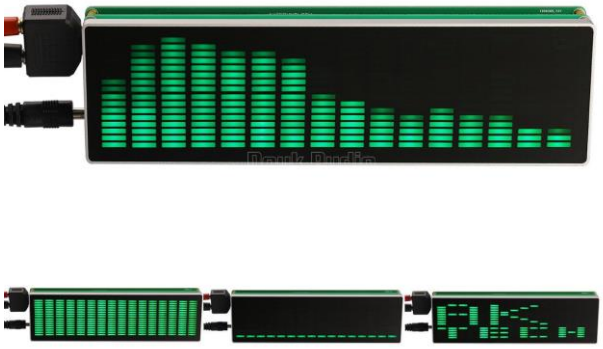
Génie électrique / électronique

Travail de diplôme 2024

Rapport de Diplôme

2225_VumetreFrequentiel



Réalisé par :

Loïc David

À l'attention de :

Philippe Bovey

Daniel Bommottet

Ricardo Neves Pereira

Début

19 août 2024

Fin

24 septembre 2024

1 Table des matières

2 Introduction

2.1 But du projet

Pour ce projet le but est de lire une entrée audio, filtrer différentes fréquences pour ainsi l'afficher sur dix LEDs RGB le niveau sonore des différentes fréquences ainsi que d'avoir une sortie bypass, une sortie avec le signal filtré et une sortie sur une matrice à LEDs.
Plus de détail en annexe (13.3).

3 Pré-étude

3.1 Schéma bloc

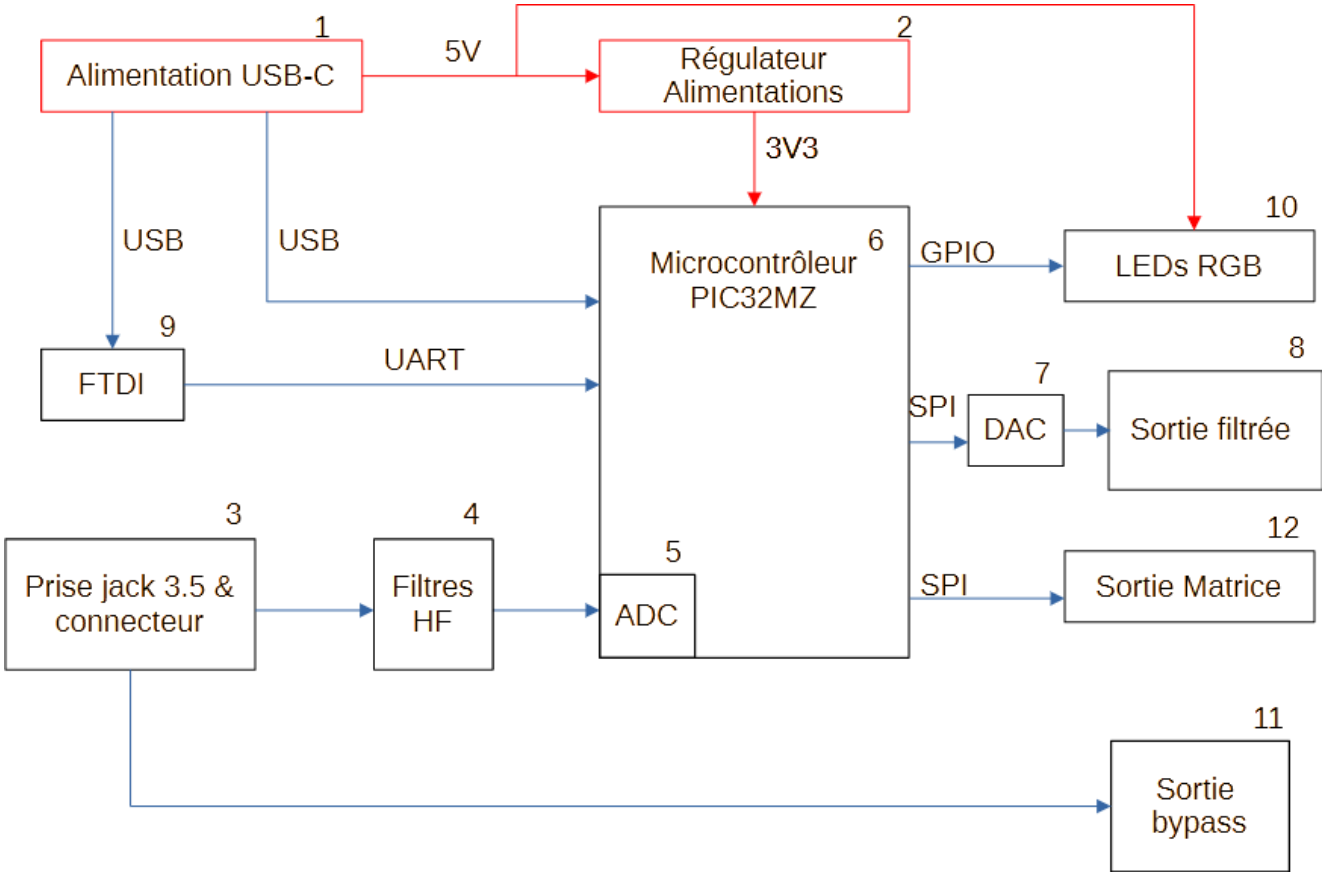


Figure 1 schéma block

(1) Alimentation USB-C

Pour alimenter notre carte, nous utilisons une alimentation en USB-C. Une alimentation en USB-C peut délivrer différentes tensions mais uniquement si on la configure en mode power delivery. Si un USB est branché sans autre configuration software, il délivrera directement 5[V].
Vu que j'utilise de l'USB pour communiquer avec un PC, j'ai décidé de prendre celui de l'alimentation (D+ - D-) pour communiquer avec l'ordinateur.

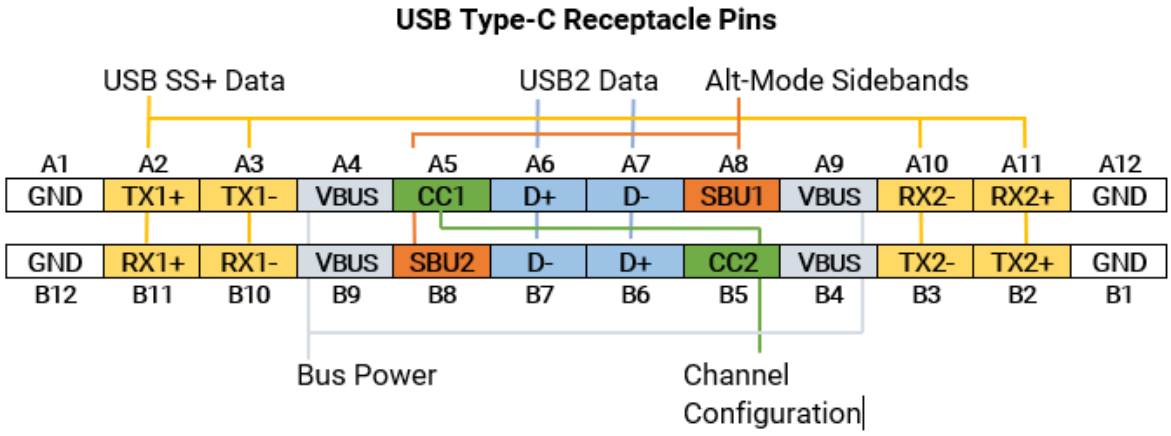


Figure 2 Mapage USB

Dans le cahier des charges, je dois utiliser un FTDI afin de communique en UART avec de l'USB.
Lors du choix du microcontrôleur, j'ai constaté qu'il était capable de directement décoder l'USB.
Le connecteur USB_C possède deux paires de données car il peut être mis dans les deux sens, avec un connecteur à 1 USB-C et 2 USB-A nous pourrions utiliser les deux communications en même temps.

Ce qui nous permettrai soit d'utiliser les deux communications en même temps pour différentes applications ou d'avoir deux solutions dans le cas où une des deux ne fonctionne pas.
Cette alimentation permet d'alimenter le régulateur 3V3 ainsi que les LEDs.

(2) Régulateur alimentations

Pour alimenter notre microcontrôleur, je dois ajouter une régulateur DC-DC qui me permettra d'avoir 3,3 [V].
Vu que pour une alimentation de microcontrôleur nous n'avons pas besoin d'un grand courant, le plus adapté serai de prendre une alimentation Whurt car elles sont blindées et en ayant utilisé ces alimentations dans le passé, elles sont fiables.

C'est pour cela que j'ai choisi une alimentation Whurt 173950336.

Figure 3 alimentation Whürt

(3) Prise jack 3,5mm & connecteur

Nous aurons une prise jack 3,5mm afin d'y connecter une entrée audio.

Figure 4 prise jack

En Recherchant, j'ai pu ainsi trouver les informations électriques suivantes :

Électrique

Général

Fonction	Prise en charge des appareils	Remarques
Entraînement de tension de sortie maximale	150mV	>= 150mV sur 32 ohms Conditions d'essai : EN50332-2
Résistance de polarisation du micro	Requis	Flexible sur la méthode de détection utilisée et la sélection de la résistance de polarisation du microphone. Exiger que toutes les plages de valeurs de résistance des boutons spécifiées ci-dessous soient détectées et liées à leur fonction respective
Tension de polarisation du micro	1,8 V - 2,9 V	Pour garantir la compatibilité avec les capsules de microphones courantes.

Figure 5 information JACK sur Android

Ainsi qu'un connecteur BCN que nous ajoutons afin d'y mettre un générateur de signal.

Figure 6 câbles BCN

Ceci afin de pouvoir faire tous les tests avec des fréquences et amplitude précises.

(4) Filtre HF

Vu que sur le signal audio il y aura probablement beaucoup de bruit :

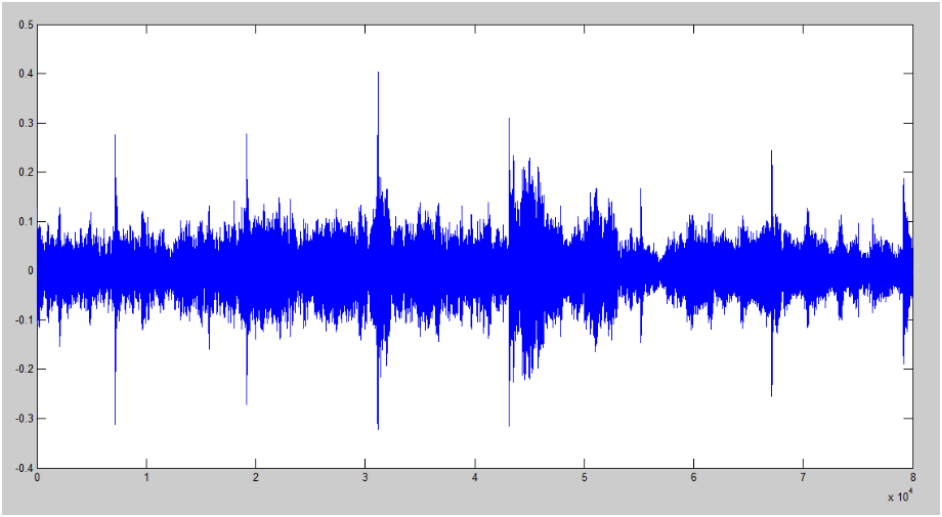


Figure 7 exemple signal audio

Le but de cette étape est déjà de faire un premier filtre passe bas afin d'enlever un maximum de bruit et avoir des mesures plus précises.

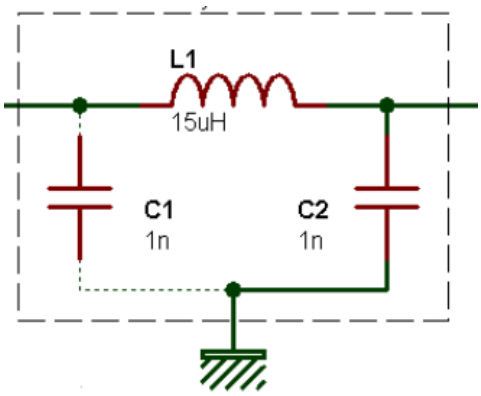


Figure 8 Filtre en PI

Il s'agit d'un filtre en PI.
C'est un filtre beaucoup utilisé dans l'audio qui permet de bien filtrer les hautes fréquences.

(5) ADC

Nous utilisons une ADC afin de pouvoir lire les signaux d'entrée.
Nous faisons une mesure jusqu'à 20 [kHz] nous voudrons probablement une fréquence d'échantillonnage au moins 10 fois supérieur à la fréquence la plus haute donc 200 [kHz].
Pour cela j'ai choisi d'utiliser l'ADC interne du microcontrôleur.

(6) Microcontrôleur

Lors de la version précédente, le microcontrôleur n'était pas adapté.
Pour cette version, je dois changer de famille de PIC32 afin d'en avoir un qui est adapté au traitement de signal audio. Vu que j'ai la possibilité de prendre un IO expender pour les LEDs, le nombre de pin n'est pas un problème majeur.
Mais comme noté dans le cahier des charges, j'ai choisi un PIC32MZ.

Pour le microcontrôleur j'ai besoin des périphériques suivant

- Convertisseur Analogique-Digital (entrée audio)
- 2 SPI
  - 1 pour le DAC
  - 1 pour la sortie matrice à LED
- 1 UART (FTDI)
- Eventuellement traitement USB

Les caractéristiques recherchées sont :

- La vitesse de travail car je traite un signal audio avec 10 filtres numériques
- Possibilité de travailler avec de l'audio

C'est pour cela que j'ai choisi le microcontrôleur suivant :

- Le PIC32MZ2048EFH064-250I_PT

(7) DAC

Pour le DAC, j'ai choisi de prendre le AD5620CRMZ-1.
Qui a une résolution de 13 bits ce qui est largement suffisant vu que nous n'allons pas réutiliser le signal mais il est plutôt là pour nous permettre de le visualiser.
Il peut aller jusqu'à 30 [MHz] ce qui très rapide et assez rapide pour de l'audio

(8) Sortie signaux filtrés

Cela sera la sortie après le DAC qui sera une reconstitution des signaux filtrés dans notre microcontrôleur. Cette sortie sera connectée à l'aide d'un connecteur BNC. J'utilise le même connecteur que pour l'entrée.

(9) FTDI

Le FTDI me permettra d'avoir la relation entre l'USB et le microcontrôleur.
Il fait la conversion UART-USB afin de le lire directement sur le micro.
Pour choisir le FTDI, une liste de différents composant m'as été fournie.

FTDI choisi : FT232RN.

Fonctionne sur 3V3 et peut être connecté à un micro.

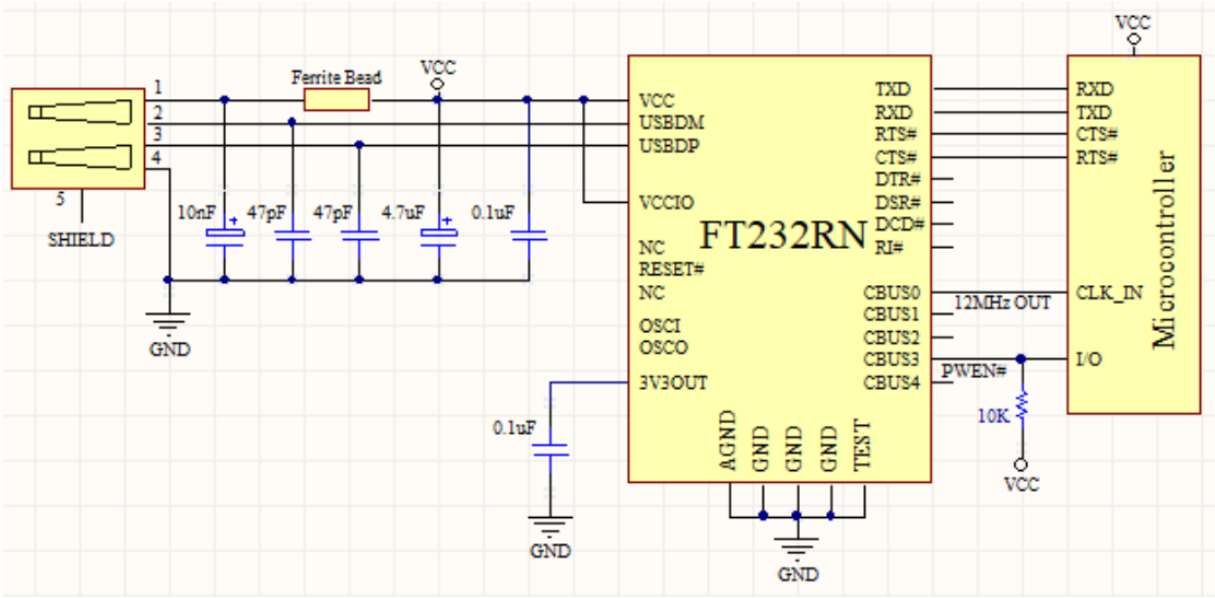


Figure 7.4 USB to MCU UART Interface

Figure 9 exemple datasheet connexion micro-USB

(Schéma datasheet)
Les deux sortie CBUS0 & CBUS3 ne seront pas implémentées car ce sont des sorties non obligatoires configurable depuis l'EPROM

(10)LEDs RGB

Pour les LEDs RGB différents choix s'offraient à moi :

- Prendre un des LEDs 3 couleurs avec pilotable par un io explander
- Prendre des LEDs RGB adressable

Finalement j'ai opté pour la première solution car il y aura déjà beaucoup de travail au niveau software pour ce projet et cela me permet d'éviter de rajouter des difficultés lors de la programmation.

