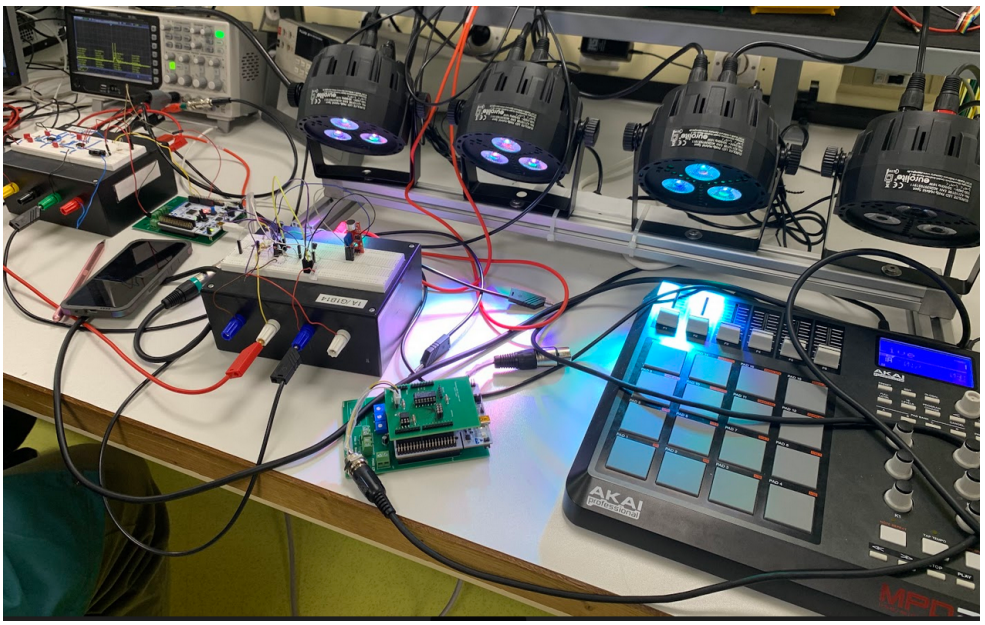
Rapport technique

Yihao Yuan/ Chatony Malcolm/
Li Yi/ Mingrui Ma/ Robin Fournel

1. Pourquoi réaliser le BeatBox ?

Lors des soirées Supop, de nombreuses associations à caractère événementiel ont besoin de pouvoir gérer la lumière et le son de manière simultanée. Souvent, l'intermédiaire d'un ordinateur est nécessaire. Nous voulons simplifier la tâche des organisateurs en remplaçant l'ordinateur par un pad de commande. Les pads de commande sont plus intuitifs et faciles d'utilisation lors de la soirée. De plus, ces derniers évitent notamment les problèmes liés à l'exposition de l'ordinateur face aux boissons. Enfin, codé en amont, le pad évite d'avoir à jongler le jour J entre les différentes fonctions d'allumage. Accompagné de contrôles sur les curseurs, il sera donc plus facile et agréable pour l'utilisateur de réaliser les effets artistiques souhaités.

Pour cela nous allons distinguer deux modes de fonctionnement principaux. L'un concernant l'éclairage de lyres en fonction d'un son externe joué aux alentours du pad. Cela correspond au mode avec le son. L'autre étant l'allumage de lyres en fonction de séquences d'allumage sur chaque touche du pad avec des contrôles supplémentaires avec les curseurs.



