BEATBOX AND LIGHT

Mahomet Boumard, Diane de Dietrich, Vincent Joumani, Stéphanie Lods

Groupe 3 - 2022/2023

Nous attestons que ce travail est original, que nous citons en référence toutes les sources utilisées et qu'il ne comporte pas de plagiat.

I. Description du projet

1) Introduction

Le projet Beatbox and Light consiste à la création d'un instrument musical et lumineux pouvant être utilisé pendant des fêtes. L'ambiance lumineuse, c'est-à-dire l'intensité, les couleurs et les modes, saura s'adapter selon l'envie de l'utilisateur. Les différentes interfaces (clavier et Pad Akai) et spots lumineux (LEDs et Lyre) pourront offrir une multitude de possibilités d'ambiances lumineuses.

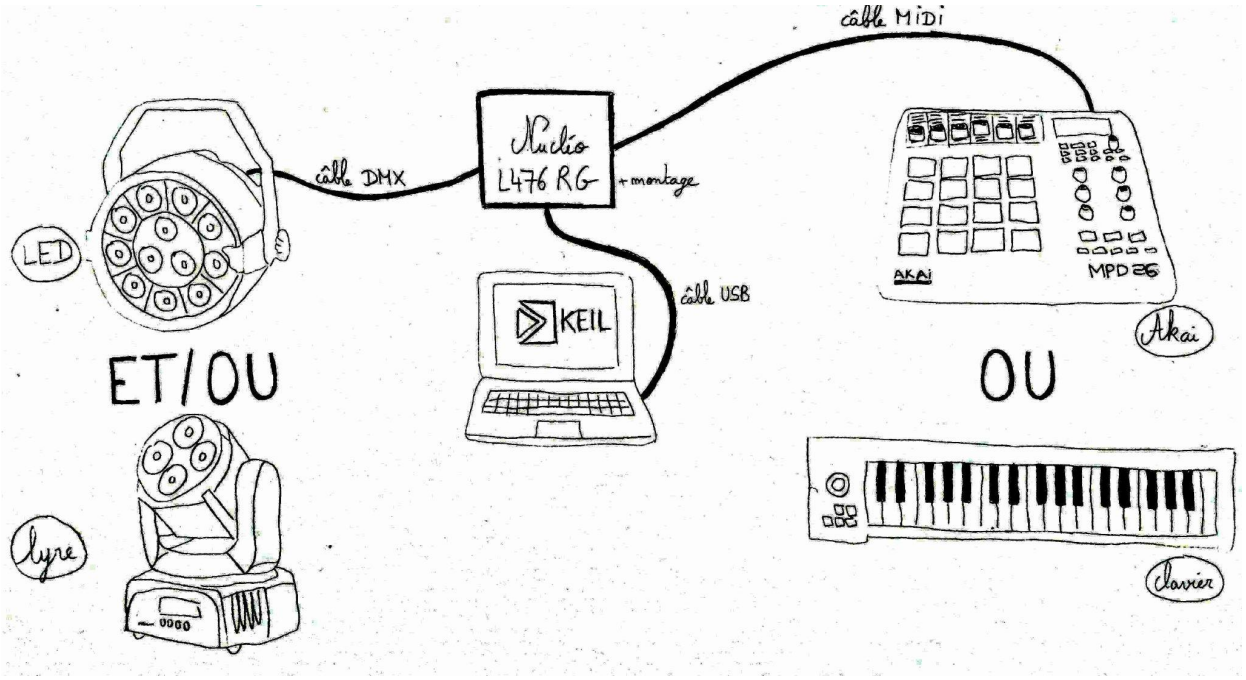

Figure 1 - Schéma du projet "Beatbox and Light"

Le choix de ce projet a été fait en symbiose avec les différents membres de l'équipe. Diane fait partie de l'association du Laserwave donc l'aspect technique du projet permettait une compréhension plus poussée des systèmes utilisés dans l'association. Stéphanie et

Mahomet étaient intéressés par l'aspect visuel et artistique du projet. Vincent maîtrise bien l'aspect informatique et voulait comprendre les processus DMX et MIDI. Nous avons donc élaboré un projet en harmonie avec les envies et les intérêts de chacun.

