Cathala Pierre Gr 3
Bidard Martin
Burton Cristobal
Pearlstein Martin

PAY DAL

I. Introduction

A) Objectifs et cahier des charges.

L'objectif est de créer une pédale loop pour guitare électrique. Cette pédale permet d'enregistrer une bande de son pour pouvoir jouer un autre morceau par dessus. Nous voulons utiliser une carte Nucleo et une carte SD pour acquérir et ressortir le son sur l'amplificateur de guitare. Il faut alors créer un code en C++ sur Mbed Keil Studio Arm.
On peut détailler les deux schémas de principes possibles:

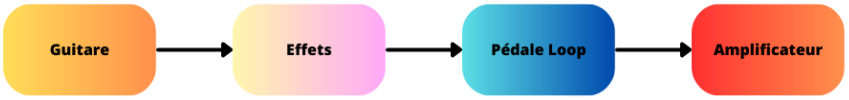


Fig 1: Schéma de principe sans modifications sur l'ampli.

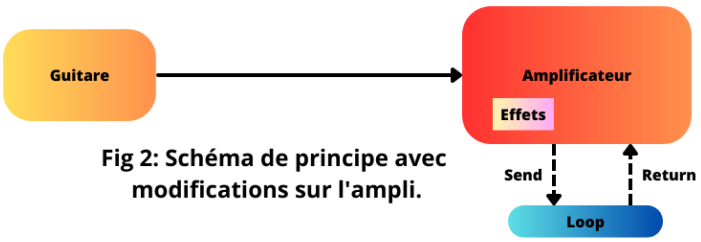


Fig 2: Schéma de principe avec modifications sur l'ampli.

On a choisi de procéder comme sur la figure 1. On a préféré rajouter une étape intermédiaire entre l'ampli et la guitare, ce qui nous a évité de devoir modifier l'amplificateur de guitare (il coûte cher !).

On a pu définir notre cahier des charges pour ce projet:

1. L'objectif est de créer **3 fonctions sur Mbed pour piloter trois boutons différents**, qui seront reliés chacun à la carte Nucleo:
   - 1 bouton pour démarrer l'enregistrement et l'arrêter.
   - 1 bouton pour vider la carte SD
   - 1 bouton pour lancer la boucle enregistrée, qui sera jouée sur l'ampli.

2. On veut pouvoir enregistrer des **boucles de 3 minutes chacune**. Cela nécessite **une carte SD avec 10 Mo de mémoire.**
3. On doit pouvoir **échantillonner à 44,4 kHz sur 8 bits.**
4. La nucléo doit être alimentée sous **3,3 V.**
5. Possibilité de **concaténer plusieurs enregistrements** dans la limite des 3 minutes possibles.

B) Schéma fonctionnel de notre pédale loop.

