

# Tube Pitot déporté

Détection de l'angle d'incidence au décrochage d'un avion par mesure de la vitesse et des accélérations

Meven Ricchieri  
2022 - 2023

Numéro de projet : 2230

Responsable : M. Juan José Moreno  
Mandataire : M. Vincent Seguin (pour AMPA)



## Table des matières

<b>1 Cahier des charges</b>	<b>3</b>
<b>2 Pré-étude</b>	<b>3</b>
2.1 Contexte . . . . .	3
2.2 Schéma de principe . . . . .	3
2.3 Schéma bloc détaillé . . . . .	4
2.4 Descriptions des blocs détaillées . . . . .	4
2.4.1 Bloc Générateur (turbine) . . . . .	4
2.4.2 Bloc Capteur de pression . . . . .	5
2.4.3 Bloc Tube Pitot . . . . .	5
2.4.4 Bloc IMU . . . . .	6
2.4.5 Bloc Connecteur USB . . . . .	6
2.4.6 Bloc Module BLE . . . . .	7
2.4.7 Bloc LED bleue . . . . .	7
2.4.8 Bloc MCU . . . . .	7
2.4.9 Bloc Alimentation . . . . .	7
2.4.10 Bloc Appareil Android . . . . .	8
2.5 Design mécanique . . . . .	8
2.6 Estimation du prix . . . . .	9
2.7 Faisabilité du projet . . . . .	9
<b>3 Design électronique</b>	<b>10</b>
3.1 Introduction . . . . .	10
3.2 Générateur . . . . .	10
3.3 LED de signalisation . . . . .	12
3.4 Piles rechargeables . . . . .	13
3.5 Capteur de pressions différentielles . . . . .	14
3.6 IMU . . . . .	15
3.7 Module Bluetooth . . . . .	16
3.8 MCU . . . . .	17
3.9 Alimentations . . . . .	19
3.10 Pull-up I2C . . . . .	20
<b>4 Design mécanique</b>	<b>21</b>
4.1 Introduction . . . . .	21
4.2 Design . . . . .	21
4.3 Modifications . . . . .	25
4.3.1 Circuit imprimé . . . . .	25
4.3.2 Boîtier . . . . .	25
<b>5 Mise en service</b>	<b>26</b>
5.1 Introduction . . . . .	26
5.2 Test des alimentations . . . . .	26
5.3 Consommation . . . . .	26
5.4 Périphériques . . . . .	26
5.4.1 Signaux USART . . . . .	27
5.4.2 Signaux I2C . . . . .	28
5.5 Améliorations possibles . . . . .	29

<b>6 Software MCU</b>	<b>30</b>
6.1 Introduction . . . . .	30
6.2 Paramétrage Harmony . . . . .	30
6.2.1 Fréquence d'horloge . . . . .	30
6.2.2 Timers . . . . .	31
6.2.3 Entrées sorties . . . . .	32
6.2.4 Communications série . . . . .	33
6.3 Flowcharts . . . . .	34
6.3.1 Machine d'état principale . . . . .	34
6.3.1.1 Etat d'initialisation . . . . .	35
6.3.1.2 Etat de service . . . . .	36
6.4 Codes C . . . . .	37
<b>7 Software Android</b>	<b>39</b>
7.1 Introduction . . . . .	39
7.2 Aperçu des interfaces . . . . .	39
<b>8 Résultat final</b>	<b>40</b>
8.1 Introduction . . . . .	40
8.2 État d'avancement . . . . .	41
8.3 Tâches restantes . . . . .	41
<b>9 Conclusion</b>	<b>42</b>
<b>10 Annexes</b>	<b>43</b>

## 1 Cahier des charges

Le but du projet consiste à développer un système permettant de détecter l'angle d'inclinaison et la vitesse au décollage d'un avion. La fixation de ce système doit être flexible afin de pouvoir l'installer sur différents types d'avions. L'emplacement de fixation ne doit pas se trouver dans le flux d'air provenant de l'hélice afin d'éviter que la mesure de vitesse ne soit faussée. Il doit également être miniaturisé au maximum afin de produire le minimum de traînée possible et de ne pas dépasser un poids de 500g. Les données acquises par les capteurs doivent être transmises de la partie déportée à un appareil Android se trouvant dans le cockpit de l'avion à travers une communication sans fil. L'appareil Android doit traiter et afficher les données reçues, si possible graphiquement (optionnel).

Le cahier des charges complet se trouve en annexes.

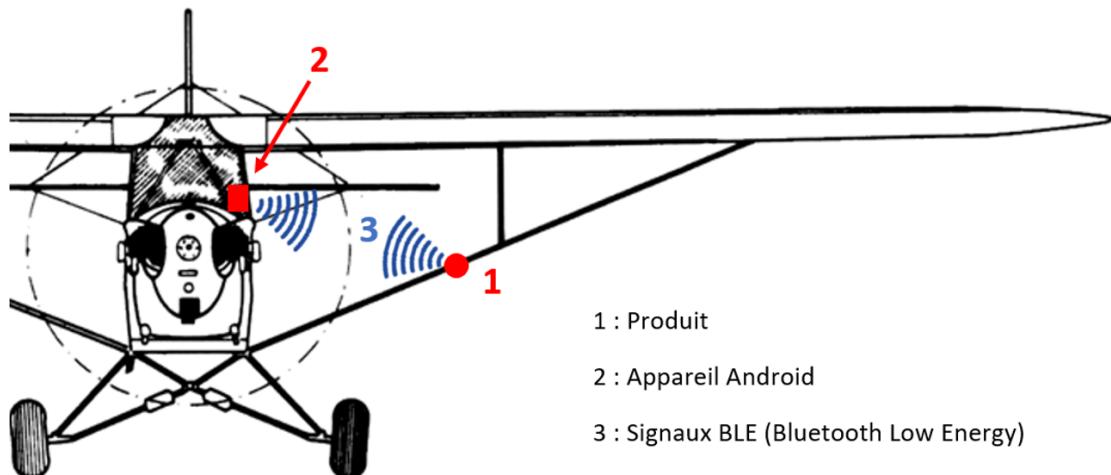
## 2 Pré-étude

### 2.1 Contexte

Cette pré-étude consiste à mener des recherches qui conduiront à une décision de lancer ou non le projet. Cette décision va reposer sur les difficultés de réalisation électrique, mécanique et software. Elle va aussi permettre d'imaginer et de comparer les différentes manières de réaliser les tâches et ainsi choisir la meilleure.

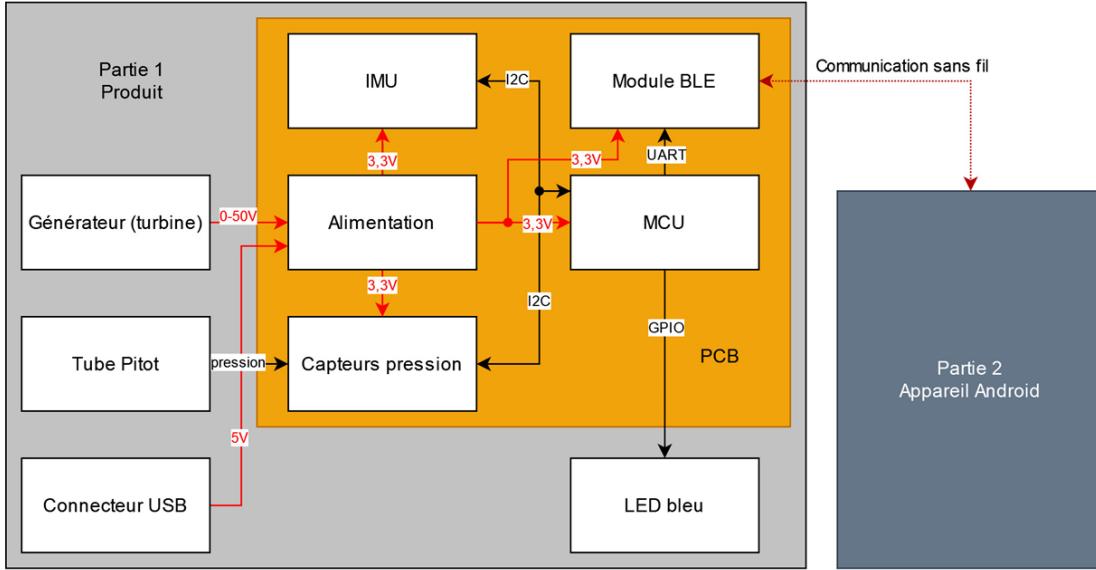
### 2.2 Schéma de principe

FIGURE 1 – Schéma de principe



## 2.3 Schéma bloc détaillé

FIGURE 2 – Schéma bloc détaillé



## 2.4 Descriptions des blocs détaillées

### 2.4.1 Bloc Générateur (turbine)

L'intérêt d'utiliser un générateur comme source d'énergie permet de simplifier l'utilisation du système, il n'y aura aucune maintenance à effectuer ou de batterie à charger. L'énergie cinétique de l'air en mouvement, par rapport à l'avion, va être captée par les aubes de la turbine axée à un générateur, ce qui va produire une tension. Le système va se mettre en marche une fois que l'axe du générateur-turbine aura atteint une certaine vitesse de rotation. Plus la vitesse sera grande, plus la tension générée sera élevée. Un simple moteur DC, va permettre de transformer cette énergie mécanique en énergie électrique. Il est possible d'utiliser d'autres technologies comme les moteurs brushless, qui ont un rendement bien meilleur, mais cela complexifie le circuit et augmente le prix. Dans ce projet, le rendement n'est pas primordial car le système ne consommera que très peu d'énergie.

$$U_t(V) = \frac{n}{k_n} - R_{mot} * I_L \quad [2] \quad (1)$$

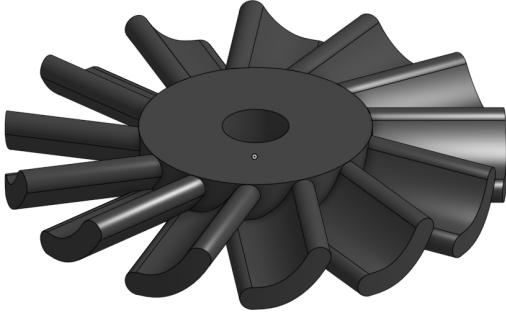
où :

- $U_t$  = Tension aux bornes du moteur DC (V)
- $k_n$  = Constante de vitesse du moteur DC (rpm/V)
- $R_{mot}$  = Résistance aux bornes du moteur DC ( $\Omega$ ) Puisque la valeur de la tension générée est dépendante de la vitesse air de l'avion, celle-ci ne sera jamais stable. Il faudra donc la transformer et la réguler avant de la transmettre aux composants électroniques, ce à quoi le bloc "Alimentation"
- $n$  = Vitesse moteur DC (rpm)
- $I_L$  = Courant de charge (I)

sera destiné.

N'ayant pas de grandes connaissances dans le dimensionnement et le design de générateur-turbines, des recherches plus poussées seront nécessaires afin de rendre le système performant. Pour me faire une première idée, j'ai dessiné et imprimé une première turbine composée de 13 aubes puis j'ai effectué quelques tests. A l'aide d'un moteur DC, j'ai réussi à générer une tension d'environ 3V à une vitesse de 80km/h.

FIGURE 3 – Turbine de test N°1



#### 2.4.2 Bloc Capteur de pression

Le système sera équipé d'un capteur de pression différentielle qui permettra de mesurer la différence entre la pression statique et la pression totale. "La pression statique, dans un fluide en mouvement, est la pression que mesure un capteur qui se déplace à la même vitesse que le fluide" [3]. "La pression totale dans un fluide (eau, air, etc.) est la somme de la pression statique, de la pression dynamique, et de la densité volumique d'énergie potentielle de gravité" [4]. Grâce aux valeurs ainsi mesurées, il sera possible de calculer la vitesse du fluide, ce qui correspondra à la vitesse air de l'avion. Pour des vitesses inférieures à Mach 0.3 (200 nuds), l'effet de compressibilité du fluide peut être négligé [5].

$$v(m/s) = \sqrt{\frac{2 * (p_t - p_s)}{\rho}} \quad [5] \quad (2)$$

où :

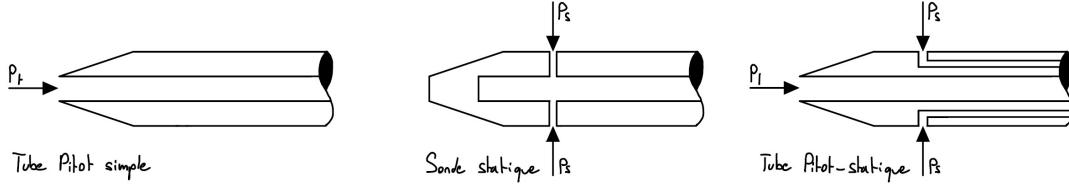
- $v$  = Vélocité du fluide (m/s)
- $p_t$  = Pression totale (Pa)
- $p_s$  = Pression statique (Pa)
- $\rho$  = Masse volumique du fluide ( $kg/m^3$ )

#### 2.4.3 Bloc Tube Pitot

Comme expliqué au point précédent, la vitesse se calcule grâce à la pression statique et totale, pour que le capteur différentielle ait accès à ces deux pressions, il faut ajouter deux entrées d'air. Ces entrées doivent être placées correctement car dans le cas contraire, la vitesse calculée ne sera pas correcte. L'entrée de la pression statique doit être perpendiculaire à l'écoulement local du fluide (non perturbé) alors que celle de la pression totale doit être parallèle à ce flux. Il existe 3 tubes différents, le premier, appelé tube Pitot simple, est simplement un tube perforé en son centre faisant office d'entrée d'air pour la pression totale. Le second, appelé sonde statique, est perforé latéralement permettant

l'entrée d'air pour la pression statique. Le troisième est une combinaison des deux premiers, il dispose d'une entrée pour la pression totale et d'une autre pour la pression statique. L'idéal serait d'utiliser ce dernier car il simplifierait l'implémentation mécanique du système. Les deux sorties du tube seraient reliées au capteur de pression différentielle par des tuyaux. Dans le cas où un tube Pitot simple serait utilisé, il serait nécessaire de percer le boîtier du produit afin d'avoir accès à la pression statique. La décision du type de tube sera effectuée lors du design mécanique car il n'influence aucunement la conception électronique.

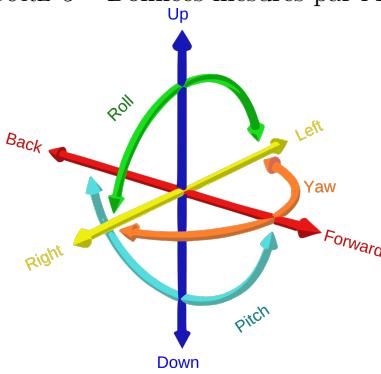
FIGURE 4 – Types de tubes Pitot



#### 2.4.4 Bloc IMU

La centrale inertie comportera 6 capteurs, 3 gyromètres et 3 accéléromètres. *"Les gyromètres mesurent les trois composantes du vecteur vitesse angulaire (vitesses de variation des angles de roulis, de tangage et de lacet. Les accéléromètres mesurent les trois composantes du vecteur force spécifique. La force spécifique est la somme des forces extérieures autres que gravitationnelles divisée par la masse"* [1]. C'est grâce à ses capteurs qu'il sera possible de détecter le moment où l'avion commence à décrocher. La plupart de ces IMUs ont également un capteur de température intégré, la mesure de cette grandeur peut être intéressante puisqu'elle a une influence sur la valeur de la pression.

FIGURE 5 – Données mesurés par l'IMU



#### 2.4.5 Bloc Connecteur USB

Étant donné que le système n'aura pas de source d'énergie interne, un connecteur USB-C sera implémenté afin de pouvoir tester la communication ainsi que les capteurs, en amont du vol. Il permettra également d'alimenter le système lors de son installation sur l'avion, car il devra être positionné correctement à l'horizontale pour que les mesures soient fiables.

#### 2.4.6 Bloc Module BLE

Le transfert de données entre le produit et l'appareil Android, se fera au travers d'une communication sans fil Bluetooth Low Energy (BLE). Cette variante de Bluetooth permet de réduire significativement sa consommation par rapport au mode normal. Il sera tout de même possible qu'un module permettant de choisir le mode de fonctionnement soit implémenté. Les données transmises du produit à l'appareil Android seront principalement les valeurs lues par les capteurs alors que celle provenant de l'appareil seront plutôt des commandes ou réglages.

#### 2.4.7 Bloc LED bleue

Comme énoncé dans le CDC, la LED bleue va permettre d'indiquer l'état du système. La LED éteinte va signifier que le système n'est pas alimenté. Un clignotement toutes les 2 secondes correspondra au système alimenté mais non appairé à un périphérique Bluetooth et finalement, 2 clignotements toutes les 2 secondes correspondra à l'état fonctionnel, appairé et transmettant les données. La LED devra être visible de jour donc son emplacement et son intensité lumineuse devront être correctement défini.

#### 2.4.8 Bloc MCU

Le microcontrôleur sera obligatoirement un modèle 32Bit du fabricant Microchip. Il va faire le lien entre tous les périphériques implantés, c'est à dire les composants d'entrées, IMU et capteur de pression et ceux de sortie, module BLE et LED. Selon les recherches effectués précédemment, le microcontrôleur devra au minimum contenir :

- Un module UART pour la communication avec le module Bluetooth
- Un module I2C pour la communication avec l'IMU et le capteur de pression
- Une PIN GPIO pour le contrôle de la LED

#### 2.4.9 Bloc Alimentation

Puisque la grande majorité des composants nécessaires, tel que le PIC ou les capteurs, fonctionnent avec une tension de 3.3V, il faudra donc que la sortie de ce bloc soit proche de cette valeur. Le générateur va produire une tension bien supérieur à 3.3V, donc la meilleure option est d'utiliser un convertisseur Buck afin de l'abaisser. Il permettra d'avoir un bien meilleur rendement qu'un régulateur linéaire et donc limiter la consommation.

Composant	min	max
Module BLE (consommation non continue)	5mA	10mA
Microcontrôleur	5mA	10mA
Capteur pression différentielle	3mA	4mA
LED (consommation non continue)	10mA	20mA
Centrale inertie	1mA	3mA
Total	24mA	47mA

La consommation totale devrait se situer entre 24mA et 47mA

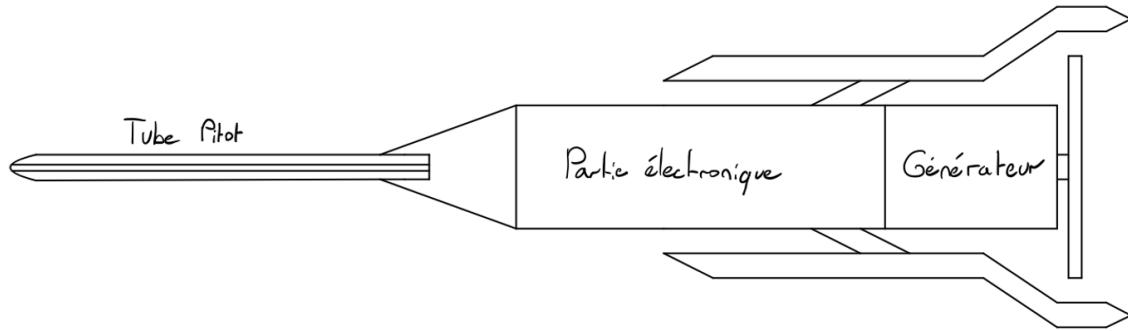
#### 2.4.10 Bloc Appareil Android

Toutes les données obtenues au travers de la communication sans fil seront traitées et affichées sur l'écran de l'appareil. Ces données seront affichées sous forme numérique et dans l'idéal, sous forme graphiques, mais cela reste optionnel. L'application permettra d'envoyer certaines commandes au produit en appuyant sur les différents boutons qui seront affichés. Il sera aussi possible d'ajouter une valeur d'offset aux mesures pour compenser une éventuelle erreur.

### 2.5 Design mécanique

Comme expliqué dans le cahier des charges, l'idée est d'adapter tout le système pour qu'il puisse se trouver dans un cylindre à l'arrière du tube Pitot. Le boîtier et les autres parties mécaniques seront imprimé en une matière plastique selon la disponibilité de matériaux de l'école. L'idéal serait d'utiliser de l'acrylonitrile butadiène styrène (ABS) car ses caractéristiques mécaniques sont bien supérieures à l'acide polylactique (PLA). La grosse difficulté va être de miniaturiser le design du PCB afin de rendre la section du tube la plus mince possible. Le système de fixation sera réalisé grâce à une boule style RAM-Mounts vissé au boîtier.

FIGURE 6 – Premier croquis du design mécanique



## 2.6 Estimation du prix

L'estimation du prix est actuellement difficile car certains choix doivent encore être pris et la disponibilité des composants peut avoir une influence, j'ai donc mis un plage de prix.

Composant	min	max
Générateur (moteur DC)	5.-	50.-
Turbine (ABS)	0.5.-	1.-
Capteur pression différentielle	30.-	70.-
Tube Pitot	0.-	30.-
Centrale inertielle	5.-	12.-
Connecteur USB-C	1.-	3.-
Module Bluetooth	5.-	15.-
LED	0.5.-	3.-
MCU	1.-	3.-
Convertisseur Buck	3.-	8.-
Boîtier (ABS)	1.-	2.-
Boule RAM-Mounts	15.-	25.-
PCB	50.-	75.-
Total	126.-	396.-

Le prix devrait se situer entre 126.- et 396.-

## 2.7 Faisabilité du projet

D'après les informations trouvées, le projet est totalement réalisable. Il y aura sans doute quelques difficultés au niveau du dimensionnement de la partie turbine-générateur.

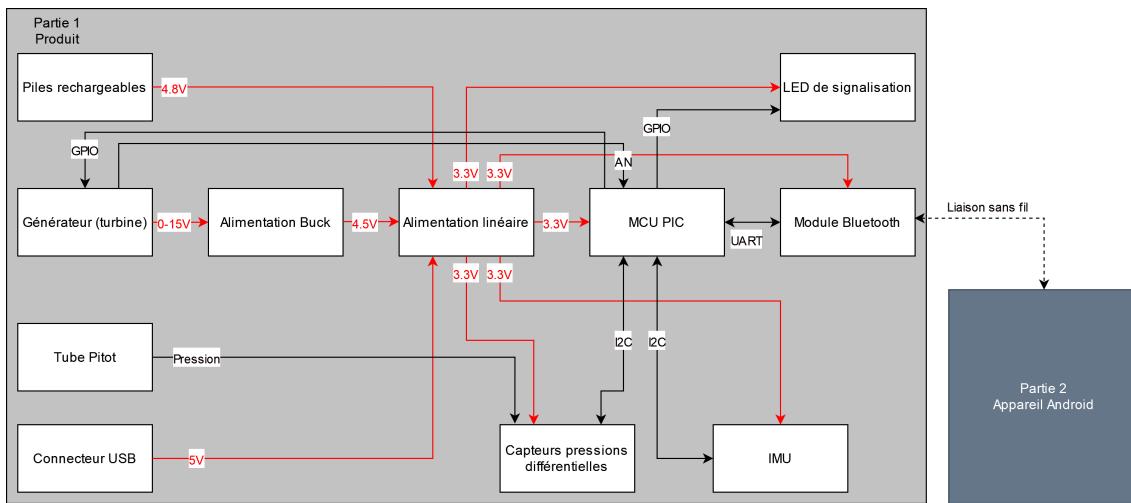
### 3 Design électronique

#### 3.1 Introduction

La partie Design électronique consiste à réaliser le schéma électrique, choisir les différents composants et réaliser les dimensionnements nécessaires. Certaines modifications ont suivi la présentation de la pré-étude, ceux-ci concernent :

- La source d'alimentation
- La régulation de tension

FIGURE 7 – Schéma bloc détaillé



#### 3.2 Générateur

Selon l'estimation de consommation totale effectuée lors de la pré-étude, il est maintenant possible de calculer approximativement la puissance minimum que le générateur devra délivrer au système.

$$P(W) = U * I \quad (3)$$

où :

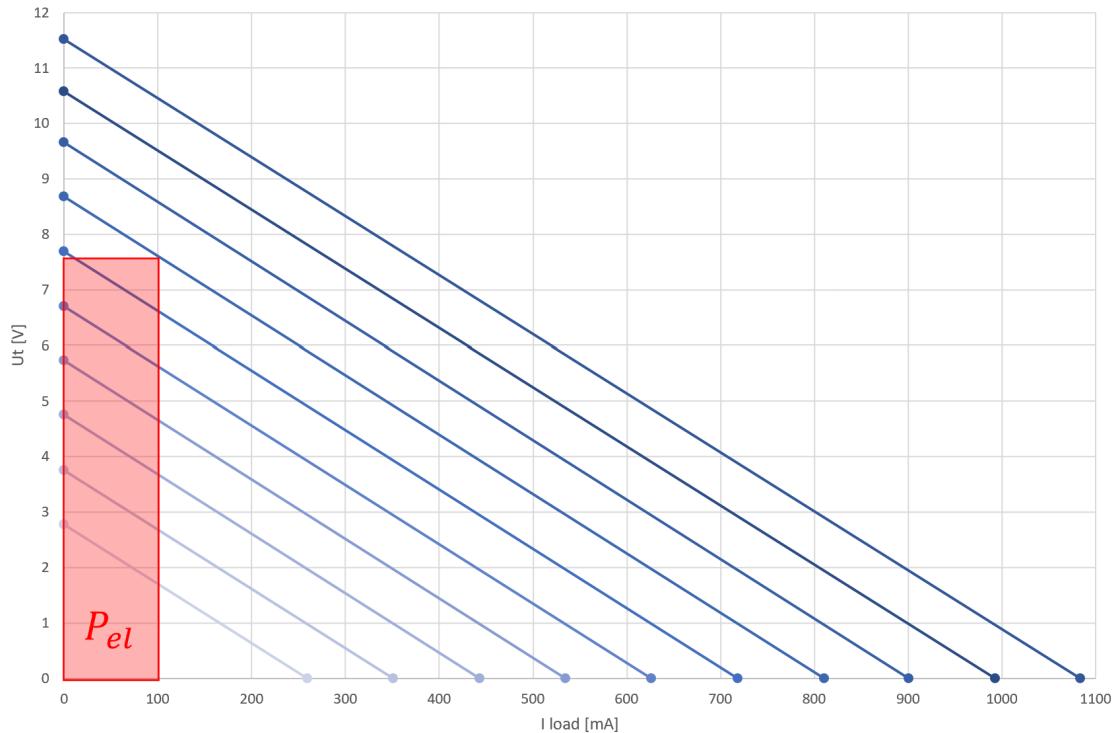
- $P$  = Puissance nécessaire (W)
- $U$  = Tension du circuit (V)
- $I$  = Courant consommé (A)

Avec la formule (3) :

$$P(W) = 3.3 * 0.047 = 0.1551W = 155mW$$

Les 155mW correspondent à la puissance maximale approximative consommée après transformation, ce qui signifie qu'il y aura quelques pertes au niveau des abaisseurs de tension. Afin de ne pas avoir de mauvaises surprises, j'ai décidé de prendre une certaine marge, au lieu de 155mW, la puissance sera de 250mW. Cette puissance étant tout de même très faible, le système sera capable de fonctionner même à faibles vitesses tant que la tension minimale est atteinte. Ayant effectué des tests lors de ma pré-étude avec un moteur 3VDC et une première turbine, mon professeur M.Moreno m'en a fourni un différent, RF-370A-15370, fonctionnant jusqu'à une tension nominale de 12VDC. Pour vérifier si ce moteur pourra être utilisé, j'ai réalisé quelques mesures et calculs afin de visualiser les lignes de tension-courant du générateur.

FIGURE 8 – Lignes tension-courant du générateur



Le rectangle rouge sur la figure 8 permet de visualiser la puissance électrique. Avec celle-ci nous pouvons savoir quelle tension sera disponible au borne du moteur en fonction de la vitesse de rotation (tension à vide) et du courant consommé. Le rectangle actuel montre que la tension est d'environ 7.8V lorsque la tension à vide est de 8V (environ 3600rpm) avec un courant consommé de 100mA.

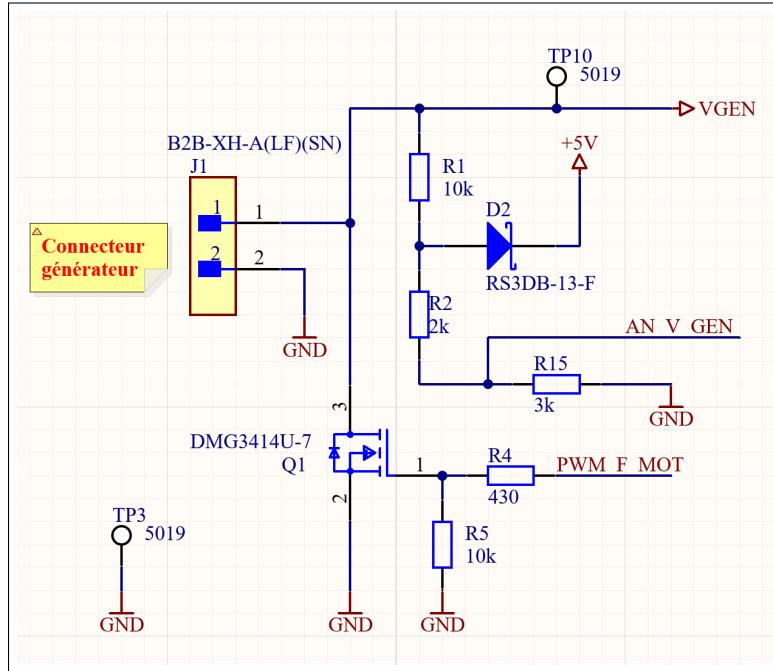
Avec la formule (3) :

$$P(W) = 7.8 * 0.100 = 0.78W = 780mW$$

Avec ce calcul, il est évident que le moteur correspond parfaitement puisque celui-ci devra fournir bien moins de puissance au système. Ce moteur là est plus adapté car il génère une tension supérieurs à la tension après régulation, 3.3VDC.

Le problème de ce moteur est qu'au moment où sa vitesse est trop élevée, cela peut le dégrader. J'ai donc décidé d'ajouter au schéma un système de freinage. La tension générée par le moteur passera par un pont diviseur avant d'entrer dans une entrée analogique du MCU. Une diode de protection permettra de court-circuiter la surtension lorsqu'elle est trop importante pour protéger le MCU. Une sortie PWM sera connectée à la gate d'un N-MOSFET afin de court-circuiter le moteur afin de le freiner lorsque la lecture de l'entrée analogique indiquera que la vitesse est trop élevée. La tension entrant dans le MCU ne pourra pas dépasser la tension max de 3.3V.

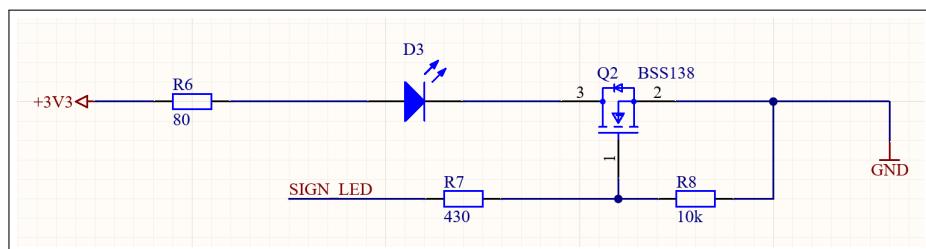
FIGURE 9 – Générateur, circuit de mesure et de freinage



### 3.3 LED de signalisation

La LED de signalisation sera commandée par une GPIO du MCU et permettra d'indiquer l'état du système à l'utilisateur.