LED choisie : APTF1616SURKCGKSYKC

(11) Sortie bypass

Cette sortie est une simple sortie directement reliée à l'entrée sans aucun filtrage qui permet d'avoir le signal de référence. Cette sortie permet justement que ce projet soit vu comme une boîte noire ce qui fais que nous pouvons avoir les différents filtrages du son en parallèle que la musique se joue toujours en sortie.

(12) Sortie Matrice

Pour la matrice à LED je dois pouvoir communiquer avec la matrice existante du projet 2227. Pour cela j'ai dû aller regarder les connecteurs utilisés ainsi que le maping des pins. Cette matrice fonctionne en SPI s'est pour cela que je dois utiliser le SPI2 du microcontrôleur.

4 Schématique

4.1 Alimentations

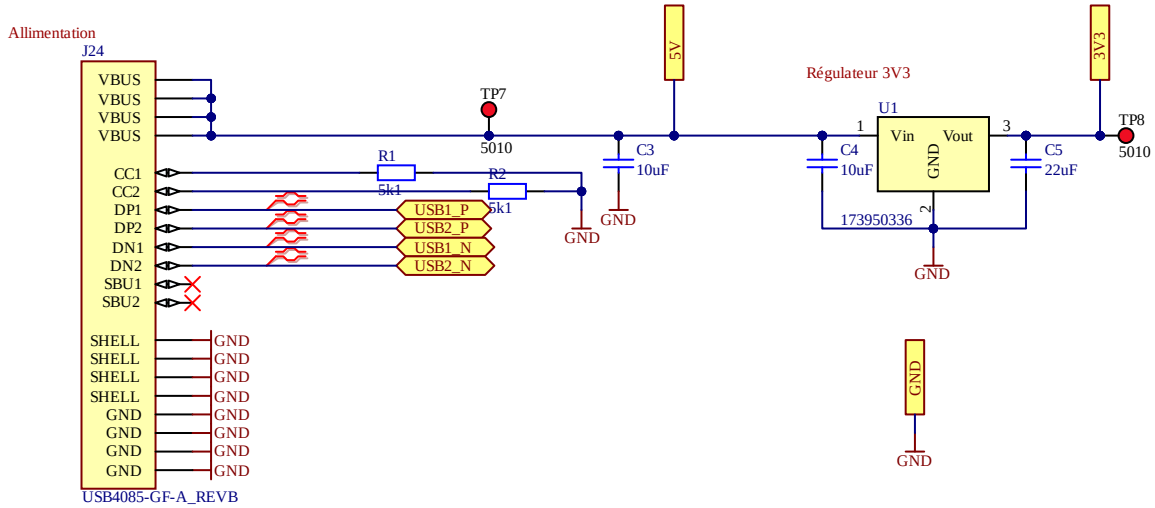


Figure 10 schéma alimentation

2

Pour l'alimentation principale, il y a un connecteur USB-C. Pour le connecteur, j'ai choisi un connecteur traversant :

Figure 11 connecteur USB-C

Pour les deux raisons suivantes :

- Simplification lors du montage et éviter d'avoir des ponts entre les différents pins
- Pour éviter que les efforts mécaniques en enlevant et mettant la prise n'endommagent les connexions.

R1 & R2 sont les deux résistances permettant de réguler le courant, elles sont connectées aux pins CC1 & CC2 qui sont les pins de commande. Avec une résistance de 5,1 [kΩ] nous limitons le courant à 500 [mA].

Du fait que nous utilisons l'alimentation 5 [V] pour alimenter les LEDs ainsi que la matrice à LED, un condensateur de découplage a été rajouté (C3).

Les signaux que nous utilisons avec l'USB-C, nous devons utiliser des paires différentielles sur les signaux D+ et D-.

Pour l'alimentation nous avons un 173950336 qui est un régulateur step-down Whurt blindé.
A l'intérieur de ce composant, nous pouvons retrouver un régulateur classique

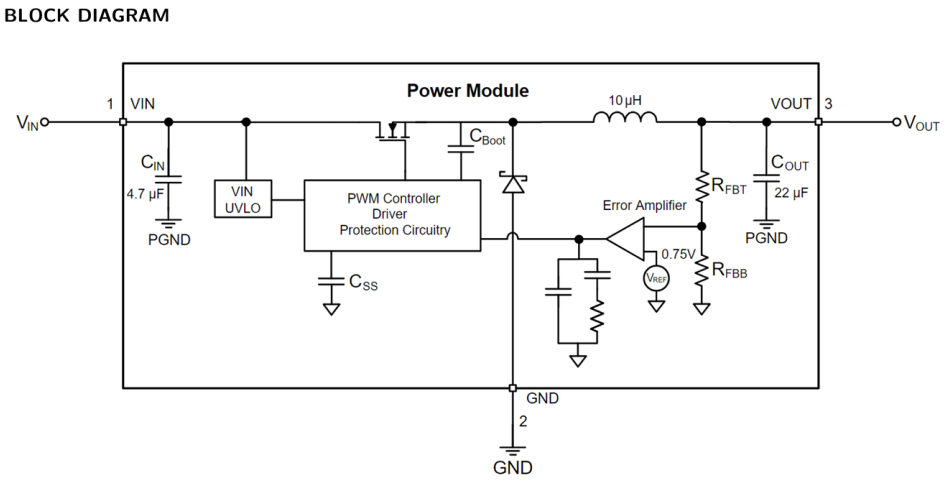


Figure 12 diagramme block alimentation Whürt

Avec Rfbt et Rfbb qui permettent de mesurer la tension de sortie ainsi qu'un PWM Controller driver qui permet de gérer le PWM sur le MOSFET qui se situe avant la bobine.
Ce schéma ressemble exactement à ce que nous avons pu voir dans notre cours d'électronique d'industrialisation :

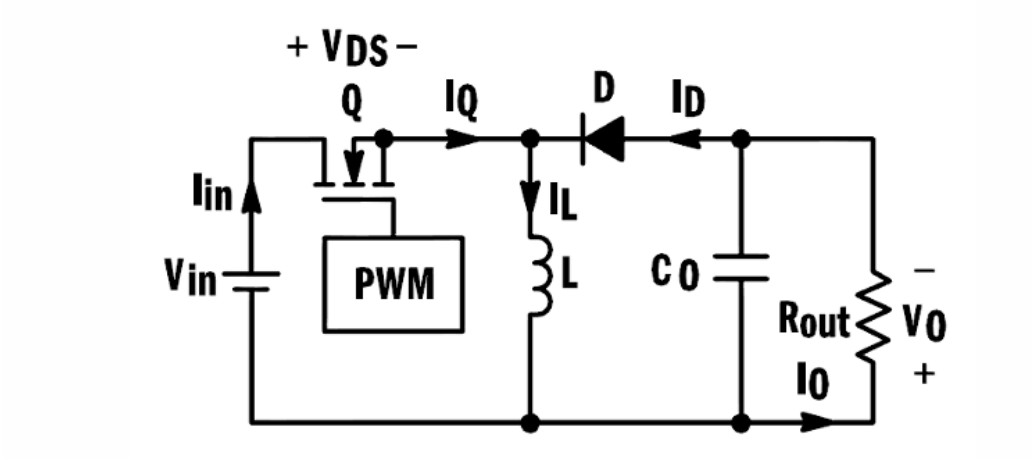


Figure 13 schéma pour buck

Le schéma d'application typique de la datasheet nous indique que nous devons rajouter un condensateur de découplage à l'entré (C4) et à la sortie (C5).

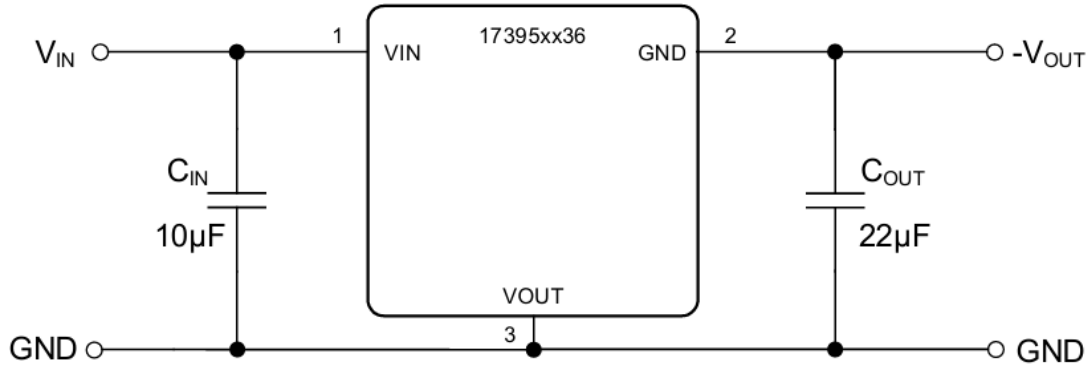


Figure 14 utilisation typique alimentation Whürt

4.2Microcontrôleur

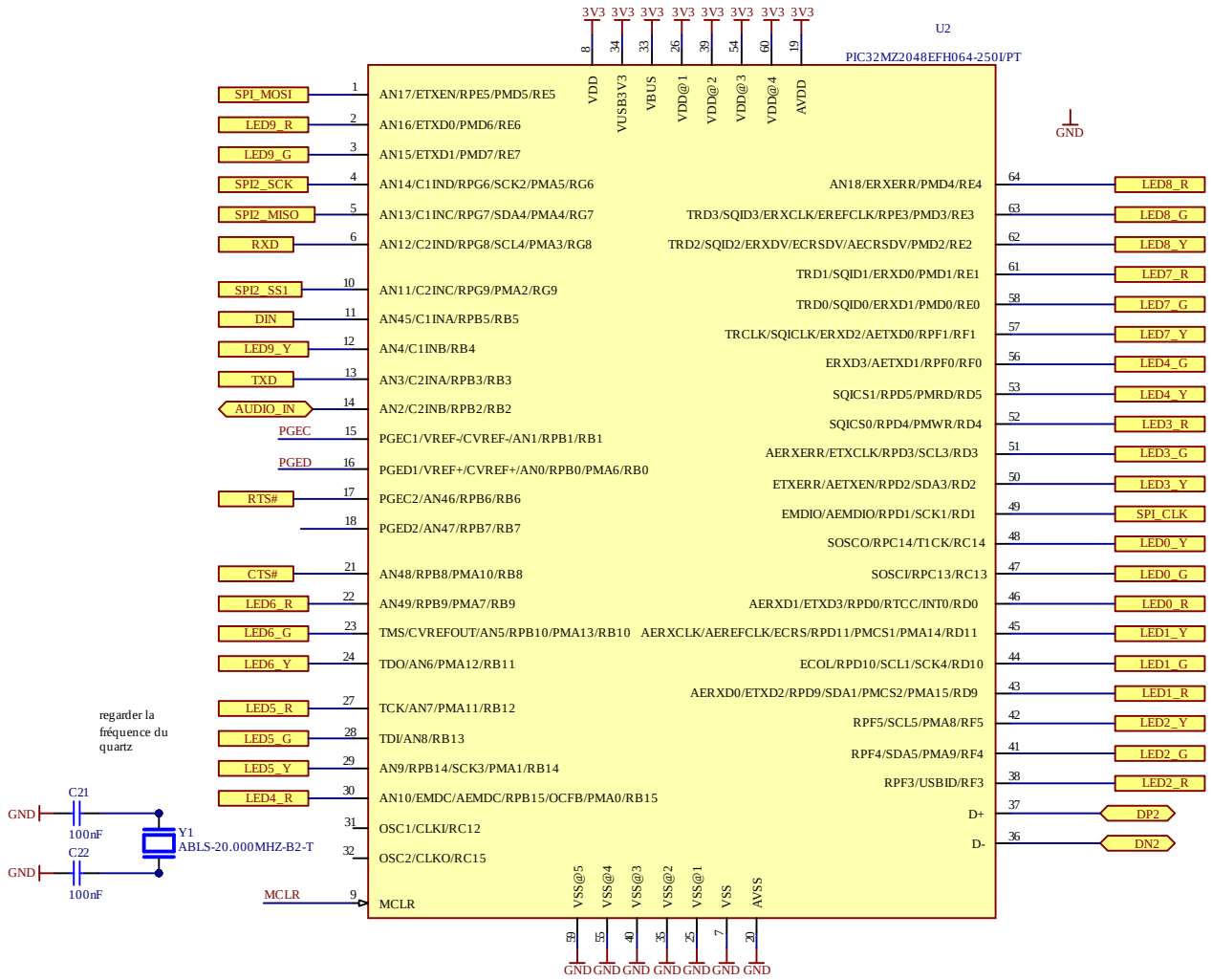


Figure 15 schéma uc

Voici le différent mapping des Entrées sorties du microcontrôleur ainsi que les périphérique et communication utilisés :

Périphérique	Communication	Fonction	Pin	Nom Uc
DAC	SPI	MOSI	1	SDO1
		CLK	49	SCK1
		Synchro data	18	GPIO_OUT
FTDI	UART	RXD	6	U1RX
		TXD	13	U1TX
		RTS#	17	U1RTS
		CTS#	21	U1CTS
USB	*USB*	*DP2*	*37*	*D+*
		DN2	36	D-
Entrée audio	ADC	AN2	14	AN2

Périphérique	Communication	Fonction	Pin	Nom Uc
Quartz	-	-	31	OSC1
		-	32	OSC2
Leds	I/O	LED0_R	46	GPIO_OUT
		LED0_G	47	GPIO_OUT
		LED0_Y	48	GPIO_OUT
		LED1_R	43	GPIO_OUT
		LED1_G	44	GPIO_OUT
		LED1_Y	45	GPIO_OUT
		LED2_R	38	GPIO_OUT
		LED2_G	41	GPIO_OUT
		LED2_Y	42	GPIO_OUT
		LED3_R	52	GPIO_OUT
		LED3_G	51	GPIO_OUT
		LED3_Y	50	GPIO_OUT
		LED4_R	30	GPIO_OUT
		LED4_G	56	GPIO_OUT
		LED4_Y	53	GPIO_OUT
		LED5_R	27	GPIO_OUT
		LED5_G	28	GPIO_OUT
		LED5_Y	29	GPIO_OUT
		LED6_R	22	GPIO_OUT
		LED6_G	23	GPIO_OUT
		LED6_Y	24	GPIO_OUT
		LED7_R	61	GPIO_OUT
		LED7_G	58	GPIO_OUT
		LED7_Y	57	GPIO_OUT
		LED8_R	64	GPIO_OUT
		LED8_G	63	GPIO_OUT
		LED8_Y	62	GPIO_OUT
		LED9_R	2	GPIO_OUT
		LED9_G	3	GPIO_OUT
		LED9_Y	12	GPIO_OUT
Matrice LED	SPI	SPI2_MOSI	1	SDO2
		SPI2_MISO	11	SDI2
		SPI2_SS1	10	SS2
		SPI2_CLK	4	SCK2
PGEx	Port prog	PGEC	15	-
		PGED	16	-

4.2.1.1Bouton reset

Le bouton reset est là afin de pouvoir reset le microcontrôleur manuellement, le schéma de ce bouton est celui trouvé dans la datasheet du microcontrôleur :

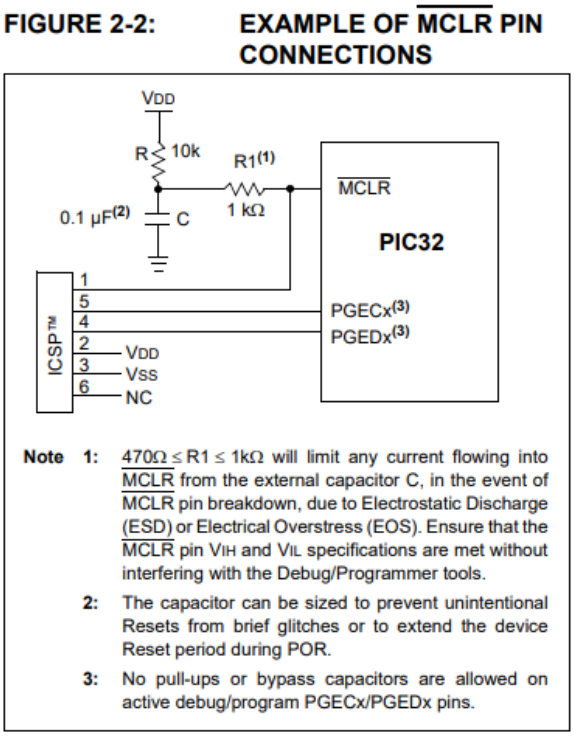


Figure 16 exemple connexion MCLR et PGEx

J'y ai simplement ajouté un bouton afin que je puisse le clear à volonté.

