Rapport technique
Projet Beatbox & Lights

BROUSSOUX Jeanne
SEBOUAÏ Arno
ESVARANATHAN Janusan
PLESSE--COSTA Ferdinand
G3

Session 2021.2022

INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech
LEnsE Laboratoire d'Enseignement Expérimental

Table des matières

I. Introduction

Ce projet *Beatbox & Lights* consiste à élaborer un système utilisable lors d'évènements associatifs de l'Institut d'Optique (Optibars, foyer, soirées…) permettant de coordonner la lumière et le son. Il s'agit ici de connecter une interface MIDI à une série de projecteurs illuminant une salle. Cette interface MIDI peut prendre la forme d'un clavier ou bien d'un pad. Connectée à un éditeur de partitions musicales (logiciel MuseScore), cette dernière nous permet aussi de produire des sons (instruments de musique, électroniques…). L'objectif de notre projet est donc de faire jouer de la musique à la lumière afin de créer des ambiances de salle.

1. Problématique

Dans quelle mesure pouvons-nous utiliser la technologie MIDI pour améliorer la cohérence entre lumière et son lors d'événements associatifs ?

2. Objectifs et démarche suivie

Cahier des charges :

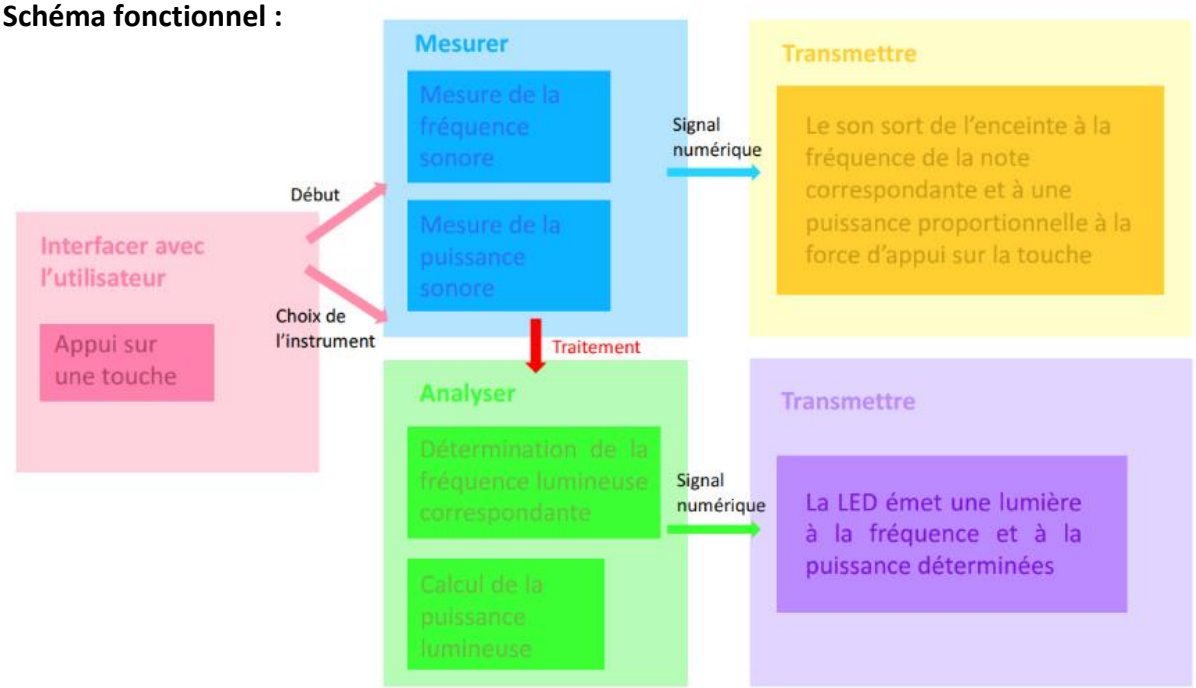
Fonction	Le produit doit permettre d'ajouter à l'ambiance sonore d'une salle de musique une ambiance lumineuse correspondante.
Projecteurs utilisés	Le système doit pouvoir s'adapter au stroboscope LED DMX Eurolite (12W) ainsi qu'au Renkforce LVPT12 (15W)
Nombre de projecteurs	Le nombre de projecteurs utilisables doit être compris entre 1 et 3.
Gamme de fréquences lumineuses	Les couleurs associées à chaque touche doivent être choisies selon le tableau de « Correspondance des fréquences musique/couleurs »

Les références des spots (Eurolite et Renkforce) sont disponibles sur le site du LEnsE

On a accompli notre objectif pour 4 projecteurs

La correspondance des couleurs initiales vient du site : https://musicordes.fr/tableau-correspondance-notes-couleur/

