config */
178 DemulCBOff();
179 DemulCCOn();
180
181 /* Enable 5V regulator */
182 EN_5VOn();
183
184
185 }
186
187 ****
188 Function:
189 void APP_Tasks ( void )
190
191 Remarks:
192 See prototype in app.h.
193 */
194
195
196 void APP_Tasks ( void )
197 {
198 /* Local bno055 data */
199 s_bno055_data bno055_local_data;
200 static bool Hold = false;
201 static uint8_t flagMeas = false;
202 /* Check the application's current state.*/
203 switch ( appData.state )
204 {
205 /* Application's initial state.*/
206 case APP_STATE_INIT:
207 {
208 // Init delay
209 BNO055_delay_msek(500);
210 // Init and Measure set
211 bno055_init_readout();
212 /* go to service task */
213 appData.state = APP_STATE_LOGGING;
214 /* Init ltime counter */
215 timerData.ltime = 0;
216 /* Init first measure flag */
217 flagMeas = FLAG_MEAS_OFF;
218 break;
219 }
220
221 case APP_STATE_LOGGING:

```

```

222 {
223     /* Display period */
224     if(timerData.TmrDisplay >= 320)
225         timerData.TmrDisplay = 0;
226     // --- Display LED ---
227     if((timerData.TmrDisplay <= 1)&&(sd_getState() != APP_MOUNT_DISK))
228         LED_GOn();
229     else
230         LED_GOff();
231
232     if((timerData.measTodoFlag == true )&&(sd_getState() == APP_IDLE))
233     {
234         /* BNO055 Read all important info routine */
235         bno055_local_data.comres = bno055_read_routine(&bno055_local_data);
236         /* Delta time */
237         bno055_local_data.d_time = timerData.TmrMeas - timerData.ltime;
238         /* Pressure measure */
239         bno055_local_data.pressure = Press_readPressure();
240         /* Flag measure value */
241         bno055_local_data.flagImportantMeas = flagMeas;
242         /* Display value via UART */
243         //serDisplayValues(&bno055_local_data);
244         /* Write value to sdCard */
245         sd_BNO_scheduleWrite(&bno055_local_data);
246         /* Reset measure flag */
247         if(flagMeas == FLAG_MEAS_ON){
248             /* Rest important measure flag */
249             flagMeas = FLAG_MEAS_OFF;
250             LED_BOff();
251         }
252         /* Reset measure flag */
253         timerData.measTodoFlag = false;
254         /* Update last time counter */
255         timerData.ltime = timerData.TmrMeas;
256     }
257     else
258     {
259         /* No comm, so no error */
260         bno055_local_data.comres = 0;
261     }
262
263     /* If error detected : error LED */
264     if((bno055_local_data.comres != 0)|| (sd_getState() == APP_MOUNT_DISK))
265         LED_ROn();
266     else
267         LED_ROff();
268
269     /* --- SD FAT routine --- */
270     sd_fat_task();
271
272     /* Button management : if rising edge detected */
273     if((ButtonMFStateGet())||(Hold == true))
274     {
275         /* Hold until falling edge */
276         Hold = true;
277         /* Start counting pressed time */
278         timerData.flagCountBtnPressed = true;
279         /* If falling edge detected */
280         if (ButtonMFStateGet() == 0)
281         {
282             /* Reset flag and switchdescr */
283             timerData.flagCountBtnPressed = false;
284             DebounceClearReleased(&switchDescr);
285             /* If pressed time less than power off time */
286             if((timerData.TmrBtnPressed <= TIME_POWER_OFF)&&(sd_getState() != APP_MOUNT_DISK)){
287                 flagMeas = FLAG_MEAS_ON;
288                 LED_BOn();
289             }
290             else{
291                 /* Power off the system */
292                 appData.state = APP_STATE_SHUTDOWN;
293             }
294             timerData.TmrBtnPressed = 0;
295             Hold = false;
296         }

```

```

297     }
298
299     break;
300 }
301 case APP_STATE_SHUTDOWN:
302 {
303     /* Display shutting off mode */
304     LED_BOff();
305     LED_GOff();
306     LED_ROn();
307
308     /* If and SD card is mounted */
309     if(sd_getState() != APP_MOUNT_DISK){
310         /* Wait until SD available */
311         while(sd_getState() != APP_IDLE){
312             /* SD FAT routine */
313             sd_fat_task();
314         }
315         /* Unmount disk */
316         sd_setState(APP_UNMOUNT_DISK);
317         /* Wait until unmounted*/
318         while(sd_getState() != APP_IDLE){
319             sd_fat_task();
320         }
321     }
322
323     /* turn off the device */
324     PwrHoldOff();
325
326     break;
327 }
328
329 /* TODO: implement your application state machine.*/
330
331
332 /* The default state should never be executed.*/
333 default:
334 {
335     /* TODO: Handle error in application's state machine.*/
336     break;
337 }
338 }
339 }
340
341 void App_resetMeasFlag( void )
342 {
343     timerData.measTodoFlag = false;
344 }
345
346
347 ****
348 End of File
349 */
350

```

```
1 //*****
2 // MPLAB Harmony Application Header File
3 //
4 // Company:
5 // Microchip Technology Inc.
6 //
7 // File Name:
8 // app.h
9 //
10 // Summary:
11 // This header file provides prototypes and definitions for the application.
12 //
13 // Description:
14 // This header file provides function prototypes and data type definitions for
15 // the application. Some of these are required by the system (such as the
16 // "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
17 // internally by the application (such as the "APP_STATES" definition). Both
18 // are defined here for convenience.
19 *****/
20
21 //DOM-IGNORE-BEGIN
22 *****
23 Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.
24
25 Microchip licenses to you the right to use, modify, copy and distribute
26 Software only when embedded on a Microchip microcontroller or digital signal
27 controller that is integrated into your product or third party product
28 (pursuant to the sublicense terms in the accompanying license agreement).
29
30 You should refer to the license agreement accompanying this Software for
31 additional information regarding your rights and obligations.
32
33 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
34 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
35 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
36 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
37 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
38 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
39 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
40 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
41 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
42 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
43 *****/
44 //DOM-IGNORE-END
45
46 #ifndef _APP_H
47 #define _APP_H
48
49 // *****
50 // *****
51 // Section: Included Files
52 // *****
53 // *****
54
55 #include <stdint.h>
56 #include <stdbool.h>
57 #include <stddef.h>
58 #include <stdlib.h>
59 #include "system_config.h"
60 #include "system_definitions.h"
61 #include "bno055.h"
62
63 // DOM-IGNORE-BEGIN
64 #ifdef __cplusplus // Provide C++ Compatibility
65
66 extern "C" {
67
68 #endif
69 // DOM-IGNORE-END
70
71 #define TIME_OUT 80000000U
72 #define TIME_POWER_OFF 500
```