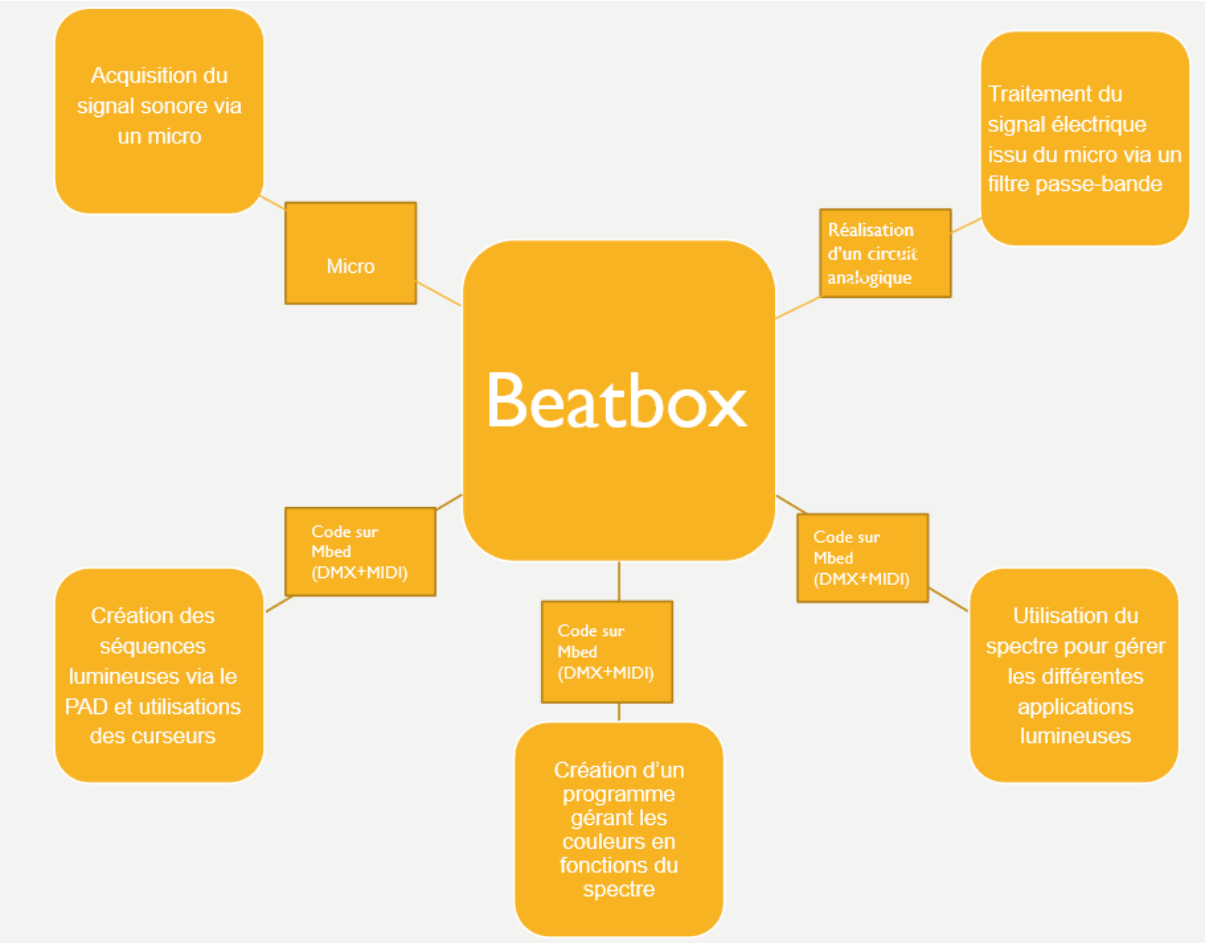
2. Découpage fonctionnel

Un des enjeux de ce projet est d'arriver à gérer l'acquisition et le traitement d'un signal sonore environnant. Un autre enjeu est d'appréhender le fonctionnement de la communication MIDI (pour le pad) et DMX (pour la lyre). Enfin, il faut être capable de concevoir des séquences d'allumage qui correspondent à des effets artistiques.

L'objectif de cette partie est de comprendre la fonctionnalité des 5 sous-parties du système. Ces dernières sont reliées à deux grandes familles : acquisition/traitement du signal et effets stylistes lumineux. Finalement, voici l'organisation choisi reposant sur le fonctionnement logique du système :

I. Acquisition/traitement du signal
   a. Schéma fonctionnel
   b. Acquisition du signal sonore via un micro
   c. Traitement du signal électrique issu du micro via un filtre passe-bande
   d. Utilisation du spectre pour gérer les différentes applications lumineuses

II. Effets stylistes lumineux
   a. Introduction du matériel utilisé
   b. Les protocoles DMX et MIDI
   c. Création d'un programme gérant les couleurs en fonctions du spectre
   d. Création des séquences lumineuses via le PAD et utilisations des curseurs



I. Acquisition/traitement du signal

a. Schéma fonctionnel

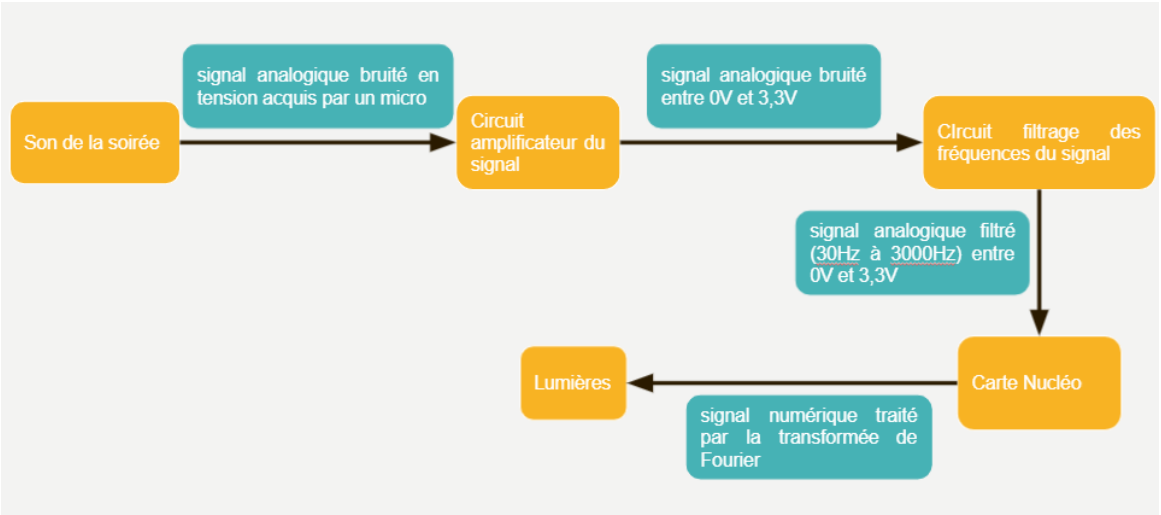


Figure 0 *- Schéma fonctionnel.*

b. Acquisition du signal sonore via un micro : circuit amplificateur

L'objectif ici est de capter un signal sonore, et de l'amplifier de telle sorte qu'il soit traitable par la carte Nucléo. En effet,cette dernière admet des tensions comprises entre 0 et 3,3V.

