CRÉATION D'UN SYNTHÉTISEUR COMMANDÉ PAR UNE INTERFACE WEB

Rapport de projet

Quatrième année département

Informatique, Microélectronique et Automatique



Auteurs :
Lu XIA
Hidéo VINOT

Année :
2014/2015

Nom de la matière :
Projet de 4ème année

A l'attention de :
Mr. Alexandre BOE
Mr. Thomas VANTROYS
Mr. Xavier REDON

Remerciements

Nous tenons tout d'abord à remercier l'équipe pédagogique du département I.M.A. pour nous avoir permis de réaliser ce projet et d'acquérir de nouvelles compétences dans les divers domaines que nous avons abordés.

Nous voulons particulièrement remercier Monsieur Alexandre BOE et Monsieur Thomas VANTROYS, les encadrants du projet, qui nous ont soutenu et appuyé lors des différentes étapes.

Nous souhaitons également remercier M. Xavier REDON pour sa participation active et pour avoir répondu à toutes nos questions.

Nous souhaitons aussi remercier M. Thierry FLAMEN pour son aide et les gravures de cartes.

Sommaire

Remerciements.....2

Sommaire.....3

Introduction.....4

Présentation du projet.....5

1 Présentation générale.....5

2 Vision initial du projet.....5

3 Matériel nécessaire.....6

4 Étapes du projet.....7

Contenu du Projet.....9

1 Partie Électronique numérique.....9

2 Partie Réseau.....14

3 Partie Électronique analogique.....16

Ressentis du projet.....18

1 Difficultés rencontrées.....18

2 Bilan personnel.....18

Conclusion.....19

Annexes.....20

Introduction

Ce rapport de projet présente la réalisation d'un synthétiseur monophonique commandé depuis une page web. Nous sommes un binôme composé de Lu XIA et de Hidéo VINOT, étudiants en 4ème année à l'école Polytech'Lille en Informatique, Microélectronique et Automatique. Durant 12 semaines, nous avons travaillé selon un cahier des charges défini par M. Thomas VANTROYS et M. Alexandre BOE, nos encadrants. Ce synthétiseur opérationnel permet de jouer des notes de musique et de jouir de plusieurs modulations permettant d'en faire un instrument de musique tout à fait ludique.

Nous présenterons dans un premier temps l'idée motrice du projet, le matériel nécessaire ainsi que les différentes étapes de conception. Puis dans une seconde partie, nous aborderons en détails les parties techniques : électronique numérique, réseau et électronique analogique. Enfin, nous exposerons nos ressentis en abordant d'une part les difficultés que nous avons surmontées et d'autre part, les apports personnels du projet.

Présentation du projet

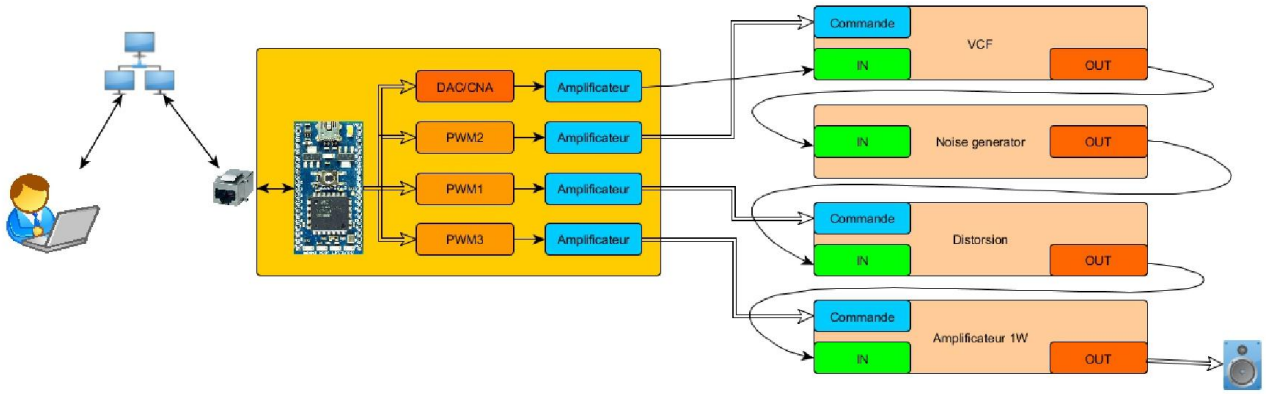
1 Présentation générale

Le but du projet est de réaliser un synthétiseur commandable par une interface web. Cette interface doit pouvoir gérer plusieurs paramètres comme la hauteur de la note, les éléments de l'enveloppe ADSR, la fréquence du LFO… Pour ce faire, nous allons utiliser la technologie SMEWS, qui permet de créer, sur un microcontrôleur, un serveur web qui commandera des fonctions de bases (PMW, DAC, ADC…). Nous avons comme matériel de départ un microcontrôleur fonctionnant à 96 Mhz ainsi que sa plate forme de développement.

Nous avons aussi à notre disposition une salle de travail et six heures de cours aménagés chaque semaine pour réaliser ce projet.

2 Vision initial du projet

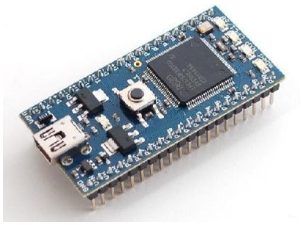
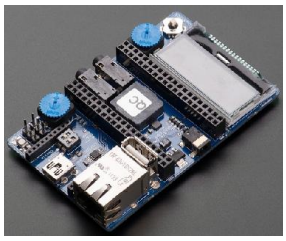
Nous avons pensé réaliser un système couplant électronique analogique et numérique. Le microcontrôleur pourrai contrôler via le DAC les notes et les modulations de bases du signal. Par ailleurs, les sorties PWM pourraient contrôler différents montage analogique, comme un *noise generator* par exemple. Voici le schéma que nous avons élaboré pour représenter le système idéal final :



L'utilisateur se connecte via le réseau sur le serveur du microcontrôleur, il active les notes, change les paramètres… Il peut aussi, par les PWM, changer les paramètres de différentes cartes.

3 Matériel nécessaire

Voici le matériel que nous avons utilisé tout au long des 12 semaines de développement.

MBED LPC1768	
Lab Board	
Bibliothèque RFLPC	
Oscilloscope	
Alimentation stabilisée	
Divers composants électroniques	
Enceinte amplifié Bose	

Le microcontrôleur est le cœur du projet. C'est à travers lui que nous allons travailler tout au long du projet. Voici les caractéristiques globales du MBED LPC1768 extraites de la *datasheet* constructeur :

Overview

Type Number	Flash	SRAM in kB				Ethernet	USB	CAN	I²	DAC	Maximum CPU Opérating fequency
		CPU	AHM SRAM0	AHM SRAM1	Total						
LPC 1768***/***	512kB	32	16	16	64	yes	Device/host/OTG	2	yes	yes	100Mhz

4 Étapes du projet

Afin de mener à bien le projet, nous avons mis en évidence différents objectifs :

- Génération de formes d'ondes complexes
- Prise en main de la technologie SMEWS
- Création de la page web
- Choix de modulation (AM, FM, phase…)
- Mise en place de l'enveloppe ADSR
- Commande externe PWM
- Réalisation de carte de modulation

Nous avons réparti les taches de manière à optimiser le temps de réalisation du projet.

a) Génération de formes d'ondes complexes

Il faut dans un premier temps découvrir comment le microcontrôleur peut générer des ondes. En d'autres termes, connaissant les caractéristiques du DAC, trouver les équations qui permettent de créer les formes d'ondes classiques d'un synthé (dent de scie, sinus, carré).

b) Prise en main de la technologie SMEWS

L'objectif de cette partie est de fabriquer un programme sur le MBED qui permette par l'appuie sur un bouton d'une interface web d'allumer une led par exemple. C'est la première étape pour comprendre le lien entre le code de la page web et le code du microcontrôleur.

c) Création de la page web

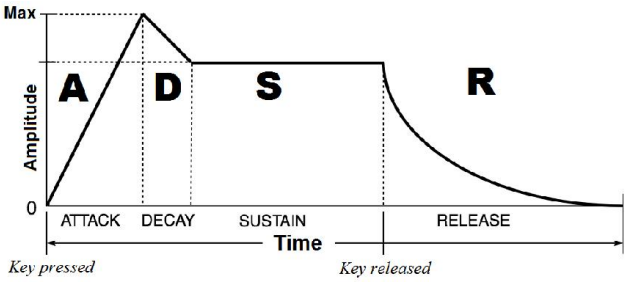
Il faudra réaliser une page web qui sera hébergée sur le serveur du microcontrôleur et qui commandera les notes et d'autres paramètres du synthétiseur.

d) Choix de modulation (AM, FM, phase…)

Il est question de mettre en place une modulation pour complexifier la texture du son. Nous allons devoir choisir une modulation et mettre en place un LFO.

e) Mise en place de l'enveloppe ADSR

L'enveloppe ADSR définit l'évolution du volume du son dans les temps à l'appui et que l'on relâche une touche comme le rappelle ce schéma :



Ici on devra pouvoir gérer les temps correspondant au période A, D et R ainsi que le niveau de S.

f) Commande externe PWM

Sur l'interface web, on pourra trouver des curseurs permettant de faire varier les valeurs de sortie des PWM. Par exemple, on pourra imaginer une tension continue variable par le biais d'un de ces curseurs ou encore faire varier la fréquence d'un signal sinusoïdal.

g) Réalisation de cartes de modulation

Pour finir, on pourra réaliser quelques cartes de modulation qui viendront ajouter encore plus de texture au son. On pense par exemple à un filtre contrôlable en tension (*Voltage Controlled Filter* VCF), au *noise générator*, ou encore à un distorsion contrôlable en tension (*Voltage Controlled Distorsion* VCD).

Contenu du Projet

1 Partie Électronique numérique

a) Génération des ondes simples

Nous allons exposer ici la technique que nous avons utiliser pour générer une forme d'onde simple :

La première étape consiste à déclarer un tableau que nous allons initialiser avec la forme d'onde simple que nous désirons, ici dans cet exemple, nous prendrons la dent de scie :

```
#define size 60
int Shape[size];
for (i=0; i<=size-1; i++)
                        {
                                if (i==0) Shape[i]=-512;
                                else Shape[i]=Shape[i-1]+(1023/size);
                        }
```

Il est à noter que la valeur pleine échelle du DAC est de 1023.

