

Travail de diplôme

École supérieure
Électronique

Salle R112

Numéro de projet : 2015

Surveillance de paramètres moteur en vol (avion)

Réalisé par :

Dimitrijevic Nikola

À l'attention de :

Juan José Moreno
Thierry Berney
Association AMPA

Date :

Début du travail de diplôme : 25 aout 2020
Fin du travail de diplôme : 29 septembre 2020

Table des matières :

1 But du projet.....	6
2 Cahier des charges	6
3 Disposition des circuits.....	6
4 Avant-projet :.....	7
4.1 PCB_Capteurs_Berney	7
4.2 PCB_TrameRS485.....	8
5 Schéma bloc.....	9
5.1 PCB_TrameRS485.....	9
5.1.1 Explication des blocs :	9
5.2 PCB_Bluetooth.....	10
5.2.1 Explication des blocs :	10
6 Choix technologiques	11
6.1 PCB_Capteurs_AMPA	11
6.1.1 Connecteurs.....	11
6.2 PCB_Capteurs_Berney	13
6.2.1 Connecteurs.....	13
6.3 Boitier (pour M.Berney et l'AMPA).....	15
6.4 Connexion des cartes PCB_Capteurs et PCB_TrameRS485	17
6.5 PCB_TrameRS485.....	18
6.5.1 Corrections à faire	18
6.5.1.1 L'alimentation à découpage 12V-5V.....	18
6.5.1.2 Interface analogique - Protection des entrées ADC contre surtension.....	20
6.5.2 Travaux restants	22
6.5.2.1 RPM de l'alternateur	22
6.5.3 Capteur de débit d'essence	26
6.5.3.1 Récupération de la tension de la batterie	29
6.5.3.2 RPM_projet2007	30
6.5.3.3 Trame RS485	30
6.5.4 Microcontrôleur	32
6.6 PCB_Bluetooth.....	34
6.6.1 Alimentation	34
6.6.2 RS485 to UART.....	35
6.6.3 UART to Bluetooth.....	35
6.6.4 Microcontrôleur	37
6.6.5 Boitier	38
7 Schéma électrique.....	39
7.1 PCB_Capteurs_AMPA	39
7.2 PCB_Capteurs_Berney	39
7.3 PCB_TrameRS485.....	39
7.4 PCB_Bluetooth.....	39
8 Routage	40
8.1 Paramètres généraux.....	40
8.1.1 Taille des pistes	40
8.1.2 Nombre de couches.....	40
8.2 PCB_TrameRS485.....	41

8.2.1 Forme du circuit :	41
8.2.2 Contraintes mécaniques.....	44
8.3 PCB_Capteurs_Berney	45
8.3.1 Contraintes mécaniques.....	45
8.4 PCB_Capteurs_AMPA	46
8.4.1 Contraintes mécaniques.....	46
8.5 PCB_Bluetooth	47
8.5.1 Contraintes mécaniques.....	48
9 Montage	49
9.1 PCB_TrameRS485.....	49
9.2 PCB_Bluetooth.....	52
10 Software	55
10.1 PCB_trameRS485	56
10.1.1 Création de projet	56
10.1.2 Configurations dans l'Harmony	57
10.1.2.1 Configuration du clock	57
10.1.2.2 Configuration des timers.....	58
10.1.2.3 Configuration des ADC	60
10.1.2.4 Configuration du SPI	62
10.1.2.5 Configuration de l'UART.....	64
10.1.2.6 Configuration de l'input capture.....	65
10.1.3 Machine d'état	66
10.1.4 Réception de la trame UART	67
10.1.4.1 Calcul des RPM de l'alternateur.....	71
10.1.5 Capteur de débit d'essence	76
10.1.6 Envoi de la trame UART.....	80
10.2 Module Bluetooth.....	88
10.3 PCB_bluetooth	93
10.3.1 Création du projet	93
10.3.1.1 Configurations dans l'Harmony.....	93
10.3.1.2 Configuration de l'UART	94
10.3.1.3 Machine d'état	95
10.3.1.4 Initialisation du module Bluetooth	96
11 Liste de matériel utilisé	100
12 Problèmes rencontrés	100
13 Améliorations à faire	100
14 Évaluation du projet.....	101
14.1 État d'avancement du projet	101
14.1.1 PCB_Capteurs_Berney.....	101
14.1.2 PCB_Capteurs_AMPA	101
14.1.3 PCB_TrameRS485	101
14.1.4 PCB_Bluetooth	101
15 Conclusion	102
16 Remerciements	102
17 Annexes	103
17.1 Annexe 1 : Planification	103
17.2 Annexe 2 : Journal de travail	103
17.3 Annexe 3 : Liste de personnes de contact	103

17.4 Annexe 4 : Schéma électrique - PCB_Capteurs_Berney	103
17.5 Annexe 5 : Schéma électrique - PCB_Capteurs_AMPA	103
17.6 Annexe 6 : Schéma électrique - PCB_Capteurs_TrameRS485	103
17.7 Annexe 7 : Schéma électrique - PCB_Capteurs_Bluetooth	103
17.8 Annexe 8 : Circuit - PCB_Capteurs_AMPA	103
17.9 Annexe 9 : Circuit - PCB_Capteurs_Berney	103
17.10 Annexe 10 : Circuit - PCB_Capteurs_TrameRS485	103
17.11 Annexe 11 : Circuit - PCB_Bluetooth	103
17.12 Annexe 12 : Plans de perçage	103
17.13 Annexe 13 : Mode d'emploi.....	103
17.14 Annexe 14 : Modifications - PCB_TrameRS485	103
17.15 Annexe 15 : Modifications - PCB_Bluetooth	103
17.16 Annexe 16 : Code PCB_TrameRS485	103
17.17 Annexe 17 : Code PCB_Bluetooth.....	103

1 But du projet

Le but de ce projet est de réaliser un petit module qui récupère des données de mesures d'un moteur d'avion par une ligne RS-485 et les transmet sur une liaison Bluetooth.

Ces mesures sont par exemple, températures de cylindres, températures des gaz d'échappement, pression d'admission, d'huile, d'essence, RPM, ...

Elles pourront ainsi être affichables sur une tablette dans le cadre d'un autre projet. Il s'agit d'une extension du projet 1923 (projet de semestre – paramètres moteur pour ULM).

En outre, des compléments et corrections sur les 2 PCBs, MCU et interface capteurs de ce projet 1923 seront réalisées, ainsi que l'émission des trames des paramètres mesurés.

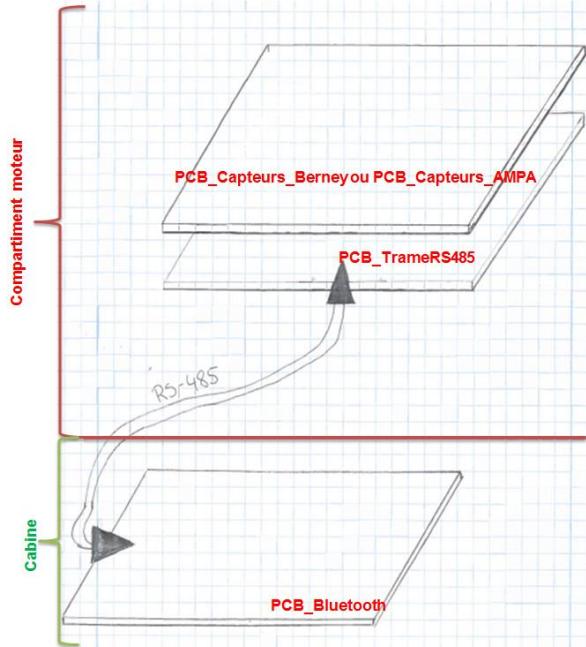
Ce projet intéresse deux clients :

- M.Berney Thierry
- M.Schubert Edouard -Association AMPA

2 Cahier des charges

Veuillez consulter l'Annexe 3 afin de voir le cahier des charges complet.

3 Disposition des circuits



- PCB_Capteurs : sur ce circuit, les différents capteurs seront connectés. Il y aura un PCB_Capteurs pour chaque client parce que les différents connecteurs seront utilisés.
- PCB_TrameRS485 : avec ce circuit, nous allons mettre en forme les signaux des capteurs avant de les connecter au microcontrôleur. Avec ce dernier, nous allons lire la valeur des capteurs et envoyer la trame RS485. Pour qu'on puisse utiliser le même circuit pour les deux clients, il doit être compatible avec les deux PCB_Capteurs.
- PCB_Bluetooth : Comme le compartiment moteur est un endroit avec beaucoup de perturbations, nous avons décidé de rajouter ce circuit dans la cabine, sur lequel nous allons mettre le module Bluetooth.

4 Avant-projet :

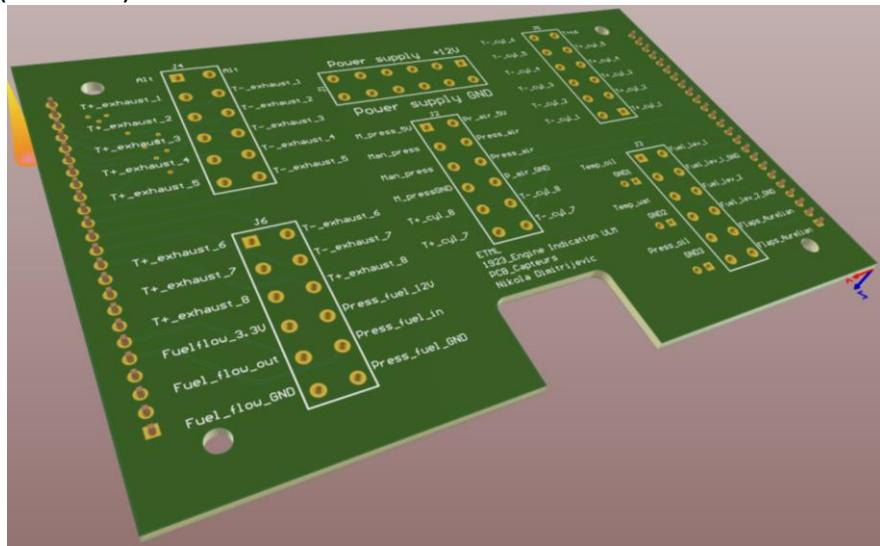
Ce travail de diplôme est donc la suite du projet de semestre : 1923_EngineIndicationUlm.

Dans ce chapitre, je vais expliquer brièvement ce qui a été fait dans ce projet ainsi les choses à corriger :

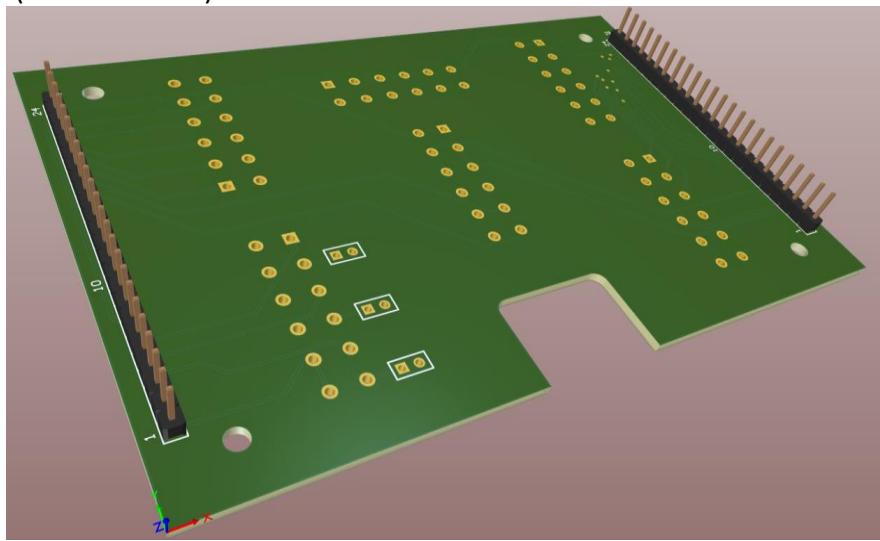
4.1 PCB_Capteurs_Berney

Pendant le travail de semestre, le circuit "PCB_Capteurs" pour M.Berney a été réalisé :

Vue de dessus (face TOP)



Vue de dessous (face BOTTOM)



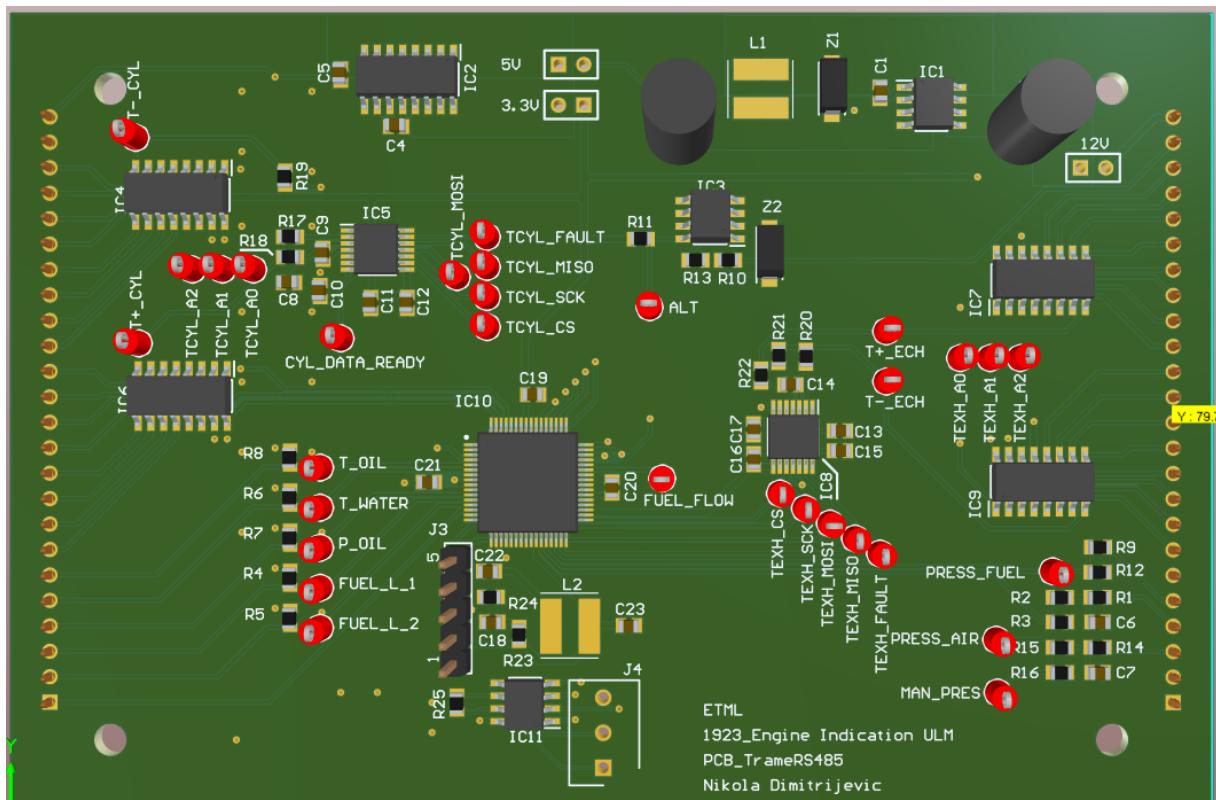
À corriger :

- Sur ce circuit, il faut corriger le footprint des connecteurs.

Reste à faire :

- Rajouter la possibilité de connecter 8 entrées de capteurs supplémentaires pour pouvoir mesurer la température de plusieurs cylindres.
- Faire un nouveau circuit PCB_Capteurs pour l'AMPA pour pouvoir connecter les capteurs sur un connecteur D-SUB.

4.2 PCB_TrameRS485



À corriger :

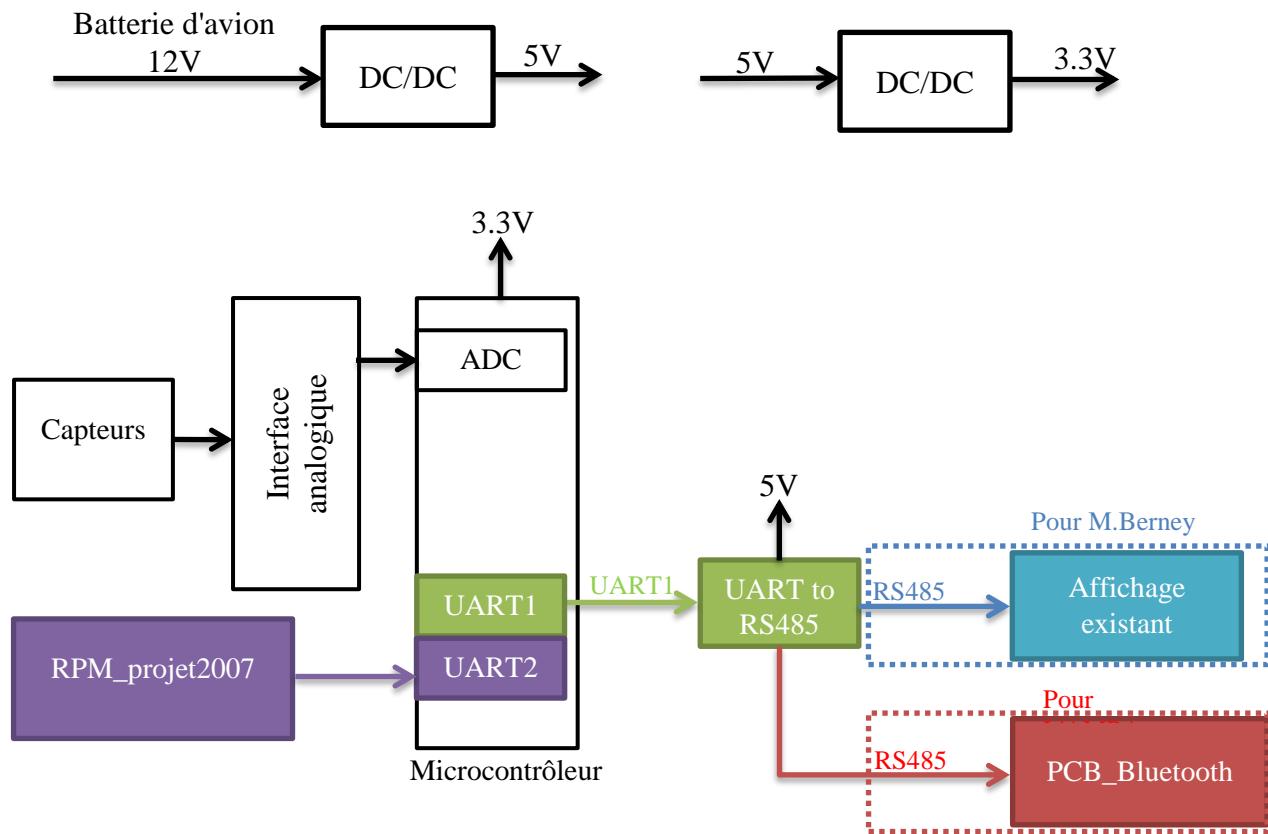
- L'alimentation à découpage 12V-5V.
- Mettre en place une interface analogique afin de protéger les entrées du microcontrôleur lorsque le capteur n'est pas connecté.

Reste à faire :

- Lire le capteur de débit d'essence.
- Faire le code pour envoyer la trame RS485.
- Mettre en place une interface analogique pour récupérer les RPM de l'alternateur.
- Rajouter 8 entrées de capteurs supplémentaires (thermocouples) pour pouvoir mesurer la température de plusieurs cylindres.

5 Schéma bloc

5.1 PCB_TrameRS485



5.1.1 Explication des blocs :

Capteurs

Voici la liste de différentes choses qui seront connectées :

Capteur en question	Nombre de capteurs à connecter
Capteur de température de tête de cylindre	8
Capteur de température des gaz d'échappement	8
8 entrées des thermocouples supplémentaires	8
Capteur de température de l'huile	1
Capteur de température de l'eau	1
Capteur de pression de l'huile	1
Capteur de pression d'essence	1
Capteur de pression d'admission	1
Capteur de pression d'air	1
Capteur de niveau d'essence	2
Mesure des RPM sur l'alternateur	1
Récupération des RPM du projet 2007	1
Débit d'essence	1
Position des flaps	1

Interface analogique

Le bloc "interface analogique" sert à modifier les signaux analogiques des capteurs, avant de les connecter sur le microcontrôleur.

Microcontrôleur

Nous avons besoin d'un microcontrôleur pour convertir en numérique les valeurs analogiques reçues par les différents capteurs, pour lire la TRAME UART qui contient les RPM (projet 2007) et pour envoyer la trame RS485 avec la valeur de différents capteurs.

UART to RS485

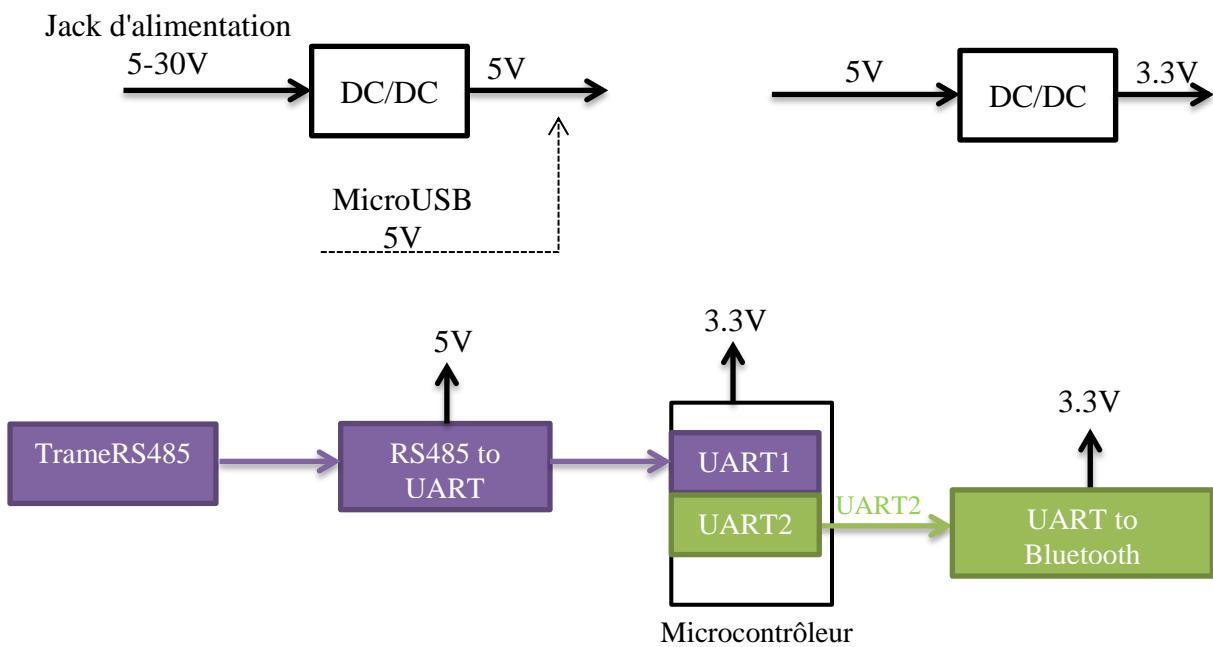
Le client M.Berney nous a imposé le format de la trame RS485 afin de pouvoir piloter un affichage existant.

L'exemple de la trame :

START::OILT=250::WAT=87::CHT1=365::CHT2=384...

Afin de pouvoir utiliser le même circuit PCB_TrameRS485 pour les deux clients, la même trame sera envoyée au circuit PCB_Bluetooth.

5.2 PCB_Bluetooth



5.2.1 Explication des blocs :

Alimentation :

Selon le cahier des charges, il faut prévoir la possibilité d'alimenter le circuit avec une tension de 5-30V à l'aide d'un jack d'alimentation ou par micro USB.

RS485 to UART

Pour pouvoir lire la trame RS485 avec le microcontrôleur, je vais utiliser un module qui va la convertir en UART.

Microcontrôleur

Nous avons besoin du microcontrôleur pour recevoir la trame UART avec la valeur des capteurs. Une fois les données lues, une autre trame UART sera envoyée au module Bluetooth.

UART to Bluetooth

Je vais utiliser un module qui va recevoir les données en UART et qui va les envoyer en Bluetooth.

6 Choix technologiques

6.1 PCB_Capteurs_AMPA

6.1.1 Connecteurs

En discutant avec M. Schubert Edouard, l'association AMPA aimeraient utiliser des capteurs D-SUB pour connecter les différents capteurs sur l'avion.

J'ai fait le tableau suivant afin de voir le nombre d'entrées nécessaires pour connecter tous les capteurs :

Capteur en question	Nombre de capteurs à connecter	Nombre de connexions par capteur	Nombre de connexions tot.
Température de tête de cylindre	8	2	16
Température des gaz d'échappement	8	2	16
Entrées des thermocouples supplémentaires	8	2	16
Capteur de température de l'huile	1	1	1
Capteur de température de l'eau	1	1	1
Capteur de pression de l'huile	1	2	2
Capteur de pression d'essence	1	3	3
Capteur de pression d'admission	1	3	3
Capteur de pression d'air	1	3	3
Capteur de niveau d'essence	2	1	2
RPM de l'alternateur	1	2	2
RPM du projet 2007	1	3	3
Capteur de débit d'essence	1	3	3
Position des flaps	1	1	1
Alimentation	1	2	2
Total			72

Comme il y a beaucoup de connexions, je vais utiliser deux connecteurs D-SUB afin de pouvoir connecter tous les capteurs. Je vais donc choisir le connecteur D-SUB avec 37 positions.

Principe de connexion :

Comme le compartiment moteur est un endroit très bruité et que nous devons assurer une bonne connexion, je vais utiliser le principe de verrouillage "SnapLock". Ce système est représenté sur les images ci-dessous :



Sur un des connecteurs (mâle ou femelle), nous allons fixer les vises et par la suite, venir se clipser avec le deuxième connecteur (femelle ou mâle).

Références des composants choisis :

- Connecteur D-SUB mâle 37 positions
Fabricant : TE Connectivity AMP Connectors
Fournisseur : Digi - Key

Référence fabricant : 5747835-4
Référence fournisseur : A32082-ND



- Connecteur D-SUB femelle 37 positions
Fabricant: NorComp Inc.
Fournisseur : Digi - Key

Référence fabricant: 171-037-203L001
Référence fournisseur : 237FE-ND



- Kit de verrouillage
Fabricant: Conec
Fournisseur : Digi - Key

Référence fabricant: 16-002190E
Référence fournisseur: 626-1724-ND



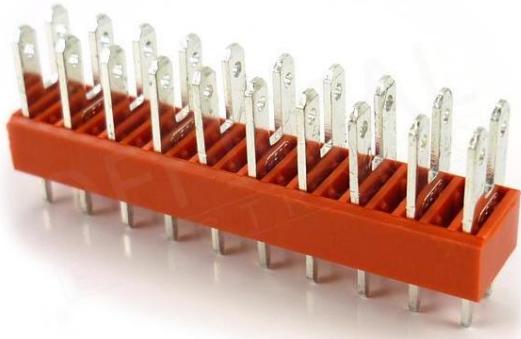
- Capuchon pour le connecteur D-SUB 37 avec le système de verrouillage (SnapLock)
Fabricant: Conec
Fournisseur : Digi - Key
- Référence fabricant: 16-001840E
Référence fournisseur : 626-1721-ND



6.2 PCB_Capteurs_Berney

6.2.1 Connecteurs

Le client nous a imposé et fourni le modèle suivant :



Fabricant : Weco
Référence fabricant : 016002263

Ce connecteur a été commandé sur le site suivant :

<https://www.official.cz/z16454-konektor-do-dps-weco-900-sh-5-10?lang=en>

Voici les cosses utilisées pour ce type de connecteurs :



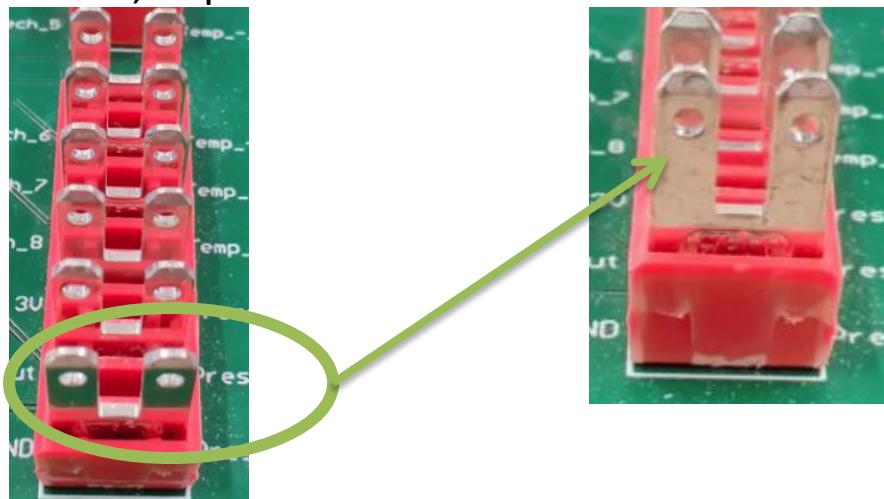
Fabricant : TME
Référence fabricant : 1361.68

Fournisseur : Conrad
Référence fournisseur : 526178-TX

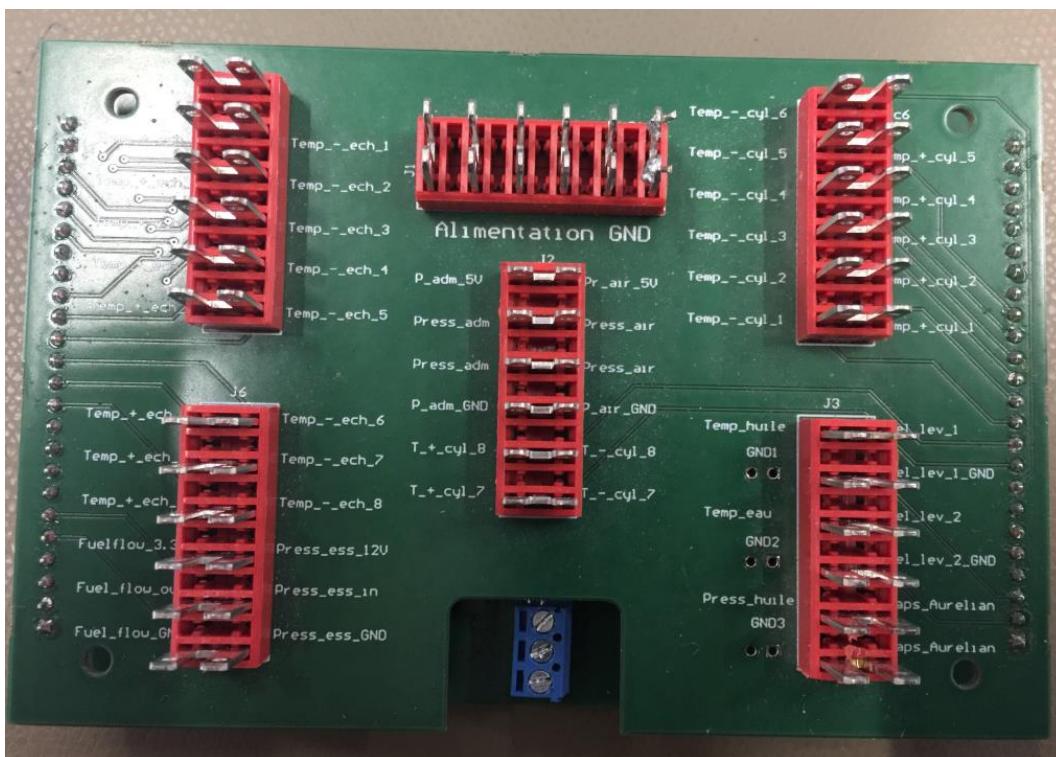
Comme pour PCB_Capteurs_AMPA, je vais prévoir 72 connexions pour connecter tous les capteurs (voir la page 10 pour plus de détails).

Sur un connecteur nous pouvons effectuer 6 connexions, il faudra donc placer au moins 12 connecteurs.

Attention, chaque connexion est doublée:



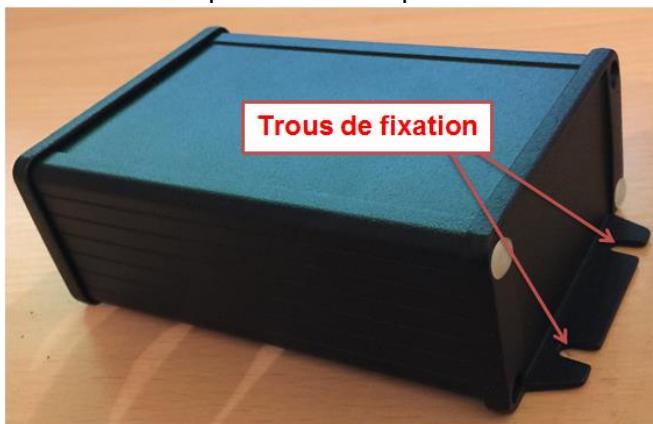
Le circuit PCB_Capteurs réalisé pendant le travail de semestre mesure 120x80 :



Sur ce circuit, 6 connecteurs ont été montés. Afin de pouvoir en mettre 12 et pour qu'on puisse lire la sérigraphie, j'ai décidé de faire un circuit plus grand donc je vais choisir un boîtier plus grand.

6.3 Boitier (pour M.Berney et l'AMPA)

Voici le boîtier qui a été choisi pendant le travail de semestre :



Fabricant: Hammond Manufacturing
Référence fabricant: 1457K1202BK

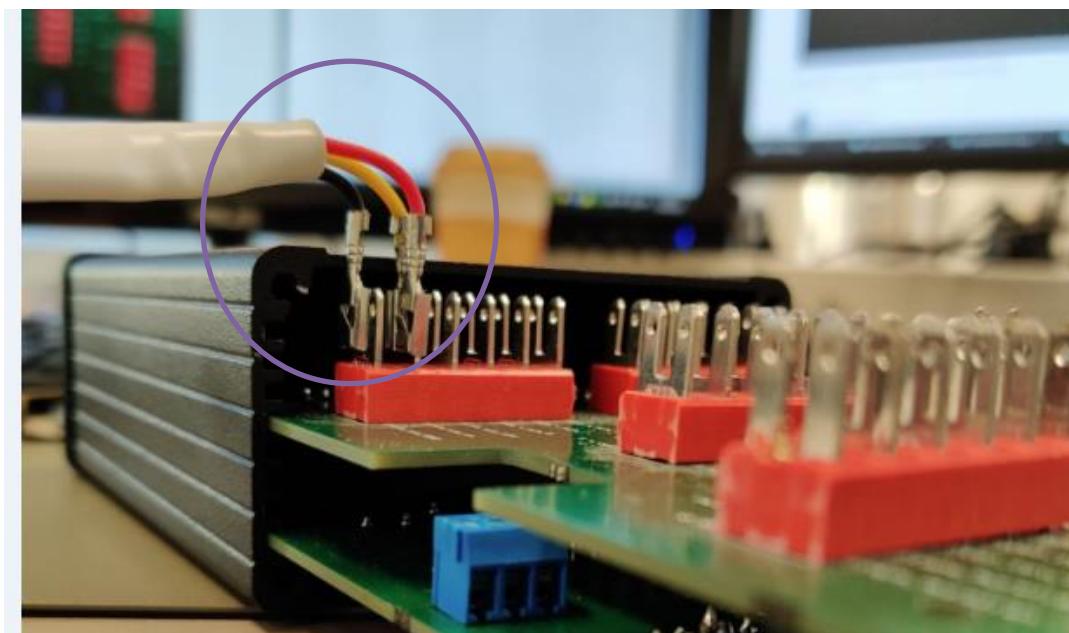
Fournisseur : Digi-Key
Référence fournisseur: HM1023-ND



Taille du boîtier: 120x84x44 [mm]

Problème du boîtier existant :

Une fois les cosses mises sur les fils de connexion, il est impossible de glisser les circuits dans le boîtier :



Je vais donc chercher un boîtier plus grand (pour pouvoir mettre au moins 12 connecteurs sur le circuit PCB_Capteurs_Berney) et également plus haut pour pouvoir glisser les circuits une fois les connecteurs câblés.

Chez le fabricant de boitiers "Hammond Manufacturing", j'ai cherché le même type de boîtier, mais avec les dimensions différentes.

