**Laurent Tremblay**

*Gr.001*

**Projet final**

*Pédale numérique pour effets de guitare*

Travail présenté à

**M. Jasmin St-Laurent, M. Claude Barbaud et Mme Chantal Milette**

dans le cadre du cours

*Projet de fin d’études, 243-631 — MA*

**Collège Maisonneuve**

2018-05-14

Table des matières

[Introduction 2](#_Toc513664118)

[Bilan des travaux 3](#_Toc513664119)

[Description générale 4](#_Toc513664120)

[Schéma synoptique 4](#_Toc513664121)

[Schéma de principe 4](#_Toc513664122)

[Description précise 5](#_Toc513664123)

[Schéma de montage 6](#_Toc513664124)

[Plaquette principale 6](#_Toc513664125)

[Circuit pour l’écran graphique 9](#_Toc513664126)

[Circuit imprimé 10](#_Toc513664127)

[Circuit imprimé pour le circuit principal 10](#_Toc513664128)

[Circuit imprimé pour l’écran LCD 12](#_Toc513664129)

[Câbles 12](#_Toc513664130)

[Programmes 13](#_Toc513664131)

[Boîtier 17](#_Toc513664132)

[Caractéristiques et limites de l’appareil 17](#_Toc513664133)

[Liste de matériel 18](#_Toc513664134)

[Évaluation des coûts de production 20](#_Toc513664135)

[Procédure d’assemblage 21](#_Toc513664136)

[Procédure de calibration 21](#_Toc513664137)

[Manuel de l’utilisateur 22](#_Toc513664138)

[Conclusion 23](#_Toc513664139)

[Évaluation du choix matériel 23](#_Toc513664140)

[Modifications éventuelles à apporter 23](#_Toc513664141)

[Difficultés rencontrées 24](#_Toc513664142)

[Connaissances acquises pendant le projet 25](#_Toc513664143)

[Annexes 26](#_Toc513664144)

# Introduction

Dans le cadre du cours de projet de fin d’études, j’ai réalisé un projet électronique ayant comme fonction de modifier un signal audio provenant d’une guitare à l’aide d’un FPGA. Celui-ci possède une interface graphique pour afficher à l’utilisateur l’effet actif ainsi que les paramètres de cet effet. L’utilisateur a à sa disposition deux boutons poussoirs pour avancer et reculer entre les multiples effets ainsi qu’un bouton pour activer les effets audios ou non. Parmi ces effets, on peut retrouver un effet de distorsion et un effet de trémolo. L’effet de distorsion consiste à limiter le signal audio pour ajouter des harmoniques à celui-ci, et l’effet de trémolo consiste à moduler le signal audio à l’aide d’une onde de fréquence très basse, ce qui donne l’impression qu’on augmente et réduit automatiquement le volume du son.

J’ai choisi ce projet, car je joue de la guitare et qu’il est important d’avoir des pédales à effets pour modifier le son de la guitare. Il y a aussi le fait que les pédales à effets pour guitare sont très couteuses, car une pédale peut coûter entre 75 et 500 dollars, et celle-ci ne produit qu’un seul effet. On peut donc s’imaginer qu’on peut se retrouver avec une grosse facture seulement en possédant quelques effets.

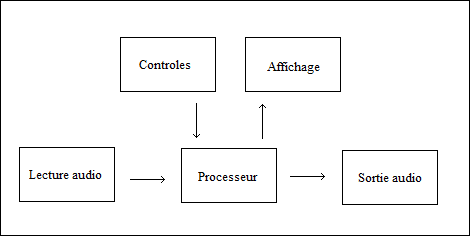
L’objectif du projet était donc de pouvoir créer tous les effets voulus pour une fraction du prix que l’ensemble des pédales standard pourraient couter. De plus, ce projet me donnait raison d’utiliser un FPGA, car ceux-ci sont extrêmement puissants pour le traitement de signaux numériques, et par ce fait approfondir mes connaissances au niveau de la composante et du langage VHDL.

# Bilan des travaux

Dans mon cahier des charges, il était spécifié que je devais créer cinq effets différents, chacun ayant trois paramètres modifiables, ainsi qu’implémenté une mémoire vive statique et un écran graphique pour pouvoir créer une pédale numérique. Cependant, suite au modifications apportés au cours, je me suis vu enlevé quelques modules du projet. Ceux-ci était trois des cinq effets ainsi que la mémoire vive statique. Suites à ces changements, je devais créer deux effets ayant trois paramètres modifiables, soit un effet de distorsion ayant comme paramètres le contrôle du niveau de distorsion, le contrôle du « ton » du son, soit plusieurs filtres passe bas pour couper les hautes fréquences, et un paramètre pour le volume du son, soit un effet de trémolo ayant comme paramètres la fréquence de l’onde modulant le signal, la forme de l’onde modulant le signal et l’intensité de l’effet, ainsi que d’implémenter un afficheur graphique pour l’interface utilisateur. Parmi ces obligations, la seule que je n’ai pu réaliser était le paramètre pour modifier le ton de l’effet de distorsion. Cela est dû au fait que je n’ai pas implémenté la mémoire vive statique dans le projet, et donc dans le FPGA je n’avais pas assez d’espace pour implémenter les filtres nécessaires pour réaliser le circuit. Il y avait une autre option, soit d’utiliser une transformation de fourrier pour faire une analyse fréquentielle du signal et de multiplier les données par des coefficients, cependant je n’ai pas eu le temps d’analyser de sujet et de comprendre comment l’implémenter dans le projet.

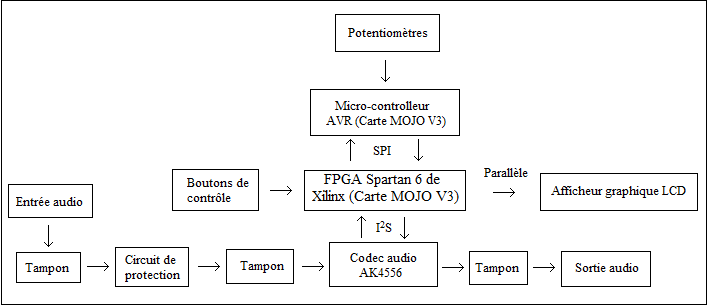
# Description générale

## Schéma synoptique



Le schéma démontre que le projet repose sur un module qui échantillonne un signal audio, et qui envoie les données au processeur, celui-ci traitant le signal qu’il lui est passé dépendamment des contrôles spécifiés par l’utilisateur.

