Procédés de Traitement de l'Information et du Signal

Application pour pilotage DMX/MIDI

Agathe CHIRIER
Audrey CORON
Sapna HASSANALY
Simon KOUBA

8 Avril 2022

Table des matières

Introduction

Description du projet

Le monde de l'événementiel utilise des projecteurs pilotables (projecteurs PAR, lyres, scanners...) afin de pouvoir créer des scénographies particulières et adaptées à un contexte. La majorité des projecteurs utilisés sont pilotables à l'aide d'un protocole nommé DMX512 (permettant de piloter jusqu'à 512 canaux différents).

Pour pouvoir piloter et programmer les jeux de lumières, en fonction d'un timing particulier ou de la musique, les techniciens utilisent des interfaces informatiques (avec un convertisseur USB-DMX) ou des interfaces directement DMX. Certains utilisent également des contrôleurs MIDI (norme prévue initialement pour la musique numérique).

Afin de faciliter la programmation et l'interfaçage, SOLEC souhaite proposer une gamme complète de produits pour la scénographie, en commençant par une interface MIDI/DMX afin de s'affranchir d'un ordinateur pour la gestion des jeux de lumière.

Source : SOLEC

Cahier des charges

Ce projet pourra être décomposé en deux sous-ensembles :
- *Une interface matérielle de conversion MIDI vers DMX,*
- *Une interface logicielle pour la programmation.*

Contraintes liées à l'interface matérielle : *celle-ci devra permettre de piloter jusqu'à 16 groupes de projecteurs différents (de manière indépendante d'un groupe à l'autre).*

Plusieurs modes doivent être prévus :
- *Mode Scriabin (une note MIDI correspond à une couleur),*
- *Mode Contrôleur MIDI (chaque note est programmée pour afficher un motif particulier sur l'ensemble des 16 groupes de spot),*
- *Mode Séquenceur (une séquence de 16 motifs pourra être lancée, avec un réglage de la vitesse d'itération).*

L'interface permettra de piloter indépendamment (pour chaque projecteur) :
- *Les couleurs : rouge, vert, bleu, ambre, blanc, UV,*
- *L'intensité globale (dimmer),*
- *Les rotations selon les deux angles (tilt et pan) - cas des lyres ou scanners,*
- *Gestion par potentiomètre de l'intensité globale (et des différentes couleurs).*

L'application de programmation permettra de :
- *Créer la liste des projecteurs,*
- *Sélectionner leurs adresses et le nombre de canaux utilisables,*
- *Configurer les différents types de canaux (RGB,W,UV...),*
- *Créer des scènes pour les différentes notes MIDI,*
- *Transférer la configuration à la carte Nucléo.*

Source : SOLEC

1 Démarche de résolution

Afin de répondre aux exigences de ce projet, nous souhaitons apporter une solution technique simple d'utilisation et s'affranchissant totalement d'une liaison avec un ordinateur. Pour ce faire, nous utilisons un clavier électrique qui fournit les notes jouées, une carte NUCLEO munie d'une extension DMX/MIDI, d'une

extension MIDI et d'un potentiomètre ainsi que d'un banc de 4 projecteurs, les Eurolite LED PARty **Figure 1**.

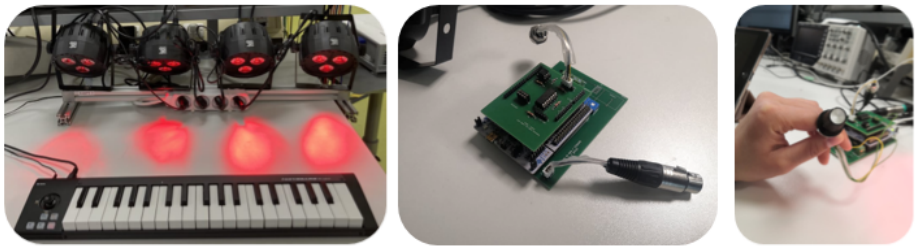

FIGURE 1 – Matériel utilisé : piano et projecteurs (à gauche), carte NUCLEO munie d'une extension DMX/-MIDI (au centre) et potentiomètre (à droite)

Afin de répondre au cahier des charges, 3 modes ont été imaginé. Chaque mode répond à des applications scénographiques différentes. Les trois modes sont :

— Le mode **Scriabin**, qui, pour chaque note, associe la même couleur au banc de projecteurs.
— Le mode **Controleur MIDI**, qui à chaque note associe un motif différent aux projecteurs.
— Le mode **Séquenceur**, qui pour chaque note lance une séquence de 16 motifs.