J'ai trouvé donc le boîtier qui mesure 160x104x55 :



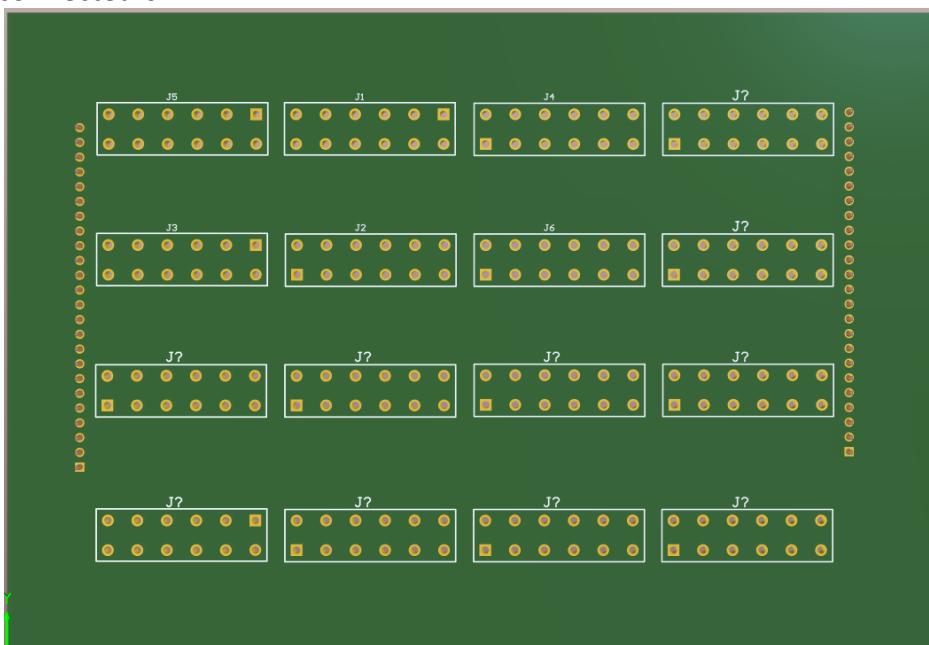
Fabricant: Hammond Manufacturing
Fournisseur : Mouser

Référence fabricant: 1457N1602BK
Référence fournisseur : 546-1457N1602BK

La taille maximale du PCB qu'on peut mettre dans le boîtier est : 160x100 [mm]

Vérification de la taille du boîtier :

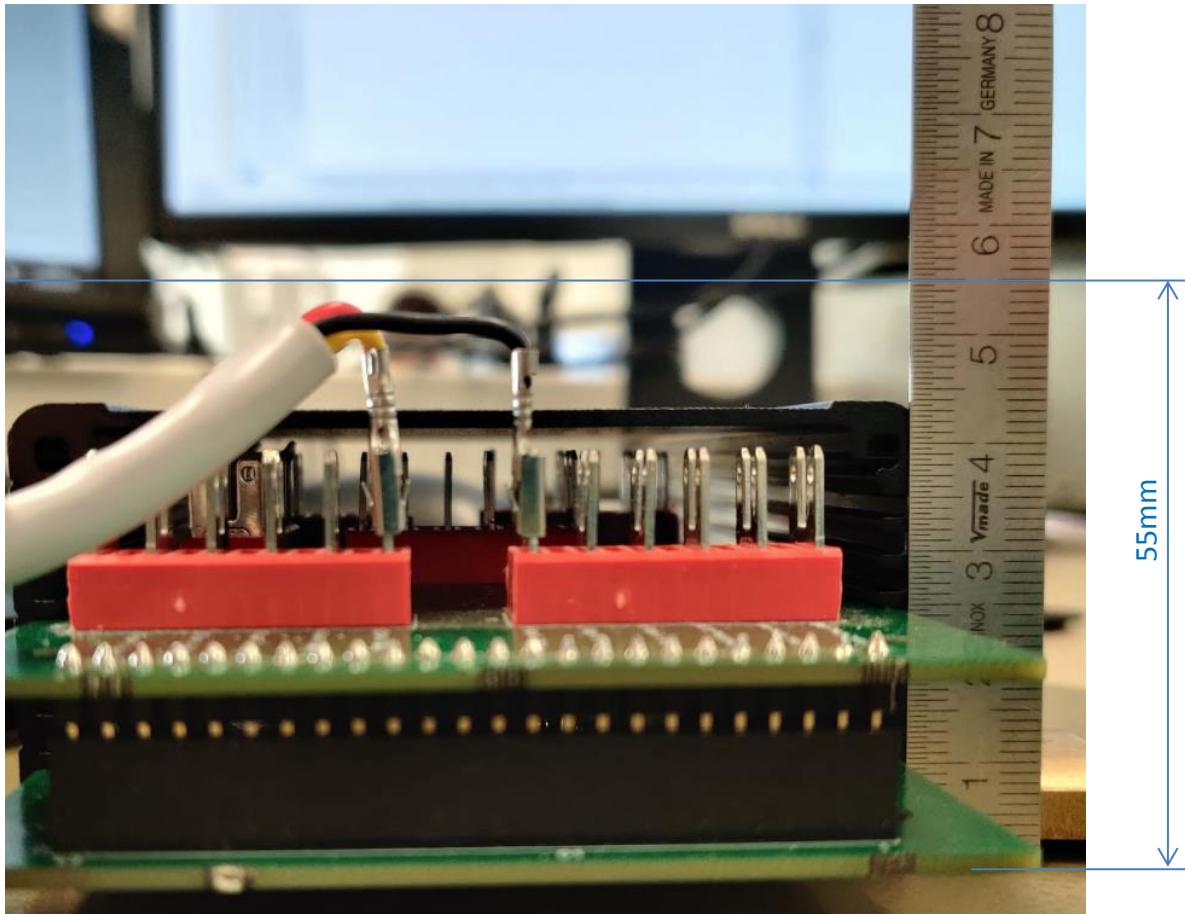
Afin de m'assurer que la taille du PCB (160x100) est suffisante pour mettre au moins 12 connecteurs que M.Berney nous a fournis, j'ai fait un circuit d'essai sur l'Altium et j'ai placé les connecteurs :



Nous pouvons voir qu'il y a assez de place pour mettre au moins 12 connecteurs et que la place est suffisante pour la sérigraphie.

Vérification de la hauteur du boitier :

Le boitier choisi mesure 55mm de haut. Afin de vérifier que cette hauteur est suffisante, j'ai mis les deux circuits réalisés pendant le travail de semestre (PCB_Capteurs_Berney et PCB_TrameRS485) et ensuite j'ai mis une règle à côté :



En traçant une ligne à une hauteur de 55mm, nous pouvons constater qu'il y a assez de place et que les circuits pourront être glissés une fois le câblage effectué.

6.4 Connexion des cartes PCB_Capteurs et PCB_TrameRS485

Pour connecter le circuit "PCB_Capteurs" avec le circuit "PCB_TrameRS485", j'ai décidé d'utiliser les barrettes mâles sur le circuit "PCB_Capteurs" et les barrettes femelles sur le circuit "PCB_TrameRS485", comme indiqué sur l'image ci-dessous :



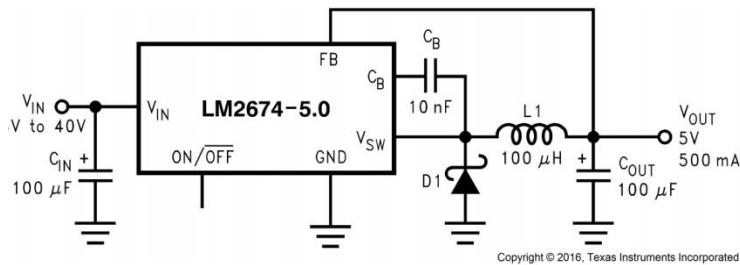
6.5 PCB_TrameRS485

6.5.1 Corrections à faire

6.5.1.1 L'alimentation à découpage 12V-5V

Comme alimentation à découpage, je vais utiliser **LM2674**. Je pars du principe que la consommation du circuit ne va pas dépasser 500mA.

En regardant le datasheet, nous pouvons trouver le schéma électrique proposé par le fabricant :



La tension d'entrée peut varier de 8V à 40V et le courant maximal de sortie est de 500mA.

Choix technologiques :

Comme le circuit sera placé dans le compartiment moteur, où il y a de grands changements de température, j'ai décidé de faire une étude afin de choisir la meilleure technologie des condensateurs à utiliser pour le projet.

1. Condensateurs au mica :

- Coefficient de température très faible.
- Gamme de capacité : quelque pF à quelque centaine de nF.
- Plage de tension : 300V à 2500V

2. Condensateurs céramiques :

- Plusieurs classes de condensateurs céramiques sont définies en fonction de leur tenue en température :
 - COG /NPO : grande stabilité en température. La valeur de capacité très faible (quelques nano-farad)
 - Les condensateurs céramiques (X7R) : variation de 10% lors d'un fonctionnement entre -10 et +60°C.
 - Y4T et Z5U : dérivée en température d'environ 50%.

3. Condensateurs en tantale :

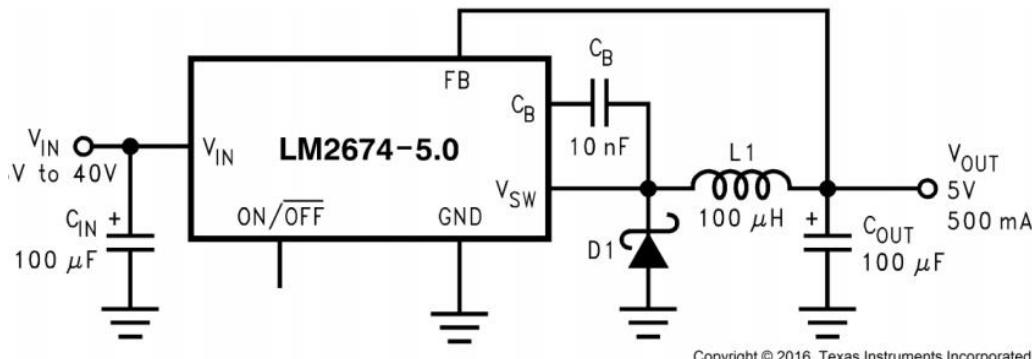
- Les caractéristiques des condensateurs en tantale sont beaucoup plus avantageuses que ceux en aluminium.
- Les condensateurs en tantale peuvent être utilisés à des températures maximales de 125°C alors que ceux en aluminium sont utilisables jusqu'à 85°C.
- Disponibilité en petites dimensions

Les informations ci-dessus ont été prises sur les sites suivants :

1. <http://yann.lelogeais.free.fr/science/condensateurs.html>
2. <https://fr.wikipedia.org/wiki/Condensateur>

Prise de décisions :

- Pour les condensateurs de grande capacité, je vais utiliser des condensateurs en tantale.
- Pour les condensateurs de capacité moyenne, je vais utiliser ses condensateurs céramiques (X7R).

Choix des composants de l'alimentation à découpage :**Cin et Cout:**

Fabricant : Vishay

Référence fabricant : 74-593D107X0020E2TE3

Fournisseur : Mouser

Référence fournisseur : 74-593D107X0020E2TE3

En regardant le datasheet, nous pouvons voir que ce condensateur peut supporter des températures élevées (jusqu'à 125°C) ce qui est largement suffisant pour notre utilisation :

TEMPERATURE RANGE	-55 °C to +125 °C
-------------------	-------------------

*Extrait du datasheet, page 3**Lien : <https://www.mouser.ch/datasheet/2/427/593d-1763502.pdf>***C_b :**

Fabricant : Taiyo Yuden

Référence fabricant : TMF212B7103MGHT

Fournisseur : Mouser

Référence fournisseur : 963-TMF212B7103MGHT

Ce condensateur peut travailler dans les mêmes plages de température que Cin :

B7	EIA	X7R	-55~+125	25	±15%	±10%	K
						±20%	M

*Extrait du datasheet, page 18**Lien : https://www.mouser.ch/datasheet/2/396/taiyo_yuden_07182019_MLCC_AECQ200_Grade_1_X7R_data-1622917.pdf***D1 :**

J'ai décidé de garder la même diode Schottky, comme utilisé pour l'Interface analogique (voir la page suivante pour plus de détails).

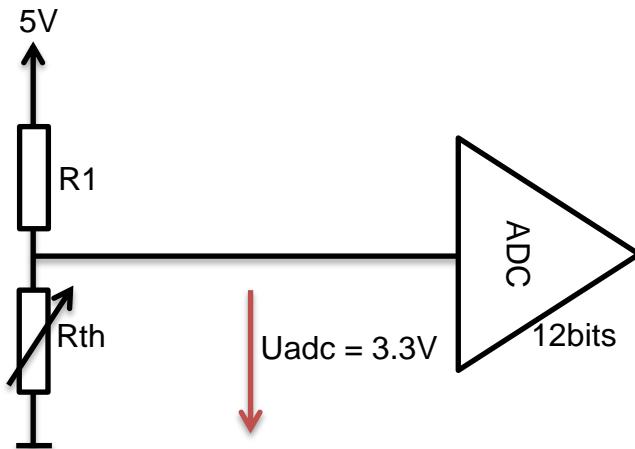
Elle peut supporter un courant maximum de 1A et une tension inverse de 40V, donc nous pouvons la utiliser dans ce montage.

6.5.1.2 Interface analogique - Protection des entrées ADC contre surtension

Pendant le travail de semestre, j'ai remarqué qu'il y a un risque de surtension sur les entrées ADC de microcontrôleur lorsque le capteur n'est pas connecté.

Principe de fonctionnement:

Pour les thermistances, un pont diviseur est mis en place, comme illustrée sur l'image ci-dessous :



R_{th} = la valeur de thermistance qui varie en fonction de la température/pression.

Si le capteur n'est pas connecté (absence de la résistance R_{th} sur l'image ci-dessus), nous pourrons se retrouver avec une tension d'environ 5V à l'entrée de l'ADC.

En regardant dans le datasheet du PIC32MKGPEXXX, nous pouvons voir que la tension sur les pins qui ne supportent pas 5V ne doit pas dépasser la tension $VDD+0.3$, donc une tension de 3.6V :

2.9.1 NON-5V TOLERANT INPUT PINS

A quick review of the absolute maximum rating section in **36.0 “Electrical Characteristics”** will indicate that the voltage on any non-5v tolerant pin may not exceed $VDD + 0.3V$ unless the input current is limited to meet the respective injection current specifications defined by parameters DI60a, DI60b, and DI60c in **Table 36-**

Extrait du datasheet, page 41

Lien : http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MK_GP_MC_Family_Datasheet_60001402G.pdf

Pour protéger les entrées ADC de microcontrôleur contre les surtensions, j'ai décidé d'utiliser la diode Schottky. En regardant dans le stock de l'ETML, j'ai trouvé le modèle suivant :



Références du composant :

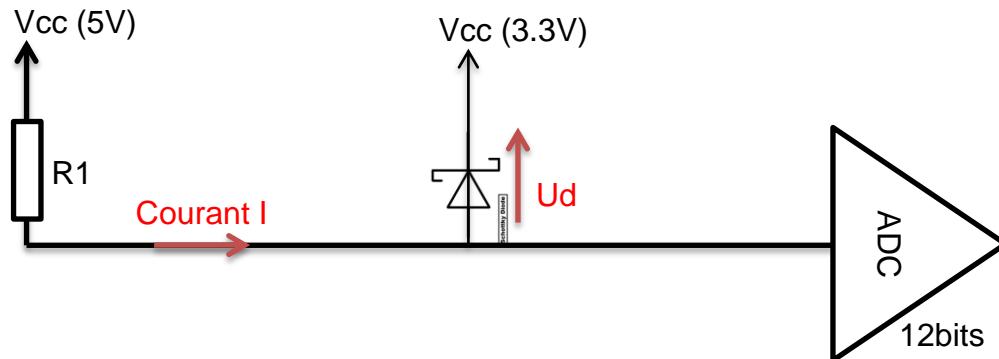
Fabricant : On Semiconductor

Fournisseur : Mouser

Référence fabricant : 512-SS14

Référence fournisseur : SS14

Je vais donc prendre en compte le pire des cas afin de m'assurer que cette diode peut être utilisée:



La résistance R1 a été dimensionnée lors du projet de semestre et elle est différente pour chaque capteur. Sa plus faible valeur est de 91Ω . Je vais donc calculer le courant qui passera dans la diode pour m'assurer qu'elle le pourra supporter.

Je pars du principe que l'impédance de l'ADC est très grande et que tout le courant passera dans la diode :

$$Ud = 0.2V \text{ (valeur théorique)}$$

$$I = \frac{Vcc(5V) - Ud - Vcc(3.3V)}{R1} = \frac{5 - 0.2 - 3.3}{91} = 16.48mA$$

En prenant le pire des cas, le courant que la diode doit supporter est d'environ 20mA.

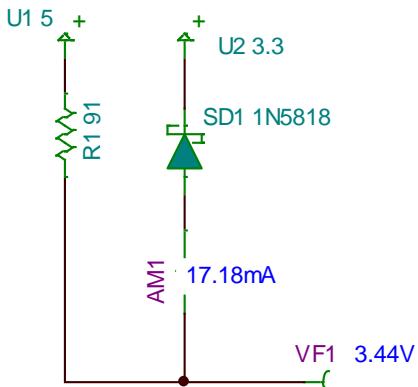
J'ai regardé donc les spécificités du fabricant :

MAXIMUM RATINGS ($T_A = 25^\circ C$ unless otherwise noted)							
PARAMETER	SYMBOL	SS12	SS13	SS14	SS15	SS16	UNIT
Device marking code		S2	S3	S4	S5	S6	V
Maximum repetitive peak reverse voltage	V_{RRM}	20	30	40	50	60	V
Maximum RMS voltage	V_{RMS}	14	21	28	35	42	V
Maximum DC blocking voltage	V_{DC}	20	30	40	50	60	V
Maximum average forward rectified current at T_L (fig. 1)	$I_{F(AV)}$			1.0			A
Peak forward surge current 8.3 ms single half sine-wave superimposed on rated load	I_{FSM}			40			A
Voltage rate of change (rated V_R)	dV/dt			10 000			$V/\mu s$
Operating junction temperature range	T_J			-65 to +150			$^\circ C$
Storage temperature range	T_{STG}			-65 to +150			$^\circ C$

Le courant maximum que la diode peut supporter est de 1A et la tension inverse maximale est de 40V. Ces caractéristiques sont largement suffisantes pour notre utilisation, donc nous pouvons utiliser cette diode pour ce montage.

Simulation :

J'ai décidé de tester le montage dans le programme simulation "Tina". Comme la diode SS14 n'existe pas dans le logiciel, la diode 1N5818 a été utilisée:



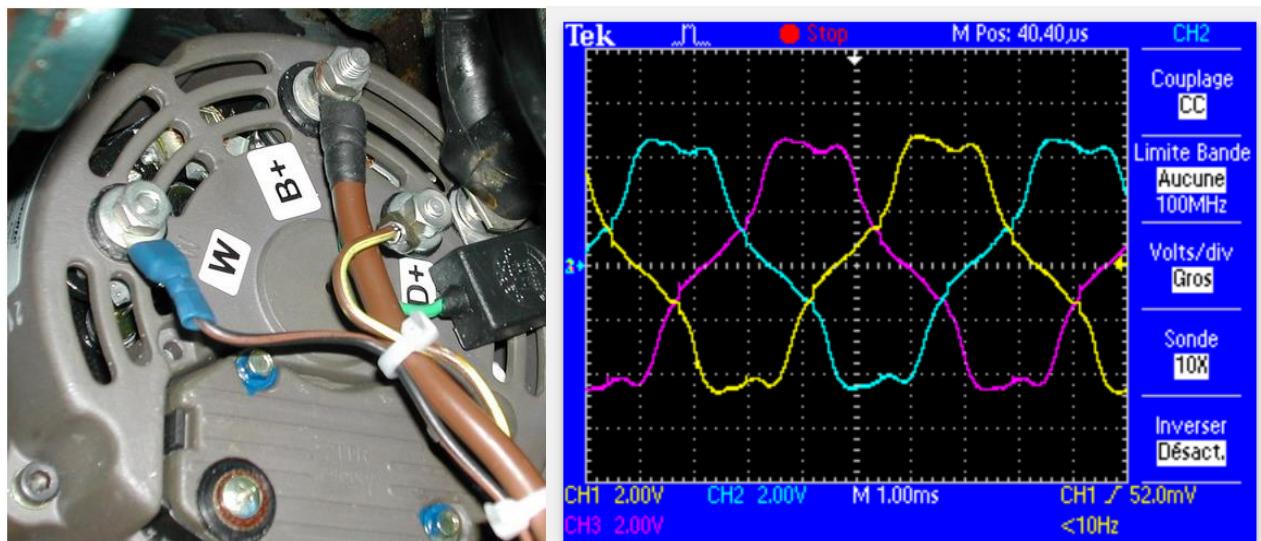
Nous pouvons constater que nous retrouvons un courant d'environ 17mA et que la tension de sortie ne dépasse pas 3.6V.

6.5.2 Travaux restants

6.5.2.1 RPM de l'alternateur

Étude théorique :

Les connexions seront faites sur le connecteur W de l'alternateur. En mesurant sur les trois phases, le signal suivant a été obtenu (la mesure a été envoyée par le client, M.Berney) :

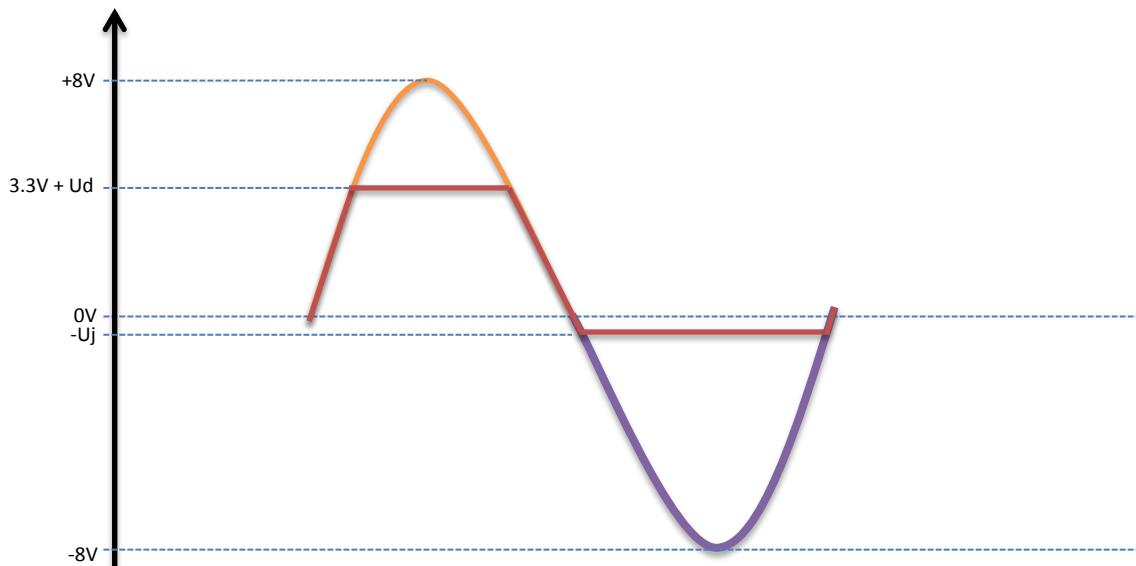
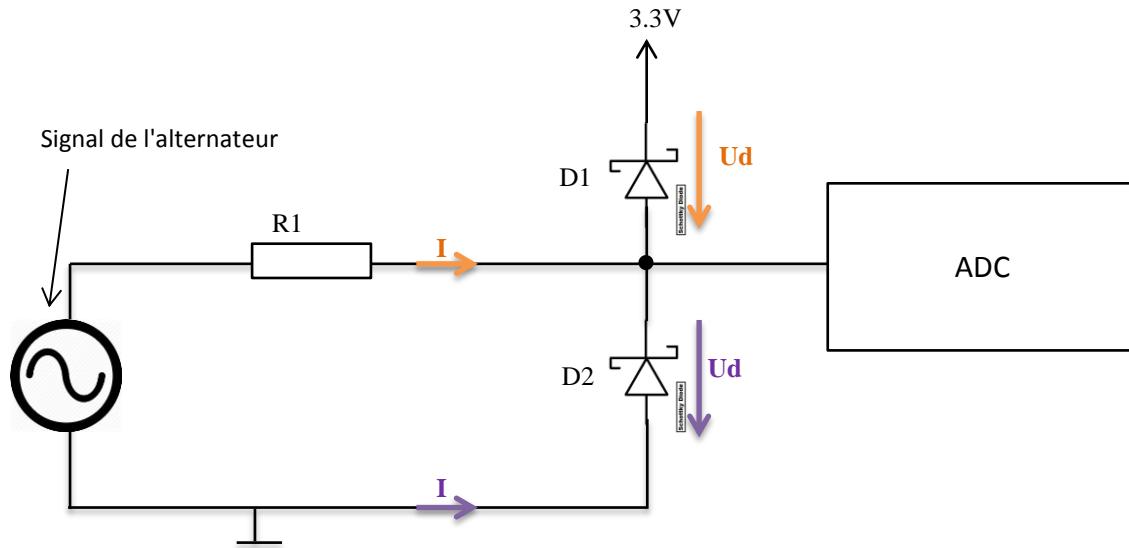


Nous pouvons donc récupérer le signal sur n'importe quelle phase de l'alternateur. Chaque signal a la même amplitude et la période. C'est le déphasage du signal qui change : chaque signal est déphasé de 120°.

Sur l'image ci-dessus, nous pouvons voir que le signal a une amplitude d'environ 8Vpp. Je pars donc du principe que la tension pic du signal peut varier de 3V à 30V maximum.

Étude théorique :

Schéma de départ :



Lorsque les RPM augmentent, la fréquence du signal augmente également.

Rôle des diodes :

Lorsque nous avons l'alternance positive, la diode D1 se met à conduire. La tension à l'entrée de l'ADC est de : 3.3V plus la tension de jonction de la diode.

À l'alternance négative, la diode D2 conduit donc la tension à l'entrée de l'ADC est égale à la tension de jonction de la diode (- U_d).

Dimensionnement de la résistance R1 :

L'amplitude (pic) maximale du signal d'alternateur = 30V

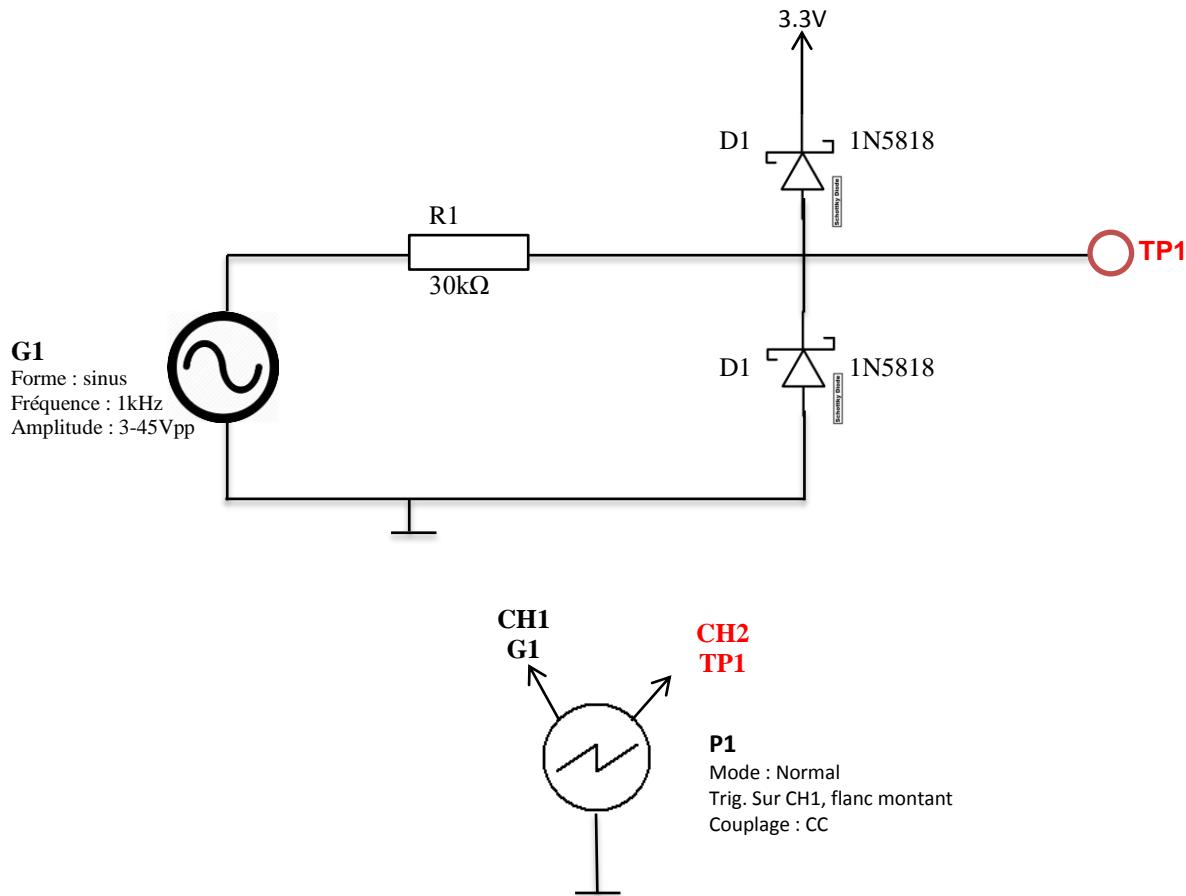
Courant maximale @ Tension maximale = 1mA (valeur choisie)

$U_d = 0.2V$ (valeur théorique)

$$R1 = \frac{3.3 - U_d}{I} = \frac{3.3 - 0.2}{1m} = 30k\Omega$$

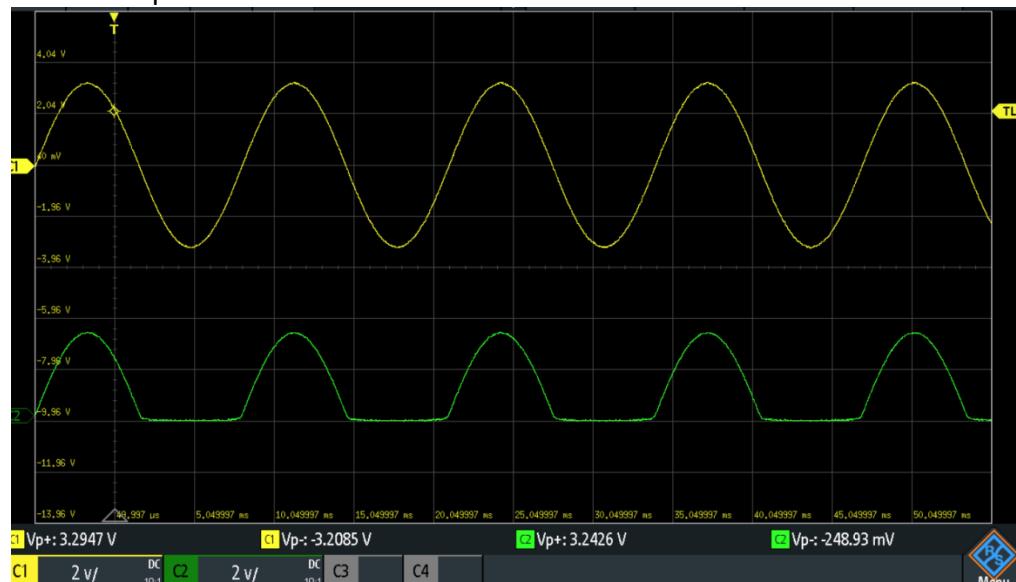
J'ai décidé donc de faire une mesure avec les diodes Schottky que nous avons dans le stock:

Schéma de mesure :



Résultat de la mesure :

Tension du G1 : 3.3V pic



En appliquant une amplitude de 3.3V pic, nous pouvons voir que la diode Schottky ne laisse pas passer l'alternance négative.

Tension du G1 : 45 pic-pic



Avec cette mesure, nous pouvons voir que les signaux correspondent bien à l'étude théorique. Lorsque nous avons l'alternance positive, la tension de sortie est de 3.4V et à l'alternance négative elle est de -0.3V.

Ce signal peut être connecté à l'entrée input capture du microcontrôleur afin de mesurer le temps entre chaque flanc montant.

6.5.3 Capteur de débit d'essence

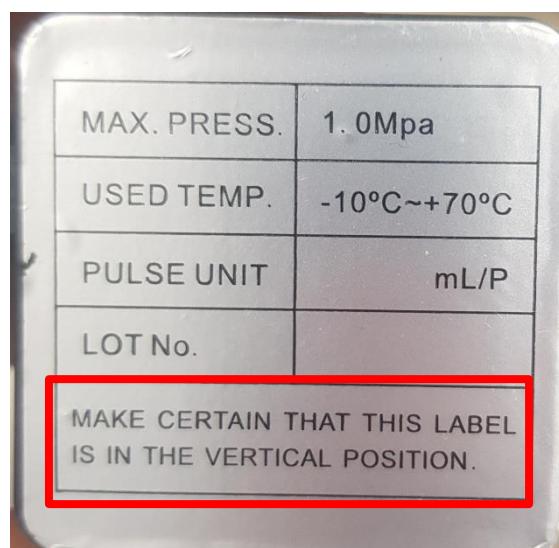


Spécifications du capteur :

- Plage de mesure : 0-10 litres/minute
- Tension alimentation : 3-12V (DC)
- Precision : 0.5%
- Plage de température : -10 à 70°C
- Pression maximale = 1Mpa = 10 bar
- Plage de mesure : 0 à 10litres/minute
- Fixation : ce capteur va venir se connecter sur le tuyau de carburant.



Le capteur doit être placé verticalement.



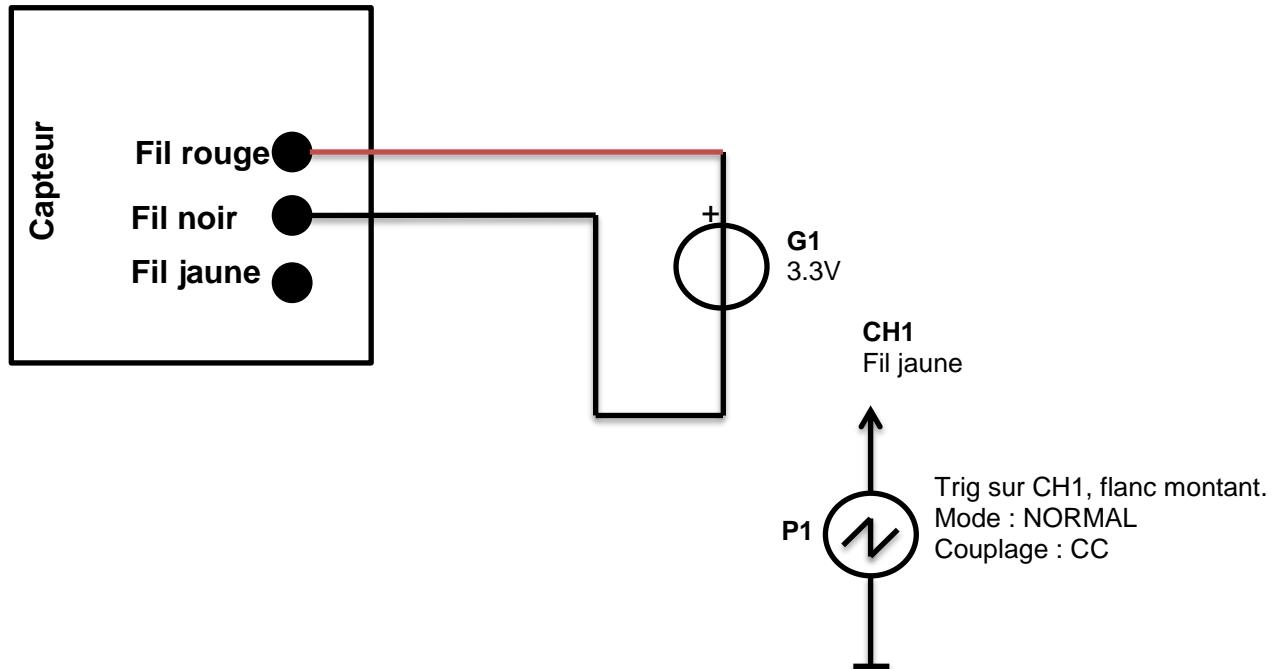
Ce capteur possède 3 fils :



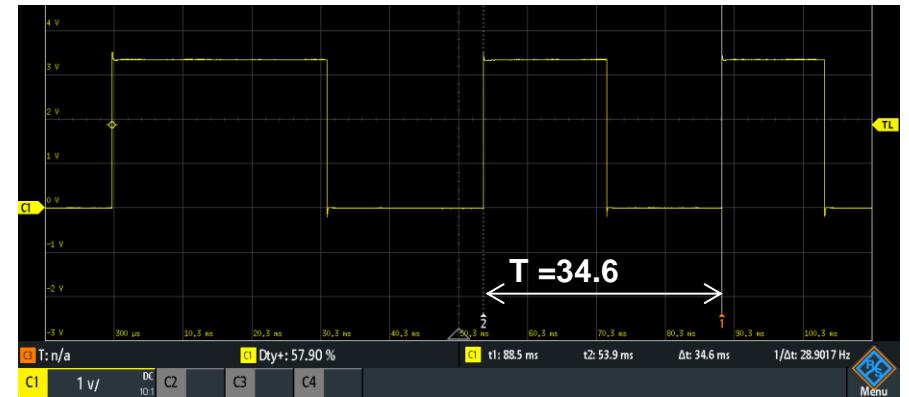
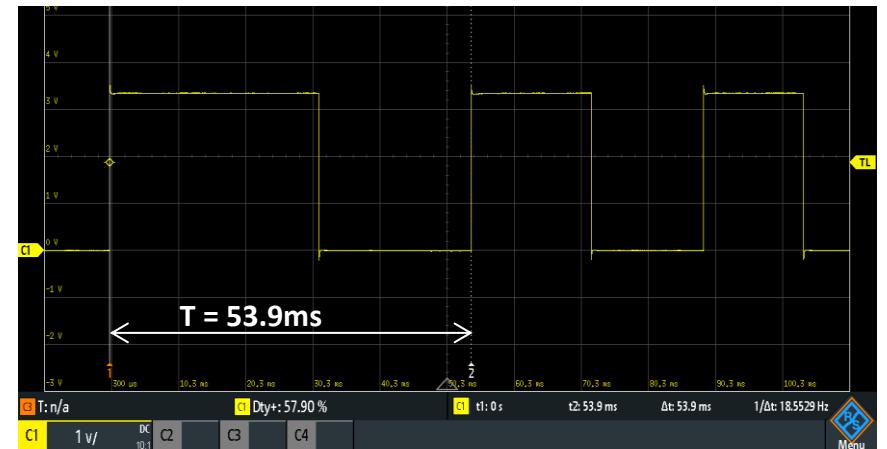
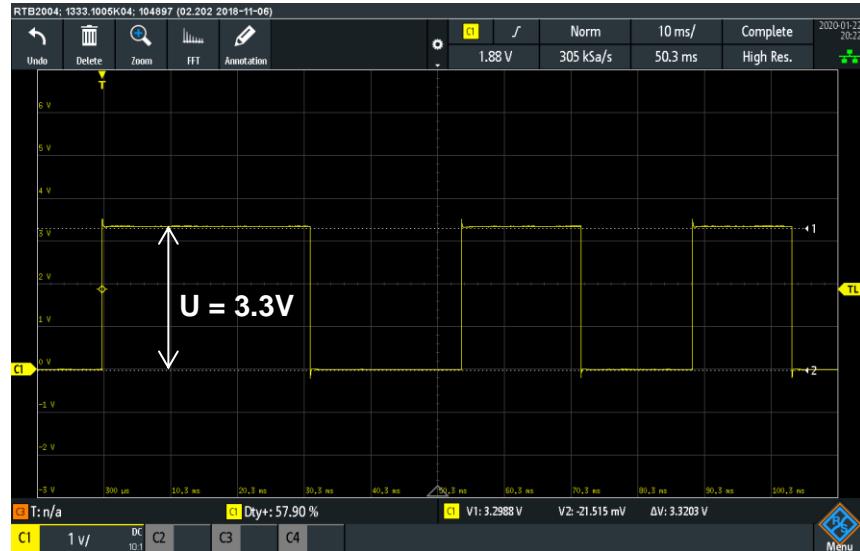
Red---Power + black---Power - yellow ---Pulse singal

Pour visualiser le signal de sortie de ce capteur, j'ai décidé de faire une mesure.
Comme ce capteur peut être alimenté entre 3 à 12V, j'ai décidé de l'alimenter en 3.3V.

Schéma de mesure :



Résultat de la mesure :

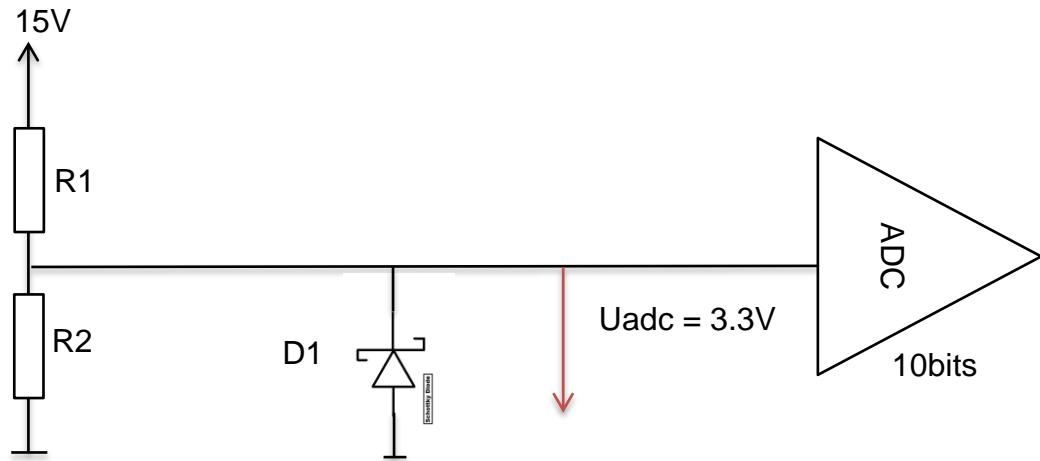


Nous pouvons voir que l'amplitude de signal correspond bien à la tension d'alimentation. Le signal n'est pas bruité donc nous pouvons le connecter tel quel à l'entrée IC du microcontrôleur, sachant que l'amplitude du signal ne dépasse pas 3.3V.