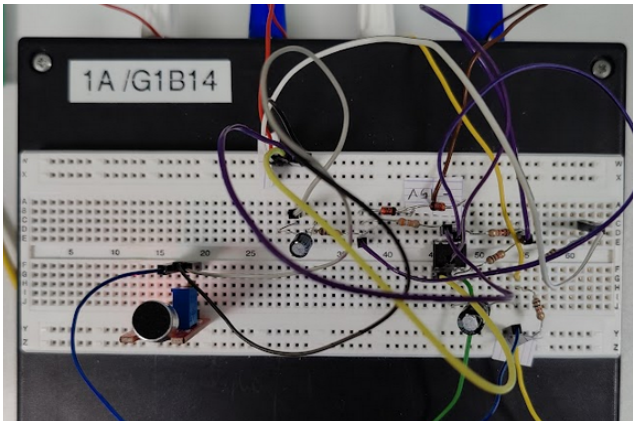


Figure 1 *- Montage électrique du micro et du système d'amplification.*

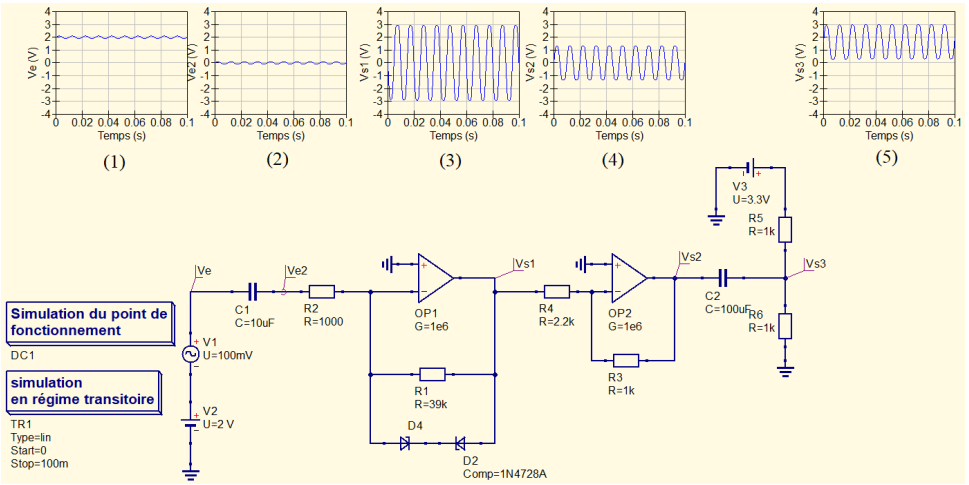


Figure 2 *- Simulation électrique du système d'amplification.*

Que réalise chacune des parties du circuit de la figure 2, de la gauche vers la droite ?

Le graphique 1 est une représentation du signal sonore d'entrée, représenté par le signal V1+V2.

D'abord, le circuit supprime l'offset du signal par un condensateur. Ensuite, un circuit avec un amplificateur opérationnel et des diodes Zeners amplifie le signal pour que son amplitude absolue maximale soit inférieure à 3,3V. Effectivement, l'amplificateur opérationnel amplifie et les diodes Zener limite l'amplitude maximale.

Puisque la carte Nucléo n'accepte que des tensions comprises entre 0V et 3,3V, nous avons utilisé un circuit amplificateur inverseur pour avoir un signal de sortie d'amplitude crête à crête de 3,3V. Enfin, nous ajoutons un offset de 1,65V avec une alimentation de tension 3,3V et un petit circuit diviseur de tension.

c. Traitement du signal issu du micro via un filtre passe-bande

L'objectif ici est de sélectionner uniquement les fréquences utiles, celles de la voix humaine. Ces dernières appartiennent à l'intervalle de fréquence 300 Hz à 3000 Hz. On utilise donc un filtre passe - bande du 4ème ordre. Ce filtre permet une amplification constante pour les fréquences de la bande passante.