2) Objectifs

Nous avons établis une liste d'objectifs à réaliser pendant les séances de IéTi

Objectifs principaux :

- ☑ Associer un pad au spot lumineux via un programme
- ☑ Associer un piano numérique au spot lumineux
- ☑ Contrôler des couleurs, de l'intensité, de l'orientation du spot
- ☑ Utiliser ou créer un programme qui convertit données MIDI en DMX
- ☑ Réponse en temps réel (mesure du temps de réaction et adaptation si nécessaire)
- ☑ Ergonomie et praticité
- ☐ Adapter des tensions, mesurer du bruit, ...

Objectifs secondaires :

- ☑ Jouer la musique sur des haut-parleurs (via prise Jack) avec son éclairage associé en temps réel
- ☐ Extraire une partie d'une musique et associer une séquence de lumière correspondante
- ☐ Calcul des dépenses énergétiques du système

On définit dans un premier temps un diagramme fonctionnel pour l'utilisation d'un pad et d'un spot lumineux :

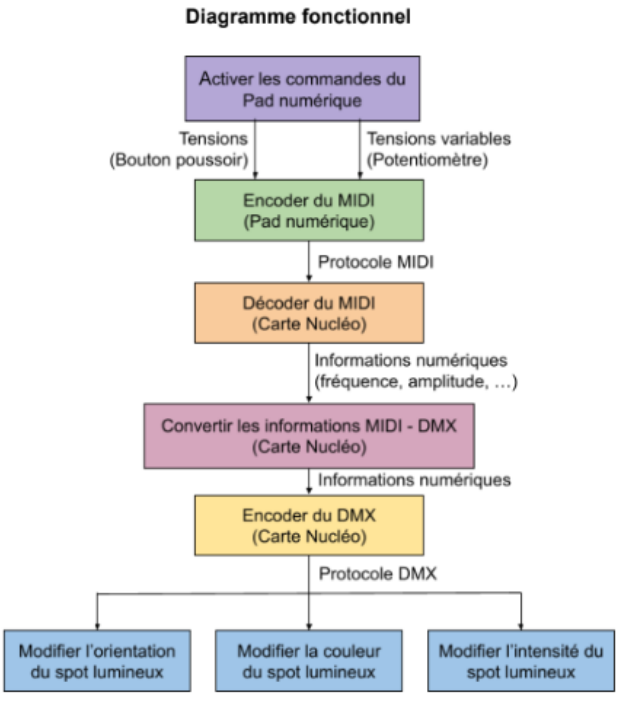

Figure 2 - Diagramme fonctionnel pour le pad et un spot lumineux

3) Matériel

La liste du matériel dont nous disposions est la suivante :

- Le site Keil Studio (https://studio.keil.arm.com/)
- Le logiciel Midi Monitor
- 4 LEDs Renkforce LVPT12 / RGBW - 15 W (notice : https://fr-data.manualslib.com/pdf/fr/pdf1/5/472/47137-renkforce/lvpt12.pdf?ab384e41f3e25e369ed6a1c7a2ede077)
- Une lyre Eurolite TMH-46 / Lyre RGWAUV (notice : https://www.steinigke.fr/download/51785892-Instructions-128780-1.100-eurolite-lyre-led-tmh-46-wash-de_en.pdf)
- Un contrôleur MIDI Akai MPD26 (notice : https://cdn.inmusicbrands.com/akai/attachments/MPD26/mpd26_quickstart_guide_reva_00.pdf)
- Un clavier maître Icon iKeyBoard (notice : https://s3.amazonaws.com/assets.iconproaudio.com/wp-content/uploads/2021/01/18215601/IKB-mini-PD3V103-French.pdf)
- Un code permettant la conversion des données MIDI en données DMX (lien : https://os.mbed.com/teams/IOGS_France/code/Arts_DMX512_carteV3_MIDI_nRF/)
- Une carte Nucléo L476RG accompagnée du montage nécessaire au bon fonctionnement du précédent code