6.5.3.1 Récupération de la tension de la batterie

Dans la trame imposée par le client (M.Berney), il faut également envoyer la tension de la batterie de l'avion.

J'ai mis en place un pont diviseur :



Je pars du principe que la tension de la batterie ne va pas dépasser les 15V.

$R_2 = 2.2k\Omega \rightarrow$ valeur choisie

Formule du pont diviseur :

$$V_{adc} = \frac{R_2}{R_2 + R_1} * V_{cc}$$

$$\frac{V_{adc}}{V_{cc}} = \frac{R_2}{R_2 + R_1}$$

$$V_{cc} * R_{th} = V_{adc} * (R_2 + R_1)$$

$$V_{cc} * R_2 - V_{adc} * R_2 = V_{adc} * R_1$$

$$R_{th} * (V_{cc} - V_{adc}) = V_{adc} * R_1$$

$$R_1 = \frac{R_2 * (V_{cc} - V_{adc})}{V_{adc}}$$

$$R_1 = \frac{2200 * (15 - 3.3)}{3.3} = 7.8k\Omega \quad \rightarrow E24 = 8.2k\Omega$$

Vérification :

$$V_{adc} = \frac{2200}{2200+8.2k} * 15 = 3.17V$$

6.5.3.2 RPM_projet2007

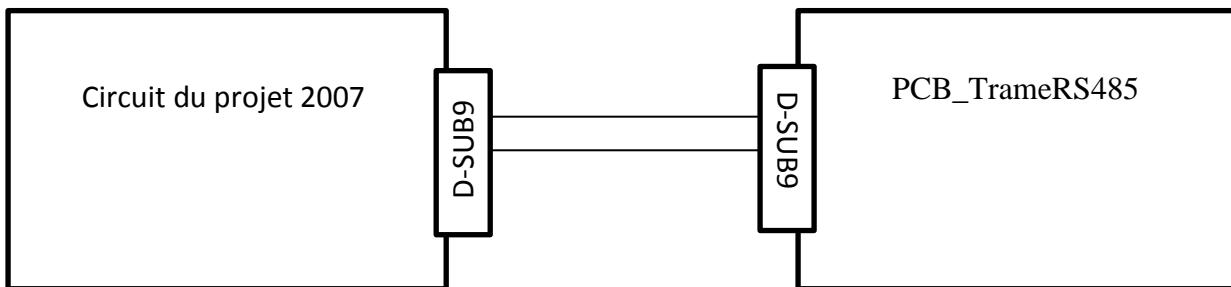
Un autre travail de diplôme (projet N°2007) consiste à récupérer les RPM d'un avion en utilisant les méthodes suivantes :

- Optique - détecter le passage d'hélice et en fonction du nombre de cylindres de l'avion récupérer par calcul les RPM.
- Mesure du bruit : le principe consiste à mesurer le bruit lorsque l'hélice tourne et grâce au filtrage numérique, récupérer les RPM.
- Accéléromètre : mesurer les vibrations de l'avion lorsque l'hélice tourne et en fonction des vibrations déduire le nombre de tours par minute.

L'utilisateur va donc choisir une des méthodes à utiliser pour récupérer les RPM et la trame RS485 sera envoyée sur le circuit "PCB_TrameRS485" avec la forme suivante :

START::RPM=XX

Principe de connexion entre les deux cartes :



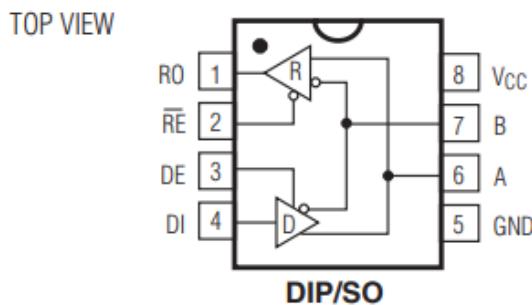
Pour convertir la trame RS485 en UART, je vais utiliser MAX485.

Pour plus de détails sur ce composant, veuillez voir le chapitre suivant.

6.5.3.3 Trame RS485

Pour envoyer la trame RS485, je vais utiliser MAX485.

Pins :



Ro = receiver output (UART)

RE = receiver enable

DE = driver enable

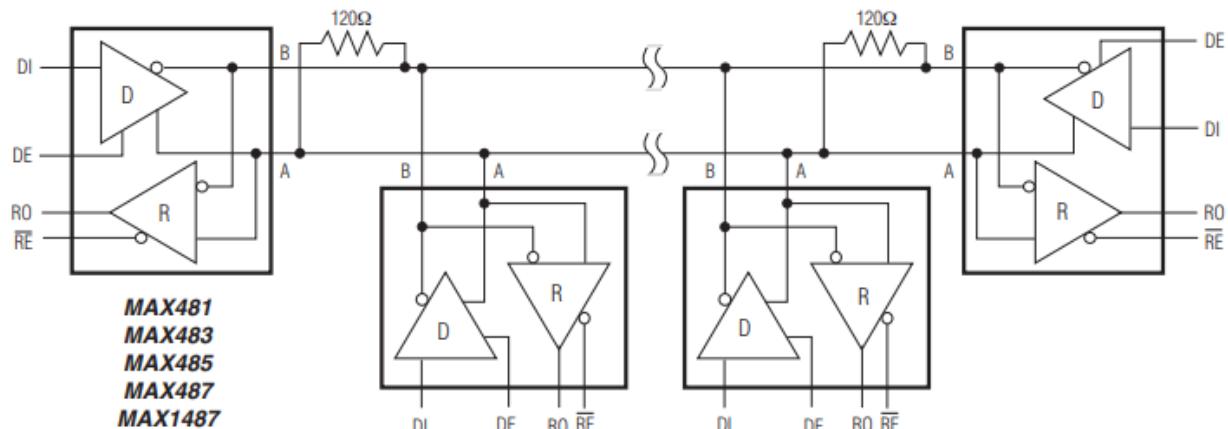
DI = data input (UART)

VCC = la tension d'alimentation (5V)

A = reception frame RS485

B = send frame RS485

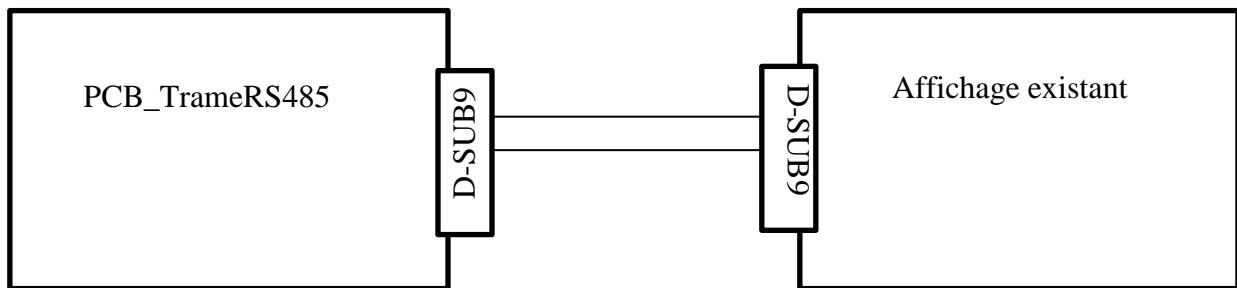
Voici la configuration proposée dans le datasheet :



Il nous est proposé de rajouter une résistance de 120Ω .

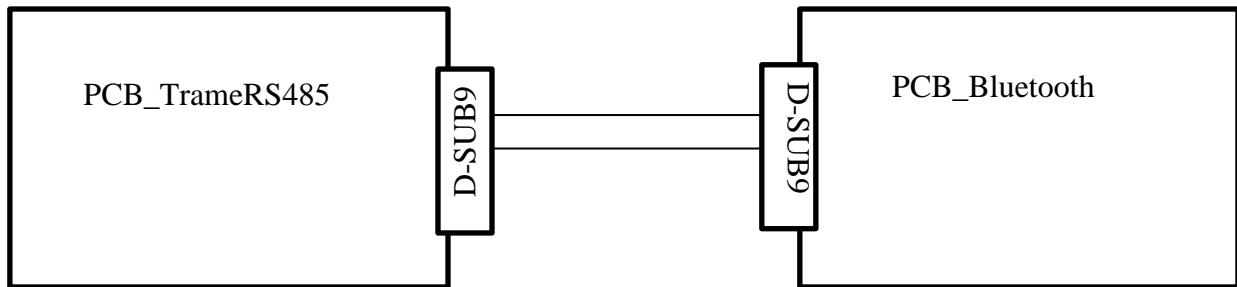
Comme le circuit PCB_TrameRS485 est utilisé pour les deux clients, la même trame sera envoyée.

Principe de connexion pour M.Berney:



Pour M.Berney la trame RS485 est imposée afin de piloter un affichage existant.

Principe de connexion pour l'AMPA:



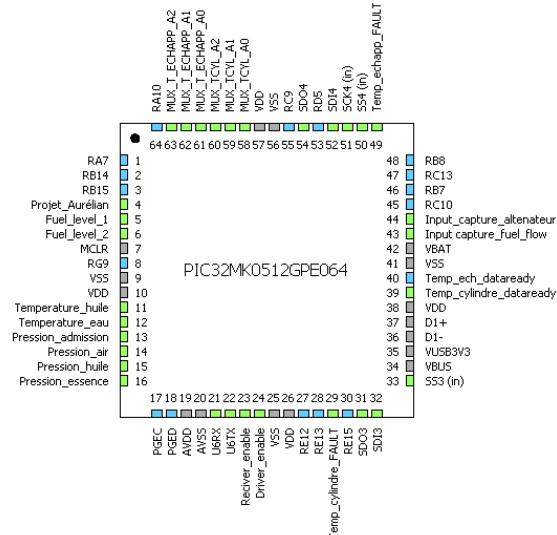
Pour l'AMPA, le circuit PCB_Bluetooth recevra la même trame et par la suite, la même trame sera envoyée au module Bluetooth.

6.5.4 Microcontrôleur

Voici la liste des choses à connecter en plus par rapport au travail de semestre :

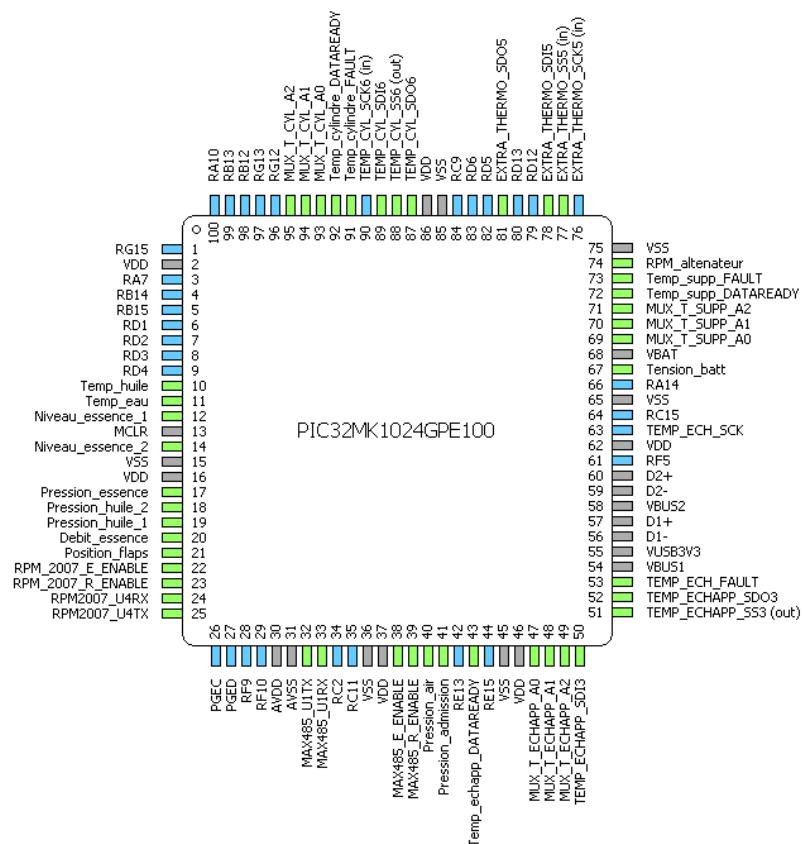
- 8 entrées supplémentaires pour la lecture des thermocouples, donc besoin d'une entrée SPI supplémentaire
- Réception de la trame UART contenant les RPM du projet 2007

Pendant le travail de semestre, le PIC32MK1024GPE064 a été utilisé :



Comme il n'y a pas beaucoup d'entrées disponibles et qu'il faut rajouter des entrées supplémentaires, j'ai décidé de garder le même modèle et prendre la version avec 100 pattes.

Pour s'assurer d'utiliser les bonnes entrées du uc, j'ai configuré toutes les pins dans l'Harmony. Voici la configuration obtenue :



Afin de bien connecter les pins VBUS, D+ et D-, j'ai cherché l'information dans le datasheet :

TABLE 1-15: USB1 AND USB2 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	100-pin TQFP	64-pin QFN/TQFP			
VUSB3V3	55	35	P	—	USB internal transceiver supply. This pin should be connected to VDD.
VBUS1	54	34	I	Analog	USB1 Bus Power Monitor (Tied to VSS if USB1 not used.)
VBUSON1	4	2	O	CMOS	USB1 VBUS Power Control Output
VBUSON2	10	—	O	CMOS	USB2 VBUS Power Control Output
D1+	57	37	I/O	Analog	USB1 D+ (Connect through 10K to VSS if USB1 not used.)
D1-	56	36	I/O	Analog	USB1 D- (Connect through 10K to VSS if USB1 not used.)
USBID1	69	43	I	ST	USB1 OTG ID Detect
VBUS2	58	—	I	Analog	USB2 Bus Power Monitor (Tied to VSS if USB2 not used.)
D2+	60	—	I/O	Analog	USB2 D+ (Connect through 10K to VSS if USB2 not used.)
D2-	59	—	I/O	Analog	USB2 D- (Connect through 10K to VSS if USB2 not used.)
USBID2	77	—	I	ST	USB2 OTG ID detect

Legend:
 CMOS = CMOS-compatible input or output
 ST = Schmitt Trigger input with CMOS levels
 TTL = Transistor-transistor Logic input buffer

Analog = Analog input
 O = Output
 PPS = Peripheral Pin Select

P = Power
 I = Input

Vbus3v3 : à connecter sur Vdd

Vbus1, Vbus2 : à connecter à la masse si l'USB n'est pas utilisé

D1+, D2+ : à connecter à la masse à travers la résistance de 10k

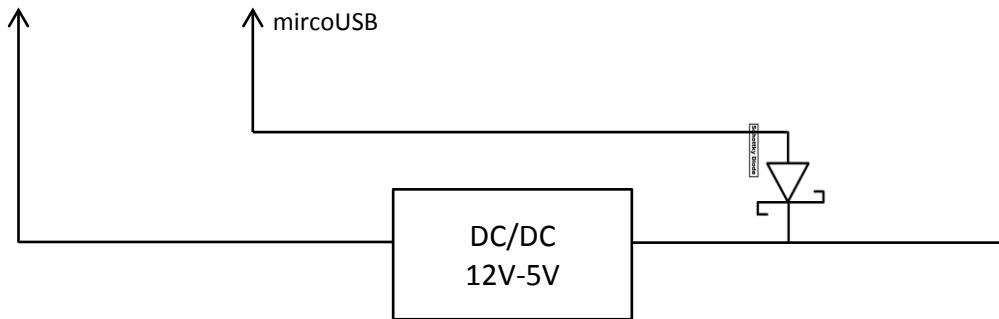
D1-, D- : à connecter à la masse à travers la résistance de 10k

6.6 PCB_Bluetooth

6.6.1 Alimentation

Selon le cahier des charges, il faut alimenter le circuit avec un micro USB ou avec un jack d'alimentation.

Jack d'alimentation

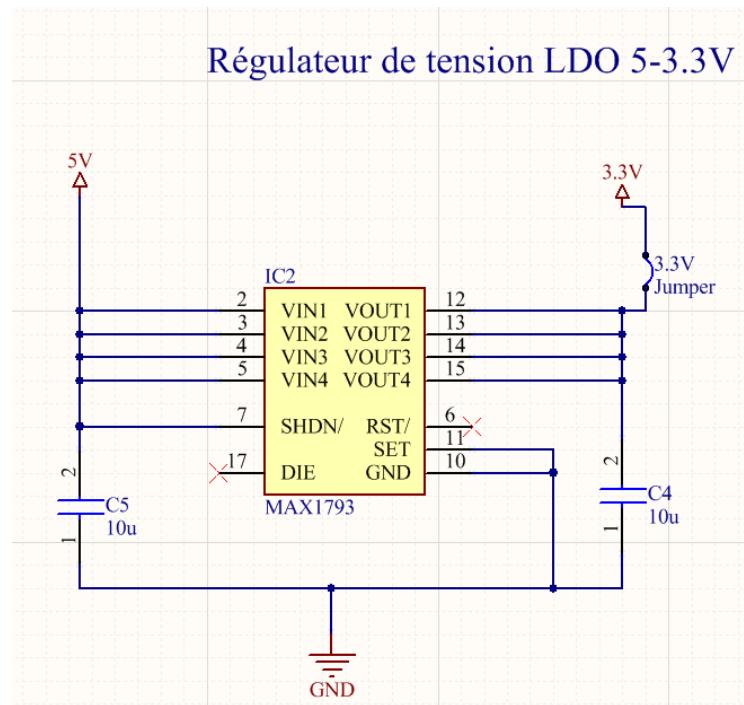


Dans le cas où le circuit est alimenté avec un jack, l'alimentation à découpage est utilisée afin d'obtenir une tension de sortie de 5V.

Si l'utilisateur connecte le câble USB, nous aurons aussi une tension de 5V à la sortie de l'alimentation à découpage.

Comme protection, j'ai utilisé la diode Schotky, ce qui va empêcher d'avoir une tension plus grande de 5V sur le bus USB.

Ensuite, nous avons besoin d'avoir une tension de 3.3V pour alimenter le microcontrôleur. Pour le faire, comme sur le circuit PCB_TrameRS485, je vais utiliser le régulateur de tension MAX1793.



6.6.2 RS485 to UART

Comme sur le circuit PCB_TrameRS485, je vais utiliser le MAX485 pour convertir la trame RS485 en UART.

Pour plus d'explication, veuillez consulter le chapitre 6.5.4 TrameRS485.

6.6.3 UART to Bluetooth

En discutant avec M.Castoldi (l'enseignant à l'ETML-ES), il m'a proposé d'utiliser le module Bluetooth RN42.

En regardant sur le site de Microchip, ce module Bluetooth n'est pas recommandé pour les nouveaux designs. Comme c'est un module Bluetooth qui a déjà été utilisé à l'école, et comme il est encore disponible chez les fournisseurs, suite à la discussion avec mon responsable de projet M.Moreno nous avons décidé de l'utiliser :

RN42 ☆

Status: Not Recommended for new designs

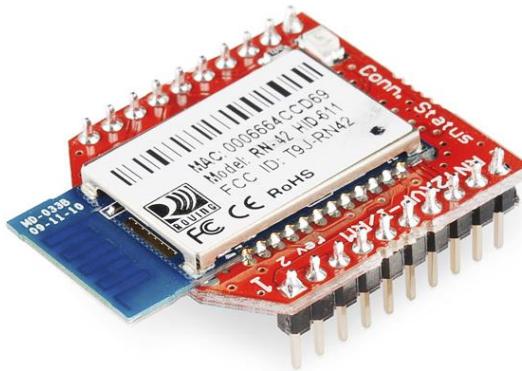
 View Datasheets

 View Comparisons

Features:

- Fully certified Class 2 Bluetooth 2.1 + EDR module
- Onboard embedded Bluetooth stack (no host processor required)
- UART (SPP or HCI) and USB (HCI only) data connection hardware interfaces

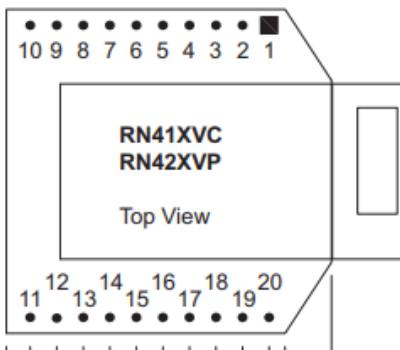
Afin de simplifier le routage du PCB, je vais choisir le module monté sur un socle :



Fabricant : Microchip technology
Fournisseur : Mouser

Référence fabricant : RN42XVP-I/RM
Référence fournisseur : 765-RN42XVP-I/RM

Les pins utilisés :



Pin Number	Signal Name	Description	Optional Function	Direction
1	VDD_3V3	3.3 V regulated power input to the module.		Power
2	TXD	UART TX, 8 mA drive, 3.3-V tolerant.		From module
3	RXD	UART RX, 3.3 V tolerant.		To module
4	GPIO7	GPIO, 24 mA drive, 3.3 V tolerant/ADC input.		I/O
5	RESET_N	<i>Optional</i> module reset signal (active low), 100 k pull up, apply pulse of at least 160 µs, 3.3 V tolerant.		Input
6	GPIO6	GPIO, 24 mA drive, 3.3-V tolerant/ADC input.	Data TX/RX	From module
7	GPIO9	GPIO, 24 mA drive, 3.3-V tolerant/ADC input.		I/O
8	GPIO4	GPIO, 24 mA drive, 3.3 V tolerant/ADC input.		I/O
9	GPIO11	GPIO, 8 mA drive, 3.3 V tolerant.		I/O
10	GND	Ground.		Ground
11	GPIO8	GPIO, 8 mA drive, 3.3-V tolerant. The RN41XV and RN42XV drive GPIO8 high on powerup, which overrides software configured powerup values, on GPIO8.		I/O
12	RTS	UART RTS flow control, 8 mA drive, 3.3 V tolerant.		From module
13	GPIO2	GPIO, 24 mA drive, 3.3 V tolerant/ADC input.		I/O
14	Not Used	No connect.		No Connect
15	GPIO5	GPIO, 24 mA drive, 3.3 V tolerant/ADC input.		I/O
16	CTS	UART CTS flow control, 3.3 V tolerant.		To module
17	GPIO3	GPIO, 24 mA drive, 3.3 V tolerant/ADC input.		I/O
18	GPIO7	GPIO, 24 mA drive, 3.3 V tolerant/ADC input.		I/O
19	AIO0			
20	AIO1			

Pour utiliser ce module Bluetooth, nous avons besoin de connecter les pins suivantes (entouré en rouge) :

- Pin 1 et 10 : Alimentation
- Pin 5 : Reset
- Pin 2, 3, 12, 16 : bus de communication UART

En ce qui concerne les GPIO je n'ai pas trouvé des explications supplémentaires. Comme nous avons de la place sur le microcontrôleur les GPIO seront connectées et nous pourrons les utiliser au cas où on aura besoin.

6.6.4 Microcontrôleur

Nous avons donc besoin du microcontrôleur pour lire la trame RS485 provenant du circuit "PCB_TrameRS485" et d'envoyer la trame UART au module Bluetooth.

Je vais donc choisir un petit microcontrôleur PIC32 à 28 pattes de la famille MX (famille de microcontrôleurs utilisés à l'école).

En regardant dans le stock de l'ETML, j'ai trouvé le modèle suivant : PIC32MX130F256B.

Par la suite, je suis allé sur le site de Microchip afin de voir les spécificités de ce microcontrôleur :

- Taille de la mémoire programme : 256kB
- Taille de la mémoire RAM : 16kB
- Fréquence d'horloge max : 40Mhz
- Tension d'alimentation : 3.3V
- 2 UART

Référence du microcontrôleur :



Fabricant: Microchip Technology
Référence fabricant: PIC32MX130F256B-I/SS

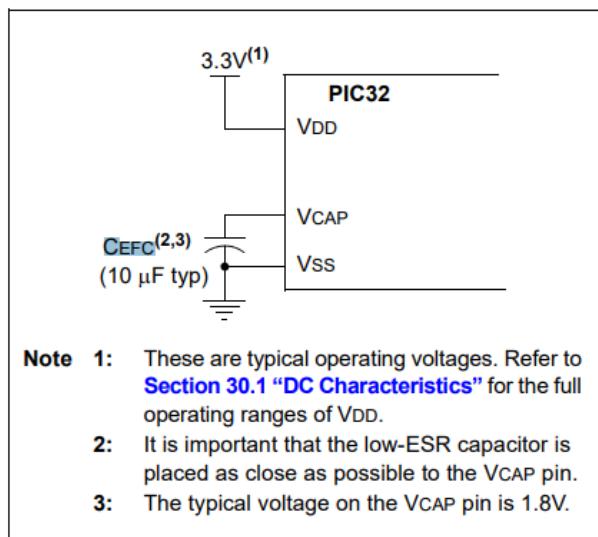
Fournisseur : Mouser
Référence fournisseur : PIC32MX130F256BISS

Pour s'assurer d'utiliser les bonnes entrées du uc, j'ai configuré toutes les pins dans l'Harmony.

Voici la configuration obtenue :

		PIC32MX130F256B	
MCLR	1	28	AVDD
BLUETOOTH_U2RTS	2	27	AVSS
BLUETOOTH_U2RX	3	26	GPIO7_2
BLUETOOTH_U2TX	4	25	GPIO3
GPIO7_1	5	24	GPIO7_2
GPIO6	6	23	GPIO2
GPIO9	7	22	PGEC
VSS	8	21	PGED
BLUETOOTH_U2CTS	9	20	VCAP
GPIO11	10	19	VSS
GPIO4	11	18	MAX485_E_ENABLE
GPIO8	12	17	MAX485_R_ENABLE
VDD	13	16	MAX485_U1TX
RB5	14	15	MAX485_U1RX

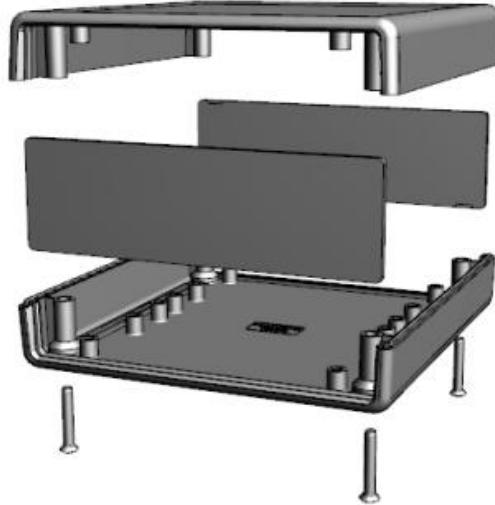
Pour connecter correctement la pin "VCAP", j'ai cherché l'information dans le datasheet :



Je vais utiliser le même montage comme proposé dans le datahseet, donc un condensateur de 10uF sera connecté à la masse.

6.6.5 Boitier

En regardant chez le fabricant "Hammond Manufacturing" j'ai trouvé le boitier suivant :



Ce boitier a été choisi pour les raisons suivantes :

- La face avant et la face arrière s'enlèvent facilement
- Sur la face avant, nous pouvons faire les ouvertures pour le connecteur USB et le connecteur JACK
- Sur la face arrière, le connecteur D-SUB peut être fixé
- Il existe des colonnettes pour fixer le PCB

Références du boitier :

Fabricant : Hammond Manufacturing
Fournisseur : Mouser

Référence fabricant : 1599WBK
Référence fournisseur : 546-1593WBK

7 Schéma électrique

7.1 PCB_Capteurs_AMPA

Veillez consulter l'annexe 5 afin de voir schéma électrique du circuit PCB_Capteurs_AMPA.

7.2 PCB_Capteurs_Berney

Veillez consulter l'annexe 4 afin de voir schéma électrique du circuit PCB_Capteurs_Berney.

7.3 PCB_TrameRS485

Veillez consulter l'annexe 6 afin de voir schéma électrique du circuit PCB_TrameRS485.

7.4 PCB_Bluetooth

Veillez consulter l'annexe 7 afin de voir schéma électrique du circuit PCB_Bluetooth.

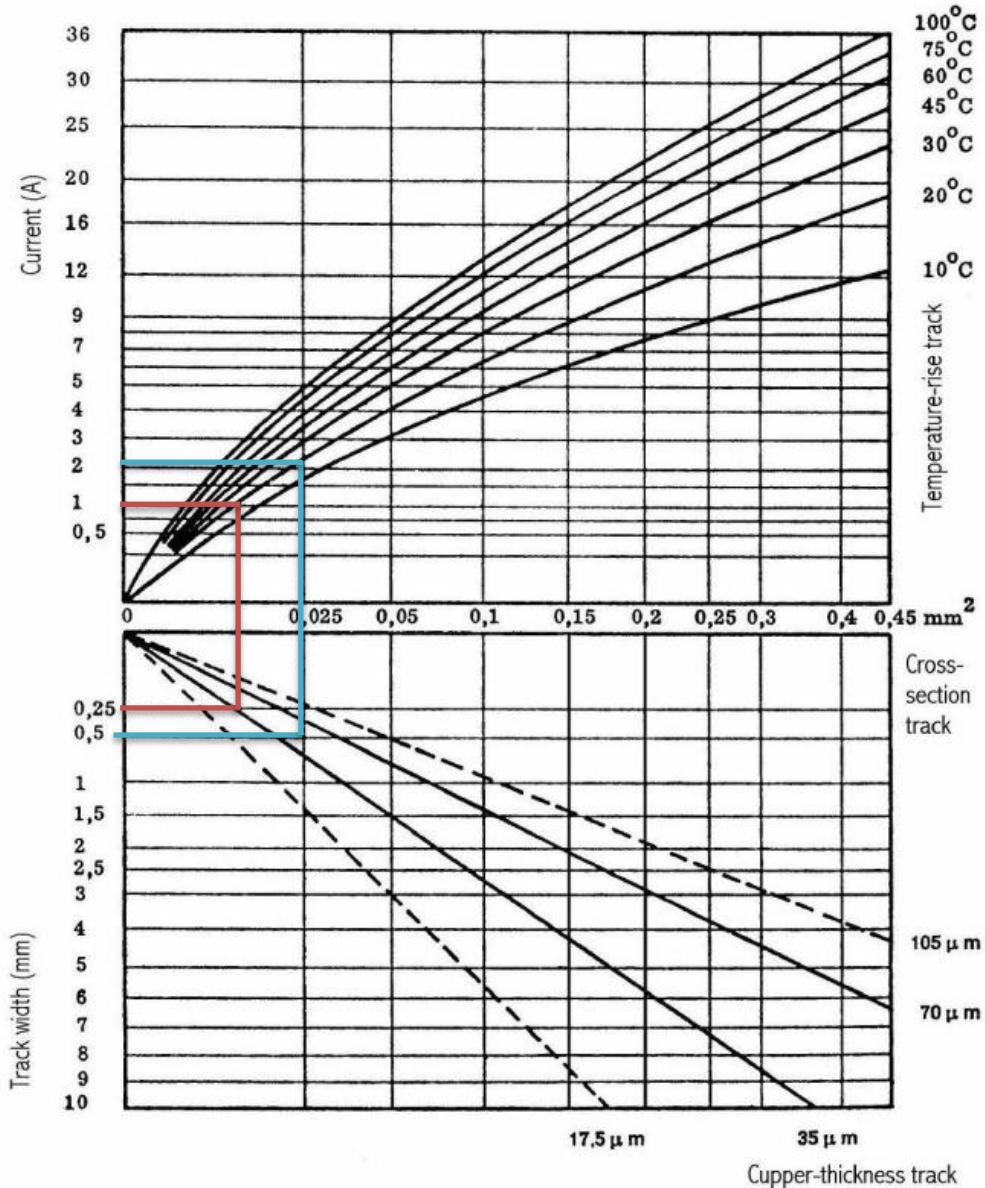
8 Routage

8.1 Paramètres généraux

8.1.1 Taille des pistes

Pour le routage des pistes de l'alimentation, la largeur de piste est de 20mils (0.508mm) et pour le routage des autres signaux, la largeur est de 10mils (0.254mm).

Les recherches ont été faites sur le site d'Eurocircuit, afin de connaitre le courant maximal qui peut circuler en utilisant ces largeurs de pistes :



En utilisant la largeur de piste de 20mils, le courant maximal est de 2A et pour une largeur de piste de 10mils, le courant maximal est de 1A. Sachant que la consommation du circuit ne va pas dépasser les 500mA, donc nous pouvons utiliser ces largeurs de pistes.

8.1.2 Nombre de couches

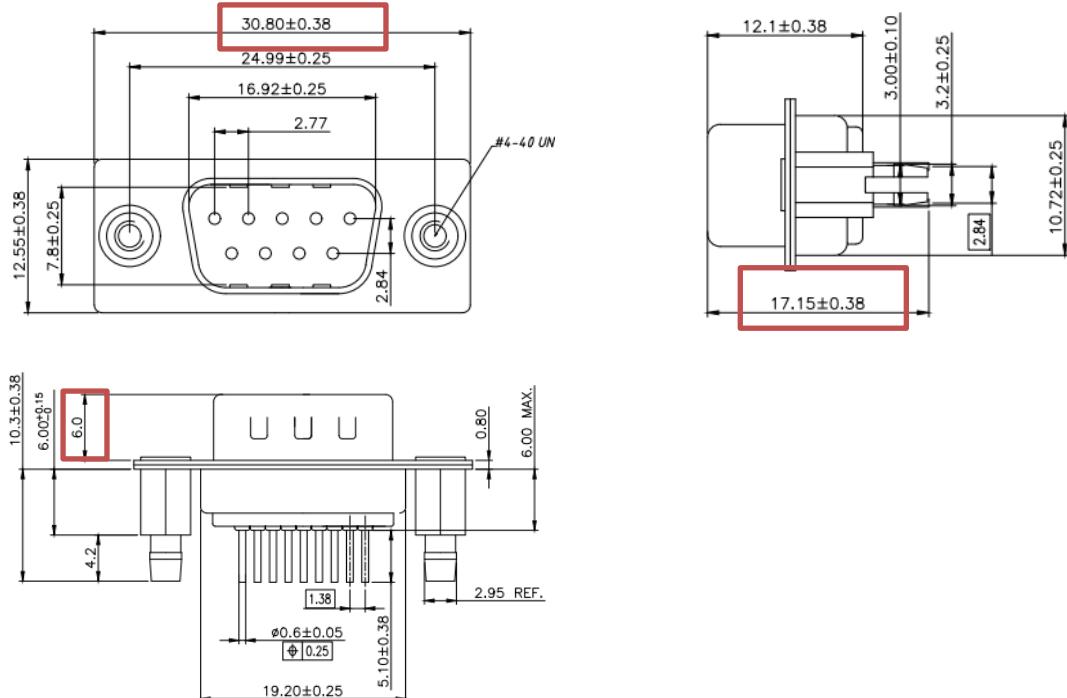
Les circuits ont été routés en deux couches : couche TOP et la couche BOTTOM. Ces paramètres sont importants pour la création de panel.

8.2 PCB_TrameRS485

8.2.1 Forme du circuit :

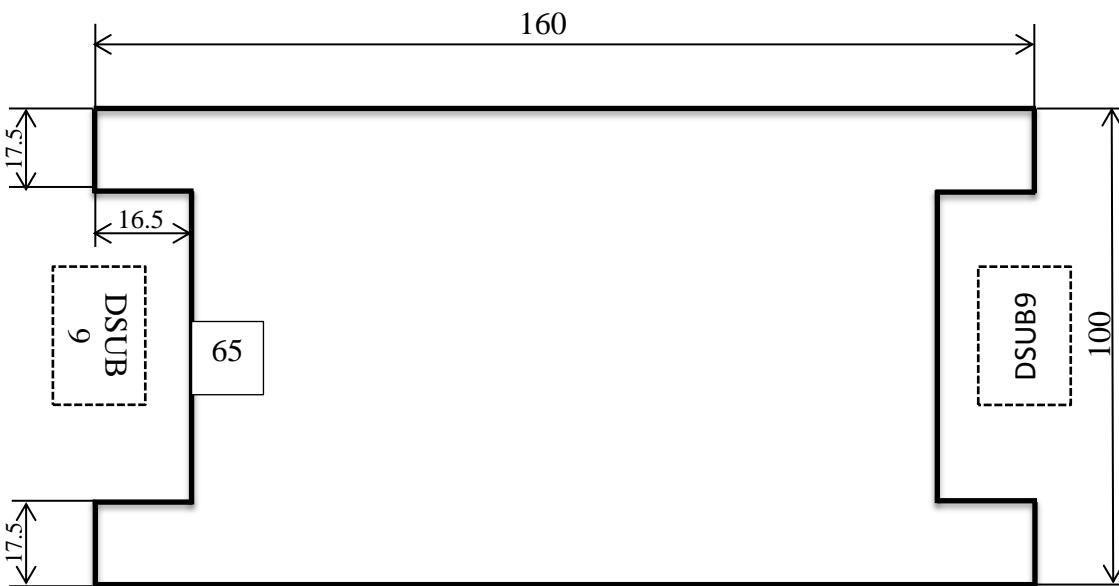
Comme sur la face avant et arrière du boîtier il faut fixer le connecteur DSUB9, j'ai commencé par définir la forme du circuit PCB_TrameRS485.

Dans un premier temps, j'ai regardé les dimensions du connecteur D-SUB9 :



Le connecteur D-SUB à une longueur de 30.8mm et une largeur de 17.15mm. Comme les 6mm vont dépasser du boîtier, il faut prévoir une rainure d'au moins 11.15mm (17.15mm-6mm = 11.15mm).

Voici la forme du circuit que j'ai prévu :



Pour s'assurer que la forme et les dimensions prévues sont bonnes, j'ai décidé de vérifier en utilisant la vue 3D de l'Altium.