## Schéma de principe



Il y a plusieurs principes utilisés dans ce projet. Ceux-ci peuvent être séparés en deux : soit une partie analogique et une partie numérique. Pour la partie analogique, il y a l’utilisation de l’amplificateur TL04 comme tampon et comme circuit de protection pour transmettre le signal audio au codec audio AK4556. Il est important d’utiliser un circuit tampon pour isoler les impédances, soit celle de l’entrée audio, celle de l’entrée du circuit d’acquisition du codec audio et celle à la sortie du codec audio. De plus, il faut aussi utiliser un circuit de protection pour limiter le niveau de tension de l’entrée audio, car elle pourrait endommager l’entrée du circuit d’acquisition du codec. Dans mon cas, j’ai choisi d’utiliser un circuit de compression audio : ce circuit ajuste automatiquement la tension du signal audio pour qu’elle soit à une tension fixe, prévenant les surtensions. En plus, ce choix permet d’avoir une meilleure résolution de notre signal audio, cependant il vient couper la dynamique du signal audio. Le deuxième aspect de la partie analogique est l’utilisation d’un codec audio, celui-ci permettant le suréchantillonnage et le filtrage du signal audio.

\*\* ajouter explication suréchantillonnage + filtrage \*\*

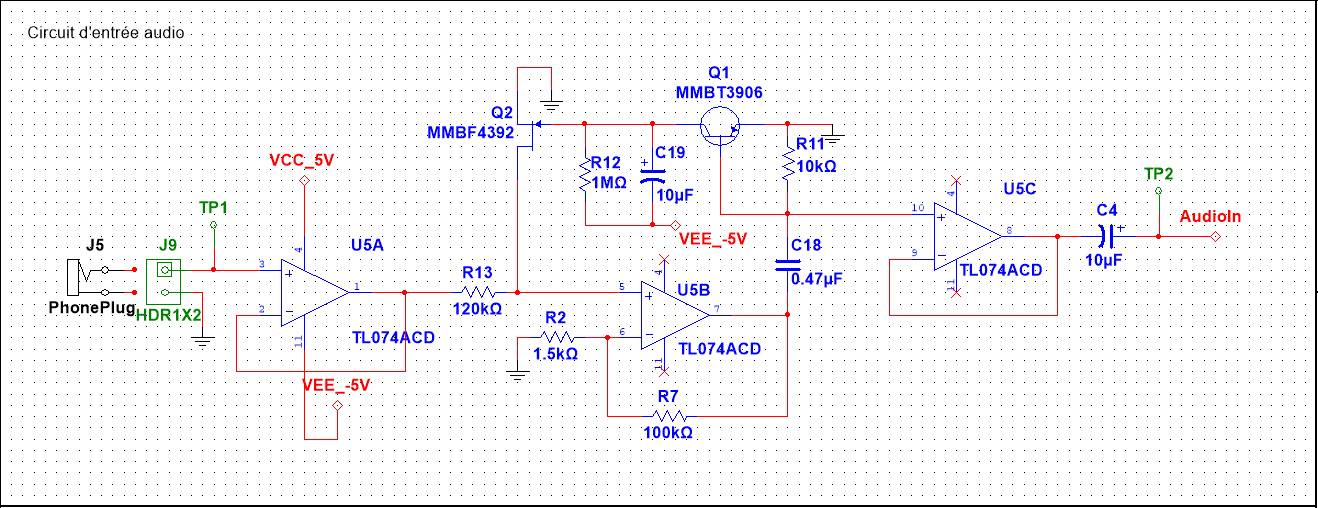
La communication entre le codec audio et le FPGA se fait à l’aide du protocole I2S.

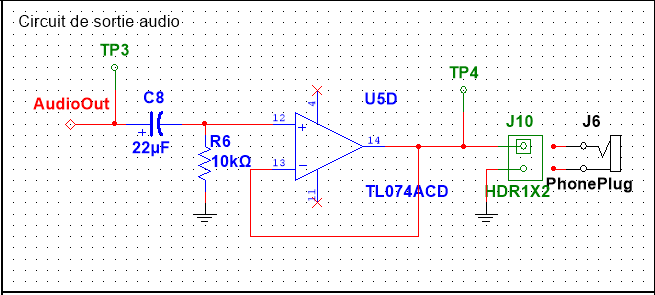
Pour la partie numérique, on peut retrouver une interface pour l’utilisateur ainsi que plusieurs protocoles de communications. L’utilisateur possède trois boutons de contrôles, soit deux pour changer de pages (précédent et prochain) ainsi qu’un bouton activant la pédale. L’utilisateur peut aussi modifier les paramètres des effets à l’aide de trois potentiomètres. Les tensions de ces potentiomètres sont lues à l’aide des convertisseurs analogiques à numérique du microcontrôleur AVR sur la carte MOJO V3. Ils sont ensuite transmis à l’aide du protocole SPI au FPGA Spartan 6. Le dernier item de l’interface utilisateur est un afficheur graphique LCD. Le FPGA envoie des données en parallèle à l’afficheur car l’interface est plus facile à implémenter par rapport à un protocole comme le SPI ou le I2C.