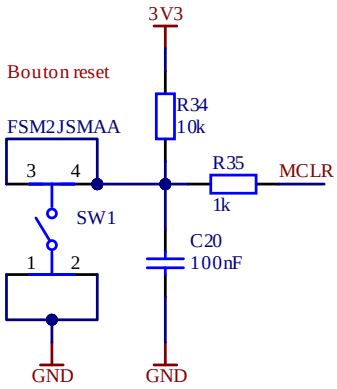


Figure 17schéma bouton reset

4.2.1.2Condensateur de découplage

Pour mettre les condensateurs de découplage ainsi que les alimentations, j'ai utilisé la datasheet avec les différentes valeurs annotées

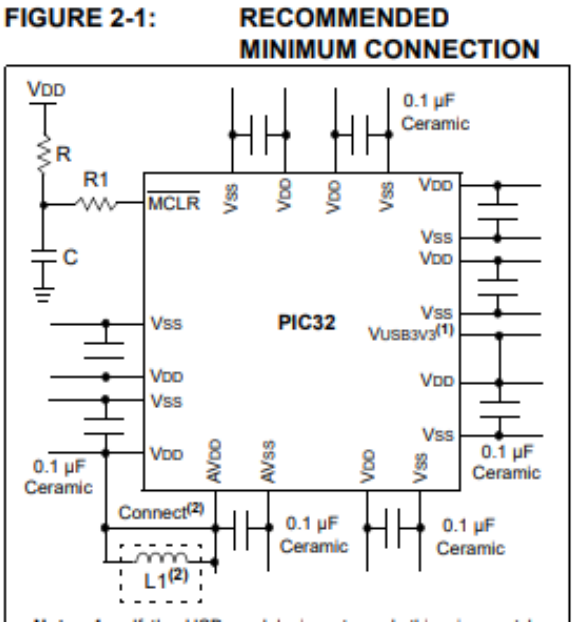


Figure 18 condensateur de découplage recommandés

J'ai ajouté les condensateurs de découplage à côté du microcontrôleur pour que lors du placement des composants, je puisse ajouter un condensateur sur chaque alimentation.

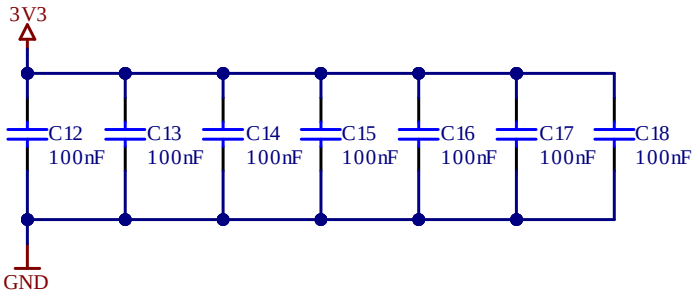


Figure 19 schéma condensateur découplage uc

4.2.1.3Port de programmation

Pour le port de programmation, j'ai réutilisé la figure 2-2 de la datasheet du microcontrôleur pour me permettre de lire et écrire sur celui-ci.

J'ai repris le schéma donne dans la figure 16 afin de faire le port de programmation que j'ai reproduit dans la schématique :

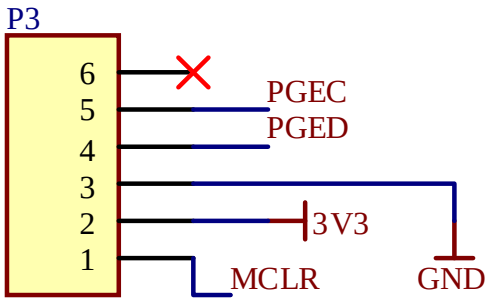


Figure 20 schéma port prog

4.3LEDs RGB

Notre LED est tout simplement 3 LED :

- Une rouge
- Une verte
- Un bleu

Qui sont dans le même boîtier avec une anode commune

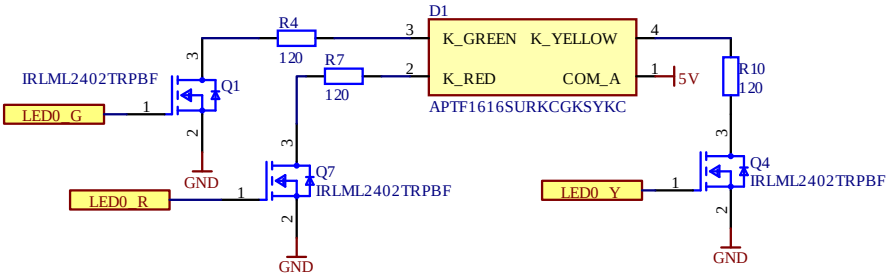


Figure 21 schéma LEDs

Si nous enlevons le boîtier nous pourrions voir Ceci :

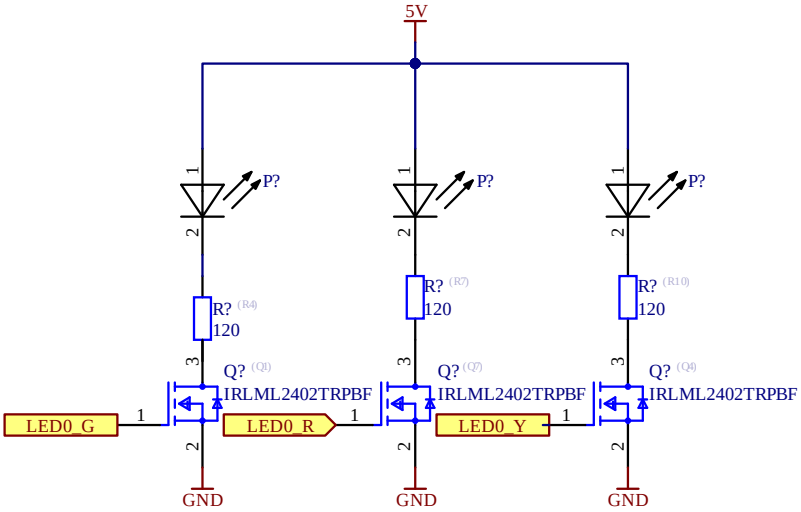


Figure 22 LED vue intérieur

J'ai choisi comme transistor le IRLML2402 car c'est le transistor que nous avons dans le stock-ES. Malheureusement ce composant est obselet mais nous pouvons le remplacer par un IRLML2502TRPBF qui avec ces caractéristiques peut le remplacer.
Dimensionnement des résistances :

Tension gate transistor :

Gate-threshold voltage	$V_{GS(th)}$	V_{DS} =V_{GS}, I_D =50µA	0.65		1.2	V

Figure 23 tension gate transistor

Tension sur les LEDs :

Forward Voltage I_F = 20mA	V_F [2]	Hyper Red Green Super Bright Yellow	1.95 2.1 2	2.5 2.5 2.5	V

Figure 24 tension sur les LEDs

Courant max sur les LEDs :

DC Forward Current	I_F	30	30	30	mA

Figure 25 courant max sur les LEDs

Vu que les courant de test sont de 10 mA, j'ai décidé de dimensionner les résistances pour 10 mA

$$R = \frac{Vcc - Uled - Vds}{If} = \frac{5 - 2.5 - 1.2}{0.010} = \ 130\ oHm - e12 \rightarrow 120\ oHm$$

4.4DAC

Pour connecter le DAC, j'ai utilisé la description des pins afin de savoir ce que je devais faire des Pins que je n'utilise pas.

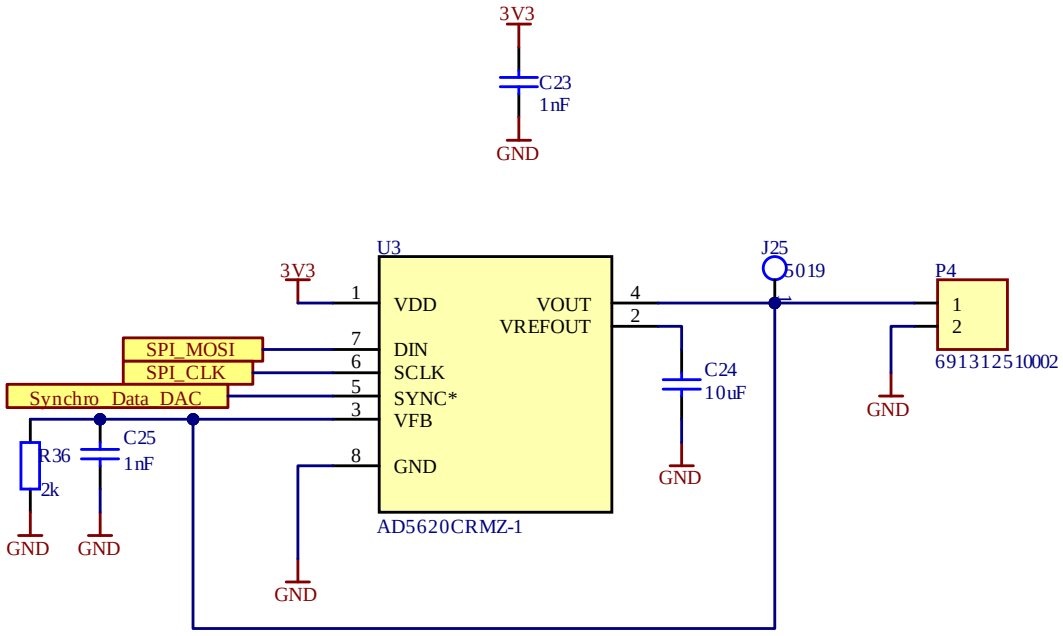


Figure 26 schéma DAC

Table 6. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	V_{DD}	Power Supply Input. These parts can operate from 2.7 V to 5.5 V. V_{DD} should be decoupled to GND.
2	V_{REFOUT}	Reference Voltage Output.
3	V_{FB}	Feedback Connection for the Output Amplifier. V_{FB} should be connected to V_{OUT} for normal operation.
4	V_{OUT}	Analog Output Voltage from DAC. The output amplifier has rail-to-rail operation.
5	$\overline{SYNC}$	Level-Triggered Control Input (Active Low). This is the frame synchronization signal for the input data. When $\overline{SYNC}$ goes low, it enables the input shift register and data is transferred in on the falling edges of the following clocks. The DAC is updated following the 24th clock cycle for the AD5660 and the 16th clock cycle for AD5620/AD5640 unless $\overline{SYNC}$ is taken high before this edge. In this case, the rising edge of $\overline{SYNC}$ acts as an interrupt, and the write sequence is ignored by the DAC.
6	SCLK	Serial Clock Input. Data is clocked into the input shift register on the falling edge of the serial clock input. Data can be transferred at rates up to 30 MHz.
7	DIN	Serial Data Input. The AD5660 has a 24-bit shift register, and the AD5620/AD5640 have a 16-bit shift register. Data is clocked into the register on the falling edge of the serial clock input.
8	GND	Ground Reference Point for all Circuitry on the Part.

Figure 27 Informations pins DAC datasheet

Vfref out est la référence de tension de sortie, J'utilise le DAC dans une plage de 0 [V] – 3,3 [V]. Ce qui fais que cette sortie ne m'est pas utile.

VFB peut être connecté a Vout pour des opérations normales.

C23 est un condensateur de découplage rajouté afin de garantir que le composant fonctionne correctement

4.5FTDI

Pour router le FTDI j'ai utilisé la figure USB to MCU UART comme exemple :

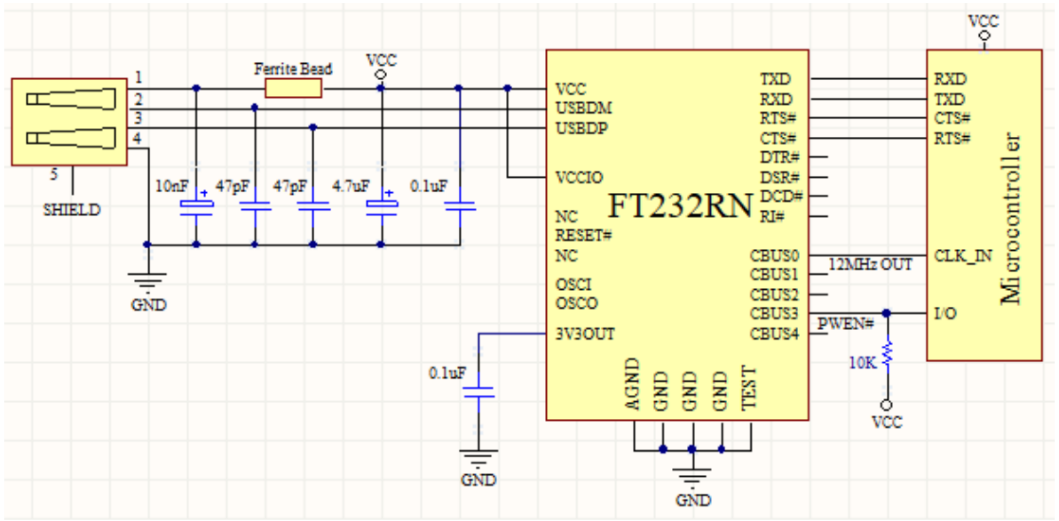


Figure 7.4 USB to MCU UART Interface

Figure 28 schéma typique datasheet FTDI

La pin VCCIO permet d'avoir une sortie à une certaine tension, nous pouvons la connecter à du 3V3. Vu que nous avons une sortie de 3V3 directement sur le composant j'utiliserai cette sortie comme alimentation.

Pin No.	Name	Type	Description
4	VCCIO	PWR	+1.8V to +5.25V supply to the UART Interface and CBUS group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect this pin to 3V3OUT pin to drive out at +3.3V levels or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive outputs at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should be used.

Figure 29 information pin VCCIO FTDI datasheet

La pin reset peut être connectée au Vcc avec une pull-up. Comme Pull-up j'ai choisi une résistance de 100k.

19	RESET#	Input	Active low reset pin. This can be used by an external device to reset the FT232RN. If not required can be left unconnected or pulled up to VCC.

Figure 30 information pin reset datasheet FTDI

Les signaux CBUSx sont des I/O configurable que nous pouvons utiliser pour différentes fonctionnalités. Vu que je n'ai pas besoin de ces fonctionnalités, ces pistes sont laissées non connectées

12	CBUS4	I/O	Configurable CBUS output only Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is SLEEP#. See CBUS Signal Options, Table 3.9.
13	CBUS2	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is TXDEN. See CBUS Signal Options, Table 3.9.
14	CBUS3	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is PWREN#. See CBUS Signal Options, Table 3.9. PWREN# should be used with a 10kΩ resistor pull up.
22	CBUS1	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is RXLED#. See CBUS Signal Options, Table 3.9.
23	CBUS0	I/O	Configurable CBUS I/O Pin. Function of this pin is configured in the device internal EEPROM. Factory default configuration is TXLED#. See CBUS Signal Options, Table 3.9.

Figure 31 information pins CBUS datasheet FTDI

Voici le schéma effectué du FTDI :

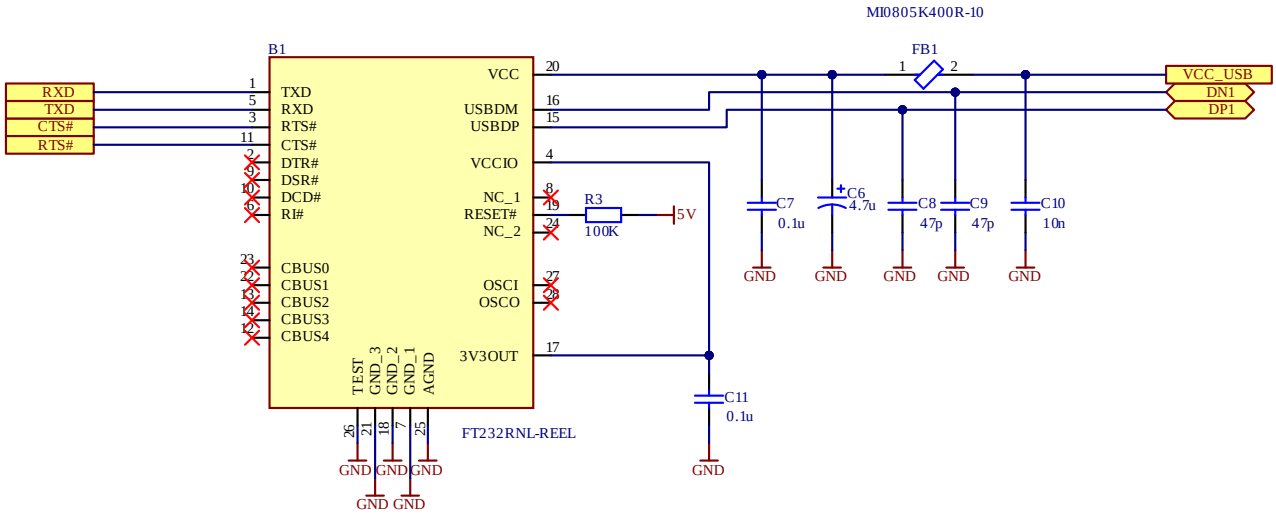


Figure 32 schématique FTDI

Pour choisi la ferrite FB1, j'ai utilisé la ferrite conseillée par la datasheet :

A ferrite bead is connected in series with the USB power supply to reduce EMI noise from the FT232RN and associated circuitry being radiated down the USB cable to the USB host. The value of the Ferrite Bead depends on the total current drawn by the application. A suitable range of Ferrite Beads is available from Laird (www.laird.com), for example Laird Part # MI0805K400R-10.