```

73
74 // ****
75 // ****
76 // Section: Type Definitions
77 // ****
78 // ****
79 typedef struct {
80     s32 comres;
81     bool flagMeasReady;
82     uint8_t flagImportantMeas;
83     struct bno055_gravity_double_t gravity;
84     struct bno055_linear_accel_double_t linear_accel;
85     struct bno055_euler_double_t euler;
86     struct bno055_gyro_double_t gyro;
87     struct bno055_mag_double_t mag;
88     struct bno055_quaternion_t quaternion;
89     unsigned long time;
90     unsigned long l_time;
91     uint16_t d_time;
92     float pressure;
93 }s_bno055_data;
94 // ****
95 /* Application states
96
97 Summary:
98 Application states enumeration
99
100 Description:
101 This enumeration defines the valid application states. These states
102 determine the behavior of the application at various times.
103 */
104
105 typedef enum
106 {
107     /* Application's state machine's initial state.*/
108     APP_STATE_INIT=0,
109     APP_STATE_LOGGING,
110     APP_STATE_FLAG_MEAS,
111     APP_STATE_SHUTDOWN
112     /* TODO: Define states used by the application state machine.*/
113
114 } APP_STATES;
115
116
117 // ****
118 /* Application Data
119
120 Summary:
121 Holds application data
122
123 Description:
124 This structure holds the application's data.
125
126 Remarks:
127 Application strings and buffers are be defined outside this structure.
128 */
129
130 typedef struct
131 {
132     /* The application's current state */
133     APP_STATES state;
134 } APP_DATA;
135
136 typedef struct
137 {
138     /* Main Timer (1ms) */
139     uint32_t mainTmrCnt;
140     /* Timer precis (1us) */
141     bool TmrTickFlag;
142     uint32_t TmrCnt;
143     /* Measure todo flag */
144     unsigned long TmrMeas;
145     unsigned long ltime;
146     bool measTodoFlag;
147     /* Timer display */

```

```
148     uint32_t TmrDisplay;
149     /* Tmr wait shutdown */
150     bool flagCountBtnPressed;
151     uint32_t TmrBtnPressed;
152 }TIMER_DATA;
153
154 // ****
155 // ****
156 // Section: Application Callback Routines
157 // ****
158 // ****
159 // ****
160 // ****
161 // Section: Application Initialization and State Machine Functions
162 // ****
163 // ****
164
165 // ****
166 Function:
167 void APP_Initialize ( void )
168
169 Summary:
170 MPLAB Harmony application initialization routine.
171
172 Description:
173 This function initializes the Harmony application. It places the
174 application in its initial state and prepares it to run so that its
175 APP_Tasks function can be called.
176
177 Precondition:
178 All other system initialization routines should be called before calling
179 this routine (in "SYS_Initialize").
180
181 Parameters:
182 None.
183
184 Returns:
185 None.
186
187 Example:
188 <code>
189 APP_Initialize();
190 </code>
191
192 Remarks:
193 This routine must be called from the SYS_Initialize function.
194 */
195
196 void APP_Initialize ( void );
197
198 void prepareBuffer( char * buffer );
199
200 void App_resetMeasFlag( void );
201
202 // ****
203 Function:
204 void APP_Tasks ( void )
205
206 Summary:
207 MPLAB Harmony Demo application tasks function
208
209 Description:
210 This routine is the Harmony Demo application's tasks function. It
211 defines the application's state machine and core logic.
212
213 Precondition:
214 The system and application initialization ("SYS_Initialize") should be
215 called before calling this.
216
217 Parameters:
218 None.
219
220 Returns:
221 None.
```

```
222
223 Example:
224 <code>
225 APP_Tasks();
226 </code>
227
228 Remarks:
229 This routine must be called from SYS_Tasks() routine.
230 */
231
232 void APP_Tasks( void );
233
234
235 // Callback main timer
236 void MainTimer_callback( void );
237
238 // Callback display timer
239 void DisplayTimer_callback( void );
240
241 #endif /* _APP_H */
242
243 //DOM-IGNORE-BEGIN
244 #ifdef __cplusplus
245 }
246 #endif
247 //DOM-IGNORE-END
248
249 ****
250 End of File
251 */
252
253
```

```

1 /**
2 * Copyright (c) 2020 Bosch Sensortec GmbH. All rights reserved.
3 *
4 * BSD-3-Clause
5 *
6 * Redistribution and use in source and binary forms, with or without
7 * modification, are permitted provided that the following conditions are met:
8 *
9 * 1. Redistributions of source code must retain the above copyright
10 * notice, this list of conditions and the following disclaimer.
11 *
12 * 2. Redistributions in binary form must reproduce the above copyright
13 * notice, this list of conditions and the following disclaimer in the
14 * documentation and/or other materials provided with the distribution.
15 *
16 * 3. Neither the name of the copyright holder nor the names of its
17 * contributors may be used to endorse or promote products derived from
18 * this software without specific prior written permission.
19 *
20 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
21 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
22 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
23 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
24 * COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
25 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
26 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
27 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
28 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
29 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
30 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
31 * POSSIBILITY OF SUCH DAMAGE.
32 *
33 * @file bno055_support.c
34 * @date 10/01/2020
35 * @version 2.0.6
36 *
37 */
38
39 /*-----*
40 * Includes
41 *-----*/
42 #include "app.h"
43 #include "bno055.h"
44 #include "bno055_support.h"
45 #include "Mc32_I2cUtilCCS.h"
46 #include "driver/tmr/drv_tmr_static.h"
47
48 // Global variable
49 TIMER_DATA timerData;
50
51 #ifdef BNO055_API
52
53 s32 bno055_read_routine(s_bno055_data *data)
54 {
55     /* Variable used to return value of
56     * communication routine*/
57     s32 comres = BNO055_ERROR;
58
59     /* variable used to set the power mode of the sensor*/
60     //u8 power_mode = BNO055_INIT_VALUE;
61
62     /* For initializing the BNO sensor it is required to the operation mode
63     * of the sensor as NORMAL
64     * Normal mode can set from the register
65     * Page - page0
66     * register - 0x3E
67     * bit positions - 0 and 1*/
68     //power_mode = BNO055_POWER_MODE_NORMAL;
69
70     /* set the power mode as NORMAL*/
71     //comres += bno055_set_power_mode(power_mode);
72

```