FIGURE 10 – LED de signalisation



### 3.4 Piles rechargeables

Suite à la présentation de ma pré-étude, il a été décidé que le produit possédera deux sources d'énergie différentes, l'intérêt est de rendre le système plus flexible. La première source sera celle de base, alimentation par un générateur-turbine, et la seconde sera une alimentation à piles rechargeables. Il sera physiquement pas possible de monter les deux sources en même temps, l'idée est d'avoir un produit modulable.

Le système possédera 4 piles rechargeables NiMH AA de 1.2V chacune, connectées en série afin obtenir une tension de 4.8V en sortie. Celles-ci seront placées dans un support à piles cylindrique afin de respecter au maximum l'idée de design mécanique présenté dans la pré-étude (Figure 6). Aucun système de charge ne sera intégré au produit, il sera donc nécessaire d'utiliser un chargeur externe.

La durée d'utilisation totale est obtenue en effectuant la formule ci-dessous :

$$t(h) = \frac{C_A}{I} \quad (4)$$

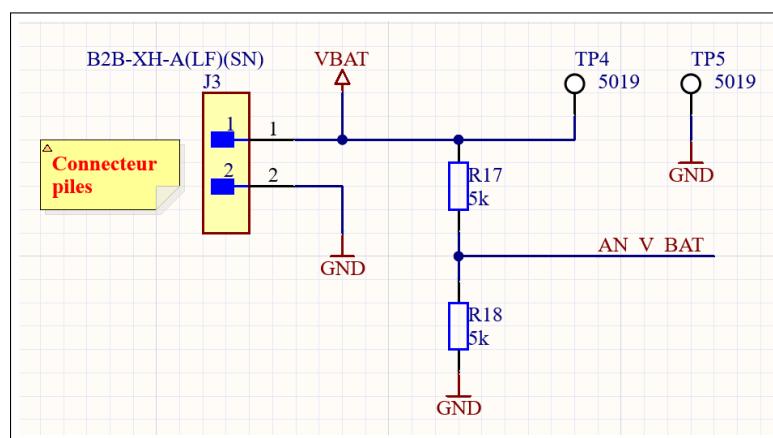
où :

- $t$  = Durée de vie (h)
- $I$  = Courant tiré (mA)
- $C_A$  = Capacité batterie (mAh)

$$t(h) = \frac{2700}{50} = 54h$$

La tension d'une cellule NiMH est d'environ 1.25V chargée et de 1.1V lorsqu'elle est déchargée. De ce fait, la tension va varier entre 5V et 4.4V aux bornes du connecteur. La tension sera lue par une entrée analogique du MCU afin de connaître son état et donc avertir l'utilisateur lorsque les batteries doivent être chargées. Un diviseur de tension est nécessaire car les entrées du microcontrôleur ne supportent pas une tension de 5V.

FIGURE 11 – Connecteur piles



### 3.5 Capteur de pressions différentielles

Le capteur de pressions différentielles choisi est le HSCMRRN001PD2A3 du fabricant Honeywell. Il s'agit d'un capteur ayant 2 entrées d'air, une pour la pression statique et la seconde pour la pression totale. La communication se fera au moyen d'une liaison série I2C. Le composant ne nécessite pas de dimensionnement particulier au niveau hardware. Son adresse I2C est fixe et est 0b00101000 (0x28). La précision de ce capteur est de  $\pm 1\%$ .

Product Series	HSC	High Accuracy, Compensated/Amplified
Package	M	SMT (Surface Mount Technology)
Pressure Port	RR	Dual radial barbed ports, same side
Options	N	Dry gases only, no diagnostics
Pressure Range	001PD	$\pm 1$ psi (differential)
Output Type	2	I2C, Address 0x28
Transfer Function	A	10% to 90% of Vsupply (analog), 214 counts (digital)
Supply Voltage	3	3.3Vdc

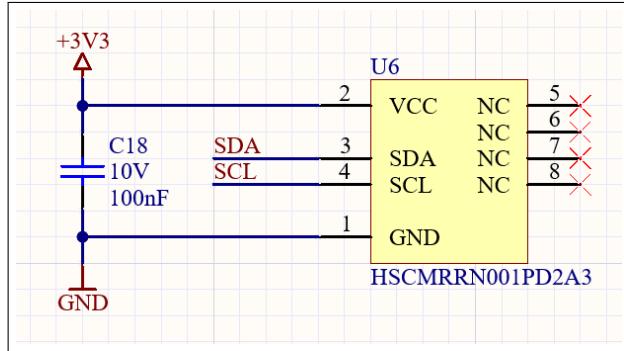
La plage de pression différentielle mesurable de ce capteur est de  $\pm 1$  psi, ce qui correspond à une pression de 6'894,76Pa.

Avec la formule (2) :

$$v(m/s) = \sqrt{\frac{2 * 6'894,76}{1.2}} = 107.197m/s = 385.91km/h$$

En utilisant la formule de la vitesse cité lors de la pré-étude, la vitesse maximale mesurable avec ce capteur est de 385.91km/h.

FIGURE 12 – HSCMRRN001PD2A3

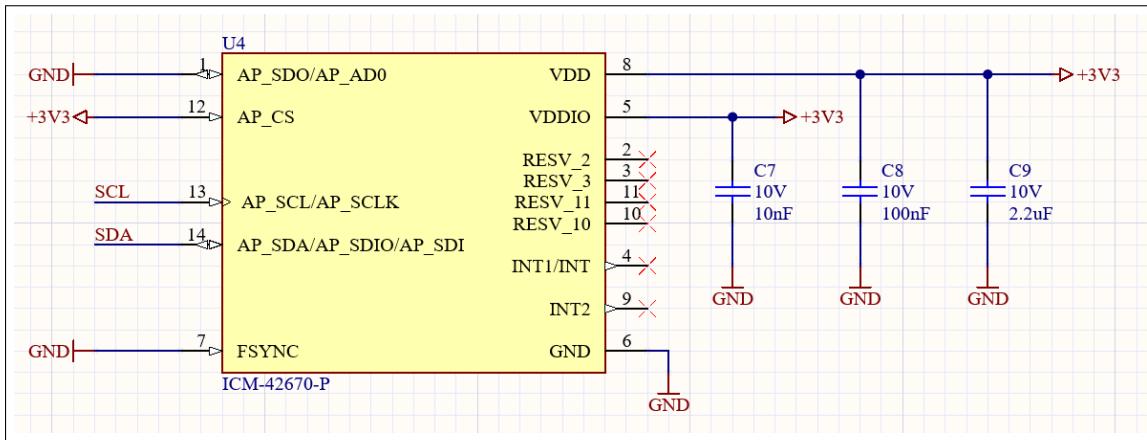


### 3.6 IMU

La centrale inertielles choisie est la ICM-42670-P du fabricant TDK. Il s'agit d'un composant intégrant un gyroscope 3 axes et un accéléromètre 3 axes et ayant une très faible consommation. La communication se fera au moyen d'une liaison série I2C, son adresse est b01101000 (0x68) puisque sa PIN AP\_AD0 est connecté volontairement à la masse.

Au niveau du gyroscope, les axes X, Y et Z ont une plage pleine échelle programmable de  $\pm 250$ ,  $\pm 500$ ,  $\pm 1000$  et  $\pm 2000$  degrés/sec. La plage des accélération en X, Y et Z est également programmable de  $\pm 2g$ ,  $\pm 4g$ ,  $\pm 8g$  et  $\pm 16g$ .

FIGURE 13 – ICM-42670-P



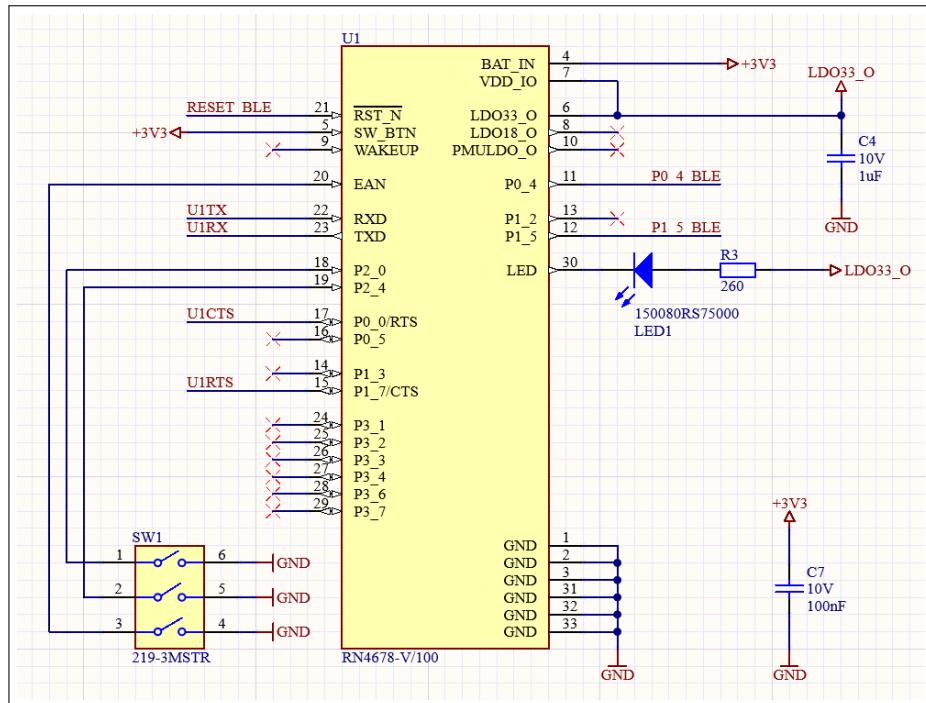
### 3.7 Module Bluetooth

Le module choisi, RN4678-V/100 de Microchip, possède deux technologie, le Bluetooth classique et le Bluetooth low-energy (BLE). Il est possible d'utiliser ce module en mode BLE, classique ou dual. Je l'ai sélectionné car il a été utilisé par plusieurs anciens élèves de l'ETML-ES et donc possède un driver connu et fonctionnel. Le DIP-switch SW1 à 3 entrées permet de modifier le mode opérationnel du module, c'est à dire qu'il est possible le mettre dans un mode fonctionnel, un mode écriture de l'EEPROM et test ou un mode écriture de la flash.

Le Baudrate par défaut du module est de 115200 Bauds d'où la fréquence du microcontrôleur particulière, expliquée au point MCU. Le débit du module peut aller jusqu'à 7kB/s en mode BLE et jusqu'à 32kB/s en mode classique. La LED va permettre d'afficher l'état du module lors du développement logiciel :

- Standby
- Link Back
- Low Battery
- Inquiry
- Link

FIGURE 14 – RN4678-V/100



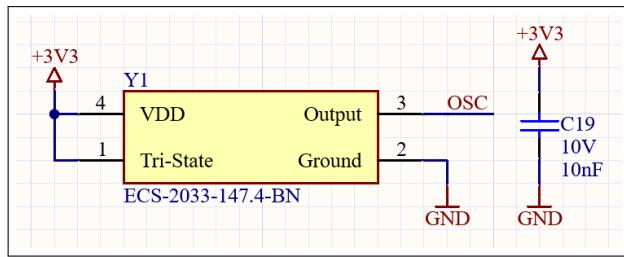
### 3.8 MCU

Le microcontrôleur choisi est le PIC32MX130F064B-I/SS, il possède au total 28 Pins. J'ai choisi celui-la en fonction du nombre d'entrées sorties disponibles mais également car il est en stock à l'ETML-ES. Ses principales fonctionnalités sont affichées dans le tableau ci-dessous.

PIC32MX130F064B	Valeur
Pins	28
Program Memory (kB)	64+3
Data Memory (kB)	16
Remappable Pins	20
Timers/Capture/Compare	5/5/5
UART	2
SPI/I2S	2
External Interrupts	5
Analog Comparators	3
USB On-The-Go (OTG)	No
I2C	2
PMP	Yes
10-bit 1 Msps ADC (Channels)	10
RTCC	Yes
I/O Pins	21
JTAG	Yes

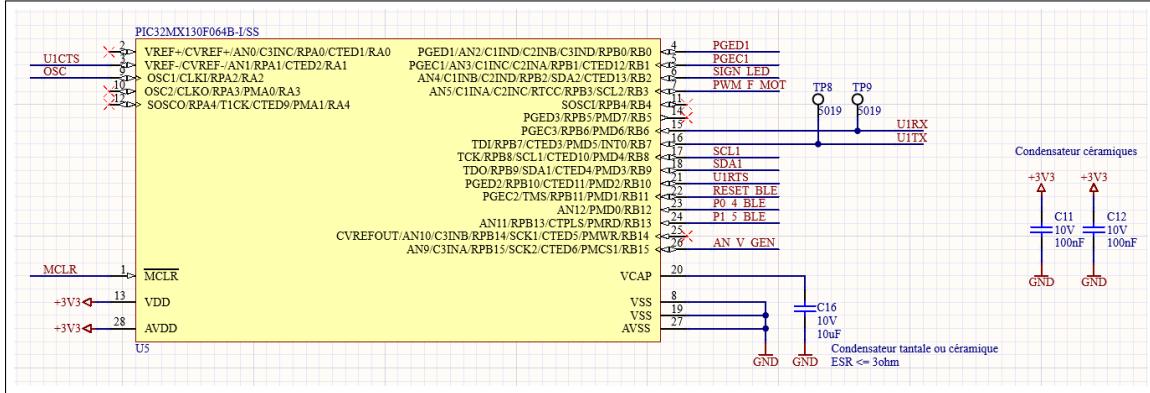
La fréquence du microcontrôleur proviendra d'un oscillateur externe de 14.7456MHz afin d'obtenir un multiple du Baudrate du module Bluetooth qui est de 115200Bd de base. 14.7456MHz correspond à  $128 \times 115200\text{Bd}$ .

FIGURE 15 – Oscillateur de 14.7456MHz



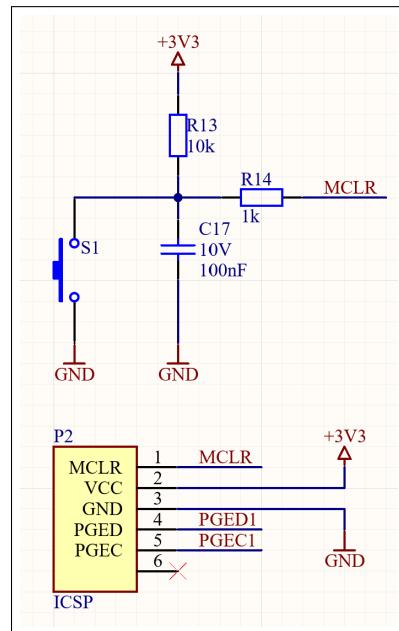
Les entrées et sorties du microcontrôleur ont été choisi à l'aide de l'utilitaire Harmony de MPLABX. Grâce à celui-ci il est plus simple de remapper les périphériques correctement.

FIGURE 16 – PIC32MX130F064B



Le circuit ci-dessous permet d'une part de programmer le microcontrôleur à l'aide d'un ICD3 ou ICD4 mais également de reset manuellement le MCU dans le cas où il y a un problème lors de la programmation.

FIGURE 17 – Circuit de reset et connecteur programmeur

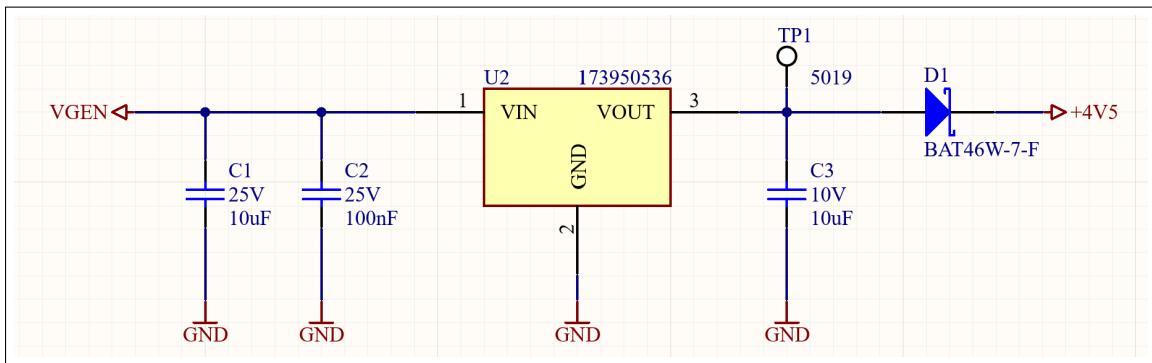


### 3.9 Alimentations

Le circuit d'alimentation sera composé de deux régulateurs de tension. La tension générée par la turbine entrera directement dans le premier régulateur de type Buck. La tension pouvant atteindre plus de 12V, j'ai choisi d'utiliser un régulateur Buck afin de ne pas dissiper toute la tension en chaleur. Sa sortie sera directement connectée à l'entrée du second régulateur. Celui-ci sera de type linéaire car la tension d'alimentation de la centrale inertielle ne doit pas être bruitée afin d'obtenir les mesures les plus précises possible. La diode Schottky permettra d'éviter qu'une tension n'aille dans la sortie du régulateur lorsque le module est alimenté par pile ou par USB et non pas par la turbine.

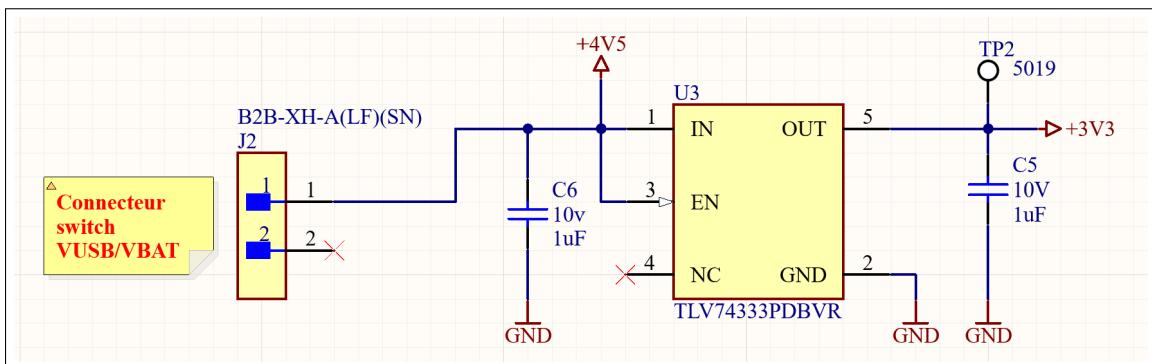
L'alimentation à découpage Wurth 173950536 est un composant intégrant l'alimentation et ses composants externes dans un même boîtier permettant ainsi de gagner de l'espace sur le PCB. La tension directe de la diode Schottky BAT45W-7-F est de 0.5V avec un courant d'environ 50mA ce qui baisse la tension entrant dans l'alimentation linéaire à 4.5V.

FIGURE 18 – 173950536



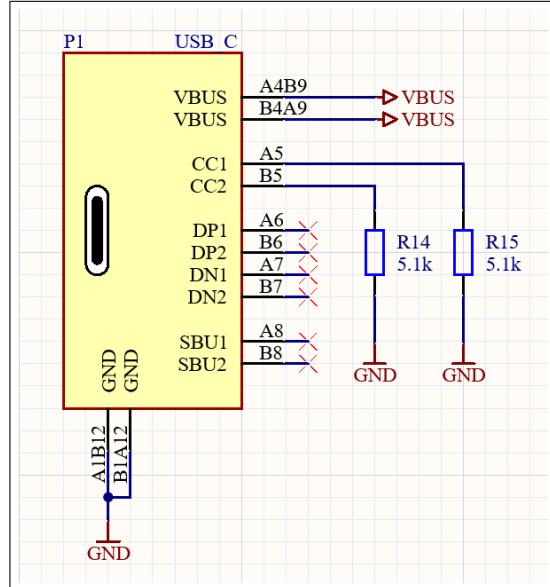
L'alimentation linéaire choisie est la TLV74333PDBVR du fabricant Texas Instrument. Son entrée sera connectée à un switch permettant de choisir soit la tension USB soit la tension des piles et comme expliqué précédemment à la sortie de l'alimentation Buck. Dans le cas où le circuit est alimenté au sol par un câble USB, le switch fera office de bouton ON/OFF.

FIGURE 19 – TLV74333PDBVR



Le connecteur USB sera de type C car c'est de nos jours le plus commun pour les nouveaux appareils. Les résistances de 5.1k branchées à la masse servent à indiquer qu'il s'agit d'un USB 2.0, bien qu'il n'y ait pas de communication.

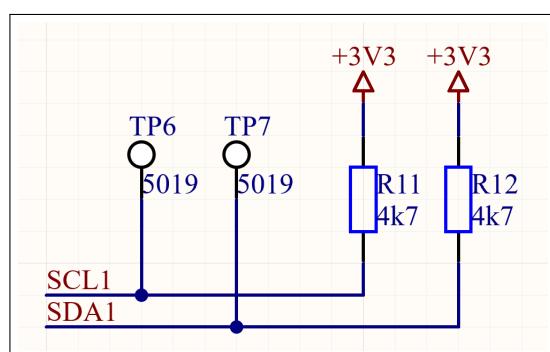
FIGURE 20 – Connecteur USB-C 2.0



### 3.10 Pull-up I2C

Les deux capteurs se trouvant sur le PCB utiliseront le bus I2C pour communiquer avec le MCU. Les deux lignes, SDA et SCL nécessitent l'ajout de pull-up afin de fonctionner correctement. Ces résistances peuvent avoir différentes valeurs en fonction de la vitesse de transmission. Initialement, j'ai envisagé d'utiliser des résistances de 10k ohms. Cependant, en raison de la configuration de communication en mode rapide (Fast mode), j'ai décidé de les remplacer par des résistances de 4.7k ohms.

FIGURE 21 – Pull-up I2C



## 4 Design mécanique

## 4.1 Introduction

J'ai pris la décision de fusionner le chapitre du design du circuit imprimé (PCB) avec celui du design mécanique, car ces deux aspects sont étroitement liés et cela contribue à une meilleure compréhension. Ce chapitre aborde les décisions prises lors de la conception du PCB sur Altium, notamment le placement et le routage des composants, ainsi que les choix de design du boîtier. J'ai adapté mon planning en développant simultanément la partie mécanique et celle du PCB. Cette approche m'a permis d'optimiser le volume et la résistance à l'air au maximum, tout en respectant les contraintes électriques. Le design du boîtier mécanique a été réalisé sur Onshape afin de générer des fichiers STL pour l'impression 3D.

## 4.2 Design

Le circuit a une forme rectangulaire mesurant 30 mm sur 120 mm, ce qui lui permet d'être inséré facilement dans une glissière prévue à cet effet dans le boîtier. Il est perforé par 2 trou de 11mm de diamètre. Ces trous permettent de faire passer deux entretoises métalliques et vis afin de fixer une boule RAM-Mounts au boîtier. Ce mode de fixation est connu et a déjà été testé par Monsieur Moreno, qui m'a donc conseillé de l'utiliser dans ce projet. Ce système de fixation offre une grande flexibilité en permettant de monter le système dans pratiquement toutes les positions et angles possibles tout en garantissant une grande fiabilité.

Sur ce circuit, les composants ont été disposés de manière à prendre en compte leur sensibilité aux perturbations. Ainsi, les capteurs sont placés à l'extrémité avant afin d'être alimentés par une source de tension propre, ce qui garantit des mesures fiables. D'autre part, la partie bruyante comprenant la MCU et l'alimentation à découpage est positionnée de l'autre côté. Le placement du capteur de pression différentielle à l'avant, permet également de réduire la longueur des tuyaux allant jusqu'au tube Pitot qui se trouve à l'extrémité avant du boîtier.

FIGURE 22 – Circuit vue de dessus

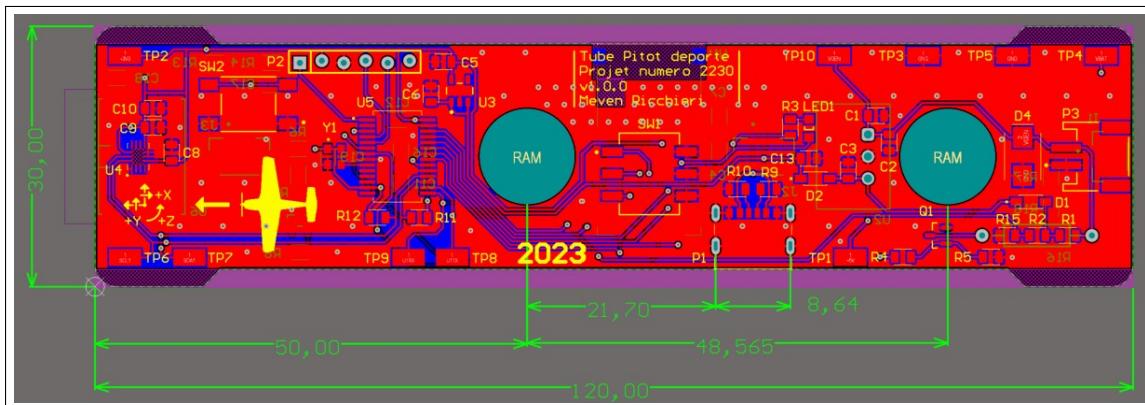
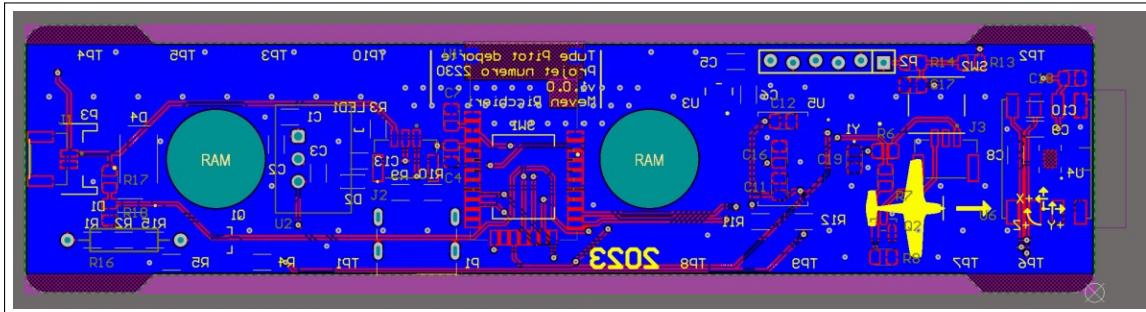
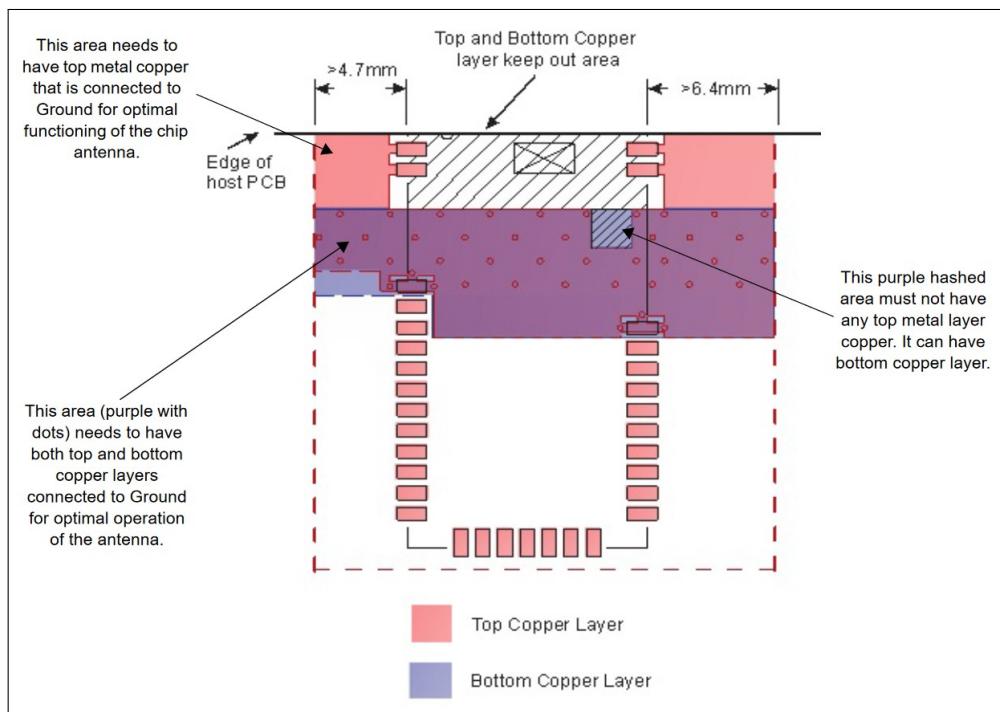


FIGURE 23 – Circuit vue de dessous



Concernant le module Bluetooth, celui-ci à du être placé de sorte à ce que son antenne soit en bordure de PCB afin d'optimiser sa portée. Le design suggéré par Microchip à également été utilisé afin de rendre le système plus robuste. Sur la Figure 24 se trouve le design suggéré par le fabricant, se trouvant dans la documentation technique du module Bluetooth RN4678.

FIGURE 24 – PCB suggéré par Microchip



Au niveau des largeurs de piste, les alimentations (VGEN, VBUS, VBAT, GND) ont une largeur de 0.3mm. Toutes les autres pistes ont une largeur de 0.254mm. Le circuit étant spécifiquement conçu pour une consommation de courant minimale, ne dépassant pas 50-60mA, les pistes sont largement surdimensionnées.

Les connecteurs sont placés de sorte à ce que le câblage soit le plus simple plus possible. Les 2 connecteurs responsables de l'alimentation se situent à l'arrière du circuit, de ce fait, ils sont au plus proche de leur source d'énergie et donc facilite le branchement. Le connecteur de la LED de signalisation et celui du switch ON/OFF sont situés sur la partie basse du PCB puisqu'ils se situent physiquement sur la partie basse du boîtier. Le connecteur USB type C est quant à lui sur le coté, ce qui permet d'y avoir accès lorsque le PCB est intégré dans le boîtier.

