Groupe 1-3
Antoine Bouchut
Bastien La Torre
Mohammed El Azhar
Florence Daniel
Aglaé Bridonneau

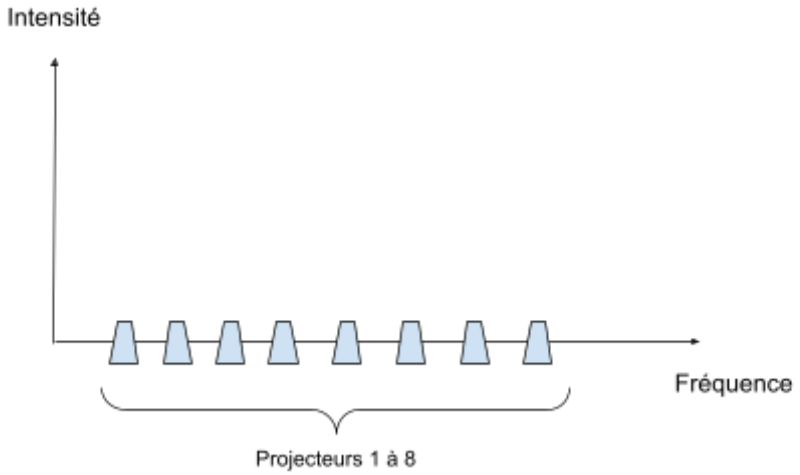
RAPPORT TECHNIQUE
Projet : Sonolux sur projecteurs

I - Introduction

Le but de notre projet est d'améliorer l'ambiance lumineuse dans les soirées à Supoptique.
Il faut que les lumières soient dynamiques afin de rendre la salle plus vivante.

Notre système de projecteurs doit pouvoir capter un signal sonore et afficher en temps réel le spectre de la musique qui passe en direct, et tout ça de manière autonome.
Pour cela, nous allons disposer de 8 projecteurs qui auront chacun une couleur attribuée. La valeur des fréquences ne sera pas représentée par des barres de niveau comme un sonolux classique mais par les intensités lumineuses. Plus l'amplitude des fréquences sera élevée et plus l'intensité lumineuse sera forte.

L'objectif est de répartir les différentes plages de fréquences sur chacun des projecteurs et de faire la moyenne de leurs amplitudes pour déterminer l'intensité à avoir.