II. Démarche

1) Découverte du projet

Pour la découverte et la compréhension du sujet, nous nous sommes d'abord intéressés au code fourni par M. Villemejane. Pendant les premières séances, nous avons avancé à tâtons car nous avons rencontré de grandes difficultés pour comprendre le code (initilisation de liaisons, conversion DMX-MIDI, significations des variables utilisées, etc). Nous avons donc décidé de mettre le code de côté et de chercher à comprendre le fonctionnement du Pad via MidiMonitor. Ici aussi nous avons eu beaucoup de mal car il fallait une configuration spéciale du logiciel afin qu'il détecte les informations envoyées par l'Akai. Nous avons ensuite cherché à modifier le code afin d'ajouter des messages de détection lorsqu'une note était jouée via TeraTerm. Cela nous aurait permis de mieux comprendre l'appellation des touches du Pad dans le code. Mais cela n'a pas marché car M.Villemejane nous a expliqué que notre Pad ne fonctionnait pas correctement lors de la séance 4. Lors de ces 4 premières séances, on a également approfondi toutes les documentations disponibles notamment pour le protocole DMX des LEDs ou encore l'interface MIDI.

Test des LEDs sous Keil Studio

Avant d'utiliser le contrôleur MIDI Akai, nous nous sommes tout d'abord concentrés sur une des quatre LEDs mises à notre disposition. Nous voulions vérifier la liaison entre Keil Studio

et la LED via la carte Nucléo. Afin qu'on puisse correctement utiliser la LED, l'adressage IP01 était à renseigner sur celle-ci.

Canal DMX	Valeur	Fonction	Remarque
1	000-010	Réglage des couleurs par le biais du canal 4-7	Canaux 2 et 3 sans fonction
	011-050	couleur statique	Sélection de la couleur par le biais du canal 2 Canaux 3-7 sans fonction
	051-100	Fonction changement de couleur	Vitesse par le biais du canal 3 Canaux 2 et 4-7 sans fonction
	101-150	Fonction Fade	Sélection de la couleur par le biais du canal 2 Vitesse par le biais du canal 3 Canaux 4-7 sans fonction
	151-200	Fonction Sound-to-Light	Sélection de la couleur par le biais du canal 2 Canaux 3-7 sans fonction
	201-255	Effet stroboscopique	Sélection de la couleur par le biais du canal 2 Vitesse par le biais du canal 3 Canaux 4-7 sans fonction
2	000-255	Sélection de la couleur	
3	000-255	Vitesse	
4	000-255	Luminosité totale sombre >> claire	active uniquement lorsque canal 1= 000-010
5	000-255	Luminosité rouge foncé >> clair	active uniquement lorsque canal 1= 000-010
6	000-255	Luminosité vert foncé >> clair	active uniquement lorsque canal 1= 000-010
7	000-255	Luminosité bleu foncé >> clair	active uniquement lorsque canal 1= 000-010

Figure 3 - Protocole DMX de la LED (source : notice LED)

Dans le code qui nous été donné, dmx_data[k-1] représente le canal DMX numéro k car le langage utilisé sous Keil Studio est le langage C donc les indices commencent à 0. Il suffisait ensuite d'associer une valeur aux canaux qui nous intéressaient pour obtenir l'effet escompté, le tout en s'aidant du tableau ci-dessus. Ainsi, pour que la LED soit de couleur magenta, le code à rentrer était :

```
dmx_data[4] = 255 ;
dmx_data[6] = 255 ;
```

Si maintenant nous voulions commander les quatre LEDs à la fois, il fallait renseigner les adresses suivantes : IP01 pour la LED1, IP02 pour la LED2, IP03 pour la LED3 et IP04 pour la LED4. Il est important de savoir que les LEDs possèdent sept canaux DMX chacune, et que ces canaux se suivent. Ainsi le dernier canal de la LED1 est dmx_data[6] et le premier canal de la LED2 est dmx_data[7]. Avec quatre LEDs, 28 canaux peuvent être utilisés.

