Romane Dorino
Kyliann Robert
Justin Dovillaire
Adrien Maltese
Aliénor Braley

Groupe 3
8 avril 2022

Rapport technique

Application pour pilotage DMX/MIDI

Le 26 janvier 2022, la société SOLEC nous a confié une mission relative au pilotage DMX/MIDI de projecteurs.
Dans le monde de l'événementiel, on 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 des jeux de lumières, en fonction d'un timing choisi 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 interface MIDI/DMX afin de s'affranchir d'un ordinateur pour la gestion des jeux de lumière.
Ce projet pourra être décomposé en deux sous-ensembles :

- une interface matérielle de conversion MIDI vers DMX ;
- une interface matérielle de conversion signal analogique vers DMX ;
- une interface logicielle pour la programmation ;

L'objet de ce rapport technique est d'expliquer en détails la démarche qui nous a permis de mener à bien ce projet, et de détailler les fonctionnalités développées. Il s'agira de décrire le projet et de présenter un schéma fonctionnel ainsi qu'un cahier des charges. Nous passerons en revue la démarche utilisée pour aboutir au prototype final. Enfin, ce rapport est l'occasion de faire un bilan sur ce projet d'un point de vue technique, avec entre autres les problèmes rencontrés et les solutions apportées, et sur les compétences acquises par chacun et par le groupe.

Table des matières

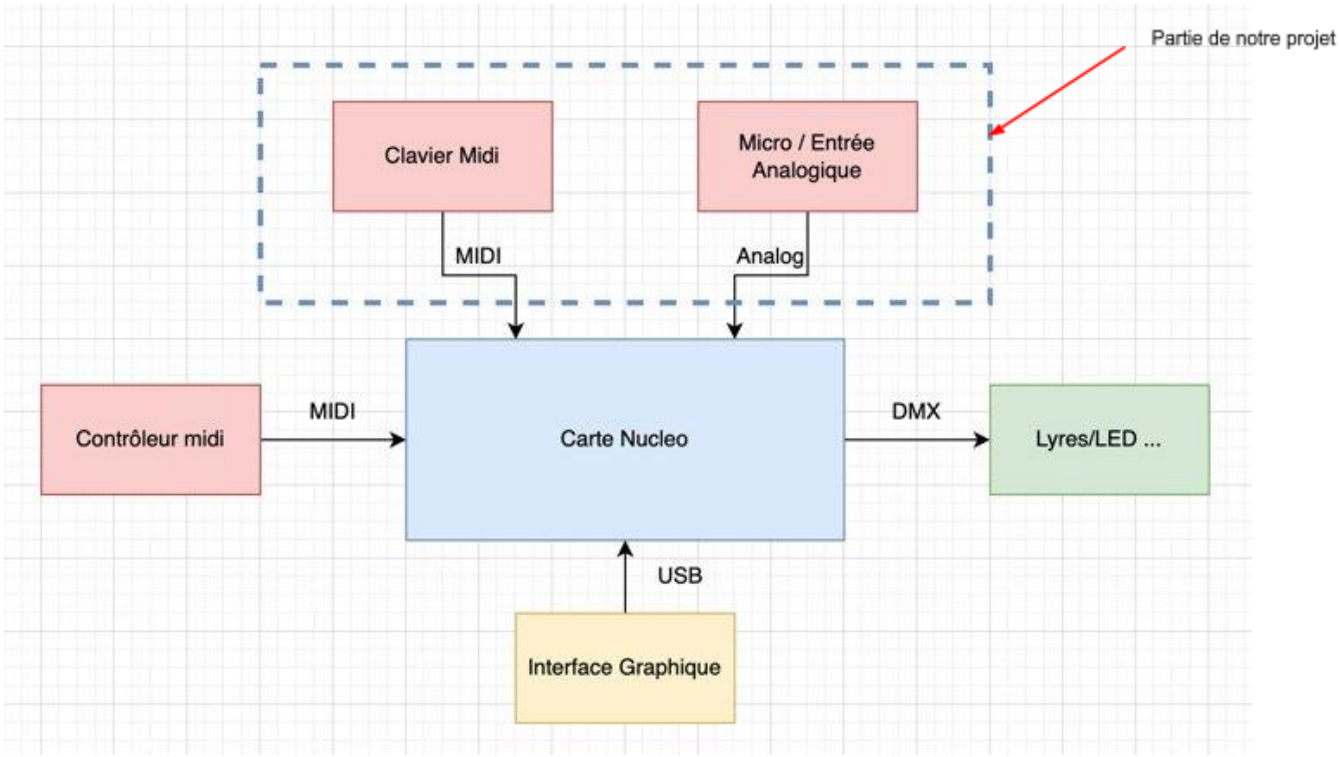
1/ Vue générale du projet

a) Objectifs et cahier des charges

Le but est de pouvoir faciliter l'usage de projecteurs pilotables dans le monde de l'événementiel. Par exemple, lors d'une soirée, on veut pouvoir adapter facilement le rythme d'une musique avec la lumière que l'on envoie, et également le type de lumière que l'on choisit d'envoyer. Les objectifs du projet sont donc de :