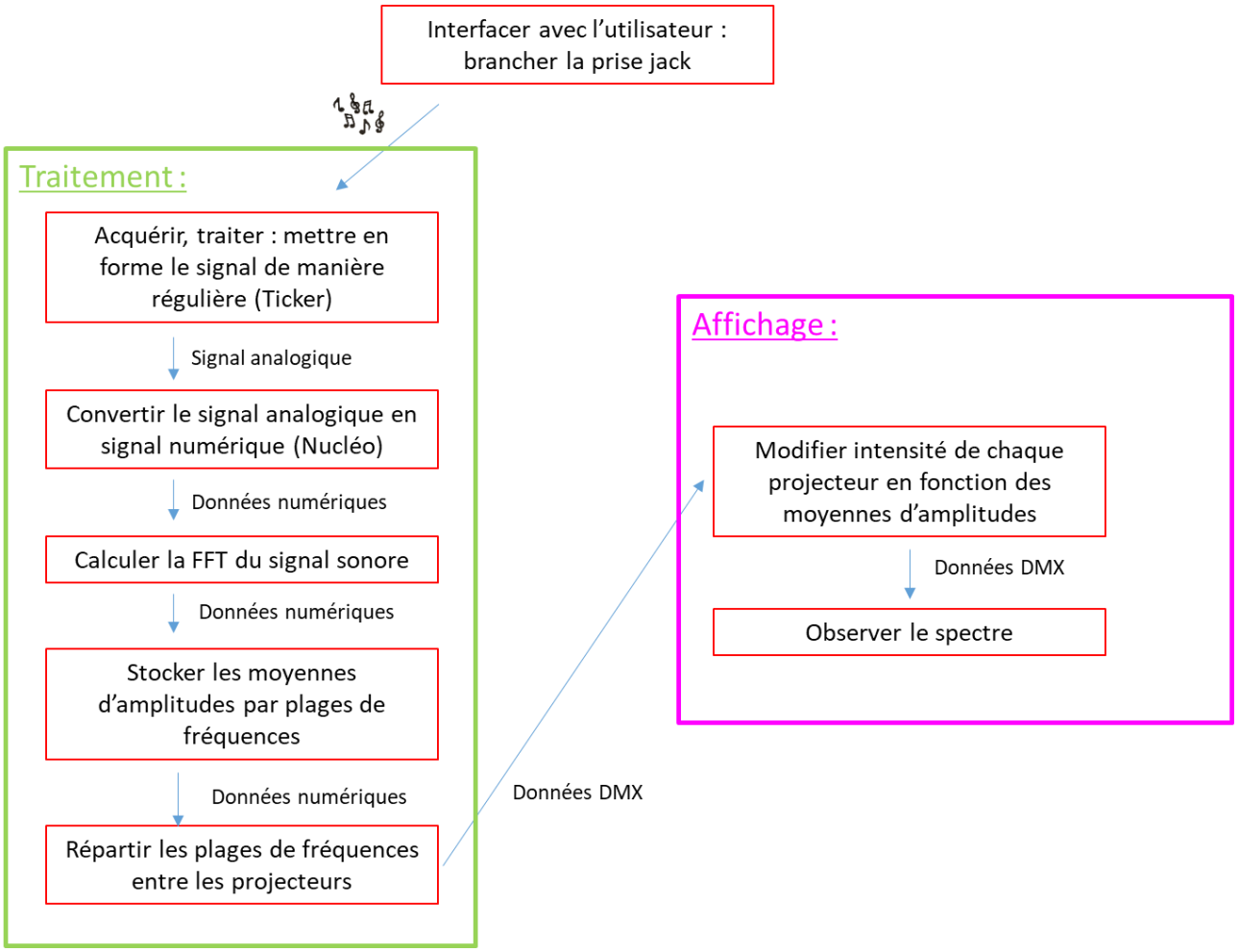


Pour analyser le son il faudra stocker les données de l'échantillonnage pour calculer la FFT et l'afficher grâce aux projecteurs.

L'échantillonnage, le calcul de la transformée de Fourier et les commandes des projecteurs se feront avec une carte nucléo.

Les projecteurs utilisés seront le jeu de lumière DMX Renkforce LVPT12 / RGBW – 15 W.

II - Description fonctionnelle



III - Réalisation du prototype

Le système est composé d'un montage électrique qui va capter le signal électrique du son envoyé et va le mettre en forme entre 0 et 3,3V. Ce signal est envoyé sur la carte Nucléo qui va faire les calculs demandés par le code pour finalement envoyer les commandes via DMX aux projecteurs afin de les mettre en marche.

Figure 1 : Projecteur DMX Renkforce LVPT12 / RGBW – 15 W

1 - Montage électrique pour le traitement du son

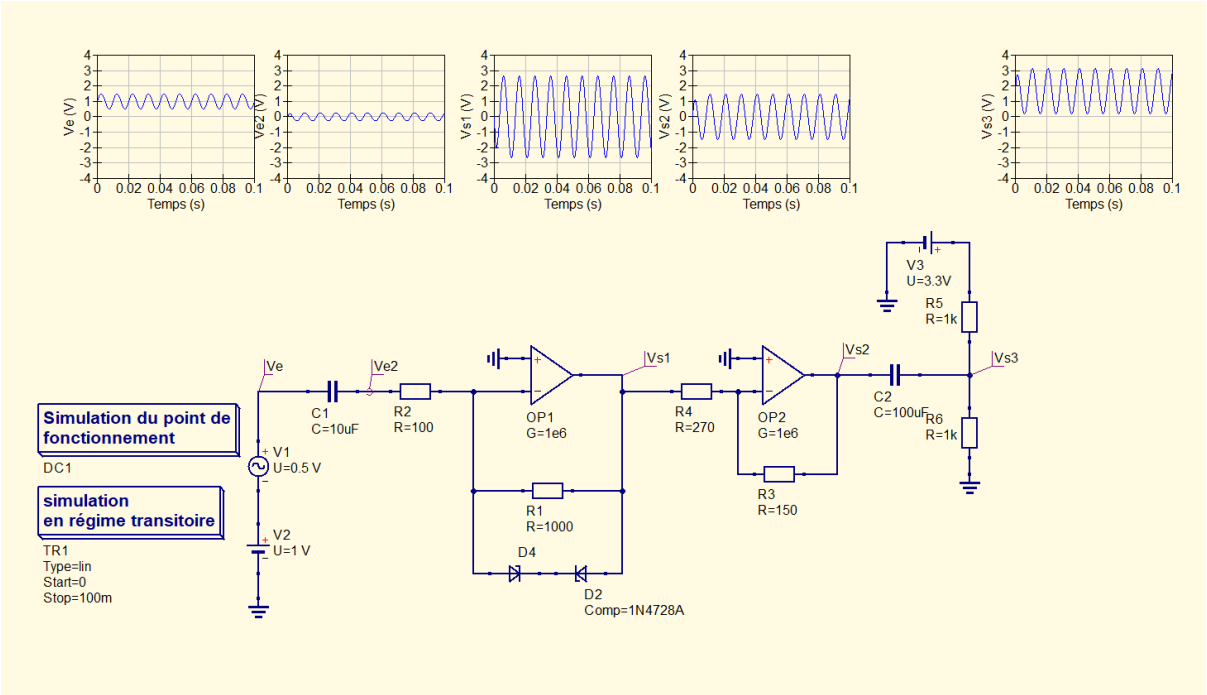


Figure 2 : Circuit modélisé sur l'interface QUCS

Dans ce circuit électronique, Ve correspond au signal électrique audio. En Ve2, le signal est passé par un filtre passe-haut pour éliminer toute éventuelle composante continue. Ensuite, le signal est écrêté par un premier amplificateur à