```

73 /*-----*
74 ***** END INITIALIZATION *****
75 *-----*/
76
77 //***** START READ RAW FUSION DATA *****
78 * For reading fusion data it is required to set the
79 * operation modes of the sensor
80 * operation mode can set from the register
81 * page - page0
82 * register - 0x3D
83 * bit - 0 to 3
84 * for sensor data read following operation mode have to set
85 * FUSION MODE
86 * 0x08 - BNO055_OPERATION_MODE_IMUPLUS
87 * 0x09 - BNO055_OPERATION_MODE_COMPASS
88 * 0x0A - BNO055_OPERATION_MODE_M4G
89 * 0x0B - BNO055_OPERATION_MODE_NDOF_FMC_OFF
90 * 0x0C - BNO055_OPERATION_MODE_NDOF
91 * based on the user need configure the operation mode*/
92 //comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_NDOF);
93
94 /* Raw Quaternion W, X, Y and Z data can read from the register
95 * page - page 0
96 * register - 0x20 to 0x27 */
97 comres += bno055_read_quaternion_wxyz(&data->quaternion);
98 //***** END READ RAW FUSION DATA *****
99 //*****START READ CONVERTED SENSOR DATA*****
100 /* API used to read mag data output as double - uT(micro Tesla)
101 * float functions also available in the BNO055 API */
102 comres += bno055_convert_double_mag_xyz_uT(&data->mag);
103 /* API used to read gyro data output as double - dps and rps
104 * float functions also available in the BNO055 API */
105 comres += bno055_convert_double_gyro_xyz_dps(&data->gyro);
106 /* API used to read Euler data output as double - degree and radians
107 * float functions also available in the BNO055 API */
108 comres += bno055_convert_double_euler_hpr_deg(&data->euler);
109 /* API used to read Linear acceleration data output as m/s2
110 * float functions also available in the BNO055 API */
111 comres += bno055_convert_double_linear_accel_xyz_msq(&data->linear_accel);
112 comres += bno055_convert_double_gravity_xyz_msq(&data->gravity);
113
114 /*-----*
115 ***** START DE-INITIALIZATION *****
116 *-----*/
117
118 /* For de - initializing the BNO sensor it is required
119 * to the operation mode of the sensor as SUSPEND
120 * Suspend mode can set from the register
121 * Page - page0
122 * register - 0x3E
123 * bit positions - 0 and 1*/
124 //power_mode = BNO055_POWER_MODE_SUSPEND;
125
126 /* set the power mode as SUSPEND*/
127 //comres += bno055_set_power_mode(power_mode);
128
129 /* Flag measure ready */
130 data->flagMeasReady = true;
131
132 /*-----*
133 ***** END DE-INITIALIZATION *****
134 *-----*/
135 return (comres+1);
136 }
137
138 /*
139 * The following API is used to map the I2C bus read, write, delay and
140 * device address with global structure bno055_t
141 */
142
143 /*
144 * By using bno055 the following structure parameter can be accessed
145 * Bus write function pointer: BNO055_WR_FUNC_PTR
146 * Bus read function pointer: BNO055_RD_FUNC_PTR
147 * Delay function pointer: delay_msec

```

```

148 * I2C address: dev_addr
149 *-----*/
150 s8 I2C_routine(void)
151 {
152     bno055.bus_write = BNO055_I2C_bus_write;
153     bno055.bus_read = BNO055_I2C_bus_read;
154     bno055.delay_msec = BNO055_delay_msek;
155     bno055.dev_addr = BNO055_I2C_ADDR1;
156     return BNO055_INIT_VALUE;
157 }
158
159 /****** I2C buffer length*****/
160
161 #define I2C_BUFFER_LEN 8
162 #define I2C0      5
163
164 /*-----*/
165 *
166 * This is a sample code for read and write the data by using I2C
167 * Use either I2C based on your need
168 * The device address defined in the bno055.h file
169 *
170 */
171
172 /* \Brief: The API is used as I2C bus write
173 * \Return : Status of the I2C write
174 * \param dev_addr : The device address of the sensor
175 * \param reg_addr : Address of the first register,
176 * will data is going to be written
177 * \param reg_data : It is a value hold in the array,
178 * will be used for write the value into the register
179 * \param cnt : The no of byte of data to be write
180 */
181 s8 BNO055_I2C_bus_write(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt)
182 {
183     s8 BNO055_iERROR = BNO055_INIT_VALUE;
184     u8 array[I2C_BUFFER_LEN];
185     u8 stringpos = BNO055_INIT_VALUE;
186
187     array[BNO055_INIT_VALUE] = reg_addr;
188
189     i2c_start();
190     BNO055_iERROR = i2c_write(dev_addr<<1);
191
192     for (stringpos = BNO055_INIT_VALUE; stringpos < (cnt+BNO055_I2C_BUS_WRITE_ARRAY_INDEX); stringpos++)
193     {
194         BNO055_iERROR = i2c_write(array[stringpos]);
195         array[stringpos + BNO055_I2C_BUS_WRITE_ARRAY_INDEX] = *(reg_data + stringpos);
196     }
197
198     i2c_stop();
199
200
201 /*
202 * Please take the below APIs as your reference for
203 * write the data using I2C communication
204 * "BNO055_iERROR = I2C_WRITE_STRING(DEV_ADDR, ARRAY, CNT+1)"
205 * add your I2C write APIs here
206 * BNO055_iERROR is an return value of I2C read API
207 * Please select your valid return value
208 * In the driver BNO055_SUCCESS defined as 0
209 * and FAILURE defined as -1
210 * Note :
211 * This is a full duplex operation,
212 * The first read data is discarded, for that extra write operation
213 * have to be initiated. For that cnt+1 operation done
214 * in the I2C write string function
215 * For more information please refer data sheet SPI communication:
216 */
217
218 /*if(BNO055_iERROR)
219     BNO055_iERROR = -1;
220 else
221     BNO055_iERROR = 0;
222
```

```

223     return (s8)(BNO055_iERROR);*/
224 // Error comm return
225
226 if(BNO055_iERROR-1 != 0)
227     BNO055_iERROR = -1;
228 else
229     BNO055_iERROR = 0;
230
231 return (s8)(BNO055_iERROR);
232 }
233
234 /* \Brief: The API is used as I2C bus read
235 * \Return : Status of the I2C read
236 * \param dev_addr : The device address of the sensor
237 * \param reg_addr : Address of the first register,
238 * will data is going to be read
239 * \param reg_data : This data read from the sensor,
240 * which is hold in an array
241 * \param cnt : The no of byte of data to be read
242 */
243 s8 BNO055_I2C_bus_read(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt)
244 {
245     s8 BNO055_iERROR = BNO055_INIT_VALUE;
246     u8 array[I2C_BUFFER_LEN] = { BNO055_INIT_VALUE };
247     u8 stringpos = BNO055_INIT_VALUE;
248
249     array[BNO055_INIT_VALUE] = reg_addr;
250
251     i2c_start();
252     // Write asked register
253     BNO055_iERROR = i2c_write(dev_addr<<1);
254     BNO055_iERROR = i2c_write(reg_addr);
255     // Send read address
256     i2c_reStart();
257     dev_addr = (dev_addr<<1) | 0b00000001;
258     BNO055_iERROR = i2c_write(dev_addr);
259
260     /* Please take the below API as your reference
261     * for read the data using I2C communication
262     * add your I2C read API here.
263     * "BNO055_iERROR = I2C_WRITE_READ_STRING(DEV_ADDR,
264     * ARRAY, ARRAY, 1, CNT)"
265     * BNO055_iERROR is an return value of SPI write API
266     * Please select your valid return value
267     * In the driver BNO055_SUCCESS defined as 0
268     * and FAILURE defined as -1
269     */
270     for (stringpos = BNO055_INIT_VALUE; stringpos < cnt; stringpos++)
271     {
272
273         if((stringpos+1) < cnt)&&(cnt > BNO055_I2C_BUS_WRITE_ARRAY_INDEX))
274             array[stringpos] = i2c_read(1);
275         else
276             array[stringpos] = i2c_read(0);
277
278         *(reg_data + stringpos) = array[stringpos];
279     }
280
281     i2c_stop();
282
283     // Error comm return
284     if(BNO055_iERROR-1 != 0)
285         BNO055_iERROR = -1;
286     else
287         BNO055_iERROR = 0;
288
289     return (s8)(BNO055_iERROR);
290 }
291
292
293 /* Brief : The delay routine
294 * \param : delay in ms
295 */
296 void BNO055_delay_msek(u32 msek)
297 {

```