Figure 33 information datasheet FTDI sur la férite

4.6 Connecteur Matrice LED

Pour la matrice à LEDs j'ai repris le connecteur sortant du projet existant afin de l'implémenter dans le miens :

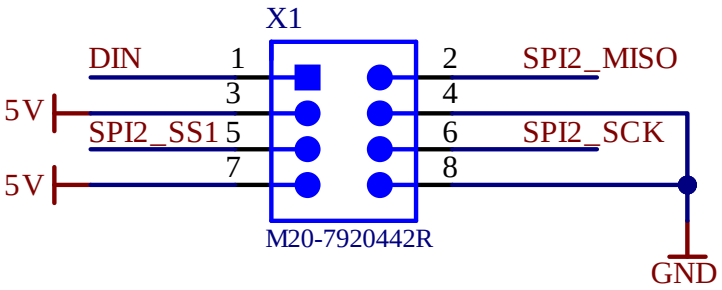


Figure 34 schéma du connecteur matrice à LED

Le but est d'avoir les différentes parties du SPI connectées afin que, dans une version ultérieure, une matrice à LEDs soit connectée à celui-ci.

4.7 Entrées

Pour l'es entrées, j'ai mis le connecteur JACK pour l'entrée audio ainsi que le connecteur BNC pour avoir une entrée que nous pilotons avec un générateur de fonction.

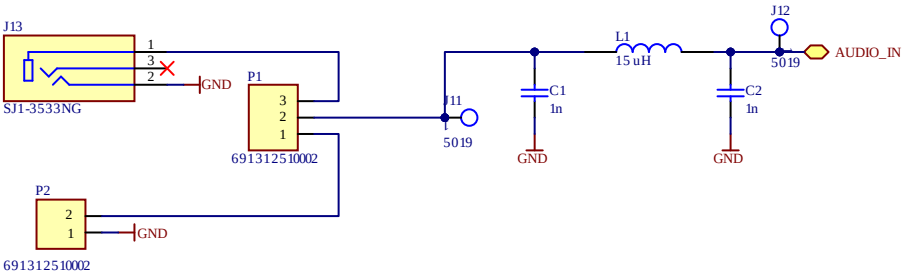


Figure 35 schématique des entrées

Le Connecteur P1 permet de choisir quelle entrée nous voulons utiliser pour éviter que les deux entrées se mélangent et que par exemple un signal non souhaité soit envoyé vers un appareil qui pourrai l'endommager.

Nous avons ensuite les composants L1, C1 et C2 qui forment un philtre en PI. Ce filtre est un filtre passe-bas anti-harmonique qui permet de filtrer les hautes fréquences qui pourraient parasiter les fréquences audios.

5 Design

5.1Placement

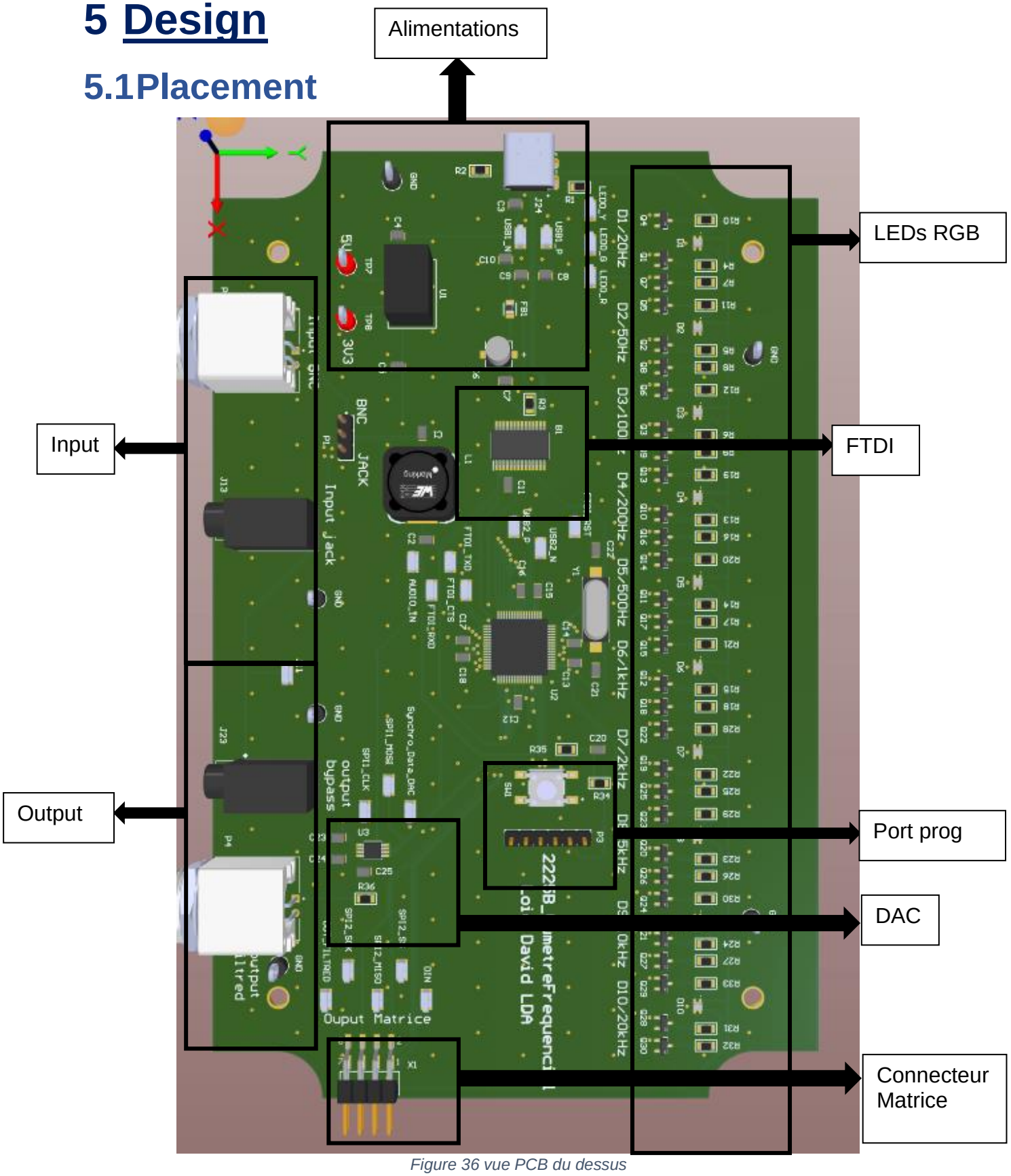


Figure 36 vue PCB du dessus

5.2Largeur des pistes + paires différentielles

Pour la largeur des pistes, j'ai utilisé une marge afin de garantir le bon fonctionnement de la carte :

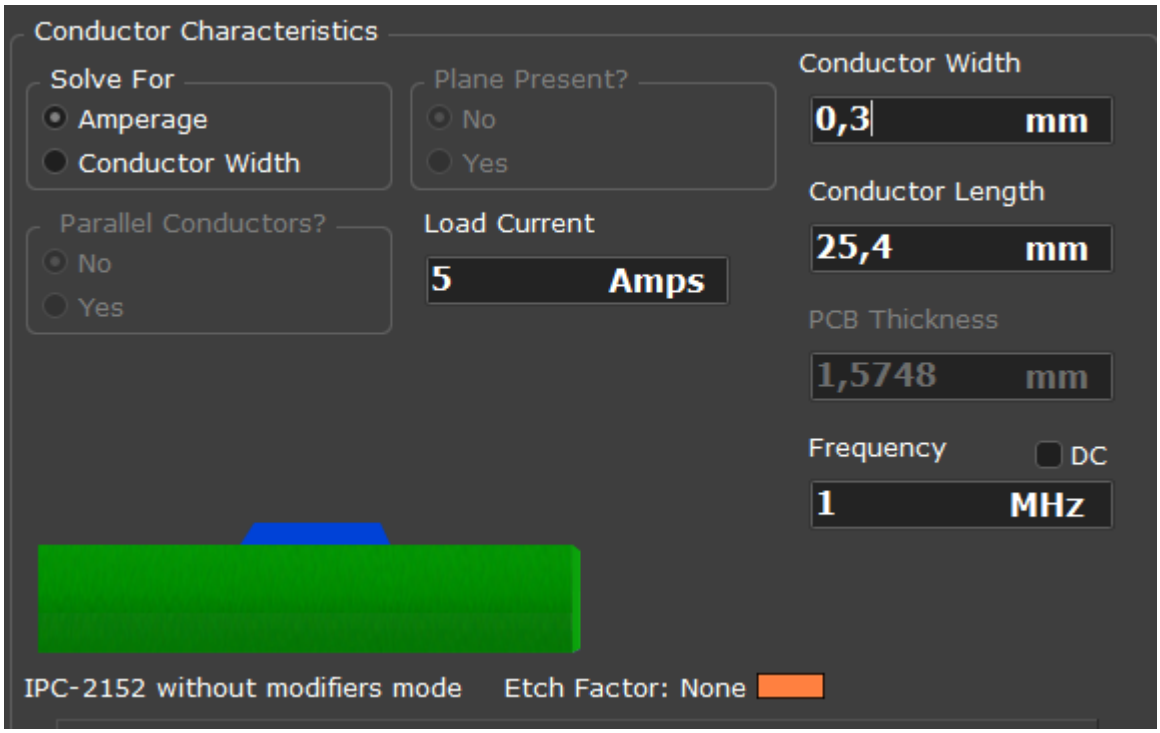


Figure 37 Saturn largeur piste 1

Ce qui me laisse une marge avec un courant pouvant aller jusqu'à 915 [mA].
J'ai décidé de prendre cette marge car je craignais que mes LEDs tirent trop de courant et que je doive modifier l'USB afin qu'il puisse en fournir plus.

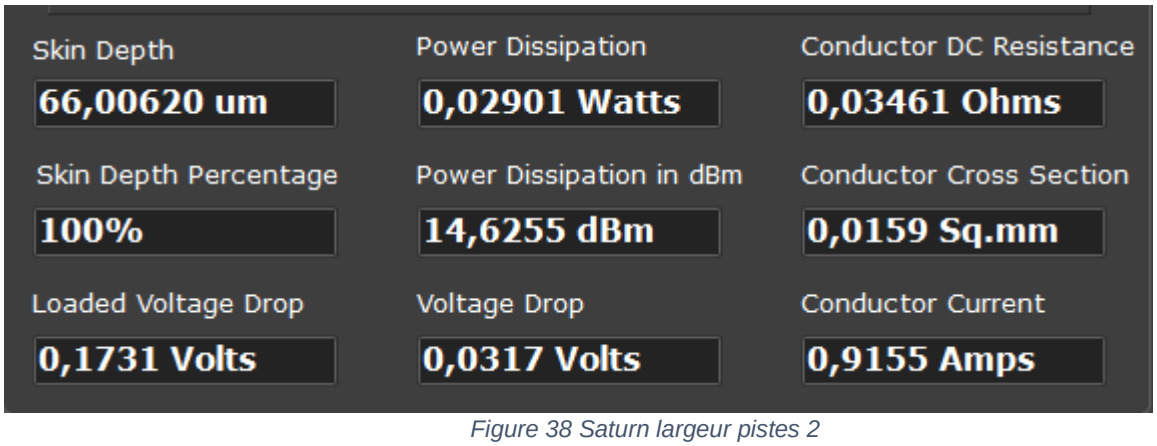


Figure 38 Saturn largeur pistes 2

Les pairs différentiels sont directement ajoutés sur altium et me permet de tirer une seul des deux lignes afin de créer la seconde automatiquement.

5.3Via Stitching

Pour faire le via stitching, j'ai utilisé la fonction de Altium qui permet de faire un stitching automatique. Mais j'ai dû faire une partie manuellement pour bien lier les plans là où le stitching automatique ne fonctionne pas.

6 Test de la carte

Pour tester la carte, en premier lieux, je dois monter les deux alimentations (USB-C et Régulateur Step-Down) afin d'éviter d'avoir des problèmes avec les autres composants que je n'aurais pas encore montés.

Si tout cela est validé, le microcontrôleur peut être monté et testé pour ensuite monter le reste des périphériques.

6.1 Test régulateur :

6.1.1 Méthode de mesure

Pour commencer à monter ma carte, j'ai d'abord monté uniquement l'alimentation ainsi que le régulateur pour le tester. J'ai ensuite mis une charge au régulateur afin de me permettre de tester à un courant voulu.

Pour le courant, j'ai choisi de mettre 300 mA en sortie du régulateur :

$$R_{ch} = \frac{U}{I} = \frac{3.3}{0.3} = 11\ \Omega$$

J'ai choisi comme résistance de charge une charge variable jusqu'à 16.5 Ω → R1 27.510.

6.1.2 Schéma de mesure :

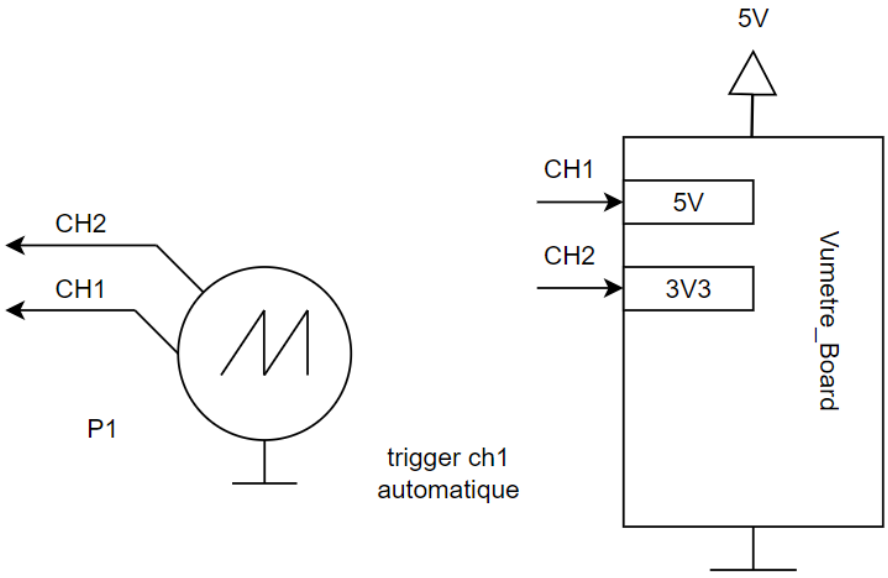


Figure 39 schéma mesure alimentations

6.1.3 Mesure

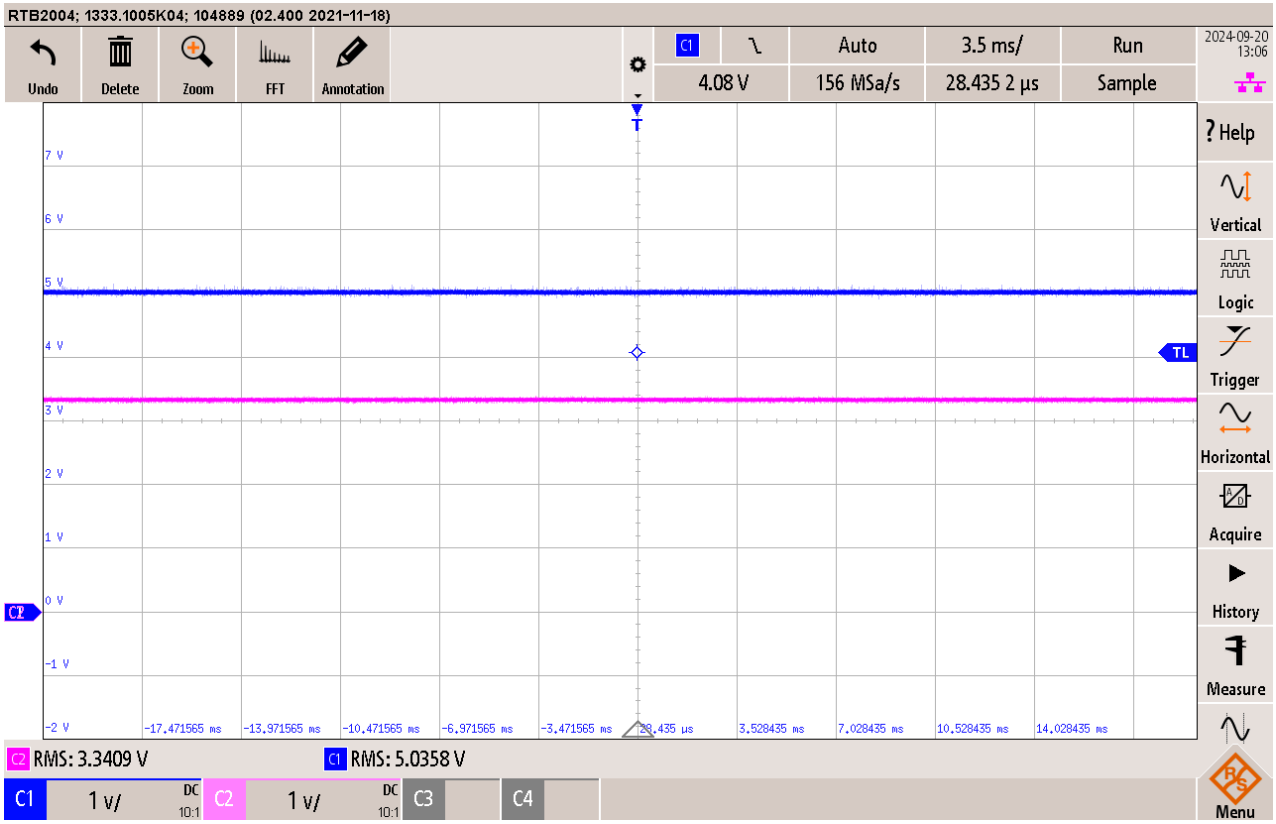


Figure 40 mesure alimentation

Je peux constater que les deux alimentations sont stables et n'ont pas d'oscillation ce qui aurai pu arriver avec le régulateur STEP-DOWN.