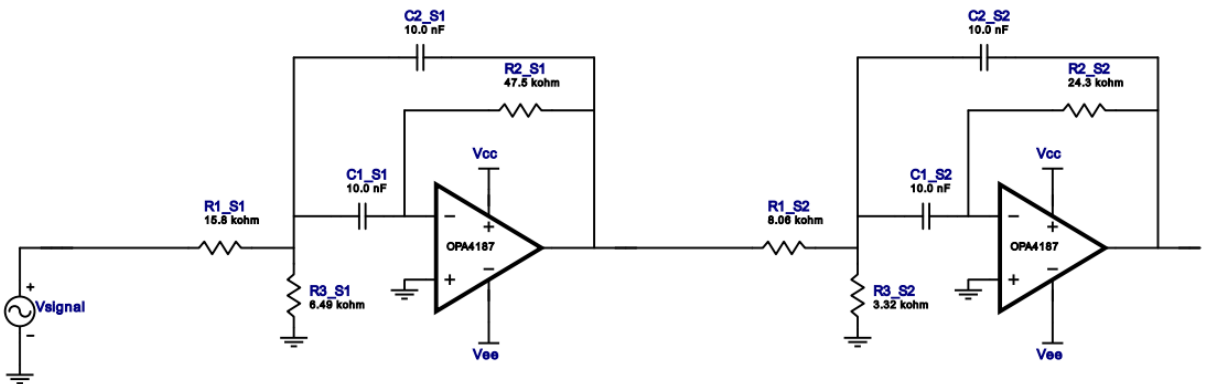


Figure 3 *- Passe bande, les valeurs des composants ne sont pas les valeurs réelles.*

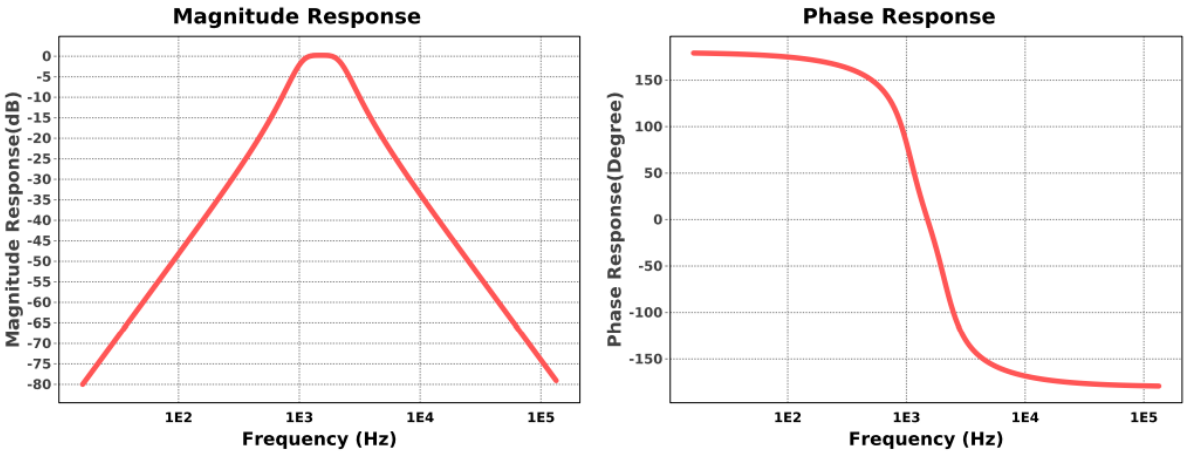


Figure 4 *- Réponse fréquentielle du gain et de la phase du filtre.*

En consultant des documentations, nous avons obtenu les fréquences sonores les plus courantes utilisées dans les soirées: 30 Hz à 3000 Hz. Afin de réduire la difficulté du processus de traitement du signal et de s'assurer que le microcontrôleur obtient les fréquences sonores au-dessus d'une certaine intensité, nous avons choisi de concevoir un filtre avec un gain plat sous la forme d'un filtre Butterworth et, sur la base du modèle de conception correspondant, nous avons calculé les données pour répondre aux exigences.

Expérimentalement, nous avons constaté que le signal de sortie à la fin du filtre était faible, compris entre 0V et 1V alors que le signal d'entrée du circuit est compris entre 0V et 3.3V. Nous avons donc ajouté un circuit pour donner au signal un gain entre 0V et 3,3V.

Par conséquent, le signal d'entrée du microcontrôleur se situe dans la gamme de fréquences que nous avons choisie et peut être utilisé directement sans traitement supplémentaire dans le logiciel, tout en ne causant pas beaucoup de perte et de distorsion du signal.

d. Traitement numérique du signal

L'objectif ici est de rendre la transformée de Fourier (TF) du signal reçu par la carte Nucleo utilisable pour les autres applications

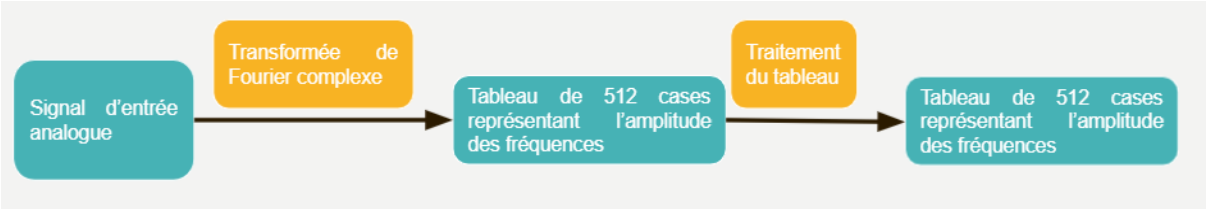


Figure 5 *- Algorithme du traitement numérique du signal*

En effectuant une transformée de Fourier sur le signal reçu par la carte Nucléo, nous obtenons un tableau de taille 512 cases où

- les 256 premières cases correspondent à la partie réelle
- les 256 cases restantes correspondent à la partie imaginaire.