Le fonctionnement détaillé de chaque mode est présenté par la suite.

Le schéma bloc **Figure 2** illustre la solution retenue. Lorsqu'on joue sur le piano, on récupère la note jouée au format MIDI. Une carte Nucléo permet de lire ce signal. De plus, un code en langage C associe à la note un motif pour le banc de projecteurs, selon le mode choisi. La carte nucléo renvoie en sortie un signal DMX, qui contrôle les projecteurs. Un potentiomètre permet de passer facilement d'un mode à l'autre en s'affranchissant d'un ordinateur.

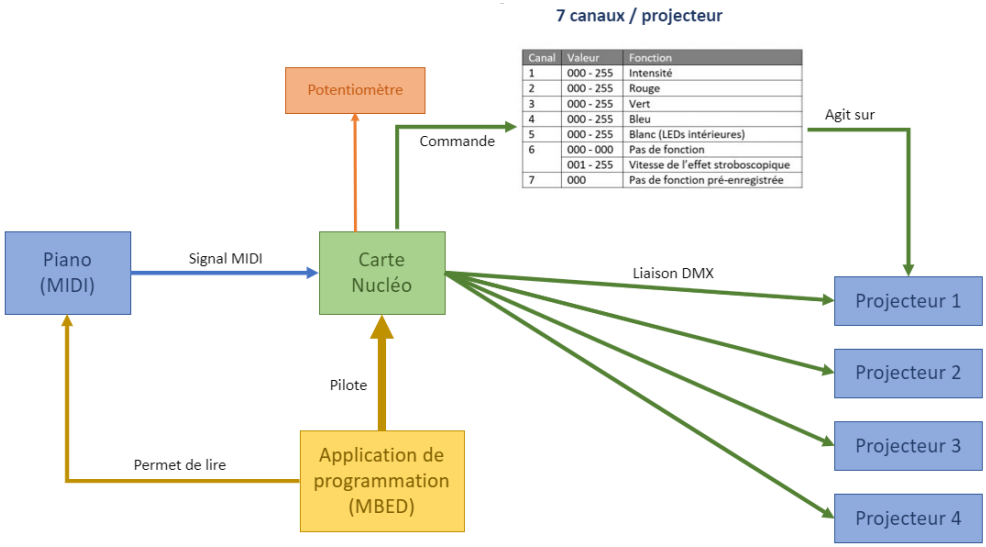

FIGURE 2 – Schéma bloc de la solution technique

2 Fonctionnement du DMX et MIDI

2.1 Entrée MIDI

Lorsque qu'on appuie sur une note sur le clavier, le signal MIDI génère 3 informations (ANNEXE LIGNE 122) :

— **channel_data** que nous n'utilisons pas.

— **note_data** qui donne la note du clavier jouée. Nous avons décidé que c'était la note (en comptant les touches noires) qui comptait et pas l'octave : nous avons donc travaillé avec note=note_data%12. Ainsi, note est un entier qui prend des valeur comprises entre 0 et 12 : 0 correspond à la note Do et 12 correspond à la note Si. De plus on voit que l'action de "lâcher" une touche est interprétée de la même manière que l'action d'appuyer sur une touche.
— **velocity_data** qui donne la vitesse avec laquelle on appuie sur la touche. plus on appuye vite plus sa valeur est élevée et elle est égale à zéro quand on lâche une touche.

Afin d'utiliser ces informations, la liaison MIDI doit être initialiser dans le code (Annexe ligne 117) par une fonction d'initialisation (Annexe ligne 226-231) liée à une fonction d'interruption (Annexe ligne 240-263).

2.2 Sortie DMX

Pour contrôler indépendemment chaque projecteur avec une liaison DMX, on leur associe un numéro différent appelé **adresse**. Pour nos quatre projecteurs, on a choisi les adresses 1, 9, 17 et 25. On peut alors contrôler différents paramètres :
— l'**intensité lumineuse** des LEDs, avec des valeurs comprises entre 0 et 255 : 0 le projecteur est éteint, 255 l'intensité est maximale.
— les composantes **rouge, verte et bleue** (RGB), avec des valeurs comprises entre 0 et 255.
— l'**effet stroboscopique** et la vitesse de du stroboscope avec des valeurs comprises entre 0 et 255 : 0 correspond à pas d'effets stroboscopiques, 1 correspond à la vitesse de clignotement la plus faible et 255 la vitesse la plus élevée.
— La croix de **LEDs blanches** au centre avec des valeurs comprises entre 0 et 255 : 0 les LEDs blanches sont éteintes, 255 les LEDs blanches sont allumées avec une intensité lumineuse maximale.