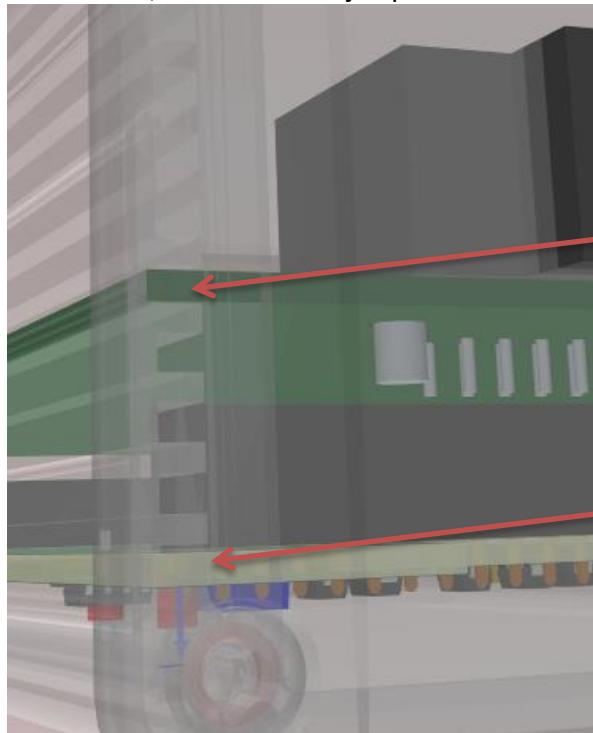
Afin de connaître la hauteur entre les circuits, j'ai utilisé les circuits réalisés pendant le travail de semestre que j'ai placé dans les rainures du nouveau boîtier :



La cinquième rainure

La première rainure

Par la suite, dans l'altium j'ai placé les circuits de la même manière :



La cinquième rainure

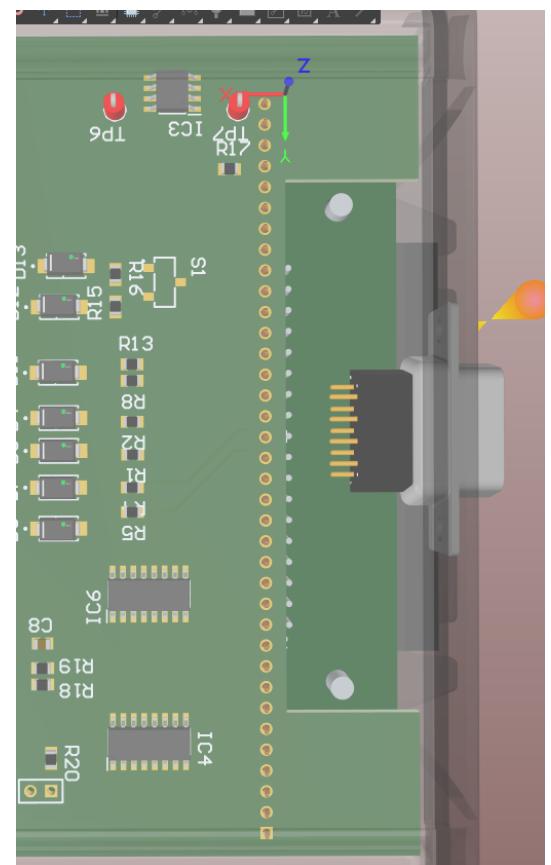
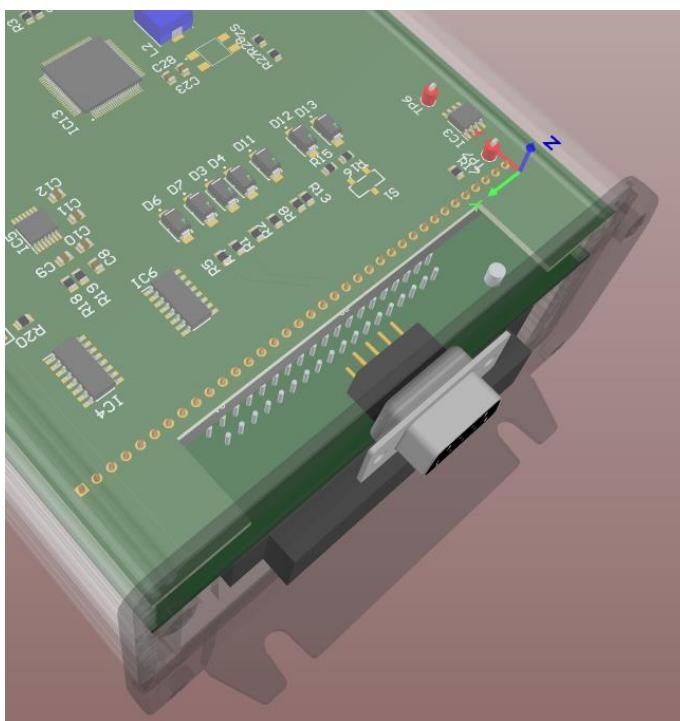
La première rainure

Pour faire l'assemblage, le circuit PCB_Capteurs_AMPA a été utilisé, parce que nous avons les connecteurs D-SUB37 qui seront fixés sur la face avant et arrière.

Sur cette image nous pouvons voir l'emplacement du connecteur D-SUB37 une fois le circuit PCB_Capteurs_AMPA placé :

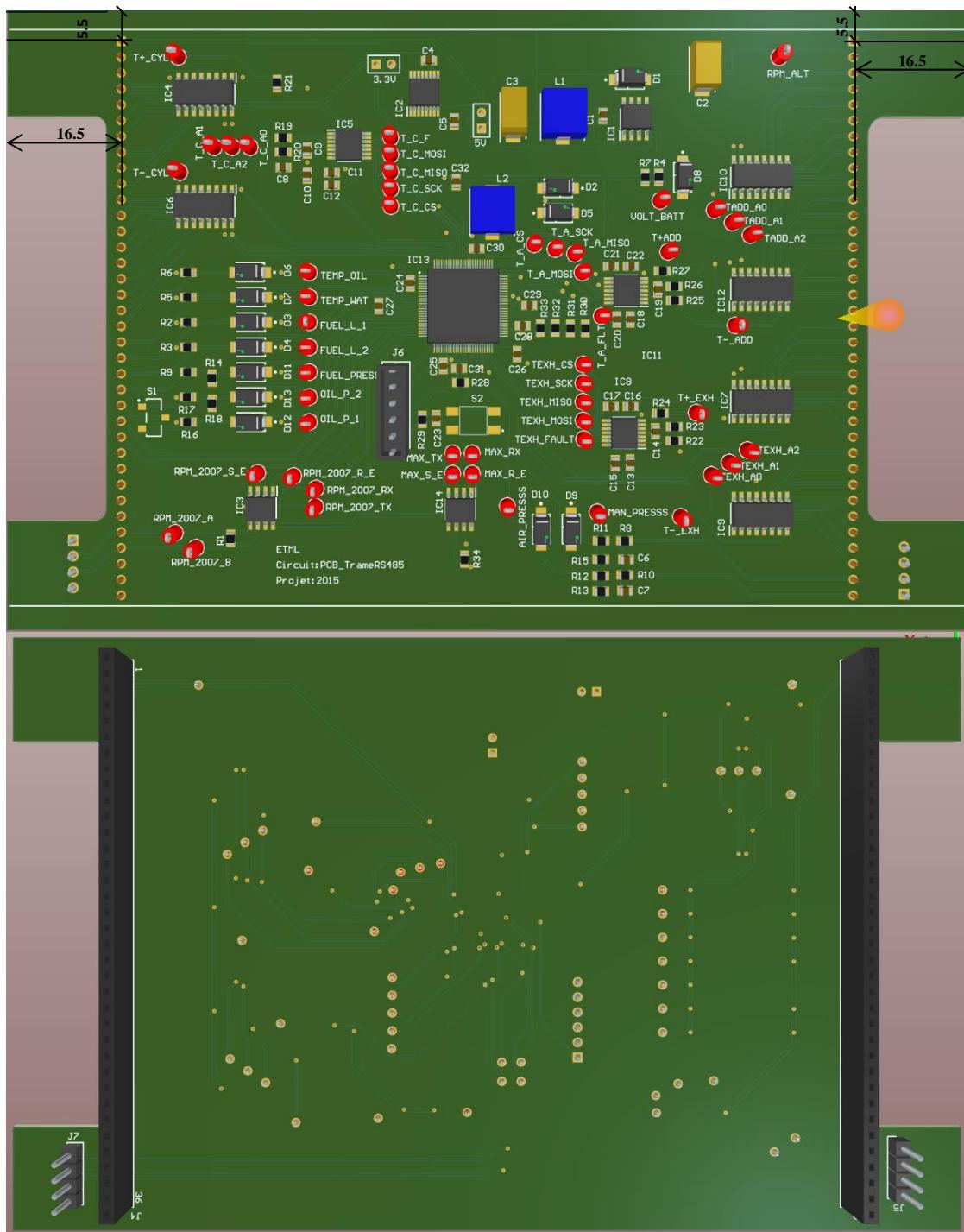


Par la suite, j'ai mis le connecteur D-SUB9 à l'endroit souhaité et j'ai vérifié que nous avons assez d'espace pour faire le câblage nécessaire :



Grâce à la vue 3D de l'Altium, j'ai pu m'assurer que la forme du circuit et la taille de l'ouverture sont correctes.

8.2.2 Contraintes mécaniques

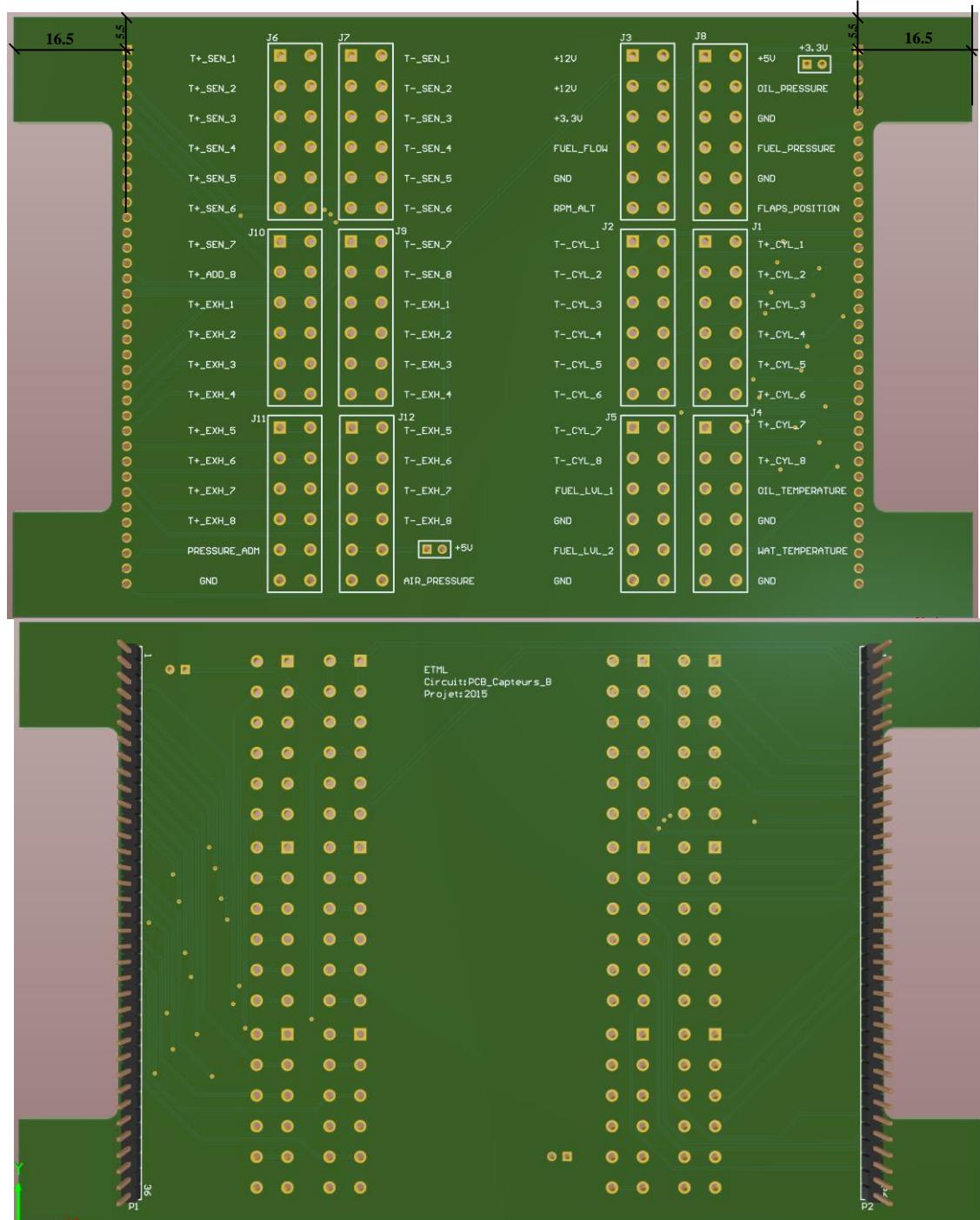


- La taille du circuit est définie en fonction du boîtier choisi (160x100mm)
- Pour pouvoir glisser le circuit dans le boîtier, il faut prévoir une distance de 2mm depuis le bord du circuit. Pour ne pas oublier cette contrainte mécanique, la ligne a été dessinée sur la sérigraphie ,à une distance de 4mm depuis le bord.
- Pour pouvoir placer le circuit PCB_Capteurs_AMPA et PCB_Capteurs_Berney sur les barrettes femelles, il faut respecter les cottes dessinées sur l'image ci-dessus.

Veuillez consulter l'annexe 10 afin de voir le circuit PCB_TrameRS485

8.3 PCB_Capteurs_Berney

8.3.1 Contraintes mécaniques

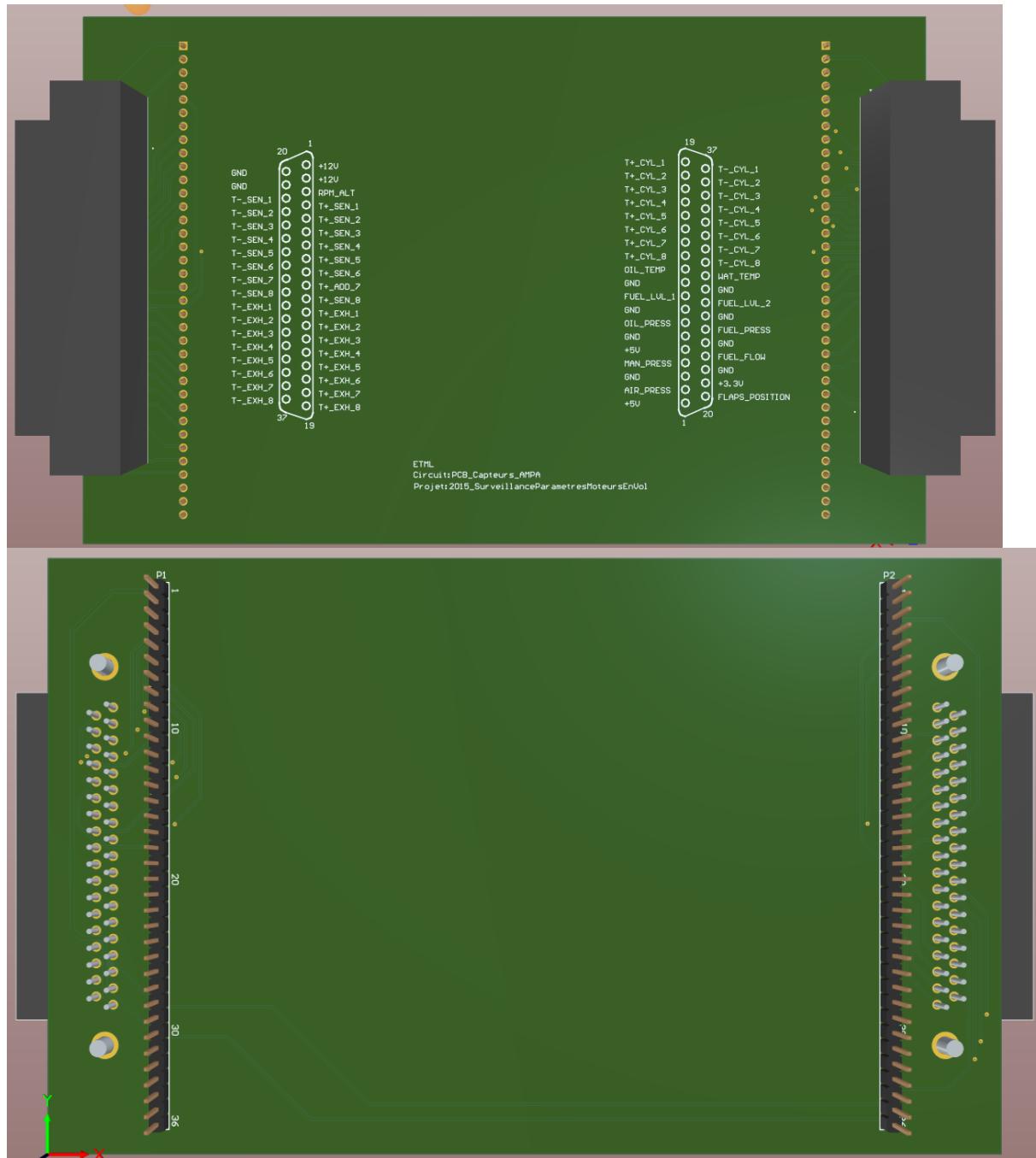


- La taille du circuit est définie en fonction du boîtier choisi (160x100mm)
- Pour pouvoir glisser le circuit dans le boîtier, il faut prévoir une distance de 2mm depuis le bord du circuit.
- Pour pouvoir placer ce circuit sur les barrettes femelles du circuit PCB_TrameRS485, il faut respecter les cottes indiquées sur l'image ci-dessus.
- J'ai décidé de garder la même forme du circuit comme le PCB_TrameRS485 afin de simplifier le montage du connecteur DSUB9 et pour simplifier la fermeture du boîtier une fois les capteurs connectés.

Veuillez consulter l'annexe 9 afin de voir le circuit PCB_Capteurs_Berney

8.4 PCB_Capteurs_AMPA

8.4.1 Contraintes mécaniques

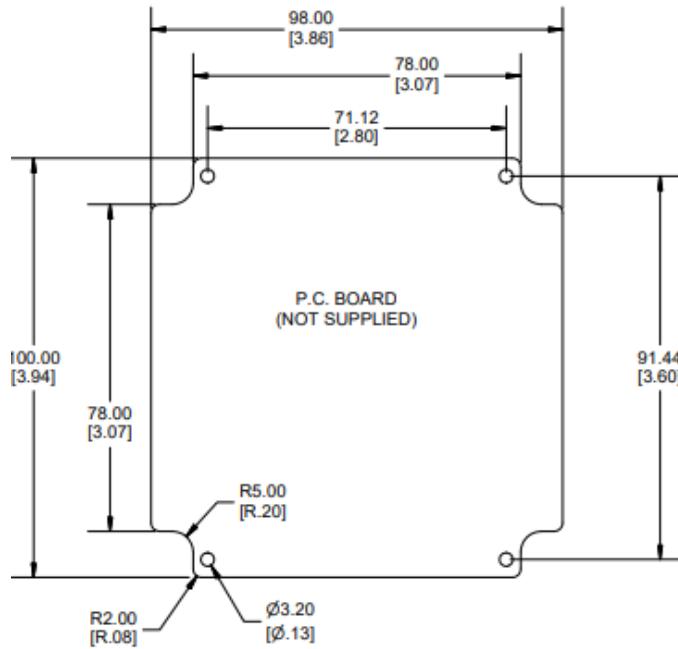


- La taille du circuit est définie en fonction du boîtier choisi (160x100mm)
- Pour pouvoir glisser le circuit dans le boîtier, il faut prévoir une distance de 2mm depuis le bord du circuit.
- Pour pouvoir placer ce circuit sur les barrettes femelles du circuit PCB_TrameRS45, il faut respecter les cottes indiquées sur l'image ci-dessus.
- Le connecteur D-SUB mâle doit dépasser le boîtier pour qu'on puisse brancher le connecteur DSUB femelle une fois le boîtier fermé.

Veuillez consulter l'annexe 8 afin de voir le circuit PCB_Capteurs_AMPA

8.5 PCB_Bluetooth

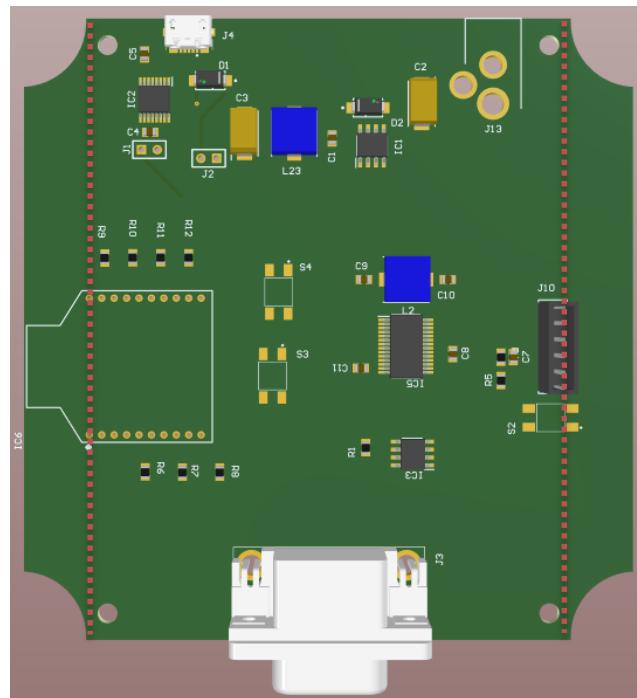
Voici la forme et la taille du PCB proposé par le fabricant du boîtier :



Extrait du datasheet, page 2

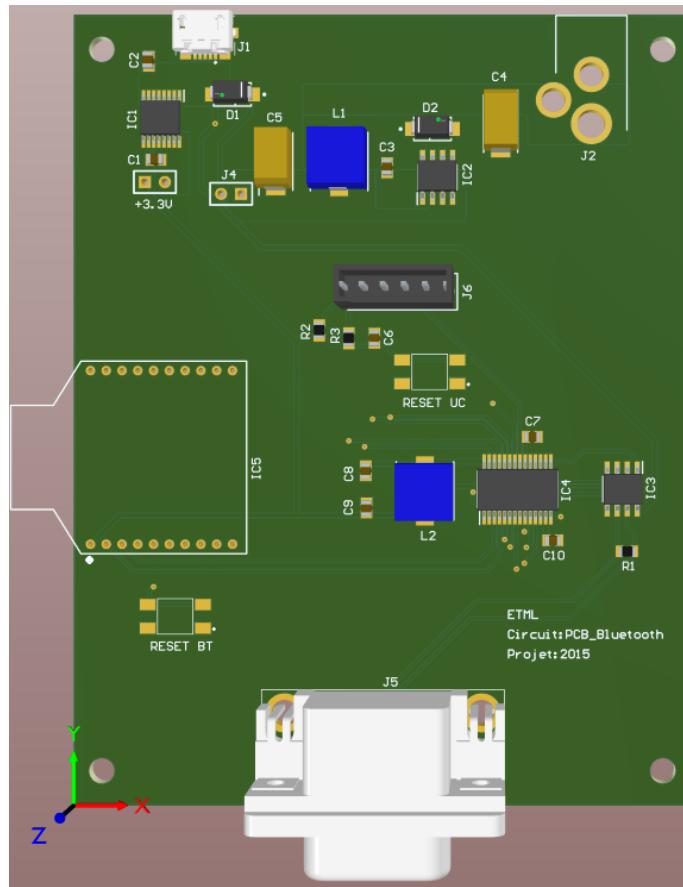
Lien : <https://www.hammfa.com/files/parts/pdf/1593WBK.pdf>

Dans un premier temps, j'ai pris exactement la même forme et taille comme proposé dans le datasheet :



Dans le datasheet du module Bluetooth, il est conseillé de ne pas avoir de plan de masse sous l'antenne Bluetooth. Comme nous avons assez de place sur le circuit, j'ai décidé de redimensionner le circuit selon la largeur de l'antenne (indiqué en rouge sur l'image ci-dessus) et de faire la même chose sur le côté opposé afin de simplifier la mise en boîtier.

Voici la forme du circuit obtenu :



8.5.1 Contraintes mécaniques

- Le connecteur microUSB et le jack d'alimentation sont placés sur la partie supérieure du circuit afin de pouvoir faire les ouvertures nécessaires sur la face avant du boîtier
- Le connecteur DSUB est mis sur la face inférieure du circuit afin de faire l'ouverture nécessaire sur la face arrière du boîtier
- L'antenne du module Bluetooth dépasse le circuit afin d'éviter les perturbations
- Faire des trous de fixations
- Il n'y a pas de composants sur le bottom afin de pouvoir plaquer le circuit dans le boîtier

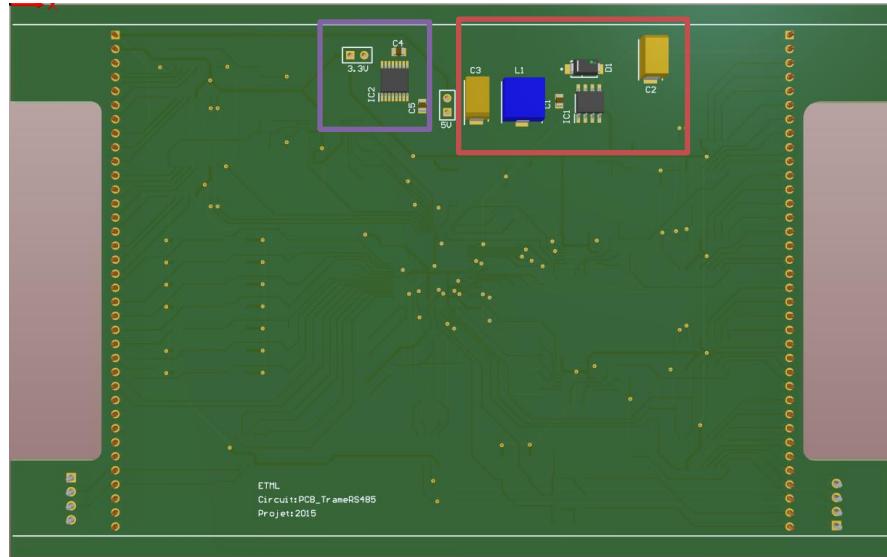
Veillez consulter l'annexe 11 afin de voir le circuit PCB_Bluetooth

9 Montage

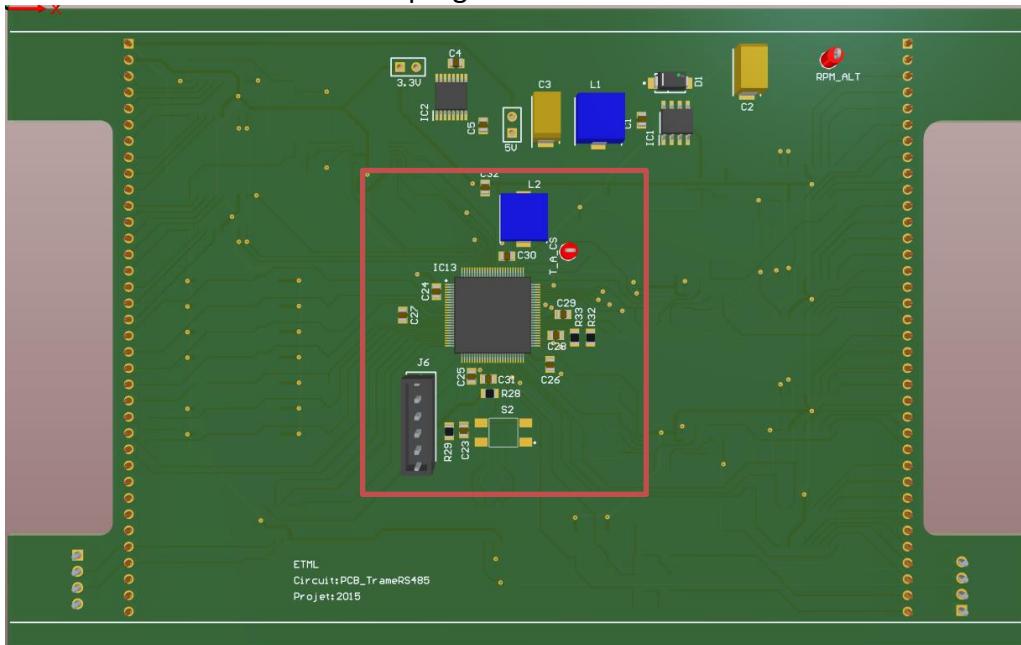
Le montage des composants sur les deux circuits a été fait en parallèle avec la programmation. Dès qu'un bloc des composants (décrit ci-dessous) a été monté, le code nécessaire a été intégré.

9.1 PCB_TrameRS485

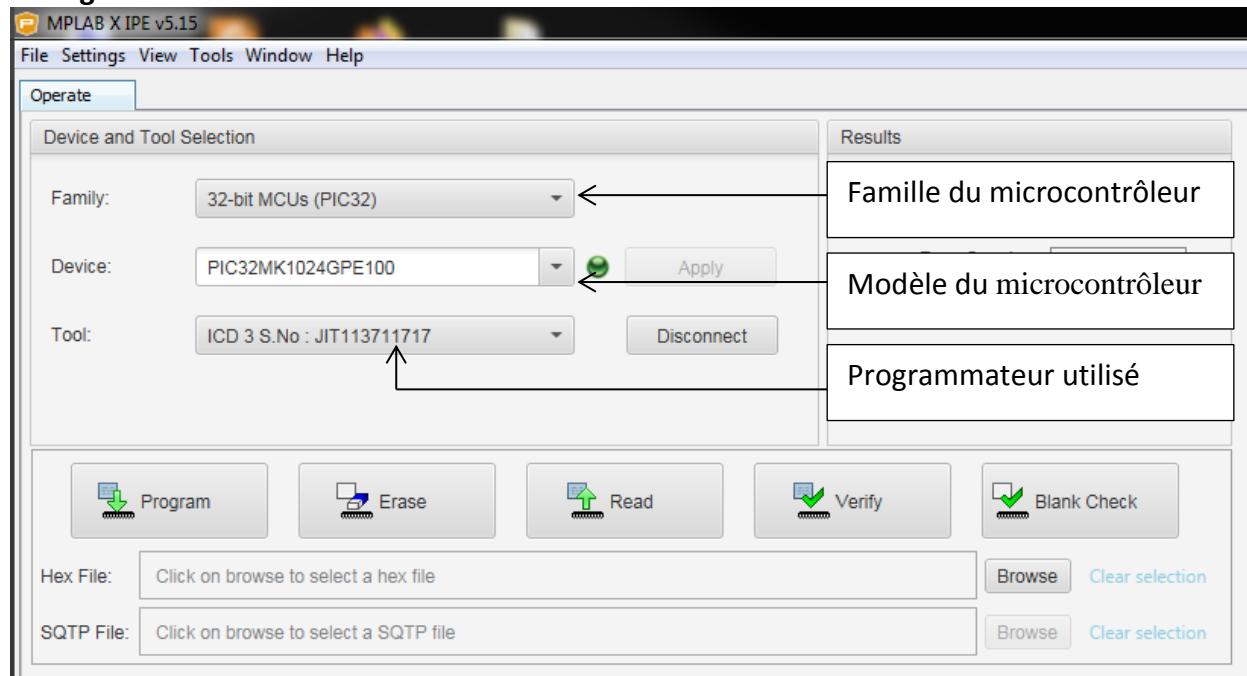
Dans un premier temps, les composants du bloc "Alimentation" ont été montés. L'alimentation à découpage permettant de passer de 12V-5V (entouré en rouge sur l'image ci-dessous) ainsi que le convertisseur DC-DC 5V-3.3V (entouré en violet) ont été montés :



Avant de monter le reste des composants, les tensions ont été mesurées au voltmètre. À la sortie de l'alimentation à découpage la tension de 5V a été mesurée et à la sortie du convertisseur DC-DC la tension était de 3.3V. Comme cette partie est fonctionnelle, j'ai poursuivi par le montage du microcontrôleur et du connecteur de programmation.



Afin de s'assurer que le microcontrôleur a bien été monté, j'ai utilisé le MPLAB X IPE pour essayer de lire la signature du microcontrôleur.

Configurations du MPLAB X IPE :

Dans un premier temps, il faut choisir la famille du microcontrôleur (32-bit MCUs) ainsi que le modèle utilisé (PIC32MK1024GPE100). Par la suite, en appuyant sur bouton "Apply", le message suivant a été observé :

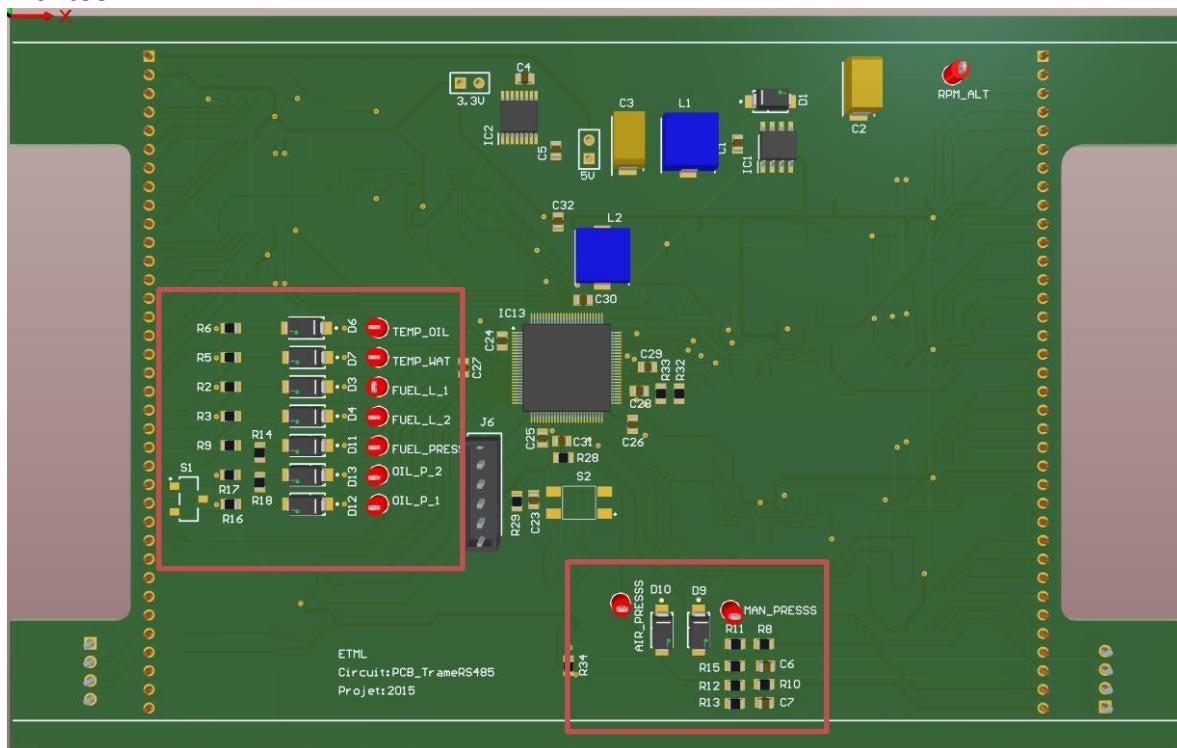
```
Connecting to MPLAB ICD 3...

Currently loaded firmware on ICD 3
Firmware Suite Version.....01.55.01
Firmware type.....PIC32MZ
Target voltage detected
Target device PIC32MK1024GPE100 found.
Device ID Revision = A2
DEVSNO = 002b0055
DEVSN1 = 50525253
```

Nous pouvons constater que le programmeur l'ICD3 a été bien connecté et que le modèle du microcontrôleur a été reconnu.

Le numéro de série du microcontrôleur est contenu dans les registres DEVSNO et DEVSN1 et nous arrivons à les lire.

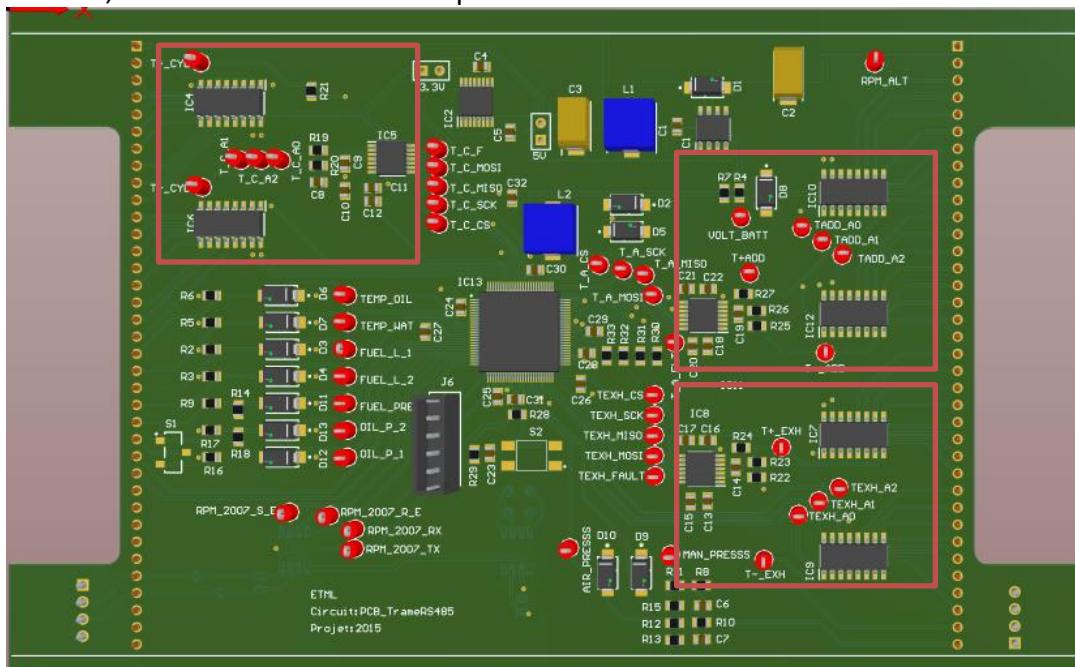
Une fois la signature du microcontrôleur lue, l'interface analogique des thermistances a été montée :



À cette étape, le projet avec MPLABX a été créé. Le code réalisé pendant le travail de semestre permettant de lire les thermistances a été intégré, et un rapide test de fonctionnement a été réalisé.

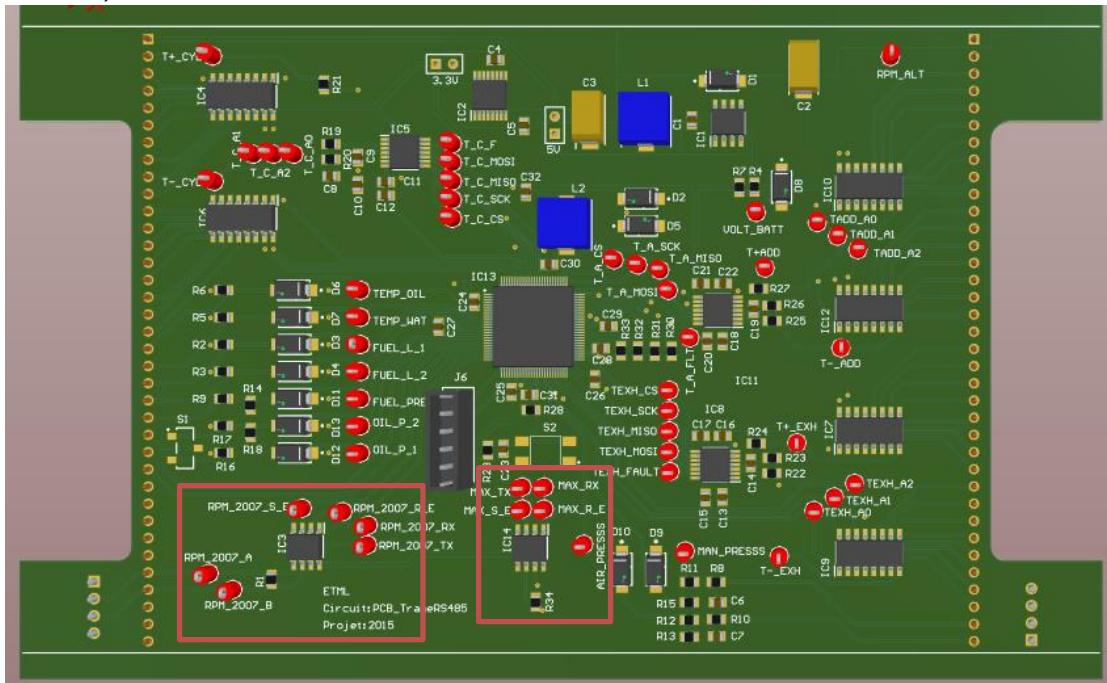
Pour plus de précision sur la création du projet avec MPLABX, veuillez consulter le chapitre 11.1.1

Par la suite, l'interface des thermocouples a été montée :



Comme pour l'interface analogique des thermistances, le code réalisé pendant le travail de semestre a été intégré.

