Une image contenant texte, clipart

Description générée automatiquement 

Génie électrique / électronique 3e semestre 2022-2023

**Rapport de Projet**

2227\_BoiteBruitage

|  |
| --- |
|  |

Réalisé par : À l’attention de :

Loïc David Serge Castoldi

Jonathan Braun

Début Fin

24 novembre 2022 22 décembre 2022

Table des matières

[Pré-étude 1](#_Toc137782441)

[Schéma block 1](#_Toc137782442)

[Détails des blocks 2](#_Toc137782443)

[(A) Alimentation 10-15 VAC 2](#_Toc137782444)

[(B) Pont de greatz 2](#_Toc137782445)

[(C) Redresseur 2](#_Toc137782446)

[(D) Entrées 2](#_Toc137782447)

[(E) Optocoupleur 3](#_Toc137782448)

[(F) Potentiomètre 3](#_Toc137782449)

[(G) Port USB ou carte micro-SD 3](#_Toc137782450)

[(H) µc 3](#_Toc137782451)

[(I) DAC 4](#_Toc137782452)

[(J) Ampli 4](#_Toc137782453)

[(K) Haut-parleur 4](#_Toc137782454)

[Sortie Jack 4](#_Toc137782455)

[Boîtier 4](#_Toc137782456)

[Choix composant critique 5](#_Toc137782457)

[Evaluation des couts 1](#_Toc137782458)

[Planification 1](#_Toc137782459)

[Conclusion et perspective 2](#_Toc137782460)

[Rapport de schématique 3](#_Toc137782461)

[Schéma générale 3](#_Toc137782462)

[Block schéma 3](#_Toc137782463)

[Connecteurs entrée-sortie 3](#_Toc137782464)

[Alimentations et pont de Greatz 3](#_Toc137782465)

[Régulateur 5V 4](#_Toc137782466)

[Régulateur 3,3V 5](#_Toc137782467)

[Microcontrôleur 6](#_Toc137782468)

[Port de programmation 7](#_Toc137782469)

[Potentiomètre 7](#_Toc137782470)

[Carte SD 8](#_Toc137782471)

[Entrées avec les optocoupleurs 9](#_Toc137782472)

[Ampli audio 10](#_Toc137782473)

[Sortie Jack : 11](#_Toc137782474)

[Conclusion rapport schéma 12](#_Toc137782475)

[Routage 13](#_Toc137782476)

[Largeur des pistes 13](#_Toc137782477)

[Positionnement des différentes parties 14](#_Toc137782478)

[Montage & premiers tests 15](#_Toc137782479)

[Alimentations 15](#_Toc137782480)

[Modification alimentation 3,3 V 15](#_Toc137782481)

[Mesure des alimentations 16](#_Toc137782482)

[Reste du montage 18](#_Toc137782483)

[Configuration des pins 19](#_Toc137782484)

[Modifications du schéma 19](#_Toc137782485)

[Schéma modifié 19](#_Toc137782486)

[Configuration harmony 20](#_Toc137782487)

[Configuration Sysclock 21](#_Toc137782488)

[Test Software + création librairie 22](#_Toc137782489)

[Test communiquer avec microcontrôleur 22](#_Toc137782490)

[Configuration harmony 22](#_Toc137782491)

[L’app.C 23](#_Toc137782492)

[Potentiomètre 24](#_Toc137782493)

[Test du potentiomètre 24](#_Toc137782494)

[Carte SD 25](#_Toc137782495)

[Fonctions programme micro chip 25](#_Toc137782496)

[Configuration SD et SPI 26](#_Toc137782497)

[Machine d’état du programme Micro Chip 27](#_Toc137782498)

[Test carte SD 28](#_Toc137782499)

[Mesure trame 28](#_Toc137782500)

[Test complet pour la machine d’état MicroChip. 29](#_Toc137782501)

[Pour le programme principal 29](#_Toc137782502)

[Information sur les fichiers .wav 30](#_Toc137782503)

[Entrées optocouplés + anti-rebond 31](#_Toc137782504)

[Montage pour faire les tests 31](#_Toc137782505)

[Anti-rebond 31](#_Toc137782506)

[Communication DAC 33](#_Toc137782507)

[Fonctionnement du DAC 33](#_Toc137782508)

[Modifications sur le AD5620 33](#_Toc137782509)

[Configuration du SPI 34](#_Toc137782510)

[Fonction d’envoi 34](#_Toc137782511)

[Gérer la fréquence d’envoi 35](#_Toc137782512)

[Test de l’ADC 35](#_Toc137782513)

[Test sorties audios 37](#_Toc137782514)

[Conclusion software 38](#_Toc137782515)

[Modification à apporter 38](#_Toc137782516)

[Evaluation du projet 39](#_Toc137782517)

[Etat d’avancement du projet 39](#_Toc137782518)

[Carte SD 39](#_Toc137782519)

[Entrées optocouplées 39](#_Toc137782520)

[DAC 39](#_Toc137782521)

[Potentiomètre 39](#_Toc137782522)

[Sortie audio 39](#_Toc137782523)

[Conclusion 40](#_Toc137782524)

[Annexes 41](#_Toc137782525)

[Cahier des charges du projet 41](#_Toc137782526)

[Dossier de CAO 41](#_Toc137782527)

[Liste de pièce 41](#_Toc137782528)

[Listing code 41](#_Toc137782529)

[Journal de travail 41](#_Toc137782530)

[Mode d’emploi du système 41](#_Toc137782531)

[Résumé du projet 41](#_Toc137782532)

[Affiche du projet 41](#_Toc137782533)

# Pré-étude

## Schéma block

Pont de greatz

(B)

Haut-parleur ou port jack line-out

(K)

Ampli

(J)

DAC

(I)

10 entrées (D)

Optocoupleur (E)

Régulateur 3,3 V

(C)

Alimentation 10-15 VAC (A)

µc

(H)

POT

(F)

Port USB ou carte micro-SD

(G)

## Détails des blocks

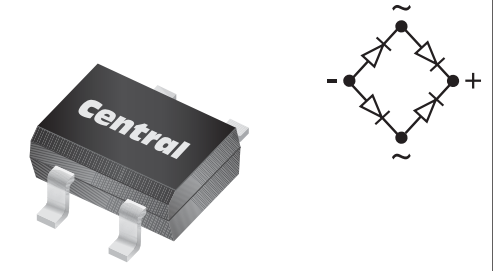
### (A) Alimentation 10-15 VAC

La carte a une alimentons une tension de 10-15 VAC.

Pour cela je dois installer des borniers a vis. Vu que nous n’avons pas une tension très élevée que nous essayons de faire une carte compacte, Des bornier de petite taille sont adaptés pour cela.

### (B) Pont de greatz

Nous avons en entrée une tension alternative depuis l’alimentation. Pour la redresser j’utiliserai un pont de greatz. Deux possibilités s’offrent à moi : prendre un IC avec un pont de greatz intégrer ou le faire directement avec des diodes. Ce signal purement positif va dans le redresseur



### (C) Redresseur

Pour alimenter le microcontrôleur, je dois avoir une tension de 3,3V : Pour avoir cette tension le plus adapté serait d’utiliser un régulateur linéaire. Cette tension permettra d’alimenter la plupart des composants.

Avec un redresseur 5 v pour avoir une sortie hautparleur de au moins 1W

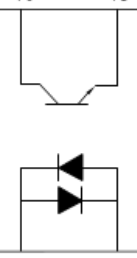
### (D) Entrées

Il y a 10 entrées qui peuvent être continue ou alternatif. Pour la connexion j’utilise aussi des borniers a vis et du fait que nous utilisons une tension semblable à celle d’alimentation, j’utiliserai les mêmes borniers a vis.

### (E) Optocoupleur

Vu que nous avons une tension qui est soit continue ou alternatif j’ai recherché un moyen de lires facilement les entrées.

Pour cela je pense utiliser un optocoupleur alternatif qui nous permet de lire les entrées en AC et en DC.



Mais vu que le temps que les entrées sont actives sont de lors des centaines de ms, je peux aussi utiliser un optocoupleur simple car ils y’aura aussi une phase positive à un certain moment.

Le seul problème que je pourrai rencontrer est que la LED sera en tension négative donc il nous en faut qui puissent le supporter sur assez de tension. Les signaux de sortie seront lus par le µc afin d’activer un des sons par apport à l’entrée active.

### (F) Potentiomètre

On utilise on potentiomètre à l’avant de la carte, le mieux serai de trouver un potentiomètre qui s’utilise à l’horizontale pour mieux rentrer dans le boitier. Pour mesurer ce potentiomètre, j’utiliserai directement de l’ADC du microcontrôleur.

### (G) Port USB ou carte micro-SD

Pour ce projet, une des questions qui s’est posée lors du cahier des charges nous nous demandions si nous utilisons une carte SD ou une clé USB afin de stocker et lire les sons.

Les deux périphériques semblent avoir beaucoup de similitude, le seul point qui fait pencher la balance vers une carte SD est que nous avons une tension de 3,3 V et que l’USB fonctionne sur du 5 V. Ce qui me ferait refaire une alimentation pour avoir 5 V uniquement pour l’USB.

Mais je garderai l’USB de côté dans le cas où j’aurai assez de temps afin de faire une version avec USB.

### (H) µc

Vu qu’en cours nous avons des PIC32, nous allons utiliser le même type de microcontrôleur afin d’éviter de perdre du temps à comprendre le fonctionnement d’un nouveau. Ainsi que réduire le risque que nous devrions attendre sur la commande d’un nouveau µc

### (I) DAC

Pour avoir une sortie audio nous devons utiliser un convertisseur Digital-Analogique, pour le convertisseur, un convertisseur 8bits peut largement suffire à avoir une sortie audible. Le seul point à faire attention est la vitesse de travail pour avoir de bonne fréquence de fonctionnement sur l’hautparleur de sortie.

### (J) Ampli

Après avoir générer le signal avec le DAC, dois avoir une sortie un minimum plus puissant que du 3,3V afin d’avoir un volume correct. Pour cela je dois amplifier le signal à la sortie à l’aide d’un AOP. Pour ensuite aller directement sur le haut-parleur

Pour avoir une sortie à 5 W je dois avoir une alimentation de 5V.

Pour la sortie sur le jack, je dois avoir un autre ampli OP afin de

### (K) Haut-parleur

Pour la sortie nous avons un haut-parleur, nous devons avoir au moins 1W et 80 dB.

Pour cet hautparleur je vais choisir un hautparleur assez gros pour donner un minimum de puissance mais pas trop gros pour pouvoir faire un boîtier plus compact.

### Sortie Jack

Pour avoir une sortie polyvalente, nous avons une sortie Jack 3,5 mm.