```

298 /*Delay routine*/
299 DRV_TMR0_Stop();
300 DRV_TMR0_CounterClear();
301 timerData.TmrCnt = 0;
302 DRV_TMR0_Start();
303 while (timerData.TmrCnt < msec)
304 {
305 DRV_TMR0_Stop();
306 }
307
308 #endif
309
310
311 s32 bno055_init_readout(void)
312 {
313 /* Variable used to return value of
314 * communication routine*/
315 s32 comres = BNO055_ERROR;
316
317 /* variable used to set the power mode of the sensor*/
318 u8 power_mode = BNO055_INIT_VALUE;
319
320
321 /* variable used to read the accel xyz data */
322 struct bno055_accel_t accel_xyz;
323
324 /*****read raw mag data******/
325 /* structure used to read the mag xyz data */
326 struct bno055_mag_t mag_xyz;
327
328 /*****read raw gyro data******/
329 /* structure used to read the gyro xyz data */
330 struct bno055_gyro_t gyro_xyz;
331
332 /*****read raw Euler data******/
333 /* structure used to read the euler hrp data */
334 struct bno055_euler_t euler_hrp;
335
336 /*****read raw quaternion data******/
337 /* structure used to read the quaternion wxyz data */
338 struct bno055_quaternion_t quaternion_wxyz;
339
340 /*****read raw linear acceleration data******/
341 /* structure used to read the linear accel xyz data */
342 struct bno055_linear_accel_t linear_acce_xyz;
343
344 /*****read raw gravity sensor data******/
345 /* structure used to read the gravity xyz data */
346 struct bno055_gravity_t gravity_xyz;
347
348 /*****read accel converted data******/
349 /* structure used to read the accel xyz data output as m/s2 or mg */
350 struct bno055_accel_double_t d_accel_xyz;
351
352 /*****read mag converted data******/
353 /* structure used to read the mag xyz data output as uT*/
354 struct bno055_mag_double_t d_mag_xyz;
355
356 /*****read gyro converted data******/
357 /* structure used to read the gyro xyz data output as dps or rps */
358 struct bno055_gyro_double_t d_gyro_xyz;
359
360 /*****read euler converted data******/
361 /* variable used to read the euler h data output
362 * as degree or radians*/
363 double d_euler_data_h = BNO055_INIT_VALUE;
364 /* variable used to read the euler r data output
365 * as degree or radians*/
366 double d_euler_data_r = BNO055_INIT_VALUE;
367 /* variable used to read the euler p data output
368 * as degree or radians*/
369 double d_euler_data_p = BNO055_INIT_VALUE;
370 /* structure used to read the euler hrp data output
371 * as as degree or radians */
372 struct bno055_euler_double_t d_euler_hpr;

```

```

373 ****read linear acceleration converted data*****
374 /* structure used to read the linear accel xyz data output as m/s2*/
375 struct bno055_linear_accel_double_t d_linear_accel_xyz;
376
377 ****Gravity converted data*****
378 /* structure used to read the gravity xyz data output as m/s2*/
379 struct bno055_gravity_double_t d_gravity_xyz;
380
381 /*
382 *-----*
383 **** START INITIALIZATION ****
384 *-----*/
385 #ifdef BNO055_API
386
387 /* Based on the user need configure I2C interface.
388 * It is example code to explain how to use the bno055 API*/
389 I2C_routine();
390 #endif
391
392 /*
393 * This API used to assign the value/reference of
394 * the following parameters
395 * I2C address
396 * Bus Write
397 * Bus read
398 * Chip id
399 * Page id
400 * Accel revision id
401 * Mag revision id
402 * Gyro revision id
403 * Boot loader revision id
404 * Software revision id
405 */
406 comres = bno055_init(&bno055);
407
408 /* For initializing the BNO sensor it is required to the operation mode
409 * of the sensor as NORMAL
410 * Normal mode can set from the register
411 * Page - page0
412 * register - 0x3E
413 * bit positions - 0 and 1*/
414 power_mode = BNO055_POWER_MODE_NORMAL;
415
416 /* set the power mode as NORMAL*/
417 comres += bno055_set_power_mode(power_mode);
418
419 /*
420 **** END INITIALIZATION ****
421 */
422
423 **** START READ RAW SENSOR DATA*****
424
425 /* Using BNO055 sensor we can read the following sensor data and
426 * virtual sensor data
427 * Sensor data:
428 * Accel
429 * Mag
430 * Gyro
431 * Virtual sensor data
432 * Euler
433 * Quaternion
434 * Linear acceleration
435 * Gravity sensor */
436
437 /* For reading sensor raw data it is required to set the
438 * operation modes of the sensor
439 * operation mode can set from the register
440 * page - page0
441 * register - 0x3D
442 * bit - 0 to 3
443 * for sensor data read following operation mode have to set
444 * SENSOR MODE
445 * 0x01 - BNO055_OPERATION_MODE_ACONLY
446 * 0x02 - BNO055_OPERATION_MODE_MAGONLY
447 * 0x03 - BNO055_OPERATION_MODE_GYRONLY

```