# Description précise

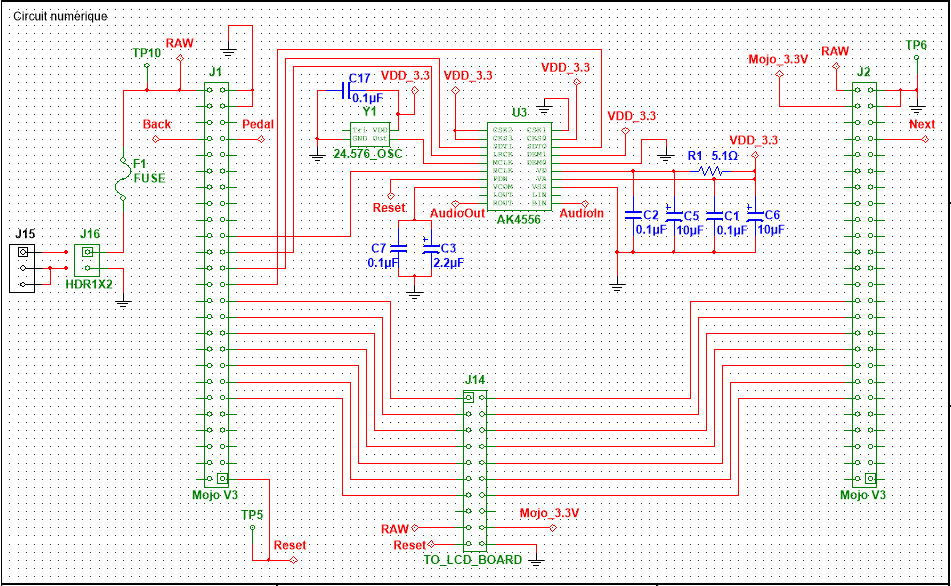
## Schéma de montage

### Plaquette principale

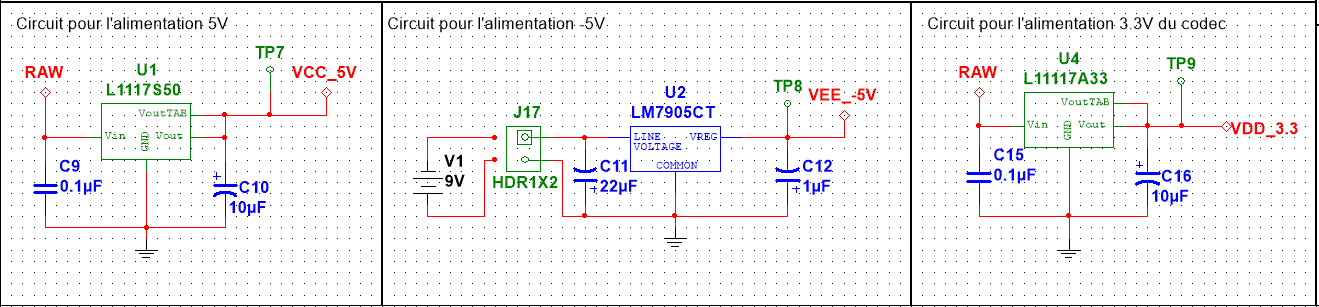
Le circuit d’entrée audio comporte trois étages. Le premier étage est un suiveur qui isole l’impédance de la guitare et celle du circuit. Le deuxième étage est un circuit de compression audio, régulant le signal d’entrée pour qu’il soit toujours à 1,31 volt. Ce circuit est important pour protéger l’entrée du convertisseur analogique à numérique du codec audio AK4556. De plus, il permet d’utiliser une plus grande résolution du convertisseur analogique à numérique. Le troisième étage est un autre suiveur qui isole l’impédance du circuit de compression et l’impédance du convertisseur analogique à numérique du codec audio. Il possède un condensateur à sa sortie pour couper le décalage courant continu.



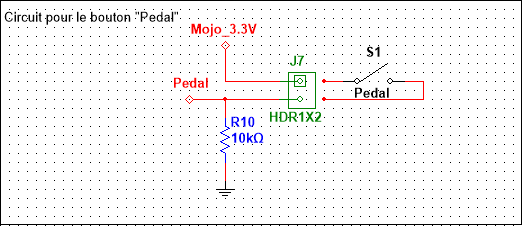
En contrepartie, le circuit audio de sortie comporte un seul étage, soit un suiveur qui vient isoler la sortie du convertisseur numérique à analogique du codec audio AK4556. Il possède aussi un filtre passe-haut à son entrée pour couper le décalage courant continu.



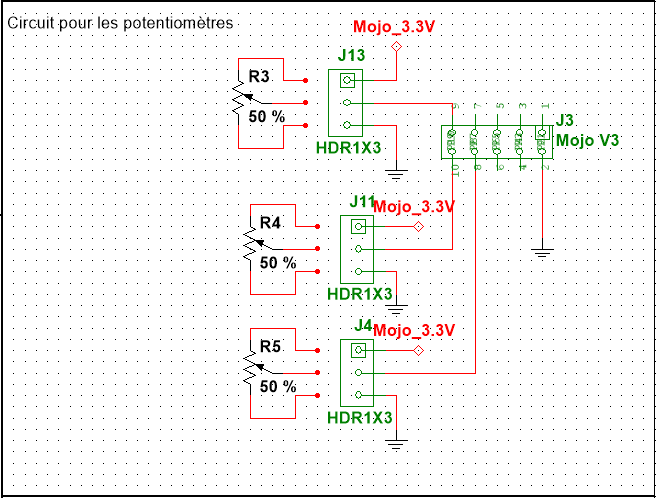
Le circuit principal du projet se retrouve ici. Le connecteur J15 est une entrée pour le bloc d’alimentation 9 V. Le connecteur J14 permet le lien entre la plaquette principale et la plaquette pour l’écran graphique. Les connecteurs J1 et J2 font le lien entre la carte Mojo et les circuits du projet. Dans ce circuit, on retrouve le codec AK4556, qui est responsable pour l’échantillonnage du signal audio.



La plaquette utilise trois alimentations, soit 5 volts et -5 volts pour l’alimentation de l’amplificateur opérationnel TL074 utilisé dans les circuits audios, ainsi que 3,3 V pour alimenter le codec audio. Il est important de noter que la carte Mojo V3 possède ses propres régulateurs, alors la tension brute du bloc d’alimentation est directement lié à l’entrée de la carte. Aussi, une batterie 9V est nécessaire pour obtenir la tension négative requise de l’amplificateur opérationnel TL074.