Nous avons finalement codé plusieurs fonctions pour différentes couleurs, prenant comme argument le numéro du premier canal de chaque LED codé en C (numero = 0, 7, 14 et 21

respectivement pour les LEDs 1, 2, 3 et 4). Par exemple, ci-dessous la fonction codant le violet :

```
void violet(int numero)
{
        dmx_data[numero + 4] = 121 ;
        dmx_data[numero + 5] = 28 ;
        dmx_data[numero + 6] = 248 ;
}
```

Test du contrôleur MIDI Akai sous Midi Monitor

Midi Monitor est une application qui permet de voir en instantané les données MIDI envoyées à l'ordinateur depuis le contrôleur, lors de l'appui d'une touche du pad par exemple. Un message MIDI comprend trois octets.

Le premier octet renseigne le type de message (bits de poids forts) ainsi que le numéro de canal MIDI (bits de poids faibles). Par exemple, le bit de NoteOff et NoteOn avec n le numéro de canal MIDI est respectivement $8n et $9n en hexadécimal.

Le deuxième octet indique quant à lui le numéro de note jouée. Le tableau ci-dessous renseigne le code en décimal (en bleu) et en hexadécimal (en vert) pour chacune des notes (de l'octave -2 à l'octave 8). Il est à noter qu'il faut ajouter la syntaxe "0x" devant la note codée en hexadécimal. Il s'agit d'une convention utilisée en C pour indiquer que l'on code en hexadécimal.

	Do	**Do#**	**Ré**	**Ré#**	**Mi**	**Fa**	**Fa#**	**Sol**	**Sol#**	**La**	**La#**	**Si**
octave -2	0 00	1 01	2 02	3 03	4 04	5 05	6 06	7 07	8 08	9 09	10 0A	11 0B
octave -1	12 0C	13 0D	14 0E	15 0F	16 10	17 11	18 12	19 13	20 14	21 15	22 16	23 17
octave 0	24 18	25 19	26 1A	27 1B	28 1C	29 1D	30 1E	31 1F	32 20	33 21	34 22	35 23
octave 1	36 24	37 25	38 26	39 27	40 28	41 29	42 2A	43 2B	44 2C	45 2D	46 2E	47 2F
octave 2	48 30	49 31	50 32	51 33	52 34	53 35	54 36	55 37	56 38	57 39	58 3A	59 3B
octave 3	60 3C	61 3D	62 3E	63 3F	64 40	65 41	66 42	67 43	68 44	69 45	70 46	71 47
octave 4	72 48	73 49	74 4A	75 4B	76 4C	77 4D	78 4E	79 4F	80 50	81 51	82 52	83 53

octave 5	84 54	85 55	86 56	87 57	88 58	89 59	90 5A	91 5B	92 5C	93 5D	94 5E	95 5F
octave 6	96 60	97 61	98 62	99 63	100 64	101 65	102 66	103 67	104 68	105 69	106 6A	107 6B
octave 7	108 6C	109 6D	110 6E	111 6F	112 70	113 71	114 72	115 73	116 74	117 75	118 76	119 77
octave 8	120 78	121 79	122 7A	123 7B	124 7C	125 7D	126 7E	127 7F	- décimal - hexadécimal			

Figure 4 - Tableau récapitulatif pour le codage des notes de musique

Ainsi si l'on joue la note Sol1, le deuxième bit en binaire sera 00101011 = 43d = $2B, car pour le décimal $2^0 + 2^1 + 2^3 + 2^5 = 43$, et pour l'hexadécimal :

- les bits de poids forts (les quatre bits les plus à gauche) donnent $2^1 = 2$;
- les bits de poids faibles (les quatre bits les plus à droite) donnent $2^0 + 2^1 + 2^3 = 11$ = B (car A=10, B=11, …, F=15).