```

448 * 0x04 - BNO055_OPERATION_MODE_ACCMAG
449 * 0x05 - BNO055_OPERATION_MODE_ACCGYRO
450 * 0x06 - BNO055_OPERATION_MODE_MAGGYRO
451 * 0x07 - BNO055_OPERATION_MODE_AMG
452 * based on the user need configure the operation mode*/
453 comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_AMG);
454
455 /* Raw accel X, Y and Z data can read from the register
456 * page - page 0
457 * register - 0x08 to 0x0D*/
458 comres += bno055_read_accel_xyz(&accel_xyz);
459
460 /* Raw mag X, Y and Z data can read from the register
461 * page - page 0
462 * register - 0x0E to 0x13*/
463 comres += bno055_read_mag_xyz(&mag_xyz);
464
465 /* Raw gyro X, Y and Z data can read from the register
466 * page - page 0
467 * register - 0x14 to 0x19*/
468 comres += bno055_read_gyro_xyz(&gyro_xyz);
469
470 /***** END READ RAW SENSOR DATA *****/
471
472 /***** START READ RAW FUSION DATA *****/
473 * For reading fusion data it is required to set the
474 * operation modes of the sensor
475 * operation mode can set from the register
476 * page - page0
477 * register - 0x3D
478 * bit - 0 to 3
479 * for sensor data read following operation mode have to set
480 * FUSION MODE
481 * 0x08 - BNO055_OPERATION_MODE_IMUPLUS
482 * 0x09 - BNO055_OPERATION_MODE_COMPASS
483 * 0x0A - BNO055_OPERATION_MODE_M4G
484 * 0x0B - BNO055_OPERATION_MODE_NDOF_FMC_OFF
485 * 0x0C - BNO055_OPERATION_MODE_NDOF
486 * based on the user need configure the operation mode*/
487 comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_NDOF);
488
489 /* Raw Euler H, R and P data can read from the register
490 * page - page 0
491 * register - 0x1A to 0x1E */
492 //comres += bno055_read_euler_h(&euler_data_h);
493 //comres += bno055_read_euler_r(&euler_data_r);
494 //comres += bno055_read_euler_p(&euler_data_p);
495 comres += bno055_read_euler_hrp(&euler_hrp);
496
497 /* Raw Quaternion W, X, Y and Z data can read from the register
498 * page - page 0
499 * register - 0x20 to 0x27 */
500 //comres += bno055_read_quaternion_w(&quaternion_data_w);
501 //comres += bno055_read_quaternion_x(&quaternion_data_x);
502 //comres += bno055_read_quaternion_y(&quaternion_data_y);
503 //comres += bno055_read_quaternion_z(&quaternion_data_z);
504 comres += bno055_read_quaternion_wxyz(&quaternion_wxyz);
505
506 /* Raw Linear accel X, Y and Z data can read from the register
507 * page - page 0
508 * register - 0x28 to 0x2D */
509 //comres += bno055_read_linear_accel_x(&linear_accel_data_x);
510 //comres += bno055_read_linear_accel_y(&linear_accel_data_y);
511 //comres += bno055_read_linear_accel_z(&linear_accel_data_z);
512 comres += bno055_read_linear_accel_xyz(&linear_acce_xyz);
513
514 /* Raw Gravity sensor X, Y and Z data can read from the register
515 * page - page 0
516 * register - 0x2E to 0x33 */
517 //comres += bno055_read_gravity_x(&gravity_data_x);
518 //comres += bno055_read_gravity_y(&gravity_data_y);
519 //comres += bno055_read_gravity_z(&gravity_data_z);
520 comres += bno055_read_gravity_xyz(&gravity_xyz);
521
522 /***** END READ RAW FUSION DATA *****/

```

```

523 ****START READ CONVERTED SENSOR DATA*****
524
525 /* API used to read accel data output as double - m/s2 and mg
526 * float functions also available in the BNO055 API */
527 //comres += bno055_convert_double_accel_x_msq(&d_accel_datax);
528 //comres += bno055_convert_double_accel_x_mg(&d_accel_datax);
529 //comres += bno055_convert_double_accel_y_msq(&d_accel_datay);
530 //comres += bno055_convert_double_accel_y_mg(&d_accel_datay);
531 //comres += bno055_convert_double_accel_z_msq(&d_accel_dataz);
532 //comres += bno055_convert_double_accel_z_mg(&d_accel_dataz);
533 comres += bno055_convert_double_accel_xyz_msq(&d_accel_xyz);
534 comres += bno055_convert_double_accel_xyz_mg(&d_accel_xyz);
535
536 /* API used to read mag data output as double - uT(micro Tesla)
537 * float functions also available in the BNO055 API */
538 //comres += bno055_convert_double_mag_x_uT(&d_mag_datax);
539 //comres += bno055_convert_double_mag_y_uT(&d_mag_datay);
540 //comres += bno055_convert_double_mag_z_uT(&d_mag_dataz);
541 comres += bno055_convert_double_mag_xyz_uT(&d_mag_xyz);
542
543 /* API used to read gyro data output as double - dps and rps
544 * float functions also available in the BNO055 API */
545 //comres += bno055_convert_double_gyro_x_dps(&d_gyro_datax);
546 //comres += bno055_convert_double_gyro_y_dps(&d_gyro_datay);
547 //comres += bno055_convert_double_gyro_z_dps(&d_gyro_dataz);
548 //comres += bno055_convert_double_gyro_x_rps(&d_gyro_datax);
549 //comres += bno055_convert_double_gyro_y_rps(&d_gyro_datay);
550 //comres += bno055_convert_double_gyro_z_rps(&d_gyro_dataz);
551 comres += bno055_convert_double_gyro_xyz_dps(&d_gyro_xyz);
552 //comres += bno055_convert_double_gyro_xyz_rps(&d_gyro_xyz);
553
554 /* API used to read Euler data output as double - degree and radians
555 * float functions also available in the BNO055 API */
556 comres += bno055_convert_double_euler_h_deg(&d_euler_data_h);
557 comres += bno055_convert_double_euler_r_deg(&d_euler_data_r);
558 comres += bno055_convert_double_euler_p_deg(&d_euler_data_p);
559 //comres += bno055_convert_double_euler_h_rad(&d_euler_data_h);
560 //comres += bno055_convert_double_euler_r_rad(&d_euler_data_r);
561 //comres += bno055_convert_double_euler_p_rad(&d_euler_data_p);
562 comres += bno055_convert_double_euler_hpr_deg(&d_euler_hpr);
563 //comres += bno055_convert_double_euler_hpr_rad(&d_euler_hpr);
564
565 /* API used to read Linear acceleration data output as m/s2
566 * float functions also available in the BNO055 API */
567 //comres += bno055_convert_double_linear_accel_x_msq(&d_linear_accel_datax);
568 //comres += bno055_convert_double_linear_accel_y_msq(&d_linear_accel_datay);
569 //comres += bno055_convert_double_linear_accel_z_msq(&d_linear_accel_dataz);
570 comres += bno055_convert_double_linear_accel_xyz_msq(&d_linear_accel_xyz);
571
572 /* API used to read Gravity sensor data output as m/s2
573 * float functions also available in the BNO055 API */
574 //comres += bno055_convert_gravity_double_x_msq(&d_gravity_data_x);
575 //comres += bno055_convert_gravity_double_y_msq(&d_gravity_data_y);
576 //comres += bno055_convert_gravity_double_z_msq(&d_gravity_data_z);
577 comres += bno055_convert_double_gravity_xyz_msq(&d_gravity_xyz);
578
579 /*
580 ***** START DE-INITIALIZATION *****
581 */
582
583 /* For de - initializing the BNO sensor it is required
584 * to the operation mode of the sensor as SUSPEND
585 * Suspend mode can set from the register
586 * Page - page0
587 * register - 0x3E
588 * bit positions - 0 and 1*/
589 //power_mode = BNO055_POWER_MODE_SUSPEND;
590
591 /* set the power mode as SUSPEND*/
592 //comres += bno055_set_power_mode(power_mode);
593 comres += bno055_set_operation_mode(BNO055_OPERATION_MODE_NDOF);
594
595 /*
596 ***** END DE-INITIALIZATION *****
597 */
598 return comres;

```

598 }
599