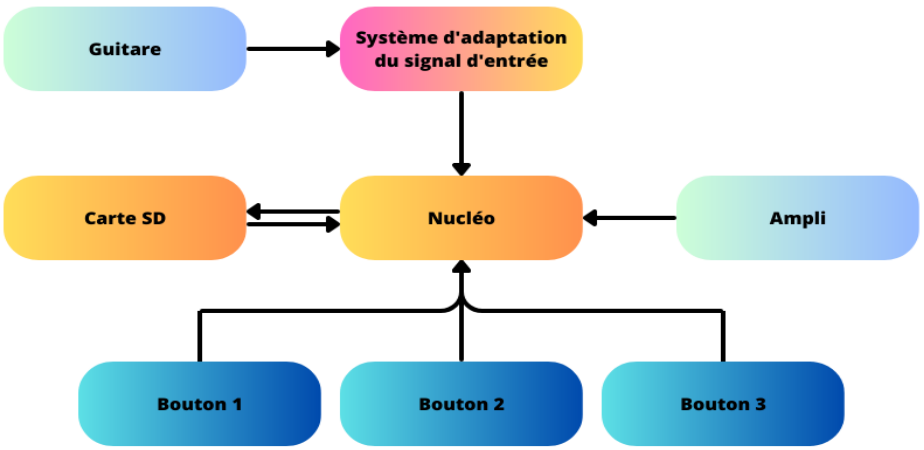


Fig 3 : Schéma fonctionnel de la *PAY DAL*

C) Notice d'utilisation

Il y a deux entrées jack sur la pédale. Il faut câbler la guitare à la première entrée jack et l'amplificateur à la deuxième. La pédale dispose de trois boutons. On appuie sur le bouton 1, le plus petit en haut à droite, pour vider la carte SD. On appuie sur le bouton 2, le plus gros, la pédale de droite, qui permet d'enregistrer le son joué par la guitare. Il faut rester appuyé sur ce bouton pour enregistrer (conseil: appuyez avec votre pied en jouant). On appuie enfin sur le bouton 3, le bouton à gauche, pour que l'enregistrement se joue en boucle. Le guitariste pourra alors continuer de jouer un morceau, alors que la boucle continue d'être jouée. Voici le prototype de la guitare en 2D et 3D:

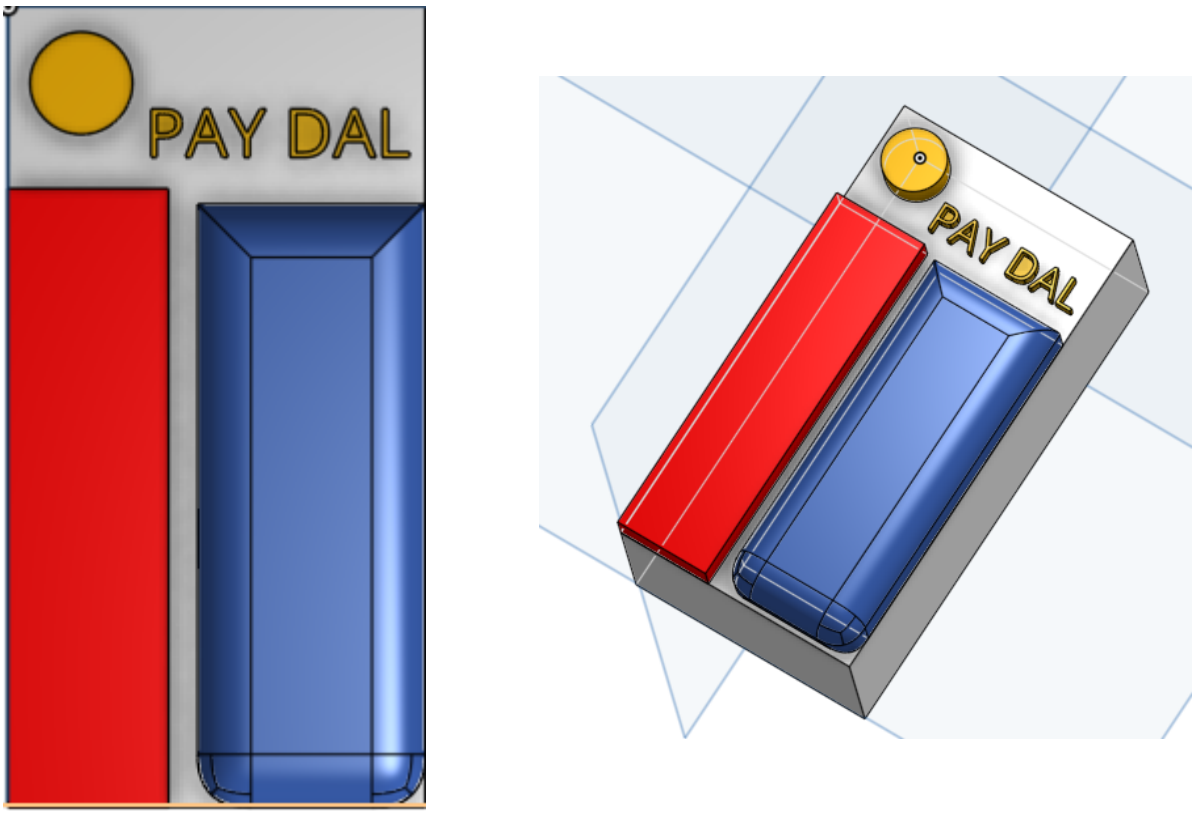
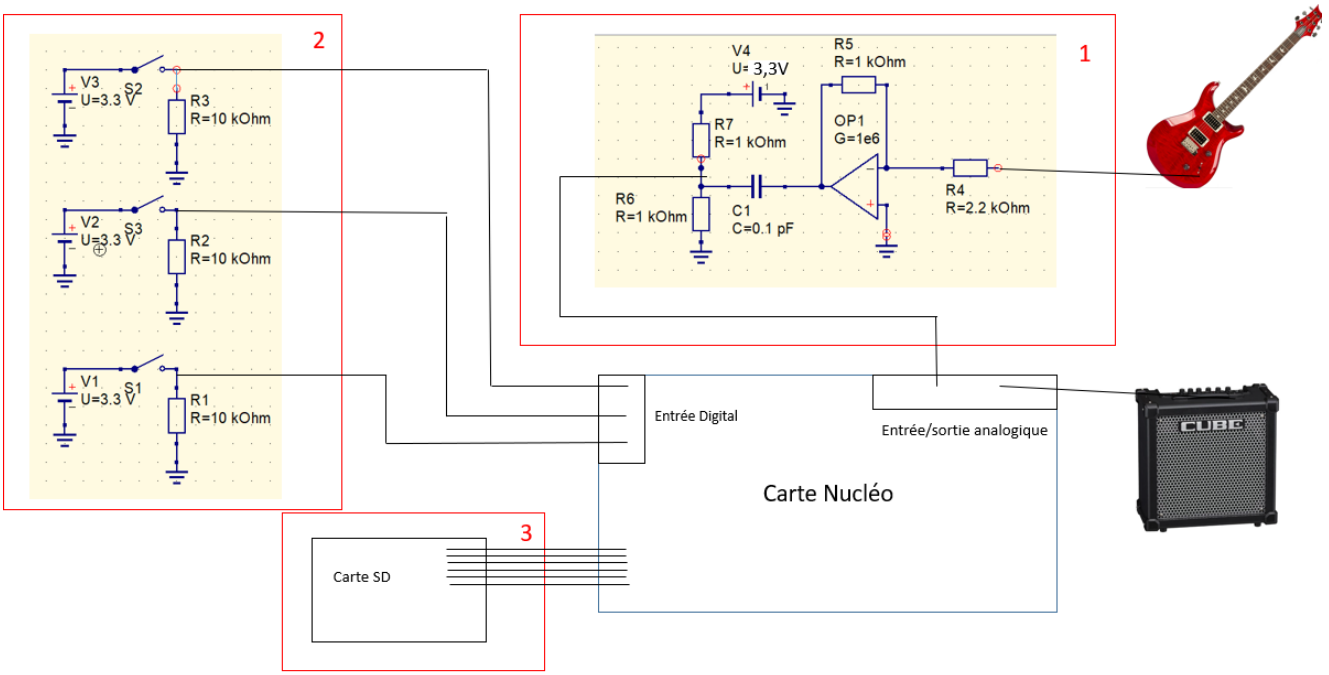


Fig 3: Prototype de la PAY DAL (*fait maison*)