Point mesuré	Tensions voulues [V]	Tension obtenue	OK/NOK
3V3	3,3	3,3	OK
5V	5	5	OK

Je peux constater à la mesure que nous avons bien une alimentation de 5V en entrée [C1] grâce à l'USB ainsi qu'une tension de 3V3 à la sortie du régulateur [C2].

6.2Test microcontrôleur :

Pour vérifier mes configurations sur MPlabX ainsi que je n'aie pas de court-circuit sur mon microcontrôleur, je n'ai monté que mon micro ainsi que le bouton reset et le port de programmation.

J'ai dû configurer les pins de programmation du MPlabX (PGDx) pour me permettre la communication :

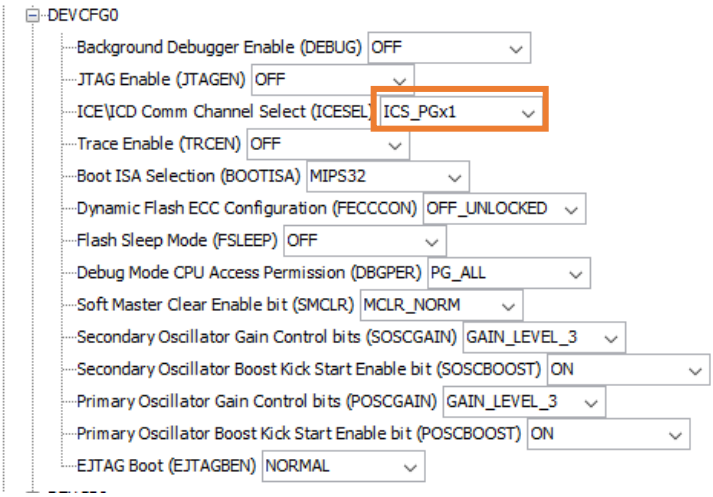


Figure 41 harmony PGDx

J'ai configuré un nouveau projet vierge sur MPlabX avec les configurations correspondant au micro et ai teste de mettre le code vierge sur le micro.

```
**************************************************


Calculating memory ranges for operation...

Erasing...

The following memory area(s) will be programmed:
program memory: start address = 0x1d000000, end address = 0x1d001fff
configuration memory
boot config memory

Programming/Verify complete

Running
```

Figure 42 réponse du microcontrôleur

J'ai pu ainsi constater que je pouvais programmer le microcontrôleur et qu'il répond.

7 Tests et mesures

7.1Configuration des différents pins :

Les différents pins ont été configurée celons ce que nous avons mappés dans la partie schématique.

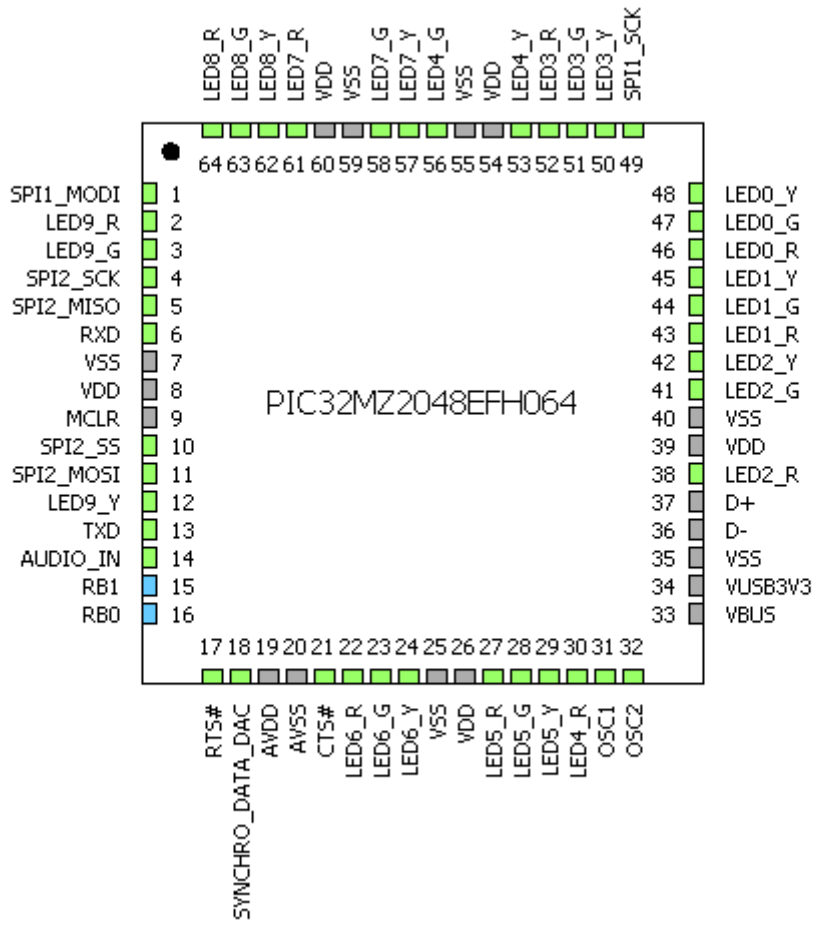


Figure 43 Pins du microcontrôleur

Les pins 37 et 36 sont dédiées à l'USB et ne sont pas modifiable tous les comme les pins du quartz (31 et 32) qui sont automatiquement attribuées.

7.2Calcul timer

Dans la configuration du SYSCLOCK, je peux configurer séparément les différents périphériques du microcontrôleur :

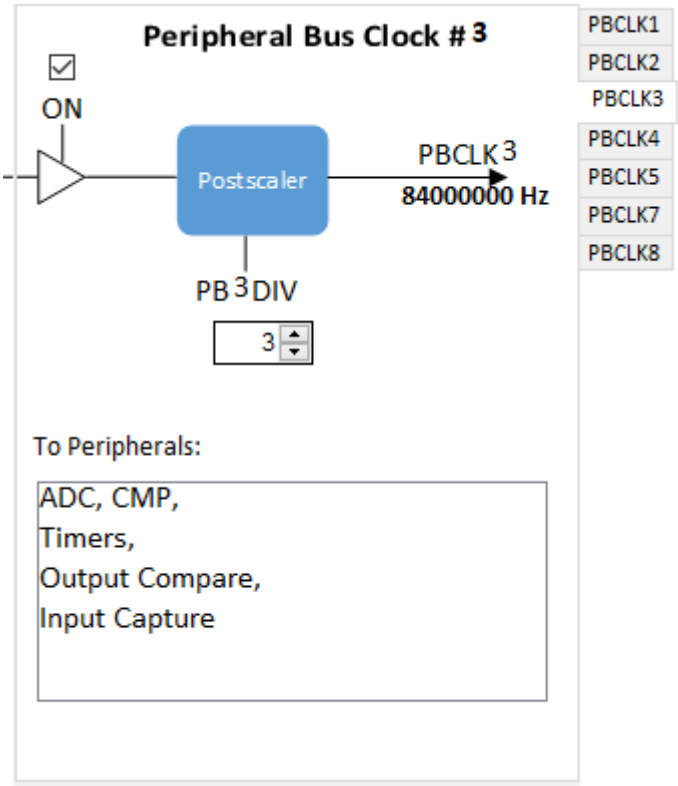


Figure 44 harmony gestion clock timer

Pour avoir une bonne fréquence d'échantillonnage, j'ai choisi de prendre la plus haute fréquence et de la multiplier par 10.

$$F_{ech} = F_{max} * 10 = 20 * 10^3 * 10 = 200\ kHz$$

Ce qui fais que je dois régler mon timmer à 200 [kHz]

Vu que cela sera la fréquence la plus haute dont j'aurai besoin pour mes timers, j'ai utilisé la configuration Peripheral Bus Clock #3 (PBCLK3) afin d'avoir une vitesse de comptage sur mes timers de 84 [MHz] aux lieux de 252 [MHz] car je dois faire clignoter une LED et cela me permet d'avoir une marge afin que cela puisse mieux se voir à l'œil nu.

7.2.1 Calcul timer 1

Je dois calculer le timer 1 avec une fréquence de 200 [kHz] et comment fréquence de clock de 84 [MHz] je dois donc transformer ces deux valeurs en temps :

$$T_{timer1} = \frac{1}{f_{voulue}} = \frac{1}{200 \cdot 10^3} = 5[us]$$

$$T_{Oscillateur} = \frac{1}{f_{oscillateur}} = \frac{1}{84 \cdot 10^6} = 12[ns]$$

Avec cela je peux calculer le nombre de tick nécessaire afin d'avoir la bonne fréquence sur le timer car j'utilise un pré-scaler de 1 donc je ne divise pas la fréquence du clock :

$$T_{timer1} = \frac{T_{timer1}}{T_{Oscillateur}} = \frac{5 \cdot 10^{-6}}{12 \cdot 10^{-9}} = 417\ tick$$

7.2.2 Vérification valeur timer 1

7.2.2.1 Méthode de mesure

Pour cette mesure, j'utilise la fonction toggle du GPIO_OUT de la LED D1 avec la couleur rouge, cela me permettra de mesurer la fréquence de mon timer.

7.2.2.2 Schéma de mesure

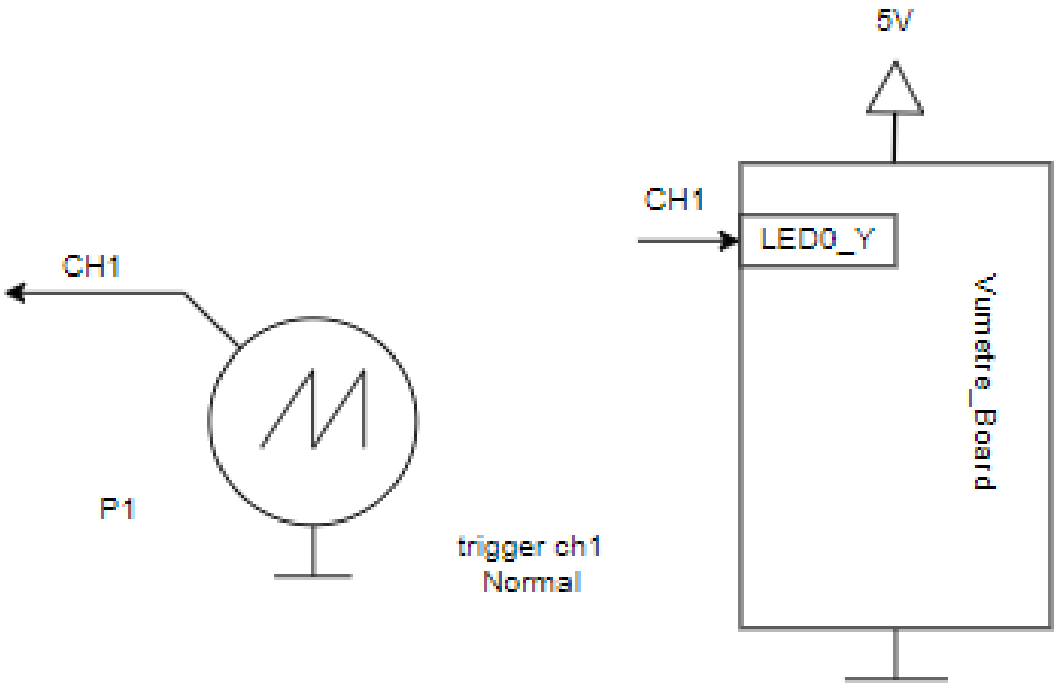


Figure 45 schéma mesure timer 1

7.2.2.3Résultats de la mesure

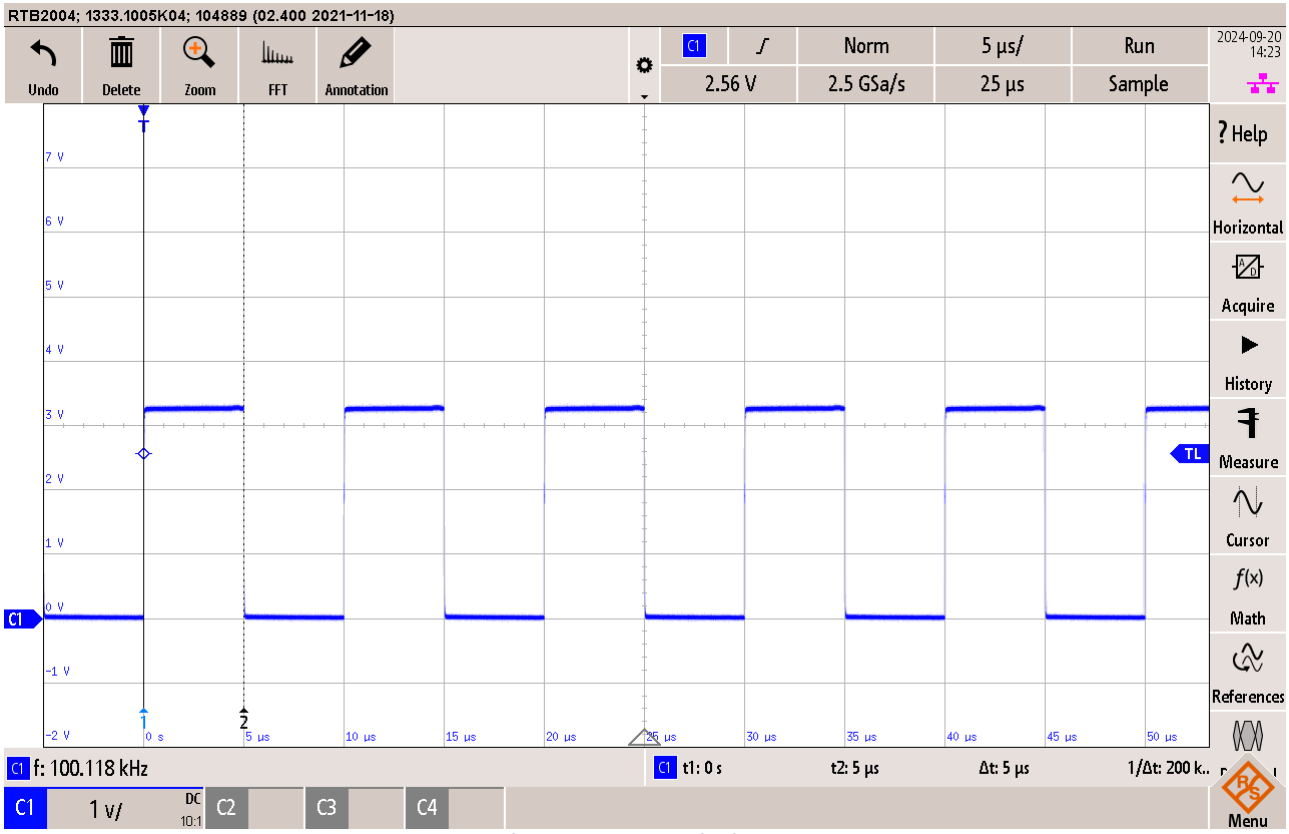


Figure 46 mesure du timer 1

La mesure de la fréquence automatique est fausse car elle mesure entre deux flans montant ce qui fais que je dois multiplier la fréquence par 2 ce qui nous donne bien 100 [kHz].

Point mesuré	Temps entre changement de flan voulu [us]	Temps entre changement de flan mesuré [us]	OK/NOK
LED0_Y	5	5	OK

Il est important que le timer soit précis car la vitesse de celui-ci va déterminer la fréquence d'échantillonnage. Plus il sera précis, plus nos mesures de fréquences le seront aussi.

7.3DAC+ADC

7.3.1 Configuration DAC

Pour configurer le DAC, j'ai utilisé le configurateur graphique d'Harmony :

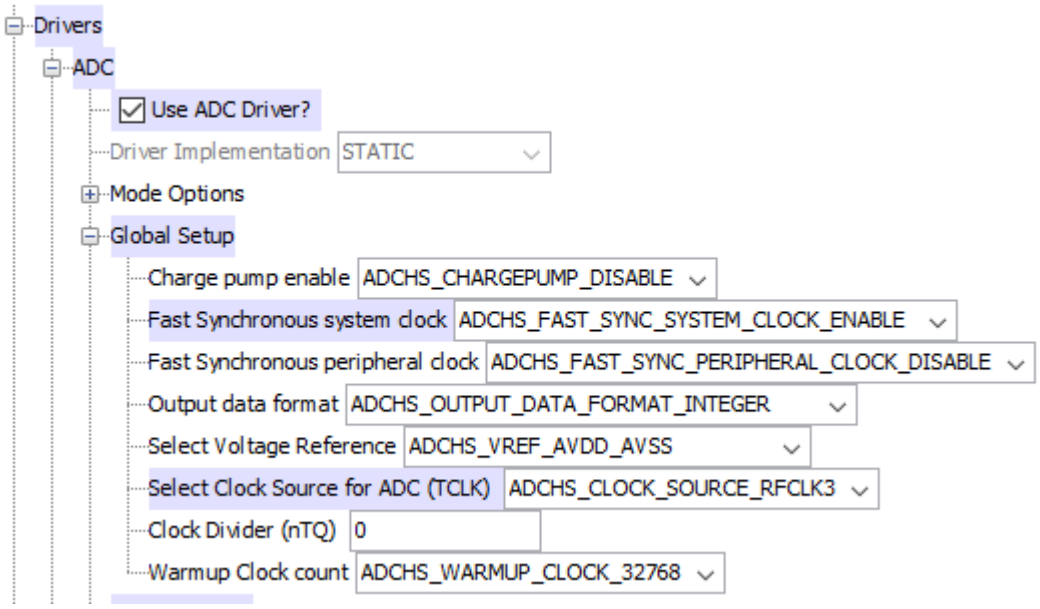


Figure 47 hamony configuration DAC 1

Pour la première partie j'ai sélectionné le clock RFCLK3 car c'est exactement le même que le timer 1 et que je voudrais le synchroniser avec.