La deuxième étape est d'utiliser un des *timers* du microcontrôleur pour créer une variable qui va parcourir le tableau à une fréquence bien définie (si possible la fréquence de la note). On initialise un compteur. Lorsque celui-ci atteint la valeur 6100, il bascule vers la fonction inter_TIMER1_Shape().

```
rflpc_dac_init(RFLPC_CCLK_8);                                   //DAC initialisation

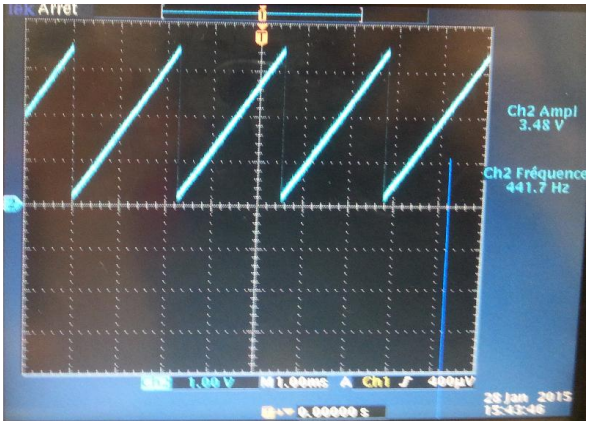
rflpc_timer_enable(RFLPC_TIMER1);                               //Timer1 enable
rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 6100); //valeur overflow=6100
rflpc_timer_set_callback(RFLPC_TIMER1, inter_TIMER1_Shape);     //callback sur inter_TIMER1_Shape()
rflpc_timer_set_clock(RFLPC_TIMER1, RFLPC_CCLK);
rflpc_timer_set_irq_on_match (RFLPC_TIMER1,RFLPC_TIMER_MATCH1,RFLPC_TIMER_IRQ_ON_MATCH |
RFLPC_TIMER_RESET_ON_MATCH);                                    // interruption sur RFLPC_TIMER_MATCH1
rflpc_timer_start(RFLPC_TIMER1);                                //Start timer1
```

La troisième chose à faire est de créer un sous-compteur dont la variable (ici FORME déclaré en variable globale) parcourt le tableau de la forme d'onde tandis que le DAC écrit cette

valeur en sortie :

```
void inter_TIMER1_Shape(void){
        FORME++ ;
        if (FORME>size) FORME=0;
        rflpc_dac_write(512+Shape[FORME]);                    //écriture sur le DAC
        rflpc_timer_reset_irq(RFLPC_TIMER1, RFLPC_TIMER_MATCH1);    //reset l'interruption du timer1
}
```

On obtient en sortie un signal de cette forme :



On remarquera que la taille du tableau correspond à la résolution de la forme d'onde. Plus celle-ci est basse, plus on retrouve des harmoniques impaires indésirables.

b) Génération des ondes complexes

Une fois que l'on a compris le principe de fabrication d'une onde simple, on peut en fabriquer d'autres et les moduler ensemble. Il ne reste plus qu'à choisir une modulation pour les assembler. On peut, par exemple, pratiquer la synthèse additive. On crée deux ondes simples à deux fréquences différentes (avec deux compteurs qui fonctionnent à des valeurs d'*overflow* différentes) puis on les additionne directement sur le DAC:

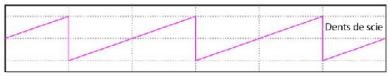
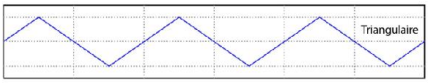
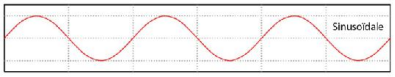
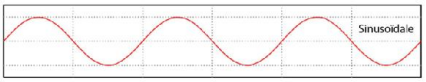
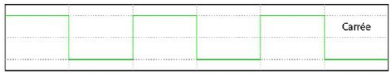
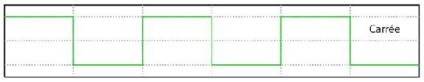
```
void inter_TIMER1_Shape1(void){
        FORME1++ ;
        if (FORME1>size1) FORME1=0;
        rflpc_dac_write(1024+Shape1[FORME1]+Shape2[FORME2]);    //écriture sur le DAC
        rflpc_timer_reset_irq(RFLPC_TIMER1, RFLPC_TIMER_MATCH1);    //reset l'interruption du timer1
}
void inter_TIMER2_Shape2(void){
        FORME2++ ;
        if (FORME2>size2) FORME2=0;
        rflpc_timer_reset_irq(RFLPC_TIMER2, RFLPC_TIMER_MATCH2);    //reset l'interruption du timer2
}
```

c) Modulation AM

Nous avons choisis d'utiliser la modulation AM parce que le nombre de compteur ne nous permet pas de créer des modulations complexes de phases ou de fréquences. Par ailleurs cette technique va nous servir pour créer l'enveloppe ADSR.

Définition : La modulation AM en audio correspond à une variation lente et périodique du volume de la note jouée.

Nous avons donc besoin de LFO (*Low Frequency Oscillator*). Nous avons choisis pour le projet d'utiliser trois formes d'ondes au choix pour la note qui est jouée (assimilé à la porteuse) et trois formes de LFO. Ce qui nous donne $3^2=9$ possibilités de formes d'ondes résultantes.

Forme d'onde de la note (porteuse)	**Représentation Temporelle**	**Forme d'onde du LFO**	**Représentation Temporelle**
Dent de scie		**Triangle**	
Sinus		**Sinus**	
Carrée		**Carrée**	

Voici un extrait de code qui permet de générer une dent de scie modulée AM par une onde carrée :

```
rflpc_dac_init(RFLPC_CCLK_8);                                   //DAC enable
rflpc_timer_enable(RFLPC_TIMER1);                               //Timer1 enable
rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 6100); //valeur overflow=6100
rflpc_timer_set_callback(RFLPC_TIMER1, inter_TIMER1_Shape);     //callback sur inter_TIMER1_Shape()
rflpc_timer_set_clock(RFLPC_TIMER1, RFLPC_CCLK);
rflpc_timer_set_irq_on_match (RFLPC_TIMER1,RFLPC_TIMER_MATCH1,RFLPC_TIMER_IRQ_ON_MATCH |
RFLPC_TIMER_RESET_ON_MATCH);          // On définie une interruption sur RFLPC_TIMER_MATCH1
rflpc_timer_start(RFLPC_TIMER1);                                //Start timer1

rflpc_timer_enable(RFLPC_TIMER2);                               //Idem TIMER1
rflpc_timer_set_match_value(RFLPC_TIMER2,RFLPC_TIMER_MATCH2, 900000);
rflpc_timer_set_callback(RFLPC_TIMER2, inter_TIMER2_LFO);
rflpc_timer_set_clock(RFLPC_TIMER2, RFLPC_CCLK);
rflpc_timer_set_irq_on_match (RFLPC_TIMER2,RFLPC_TIMER_MATCH2,RFLPC_TIMER_IRQ_ON_MATCH |
RFLPC_TIMER_RESET_ON_MATCH);
rflpc_timer_start(RFLPC_TIMER2);
```

On initialise les *timers* 1 et 2 de la même façon, à la différence que le *timer* 2 va entrer en interruption moins fréquemment que le *timer* 1. Ainsi, le *timer* 1 correspond à l'onde de la note et le *timer* 2 correspond à l'onde du LFO.

On initialise les tableaux de formes d'ondes :

```
#define size 60
#define sizeLFO 60
int Shape [size];
int SinusLFO [sizeLFO] ;

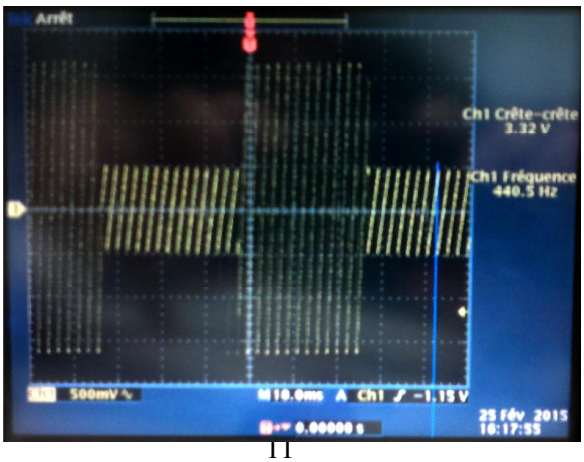
for (i=0;i<=size-1; i++)                                    //dent de scie
                                {
                                        if (i==0) Shape[i]=-512;
                                        else Shape[i]=Shape[i-1]+(1023/size);
                                }
for (i=0; i<=sizeLFO-1; i++)                                //carré
                                {
                                        if (i<=sizeLFO/2) SinusLFO[i]= 300;
                                        else  SinusLFO[i]= 1023;
                                }
```

On parcourt les tableaux grâce à des sous compteurs présents dans les *timers* 1 et 2 et on écrit sur le DAC une équation mélangeant les deux signaux:

```
void inter_TIMER1_Shape(void){                              //fonction de callback du TIMER1
        FORME++;
        if (FORME>size) FORME=0;
        rflpc_dac_write((512+Shape[FORME]*SinusLFO[LFO]/1024)+Shape[FORME]/300);
        rflpc_timer_reset_irq(RFLPC_TIMER1, RFLPC_TIMER_MATCH1);
}

void inter_TIMER2_LFO(void){                                //fonction de callback du TIMER2
        LFO++;
        if (LFO>sizeLFO)LFO=0;
        rflpc_timer_reset_irq(RFLPC_TIMER2, RFLPC_TIMER_MATCH2);
}
```

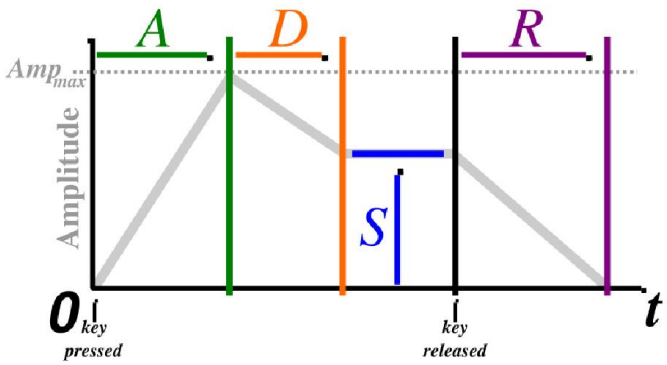
On obtient en sortie une onde de ce type :



Nous avons pu, en jouant sur les valeurs d'*overflow* des compteurs, changer la fréquence de la note et du LFO ce qui permet de jouer toute une gamme de notes et de multiplier les possibilités de textures sonores de l'instrument.

d) Généralisation à l'enveloppe ADSR

L'enveloppe ADSR représente elle aussi une modulation d'amplitude. Celle ci suit le schéma suivant :

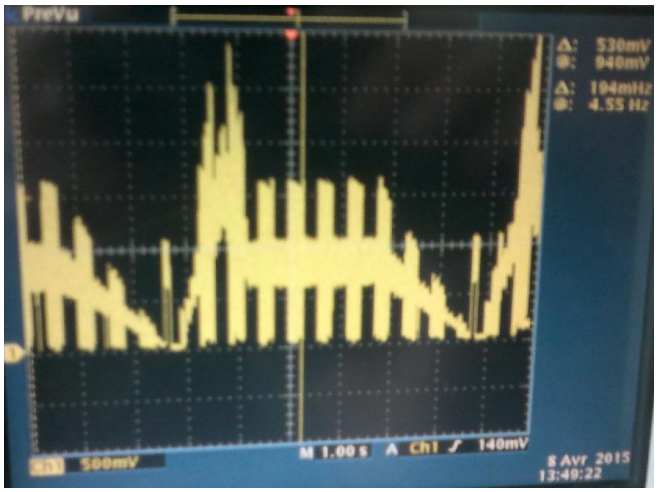


Il nous suffit de réaliser cette forme dans un tableau et de parcourir celui ci à l'aide d'un *timer* du MBED de la même façon que pour la note et le LFO. Ce *timer* entrera en interruption beaucoup moins souvent que le LFO.