II. Description du système



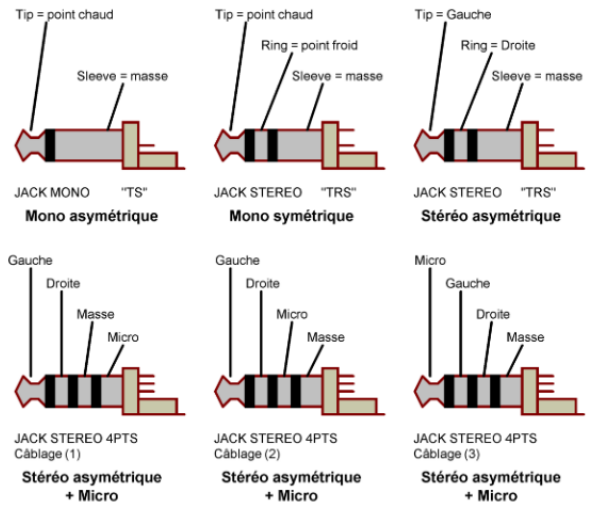
A) Mise en forme du signal (1)

1) Objectif

La première partie du montage a pour fonction de mettre en forme le signal analogique pour obtenir un signal qui soit traitable par la carte Nucléo.
La première étape est de connaître le signal qui sort directement de la guitare et de le récupérer correctement. La sortie de la guitare est une sortie jack. Une fois le signal récupéré il faut le mettre en forme pour qu'il soit adapté à la Nucléo. C'est le rôle du montage électronique qui suit. Une fois le signal mis en forme on peut l'envoyer à la carte Nucleo.

2) Réalisation

La sortie de la guitare est une sortie jack on récupère donc le signal dans un premier temps avec un câble "double jack". Un câble jack peut avoir plusieurs fonctionnalités selon que ce soit un câble mono, stéréo, stéréo+Micro....
Pour des raisons de simplicité nous utiliserons des câble jack mono le signal qui sort de la guitare n'est ainsi que sur une sortie. Il n'est pas séparé sur un côté gauche et ou sur un côté droit. On utilise un adaptateur pour passer du câble jack à des fils que l'on peut brancher sur la plaquette. Sur cet adaptateur nous avons soudé un fil à la masse et un fil aux sortie "droite" et "gauche" qui n'ont plus de raison d'être puisque l'on utilise un câble mono.



Source : Sonoelecmusic.com

On a donc récupéré notre signal il faut maintenant le mettre en forme pour qui soit exploitable par la Nucléo. Le signal commence par passer par un amplificateur/inverseur qui est là pour amplifier le signal qui est initialement très faible (quelques centaines de mV) on choisit une résistance de 2,2 kOhm en entrée et 1 kOhm

sur la boucle de rétroaction. Une fois le signal amplifié on l'envoie dans une capacité de 0,1 pF.
La dernière partie du montage est un pont diviseur de tension on met deux résistances de même valeur, on impose une tension de 3,3V (on le fait en se branchant à la carte Nucléo) à l'une des deux résistance et on récupère le signal de sortie que l'on enverra à la carte Nucléo. Le signal est ainsi compris entre 0 et 3,3V.

3) Test de validation et résultats

Pour tester cette partie du système on place un oscilloscope à différent point du circuit et on regarde le signal qui y passe. Au final le signal récupéré après le passage dans cette partie du système et qui va être utilisé par la Nucléo est bien un signal centré sur 1,65 V cependant ces variations sont faibles, la plage de variation est d' environ 200 mV ce qui est faible, la partie mise en forme du signal est donc une partie que nous pourrions améliorer pour utiliser toute la plage de tension et ainsi restituer au mieux le signal.

B) Acquisition et Gestion de l'Information

1) Définitions et Réglages

Maintenant que le signal est préparé pour être acquis par la NUCLEO, l'enjeu est de réaliser un algorithme sur Keil Studio qui réalise cette acquisition tout en respectant les critères d'échantillonnage que ce soit du point de vue du microprocesseur ou du critère de Shannon lors de la restitution ultérieure.