```

1 /**
2 * Copyright (c) 2020 Bosch Sensortec GmbH. All rights reserved.
3 *
4 * BSD-3-Clause
5 *
6 * Redistribution and use in source and binary forms, with or without
7 * modification, are permitted provided that the following conditions are met:
8 *
9 * 1. Redistributions of source code must retain the above copyright
10 * notice, this list of conditions and the following disclaimer.
11 *
12 * 2. Redistributions in binary form must reproduce the above copyright
13 * notice, this list of conditions and the following disclaimer in the
14 * documentation and/or other materials provided with the distribution.
15 *
16 * 3. Neither the name of the copyright holder nor the names of its
17 * contributors may be used to endorse or promote products derived from
18 * this software without specific prior written permission.
19 *
20 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
21 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
22 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
23 * FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
24 * COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
25 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
26 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
27 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
28 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
29 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
30 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
31 * POSSIBILITY OF SUCH DAMAGE.
32 *
33 * @file bno055_support.c
34 * @date 10/01/2020
35 * @version 2.0.6
36 *
37 */
38
39 /**
40 * Includes
41 */
42 #include "bno055.h"
43
44 #define BNO055_API
45
46 #define FLAG_MEAS_ON 1
47 #define FLAG_MEAS_OFF 0
48 /**
49 * The following APIs are used for reading and writing of
50 * sensor data using I2C communication
51 */
52 #ifdef BNO055_API
53 #define BNO055_I2C_BUS_WRITE_ARRAY_INDEX ((u8)1)
54
55 /* \Brief: The API is used as I2C bus read
56 * \Return : Status of the I2C read
57 * \param dev_addr : The device address of the sensor
58 * \param reg_addr : Address of the first register,
59 * will data is going to be read
60 * \param reg_data : This data read from the sensor,
61 * which is hold in an array
62 * \param cnt : The no of byte of data to be read
63 */
64 s8 BNO055_I2C_bus_read(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt);
65
66 /* \Brief: The API is used as SPI bus write
67 * \Return : Status of the SPI write
68 * \param dev_addr : The device address of the sensor
69 * \param reg_addr : Address of the first register,
70 * will data is going to be written
71 * \param reg_data : It is a value hold in the array,

```

```

72 * will be used for write the value into the register
73 * \param cnt : The no of byte of data to be write
74 */
75 s8 BNO055_I2C_bus_write(u8 dev_addr, u8 reg_addr, u8 *reg_data, u8 cnt);
76
77 /*
78 * \Brief: I2C init routine
79 */
80 s8 I2C_routine(void);
81
82 /* Brief : The delay routine
83 * \param : delay in ms
84 */
85 void BNO055_delay_msek(u32 msek);
86
87 #endif
88
89 /*****End of I2C APIs declarations*****/
90
91 /* This API is an example for reading sensor data
92 * \param: None
93 * \return: communication result
94 */
95 s32 bno055_init_readout(void);
96
97 s32 bno055_read_routine(s_bno055_data *data);
98
99 -----
100 * struct bno055_t parameters can be accessed by using BNO055
101 * BNO055_t having the following parameters
102 * Bus write function pointer: BNO055_WR_FUNC_PTR
103 * Bus read function pointer: BNO055_RD_FUNC_PTR
104 * Burst read function pointer: BNO055_BRD_FUNC_PTR
105 * Delay function pointer: delay_msec
106 * I2C address: dev_addr
107 * Chip id of the sensor: chip_id
108 *
109 struct bno055_t bno055;
110

```

```

1 /* **** */
2 /** Descriptive File Name
3
4 @Company
5 Company Name
6
7 @File Name
8 filename.c
9
10 @Summary
11 Brief description of the file.
12
13 @Description
14 Describe the purpose of this file.
15 */
16 /* **** */
17
18 /* **** */
19 /* **** */
20 /* Section: Included Files */
21 /* **** */
22 /* **** */
23 #include "Mc32_PressAdc.h"
24 #include "app.h"
25 #include "peripheral/adc/plib_adc.h"
26 /* This section lists the other files that are included in this file.
27 */
28
29 /* TODO: Include other files here if needed. */
30
31
32 /* **** */
33 /* **** */
34 /* Section: File Scope or Global Data */
35 /* **** */
36 /* **** */
37
38 /* A brief description of a section can be given directly below the section
39 banner.
40 */
41
42 /* **** */
43
44
45
46 /* **** */
47 /* **** */
48 // Section: Local Functions
49 /* **** */
50 /* **** */
51
52 /* **** */
53
54
55 /* **** */
56 /* **** */
57 // Section: Interface Functions
58 /* **** */
59 /* **** */
60
61 /* A brief description of a section can be given directly below the section
62 banner.
63 */
64
65 // ****
66
67 void Press_InitADC (void){
68     //Configuration de l'adresse choisi ADC
69     PLIB_ADC_InputScanMaskAdd(ADC_ID_1, ADC_AN_SCAN_ADDRES);
70     // Configure l'ADC en mode alterné
71     PLIB_ADC_ResultFormatSelect(ADC_ID_1, ADC_RESULT_FORMAT_INTEGER_16BIT);
72     //Choisir ce mode -> Buffer alterné

```

```

73 PLIB_ADC_ResultBufferModeSelect(ADC_ID_1, ADC_BUFFER_MODE_TWO_8WORD_BUFFERS);
74 //mode multiplexage
75 PLIB_ADC_SamplingModeSelect(ADC_ID_1, ADC_SAMPLING_MODE_MUXA);
76
77 //la lecture des ADC est cadensée par le timer interne
78 PLIB_ADC_ConversionTriggerSourceSelect(ADC_ID_1, ADC_CONVERSION_TRIGGER_INTERNAL_COUNT);
79 //Tension de référence de l'ADC alimentation 3V3
80 PLIB_ADC_VoltageReferenceSelect(ADC_ID_1, ADC_REFERENCE_VDD_TO_AVSS);
81 PLIB_ADC_SampleAcquisitionTimeSet(ADC_ID_1, 0x1F);
82 PLIB_ADC_ConversionClockSet(ADC_ID_1, SYS_CLK_FREQ, 32);
83
84 //ADC fait 3 mesures par interruption (car 3 canaux utilisés) -> adapter en fct des ADC utilisés
85 PLIB_ADC_SamplesPerInterruptSelect(ADC_ID_1, ADC_1SAMPLE_PER_INTERRUPT);
86 //active le scan en mode multiplexage des entrées AN
87 PLIB_ADC_MuxAInputScanEnable(ADC_ID_1);
88
89 // Enable the ADC module
90 PLIB_ADC_Enable(ADC_ID_1);
91
92 }
93
94 S_ADCResults Press_ReadAllADC( void ) {
95 //structure de valeurs brutes des ADCs
96 volatile S_ADCResults rawResult;
97 // Traitement buffer
98 ADC_RESULT_BUF_STATUS BufStatus;
99 //stop sample/convert
100 PLIB_ADC_SampleAutoStartDisable(ADC_ID_1);
101 // traitement avec buffer alterné
102 BufStatus = PLIB_ADC_ResultBufferStatusGet(ADC_ID_1);
103 //Buffer 8 bits -> 0 à 7 -> expliqué après
104 if(BufStatus == ADC_FILLING_BUF_0TO7) {
105     rawResult.AN3 = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 0);
106 }
107 else //Buffer 8 bits -> 8 à 15
108 {
109     rawResult.AN3 = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 8);
110 }
111 // Retablit Auto start sampling
112 PLIB_ADC_SampleAutoStartEnable(ADC_ID_1);
113
114 //retourner valeurs lue
115 return rawResult;
116 }
117
118 float Press_RawToVoltage(float raw){
119     float voltage = 0;
120     /* Raw ADC to voltage */
121     voltage = raw * ADC_RES;
122     /* Voltage before op-amp */
123     voltage = voltage / OPAMP_GAIN;
124     return voltage;
125 }
126
127 float Press_voltageToPressure(float voltage) {
128     float pressure = 0;
129     /* Convet voltage to pressure in bar */
130     pressure = ((voltage - V_MIN)*P_RANGE)/V_MAX;
131
132     return pressure;
133 }
134
135 float Press_readPressure( void ) {
136     //structure de valeurs brutes des ADCs
137     volatile S_ADCResults rawResult;
138     /* Voltage variable */
139     float voltage = 0;
140     /* Pressure variable */
141     float pressure = 0;
142     /* Read ADC */
143     rawResult = Press_ReadAllADC();
144     /* Convert raw data to voltage */
145     voltage = Press_RawToVoltage(rawResult.AN3);
146     /* Get the pressure from the voltage */

```