Le pilotage de chaque paramètre se fait en associant le canal correspondant (voir Figure 2). L'adresse du projecteur noté a est aussi le canal qui contrôle l'intensité lumineuse. Pour contrôler les autres paramètres, il faut choisir les canaux suivants :
— Pour l'intensité lumineuse : canal a.
— Pour la composante rouge : canal $a + 1$.
— Pour la composante verte : canal $a + 2$.
— Pour la composante bleue : canal $a + 3$.
— Pour les LEDs blanches : canal $a + 4$.
— Pour l'effet stroboscopique : canal $a + 5$.

Il est possible de paramétrer d'autres fonctions comme la sensibilité aux ondes sonores, avec le canal $a+6$. Cependant, ces fonctions ne sont pas utilisées pour notre projet, de ce fait, la valeur associée à ce canal sera 0.

Lorsqu'on code les différentes adresses et canaux, il faut faire attention à l'indexation en langage C. En effet, dans ce langage, l'indexation commence à 0. Ainsi, pour associer l'adresse 1 au premier projecteur, il faut mettre la valeur 0 dans le code. On retrouvera ce décalage de 1 pour tous les canaux et adresses.

Tout comme l'entrée MIDI, il est nécessaire d'initialiser la liaison DMX (Annexe ligne 133-142) avec une fonction d'initialisation (Annexe ligne 196-207), de la liaison DMX mais également de chaque canal. Cette fonction est liée à une fonction de mise à jour de la liaison DMX `updateDMX()`, qui suit un protocole précis (Annexe ligne 211-223) afin que les projecteurs qui reçoivent l'information puissent se rendre compte- si cette dernière est nouvelle ou non. Cette fonction `updateDMX()` est donc utilisé après chaque détection de nouvelle note.

3 Implémentation du mode Scriabin

Le premier mode est appelé **mode Scriabin**. il a été imaginé dans un contexte où un pianiste joue sur le clavier directement relié aux projecteurs. Ce mode permet en effet d'associer une couleur à chaque note

jouée pour un effet d'harmonisation. Les couleurs choisies pour chaque note sont présentées **Figure 3**.

FIGURE 3 – Codes RGB des 12 couleurs associées aux 12 notes pour le mode Scriabin

Pour le mode Scriabin nous avons créé 3 vecteurs lignes de 12 éléments, un vecteur ligne par composante de couleur (Rouge Vert Bleue) et où un élément correspond à une note (ANNEXE LIGNE 23-25). Lorsque qu'une note est détectée, on mets à jour la valeur de la variable `note`, chaque paramètre est alors modifié en prenant les valeurs de même indice que `note` dans les vecteurs lignes correspondants (ANNEXE LIGNE 133-142).

Dans la première version du code que nous avons élaboré, le fait de relâcher une note lance l'affichage de la couleur correspondante à cette note. Lorsque l'on joue du piano, on appuie souvent sur une deuxième touche sans avoir lâcher la première, et donc ce code cassait la fluidité des couleurs jouées. Nous avons donc rajouter une condition (ANNEXE LIGNE 123) qui fait que la note détectée n'est prise en compte que si la vélocité est non nulle, ce qui correspond au fait d'appuyer sur une touche, et ainsi l'action de lâcher une note n'est pas prise en compte.

4 Implémentation du mode Contrôleur MIDI

Le deuxième mode requis par le cahier des charges est le **mode Contrôleur MIDI** : ce mode a été pensé principalement dans un environnement festif avec des transitions de lumière brutales et l'utilisation pour certain motif de l'effet stroboscopique.