- effectuer le pilotage de lampes au format DMX à partir d'un clavier au format MIDI ;
- réaliser le pilotage des lampes à partir d'un signal analogique ;

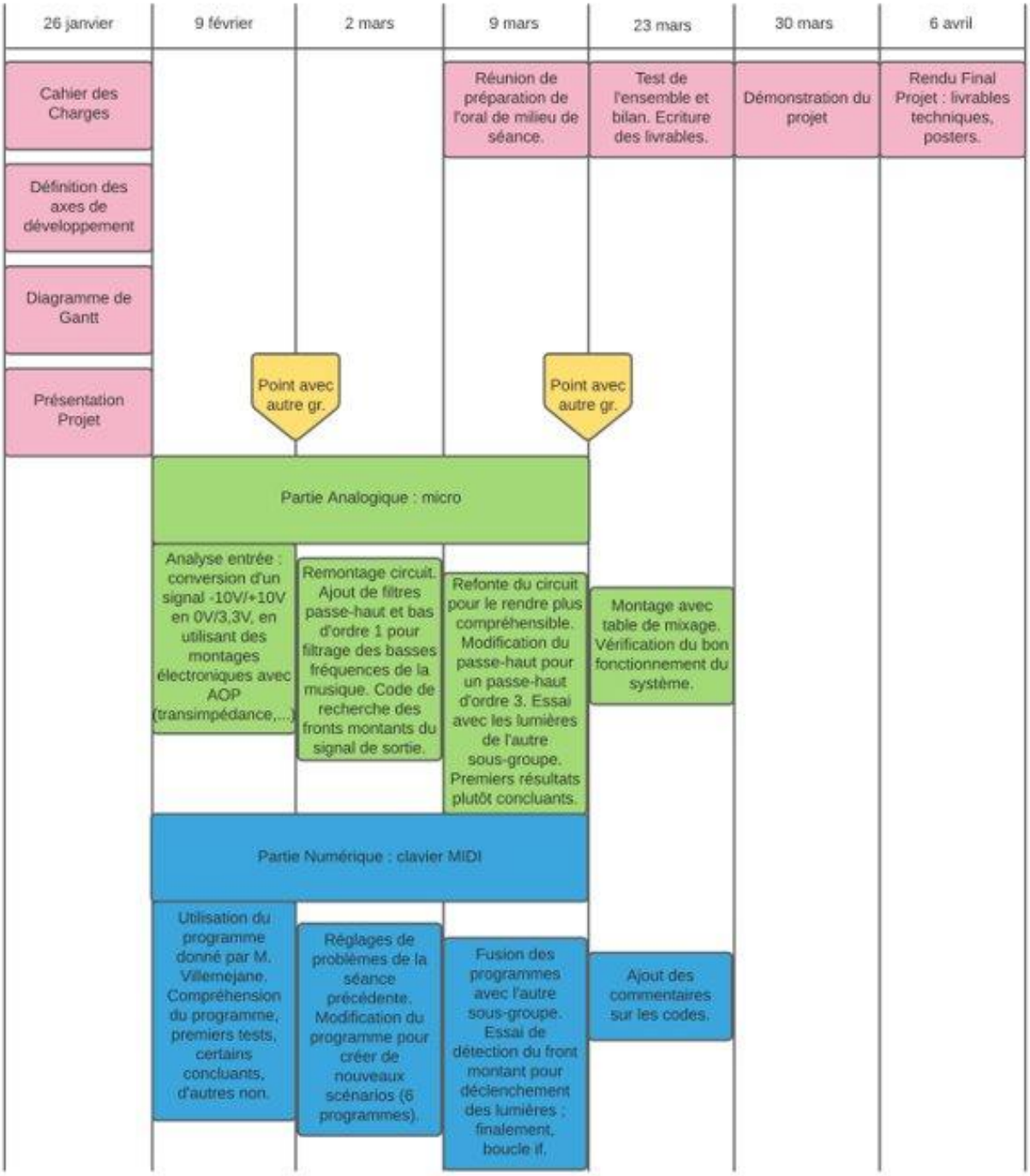
Nous décrirons les problèmes auxquels nous avons dû faire face et comment nous y avons remédié. Les programmes des différentes parties ainsi que les schémas de montage qui ont été utilisés seront décrits et commentés.



b) Organisation du groupe

Le projet général traitant d'aspects techniques différents, a été partagé en deux groupes : notre groupe a décidé de se concentrer sur la partie qui gère l'interface clavier MIDI/DMX, ainsi que l'envoi d'un signal analogique. Comme indiqué sur le schéma ci-après, on se concentre donc sur la partie encadrée.