```
147     pressure = Press_voltageToPressure(voltage);
148
149     return pressure;
150 }
151
152
153 /* ****
154 End of File
155 */
156
```

```

1 /* **** */
2 /** Descriptive File Name
3
4 @Company
5 Company Name
6
7 @File Name
8 filename.h
9
10 @Summary
11 Brief description of the file.
12
13 @Description
14 Describe the purpose of this file.
15 */
16 /* **** */
17
18 #ifndef _PRESS_ADC_H /* Guard against multiple inclusion */
19 #define _PRESS_ADC_H
20
21
22 /* **** */
23 /* **** */
24 /* Section: Included Files */
25 /* **** */
26 /* **** */
27 #include "app.h"
28 /* This section lists the other files that are included in this file.
29 */
30
31 /* TODO: Include other files here if needed. */
32
33
34 /* Provide C++ Compatibility */
35 #ifdef __cplusplus
36 extern "C" {
37 #endif
38
39
40 /* **** */
41 /* **** */
42 /* Section: Constants */
43 /* **** */
44 /* **** */
45 #define V_MIN      0.5
46 #define V_MAX      4
47 #define P_RANGE    10.0
48 #define OPAMP_GAIN 0.5913
49 #define ADC_RES    (3.3/1024)
50 #define ADC_AN_SCAN_ADDRES 0x0008
51 /* **** */
52
53
54 // ****
55 // ****
56 // Section: Data Types
57 // ****
58 // ****
59 typedef struct{
60     uint16_t AN3;
61 }S_ADCResults;
62 // ****
63
64
65 // ****
66 // ****
67 // Section: Interface Functions
68 // ****
69 // ****
70
71 /* Convert voltage into pressure in [Bar] */
72 float Press_voltageToPressure(float voltage);

```

```
73
74 /* Convert raw adc value to voltage */
75 float Press_RawToVoltage(float raw);
76
77 void Press_InitADC (void);
78
79 S_ADCResults Press_ReadAllADC( void );
80
81 float Press_readPressure( void );
82 // ****
83
84
85
86 /* Provide C++ Compatibility */
87 #ifdef __cplusplus
88 }
89 #endif
90
91 #endif /* _EXAMPLE_FILE_NAME_H */
92
93 // ****
94 End of File
95 */
96
```

```

1 /* **** */
2 /** Descriptive File Name
3
4 @Company
5 ETML-ES
6
7 @File Name
8 sd_fat_gest.c
9
10 @Summary
11 SD card fat system management
12
13 @Description
14 SD card fat system management
15 */
16 /* **** */
17
18 /* **** */
19 /* **** */
20 /* Section: Included Files */
21 /* **** */
22 /* **** */
23
24 /* This section lists the other files that are included in this file.
25 */
26
27 #include "Mc32_sdFatGest.h"
28 #include <stdio.h>
29 #include "app.h"
30 #include "bno055_support.h"
31
32 /* **** */
33 /* **** */
34 /* Section: File Scope or Global Data */
35 /* **** */
36 /* **** */
37
38 APP_FAT_DATA_COHERENT_ALIGNED appFatData;
39 /* **** */
40 /** Descriptive Data Item Name
41
42 @Summary
43 Brief one-line summary of the data item.
44
45 @Description
46 Full description, explaining the purpose and usage of data item.
47 <p>
48 Additional description in consecutive paragraphs separated by HTML
49 paragraph breaks, as necessary.
50 <p>
51 Type "JavaDoc" in the "How Do I?" IDE toolbar for more information on tags.
52
53 @Remarks
54 Any additional remarks
55 */
56
57
58 /* **** */
59 /* **** */
60 // Section: Local Functions
61 /* **** */
62 /* **** */
63
64 /* **** */
65
66
67
68 /* **** */
69 /* **** */
70 // Section: Interface Functions
71 /* **** */
72 /* **** */
73
74 void sd_fat_task ( void )
75 {
76     /* The application task state machine */
77     switch(appFatData.state)
78     {
79         case APP_MOUNT_DISK:
80             if(SYS_FS_Mount("/dev/mmcblk1", "/mnt/myDrive", FAT, 0, NULL) != 0)
81             {
82                 /* The disk could not be mounted. Try
83                  * mounting again until success. */
84
85                 appFatData.state = APP_MOUNT_DISK;
86             }
87         else
88         {
89             /* Mount was successful. Unmount the disk, for testing. */
90
91             appFatData.state = APP_SET_CURRENT_DRIVE;
92         }
93         break;
94
95     case APP_SET_CURRENT_DRIVE:
96         if(SYS_FS_CurrentDriveSet("/mnt/myDrive") == SYS_FS_RES_FAILURE)
97         {
98             /* Error while setting current drive */
99             appFatData.state = APP_ERROR;
100        }
101    else
102    {
103        /* Open a file for reading. */
104        appFatData.state = APP_IDLE;
105    }
106    break;

```