La différence par rapport au mode Scriabin réside dans le fait qu'il associe à chaque note un motif (ensemble de 4 couleurs différentes avec ou sans effet stroboscopique et avec ou sans LEDs blanches centrales présenté dans la **Figure 4**) et non plus une couleur unique pour l'ensemble des 4 projecteurs. En d'autres termes, les quatre projecteurs sont cette fois commandés indépendamment les uns des autres, là où ils recevaient les mêmes consignes dans le cas du premier mode. De ce fait, on crée cette fois ci un vecteur ligne de 48 valeurs pour chaque canal à contrôler (ANNEXE LIGNE 27-37) car chaque vecteur correspondant à un canal est composé de 12 valeurs $\times$ projecteurs afin de coder chaque motif. On modifie la valeur de chaque paramètre à l'aide de boucle `for` (ANNEXE LIGNE 146-154).

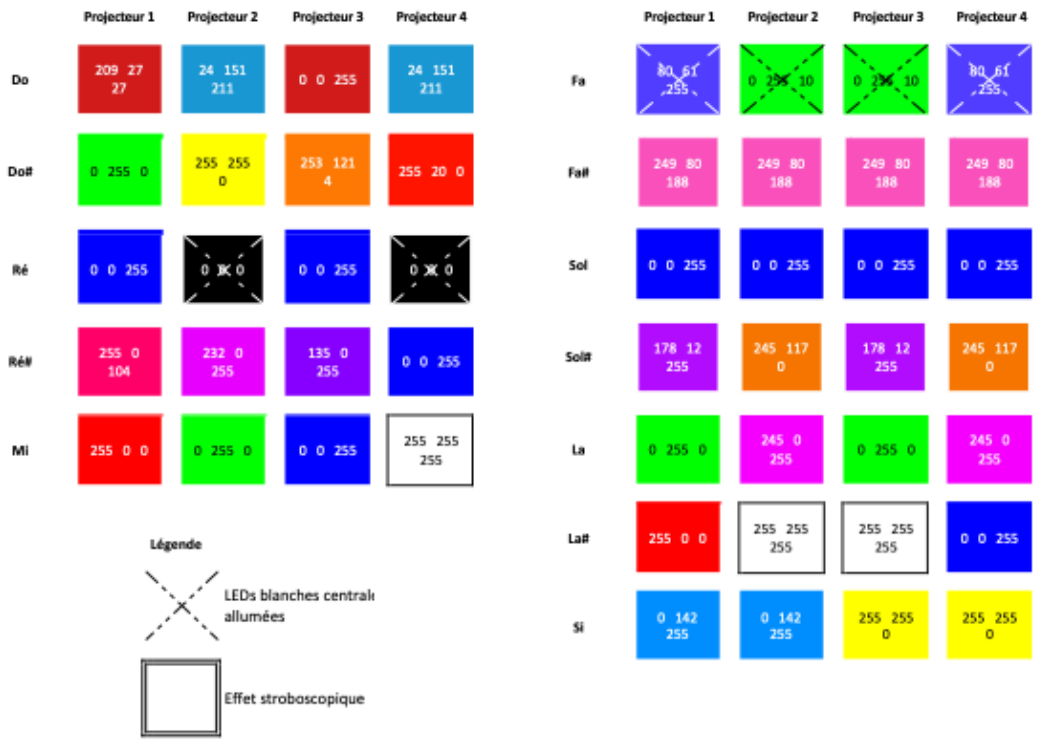

FIGURE 4 – Codes RGB des 12 modes associés aux 12 notes pour le mode Contrôleur MIDI

5 Implémentation du mode Séquenceur

Le troisième et dernier mode requis par le cahier des charges est le **mode Séquenceur** : à la différence des deux premiers, il n'associe pas à une note une couleur ou un motif, mais une vitesse d'itération pour une séquence de 16 motifs consécutifs préalablement définis (**Figure 5**). Ainsi, la note Si correspond à la vitesse d'itération la plus faible tandis que la note Do correspond à la plus grande vitesse d'itération.

Pour ce mode on crée des vecteurs lignes correspondant aux différents paramètres que nous voulons contrôler (ANNEXE LIGNES 41-50). Ces vecteurs sont composés de 4×16 éléments (les 16 premiers éléments correspondent au motif que va afficher le premier projecteur, les 16 suivants au motif du deuxième, etc...). On ajoute un autre vecteur ligne (ANNEXE LIGNE 53) qui permet de contrôler la vitesse de la séquence du motif. Il est composé de 12 éléments qui correspondent aux différentes notes.