Chacune de ces cases contient l'amplitude d'une fréquence sonore donnée et l'indice d'une case correspond à la fréquence. Ici nous ne travaillons qu'avec la partie réelle du tableau, soit les 256 premières cases.

Nous savons que le spectre de la musique en soirée s'étend jusqu'à 3 kHz. Afin de satisfaire le critère de Shannon Nyquist, nous avons pris une fréquence d'échantillonnage de 10 kHz.

La résolution est de 12 Hz, obtenue en divisant la fréquence maximale 3 kHz par le nombre de cases 256.

Ainsi, à la fin de cette étape, nous obtenons un tableau de 256 cases, représentant l'amplitude des fréquences allant de 1 Hz jusqu'à 3 kHz avec un pas de 12 Hz. Ce tableau sera repris dans la partie suivante.

II. Effets stylistes lumineux

a. Introduction des matériels utilisés

Nous utilisons ici un pad AKAI professional avec 16 touches et 6 curseurs dont nous pouvons coder les fonctions. La communication avec la carte Nucléo se fera par protocole MIDI.

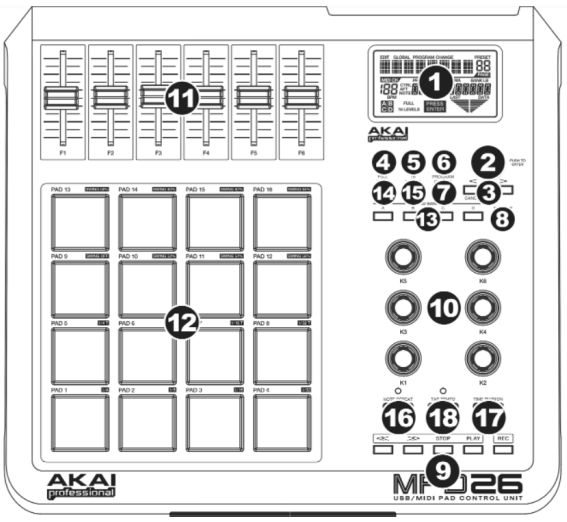


Figure 6 *- Le pad MIDI*

De plus, nous contrôlons des lyres de type Eurolite LED Party Spot. La communication avec la carte Nucléo se fera par le protocole DMX.

b. Description des protocoles MIDI et DMX

Nous allons commencer par décrire le protocole de MIDI qui est lui aussi une méthode de transfert de données conventionnée. Il s'applique à tous les appareils de la gamme MIDI (pad, piano, etc…).

En bougeant un **curseur** on récupère un train de 3 octets, le premier étant un octet de type, le second contenant la position du curseur récupéré dans le variable "**midi_data"** et le troisième étant un octet de valeur que l'on utilise pas.

Quand on appuie sur un **pad** on obtient aussi un train de 3 octets, le premier étant un octet de type, le second étant le numéro du pad récupéré dans le variable "**control_ch".** Le dernier étant un octet de force qui quantifie avec une valeur entre 0 et 127 la force avec laquelle on appuie sur un pad. Par soucis de simplicité nous n'utilisons pas cette fonctionnalité.

L'objectif est à chaque fois de récupérer la valeur comprise dans le deuxième octet du train de données (**midi_data ou control_ch)**. Pour chaque curseur, on reliera la valeur qu'on récupère à une fonctionnalité DMX afin d'avoir une action sur l'allumage des lyres.

Tout d'abord, nous précisons que l'utilisation du terme "lyre" et "lampe" sont identiques pour nous.

Les lyres utilisent le protocole de communication DMX, il s'agit d'un protocole de transfert de données conventionné. Les ordres d'allumage sont envoyés ainsi par une carte Nucléo par le biais d'un câble vers les lampes. Afin que chaque lampe puisse être indépendante on leur donne à chacune des adresses DMX séparées d'au moins 7 canaux. On définit l'adresse d'une lyre comme le premier canal utilisable.

On note que le protocole DMX (lyres) et MIDI (pad) ne sont pas identiques, la difficulté sera donc d'arriver à manier les 2 protocoles de communication de manière cohérente.

On finit par décrire les canaux de communication des lyres suivant (Rampe 3):

Channel	Value	Function
1 **Master dimmer**	000 – 255	Total brightness 0-100 %
2	000 – 255	Red 0-100%
3	000 – 255	Green 0-100%
4	000 – 255	Blue 0-100%
5	000 – 255	White 0-100%
6 **Strobe effect** **Speed channel 7**	000 – 000	No function
	001 – 255	Strobe effect with increasing speed (Channel 1, 2, 3, 4, 5)
	000 – 255	Speed with increasing speed channel 7
7	000 – 050	No function
	051 – 100	Color change, "Switching"
	101 – 200	Color change, "Fading"
	201 – 255	Sound control

***Figure 7** - Fonctions DMX.*

On voit qu'une lampe est contrôlée avec 7 différents canaux. On ne peut mettre que certaines valeurs particulières dans chaque canal. Le canal 1 contrôle l'intensité globale de la lampe. Les canaux 2,3 et 4 contrôlent respectivement les intensités de la proportion de rouge de vert et de bleu pour chaque lampe. Le canal 5 sert à contrôler la proportion de blanc dans chaque lampe, il ne sera pas utilisé. On note que le canal 6 et le canal 7 sont liés. En effet, avec une valeur nulle sur le canal 7 et une valeur non nulle sur le canal 6, alors on ne verra qu'un effet stroboscopique classique. Alors que si on met une valeur non nulle sur les canaux 6 et 7 on aura un effet fading/switching avec une certaine vitesse contrôlée sur le canal 6.