l'aide de deux diodes Zener 1N4728. Le signal est donc écrêté à 3,3V, pour obtenir Vs1. Le signal ne doit pas être trop écrêté pour garder l'aspect sinusoïdal le plus possible. Pour cela il faut judicieusement choisir R1 et R2. Puis, le signal passe dans un amplificateur inverseur et divise le signal Vs1, de manière optimale, par deux avec le rapport entre les résistances R3 et R4. On obtient le signal Vs2. Enfin, Vs2 est surélevé de 3,3V grâce à la carte Nucleo, par un bloc sommateur. On obtient donc le signal toujours sinusoïdal compris entre 0 et 3,3V, en Vs3, afin qu'il soit lisible pour la carte Nucleo. Vs3 est, enfin, envoyé à la carte Nucleo sur la pin A3.

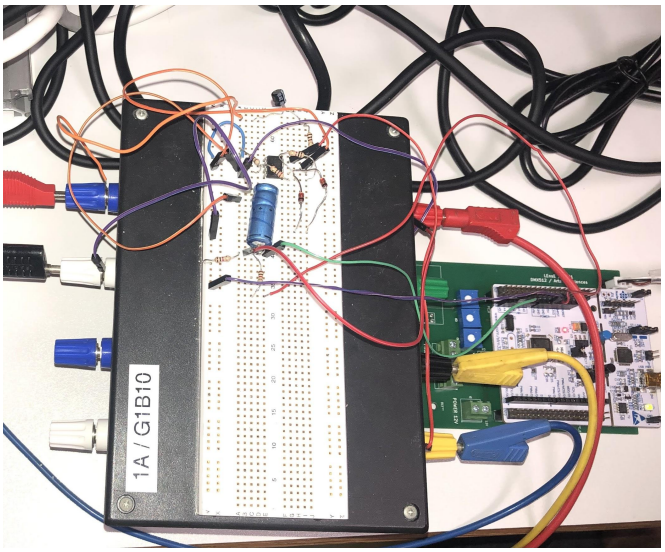


Figure 3 : Photo du montage réalisé

2 - Structure du code utilisé

1- Acquisition du signal sonore.
On utilise le montage précédent ainsi qu'un ticker pour relever les valeurs à intervalles réguliers.

2- Calcul de la transformée de Fourier.
Les valeurs des amplitudes sont stockées dans un tableau ''Output'' de 256 cases.

3- Calcul de la moyenne des amplitudes correspondant à une certaine plage de fréquences.
moyenne 1 : 0 à 500Hz
moyenne 2 : 500 à 1 kHz
moyenne 3 : 1 à 2 kHz
moyenne 4 : 2 à 3 kHz
moyenne 5 : 3 à 4 kHz
moyenne 6 : 4 à 6 kHz
moyenne 7 : 6 à 8 kHz
moyenne 8 : 8 à 12 kHz
On stocke ces valeurs d'amplitudes dans un liste

4- Normalisation des valeurs.
La valeur maximale du tableau ‘’Output’’ doit correspondre à la valeur 255 (max de l’intensité). On prend des valeurs entières.

5- Attribution des couleurs et des plages de fréquence pour chaque projecteur.
COULEURS :
proj 1 : rouge (rouge 255, vert 0, bleu 0)
proj 2 : orange (R 255, V 100, B 0)
proj 3 : jaune (R 100, V100, B 0)
proj 4 : vert (R 0, V 255, B 0)
proj 5 : cyan/bleu clair (R 0, V 255, B 100)
proj 6 : bleu (R 0, V 0, B 255)
proj 7 : violet (R 255, V 0, B 255)
proj 8 : blanc (R 255, V 255, B 255)

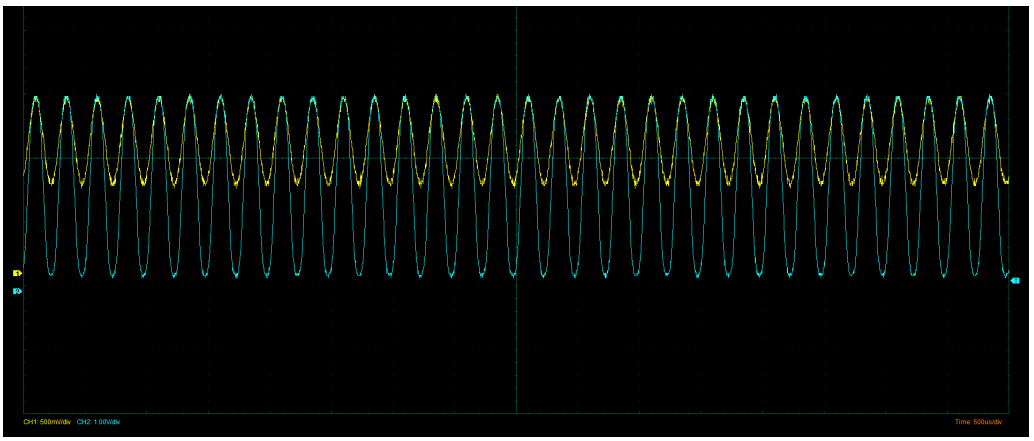
FRÉQUENCES :
proj 1 : 0 à 500Hz
proj 2 : 500 à 1 kHz
proj 3 : 1 à 2 kHz
proj 4 : 2 à 3 kHz
proj 5 : 3 à 4 kHz
proj 6 : 4 à 6 kHz
proj 7 : 6 à 8 kHz
proj 8 : 8 à 12 kHz