### Boîtier

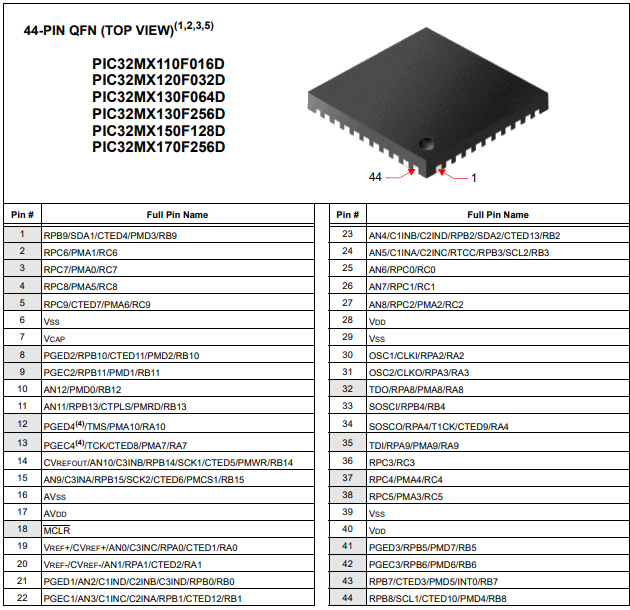
Pour le boitier je pensais prendre un boitier déjà existant pour ne pas avoir besoin de prendre du temps à faire un boiter 3D. Mais après réflexion, je pense que je n’aurai pas de contrainte majeure sur ce boitier donc je peux prendre directement un boitier existant et faire des modifications rapides sur celui-ci.

## Choix composant critique

Regarder pour un ucotroleur qui possède directement un DAC intégré pour éviter de devoir en utiliser un externe.

Dans ce projet, j’ai seulement comme composant critique le µc car c’est un composant qui peut vite devenir chez ainsi que difficile à trouver. Nous avons reçu une liste de µc à utiliser.

J’ai choisi de prendre un PIC32MX230F256D car je n’ai pas besoin que le µc ait une fréquence de 80M, 40M me suffit largement. Celui-ci est aussi plus petit et est largement suffisant. Il y aurait eu possibilité de prendre un µc encore plus petit mais dans le cas où nous voulons faire une version avec USB, j’ai choisi ce composant avec possibilité de communication USB incluse dans le µc.

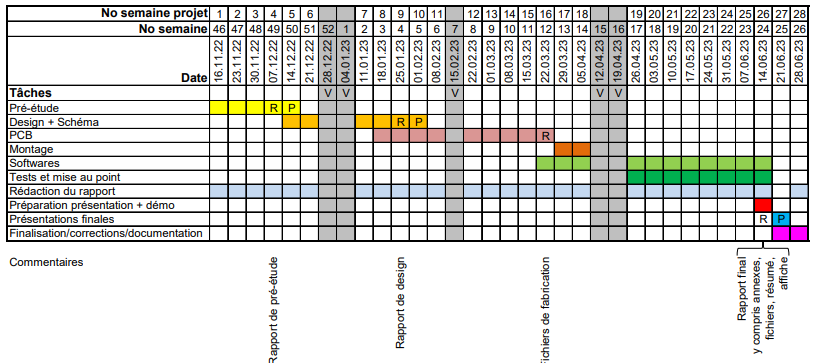


## Evaluation des couts

Pour estimer les couts de la carte, j’ai regardé sur différents fournisseur différent produit pouvant être utiliser et ait arrondi au-dessus.

|  |  |  |
| --- | --- | --- |
| Composant | quantité | prix approximatif [CHF] |
| Bornier a vis | 2 | 4 |
| port micro-SD | 1 | 2,5 |
| pont greatz | 1 | 0,7 |
| LDO | 1 | 0,9 |
| Optocoupleur | 1 | 4 |
| Potentiomètre | 1 | 2,5 |
| DAC | 1 | 4 |
| Ampli | 1 | 3 |
| Haut-Parleur | 1 | 10 |
| Microcontrôleur | 1 | 5 |
| Production carte | 1 | 30 |
| Résistances | ~30 | 2 |
| Condensateurs | ~30 | 2 |
| total | | 70,6 |

## Planification



Pour la planification je compte suivre le plan qui nous a été fourni. Mais quand le design sera terminé, il faudra faire le boitier 3D en parallèle du code qui ne sera pas très imposant.

## Conclusion et perspective

Pour ce projet je dois lire 10 entrées optocoupleur et interagir avec une micro-SD afin de pouvoir lire les fichiers audios et les lire sur un haut-parleur. Pour aller sur le haut-parleur je dois faire un convertisseur digital-analogique puis un amplificateur. Pour régler le volume, je dois pouvoir utiliser un potentiomètre. Cette carte a une alimentation alternative s’est pour cela que je dois la redresser et l’ajuster à 3,3V. Dans cette pré-étude j’ai pu faire plusieurs choix qui me seront utiles pour la suite.

Je dois encre dimensionner les composant ainsi que de déterminer si je vais avoir le temps de faire une variante avec l’USB.

# Rapport de schématique

## Schéma générale

Le schéma complet se situe en annexe.

## Block schéma

### Connecteurs entrée-sortie

Pour les connecteurs entrée-sortie, j’utiliserai un connecteur Würt electronique.

S/N : 691312510002



Ce connecteur me permettra de pouvoir venir connecter et déconnecter les entrées sorties plus facilement. Mais aussi pour le boitier car il n’y aura pas besoin d’ouverture sur celui-ci.

### Alimentations et pont de Greatz

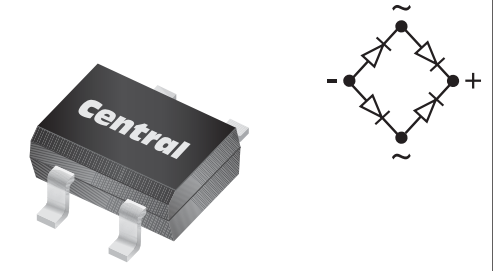


Dans cette partie nous avons le connecter (point 2.2.1) ainsi qu’un pont de greatz.

Composant : CBRHDSH1-40L

J’ai choisi celui-ci car supporte une tension de 40V et à une grande quantité en stock.

Ce montage me permet d’avoir une tension sinusoïdale positive à la sortie qui directement sur le régulateur 5V.



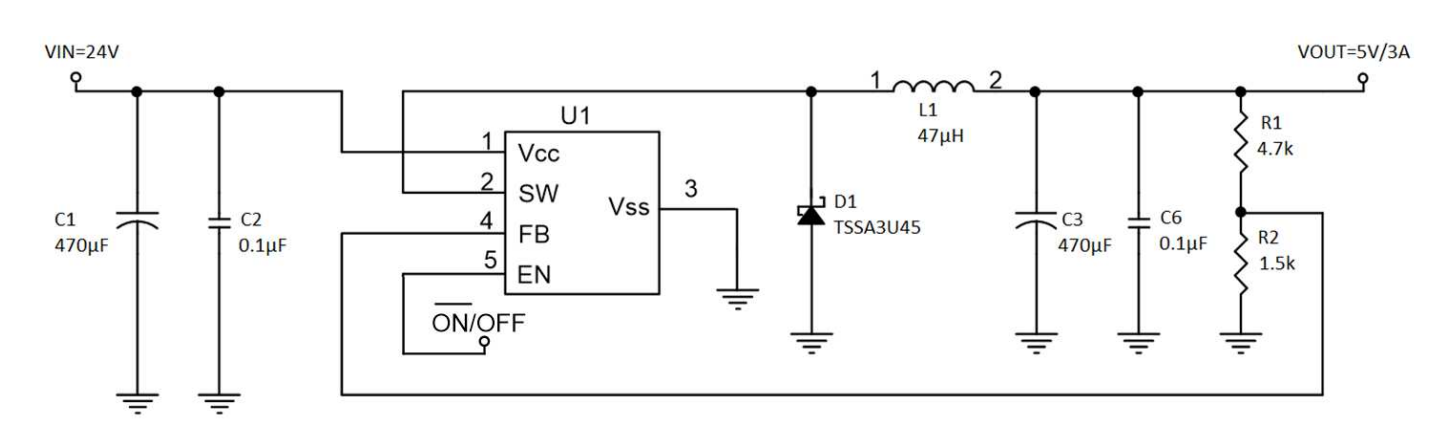
### Régulateur 5V



Pour le régulateur 5V j’ai choisi un TS2596CM550 RNG.

J’ai choisi ce régulateur car il y a beaucoup de stock, il me permet d’avoir une tension de 5V fixe à la sortie facilement.

Pour faire le schéma, j’ai recherché le schéma d’application typique dans la datasheet :



J’ai donc adapté ce schéma sur Altium.

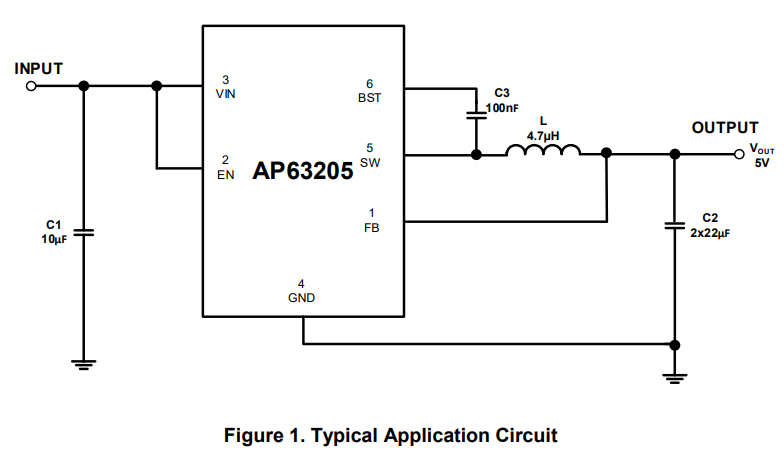
Ce régulateur me permet d’avoir une sortie DC fixe de 5V, il alimente l’amplificateur audio ainsi que le DAC et va aussi sur un régulateur 3,3V afin d’alimenter le reste du schéma.

### Régulateur 3,3V



Pour le régulateur 3,3V j’ai choisi un AP63203WU.

Pour faire le schéma, j’ai recherché le schéma d’application typique dans la datasheet :



J’ai donc adapté ce schéma sur Altium.

Ce régulateur me permet de sortir une tension fixe de 3,3 V à la sortie ce qui permet d’alimenter tout le reste de mon schéma car le µContrôleur fonctionne sur du 3,3V.

### Microcontrôleur



Le microcontrôleur à été choisi lors du point 1.3

J’utilise une entrée analogique (AN0) afin de lire la valeur du potentiomètre.

Deux port SPI :

Pour les ports SPI seul leurs Clock est un pin assigné sinon les autres pins sont mapable.