FIGURE 25 – Circuit vue de dessus

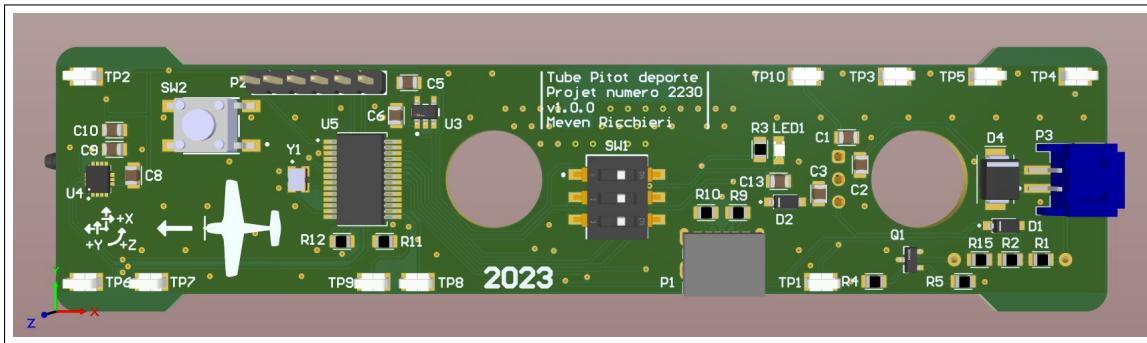
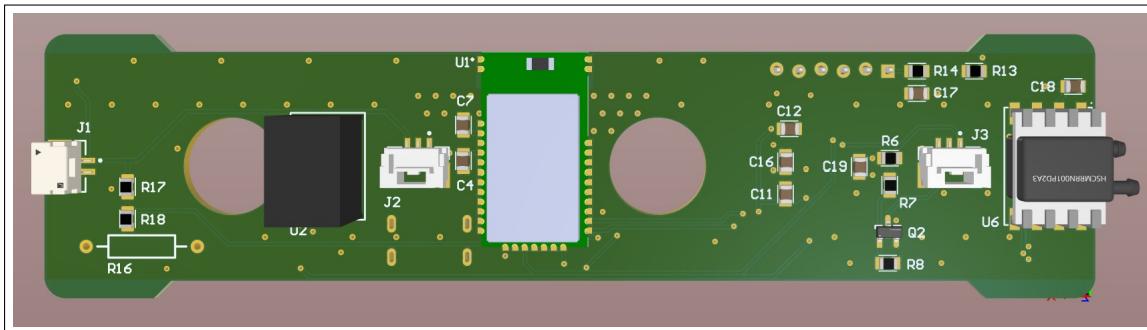


FIGURE 26 – Circuit vue de dessous



Sur la Figure 27 et sur la Figure 28, il est possible de voir comment le système complet s'intègre dans le boîtier avec les deux types d'alimentation, turbine et batteries. Les deux assemblages sont affichés en transparence, il est donc facile d'apercevoir toutes les parties du système. Les éléments importants sont énumérés ci-dessous :

1. Tube Pitot
2. Vis M2.5 permettant de fixer la partie A et B
3. Vis M5 avec entretoise de 11mm
4. Boule RAM-Mounts
5. Connecteur VGEN (tension venant du générateur)
6. Vis M2.5 permettant de fixer la partie B et C
7. Moteur DC (générateur)
8. Axe du moteur permettant de fixer la partie C et D
9. 2 Vis M3 permettant de fixer le moteur DC 7 au boîtier C
10. Switch ON/OFF
11. Connecteur VBAT (tension venant des batteries)
12. Connecteur switch ON/OFF
13. Connecteur USB type C
14. Circuit imprimé
15. Connecteur LED signalisation
16. LED de signalisation

FIGURE 27 – Assemblage complet avec turbine

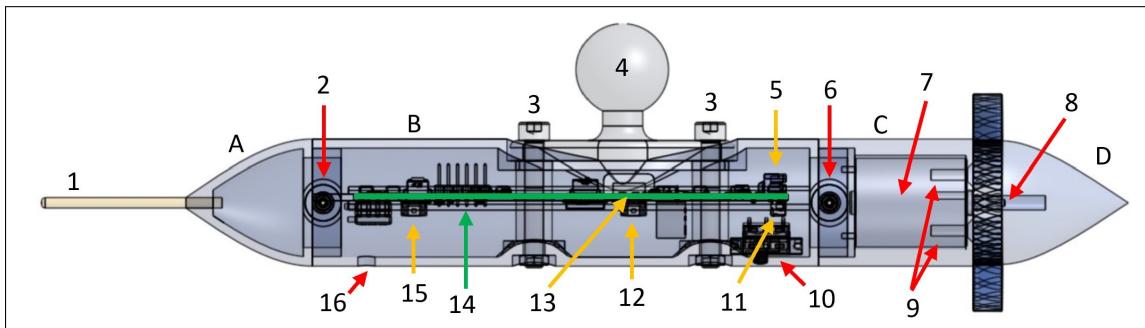
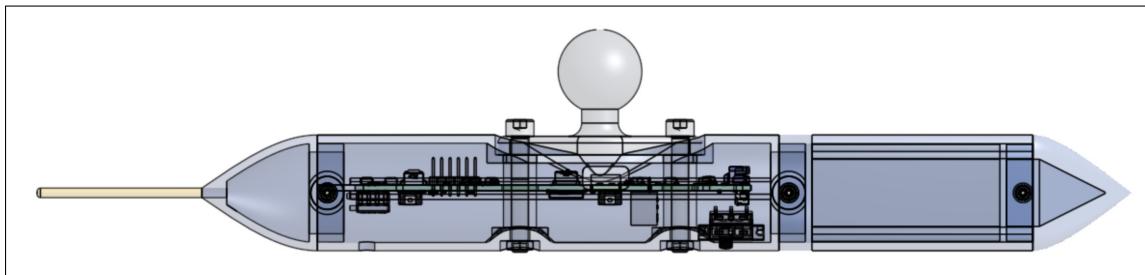
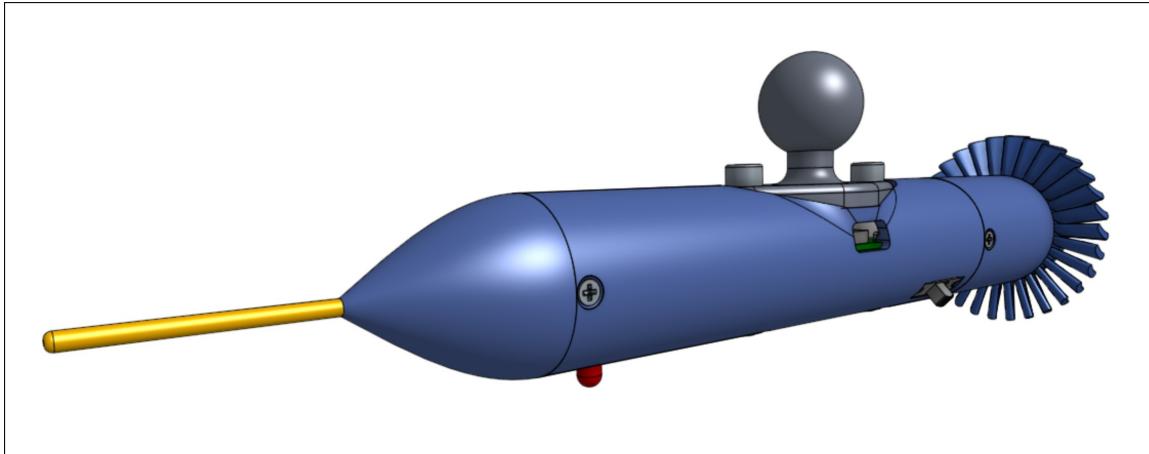


FIGURE 28 – Assemblage complet avec batteries



La Figure 29 présente le rendu final du système, intégré dans son boîtier. L'intégration est particulièrement simple et efficiente.

FIGURE 29 – Assemblage final



### 4.3 Modifications

Le design comporte quelques erreurs mineures qui doivent être corrigée dans le cas d'une nouvelle itération de ce projet en utilisant les mêmes documents Altium. Ces détails sont décrits ci-dessous.

#### 4.3.1 Circuit imprimé

Au niveau du circuit imprimé, le footprint du bouton reset dénommé SW2 n'est pas le bon. En conséquent, un switch différent a été brasé sur le PCB. Sans modification, le circuit fonctionnera tout de même. Le second point à corriger concerne les lignes de la communication USART. L'erreur va jusqu'à la modification du schéma, les PINs 21 (RESET\_BLE) et 22 (U1RTS) du MCU doivent être intervertis afin de pouvoir profiter du contrôle de flux hardware. Sur la version actuelle, il est fonctionnel mais nécessite un contrôle software manuel en activant et désactivant la PIN U1RTS, raison pour laquelle il n'est pas utilisé.

#### 4.3.2 Boîtier

Au niveau du boîtier, la partie turbine pose un problème au niveau de sa fixation à l'axe du moteur. Par manque de temps, cette partie n'a pas encore été résolu et donc nécessite de la recherche. Une amélioration serait également de passer de 4 batteries 1.5V à une seule batterie différente afin de rendre le système plus léger et plus équilibré entre l'avant et l'arrière.

## 5 Mise en service

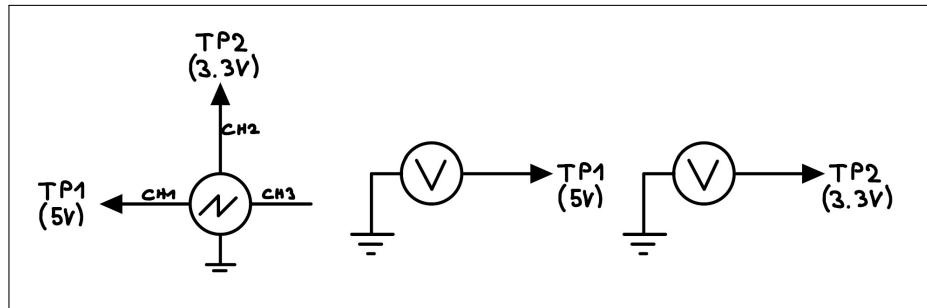
### 5.1 Introduction

Dans cette partie, j'ai réalisé une série de tests dans le but de valider le circuit imprimé (PCB). Notamment les sorties des alimentations, le courant consommé et le test de tous les périphériques telles que le module Bluetooth et les différents capteurs.

### 5.2 Test des alimentations

Les deux alimentations présentes sur le circuit fonctionnent correctement, le régulateur Buck 5V a une tension de sortie de 4.97V et le régulateur linéaire 3.3V a une tension de sortie de 3.29V. J'ai également contrôlé la qualité des sorties en mesurant la sortie de ces alimentations à l'oscilloscope.

FIGURE 30 – Schéma de mesure



### 5.3 Consommation

La consommation du circuit lorsque le MCU ne tourne pas est de 22.05mA avec la tension d'alimentation provenant de VBAT (6V). Lorsque le circuit est alimenté par VGEN (12V), la consommation est de 12.13mA.

### 5.4 Périphériques

Une fois les diverses configurations Harmony, décrites dans la section suivante de ce document, effectuées, j'ai procédé au test du capteur de pression différentielle, de la centrale inertuelle et du module Bluetooth. Tous ces composants communiquent correctement et renvoient des valeurs. Les mesures qui suivent ont été effectuées à l'aide d'un analyseur logique Saleae branché sur les points TP8 (PIC TX) et TP9 (PIC RX) pour l'USART et TP6 (SCL) et TP7 (SDA) pour l'I2C.

#### 5.4.1 Signaux USART

Sur les deux mesures ci-dessous, on peut observer que l'intégrité des signaux USART est très bonne. Sur la Figure 31, il s'agit des signaux provenant du microcontrôleur et allant au module Bluetooth. En utilisant la formule 5 il est possible de calculer la vitesse de la communication. 4 Bytes sont reçus en l'espace de 339.16us ce qui correspond à une vitesse de 114989 bit/s. L'erreur de vitesse est calculable à l'aide de la formule 6. L'erreur de Baudrate du PIC est d'environ 0.18% par rapport à la vitesse désirée ce qui est très clairement bon.

Sur la Figure 32, il s'agit des signaux allant du module Bluetooth au PIC. 5 Bytes sont reçus en l'espace de 416.48us ce qui correspond à une vitesse de 117653 bit/s. L'erreur de vitesse est d'environ 2.13% par rapport à la vitesse voulue. Ce pourcentage d'erreur n'est pas exceptionnel et ne peut malheureusement pas être corrigé. Cependant, malgré cette légère erreur, la communication fonctionne correctement et ne pose pas de problème significatif.

$$\text{Baudrate}[bit/s] = \frac{N_{bit}}{T_{tot}} \quad (5)$$

$$\text{erreurBaudrate}[\%] = \frac{Bd_{voulu} - Bd_{mesur}}{Bd_{mesur}} * 100 \quad (6)$$

FIGURE 31 – Mesure trame USART PIC → BT

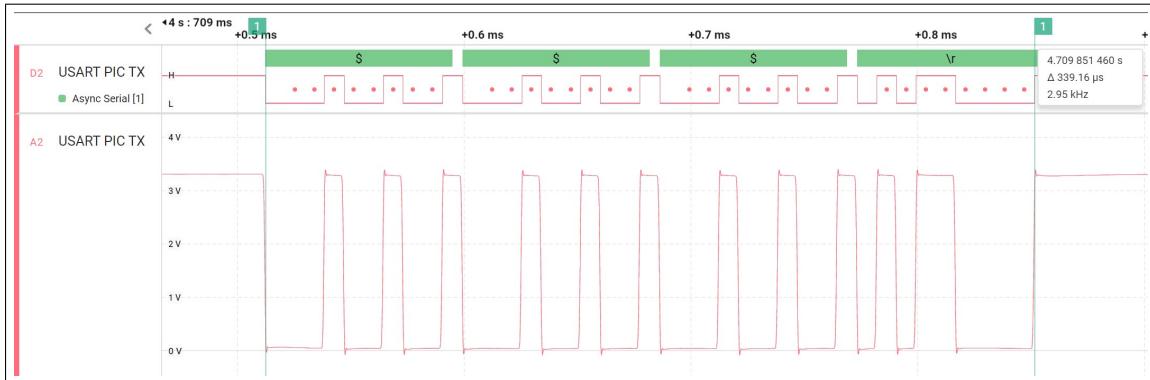
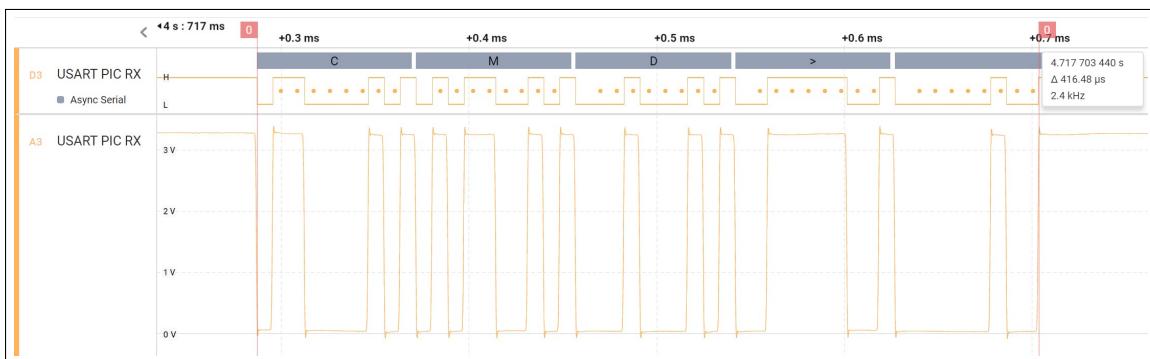


FIGURE 32 – Mesure trame USART BT → PIC



### 5.4.2 Signaux I2C

Sur les mesures qui suivent, nous pouvons observer que l'intégrité des signaux I2C n'est pas exceptionnelle. La partie haute des signaux n'atteint pas tout à fait la tension VCC et la partie basse n'atteint pas non plus le GND. La solution à ce problème est soit de diminuer vitesse en passant de 400kHz à 100kHz ou en réduisant la valeur des pull-up. Toutefois la communication fonctionne parfaitement car les seuils de déclenchement sont quand même franchis. Bien que cela fonctionne, j'ai décidé de réduire la vitesse de communication. L'effet de ce changement est visible sur la Figure 35.

FIGURE 33 – Mesure trame I2C à 400kHz

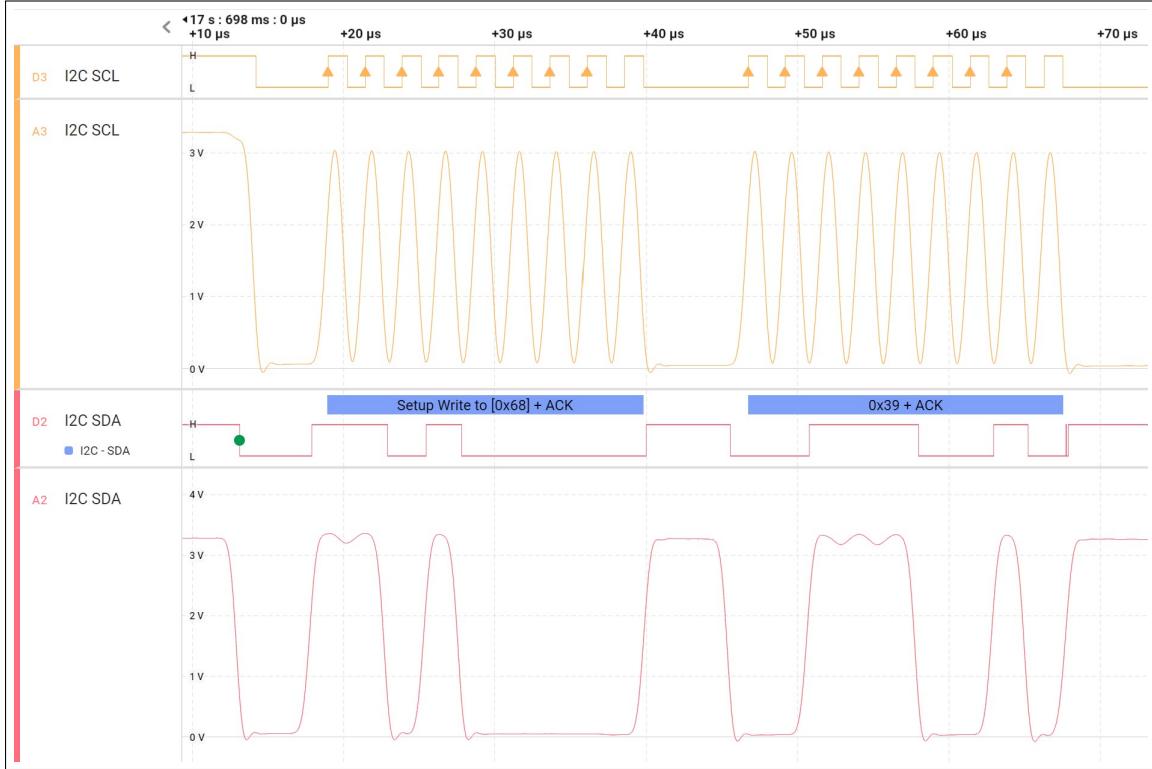


FIGURE 34 – Mesure d'une période SCL à 400kHz

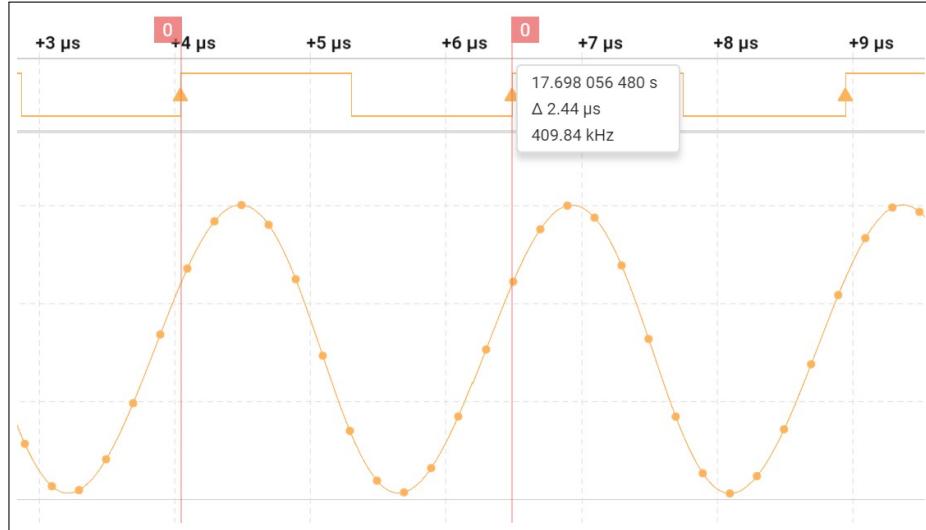
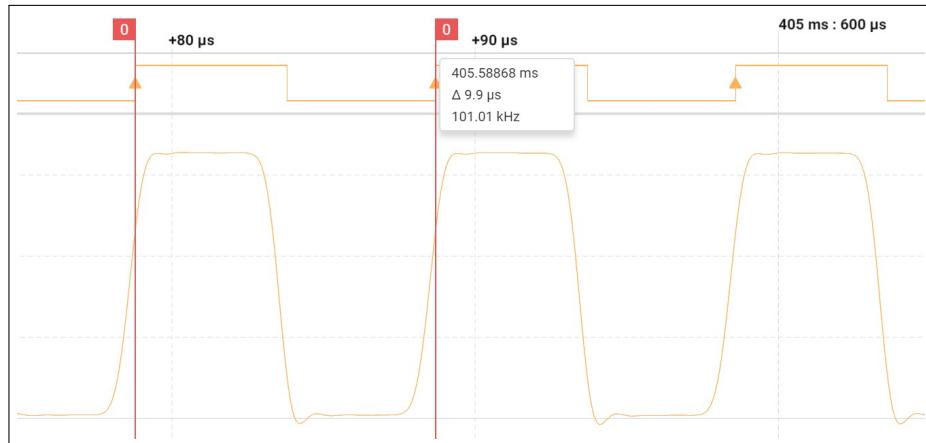


FIGURE 35 – Mesure trame I2C à 100kHz



## 5.5 Améliorations possibles

Après avoir effectué plusieurs tests d'alimentation du circuit à l'aide de la turbine, un problème potentiel peut se présenter. Lorsque la turbine commence à générer une tension suffisamment élevée pour activer les alimentations, le courant disponible peut ne pas être suffisant pour alimenter tous les composants. Par conséquent, cela entraîne une chute de tension et peut provoquer le blocage du MCU. L'idée serait d'implémenter un amorceur de démarrage à l'aide d'un comparateur déclenchant un MOSFET.

## 6 Software MCU

### 6.1 Introduction

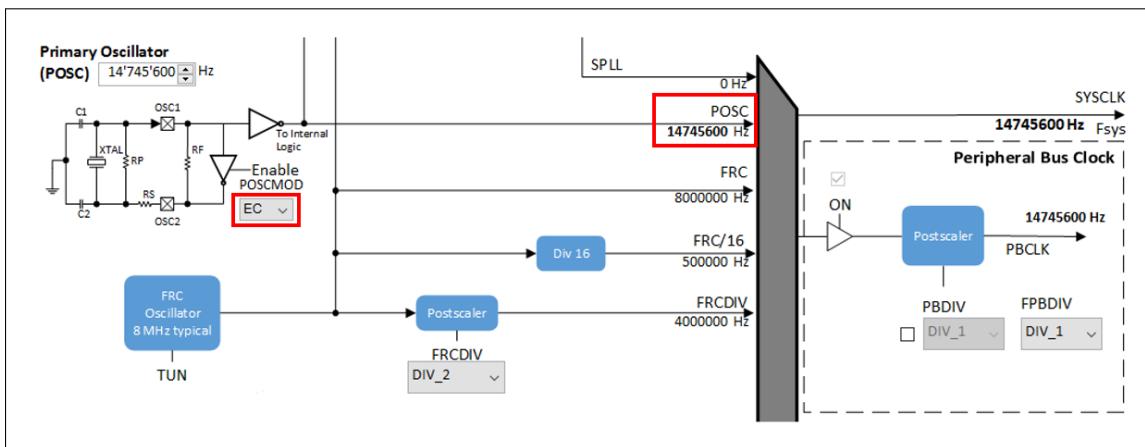
Le développement du logiciel du MCU a pris considérablement plus de temps que prévu, principalement en raison de la complexité rencontrée lors de l'intégration du driver de la centrale inertuelle, celle-ci composée de nombreux fichiers et étant pas très compréhensible. L'emplacement du projet se situe sous : C:/microchip/harmony/v2\_06/apps/PROJ

### 6.2 Paramétrage Harmony

#### 6.2.1 Fréquence d'horloge

Comme décrit dans la pré-étude, le microcontrôleur est cadencé par un oscillateur externe à une fréquence de 14.7456MHz. Cela permet d'optimiser la communication USART puisque cette fréquence est un multiple de 115200Hz ( $115200 * 128 = 14745600\text{Hz}$ ), fréquence à laquelle le MCU communique avec le module Bluetooth. Le diagramme d'horloge présenté dans la Figure 36 illustre la configuration du microcontrôleur. La fréquence du système (SYSCLK) provient de l'oscillateur primaire (POSC) en tout en spécifiant qu'il s'agit d'un oscillateur externe (EC).

FIGURE 36 – Diagramme d'horloge



### 6.2.2 Timers

Le calcul utilisé afin d'obtenir le nombre de ticks (Timer period) des trois timers utilisés est le suivant :

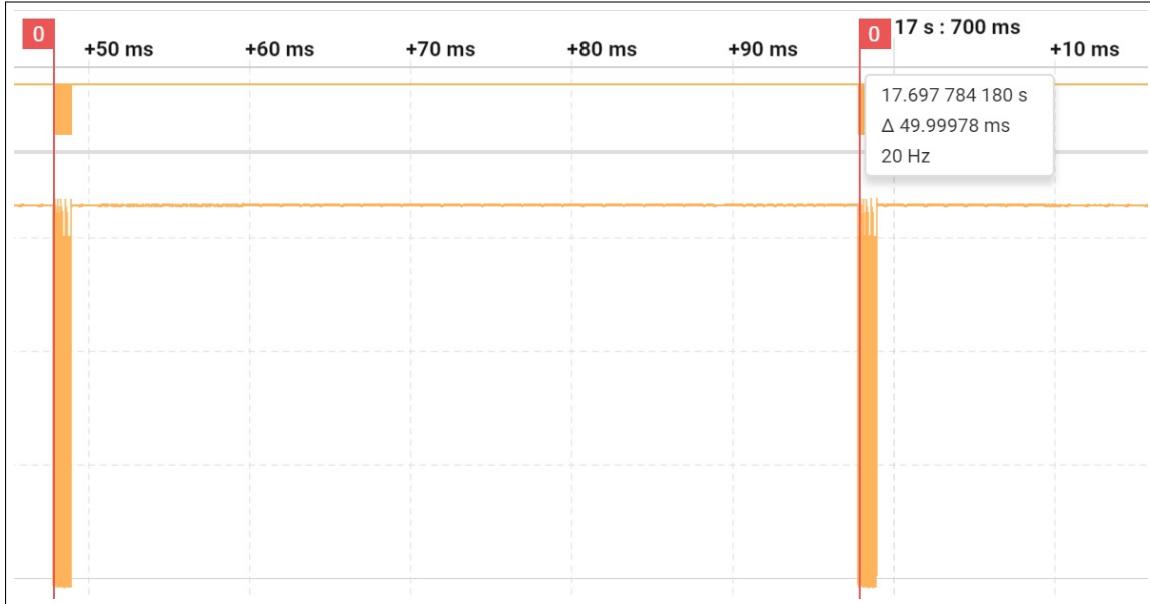
$$N_{Ticks}[-] = \frac{f_{SYS}}{f_{Timer} * PRESCL} - 1 \quad (7)$$

	Timer1	Timer2	Timer3
Instance	1	2	3
ID	2	4 et 5	3
Mode	16 bit	32 bit	16 bit
Fréquence	20Hz	3.43mHz	225Hz
Prescaler	256	1	1
Timer period	2879	4294967295	65535
Interruption priorité	Level 2	Level 0	Level 0
Enabled	YES	YES	NO

Le Timer1 a pour but de cadencer le programme principal, c'est à dire que la machine d'état principale est exécutée 20 fois par secondes. Le Timer2 est utilisé dans les fonctions de gestion de temps, nécessaire au bon fonctionnement de l'IMU et au module Bluetooth. Le Timer3 est quant à lui désactivé, il a été utilisé afin de réaliser différents tests.

Sur la Figure 37, nous pouvons voir que les communications s'effectuent bien toutes les 50ms (20Hz).

FIGURE 37 – Mesure cadence Timer1



### 6.2.3 Entrées sorties

Les PINs ont été configurées de sorte à respecter le schéma du circuit. Ci-dessous (Figure 38), se trouve la configuration de toutes les entrées et sorties du microcontrôleur. Dans la colonne "Function" nous pouvons voir à quel périphérique sont connectés les PINs. Les fonctions nommées "Available" sont des PINs son utilisées.

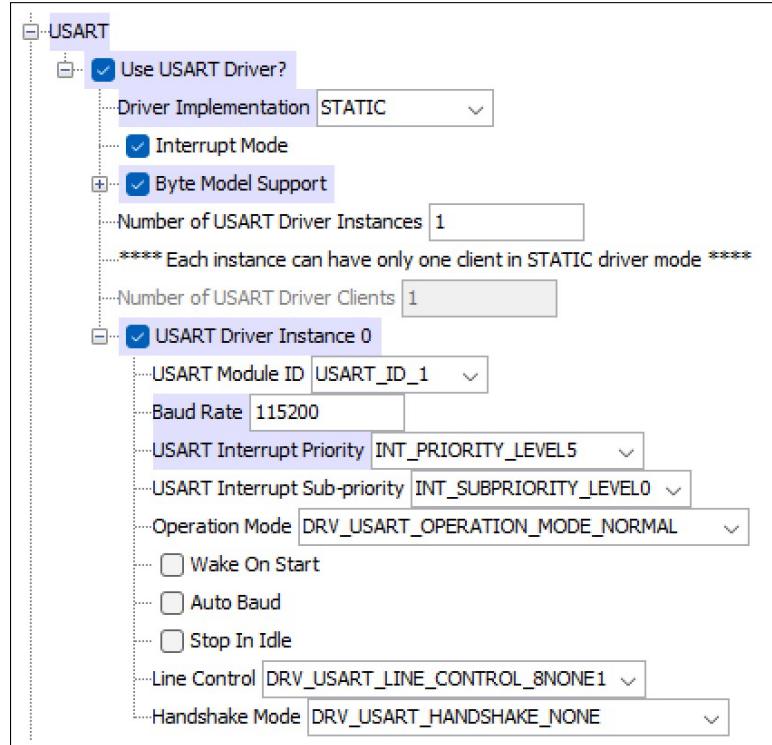
FIGURE 38 – Paramétrage des PINs

Pin Number	Pin ID	Voltage Tolerance	Name	Function	Direction (TRIS)	Mode (ANSEL)
1	MCLR	5V			In	Dig...
2	RA0			Available	In	An...
3	RA1		U1CTS_MANUAL_PIC_IN	GPIO_IN	In	Dig...
4	RB0			Available	In	An...
5	RB1			Available	Out	Dig...
6	RB2		SIGN_LED	GPIO_OUT	Out	Dig...
7	RB3			Available	In	An...
8	VSS				In	Dig...
9	RA2			Available	In	An...
10	RA3			Available	In	An...
11	RB4		P0_4	GPIO_IN	In	Dig...
12	RA4		P1_5	GPIO_IN	In	Dig...
13	VDD				In	Dig...
14	RB5	5V	TEST_OUT	GPIO_OUT	Out	Dig...
15	RB6	5V	U1RX	U1RX	n/a	Dig...
16	RB7	5V	U1TX	U1TX	n/a	Dig...
17	RB8	5V	SCL1	SCL1	n/a	Dig...
18	RB9	5V	SDA1	SDA1	n/a	Dig...
19	VSS				In	Dig...
20	VCAP				In	Dig...
21	RB10	5V	RESET_BLE	GPIO_OUT	Out	Dig...
22	RB11	5V	U1RTS_MANUAL_PIC_OUT	GPIO_OUT	Out	Dig...
23	RB12			Available	In	An...
24	RB13			Available	In	Dig...
25	RB14		AN_V_BAT	AN10	n/a	An...
26	RB15		AN_V_GEN	AN9	n/a	An...
27	AVSS				In	Dig...
28	AVDD				In	Dig...

#### 6.2.4 Communications série

La communication USART a été configurée depuis l'utilitaire Harmony. Les paramètres sont affichés dans la Figure 39.