L’intensité lumineuse de chaque projecteur correspond à la moyenne calculée juste avant.
Seuil mis en place : si la moyenne calculée et normalisée est inférieure à 20, le projecteur ne s’allume pas. Cela permet d'enlever le bruit résiduel pris en compte dans le calcul de la TF.

IV - Tests et validation

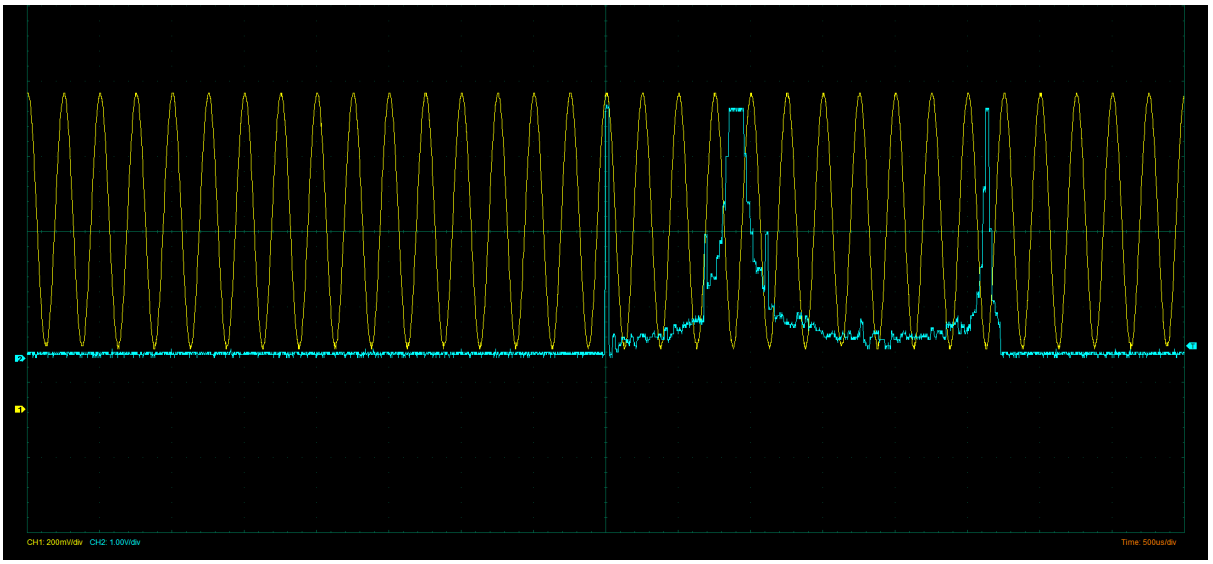
1 - Test du montage électrique pour le traitement du son

Vérification d’ajustement et d’adaptation des niveaux de tension de la sortie sonore d’un ordinateur à l’entrée analogique pour l’acquisition, à fréquence constante, par une carte Nucléo.
Sur l'oscilloscope, image des tensions que l’on obtient en sortie :



<u>Figure 4 :</u> Sinusoïde de sortie (sortie Vs3, cf figure 2) entre 0 et 3V (entrée en bleu, sortie en jaune)

On visualise ensuite sur l'oscilloscope la tf du signal en sortie :



<u>Figure 5 :</u> Courbe de la TF (en bleu) obtenue sur l'oscilloscope (sortie A2 de la carte Nucléo)

<u>2 - Test du seuil</u>

On regarde avec TeraTerm la valeur de l'intensité lorsqu'un projecteur clignote alors qu'il devrait être éteint : on observe des valeurs entre 0 et 20 environ. On choisit donc un seuil de 20.
Après création du seuil, les projecteurs concernés ne s'allument plus lorsque leur intensité est trop faible.
Il reste quand même quelques clignotements sur les basses fréquences, peut-être à cause des plages de fréquences plus rapprochées. Mais on ne peut pas encore élever le seuil car sinon dans les hautes fréquences (où il y a des intensités plus faibles), les projecteurs ne pourraient pas s'allumer.