* Pin 14 pour le SPI1
* Pin 15 pour le SPI2

Les entrées pour le debbuger sont sur leurs pins assignées :

* Pin 13 pour PGEC
* Pin 12 pour PGED

### Port de programmation



Afin de programmer le microcontrôleur, nous avons le connecteur ainsi que les connexions que nous avoir dans un dossier fourni par l’ES. Ceci nous permet de le faire directement dans le schéma. Ce port de programmation va sur les pins de microcontrôleur prévues à cet effet.

### Potentiomètre



Pour le potentiomètre, je me suis inspiré de ce qui a été fait pour le qui PIC32MX que nous utilisons pour ainsi faire le schéma de celui-ci. Celui-ci est directement relié à une entrée analogique du microcontrôleur.

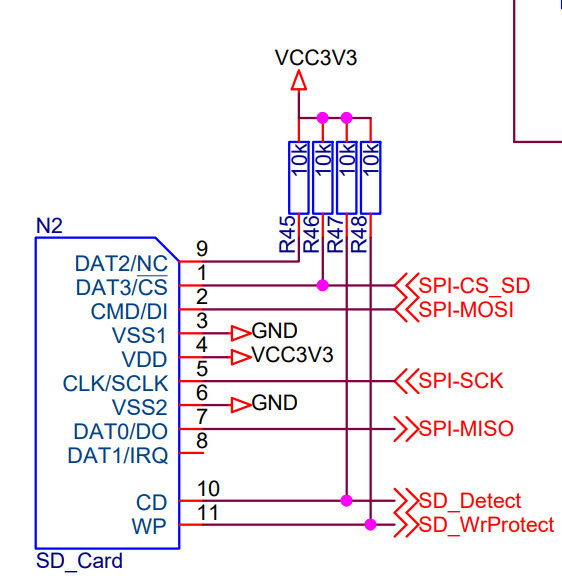
### Carte SD



Pour l’entrée de la carte SD, j’ai choisi un connecteur GMC020080HR.

Pour la communication de la carte SD, cela se fait en liaison SPI.

Pour faire le schéma, j’ai utilisé le schéma du PIC32 que nous avons en laboratoire.



### Entrées avec les optocoupleurs

Pour entrée optocoupleur dimmensionnement :



J’ai utilisé des IS2805-4 comme optocoupleurs afin de gérer les entées pour garantir une isolation galvanic sur le microcotrôleur. J’ai décidé de prendre des ICs avec 4 entrées optocouplée car nous avons 10 entrés et cela prendra moins de place.

DAC

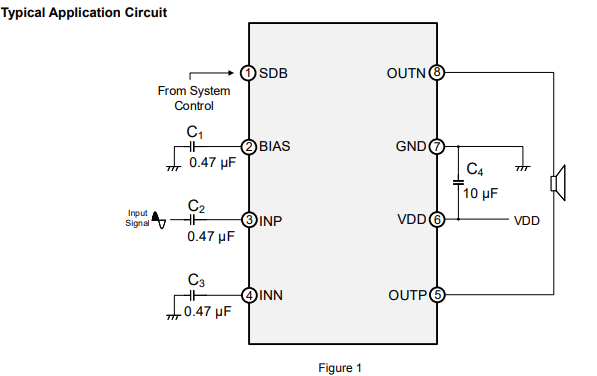


Pour le DAC, j’ai utilisé un AD5620CRMZ-1. Ce convertisseur fonctionne avec un port de communication SPI et travaille sur 12Bits. Ce qui nous permet de faire les signaux que nous aurions lu sur la carte SD. Celui-ci se communique en SPI ce qui prend beaucoup moins de place par apport à un DAC avec des entrées en parallèle.

### Ampli audio



J’ai choisi un BD7831EFJ-ME2 comme amplificateur audio. Celui-ci est alimenté en 5V et permet de sortir une puissance de 1,2 W ce qui correspond au cahier des charges. A l’entrée de celui-ci, nous avons la sortie du convertisseur analogique-digitale. Puis sur la sortie, nous avons la prise Jack reliée au haut-parleur.



Le schéma d’application typique de la datasheet m’as guidé dans la schématique.

### Sortie Jack :



J’ai choisi un adaptateur JACK 3,5 mm SJ1-3533NG. Celui-ci est en grande quantité en stock.

Le GND de la prise Jack se trouve le plus loin du bout s’est pour cela que le bout et le signal d’a pin3 sont les signaux de la sortie de l’amplificateur audio.

## Conclusion rapport schéma

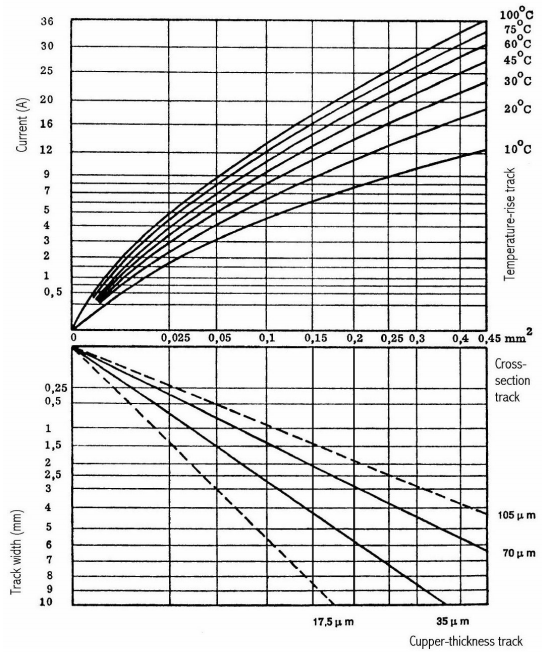
Pour faire le schéma, j’ai beaucoup pu être aidé par le schéma de la plaque que nous avons en laboratoire car elle utilise un pic32 avec certains périphériques que je n’avais jamais utilisés.

Pour la suite il me faudra encore faire certains footprints ainsi que de trouver un boîtier avant de commencer le design afin de directement le faire à la bonne dimension.

# Routage

## Largeur des pistes

Dans mon montage, je n’aurai pas plus de 500mA



Pour faire le routage des pistes, nous n’avons pas un grand courant j’ai donc pu router la carte avec des piste à un minimum de 0,2 mm.

## Positionnement des différentes parties

Sortie haut-parleur

Sortie Jack

DAC

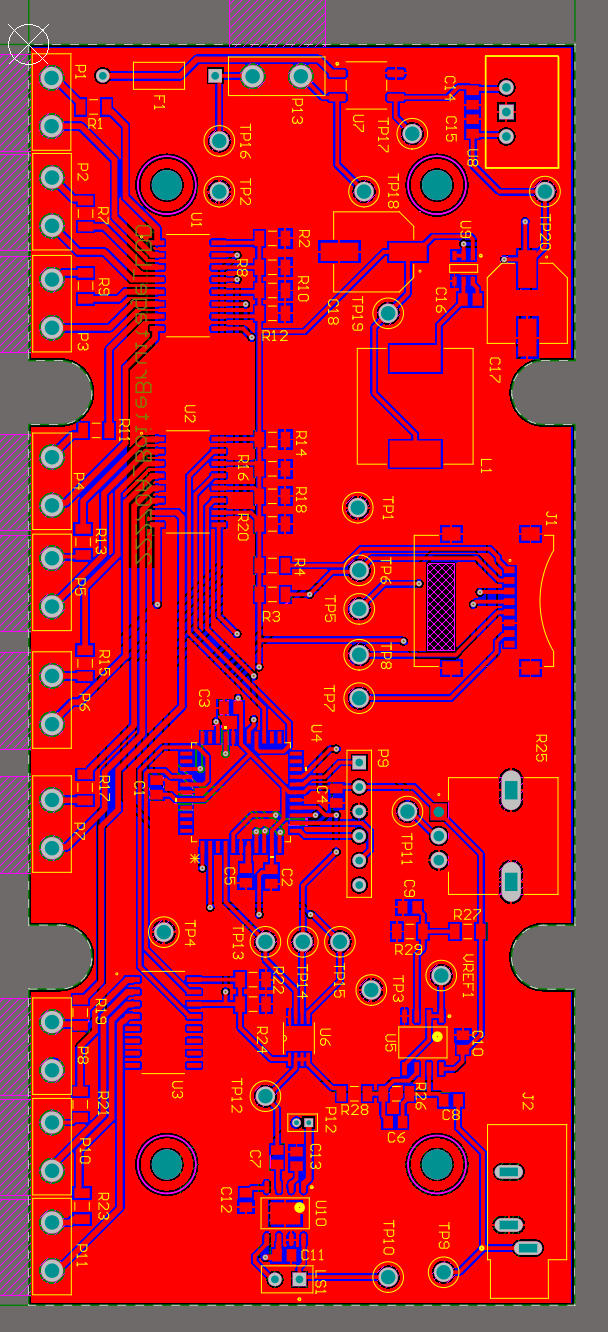
Microcontrôleur

Potentiomètre

Carte microSD

Entrées optocouplées

Alimentation 3,3 V & 5V



# Montage & premiers tests

## Alimentations

Pour commencer, j’ai routé uniquement les alimentations et les ai testées afin de vérifier que notre montage et notre microcontrôleur pour éviter d’avoir de gros problèmes pour la suite.

### Modification alimentation 3,3 V

Notre alimentation 3,3 V avais un problème et me donnais un signal trop faible avec une trop grande variation. Cela vient du fait que pour générer le 3,3 V, j’au utilisé un régulateur step-down et je l’ai mis directement après le régulateur 5V. La différence de tension en entre le 5V et le 3,3V est de 1,7V.