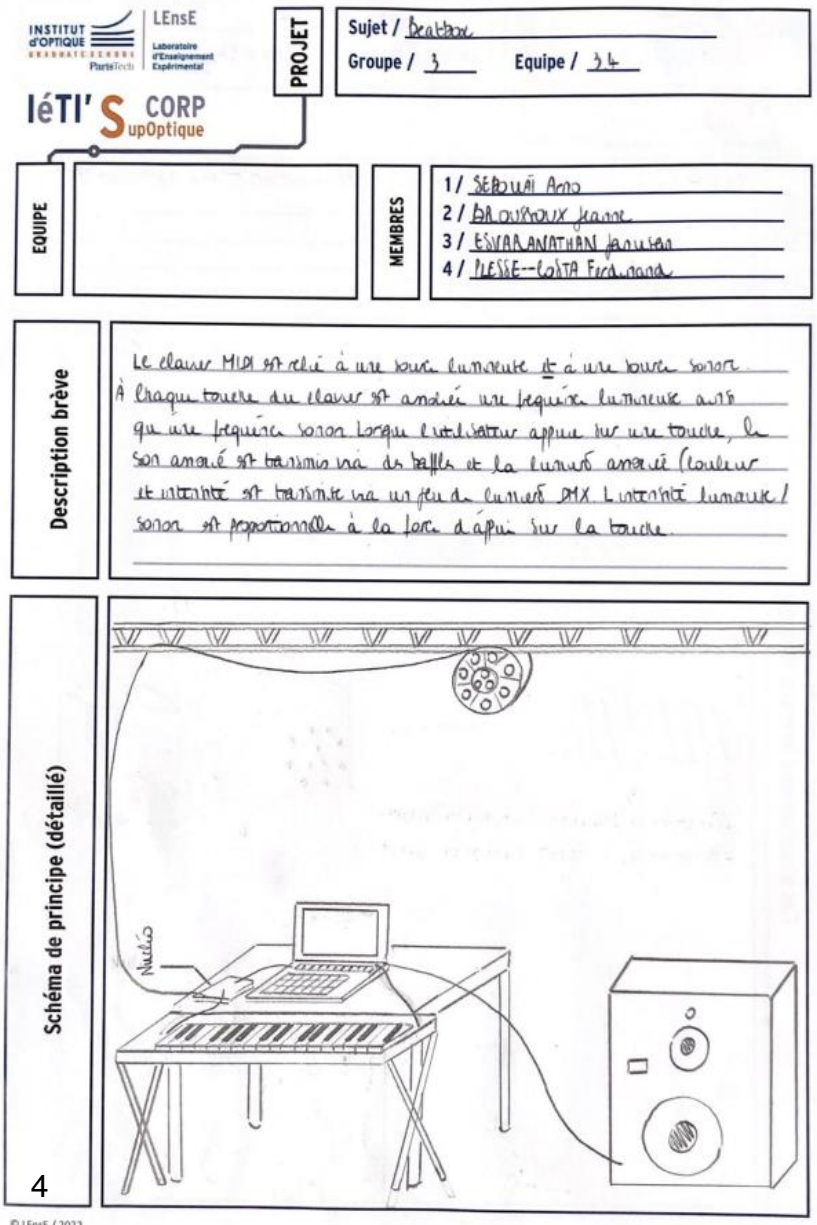
Schéma fonctionnel :

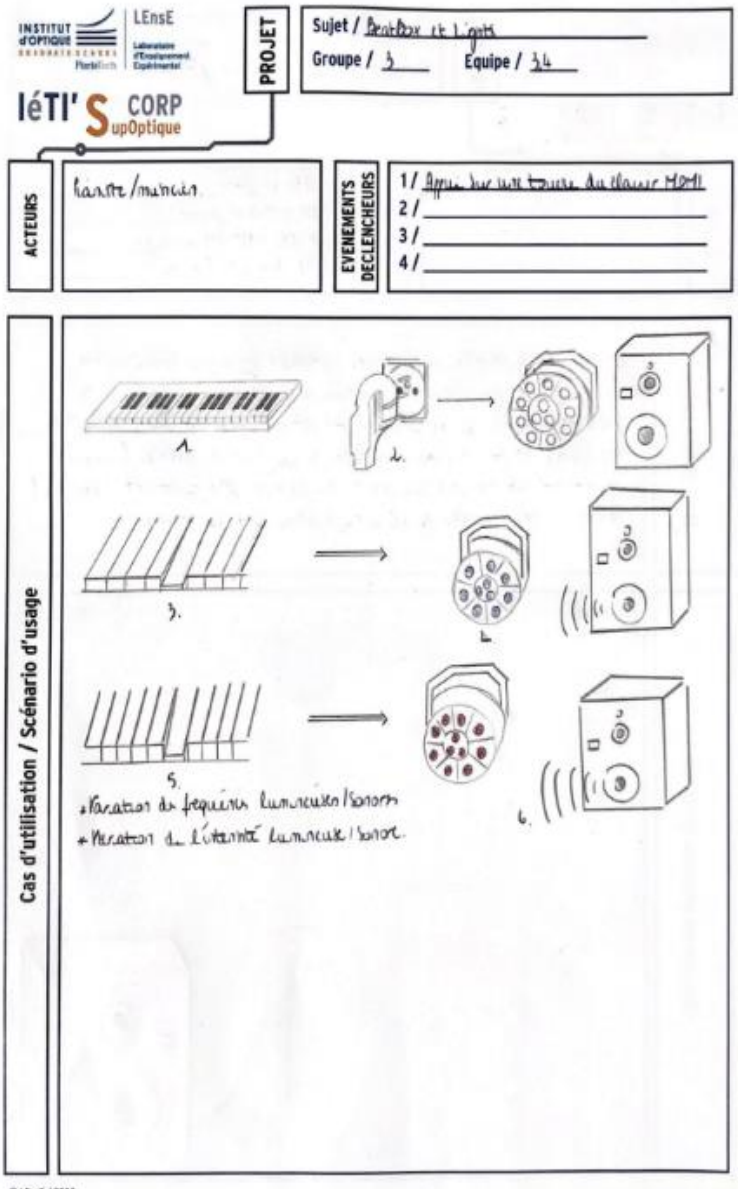


Plan de formation :