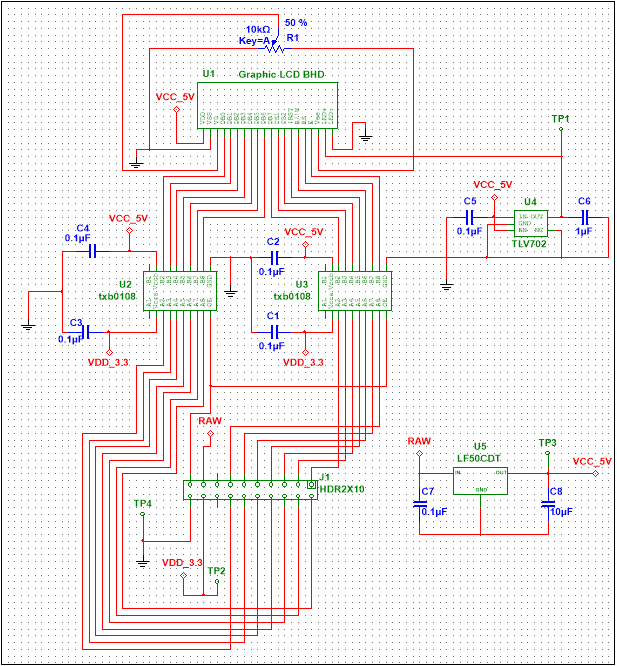
Le même circuit est utilisé pour le les boutons « Pédale », « Suivant » et « Précédent ».



Ce circuit est utilisé pour pouvoir lire la tension des potentiomètres utilisés pour modifier les paramètres des effets. Ceux-ci sont connectés au connecteur J3, qui est directement lié aux convertisseurs analogiques à numériques du microcontrôleur Atmel de la carte Mojo.

\*\*Le circuit principal complet peut être trouvé à l’annexe.

### Circuit pour l’écran graphique

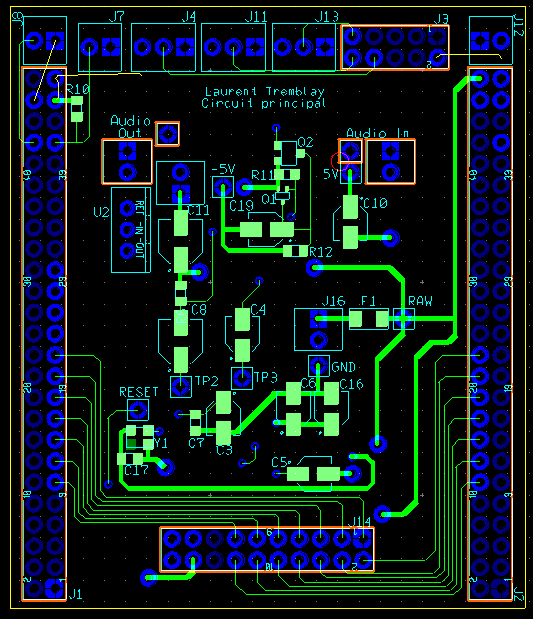


Ce circuit est le circuit complet pour interfacer l’afficheur LCD avec le FPGA. Le connecteur J1 permet la liaison entre les deux plaquettes. Le circuit possède deux régulateurs : un LF50CDT régule la tension à 5 V, soit l’alimentation nécessaire pour la logique de l’écran, et un TLV02 qui régule la tension à 3,5 V, soit l’alimentation nécessaire pour alimenter le rétroéclairage de l’écran. Le circuit possède aussi deux TXB0108, qui eux convertissent la tension provenant des tampons de sortie du FPGA, soit 3,3 V, à la tension nécessaire pour la logique de l’écran, soit 5V.

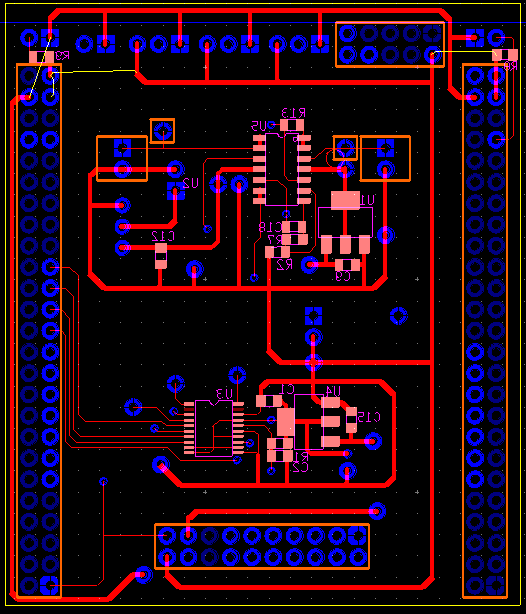
## Circuit imprimé

### Circuit imprimé pour le circuit principal

*Copper Top*

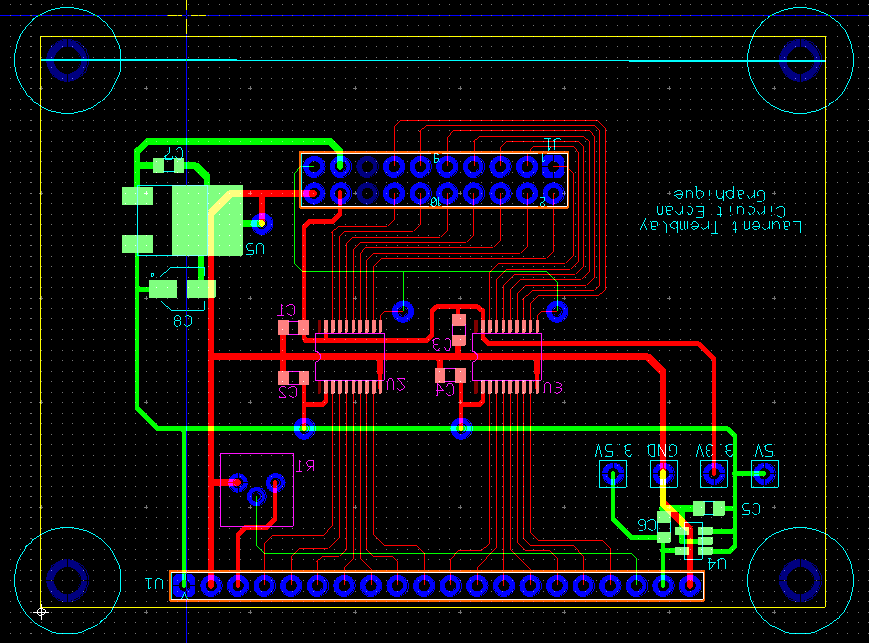


*« Copper Bottom »*



Les images ci-haut sont les côtés haut (*top)* et bas (*botom)* du circuit principal. Les connexions non reliées en jaune sont effectuées lorsque le circuit est inséré dans la carte Mojo V3. De plus, l’erreur indiquée par le point de test de l’entrée audio et de l’alimentation 5V étant trop rapproché n’est pas réellement une erreur étant donné que ceux-ci sont à 100 millièmes de pouce de distance.