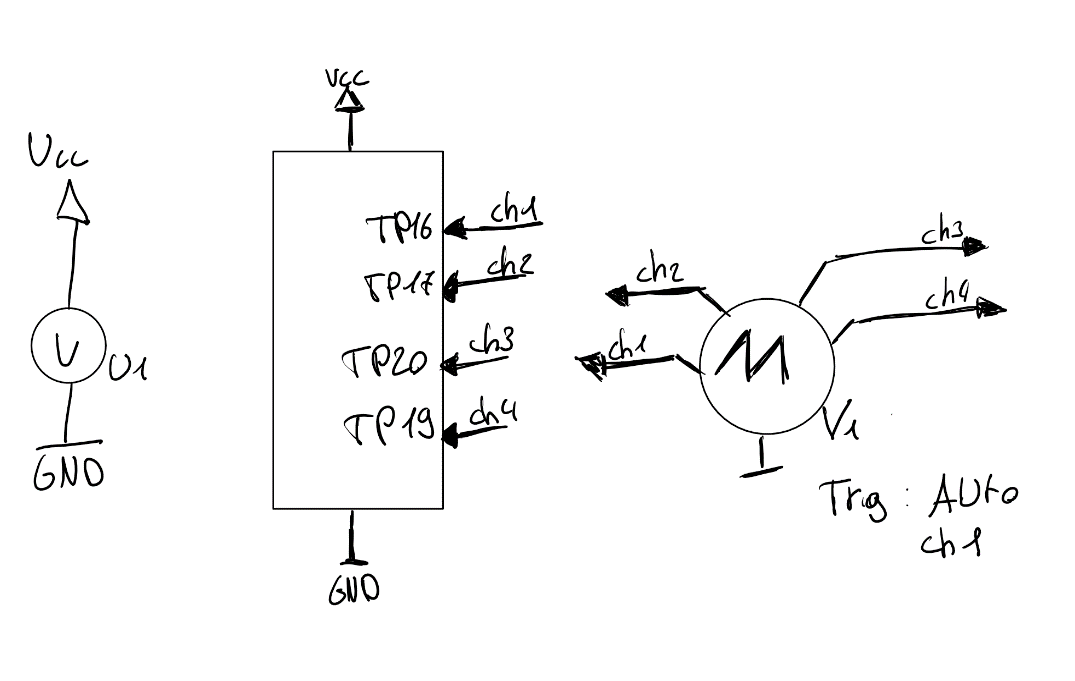
La différence de tension est faible et pour palier au problème, j’ai directement relié l’entrée du régulateur à la sortie du pont de greatz :

Une image contenant texte, diagramme, Plan, ligne

Description générée automatiquement

### Mesure des alimentations

#### Schéma de mesure



#### Liste instrument de mesure

V1 = oscilloscope Rohde&Schwarz RTB2004 S/N : ES.SLO2.05.01.09

U1 = alimentation DC 0-30V Gwenstek GPA-3303 S/N : ES.SLO2.00.00.28

#### Méthode de mesure

Dans cette mesure le but est de prouver que nos alimentations fonctionnent correctement.

Pour cela, j’ai placé une résistance de charge afin de simuler le circuit.

Pour avoir une résistance de charge, j’ai utilisé ce type de potentiomètre :



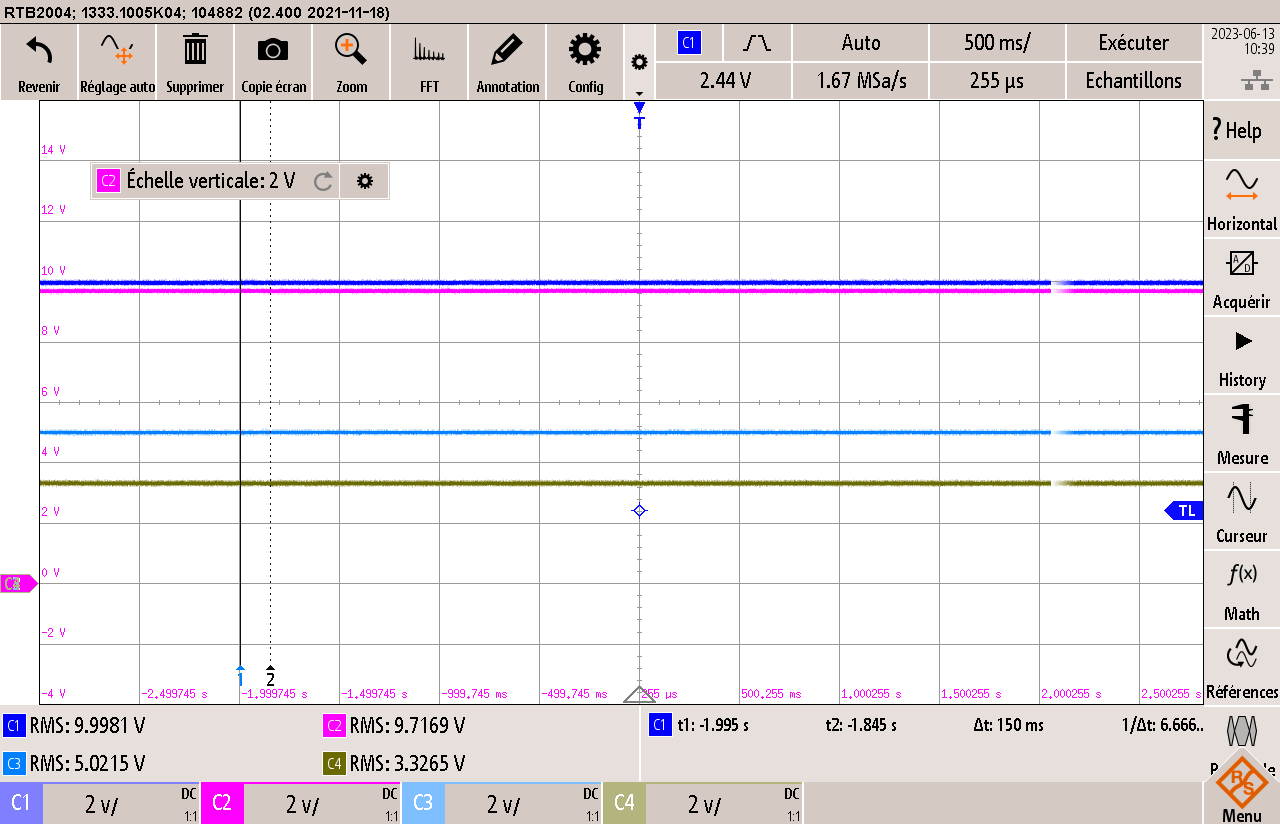
Ce qui me permet de régler le courant de sortie à ~500mA.

Pour avoir une tension moyenne, j’ai utilisé la fonction mesure de l’oscillogramme.

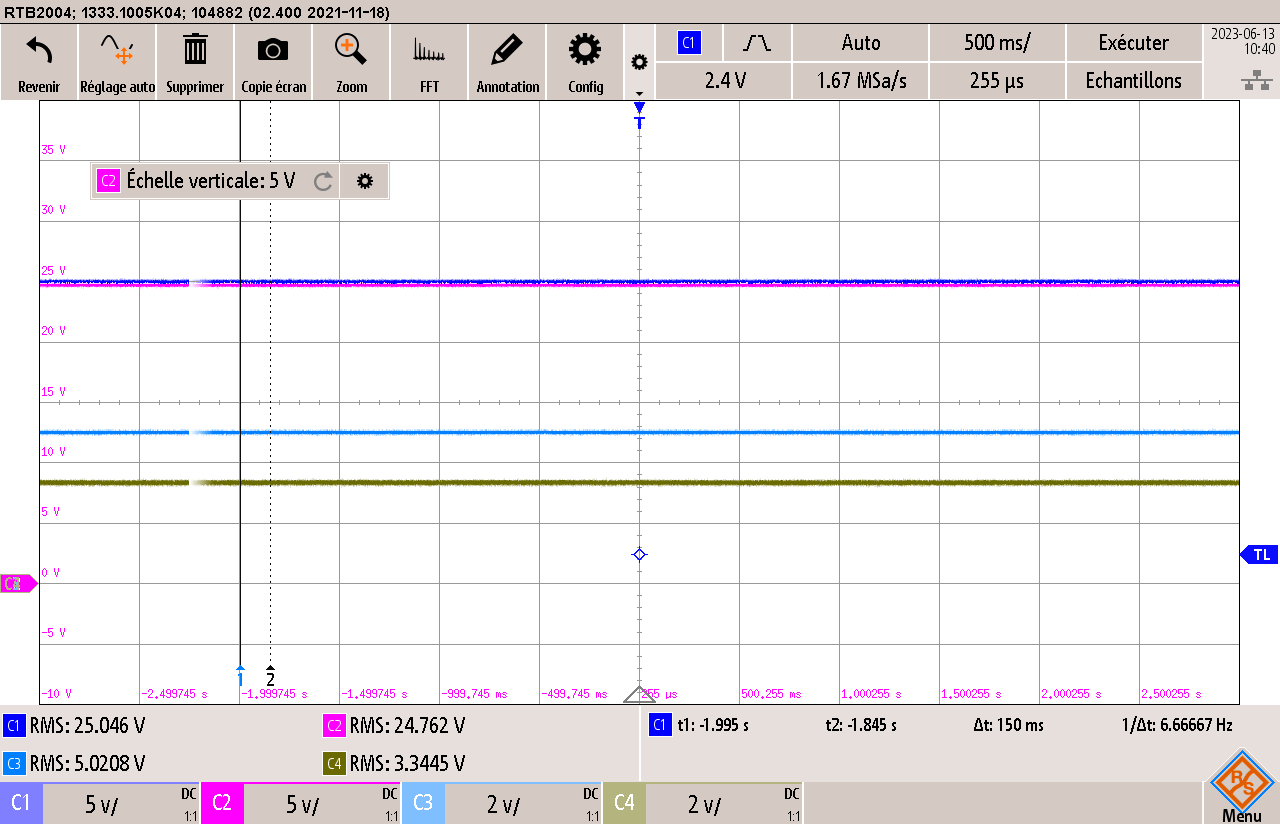
Je dois avoir 5 V sur le chenal 3 et 3,3V sur le chenal 4.

#### Oscillogrammes

##### Alimentation 10V DC



##### Alimentation 25V DC



##### Alimentation 10V AC

Une image contenant texte, capture d’écran, nombre, Tracé

Description générée automatiquement

#### Tableau mesure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Uin | U Ch3  Théorique [V] | U Ch4 théorique [V] | U Ch3  Mesuré [V] | U Ch4  Mesuré [V] |
| 10V DC | 5 | 3,3 | 5,02 | 3,32 |
| 25V DC | 5 | 3,3 | 5,02 | 3,34 |
| 10V AC | 5 | 3,3 | 4,69 | 3,32 |

J’ai pu constater que nos mesures montrent que l’alimentation est stable et me permet de continuer en brasant tous les composants restant.

## Reste du montage

J’ai finalement braser tout le reste du montage. Quand celui-ci a été alimenté, aucun court-circuit ce qui est grandement encourageant. Pour la suite des tests, il faut programmer la carte afin de tester les différents modules.

## Configuration des pins

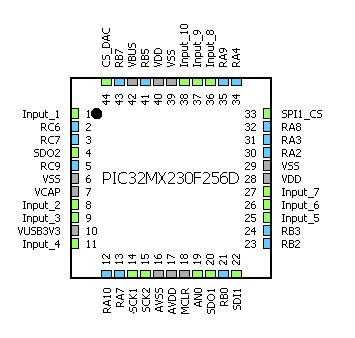
### Modifications du schéma

Lors de la programmation des pins, je me suis rendu compte que je n’avais pas utilisé les bonnes entrés sur le microcontrôleur. Pour le modifier, j’ai utilisé harmony afin de voir les pins utilisables pour les SPI.