Le troisième octet donne pour finir la valeur de la vélocité (ie la force de frappe). Cet octet est compris entre 0 et 127 en décimal.

```
void ISR_midi_in(void){
    debug_out = !debug_out;
    char data = midi.getc();
     if(data >= 128) // stockage des 3 octets (trame MIDI) >= 128 qui
correspond au premier octet car il commence pas 1xxxxx
        cpt_midi = 0;
    else
        cpt_midi++; // stockage des 2 autres octets
    midi_data[cpt_midi] = data;
    if(cpt_midi == 2){
        cpt_midi = 0;
              if((midi_data[0]  ==  MIDI_NOTE_ON)  ||  (midi_data[0]  ==
MIDI_NOTE_OFF)){ // detection d'une note
            new_note_midi = 1;
            note_data = midi_data[1];
            velocity_data = midi_data[2];
           // printf("k\n");
        }
        else{
            if(midi_data[0] == MIDI_CC){
```

```
                new_data_midi = 1;
                control_ch = midi_data[1];
                control_value = midi_data[2];
            }
        }
    }
}
```

Figure 5 - Code pour récupérer le message MIDI sur 3 octets

2) Pad Akai et LEDs

L'objectif de cette partie est de pouvoir contrôler 4 spots avec un Pad Akai. Le Pad mis à disposition (voir figure ci-dessous) permet un large contrôle de la couleur et l'intensité des spots lumineux, tout cela de façon indépendante.

Figure 6 - Photo du Pad AKai MPD26

On utilisera les curseurs pour contrôler l'intensité des différents spots. On choisit aussi de sélectionner les spots avec 4 curseurs restants. Ainsi, lorsque que le curseur est en haut, le spot correspondant est sélectionné. Le code écrit ne permet pas de sélectionner deux spots en même temps lorsque, par exemple, deux curseurs de sélection sont levés. On remarque que le code choisit le dernier curseur levé vers le haut. Enfin, on élabore une palette de couleur complète avec les 32 touches restantes. Pour créer les couleurs, on doit coder différents niveaux de rouge, vert et bleu. En effet, les spots sont codés en RVB qu'on l'évalue entre 0 et 255 dans le code. Pour créer du jaune (mélange de rouge et de vert) on écrira le code suivant :

```
void jaune(int numero)
{
    dmx_data[numero + 4] = 255;
    dmx_data[numero + 5] = 255;
    dmx_data[numero + 6] = 0;
}
```

Figure 7 - Code définir la couleur jaune sur un spot

La donnée “numéro” permet de repérer le numéro du spot.
On veut de plus pouvoir intégrer un touche qui permet de mettre un mode stroboscopique. Cela est directement possible grâce aux spots lumineux qui possèdent déjà ce mode dans leurs programmes. Ainsi, on peut écrire un code complet (code simplifié à 1 cas pour des soucis de lisibilité):

```
 while(1) //akai + 4 projecteurs
    {   if(isCCMIDIdetected()){
            //playNoteMIDI2(control_ch,control_value);
             if(control_ch == 12){ //curseur 1 en partant de la gauche,
sélection lumière 1 en partant de la gauche (IP01)
                numero = 0; //lumière 1     }
             if(control_ch == 16){ //curseur 5 en partant de la gauche,
intensité lumière 1 en partant de la gauche (IP01)
                int a;
                a=2* control_value;
                dmx_data[3] = a;     }
          resetCCMIDI(); // permet de revenir dans la boucle pour
reperer un changement de valeur  }

     if(isNoteMIDIdetected()){
           if(note_data == 43){ //pad 8
               jaune(numero);       }
           if(note_data == 48){ //pad 13
               stroboscopique(numero);   //effet stroboscopique pour
1 spot sélectionner   }
           if(note_data == 49){ //pad 14
               nonstroboscopique(numero); //stop effet   }
           if(note_data == 50){ //pad 15
               stroboscopique4();   //effet stroboscopique pour
les 4 spots en même temps }
           if(note_data == 51){ //pad 16
               nonstroboscopique4(); //stop effet   }
           resetNoteMIDI(); }
       updateDMX();
       wait_us(10000);  }
```

Figure 8 - Code complet pour contrôler les spots lumineux avec le Pad Akai

En résumé, le set du Pad qui permet de :