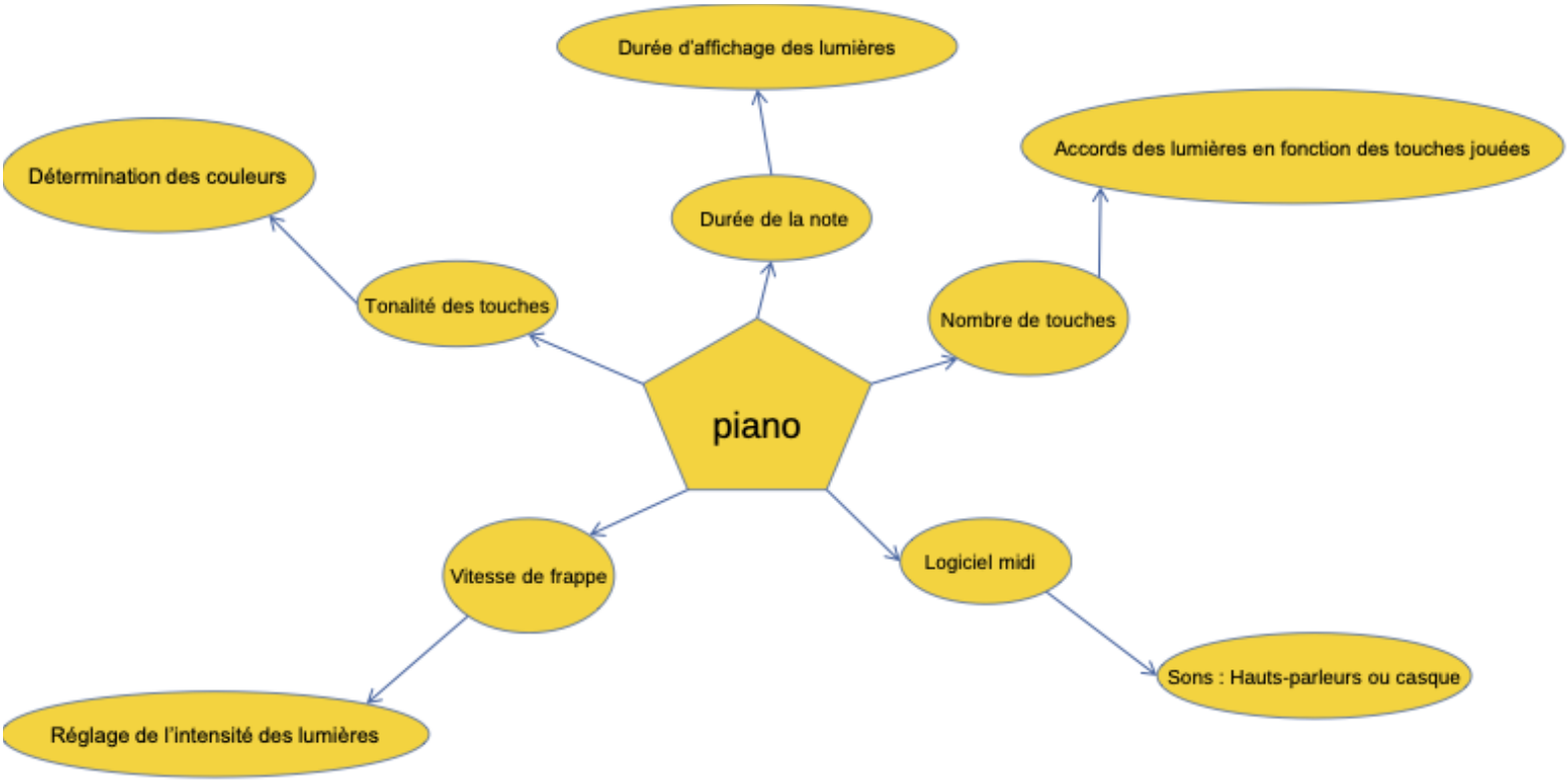
Arno	Ferdinand	Janusan	Jeanne
Créer un projet sur MBED	Tester les programmes sur Nucléo	Déboguer son programme et utiliser l'affichage série ?	Trouver des informations dans la documentation
Connecter une source sonore		Générer une tension analogique ?	Récupérer une information numérique
Faire une action instantanément lors d'un évènement externe			
Faire communiquer deux systèmes	Configurer une communication MIDI	Interfacer une liaison MIDI/DMX	Echanger des données entre un PC et un microcontrôleur	Traitement numérique

On a réalisé la division du travail à travers cette première division des compétences lors des deux premières séances pour être efficace, on a cependant aussi beaucoup travaillé ensemble quand on en a eu besoin. Plan détaillé des séances dans la partie **V**.





II. Découpage fonctionnel



On a du faire des choix pour le fonctionnement de notre système.