### Schéma modifié

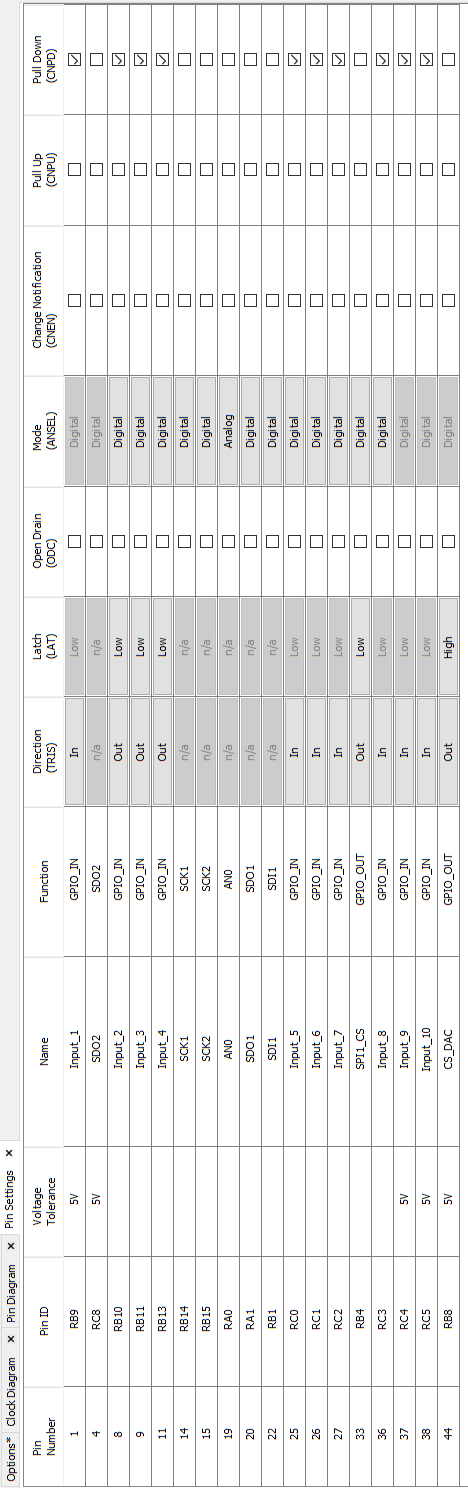


Cela nous donne ceci sur harmonie :



### Configuration harmony

Pour les entrées j’ai initialiser les deux SPI (SCK, CS, SDO, SDI), sur toutes les entrées, j’ai utilisé la fonction Pull Down pour avoir un état beaucoup plus stable, je les aie initialisées en GPIO\_IN. L’entrée du potentiomètre est sur l’entrée AN0.

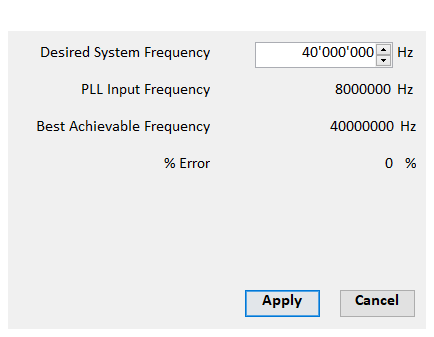


## Configuration Sysclock

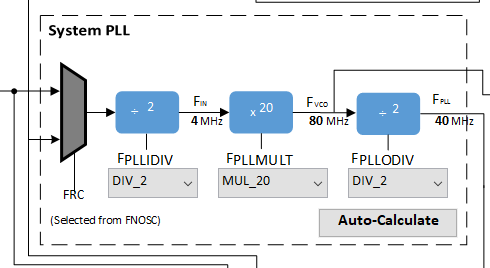
Pour configurer le sysclock, j’ai utilisé le menue de configuration des clock.

Le microcontrôleur peut aller jusqu’à 48MHz, Pour que cela soit plus facilement divisible et que le clock soit plus fluide, j’ai préféré le configurer à 40 MHz.

J’ai utilisé la fonction de calcul automatique :



Pour ainsi avoir 40 MHz en sortie :

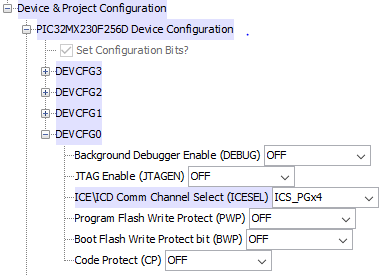


# Test Software + création librairie

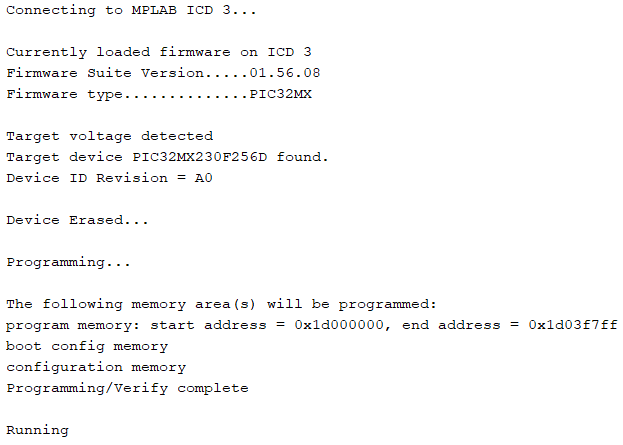
## Test communiquer avec microcontrôleur

### Configuration harmony

Pour communiquer avec le microcontrôleur, il faut configurer harmony pour avoir les bons pins pour le PGEC et PGED correcte.



Puis générer le code pour ainsi programmer un code vierge sur le microcontrôleur.



J’ai ainsi pu constater que celui-ci répondait correctement.

## L’app.C

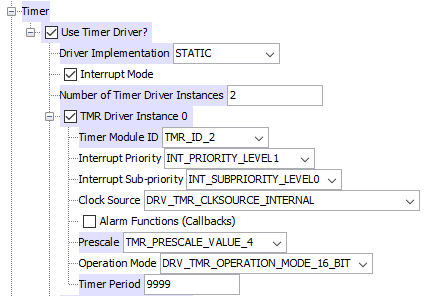
L’app.C est une machine d’état avec 3 états :

* Init
* Service task
* Wait

L’init s’éffectue uniquement une fois au lancement du programme et initialise les timers et le SPI du DAC. Puis active ceux-ci.

Le Service Task d’effectue toutes les millisecondes à l’aide de l’interruption du Timer 2 :

Calcul :



Dans cette fonction, il y aura la lecture des entrées ainsi que la gestion du potentiomètre.

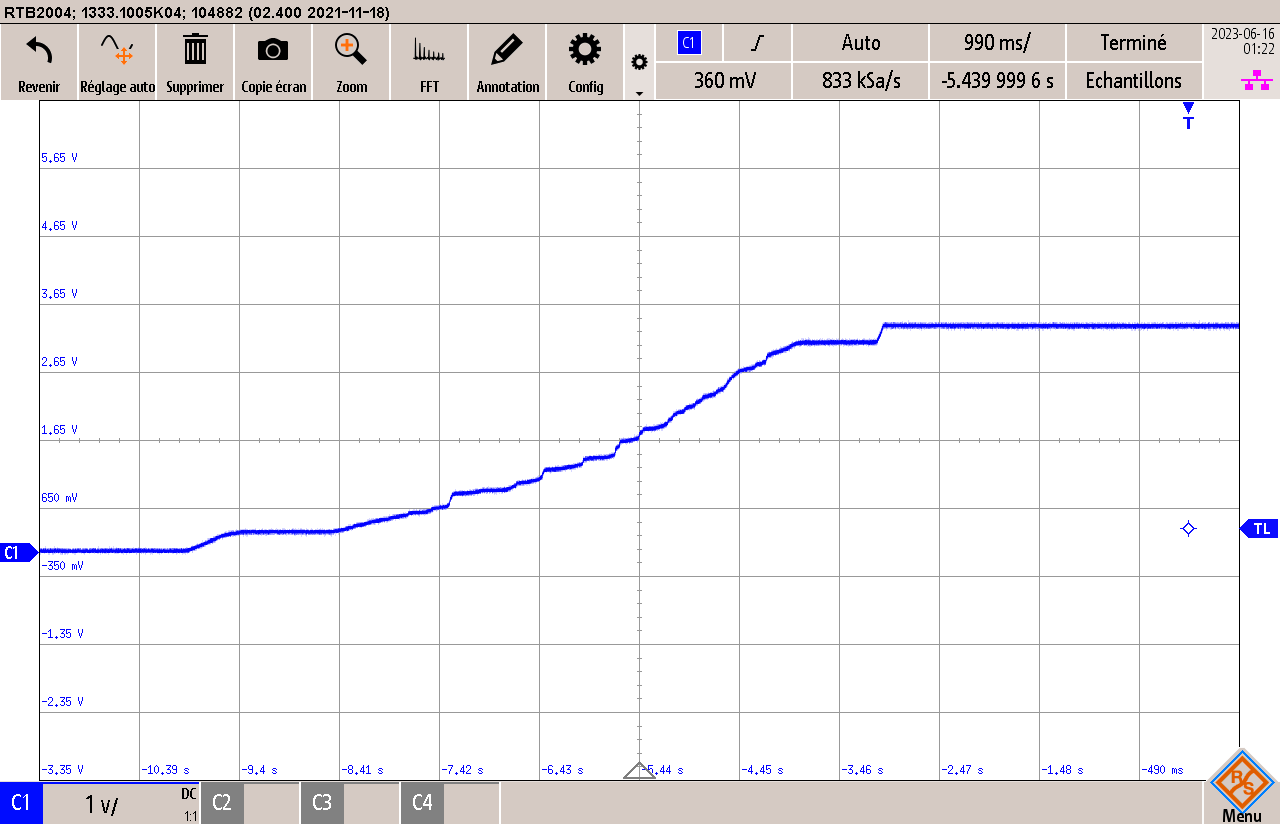
## Potentiomètre

### Test du potentiomètre

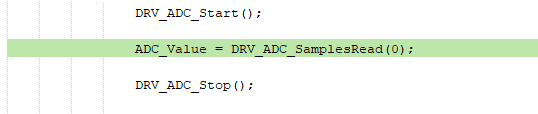
Premièrement pour tester le potentiomètre, j’ai directement mesuré le signal de potentiomètre avec l’oscilloscope.

CH1🡪 TP11

Dans cette mesure je tourne le potentiomètre du minimum au maximum, je devrais voir un signal qui part de 0V et qui monte jusqu’à 3,3V :



Pour vérifier que je lire le potentiomètre, j’ai rajouté une ligne pour lire l’ADC



Et j’ai directement été regardé la valeur sur l’ADC quand je le met au max :



Et au min :



## Carte SD

Pour utiliser la carte SD, j’ai utilisé un programme de test fait par Micro chip.