### Circuit imprimé pour l’écran LCD



Ce circuit imprimé est utilisé pour alimenter l’écran graphique et pour transformer la tension 3.3V provenant FPGA pour qu’il soit converti à 5 V. La ligne bleue sérigraphié vers le bas de la plaquette nous indique où il faut couper pour enlever l’excès.

## Câbles

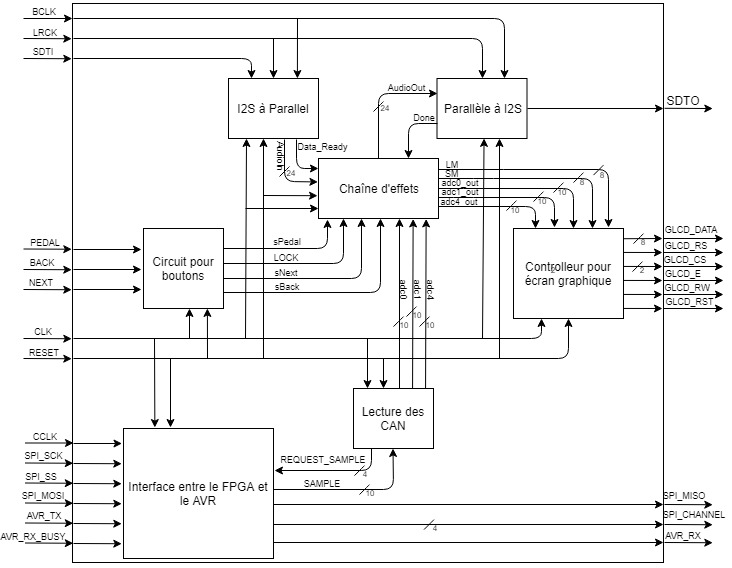
Le projet utilise deux types de câbles : un câble de type ruban plat et des câbles sertis qui sont insérés dans les connecteurs.

Le câble plat est utilisé pour échanger des données entre la carte principale et la carte de l’écran LCD. Il comporte 20 conducteurs ayant une taille de 28 AWG et doit être d’une longueur de 7 pouces.

Les câbles sertis sont utilisés pour attacher les composantes externes, soit les boutons, les potentiomètres, l’entrée et la sortie audio, ainsi que l’alimentation, au circuit principal. Ils ont tous une taille de 22 AWG. Les câbles des alimentations ont une longueur d’environ cinq pouces et doivent être de couleur rouge pour l’alimentation, et noire pour la masse. Les câbles pour les boutons et les potentiomètres, qui eux ont une longueur d’environ 2,5 pouces, suivent la même règle de couleur pour les alimentations, cependant le câble comportant les données numériques ou analogiques doit être de couleur blanche. Finalement, les câbles portant les informations du signal audio sont de couleur bleue et à une longueur d’environ 2,5 pouces.

## Programmes

**Circuit principal (« TOP »)**



**I2S à Parallèle**

Ce module permet la réception de données I2S provenant du codec audio AK4556. Il reçoit une trame de 24 bits qui sont échantillonnés sur front montant de l’horloge « bit », et qui représentent les canaux audios droit et gauche dépendamment du niveau de l’horloge « Left/Right (LR) ». Le processus est réalisé à l’aide d’un registre à décalage, qui lorsque plein, envoie un signal à l’utilisateur.

**Parallèle à I2S**

Ce module permet de prendre un registre et de le convertir en trame I2S. Pour ce faire, le module détecte un changement de niveau logique dans l’horloge sélectionnant le canal droit ou gauche, puis envoie les bits du registre à l’aide d’un registre à décalage. Lorsque toutes les données sont envoyées, le module envoie un signal disant qu’il est prêt à recevoir une nouvelle donnée dans son registre.

**Interface entre le FPGA et le AVR**

Ce module a été écrit par l’équipe de Embedded Micro en Verilog, puis a été traduite en VHDL par l’utilisateur Xark. Il est libre d’utilisation pour tous. Le module permet de faire la communication entre le microcontrôleur AVR et le FPGA se retrouvant sur la carte Mojo V3 pour ainsi lire les convertisseurs analogiques à numérique du microcontrôleur AVR. La communication entre les deux composantes se fait à l’aide du protocole SPI. L’interface permet aussi d’envoyer des données UART, cependant cette fonctionnalité n’est pas utilisée dans le cadre du projet.

**Lecture des CAN**

Ce module a été écrit dans le but de gérer l’entrée de donnée des convertisseurs analogiques à numériques. Il communique directement avec le circuit d’interface entre le FPGA et le AVR en lui demandant le canal du convertisseur analogue à numérique qu’il veut lire. Le module échantillonne 10 fois le convertisseur analogue à numérique avant d’entreposer la 11e donnée dans un registre. Cela permet au convertisseur du microcontrôleur AVR de se stabiliser, car sinon les données sont erronées. Il exécute les mêmes actions pour les autres canaux, pour revenir finalement au canal de départ.

**Chaîne d’effets**

Ce module permet de facilement enfiler et de sélectionner les effets programmés dans le FPGA. Il permet aussi de facilement interfacer les paramètres actifs de l’effet ainsi que le statut de l’effet, soit s’il est sélectionné ou s’il est bloqué, au circuit du contrôleur de l’écran graphique.