- sélectionner un des 4 spots
- régler l'intensité
- choisir une couleur
- créer un mode stroboscopique individuel et multiple

est représenté sur le schéma suivant :

Curseurs

sélection : position haute Intensité max : position haute

select IP01	select IP02	select IP03	select IP04	Intens IP01	Intens IP02	Intens IP03	Intens IP04

Boutons

Strobo multicolor individuel	Stop Strobo	Strobo blanc les 4 (vitesse différente)	Strobo
Violet	Sarcelle	Bleu électrique	Orange
Magenta	Lime	Cyan	Jaune
Rouge	Vert	Bleu	Blanc

Figure 9 - Schéma du Pad Akai associés aux 4 spots lumineux

Figure 10 - Photo du système Pad Akai + 4 spots lumineux

3) Clavier Icon et Lyre motorisée

Le principe de cette partie est d'associer le clavier Icon avec une lyre lumineuse. L'objectif est donc de pouvoir jouer un morceau de piano et ainsi créer une ambiance lumineuse selon les notes jouées. Pour faire cela, nous avons d'abord créé une palette d'ambiance avec un camaïeu de bleu. Cette ambiance s'appellera l'ambiance océanique, que l'on pourra modifier par la suite. On crée un dossier fonction.cpp où l'on code avec des fonctions "void" le set de couleurs nécessaires au nuancier de bleu avec un fonction.h associé. Pour une octave nous avons donc 12 nuances de bleu qui se répètent à chaque octave.

```
void bleu2piano(void)
{
    dmx_data[3] = 180; //intensité
    dmx_data[5] = 30;//rouge
    dmx_data[6] = 136;// vert
    dmx_data[7] = 229; //bleu
}
```

Figure 11 - Code pour une nuance de bleu

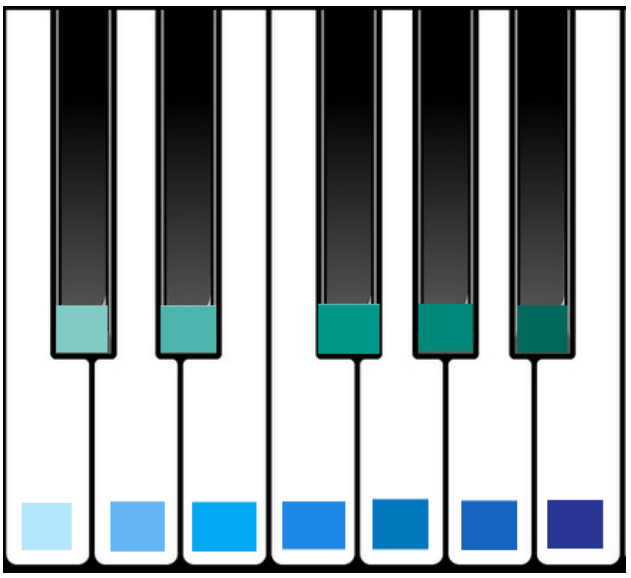

Figure 12 - Schéma des notes associées à leur couleur pour une octave (ambiance océanique)

Figure 13 - Photo du système Clavier Icon + lyre

Le défi principal est de modifier le programme fait dans la partie précédente pour pouvoir l'adapter au Clavier Icon et la lyre. En effet, le protocole DMX de la lyre n'est pas la même que celle des spots lumineux. Nous avons donc utilisé la DMX 16 channels (voir figure ci-dessous)

8	**10**			**Blue**
		0	255	Gradual adjustment of the intensity from 0 to 100 %
9	**11**			**White**
		0	255	Gradual adjustment of the intensity from 0 to 100 %
10	**12**			**Amber**
		0	255	Gradual adjustment of the intensity from 0 to 100 %
11	**13**			**UV**
		0	255	Gradual adjustment of the intensity from 0 to 100 %
12	**14**			**Motion macros, sound controlled motion macros**
		0	15	No function
		16	44	Internal program 1
		45	74	Internal program 2
		75	104	Internal program 3
		105	128	Internal program 4
		129	157	Sound controlled internal program 1
		158	187	Sound controlled internal program 2
		188	217	Sound controlled internal program 3
		218	255	Sound controlled internal program 4