Pour coder ce mode (ANNEXE LIGNES 157-186), on a créé deux boucles `for` : la boucle `i` qui affectent les valeurs voulues aux 4 projecteurs pour 1 motif, et la boucle `j` qui change le motif. Il y a, à la fin de cette boucle (ANNEXE LIGNE 184), un `wait` qui permet de contrôler la vitesse d'itération des motifs.

Cependant si on se contente de faire cela, il se pose deux problèmes : le motif ne se répète pas et il faut attendre qu'un motif soit fini pour qu'appuyer sur une note fonctionne et lance le motif avec une autre vitesse. Pour régler ces deux problèmes nous avons mis une boucle `while` à la place de la boucle `if` initialement mise LIGNE 157. De plus, nous avons rajouté dans la boucle `j` (qui permet d'enchaîner les différents motifs) une condition qui permet de sortir de cette boucle si une note est détectée et de déclencher la succession de motifs avec la vitesse correspondant à cette nouvelle note (ANNEXE LIGNES 170-183).

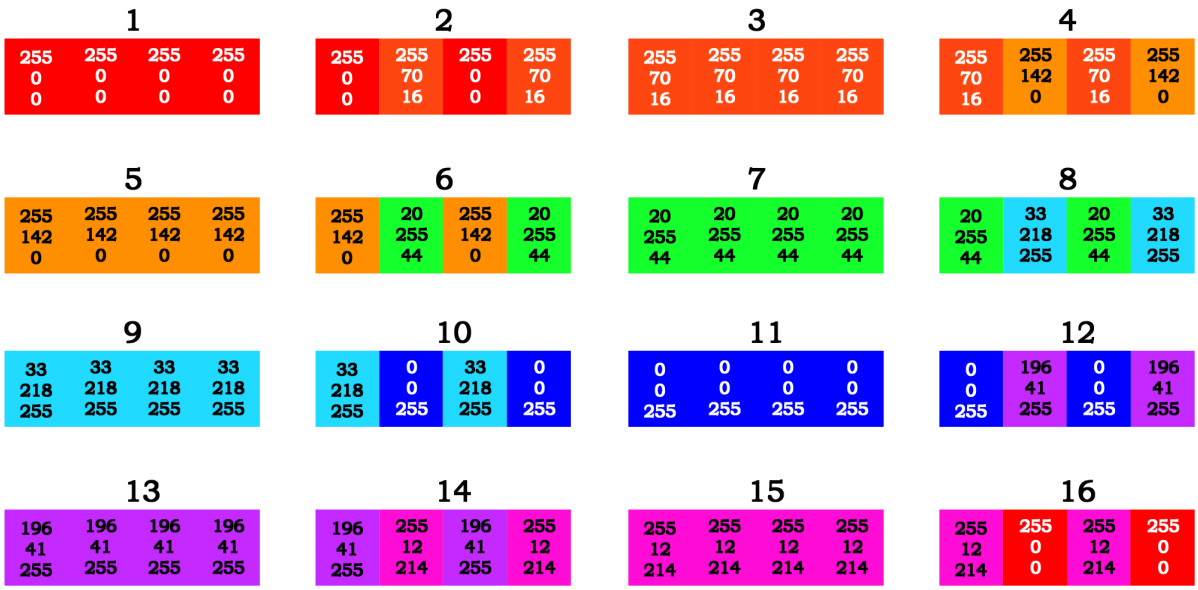

FIGURE 5 – Codes RGB des 12 motifs qui défilent successivement sur les 4 projecteurs

Ce dernier modes a été penser, contrairement au deux premiers, dans un contexte scénographique plus calme, comme un scène de théâtre par exemple. En effet les motifs choisis illustrés ci-dessus ont été créés pour former un dégradé de couleurs pour habiller une scène, avec une itération plus ou moins rapide suivant l'effet souhaité.

6 Interface de pilotage

Après avoir implémenté avec succès les trois modes présentés dans le cahier des charges, nous avons souhaité les réunir au sein d'un seul et unique fichier de code, de sorte à pouvoir passer d'un mode à l'autre à l'aide d'une **interface de pilotage** que l'on reliera à la carte Nucléo.

L'interface de pilotage se base sur l'utilisation d'un **potentiomètre** : ce dispositif se présente sous la forme d'un bouton rotatif qui à chaque position associe une tension comprise entre 0 et 1 V. Le montage électrique ainsi modifié est représenté en **Figure** 6.