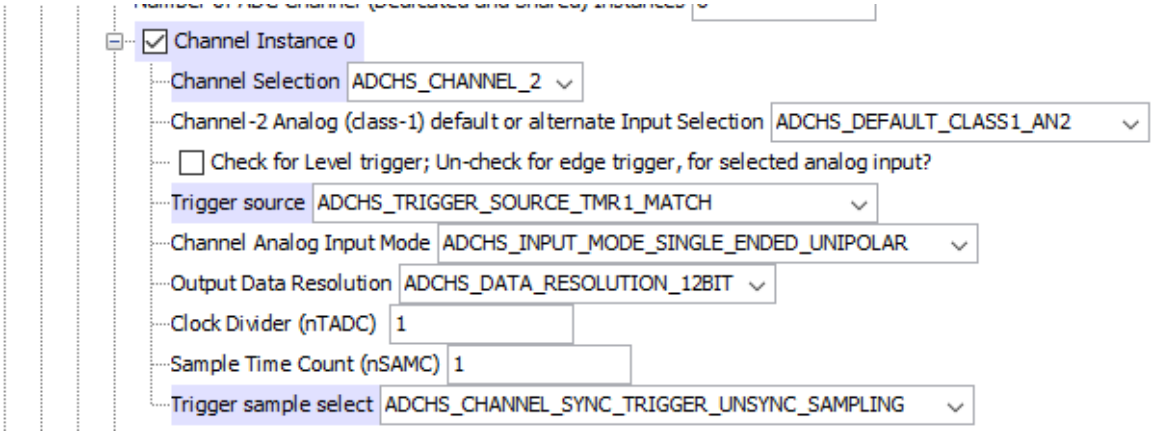


Figure 48 harmony configuration du DAC 2

J'ai configuré l'ADC sur le chanel 2 pour ainsi avoir une lecture sur la bonne pin.
Trigger source est sur ADCHS_TRIGGER_SOURCE_TMR1_MATCH afin que les échantillons soit synchronisé sur le timer 1, cela me permet de garantire la vitesse d'échantillonage stable et correcte.
Le dernier paramètre que j'ai configuré me permet d'avoir le trigger synchronisé mais pas les échantillons.

7.3.2 Configuration ADC

Pour utiliser l'ADC, j'ai utilisé une librairie que j'avais faite qui me permet d'initialiser l'ADC ainsi que de l'utiliser beaucoup plus facilement.

7.3.2.1 Fonctions pour l'ADC

Nom fonction	DAC_Init ()
Paramètre I/O	Aucun
Description	Fonction pour initialiser le DAC

Dans cettes fonction, il y a uniquement des fontions permettant d'initaliser le SPI afin de communiquer correctement avec le DAC.

Nom fonction	Dac_Write
Paramètre I/O	Uint16_t ech
Description	Fonction pour envoyer une trame à l'ADC, envoie 2 fois 8bits en SPI.

Cette fonction pemet d'avoir en entrée les datas sur 16 bits et ainsi pouvoir les mettre dans l'ordre et envoyer deux fois 8 Bits à l'aide du SPI. Pour envoyer ces deux trammes de 8 bits j'uilise la fonction suivante :

Nom fonction	Spi_write
Paramètre I/O	Uint8_t Val
Description	Envoie la valeur sur le SPI et attend que la trame ait fini de s'envoyer.

7.3.3 Test de l'ADC et et DAC

7.3.3.1 Méthode de mesure

Pour tester l'ADC et le DAC, le but est simplement de mettre une tension fixe sur l'entrée audio et de la reproduire à la sortie. Car le DAC et l'ADC sont deux périphériques fonctionnants sur la même … ce qui me permet de mettre directement en sortie ce que je peux lire en entrée.
Pour commencer, je vais mesurer le signal que je mets en entrée ainsi que la sortie du DAC. Cela pourra me confirmer que les deux périphériques fonctionnent. Le but est d'avoir la même tension à l'entrée que à la sortie.

Pour ensuite mesurer les datas envoyées sur le DAC en SPI et pouvoir décoder la trame. Je pourrai ainsi la comparer en utilisant MPlabX en debugger ce qui me permettra de voir la valeur que je lis pour ensuite la comparer avec ce que j'envoie.

7.3.3.2Schéma de mesure

7.3.3.2.1 Mesure tension entrée / sortie

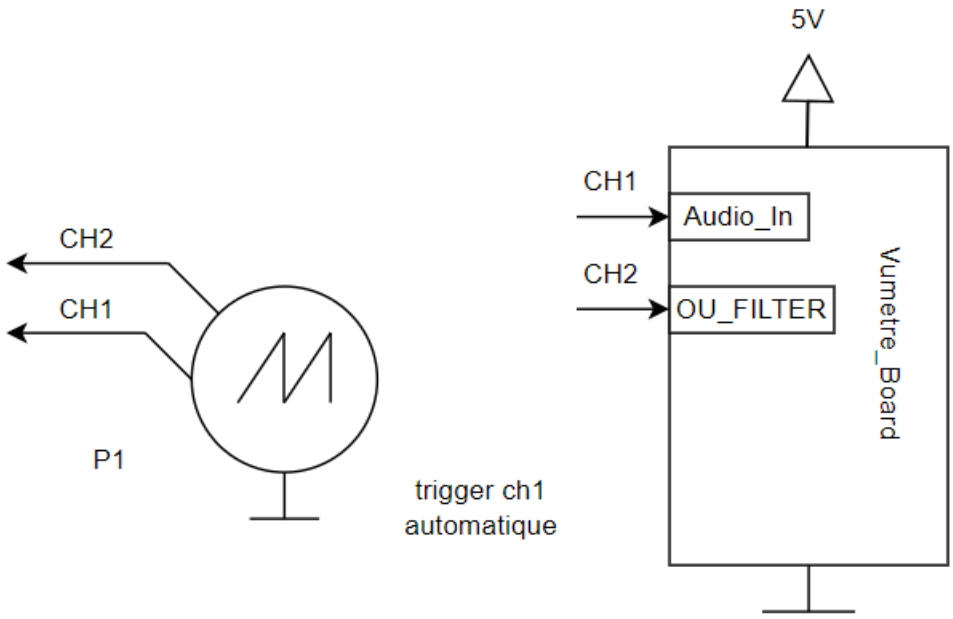


Figure 49 schéma de mesure tension entrée / sortie

7.3.3.2.2 Mesure Datas

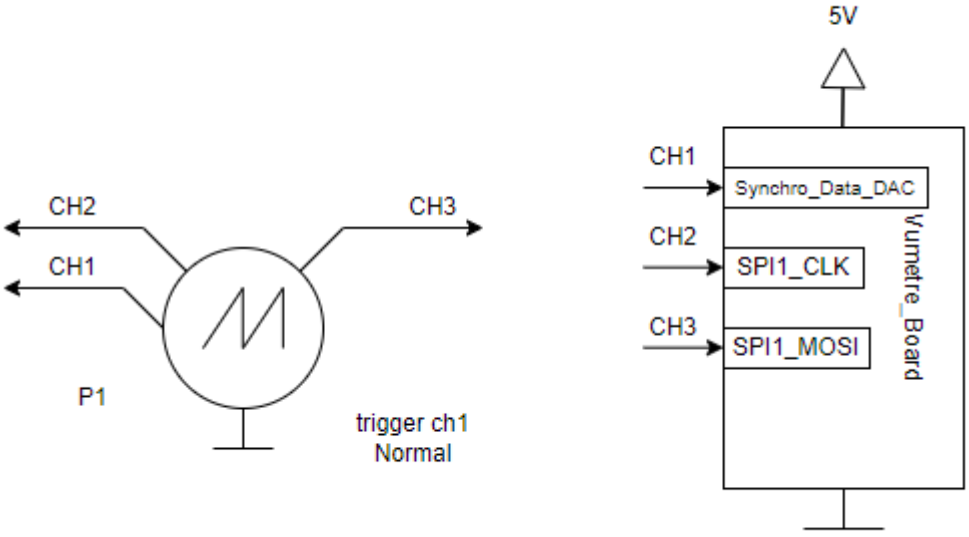


Figure 50 schéma de mesure datas

7.3.3.3 Mesure tension entrée / tension de sortie

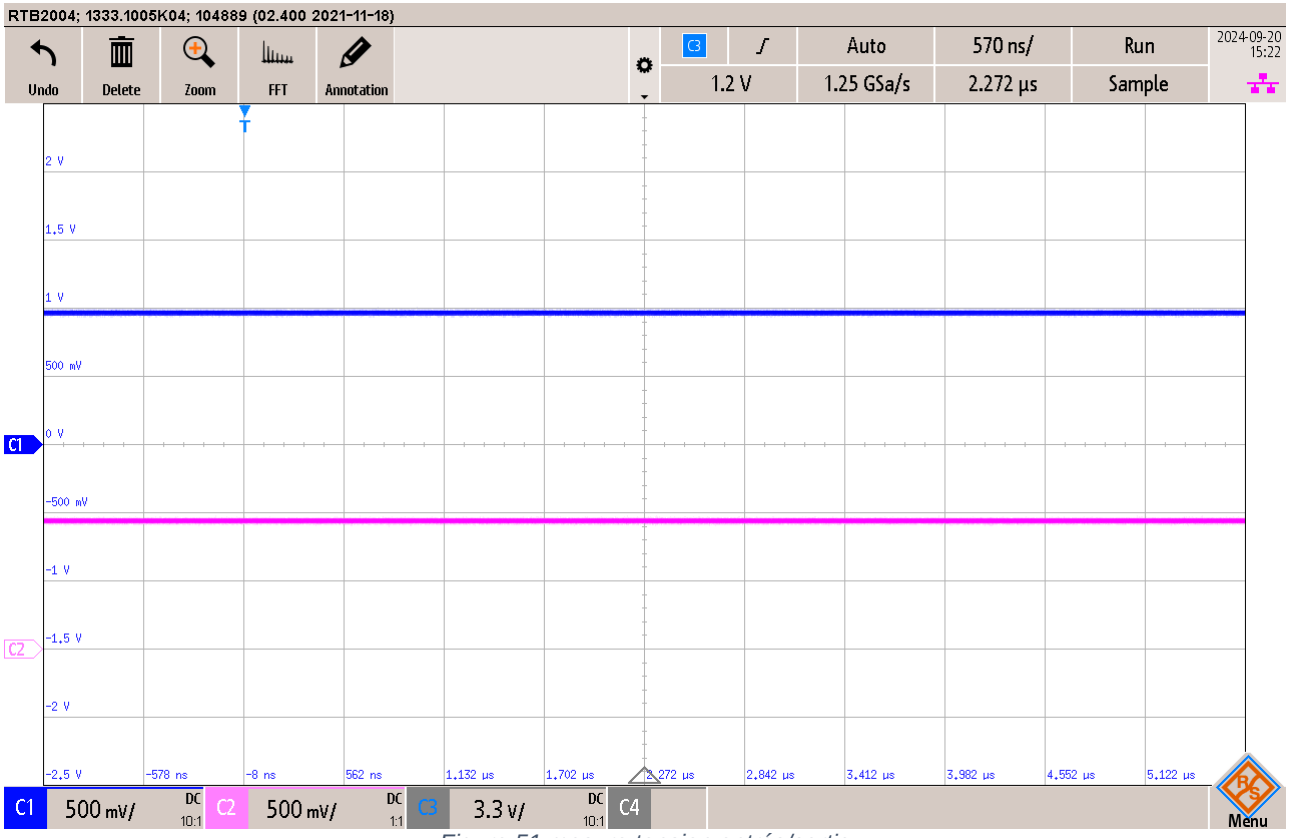


Figure 51 mesure tension entrés/sortie

Sur le CH1 nous pouvons constater que nous avons bien une tension de 1 [V]. Sur la sortie qui est le CH2 nous avons une tension qui est à -1/2 carré de 1 [V] ce qui nous donne ~950 [mV] ce qui est tolérable et nous montre que nous avons en sortie une tension similaire qu'en entrée.

7.3.3.4Mesure des datas

7.3.3.4.1 Mesure ADC

Pour avoir une variable de référence, j'ai utilisé le debugger de MPlabX, mis un point d'arrêt avant d'envoyer la trame pour ainsi lire la valeur que nous voulons envoyer.
Voilci les valeurs mesurée pour 1 V :

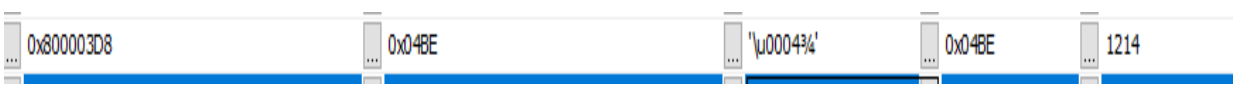


Figure 52 valeur lue par l'ADC

Nous avons donc une valeur de 1214. Je peux calculer ce que ça donne en tension :

$$Taille\ ADC = 12\ bits \rightarrow\ résolution = 0 - 4096\ [-]$$
$$\frac{3{,}3V}{4096} = \frac{?}{1214} - règle\ de\ 3 \rightarrow \frac{3{,}3 * 1214}{4096} = 978\ [mV]$$

Ce qui nous donne bien une valeur très proche des 1 [V] recherchés

7.3.3.5Mesure DAC

Pour mesurer la trame du DAC, j'ai configuré la vitesse du SPI à 300 [kHz] car si je la mets trop haute, son fonctionnement n'est compromis cependant les valeurs sont beaucoup moins lisibles et décodables.

Pour lire la trame, j'ai utilisé la fonction Protocol de l'oscilloscope afin de de pouvoir décoder la trame en direct. J'ai pu ainsi configurer cette fonction affine de lire sur 12 bits dans le bon sens et de l'afficher en décimale :

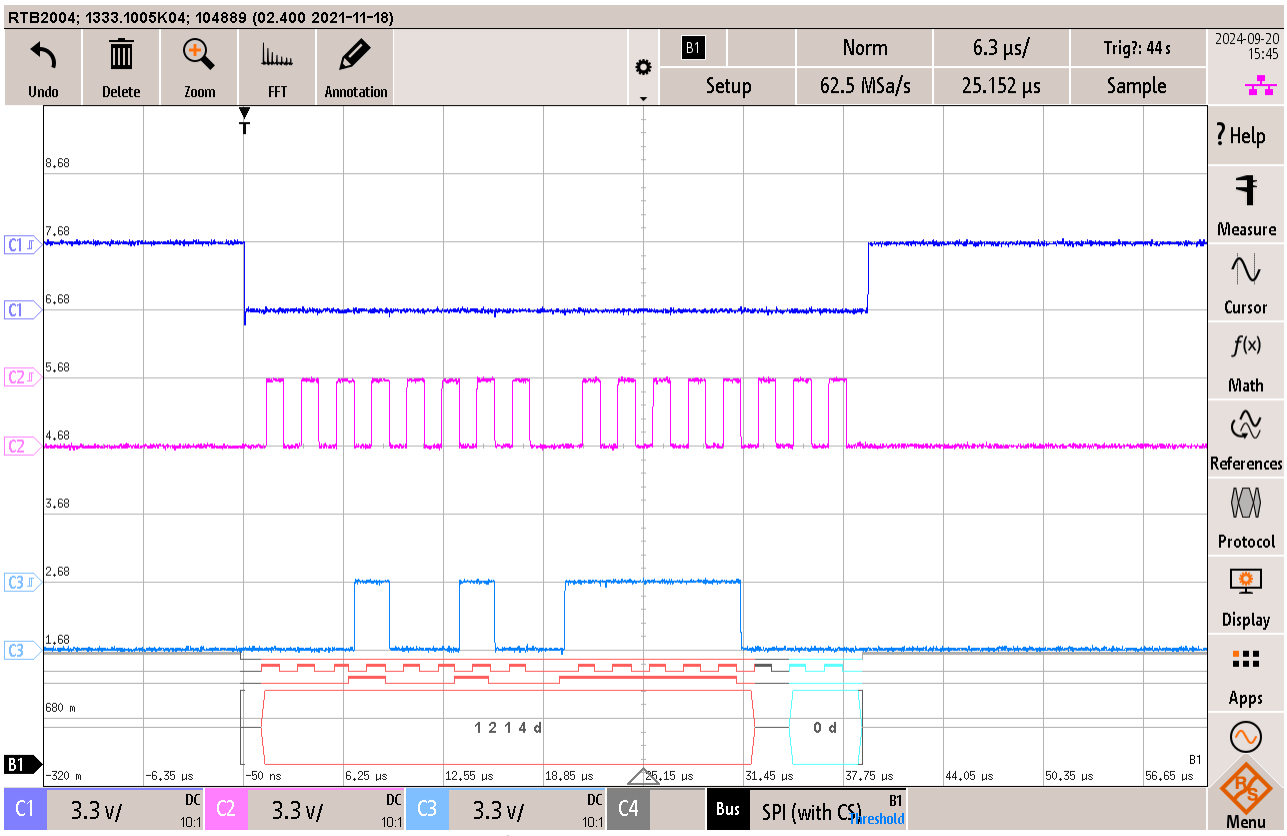


Figure 53 mesure trame DAC

Ce qui me permet de faire corresponde les datas envoyer sur le DAC avec les valeurs lues sur l'ADC. Nous retrouvons ainsi bien la valeur 1214 que nous avions dans l'ADC.