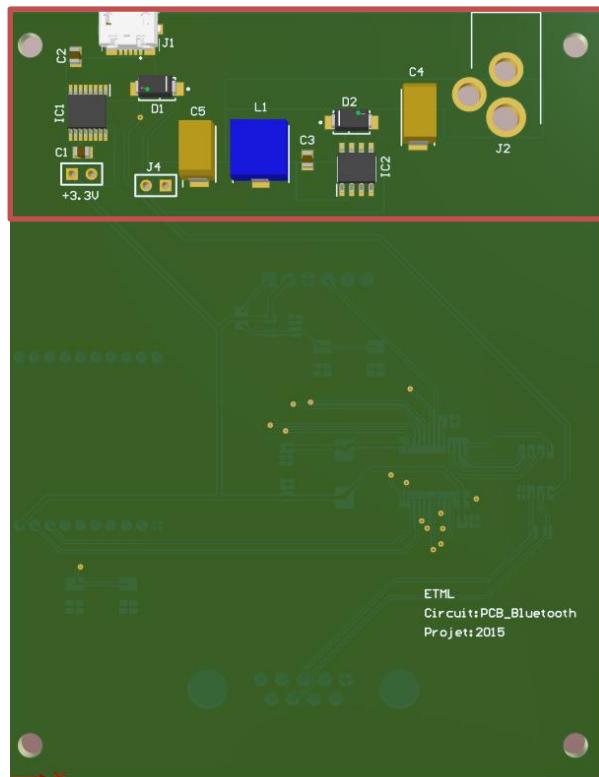
Pour finir, les convertisseurs UART-RS485 ont été montés :



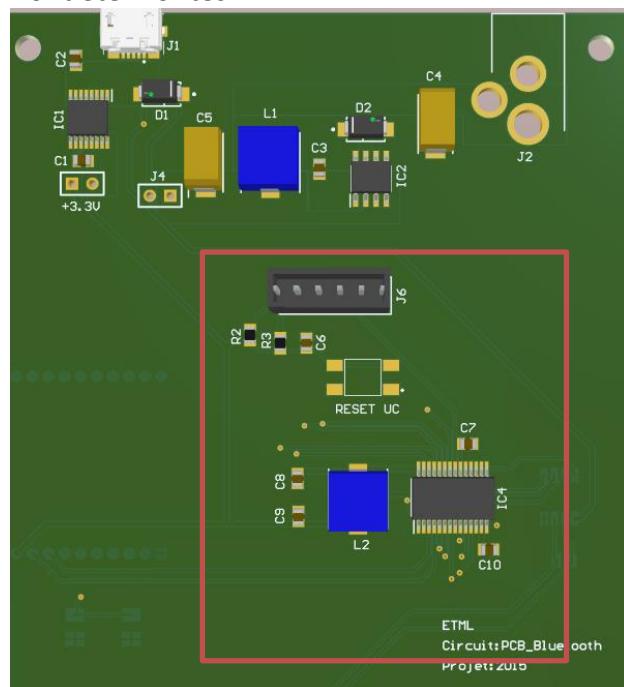
9.2 PCB_Bluetooth

Le PCB_Bluetooth a été monté une fois la bonne trame envoyée par le circuit PCB_TrameRS485.

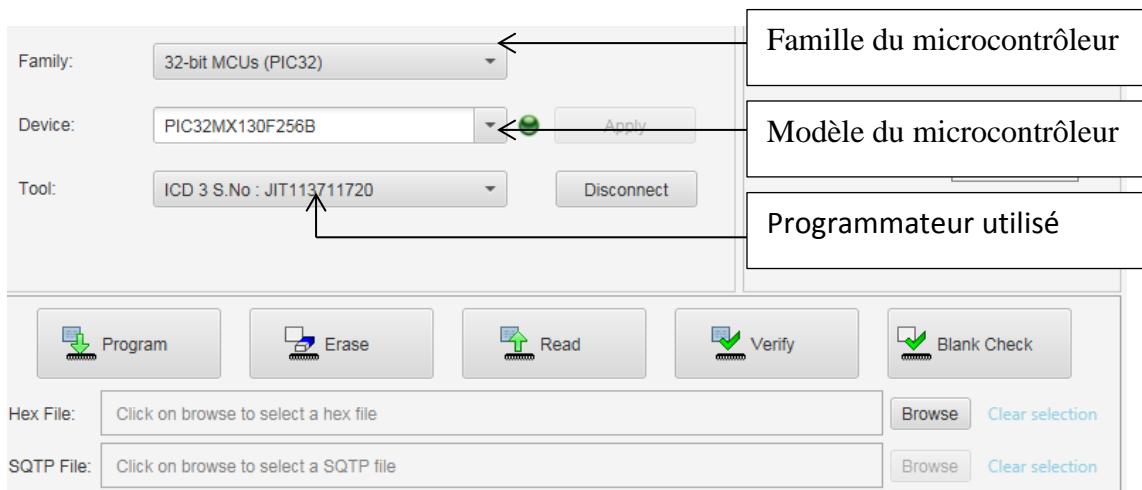
Dans un premier temps, les composants du bloc "Alimentation" ont été montés. L'alimentation à découpage permettant de passer de 12V-5V ainsi que le convertisseur DC-DC 5V-3.3V ont été brasés :



Une fois les bonnes tensions de sortie mesurées, le microcontrôleur et le connecteur de programmation ont été montés :



Ensuite, j'ai essayé de lire la signature du microcontrôleur en utilisant MPLAB x IPE.



La famille du microcontrôleur est 32-bit MCUs et le modèle utilisé est PIC32MX130F256B. Par la suite, en appuyant sur bouton "Apply", le message suivant a été observé :

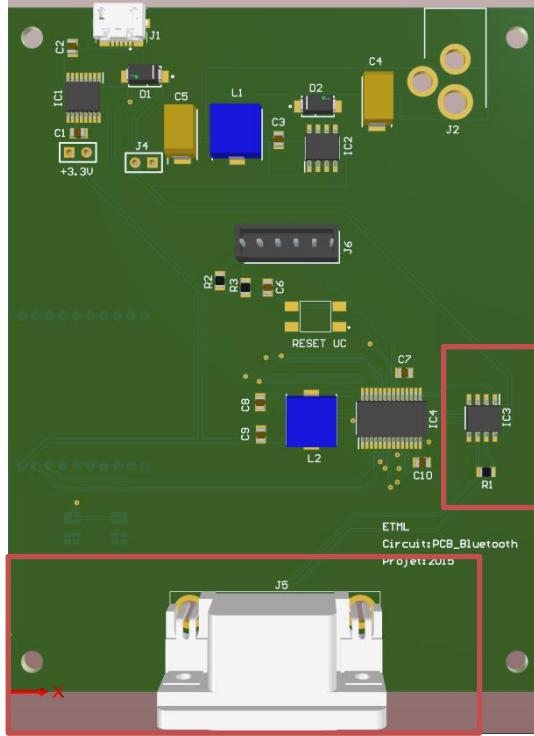
```

Connecting to MPLAB ICD 3...

Currently loaded firmware on ICD 3
Firmware Suite Version.....01.55.01
Firmware type.....PIC32MX
Target voltage detected
Target device PIC32MX130F256B found.
Device ID Revision = A0
  
```

Nous pouvons constater que le modèle du microcontrôleur a bien été trouvé. La signature (numéro de série du microcontrôleur) n'a pas été renvoyée par le logiciel. Ce test a été fait afin de s'assurer que le microcontrôleur est reconnu donc j'ai poursuivi par le montage des autres composants.

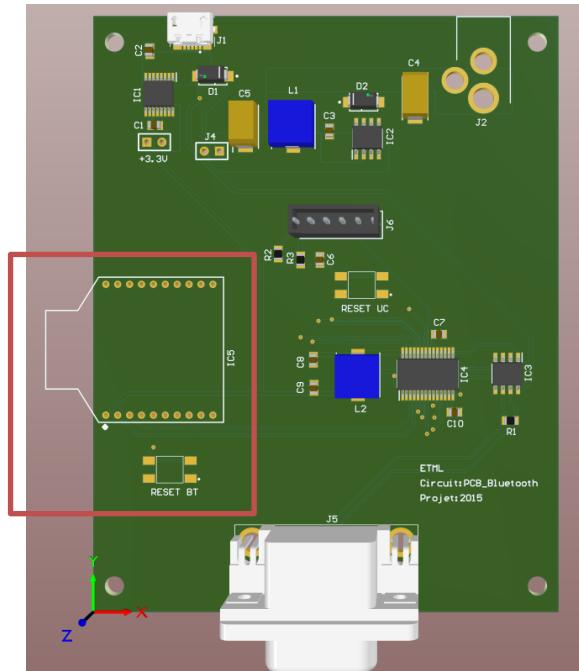
Le convertisseur RS485 - UART a été monté avec le connecteur D-SUB9 :



À cette étape, le code a été mis en place pour recevoir la trame provenant du circuit PCB_TrameRS485 et pour envoyer cette même trame au module Bluetooth.

Pour plus de précision sur le code du circuit PCB_Bluetooth veuillez consulter le chapitre 12.1.1

Le module Bluetooth a été monté une fois la bonne trame mesurée sur sa pin Tx .



10 Software

Pour établir la communication UART, M. Moreno nous a laissé le choix entre utiliser les fichiers de l'école ou d'utiliser les fichiers qu'il a développés.

J'ai décidé d'utiliser ses fichiers parce qu'ils sont complets et les fonctions peuvent utilisées sans avoir besoin de modification.

Les fichiers suivants ont été mis à disposition :

- Serial_Drv.h - gestion de la communication UART
- Serial_Drv.c - gestion de la communication UART
- Circ_Buffer.c - gestion du buffer circulaire
- Circ_Buffer.h - gestion du buffer circulaire

Voici la liste des fonctions contenues dans le fichier Serial_Drv :

- Serial_Init : permet d'initialiser la communication UART
- Serial_ReadChar : lecture d'un caractère
- Serial_ReadString : lecture d'une chaîne de caractère
- Serial_WriteChar : écriture d'un caractère
- Serial_WriteString : écriture d'une chaîne de caractère

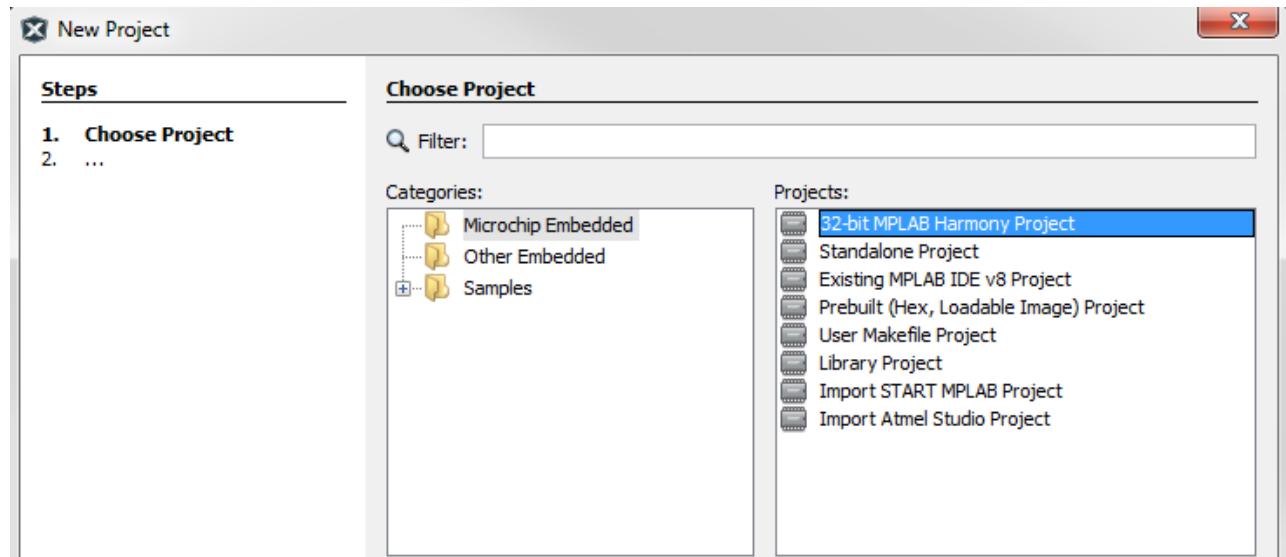
Les fichiers Circ_Buffer sont utilisés pour gérer le buffer de réception et d'émission.

10.1 PCB_trameRS485

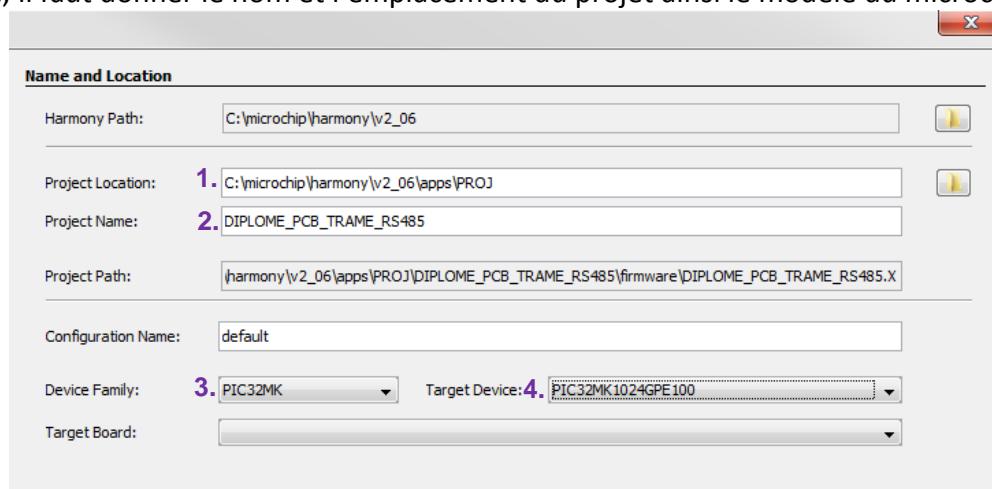
10.1.1 Création de projet

Une fois la lecture de la signature du microcontrôleur effectuée, le projet a été créé avec MPLABX IDE. Voici les différentes étapes lors de sa création :

Pour créer le projet, il faut choisir File → New Project. Comme type de projet, j'ai choisi « 32-bit MPLAB Harmony Project » :



Par la suite, il faut donner le nom et l'emplacement du projet ainsi le modèle du microcontrôleur :



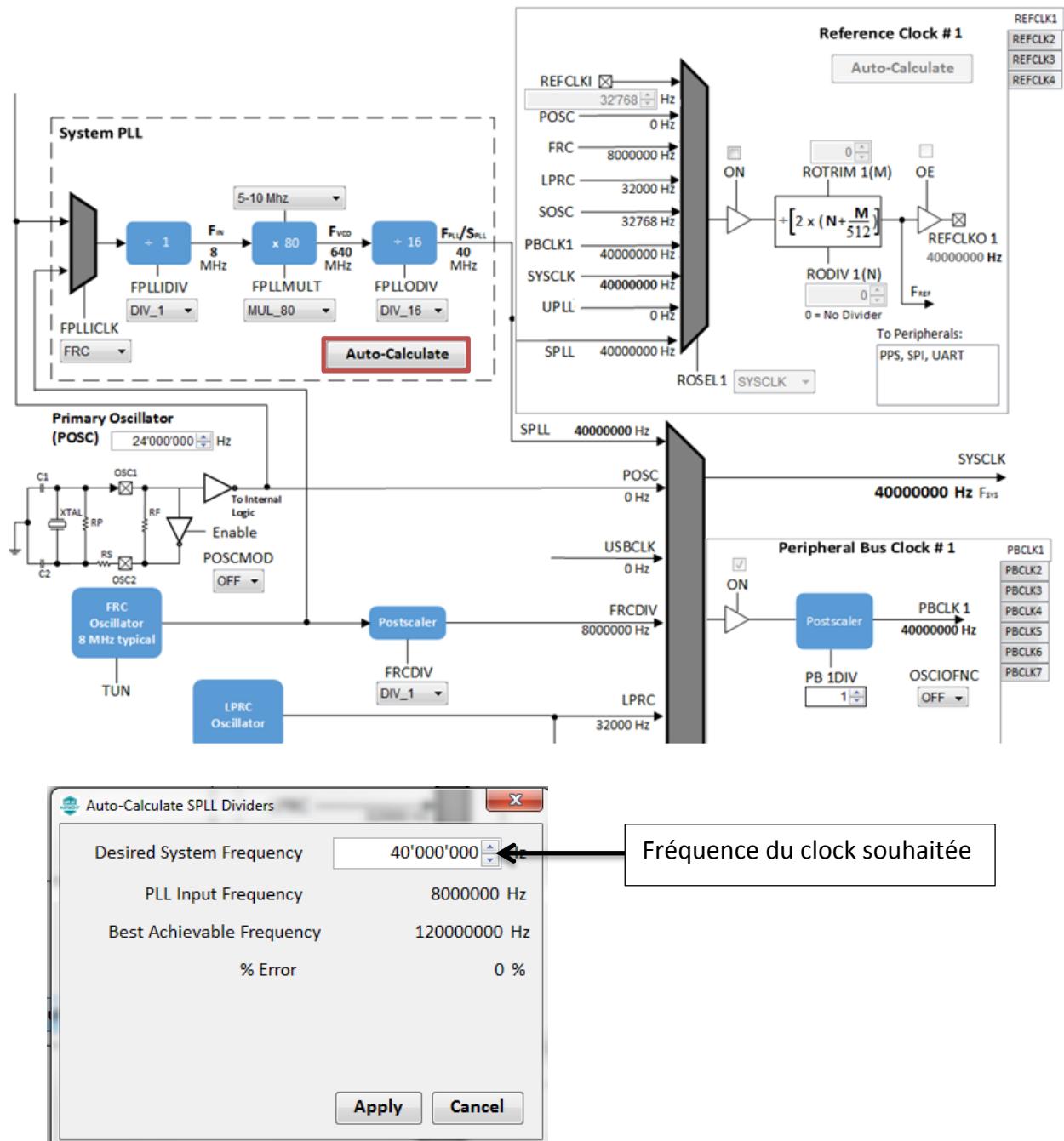
1. Emplacement du projet : C:\Microchip\harmony\v2_06\apps\PROJ
2. Nom du projet : DIPLOME_PCB_TRAME_RS485
3. La famille du microcontrôleur : PIC32MK
4. Le modèle du microcontrôleur : est PIC32MK1024GPE0100

10.1.2 Configurations dans l'Harmony

10.1.2.1 Configuration du clock

En utilisant le clock interne, ce microcontrôleur peut avoir une fréquence d'horloge maximale de 120MHz. Comme c'est relativement rapide j'ai décidé de configurer l'horloge interne à 40MHz.

Pour le faire, l'option « Auto-Calculate » a été utilisée :



10.1.2.2 Configuration des timers

J'ai décidé d'utiliser le Timer 1 pour les comptages de base.

Calcul de la valeur de recharge du Timer 1 :

Période voulue = 200ms

F_{sys} = 40MHz → T_{sys} = 25ns

- Est-ce que nous avons besoin d'un facteur de division ?

$$\text{valeur de recharge} = \left(\frac{T_{voulue}}{T_{sys}} \right) - 1$$

$$\text{valeur de recharge} = \left(\frac{200 * 10^{-3}}{25 * 10^{-9}} \right) - 1 = 7'999'999$$

La valeur de recharge est > 65'535 donc il faut mettre en place un facteur de division.

- Recherche du facteur de division

$$\text{facteur de division} = \frac{\text{valeur de recharge (trouvée ci-dessus)}}{65'535}$$

$$\text{prescaler} = \frac{15'999'999}{65'535} = 122$$

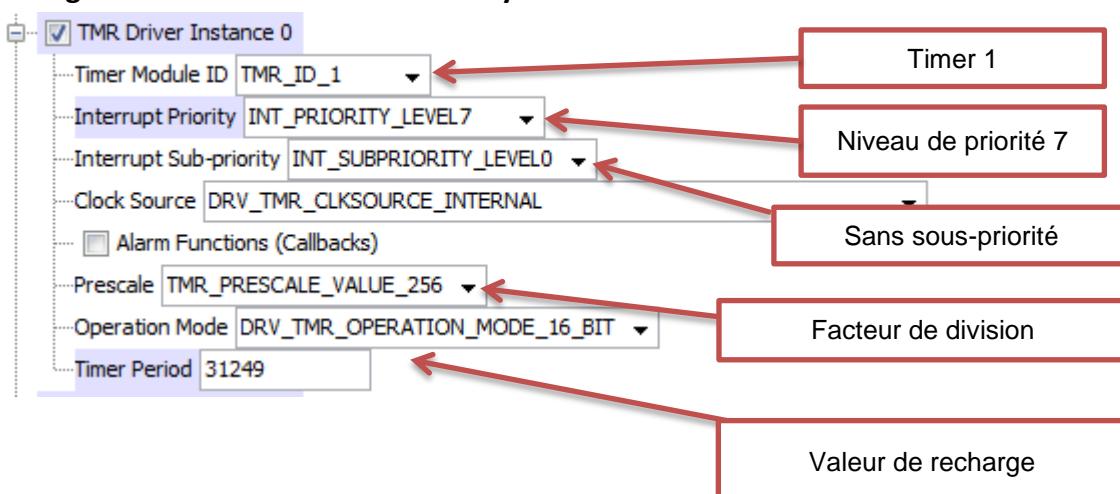
Je vais utiliser donc un facteur de division de 256.

- Valeur de recharge devient donc

$$\text{valeur de recharge} = \left(\frac{T_{voulue}}{T_{sys} * \text{prescaler}} \right) - 1$$

$$\text{valeur de recharge} = \left(\frac{200 * 10^{-3}}{25 * 10^{-9} * 256} \right) - 1 = 31'249$$

Configuration du timer dans l'Harmony



Timer 2

Le timer 2 est utilisé pour les IC (input capture), afin de lire le capteur de débit d'essence et les RPM.

La valeur de recharge sera donc calculée en prenant en compte la valeur minimale des RPM. Je pars du principe que la valeur minimale est de 150 [tours/minute], ce qui correspond à une fréquence de 2.5Hz, donc une période de 400ms.

Calcul de la valeur de recharge du Timer 2 :

Période voulue = 4000ms

F_{sys} = 40MHz → T_{sys} = 25ns

- Est-ce que nous avons besoin d'un facteur de division

$$\text{valeur de recharge} = \left(\frac{T_{voulue}}{T_{sys}} \right) - 1$$

$$\text{valeur de recharge} = \left(\frac{400 * 10^{-3}}{25 * 10^{-9}} \right) - 1 = 15'999'999$$

La valeur de recharge sans facteur de division est > 65'535 donc il faut mettre en place un facteur de division.

- Recherche du facteur de division

$$\text{facteur de division} = \frac{\text{valeur de recharge (trouvée ci-dessus)}}{65'535}$$

$$\text{prescaler} = \frac{15'999'999}{65'535} = 244$$

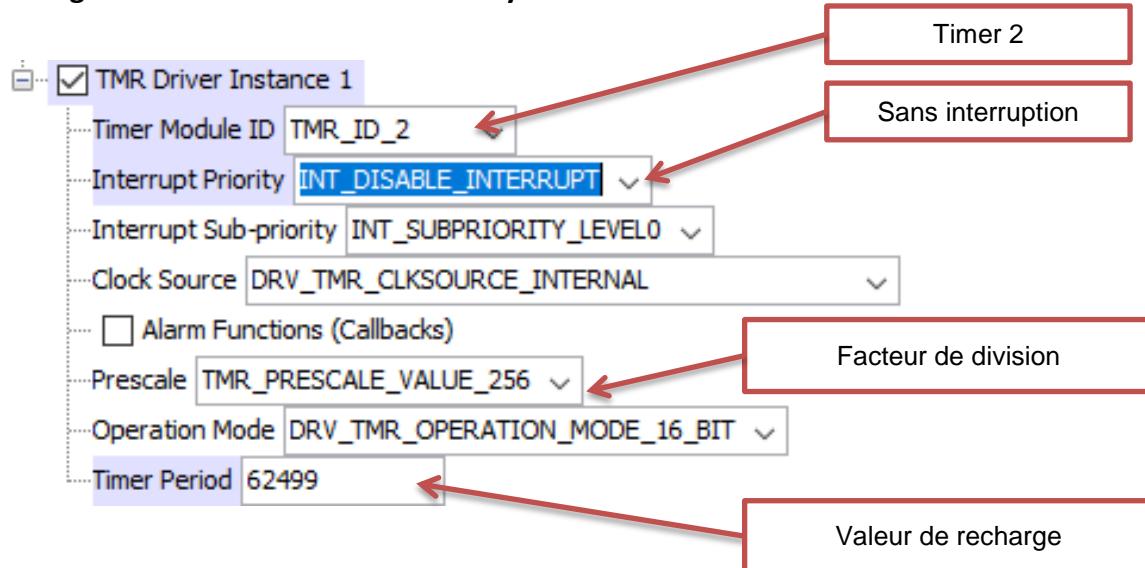
Je vais utiliser donc un facteur de division de 256.

- Valeur de recharge devient donc

$$\text{valeur de recharge} = \left(\frac{T_{voulue}}{T_{sys} * \text{prescaler}} \right) - 1$$

$$\text{valeur de recharge} = \left(\frac{400 * 10^{-3}}{25 * 10^{-9} * 256} \right) - 1 = 62'499$$

Configuration du timer dans l'Harmony



10.1.2.3 Configuration des ADC

Le PIC32MK est équipé d'un module ADC 12-bits. Voici les caractéristiques de ce convertisseur :

- L'échantillonnage est déclenché par un évènement comme : timer, software, interruptions.
- Il existe trois classes d'entrées analogiques :
 - Classe 1 : Entrées AN0 – AN5 ; AN6-AN9 et AN24-AN26
 - Classe 2 : Entrées AN6-AN27, multiplexées sur le canal 7
 - Classe 3 : Entrées AN47-ANA53, multiplexées avec la classe 2, mais seulement en mode 'Input scan' (tous déclenchés par le même évènement)

Entrés ADC utilisé :

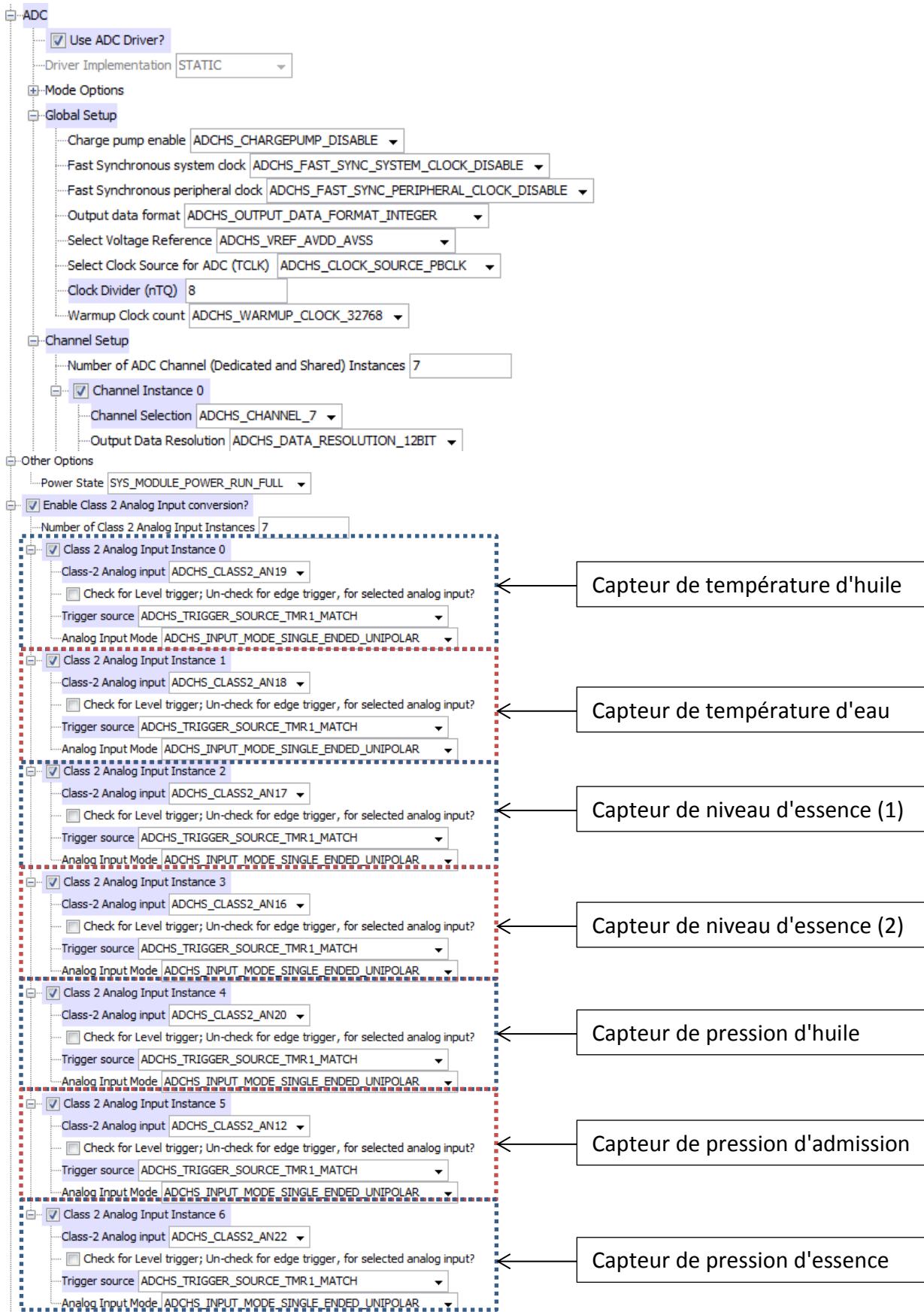
Pin du microcontrôleur	Entrée ADC	Classe ADC	Description
10	AN19	Classe 2	Température de l'huile
11	AN18	Classe 2	Température de l'eau
12	AN17	Classe 2	Niveau d'essence 1
14	AN16	Classe 2	Niveau d'essence 2
17	AN22	Classe 2	Pression d'essence
18	AN20	Classe 2	Pression d'huile 1
41	AN12	Classe 2	Pression d'admission
67	AN47	Classe 3	Tension de la batterie

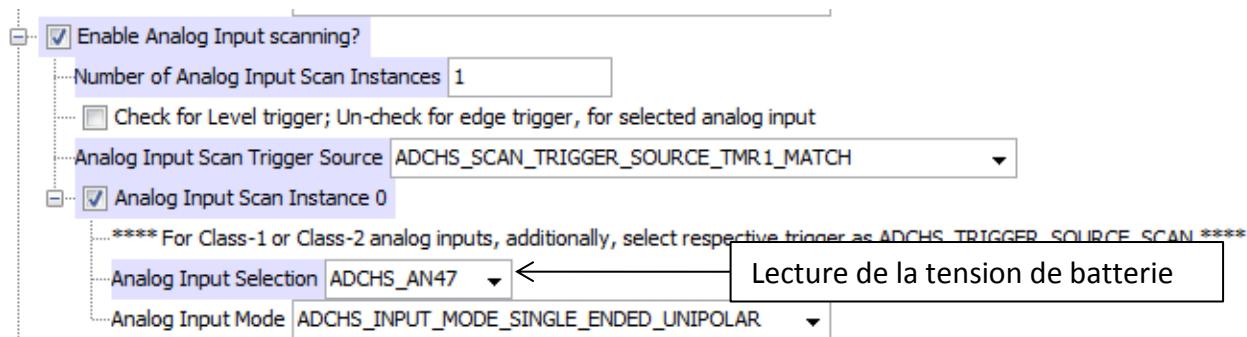
L'échantillonnage (sampling) est déclenché par le timer 1.

Configuration des ADC dans l'Harmony

La configuration de la page suivante est même comme pour le projet de semestre 1923 – EngineIndicationUlm.

L'entrée analogique AN47 a été rajoutée pendant le travail de diplôme afin de lire la tension de la batterie.





10.1.2.4 Configuration du SPI

La configuration du SPI est la même comme pour le projet de semestre 1923 – EngineIndicationUlm.

Entrées SPI utilisées :



SPI Driver Instance 1

- SPI Module ID: SPI_ID_3
- Driver Mode
- Master\Slave Mode
- Data Width
- Buffer Mode
 - Allow Idle Run
- Protocol Type: DRV_SPI_PROTOCOL_TYPE_STANDARD
- Baud Clock Source: SPI_BAUD_RATE_PBCLK_CLOCK
- Clock\Baud Rate - Hz: 1000000
- Clock Mode: DRV_SPI_CLOCK_MODE_IDLE_LOW_EDGE_RISE
- Input Phase: SPI_INPUT_SAMPLING_PHASE_IN_MIDDLE
- Dummy Byte Value: 0xFF
- Max Jobs In Queue: 10
- Minimum Number Of Job Queue Reserved For Instance: 1

SPI Driver Instance 2

- SPI Module ID: SPI_ID_5
- Driver Mode
- Master\Slave Mode
- Data Width
- Buffer Mode
 - Allow Idle Run
- Protocol Type: DRV_SPI_PROTOCOL_TYPE_STANDARD
- Baud Clock Source: SPI_BAUD_RATE_PBCLK_CLOCK
- Clock\Baud Rate - Hz: 1000000
- Clock Mode: DRV_SPI_CLOCK_MODE_IDLE_LOW_EDGE_RISE
- Input Phase: SPI_INPUT_SAMPLING_PHASE_IN_MIDDLE
- Dummy Byte Value: 0xFF
- Max Jobs In Queue: 10
- Minimum Number Of Job Queue Reserved For Instance: 1

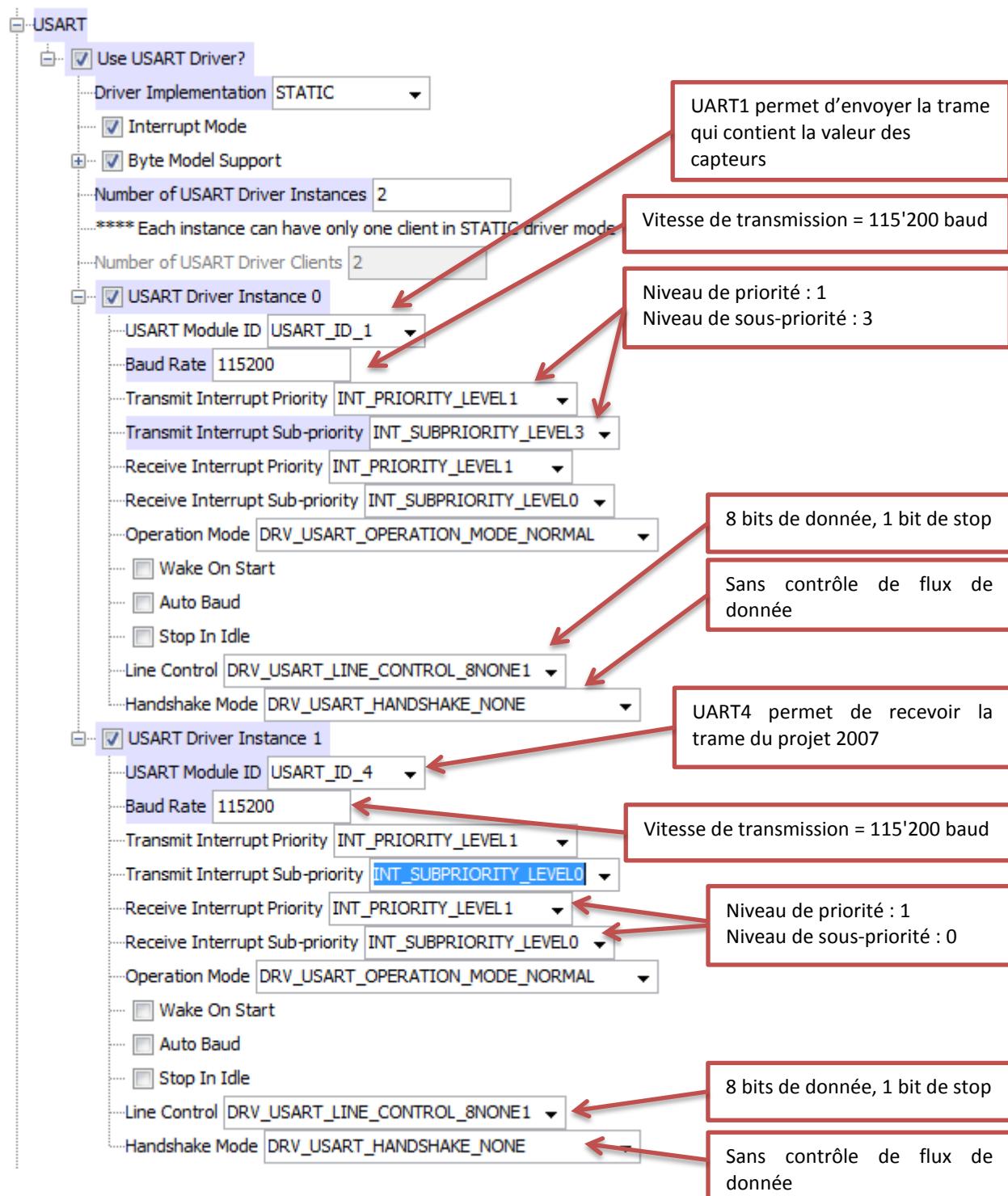
Capteur de température des gaz d'échappement

Lecture de température des capteurs supplémentaires

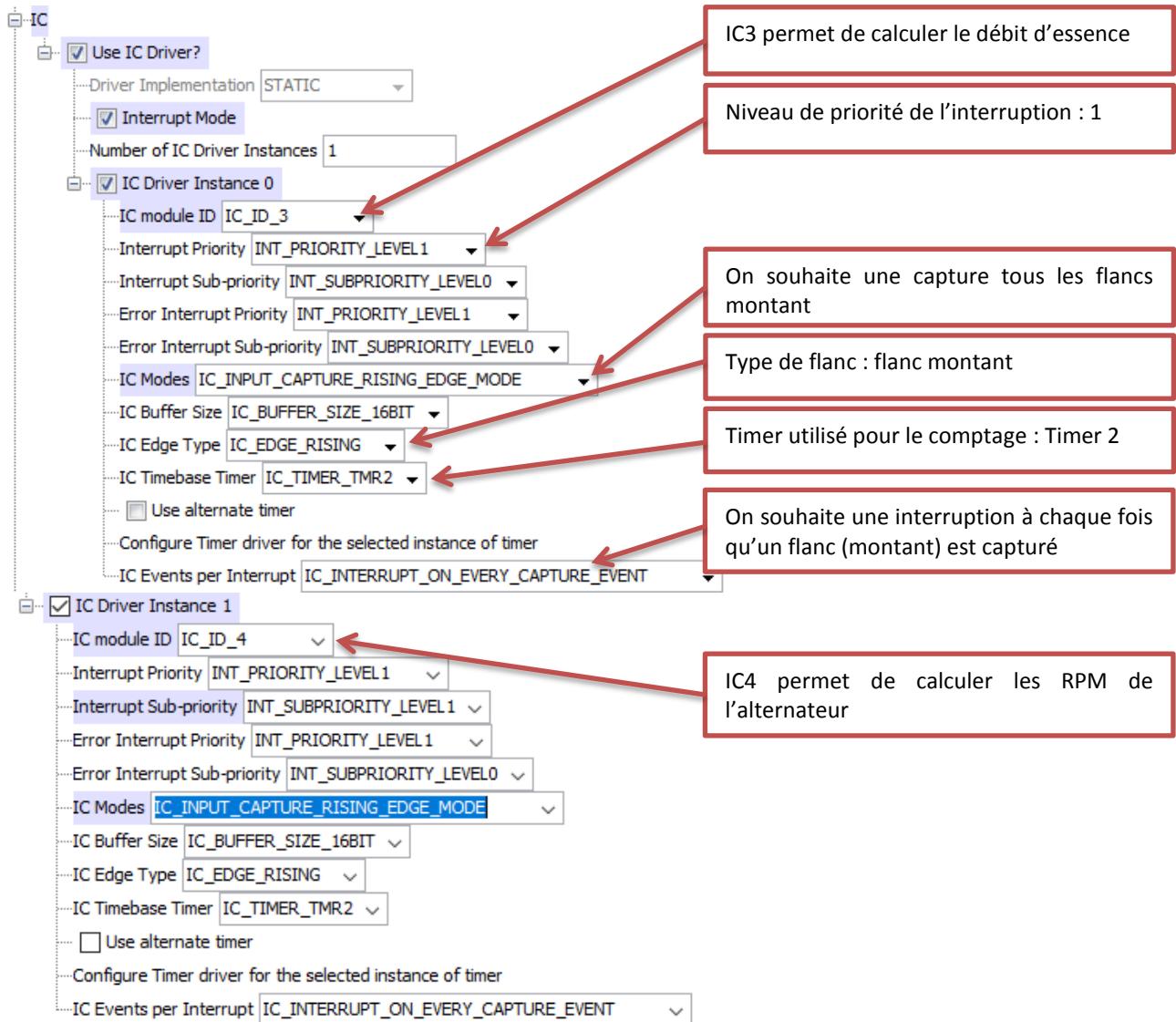
10.1.2.5 Configuration de l'UART

Pour envoyer la trame UART, j'ai décidé d'utiliser la vitesse de transmission de 115'200 Baud, sans contrôle de flux de donnée.