Voici l'équation prenant en compte la forme de la note, la forme du LFO ainsi que l'enveloppe ADSR :

```
rflpc_dac_write(((512+Shape[FORME]*SinusLFO[LFO]/1024)+Shape[FORME]/300)*Adsr[ADSR]/1024);
```

Voici ce que l'on obtient en sortie du DAC :



On peut voir l'enveloppe ADSR ainsi que la modulation AM du LFO, mais bien sûr la note étant bien plus haute fréquence que l'enveloppe, on ne distingue pas ses périodes.

e) En résumé

Voici les différentes fonctionnalités que nous avons mises en place pour la réalisation finale :

- une gamme de 12 notes accordée
- une modulation AM dont on peut changer la fréquence d'oscillation
- trois formes d'ondes pour la note
- trois formes d'ondes pour le LFO
- une enveloppe ADSR modulable
- un sinus PWM à fréquence variable

On peut retrouver tous ces éléments dans les codes qui sont fournis en annexe.

2 Partie Réseau

Dans cette partie, nous allons présenter le travail que nous avons réalisé pour créer la commande du synthétiseur.

a) Exemple de structure de code

Voici un code simple qui génère une page web avec deux boutons permettant d'allumer et d'éteindre une LED sur le MBED :

```
<html>
 <head>
  <title>Hello World</title>
  <script language="javascript" type="text/javascript" src="ajax.js"></script>
  <script language="javascript" type="text/javascript">

function led(state){ doAjax("led","state="+state); }
function sinus(state){ doAjax("sinus","state="+state); }
  </script>
 </head>
 <body>
  Hello World!
  <button onclick="led('0');">Eteindre</button>                    //bouton Eteindre
  <button onclick="led('1');">Allumer</button>                     //bouton Allumer
 </body>
</html>
```

Voici le rendu sur le navigateur web :



Le code du fichier led.c permet d'utiliser les fonctions du MBED. Il est composé d'un entête (en vert) qui définie une fonction d'initialisation, ici init_led(), et donne le nom de la fonction qui reçoit l'argument passé sur les boutons. La valeur est contenue dans args->state.

```
/*
<generator>
      <handlers init="init_led" doGet="get_led"/>
            <properties persistence="volatile" />
            <args>
                  <arg name="state" type="uint16" />
            </args>
</generator>
*/
#include <rflpc17xx/rflpc17xx.h>
static char init_led(void)
{
   rflpc_led_init();
   return 0;
}
static char get_led(struct args_t *args)
{
            if(args->state==0) rflpc_led_clr(RFLPC_LED_1);          //Allumer la led
            else rflpc_led_set(RFLPC_LED_1);                        //Eteindre la led
            return 1;
}
```

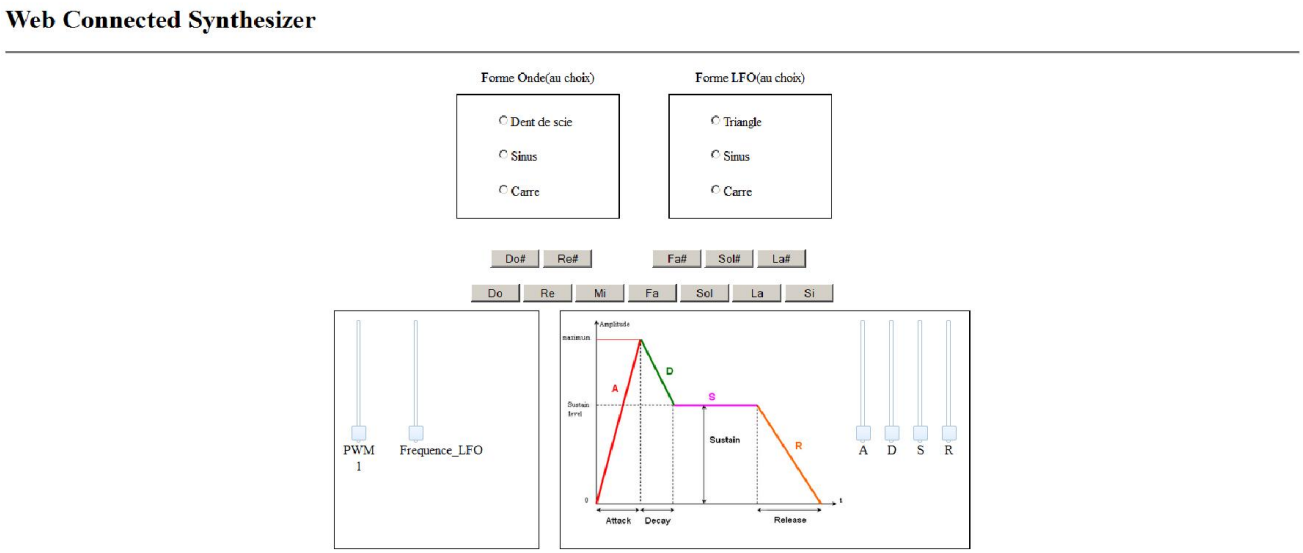
b) Compilation

Nous utilisons les fonctions de la librairie RFLPC et l'outil SMEWS qui permet d'embarquer

un serveur sur un MBED. Ces deux outils sont développés à l'Université Lille 1. On retrouve des exemples et toutes les fonctions classiques en c d'un microcontrôleur.

c) Page web finale

Voici à quoi ressemble l'interface que nous avons créé :



Le code de cette page web est disponible en **annexe 1**.

3 Partie Électronique analogique

a) Création d'un support

Le but de cette carte est de pouvoir se passer de la carte de support du MBED et d'intégrer des fonctions supplémentaires comme le filtrage des PWM. On peut voir son schéma, son PCB (non soudé) ainsi que sa photo, en **annexe 2**. Sur cette carte on retrouve un montage qui convertit la tension d'entrée (jusqu'à 18V) en une tension 5V qui va alimenter le MBED. On trouve aussi un filtrage d'ordre 2 pour les PWM suivi d'un étage d'amplification pour attaquer les cartes de modulations. Enfin, on a un étage suiveur pour la sortie du DAC afin d'adapter son impédance de sortie à n'importe quel montage.

b) Création d'un V*oltage Controlled Filter*

Ce montage est un filtre passe bas d'ordre 2 actif à base de transistors avec une résonance. A l'aide d'une tension de commande entre 0V et la tension d'alimentation, on peut faire varier la fréquence de coupure de ce filtre entre 5kHz et 800 Hz. Ainsi, si on commande avec la sortie PWM

sinusoïdale ce filtre, on obtient une auto-wah. Le *slider* PWM de la page web fait varier la fréquence de cette sinusoïde. Schéma, photos et PCB de la carte en **annexe2**.

c) Création d'un ampli

Le cahier des charges demandait de réaliser un ampli pour relier un haut parleur ou une petite enceinte. Nous avons réalisé un montage amplificateur 1W grâce à un LM386. Schéma, photos et PCB de la carte en **annexe2**.

d) Début de création d'un VCD

Nous avons commencé à travailler sur une carte de distorsion contrôlée par une tension. Nous nous somme inspiré du montage le plus simple de la *triangle big muff*, pédale de distorsion pour guitare.

Ressentis du projet

1 Difficultés rencontrées

Le cahier des charges a été respecté et le produit final a été réalisé dans les temps demandés malgré que nous ayons accumulé un fort retard au début. Nous avons eu différents problèmes avec l'installation de l'environnement de développement. En effet, la compilation du premier programme avec SMEWS n'a pas été facile. Mais, le fait de travailler dans l'urgence ne nous à pas donné d'autre choix que de créer par la suite efficacement toutes les fonctionnalités. Par ailleurs, l'IDE fourni par MBED.org nous a permis de faire les premiers tests concernant le traitement du signal. Ainsi, nous avons pu compenser le retard du aux problèmes d'installations.

Nous avons eu aussi quelques problèmes avec la carte de support qu'il a fallu produire deux fois à cause d'une conception un peu précipitée. En effet, le câblage du RJ45 n'était pas bon.

2 Bilan personnel

Nous avons trouvé que ce projet était enrichissant aussi bien sur le plan technique et théorique que sur le plan relationnel. D'abord parce que le sujet était motivant et que nous voulions entendre un résultat. Puis, parce que nous avons pu travailler ensemble et mettre en avant nos compétences respectives. L'association que nous avons faite nous a permis de progresser et de jouer du synthétiseur.

La relation avec les encadrants est également intéressante, car nous pouvons travailler sur un pied d'égalité dans le but d'apprendre de nouvelles choses sans la pression de la relation enseignant-élève. L'échange en général, avec les autres étudiants, les techniciens, le personnel travaillant au Fabricarium, donne une dimension cohérente, motivante et projette vers le haut le travail de tout à chacun.

Il serait peut-être encore plus profitable pour la réalisation des projets que nous menons au S8 de les faire commencer au S7 comme cela se fait dans d'autres écoles afin de bénéficier de plus de temps et du recule nécessaire à la gestion de projet. Nous aurions voulu porter plus loin ce projet et lui ajouter plus d'options.

Conclusion

Lors de ces douze semaines de projet, nous avons pu mettre en pratique les connaissances acquises durant notre formation et en développer de nouvelles. Cette expérience nous a permis d’anticiper les problèmes, de travailler en équipe et de réaliser un projet depuis sa genèse jusqu'à sa conclusion. Il nous a essentiellement permis de jouer avec un instrument de musique que nous modifions au fur et à mesure, ce qui donne de l'entrain à la réalisation . Et c'est sur cette note que se termine ce rapport.