*Distorsion*

L’effet de distorsion consiste à ajouter des harmoniques au signal échantillonné. Le principe utilisé pour appliquer cet effet est d’appliquer un seuil au signal d’entré, et de couper tout ce qui plus grand (et plus petit lorsque le signal est négatif). Plus on augmente le niveau de distorsion, plus on augmente le gain appliqué à l’onde d’entrée, et étant donné que le seuil reste fixe, la forme de l’onde devient de plus en plus carrée. On peut ensuite décider d’amplifier le volume de l’onde distorsionnée à l’aide du paramètre de volume, qui vient multiplier cette onde par un gain variable. Bien que le paramètre modifiant le ton du signal n’ai pas été implémenté, il peut être retrouvé dans le fichier *toneFiltreFIR.vhd* inclus à l’annexe. Il consiste de 15 filtres impulsionnels à réponse fini de 39 coefficients qui sont sélectionnable à l’aide de la valeur du convertisseur analogue à numérique #1. En changeant la fréquence des filtres, on vient faire en sorte que la tonalité du signal devient plus gras ou plus aigu. Le programme devrait être plus optimisé lors de l’utilisation d’une mémoire vive statique.

*Trémolo*

L’effet de trémolo consiste à multiplier le signal échantillonné par une onde de basse fréquence, ce qui donne l’impression qu’on augmente et diminue le son du signal rapidement. Pour appliquer cet effet, le module génère une onde triangulaire ou carrée, dépendamment du paramètre nommé « wave », qui signifie forme de l’onde, puis on multiplie le signal entrant dans le module par la cette onde. On peut choisir de réduire ou d’augmenter la fréquence de l’onde généré en modifiant le paramètre « rate ». On peut aussi modifier l’intensité de l’effet en modifiant le paramètre « depth ». Le principe de l’intensité consiste à augmenter ou réduire la plage de l’onde généré. En effet, si on assume que l’effet n’est en fait qu’un gain variable, en réduisant la plage du gain, on réduit l’écart entre les niveaux de tensions, ce qui fait en sorte que l’effet est moins remarqué.

**Circuit pour boutons**

Ce module permet d’analyser les boutons disponibles à l’utilisateur. Ce module se sépare en deux catégories : le bouton qui active la pédale et les deux boutons pour changer le type d’effet appliqué. Parmi les sous-modules utilisés, il y a un module d’anti rebond créer par le professeur Jasmin St-Laurent du collège de Maisonneuve. J’ai obtenu son autorisation pour l’utilisation de son module.

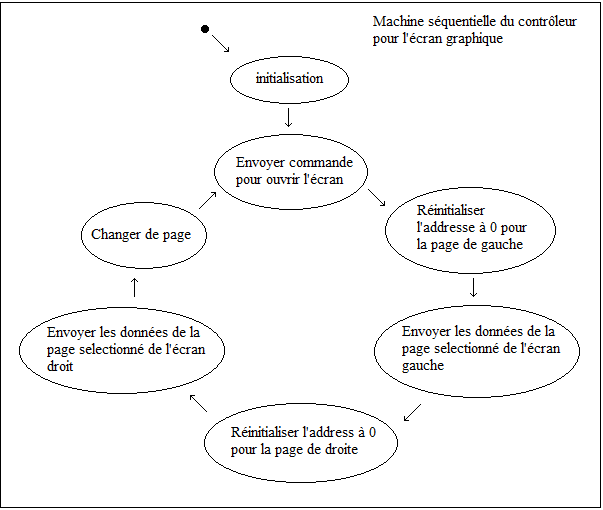
*Bouton « pédale »*

Pour le signal du bouton contrôlant si la pédale est active ou non, étant donné que le circuit de ce bouton est en fait un interrupteur ouvert ou fermé, on applique un anti-rebond de 10 millisecondes. En parallèle, il y a un processus qui observe si l’utilisateur réappuie sur le bouton dans l’espace de 500 millisecondes. Si c’est le cas, un signal est envoyé au module s’occupant des effets, indiquant que l’utilisateur veut bloquer les paramètres de l’effet actuellement actif.

*Bouton « suivant » et « précédent »*

Pour le signal des boutons suivant et précédent, on applique un anti-rebond de 10 millisecondes, puis lorsque l’anti-rebond est appliqué, on envoie un signal durant un coup d’horloge au module d’effet indiquant qu’on change d’effet.

**Contrôleur pour écran graphique**



Le module contrôlant l’écran graphique est une machine séquentielle ayant 7 états, ceux-ci étant synchronisés sur une horloge nommée E pour « enable » (activé). Lorsqu’il y a un front descendant sur cette horloge, le contrôleur sur la carte de l’écran graphique lit les données. La machine séquentielle recharge ses données sur le front montant de cette horloge, puis effectue ses actions sur le front descendant, soit de changer d’état ou d’incrémenter des compteurs, étant donné que l’écran lit les données chargées lors du front montant. Le premier état est celui de l’initialisation, où on met tous les signaux ainsi que les variables du processus a leurs valeurs initiales. De plus, on applique un signal « reset » au contrôleur de l’écran pendant 2,4 microsecondes. On tombe ensuite dans l’état où on envoie la commande nécessaire pour ouvrir l’écran. On passe ensuite au prochain état où on envoie les 64 premiers octets de la page, soit la moitié de l’écran. Lorsque tout est envoyé, on change pour le côté droit de l’écran et on réinitialise l’adresse à 0. On envoie ensuite les 64 derniers octets pour du tableau de donnés, et lorsque cela est fait, on envoie une commande à l’écran pour qu’il change de page. Pour s’assurer que l’écran fonctionne toujours, même si on le débranche ou que le FPGA s’allume avant celui-ci, on renvoie la commande pour allumer l’écran.