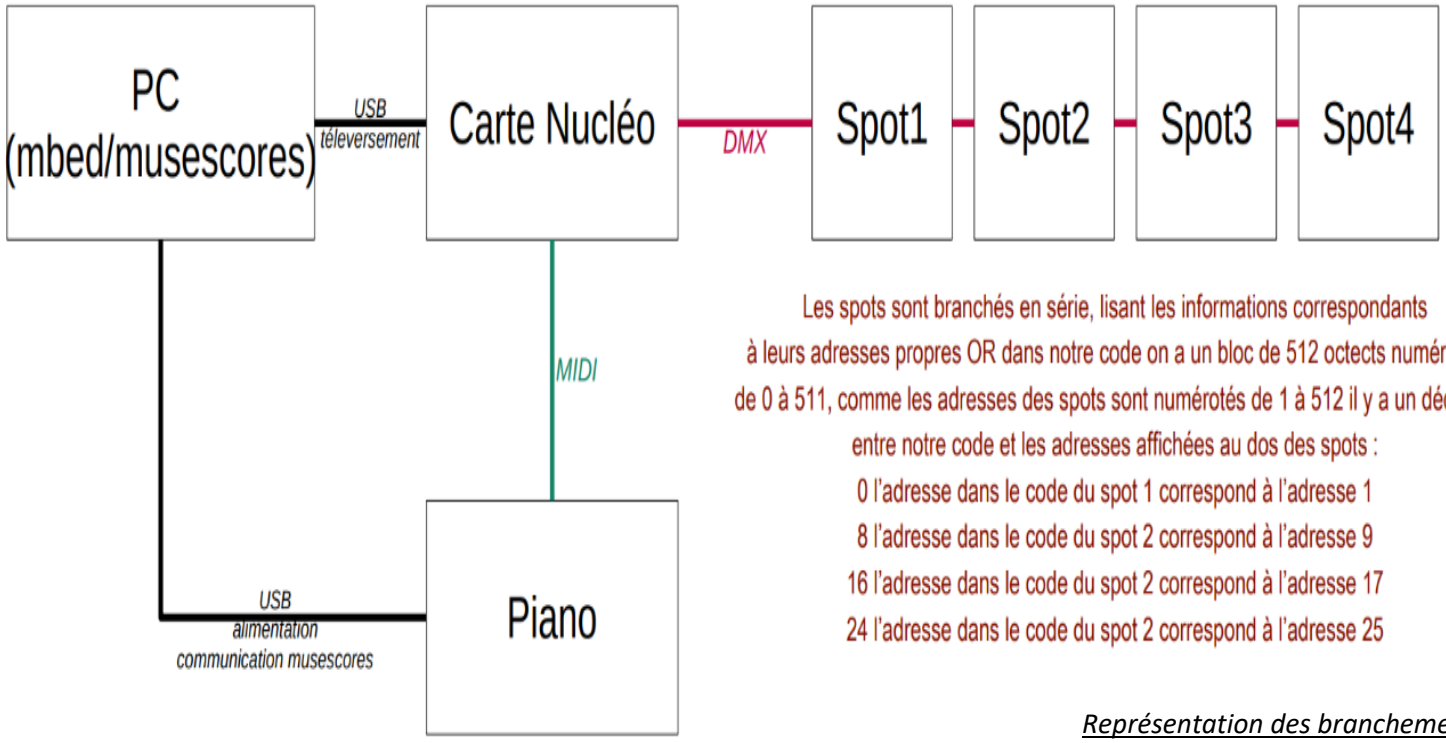
Pour le son, on a choisit d'utiliser un logiciel MIDI, c'est très accessible et très souple, on peut choisir différents types de sons, utiliser des plug-ins VST, il suffit de brancher l'instrument (piano) à un ordinateur disposant du logiciel, pour notre part, le logiciel Musescore3.

Pour l'intensité lumineuse, on a étudié les principales possibilités de fonctionnement, comme par exemple avec les spots constamment allumés qui changent à chaque touches jouées, nous avons choisit de faire varier l'intensité lumineuse avec la vélocité des notes (force d'appui des touches).

Pour déterminer les couleurs, on a choisit 12 couleurs (pour les 12 demi-tons composant un octave), variées entre elles mais avec un dégradé de sorte qu'on ait une proximité entre les deux couleurs de deux touches voisines.

Pour jouer simultanément plusieurs touches, on a choisit de faire des accords de couleurs, à un nombre de touches appuyées (entre 1 et 4) correspond un mode des spots :

Pour la durée d'affichage des spots, on a choisit qu'ils s'éteignent une fois la touche retirée, avec une 'extinction' lumineuse, c'est à dire que l'intensité diminue progressivement après avoir retiré la touche, on a choisit une décroissance exponentielle qui est plus naturelle et rapide qu'une décroissante linéaire.