Annexes

Sommaire

Annexes.....17

Annexe 1 : Codes sources.....19

1 Code HTML.....19

2 Code Microcontrôleur.....26

Annexe 2 :Carte électronique Support.....44

1 Schéma des cartes.....44

2Typon échelle 1:1 45

3 Photo de la carte (en cours de développement).....45

Annexe 3 : Carte électronique Voltage Controlled Filter. . .47

1 Schéma des cartes.....47

2 Typon échelle 1:1 47

3Photos de la carte.....48

Annexe 4 :Carte électronique Ampli.....49

1 Schéma des cartes.....49

2Typon échelle 1:1 49

3 Photos de la carte (en cours de développement).....49

Annexe 1 : Codes sources

1 Code HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Synthetiseur</title>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
            <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
        <script language="javascript" type="text/javascript" src="ajax.js"></script>
        <script language="javascript" type="text/javascript">

        //ajax permettant la mise à jour sans ouvrir de nouvelle page.
        function sinus(state){ doAjax("sinus","state="+state); }
        function LFOshape(state){ doAjax("LFOshape","state="+state); }
        function FORMshape(state){ doAjax("FORMshape","state="+state); }
        function PWM1(state){ doAjax("PWM1","state="+state); }
        function A(state){ doAjax("A","state="+state); }
        function D(state){ doAjax("D","state="+state); }
        function S(state){ doAjax("S","state="+state); }
        function R(state){ doAjax("R","state="+state); }
        function FrqLFO(state){ doAjax("FrqLFO","state="+state); }

        </script>

            <script src="dhtmlx.js" type="text/javascript"></script>
        <link href="dhtmlx.css" type="text/css" rel="STYLESHEET">

        <style>
            .dhtmlxslider_dhx_web {
                display:inline-block;
                margin-left: 25px;
                margin-right: 25px;
            }
            .formes {
                display:inline-block;
                width: 200px;
                height: 170px;
                text-align: center;
                border: 1px solid;
                margin: 10px 90px;
            }
            .tout {
                text-align:center;
            }
            .element {
                clear: both;
            }
            .dieses {
                display:inline-block;
                margin: 20px 0;
            }
            .dieses_gauche  {
                display:inline-block;
                margin: 10px;
            }
            .dieses_gauche_position {
                   position:relative;
                left:-40px;
            }
            .dieses_gauche button {
```

```
        display:inline-block;
    }
    .dieses_droit  {
        display:inline-block;
        margin-left: 10px;

    }
    .dieses_droit_position{
            position:relative;
        left:70px;
    }
    .dieses_droit button {
        display:inline-block;

    }
    .notes  {
        margin: 20px 0;
    }
    .notes button{
        display:inline-block;
    }
    .radiobutton{
        margin-top:22px;
        text-align:left;
        padding-left:45px;
}
.divcenter{
    margin-left: auto;
    margin-right: auto;
}
</style>
<script>

var mySlider1,mySlider2,mySlider3,mySlider4,mySlider5,mySlider6,mySlider7,mySlider8;
var myPop1,myPop2,myPop3,myPop4,myPop5,myPop6,myPop7,myPop8;

function doOnLoad() {

    // init slider
    //caractéristique des sliders
    mySlider1 = new dhtmlXSlider({
        parent: "PWM_1",
        size: 150,
        value: 5,
        step: 1,
        min: 0,
        max: 255,
        inverse:true,
        vertical: true

    });

    mySlider4 = new dhtmlXSlider({
        parent: "FQ_LFO",
        size: 150,
        value: 5,
        step: 1,
        min: 0,
        max: 255,
```

```
        inverse:true,
        vertical: true

    });
    mySlider5 = new dhtmlXSlider({
        parent: "A",
        size: 150,
        value: 5,
        step: 1,
        min: 0,
        max: 255,
        inverse:true,
        vertical: true

    });
    mySlider6 = new dhtmlXSlider({
        parent: "D",
        size: 150,
        value: 5,
        step: 1,
        min: 0,
        max: 255,
        inverse:true,
        vertical: true

    });
    mySlider7 = new dhtmlXSlider({
        parent: "S",
        size: 150,
        value: 5,
        step: 1,
        min: 0,
        max: 255,
        inverse:true,
        vertical: true

    });
    mySlider8 = new dhtmlXSlider({
        parent: "R",
        size: 150,
        value: 5,
        step: 1,
        min: 0,
        max: 255,
        inverse:true,
        vertical: true

    });

    // attach popup to slider
    myPop1 = new dhtmlXPopup({ slider: mySlider1 });
    myPop2 = new dhtmlXPopup({ slider: mySlider2 });
    myPop3 = new dhtmlXPopup({ slider: mySlider3 });
    myPop4 = new dhtmlXPopup({ slider: mySlider4 });
    myPop5 = new dhtmlXPopup({ slider: mySlider5 });
    myPop6 = new dhtmlXPopup({ slider: mySlider6 });
    myPop7 = new dhtmlXPopup({ slider: mySlider7 });
    myPop8 = new dhtmlXPopup({ slider: mySlider8 });
```

```
            // change popup value when slider moved
            mySlider1.attachEvent("onChange", function(value){
                PWM1(value);        //envoie dans la fonction PWM1 la valeur du slider
                updatePopupValue(value);});
            mySlider4.attachEvent("onChange", function(value){
                FrqLFO(value);      //envoie dans la fonction FrqLFO la valeur du slider
                updatePopupValue(value);});
            mySlider5.attachEvent("onChange", function(value){
                A(value);           //envoie dans la fonction A la valeur du slider
                updatePopupValue(value);});
            mySlider6.attachEvent("onChange", function(value){
                D(value);           //envoie dans la fonction D la valeur du slider
                updatePopupValue(value);});
            mySlider7.attachEvent("onChange", function(value){
                S(value);           //envoie dans la fonction S la valeur du slider
                updatePopupValue(value);});
            mySlider8.attachEvent("onChange", function(value){
                R(value);           //envoie dans la fonction Rla valeur du slider
                updatePopupValue(value);});

            // set initial value for popup
            updatePopupValue(mySlider1.getValue());
            updatePopupValue(mySlider2.getValue());
            updatePopupValue(mySlider4.getValue());
            updatePopupValue(mySlider5.getValue());
            updatePopupValue(mySlider6.getValue());
            updatePopupValue(mySlider7.getValue());
            updatePopupValue(mySlider8.getValue());
        }
        //permet l'affichage de la valeur courante du slider
        function updatePopupValue(value) {
            myPop1.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
            myPop2.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
            myPop4.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
            myPop5.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
            myPop6.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
            myPop7.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
            myPop8.attachHTML(window.dhx4.template("Current value: #value#", {value:value}));
        }

        </script>
    </head>

     <body  onload="doOnLoad()">
<!-- titre de la page -->
        <h1>Web Connected Synthesizer</h1><HR>

        <div class="tout">
```

```
        <div class="element">
            <div class="divcenter" style=
            "width:600px;text-align:center;height:200px;margin-top:20px;">
                <div style="width:200px;float:left;margin-left:60px">
                <!-- radiobouton pour le choix de l'onde-->
                    <div>Forme Onde(au choix) </div>
                    <div style="border:1px solid;height:150px;margin-top:10px;">
                        <div class="radiobutton"><label><input onchange="LFOshape('3')"name=
                        "onde" type="radio">Dent de scie</label></div>
                        <div class="radiobutton"><label><input onchange="LFOshape('4')"name=



                        "onde" type="radio">Sinus</label></div>
                        <div class="radiobutton"><label><input onchange="LFOshape('5')" name=
                        "onde" type="radio">Carre</label></div>
                    </div>
            </div>

                <div style="width:200px;float:left;margin-left:60px;">
                <!-- radiobouton pour le choix du LFO -->
                    <div>Forme LFO(au choix)</div>
                    <div style="border:1px solid;height:150px;margin-top:10px;">
                        <div class="radiobutton"><label><input onchange="LFOshape('0')" name=
                        "lfo" type="radio" >Triangle</label></div>
                        <div class="radiobutton"><label><input onchange="LFOshape('1')" name=
                        "lfo" type="radio" >Sinus</label></div>
                        <div class="radiobutton"><label><input onchange="LFOshape('2')" name=
                        "lfo" type="radio" >Carre</label></div>
                    </div>
                </div>
            </div>
        </div>

        <div class="element">
!-- Bouton pour les notes -->
!-- lorsque l'on clic sur une des notes, on envoie une valeur à la fonction sinus -->
        <div class="dieses">
           <span class="dieses_gauche">
           <span class="dieses_gauche_position">
             <button onclick="sinus('0')"type="button"style="width:60px">Do#</button>
             <button onclick="sinus('1')" type="button"style="width:60px">Re#</button>
           <span class="dieses_droit_position">
           </span>
           <span class="dieses_droit">
           <span class="dieses_droit_position">
             <button onclick="sinus('2')"type="button" style="width:60px">Fa#</button>
             <button onclick="sinus('3')"type="button"style="width:60px">Sol#</button>
             <button onclick="sinus('4')"type="button"style="width:60px">La#</button>
           </span>
        </div>
         <div class="notes">
           <button onclick="sinus('5')"type="button"style="width:60px">Do</button>
           <button onclick="sinus('6')"type="button"style="width:60px">Re</button>
           <button onclick="sinus('7')"type="button"style="width:60px">Mi</button>
           <button onclick="sinus('8')"type="button"style="width:60px">Fa</button>
           <button onclick="sinus('9')"type="button"style="width:60px">Sol</button>
           <button onclick="sinus('10')"type="button"style="width:60px">La</button>
           <button onclick="sinus('11')"type="button"style="width:60px">Si</button>
         </div>
        </div>
```

```
<div class="element">
    <div class="divcenter" style=
    "width:850px;text-align:center;height:300px;margin-top:5px">
        <div style="border:1px
        solid;width:250px;height:290px;float:left;margin-top:5px;margin-left:35px">
            <div>
                <div style="width:40px;height:200px;float:left;margin:10px 10px 0
                10px">
                    <div class="divcenter" id="PWM_1" style=
                    "height:100px;width:20px;"></div>



                    <div style="margin-top:5px;">PWM 1</div>
                </div>
            </div>
            <div>
                <div style="width:40px;height:200px;float:left;margin:10px 0 0 20px">
                    <div class="divcenter" id="FQ_LFO" style=
                    "height:100px;width:20px;"></div>
                    <div style="margin-top:5px;">Frequence_LFO</div>
                </div>
            </div>

        </div>

        <div style="border:1px
        solid;width:500px;height:290px;float:left;margin-top:5px;margin-left:25px;">
            <div style="float:left;">
            <!-- image -->
                <div><img src="adsr_envelope01.gif" width="350px"></div>
            </div>

            <div style="width:150px;float:right;">
            <div>
                <div style="float:left;margin:10px 5px 0 10px">
                    <div class="divcenter" id="A" style="height:100px;width:20px;"
                    ></div>
                    <div style="margin-top:5px;">A</div>
                </div>
            </div>
            <div>
                <div style="float:left;margin:10px 5px 0 10px">
                    <div class="divcenter" id="D" style="height:100px;width:20px;"
                    ></div>
                    <div style="margin-top:5px;">D</div>
                </div>
            </div>
            <div>
                <div style="float:left;margin:10px 5px 0 10px">
                    <div class="divcenter" id="S" style="height:100px;width:20px;"
                    ></div>
                    <div style="margin-top:5px;">S</div>
                </div>
            </div>
            <div>
```

```
                    <div style="float:left;margin:10px 5px 0 10px">
                        <div class="divcenter" id="R" style="height:100px;width:20px;"
                        ></div>
                        <div style="margin-top:5px;">R</div>
                    </div>
                </div>
                </div>
            </div>
        </div>


        </div>
        </div>
  </body>
</html>
```

2 Code Microcontrôleur

a) sinus.h

```
#ifndef __SINUS_H__
#define __SINUS_H__

#include <rflpc17xx/rflpc17xx.h>
#include <math.h>
#include <time.h>

/*Constante*/
#define size 60
#define sizeLFO 60
#define sizeADSR 120
#define sizePWM_LFO 30
#define PWM_PIN3 MBED_DIP21
#define PWM_PIN4 MBED_DIP23