Commençons par définir les entrées et sorties qui nous seront utiles: hormis les boutons poussoirs et le câblage de la carte SD (cf partie x) il faut tout simplement introduire une entrée et une sortie analogique.

```
SDBlockDevice   blockDevice(D11, D12, D13, D8);  // mosi, miso, sck, cs
FATFileSystem   fileSystem("fs");
DigitalIn  boutonenregistrement(D4);
DigitalIn  boutonvide(D2);
DigitalIn  boutonmusique(D7);
AnalogOut tensionsortie(A2);
AnalogIn signal(A0);
```

On importe en parallèle de cela la bibliothèque *FATFileSystem* qui permet d'interfacer la carte SD et la carte NUCLEO

```
#include "mbed.h"
#include "SDBlockDevice.h"
#include "FATFileSystem.h"
```

Enfin, avant de pouvoir se lancer pleinement dans le code il faut s'assurer que la carte SD et la NUCLEO sont bien "connectées". Pour cela, on modifie le fichier *mbed_app.json* de manière à cibler ("*traget.components_add*") la carte SD :

```
{
    "requires": [ "bare-metal", "rtos-api", "sd","filesystem","fat_chan"],
    "target_overrides": {
      "*": {
        "target.printf_lib": "minimal-printf",
        "platform.minimal-printf-enable-floating-point": true,
        "platform.stdio-baud-rate": 115200,
        "target.components_add": [ "SD" ]
      }
    }
}
```

2) Introduction du code

```
int err = fileSystem.mount(&blockDevice);
printf("%s\n", (err ? "Fail :(" : "OK"));

// on fait l'aquisition
while(1){
    double note;
    int a = boutonenregistrement.read();
    int b = boutonmusique.read();
    int c = boutonvide.read();
```

La carte NUCLEO est donc maintenant prête à acquérir les informations analogiques! Pour entamer le code on commence par une batterie de tests pour s'assurer que le fichier est bien ouvert et la carte SD bien reliée au montage. On vérifiera ensuite à chaque utilisation des boutons qu'il n'y pas d'erreur avec la commande *if (!err)*. L'instruction "*printf*" permet de visualiser via Teraterm l'état du montage.

On introduit ensuite nos trois boutons poussoirs qui seront traités dans trois parties distinctes ultérieurement ainsi que la variable utilisée pour réaliser l'acquisition de la tension associée à la note jouée. Le montage final étant censé être un montage embarqué, nous avons besoin que le code tourne continuellement. On crée pour cela une boucle infinie avec l'instruction *while(1)*.

3) Bouton d'enregistrement

Lorsque le bouton n°1 sera activé on va effectuer à intervalle de temps régulier l'acquisition de la tension. On utilise alors la fonction .read qui réalise l'acquisition analogique et convertit la tension en un flottant compris entre 0.0 et 1.0 qui sont ensuite transmis sur la carte SD simplement via l'instruction "*fprintf*".

Lors des séances de réalisation on a pu s'assurer que la carte SD se remplissait bien mais nous avons eu des problèmes quant à la restitution du signal. Les instructions initiales sont donc en commentaires sur les figures . La méthode à ce stade serait alors de modifier le fonctionnement de la carte SD pour utiliser comme une mémoire vive. En effet, l'écriture dans une carte SD fonctionne comme un livre, ce qui induit un délai supplémentaire à chaque fois que l'on doit "changer de page" (le signal restitué était alors saccadé à intervalle régulier. Cependant faire les recherches sur ce mode de fonctionnement aurait pris trop de temps dans le cadre du projet et nous avons alors décidé d'utiliser la mémoire de la carte NUCLEO pour diminuer le temps de restitution, même si elle est très limitée (de l'ordre de 1 seconde). On définit alors un compteur *ind_sd* qui augmente de 1 à chaque acquisition jusqu'à une valeur seuil défini par "NN = 8000" qui respecte la taille de mémoire utilisée.