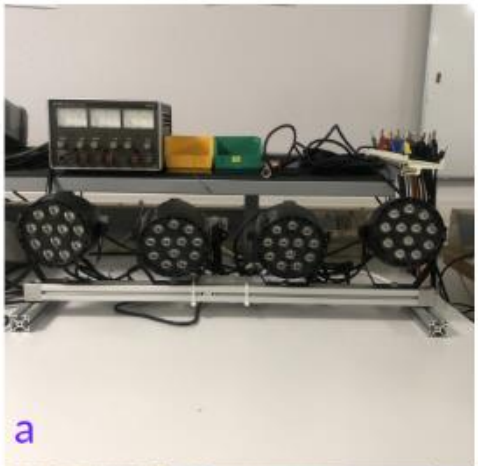


Représentation des branchements

III. Réalisation du prototype

1. Matériel utilisé

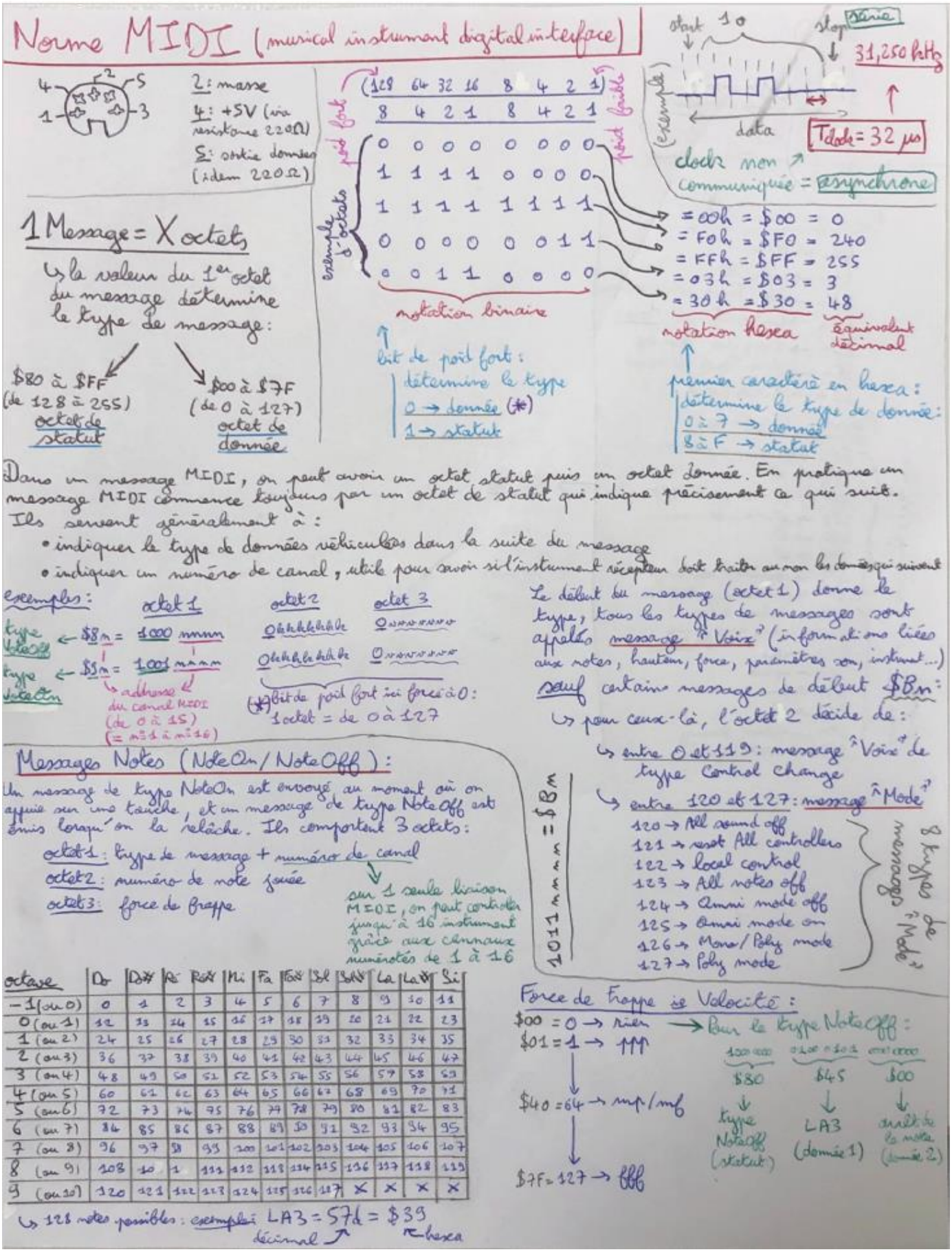
a. LEDs
b. Clavier MIDI / Ordinateur
c. Logo du logiciel MuseScore
d. Logo du logiciel MBED
e. Carte Nucléo
f. Enceintes branchées à l'ordinateur