13	15			**Speed motion macros**
		0	255	Decreasing speed
14	16			**Color macros, sound controlled color macros**
		0	0	No function
		1	7	Color fade
		8	39	Internal program 1
		40	72	Internal program 2
		73	105	Internal program 3
		106	138	Internal program 4
		139	140	Internal program 5
		141	167	Sound controlled internal program 1
		168	195	Sound controlled internal program 2
		196	223	Sound controlled internal program 3
		224	251	Sound controlled internal program 4
		252	255	Sound controlled internal program 5
15	17			**Speed color macros**
		0	255	Increasing speed
16	18			**Reset**
		0	249	No function
		250	255	Reset

Figure 11 - Protocole DMX de la lyre (source : notice lyre)

Version saccadée (on utilise la vélocité)

Nous remarquons que lorsque l'on enchaîne très vite 2 notes, les couleurs de la lyre se saccade. Cela ne rend pas fluide l'ambiance d'un morceau de piano. Ce problème vient de l'intensité pris en compte dans le programme (voir figure 11). Dans un premier temps, on avait défini l'intensité en fonction de la force de frappe appliquée sur une touche (correspondant à la donnée velocity_data dans la figure 5).

```
    dmx_data[3] = velocity_data ; //intensité
```

Version ambiance (on n'utilise pas la vélocité)

Pour rendre plus fluide la transition entre les 2 notes de piano, on met une donnée fixe, par exemple 180.

```
    dmx_data[3] = 180; //intensité
```

Version dynamique (on fait tourner la lyre pour chaque touche)

La version dynamique consiste à ajouter la motorisation de la lyre. On garde le code de la version précédente en ajoutant dmx_data[0] et dmx_data[1], d'après le protocole DMX, pour le mouvement de la lyre. On définit ainsi 4 touches tout à droite du clavier pour définir les 4 rotations possibles de la lyre. Cela impose potentiellement la présence de 2 personnes sur le clavier : l'un joue le morceau donc contrôle les couleurs, l'autre contrôle le mouvement de la lumière. On adapte aussi dans le code la vitesse de rotation qui doit être en harmonie avec un morceau de piano plutôt doux.

```
void gauche(void)
```

```
{
    dmx_data[0]=0; //rotation vers la gauche}

void droite(void)
{
    dmx_data[0]=255; //rotation vers la droite }

void avant(void)
{
    dmx_data[1]=0;//rotation vers l'avant }

void arriere(void)
{
    dmx_data[1]=255;//rotation vers l'arrière}
```

Figure 14 - Code pour contrôler le mouvement de la Lyre

Calcul d'un temps de réponse

On ajoute un timer au début de la fonction main pour pouvoir calculer le temps que met le programme pour : 1) Détecter la note ,2) Identifier la note et 3) Afficher la couleur de la note associée

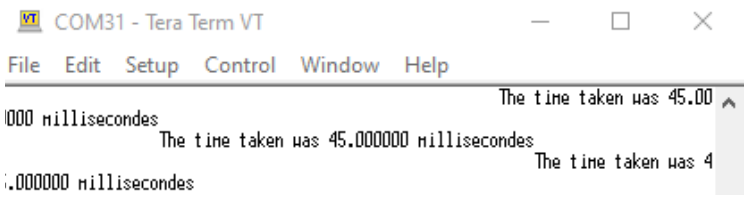

Figure 15 - Interface TeraTerm donnant le temps de réponse du système

On mesure un temps de réponse de **45ms**. Ce temps est suffisant pour que l'œil humain ne perçoive pas de saccade lors de l'utilisation du système. En effet, ce dernier à une sensibilité de 50Hz donc de 20ms, ce qui deux fois moins élevé que le temps de réponse mesuré.