3 - Test du système complet

Pour vérifier que le système entier fonctionne bien, on fait défiler les fréquences petit à petit des basses fréquences vers les hautes fréquences.
Les projecteurs s'allument et s'éteignent chacun leur tour et leur intensité varie de manière sinusoïdale.
On a d'abord utilisé le générateur de tension et sa fonction balayage (sweep). Puis pour tester avec du son, on a utilisé une vidéo YouTube qui faisait défiler toutes les fréquences.
Les projecteurs s'allument bien au moment où du son est émis dans leur plage de fréquence.

Cependant, on observe quelques défauts : certains projecteurs sont allumés en même temps alors qu'un seul devrait être allumé et tous les autres éteints. L'intensité du projecteur qui devrait être éteint reste tout de même plus faible que celle de celui allumé dans sa plage de fréquences.

On envoie ensuite une musique pour vérifier le bon fonctionnement du prototype. On observe bien les projecteurs s'allumer et s'éteindre au cours de la musique avec les bonnes couleurs. Les éclairages semblent correspondre aux fréquences perçues par nos oreilles (aigu et grave).

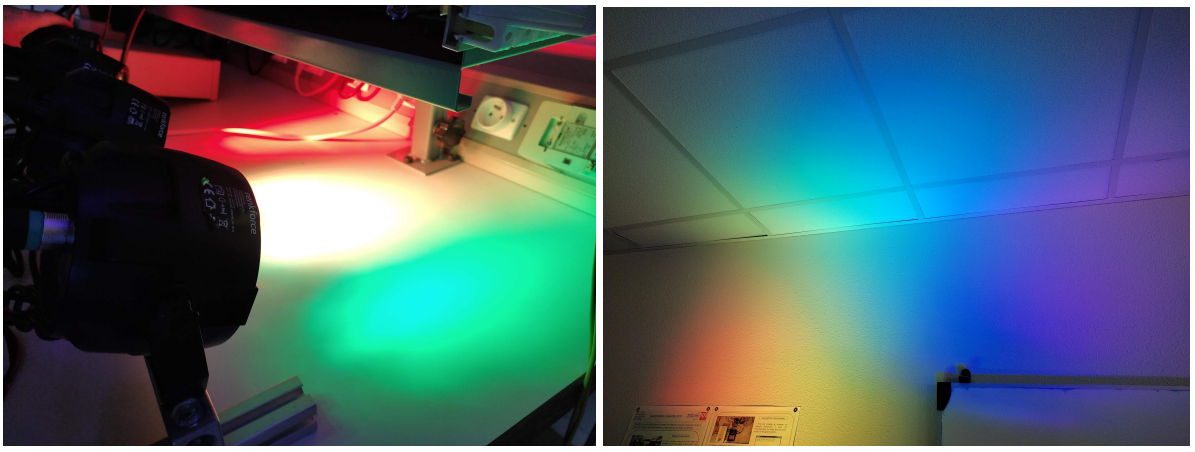


Figure 6 : Résultat des projecteurs mis en marche

V - Étapes de réalisation et choix

1 - Planning

Séance 1 :
Choix du thème du projet et réalisation de la fiche descriptive

Séance 2 :
Découverte du fonctionnement des projecteurs Renkforce LVPT12 / RGBW - 15W, notamment comment les commander et quelles adresses correspondent à l'intensité, aux couleurs rouge, bleue et verte.
Rendu des livrables intermédiaires avec le cahier des charges, le schéma des différentes fonctions techniques réalisées par le prototype, le plan de formation incluant les compétences visées.
Construction du montage électrique pour le traitement du son, c'est-à-dire transmettre le signal de l'ordinateur à la carte Nucléo.

Séance 3 :
Appropriation du code de la TF pour en extraire les données nécessaires à la découpe en plage de fréquences ;
Réalisation du programme qui calcule la moyenne du module de la TF sur chaque plage de fréquence ;
Assemblage des 3 codes différents (FFT, calcul de moyenne et commande des projecteurs) ;
Visualisation de la TF sur l'oscilloscope pour s'assurer du fonctionnement du code ;
Premier test avec 4 projecteurs.

Séance 4 :
Perfectionnement des composants du montage pour obtenir en sortie un signal le plus sinusoïdal possible, compris strictement entre 0 et 3,3V ;
Premier test avec 8 projecteurs au lieu de 4 ;
Choix de la répartition des couleurs sur les différents projecteurs ;
Nouvelle répartition des fréquences pour chaque projecteur ;

Séance 5 :
Mise en place d'un seuil pour éviter les clignotements parasites et pour ne pas que les projecteurs s'allument à cause du bruit dans le calcul de la TF ;
Début de la rédaction du rapport technique.

Séance 6 :
Vérification du bon fonctionnement du système complet ;
Suite de la rédaction du rapport technique ;
Début de l'élaboration du poster.

Séance 7:
Suite de l'élaboration du poster.