La même configuration est gardée pour recevoir la trame UART qui contient la valeur des RPM du projet 2007 :



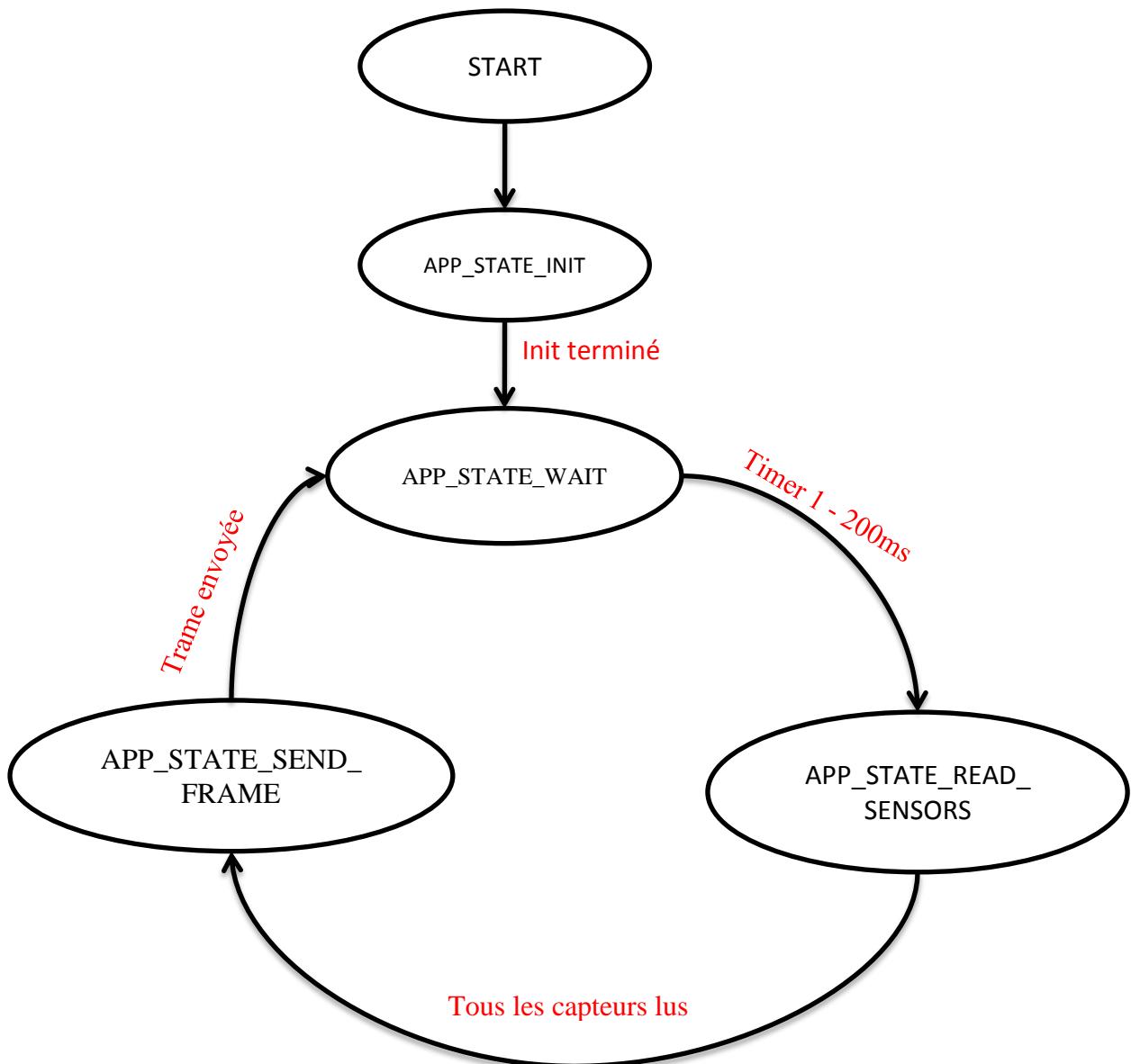
10.1.2.6 Configuration de l'input capture



La configuration de l'input capture 4 est la même comme pour l'IC3.

10.1.3 Machine d'état

Voici la machine une représentation de la machine d'état:



- **APP_STATE_INIT** : Dans cet état, les capteurs seront initialisés. Une fois terminé, le programme va rentrer dans **APP_STATE_WAIT**.
- Le Timer 1 est configuré afin de déclencher une interruption toutes les 200ms. Dans cet état, nous allons lire tous les capteurs ainsi la trame UART contenant des RPM (projet 2007).
- Une fois la lecture de tous les capteurs terminés, la trame RS485 est envoyée.
- Pour finir, une fois la trame envoyée, le programme revient dans **APP_STATE_WAIT** et reste jusqu'à la prochaine interruption.

10.1.4 Réception de la trame UART

La trame contenant les RPM du projet 2007 est envoyée toutes les 10ms.

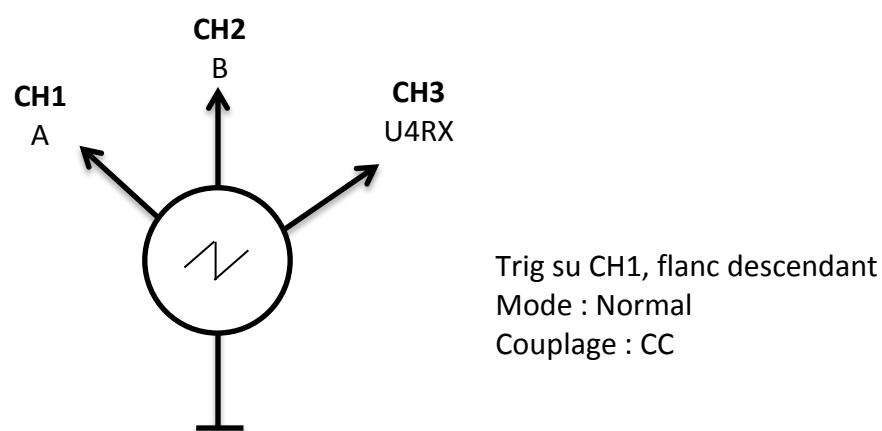
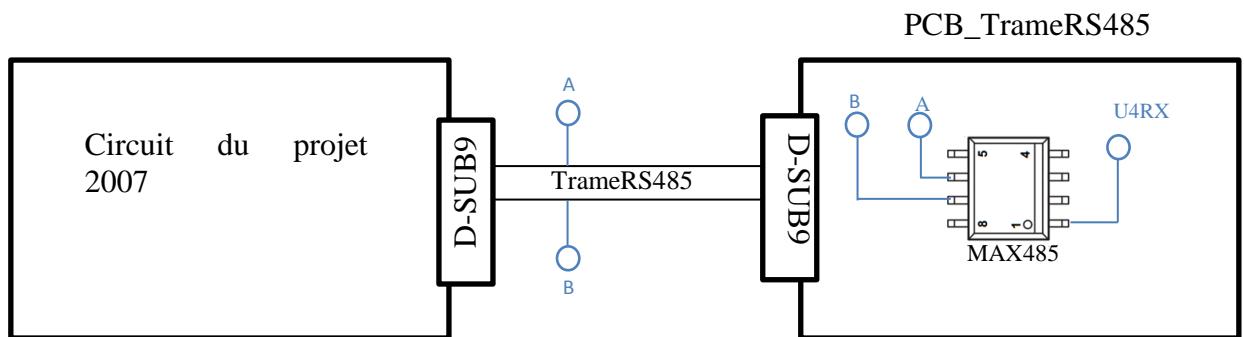
La trame a le format identique à celle de la ligne principale :

START::RPM=XXXX::STOP#

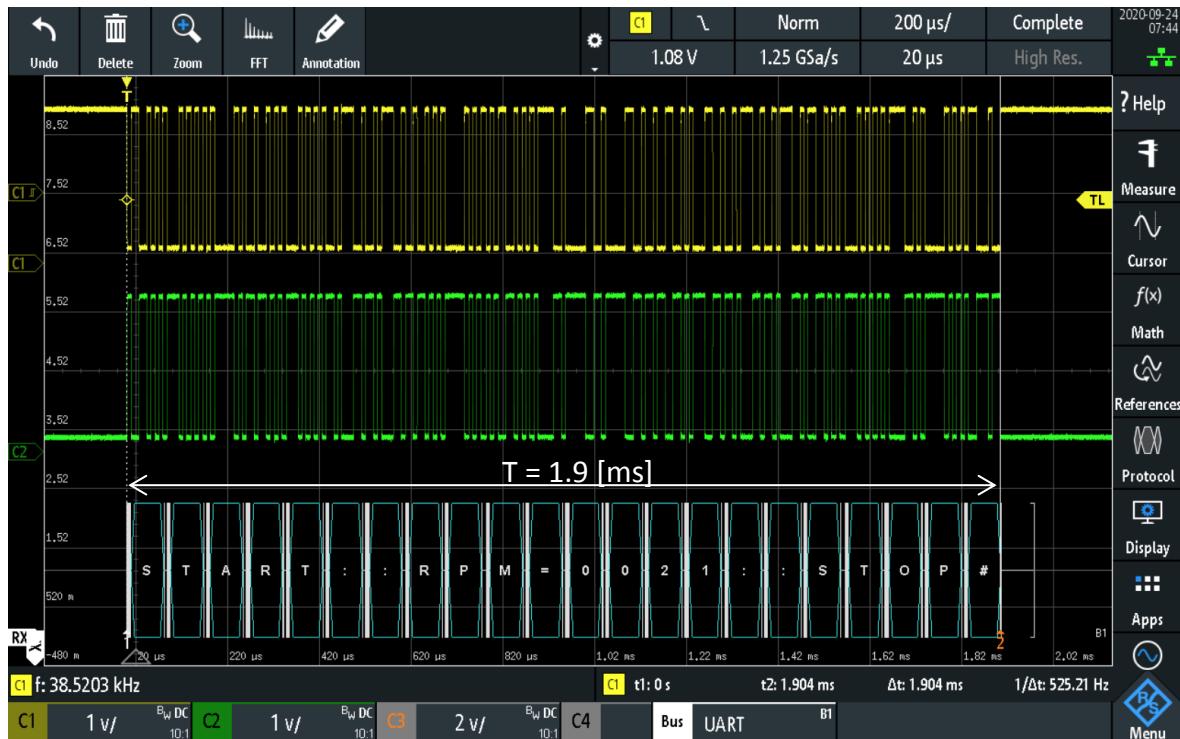
Le "#" a été rajouté afin de définir la fin de la trame.

Dans un premier temps, j'ai décidé de faire une mesure de la trame du projet 2007.

Schéma de mesure :



Résultats de mesure :



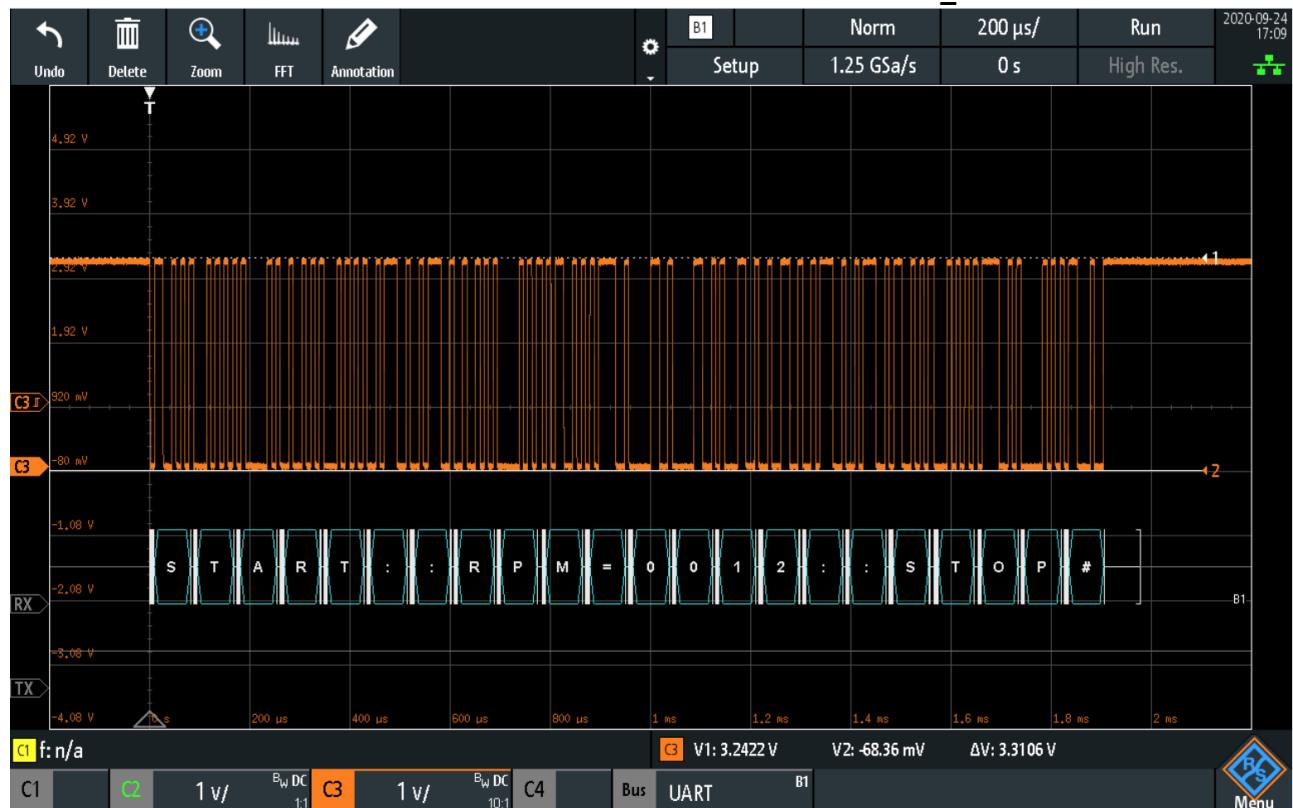
La vitesse de transmission de la trame est de 115'200 baud, donc la durée d'un bit est de 8.68μs.
Pour envoyer un caractère, 10 bits ont été utilisés (START, 8 bits de données, STOP) et au total 22 caractères ont été reçus.

Calcul théorique de la durée de la trame :

*La durée de la trame = durée d'un bit * nombre de bits (un caractère) * nombre de car.*
*La durée de la trame = 8.68μs * 10 * 22 = 1.90ms*

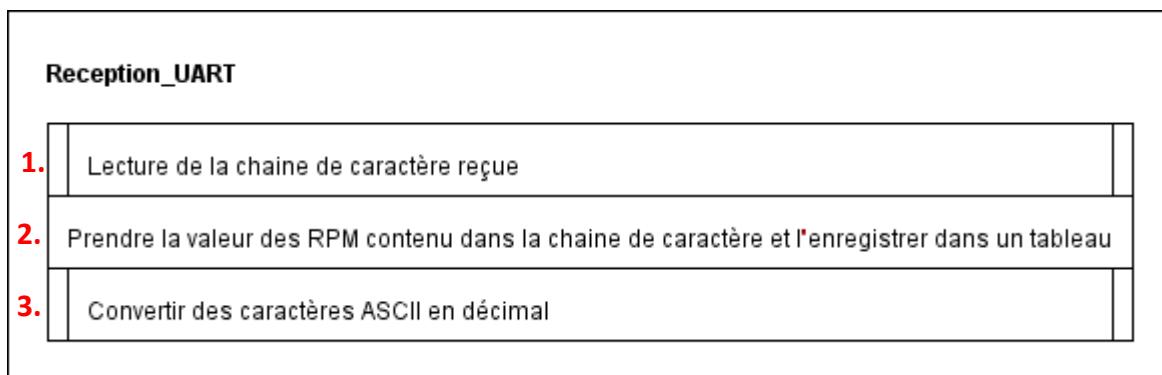
En regardant la première mesure, nous pouvons constater que la durée de la trame est bien de 1.90ms, ce qui correspond bien au calcul théorique.

Cette trame est envoyée tous les 10ms.

Mesure de la trame sur l'entrée Rx du microcontrôleur de la carte PCB_TrameRS485 :

Cette mesure a été faite afin de s'assurer que le signal a une amplitude entre 0 et 3.3V et qu'on retrouve les bons caractères sur la ligne Rx.

Avec cette mesure, nous pouvons voir que c'est le cas, donc nous pouvons procéder à la lecture de la trame.

Structogramme

- Pour lire une chaîne de caractère, nous avons à disposition la fonction "Serial_ReadString" contenue dans le fichier "Serial_Drv.c".

2. Grâce à la fonction expliquée précédemment, la chaîne de caractère est sauvegardée dans un tableau. Nous allons donc prendre la valeur de RPM :

Tableau [0]	S
Tableau [2]	T
Tableau [2]	A
Tableau [3]	R
Tableau [4]	T
Tableau [5]	:
Tableau [6]	:
Tableau [7]	R
Tableau [8]	P
Tableau [9]	M
Tableau [10]	=
Tableau [11]	X
Tableau [12]	X
Tableau [13]	X
Tableau [14]	X

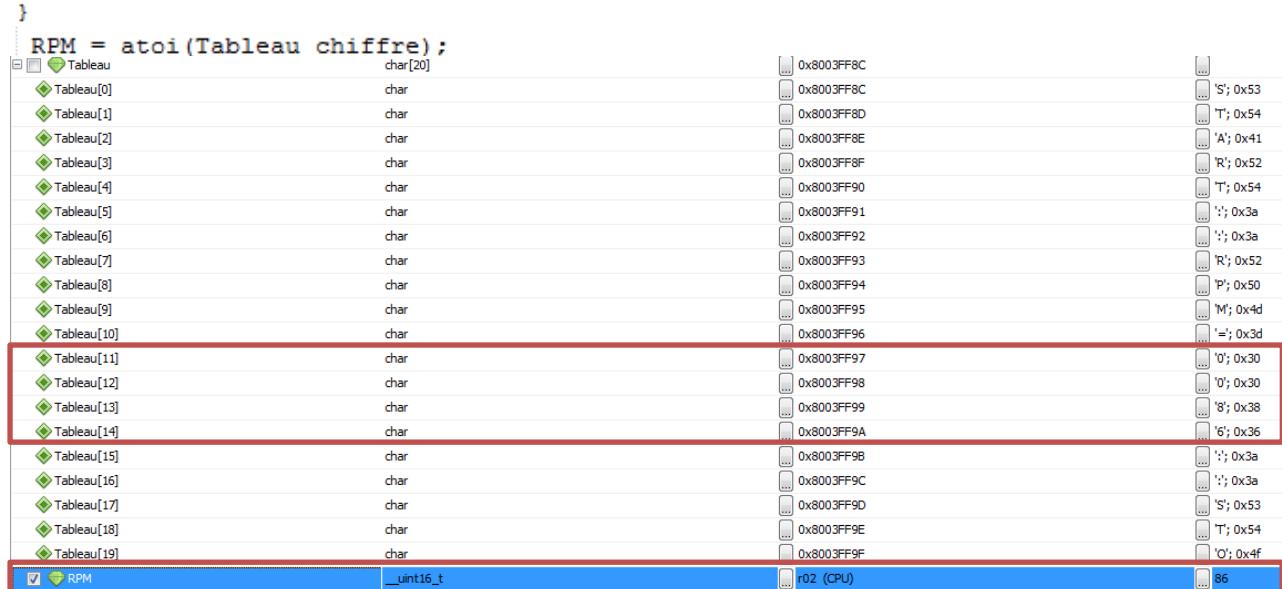
La valeur des RPM est contenue dans les cases 11 à 14. Nous allons donc enregistrer ces valeurs dans un autre tableau (Tableau_chiffre).

3. Pour convertir la chaîne de caractère en décimal, la fonction "atoi" peut être utilisée

Code :

```
Serial_ReadString(&driver1, &Tableau, '#');

for (compt = 0; compt <= 4; compt++)
{
    Tableau_chiffre[compt] = Tableau[compt+11];
}
```



Nous pouvons voir que dans la variable RPM nous retrouvons la valeur décimale des tours par minute et qu'elle correspond bien à la valeur reçue.

10.1.4.1 Calcul des RPM de l'alternateur

Calcul mathématique permettant d'obtenir les RPM :

1. $RPM = \frac{f_{RPM}}{60} = \frac{1}{60 * Période\ RPM}$
2. $Période\ tick = Nouveau\ flanc - Ancien\ flanc [tick]$
3. $tick = \frac{1}{F_{sys}} * facteur\ de\ division = \frac{facteur\ de\ division}{F_{sys}}$
4. $Période\ RPM = Période\ tick * tick$

$$Période\ RPM = Nouveau\ flanc - Ancien\ flanc * \frac{facteur\ de\ division}{F_{sys}}$$

$$1. RPM = \frac{1}{Période\ RPM * 60} = \frac{1}{Nouveau\ flanc - Ancien\ flanc * \frac{facteur\ de\ division}{F_{sys}} * 60}$$

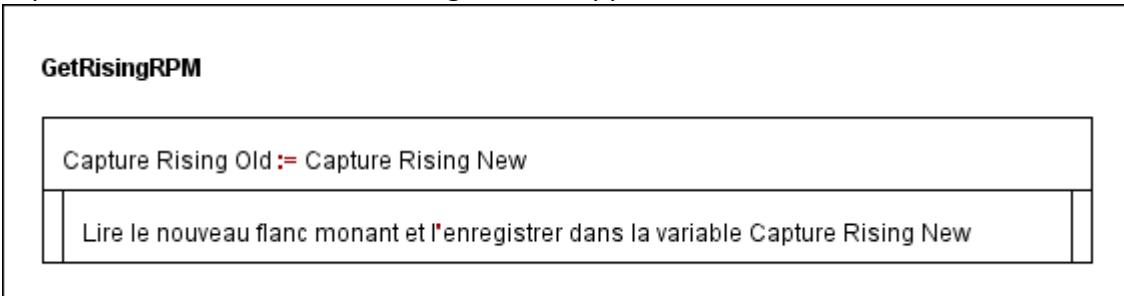
Explication du calcul :

1. Les RPM (rotation per minute) représentent le nombre de rotations du moteur en une minute.
2. Le signal de l'alternateur est connecté sur l'entrée IC (input capture) du microcontrôleur. Ce dernier est paramétré afin de déclencher une interruption lors de chaque flanc montant. Nous allons donc calculer la différence de temps entre les deux flancs montants. Ce temps est exprimé en nombre de tick.
3. Pour trouver la valeur d'un tick, il faut prendre en compte la fréquence du système ainsi le facteur de division paramétré dans l'Harmony.
4. Pour trouver la période des RPM, il faut donc prendre la période du tick et multiplier par le nombre de tick.
5. Finalement, pour trouver le RPM, il faut faire $1/(la\ période\ des\ RPM)$ pour obtenir la fréquence de rotation et multiplier par 60 afin d'obtenir les tours par minute.

Structogramme :

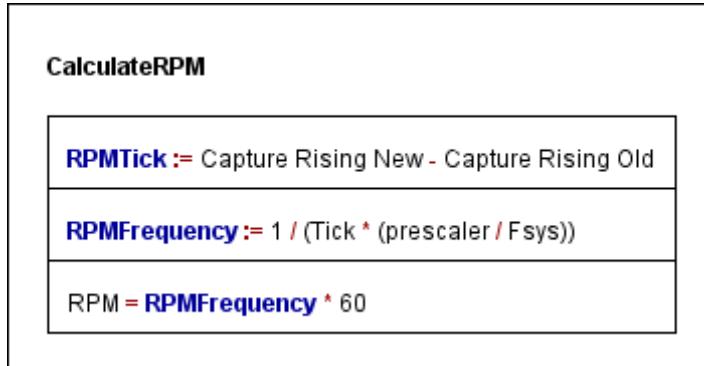
Pour calculer les RPM, les fonctions suivantes ont été utilisées :

Input Capture est paramétré afin d'avoir une interruption lors de chaque flanc montant. Quand l'interruption a lieu, la fonction GetRisingRPM est appelée.



Cette fonction permet de capturer deux flancs montants : Capture Rising New et Capture Rising Old.

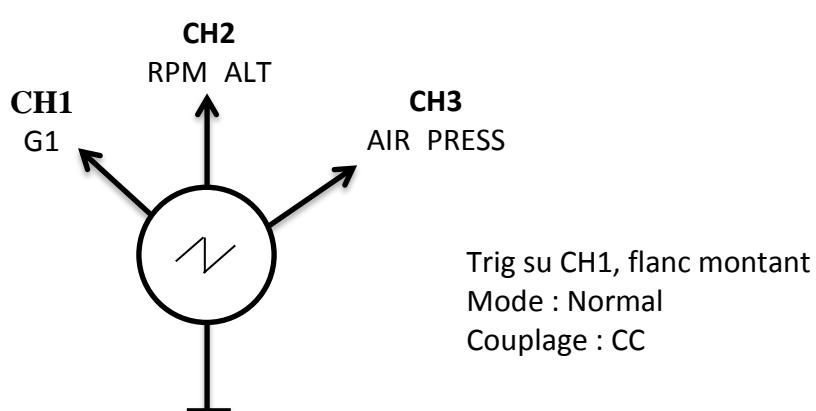
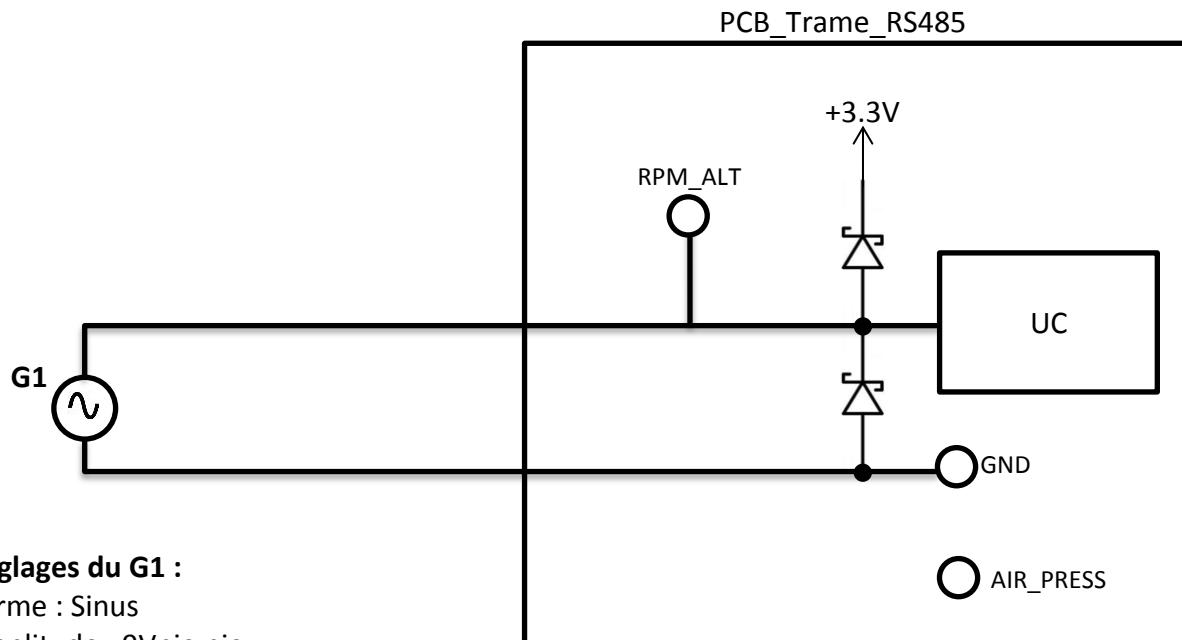
Ensuite, la fonction CalculateRPM permettra de faire les calculs mathématiques afin de trouver les tours par minute :



Les calculs ci-dessus ont été expliqués à la page précédente.

Test fonctionnel :

Schéma de mesure :



Pour simuler le signal de l'alternateur, j'ai utilisé un générateur de fonction. Une amplitude de 7V pic - pic a été choisie pour voir si les diodes vont bien protéger le microcontrôleur et la fréquence de 200Hz a été choisie pour simuler 12'000 tours par minute.

Signal du générateur de fonction :



Avec cette mesure, nous pouvons voir que le signal du générateur de fonction a une amplitude de 7V pic - pic et que la fréquence est de 200Hz.



Sur cette image, nous constatons que les diodes Schottky ne laissent pas passer l'alternance négative et protègent le microcontrôleur contre les surtensions. Si on continu d'augmenter la tension du générateur de fonction, la tension reste bloqué à 3.7V et le sinus "s'aplatit". C'est une tension légèrement élevée sachant que le microcontrôleur supporte une tension maximale de $V_{cc} + 0.3\text{V}$ donc 3.6V. Idéalement, il faudra utiliser la pin du microcontrôleur qui supporte 5V.

Pour faire les mesures, je vais m'assurer que la tension à l'entrée du microcontrôleur ne dépasse pas 3.6V .

J'ai fait « toogle » de la pin "AIR_PRESS" lorsque l'interruption d'input capture a lieu. Nous pouvons voir qu'elle se produit bien à chaque flanc montant.

Code :

```

uint16_t CalculateRPM(void)
{
    uint16_t RPM_alternator;

    //Calculer le nombre de ticks
    if (CaptureRPM.Rising_New >= CaptureRPM.Rising_Old)
    {
        CaptureRPM.Number_Ticks = CaptureRPM.Rising_New - CaptureRPM.Rising_Old;
    }
    else
    {
        CaptureRPM.Number_Ticks = 62500-CaptureRPM.Rising_Old + CaptureRPM.Rising_New + 1;
    }

    // Protection contre division par 0
    if( CaptureRPM.Number_Ticks != 0)
    {
        //Calculer la fréquence de rotation
        CaptureRPM.Value = (1/( CaptureRPM.Number_Ticks * 0.0000064));

        //Fréquence de rotation * 60 = RPM
        //On fait /10 pour exprimer la valeur en dixième de RPM
        RPM_alternator = (CaptureRPM.Value * 60)/10;
    }

    //Retourner la valeur calculée
    return (RPM_alternator);
}

```

En faisant du pas à pas, le programme a été arrêté dans la fonction "CalculateRPM" et voici le contenu dans les différentes variables :

Variables				
	Name	Type	Address	Value
<input checked="" type="checkbox"/>	CaptureRPM	IC_RPM_ALT_S	0x800003B0	
<input checked="" type="checkbox"/>	Rising_Old	_uint16_t	0x800003B0	30222
<input checked="" type="checkbox"/>	Rising_New	_uint16_t	0x800003B2	31001
<input checked="" type="checkbox"/>	Number_Ticks	_uint16_t	0x800003B4	779
<input checked="" type="checkbox"/>	Value	_uint16_t	0x800003B6	200
<input type="checkbox"/>	<Enter new watch>			
<input checked="" type="checkbox"/>	RPM_alternator	_uint16_t	0x8003FF68	1200

Nous pouvons voir que les flancs ont bien été capturés et. La valeur est exprimée en 1/10 de RPM, c'est pour cette raison qu'on retrouve 1200.

10.1.5 Capteur de débit d'essence

Pour plus de précision sur ce capteur, veillez voir le chapitre XXXX.

L'unité de mesure est exprimée en ml/pulse. Voici deux exemples permettant de faire la démonstration :

Exemple N°1 – Période mesurée = 50ms

$$\text{Débit d'essence} = \left[\frac{\text{ml}}{\text{période mesurée}} \right]$$

$$\text{Débit d'essence} = \frac{\text{ml}}{50\text{ms}} = \frac{1\text{l}}{50\text{s}}$$

Lorsque le débit d'essence augmente, la période mesurée diminue et nous sommes sensé trouver un débit plus grand :

Exemple N°2 – Période mesurée = 10ms

$$\text{Débit d'essence} = \left[\frac{\text{ml}}{\text{période de la pulse}} \right]$$

$$\text{Débit d'essence} = \frac{\text{ml}}{10\text{ms}} = \frac{1\text{l}}{10\text{s}}$$

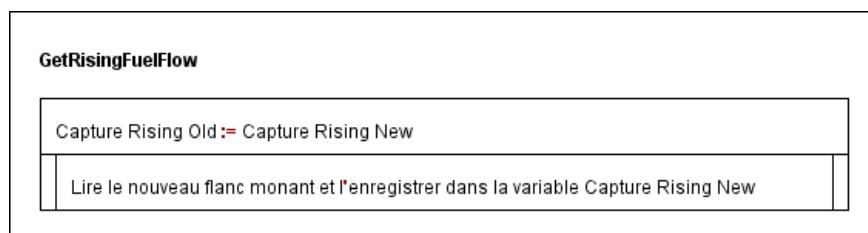
Nous constatons que le calcul correspond. Lorsque la période mesurée était de 50ms, 1 litre était consommé toutes les 50s et lorsque la période est de 10ms, 1 litre est consommé toutes les 10s.

En parlant avec le responsable du projet, nous avons décidé d'exprimer la mesure en [ml/s].

Structogramme :

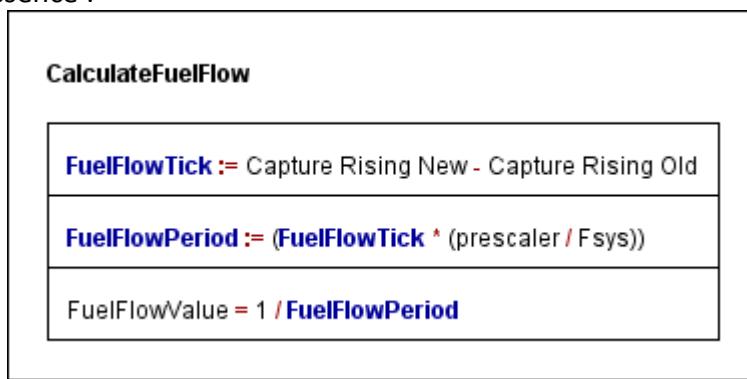
Les fonctions ci-dessous sont similaires aux fonctions permettant de récupérer les RPM.

L'Input Capture est paramétré afin d'avoir une interruption lors de chaque flanc montant. Quand l'interruption a lieu, la fonction GetRisingFuelFlow est appelée.



Cette fonction permet de capturer deux flancs montants : Capture Rising New et Capture Rising Old.

Ensuite, la fonction CalculateFuelFlow permettra de faire les calculs mathématiques afin de trouver le débit d'essence :

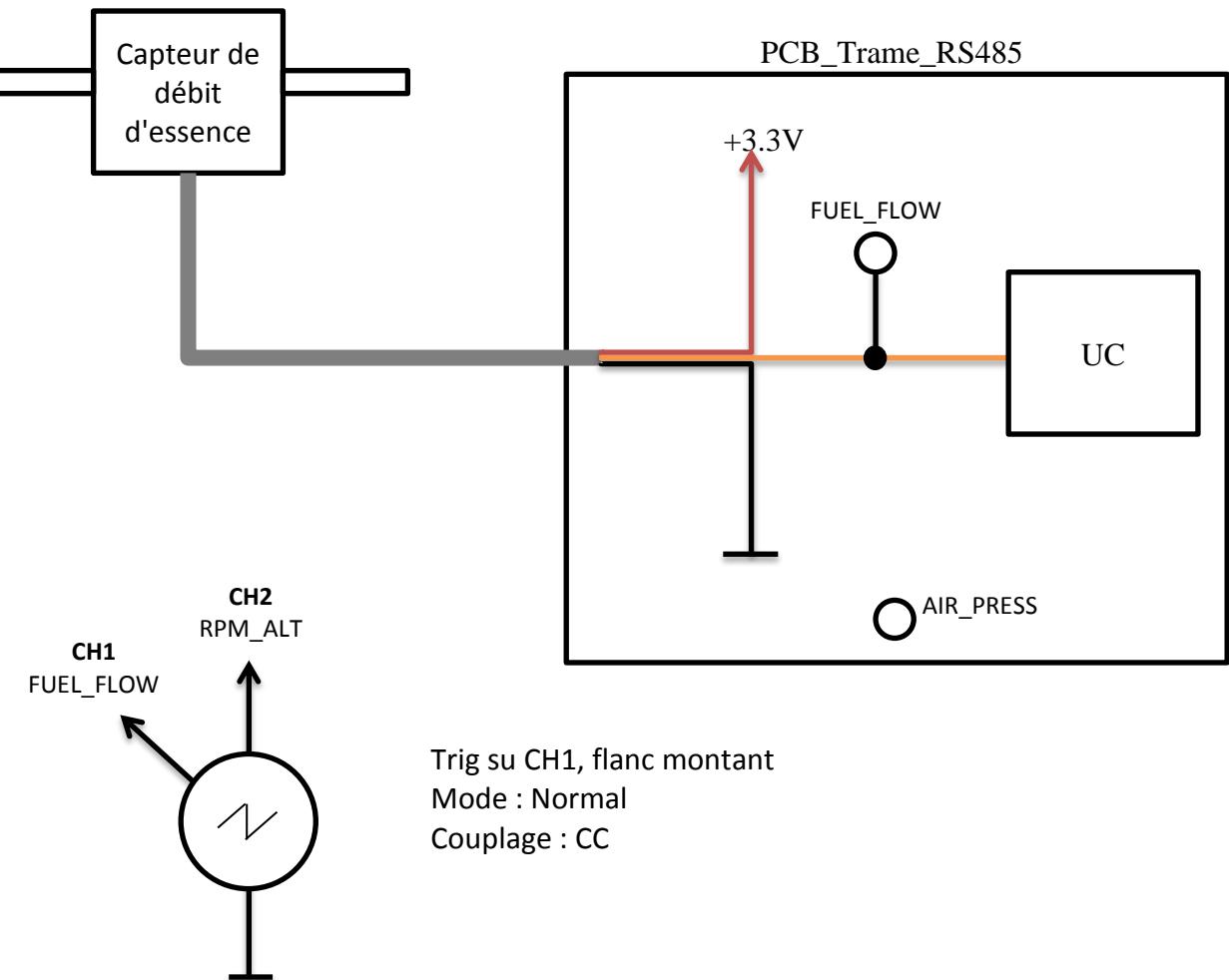


Dans la variable FuelFlowTick, nous allons enregistrer le nombre de tick entre les deux flancs montants.