/*prototype*/
static char init_A(void);
static char A1(struct args_t *args1);
static char init_D(void);
static char D1(struct args_t *args1);
static char init_S(void);
static char SSS(struct args_t *args1);
static char init_R(void);
static char RRR(struct args_t *args1);
static char init_FrqLFO(void);
static char FrqLFO(struct args_t *args1);
static char init_jjj(void);
static char init_ccc(void);
static char init_mmm(void);
static char init_PWM1(void);
static char init_PWM2(void);
static char init_dac();
void init_Shapes();
static char sinus(struct args_t *args1);
void inter_TIMER1_Shape();
void inter_TIMER2_LFO();
void inter_TIMER3_ADSR();

/*variable globales*/
int  FORME, LFO, ADSR;
int SinusLFO [sizeLFO],i, Adsr[sizeADSR];
int Shape [size];
int Shape_PWM_LFO[sizePWM_LFO];
int A,D,S,R;
int maxA, sustain;
int PWM_LFO, frq_PWM;
int note_On_Off;
int cpt;

#endif
```

b) sinus.c

```
/*
<generator>
        <handlers init="init_dac" doGet="sinus"/>
        <properties persistence="volatile" />
        <args>
                <arg name="state" type="uint16" />
        </args>
</generator>
*/

#include "sinus.h"

static char init_dac(void)
{
    init_Shapes();
    rflpc_dac_init(RFLPC_CCLK_8);                                   //DAC enable

    rflpc_timer_enable(RFLPC_TIMER1);                               //Timer1 enable
    rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 6100);     //valeur
    overflow=6100
    rflpc_timer_set_callback(RFLPC_TIMER1, inter_TIMER1_Shape);             //callback sur
    inter_TIMER1_Shape()
    rflpc_timer_set_clock(RFLPC_TIMER1, RFLPC_CCLK);
    rflpc_timer_set_irq_on_match (RFLPC_TIMER1,RFLPC_TIMER_MATCH1,RFLPC_TIMER_IRQ_ON_MATCH |
    RFLPC_TIMER_RESET_ON_MATCH); // On définie une interruption sur RFLPC_TIMER_MATCH1
    rflpc_timer_start(RFLPC_TIMER1);                                //Start timer1

    rflpc_timer_enable(RFLPC_TIMER2);                               //Idem TIMER1
    rflpc_timer_set_match_value(RFLPC_TIMER2,RFLPC_TIMER_MATCH2, 900000);
    rflpc_timer_set_callback(RFLPC_TIMER2, inter_TIMER2_LFO);
    rflpc_timer_set_clock(RFLPC_TIMER2, RFLPC_CCLK);
    rflpc_timer_set_irq_on_match (RFLPC_TIMER2,RFLPC_TIMER_MATCH2,RFLPC_TIMER_IRQ_ON_MATCH |
    RFLPC_TIMER_RESET_ON_MATCH);
    rflpc_timer_start(RFLPC_TIMER2);

    rflpc_timer_enable(RFLPC_TIMER3);                               //Idem TIMER1
    rflpc_timer_set_match_value(RFLPC_TIMER3,RFLPC_TIMER_MATCH3, 3600000/2);
    rflpc_timer_set_callback(RFLPC_TIMER3, inter_TIMER3_ADSR);
    rflpc_timer_set_clock(RFLPC_TIMER3, RFLPC_CCLK);
    rflpc_timer_set_irq_on_match (RFLPC_TIMER3,RFLPC_TIMER_MATCH3,RFLPC_TIMER_IRQ_ON_MATCH |
    RFLPC_TIMER_RESET_ON_MATCH);
    rflpc_timer_start(RFLPC_TIMER3);

    return 1;
}

void inter_TIMER1_Shape(void){                                      //fonction de
callback du TIMER1
    FORME++;
    if (FORME>size) FORME=0;

    if(note_On_Off==1)
    {
        /****** Ecriture sur le DAC  ******/
        rflpc_dac_write(((512+Shape[FORME]*SinusLFO[LFO]/1024)+Shape[FORME]/300)*Adsr[ADSR]/
        1024);
    }
    rflpc_timer_reset_irq(RFLPC_TIMER1, RFLPC_TIMER_MATCH1);            //reset interruption
    TIMER1
```

```
}

void inter_TIMER2_LFO(void){                                    //fonction de
callback du TIMER2
    LFO++;
    if (LFO>sizeLFO)LFO=0;
    rflpc_timer_reset_irq(RFLPC_TIMER2, RFLPC_TIMER_MATCH2);    //reset interruption
    TIMER2
}

void inter_TIMER3_ADSR(void){                                   //fonction de
callback du TIMER3
    if(note_On_Off==1)
    {
        ADSR++;
        if (ADSR>sizeADSR){ADSR=0;note_On_Off=0;}
    }
    else ADSR=0;
    //LFO PWM
    cpt++;
    if (cpt>frq_PWM)
    {
        PWM_LFO++;
        cpt=0;
        if (PWM_LFO>sizePWM_LFO){PWM_LFO=0;}
    }
    rflpc_timer_reset_irq(RFLPC_TIMER3, RFLPC_TIMER_MATCH3);    //reset interruption
    TIMER3

}


void init_Shapes(void)                          //initialisation des tableaux en début
de programme
{
        /******Shape*******/
    for (i=0; i<=size-1; i++)
        {
            if (i==0) Shape[i]=-512;
            else Shape[i]=Shape[i-1]+(1023/size);
        }
        /******LFO*******/
    for (i=0; i<=sizeLFO-1; i++)
        {
            if (i<=sizeLFO/2) SinusLFO[i]= 300;
            else  SinusLFO[i]= 1023;
        }
        /******ADSR*******/
    A=10; D=45; S=46; R=sizeADSR-1; maxA=1023; sustain=511;
    for (i=0; i<=R; i++)
    {
        if(i<A)
        {
            if (i==0) Adsr[i]= 0;
            else Adsr[i]=Adsr[i-1]+(maxA/(A-1));
        }
        else Adsr[i]=sustain;
        if(A<=i&&i<D)
        {
```

```
            Adsr[i]=Adsr[i-1]-((maxA-sustain)/(D-A));
        }
        else  if(D<=i&&i<S)
                {
                    Adsr[i]=sustain;
                }
                else if(S<=i&&i<R)
        {
            Adsr[i]=Adsr[i-1]-(sustain/(R-S));
        }
    }
    /******Initialisation de la forme de la PWM*******/
    int Shape_PWM_LFO1[30]= {0,106,208,301,380,443,487,509,509,487,444,381,301,209,107,1,-106
    ,-207,-300,-380,-443,-501,-512,-501,-468,-415,-344,-257,-160,-55};
    for (i=0;i<=size-1; i++)
            {
                Shape_PWM_LFO[i]=2*Shape_PWM_LFO1[i];
            }
    note_On_Off=0;
    PWM_LFO=0;
    frq_PWM=0;
}

static char sinus(struct args_t *args1)                    //Fonction sinus permet de
changer la note jouée
{
     if (args1->state==0)                                  //Do#
    {
        if (note_On_Off==0)
        {
            note_On_Off=1;
            rflpc_timer_stop(RFLPC_TIMER1);                //stop TIMER1
            rflpc_timer_stop(RFLPC_TIMER3);                //stop TIMER3

            //traitement
            ADSR=0;                                        // la note reprend au début de
            l'enveloppe ADSR
            rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 5734); //changement
            de la valeur d'overflow
            rflpc_timer_reset(RFLPC_TIMER1);               //IMPORTANT: reset le TIMER1
            (fait perdre une periode) mais évite le blocage

            rflpc_timer_start(RFLPC_TIMER1);               //start TIMER1
            rflpc_timer_start(RFLPC_TIMER3);               //start TIMER3
        }
        else
        {
            rflpc_timer_stop(RFLPC_TIMER3);                //idem
            rflpc_timer_stop(RFLPC_TIMER1);

            //traitement
            ADSR=0;
            rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 5734);
            rflpc_timer_reset(RFLPC_TIMER1);

            rflpc_timer_start(RFLPC_TIMER1);
            rflpc_timer_start(RFLPC_TIMER3);
        }
    }
```

```
else if (args1->state==1)                                   //Ré#
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 5107);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 5107);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }

}
else if (args1->state==2)                                   //Fa#
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4293);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4293);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
```

```
    }
}
else if (args1->state==3)                                   //Sol#
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3827);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3827);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==4)                                   //La#
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3409);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3409);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
```

```
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==5)                                   //Do
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 6086);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 6086);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==6)                                       //Ré
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 5403);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 5403);
        rflpc_timer_reset(RFLPC_TIMER1);
```

```
        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==7)                                   //Mi
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4814);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4814);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==8)                                     //Fa
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4552);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4552);
        rflpc_timer_reset(RFLPC_TIMER1);
```

```
        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==9)                                          //Sol
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4062);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 4062);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
}
else if (args1->state==10)                                         //La
{
    if (note_On_Off==0)
    {
        note_On_Off=1;
        rflpc_timer_stop(RFLPC_TIMER1);
        rflpc_timer_stop(RFLPC_TIMER3);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3610);
        rflpc_timer_reset(RFLPC_TIMER1);

        rflpc_timer_start(RFLPC_TIMER1);
        rflpc_timer_start(RFLPC_TIMER3);
    }
    else
    {
        rflpc_timer_stop(RFLPC_TIMER3);
        rflpc_timer_stop(RFLPC_TIMER1);

        //traitement
        ADSR=0;
        rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3610);
```

```
            rflpc_timer_start(RFLPC_TIMER3);
        }
    }
    else if (args1->state==11)                                          //Si
    {
        if (note_On_Off==0)
        {
            note_On_Off=1;
            rflpc_timer_stop(RFLPC_TIMER1);
            rflpc_timer_stop(RFLPC_TIMER3);

            //traitement
            ADSR=0;
            rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3222);
            rflpc_timer_reset(RFLPC_TIMER1);

            rflpc_timer_start(RFLPC_TIMER1);
            rflpc_timer_start(RFLPC_TIMER3);
        }
        else
        {
            rflpc_timer_stop(RFLPC_TIMER3);
            rflpc_timer_stop(RFLPC_TIMER1);

            //traitement
            ADSR=0;
            rflpc_timer_set_match_value(RFLPC_TIMER1,RFLPC_TIMER_MATCH1, 3222);
            rflpc_timer_reset(RFLPC_TIMER1);

            rflpc_timer_start(RFLPC_TIMER1);
            rflpc_timer_start(RFLPC_TIMER3);
        }
    }
    return 1;
}
```

c) LFOshape.c