FIGURE 39 – Paramétrage de la communication USART



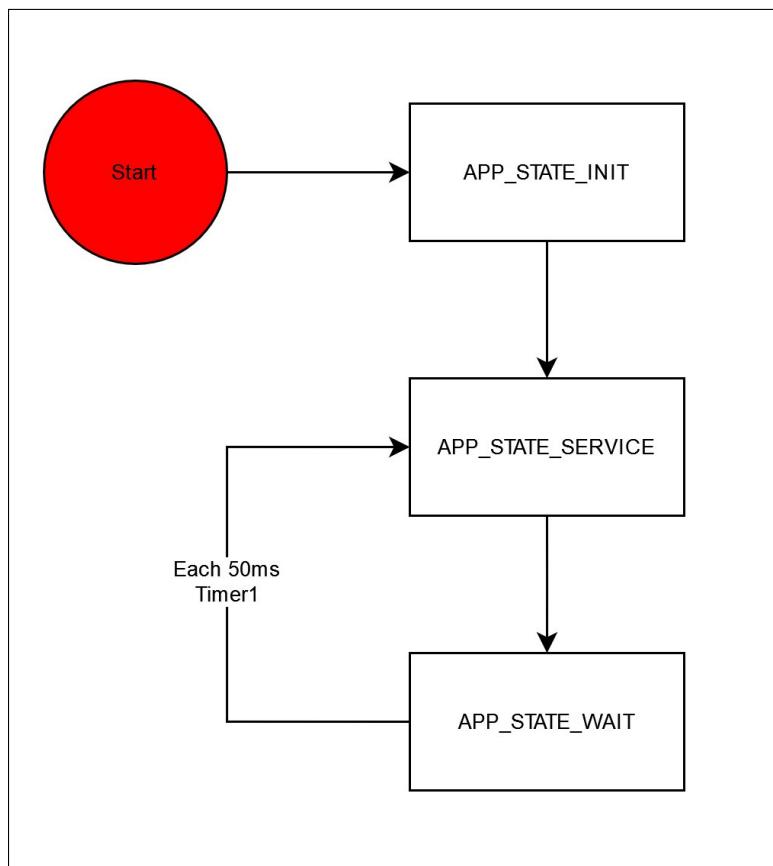
L'I2C à quant-à lui n'a pas été configuré depuis Harmony mais à l'aide du fichier C nommé "Mc32\_I2cUtilCCS". Le paramètre principal, la vitesse, est réglée en SLOW mode (100kHz) puisque le FAST mode pose quelques problèmes de distorsion.

## 6.3 Flowcharts

### 6.3.1 Machine d'état principale

La Figure 40 présente le diagramme de flux de la machine d'état principale, qui est cadencée par le Timer1 à une fréquence de 20Hz. Au début, tous les périphériques du système sont initialisés avant d'entrer pour la première fois dans l'état de service. Après chaque passage dans l'état de service, le programme entre dans l'état d'attente, où le MCU ne fait rien, jusqu'à ce que l'interruption du Timer1 déclenche à nouveau l'état de service.

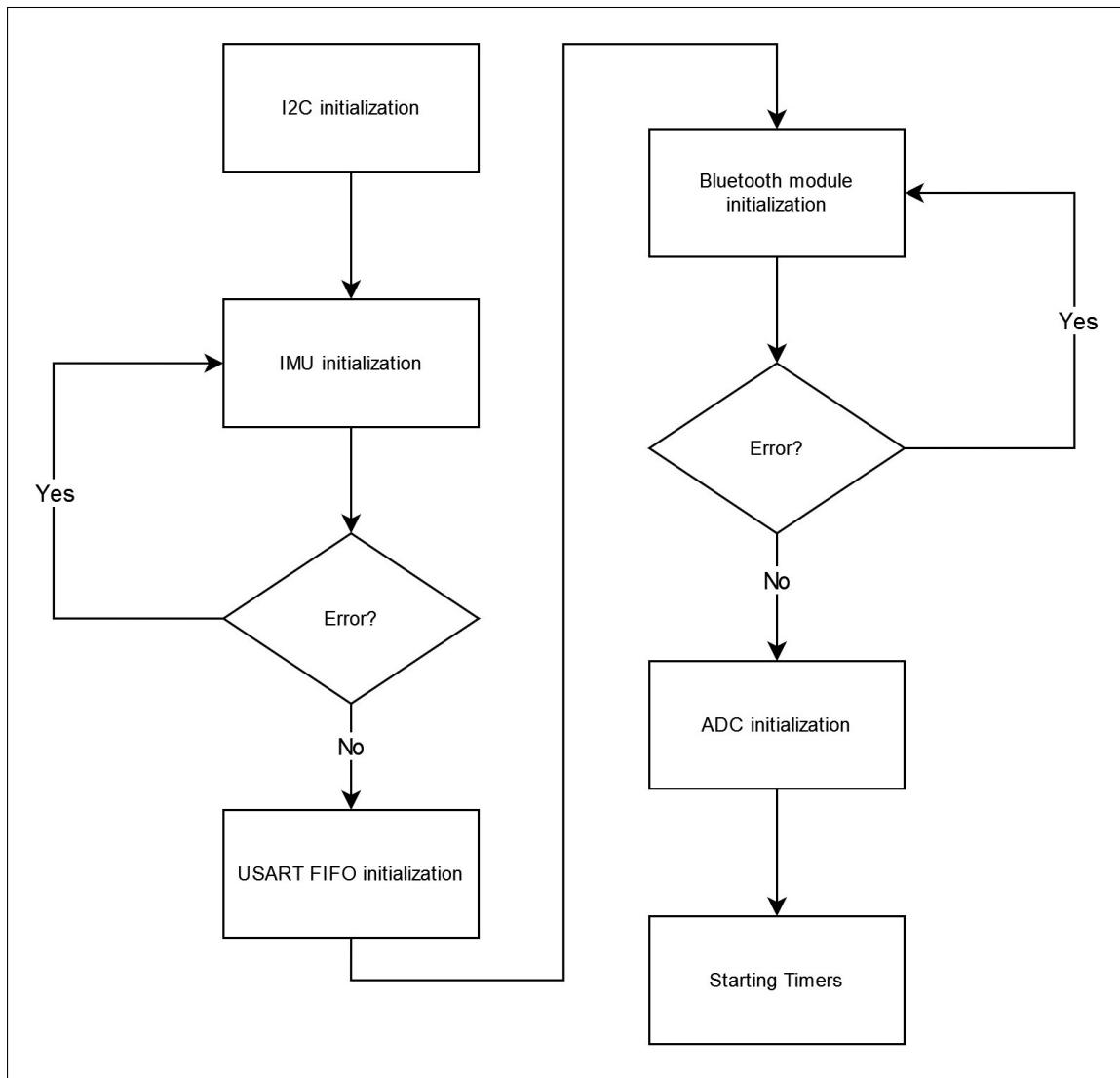
FIGURE 40 – Machine d'état principale



### 6.3.1.1 Etat d'initialisation

La Figure 41 montre toute la phase d'initialisation du système. Les deux composants principaux bloquent le programme tant qu'ils ne sont pas correctement initialisé. Cela permet de garantir que le système fonctionne correctement.

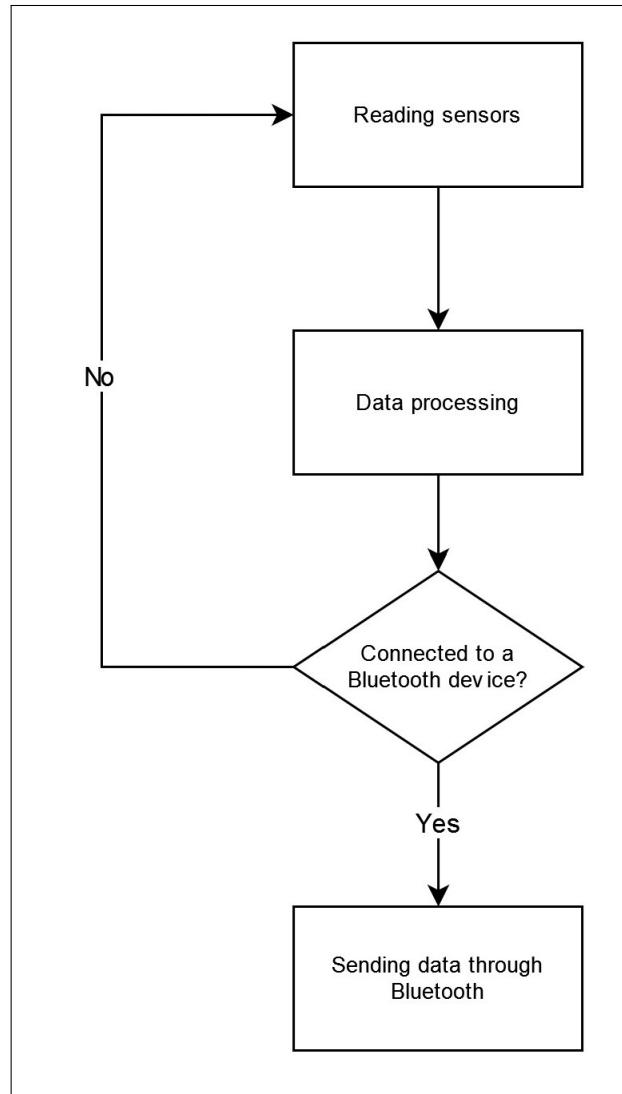
FIGURE 41 – Etat d'initialisation



### 6.3.1.2 Etat de service

L'état de service contient une sous-machine d'état composée de trois autres états. Le premier état a pour but de lire les capteurs, le suivant effectue les traitements nécessaires, et enfin le dernier état envoie les données au module Bluetooth qui les transmet à l'appareil Android connecté.

FIGURE 42 – Etat de service



## 6.4 Codes C

La mise en forme de la trame ce fait dans cette fonction ci-dessous et permet de faciliter le traitement des données aux niveau de l'application Android.

---

```
inline void frameFormatting(char* a_dataToSend, const SENS_DATA* sensData){

    // Saves all data into a simple frame
    // Speed in [km/h]
    // Gyros in [dps]
    // Angles in [degrees]
    // Accelerations in [g]
    // VB and VG in [V]
    sprintf(a_dataToSend, "S=%03d GX=%+.02f GY=%+.02f GZ=%+.02f GAX=%+.02f "
            "GAY=%+.02f GAZ=%+.02f AX=%+.02f AY=%+.02f AZ=%+.02f VB=%+.02f "
            "VG=%+.02f\n\r",
            sensData->velocity,
            sensData->gyroX, sensData->gyroY, sensData->gyroZ,
            sensData->GyrAngleX, sensData->GyrAngleY, sensData->GyrAngleZ,
            sensData->accelX, sensData->accelY, sensData->accelZ,
            sensData->batVoltage, sensData->genVoltage);
}
```

---

La fonction ci-dessous permet de lire les statuts envoyé par le module Bluetooth. Pour l'instant elle effectue uniquement un

---

```
void USART1_Callback_Function(void){

    char a_received[50];
    char* result;

    // Gets new data from FIFO
    getUsartData(&a_received[0]);

    // If "<RFCOMM_OPEN>" is present in the array received
    result = strstr(a_received, "<RFCOMM_OPEN>");
    if(result != NULL){

        appData.isBluetoothDiscoverable = false;
        appData.isBluetoothConnected = true;
    }

    // If "<RFCOMM_CLOSE>" is present in the array received
    result = strstr(a_received, "<RFCOMM_CLOSE>");
    if(result != NULL{

        // turnOnDiscoverBT();
        appData.isBluetoothConnected = false;
    }
}
```

---

La fonction de callback ci-dessous est appelée après chaque lecture de la centrale inertuelle. Elle est appelée automatiquement par le driver fourni par le fabricant (TDK InvenSense) lorsque les valeurs sont disponibles dans la structure "event". Des offsets on été ajoutés manuellement

---

```
void imu_callback(inv_imu_sensor_event_t *event){

    // Transforms 16bits values into degrees and saves them in the sensor data
    // structure
    // 250 dps
    sensData.gyroX = (event->gyro[0])/250 + 1; // + offset
    sensData.gyroY = (event->gyro[1])/250 + 1; // + offset
    sensData.gyroZ = (event->gyro[2])/250;

    // Reads and transforms 16bits values into
    // Transforms 16bits values into g acceleration and saves them in the sensor
    // data structure (8192 bits per g)
    sensData.accelX = (float)(event->accel[0])/8192.0;
    sensData.accelY = (float)(event->accel[1])/8192.0;
    sensData.accelZ = (float)(event->accel[2])/8192.0 + 0.075; // + offset

    // Calculate the angle with the gyro values
    // 0.05 correspond to the period between each reading 1/20Hz = 0.05s
    sensData.GyrAngleX += sensData.gyroX * 0.05;
    sensData.GyrAngleY += sensData.gyroY * 0.05;
    sensData.GyrAngleZ += sensData.gyroZ * 0.05;
}
```

---

Le reste du code se trouve en annexe. Dans ce sous chapitre, quelques fonctions importants sont brièvement expliquées. Le reste du code se trouve en annexe car tout est bien commenté et facile à comprendre.

## 7 Software Android

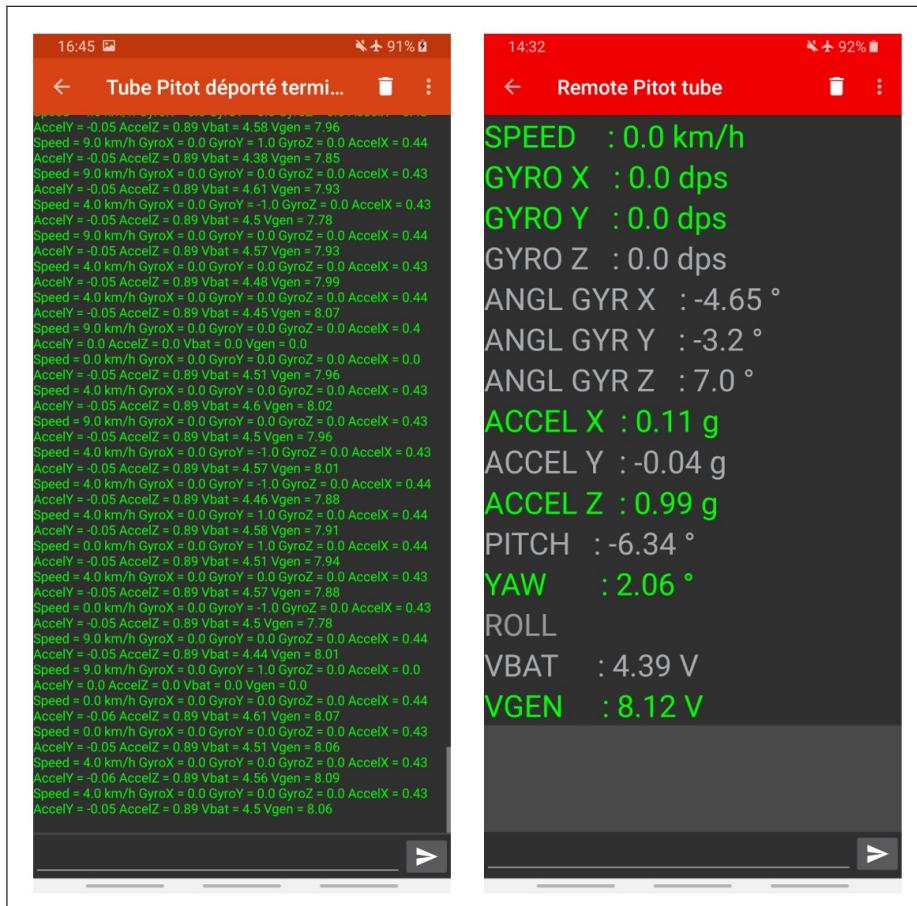
### 7.1 Introduction

Cette partie de rapport n'est pas documenté car l'application Android n'est pas terminé et nécessite encore un certain temps de travail. En premier lieu, J'ai penser développer une application de A à Z. Cependant la difficulté était bien supérieure à celle à laquelle je m'attendais. En conséquent, j'ai fork le repos github d'un développeur ayant réalisé un terminal Bluetooth afin de modifié son application. Le lien de son github : <https://github.com/kai-morich/SimpleBluetoothTerminal>

Un aperçu des deux applications est disponible ci-dessous. La capture d'écran de gauche correspond à l'application de base provenant de github et celle de droite est celle modifié pour le projet.

### 7.2 Aperçu des interfaces

FIGURE 43 – Aperçu des deux applications



## 8 Résultat final

### 8.1 Introduction

Dans ce chapitre, il est possible de voir à quel point le projet est avancé et ce qu'il reste à réaliser pour qu'il soit totalement abouti.

FIGURE 44 – Photo 1 du résultat final



FIGURE 45 – Photo 2 du résultat final



## 8.2 État d'avancement

L'état d'avancement actuel au jour du 15 juin 2023 est décrit ci-dessous. Le logiciel intégré dans le microcontrôleur est capable de réaliser toutes les fonctions énumérées ci-dessous :

1. Mesure de la vitesse air entre 0 et 386km/h
2. Mesure des accélérations X, Y et Z
3. Mesure des gyroscopes X, Y et Z
4. Mesure de la tension des batteries et du générateur
5. Envoi des données au travers d'une communication Bluetooth Classique

Le logiciel intégré dans l'appareil Android est capable de réaliser toutes les fonctions énumérées ci-dessous :

1. Connexions à un appareil appairé (appairage par le bias du système)
2. Réception et traitement de toutes données envoyées par l'appareil conçu
3. Affichage de toutes les valeurs brutes reçues sous forme de variables
4. Affichage de valeurs traitées par le logiciel Android

## 8.3 Tâches restantes

Les tâches incomplètes ou non débutées sont énumérées ci-dessous :

1. Terminer le développement de l'application Android
  - (a) Cacher les données non essentielles
  - (b) Moyenner les données
  - (c) Optimiser la facilité de lecture des données
  - (d) Ajouter la fonction de mise à zéro (offsets)
  - (e) Ajouter la fonction de logging des données
2. Effectuer un test réel en vol
3. Dessiner la nouvelle turbine avec un système de fixation plus robuste

## 9 Conclusion

En conclusion, ce projet de développement d'un système de détection de l'angle d'incidence et de la vitesse au décrochage d'un avion présente des défis techniques importants. L'objectif était de concevoir un système flexible, pouvant être installé sur différents types d'avions, tout en évitant l'interférence du flux d'air provenant de l'hélice pour assurer des mesures précises de vitesse. De plus, la contrainte de miniaturisation était primordiale pour minimiser la traînée et respecter une limite de poids de 500g.

La réalisation de ce projet a été une expérience inestimable qui m'a permis d'élargir mes compétences en travaillant avec des composants variés que je n'avais jamais manipulés auparavant. Cette opportunité a renforcé ma détermination et mon dévouement envers le projet, car il incarnait une occasion unique de contribuer à un domaine qui suscite en moi une profonde inspiration et fascination : l'aéronautique. Travailler sur ce projet a été une expérience incroyablement gratifiante et motivante, comblant mes aspirations en tant qu'enthousiaste passionné d'aéronautique.

Je tiens à exprimer mes sincères remerciements à M. Castoldi et à M. Moreno pour leur précieuse assistance tout au long de ce projet.

Lausanne, le 16 juin 2023

Meven Ricchieri

## 10 Annexes

- Références
- Cahier des charges
- Schéma électrique
- Fichiers C et H réalisés ou modifiés
- Fichier Java
- Documents CAO
- BOM
- Ficher de modifications
- Journal de travail
- Planning
- Mode d'emploi
- Affiche

## Références

- [1] *Centrale à inertie.* fr. Page Version ID : 192388914. Mars 2022. URL : [https://fr.wikipedia.org/w/index.php?title=Centrale\\_%C3%A0\\_inertie&oldid=192388914](https://fr.wikipedia.org/w/index.php?title=Centrale_%C3%A0_inertie&oldid=192388914) (visité le 07/12/2022).
- [2] *maxon Motors as Generators.* en-US. URL : <https://support.maxongroup.com/hc/en-us/articles/360004496254-maxon-Motors-as-Generators> (visité le 04/12/2022).
- [3] *Pression statique.* fr. Page Version ID : 169954162. Avr. 2020. URL : [https://fr.wikipedia.org/w/index.php?title=Pression\\_statique&oldid=169954162](https://fr.wikipedia.org/w/index.php?title=Pression_statique&oldid=169954162) (visité le 03/12/2022).
- [4] *Pression totale.* fr. Page Version ID : 198361997. Nov. 2022. URL : [https://fr.wikipedia.org/w/index.php?title=Pression\\_totale&oldid=198361997](https://fr.wikipedia.org/w/index.php?title=Pression_totale&oldid=198361997) (visité le 03/12/2022).
- [5] *Tube de Pitot.* fr. Page Version ID : 198892898. Nov. 2022. URL : [https://fr.wikipedia.org/w/index.php?title=Tube\\_de\\_Pitot&oldid=198892898](https://fr.wikipedia.org/w/index.php?title=Tube_de_Pitot&oldid=198892898) (visité le 04/12/2022).

# Projet ETML-ES- Cahier des charges

<b>Tube Pitot déporté</b>
<b>2230</b>

A remplir par l'initiateur / porteur de projet

Entreprise/Client:		Département:	
Demandé par (Prénom, Nom):		Date:	

A remplir par le gestionnaire de projet (étudiant)

Auteur (ETML-ES):	Meven Ricchieri	Filière:	SLO
		Date:	16.11.2022

## 1 But du projet

Le but du projet consiste à développer un système permettant de détecter l'angle d'incidence et la vitesse au décrochage d'un avion. La fixation de ce système doit être flexible afin de pouvoir l'installer sur différents types d'avions. L'emplacement de fixation ne doit pas se trouver dans le flux d'air provenant de l'hélice afin d'éviter que la mesure de vitesse ne soit faussée. Il doit également être miniaturisé au maximum afin de produire le minimum de trainée possible et de ne pas dépasser un poids de 500g. Les données acquises par les capteurs doivent être transmises de la partie déportée à un appareil Android se trouvant dans le cockpit de l'avion au travers une communication sans fil. L'appareil Android doit traiter et afficher les données reçues, si possible graphiquement (optionnel).

### 1.1 Schéma de principe

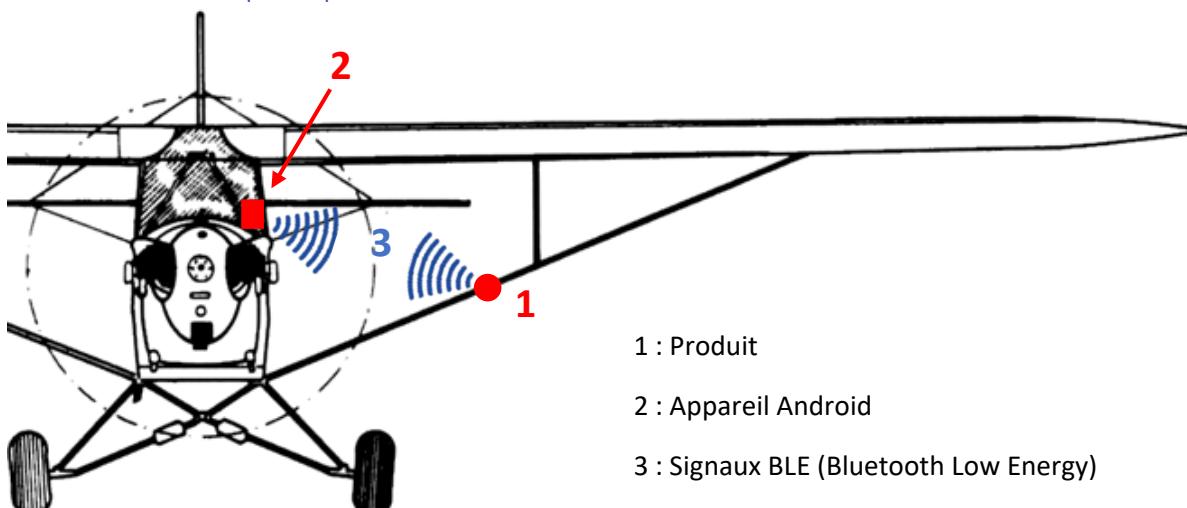


Figure 1 -

## 2 Spécifications du projet

### 2.1 Schéma bloc

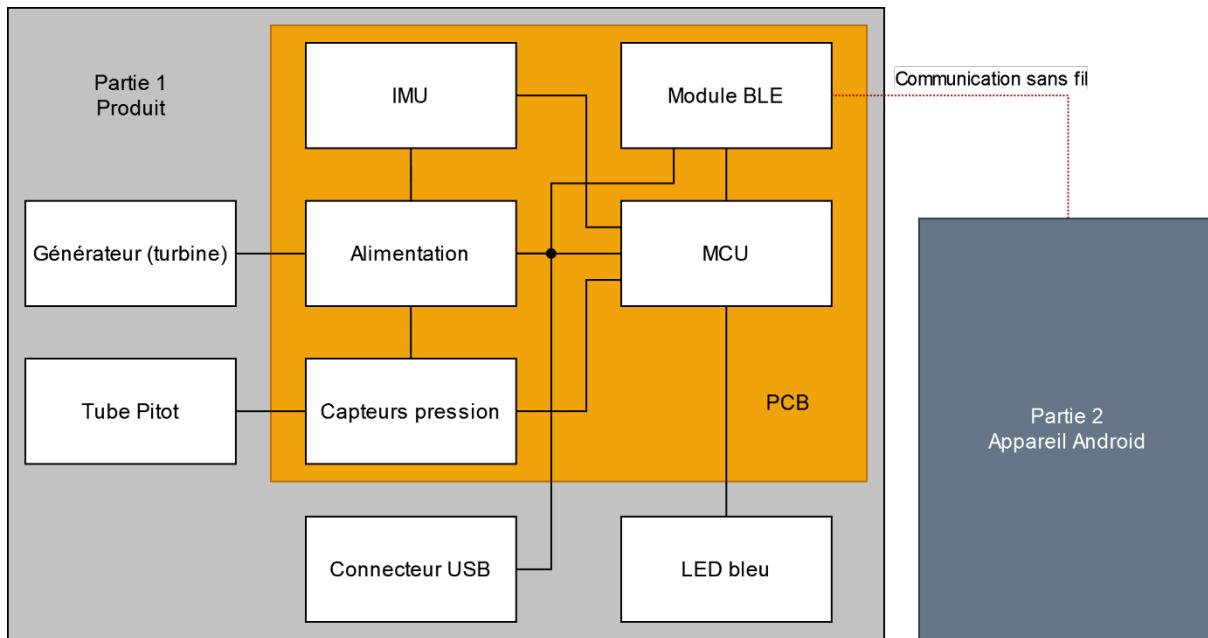
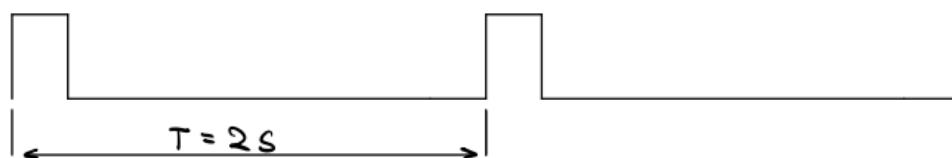


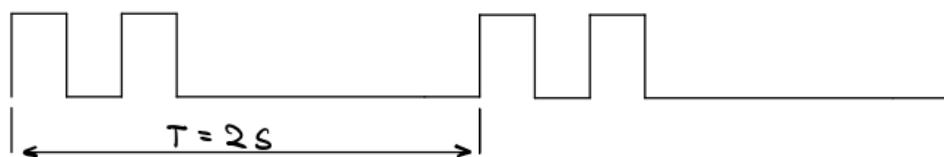
Figure 2 - Schéma bloc

#### 2.1.1 Descriptions simples des blocs

- Générateur (turbine), l'idée est d'intégrer un générateur-turbine, montés à l'arrière du produit (voir Figure 2) afin de générer l'électricité nécessaire au système. Dans le cas où la réalisation mécanique de cela serait trop difficile, une batterie ou pile serait l'alternative.
- LED bleue, permettra d'indiquer l'état du système à un moment donné, elle devra être visible depuis l'intérieur du cockpit. Le comportement sera le suivant :
  - LED éteinte, le système non alimenté.
  - Simple clignotement à une fréquence de 0.5Hz, le système n'est pas appairé à un périphérique Bluetooth.



- Double clignotement à une fréquence de 0.5Hz, le système est appairé à un périphérique Bluetooth et fonctionnel.



- Connecteur USB, le connecteur permettra d'alimenter le système afin de le tester lorsqu'il est au sol.
- Tube Pitot, servira d'entrée d'air pour les capteurs de pression.
- Capteurs pression, ces capteurs liront les valeurs de la pression statique et totale pour ainsi calculer la vitesse air.
- Alimentation, l'alimentation transformera la tension provenant de la source d'énergie en une tension régulée et stable.
- IMU, la centrale inertie servira à mesurer les vecteurs des vitesses angulaires et les accélérations.
- MCU, le microcontrôleur va être le cerveau du produit, il sera responsable de commander tous les autres composants.
- Module BLE, le module Bluetooth Low Energy permettra d'établir une communication basse consommation entre le produit et l'appareil Android. C'est par le biais de cette liaison que seront transmises toutes les données.
- Appareil Android, l'appareil recevra, traitera et affichera les données sur son écran. Il permettra également d'envoyer des commandes tels que la fréquence d'envoi ou d'autres paramètres.

Les blocs seront décrits en détail dans la pré-étude.

## 2.2 Croquis du produit

L'intégration du système se fera dans un cylindre à l'arrière du tube Pitot comme représenté ci-dessous (vue en coupe). Dans le cas où la vitesse de rotation du générateur est trop élevée, l'ajout d'un carénage permettrait de diminuer la vitesse du flux d'air en augmentant la section de sortie par rapport à l'entrée. La fixation se réalisera à l'aide d'une boule style RAM-Mounts.