Pour retrouver la période de signal, il faut prendre en compte le facteur de division du timer utilisé pour l'input capture (dans notre cas c'est le timer 2) et la fréquence du système qui est de 40Mhz.

Finalement, pour trouver la valeur de débit d'essence, il faut prendre 1 (litre) et diviser par la période mesurée afin d'exprimer la valeur en [ml/s].

Schéma de mesure :



Résultat de mesure :

Sur CH1, nous pouvons voir le signal de sortie du capteur. Dans la réponse d'interruption d'input capture, j'ai "toogle" la sortie "AIR_PRESS" et nous constatons que l'interruption a lieu à chaque flanc montant.

La période mesurée est de 24.9ms.

Code :

```
uint16_t Calculate_FUEL_FLOW(void)
{
    uint16_t Fuel_flow;

    //Calculer le nombre de ticks
    if (CaptureFuelFlow.Rising_New >= CaptureFuelFlow.Rising_Old)
    {
        CaptureFuelFlow.Number_Ticks = CaptureFuelFlow.Rising_New - CaptureFuelFlow.Rising_Old;
    }
    else
    {
        CaptureFuelFlow.Number_Ticks = RECHARGE_VALUE_TMR2-CaptureFuelFlow.Rising_Old + CaptureFuelFlow.Rising_New + 1;
    }

    // Protection contre division par 0
    if(CaptureFuelFlow.Number_Ticks != 0)
    {
        //Dabit d'essence = 1 (litre) / période mesurée
        Fuel_flow = (1/(CaptureFuelFlow.Number_Ticks * 0.0000064));
    }

    return (Fuel_flow);
}
```

	Name	Type	Address	Value
<input checked="" type="checkbox"/>	CaptureFuelFlow	IC_FUEL_FLOW_S	0x800003A8	
<input checked="" type="checkbox"/>	Rising_Old	_uint16_t	0x800003A8	26019
<input checked="" type="checkbox"/>	Rising_New	_uint16_t	0x800003AA	29919
<input checked="" type="checkbox"/>	Number_Ticks	_uint16_t	0x800003AC	3900
<input type="checkbox"/>	<Enter new watch>			
<input checked="" type="checkbox"/>	Fuel_flow	_uint16_t	0x8003FF68	40

Le nombre de ticks = 3900.

$$\text{Période RPM} = \text{Nouveau flanc} - \text{Ancien flanc} * \frac{\text{facteur de division}}{F_{\text{sys}}}$$

$$\text{Période RPM} = 3900 * \frac{256}{40M} = 24.96ms$$

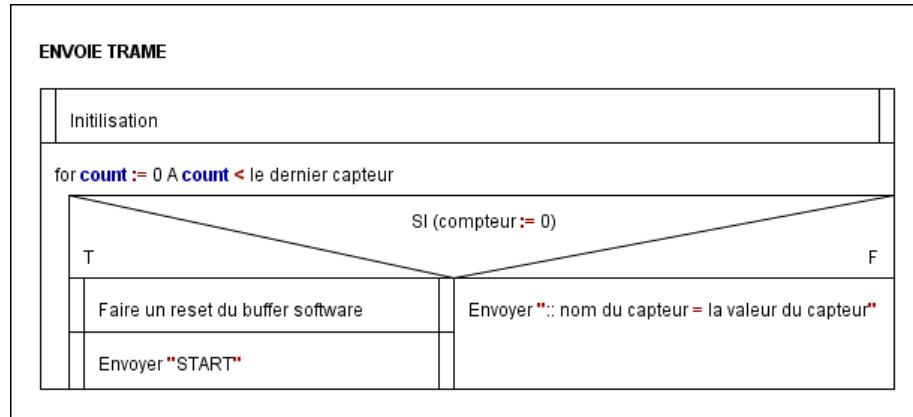
On retrouve bien la période mesurée.

$$\text{Débit d'essence} = \left[\frac{ml}{25ms} \right] = \frac{1l}{25s} = 40 \left[\frac{ml}{s} \right]$$

10.1.6 Envoi de la trame UART

Structogramme

Voici l'idée de départ permettant d'envoyer la trame UART contenant la valeur de tous les capteurs :



Le principe consiste à utiliser un compteur qui va compter de 0 jusqu'au dernier capteur.

Si le compteur vaut 0, cela veut dire que c'est le début de message et j'ai décidé de vider le buffer par précaution. Une fois le buffer vidé, un "START" est envoyé pour indiquer le début de message. Ensuite, nous allons **envoyer :: le nom de chaque capteur = la valeur du capteur**.

Comme nous avons beaucoup de capteurs, dans le code j'ai utilisé deux boucles for :

```

for (count1 = 0; count1<E_t_exh_last;count1++)
{
    if(count1==0)
    {
        CircBuffer_Reset(&driver_send_frame.txBuffer);
        Serial_WriteChar(&driver_send_frame,'S');
        Serial_WriteChar(&driver_send_frame,'T');
        Serial_WriteChar(&driver_send_frame,'A');
        Serial_WriteChar(&driver_send_frame,'R');
        Serial_WriteChar(&driver_send_frame,'T');

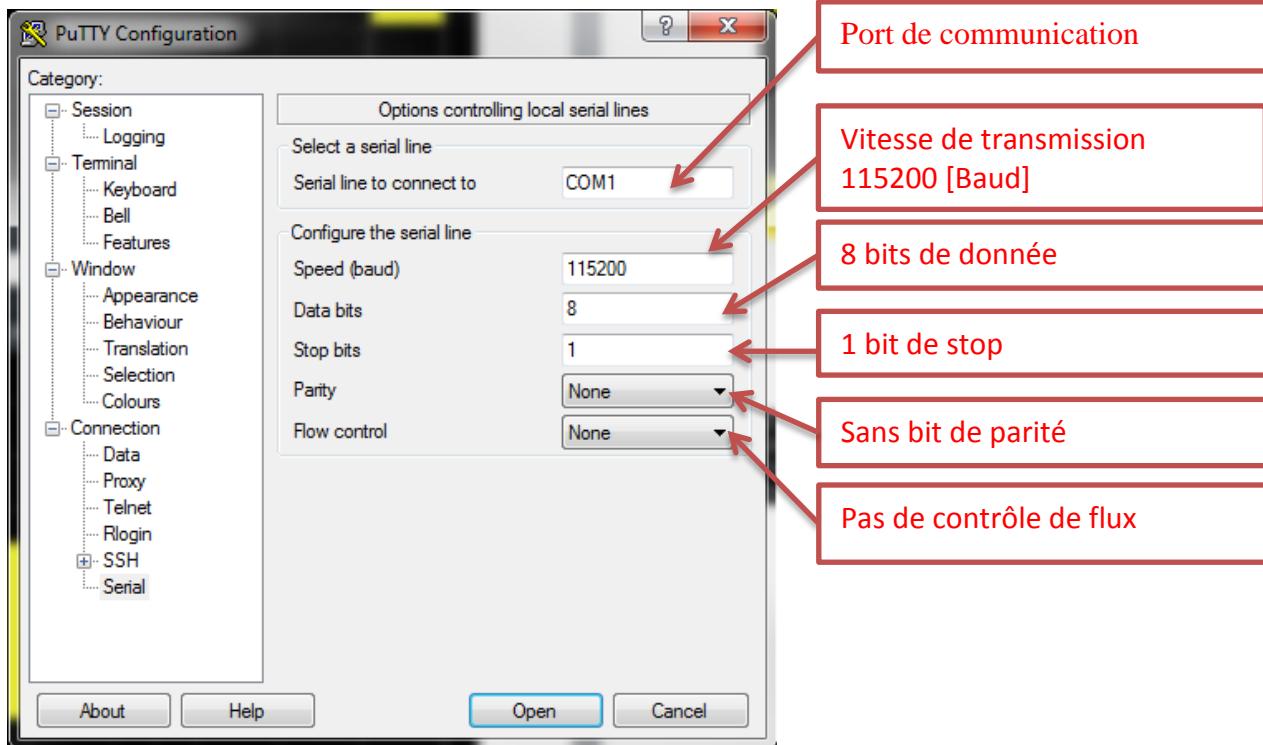
    }
    else if((count1>0) && (count1<=E_tech_8))
    {
        sprintf(str_send_frame_1, "::%s=%03d", &DATA_SENSOR_CYL_EXH[count1],VAL_SENSOR_CYL_EXH[count1]);
        while(!Serial_WriteString(&driver_send_frame,str_send_frame_1));
    }
}

for (count2 = 0; count2<E_other_sensor_last;count2++)
{
    if(count2<=E_tension_batt)
    {
        sprintf(str_send_frame_2, "::%s=%03d", &DATA_OTHER_SENSOR[count2],VAL_OTHER_SENSOR[count2]);
        while(!Serial_WriteString(&driver_send_frame,str_send_frame_2));
    }
}
  
```

Dans la première boucle for, la valeur de capteur de température de tête de cylindre et le capteur de température des gaz d'échappement est envoyé. Dans la deuxième boucle for, nous allons envoyer le reste de la trame.

Afin de s'assurer que la bonne trame est envoyée, sur une plaque d'essai j'ai câblé **ICL232CPE** qui permet de convertir la trame UART en RS232. De cette façon, nous pouvons visualiser la trame sur l'ordinateur.

Sur l'ordinateur, le logiciel "Putty" a été utilisé. Voici la configuration à faire :



Voici la trame observée.

```
TL=011START::CHT1=000::CHT2=034::CHT3=000::CHT4=000::CHT5=000::CHT6=000::CHT7=000::CHT8=000::EXT1=000::EXT2=032::EXT3=000::EXT4=000::EXT5=000::EXT6=000::EXT7=000::EXT8=000::SNT1=000::SNT2=035::SNT3=000::SNT4=000::SNT5=000::SNT6=000::SNT7=000::SNT8=000::TOIL=020::TWAT=021::POIL=001::PFUE=001::PMAN=000::PAIR=000::FLV1=092::FLV2=092::ARP=M=1200::MRPM=386::FFLW=028::BATL=011
```

CHT1 – CHT8 : Température de tête de cylindre. CHT2 a été connecté et nous pouvons voir que la température mesurée est de 34 degrés.

EXT1 – EXT8 : Température des gaz d'échappement. EXHT2 a été connecté et nous pouvons voir que la température mesurée est de 32 degrés.

SNT1 – SNT8 : Entrées des thermocouples supplémentaires. SNT2 a été connecté et nous pouvons voir que la température mesurée est de 35 degrés.

TOIL : Température d'huile – 20[°C]

TWAT : Température d'eau – 21[°C]

POIL : Pression d'huile – 1[bar]

PFUE : Pression d'essence – 1[bar]

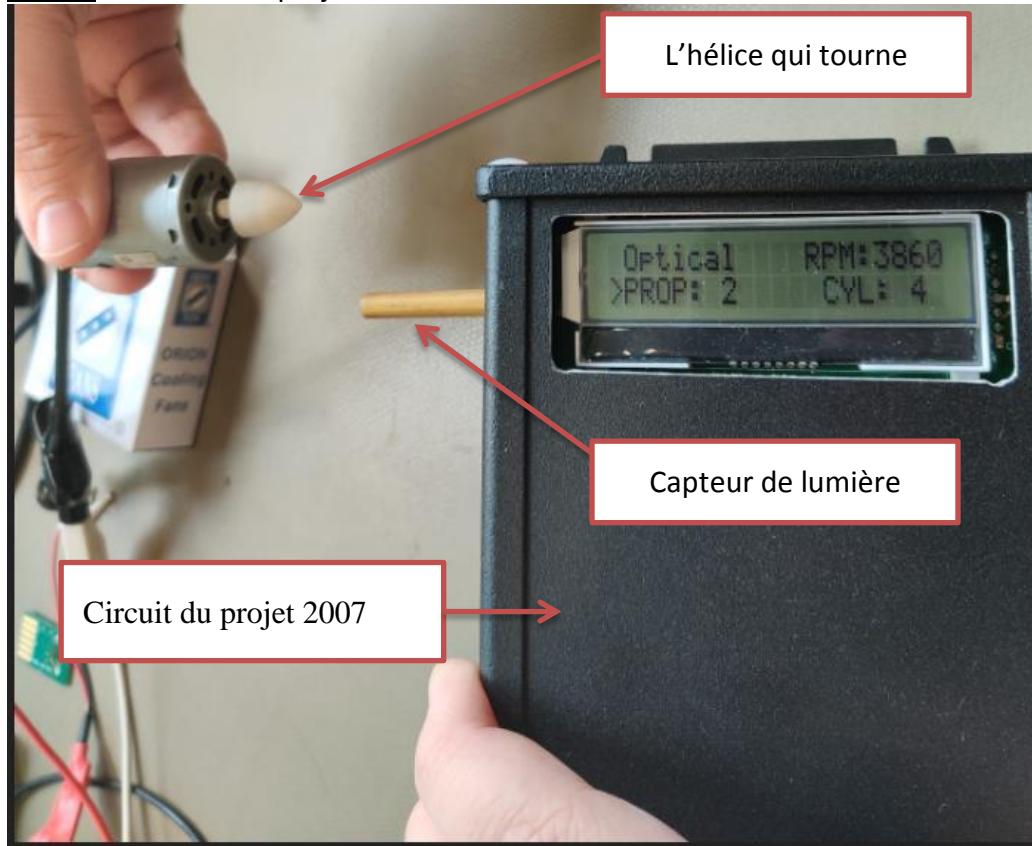
PMAN: Pression d'admission– Ce capteur a été cassé pendant le travail de semestre parce qu'une pression trop élevée a été appliquée

PAIR : Pression d'air – Ce capteur n'a pas été fourni par le client.

FLV1 et FLV2 : Niveau d'essence – 92[%]. Le client nous a fourni un capteur. J'ai donc fait un pont sur le circuit « PCB_Capteurs_Berney » entre chaque entrée des deux capteurs. C'est donc pour cette raison qu'on voit la même valeur.

ARPM : Les RPM de l'alternateur – Fréquence appliquée est de 200[Hz] et on affiche 1200 [1/10 tours par minute]

MRPM : Les RPM du projet 2007



Réglage du projet 2007 effectuée :

Pour mesurer les RPM, l'option « Optique » a été choisie. Le capteur de lumière va détecter le passage d'hélice et en fonction du nombre de cylindre les RPM sont calculés.

Sur l'écran, nous pouvons voir que l'appareil a mesurée 3860 RPM. Sur le terminal, la mesure

La valeur lue sur le terminal (Putty) est de 386. Une fois la trame réceptionnée, j'ai fait une division par 10 afin d'exprimer la mesure en dixième de tours minute, comme c'est le cas pour les RPM de l'alternateur.

FFLW : Débit d'essence. La valeur mesurée est de 28 [ml/s]

BATL : La tension de batterie. Comme le circuit est alimenté avec une alimentation laboratoire, en faisant varier la tension, j'ai pu observer la bonne tension au terminal.

Remarques : En observant la trame, je me suis rendu compte que les thermocouples n'affichaient pas précisément la bonne valeur. La température mesurée était plutôt autour des 22°C, comme les capteurs de température d'huile et de l'eau ont affichés.

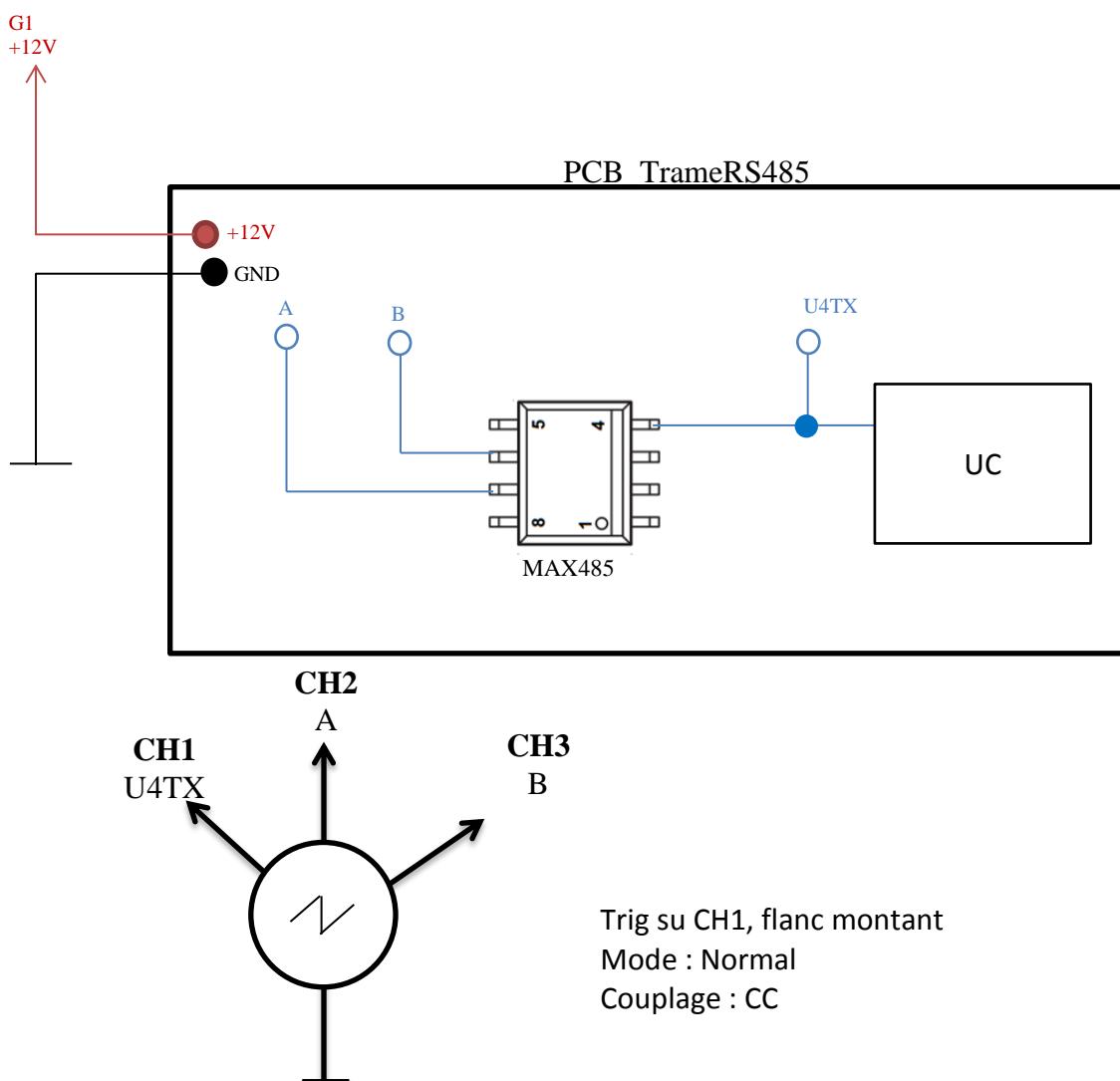
Comme la lecture des thermocouples n'est pas l'objectif principal du travail de diplôme, ce problème est à résoudre en fonction du temps restant.

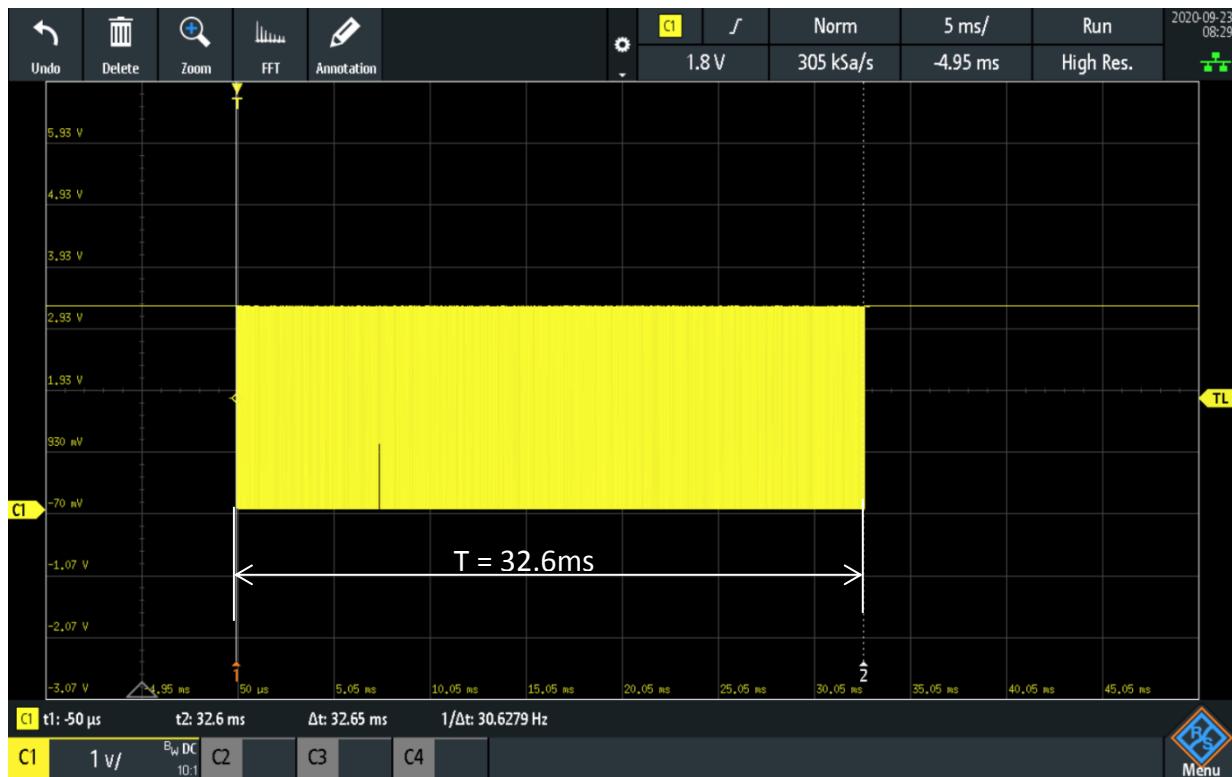
En ce qui concerne les autres capteurs, nous pouvons constater qu'ils affichent de bonnes valeurs. Les capteurs de pressions ont mesurés 1[bar], ce qui correspond à la pression atmosphérique, les niveaux d'essence affichent la bonne valeur lorsqu'on fait varier le flotteur, les RPM de l'alternateur correspondent bien à la fréquence appliquée et les RPM du projet 2007 ont bien été reçus.

Le débit d'essence est de 28 ml/s, ce qui correspond à une consommation de 1.68l/minute, donc 100 litre/heure. Si on prend en compte que pour faire la mesure, j'ai soufflé dans le capteur pour faire tourner l'hélice qui se trouve à l'intérieur, cette valeur peut être juste. Il faudra donc faire un essai sur l'avion.

J'ai fait une mesure, pour s'assurer que la trame est bien envoyée toutes les 200ms et pour vérifier la durée de la trame.

Schéma de mesure :



Mesure de la durée de la trame :**Calcul théorique de la durée de la trame :**

La vitesse de transmission est de 115'200 baud, donc la durée d'un bit est de 8.68us.
Pour envoyer un caractère, 10 bits ont été utilisés (START, 8 bits de données, STOP).

Voici la marche à suivre permettant de calculer la durée de la trame :

- **Nombre de capteurs : 36**
- Il faut envoyer le **start** = 5 caractères
- Pour chaque capteur il faut envoyer ::XXXX=YYY . X représente le nom du capteur et Y sa valeur.
Nous avons donc **10 caractères pour chaque capteur**.
- Nombre de caractères a envoyer = Nombre de capteurs * nombre de caractères par capteur + nombre de caractères pour le start
- Nombre de caractères total = $(36 \times 10) + 5 = 360 + 5 = 365$ caractères
- Nombre de bits total = Nombre de caractères total * 10bits = $365 \times 10 = 3'650$ caractères
- La durée totale de la trame = Nombre de bits total x la durée d'un bit = $3650 \times 8.68\text{us} = 31.68\text{ms}$

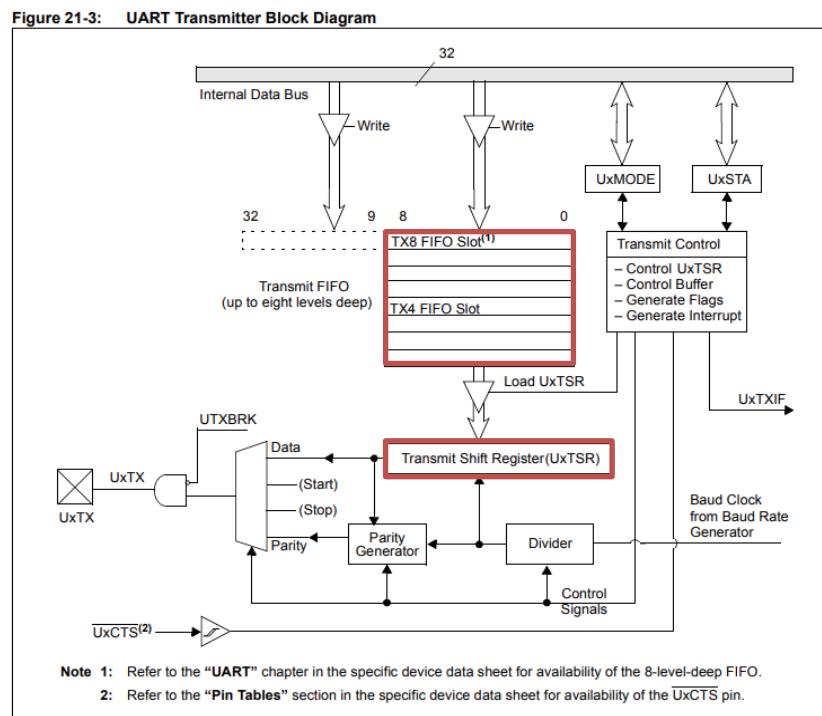
En regardant la mesure, ci-dessus la trame dure 32.6ms.

Pour comprendre pourquoi la trame dure plus longtemps nous pouvons observer la mesure suivante :



Avec cette mesure, nous pouvons voir qu'il y a un délai entre l'envoi de 9 caractères.

Pour savoir pourquoi ce délai est présent, j'ai regardé le datasheet du microcontrôleur :



Extrait du datasheet, page 21

Lien : <http://ww1.microchip.com/downloads/en/DeviceDoc/60001107H.pdf>

Avec ce graphique, nous pouvons voir que les 8 caractères peuvent être stockés dans le Fifo de l'UART et un caractère dans "Transmitt Shift Register", donc 9 caractères au total.

L'interruption est réglée afin d'envoyer la trame UART tant que le buffer n'est pas plein :

```
while (!PLIB_USART_TransmitterBufferIsFull(USART_ID_1) && moreBytesToSend) {
    moreBytesToSend = Serial_GetCharFromTxBuffer(1, &c);
    if (moreBytesToSend) {
        PLIB_USART_TransmitterByteSend(USART_ID_1, c);
    }
}
```

Le code du système d'interruption.

Le buffer va se donc remplir avec les caractères. Une fois qu'il arrive à 8 caractères (+1), la trame UART est envoyée.

Le délai mesuré avec l'oscilloscope représente le temps que le microcontrôleur a besoin pour remplir le fifo et pour effectuer les tâches dans le système d'interruption.

Pour régler ce problème, la trame UART peut être envoyée dès qu'il y a un espace libre dans le buffer UART, au lieu d'attendre qu'il soit rempli.

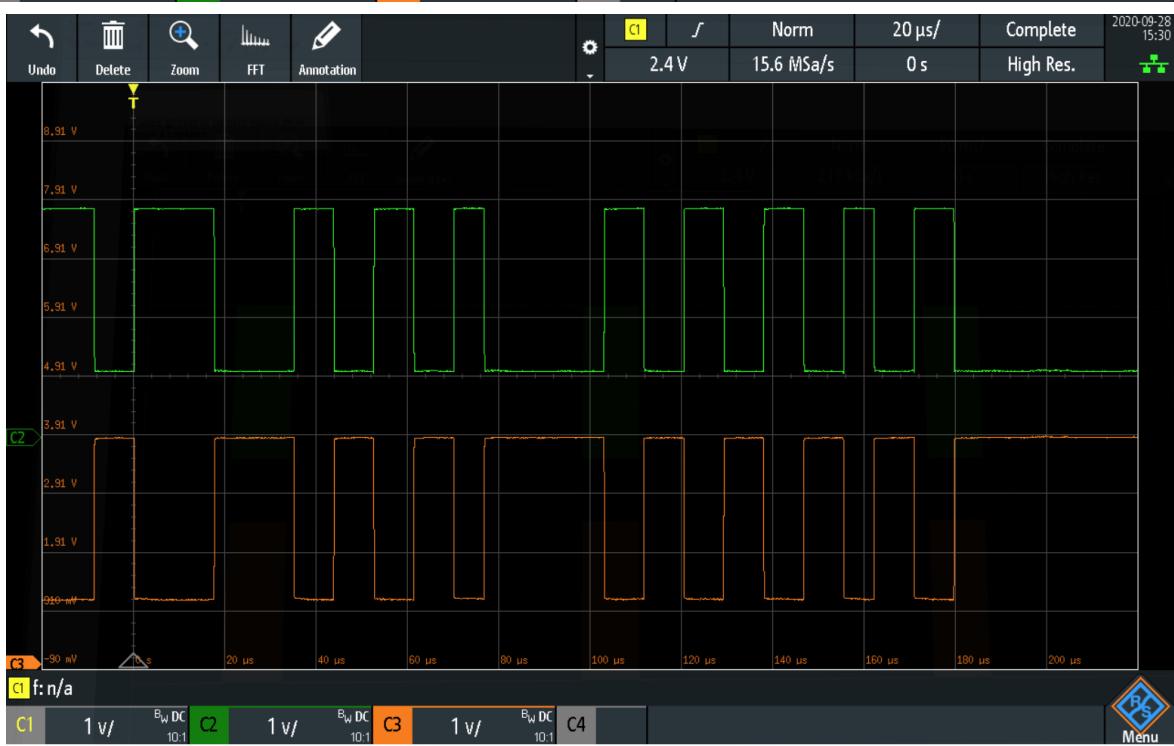
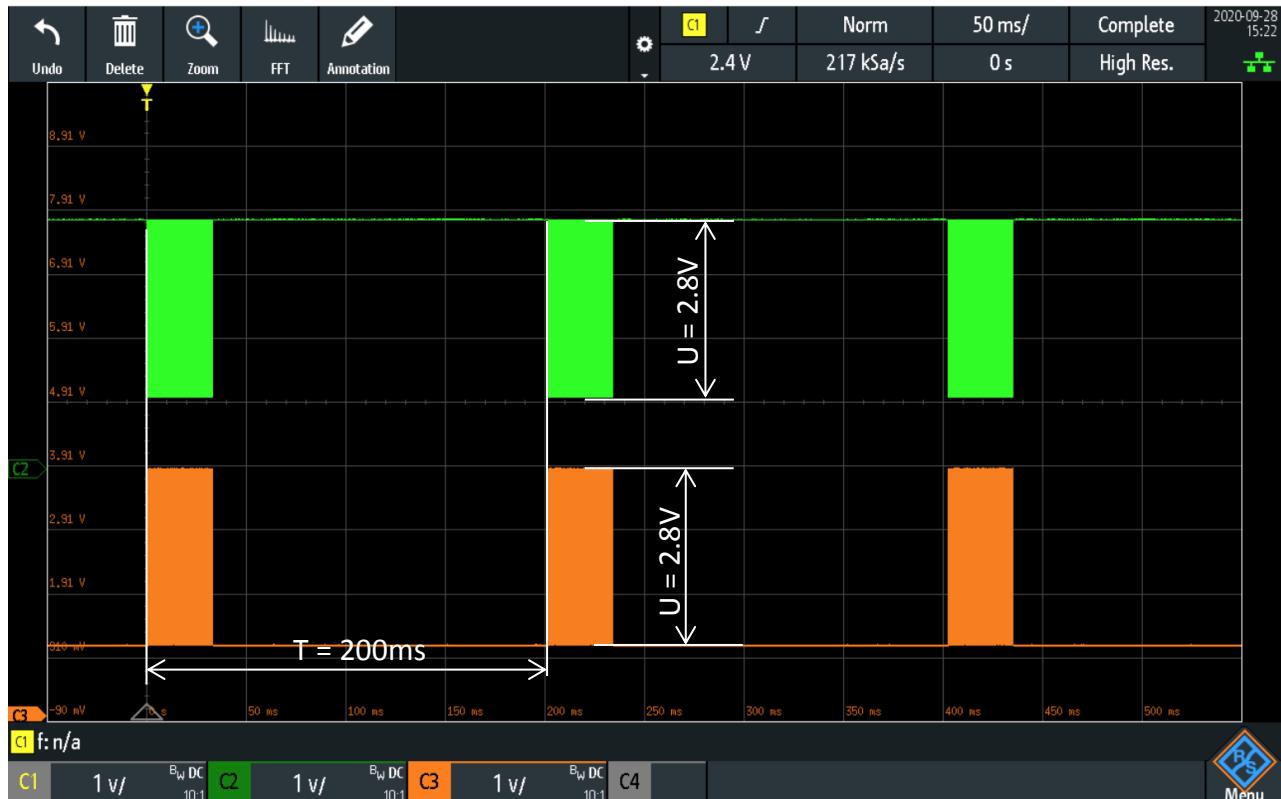
Les 1ms en plus, n'est pas dérangeant pour ce projet. La recherche ci-dessus a été faite afin de comprendre d'où vient le délai entre l'envoi des 9 caractères.

Mesure de la répétition de la trame :



Avec cette mesure, nous pouvons voir que la trame est envoyée toutes les 200ms.

Mesure des signaux A et B - RS485 :



La première mesure confirme que la trame RS485 est envoyée toutes les 200ms.

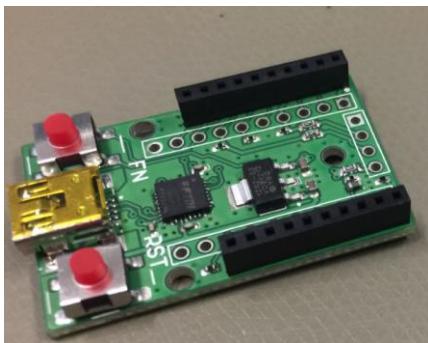
Grâce à la deuxième mesure, nous pouvons voir le fonctionnement différentiel. Lorsque nous avons un "1" logique sur la ligne A, sur la ligne "B" on peut voir "0" logique.

Selon la norme RS485, le niveau de tension est de ± 1.5 à ± 6 V. L'amplitude des signaux A et B est de 2.8V, la norme est donc respectée.

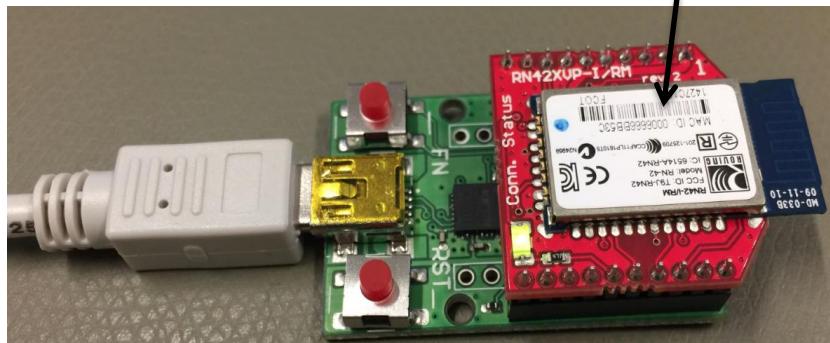
10.2 Module Bluetooth

Pendant que les pcb étaient en fabrication, le kit d'évaluation suivant a été utilisé pour paramétriser le module Bluetooth :

Kit d'évaluation :



Module Bluetooth RN42



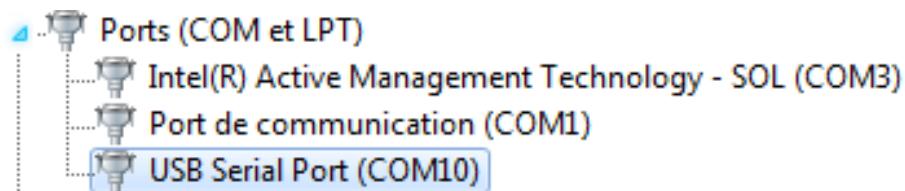
Principe de fonctionnement :

Le kit d'évaluation est branché sur l'ordinateur via le câble USB. Sur le kit d'évaluation, il y a un module qui permet de convertir la trame USB en UART.

Sur les connecteurs berg femelle, on insère le module Bluetooth RN42 et les commandes ASCII peuvent être envoyées via le PC afin de paramétriser le module Bluetooth.

Connexion du module Bluetooth sur le PC :

Dans un premier temps, le câble USB a été connecté. En allant dans le gestionnaire de périphériques -> section Ports (COM et LPT) nous pouvons voir que l'USB Serial Port a été reconnu et qu'il se trouve sur le COM 10 :



Paramètres du port série :

TABLE 1-1: SERIAL PORT SETTINGS

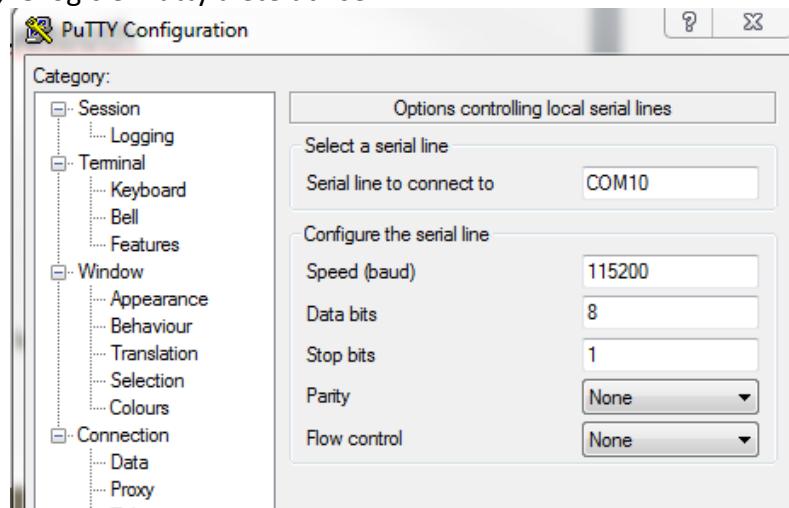
Setting	Value
Port	COM port to which you attached the module
Baud rate	115200
Data rate	8 bits
Parity	None
Stop bits	1
Flow control	None

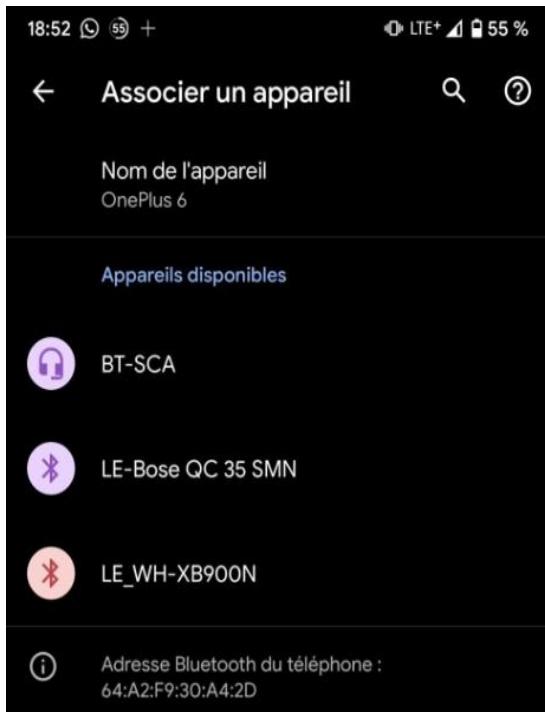
Par la suite, je me suis renseigné sur les différentes commandes à envoyer au module Bluetooth afin d'obtenir les réglages voulus.