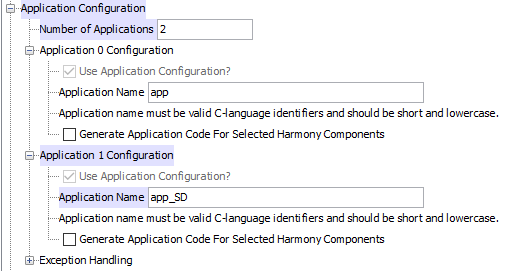
### Fonctions programme micro chip

Le programme micro-chip fait les actions suivantes :

* Monte la carte
* Démonte la carte
* Remonte la carte
* Set le fichier courant
* Lit le fichier nommé « FILE\_TOO\_LONG\_NAME\_EXAMPLE\_123.JPG »
* Crée un nouveau répertoire nommé « Dirl »
* Crée un nouveau ficher avec le même nom que celui ouvert précédemment
* Copie le fichier « FILE\_TOO\_LONG\_NAME\_EXAMPLE\_123.JPG » dans ce nouveau fichier
* Ferme les deux fichiers
* Attend indéfiniment

Ce programme a été conçu pour le stater kit avec le PIC32MX795F512.

Pour cela, j’ai dû créer une seconde application dans ce projet :



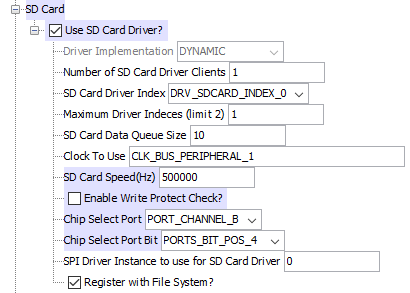
Puis j’ai copié collé la machine d’état d’exemple dans celle-ci.

Pour finalement renommer toutes les structure, énumérations et fonction afin que cela soit compatible avec l’APP. Pour finir j’ai include app\_SD.h dans app.h.

### Configuration SD et SPI

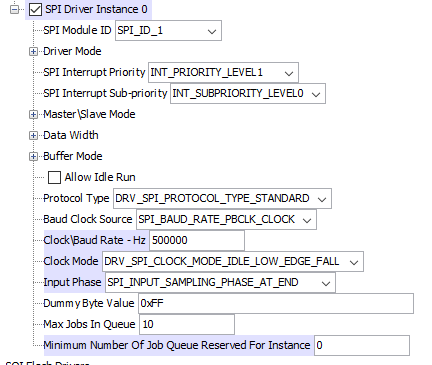
Configuration carte SD :

Choix du pin du Chip select

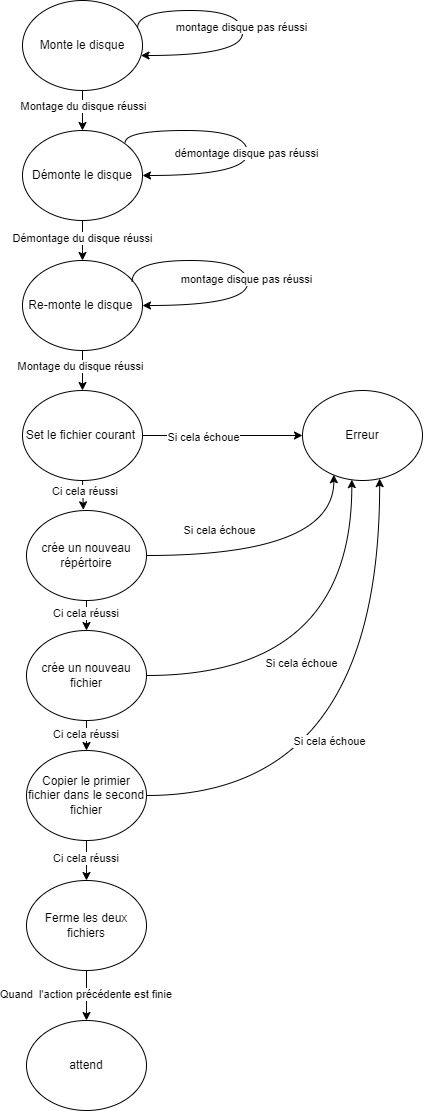


Configuration SPI

Choix polarité du clock

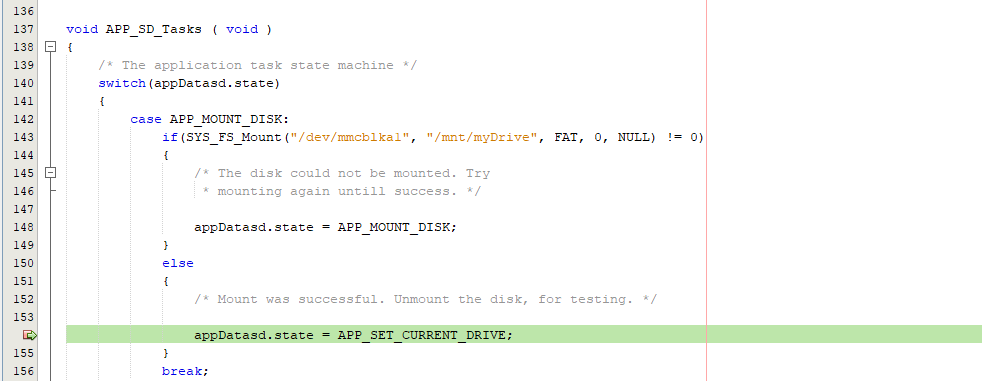


### Machine d’état du programme Micro Chip



### Test carte SD

Pour commencer, j’ai testé si la carte de monte correctement :



### Mesure trame

Pour vérifier que le SPI est correct, j’ai mesuré une trame d’envoi.

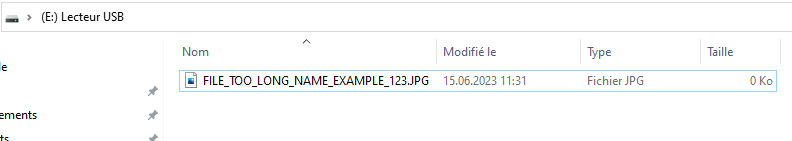
|  |  |
| --- | --- |
| Ch1 | TP6 |
| CH2 | TP5 |
| CH3 | TP8 |

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

### Test complet pour la machine d’état MicroChip.

Pour cela j’ai créé le fichier dont j’ai besoin pour le programme :



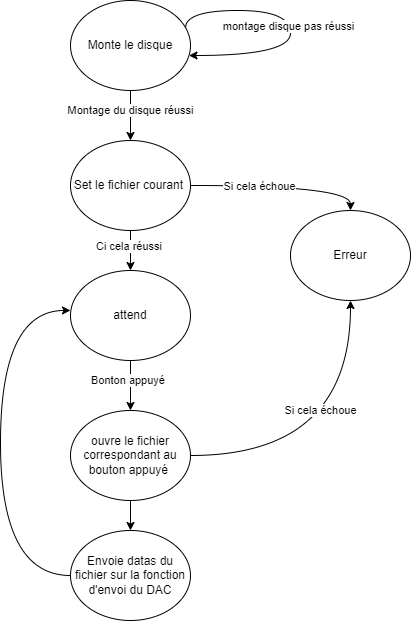
Puis j’ai lancé le programme et j’ai obtenu un nouveau dossier :



Nous pouvons voir que le programme de teste ne recrée pas de nouveau fichiers. Mais cela n’as pas grande importance car le but était de prouver que la communication de la carte SD est fonctionnelle.

### Pour le programme principal

Pour le programme principal, il faut modifier cette machine d’état.



### Information sur les fichiers .wav

Pour lire les fichiers .wav correctement il faut nommer les fichiers d’un manière précise.

Cela il faut les nommer ainsi « XX.wav ».

Les fichiers .wav sont composés de 44 octets d’entête. Ces 44 octets donnent les informations suivantes :

* Octets 0-3 : "RIFF" (4 octets) - Indique le début du fichier RIFF (Resource Interchange File Format).
* Octets 4-7 : Taille du fichier (4 octets) - Indique la taille totale du fichier WAV en octets (hors les 8 octets des premiers octets de l'en-tête).
* Octets 8-11 : "WAVE" (4 octets) - Indique le type de format du fichier (format WAVE).
* Octets 12-15 : "fmt " (4 octets) - Indique le début de la section de format.
* Octets 16-19 : Taille de la section de format (4 octets) - Indique la taille de la section de format (16 pour PCM).
* Octets 20-21 : Format audio (2 octets) - Indique le type de format audio utilisé (1 pour PCM non compressé).
* Octets 22-23 : Nombre de canaux audio (2 octets) - Indique le nombre de canaux audio (1 pour mono, 2 pour stéréo).
* Octets 24-27 : Fréquence d'échantillonnage (4 octets) - Indique la fréquence d'échantillonnage en hertz (ex. 44100 Hz).
* Octets 28-31 : Débit binaire (4 octets) - Indique le débit binaire en octets par seconde.
* Octets 32-33 : Taille d'échantillon (2 octets) - Indique la taille de l'échantillon en bits (ex. 16 bits).
* Octets 34-35 : Bits par canal (2 octets) - Indique le nombre de bits utilisés par canal (ex. 16 bits).
* Octets 36-39 : "data" (4 octets) - Indique le début de la section de données audio.
* Octets 40-43 : Taille de la section de données audio (4 octets) - Indique la taille de la section de données audio en octets.

Ces octets permettent de savoir toutes les informations sur cette piste audio.

Puis la suite des données sont les données audio brutes. Qui faudra calculer pour l’ampli OP qui est en 12 Bits.

## Entrées optocouplés + anti-rebond

### Montage pour faire les tests

Pour utiliser les entrées, j’ai utilisé ma carte de test pour faire 10 entrées avec un switch :

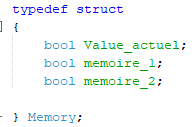


J’ai ensuite mis 10 boutons pour aller sur toutes les entrées.

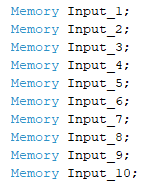
### Anti-rebond

Pour faire l’anti-rebond, je le fais dans l’app. Comme cela je peux lire les entrées toutes les 1ms et je fais deux mémoire afin d’avoir un meilleure anti-rebond. Pour cela j’ai créé une fonction qui lis l’entrée actuelle et met en mémoire les deux cycles précédents, Cette fonction se nomme : **Debunce\_Input ()**.

Pour simplifier le reste, d’utilise un typedef pour avoir dans une variable l’état actuelle et les deux anciens :



Et ensuite je crée toutes les variables qui correspondent toutes à une entrés



Et ensuite, j’utilise une fonction **Mémoire\_Save (Memory \*Input1)** cette variable va pointer sur la structure mise en paramètre.



Il faut appeler cette fonction pour toutes chaque entrées (10 fois)

Pour finalement enregistrer l’état actuelle avec la fonction **Input\_1StateGet()** qui renvoie l’état actuel de l’entrée.