On finit par décrire la répartition des adresses de chaque lampe comme suit :

Figure 8 *- Répartition des canaux pour la rampe 3.*

c. Création d'un programme gérant les couleurs en fonction du spectre

L'objectif est de réaliser une fonction qui gère les lumières à partir du spectre de la précédente partie. Son argument d'entrée est donc un tableau constituant le spectre. Son argument de sortie est un ordre pour les lumières.

```
void TF_to_Light( TF[] ){
    double r = 0 , g = 0 , b = 0 ;
    int fr = sizeof( TF ) ;
    int band = floor( fr / 3.0 ) ;
    int i = 0 ;
    for ( i = 0 , i < band , i++ ){
        r = r + TF[ i ]; }
    r = r / band ;
    for ( i = band , i < 2 * band , i++ ){
        g = g + TF[ i ]; }
    g = g / band ;
    for ( i = 2 * band , i < fr , i++ ){
        b = b + TF[ i ]; }
    b = b / band ;
    double tot = r + g + b ;
    r = floor( 250 * r / tot ) ;
    g = floor( 250 * g / tot ) ;
    b = floor( 250 * b / tot ) ;
    dmx_data[64] = 250;
    dmx_data[65] = r;
    dmx_data[66] = g;
    dmx_data[67] = b;
    dmx_data[68] = 0;
    dmx_data[69] = 100;
    dmx_data[70] = 55;


    dmx_data[72] = 250;
    dmx_data[73] = r;
    dmx_data[74] = g;
    dmx_data[75] = b;
    dmx_data[76] = 0;
    dmx_data[77] = 100;
    dmx_data[78] = 55;


    dmx_data[80] = 250;
    dmx_data[81] = r;
    dmx_data[82] = g;
    dmx_data[83] = b;
    dmx_data[84] = 0;
    dmx_data[85] = 100;
    dmx_data[86] = 55;
    new_note_midi = 0;
}
```

Figure 9 *- Programmation TF_to_Light.*

Cette fonction réalise les actions suivants :

- Découpage du tableau en 3 parties : le rouge(basses fréquences), le vert(moyennes fréquences) et le bleu(hautes fréquences).
- Moyennage des amplitudes de chaque division.
- On applique la précédente valeur à l'intensité de chaque couleur

La figure 5 montre l'algorithme de programmation TF_to_Light, d'abord on divise le tableau obtenu après la transformée de Fourier en trois parties, correspondant aux hautes fréquences, aux moyennes fréquences et aux basses fréquences, puis on calcule la valeur moyenne de chaque partie.

Si le nombre dans le tableau est inférieur à 0,1, on le considère comme du bruit et l'ignore. Cette valeur de 0.1 est obtenue expérimentalement. La valeur maximale d'une fréquence donnée est de 1. Nous avons vérifié qu'en ignorant les valeurs inférieures à 0.1, l'allure du signal est conservée.

Une fois que nous avons obtenu les trois moyennes, nous calculons la proportion de la moyenne de chaque partie dans le tout (la somme des trois moyennes) et enfin nous multiplions chaque proportion par 250 comme la valeur rouge, verte et bleue attribuée à la led.