```
/*
<generator>
        <handlers init="init_mmm" doGet="LFOshape"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/
#include "sinus.h"

static char init_mmm(void){
    return 1;
}

static char LFOshape(struct args_t *args1)
{
                /****LFO -----> Triangle***/
    if(args1->state==0)
    {
            rflpc_timer_stop(RFLPC_TIMER1);                          //stop TIMER1
            for (i=0;i<=sizeLFO-1; i++)
            {
                if (i==0) SinusLFO[i]=-255;
                else SinusLFO[i]=(SinusLFO[i-1]+(2047/sizeLFO));
                if (i>=sizeLFO/2)SinusLFO[i]=SinusLFO[i-1]-(2047/sizeLFO);
            }
            rflpc_timer_start(RFLPC_TIMER1);                         //start TIMER1
    }
                /****LFO -----> Sinus***/
    else if(args1->state==1)
        {
            rflpc_timer_stop(RFLPC_TIMER1);                          //stop TIMER1
            int SinusLFO1[60]={0,53,106,158,208,256,301,342,380,414,443,468,487,501,
            509,512,509,501,487,468,444,415,381,343,301,257,209,159,107,54,1,-53,
            -106,-157,-207,-255,-300,-342,-380,-414,-443,-467,-487,-501,-509,-512,
            -509,-501,-487,-468,-444,-415,-381,-344,-302,-257,-210,-160,-108,-55};
            for (i=0;i<=sizeLFO-1; i++)
            {
                SinusLFO[i]=SinusLFO1[i];
            }
            rflpc_timer_start(RFLPC_TIMER1);                         //start TIMER1
        }
                /****LFO -----> Carré***/
        else if(args1->state==2)
            {
                rflpc_timer_stop(RFLPC_TIMER1);                      //stop TIMER1
                for (i=0; i<=sizeLFO-1; i++)
                {
                    if (i<=sizeLFO/2) SinusLFO[i]= 300;
                    else  SinusLFO[i]= 1023;
                }
                rflpc_timer_start(RFLPC_TIMER1);                     //start TIMER1
            }
```

```
            /****Forme d'onde -----> Dent de scie****/
        else if(args1->state==3)
        {
            rflpc_timer_stop(RFLPC_TIMER1);                          //stop TIMER1
            for (i=0;i<=size-1; i++)
            {
                if (i==0) Shape[i]=-512;
                else Shape[i]=Shape[i-1]+(1023/size);
            }
            rflpc_timer_start(RFLPC_TIMER1);                         //start TIMER1
        }
            /****Forme d'onde -----> Sinus ****/
            else if(args1->state==4)
            {
                rflpc_timer_stop(RFLPC_TIMER1);                      //stop TIMER1
                int Shape1[60]={0,53,106,158,208,256,301,342,380,414,443,468,487,
                501,509,512,509,501,487,468,444,415,381,343,301,257,209,159,107,
                54,1,-53,-106,-157,-207,-255,-300,-342,-380,-414,-443,-467,-487,
                -501,-509,-512,-509,-501,-487,-468,-444,-415,-381,-344,-302,-257,
                -210,-160,-108,-55};
                for (i=0;i<=size-1; i++)
                {
                    Shape[i]=Shape1[i];
                }
                rflpc_timer_start(RFLPC_TIMER1);                     //start TIMER1
            }
            /****Forme d'onde -----> Carré ****/
                else if(args1->state==5)
                {
                    rflpc_timer_stop(RFLPC_TIMER1);                  //stop TIMER1
                    for (i=0; i<=size-1; i++)
                    {
                        if (i<=size/2) Shape[i]=-512;
                        else Shape[i]=511;
                    }
                    rflpc_timer_start(RFLPC_TIMER1);                 //start TIMER1
                }
    return 1;
}
```

d) FrqLFO.c

```
/*
<generator>
        <handlers init="init_FrqLFO" doGet="FrqLFO"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/

#include "sinus.h"
static char init_FrqLFO(void){
    return 1;
}

static char FrqLFO(struct args_t *args1){

    rflpc_timer_stop(RFLPC_TIMER2);
    //Stop TIMER2
    rflpc_timer_set_match_value(RFLPC_TIMER2,RFLPC_TIMER_MATCH2, (15000+1500*args1->state));
    //Changement de la valeur d'overflow
    rflpc_timer_reset(RFLPC_TIMER2);
    //Reset TIMER2
    rflpc_timer_start(RFLPC_TIMER2);
    //Start TIMER2
    return 1;
}
```

e) A.C

```
/*
<generator>
        <handlers init="init_A" doGet="A1"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/

#include "sinus.h"
static char init_A(void){

    return 1;
}

static char A1(struct args_t *args1){

    rflpc_timer_stop(RFLPC_TIMER1);                                    //stop TIMER1
    A=19*args1->state/255+1;                                           //Changement de
    la valeur A

    for (i=0; i<=sizeADSR-1; i++)                                      //Calcul de
    l'envellope
    {
        if(i<A)
        {
            if (i==0) Adsr[i]= 0;
            else Adsr[i]= Adsr[i-1]+maxA/A ;
        }

        else if(A<=i&&i<D)
        {
            if (i==A) Adsr[i]= maxA;
            else  Adsr[i]= Adsr[i-1]-(maxA-sustain)/(D-A);
        }

        else if(D<=i&&i<S)
                {
            Adsr[i]=sustain;
                }

        else if(S<=i&&i<sizeADSR-1)
        {
            if (i==S) Adsr[i]= sustain;
            if (i==sizeADSR-1) Adsr[i]=0;
            if (i<sizeADSR-1)  Adsr[i]= Adsr[i-1]-sustain/(sizeADSR-S);
        }
    }
    rflpc_timer_start(RFLPC_TIMER1);                                   //start TIMER1
    return 1;
}
```

f) D.c

```
/*
<generator>
        <handlers init="init_D" doGet="D1"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/

#include "sinus.h"
static char init_D(void){


    return 1;
}

static char D1(struct args_t *args1){

    rflpc_timer_stop(RFLPC_TIMER1);                                    //stop TIMER1
    D=39*(args1->state)/255+21;                                        //Changement de
    la valeur D

    for (i=0; i<=sizeADSR-1; i++)                                      //Calcul de
    l'envellope
    {
        if(i<A)
        {
            if (i==0) Adsr[i]= 0;
            else Adsr[i]= Adsr[i-1]+maxA/A ;
        }

        else if(A<=i&&i<D)
        {
            if (i==A) Adsr[i]= maxA;
            else  Adsr[i]= Adsr[i-1]-(maxA-sustain)/(D-A);
        }

        else if(D<=i&&i<S)
                {
            Adsr[i]=sustain;
                }

        else if(S<=i&&i<sizeADSR-1)
        {
            if (i==S) Adsr[i]= sustain;
            if (i==sizeADSR-1) Adsr[i]=0;
            if (i<sizeADSR-1)  Adsr[i]= Adsr[i-1]-sustain/(sizeADSR-S);
        }
    }
    rflpc_timer_start(RFLPC_TIMER1);                                   //start TIMER1
    return 1;
}
```

g) S.c

```
/*
<generator>
        <handlers init="init_S" doGet="SSS"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/

#include "sinus.h"
static char init_S(void){


    return 1;
}

static char SSS(struct args_t *args1){

    rflpc_timer_stop(RFLPC_TIMER1);                                    //stop TIMER1
    sustain=4*(args1->state);                                          //Changement de
    la valeur sustain

    for (i=0; i<=sizeADSR-1; i++)                                      //Calcul de
    l'envellope
    {
        if(i<A)
        {
            if (i==0) Adsr[i]= 0;
            else Adsr[i]= Adsr[i-1]+maxA/A ;
        }

        else if(A<=i&&i<D)
        {
            if (i==A) Adsr[i]= maxA;
            else  Adsr[i]= Adsr[i-1]-(maxA-sustain)/(D-A);
        }

        else if(D<=i&&i<S)
                {
            Adsr[i]=sustain;
                }

        else if(S<=i&&i<sizeADSR-1)
        {
            if (i==S) Adsr[i]= sustain;
            if (i==sizeADSR-1) Adsr[i]=0;
            if (i<sizeADSR-1)  Adsr[i]= Adsr[i-1]-sustain/(sizeADSR-S);
        }
    }
    rflpc_timer_start(RFLPC_TIMER1);                                   //start TIMER1
    return 1;
}
```

h) R.c

```
/*
<generator>
        <handlers init="init_R" doGet="RRR"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/

#include "sinus.h"
static char init_R(void){


    return 1;
}

static char RRR(struct args_t *args1){

    rflpc_timer_stop(RFLPC_TIMER1);                                    //stop TIMER1
    S=19*(args1->state)/255+100;                                       //Changement de
    la valeur S

    for (i=0; i<=sizeADSR-1; i++)                                      //Calcul de
    l'envellope
    {
        if(i<A)
        {
            if (i==0) Adsr[i]= 0;
            else Adsr[i]= Adsr[i-1]+maxA/A ;
        }

        else if(A<=i&&i<D)
        {
            if (i==A) Adsr[i]= maxA;
            else  Adsr[i]= Adsr[i-1]-(maxA-sustain)/(D-A);
        }

        else if(D<=i&&i<S)
                {
            Adsr[i]=sustain;
                }

        else if(S<=i&&i<sizeADSR-1)
        {
            if (i==S) Adsr[i]= sustain;
            if (i==sizeADSR-1) Adsr[i]=0;
            if (i<sizeADSR-1)  Adsr[i]= Adsr[i-1]-sustain/(sizeADSR-S);
        }
    }
    rflpc_timer_start(RFLPC_TIMER1);                                   //start TIMER1
    return 1;
}
```

i) PWM1.c

```
/*
<generator>
        <handlers init="init_PWM1" doGet="PWM1"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/
#include "sinus.h"

static char init_PWM1(void){

    rflpc_pwm_init (PWM_PIN3);                  //initialisation de PWM3
    rflpc_pwm_enable(PWM_PIN3);                 //PWM3 enable
    rflpc_pwm_single_edge (PWM_PIN3, 1000);     //PWM3 en single edge (non reversible)
    rflpc_pwm_set_period(20);//in µs            //période de découpage 2ms
    rflpc_pwm_start();                          //start PWM
    return 1;
}