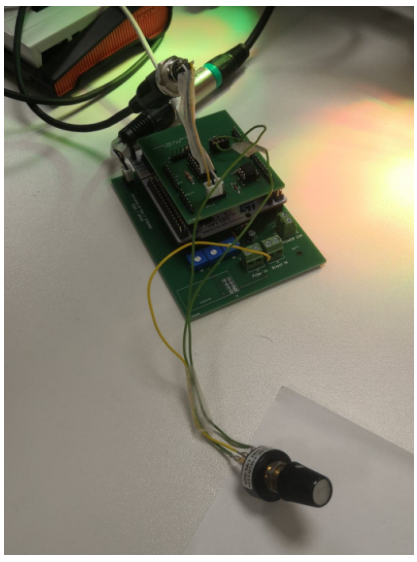

FIGURE 6 – Photo du montage avec le potentiomètre

Le code utilisé est présenté en ANNEXE 4 et fonctionne de la façon suivante : selon la valeur prise par le

potentiomètre (choisie en temps réel par l'utilisateur), le code redirige vers l'un ou l'autre des modes à l'aide de conditions `if`. Le contenu de ces conditions est alors identique aux codes présentés en Annexes 1, 2 et 3. Plus précisément comme illustré **Figure 7** :

— Pour les potentiels compris entre 0 V et 0,33 V, le code redirige vers le mode Scriabin.
— Pour les potentiels compris entre 0,33 V et 0,65 V, le code redirige vers le mode Contrôleur MIDI.
— Pour les potentiels compris entre 0,65 V et 1 V, le code redirige vers le mode Séquenceur.

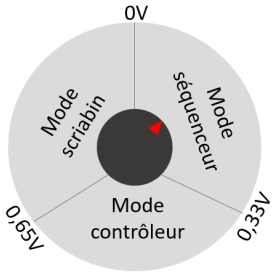

Figure 7 – Schéma du potentiomètre

Conclusion

Notre projet répond au cahier des charges et est fonctionnel. Il est possible de piloter les LEDs à l'aide du piano en s'affranchissant de l'ordinateur grâce au potentiomètre, ce qui était pour nous le principal objectif de ce projet. Il est toutefois possible d'améliorer ce prototype.

Perspectives d'amélioration

Actuellement, le principal problème de notre dispositif est sa rigidité. En effet, si une personne veut modifier les couleurs ou motifs d'un mode, ou veut changer les vitesse du mode séquenceur, elle sera obligée d'ouvrir le code avec Mbed et devra modifier manuellement les éléments des vecteurs lignes correspondant. Ces vecteurs étant assez conséquents, ce travail est très fastidieux et pas du tout intuitif pour un utilisateur. On peut donc envisager de créer une interface avec Matlab qui permettrait de choisir les couleurs des différents modes avec par exemple une roue colorimétrique.

De plus notre code ne fonctionne actuellement que pour 4 projecteurs, on pourrait imaginer aisément faire un code général avec N projecteurs et le choix du nombre de projecteurs se ferait sur l'interface Matlab évoquée précédemment.

Retour sur le projet

Le sujet du projet a été très intéressant à traiter. Nous avons principalement appris le fonctionnement et l'utilisation de liaisons MIDI et DMX. L'avantage de ce projet d'application est que des résultats sont rapidement visibles. Par ailleurs, ils nous a permis de revoir le langage C, qui a été vu en cours l'année dernière. De plus, obtenir un prototype fonctionnel tel que nous l'avions imaginé au départ a été très gratifiant.

En terme d'organisation, Nous avons principalement travaillé par binôme lors des séances de travail afin de couvrir tous les aspects du projet et nous communiquions au travers de l'application TEAMS afin de conserver une trace écrite de l'avancée de notre projet à chaque étape. La communication et la répartition du travail s'est plutôt bien passée. Nous n'avons pas eu de grand problème au sein de notre équipe.