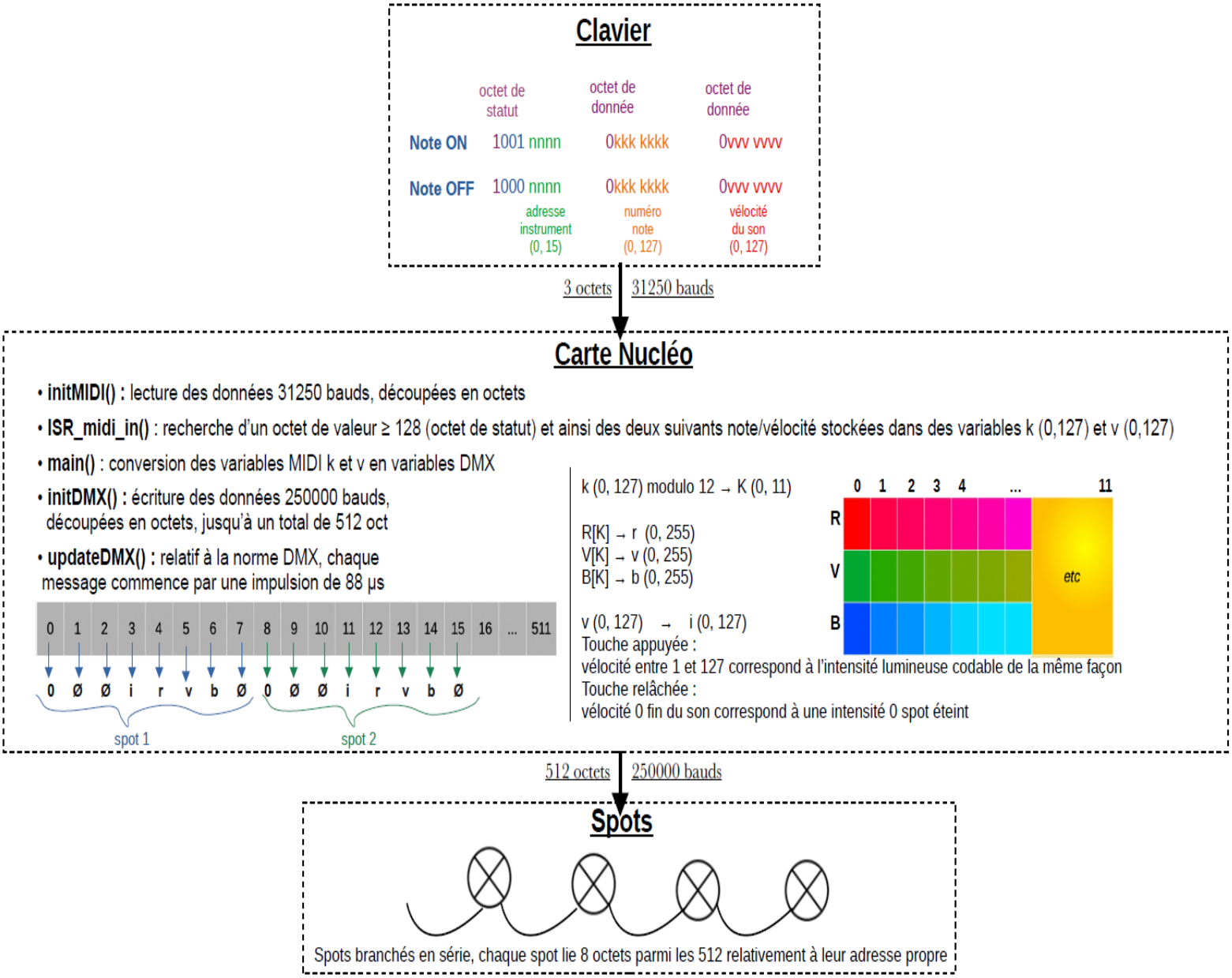
musescore c

mbed d

2. Explication de la norme MIDI



3. Interface MIDI -> DMX



En appuyant sur une touche, le clavier envoie 3 octets selon la norme MIDI à 31250 Bauds. Une première fonction va lire les données reçu par la nucléo, et une autre va rechercher les octets contenants les valeurs utiles (numéro de la touche et sa vélocité).
On converti ces variables MIDI en variables DMX, c'est-à-dire en 3 variables RVB et en 1 variable d'intensité lumineuse.
Enfin une autre fonction va écrire un nouveau message DMX dans la bonne procédure et à 250000 Bauds. Il s'agit de 512 octets dont seuls 8 fois le nombre de spots sont utiles (32 octets) qui sera lu indépendamment pour chaque spot.

IV. Tests et validation du projet

Essaies vidéos

https://
www.youtube.com/
watch?v=itdnAFrneY4

https://www.youtube.com/watch?v=itdnAFrneY4

IETI

V. Comprendre les étapes de réalisation

1. Rétroplanning

Séance 1 : 19.01.2022
Découverte du système, imagination du projet. Lors de cette séance, nous avons complété une fiche descriptive du schéma de principe et du scénario d'usage. Nous avons principalement fait de l'étude de documents. Nous avons créé un groupe sur la plateforme Trello afin de répertorier les documents importants.

Séance 2 : 26.01.2022
Durant cette séance, nous avons établi un schéma fonctionnel, un plan de formation ainsi qu'un cahier des charges. Nous avons poursuivi l'étude de documents et nous sommes intéressés au code.

Séance 3 : 16.02.2022 PAD
Lors de cette séance, nous avons réalisé le montage imaginé pour notre projet. Nous avons rencontré quelques difficultés lors des branchages via la carte Nucléo pour contrôler les LED. Nous nous sommes intéressés au code en détail et avons commencé à l'adapter.

Séance 4 : 02.03.2022
Avancée du code informatique, mise en fonctionnement du système. Durant cette séance, nous nous sommes questionnés quant à la cohérence entre les notes et la lumière (choix des couleurs, de l'intensité lumineuse en fonction de la force de frappe...). Si on appuie sur une seule touche, toutes les lampes renvoient la même couleur. Si on appuie sur plusieurs touches simultanément, les lampes affichent des couleurs différentes (une lampe est associée à une note) selon l'ordre de note croissant.
Nous partions pendant cette séance sur une extinction immédiate des lumières après relâchement, chose que nous avons changé à la séance suivante pour un effet plus agréable.

Séance 5 : 21.03.2022 PAD
Difficultés rencontrées : Les lumières restantes après avoir relâché une note ne correspondait pas à la note que l'on maintenait enfoncée. Mises-en forme des accords sur le clavier midi.

Séance 6 : 28.03.2022 PAD

Finalisation du code information et réflexions sur l'aspect artistique de notre système (choix des
couleurs, cohérence des lumières lorsque l'on joue un accord etc.). Utilisation de la plateforme Musescore : jusqu'où pouvons-nous nous en servir ? On a tenté plusieurs instruments, d'ajuster les lumières en fonctions de ceux-ci mais en vain. On s'en est donc servi exclusivement pour transformer le signal midi en signal sonore.

Séance 7 : 11.04.2022
Adaptation du code informatique au PAD, tests effectués sur cette nouvelle interface. Avancement de
la présentation du projet IeTi. Prise de photos et vidéos à l'appui.

Séance 8 : 16.05.2022
Présentation finale du projet.

2. Difficultés rencontrées

- Modifier les adresses et prendre en compte qu'il fallait introduire un décalage de huit adresses entre chaque notes : on a eu du mal à comprendre comment il fallait noter ces adresses.
- Lorsque deux touches étaient maintenues appuyées simultanément cela déclenchait une extinction de toutes les lumières.
- Avoir les quatre lumières allumées différemment : la carte prenait en compte uniquement la dernière note.
- Lorsqu'on enlevait une note les lumières restaient de la mauvaise couleur (cf partie précédente)
- Deux notes appuyées simultanément ne prenaient en compte qu'une seule note.