7.4USART

7.4.1 Configuration

Le but est de tester si mon FTDI fonctionne correctement. Pour cela je vais configurer l'UART afin d'activer les interruptions. Comme je veux simplement tester ce composant, je n'ai besoin uniquement de savoir si je reçois une trame correctement.

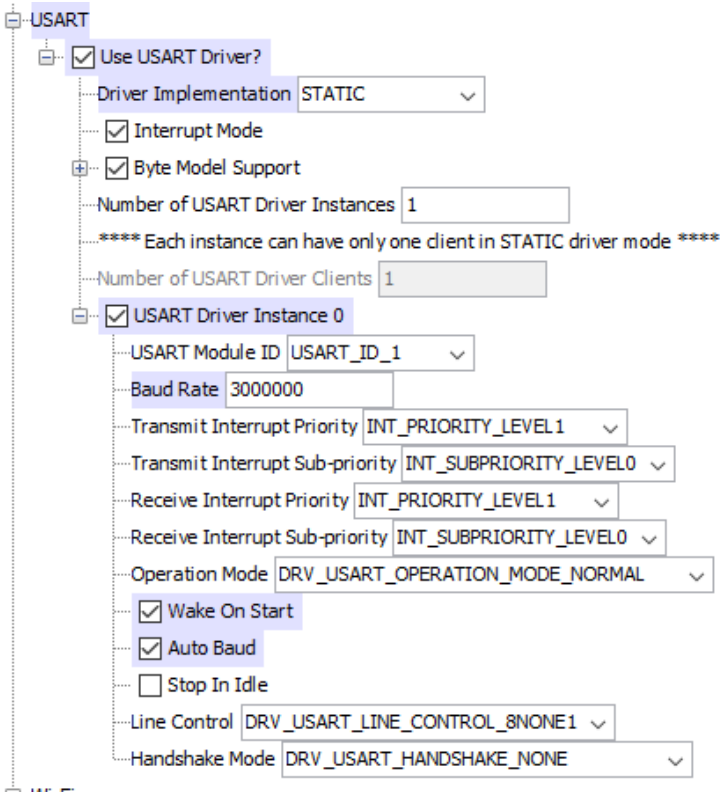


Figure 54 harmony configuration UART

J'utilise un UART static avec interruption. Le baud rate n'est pas important car j'utilise la fonction auto-Baud ce qui permet au micro à caller son baud rate sur le FTDI.

Pour envoyer une trame, j'utilise l'application PuTTY. Je crée une cession sur le COM9 car c'est le COM auquel j'ai connecté mon USB. Pour trouver ce COM, j'ai branché l'USB au PC pour ensuite aller dans le gestionnaire de périphérique et le trouver.
Le type de connexion est en série sur Telnet. La vitesse est de 11'5200 bauds, c'est la vitesse standard de l'USB

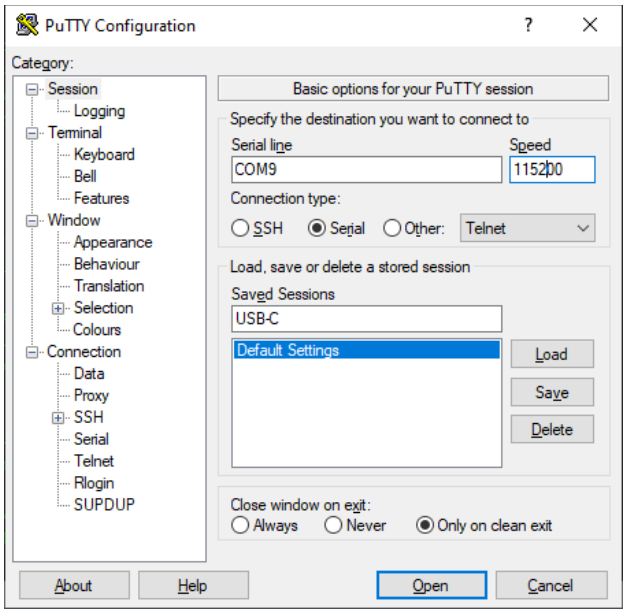


Figure 55 PuTTY configuration USB

7.4.2 Méthode de mesure

Pour vérifier que je reçoi une trame d'UART, je fais allumer/éteindre la LED0 en jaune.
Le but est de mesurer la tram UART et ainsi vérifier que la LED réagisse.
Je lance le programme PuTTY et appuie sur une de mes touches du clavier. Afin de vérifier que le décodage se fasse correctement, je décide l'envoyer un caractère 'a' pour savoir ce que je dois lire sur la trame.

7.4.3 Schéma de mesure

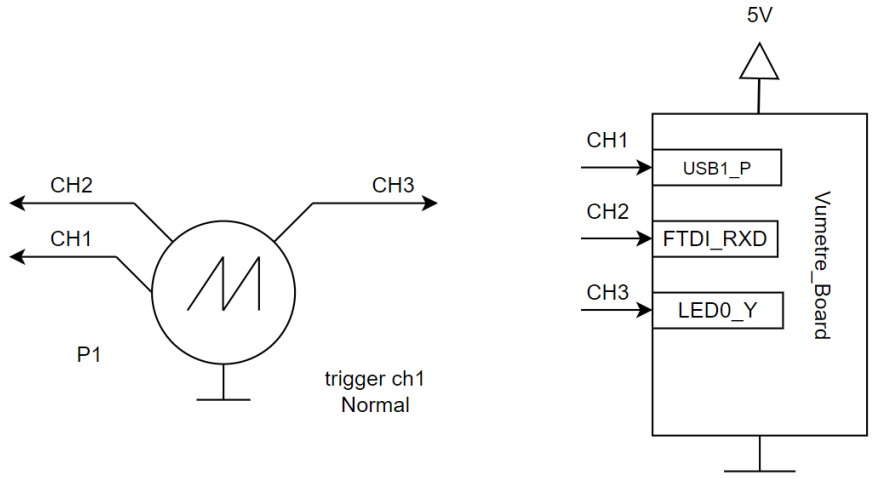


Figure 56 schéma mesure trame UART

7.4.4 Résultat et analyse

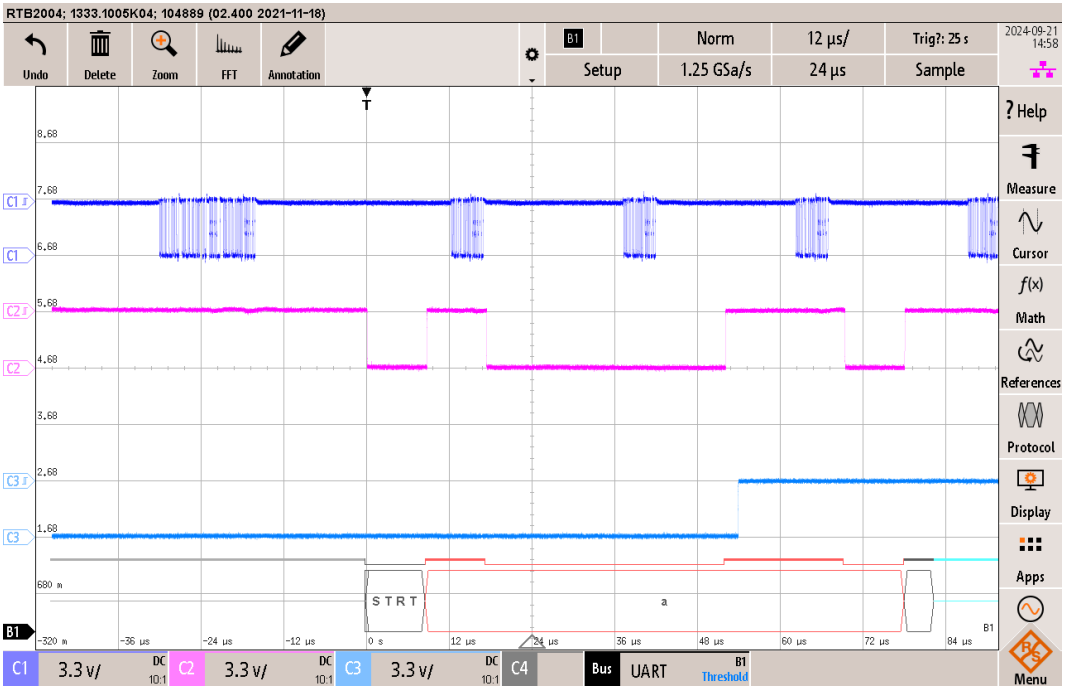


Figure 57 mesure trame UART

Je peux constater que sur le signal USB (CH1) environ 30 [us] avant que la trame UART, le signal est beaucoup plus long ce qui signifie que des datas sont transmises.

Le signal CH2 est la trame UART, sur le décodage (B1) je peux constater que j'obtiens bien le caractère 'a'. Quand la trame est détectée, la LED 0 s'allume en jaune ce qui montre que je rentre bien dans l'interruption.

8 Software

8.1 Diagrammes

8.1.1 Machine d'état générale

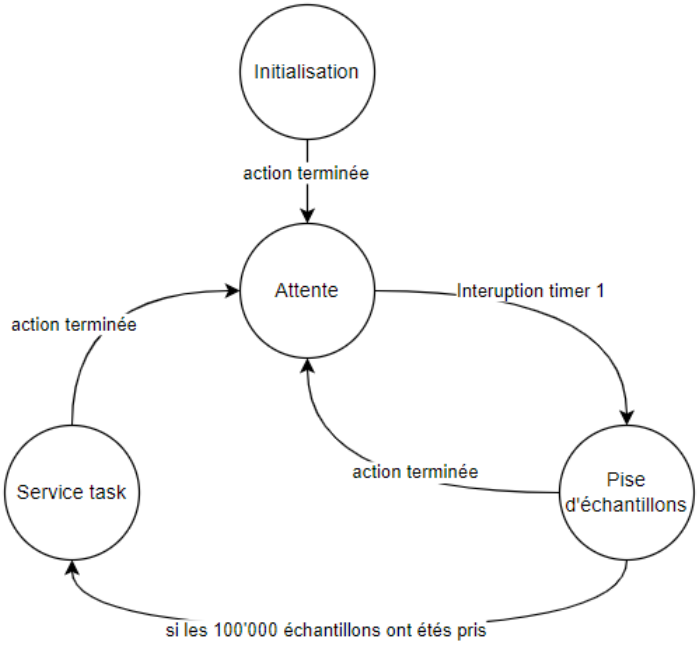


Figure 58 machine d'état générale

Dans l'état initialisation, j'initialise le timer, l'UART, les SPI et j'éteins toutes les LEDs. Dans la fonction d'attente, je ne fais rien. Toutes les 5 [us] je vais prendre un échantillon que je viens mettre dans le tableau d'échantillons :

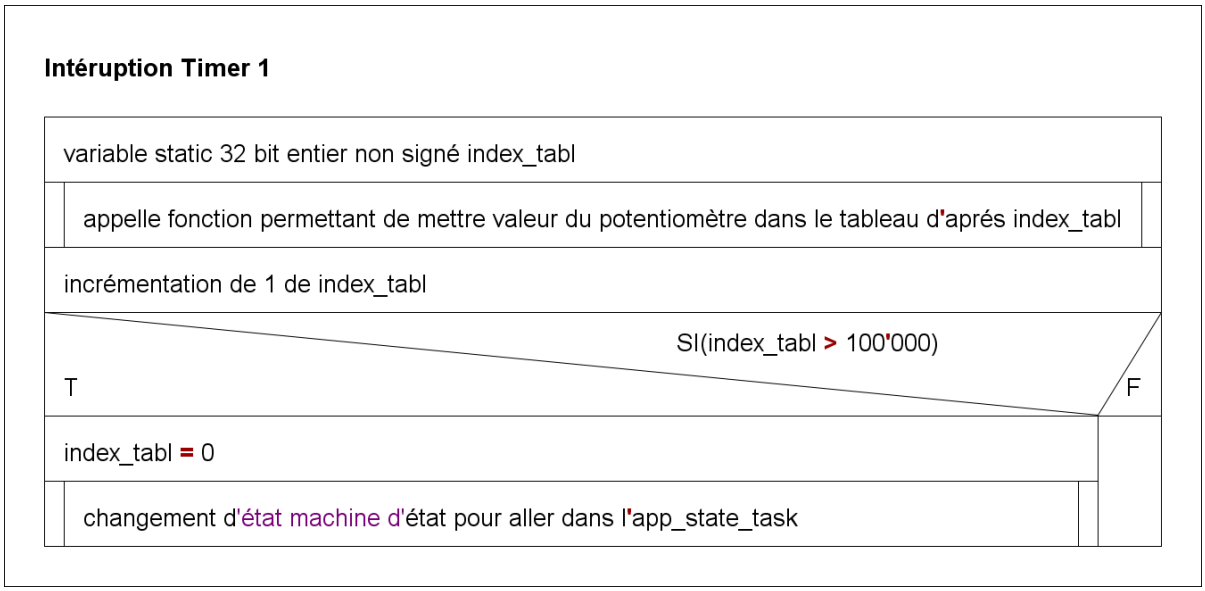


Figure 59 structogramme interruption timer 1

J'ai choisi une valeur de 100 000 car le pire des cas est quand nous mesurons un signal de 20 [Hz].

Une fois que j'ai fini de prendre des échantillons, dans le service task, mon but est de faire en sorte d'avoir le moins de code possible. Pour cela j'appelle 3 fonctions afin de gérer le filtre.

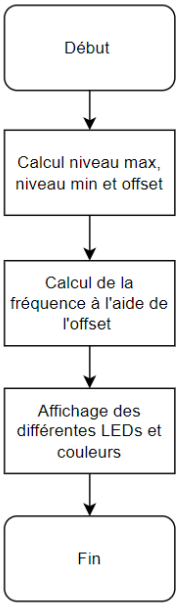


Figure 60 App_service_task

Ces 3 fonction sont dans un autre fichier .C qui se nomme Gestion_Filtre.c.
Quand nous mesurons un signal le but est de mesurer la seconde courbe vers le haut si le début des échantillons est positif :

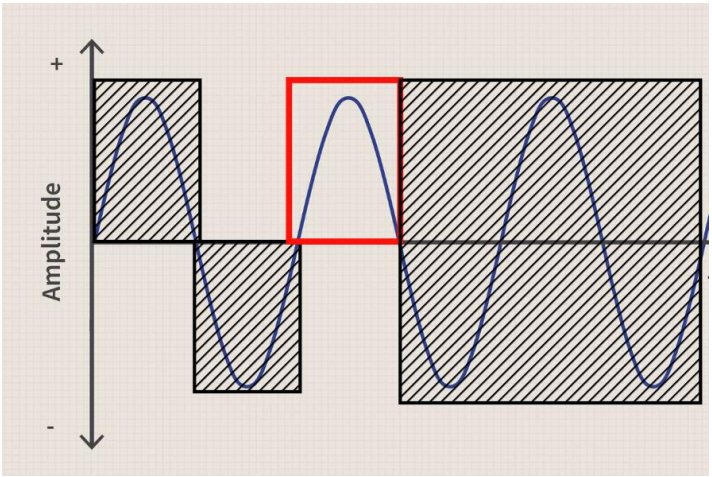


Figure 61 démonstration de ce que je veux mesurer

Car si je commence par un signal positif, je ne peux pas réellement savoir à quel moment je suis. Je peux donc avoir 3 fois 5 [ms] comme temps maximum que je dois mesurer

$$Tmax = \frac{1}{20} * 3 = 15\ [ms]$$

$$T_{ech} = \frac{1}{200\ *\ 10^3} = 5\ [us]$$

$$Ech_{min} = \frac{7.5 * 10^{-3}}{5 * 10^{-6}} = 3000$$

Et pour avoir une sécurité, j'ai décidé de prendre 100'000 échantillons.

8.2Librairies

8.2.1 Gestion_Filtre.c

Pour pouvoir mesure la tension du signal ainsi qu'avoir la valeur du 0, je dois trouver la valeur minimum ainsi que la valeur maximum du signal pour pouvoir calculer la valeur du '0'.