Annexes

Le code illustré en annexe est prévu pour une carte `Nucléo L476_RG` avec une extension `MIDI-IN/MIDI-OUT`. La version présentée dans ce document (nottament les vecteurs) a été coupée pour plus de lisibilité.

```
/*****************************************************************/
/*  PROJET SOLEC : APPLICATION POUR PILOTAGE DMX/MIDI            */
/*****************************************************************/
/*  CORON AUDREY, CHIRIER AGATHE, HASSANALY SAPNA, KOUBA SIMON   */
/*****************************************************************/
/*  Institut d'Optique Graduate School                           */
/*****************************************************************/
// Ce code permet de commander 4 projecteurs (sortie DMX)  l'aide d'un clavier
// de type piano (sortie MIDI), les deux etant relies par une carte nucleo.
// Nous avons programme 3 modes :
// - 1er mode (potentiometre <0.33V): les projecteurs affichent tous la m me couleur pour une note;
// - 2 me  mode (0.33V<potentiometre <0.65V): appuyer une une touche declenche un motif.
// - 3 me  mode (potentiometre >0.65V): les projecteurs affichent une succession de 16 motifs, et la
//note defini la vitesse avec laquelle les motifs s'encha nent ("do" est le plus
// rapide, "si" le plus lent).
// Pour changer de mode on utilise un potentiom tre.

#include "mbed.h"
#include "platform/mbed_thread.h"

//tableaux mode 1 (scriabin)
                      //notes:  1   2   3   4   5   6   7   8   9   10  11  12
const uint8_t scriabin_r[12] = {255,255,255,128,  0,  0,  0,  0,  0,127,255,255};// composantes rouges
const uint8_t scriabin_g[12] = {  0,128,255,255,255,255,255,128,  0,  0,  0,  0};// composantes vertes
const uint8_t scriabin_b[12] = {  0,  0,  0,  0,  0,128,255,255,255,255,255,127};// composantes bleues

//tableaux mode 2 (controleur)                              //projecteur 1                ...
                           //motifs:    1   2   3   4   5   6   7   8   9   10  11  12   ...
const uint8_t controleur_bright[48] = {255,255,255,255,255,255,255,255,255,255,255,255, ...}// intensite

const uint8_t      controleur_r[48] = {209,  0,  0,255,255, 80,249,  0,178,  0,255,  0, ...}// composantes rouges
const uint8_t      controleur_g[48] = { 27,255,  0,  0,  0, 61, 80,  0, 12,255,  0,142, ...}// composantes vertes
const uint8_t      controleur_b[48] = { 27,  0,255,104,  0,255,188,255,255,  0,  0,255, ...}// composantes bleues

const uint8_t      controleur_w[48] = {  0,  0,  0,  0,  0,128,  0,  0,  0,  0,  0,  0, ...}// leds blanches

const uint8_t controleur_strobe[48] = {  0,  0,  0,  0,  0,  0,200,  0,100,  0,  0,  0, ...}// effet stroboscopique


//tableaux mode 3 (sequenceur)

const       uint8_t    sequenceur_bright[64] = {...}; // intensite

const       uint8_t         sequenceur_r[64] = {...}; // composantes rouges
const       uint8_t         sequenceur_g[64] = {...}; // composantes vertes
const       uint8_t         sequenceur_b[64] = {...}; // composantes bleues

const       uint8_t         sequenceur_w[64] = {...}; // leds blanches

const       uint8_t    sequenceur_strobe[64] = {...}; // effet stroboscopique

 // reglage de la vitesse du mode sequenceur ( de 1 millieme de seconde    2.4 secondes)
```

```
const       double          vitessemode[12]={0.001, 0.01, 0.05, 0.1, 0.3, 0.6, 0.9, 1.2,
    1.5, 1.8, 2.1,2.4 };


//definition de l'entree MIDI
#define     MIDI_NOTE_ON        0x90
#define     MIDI_NOTE_OFF       0x80
#define     MIDI_CC             0xB0

#define SAMPLES     512

int i; // numero du projecteur
int j; // numero du motif pour le mode 3
int note_mode3; // note enrengistree quand on reappui sur une note alors que le motif n'est
    pas fini pour le mode 3

Serial      debug_pc(USBTX, USBRX); //debugage: permet de visualiser des informations via
    TeraTerm
InterruptIn     my_bp(USER_BUTTON);

// definition des entrees et sortie de la Nucleo

Serial      dmx(A0, A1);
Serial      midi(D8, D2);

DigitalOut  out_tx(D5);
DigitalOut  start(D4);
DigitalOut  enableDMX(D6);

AnalogIn    CV_volume(PC_1);
AnalogIn    CV_pitch(PB_0);
AnalogIn    variation(A3); //potentiometre



// 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;
char        note;


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

//Potentiometre
double potentio; // recup re la valeur du potentiom tre afin de changer de mode


// Boucle principale
int main() {
    double time;

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

    initDMX(); //initialisation de la sortie DMX
    initMIDI(); //initialisation de l'entree MIDI