2 - Difficultés rencontrées

L'une des principales difficultés a été de commencer le projet. Nous avons mis du temps à nous décider sur ce que nous voulions faire au cours de ces séances. Définir des objectifs clairs n'a pas été immédiat, nous étions assez indécis. Malgré une mise en route un peu longue, une fois lancés, nous avons quand même été efficaces et impliqués dans l'avancée du projet.

Lors de la création et l'assemblage de tous les codes, il fallait être très attentif pour que cela fonctionne. Par exemple, les projecteurs ne fonctionnaient pas de la même manière que quand on testait les codes séparément. Pour débugger le clignotement du projecteur, il fallait mettre l'entrée du son à analyser sur l'entrée A3 du Nucléo, l'entrée A0 et A1 étant déjà prises par les DMX. Il fallait donc faire attention à ce qu'il n'y ait pas de chevauchement entre les différentes entrées et sorties.

Le choix des composants n'a pas été facile pour obtenir en sortie un signal sinusoïdal. Il a fallu faire de nombreux tests sur le montage réel et sur le logiciel QUCS pour avoir un résultat satisfaisant.

L'un des problèmes également rencontré est au niveau des adresses des projecteurs. Il a fallu quelque temps au début du projet pour comprendre le fonctionnement de celles-ci et comment les utiliser pour contrôler la lumière émise des projecteurs (intensité et couleur).

3 - Analyse du travail d'équipe

Le travail a été partagé pour avancer plus efficacement. Bastien, Antoine et Mohammed ont travaillé sur le montage électrique et sur la partie acquisition du son. Florence et Aglaé ont travaillé sur la partie codage et mise en forme des données pour l'affichage sur les projecteurs. La communication entre les deux équipes s'est bien faite, il n'y a pas eu de malentendus sur les objectifs. Nous utilisions un dossier partagé où nous mettions à chaque fin de séance le code final qui fonctionnait pour ne rien perdre. Nous notions également sur un document partagé les avancées du jour, les problèmes rencontrés et les objectifs de la prochaine séance afin de savoir où nous nous situions dans le déroulement du projet.

Annexe :