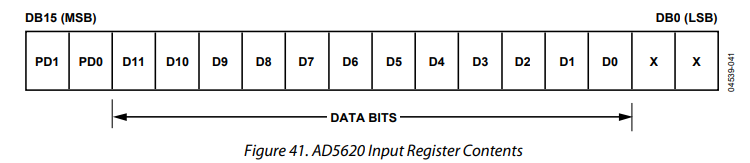
La deuxième fonction permet de vérifier qu’il y ait un flanc montant. Cette fonction est la suivante : **bool Check\_FlanMontant(uint8\_t InputNumber)** en paramètre d’entrée, il y a un nombre de 1-10 qui correspond à l’entrée choisie et renvoie un bool. Pour vérifier s’il y a biens un flanc montant, je vérifie que l’état actuelle est actif et que les deux états de mémoire sont inactifs.

Ces actions s’effectuent dans l’app.c comme cela il y a une lecture de flanc toutes les ms.

## Communication DAC

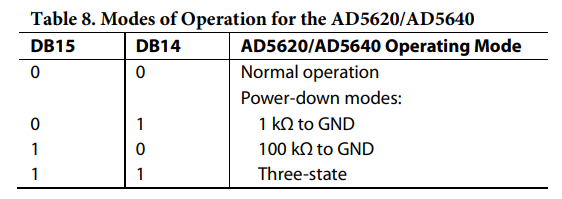
### Fonctionnement du DAC

Le DAC communique en SPI sur 12 Bits, les trames envoyées sont les suivantes :



D0-D11 sont les données de 0 à 4095.

PD0 & PD1 sont des bits de configuration :



Pour notre projet, nous le mettons en normal opération donc ces deux Bits sont à 0.

### Modifications sur le AD5620

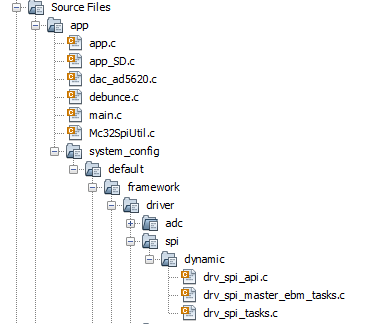
Lors de la partie schématique, je n’avais pas remarqué que l’entrée VFB est une tension de référence pour l’offset. Du coup j’ai rajouté un pont diviseur sur cette entrée :



### Configuration du SPI

Pour la configuration du DAC, vu que j’utilise une Spi en Static et que je dois le configurer en Dynamic.

Pour cela j’ai dû rajouter les configurations dynamique fait dans un autre projet MPLABX :



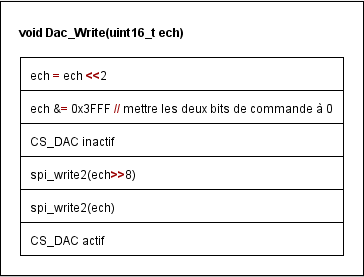
Pour ensuite le configurer avec une fonction. Ou j’ai copié collé les configurations sur un projet test dans la fonction **Dac\_Init ()**. La seule commande qui doit être retouchée est la suivante :



Qui permet de modifier la fréquence du clock du SPI.

### Fonction d’envoi

La fonction d’avoir mets les bits dans bonne position et met les bits de commande à 0.



### Gérer la fréquence d’envoi

Pour gérer la fréquence d’envoi, j’utilise l’interruption du timer 4 pour avoir une fréquence de 44,1 KHz. Car cela est une fréquence d’échantillonnage la plus fréquente dans les fichiers .wav.

### Test de l’ADC

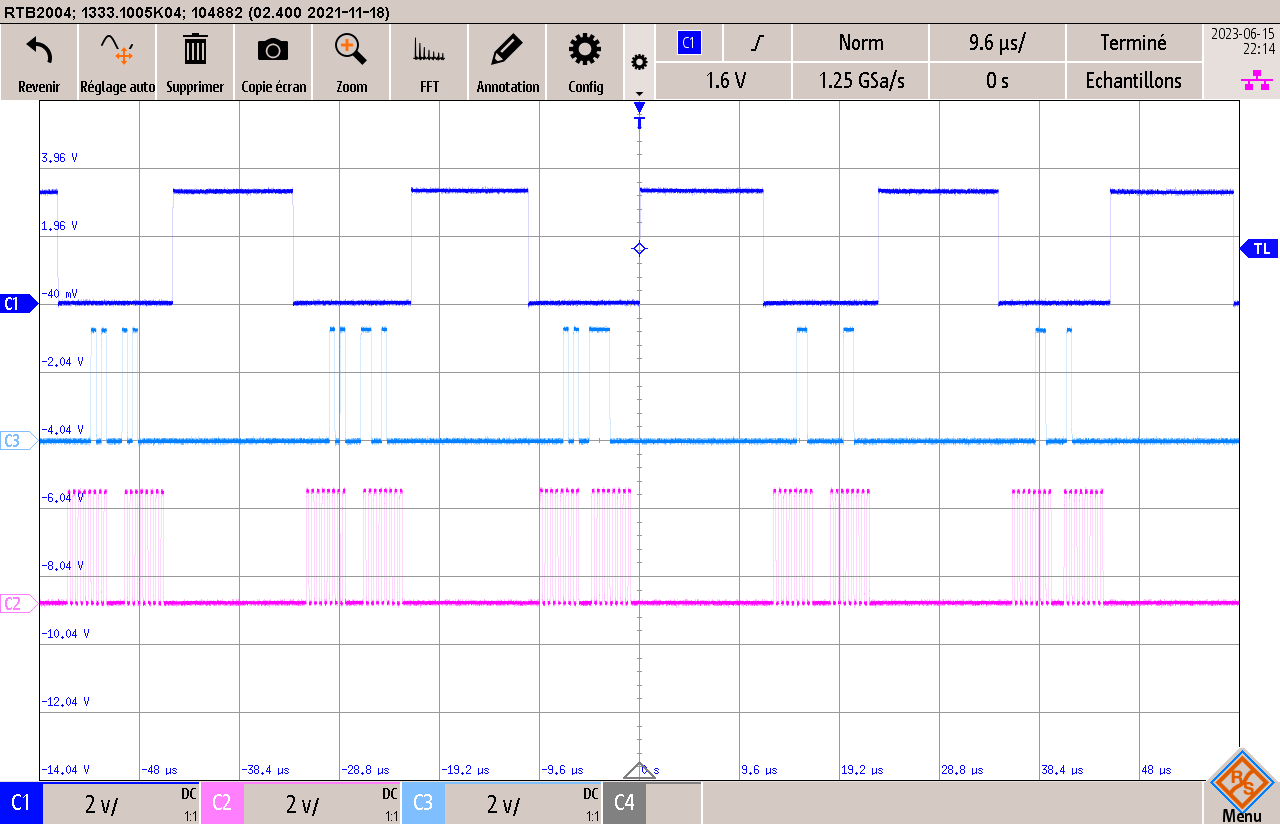
Pour tester l’ADC, j’ai fait une fonction d’envoi qui s’incrément à chaque fois que cela passe dans l’interruption du Timmer 4 et se remet à 0 quand celui-ci arrive à 4090. Ce qui devrai nous donner un signal triangle de 0-3,3 V.

#### Mesures

|  |  |
| --- | --- |
| Ch1 | TP15 |
| CH2 | TP14 |
| CH3 | TP13 |
| CH4 | TP12 |

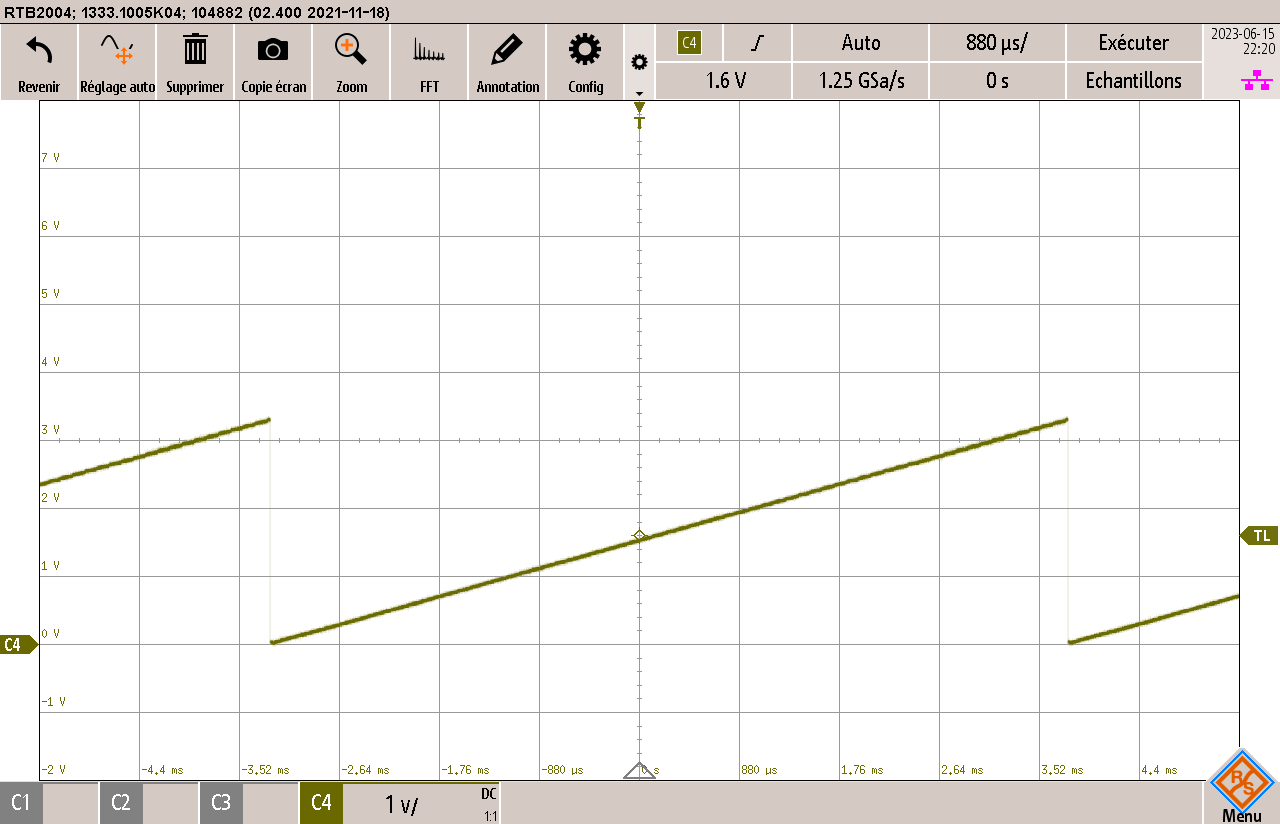
Nous voulons juste générer un signal triangle afin de vérifier le fonctionnement du DAC.