## Boîtier

;dimensions

;materiaux

;position des commandes et indicateurs

Trous :

Pots : ¼ inch

Pbuttons : 3/8 inch

Pedal : 3/8 inch

Alim : ¼

Switch :13/64 inch

Trou4coins : 3/32 inch

## Caractéristiques et limites de l’appareil

Consommation d’énergie : Alimentation positive : 1,35 Watt (9v x 150mA)

Alimentation négative : 0,045 Watt (9v x 5mA)

Autonomie : 100 heures avant de changer la batterie (~550mAh/5mA)

Type d’alimentation : Prise murale 9V et batterie 9V

Milieu : Maison et studio

Température : \*\*\*\*\*lire datasheets

Plage de fonctionnement : Fréquence minimale échantillonable : 20 Hz

Fréquence maximale échantillonnable : 24kHz

Vitesse d’acquisition du signal audio : 48 kHz

## Liste de matériel

Processeur



\* Le circuit principal est inséré dans la carte Mojo V3, d’où son absence de RefDes

Composantes passives



\* Le bouton activant la pédale ne se retrouve pas sur digikey, il peut être acheté chez Addison ([lien internet](https://addison-electronique.com/interrupteur-a-bouton-poussoir-on-off-2-pins-spst.html)).

Composantes actives



Connecteurs



\* Le connecteur est représenté par la composante de l’écran LCD

\*\* Il est préférable d’acheter des connecteurs de taille standard et de les couper, pour des raisons monétaires.

Accessoires



\* Le boitier utilisé ne se retrouve pas sur Digikey, il peut être acheté chez Abra Electronics ([lien internet](https://abra-electronics.com/enclosures-boxes/plastic-box-generic/gpb435-plastic-box-4.875l-x-2.5w-x-1.5h.html)).

## Évaluation des coûts de production

Les coûts de production se séparent en cinq catégories : le processeur, les composantes passives, les composantes actives, les connecteurs et les accessoires.

Le processeur central, soit le FPGA, se retrouve sur la carte Mojo V3, qui elle se vend à 129.96 CAD.

Les composantes passives, soit les résistances, les potentiomètres, les condensateurs, les boutons, l’interrupteur pour l’alimentation et le fusible, coûtent au maximum 22 dollars canadiens, le plus cher étant les boutons et les potentiomètres.

Les composantes actives, soit les régulateurs, les transistors, l’amplificateur opérationnel, l’adaptateur de tension, l’écran graphique ainsi que le codec audio, sont estimées à un coût total d’environ 42 dollars canadiens, le plus cher étant l’écran graphique coûtant 32 dollars canadiens.

L’ensemble des connecteurs, incluant les pièces pour sertir les câbles, le connecteur pour le bloc d’alimentation, le connecteur pour la batterie 9 volts, les connecteurs pour les entrées audios et le câble en ruban 20 positions, coûtent autour de 25 dollars canadiens, le plus cher étant le câble ruban, celui-ci coutant 7,50 $.

Les accessoires englobent toutes les parties du projet qui sont externes. Ils peuvent coûter plus ou moins cher et sont facilement remplaçables. Cela comprend les poignées des potentiomètres, la batterie 9V, le bloc d’alimentation 9 volts ainsi que le boitier. Le coût des accessoires est estimé à 35 dollars canadiens.

Au total, le projet est estimé à 255 dollars canadiens sans taxes.

# Procédure d’assemblage

Imprimer circuits,

Souder composantes,

Percer boitier,

Crimp les file,

Câble ruban,

Souder les fils aux périphériques externes,

Installer la carte mère dans le Mojo V3,

Mettre tous les connecteurs, dans les sockets

Installer le programme

# Procédure de calibration

La seule calibration nécessaire pour le projet est d’ajuster le potentiomètre « R1 », sur le circuit imprimé de l’écran, pour ajuster le rétroéclairage du LCD. Il faut ajuster cette résistance pour que la tension obtenue à la broche #3 de l’écran graphique soit de -3.5 volts.

# Manuel de l’utilisateur

# Conclusion

### Évaluation du choix matériel

S’il était question de recommencer le même projet, je crois qu’il serait plus intéressant d’utiliser un processeur adapté au traitement des signaux numériques à la place d’un FPGA. Bien sûr, le FPGA permet d’avoir une latence très faible entre l’entrée et la sortie du signal audio grâce à la parallélisation des calculs, cependant, un processeur adapté pour le traitement de signaux numérique permet d’obtenir des résultats presque équivalents beaucoup facilement. En effet, étant donné que ceux-ci sont conçus dans l’optique de traiter les signaux, ils possèdent plusieurs périphériques internes permettant d’effectuer des calculs intenses comme des transformations de fourrier et des divisions très rapidement. Ces périphériques sont déjà optimisés, ce qui est un avantage comparé au FPGA, où il faut tout concevoir de A à Z, et où le temps d’optimisation est plus important que le temps de développement. De plus, le FPGA a besoin de plusieurs périphériques pour fonctionner, comme de la mémoire, un programmeur, etc. Comparer au processeur de signaux numériques, qui peut fonctionner seul, ou avec quelques petites composantes externes comme des oscillateurs. Toutes ces composantes viennent augmenter la taille du projet, et étant donné que les pédales vendues sur le marché sont relativement petites, il serait beaucoup plus intéressant d’utiliser un processeur de signaux numériques pour réduire la taille de la plaquette.