static char PWM1(struct args_t *args1)
{
    rflpc_pwm_disable(PWM_PIN3);                //PWM3 disable
    frq_PWM=(args1->state)/40;                  //Changement de la valeur que doit coder la
    PWM (cf sinus.c-->inter_TIMER3_ADSR())
    rflpc_pwm_enable(PWM_PIN3);                 //PWM3 enable
    return 1;
}
```

j) PWM2.c

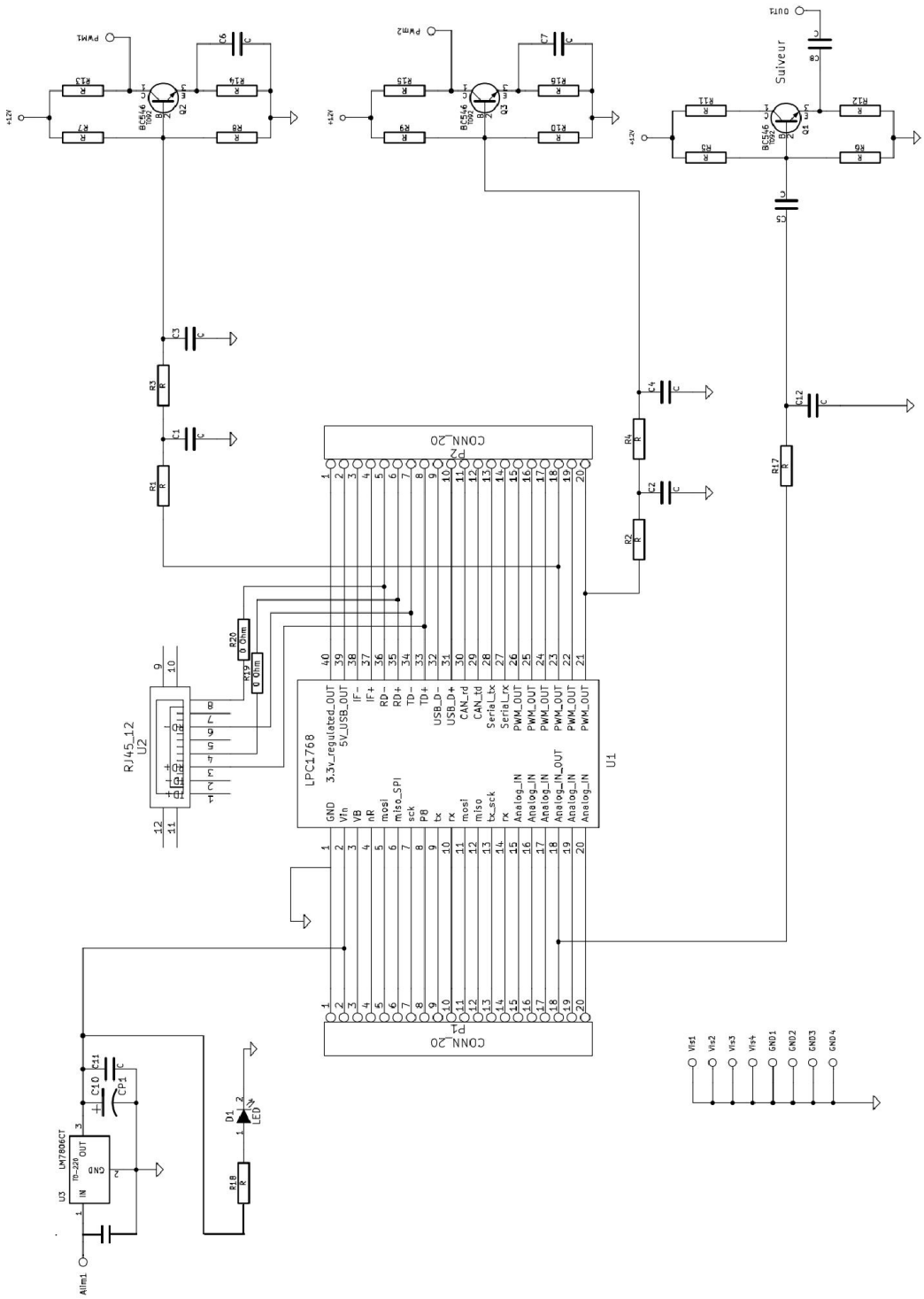
```
/*
<generator>
        <handlers init="init_PWM2" doGet="PWM2"/>
    <properties persistence="volatile" />
    <args>
            <arg name="state" type="uint16" />
    </args>
</generator>
*/
#include "sinus.h"

static char init_PWM2(void){
    rflpc_pwm_init (PWM_PIN4);                          //initialisation de PWM4
    rflpc_pwm_enable(PWM_PIN4);                         //PWM4 enable
    rflpc_pwm_single_edge (PWM_PIN4, 1000);             //PWM4 en single edge (non
    reversible)
    rflpc_pwm_set_period(2000); //in µs                 //période de découpage 2ms
    rflpc_pwm_start();                                  //start PWM
    return 1;
}