##### Signaux SPI :



Nous pouvons constater que nous avons des trames qui envoient des données différentes à la suite ainsi que le chip select qui se baisse à chaques fois qu’il envoie une tramme.

##### Forme signal

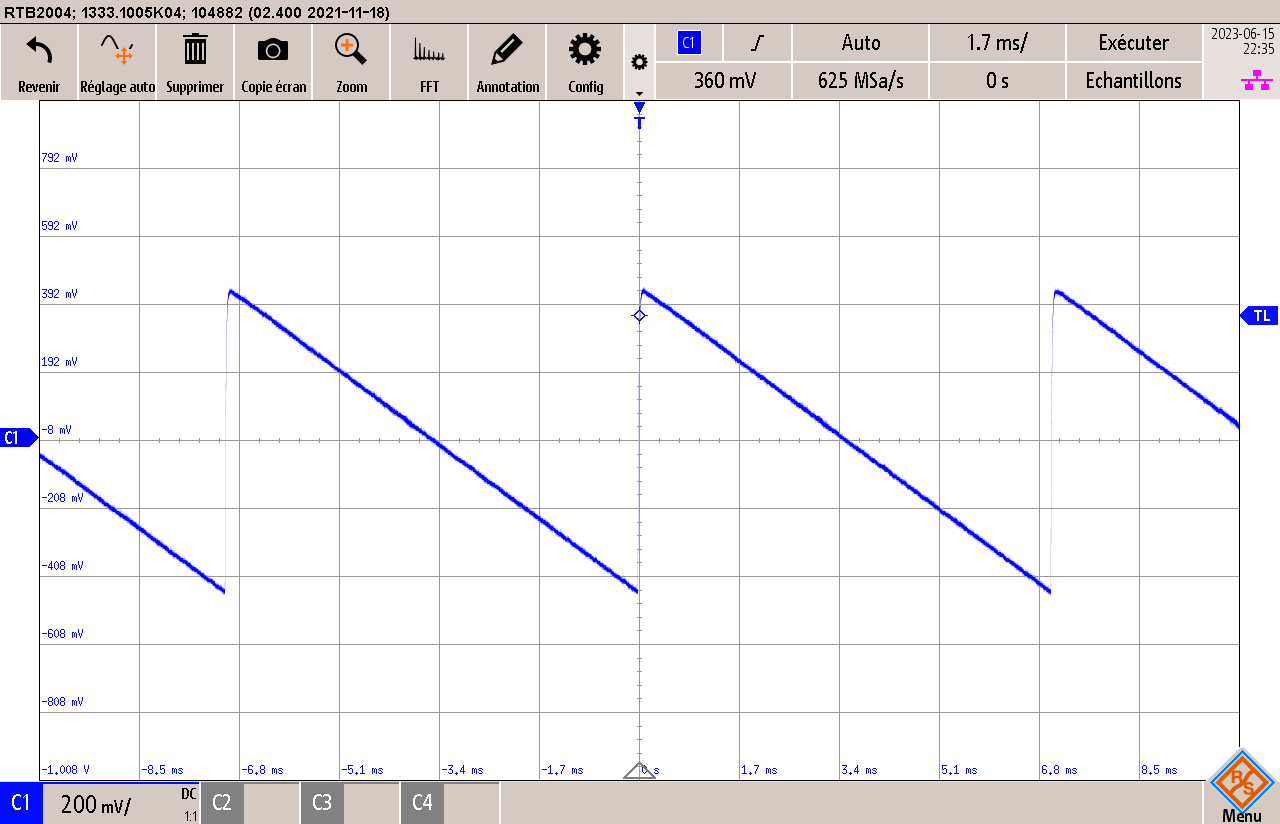


Nous pouvons voir que le signal est bien un signal triangle qui va de 0V à 3,3V.

## Test sorties audios

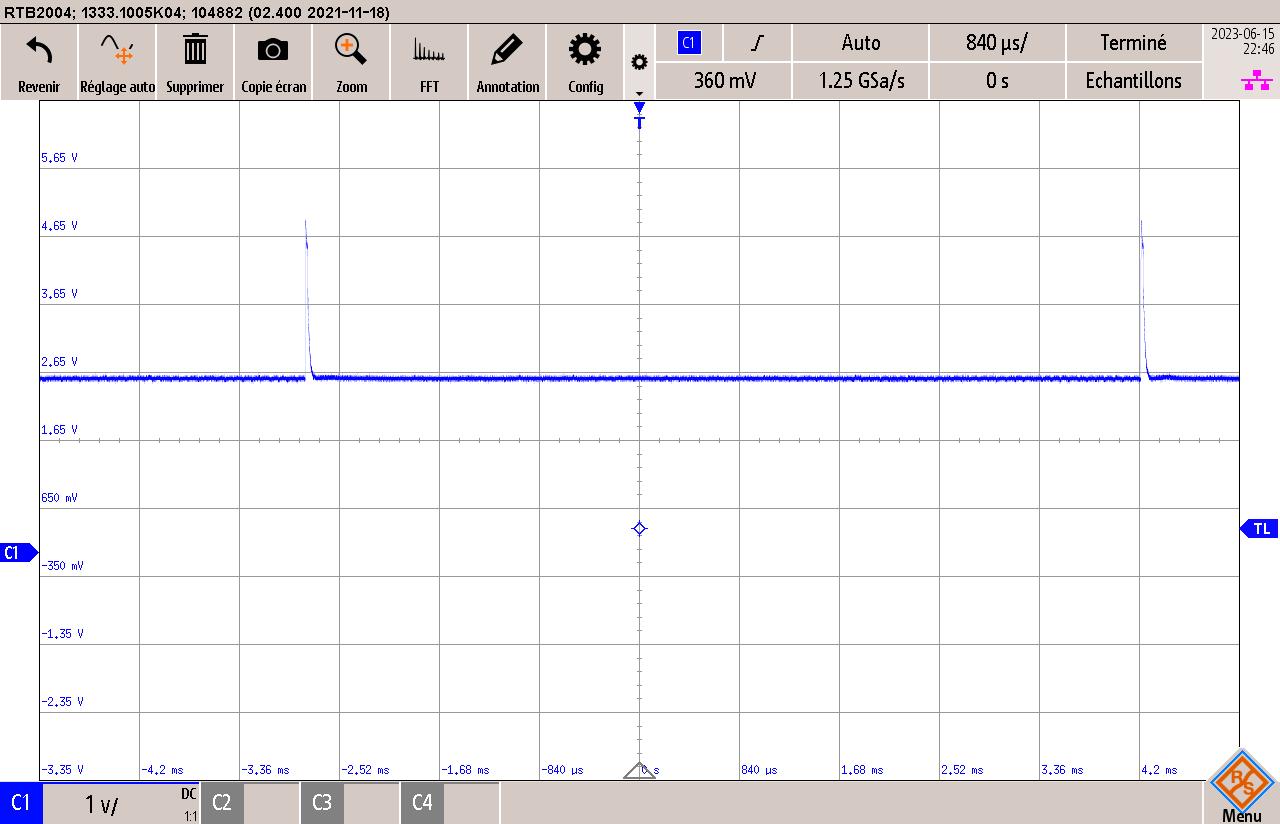
Pour vérifier les sorties, j’ai utilisé le signal que j’avais en sortie du DAC afin de vérifier que les signaux que j’ai en sortie du montage soient correcte.

Pour la sortie jack je dois avoir en sortie des mV et le même signal mais inversé car il y a une inversion.



Nous pouvons constater qu’en sortie nous avons bien le signal attendu.

Sur la sortie audio, je n’ai pas réussi à mesurer le signal correctement. Quand je tente de faire une mesure, j’obtiens les résultats suivants :



## Conclusion software

Toutes les librairies pour utiliser la carte sont faites et fonctionnent. Pour finaliser le projet, il restera plus que à faire l’état de la carte SD pour lire les fichiers .wav et ainsi pouvoir les envoyer au DAC. La majeure partie des librairies ainsi que des fonction d’envoi sur le DAC sont fonctionnelles.

# Modification à apporter

* Mettre le convertisseur 3,3V directement sur la sortie du pont de greatz ou le changer pour mettre un régulateur linéaire à la place en sortie du 5V
* Refaire les I/O du microcontrôleur afin d’avoir les bons pins pour les deux SPI
* Mettre un pont diviseur sur l’entrée VFB du convertisseur Digital-Analogique AD56200
* Mettre un qwartz externe afin d’améliorer le SYSCLK et avoir une fréquence beaucoup plus stable.
* Finir le programme afin de lire des fichiers

# Evaluation du projet

## Etat d’avancement du projet

### Carte SD

* Monté et testé
* Début de programme fonctionnel

### Entrées optocouplées

* Monté et testé
* Programme anti-rebond fait et testé

### DAC

* Monté et testé
* Programme envoi et initialisation fait et testé
* Reste à implémenter la lecture Wav en recalculant les valeurs obtenues depuis la carte SD.

### Potentiomètre

* Monté et testé
* Code non fonctionnel

### Sortie audio

* Montées et testées
* Fonctionnent en condition réelle

# Conclusion

Pour ce projet, il m’a été demandé de partir de zéro et de faire une boîte à bruitage. Le principe de celle-ci est de lire des entrées optocouplées et de ressortir un son depuis une carte SD d’après l’entrée choisie sur une sortie avec haut-parleur et une sortie jack.

La recherche de technologie n’était pas la partie la plus exigeante car en choisissant ce projet, j’avais déjà une idée du concept à faire. L’alimentation 5V Würt m’as fait gagner beaucoup de temps car cela consiste à un seul composant qui donne directement une sortie réglée.

La partie schématique et design étaient plus compliquées cor ne disposions pas de beaucoup de temps pour faire ce PCB et beaucoup d’erreur ont étés commises à cause de la précipitation et auraient pu être largement évité. Faire plus de revues auraient pu éviter ces problèmes. La recherche de composant était compliquée car nous somme en pleine crise de silicium et cela menais à beaucoup de composants avec des délais d’approvisionnement de au moins 1 mois. Heureusement que les microcontrôleurs étaient en stock et car en faisant des recherches, j’ai remarqué qu’il est très compliqué de s’en procurer.

Dans le montage et le test, cela s’est bien passé. Quelques petits problèmes qui ont vite étés réglés sauf pour celui de la carte SD. Qui celui-ci m’as fait perdre beaucoup trop de temps dans mon software et mon rapport. Heureusement, j’ai pu résoudre ce problème et continuer pour faire les autres périphériques.

Loïc David

Le16.06.2023

# Annexes

## Cahier des charges du projet

## Dossier de CAO

## Liste de pièce

## Listing code

## Journal de travail

## Mode d’emploi du système

## Résumé du projet

## Affiche du projet