```

```
119
120     while(1) {
121         if(isNoteMIDIdetected()){
122             debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
    //visualisation des donnees MIDI
123             if (velocity_data) { //boucle pour que relacher une touche ne soit pas considere
     comme une nouvelle note
124                 note= note_data%12;           // le reste de la division euclidienne par 12
    donne la hauteur de la note jouee
125             }
126             debug_pc.printf("N=%d\r\n", note); // visualisation de la note
127
128             potentio=variation.read();
129             debug_pc.printf("pressoir %lf \r\n",potentio );
130
131             if (potentio<0.33){ //Mode 1
132
133                 for(int i=0; i<4; i++){ //pour chaque projecteur
134                     dmx_data[0+i*8] = 100; // intensite
135                     dmx_data[1+i*8] = scriabin_r[note]; //rouge
136                     dmx_data[2+i*8] = scriabin_g[note]; //vert
137                     dmx_data[3+i*8] = scriabin_b[note]; //bleu
138                     dmx_data[4+i*8] = 0 ; // LEDs blanches du milieu
139                     dmx_data[5+i*8] = 0; //effet stroboscopique
140                     dmx_data[6+i*8] = 0;  // absence de fonction pre-enregistree
141                     }
142                 }
143
144             if (0.33<potentio && potentio<0.65){ //Mode 2
145
146                 for(int i=0; i<4; i++){ //pour chaque projecteur
147                     dmx_data[0+i*8] = controleur_bright[note+ i*12];  //intensite
148                     dmx_data[1+i*8] = controleur_r[note+i*12]; //rouge
149                     dmx_data[2+i*8] = controleur_g[note+i*12]; //vert
150                     dmx_data[3+i*8] = controleur_b[note + i*12]; //bleu
151                     dmx_data[4+i*8] = controleur_w[note + i*12]; //LEDs blanches du milieu
152                     dmx_data[5+i*8] = controleur_strobe[note + i*12]; //effet stroboscopique
153                     dmx_data[6+i*8] = 0;  // absence de fonction pre-enregistree
154                     }
155                 }
156
157             while (potentio>0.65){ //Mode 3 (motif en boucle infinie)
158                 for(int j=0; j<16; j++){ // pour chaque motif
159                      for(int i=0; i<4; i++){ //pour chaque projecteur
160
161                         dmx_data[0+i*8] = sequenceur_bright[j+i*16];  //intensite
162                         dmx_data[1+i*8] = sequenceur_r[j+i*16]; //rouge
163                         dmx_data[2+i*8] = sequenceur_g[j+i*16]; //vert
164                         dmx_data[3+i*8] = sequenceur_b[j+i*16]; //bleu
165                         dmx_data[4+i*8] = sequenceur_w[j+i*16]; //LEDs blanches du milieu
166                         dmx_data[5+i*8] = sequenceur_strobe[j+i*16]; //effet stroboscopique
167                         dmx_data[6+i*8] = 0;  //absence de fonction pre-enregistree
168                         }
169
170                     if (isNoteMIDIdetected()){//permet de sortir de la boucle for si on
    reactionne une note alors que le motif n'est pas fini
171                         new_note_midi = 0;
172                         updateDMX();
173                         note_mode3=note;
174                         debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data,
    velocity_data);
175                         note=note_data%12;
176
177                         if (note!=note_mode3){//sort de la boucle si la nouvelle note
    appuyee est differente de l'ancienne
178                             break;
179                             }
180                         }
181                     new_note_midi = 0;
```

```
                    updateDMX();
                    potentio=variation.read();
                    wait_us(vitessemode[note]*1E6); //reglage de la vitesse de transition
                }
        }

new_note_midi = 0;                          // reinitialisation
updateDMX();                                // envoi des mises    jour dans les projecteurs
wait_us(10000);                             // temps minimal d'attente entre deux notes
        }
    }
}


// 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 a 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 recue 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)){
```

```
250             new_note_midi = 1;
251             channel_data = midi_data[0] & 0x0F;
252             note_data = midi_data[1];
253             velocity_data = midi_data[2];
254         }
255         else{
256             if(midi_data[0] == MIDI_CC){
257                 new_data_midi = 1;
258                 control_ch = midi_data[1];
259                 control_value = midi_data[2];
260             }
261         }
262     }
263 }
```