III. Améliorations possibles

Pour la configuration avec l'Akai, on pourrait rajouter une fonction permettant de synchroniser une musique avec l'éclairage. Il faudrait alors extraire une partie d'une musique (ex : les basses), et associer des lumières correspondantes en fonction des notes, du tempo ou de l'intensité. C'est une amélioration envisageable mais qui prendrait beaucoup de temps car il faudrait trouver le moyen d'extraire une partie d'une musique et réussir à envoyer ces données dans le logiciel Keil Studio.

Pour le montage avec le clavier et la lyre motorisée, on pourrait utiliser le joystick du clavier pour contrôler les mouvements de la lyre. C'est une amélioration relativement facile une fois qu'on a trouvé comment les données du joystick sont nommées dans Keil Studio. Enfin, on

pourrait aussi utiliser un micro associé à un chanteur. Comme pour l'Akai, on synchroniserait la voix du chanteur avec l'éclairage.

IV. Conclusion

Nous avons donc réussi à remplir le cahier des charges principal pour ce projet BeatBox and Light. Nous avons rencontré de multiples difficultés sur la partie technique mais nous avons pu avancer grâce à l'aide des encadrants et en nous répartissant bien les tâches. Cela nous a permis de mieux comprendre chaque partie dans le détail et d'avoir une meilleure vision des problèmes rencontrées. Cela nous a aussi permis de découvrir le protocole MIDI et DMX, spécifique à la musique et aux sources lumineuses type spot. On a ensuite pu passer à la partie codage, en avançant en fonction des objectifs de bases et progresser vers les objectifs secondaires souvent plus complexes.

Annexe : planning

Séance 1 [18/01] Choix du sujet et discussion avec l'ensemble de l'**équipe** pour déterminer des objectifs à atteindre. Réalisation du cahier des charges et du diagramme fonctionnel du pad et d'un spot. **Stéphanie** élabore le journal de bord du projet.

Séance 2 [25/01] Découverte du matériel (spot lumineux et pad AKAI). **Vincent** et **Diane** essaient de faire une ébauche de programme sous Keil Studio pour comprendre le fonctionnement du pad AKAI. En parallèle, **Stéphanie** et **Mahomet** lisent la documentation du matériel misent à disposition.

Séance 3 [15/02] Difficulté à comprendre ce qu'il faut écrire dans le programme pour que les actions s'exécutent (par exemple afficher une couleur en appuyant seulement sur bouton du Pad). **Vincent** et **Diane** utilisent le code mis à disposition sur le Lense et essayent de le comprendre et de le décortiquer. **Stéphanie** et **Mahomet** font des recherches plus approfondies sur le protocole MIDI et DMX pour aider la compréhension du code.

Séance 4 [01/03] Le code n'arrive pas à se compiler. L'**équipe** commente étape par étape et cherche ce qui faut modifier dans le code pour enfin le faire marcher. Finalement le programme marche. Premier résultat concluant : le spot lumineux s'allume quand on appuie sur une touche du Pad.

Séance 5 [06/03] Mise en place de 4 spots lumineux reliés au Pad Akai. **Stéphanie** et **Mahomet** créent la palette de couleurs et lisent la notice sur les modes du spot lumineux. **Vincent** et **Diane** modifient le code pour ajouter les 4 spots, l'intensité des spots et un mode stroboscopique (solo ou multiple).

Séance 6 [20/03] Mise en place de la lyre reliée au piano. **Vincent** et **Stéphanie** crée une palette d'ambiance (océanique) et le code sur le programme déjà existant. **Diane** et

Mahomet adaptent le code pour la lyre et le piano et intègre le mouvement de la lyre et l'intensité. L'**équipe** filme une morceau de piano complet avec l'ambiance créée par la lyre.

Séance 7 [15/04] Bilan des 2 montages effectués et prise de vidéos de l'ensemble des éléments du projet. **Mahomet** crée le diaporama pour l'oral et répartis à chacun les parties évoquées. **Vincent** calcule le temps de réponse du système en ajoutant des lignes au code existant. **Diane** et **Stéphanie** commencent l'élaboration et le plan du rapport technique.