### Modifications éventuelles à apporter

Pour améliorer le projet, il y a plusieurs modifications à apporter. Premièrement, il faudrait utiliser un écran LCD plus petit et adapté aux tensions du processeur. En effet, l’écran prend énormément de place, et à cause de cela il faut lui faire sa propre plaquette pour pouvoir faire les connexions entre ses broches espacées. De plus, il requiert une tension de 5 V pour les niveaux logiques hauts, ce que le processeur ne peut pas fournir. Tous ces facteurs viennent grandement augmenter le coût de production, alors comme amélioration, il serait question de trouver un écran ayant un connecteur intégré qui communique avec des niveaux de tensions « haut » de même valeur que celle du processeur. Le connecteur permettrait de connecter l’écran directement au circuit principal, et étant donné que l’on n’aura pas besoin d’adaptateurs de tensions entre le processeur et l’écran, il sera possible de les connecter directement. Deuxièmement, le circuit de compression audio devrait être amélioré. En effet, ce circuit devrait être modifié pour qu’il utilise la plage entière du convertisseur 24 bits du codec audio, et il devrait avoir un temps de réponse plus rapide. De plus, celui-ci semble avoir un problème avec les tensions trop hautes, alors il serait question d’ajouter un circuit de pré amplification pour automatiquement régler la tension provenant de la guitare à un niveau relativement faible pour que le compresseur fonctionne correctement. On pourrait aussi choisir d’améliorer les composantes utilisées pour générer le moins de bruit possible. Troisièmement, il faudrait ajouter une mémoire vive statique ou une mémoire vive dynamique externe. En effet, le projet nécessite une mémoire pour temporairement entreposer les PEPS (première entrée, première sortie) utilisés lors de l’intégration de filtres à réponse impulsionnelle finie ou lors de l’intégration de filtres à réponse impulsionnelle infinie. Le FPGA Spartan 6 utilisé lors de ce projet ne possède pas assez de mémoire interne pour permettre l’intégration de multiples filtres dans le cadre de la création des effets, ce qui est peu intéressant étant donné qu’il est nécessaire de produire des filtres lorsqu’on traite un signal audio. Dernièrement, une amélioration très intéressante serait d’ajouter un circuit qui permettrait d’avoir une alimentation bipolaire à l’aide d’une alimentation unipolaire. Pour obtenir le moins de bruits possible, il est important d’utiliser un amplificateur nécessitant une alimentation bipolaire, ce qui fait en sorte qu’on se retrouve avec deux alimentations à la place d’une. Pour éliminer cela, il faudrait ajouter un circuit, soit un convertisseur courant continu à courant continu (DC-DC) pour obtenir une tension négative à l’aide de notre tension positive.

### Difficultés rencontrées

Durant la conception du projet, je suis tombé face à face avec plusieurs problèmes. Premièrement, je me suis rendu compte assez rapidement qu’on ne pouvait pas simplement générer une tension négative en inversant les broches d’un régulateur. En effet, pour que cela fonctionne, il faut utiliser un convertisseur courant continu à courant continu, soit un détail que je n’avais aucunement prévu. Comme solution rapide, j’ai dû ajouter une pile 9 volts ainsi qu’un régulateur négatif pour obtenir une tension négative régulée pour le circuit analogique nécessitant une alimentation bipolaire. Deuxièmement, lorsque je testais mes effets, je me suis rendu compte qu’il y avait des pics de tensions aléatoires qui affectaient le signal. Ce problème venait du fait que le contrôleur ATMEL de la carte Mojo V3 échantillonne beaucoup trop vite, et par ce fait ne laisse pas assez de temps à ses convertisseurs analogues à numériques pour se décharger, ce qui fait en sorte qu’ils s’influencent tous. Troisièmement, je n’avais pas prévu de circuits de protection pour l’entrée du convertisseur analogique à numérique du codec audio. Lors de mes tests, je me suis rendu compte que le signal provenant de la guitare dépassait la tension maximale du convertisseur analogique à numérique du codec audio, et il fallait donc que je trouve une solution rapide et efficace. Celle-ci était d’implémenté un circuit de compression audio qui s’occuperais de normaliser le signal à une tension fixe, soit en augmentant les tensions faibles, et en réduisant les tensions trop hautes. Le circuit fonctionne bien pour protéger l’entrée du convertisseur analogique à numérique du Codec audio, cependant il n’est pas parfait et vient causer d’autres problèmes non voulus si la tension est trop élevée, ce qui le rend sujet à une future amélioration. Finalement, lors de la conception de mes circuits imprimés, je me suis rendu compte que les connecteurs de l’écran graphique LCD étaient trop larges pour simplement y attacher un connecteur et que la tension du niveau logique haut était plus élevée que celle de mon FPGA. Cela à donc mené à devoir créer rapidement une deuxième plaquette spécialement pour l’écran, ainsi que d’imaginer une façon de connecter cette plaquette avec la plaquette principale.

### Connaissances acquises pendant le projet

Durant la conception du projet, j’ai acquis de nombreuses connaissances par rapport à plusieurs aspects. Premièrement, j’ai beaucoup appris sur le langage VHDL ainsi que sur les FPGA.

\*\*\*\* à ré-écrire, expliquer les concepts appris je crois \*\*\*\*

Il y avait énormément de connaissances qui n’avaient pas été montrées dans les cours que nous avions eus, comme l’utilisation des variables en VHDL, qui elles sont uniquement disponibles à l’intérieur des processus, l’utilisation des « for generate » et des « if generate » en VHDL pour générer plusieurs instances d’un élément logique rapidement, l’utilisation des « integer » et des nombres naturels en VHDL, la gestion de ressources dans les FPGA, soit les blocs de mémoire vive, les blocs DSP et tous les autres modules disponibles à l’intérieur du FPGA, les broches dédiées dans les FPGA, par exemple les broches dédiées aux signaux des horloges, l’utilisation plus extensive des bancs de tests pour tester plusieurs modules entre eux, et finalement cela m’a permis d’avoir une meilleure conceptualisation de la façon dont la description matérielle est implémentée au niveau des portes logiques du FPGA à l’aide du synthétiseur.

Deuxièmement, j’ai appris à utiliser un codec audio. Ceux-ci suréchantillonnent le signal audio dans le but d’améliorer la résolution du signal audio, de réduire le bruit et de réduire la taille des filtres d’anti-repliement. Aussi, ceux-ci utilisent quelques protocoles de communications dédiés au transfert de données audios modulés par impulsions codées (PCM). Le protocole que j’ai dû apprendre était le I2S, soit un standard développé par Philips Semiconductor permettant la communication de données modulées par impulsions codées entre deux composantes. Troisièmement, j’ai appris à concevoir des circuits imprimés dans l’optique de réduire le plus possible le bruit généré par les composantes numériques dans les circuits analogiques. Pour ce faire, j’ai utilisé un document mis par Normand Provencher sur le wiki TGE (voir annexe) pour comprendre plusieurs concepts importants lors de la conception de circuit imprimé comportant des circuits analogiques et numériques comme le placement des composantes par rapport au type de circuits et par rapport à la fréquence, la façon de créer un point neutre pour les retours de courants ainsi que la façon d’utiliser des condensateurs de découplage et de dérivation. Finalement, j’ai appris à utiliser un écran graphique de type LCD pour pouvoir afficher des images programmables.

# Annexes

Document sur la conception des pcb sur wiki tge¸

Tous les programmes utilisés/conçus