Enfin, le problème principal rencontré dans ce projet était la différence entre le temps d'acquisition et le temps de restitution qui était souvent plus court, ce qui signifie que lors de la restitution il y avait des coupures car le système n'avait tout simplement pas le temps d'enregistrer les notes. On a alors choisi de forcer les deux fonctions à être coordonnées en imposant un temps d'attente différent pour les deux fonctions qui compense les temps d'application respectifs (ici *wait_us*(55)).

```
if(a == 1){
    f = fopen("/fs/notes.txt", "a");
    if (!err) {
        while(a==1){

            //debug_out = 1;
            note = signal.read();
            fakesd[ind_sd] = note;
            if(ind_sd < NN) ind_sd++;
            wait_us(55);
            //fprintf(f,"%lf\n",note);
            //printf("%lf\n",note);
            a = boutonenregistrement.read();
            //debug_out = 0;
        }
        fclose(f);
    }


}
```

4) Bouton de Musique

Le fonctionnement de ce bouton est particulièrement similaire au premier, en remplaçant la commande "*fprintf*" par "*fscanf*" pour lire le fichier écrit précédemment dans la carte SD. Encore une fois, en commentaire se trouvent les instructions initiales et le code effectif est celui utilisant la mémoire de la carte NUCLEO. Pour envoyer par la sortie analogique on utilise la fonction tension.write. Par soucis de commodité on utilise cette fonction pour mettre la tension à 0 lorsqu'il n'y a plus de notes à transmettre car

```
if(b == 1){
    f = fopen("/fs/notes.txt", "r+");
    if (!err) {

        double v;
        ind_sd = 0;
        while(b==1){

            //debug_out = 1;
            //fscanf(f, "%lf", &v);
            v = fakesd[ind_sd];
            if(ind_sd < NN) ind_sd++;
            tensionsortie.write(v);
            wait_us(70);
            //printf("%lf\n",v);
            b = boutonmusique.read();
            //debug_out = 0;
        }
        fclose(f);
        tensionsortie.write(0);
    }
}
```

sinon la carte NUCLEO reste sur la dernière note jouée ce qui est très désagréable! Ici on utilise *wait_us*(70) pour accorder l'entrée et la sortie comme expliqué précédemment. En effet cette commande est plus rapide donc on lui impose un temps plus long.

5) Bouton Vide

Le code permet maintenant d'enregistrer et de jouer la musique, il suffit donc pour terminer de pouvoir vider la carte SD rapidement. On utilise pour cela l'instruction suivante qui ouvre le fichier "notes" mais avec la consigne "w", qui permet d'écrire mais en effaçant tout ce qu'il y avait avant sur le fichier!

```
if(c == 1){
        ind_sd = 0;
        f = fopen("/fs/notes.txt", "w");
        fclose(f);
    }
```

Ainsi, juste avec cette ligne on peut vider le fichier et conclure le code d'acquisition de notre pédale.

C) Stockage de l'information (3)

1) Branchement de la carte SD

On utilise un protocole de communication SPI qui permet de réaliser une transmission de données entre un maître et un esclave, ici respectivement la carte Nucléo et la carte SD. Il y a 4 fils qui font la liaison ainsi que les fils d'alimentation de la carte SD. Il y a le fil:

- SCK un signal d'horloge
- MOSI signal envoyé par le maître
- MISO signal reçu par le maître
- SS/CS signal de sélection de l'esclave

Le SS/CS est un signal qui se déclenche quand le maître transmet des informations, le signal SCK est un signal régulier qui permet de synchroniser la transmission d'information entre le maître et l'esclave. Enfin les signaux MOSI et MISO sont les signaux qui permettent de transmettre l'information.

2) Utilisation de la carte SD

On utilise la carte SD comme un système de stockage de fichier classique. On a vu que ce n'était pas l'option la plus adéquate mais c'est avec cette méthode que nous avons le plus travaillé. On utilise principalement deux bibliothèques pour utiliser la carte SD dans ces conditions, d'abord la bibliothèque *SDBlockservice* qui permet la communication en SPI, c'est de la communication bas niveau. Il y a aussi la bibliothèque *FATFileSystem* qui permet de créer une table d'allocation de fichier sur la carte SD et qui va nous servir plus concrètement pour le code, pour donner des indications de stockage de fichier. Une fois que l'on a fait les branchements et mis en place ces deux bibliothèques, la carte SD est prête à l'utilisation. On peut donc se référer au code précédemment décrit.