CODE UTILISÉ :

```
#include "mbed.h"
#include "arm_math.h"
#include "dsp.h"
#include "arm_common_tables.h"
#include "arm_const_structs.h"

// entete code proj
#include "mbed.h"

#define SAMPLES                 512
/* 256 real party and 256 imaginary parts */
#define FFT_SIZE                SAMPLES / 2
/* FFT size is always the same size as we have samples, so 256 in our case */

const         uint8_t       scriabin_r[12] = {255, 140, 255, 255, 0, 255, 68, 255, 170, 0,
255, 0};
const       uint8_t     scriabin_g[12] = {0, 0, 255, 0, 255, 0, 0, 162, 0, 255, 0, 0};
const         uint8_t      scriabin_b[12] = {0, 255, 0, 255, 170, 85, 255, 0, 255, 0, 170,
255};

#define     MIDI_NOTE_ON        0x90
#define     MIDI_NOTE_OFF       0x80
#define     MIDI_CC             0xB0

Serial      debug_pc(USBTX, USBRX);
InterruptIn     my_bp(USER_BUTTON);

Serial      dmx(A0, A1);
DigitalOut  out_tx(D5);
DigitalOut  start(D4);  //envoie des données
DigitalOut  enableDMX(D6);

AnalogIn    variationR(PC_0);
AnalogIn    variationG(PC_2);
AnalogIn    variationB(PC_3);

Serial      midi(D8, D2);

// DMX
```

```
char dmx_data[SAMPLES] = {0};
char nb = 0;

void initDMX();
void updateDMX();

// MIDI
char       cpt_midi;
char       new_data_midi, new_note_midi;
char       midi_data[3], channel_data, note_data, velocity_data;
char       control_ch, control_value;

void initMIDI(void);
void ISR_midi_in(void);
bool isNoteMIDIdetected(void);

// fin entête code proj

double L[8] = {0};   // liste dans laquelle sera stocké les valeurs d'amplitudes moyennes par plage de fréquence

float32_t Input[SAMPLES];
float32_t Output[FFT_SIZE];
bool      trig=0;         // sémaphore de blocage de l'échantillonnage
int       indice = 0;

DigitalOut myled(LED1);
AnalogIn   myADC(A3);
AnalogOut  myDAC(A2);
Serial     pc(USBTX, USBRX);
Ticker     timer;

void sample(){
   myled = 1;
   if(indice < SAMPLES){
      Input[indice] = myADC.read() - 0.5f;
      //Real part NB removing DC offset
      Input[indice + 1] = 0;
      //Imaginary Part set to zero
      indice += 2;
   }
   else{ trig = 0; }
   myled = 0;
}
```

```
int main() {
    float maxValue;           // Max FFT value is stored here
    uint32_t maxIndex;        // Index in Output array where max value is

    debug_pc.baud(115200);
    debug_pc.printf("Essai DMX512\r\n");

    initDMX();
    initMIDI();

    while(1) {
        debug_pc.printf("OK\r\n");

        if(trig == 0){
            timer.detach();
            // Initialisation du calcul de la FFT
              // NB utilisation de fonctions prédéfinies arm_cfft_sR_f32_lenXXX, où XXX
est le nombre d'échantillons, ici 256
            arm_cfft_f32(&arm_cfft_sR_f32_len256, Input, 0, 1);
            // Calcul de la FFT et stockage des valeurs dans le tableau Output
            arm_cmplx_mag_f32(Input, Output, FFT_SIZE);
            Output[0] = 0;
            // Calcul la valeur maximale du spectre - maxValue - et son indice - maxIndex
            arm_max_f32(Output, FFT_SIZE/2, &maxValue, &maxIndex);
            // Affichage de la valeur max et de son indice
            // pc.printf("MAX = %lf, %d \r\n", maxValue, maxIndex);

            // Affichage du spectre sous forme d'un signal analogique entre 0 et 3V.
            myDAC=1.0f;     // Impulsion de synchronisation
            wait_us(20);
            myDAC=0.0f;
            // Affichage des différentes valeurs du spectre
            for(int i=0; i < FFT_SIZE / 2; i++){
                myDAC=(Output[i])* 0.9;   // Mise à l'échelle de 90% de 3.3V
                wait_us(10);              // Durée de chaque palier
            }
            myDAC=0.0f;




            // code pour moyenner les amplitudes
```

```
        L[0] = 0;
        L[1] = 0;
        L[2] = 0;
        L[3] = 0;
        L[4] = 0;
        L[5] = 0;
        L[6] = 0;
        L[7] = 0;

int i;
int S = 0;

for (i=0;i<5;i=i+1) // moyenne pour les fréquences de 0 à 500 Hz
{
   S=S+Output[i];
}
L[0]=S/5.0;
//debug_pc.printf("L0=%lf\r\n", L[0]);

S = 0;
for (i=5;i<10;i=i+1) // moyenne pour les fréquences de 500 Hz à 1 kHz
{
   S=S+Output[i];
}
L[1]=S/5.0;
//debug_pc.printf("L1=%lf\r\n", L[1]);

S = 0;
for (i=10;i<20;i=i+1) // moyenne pour les fréquences de 1kHz à 2kHz
{
   S=S+Output[i];
}
L[2]=S/10.0;
//debug_pc.printf("L2=%lf\r\n", L[2]);

S = 0;
for (i=20;i<30;i=i+1) // moyenne pour les fréquences de 2kHz à 3kHz
{
   S=S+Output[i];
}
L[3]=S/10.0;
//debug_pc.printf("L3=%lf\r\n", L[3]);
```

```
   S = 0;
   for (i=30;i<40;i=i+1) // moyenne pour les fréquences de 3kHz à 4kHz
   {
      S=S+Output[i];
   }
   L[4]=S/10.0;
   //debug_pc.printf("L3=%lf\r\n", L[4]);

   S = 0;
   for (i=40;i<60;i=i+1) // moyenne pour les fréquences de 4kHz à 6kHz
   {
      S=S+Output[i];
   }
   L[5]=S/20.0;
   //debug_pc.printf("L3=%lf\r\n", L[5]);

   S = 0;
   for (i=60;i<80;i=i+1) // moyenne pour les fréquences de 6kHz à 8kHz
   {
      S=S+Output[i];
   }
   L[6]=S/20.0;
   //debug_pc.printf("L3=%lf\r\n", L[6]);