```

107
108 case APP_WRITE_MEASURE_FILE:
109     appFatData.fileHandle = SYS_FS_FileOpen("MESURES.csv",
110         (SYS_FS_FILE_OPEN_APPEND_PLUS));
111     if(appFatData.fileHandle == SYS_FS_HANDLE_INVALID)
112     {
113         /* Could not open the file. Error out*/
114         appFatData.state = APP_ERROR;
115     }
116     else
117     {
118         /* Create a directory.*/
119         appFatData.state = APP_WRITE_TO_MEASURE_FILE;
120     }
121     break;
122
123 case APP_WRITE_TO_MEASURE_FILE:
124     /* If read was success, try writing to the new file */
125     if(SYS_FS_FileStringPut(appFatData.fileHandle, appFatData.data) == -1)
126     {
127         /* Write was not successful. Close the file
128         * and error out.*/
129         SYS_FS_FileClose(appFatData.fileHandle);
130         appFatData.state = APP_ERROR;
131     }
132     else
133     {
134         appFatData.state = APP_CLOSE_FILE;
135     }
136     break;
137
138 case APP_CLOSE_FILE:
139     /* Close both files */
140     SYS_FS_FileClose(appFatData.fileHandle);
141     /* The test was successful. Lets idle. */
142     appFatData.state = APP_IDLE;
143     break;
144
145 case APP_IDLE:
146     /* The application comes here when the demo
147      * has completed successfully. Switch on
148      * green LED.*/
149     //BSP_LEDOn(APP_SUCCESS_LED);
150     LED_ROff();
151     break;
152 case APP_ERROR:
153     /* The application comes here when the demo
154      * has failed. Switch on the red LED.*/
155     //BSP_LEDOn(APP_FAILURE_LED);
156     LED_ROn();
157     break;
158 default:
159     break;
160
161 case APP_UNMOUNT_DISK:
162     if(SYS_FS_Unmount("/mnt/myDrive") != 0)
163     {
164         /* The disk could not be un mounted. Try
165          * un mounting again untill success.*/
166
167         appFatData.state = APP_UNMOUNT_DISK;
168     }
169     else
170     {
171         /* UnMount was successful. Mount the disk again */
172         appFatData.state = APP_IDLE;
173     }
174     break;
175
176 }
177
178 // _SYS_FS_Tasks();
179 } //End of APP_Tasks
180
181 void sd_BNO_scheduleWrite (s_bno055_data * data)
182 {
183     /* If sd Card available */
184     if(appFatData.state == APP_IDLE)
185     {
186         /* Next state : write to file */
187         appFatData.state = APP_WRITE_MEASURE_FILE;
188         /* Write the buffer */
189         sprintf(appFatData.data, "%d;%d;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%f;%d;%d;%d;%d";
190             ,data->flagImportantMeas, (data->d_time), data->pressure, data->gravity.x, data->gravity.y, data->gravity.z, data->gyro.x, data->gyro.y, data->gyro.z
191             ,data->mag.x, data->mag.y, data->mag.z, data->linear_accel.x, data->linear_accel.y, data->linear_accel.z
192             ,data->euler.h, data->euler.p, data->euler.r, data->quaternion.w, data->quaternion.x, data->quaternion.y, data->quaternion.z);
193         /* Compute the number of bytes to send */
194         appFatData.nBytesToWrite = strlen(appFatData.data);
195     }
196 }
197
198 APP_FAT_STATES sd_getState( void )
199 {
200     return appFatData.state;
201 }
202
203 void sd_setState( APP_FAT_STATES newState )
204 {
205     appFatData.state = newState;
206 }
207
208 /* ****
209 End of File
210 */
211

```

```

1 //*****
2 MPLAB Harmony Application Header File
3
4 Company:
5 Microchip Technology Inc.
6
7 File Name:
8 app.h
9
10 Summary:
11 This header file provides prototypes and definitions for the application.
12
13 Description:
14 This header file provides function prototypes and data type definitions for
15 the application. Some of these are required by the system (such as the
16 "APP_Initialize" and "APP_Tasks" prototypes) and some of them are only used
17 internally by the application (such as the "APP_FAT_STATES" definition). Both
18 are defined here for convenience.
19 *****/
20
21 //DOM-IGNORE-BEGIN
22 *****
23 Copyright (c) 2013-2014 released Microchip Technology Inc. All rights reserved.
24
25 Microchip licenses to you the right to use, modify, copy and distribute
26 Software only when embedded on a Microchip microcontroller or digital signal
27 controller that is integrated into your product or third party product
28 (pursuant to the sublicense terms in the accompanying license agreement).
29
30 You should refer to the license agreement accompanying this Software for
31 additional information regarding your rights and obligations.
32
33 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
34 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
35 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
36 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
37 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
38 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
39 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
40 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
41 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
42 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
43 *****/
44 //DOM-IGNORE-END
45
46 #ifndef _SD_FAT_GEST_H
47 #define _SD_FAT_GEST_H
48
49
50 // *****
51 // *****
52 // Section: Included Files
53 // *****
54 // *****
55
56 #include "app.h"
57
58 // *****
59 // *****
60 // Section: Type Definitions
61 // *****
62 // *****
63
64 #ifdef DRV_SDHC_USE_DMA
65 #define DATA_BUFFER_ALIGN      __attribute__((coherent, aligned(32)))
66 #else
67 #define DATA_BUFFER_ALIGN      __attribute__((aligned(32)))
68 #endif
69
70 // *****
71 /* Application States
72

```

```

73 Summary:
74 Application states enumeration
75
76 Description:
77 This enumeration defines the valid application states. These states
78 determine the behavior of the application at various times.
79 */
80
81 typedef enum
82 {
83     /* Application's state machine's initial state.*/
84     /* The app mounts the disk */
85     APP_MOUNT_DISK = 0,
86
87     /* Set the current drive */
88     APP_SET_CURRENT_DRIVE,
89
90     /* The app opens the file to read */
91     APP_WRITE_MEASURE_FILE,
92
93     /* The app reads from a file and writes to another file */
94     APP_WRITE_TO_MEASURE_FILE,
95
96     /* The app closes the file*/
97     APP_CLOSE_FILE,
98
99     /* The app closes the file and idles */
100    APP_IDLE,
101
102    /* An app error has occurred */
103    APP_ERROR,
104
105    /* Unmount disk */
106    APP_UNMOUNT_DISK
107
108 } APP_FAT_STATES;
109
110 // *****
111 /* Application Data
112
113 Summary:
114 Holds application data
115
116
117 Description:
118 This structure holds the application's data.
119
120 Remarks:
121 Application strings and buffers are be defined outside this structure.
122 */
123
124 typedef struct
125 {
126     /* SYS_FS File handle for 1st file */
127     SYS_FS_HANDLE     fileHandle;
128
129     /* SYS_FS File handle for 2nd file */
130     SYS_FS_HANDLE     fileHandle1;
131
132     /* Application's current state */
133     APP_FAT_STATES      state;
134
135     /* Application data buffer */
136     char          data[256] DATA_BUFFER_ALIGN;
137
138     uint32_t        nBytesWritten;
139
140     uint32_t        nBytesRead;
141
142     uint32_t        nBytesToWrite;
143 } APP_FAT_DATA;
144
145 // *****
146 // *****
147 // *****

```

```

148 // Section: Application Callback Routines
149 // ****
150 // ****
151 /* These routines are called by drivers when certain events occur.
152 */
153
154
155 // ****
156 // ****
157 // Section: Application Initialization and State Machine Functions
158 // ****
159 // ****
160
161 ****
162
163 Function:
164 void APP_Tasks ( void )
165
166 Summary:
167 MPLAB Harmony Demo application tasks function
168
169 Description:
170 This routine is the Harmony Demo application's tasks function. It
171 defines the application's state machine and core logic.
172
173 Precondition:
174 The system and application initialization ("SYS_Initialize") should be
175 called before calling this.
176
177 Parameters:
178 None.
179
180 Returns:
181 None.
182
183 Example:
184 <code>
185 APP_Tasks();
186 </code>
187
188 Remarks:
189 This routine must be called from SYS_Tasks() routine.
190 */
191
192 void sd_fat_task ( void );
193
194 void sd_BNO_scheduleWrite (s_bno055_data * data);
195
196 APP_FAT_STATES sd_getState( void );
197
198 void sd_setState( APP_FAT_STATES newState );
199
200 #endif /* _APP_H */
201 ****
202 End of File
203 */
204
205

```