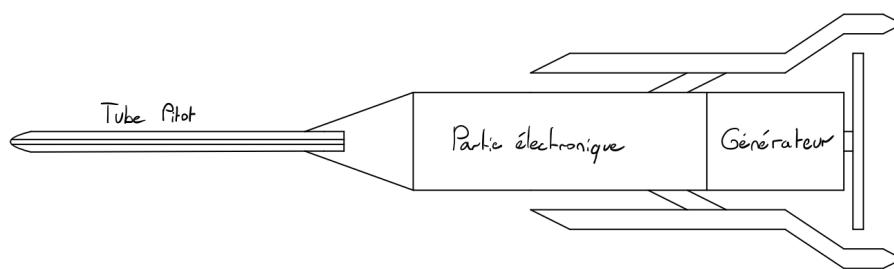


Figure 3 - Croquis du produit (vue en coupe)

## 3 Tâches à réaliser

- Pré-étude
- Dimensionnement, design et réalisation du schéma
- Commande des composants et du matériel
- Réalisation du PCB
- Design de l'assemblage mécanique avec fixations
- Montage du PCB
- Réalisation du logiciel du MCU et de l'application Android
- Impression 3D ou usinage de la partie mécanique
- Mise en service et tests

## 4 Deadlines principales

- 16 novembre 2022 : Début du projet
- 07 décembre 2022 : Rendu de la pré-étude
- 14 décembre 2022 : Présentation de la pré-étude
- 25 janvier 2023 : Rendu du design
- 01 février 2023 : Présentation du design
- 22 mars 2023 : Rendu des fichiers de fabrication PCB
- 14 juin 2023 : Rendu de toute la documentation finale
- 21 juin 2023 : Présentation finale

### 4.1 Planning

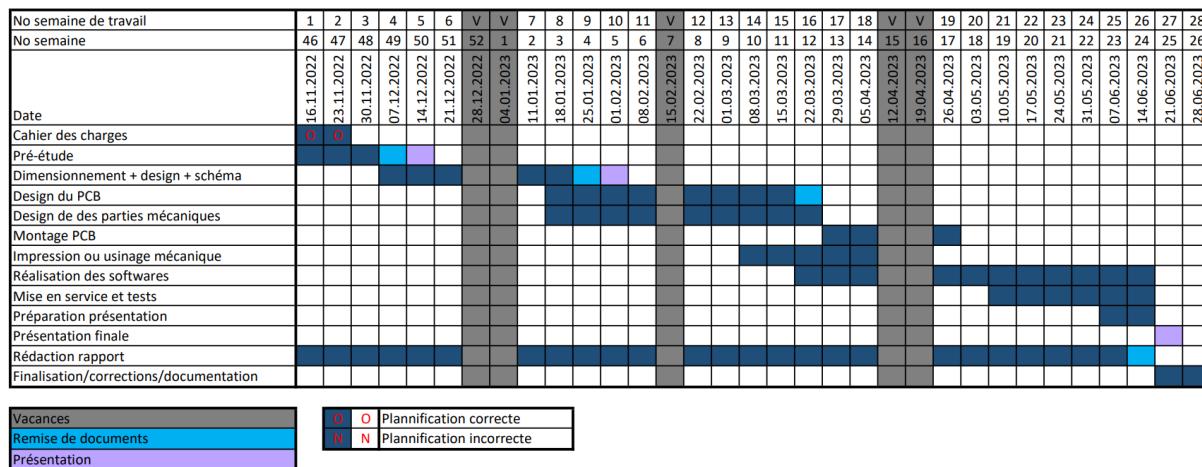
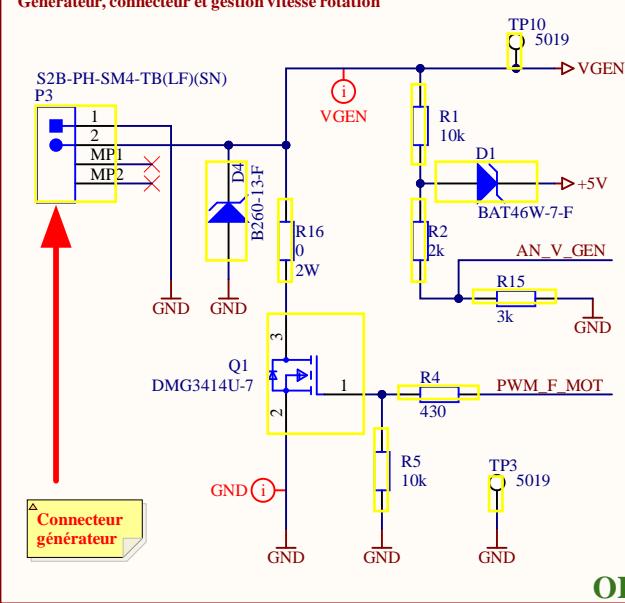


Figure 4 - Planning

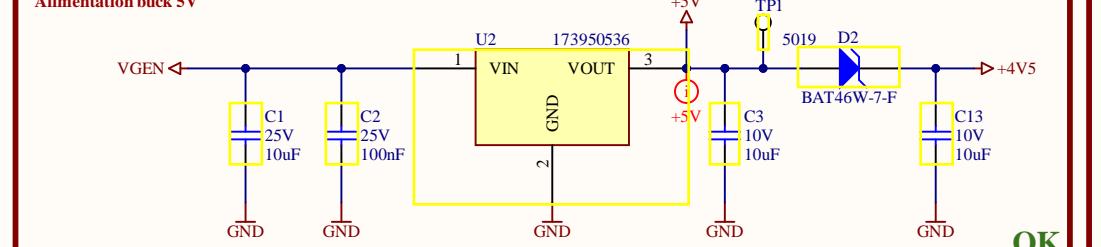
## 5 Livrables

- Les fichiers sources de CAO électronique et mécanique
- Tout le nécessaire pour fabriquer un (1) exemplaire hardware :
  - Fichiers de fabrication (GERBER)
  - Liste de pièces avec références (BOM)
  - Implantation des composants
  - Dessins mécaniques
- Les fichiers sources de programmation du microcontrôleur (.c./h)
- Tout le nécessaire pour programmer les microcontrôleurs (.hex)
- Les fichiers sources de programmation Android.
- Tout le nécessaire à l'installation de programmes sur Android.
- Un mode d'emploi du produit
- Une estimation des coûts du projet
- Un rapport contenant toutes les informations du design du produit.
- Un prototype monté et fonctionnel
- Une application Android permettant d'afficher les données :
  - Affichage numérique
  - Affichage graphique (optionnel)

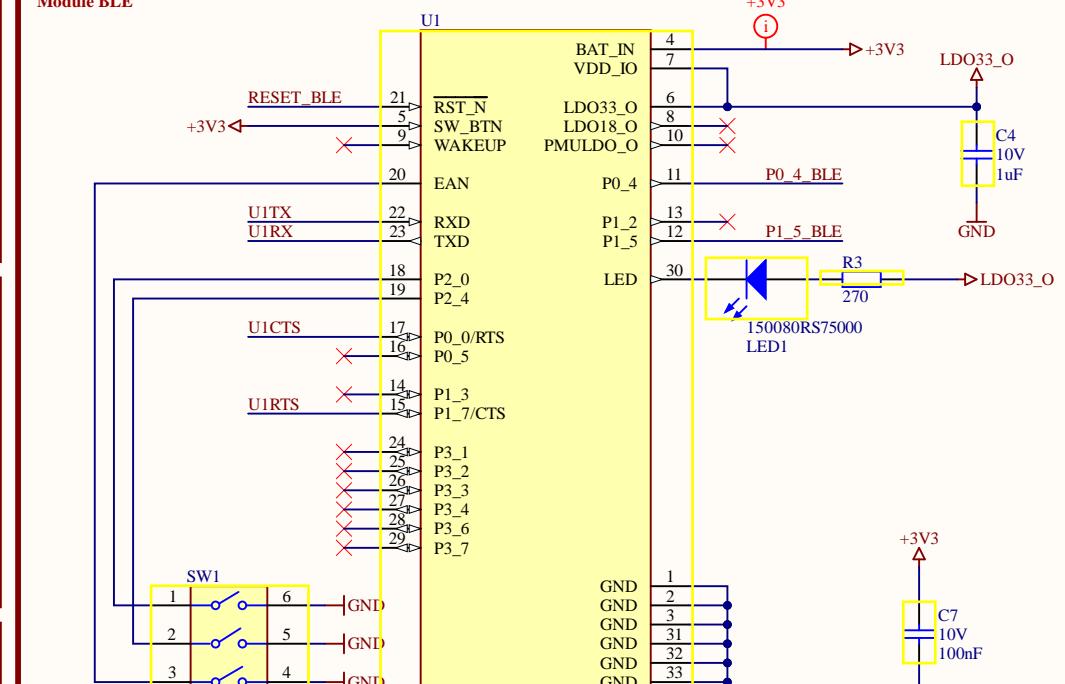
## Générateur, connecteur et gestion vitesse rotation



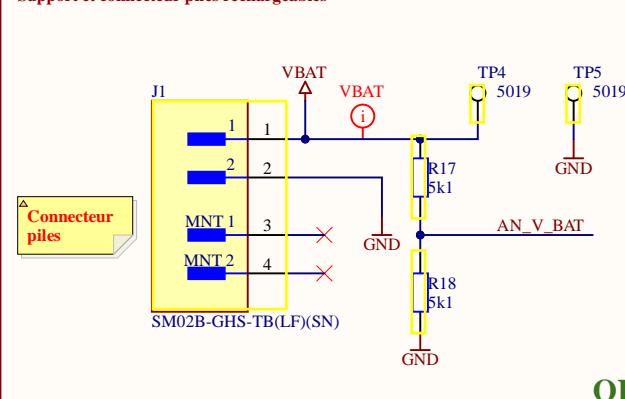
## Alimentation buck 5V



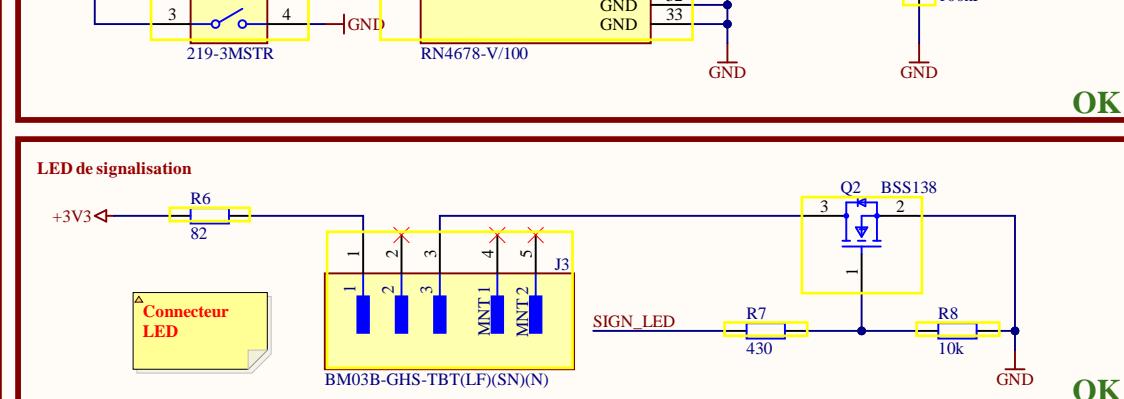
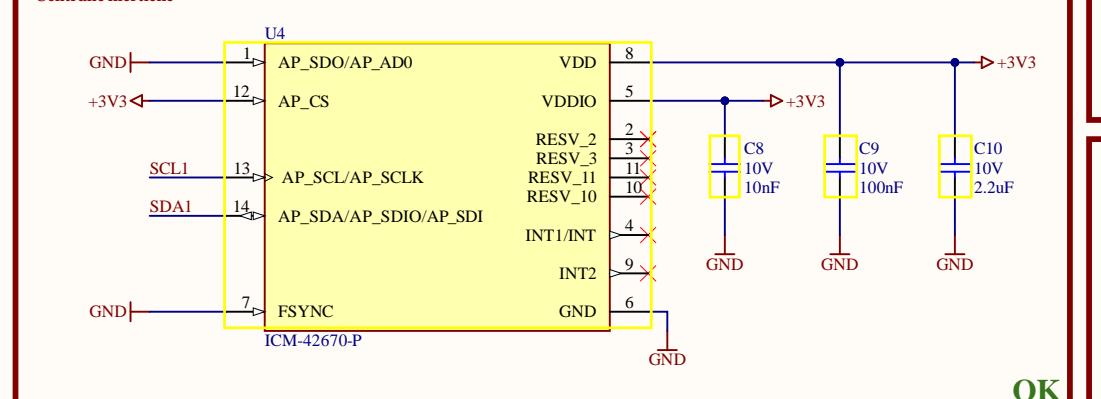
## Module BLE



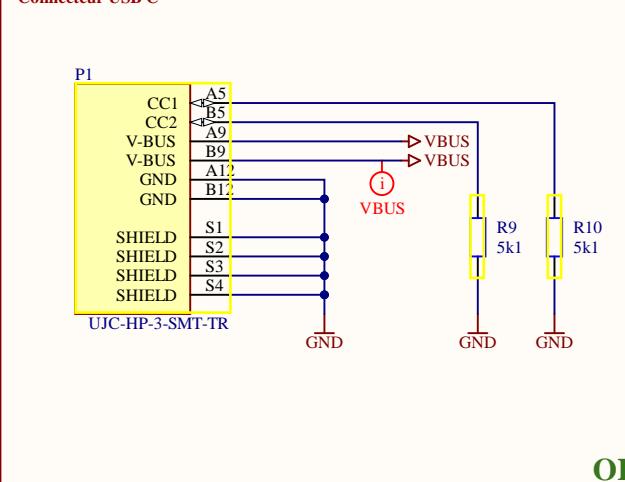
## Support et connecteur piles rechargeables



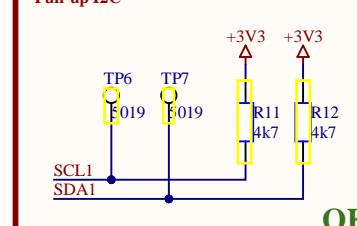
## Centrale inertuelle



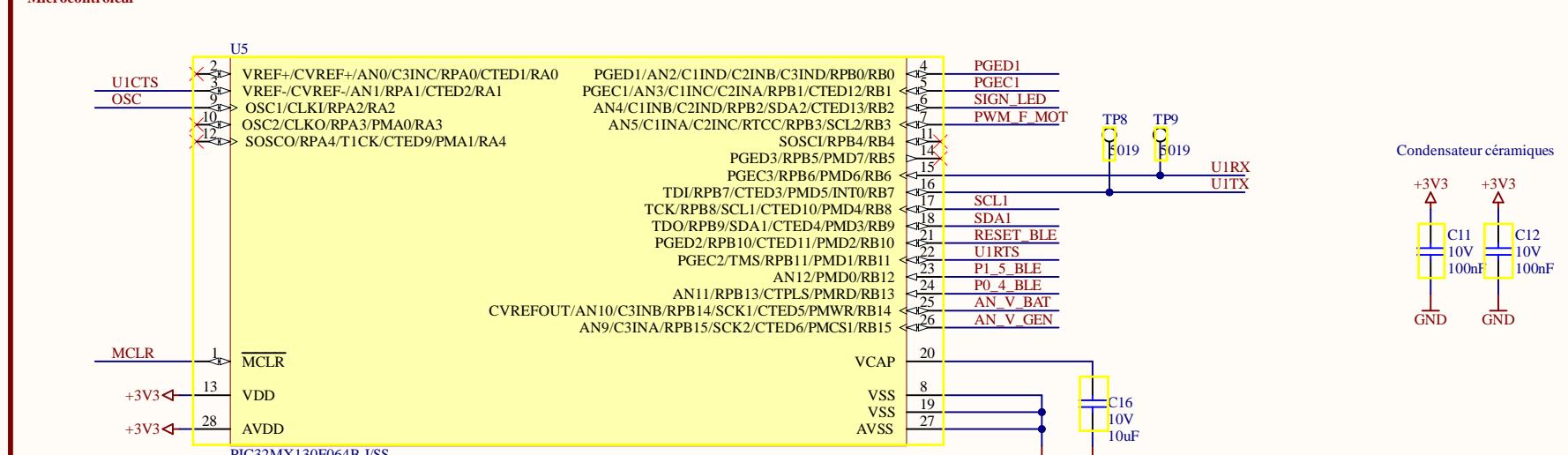
## Connecteur USB C



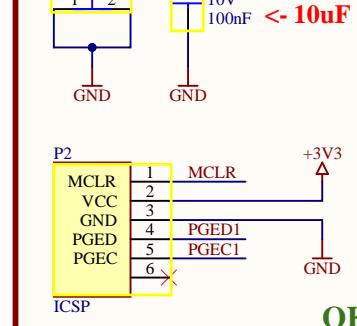
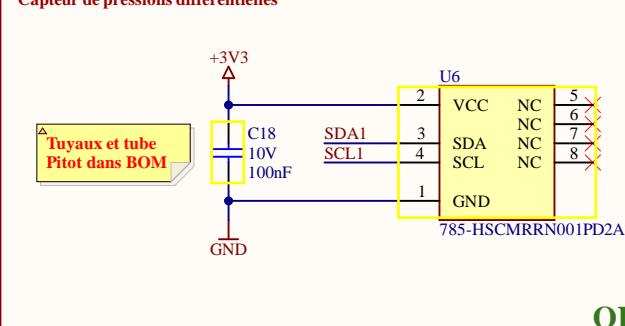
## Pull-up I2C



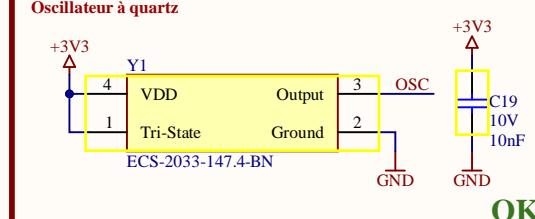
## Microcontroleur



## Capteur de pressions différentielles



## Oscillateur à quartz



Fichier : 2230\_TubePitotDeporte\_Schematic\_v1.0.0.SchDoc

Date : 16.06.2023

Heure : 03:45:02

Modif. :

a  
b  
c  
d

Auteur : \*

Nb feuillets 1 Feuille n° 1

Chemin : P:\School\PROJ\2230\_TubePitotDeporte\hard\PCB\_Project\2230\_TubePitotDeporte\_Schematic\_v1.0.0.SchDoc

Projet : 2230\_TubePitotDeporte\_v1.0.0.PjPcb

```

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 // Author M.Ricchieri
51
52
53 // ****
54 // ****
55 // Section: Included Files
56 // ****
57 // ****
58
59 #include "app.h"
60 #include "Mc32_I2cUtilCCS.h"
61
62 #include "HSCMRRN001PD2A3_driver.h"
63
64 #include "RN4678_driver.h"
65 #include "voltageADC_driver.h"
66
67
68
69 // ****
70 // ****
71 // Section: Global Data Definitions
72 // ****
73 // ****
74

```

```

75 //-----// Global data
76 APP_DATA      appData;
77 SENS_DATA     sensData;
78
79 struct inv_imu_device  myImuDevice;
80 struct inv_imu_serif   myImuSertif;
81
82
83
84 // ****
85 // ****
86 // Section: Application Callback Functions
87 // ****
88 // ****
89
90 //-----// TIMER0 callback function <--- Disabled
91 void TIMER0_Callback_Function(){ //156Hz
92 //
93 //    // xxxx
94 //}
95
96
97 //-----// TIMER1 callback function
98 void TIMER1_Callback_Function(){ // 20Hz
99
100 //    if(appData.isBluetoothDiscoverable){
101 //
102 //        if(inv_imu_get_time_us() >= 120000000){ // 120 000 000 = 120 seconds
103 //
104 //            turnOffDiscoverBT();
105 //            appData.isBluetoothDiscoverable = false;
106 //
107 //        }
108 //    Update the main state machine
109 APP_UpdateAppState(APP_STATE_SERVICE);
110 }
111
112
113 //-----// USART1_Callback_Function
114 void USART1_Callback_Function(void){
115
116     char a_received[50];
117     char* result;
118
119     // Gets new data from FIFO
120     getUsartData(&a_received[0]);
121
122     // If "<RFCOMM_OPEN>" is present in the array received
123     result = strstr(a_received, "<RFCOMM_OPEN>");
124     if(result != NULL){
125
126         appData.isBluetoothDiscoverable = false;
127         appData.isBluetoothConnected = true;
128     }
129
130     // If "<RFCOMM_CLOSE>" is present in the array received
131     result = strstr(a_received, "<RFCOMM_CLOSE>");
132     if(result != NULL){
133
134         // turnOnDiscoverBT();
135         appData.isBluetoothConnected = false;
136     }
137
138     // If "<xxxxxx>" is present in the array received
139     result = strstr(a_received, "<xxxxxx>");
140     if(result != NULL){
141
142         // Does something
143     }
144 }
145
146
147 //-----// IMU callback function
148 void imu_callback(inv_imu_sensor_event_t *event){

```

```

149
150     // Transforms 16bits values into degrees and saves them in the sensor data
151     // structure
152     // 250 dps
153     sensData.gyroX = (event->gyro[0])/250 + 1; // + offset
154     sensData.gyroY = (event->gyro[1])/250 + 1; // + offset
155     sensData.gyroZ = (event->gyro[2])/250;
156
157     // Reads and transforms 16bits values into
158     // Transforms 16bits values into g acceleration and saves them in the sensor
159     // data structure (8192 bits per g)
160     sensData.accelX = (float)(event->accel[0])/8192.0;
161     sensData.accelY = (float)(event->accel[1])/8192.0;
162     sensData.accelZ = (float)(event->accel[2])/8192.0 + 0.075; // + offset
163
164     // Calculate the angle with the gyro values
165     // 0.05 correspond to the period between each reading 1/20Hz = 0.05s
166     sensData.GyrAngleX += sensData.gyroX * 0.05;
167     sensData.GyrAngleY += sensData.gyroY * 0.05;
168     sensData.GyrAngleZ += sensData.gyroZ * 0.05;
169 }
170
171
172 // ****
173 // ****
174 // Section: Application Local Functions
175 // ****
176 // ****
177
178 //-----// APP_UpdateAppState
179 void APP_UpdateAppState(APP_STATES newState){
180
181     appData.appState = newState;
182 }
183
184
185 //-----// APP_UpdateServiceState
186 void APP_UpdateServiceState(SERVICE_STATES newState){
187
188     appData.serviceState = newState;
189 }
190
191
192 //-----// clearArray
193 void clearArray(size_t arraySize, char *pArrayToClear){
194
195     int i;
196
197     for (i = 0; i < arraySize; i++){
198
199         pArrayToClear[i] = NULL;
200     }
201 }
202
203
204 //-----// frameFormatting
205 inline void frameFormatting(char* a_dataToSend, const SENS_DATA* sensData){
206
207     // Saves all data into a simple frame
208     // Speed in [km/h]
209     // Gyros in [dps]
210     // Angles in [degrees]
211     // Accelerations in [g]
212     // VB and VG in [V]
213     sprintf(a_dataToSend, "S=%03d GX=%+.02f GY=%+.02f GZ=%+.02f GAX=%+.02f "
214             "GAY=%+.02f GAZ=%+.02f AX=%+.02f AY=%+.02f AZ=%+.02f VB=%+.02f "
215             "VG=%+.02f\n\r",
216             sensData->velocity,
217             sensData->gyroX, sensData->gyroY, sensData->gyroZ,
218             sensData->GyrAngleX, sensData->GyrAngleY, sensData->GyrAngleZ,
219             sensData->accelX, sensData->accelY, sensData->accelZ,
220             sensData->batVoltage, sensData->genVoltage);
221 }
222

```

```

223 // ****
224 // ****
225 // Section: Application Initialization and State Machine Functions
226 // ****
227 // ****
228
229 //-----// APP_Initialize
230 void APP_Initialize(void){
231
232     // Initializes the appData structure
233
234     appData.appState          = APP_STATE_INIT;
235     appData.isBluetoothModuleInit = false;
236     appData.isBluetoothConnected = false;
237     appData.isBluetoothDiscoverable = false;
238     sensData.GyrAngleX = 0;
239     sensData.GyrAngleY = 0;
240     sensData.GyrAngleZ = 0;
241 }
242
243
244 //-----// initImuInterface
245 // Initialize serial interface between MCU and IMU
246 int initImuInterface(struct inv_imu_serif *icm_serif){
247
248     // No need
249     icm_serif->context      = 0;
250     // Points to the reading function dedicated
251     icm_serif->read_reg     = ICM42670P_I2C_bus_read;
252     // Points to the writing function dedicated
253     icm_serif->write_reg    = ICM42670P_I2C_bus_write;
254     icm_serif->max_read     = 255; /* maximum number of bytes allowed per serial read */
255     icm_serif->max_write    = 255; /* maximum number of bytes allowed per serial write */
256     // Set the communication interface
257     icm_serif->serif_type   = SERIF_TYPE;
258
259     return 0;
260 }
261
262
263 //-----// APP_Tasks
264 void APP_Tasks(void){
265
266     RAW_ADC rawAdc;
267
268     // Check the application's current state
269     switch(appData.appState){
270
271         // Application's initial state
272         case APP_STATE_INIT:
273         {
274             int rc = 0;
275
276             // Initialization of the I2C communication
277             i2c_init(SLOW);
278
279             do{
280                 // Initialization of the ICM42670 interface
281                 rc |= initImuInterface(&myImuSertif);
282                 // Resets and prepares the chip for the configuration
283                 rc |= setupImuDevice(&myImuSertif);
284                 // Configures ICM42670 parameters
285                 rc |= configureImuDevice();
286
287             }while(rc != INV_ERROR_SUCCESS);
288
289             // Initialization of the USART FIFOs
290             initFifo(&uartFifoRx, FIFO_RX_SIZE, a_fifoRx, 0);
291             initFifo(&uartFifoTx, FIFO_TX_SIZE, a_fifoTx, 0);
292
293             do{
294                 // Initialization of the Bluetooth module
295                 appData.isBluetoothModuleInit = init_RN4678();
296

```

```

296
297         }while(appData.isBluetoothModuleInit == false);
298
299         // Initialization of the ADC module
300         initAdc();
301
302         // Starts TIMERS
303 //         DRV_TMR0_Start(); <--- Disabled
304         DRV_TMR1_Start();
305         DRV_TMR2_Start();
306
307         // States machines update
308         APP_UpdateAppState(APP_STATE_WAIT);
309         APP_UpdateServiceState(SERVICE_STATE_READ_SENSORS);
310         break;
311     }
312
313     case APP_STATE_SERVICE:
314     {
315         int8_t a_frameToSend[130];
316
317         switch(appData.serviceState) {
318
319             case SERVICE_STATE_READ_SENSORS:
320
321                 // Reads voltages values
322                 readRawAdc(&rawAdc);
323                 convertRawToVoltage(&rawAdc, &sensData);
324                 // Reads velocity value
325                 convertRawToVelocity(readRawDiffPress(), &sensData);
326                 // Gets new IMU data
327                 get_imu_data();
328
329
330                 APP_UpdateServiceState(SERVICE_STATE_PROCESS);
331                 break;
332
333
334             case SERVICE_STATE_PROCESS:
335
336                 // Clears the array before saving new values
337                 clearArray(sizeof(a_frameToSend), (char*)&a_frameToSend[0]);
338                 // Converts float values in a char array (frame)
339                 frameFormatting((char*)&a_frameToSend[0], &sensData);
340
341                 APP_UpdateServiceState(SERVICE_STATE_SEND_DATA_BT);
342                 break;
343
344             case SERVICE_STATE_SEND_DATA_BT:
345
346                 // Bluetooth is connected to a device
347                 if(appData.isBluetoothConnected == true){
348
349                     // Sends frame through USART
350                     sendData_RN4678(&a_frameToSend[0]);
351                     // Toggle signalisation LED
352                     SIGN_LEDToggle();
353                 }
354                 else SIGN_LEDOn();
355
356                 APP_UpdateAppState(APP_STATE_WAIT);
357                 APP_UpdateServiceState(SERVICE_STATE_READ_SENSORS);
358                 break;
359             }
360             break;
361         }
362
363     case APP_STATE_WAIT:
364     {
365         // Does nothing here
366         break;
367     }
368     default:
369     {
370         // Does nothing here

```

```
371         break;
372     }
373 }
374 }
375
376
377 /*****
378 End of File
379 */
```

```

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
47 // Author M.Ricchieri
48
49
50 #ifndef _APP_H
51 #define _APP_H
52
53 // ****
54 // ****
55 // Section: Included Files
56 // ****
57 // ****
58
59 #include <stdint.h>
60 #include <stdbool.h>
61 #include <stddef.h>
62 #include <stdlib.h>
63 #include <stdio.h>
64 #include "system_config.h"
65 #include "system_definitions.h"
66
67 #include "imu/inv_imu_driver.h"
68 #include "imu/inv_imu_transport.h"
69 #include "Invn/EmbUtils/RingBuffer.h"
70 #include "I2C_ICM42670P_Functions.h"
71 #include "inv_imu_personnal_functions.h"
72
73 #include "uart_FIFO.h"
74

```

```

75 // DOM-IGNORE-BEGIN
76 #ifdef __cplusplus // Provide C++ Compatibility
77
78 extern "C" {
79
80 #endif
81 // DOM-IGNORE-END
82
83 // ****
84 // ****
85 // Section: Type Definitions
86 // ****
87 // ****
88
89 /*
90  * Select communication link between SmartMotion and IMU
91  */
92 #define SERIF_TYPE UI_I2C
93
94 /*
95  * Set power mode flag
96  * Set this flag to run example in low-noise mode.
97
98  * Reset this flag to run example in low-power mode.
99  * Note: low-noise mode is not available with sensor data frequencies less than 12.5Hz.
100 */
100 #define USE_LOW_NOISE_MODE 1
101
102 /*
103  * Select Fifo resolution Mode (default is low resolution mode)
104  * Low resolution mode: 16 bits data format
105  * High resolution mode: 20 bits data format
106  * Warning: Enabling High Res mode will force FSR to 16g and 2000dps
107 */
108 #define USE_HIGH_RES_MODE 0
109
110 /*
111  * Select to use FIFO or to read data from registers
112 */
113 #define USE_FIFO 0
114
115 /*
116  * Print raw data or scaled data
117  * 0 : print raw accel, gyro and temp data
118  * 1 : print scaled accel, gyro and temp data in g, dps and degree Celsius
119 */
120 #define SCALED_DATA_G_DPS 1
121
122
123 #define FAST      1
124 #define SLOW      0
125
126
127
128 typedef enum
129 {
130     APP_STATE_INIT=0,
131     APP_STATE_SERVICE,
132     APP_STATE_WAIT,
133
134 } APP_STATES;
135
136
137 typedef enum
138 {
139     SERVICE_STATE_READ_SENSORS=0,
140     SERVICE_STATE_PROCESS,
141     SERVICE_STATE_SEND_DATA_BT,
142
143 } SERVICE_STATES;
144
145
146 // Application data structure
147 typedef struct{
148

```

```

149     APP_STATES appState;
150     SERVICE_STATES serviceState;
151
152     bool isBluethoothModuleInit;
153     bool isBluetoothConnected;
154     bool isBluetoothDiscoverable;
155     bool isBluetoothInOperation;
156     bool isBluetoothInCommandMode;
157
158 } APP_DATA;
159
160
161 // Sensors data structure
162 typedef struct{
163
164     uint16_t velocity;
165     float gyroX;
166     float gyroY;
167     float gyroZ;
168     float accelX;
169     float accelY;
170     float accelZ;
171     float GyrAngleX;
172     float GyrAngleY;
173     float GyrAngleZ;
174     float batVoltage;
175     float genVoltage;
176
177 } SENS_DATA;
178
179
180 // Analogic data structure
181 typedef struct{
182
183     uint16_t AN9_V_GEN;
184     uint16_t AN10_V_BAT;
185
186 }RAW_ADC;
187
188
189
190 // ****
191 // ****
192 // Section: Application Callback Routines
193 // ****
194 // ****
195 /* These routines are called by drivers when certain events occur.
196 */
197
198 // ****
199 // ****
200 // Section: Application Initialization and State Machine Functions
201 // ****
202 // ****
203
204
205
206 // Extern variables and structures
207 extern APP_DATA      appData;
208 extern SENS_DATA     sensData;
209 extern bool          isBluetoothConnected;
210 extern bool          isBluethoothModuleInit;
211
212 extern struct inv_imu_device  myImuDevice;
213 extern struct inv_imu_serif   myImuSertif;
214
215
216 // Basic functions prototypes
217 void APP_Initialize (void);
218 void APP_Tasks( void );
219 void APP_UpdateAppState(APP_STATES NewState);
220 void clearArray(size_t arraySize, char *pArrayToClear);
221 inline void frameFormatting(char* a_dataToSend, const SENS_DATA* sensData);
222