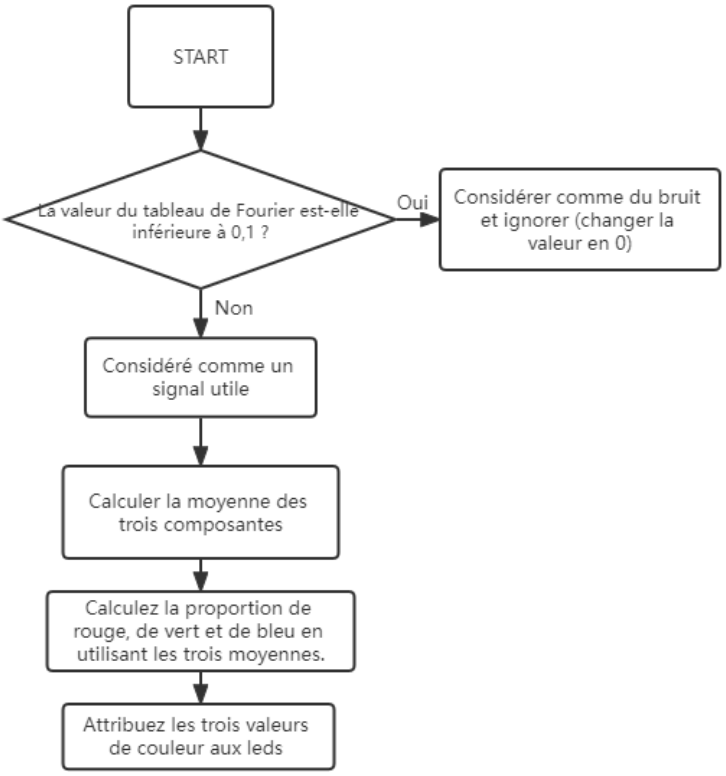


Figure 10 *- Algorithme de Programmation TF_to_Light.*

d. Création des séquences lumineuses via le PAD et utilisations des curseurs

L'objectif dans cette partie est de pouvoir allouer à chaque pad MIDI une fonction lumineuse avec, soit des éclairages fixes, soit changeants avec un certain rythme qu'on va pouvoir contrôler avec l'un des curseurs. Les pads et les curseurs sont répertoriés selon le schéma de la figure 11.

Figure 11 *- Schéma des fonctions de chaque pad et des curseurs.*

Pad	36 à 37	40 à 47 et 49	48	50	51	
Fonctionnalité	jouer une séquence fixe	jouer une séquence animée	tout éteindre	Active le mode sans micro	Active le mode avec micro	
Curseur	1	2	3	4	5	6
Fonctionnalité	contrôler l'intensité de la lyre 1	contrôler l'intensité de la lyre 2	contrôler l'intensité de la lyre 3	Gère la vitesse des séquences animées	Ajouter un effet strob OU vitesse de l'effet fading/switch	Créer un effet fading/switch

Figure 12 *- Tableau récapitulatif des fonctions de chaque pad et des curseurs.*

Pour les séquences fixes (pads 36 à 39), il suffit de mettre un nombre arbitraire entre 0 et 255 sur les canaux r g et b (canaux 1,2 et 3). Pour les séquences animées (concerne les pads 40-47 + 49), on viendra placer sur les canaux r g et b un vecteur de 20 valeurs comprises entre 0 et 255.

Ensuite, on crée un timer que l'on initialise arbitrairement. On va alors l'attribuer sur certains pads afin d'avoir une vitesse d'exécution pour nos séquences. Par la suite, on viendra modifier la période de ce timer avec le curseur 4 afin d'avoir une vitesse variable.

On rappelle que le pad 48 sert à éteindre complètement les lampes, il suffit de mettre toutes les valeurs des 7 canaux DMX à zéro pour chaque lampe.

Enfin, on attribue aux curseurs 50 et 51 un mode de fonctionnement particulier. Chaque bouton permettra de basculer d'une fonction à une autre selon l'algorithme suivant :

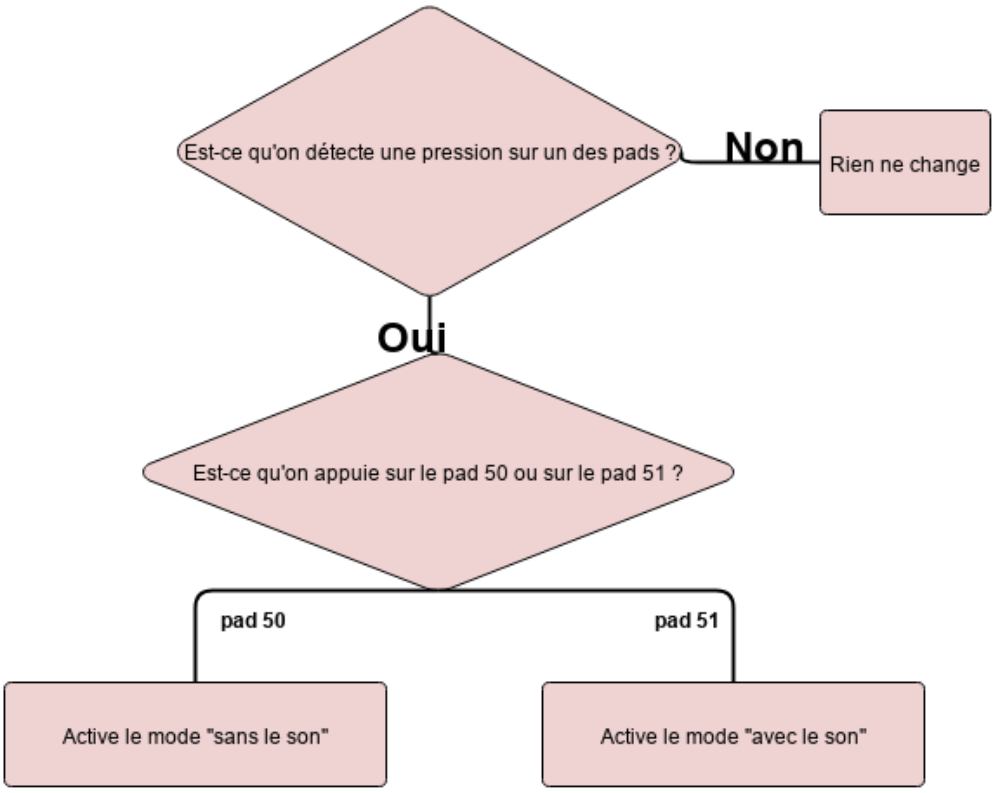


Figure 13 *- Algorithme de fonctionnement des deux modes avec les pads 50 et 51.*

Pour les curseurs 1, 2 et 3 il suffit de **relier** la valeur récupérée dans la variable **midi_data** et le **canal DMX 1** de chaque lyre pour agir sur l'intensité totale.