3. Pistes d'amélioration :

- Faire un script / une interface pour demander à l'utilisateur de choisir lui-même les douze couleurs
- Faire un programme généralisant aux cas de N spots pour pouvoir en mettre beaucoup plus lors de gros évènements
- Faire un programme pouvant convenir à la fois au pad et au piano

V. Annexe

Programme de la carte sous Mbed

```
/
**********************************************************************
**********/
/*  Test DMX512                                                 */
/
**********************************************************************
**********/
/*  LEnsE / Julien VILLEMEJANE       /   Institut d'Optique
Graduate School */
/
**********************************************************************
**********/
/*  Brochage
*/
/*      TO COMPLETE
*/
/
**********************************************************************
**********/
/*  Test réalisé sur Nucléo-L476RG
*/
/
**********************************************************************
**********/

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

const       uint8_t     scriabin_r[12] = {255, 255, 255, 255, 0,
9, 23, 49, 21, 27, 102, 255};
const       uint8_t     scriabin_g[12] = {0, 73, 127, 215, 255,
106, 101, 140, 96, 1, 0, 0};
const       uint8_t     scriabin_b[12] = {0, 1, 0, 0, 0, 9, 125,
231, 189, 155, 153, 127};

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

#define SAMPLES     512
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    CV_volume(PC_1);
AnalogIn    CV_pitch(PB_0);

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;

char memoire; // voir dans le programme ISR_midi_in tout en bas

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

void fin();

int cpttouches = 0; // compte le nb de touches appuyées, lorsque
egale a 0 signifie qu'on a zero touche
                    // exemple : si egale 3 alors c qu'on appuie
sur 3 touches etc
int cptavant = 0;

int L[4] = {0};

// Main
int main() {
    debug_pc.baud(115200);
    debug_pc.printf("Essai DMX512\r\n");

    initDMX();
    initMIDI();

    int i = 0;

    while(1) {
    if(isNoteMIDIdetected()){

        if((cptavant <= cpttouches)&&cpttouches<5){
            debug_pc.printf("oui\r\n");
            L[i%4] = note_data%12 ;
            i++ ;}
        debug_pc.printf("L = %d\t%d\t%d\t%d\r\n",
L[0],L[1],L[2],L[3]);

        if(cpttouches==1){
            debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data,
note_data, velocity_data);

            // Renkforce LPT12 - AD 1
            dmx_data[0+0] = 0;  // Spot 1 - Mode
            dmx_data[0+3] = velocity_data;  // Spot1 - Intensité
            dmx_data[0+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[0+5] = scriabin_g[L[(i-1)%4]];
```

```
            dmx_data[0+6] = scriabin_b[L[(i-1)%4]];

            // Renkforce LPT12 - AD 1
            dmx_data[8+0] = 0;  // Spot 2 - Mode
            dmx_data[8+3] = velocity_data;   // Spot 2 - Intensité
            dmx_data[8+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[8+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[8+6] = scriabin_b[L[(i-1)%4]];


            dmx_data[16+0] = 0;  // Spot 3 - Mode
            dmx_data[16+3] = velocity_data;   // Spot 3 -
Intensité
            dmx_data[16+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[16+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[16+6] = scriabin_b[L[(i-1)%4]];


            dmx_data[24+0] = 0;  // Spot 4 - Mode
            dmx_data[24+3] = velocity_data;   // Spot 4 -
Intensité
            dmx_data[24+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[24+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[24+6] = scriabin_b[L[(i-1)%4]];
            new_note_midi = 0;
        }
        if(cpttouches==2){
            debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data,
note_data, velocity_data);
            // Renkforce LPT12 - AD 1
            dmx_data[0+0] = 0;  // Spot 1 - Mode
            dmx_data[0+3] = velocity_data;  // Spot1 - Intensité
            dmx_data[0+4] = scriabin_r[L[(i-2)%4]];
            dmx_data[0+5] = scriabin_g[L[(i-2)%4]];
            dmx_data[0+6] = scriabin_b[L[(i-2)%4]];

            // Renkforce LPT12 - AD 1
            dmx_data[8+0] = 0;  // Spot 2 - Mode
            dmx_data[8+3] = velocity_data;   // Spot 2 - Intensité
            dmx_data[8+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[8+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[8+6] = scriabin_b[L[(i-1)%4]];


            dmx_data[16+0] = 0;  // Spot 3 - Mode
            dmx_data[16+3] = velocity_data;   // Spot 3 -
Intensité
            dmx_data[16+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[16+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[16+6] = scriabin_b[L[(i-1)%4]];


            dmx_data[24+0] = 0;  // Spot 4 - Mode
            dmx_data[24+3] = velocity_data;   // Spot 4 -
Intensité
            dmx_data[24+4] = scriabin_r[L[(i-2)%4]];
            dmx_data[24+5] = scriabin_g[L[(i-2)%4]];
            dmx_data[24+6] = scriabin_b[L[(i-2)%4]];
            new_note_midi = 0;
        }
```

```
        if(cpttouches==3){
            debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data,
note_data, velocity_data);
            // Renkforce LPT12 - AD 1
            dmx_data[0+0] = 0;  // Spot 1 - Mode