```

```
223 // Callback functions prototypes
224 void TIMER0_Callback_Function(void);
225 void TIMER1_Callback_Function(void);
226 void TIMER5_Callback_Function(void);
227 void USART1_Callback_Function(void);
228 void imu_callback(inv_imu_sensor_event_t *event);
229
230
231 // IMU useful functions prototypes
232 int         initImuInterface(struct inv_imu_serif *icm_serif);
233 uint64_t    inv_imu_get_time_us(void);
234
235
236
237 #endif /* _APP_H */
238
239 //DOM-IGNORE-BEGIN
240 #ifdef __cplusplus
241 }
242 #endif
243 //DOM-IGNORE-END
244
245 ****
246 End of File
247 */
```

```

1 /*
2  * File: HSCMRRN001PD2A3_driver.c
3  * Author: M.Ricchieri
4  *
5  * Created on 12. avril 2023
6  */
7
8
9 //-----// Includes
10 #include "HSCMRRN001PD2A3_driver.h"
11 #include "math.h"
12 #include "Mc32_I2cUtilCCS.h"
13 #include "peripheral\i2c\plib_i2c.h"
14
15
16 //-----// Constants
17 #define HSCMRRN001PD2A3_ADDR 0x51
18
19 #define RHO_AIR 1.2
20
21
22 //-----// readRawDiffPress
23 // Read the raw compensated differential pressure from the HSCMRRN001PD2A3 sensor
24 int16_t readRawDiffPress(){
25
26     int16_t rawDiffPress;
27     uint8_t MSB;
28     uint8_t LSB;
29
30     // I2C communication with the sensor
31     i2c_start();
32     i2c_write(HSCMRRN001PD2A3_ADDR);
33     MSB = i2c_read(1);
34     LSB = i2c_read(1);
35     // Reads 2 unused bytes to avoid bug // needs to be clarified !
36     i2c_read(1);
37     i2c_read(0); // No ACK
38     i2c_stop();
39
40     // Data formatting
41     rawDiffPress = MSB;
42     rawDiffPress = rawDiffPress << 8;
43     rawDiffPress = rawDiffPress | LSB;
44     rawDiffPress = rawDiffPress - 8192;
45     // Safety to avoid negative speeds
46     if(rawDiffPress < 0) rawDiffPress = 0;
47
48     return rawDiffPress;
49 }
50
51
52 //-----// convertRawToVelocity
53 // Convert raw compensated differential pressure to velocity (km/h)
54 void convertRawToVelocity(int16_t rawDiffPress, SENS_DATA *pSensData){
55
56     // Multiplied by 3.6 to obtain velocity in km/h instead of m/s
57     pSensData->velocity = 3.6 * (sqrtf((2*(float)rawDiffPress)/(RHO_AIR)));
58 }
```

```
1 /*
2  * File: HSCMRRN001PD2A3_driver.h
3  * Author: M.Ricchieri
4  *
5  * Created on 12. avril 2023
6  */
7
8 #ifndef HSCMRRN001PD2A3_DRIVER
9 #define HSCMRRN001PD2A3_DRIVER
10
11
12 //-----// Includes
13 #include "app.h"
14
15 #ifdef __cplusplus
16 extern "C" {
17 #endif
18
19 //-----// Functions
20 int16_t readRawDiffPress();
21 void convertRawToVelocity(int16_t rawDiffPress, SENS_DATA *pSensData);
22
23 #ifdef __cplusplus
24 }
25 #endif
26
27 #endif/* HSCMRRN001PD2A3_DRIVER */
```

```

1
2 /*
3  * File:      I2C_ICM42670P_Function.c
4  * Author:    M.Ricchieri
5  *
6  * Created on 10. mai 2023
7  *
8  * This code uses the Mc32_I2cUtilCCS.h
9  */
10
11
12 //-----// Includes
13 #include "stdint.h"
14 #include "Mc32_I2cUtilCCS.h"
15 #include "imu/inv_imu_driver.h"
16 #include "imu/inv_imu_transport.h"
17 #include "I2C_ICM42670P_Functions.h"
18
19
20 //-----// Constants
21 #define ICM42670P_INIT_VALUE 0
22 #define I2C_BUFFER_LEN 10
23 #define ICM42670P_I2C_BUS_WRITE_ARRAY_INDEX 1
24
25
26 // \Brief : Those functions are used by the ICM42670 IMU to communicate with
27 //           the chip in I2C. The first one is used to read and the second to
28 //           write in the chip. Those functions are pointed by the ICM42670
29 //           driver.
30
31 //-----// ICM42670P_I2C_bus_write
32 int ICM42670P_I2C_bus_write(struct inv_imu_serif *serif, uint8_t reg,
33     const uint8_t *buf, uint32_t len){
34
35     int cursor = 0;
36
37     i2c_start();
38     i2c_write(ICM42670P_ADDR_W);
39     i2c_write(reg);
40
41     for (cursor = 0; cursor < len; cursor++){
42
43         i2c_write(*(buf+cursor));
44     }
45     i2c_stop();
46
47     return 0;
48 }
49
50
51 //-----// ICM42670P_I2C_bus_read
52 int ICM42670P_I2C_bus_read(struct inv_imu_serif *serif, uint8_t reg,
53     uint8_t *buf, uint32_t len){
54
55     uint8_t array[len];
56     int cursor = 0;
57
58     i2c_start();
59     i2c_write(ICM42670P_ADDR_W);
60     i2c_write(reg);
61     i2c_start();
62     i2c_write(ICM42670P_ADDR_R);
63
64     for (cursor = 0; cursor < len; cursor++){
65
66         if((cursor+1) < len)
67             // With ACK
68             array[cursor] = i2c_read(1);
69         else
70             // Without ACK
71             array[cursor] = i2c_read(0);
72
73         buf[cursor] = array[cursor];
74         /*(buf+cursor) = array[cursor];
```

```
75     }
76     i2c_stop();
77
78     return 0;
79 }
```

```
1
2 /*
3  * File:      I2C_ICM42670P_Functions.h
4  * Author:    M.Ricchieri
5  *
6  * Created on 10. mai 2023
7  *
8  * This code uses the Mc32_I2cUtilCCS.h
9  */
10
11
12 //-----// Includes
13 #include "app.h"
14
15
16 #ifndef I2C_ICM42670P_FUNCTIONS_H
17 #define I2C_ICM42670P_FUNCTIONS_H
18
19 //-----// Constants
20 #define ICM42670P_ADDR_W 0b11010000 // W
21 #define ICM42670P_ADDR_R 0b11010001 // R
22
23
24 #ifdef __cplusplus
25 extern "C" {
26 #endif
27
28 //-----// Fonctions
29 int ICM42670P_I2C_bus_write(struct inv_imu_serif *serif, uint8_t reg,
30     const uint8_t *buf, uint32_t len);
31 int ICM42670P_I2C_bus_read(struct inv_imu_serif *serif, uint8_t reg,
32     uint8_t *buf, uint32_t len);
33
34 #ifdef __cplusplus
35 }
36 #endif
37
38 #endif/* I2C_ICM42670P_FUNCTIONS_H */
```

```

1
2 /*
3  * File:    adc_driver.c
4  * Author:  M.Ricchieri
5  *
6  * Created on 31. mai 2023, 08:55
7  *
8  * Inspired by the "Mc32DriverAdc.c" file
9  */
10
11 //-----// Includes
12 #include "app.h"
13 #include "I2C_ICM42670P_Functions.h"
14 #include "imu/inv_imu_driver.h"
15 #include "imu/inv_imu_transport.h"
16 #include "Invn/EmbUtils/RingBuffer.h"
17
18
19 //-----// Constants
20 #if !USE_FIFO
21 /*
22  * Buffer to keep track of the timestamp when IMU data ready interrupt fires.
23  * The buffer can contain up to 64 items in order to store one timestamp
24  * for each packet in FIFO.
25  */
26 RINGBUFFER(timestamp_buffer, 64, uint64_t);
27 #endif
28
29
30 //-----// get_imu_data
31 /** @brief this function provides a way to retrieve IMU data either from a FIFO
32  *         buffer if USE_FIFO is defined, or directly from the registers if
33  *         USE_FIFO is not defined.
34  *
35  * @param[in]      -
36  * @return         data from "inv_imu_get_data_from_registers" function or
37  *                 data from "inv_imu_get_data_from_fifo"
38 */
39 int get_imu_data(void)
40 {
41 #if USE_FIFO
42 return inv_imu_get_data_from_fifo(&myImuDevice);
43 #else
44 return inv_imu_get_data_from_registers(&myImuDevice);
45 #endif
46 }
47
48
49 //-----// configureImuDevice
50 /** @brief This function configures the device in order to output gyro and
51  *         accelerometer.
52  *
53  * @return        rc          InvError structure parameter, 0 on success
54 */
55 int configureImuDevice(void)
56 {
57 int rc = 0;
58
59 if (!USE_FIFO)
60 rc |= inv_imu_configure_fifo(&myImuDevice, INV_IMU_FIFO_DISABLED);
61
62 if (USE_HIGH_RES_MODE) {
63 rc |= inv_imu_enable_high_resolution_fifo(&myImuDevice);
64 } else {
65 rc |= inv_imu_set_accel_fsr(&myImuDevice, ACCEL_CONFIG0_FS_SEL_4g);
66 rc |= inv_imu_set_gyro_fsr(&myImuDevice, GYRO_CONFIG0_FS_SEL_250dps);
67 }
68
69 if (USE_LOW_NOISE_MODE) {
70 rc |= inv_imu_set_accel_frequency(&myImuDevice, ACCEL_CONFIG0_ODR_50_HZ);
71 rc |= inv_imu_set_gyro_frequency(&myImuDevice, GYRO_CONFIG0_ODR_200_HZ);
72 rc |= inv_imu_enable_accel_low_noise_mode(&myImuDevice);
73 } else {
74 rc |= inv_imu_set_accel_frequency(&myImuDevice, ACCEL_CONFIG0_ODR_50_HZ);

```

```

75 rc |= inv_imu_set_gyro_frequency(&myImuDevice, GYRO_CONFIG0_ODR_200_HZ);
76 rc |= inv_imu_enable_accel_low_power_mode(&myImuDevice);
77 }
78 //    rc |= inv_imu_set_accel_lp_avg(&myImuDevice, ACCEL_CONFIG1_ACCEL_FILT_AVG_8);
79
80 rc |= inv_imu_enable_gyro_low_noise_mode(&myImuDevice);
81
82 if (!USE_FIFO)
83 inv_imu_sleep_us(GYR_STARTUP_TIME_US);
84
85 return rc;
86 }
87
88
89
90 //-----// setupImuDevice
91 /** @brief This function is in charge of reseting and initializing IMU device.
92 *          It should be successfully executed before any access to IMU device.
93 */
94 * @param[in]      icm_serif      pointer to the serial interface structure
95 * @return         rc           InvError structure parameter, 0 on success
96 */
97 int setupImuDevice(struct inv_imu_serif *icm_serif) {
98
99 int rc = 0;
100 uint8_t who_am_i;
101
102 // Initialization of the device
103 rc = inv_imu_init(&myImuDevice, icm_serif, imu_callback);
104 if (rc != INV_ERROR_SUCCESS) {
105
106 return rc;
107 }
108
109 // Check WHOAMI
110 rc = inv_imu_get_who_am_i(&myImuDevice, &who_am_i);
111 if (rc != INV_ERROR_SUCCESS) {
112
113 return rc;
114 }
115
116 if (who_am_i != ICM_WHOAMI) {
117
118 return INV_ERROR;
119 }
120
121 #if !USE_FIFO
122 RINGBUFFER_CLEAR(&timestamp_buffer);
123 #endif
124
125 return rc;
126 }
127
128
129 //-----// inv_imu_sleep_us
130 /** @brief This function is in charge of delaying the program for a certain
131 *          time.
132 */
133 * @param[in]      us           time in microsecond
134 * @return         -           -
135 */
136 void inv_imu_sleep_us(uint32_t us){
137
138     uint16_t finalValue;
139
140     // Prepares the Timer3 and the counting variable
141     DRV_TMR3_CounterClear();
142     // appData.usCounter32 = 0;
143     // Starts Timer3
144     DRV_TMR3_Start();
145
146     finalValue = (us * 14.745600) + 8; //+8 pour arrondir à la 1/2 us supérieure lors du passage float -> int
147     finalValue = us * 15;
148

```

```
149 // Wait until the while loop is not true anymore
150 //     while(appData.usCounter32 < us){}
151     while (DRV_TMR3_CounterValueGet() < finalValue);
152 // Stops Timer3
153 DRV_TMR3_Stop();
154 }
155
156
157 //-----// inv_imu_sleep_ms
158 /** @brief This function is in charge of delaying the program for a certain
159 *          time.
160 *
161 *      @param[in]      ms           time in millisecond
162 *      @return         -           -
163 */
164 void inv_imu_sleep_ms(uint32_t ms){
165
166     uint32_t i;
167     for (i = 0; i < ms; i++) {
168
169         inv_imu_sleep_us(1000);
170     }
171 }
172
173
174 //-----// inv_imu_get_time_us
175 /** @brief This function is in charge of TIMEOUT uses
176 *
177 *      @param[in]      -           -
178 *      @return         xxx
179 */
180 uint64_t inv_imu_get_time_us(void){
181
182     // Not in int64 but normal
183     return DRV_TMR2_CounterValueGet() / 15; // + appData.usCounter64;
184 }
```

```
1 /*
2  * File:    inv_imu_personnal_functions.h
3  * Author:  ricch
4  *
5  * Created on 17. mai 2023, 10:56
6  */
7
8 #ifndef INV_IMU_PERSONNAL_FUNCTIONS_H
9 #define INV_IMU_PERSONNAL_FUNCTIONS_H
10
11 #ifdef __cplusplus
12 extern "C" {
13 #endif
14
15
16
17     int get_imu_data(void);
18     int configureImuDevice(void);
19     int setupImuDevice(struct inv_imu_serif *icm_serif);
20     void inv_imu_sleep_us(uint32_t us);
21     void inv_imu_sleep_ms(uint32_t ms);
22     uint64_t inv_imu_get_time_us(void);
23
24
25
26
27 #ifdef __cplusplus
28 }
29 #endif
30
31 #endif/* INV_IMU_PERSONNAL_FUNCTIONS_H */
```

```

1
2 /*
3  * File:      RN4678_driver.c
4  * Author:    M.Ricchieri
5  *
6  * Inspired by the code of S. Giuseppe
7  * Created on 12. avril 2023
8  *
9  * This code uses USART with FIFO
10 */
11
12
13 //-----// Includes
14 #include <stdbool.h>
15 #include <stdint.h>
16 #include "RN4678_driver.h"
17 #include "app.h"
18
19
20 //-----// Constants
21 // Commands
22 #define CMD_MODE_ENTER      "$$$\r"
23 #define CMD_MODE_EXIT       "---\r"
24 #define CMD_BLE_DISCOV_EN   "Q,0\r"
25 // The module is able to connect, but is undiscoverable in Bluetooth Classic
26 #define CMD_BT_DISCOV_DIS   "Q,2\r"
27 #define CMD_BLE_ONLY        "SG,1\r"
28 #define CMD_BT_CLASSIC_ONLY "SG,2\r"
29 #define CMD_PREFIX_SUFFIX   "SO,<,>\r"
30 #define CMD_REBOOT_DEVICE   "R,1\r"
31 #define CMD_BITMAP          "SQ,8000\r"
32 #define CMD_SCAN_DURATION   "SL,01\r"
33
34 // Answers
35 #define CMD_MODE_ANSWER     "CMD> "
36 #define CMD_EXIT_ANSWER    "END\r\n"
37 #define CMD_POS_ANSWER     "AOK\r\nCMD> "
38 #define CMD_NEG_ANSWER     "ERR\r\nCMD> "
39 //#define CMD_REBOOT_ANSWER "<REBOOT>"
40 #define CMD_REBOOT_ANSWER  "Rebooting\r\n"
41
42 // Device name
43 #define DEVICE_NAME         "SN,TubePitotDeporte_v1.0.0\r"
44
45
46
47 //-----// init_RN4678
48 bool init_RN4678(void){
49
50     bool initIsDone = 1;
51
52     //Resets the module for a reboot
53     RESET_BLEOff();
54     inv_imu_sleep_ms(1000);
55     RESET_BLEOn();
56     inv_imu_sleep_ms(2000);
57
58     appData.isBluetoothInCommandMode = true;
59     // Enters in command mode
60     initIsDone = sendCMD_RN4678(CMD_MODE_ENTER, sizeof(CMD_MODE_ENTER), CMD_MODE_ANSWER,
61                                 sizeof(CMD_MODE_ANSWER));
62     // Sets the name of the device
63     initIsDone &= sendCMD_RN4678(DEVICE_NAME, sizeof(DEVICE_NAME), CMD_POS_ANSWER,
64                                 sizeof(CMD_POS_ANSWER));
65     // Sets the Bluetooth mode in Classic
66     initIsDone &= sendCMD_RN4678(CMD_BT_CLASSIC_ONLY, sizeof(CMD_BT_CLASSIC_ONLY), CMD_POS_ANSWER,
67                                 sizeof(CMD_POS_ANSWER));
68     // Sets the prefix and the sufix of status
69     initIsDone &= sendCMD_RN4678(CMD_PREFIX_SUFFIX, sizeof(CMD_PREFIX_SUFFIX), CMD_POS_ANSWER,
70                                 sizeof(CMD_POS_ANSWER));
71     // Sets the scan duration to 10 seconds
72     initIsDone &= sendCMD_RN4678(CMD_SCAN_DURATION, sizeof(CMD_SCAN_DURATION), CMD_POS_ANSWER,
73                                 sizeof(CMD_POS_ANSWER));
74     // Sets the RN4678 into Fast mode

```

```

75     initIsDone &= sendCMD_RN4678(CMD_BITMAP, sizeof(CMD_BITMAP), CMD_POS_ANSWER,
76         sizeof(CMD_POS_ANSWER));
77     // Lauches a reboot command
78     initIsDone &= sendCMD_RN4678(CMD_REBOOT_DEVICE, sizeof(CMD_REBOOT_DEVICE), CMD_REBOOT_ANSWER,
79         sizeof(CMD_REBOOT_ANSWER));
80
81     inv_imu_sleep_ms(2000);
82
83     // Flag is discoverable true
84     appData.isBluetoothDiscoverable = true;
85
86     appData.isBluetoothInCommandMode = false;
87
88     return initIsDone;
89 }
90
91 //-----// turnOffDiscoverBT
92 // For the moment, this function isn't used. The Fast mode affects the detection
93 // of the "$$$\\r" message and it is impossible to inter in command mode when the
94 // data mode is enable.
95 bool turnOffDiscoverBT(void) {
96
97     appData.isBluetoothInCommandMode = true;
98     // Enters in command mode
99     sendCMD_RN4678(CMD_MODE_ENTER, sizeof(CMD_MODE_ENTER), CMD_MODE_ANSWER,
100         sizeof(CMD_MODE_ANSWER));
101    // Turn off the discoverable mode of the module
102    sendCMD_RN4678(CMD_BT_DISCOV_DIS, sizeof(CMD_BT_DISCOV_DIS), CMD_POS_ANSWER,
103        sizeof(CMD_POS_ANSWER));
104    // Exits command mode
105    sendCMD_RN4678(CMD_MODE_EXIT, sizeof(CMD_MODE_EXIT), CMD_EXIT_ANSWER,
106        sizeof(CMD_EXIT_ANSWER));
107
108    appData.isBluetoothInCommandMode = false;
109
110    return 1;
111 }
112
113
114 //-----// sendCMD_RN4678
115 bool sendCMD_RN4678(char* pArrayToSend, size_t arraySize, char* pArrayExpected,
116     size_t answerSize){
117
118     int8_t a_answer[20];
119
120     // Save data in TX FIFO
121     putStringInFifo(&uartFifoTx, arraySize, pArrayToSend);
122     // Enable USART TX interrupt
123     PLIB_INT_SourceEnable(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);
124
125     do{
126         // If the number of new char in FIFO is the same as the answer size
127         if(getReadSize(&uartFifoRx) >= answerSize - 1){
128
129             // Reads the answer received
130             getStringFromFifo(&uartFifoRx, &a_answer[0]);
131         }
132
133     }while((strstr((char*)a_answer, pArrayExpected) == NULL));
134     //if(strstr((char*)a_answer, pArrayExpected) != NULL) isInitDone = 1;
135
136     //}while(isInitDone != 1);
137
138     clearInt8Array(sizeof(a_answer), &a_answer[0]);
139
140     return 1;
141 }
142
143
144 //-----// getUsartData
145 void getUsartData(int8_t* pArrayToModify){
146
147     do{
148         // Reads the answer received

```

```
149     getStringFromFifo(&uartFifoRx, &pArrayToModify[0]);
150
151 }while(getReadSize(&uartFifoRx));
152 }
153
154
155 //-----// sendData_RN4678
156 void sendData_RN4678(int8_t* pArrayToSend) {
157
158     int cursor = 0;
159
160     // Does until character '\r' is sent
161     do{
162         // Wait for the Transmit buffer to be empty.
163         if(!PLIB_USART_TransmitterBufferIsFull(USART_ID_1)){
164
165             // Sends all data of the array
166             PLIB_USART_TransmitterByteSend(USART_ID_1, pArrayToSend[cursor]);
167             cursor++;
168         }
169     }while(pArrayToSend[cursor-1] != '\r');
170 }
171
172
173 //-----// readStatus
174 void readStatus(char *pArrayStatus) {
175
176     int cursor = 0;
177
178     while(PLIB_USART_ReceiverDataIsAvailable(USART_ID_1)){
179         // Reads and saves the characters received in an array
180         pArrayStatus[cursor] = PLIB_USART_ReceiverByteReceive(USART_ID_1);
181         // Increments the cursor value
182         cursor++;
183     }
184 }
185
186
187 //-----// clearInt8Array
188 void clearInt8Array(size_t arraySize, int8_t* arrayToClear) {
189
190     int i;
191
192     for (i = 0; i < arraySize; i++){
193
194         arrayToClear[i] = NULL;
195     }
196 }
```

```
1
2 /*
3  * File:      RN4678_driver.h
4  * Author:    M.Ricchieri
5  *
6  * Created on 12. avril 2023
7  *
8  * This code uses USART with FIFO
9  */
10
11 #ifndef RN4678_DRIVER_H
12 #define RN4678_DRIVER_H
13
14 //-----// Includes
15 #include <stdbool.h>
16 #include <stdint.h>
17 #include <stddef.h>
18
19 #ifdef __cplusplus
20 extern "C" {
21 #endif
22
23
24 //-----// Functions
25 bool init_RN4678(void);
26 bool turnOffDiscoverBT(void);
27 bool sendCMD_RN4678(char* pArrayToSend, size_t arraySize, char* pArrayExpected,
28           size_t answerSize);
29
30 void sendData_RN4678(int8_t* pArrayToSend);
31 void performAction(const char* word);
32 bool searchWord(char* message, const char* word);
33 void readStatus();
34 void clearInt8Array(size_t arraySize, int8_t* arrayToClear);
35 void getUsartData();
36
37
38 #ifdef __cplusplus
39 }
40#endif
41
42#endif/* RN4678_DRIVER_H */
```

```

1 ****
2 System Interrupts File
3
4 File Name:
5     system_interrupt.c
6
7 Summary:
8     Raw ISR definitions.
9
10 Description:
11     This file contains a definitions of the raw ISRs required to support the
12     interrupt sub-system.
13
14 Summary:
15     This file contains source code for the interrupt vector functions in the
16     system.
17
18 Description:
19     This file contains source code for the interrupt vector functions in the
20     system. It implements the system and part specific vector "stub" functions
21     from which the individual "Tasks" functions are called for any modules
22     executing interrupt-driven in the MPLAB Harmony system.
23
24 Remarks:
25     This file requires access to the systemObjects global data structure that
26     contains the object handles to all MPLAB Harmony module objects executing
27     interrupt-driven in the system. These handles are passed into the individual
28     module "Tasks" functions to identify the instance of the module to maintain.
29 ****/
30
31 // DOM-IGNORE-BEGIN
32 ****
33 Copyright (c) 2011-2014 released Microchip Technology Inc. All rights reserved.
34
35 Microchip licenses to you the right to use, modify, copy and distribute
36 Software only when embedded on a Microchip microcontroller or digital signal
37 controller that is integrated into your product or third party product
38 (pursuant to the sublicense terms in the accompanying license agreement).
39
40 You should refer to the license agreement accompanying this Software for
41 additional information regarding your rights and obligations.
42
43 SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND,
44 EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF
45 MERCHANTABILITY, TITLE, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.
46 IN NO EVENT SHALL MICROCHIP OR ITS LICENSORS BE LIABLE OR OBLIGATED UNDER
47 CONTRACT, NEGLIGENCE, STRICT LIABILITY, CONTRIBUTION, BREACH OF WARRANTY, OR
48 OTHER LEGAL EQUITABLE THEORY ANY DIRECT OR INDIRECT DAMAGES OR EXPENSES
49 INCLUDING BUT NOT LIMITED TO ANY INCIDENTAL, SPECIAL, INDIRECT, PUNITIVE OR
50 CONSEQUENTIAL DAMAGES, LOST PROFITS OR LOST DATA, COST OF PROCUREMENT OF
51 SUBSTITUTE GOODS, TECHNOLOGY, SERVICES, OR ANY CLAIMS BY THIRD PARTIES
52 (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF), OR OTHER SIMILAR COSTS.
53 ****/
54 // DOM-IGNORE-END
55
56 // ****
57 // ****
58 // Section: Included Files
59 // ****
60 // ****
61
62 #include "system/common/sys_common.h"
63 #include "app.h"
64 #include "system_definitions.h"
65
66 // ****
67 // ****
68 // Section: System Interrupt Vector Functions
69 // ****
70 // ****
71 void __ISR(_UART_1_VECTOR, ipl7AUTO) _IntHandlerDrvUsartInstance0(void)
72 {
73
74     USART_ERROR usartStatus;

```

```

75     bool      isTxBuffFull;
76     static bool      isStatusBeg = false;
77     int8_t      charReceived;
78     int8_t      charToSend;
79     int8_t      charTrash;
80     int8_t      TXsize;
81
82 //-----// RX interrupt
83 if(PLIB_INT_SourceFlagGet(INT_ID_0, INT_SOURCE_USART_1_RECEIVE) &&
84     PLIB_INT_SourceIsEnabled(INT_ID_0, INT_SOURCE_USART_1_RECEIVE)){
85
86     // Parity error or overrun
87     usartStatus = PLIB_USART_ErrorsGet(USART_ID_1);
88
89     if ((usartStatus & (USART_ERROR_PARITY | USART_ERROR_FRAMING |
90         USART_ERROR_RECEIVER_OVERRUN)) == 0){
91
92         // All char received are transferred to the FIFO
93         // 1 if ONE_CHAR, 4 if HALF_FULL and 6 3B4FULL
94         while(PLIB_USART_ReceiverDataIsAvailable(USART_ID_1)){
95
96             charReceived = PLIB_USART_ReceiverByteReceive(USART_ID_1);
97
98             putCharInFifo(&usartFifoRx, charReceived);
99
100            // Beginning of a status
101            if(charReceived == '<' && appData.isBluetoothInCommandMode == false) isStatusBeg = true;
102
103            // Ending of a status
104            if(appData.isBluetoothModuleInit && charReceived == '>' &&
105                appData.isBluetoothInCommandMode == false && isStatusBeg == true){
106
107                isStatusBeg = false;
108                USART1_Callback_Function();
109            }
110
111            // If(isStatusBeg == false && appData.isBluetoothInCommandMode ==
112                false) getCharFromFifo(&usartFifoRx, &charTrash);
113        }
114
115        // Buffer is empty, clear interrupt flag
116        PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_USART_1_RECEIVE);
117
118    }else{
119        // Deleting errors
120        // Reading errors clears them except for overrun
121        if((usartStatus & USART_ERROR_RECEIVER_OVERRUN) ==
122            USART_ERROR_RECEIVER_OVERRUN){
123
124            PLIB_USART_ReceiverOverrunErrorClear(USART_ID_1);
125        }
126    }
127
128
129 //-----// TX interrupt
130 if (PLIB_INT_SourceFlagGet(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT) &&
131     PLIB_INT_SourceIsEnabled(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT)){
132
133     TXsize = getReadSize(&usartFifoTx);
134     // i_cts = input(RS232_CTS);
135
136     isTxBuffFull = PLIB_USART_TransmitterBufferIsFull(USART_ID_1);
137
138     if /*(i_cts == 0) && *(TXsize > 0) && (isTxBuffFull == false)){
139         do{
140             getCharFromFifo(&usartFifoTx, &charToSend);
141             if(charToSend != '\0') PLIB_USART_TransmitterByteSend(USART_ID_1, charToSend);
142             /*i_cts = RS232_CTS;*/
143             TXsize = getReadSize (&usartFifoTx);
144             isTxBuffFull = PLIB_USART_TransmitterBufferIsFull(USART_ID_1);
145
146         }while/*(i_cts == 0) && *( TXsize > 0 ) && isTxBuffFull == false);
147     }
148
149     // Disables TX interrupt (to avoid unnecessary interruptions if there's

```

```
149     // nothing left to transmit)
150     if(TXsize == 0) {
151
152         PLIB_INT_SourceDisable(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);
153     }
154     // Clears the TX interrupt Flag
155     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_USART_1_TRANSMIT);
156 }
157 }
158
159
160 //-----// TIMER0 ID1 <--- Disabled
161 void __ISR(_TIMER_1_VECTOR, ipl1AUTO) IntHandlerDrvTmrInstance0(void) {
162 //
163     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_1);
164     TIMER0_Callback_Function();
165 }
166
167
168 //-----// TIMER1 ID2
169 void __ISR(_TIMER_2_VECTOR, ipl2AUTO) IntHandlerDrvTmrInstance1(void)
170 {
171     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_2);
172     TIMER1_Callback_Function();
173 }
174
175
176 //-----// TIMER2 ID5
177 void __ISR(_TIMER_5_VECTOR, ipl0AUTO) IntHandlerDrvTmrInstance2(void)
178 {
179     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_5);
180 }
181
182
183 //-----// TIMER3 ID3
184 void __ISR(_TIMER_3_VECTOR, ipl0AUTO) IntHandlerDrvTmrInstance3(void)
185 {
186     PLIB_INT_SourceFlagClear(INT_ID_0, INT_SOURCE_TIMER_3);
187 }
188
189
190 ****
191 End of File
192 */
```

```

1
2 /*
3  * File:    adc_driver.c
4  * Author:  M.Ricchieri
5  *
6  * Created on 31. mai 2023, 08:55
7  *
8  * Inspired by the "Mc32DriverAdc.c" file
9  */
10
11
12 //-----// Includes
13 #include "voltageADC_driver.h"
14
15
16 //-----// Constants
17 #define ADC_RESOLUTION 1024
18 #define ADC_VREF      3.3
19
20 //-----
21 // Help to select the SCAN_MASK value
22 //
23 // AN10  AN9   AN8   AN7   AN6   AN5   AN4   AN3   AN2   AN1   AN0
24 // |     |     |     |     |     |     |     |     |     |     |
25 // 1     1     0     0     0     0     0     0     0     0     0
26 //
27 // Value in CONFIGSCAN = 0b0011 0000 0000 = 0x600
28 //
29 #define SCAN_MASK 0x0600
30 //-----
31
32
33 //-----// initAdc
34 void initAdc(void){
35
36     // Mask configuration
37     PLIB_ADC_InputScanMaskAdd(ADC_ID_1, SCAN_MASK);
38     // Data return configuration
39     PLIB_ADC_ResultFormatSelect(ADC_ID_1, ADC_RESULT_FORMAT_INTEGER_16BIT);
40     // Alternate buffer selection
41     PLIB_ADC_ResultBufferModeSelect(ADC_ID_1, ADC_BUFFER_MODE_TWO_8WORD_BUFFERS);
42     // Multiplex mode selection
43     PLIB_ADC_SamplingModeSelect(ADC_ID_1, ADC_SAMPLING_MODE_MUXA);
44     PLIB_ADC_ConversionTriggerSourceSelect(ADC_ID_1,
45                                         ADC_CONVERSION_TRIGGER_INTERNAL_COUNT);
46     // Reference selection
47     PLIB_ADC_VoltageReferenceSelect(ADC_ID_1, ADC_REFERENCE_VDD_TO_AVSS );
48     PLIB_ADC_SampleAcquisitionTimeSet(ADC_ID_1, 0x1F);
49
50     PLIB_ADC_ConversionClockSet(ADC_ID_1, SYS_CLK_FREQ, 32);
51     // Configuration of number of readings (depends on the number of inputs)
52     PLIB_ADC_SamplesPerInterruptSelect(ADC_ID_1, ADC_2SAMPLES_PER_INTERRUPT);
53     PLIB_ADC_MuxAInputScanEnable(ADC_ID_1);
54     // Enable the ADC module
55     PLIB_ADC_Enable(ADC_ID_1);
56
57
58 //-----// readRawAdc
59 void readRawAdc(RAW_ADC *pRawAdc){
60
61     ADC_RESULT_BUF_STATUS BufStatus;
62
63     // Stop sample/convert
64     PLIB_ADC_SampleAutoStartDisable(ADC_ID_1);
65
66     // Treatment with alternating buffer
67     BufStatus = PLIB_ADC_ResultBufferStatusGet(ADC_ID_1);
68
69     if (BufStatus == ADC_FILLING_BUF_0TO7){
70
71         pRawAdc->AN9_V_GEN = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 0);
72         pRawAdc->AN10_V_BAT = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 1);
73     }else{
74