Pour le curseur 4 on fait la même chose mais en agissant cette fois ci sur la **période de notre timer**. Pour le curseur 5 on agit sur le canal DMX 6 (effet **stroboscopique**) de chaque lyre et pour le curseur 6 on agit sur la canal DMX n°7 (effet f**ading/switching**).

```
if(new_data_midi == 1){  // test si un curseur est levé
    switch(control_ch){  // switche entre les codes pour cu

        case 1 :  // changer l'intesité globale du spot 1
        control_value = midi_data[2]*2;
        dmx_data[64] = control_value;
        new_note_midi = 0;
            break;

        case 2 :  // changer l'intesité globale du spot 2
        control_value = midi_data[2]*2;
        dmx_data[72] = control_value;
        new_note_midi = 0;
            break;

        case 3 :  // changer l'intesité globale du spot 3
        control_value = midi_data[2]*2;
        dmx_data[80] = control_value;
        new_note_midi = 0;
            break;
```

Figure 14 *- Exemple de code pour les curseurs 1,2 et 3.*

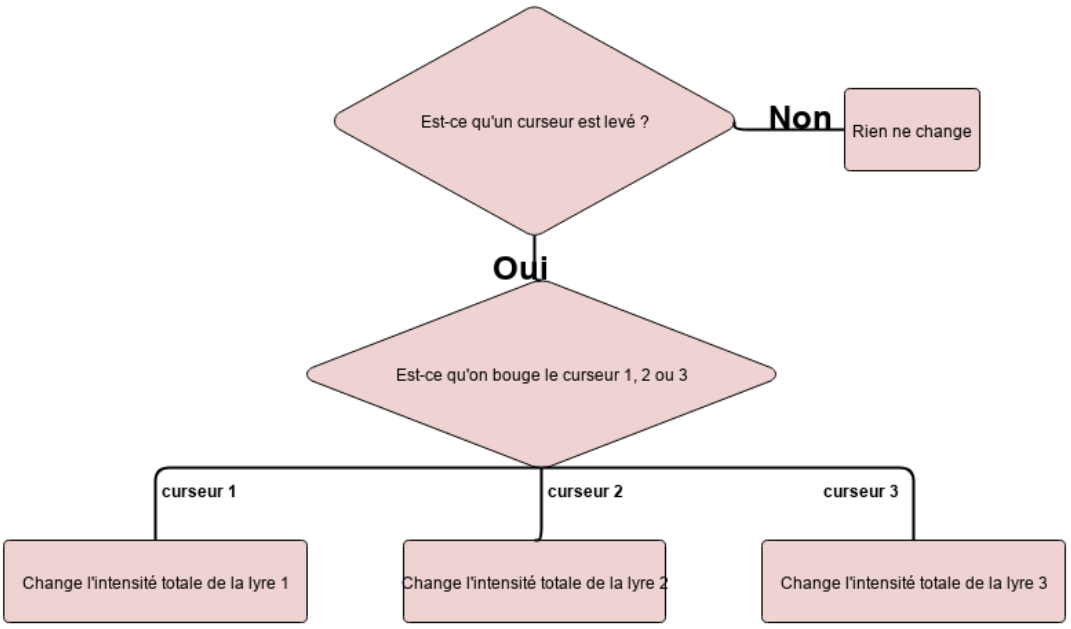


Figure 15 *- Algorithme pour les curseurs 1,2 et 3.*

NB: on multiplie par 2 la valeur de midi_data afin d'avoir une plage de données plus grande.

3. Etapes de réalisation

Pour mener à bien le projet, nous avons établi ensemble le planning de conception : https://docs.google.com/spreadsheets/u/0/d/1ni6gxkluMoqOUzhA5ytzrnIiVG8CzFwbQZy0VPARUjs/edit : *Lien vers le planning.*

4. Conclusion et difficultés rencontrées

Ce projet nous a permis de révéler des difficultés et des atouts à travailler en groupe. Pour commencer, nous avons mis une séance avant de vraiment savoir quel était l'objectif à atteindre. De plus, une de nos grosses difficultés était le branchement complet du système qui prenait beaucoup de temps. Nous avons aussi eu beaucoup de mal concernant la mise en commun des codes car nous n'avons pas déterminé au préalable les variables communes. Ainsi, lors de la mise en commun des codes, les noms des variables se sont entremêlés.

Néanmoins, la communication nous a permis d'aboutir ce projet. Une des qualités dont chacun à fait preuve, est l'entraide, malgré la précision demandée dans chaque domaine. Nous avons réparti très vite les tâches de manière équitable et nous avons été aptes à rapidement rediriger nos efforts quand l'un des membres de l'équipe avait fini avant les autres.