D’une part, on souhaite pouvoir piloter la lumière des projecteurs (autant l’intensité que la couleur) quand on appuie sur les touches d’un clavier synthétique de piano, et d’autre part on veut changer les lumières dès qu’il y a le son d’une basse dans la musique écoutée.
La partie sur le pilotage des projecteurs a été réalisée majoritairement par Romane et Aliénor et la partie sur la récupération des basses par voie analogique a été réalisée par Kyliann, Adrien et Justin. Vers la fin du projet où il était nécessaire de fusionner les 2 parties, les deux sous-équipes se sont rassemblées pour travailler ensemble.
La répartition du travail, à posteriori, peut être résumée dans le diagramme de Gantt suivant :



2/ Partie analogique

Dans le but de pouvoir capter les “kicks” (basses) des musiques lors des soirées, nous avons choisi une solution associant traitement de signal et échantillonnage numérique. Nous traiterons ici du traitement analogique que nous appliquons au signal d’entrée qui est une musique en sortie de table de mixage (Pioneer DJM-700) :

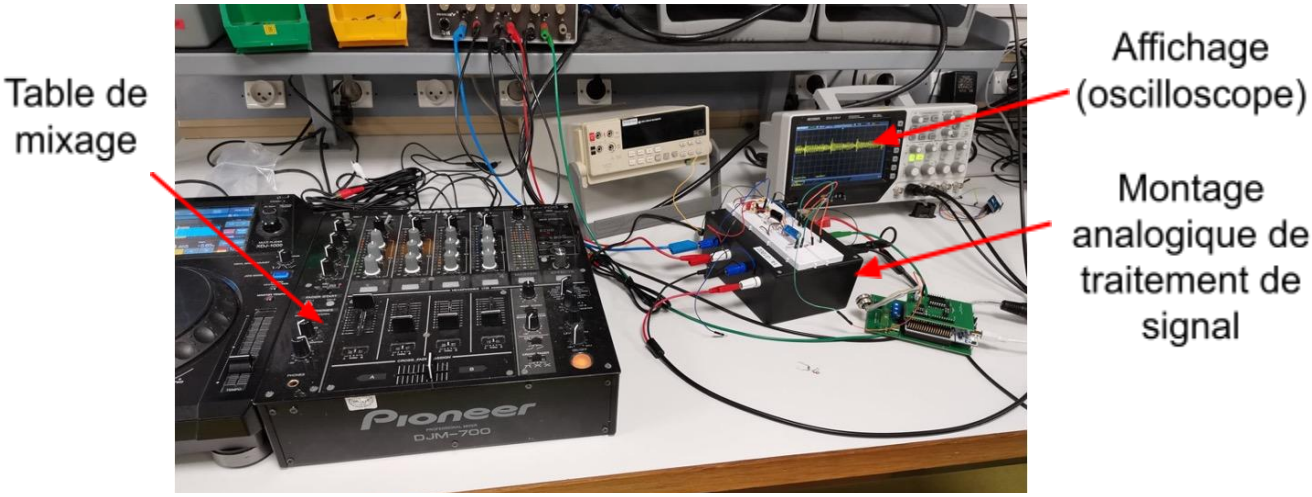


Photo du montage complet.

Après des premières mesures de tension en sortie de table de mixage à pleine puissance, nous avons mesuré un voltage pic à pic de 20V. La carte Nucléo acceptant uniquement une tension comprise entre 0 et 3.3V, il faut prévoir un étage d’amplification adapté pour pouvoir sécuriser la carte. De plus, nous souhaitons conserver uniquement les fréquences caractéristique du tempo de la musique.

Pour cela, nous avons analysé les musiques via l'oscilloscope et avons déterminé que la bande de fréquence intéressante à conserver est autour de 170 Hz. En dessous de cette valeur, on retrouve composantes correspondant plus à des vibrations. Au-dessus de cette valeur, on commence à capter certains instruments comme des pianos par exemple qui ne suivent pas le tempo. Nous n’avons pas pris le temps d’analyser logiciellement le spectre des musiques par manque de temps. Sélectionner une fréquence plus précise fait partie des améliorations possibles.