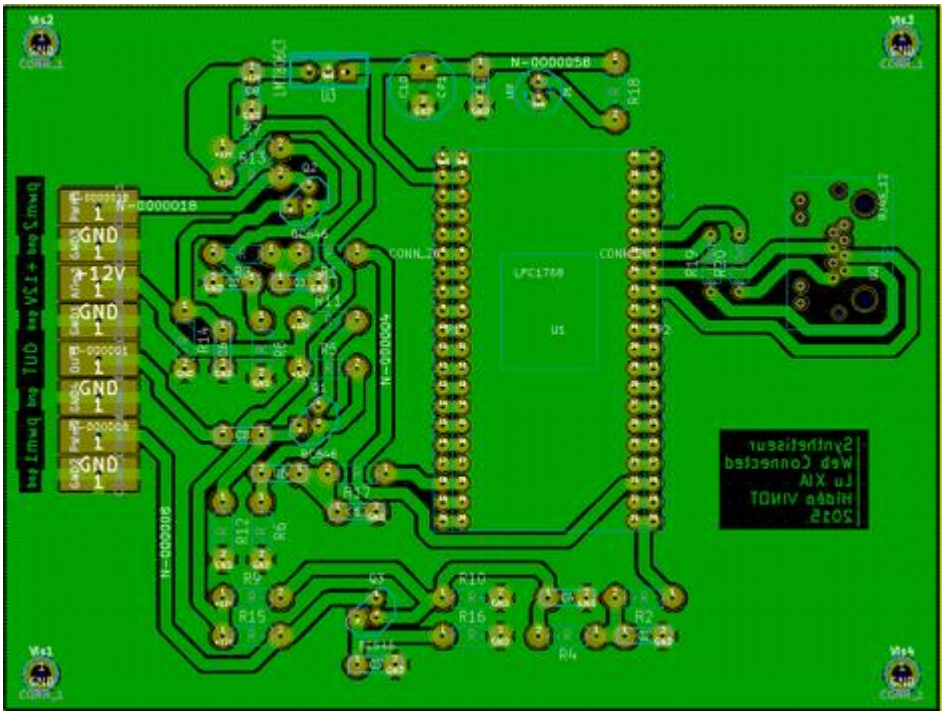
static char PWM2(struct args_t *args1)
{
    rflpc_pwm_disable(PWM_PIN4);                        //PMW4 disable
    rflpc_pwm_single_edge (PWM_PIN4, 8*(args1->state)); //Changement de la valeur que
    doit coder la PWM
    rflpc_pwm_enable(PWM_PIN4);                         //PMW4 enable
    return 1;
}
```

Annexe 2 :Carte électronique Support

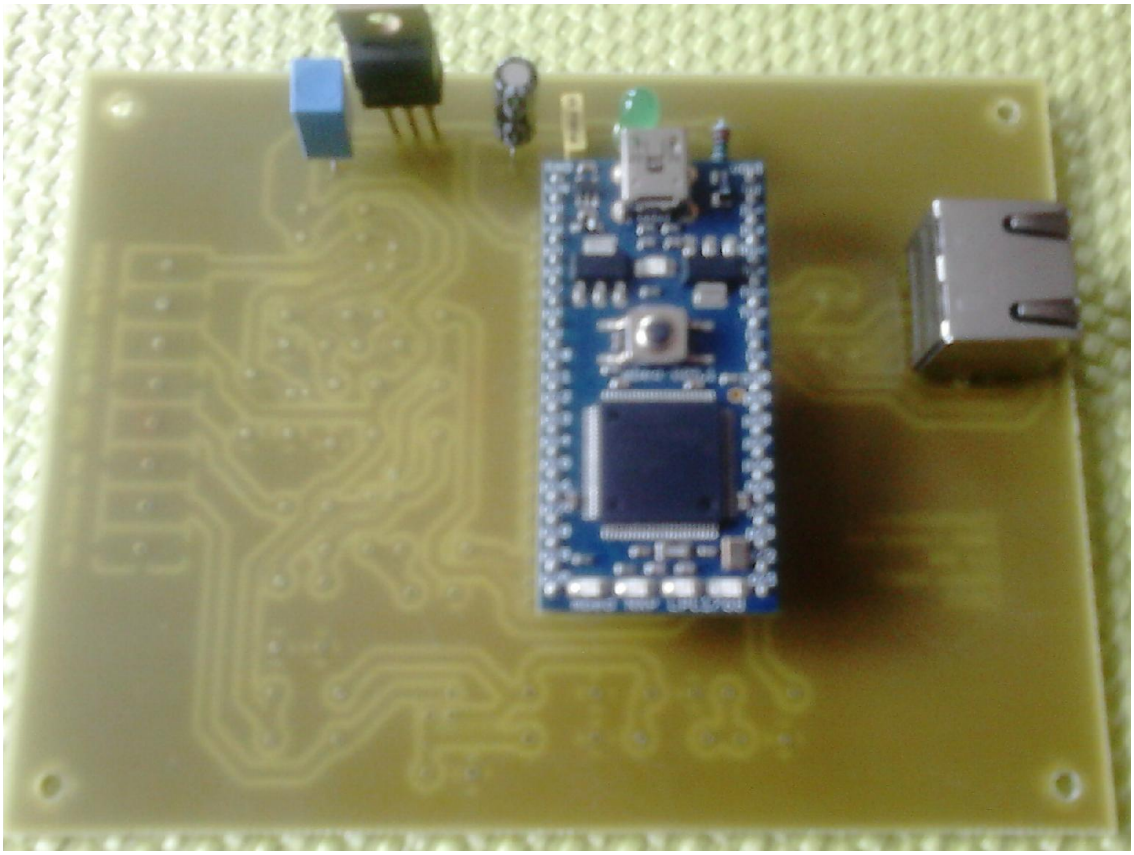
1 Schéma des cartes

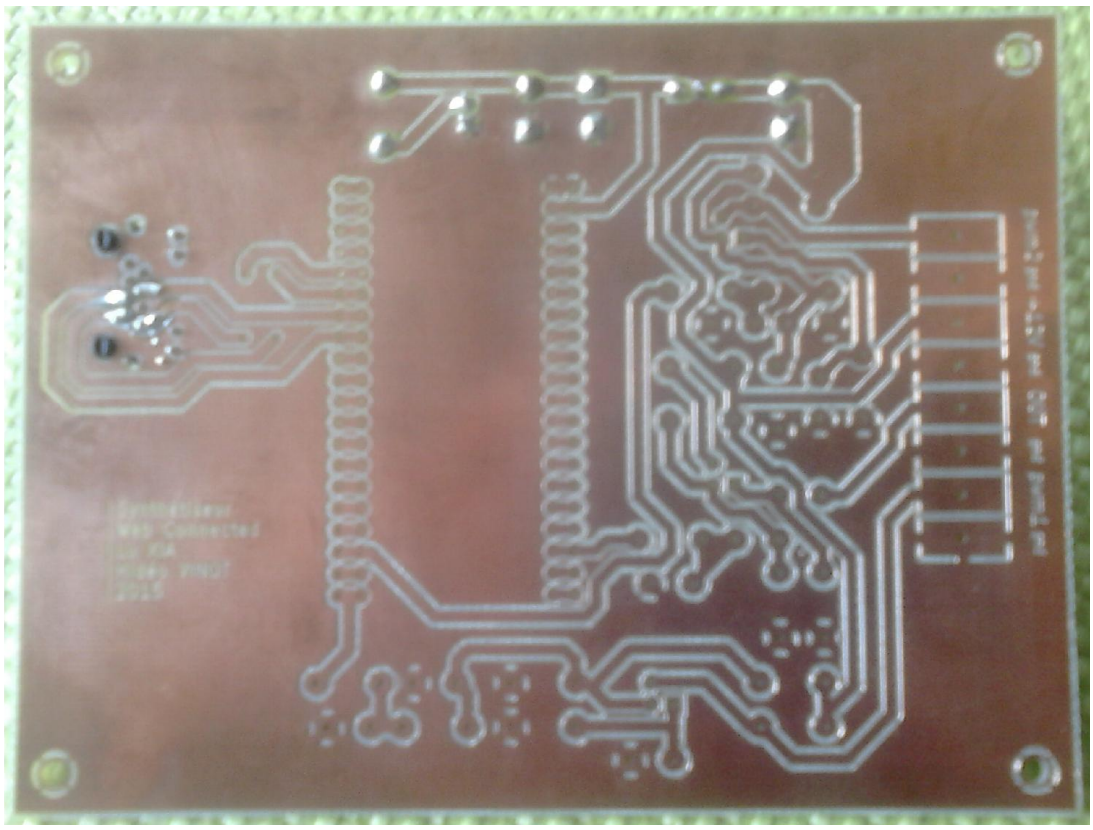


2 Typon échelle 1:1



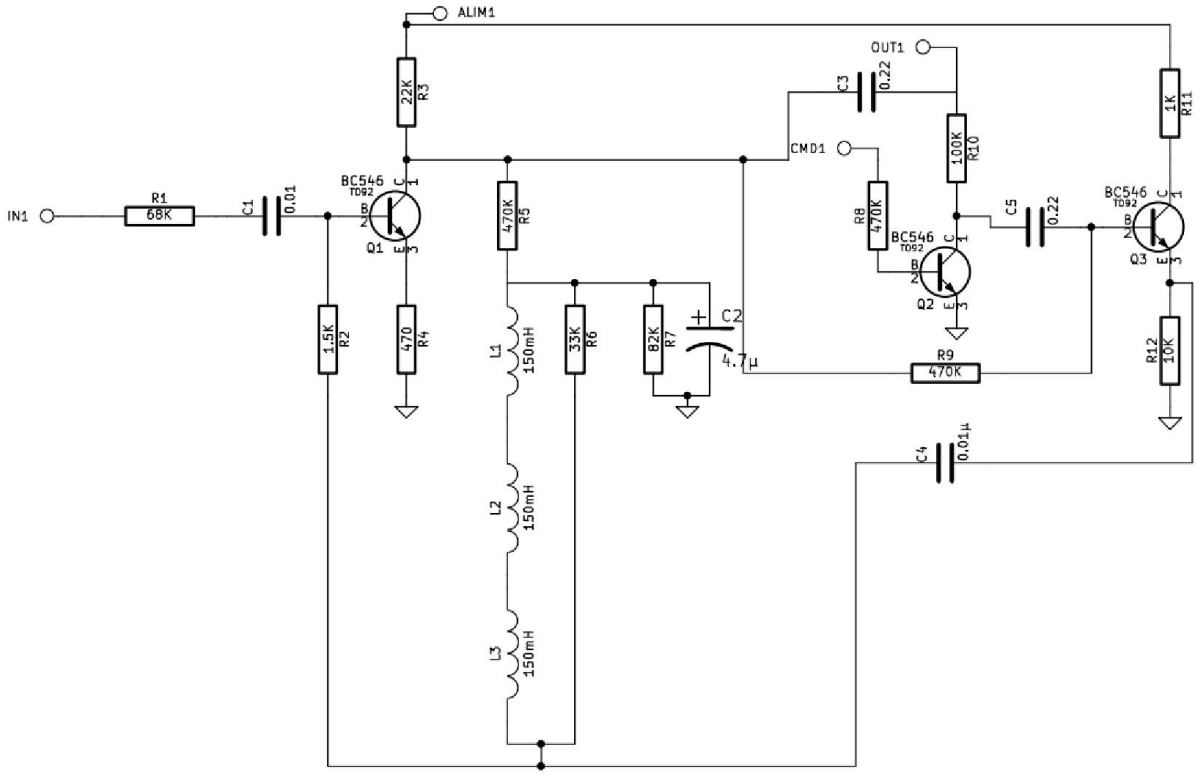
3 Photo de la carte (en cours de développement)



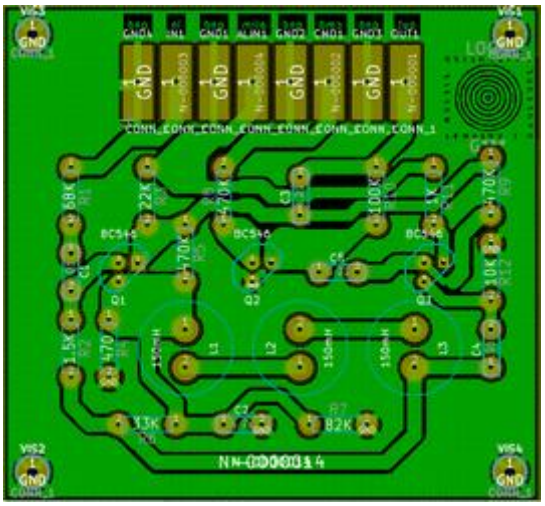


Annexe 3 : Carte électronique Voltage Controlled Filter

1 Schéma des cartes

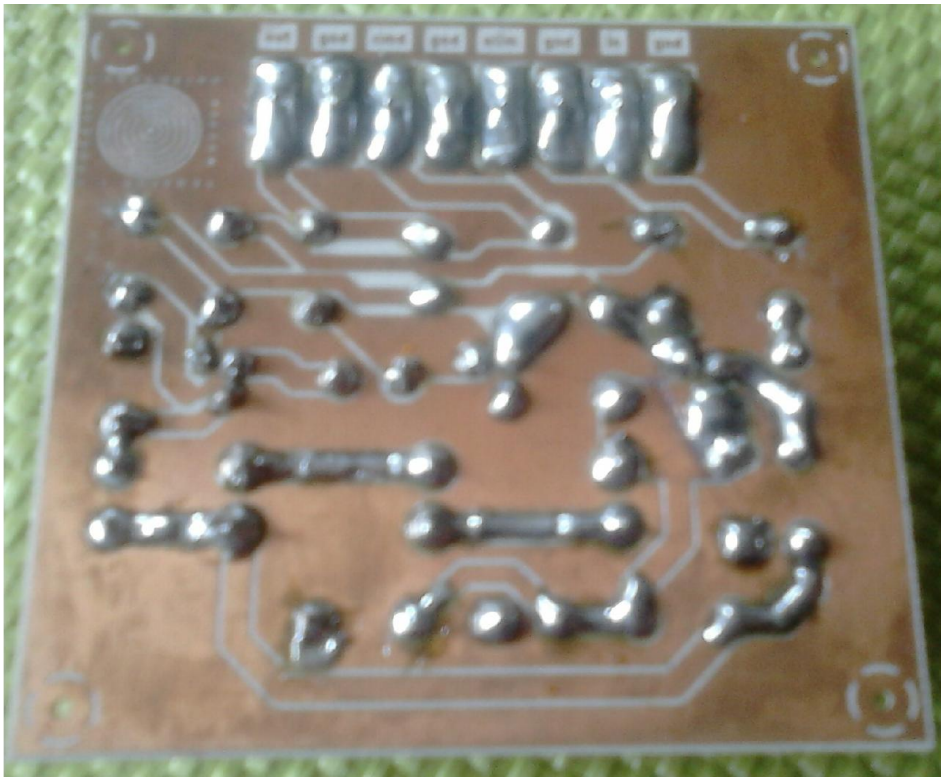


2 Typon échelle 1:1



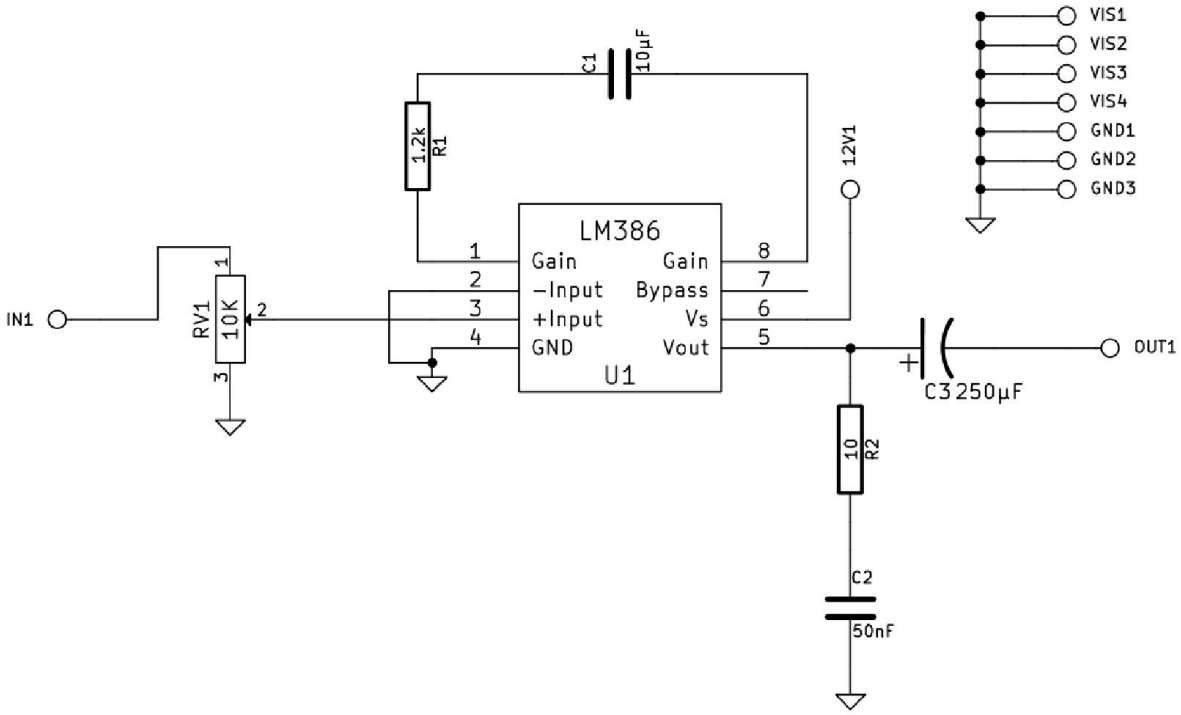
3 Photos de la carte



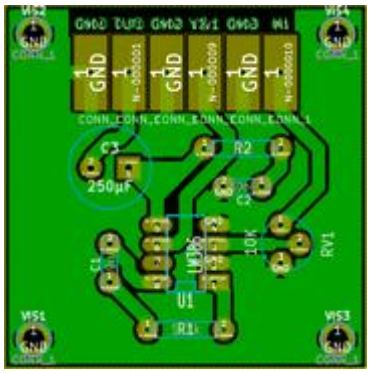


Annexe 4 :Carte électronique Ampli

1 Schéma des cartes



2 Typon échelle 1:1



3 Photos de la carte (en cours de développement)