Nom fonction	Value_search
Paramètre I/O	uint16_t t_Values [NOMBRE_ECH],uint16_t *Value_max,uint16_t *Value_min,uint16_t *Val_Zero
Description	Permet de lire l'entièreté du signal pour déterminer la valeur min et la valeur max et l'offset

Et voici le structogramme de cette fonction :

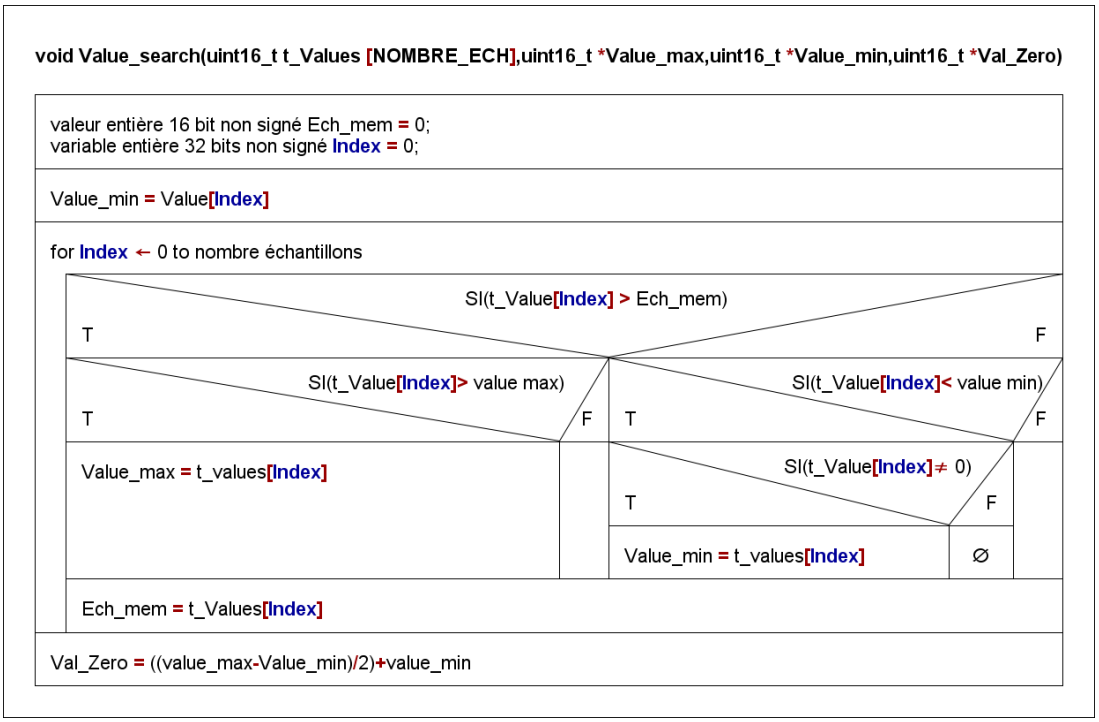


Figure 62 structogramme fonction Value_search

J'essaye d'éviter au maximum que la valeur du min soit 0 car cela peut provenir d'un problème d'échantillonnage et cela bloque la valeur minimum à 0.

La fonction suivante permet de mesurer le nombre d'échantillons que nous pouvons mesurer sur un cycle qui est au dessus du 0.

Nom fonction	Conv_Values
Paramètre I/O	uint16_t t_Values [NOMBRE_ECH], uint16_t *Nb_ech_pos,uint16_t Val_Zero
Description	Permet de compter le nombre d'échantillons lors du deuxième demi-sinus positif

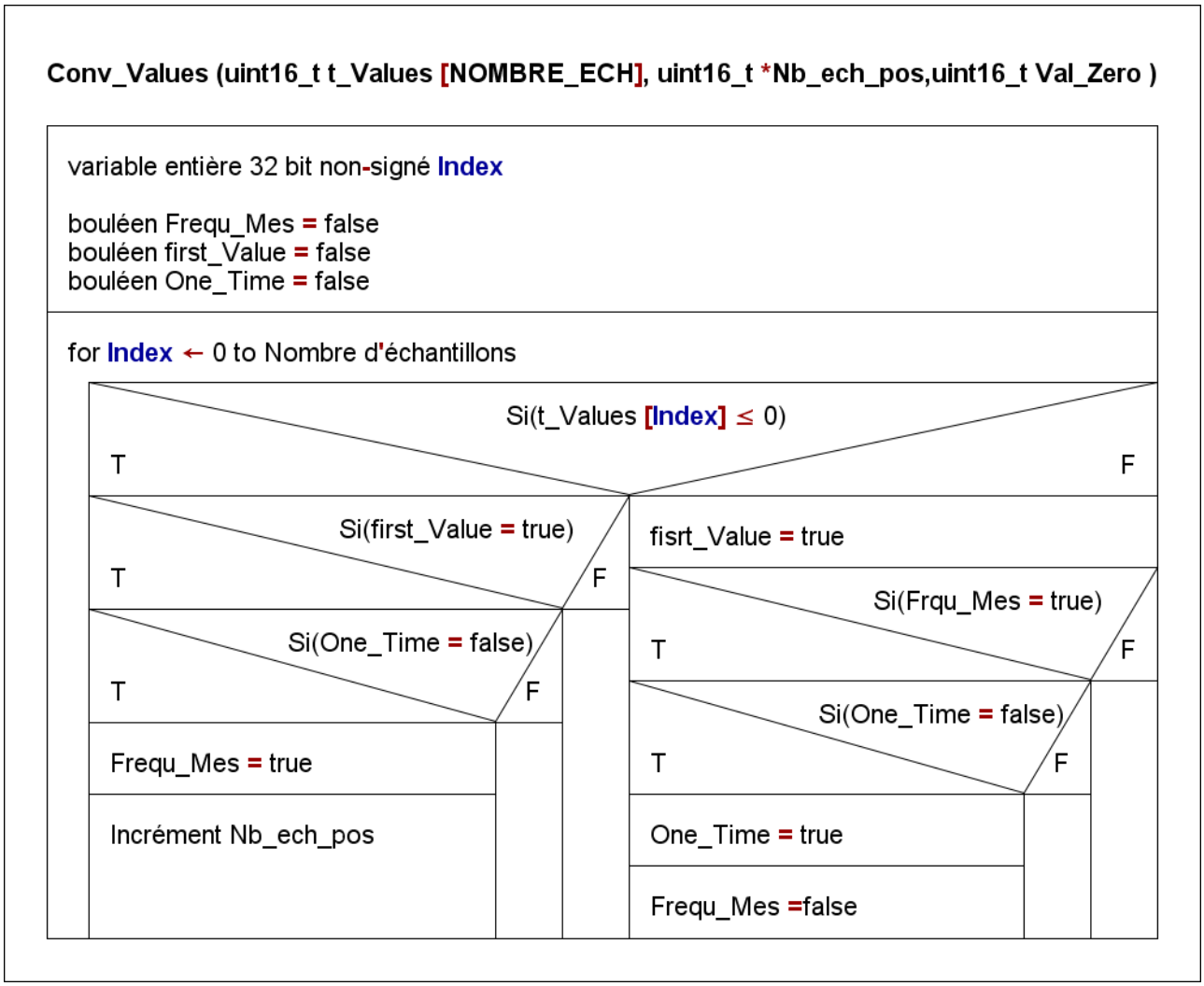


Figure 63 structogramme fonction Conv_Values

Ainsi je peux directement avoir le nombre d'échantillons sur cette période, les 3 variable booléennes me permettent de d'avoir uniquement le moment voulu dans cette mesure.

La dernière fonction est une fonction qui permet d'activer les LEDs en fonction du nombre d'échantillons que j'ai compté juste avant. J'utilise la tension max pour savoir de quel couleur mes LEDs doivent être.

Nom fonction	Calcul_Frequence_Led
Paramètre I/O	uint16_t Nb_ech,uint16_t Tension_max
Description	Permet de déterminer les LEDs à allumer ainsi que la couleur par apport à la fréquence

Pour déterminer la fréquence, je regarde combien j'ai d'échantillons et je peux ainsi déterminer la fréquence.

$$Nobre\ d'échantillons\ pile = \frac{fréquence\ voulue}{200\ [kHz] * 2}$$

Ce qui me donne les valeurs suivantes :

Fréquences voulue	Total échantillon 1 période	Nombre échantillon pile
20	10000	5000
50	4000	2000
100	2000	1000
200	1000	500
500	400	200
1000	200	100
2000	100	50
5000	40	20
10000	20	10
20000	10	5

Dans le code, j'ai utilisé un des variables au lieux de définitions parce que dans le futur, le but sera de pouvoir modifier ces valeurs à l'aide d'une application C#.

8.2.2 Dac_ad5620.c

Cette librairie est décrite dans la partie 7.3.1

8.2.3 Gest_LED.c

La première fonction me permet d'éteindre toutes les LEDs en utilisant la fonction de MPlabX LEDx_xOff ().

Nom fonction	All_LED_Off
Paramètre I/O	Aucun
Description	Fonction pour éteindre toutes les LEDs

La fonction Gest_LED me permet de de gérer plus facilement les différentes LEDs et couleur. En entrée, je peux sélectionner la LED, la couleur et si je veux qu'elle soit éteinte ou allumée.

Nom fonction	Gest_LED
Paramètre I/O	uint8_t Num_LEDs,uint8_t Color, bool LED_On_Off
Description	Fonction pour sélectionner une led et une couleur pour ainsi l'éteindre ou l'allumer

9 Modification à apporter

Lors du montage, je me suis rendu compte que le connecteur BNC à l'entrée (P2) a les signaux qui sont inversés.
Dans la version B, ce connecteur a été inversé ce qui résous ce problème.

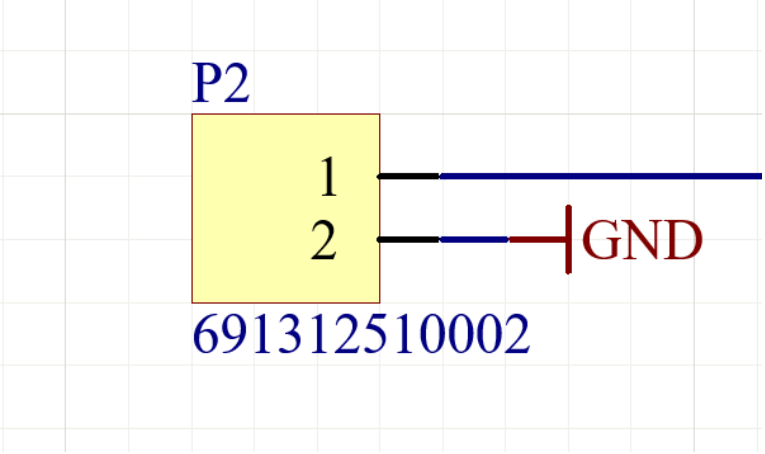


Figure 64 schématique modification

10 Evaluation du projet

10.1 Etat d'avancement du projet

10.1.1 Gestion des LEDs

- Monté et testé → LED 3 rouge ne fonctionne pas
- Programme fonctionnel

10.1.2 Échantillonnage

- Programme fait et testé

10.1.3 Filtrage

- Testé mais ne fonctionne pas à 100 %

10.1.4 ADC

- Monté et testé
- Pas implémenté au programme

10.1.5 UART

- Monté et testé
- Réceptionne une trame mais ne la décode pas

10.2 Reste à implémenter

- Filtre numérique plus performant voir FFT
- Communication UART pour modifier les valeurs des filtres
- Gestion de la mémoire
- Sortie filtrée
- Communication avec une matrice à LED

11 Auto-analyse

11.1 Acquis du travail de diplôme

Ce travail m'a permis de beaucoup en apprendre sur le filtrage numérique car nous utilisons beaucoup de filtres analogiques. C'était donc un sujet que je n'avais jamais traité et qui m'as demandé beaucoup de réflexions. J'ai beaucoup apprécié cela car ça rajoutait un certain défi et finalement cela m'a fait me rendre compte qu'avec des bases solide, je pouvais beaucoup plus facilement appréhender le sujet.
Cela m'a conforté dans le choix de la profession et m'a conforté dans le fait de continuer les études afin de me perfectionner dans ce domaine.

11.2 Prise en compte de l'environnement social

Ce projet n'a pas forcément une grande utilité mais c'est un système qui est beaucoup utilisé surtout dans les enceinte Bluetooth actuelles. Cela peut rajouter un certain design à une table de mixage ou un autre appareil audio mais ne rajoute pas forcément de plus-value. Malgré cela s'est beaucoup plus tape à l'œil d'avoir une certaine visualisation de ce qu'on écoute.

11.3 Prise en compte de l'environnement naturel

Lors du design, j'ai fait attentions à plusieurs choses afin d'améliorer la partie fabrication du produit. Surtout de garder un espace suffisant entre le composant pour faciliter la brasure pour mieux laisser passer la panne à braser ainsi que de mettre tous les composant sur le même côté du PCB pour éviter lors du montage de devoir détourner la carte. Ce qui peut être largement améliorer est le boîtier. Vu que je produisais un prototype, j'ai choisi un boîtier que j'ai usiné. Cela pourra être améliorer en faisant un boitier 3d sois même ce qui demanderai un certain travail sur le design ou en commandent un boîtier sur mesure ce qui rajouterai beaucoup de coût au projet. Ce qui joins un second problème avec le boîtier qui est la taille. Lors de la recherche du boîtier, je me suis rendu compte que je devais aligner les LEDs tout en gardant un certain espace afin d'éviter que différentes LEDs polluent celles d'as côté. J'ai choisi une longueur suffisante mais même en prennent la largeur la plus petite cela reste trop large. Le mieux sera d'en avoir un qui est moins large afin d'éviter d'avoir autant d'espace non-utilisé.
Ce qui est une peu compliqué aussi s'est de faire des filtres numériques en C. Car en python il existe des librairies pour utiliser la méthode FFT qui sont beaucoup plus simple à utiliser. Pourquoi ne pas partir sur un arduino ou un Raspberry tout en vérifiant que ceux-ci permettent un traitement de signal audio.

11.4 Leadership et développement personnel

Lors de ce travail, j'ai dû énormément me débrouiller par mes propres moyens et trouver des solutions seule, cela m'a été très stimulant et m'as ainsi demandé de faire beaucoup de recherches. Ce qui m'as beaucoup manqué est une meilleure méthodologie de travail car j'avais tendance à beaucoup me disperser. Une solution aurait été d'utiliser un logiciel comme Jira qui m'aurais permis de créer des tâches à réaliser afin de mieux fragmenter le travail et se concentrer sur une seule tâche à la fois. La relation élève-professeur étais beaucoup moins marquée ce qui a donné un certain confort et aide aussi à mieux discuter sans avoir la peur du supérieur. Cela permet ainsi un dialogue qui permet de mieux apporter ces propres idée et/ou améliorations à apporter.

12 Conclusion

Lors de ce projet, j'ai dû réaliser un vu mètre fréquentiel. Le but était de faire entrer un signal audio et de pouvoir filtrer les fréquences pour ainsi les interpréter à l'aide de LEDs RGB. Le principe est que le signal d'entrée soit retrouvé en sortie sans aucune retouche afin que le système agisse comme une boîte noire sans interférer avec la musique.

Malheureusement, mon filtrage de fréquence n'est pas opérationnel et à l'heure actuelle je ne peux que lire la fréquence sur des signaux constant. Le but pour la suite serait de trouver comment filtrer un signal audio qui aurai plusieurs fréquences. Pour cela il semblerait que la méthode FFT soit plus appropriée et pour l'utiliser des librairies sont disponibles. Il reste des fonctionnalités à implémenter comme l'application C# qui permettrai de modifier les filtres en direct ainsi que de faire la sortie filtrée afin de visualiser comment nous filtrons ces signaux.

Toutes les différentes partie Hardware son testées et sont fonctionnelles. Il ne reste plus que faire une librairie pour utiliser correctement l'UART. La partie qui demandera encore beaucoup de retouche est la partie Software. Surtout à implémenter certaines fonctions comme la mise en mémoire, la gestion avec l'application C# et la sortie DAC.

Loïc David Le 24.09.2024

Loïc David
Le 24.09.2024

13 Annexes

13.1 Liste appareils de mesure

Désignation	Marque	Type	Caractéristique	N° d'inventaire
G1	Sefram	GPS-3303	Générateur de tension	ES. SLO2 00.00.54
G2	Agilent	33521A	Générateur de signal	ES.SLO2.00.00.97
P1	Rohde & Schwarz	RTB2004	Oscilloscope	ES. SLO2.05.01.04

13.2 Bibliographie

13.2.1 Datasheets

13.2.1.1 Microcontrôleur

PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Family Silicon Errata and Data Sheet Clarification (microchip.com)

13.2.1.2 Régulateur 3V3

173950336.pdf (we-online.com)

13.2.1.3 DAC

AD5620/AD5640/AD5660 (Rev. G) (analog.com)

13.2.1.4 LED RGB

APTF1616SURKCGKSYKC(Ver.7A) (kingbrightusa.com)

13.2.1.5 FTDI

FT232RN (ftdichip.com)

13.2.1.6 MOSFET

https://www.infineon.com/dgdl/irlml2402pbf.pdf?fileId=5546d462533600a401535664e5ef25fa

13.2.2 Liens internet

13.2.2.1 Filtre RF

Electronique - Realisations - Filtre RF (sonelec-musique.com)

13.2.2.2 USB-C

USB Type-C — Wikipédia (wikipedia.org)

13.2.2.3 Prise jack

https://source.android.com/docs/core/interaction/accessories/headset/jack-headset-spec?hl=fr