            dmx_data[0+3] = velocity_data;  // Spot1 - Intensité
            dmx_data[0+4] = scriabin_r[L[(i-3)%4]];
            dmx_data[0+5] = scriabin_g[L[(i-3)%4]];
            dmx_data[0+6] = scriabin_b[L[(i-3)%4]];

            // Renkforce LPT12 - AD 1
            dmx_data[8+0] = 0;  // Spot 2 - Mode
            dmx_data[8+3] = velocity_data;   // Spot 2 - Intensité
            dmx_data[8+4] = scriabin_r[L[(i-2)%4]];
            dmx_data[8+5] = scriabin_g[L[(i-2)%4]];
            dmx_data[8+6] = scriabin_b[L[(i-2)%4]];


            dmx_data[16+0] = 0;  // Spot 3 - Mode
            dmx_data[16+3] = velocity_data;   // Spot 3 -
Intensité
            dmx_data[16+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[16+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[16+6] = scriabin_b[L[(i-1)%4]];


            dmx_data[24+0] = 0;  // Spot 4 - Mode
            dmx_data[24+3] = velocity_data;   // Spot 4 -
Intensité
            dmx_data[24+4] = scriabin_r[L[(i-3)%4]];
            dmx_data[24+5] = scriabin_g[L[(i-3)%4]];
            dmx_data[24+6] = scriabin_b[L[(i-3)%4]];
            new_note_midi = 0;
        }
        if(cpttouches>=4){
            debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data,
note_data, velocity_data);
            // Renkforce LPT12 - AD 1
            dmx_data[0+0] = 0;  // Spot 1 - Mode
            dmx_data[0+3] = velocity_data;  // Spot1 - Intensité
            dmx_data[0+4] = scriabin_r[L[i%4]];
            dmx_data[0+5] = scriabin_g[L[i%4]];
            dmx_data[0+6] = scriabin_b[L[i%4]];

            // Renkforce LPT12 - AD 1
            dmx_data[8+0] = 0;  // Spot 2 - Mode
            dmx_data[8+3] = velocity_data;   // Spot 2 - Intensité
            dmx_data[8+4] = scriabin_r[L[(i-3)%4]];
            dmx_data[8+5] = scriabin_g[L[(i-3)%4]];
            dmx_data[8+6] = scriabin_b[L[(i-3)%4]];


            dmx_data[16+0] = 0;  // Spot 3 - Mode
            dmx_data[16+3] = velocity_data;   // Spot 3 -
Intensité
            dmx_data[16+4] = scriabin_r[L[(i-2)%4]];
            dmx_data[16+5] = scriabin_g[L[(i-2)%4]];
            dmx_data[16+6] = scriabin_b[L[(i-2)%4]];
```

```
            dmx_data[24+0] = 0;  // Spot 4 - Mode
            dmx_data[24+3] = velocity_data;   // Spot 4 -
Intensité
            dmx_data[24+4] = scriabin_r[L[(i-1)%4]];
            dmx_data[24+5] = scriabin_g[L[(i-1)%4]];
            dmx_data[24+6] = scriabin_b[L[(i-1)%4]];
            new_note_midi = 0;
        }
    }
        // au moment on on relache TOUTES les touches
(cpttouches=1) on veut eteindre de maniere
        // progressive donc on va faire baisser velocity_data
jusqu'a 0 (condition nécessaire !!)
        if(cpttouches==0 && velocity_data!=0){

            i = 0;

            velocity_data = velocity_data/1.5; // ****decroissance
exponentielle****

        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] = 0;  // Spot 1 - Mode
            dmx_data[0+3] = velocity_data;  // Spot1 - Intensité

            // Renkforce LPT12 - AD 1
            dmx_data[8+0] = 0;  // Spot 2 - Mode
            dmx_data[8+3] = velocity_data;   // Spot 2 - Intensité

            dmx_data[16+0] = 0;  // Spot 3 - Mode
            dmx_data[16+3] = velocity_data;   // Spot 3 -
Intensité

            dmx_data[24+0] = 0;  // Spot 4 - Mode
            dmx_data[24+3] = velocity_data;   // Spot 4 -
Intensité

            new_note_midi = 0;

            // il faut récrire tout ça dans le main comme ça si
l'extinction commence et qu'on appuie de nouveau
            // sur une touche, alors ça interromp l'extinction
            // (EXTINCTION = decroissance de la velocity -> ça
donne de la dynamique)
            }
        cptavant = cpttouches;
        updateDMX();
    }
}


/* 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){ //sur le
clavier MIDINOTEOFF existe pas il faut tester la valeur de
velocity_data (si nulle ou non)
            new_note_midi = 1;
            channel_data = midi_data[0] & 0x0F;
            velocity_data = midi_data[2];
            if(velocity_data==0){ // NOTE OFF
            // le test est effectué apres que la velocity est
nulle mais on veut pas qu'elle soit nulle sauf si cpttouches=1
            // on va donc réattribuer la valeur de velocity juste
avant que ce declenche le NOTEOFF grace à "memoire"
                cpttouches = cpttouches - 1;
```

```
                channel_data = midi_data[0] & 0x0F;
                velocity_data = memoire;
                if(cpttouches==0){ // on veut eteindre la lumiere
pas quand on relache une touche mais quand on relache la derniere
touche appuyée
                    // vide car on a fait l'extinction dans le
MAIN !
                    }
                debug_pc.printf("cpt=%d\r\n", cpttouches);
                }
            else{ // NOTE ON
                note_data = midi_data[1];
                cpttouches = cpttouches + 1;
                new_note_midi = 1;
                debug_pc.printf("cpt=%d\r\n", cpttouches);
                memoire = velocity_data; // garde l'information du
velocity avant que ça se coupe
            }
        }
        else{
            if(midi_data[0] == MIDI_CC){
                new_data_midi = 1;
                control_ch = midi_data[1];
                control_value = midi_data[2];
            }
        }
    }
}
```