3) Test et validation

Comme évoqué précédemment cette méthode avec une table d'allocation de de fichier n'est pas la plus efficace, en effet à intervalle régulier on observe un temps pour écrire dans la carte SD beaucoup plus grand , cela est due à l'utilisation de la carte SD comme un "disque dur" et non comme une mémoire vive. A intervalle régulier notre échantillonnage est donc complètement faussé et cela ne permet pas une bonne restitution du signal d'origine lorsque l'on fait relire la carte SD. Ainsi si on branche l'oscilloscope sur la carte Nucléo et que l'on ajoute une indication dans le code pour déclencher un signal à chaque fois que l'on rentre dans la boucle, on peut mesurer le temps d'exécution de la boucle, on note que l'écriture dans le fichier prend environ 70 µs, cependant ce temps est régulièrement multiplier par 5 lorsque le système d'allocation de fichier "change de page". Ce qui provoque des soucis. Pour résoudre ce problème il faudrait utiliser la carte SD comme une mémoire vive, nous n'avons pas eu le temps de le faire donc nous avons utilisé la mémoire de la carte Nucléo même si elle est limitée, cette partie est plus détaillée dans la partie sur le code.

III) Bilan

A) Partie technique et avancement final

Nous avons connu des hauts et des bas dans ce projet. L'idée de départ était de réaliser une pédale wah-wah, mais la méthode du montage ne nous challengeais pas. Dès lors en Février 2023 nous avons décidé de réaliser une *pédale Loop*. Lors de nos recherches bibliographiques nous sommes tombés sur un tutoriel réalisé par Paul Graham sur la *pédale Loop*. Si le tutoriel était très complet, il ne réalisait pas du tout ce que nous cherchions. Cependant cela nous a permis de comprendre plus précisément notre système, comme la compréhension de l'amplificateur et nous avons pu nous même réaliser le cahier des charges ainsi que le schéma fonctionnel tel que le choix d'avoir trois boutons. S'en est suivi une période de codage et de réalisation du système par bloc.

alors problème de bibliothèque puis plusieurs précisions afin que cela fonctionne. Enfin lorsque ce fut opérationnel nous avons rencontré des problèmes dû à la carte Nucléo soit pour la lecture de note soit d'enregistrement. Avec plus de temps, nous pourrions essayer d'explorer la lecture et l'enregistrement de la carte SD comme une mémoire vive afin de pallier ces deux obstacles dans le même temps.

B) Rétro planning

Janvier	Février		Mars			Mai		
étude de la pédale wah wah	pédale loop	cahiers des charges et schéma fonctionnel	codage	montage du système électrique	premier tests	réajustement du code et du système	second tests	bilan

C) Répartition des tâches

Répartition des tâches					
Martin Pearlstein	le code		cahier des charges	rédaction des bilans	
Pierre			cahier des charges	rédaction des bilans	système électrique
Martin Bidard		schéma fonctionnel		rédaction des bilans	système électrique
Cristobal	le code	schéma fonctionnel			

D) Compétences acquises :

Au cours de ce projet nous avons appris à concevoir un système en procédant notamment par analyse synthèse car nous avons eu peu de modèle sur lesquels nous sommes basés. De plus, l'aspect informatique nous a permis de nous familiariser d'une façon différente avec les différentes bibliothèques à travers la recherche (parfois à taton) de fonctions et solutions possibles pour répondre aux problématiques auxquelles nous étions confrontés. Puis cela nous a permis de continuer d'approfondir le travail en équipe et son organisation. Enfin, elle fut l'opportunité de se familiariser encore plus avec le monde musical.