Liste des commandes utiles :

Commande à écrire	Explication
\$\$\$	Mode "envoi des paramètres"
---	Retourner dans le mode "envoi des données"
H	Afficher la liste des commandes
X	Résumé de la configuration actuelle
+	Voir les paramètres qu'on écrit
S-,<string>	Modifier le nom du module Bluetooth. La taille maximale du nom à donner est de 15 caractères ASCII et les deux derniers caractères correspondent aux derniers caractères de l'adresse MAC
SP,<string>	Modifier le code pin permettant au dispositif de s'appairer au module
SU,<valeur>	Permet de modifier le baudrate. Les valeurs sont : 1200;2400;4800;9600;19.2k;28.8k;38.4;57.6k;115k;230k;460k;921kBaud
GK	En envoyant cette commande, le module BT répondra son statut actuel. En répondant 1, le module est connecté à un appareil et on répondant 0, le module est déconnecté.

J'ai donc utilisé les paramètres ci-dessus afin de paramétrier le module Bluetooth. Pour envoyer les commandes ASCII, le logiciel Putty a été utilisé :





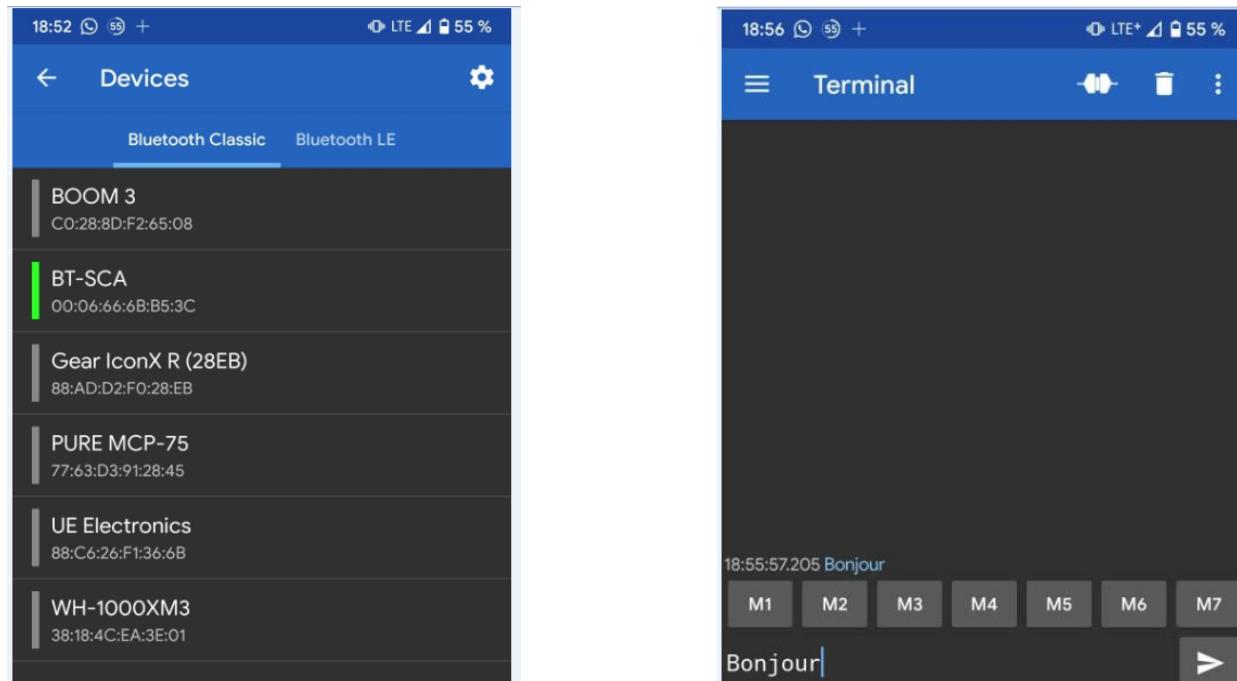
En regardant les appareils Bluetooth disponibles avec un smartphone, nous pouvons voir le dispositif BT-SCA et que l'adresse bluetooth du téléphone est 64:A2:F9:30:A4:2D

Voici ce qu'on peut observer sur Putty, en cliquant sur BT-SCA :

```
%NEW_PAIRING
%CONNECT, 64A2F930A42D, 0
```

Le smartphone s'est bien connecté au module Bluetooth. Nous pouvons constater que l'adresse bluetooth du smartphone correspond bien à l'adresse affichée sur Putty.

Pour voir les données envoyée via le module Bluetooth, l'application "Bluetooth serial terminal" a été téléchargé.



Il suffit donc de se connecter sur le module Bluetooth "BT-SCA" et nous pouvons envoyer les caractères. Pour faire l'essai, j'ai envoyé "Bonjour"

Sur le terminal du PC, nous retrouvons bien le message envoyé :

Bonjour

Modification des paramètres :

- **Modification du nom de Bluetooth**

Pour passer en mode "commande" il faut envoyer trois dollars :

\$\$\$

CMD

Afin de visualiser les paramètres actuels, la commande X a été utilisée :

Commande à écrire : X

```
Ver 6.15 04/26/2013
(c) Roving Networks
***Settings***
RTA=0006666C6A81
BTName=BT-SCA
Baudrt (SW4)=115K
Mode =Slav
Authen=0
PinCod=1234
Bonded=0
Rem=NONE SET
***ADVANCED Settings***
SrvName= SERVICE
SrvClass=0000
DevClass=1F00
InqWindw=0100
PagWindw=0100
CfgTimer=255
StatuStr=NULL
HidFlags=200
DTRtimer=8
KeySwapr=0
***OTHER Settings***
Profile= SPP
CfgChar= $
SniffEna=0
LowPower=0
TX Power=0
IOPorts= 0
IOValues=0
Sleeptmr=0
DebugMod=0
RoleSwch=0
```

Nous pouvons voir que le nom actuel est "BT-SCA".

Pour changer le nom il faut utiliser la commande SN, et écrire le nom souhaité. J'ai choisi de donner le nom "PROJ-2015"

Commande à écrire : SN,PROJ-2015

```
***Settings***  
BTA=0006666C6A81  
BTName=PROJ-2015  
Baudrt (SW4)=115K  
Mode =Slav  
Authen=0  
PinCod=1234  
Bonded=0  
Rem=NONE SET  
***ADVANCED Settings***  
SrvName= SERVICE  
SrvClass=0000  
DevClass=1F00  
InqWindw=0100  
PagWindw=0100  
CnfTimew=255
```

On envoyant le caractère X, nous pouvons voir que le nom du module est bien PROJ-2015

Modification du code pin de Bluetooth

Pour changer le code pin il faut utiliser la commande SP, et écrire le code voulu. Pour faciliter la tâche à l'utilisateur, le code correspond au numéro de projet (2015) :

Commande à écrire : SP,2015

```
***Settings***  
BTA=0006666C6A81  
BTName=PROJ-2015  
Baudrt (SW4)=115K  
Mode =Slav  
Authen=0  
PinCod=2015
```

Modification du baudrate

Pour modifier le baudrate il faut utiliser la commande SM, et écrire le baudrate souhaité. J'ai gardé le même baudrate comme sur la carte PCB_TrameRS485 (115'200 Baud)

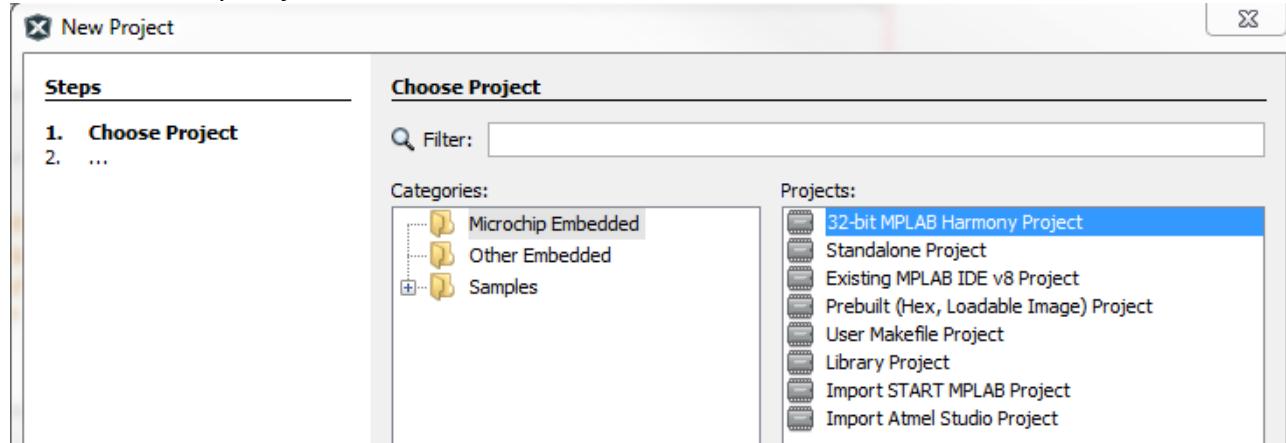
Commande à écrire : SM,115200

```
***Settings***  
BTA=0006666C6A81  
BTName=PROJ-2015  
Baudrt=115K
```

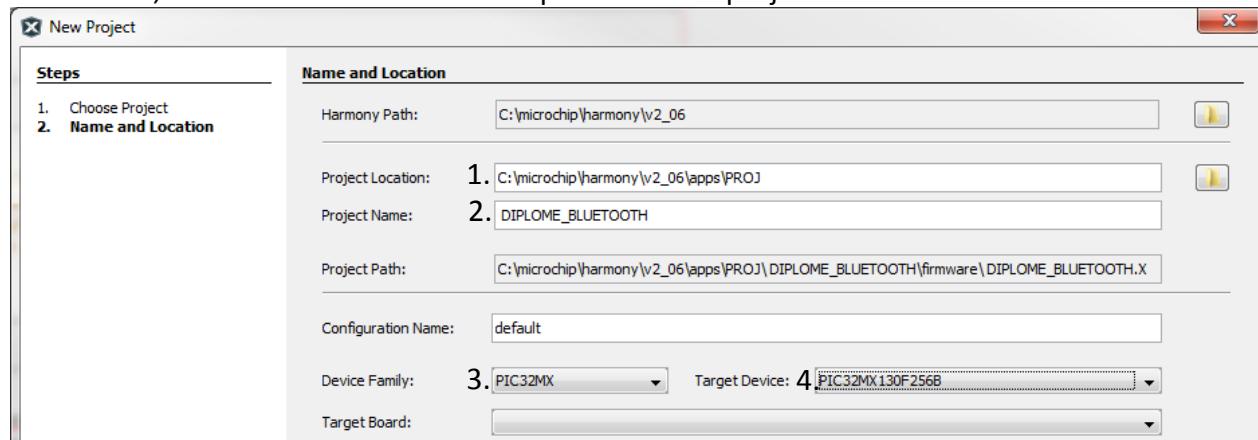
10.3 PCB_bluetooth

10.3.1 Création du projet

Pour créer le projet, il faut choisir File → New Project. Comme type de projet, j'ai choisi « 32-bit MPLAB Harmony Project » :



Par la suite, il faut donner le nom et l'emplacement du projet ainsi le modèle du microcontrôleur :

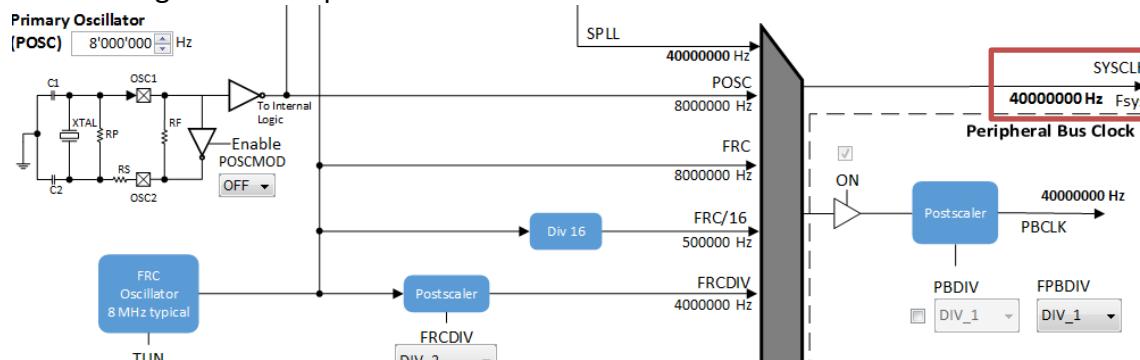


1. Emplacement du projet : C:\Microchip\harmony\v2_06\apps\PROJ
2. Nom du projet : DIPLOME_BLUETOOTH
3. La famille du microcontrôleur : PIC32MX
4. Le modèle du microcontrôleur : est PIC32MX130F256B

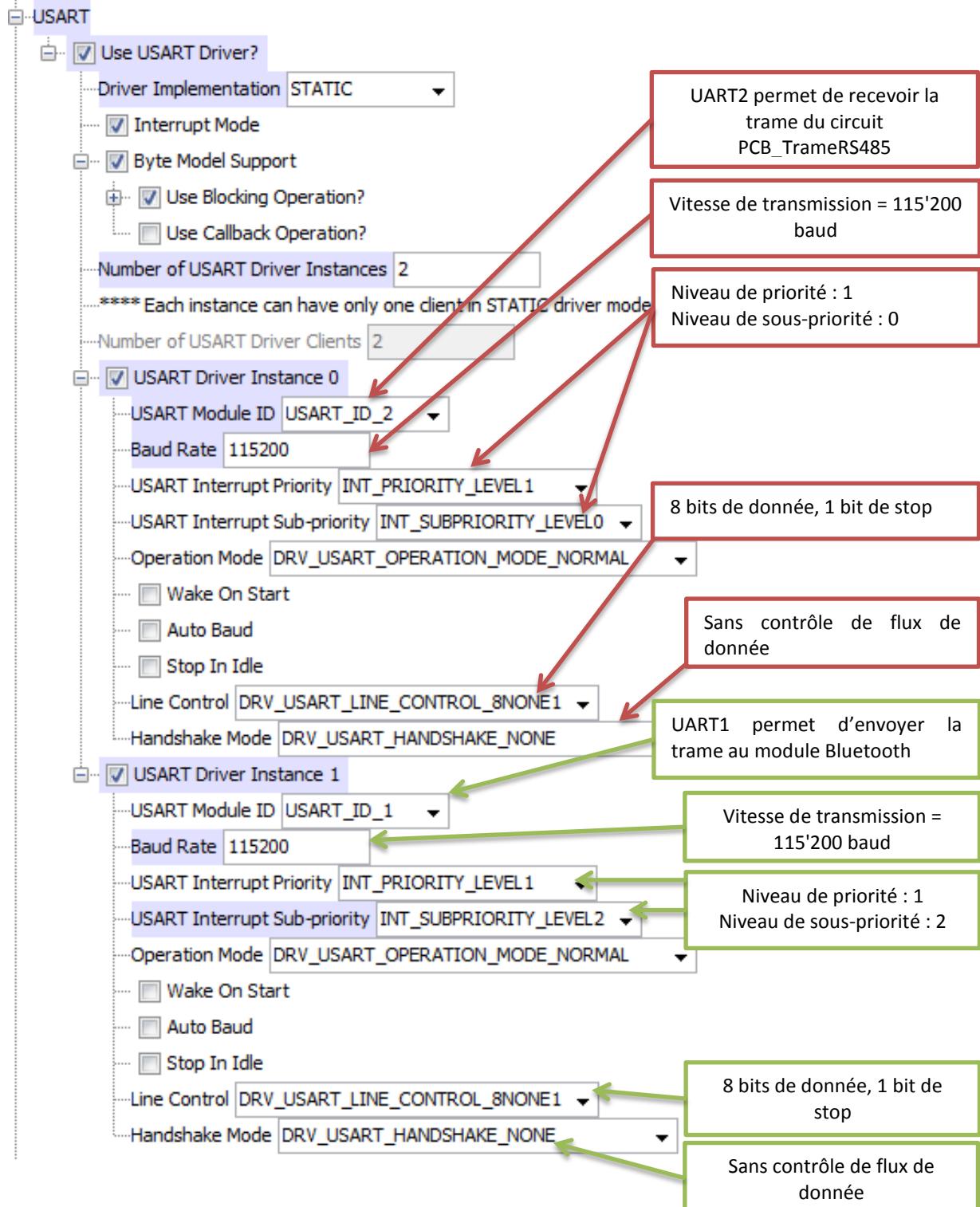
10.3.1.1 Configurations dans l'Harmony

Configuration du clock

J'ai décidé de garder la fréquence du clock de 40MHz :



10.3.1.2 Configuration de l'UART



10.3.1.3 Machine d'état

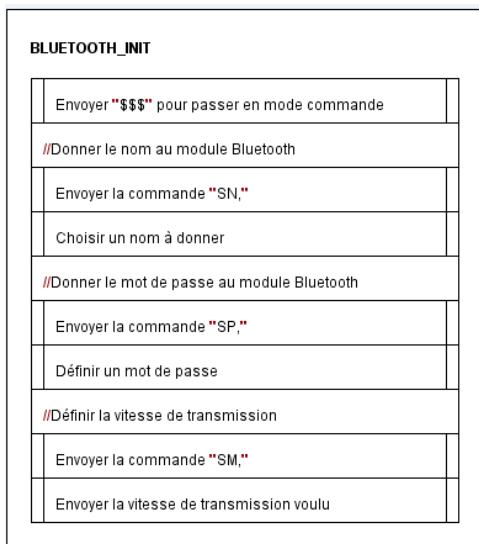


- Dans un premier temps, nous allons passer dans l'état APP_STATE_SEND_FRAME_TO_BT_INIT pour initialiser la communication UART permettant d'envoyer la trame au module Bluetooth.
- Une fois l'initialisation de la communication avec le module Bluetooth terminé, nous allons envoyer les commandes trouvées au chapitre précédent afin d'obtenir les réglages voulu. Cela permet aussi de garder les mêmes paramètres si on change le module Bluetooth.
- Après avoir initialisé le module Bluetooth, nous allons initialiser la communication UART permettant de recevoir la trame du circuit PCB_TrameRS485.
- Le programme passera ensuite dans l'état APP_STATE_WAIT et c'est donc dans le système d'interruption que nous allons envoyer la trame contenant la valeur des capteurs au module Bluetooth.
Comme il faut envoyer exactement la même trame, dès qu'on reçoit un caractère du PCB_TrameRS485, on l'envoie au module Bluetooth.

10.3.1.4 Initialisation du module Bluetooth

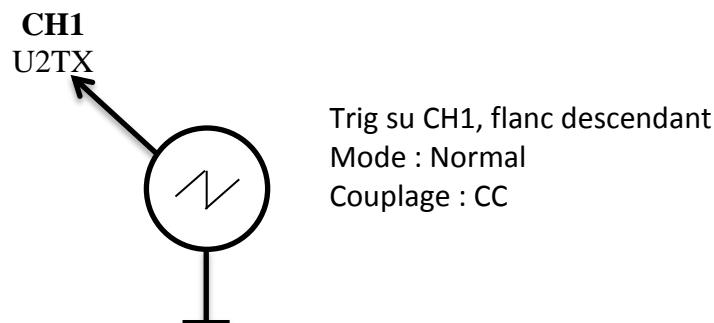
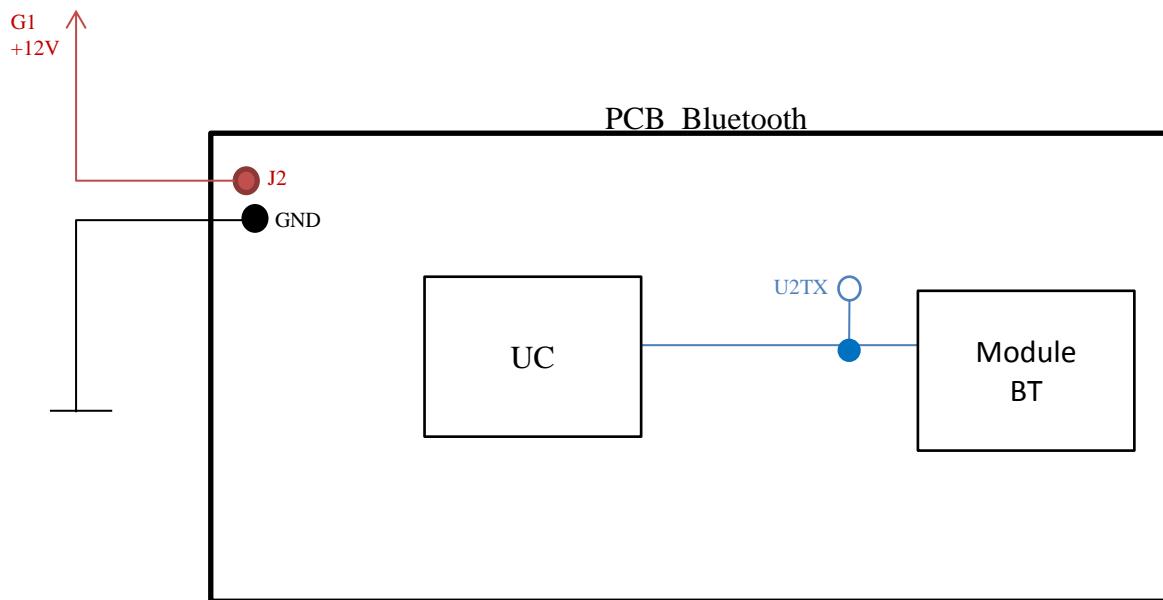
Structogramme

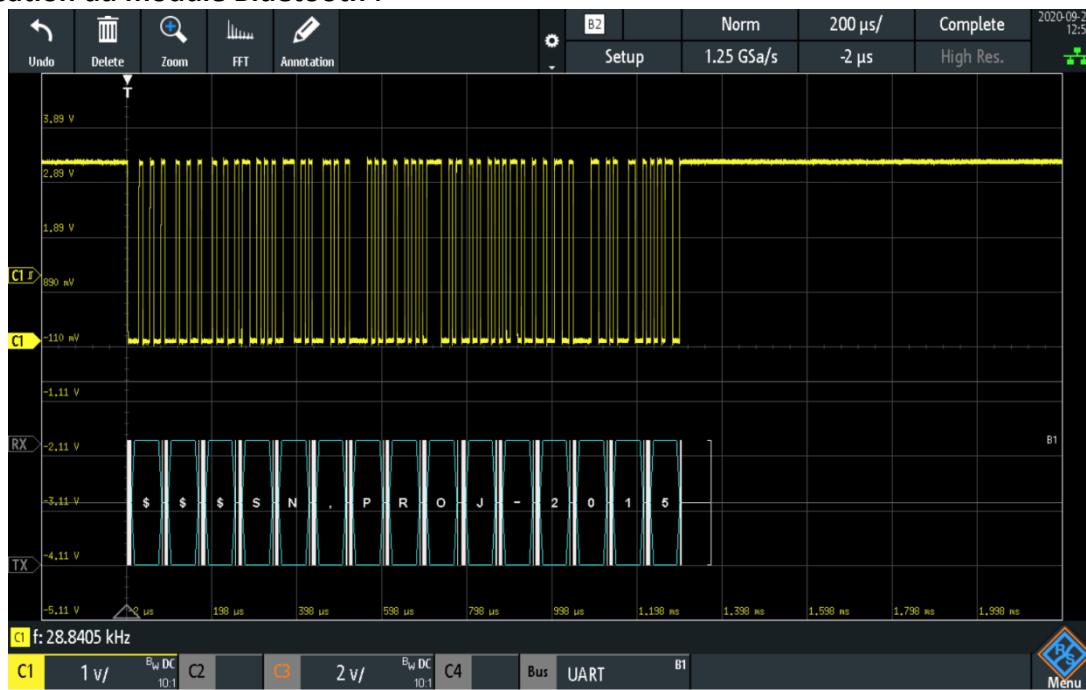
Les commandes trouvées au chapitre 11.2 m'ont permis de mettre en place le structogramme suivant afin d'initialiser le module Bluetooth :



J'ai fait les mesures suivantes pour s'assurer que les bonnes trames sont envoyées au module Bluetooth.

Schéma de mesure :



Résultat de mesure :**Initialisation du module Bluetooth :**

Avec cette mesure, nous pouvons voir que pour initialiser le module Bluetooth, la première commande envoyée est "\$\$\$".

Par la suite, le nom choisi précédemment a été envoyé.

Les commandes suivantes permettent de définir la vitesse de transmission et le mot de passe :

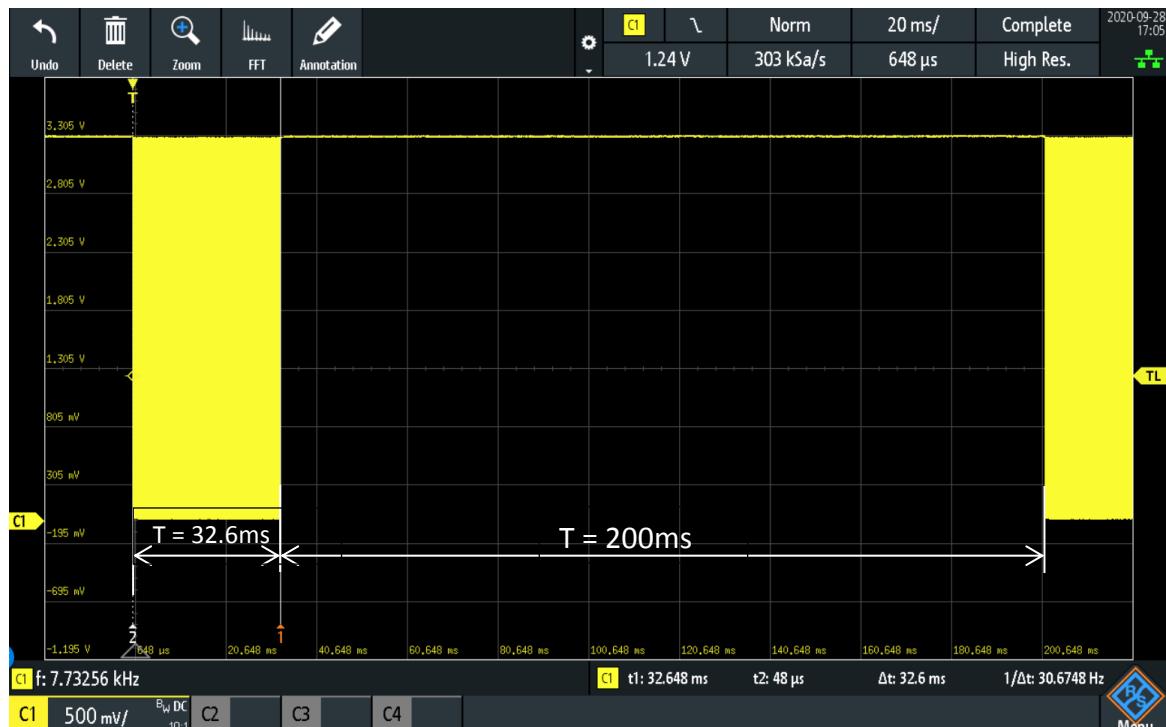


Les trois commandes ci-dessus ont été envoyées à la suite. Pour faciliter la mesure, dans un premier temps, la commande permettant de définir le nom du module Bluetooth a été envoyé et par la suite, les autres commandes ont été mesurées.

Une fois l'initialisation terminé, la trame du PCB_TrameRS485 est envoyée :

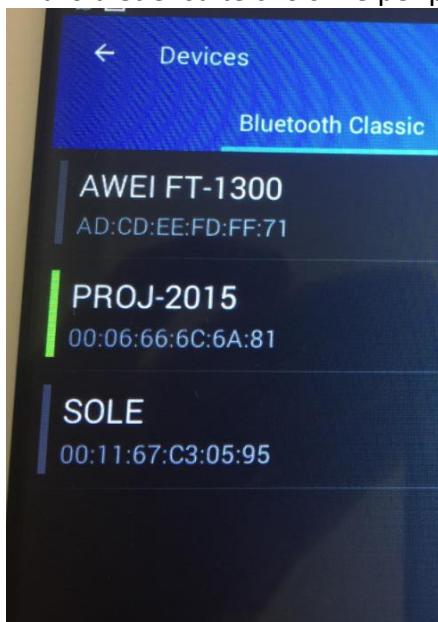


Comme la trame est assez longue, le début du message a été mesuré. On peut voir que la trame commence bien avec un START.



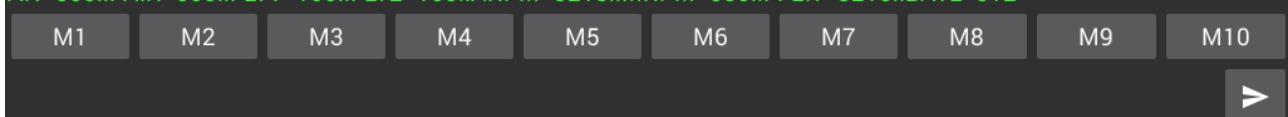
Comme mesurée au chapitre 11.1.6, la trame mesure 32.6ms et elle est bien envoyée au module Bluetooth toutes les 200ms.

Finalement, nous pouvons ouvrir l'application "Serial Bluetooth Terminal" sur une plateforme Android et ensuite choisir le périphérique PROJ - 2015 :



Voici la trame observée sur smartphone :

```
AN=000::PAIR=000::FLV1=100::FLV2=100::ARPM=3218::MRPM=000::FFLW=3218::BATL=012♦$$$SN,PROJ-2015SP,  
2015SM,115200START::CHT1=000::CHT2=000::CHT3=000::CHT4=000::CHT5=000::CHT6=000::CHT7=000::CHT8=000  
::EXT1=000::EXT2=033::EXT3=000::EXT4=000::EXT5=000::EXT6=000::EXT7=000::EXT8=000::SNT1=000::SNT2=000::S  
NT3=000::SNT4=000::SNT5=000::SNT6=000::SNT7=000::SNT8=000::TOIL=020::TWAT=021::POIL=001::PFUE=001::PM  
AN=000::PAIR=000::FLV1=100::FLV2=100::ARPM=3215::MRPM=000::FFLW=3215::BATL=012
```



11 Liste de matériel utilisé

Liste du matériel utilisé :

Désignation	Marque	Modèle	Caractéristiques	No d'inventaire
G1	Toellner	TOE7741	Générateur de fonction	ES.SLO2.00.00.11
P1	ROHDE & SHWARTZ	RTB2004	Oscilloscope	ES.SLO2.05.01.11

12 Problèmes rencontrés

Problème : L'interruption de l'IC3 n'a pas lieu. IC3 permet de lire le signal de l'alternateur afin de calculer les RPM.

En regardant le datasheet, cette pin (74) ne peut pas être utilisé en entrée :

Pin #	Full Pin Name	Pin #	Full Pin Name
71	DAC2/AN48/CVD48/RPC10/PMA14/PMCS/RC10	86	VDD
72	OA5OUT/AN25/CVD25/C5IN4-/RPB7/SCK1/INT0/RB7	87	RPF0/PMD11/RF0
73	SOSCI/RPC13 ⁽⁴⁾ /RC13 ⁽⁴⁾	88	RPF1/PMD10/RF1
74	SOSCO/RPB8 ⁽⁴⁾ /RB8 ⁽⁴⁾	89	RPG1/PMD9/RG1
75	VSS	90	RPG0/PMD8/RG0
76	TMS/OA5IN-/AN27/CVD27/LVDIN/C5IN1-/RPB9/RB9	91	TRCLK/PMA18/RF6
77	RPC6/USBID2/PMA16/RC6	92	TRD3/PMA19/RF7
78	RPC7/PMA17/RC7	93	RPB10/PMD0/RB10
79	PMD12/RD12	94	RPB11/PMD1/RB11
80	PMD13/RD13	95	TRD2/PMA20/RG14
81	RPC8/PMWRR/RC8	96	TRD1/RPG12/PMA21/RG12
82	RPD5/PMRD/RD5	97	TRD0/PMA22/RG13
83	RPD6/PMD14/RD6	98	RPB12/PMD2/RB12
84	RPC9/PMD15/RC9	99	RPB13/CTPLS/PMD3/RB13
85	VSS	100	TDO/PMD4/RA10

- Note**
- 1: The RPn pins can be used by remappable peripherals. See [Table 1](#) for the available peripherals and 13.3 "Peripheral Pin Select (PPS)" for restrictions.
 - 2: Every I/O port pin (RAx-RGx) can be used as a change notification pin (CNAx-CNGx). See [13.0 "I/O Ports"](#) for more information.
 - 3: Shaded pins are 5V tolerant.
 - 4: Functions are restricted to input functions only and inputs will be slower than standard inputs.
 - 5: The I²C library is available in MPLAB Harmony. For future hardware or silicon compatibility, it is recommended to use these pins for the I²C master/slave clock, that is, SCL.
 - 6: The I²C library is available in MPLAB Harmony. For future hardware or silicon compatibility, it is recommended to use these pins for the I²C data I/O, that is, SDA.
 - 7: VBAT functionality is compromised. For additional information, refer to specific errata documents. This pin must be connected to VDD.

Solution : J'ai connecté le signal de l'alternateur sur une autre entrée IC et ça fonctionne.

13 Améliorations à faire

- Connecter le signal de l'alternateur sur une entrée qui supporte 5V.
- La température mesurée par les thermocouples est élevée.

14 Évaluation du projet

14.1 État d'avancement du projet

14.1.1 PCB_Capteurs_Berney

- Ce circuit est monté et testé
- Aucune modification à faire

14.1.2 PCB_Capteurs_AMPA

- Ce circuit est monté et prêt à livrer au client.
- Le client va donc connecter les capteurs sur l'avion et faire le câble D-SUB37 pour pouvoir les connecter sur le circuit.

14.1.3 PCB_TrameRS485

Travaux restants :

- Lire le capteur de pression d'huile (capteur fourni pendant le travail de diplôme)
- Demander au client de nous fournir un autre capteur de pression d'admission. L'interface analogique est mise en place ainsi que le code.
- Il manque également le capteur de pression de l'air

Modifications à faire :

- Se référer au fichier 2015_SurveillanceParametresMoteurs_PCB_TrameRS485-MOD

14.1.4 PCB_Bluetooth

- Le circuit est monté
- La trame envoyée par PCB_TrameRS485 est envoyée au module Bluetooth et nous pouvons lire la trame sur la plateforme android.

Modifications à faire :

- Se référer au fichier 2015_SurveillanceParametresMoteurs_PCB_Bluetooth-MOD

15 Conclusion

En faisant ce travail de diplôme, j'ai été confronté à développer un appareil complet. Je suis très content d'avoir été en contact avec les clients ce qui nous a permis de discuter ensemble sur les différentes parties du projet. Cela m'a donné une idée du produit final, ce qui m'a aidé à réaliser ma pré-étude rapidement.

En ce qui concerne la partie schématique, je n'ai pas rencontré de grands problèmes. J'ai bien lu les datasheet et quand c'était possible les schémas proposés par le fabricant.

La partie principale de ce projet était le routage des PCB. Tout d'abord j'ai étudié le design du PCB afin de pouvoir mettre tous les connecteurs et simplifier la mise en boîtier. J'ai beaucoup utilisé la 3D de l'Altium ce qui m'a permis de faire les éventuelles corrections avant de commander les circuits.

Pour la partie software, j'ai pu utiliser une autre famille de microcontrôleur que nous l'habitude d'utiliser à l'école. La mise en place de la communication UART m'a pris du temps.

J'ai trouvé très intéressant la collaboration avec un Maxime Borlat qui a travaillé sur le projet 2007 Capteur des RPM non invasifs pour l'avion. Mon travail consistait à mettre en place la réception de la trame envoyée par son circuit. Une fois le code mis en place, nous avons pu le tester ensemble et observer la valeur envoyée sur une tablette.

En ce qui concerne le module Bluetooth, j'étais surpris du fait des réglages à faire pour pouvoir envoyer la trame. En effet, la configuration du module est assez simple et le kit d'évaluation m'a permis d'avancer plus rapidement et de trouver les commandes à envoyer.

Bien que le projet remplisse le cahier des charges, il n'est pas terminé. Il reste encore à connecter tous les capteurs sur un avion et d'observer les valeurs lues. Si besoin, il faudra prévoir une étape de calibration.

Il reste également à développer une application Android pour avoir une meilleure interface graphique.

Finalement, je suis fier du résultat du projet et j'espère pouvoir le tester sur un avion avec mon responsable de projet M.Juan José Moreno prochainement.

16 Remerciements

Je tiens à remercier M.Juan José Moreno pour sa disponibilité, son aide et ses précieux conseils. Enfin, je remercie Messieurs D.Bommottet et F.Baumgartner pour leur expertise et le temps qu'ils passeront à évaluer ce travail.

17 Annexes

- 17.1 Annexe 1 : Planification
- 17.2 Annexe 2 : Journal de travail
- 17.3 Annexe 3 : Liste de personnes de contact
- 17.4 Annexe 4 : Schéma électrique - PCB_Capteurs_Berney
- 17.5 Annexe 5 : Schéma électrique - PCB_Capteurs_AMPA
- 17.6 Annexe 6 : Schéma électrique - PCB_Capteurs_TrameRS485
- 17.7 Annexe 7 : Schéma électrique - PCB_Capteurs_Bluetooth
- 17.8 Annexe 8 : Circuit - PCB_Capteurs_AMPA
- 17.9 Annexe 9 : Circuit - PCB_Capteurs_Berney
- 17.10 Annexe 10 : Circuit - PCB_Capteurs_TrameRS485
- 17.11 Annexe 11 : Circuit - PCB_Bluetooth
- 17.12 Annexe 12 : Plans de perçage
- 17.13 Annexe 13 : Mode d'emploi
- 17.14 Annexe 14 : Modifications - PCB_TrameRS485
- 17.15 Annexe 15 : Modifications - PCB_Bluetooth
- 17.16 Annexe 16 : Code PCB_TrameRS485
- 17.17 Annexe 17 : Code PCB_Bluetooth