   S = 0;
   for (i=80;i<120;i=i+1) // moyenne pour les fréquences de 8kHz à 12kHz
   {
      S=S+Output[i];
   }
   L[7]=S/40.0;
   //debug_pc.printf("L3=%lf\r\n", L[7]);



    for (i=0;i<8;i=i+1) // normalisation des valeurs pour que le max soit à 255 pour les
projecteurs
   {
      L[i]=floor((L[i]*255)/maxValue)*2;
   }


   debug_pc.printf("L0=%lf\r\n", L[0]);
   debug_pc.printf("L1=%lf\r\n", L[1]);
```

```
   debug_pc.printf("L2=%lf\r\n", L[2]);
   debug_pc.printf("L3=%lf\r\n", L[3]);
   debug_pc.printf("L4=%lf\r\n", L[4]);
   debug_pc.printf("L5=%lf\r\n", L[5]);
   debug_pc.printf("L6=%lf\r\n", L[6]);
   debug_pc.printf("L7=%lf\r\n", L[7]);

         myDAC=1.0f;     // Impulsion de synchronisation
         wait_us(20);
         myDAC=0.0f;
         // Affichage des différentes valeurs du spectre
         for(i=0; i < 4; i++){
            myDAC=(L[i])* 0.9;   // Mise à l'échelle de 90% de 3.3V
            wait_us(20);              // Durée de chaque palier
         }
         myDAC=0.0f;



   // code pour diriger les projecteurs


    //if(isNoteMIDIdetected()){
        //           debug_pc.printf("C=%d,N=%d,V=%d\r\n",  channel_data,  note_data,
velocity_data);
    //    char note = note_data%12;
    //    debug_pc.printf("N=%d\r\n", note);
    //       Renkforce LPT12 - AD 1

            dmx_data[0] = 0;  //
            if (L[0]>20){dmx_data[3] = L[0];} // INTENSITE
            else {dmx_data[3] = 0;}
            dmx_data[4] = 255; // ROUGE
            dmx_data[5] = 0; // VERT
            dmx_data[6] = 0; // BLEU

            dmx_data[8] = 0;  //
            if (L[1]>20){dmx_data[11] = L[1];} // INTENSITE
            else {dmx_data[11] = 0;}
            dmx_data[12] = 255; // ROUGE
            dmx_data[13] = 100; // VERT
            dmx_data[14] = 0; // BLEU

            dmx_data[16] = 0;  //
```

```
if (L[2]>20){dmx_data[19] = L[2];} // INTENSITE
else {dmx_data[19] = 0;}
dmx_data[20] = 100; // ROUGE
dmx_data[21] = 100; // VERT
dmx_data[22] = 0; // BLEU

dmx_data[24] = 0;  //
if (L[3]>20){dmx_data[27] = L[3];} // INTENSITE
else {dmx_data[27] = 0;}
dmx_data[28] = 0; // ROUGE
dmx_data[29] = 255; // VERT
dmx_data[30] = 0; // BLEU
new_note_midi = 0;



dmx_data[32] = 0;  //
if (L[4]>20){dmx_data[35] = L[4];} // INTENSITE
else {dmx_data[35] = 0;}
dmx_data[36] = 0; // ROUGE
dmx_data[37] = 255; // VERT
dmx_data[38] = 200; // BLEU

dmx_data[40] = 0;  //
if (L[5]>20){dmx_data[43] = L[5];} // INTENSITE
else {dmx_data[43] = 0;}
dmx_data[44] = 0; // ROUGE
dmx_data[45] = 0; // VERT
dmx_data[46] = 255; // BLEU

dmx_data[48] = 0;  //
if (L[6]>20){dmx_data[51] = L[6];} // INTENSITE
else {dmx_data[51] = 0;}
dmx_data[52] = 255; // ROUGE
dmx_data[53] = 0; // VERT
dmx_data[54] = 255; // BLEU

dmx_data[56] = 0;  //
if (L[7]>20){dmx_data[59] = L[7];} // INTENSITE
else {dmx_data[59] = 0;}
dmx_data[60] = 255; // ROUGE
dmx_data[61] = 255; // VERT
dmx_data[62] = 255; // BLEU
new_note_midi = 0;
```

```
        updateDMX();


            // Réinitialisation du sémaphore
            trig = 1;
            indice = 0;
            timer.attach_us(&sample,40);      //40us 25KHz sampling rate
        }
    }
}


/* Fonction d'initialisation de la liaison DMX */
void initDMX(){
    // Initialisation DMX
    dmx.baud(250000);
    dmx.format (8, SerialBase::None, 2);
    enableDMX = 0;
    // Initialisation canaux DMX
    for(int k = 0; k < SAMPLES; k++){
        dmx_data[k] = 0;
    }
    updateDMX();
}

/* Fonction de mise à jour de la liaison DMX */
void updateDMX(){
        enableDMX = 1;
        start = 1;      // /start
        out_tx = 0;     // break
        wait_us(88);
        out_tx = 1;     // mb
        wait_us(8);
        out_tx = 0;     // break
        start = 0;
        dmx.putc(0);    // Start
        for(int i = 0; i < SAMPLES; i++){
            dmx.putc(dmx_data[i]);  // data
        }
        wait_us(23000); // time between frame
}

/* Fonction d'initialisation de la liaison MIDI */
```

```
void initMIDI(void){
    midi.baud(31250);
    midi.format(8, SerialBase::None, 1);
    midi.attach(&ISR_midi_in, Serial::RxIrq);
}

/* Detection d'une note reçue en MIDI */
bool isNoteMIDIdetected(void){
    if(new_note_midi == 1)
        return true;
    else
        return false;
}
/* Fonction d'interruption sur MIDI */
void ISR_midi_in(void){
    char data = midi.getc();
    if(data >= 128)
        cpt_midi = 0;
    else
        cpt_midi++;
    midi_data[cpt_midi] = data;
    if(cpt_midi == 2){
        cpt_midi = 0;
            if(((midi_data[0] & 0xF0) == MIDI_NOTE_ON) || ((midi_data[0] & 0xF0) ==
MIDI_NOTE_OFF)){
            new_note_midi = 1;
            channel_data = midi_data[0] & 0x0F;
            note_data = midi_data[1];
            velocity_data = midi_data[2];
        }
        else{
            if(midi_data[0] == MIDI_CC){
                new_data_midi = 1;
                control_ch = midi_data[1];
                control_value = midi_data[2];
            }
        }
    }
}
```