```

```
75     pRawAdc->AN9_V_GEN = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 8);
76     pRawAdc->AN10_V_BAT = PLIB_ADC_ResultGetByIndex(ADC_ID_1, 9);
77 }
78
79 // Auto start sampling
80 PLIB_ADC_SampleAutoStartEnable(ADC_ID_1);
81 }
82
83
84 //-----// convertRawToVoltage
85 void convertRawToVoltage(RAW_ADC *pRawAdc, SENS_DATA *pSensData){
86
87     // Converts RAW data of the battery voltage into a real decimal value
88     // The number 2 is present because of the hardware bridge divider
89     pSensData->batVoltage = (2 *(pRawAdc->AN10_V_BAT *
90         (ADC_VREF / ADC_RESOLUTION)));
91
92     // Converts RAW data of the generator voltage into a real decimal value
93     // The number 5 is present because of the hardware bridge divider
94     pSensData->genVoltage = (5 *(pRawAdc->AN9_V_GEN *
95         (ADC_VREF / ADC_RESOLUTION)));
96 }
```

```
1 /*
2  * File:    adc_driver.h
3  * Author:  M.Ricchieri
4  *
5  * Created on 31. mai 2023, 08:55
6  */
7
8 #ifndef ADC_DRIVER_H
9 #define ADC_DRIVER_H
10
11 #ifdef __cplusplus
12 extern "C" {
13 #endif
14
15 //-----// Includes
16 #include "app.h"
17 #include "peripheral/adc/plib_adc.h"
18
19
20 //-----// Functions
21 void initAdc(void);
22 void readRawAdc(RAW_ADC *pRawAdc);
23 void convertRawToVoltage(RAW_ADC *pRawAdc, SENS_DATA *pSensData);
24
25 #ifdef __cplusplus
26 }
27#endif
28
29#endif/* ADC_DRIVER_H */
```

```
1 package de.kai_morich.simple_bluetooth_terminal;
2
3 import android.annotation.SuppressLint;
4 import android.app.Activity;
5 import android.app.AlertDialog;
6 import android.bluetooth.BluetoothAdapter;
7 import android.bluetooth.BluetoothDevice;
8 import android.content.ComponentName;
9 import android.content.Context;
10 import android.content.Intent;
11 import android.content.ServiceConnection;
12 import android.os.Bundle;
13 import android.os.IBinder;
14 import android.text.Spannable;
15 import android.text.SpannableStringBuilder;
16 import android.text.style.ForegroundColorSpan;
17 import android.view.LayoutInflater;
18 import android.view.Menu;
19 import android.view.MenuInflater;
20 import android.view.MenuItem;
21 import android.view.View;
22 import android.view.ViewGroup;
23 import android.widget.TextView;
24 import android.widget.Toast;
25
26 import androidx.annotation.NonNull;
27 import androidx.annotation.Nullable;
28 import androidx.fragment.app.Fragment;
29
30 import java.util.ArrayDeque;
31 import static java.lang.Math.*;
32
33
34
35 public class TerminalFragment extends Fragment implements ServiceConnection,
36 SerialListener {
37
38     private enum Connected { False, Pending, True }
39
40     private String deviceAddress;
41     private SerialService service;
42
43     //private TextView receiveText;
44
45     private TextView speedText;
46     private TextView gyroXText;
47     private TextView gyroYText;
48     private TextView gyroZText;
49     private TextView anglGyrXText;
50     private TextView anglGyrYText;
51     private TextView anglGyrZText;
52     private TextView accelXText;
53     private TextView accelYText;
54     private TextView accelZText;
55     private TextView vbatText;
56     private TextView vgenText;
57     private TextView pitchText;
58     private TextView yawText;
59     private TextView rollText;
60
61     private TextView sendText;
62     private TextUtil.HexWatcher hexWatcher;
63
64     private Connected connected = Connected.False;
65     private boolean initialStart = true;
```

```
66 private boolean hexEnabled = false;
67 private boolean pendingNewline = false;
68 private String newline = TextUtil.newLine_crlf;
69
70 // Personal variables
71 public double sValue = 0.0;
72 public double gxValue = 0.0;
73 public double gyValue = 0.0;
74 public double gzValue = 0.0;
75 public double gaxValue = 0.0;
76 public double gayValue = 0.0;
77 public double gazValue = 0.0;
78 public double axValue = 0.0;
79 public double ayValue = 0.0;
80 public double azValue = 0.0;
81 public double vbatValue = 0.0;
82 public double vgenValue = 0.0;
83
84 /*
85 * Lifecycle
86 */
87 @Override
88 public void onCreate(@Nullable Bundle savedInstanceState) {
89     super.onCreate(savedInstanceState);
90     setHasOptionsMenu(true);
91     setRetainInstance(true);
92     deviceAddress = getArguments().getString("device");
93 }
94
95 @Override
96 public void onDestroy() {
97     if (connected != Connected.False)
98         disconnect();
99     getActivity().stopService(new Intent(getActivity(), SerialService.class));
100    super.onDestroy();
101 }
102
103 @Override
104 public void onStart() {
105     super.onStart();
106     if(service != null)
107         service.attach(this);
108     else
109         getActivity().startService(new Intent(getActivity(), SerialService.class));
110         // prevents service destroy on unbind from recreated activity caused by
111         // orientation change
112 }
113
114 @Override
115 public void onStop() {
116     if(service != null && !getActivity().isChangingConfigurations())
117         service.detach();
118     super.onStop();
119 }
120
121 @SuppressWarnings("deprecation") // onAttach(context) was added with API 23.
122 onAttach(activity) works for all API versions
123 @Override
124 public void onAttach(@NonNull Activity activity) {
125     super.onAttach(activity);
126     getActivity().bindService(new Intent(getActivity(), SerialService.class), this,
127     Context.BIND_AUTO_CREATE);
128 }
129
130 @Override
131 public void onDetach() {
```

```

128         try { getActivity().unbindService(this); } catch(Exception ignored) {}
129         super.onDetach();
130     }
131
132     @Override
133     public void onResume() {
134         super.onResume();
135         if(initialStart && service != null) {
136             initialStart = false;
137             getActivity().runOnUiThread(this::connect);
138         }
139     }
140
141     @Override
142     public void onServiceConnected(ComponentName name, IBinder binder) {
143         service = ((SerialService.SerialBinder) binder).getService();
144         service.attach(this);
145         if(initialStart && isResumed()) {
146             initialStart = false;
147             getActivity().runOnUiThread(this::connect);
148         }
149     }
150
151     @Override
152     public void onServiceDisconnected(ComponentName name) {
153         service = null;
154     }
155
156     /*
157      * UI
158      */
159     @Override
160     public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
161     Bundle savedInstanceState) {
162         View view = inflater.inflate(R.layout.fragment_terminal, container, false);
163         // Assignment text to textView
164         speedText = view.findViewById(R.id.speed_viewer);
165         gyroXText = view.findViewById(R.id.gyroX_viewer);
166         gyroYText = view.findViewById(R.id.gyroY_viewer);
167         gyroZText = view.findViewById(R.id.gyroZ_viewer);
168         anglGyrXText = view.findViewById(R.id.angGyrX_viewer);
169         anglGyrYText = view.findViewById(R.id.angGyrY_viewer);
170         anglGyrZText = view.findViewById(R.id.angGyrZ_viewer);
171         accelXText = view.findViewById(R.id.accelX_viewer);
172         accelYText = view.findViewById(R.id.accelY_viewer);
173         accelZText = view.findViewById(R.id.accelZ_viewer);
174         vbatText = view.findViewById(R.id.vbat_viewer);
175         vgenText = view.findViewById(R.id.vgen_viewer);
176         pitchText = view.findViewById(R.id.pitch_viewer);
177         yawText = view.findViewById(R.id.yaw_viewer);
178         rollText = view.findViewById(R.id.roll_viewer);
179
180         //receiveText = view.findViewById(R.id.receive_text);
181         // TextView performance decreases with number of spans
182         //receiveText.setTextColor(getResources().getColor(R.color.colorRecieveText));
183         // set as default color to reduce number of spans
184         //receiveText.setMovementMethod(ScrollingMovementMethod.getInstance());
185
186         sendText = view.findViewById(R.id.send_text);
187         hexWatcher = new TextUtil.HexWatcher(sendText);
188         hexWatcher.enable(hexEnabled);
189         sendText.addTextChangedListener(hexWatcher);
190         sendText.setHint(hexEnabled ? "HEX mode" : "");
191
192         View sendBtn = view.findViewById(R.id.send_btn);
193         sendBtn.setOnClickListener(v -> send(sendText.getText().toString()));

```

```
191     return view;
192 }
193
194     @Override
195     public void onCreateOptionsMenu(@NonNull Menu menu, MenuInflater inflater) {
196         inflater.inflate(R.menu.menu_terminal, menu);
197         menu.findItem(R.id.hex).setChecked(hexEnabled);
198     }
199
200     @Override
201     public boolean onOptionsItemSelected(MenuItem item) {
202         int id = item.getItemId();
203         if (id == R.id.clear) {
204             //receiveText.setText("");
205             return true;
206         } else if (id == R.id.newline) {
207             String[] newlineNames = getResources().getStringArray(R.array.newline_names);
208             String[] newlineValues = getResources().getStringArray(R.array.newline_values);
209             int pos = java.util.Arrays.asList(newlineValues).indexOf(newline);
210             AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
211             builder.setTitle("Newline");
212             builder.setSingleChoiceItems(newlineNames, pos, (dialog, item1) -> {
213                 newline = newlineValues[item1];
214                 dialog.dismiss();
215             });
216             builder.create().show();
217             return true;
218         } else if (id == R.id.hex) {
219             hexEnabled = !hexEnabled;
220             sendText.setText("");
221             hexWatcher.enable(hexEnabled);
222             sendText.setHint(hexEnabled ? "HEX mode" : "");
223             item.setChecked(hexEnabled);
224             return true;
225         } else {
226             return super.onOptionsItemSelected(item);
227         }
228     }
229
230     /*
231      * Serial + UI
232      */
233     private void connect() {
234         try {
235             BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
236             BluetoothDevice device = bluetoothAdapter.getRemoteDevice(deviceAddress);
237             status("connecting...");
238             connected = Connected.Pending;
239             SerialSocket socket = new SerialSocket(getApplicationContext(), device);
240             service.connect(socket);
241         } catch (Exception e) {
242             onSerialConnectError(e);
243         }
244     }
245
246     private void disconnect() {
247         connected = Connected.False;
248         service.disconnect();
249     }
250
251     private void send(String str) {
252         if(connected != Connected.True) {
253             Toast.makeText(getActivity(), "not connected", Toast.LENGTH_SHORT).show();
254             return;
```

```

255
256 }
257 try {
258     String msg;
259     byte[] data;
260     if(hexEnabled) {
261         StringBuilder sb = new StringBuilder();
262         TextUtil.toHexString(sb, TextUtil.fromHexString(str));
263         TextUtil.toHexString(sb, newline.getBytes());
264         msg = sb.toString();
265         data = TextUtil.fromHexString(msg);
266     } else {
267         msg = str;
268         data = (str + newline).getBytes();
269     }
270     SpannableStringBuilder spn = new SpannableStringBuilder(msg + '\n');
271     spn.setSpan(new ForegroundColorSpan(getResources().getColor(R.color.
272         colorSendText)), 0, spn.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
273     //receiveText.append(spn);
274     service.write(data);
275 } catch (Exception e) {
276     onSerialIoError(e);
277 }
278
279 @SuppressLint("SetTextI18n")
280 private void receive(ArrayDeque<byte[]> datas) {
281     SpannableStringBuilder spn = new SpannableStringBuilder();
282     for (byte[] data : datas) {
283         if (hexEnabled) {
284             spn.append(TextUtil.toHexString(data)).append('\n');
285         } else {
286             String msg = new String(data);
287             if (newline.equals(TextUtil.newline_crlf) && msg.length() > 0) {
288                 // don't show CR as ^M if directly before LF
289                 msg = msg.replace(TextUtil.newline_crlf, TextUtil.newline_lf);
290                 // special handling if CR and LF come in separate fragments
291                 if (pendingNewline && msg.charAt(0) == '\n') {
292                     if (spn.length() >= 2) {
293                         spn.delete(spn.length() - 2, spn.length());
294                     } else {
295                         //Editable edt = receiveText.getEditableText();
296                         //if (edt != null && edt.length() >= 2)
297                         //    edt.delete(edt.length() - 2, edt.length());
298                     }
299                     pendingNewline = msg.charAt(msg.length() - 1) == '\r';
300                 }
301                 spn.append(TextUtil.toCaretString(msg, newline.length() != 0));
302             }
303         }
304     }
305     //-----
306     // PERSONAL
307     String[] pairs = spn.toString().split("\\s+"); // Séparer la chaîne en paires
308     cle-valeur
309     for (String pair : pairs) {
310         String[] parts = pair.split("="); // Séparer chaque paire clé-valeur
311         if (parts.length == 2) {
312             String key = parts[0];
313             String valueStr = parts[1];
314
315             try {
316                 double value = Double.parseDouble(valueStr);

```

```

317
318         switch (key) {
319             case "S":
320                 sValue = value;
321                 break;
322             case "GX":
323                 gxValue = value;
324                 break;
325             case "GY":
326                 gyValue = value;
327                 break;
328             case "GZ":
329                 gzValue = value;
330                 break;
331             case "GAX":
332                 gaxValue = value;
333                 break;
334             case "GAY":
335                 gayValue = value;
336                 break;
337             case "GAZ":
338                 gazValue = value;
339                 break;
340             case "AX":
341                 axValue = value;
342                 break;
343             case "AY":
344                 ayValue = value;
345                 break;
346             case "AZ":
347                 azValue = value;
348                 break;
349             case "VB":
350                 vbatValue = value;
351                 break;
352             case "VG":
353                 vgenValue = value;
354                 break;
355
356         }
357     } catch (NumberFormatException e) {
358         // Gérer l'exception si la valeur n'est pas un nombre décimal valide
359     }
360 }
361
362
363
364 speedText.setText("SPEED : " + String.valueOf(sValue) + " km/h");
365 gyroXText.setText("GYRO X : " + String.valueOf(gxValue) + " dps");
366 gyroYText.setText("GYRO Y : " + String.valueOf(gyValue) + " dps");
367 gyroZText.setText("GYRO Z : " + String.valueOf(gzValue) + " dps");
368 anglGyrXText.setText("ANGL GYR X : " + String.valueOf(gaxValue) + " °");
369 anglGyrYText.setText("ANGL GYR Y : " + String.valueOf(gayValue) + " °");
370 anglGyrZText.setText("ANGL GYR Z : " + String.valueOf(gazValue) + " °");
371 accelXText.setText("ACCEL X : " + String.valueOf(axValue) + " g");
372 accelYText.setText("ACCEL Y : " + String.valueOf(ayValue) + " g");
373 accelZText.setText("ACCEL Z : " + String.valueOf(azValue) + " g");
374 vbatText.setText("VBAT : " + String.valueOf(vbatValue) + " V");
375 vgenText.setText("VGEN : " + String.valueOf(vgenValue) + " V");
376
377 // Calculation of angles
378 //pitchValue = Math.atan(axValue / (Math.sqrt(Math.pow(ayValue, 2) +
379 Math.pow(azValue, 2))));
```

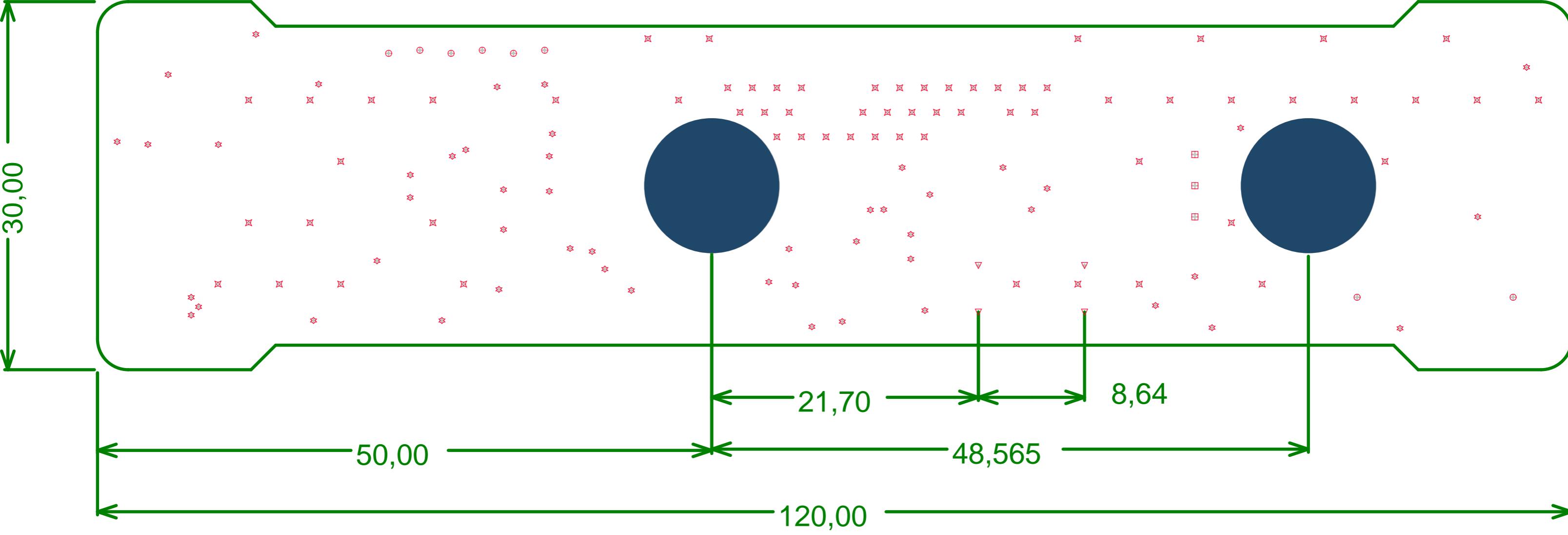
//double angle = atan2(ayValue, axValue); // Calcul de l'angle en radians

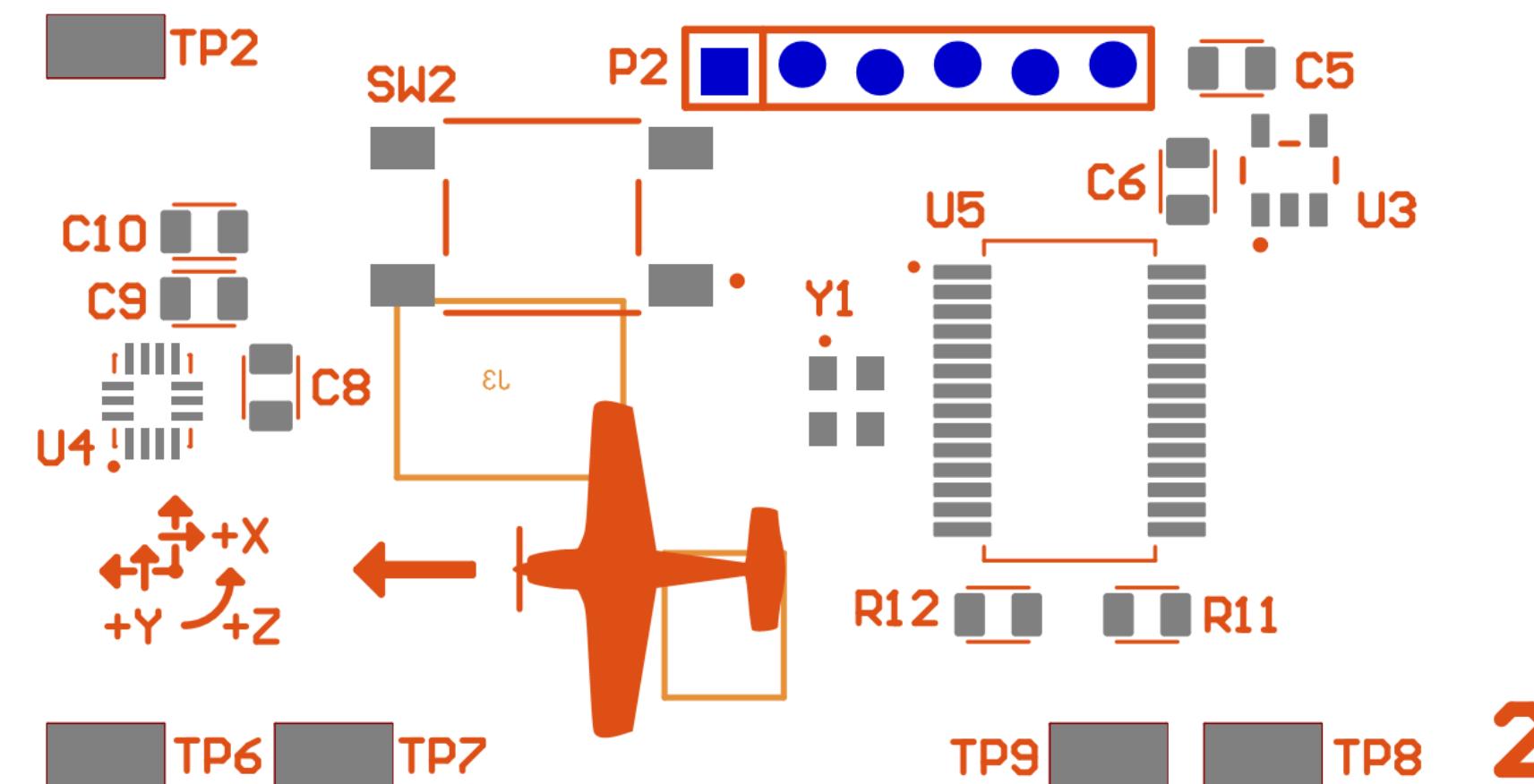
```

382     //double angle = atan(axValue/ayValue);
383     // Conversion de l'angle en degrés et en float
384     //float pitchValue = (float) toDegrees(angle);
385
386     // Calcul des angles
387     double roll = Math.atan2(ayValue, azValue);
388     double pitch = Math.atan2(-axValue, Math.sqrt(ayValue * ayValue + azValue * azValue));
389     double yaw = Math.atan2(Math.sin(roll) * axValue - Math.cos(roll) * ayValue, Math.cos(roll) * azValue);
390
391     // Conversion des angles en degrés
392     //roll = Math.toDegrees(roll);
393     pitch = Math.toDegrees(pitch);
394     yaw = Math.toDegrees(yaw);
395
396     pitchText.setText("PITCH : " + String.format("%.2f", pitch) + " °");
397     yawText.setText("YAW : " + String.format("%.2f", yaw) + " °");
398     //rollText.setText("ROLL : " + String.format("%.2f", roll) + " °");
399
400
401     /*
402      receiveText.append("Speed = " + sValue + " km/h" + " GyroX = " + gxValue + "
403      GyroY = " + gyValue +
404          " GyroZ = " + gzValue + " AccelX = " + axValue + " AccelY = " + ayValue
405          +
406          " AccelZ = " + azValue + " Vbat = " + vbatValue + " Vgen = " +
407          vgenValue + '\n');
408
409
410     */
411
412
413     private void status(String str) {
414         SpannableStringBuilder spn = new SpannableStringBuilder(str + '\n');
415         spn.setSpan(new ForegroundColorSpan(getResources().getColor(R.color.
416             colorStatusText)), 0, spn.length(), Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
417         //receiveText.append(spn);
418     }
419
420     /*
421      * SerialListener
422      */
423     @Override
424     public void onSerialConnect() {
425         status("connected");
426         connected = Connected.True;
427     }
428
429     @Override
430     public void onSerialConnectError(Exception e) {
431         status("connection failed: " + e.getMessage());
432         disconnect();
433     }
434
435     @Override
436     public void onSerialRead(byte[] data) {
437         ArrayDeque<byte[]> datas = new ArrayDeque<>();
438         datas.add(data);
439         receive(datas);
440     }

```

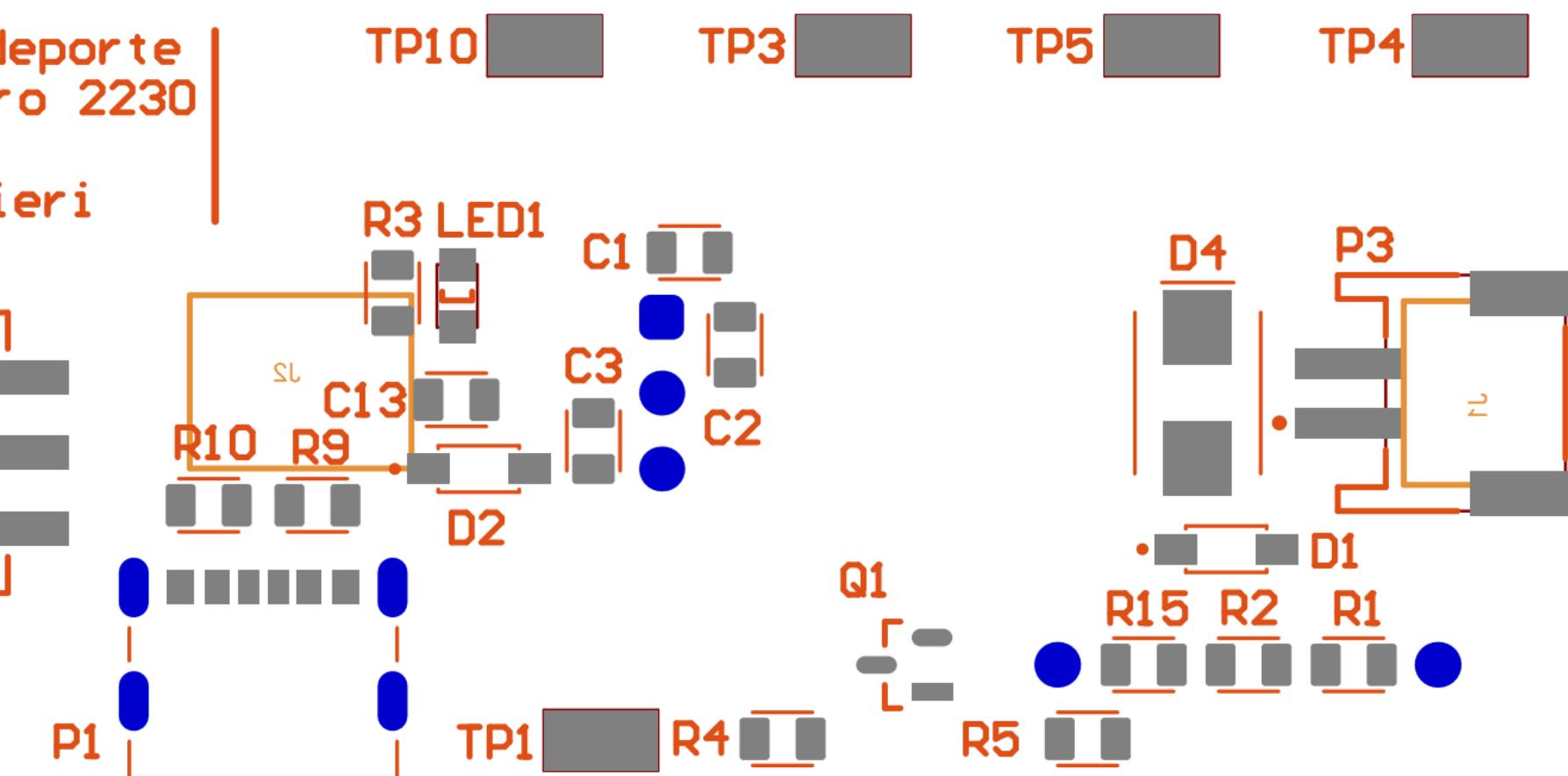
```
440
441     public void onSerialRead(ArrayDeque<byte[]> datas) {
442         receive(datas);
443     }
444
445     @Override
446     public void onSerialIoError(Exception e) {
447         status("connection lost: " + e.getMessage());
448         disconnect();
449     }
450
451 }
452
```

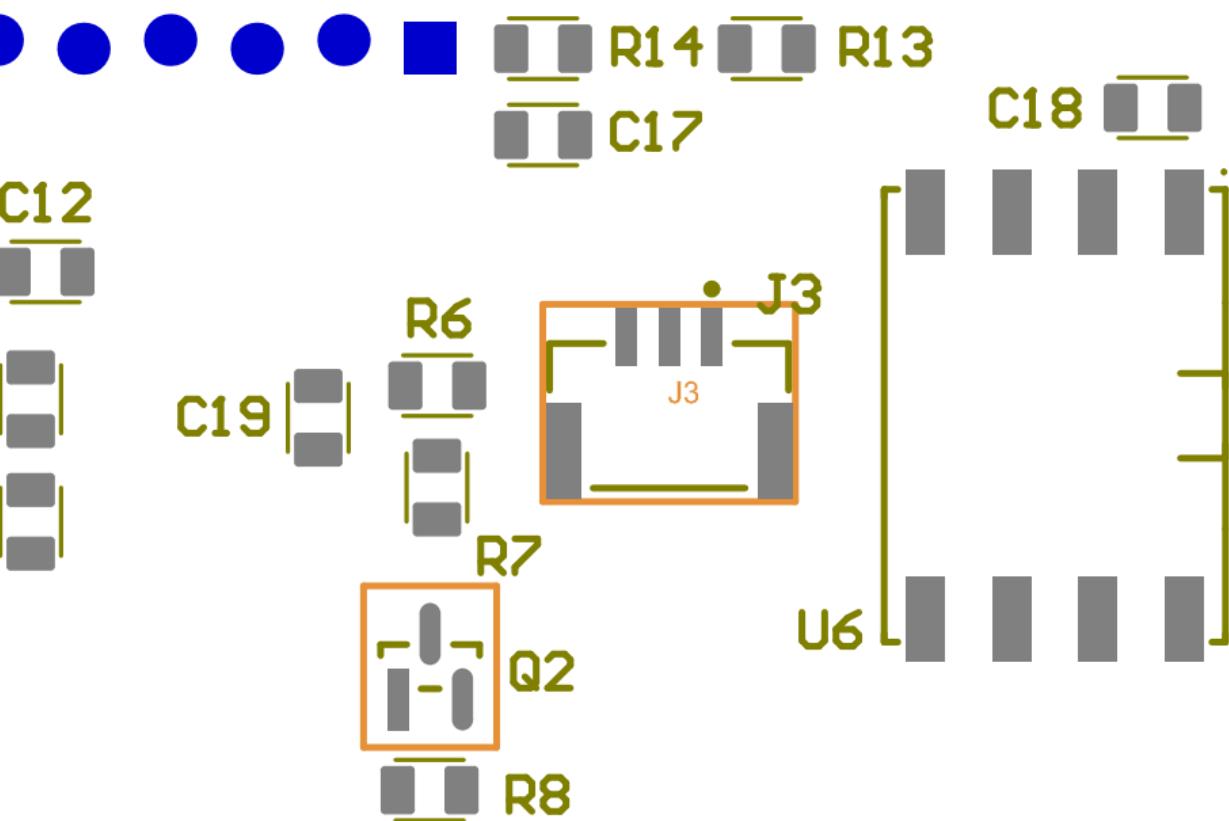
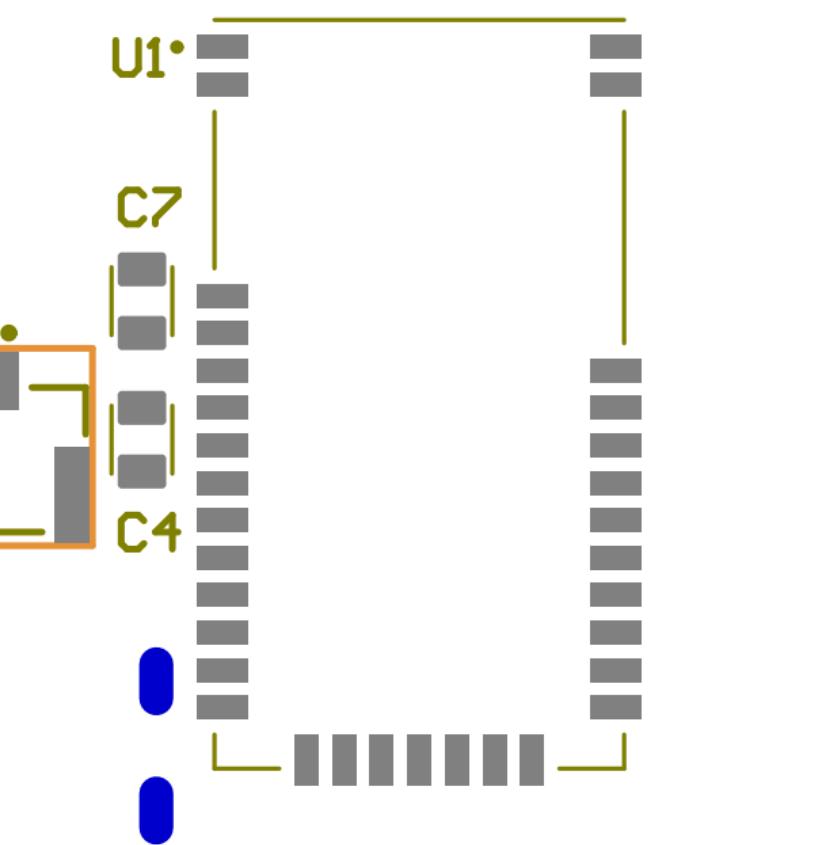
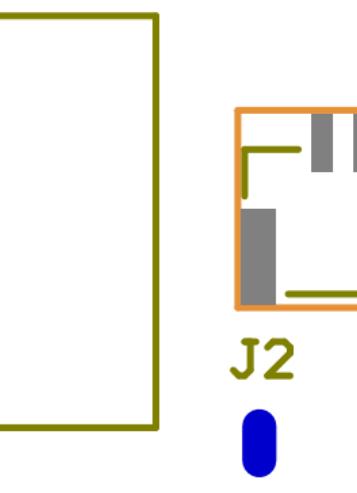
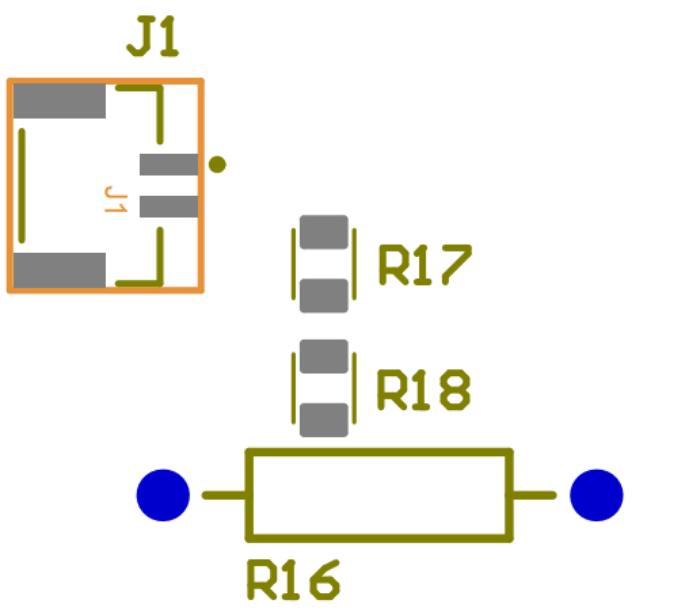




2023

Tube Pitot deporte  
Projet numero 2230  
v1.0.0  
Meven Ricchieri





UJC-HP-3-SMT-TR	USBC connector	Digi-Key	P1	No	Not managed	1 CUI Devices	UJC-HP-3-SMT-TR	Volume Production 2223-UJC-HP-3-SMT-CT-ND	1.16	1.16 Clear
2 TLV7433PDBVR	IC REG LINEAR 3.3V 300MA SOT23-5	Mouser	U3	No	CMP-04917-000051-1 New From Up to date	1 Texas Instruments	TLV7433PDBVR	Volume Production 595-TLV7433PDBVR	0.31	0.31 Clear
3 SM02B-GHS-TB(LF)(SN)	CONN HEADER SMD R/A 2P0S 1.25MM	Digi-Key	J1	No	CMP-2000-07076-2 Released Up to date	1 JST	SM02B-GHS-TB(LF)(SN)	Unknown 455-1564-2-ND	0.45	0.45 Clear
4 S2B-PH-SM4-TB(LF)(SN)	Male Header, Pitch 2 mm, 1 x 2 Position, Height 5.5 mm, -25 to 85 degC, RoHS, Tape and Reel	Digi-Key	P3	No	CMP-1755-00001-1 Released Up to date	1 IST Sales America Inc.	S2B-PH-SM4-TB(LF)(SN)	Manual Solution 455-1749-2-ND	0.53702	0.53702 Clear
5 RN4678-V/100	Bluetooth Bluetooth v5.0 Transceiver Module 2.4GHz Integrated, Chip Surface Mount	Digi-Key	U1	No	Not managed	1 Microchip	RN4678-V/RM100	Volume Production RN4678-V/RM100-ND	13.82	13.82 Clear
RAM-Mount Diamond	RAM-Mount fixation			Yes	Not managed	1				Fatal error
16 PIC32MX130F064B-I/SS	PIC32MX130F064B-I/SS	Digi-Key	U5	Yes	Not managed	1 Microchip	PIC32MX130F064B-I/SS	Volume Production PIC32MX130F064B-I/SS-ND	3.54	3.54 Clear
LTL-4223	THT LED RED	Digi-Key		No	Not managed	1 Vishay Lite-On	LTL-4223	Volume Production 160-1127-ND	0.36	0.36 Clear
Jeti Tube Pitot DUPLEX	Pitot			Yes	Not managed	1				Fatal error
ICSP	Prog	Digi-Key	P2	Yes	Not managed	1 Adam Equipment	PH1-05-UA	Unknown 2057-PH1-05-UA-ND	0.1	0.1 Clear
19 ICM-42670-P	6 axis IMU	Mouser	U4	No	Not managed	1 TDK	ICM-42670-P	Unknown 410-ICM-42670-P	5.97	5.97 Clear
GHR-03V-S	1.25mm IST female connector 3P	Conrad		No	Not managed	3 JST	GHR-03V-S	Manual Solution 1426141 0.11686 0.47471 Clear		
GHR-02V-S	1.25mm IST female connector 2P	Digi-Key		No	Not managed	2 JST	GHR-02V-S	Unknown 455-1592-ND	0.13	0.13 Clear
20 FSM2JS5MAA	SWITCH TACTILE SPST-NO 0.05A 24V	Mouser	SW2	Yes	CMP-03402-000003-1 New From Up to date	1 TE Connectivity	FSM2JS5MAA	Volume Production 506-2-1437565-8	0.23	0.23 Clear
ECS-0323-147.4-BN	ECS-2033-147.4-BN 14.7456 MHz Oscillator	Digi-Key	Y1	No	Not managed	1 ECS International	ECS-2033-147.4-BN	Volume Production XC1429CT-ND	1.19	1.19 Clear
22 DMG3414U-7	MOSFET N-CH 20V 4.2A SOT23	Digi-Key	Q1	No	CMP-12801-000101-1 New From Up to date	2 Diodes	DMG3414U-7	Volume Production DMG3414U-7-TOCT-ND	0.46	0.46 Clear
23 BSS138	MOSFET N-CH 50V 220mA SOT-23	Digi-Key	Q2	No	CMP-07173-000101-1 New From Up to date	1 ANBON SEMICONDUCTOR (INT'L) LIMITED	BSS138	Manual Solution 4530-855138CT-ND	0.17491	0.17491 Clear
25 BM02B-GHS-TBT(LF)(SN)(N)	CONN HEADER SMD 3POS 1.25MM	Digi-Key	J2, J3	No	CMP-17439-000058-1 New From Up to date	2 JST	BM02B-GHS-TBT(LF)(SN)(N)	Unknown 455-1579-1-ND	0.47	0.47 Clear
BH24AAW	Support Files AA	Digi-Key		No	Not managed	1 Memory Protection Devices	BH24AAW	Unknown BH24AAW-ND	2	2 Clear
26 BAT46W-7-F	Zener Diode	Mouser	D1, D2	No	Not managed	2 Diodes	BAT46W-7-F	Volume Production 621-BAT46W-7-F	0.41	0.82 Clear
27 B260-13-F	Zener Diode	Anrov Electronics	D4	No	Not managed	1 Diodes	B260-13-F	Volume Production 0260-13-F	0.2809	0.2809 Clear
AGHG28K102	JST 1.25mm cable			No	Not managed	0 JST	AGHG28K102	Unknown 455-4360-ND	0.382	0.382 Clear
AGHG28Z203		Digi-Key		No	Not managed	0 JST	AGHG28Z203	Unknown 455-4361-ND	0	0 Warning
28 173950536	Buck SV	Digi-Key	U2	Yes	Not managed	1 Würth Elektronik	173950536	Manual Solution 732-173950536-ND	6.14	6.14 Clear
29 150080R575000	SMD mono-color Chip LED, WL-5MCW, Red	Mouser	LED1	No	CMP-1426-00011-1 Released Up to date	1 Würth Elektronics	150080R575000	Volume Production 710-150080R575000	0.19	0.19 Clear
30 5019	Test Point, 1 Position SMD, RoHS, Tape and Reel	Mouser	TP1, TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10	No	CMP-1672-000008-1	Released Up to date	10 Keystone Electronics	5019 Volume Production 534-5019	0.33	0.33 Clear
2254-440x219H-ND		Mouser	U6	No	Not managed	10 Tri-Star	440X219H	Unknown 2254-440X219H-ND	0.144	0.144 Clear
785-HSCMRN001PD2A3	Air sensor	Digi-Key		Yes	Not managed	1 Honeywell Sensing & Control	HSCMRN001PD2A3	Volume Production 785-HSCMRN001PD2A3	68.91	68.91 Clear
5005SP151M1QE8	Switch ON/OFF	Digi-Key		No	Not managed	1 E-Switch	5005SP151M1QE8	Unknown EG2476-ND	3.02	3.02 Fatal error
440X219H	Insert M2.5	Digi-Key		No	Not managed	6 Tri-Star	440X219H	Unknown 2254-440X219H-ND	0.172	0.172 Clear
430R	Resistor	Digi-Key	R4, R7	Yes	Not managed	2 Yageo	RC0805JR-07430R	Volume Production 311-430ARCT-ND	0.1	0.2 Clear
428.351	Connector JST 1.25 mm	Digi-Key		No	Not managed	0				Fatal error
270R	Resistor	Digi-Key	R3	Yes	Not managed	1 Yageo	RC0805JR-07270R	Volume Production 311-270ARCT-ND	0.1	0.1 Clear
261 2mm JST female connector 2P		Mouser		No	Not managed	1 Adafruit Industries		261 Unknown 485-261	0.75	0.75 Clear
32 219-3MSTR	SWITCH SLIDE DIP SPST 100MA 20V	Digi-Key	SW1	No	CMP-07018-000004-1 New From Up to date	1 CTS	219-3MSTR	Volume Production CT0949CT-ND	0.69	0.69 Clear
82R	Resistor	Digi-Key	R6	Yes	Not managed	1 Yageo	RC0805JR-0782RL	Volume Production 311-82ARCT-ND	0.1	0.1 Clear
33 25V	Capacitor	Digi-Key	C1	Yes	Not managed	1 Samsung	C21A106KAYNNNE	Volume Production 1276-2891-1-ND	0.18	0.18 Warning
34 25V	Capacitor	Mouser	C2	Yes	Not managed	1 Samsung	C21B104KBCNNNC	Volume Production 187-C21B104KBCNNNC	0.1	0.1 Warning
35 10V	Capacitor	Digi-Key	C3, C13, C16	Yes	Not managed	3 Samsung	C21A106KPNNNNE	Volume Production 1276-1052-6-ND	0.11	0.33 Warning
36 10V	Capacitor	Digi-Key	C4, C5	No	Not managed	2 Kyocera AVX	0805YC105KAT2A	Volume Production 478-5034-1-ND	0.18	0.36 Fatal error
37 10V	Capacitor	Digi-Key	C7, C9, C11, C12, C17, C18	Yes	Not managed	6 Samsung	C21B104KBCNNNC	Volume Production 1276-1003-1-ND	0.044	0.44 Warning
38 10V	Capacitor	Digi-Key	C8, C19	Yes	Not managed	2 Samsung	C21B103KBAANNNC	Volume Production 1276-1015-1-ND	0.1	0.2 Clear
39 10V	Capacitor	Digi-Key	C10	No	Not managed	1 Samsung	C21B252KAFFNNE	Volume Production 1276-2953-1-ND	0.2	0.2 Clear
40 10V	Capacitor	Digi-Key	C6	No	Not managed	1 Samsung	C21B105KAFNNNE	Volume Production 1276-1066-1-ND	0.1	0.1 Warning
10k	Resistor		R11, R12	Yes	Not managed	2				Fatal error
10k	Resistor	Mouser	R1, R5, R8, R13	Yes	Not managed	4 Yageo	RC0805JR-0710KL	Volume Production 603-RC0805JR-0710KL	0.021	0.21 Clear
5k1	Resistor	Digi-Key	R9, R10, R17, R18	Yes	Not managed	4 Yageo	RC0805JR-075K1L	Volume Production 311-5.1KARCT-ND	0.022	0.22 Clear
3k	Resistor	Digi-Key	R15	Yes	Not managed	1 Yageo	RC0805FR-103KL	Volume Production 13-RC0805FR-103KLCT-ND	0.1	0.1 Clear
2k	Resistor	Digi-Key	R2	Yes	Not managed	1 Stackpole Electronics	RNCP0805FTD2K00	Volume Production RNCP0805FTD2K00CT-ND	0.1	0.1 Clear
1k	Resistor	Digi-Key	R14	Yes	Not managed	1 Stackpole Electronics	RNCP0805FTD1K00	Volume Production RNCP0805FTD1K00CT-ND	0.1	0.1 Clear
0R	Resistor	Digi-Key	R16	Yes	Not managed	1 Stackpole Electronics	RMCF0805ZTRO0	Volume Production RMCF0805ZTRO0CT-ND	0.1	0.1 Clear

# Projet ETML-ES – Modification

*Note: Les textes explicatifs en italique peuvent être supprimés*

*A remplir par l'initiateur*

<b>PROJET:</b>	2230 Tube Pitot déporté		
<b>Entreprise/Client:</b>	M. Vincent Seguin (pour AMPA)	<b>Département:</b>	SLO
<b>Demandé par (Prénom, Nom):</b>	M. Juan José Moreno	<b>Date:</b>	1
<b>Objet (No ou réf, pièce, PCB...)</b>	2230		
<b>Version à modifier:</b>	V1		

*A remplir par l'exécutant*

<b>Auteur (ETML-ES):</b>		<b>Filière:</b>	SLO
<b>Nouvelle version:</b>		<b>Date:</b>	16.06.2023

## 1 Description ou justification

*Décrire ici en quelques lignes les raisons des modifications, améliorations, justifications.*

*Pour entreprises, si confidentiel, svp, mentionner CONFIDENTIEL.*

## 2 Référence conception

*Indiquer ici le(s) dossier(s) de conception de référence et emplacement. (N/A pour entreprises)*

## 3 Détail des modifications

*Chaque rangée du tableau ci-dessous contient le détail d'une seule modification.*

*Exemples:*

- 1 / Changer tous les boîtiers de résistances 0805 en 0603      / OK    / JMO
- 2 / Remplacement U4 - TL074 par LM124                          / NOK    / SCA

#	Description	Fait	Approuvé
1	Changer footprint bouton reset SW2 (Altium)	NOK	
2	Intervertir PIN21 et 22 du MCU (Altium)	NOK	
3	Dessiner nouvelle turbine avec fixation plus fiable	NOK	
4			
5			
6			
7			
8			

## 4 Remarques

*Au besoin, indiquer ici des détails nécessaires à la compréhension, ainsi que les raisons d'une modification non effectuée ou reportée.*

*Exemple: Le point 2 (marqué NOK), est reporté pour une prochaine version pour épuiser notre stock de composants. Cette modif n'est pas critique fonctionnellement.*

## 5 Convention de nommage et liens

Le nom de ce fichier doit être unique et doit donc contenir le numéro du projet et un numéro consécutif de modification avec le format suivant :

***aaii\_MOD\_nn.docx***

ou

***NomProjet\_MOD\_nn.docx***

avec :

- MOD : pour modification
- aaii : numéro de projet, exemple 1708 pour projet de 2017 no 08
- NomProjet : Si le projet n'est pas numéroté ou mandat de client.
- nn : numéro de modification. La première est 01

Exemples :

- **1708\_MOD\_01.docx** 1ere modification pour le projet 1708
- **1708\_MOD\_02.docx** 2e modification pour le projet 1708
- **CapteurVolets\_MOD\_01.docx** Cas de projet externe

Le schéma et/ou les documents de production de la pièce ou du PCB se référeront à ce document dans les cartouches.

Si un nouveau projet reprend un design d'un autre projet, créer un document de **modification numéro 00**. Ainsi, on pourra décrire les modifications initiales dans le fichier.

Exemple :

- **1803\_MOD\_00.docx** Modification initiale pour le nouveau projet 1803 à partir d'un autre projet (par ex. 1708)

### 5.1 Stockage du fichier

Ce fichier sera stocké à la racine du dossier **/doc** d'un projet.

Ainsi, tous les fichiers de modifications des pièces ou PCBs faisant partie du projet sont centralisés dans le même répertoire. La numérotation devient implicite.

Jour	Date	Nbr période	Tâches effectuées
Mercredi	16.11.2022	8	- (Présentation des projets et choix)
Jeudi	17.11.2022	4	Rédaction du cahier des charges (but du projet, schéma de principe, schéma bloc)
Mercredi	23.11.2022	8	Rédaction du cahier des charges (description simples des blocs,
Jeudi	24.11.2022	4	Rédaction du cahier des charges (croquis mécanique, tâches à réaliser, deadlines principales, planning, livrables)
Samedi	26.11.2022	1	Rencontre de M. Vincent Seguin à l'aéroport de Lausanne pour plus de détail
Mercredi	30.11.2022	8	Finitions du cahier des charges (corrections et amélioration de certains points), rédaction de la pré-étude (contexte, schéma de principe, schéma bloc détaillé)
Samedi	03.11.2022	5	Rédaction de la pré-étude (description des blocs détaillées)
Mercredi	07.12.2022	8	Rédaction de la pré-étude (description des blocs détaillées)
Mercredi	14.12.2022	8	Présentation de la pré-étude, début du design électronique, corrections de certains points de la pré-étude
Mercredi	21.12.2022	8	Recherche d'information sur la génération d'électricité avec un moteur DC
Lundi	26.12.2022	4	Caractérisation du moteur DC RF-370A-15370
Mercredi	11.01.2023	8	Création du projet Altium et début du schéma électrique avec dimensionnement des composants
Mercredi	18.01.2023	8	Dimensionnement et choix de composants et design du schéma électriques
Dimanche	22.01.2023	4	Rédaction du rapport et design du schéma électrique
Mardi	24.01.2023	4	Rédaction du rapport et corrections du schéma électrique
Mercredi	25.01.2023	10	Rédaction du rapport et corrections du schéma électrique
Jeudi	26.01.2023	4	Rédaction du rapport et corrections du schéma électrique
Mercredi	01.02.2023	8	Correction du rapport, création de la présentation, présentation du projet
Mercredi	08.02.2023	8	Correction des points évoqués lors de la présentation, création de footprints
Jeudi	16.02.2023	2	Dessin d'une ébauche du boîtier 3D
Mercredi	22.02.2023	8	Création de footprints, modification du schéma et début du placement des composants
Mercredi	01.03.2023	6	Recherche de solutions mécaniques au niveau du boîtier et du PCB et placement des composants
Mardi	07.03.2023	2	Placement des composants
Mercredi	08.03.2023	8	Placement des composants et routage du PCB
Jeudi	09.03.2023	2	Routage du PCB
Samedi	11.03.2023	4	Routage du PCB
Mercredi	15.03.2023	8	Routage du PCB, contrôle du PCB par Ali Zoubir et modification de la BOM
Dimanche	19.03.2023	4	Design mécanique du boîtier
Mercredi	22.03.2023	8	Recherche des composants non critiques (inserts, LED 5mm, ..) et modification de la BOM
Mercredi	29.03.2023	8	Finition du design mécanique du boîtier
Mercredi	05.03.2023	8	Montage et test du PCB, configuration du MCU sur Harmony et programmation de test du MCU
Mardi	11.04.2023	8	Programmation de test du capteur de pressions différentielles et du module bluetooth
Mercredi	12.04.2023	5	Programmation du driver du capteur de pressions différentielles et du module bluetooth
Mercredi	26.04.2023	8	Mesure et test du reset en mode turbine, amélioration du système de reset
Mercredi	03.05.2023	8	Compréhension, test et programmation du driver de l'IMU
Mercredi	10.05.2023	11	Debugging du driver de l'IMU et debugging des permissions Android
Mercredi	17.05.2023	10	Debugging du driver de l'IMU
Mardi	23.05.2023	3	Mise en place de l'USART en FIFO
Mercredi	24.05.2023	10	Mise en place de l'USART en FIFO et modification du driver du module Bluetooth
Mercredi	31.05.2023	8	Correction de bugs et amélioration du code
Jeudi	01.06.2023	2	Correction de bugs et amélioration du code
Mercredi	07.06.2023	8	Correction de bugs et amélioration du code, recherche de documentation sur le développement d'application Android avec Bluetooth
Jeudi	08.06.2023	3	Développement de l'application Android
Mercredi	14.06.2023	8	Développement de l'application Android
Jeudi	15.06.2023	6	Finalisation du rapport et de la documentation
Vendredi	16.06.2023	2	Finalisation du rapport, de la documentation et rendu

Total 288

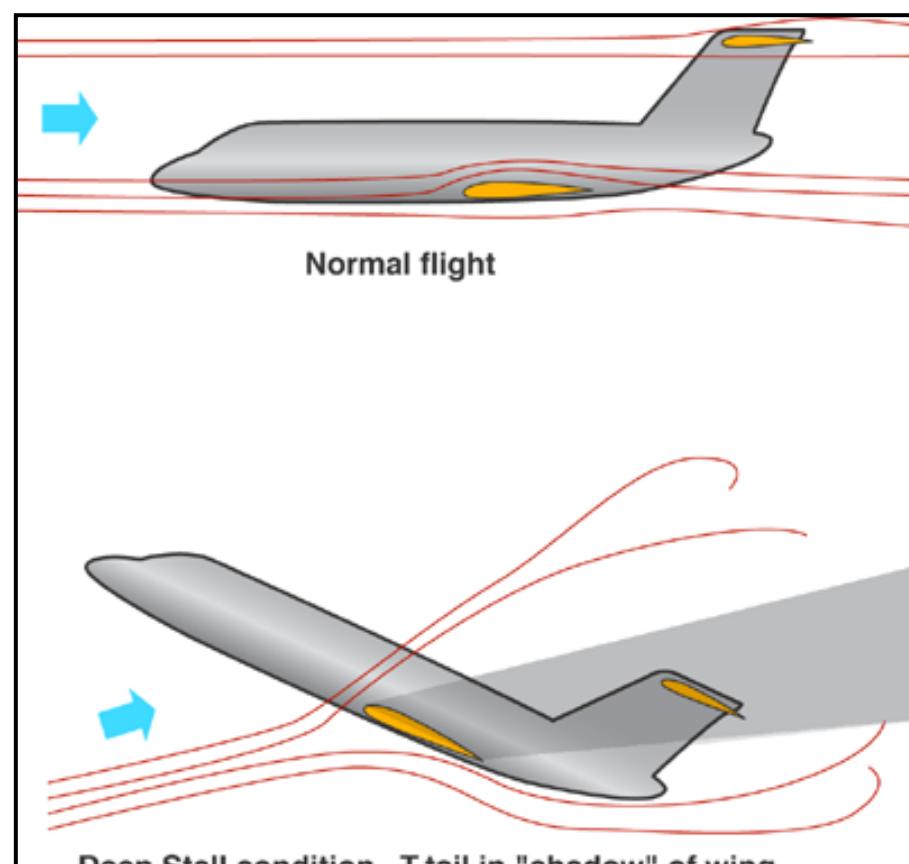
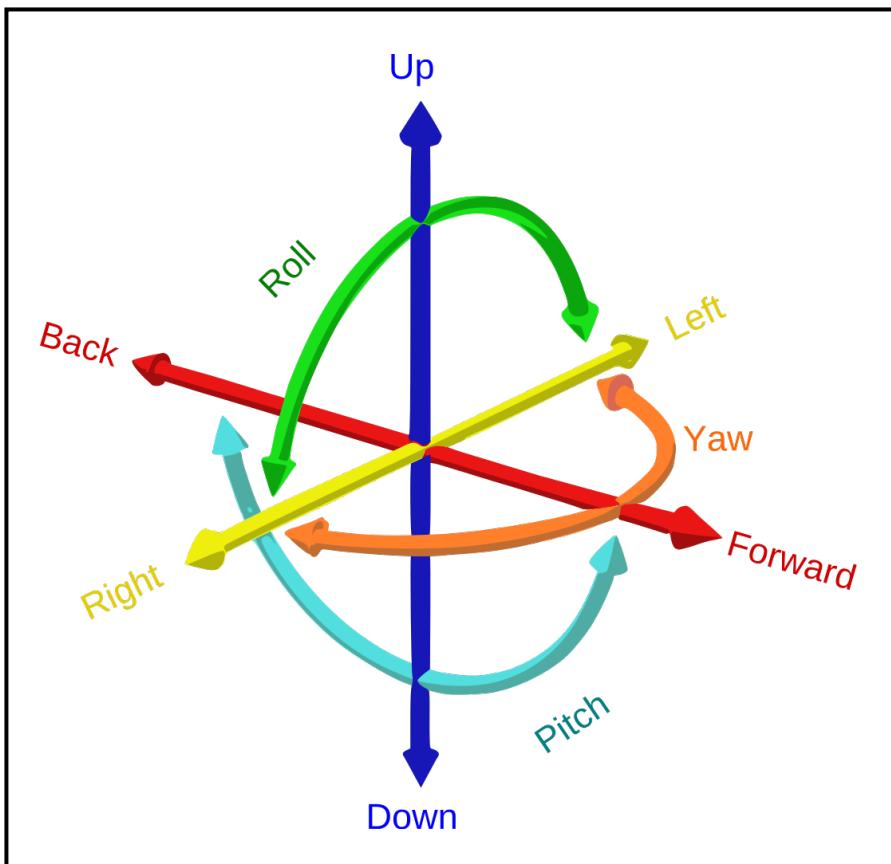
No semaine de travail	1	2	3	4	5	6	V	V	7	8	9	10	11	V	12	13	14	15	16	17	V	V	19	20	21	22	23	24	25	26	27	28		
No semaine	46	47	48	49	50	51	52	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
Date	16.11.2022	23.11.2022	30.11.2022	07.12.2022	14.12.2022	21.12.2022	28.12.2022	04.01.2023	11.01.2023	18.01.2023	25.01.2023	01.02.2023	08.02.2023	15.02.2023	22.02.2023	01.03.2023	08.03.2023	15.03.2023	22.03.2023	29.03.2023	05.04.2023	12.04.2023	19.04.2023	26.04.2023	03.05.2023	10.05.2023	17.05.2023	24.05.2023	31.05.2023	07.06.2023	14.06.2023	21.06.2023	28.06.2023	
Cahier des charges	O	O	N																															
Pré-étude	O	O	O	O	O																													
Dimensionnement + design + schéma			N	N	O																													
Design du PCB																																		
Design des parties mécaniques																																		
Montage PCB																																		
Impression ou usinage mécanique																																		
Réalisation des softwares																																		
Mise en service et tests																																		
Préparation présentation																																		
Présentation finale																																		
Rédaction rapport	O	O	O	O	O	O				O	O	O	N		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N		
Finalisation/corrections/documentation																																		

Vacances	O	O	Plannification correcte
Remise de documents	N	N	Plannification incorrecte
Présentation			

## Mode d'emplois :

1. Branchez le connecteur de la batterie.
2. Vissez le support pile au boitier principale.
3. Allumez le système à l'aide du switch.
4. Apparez le système à l'appareil Android.
5. Lancez l'application dédiée.

Le but du projet consiste à développer un système permettant de détecter l'angle d'incidence et la vitesse au décrochage d'un avion. La fixation de ce système doit être flexible afin de pouvoir l'installer sur différents types d'avions. L'emplacement de fixation ne doit pas se trouver dans le flux d'air provenant de l'hélice afin d'éviter que la mesure de vitesse ne soit faussée. Il doit également être miniaturisé au maximum afin de produire le minimum de traînée possible et de ne pas dépasser un poids de 500g. Les données acquises par les capteurs doivent être transmises de la partie déportée à un appareil Android se trouvant dans le cockpit de l'avion au travers une communication sans fil. L'appareil Android doit traiter et afficher les données reçues, si possible graphiquement (optionnel).



La réalisation de ce projet a été une expérience inestimable qui m'a permis d'élargir mes compétences en travaillant avec des composants variés que je n'avais jamais manipulés auparavant. Cette opportunité a renforcé ma détermination et mon dévouement envers le projet, car il incarnait une occasion unique de contribuer à un domaine qui suscite en moi une profonde inspiration et fascination : l'aéronautique. Travailler sur ce projet a été une expérience incroyablement gratifiante et motivante, comblant mes aspirations en tant qu'enthousiaste passionné d'aéronautique.