Étant donné que le filtrage atténue le signal d’entrée, il est important de faire le filtrage directement en sortie de la table de mixage. On peut ensuite mesurer la tension en sortie de filtre de manière à déterminer l’amplification nécessaire par la suite pour protéger la carte. On arrive à l’architecture suivante :

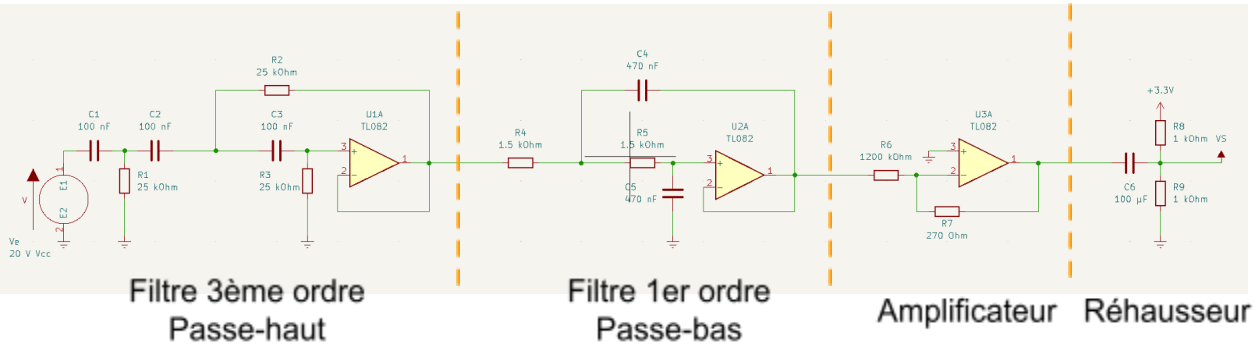
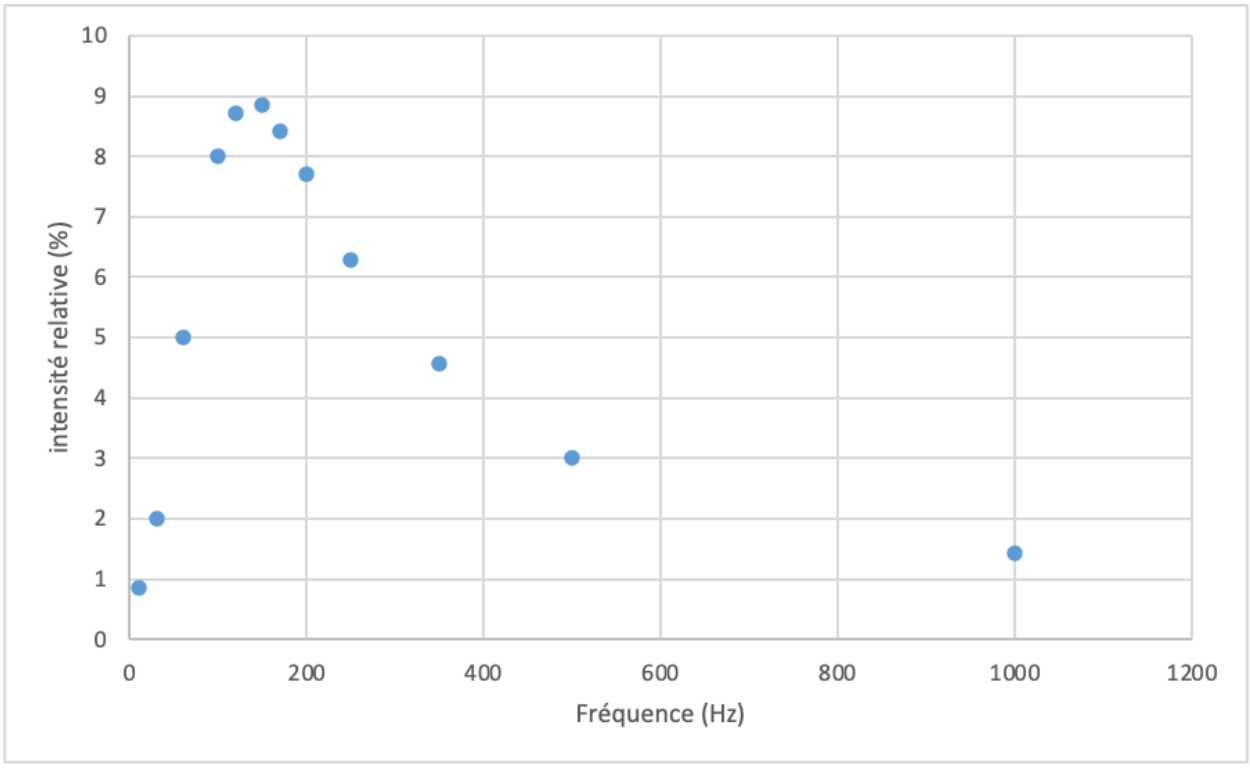


Schéma électronique de traitement du signal.

Étant donné que nous souhaitons couper au maximum les fréquences de vibration (autour de 60 Hz), nous utilisons un filtre passe haut actif du 3ème ordre afin d'avoir une pente de gain de -60dB/décade et de diminuer suffisamment l'amplitude de ces vibrations. D'autre part, nous utilisons un filtre passe bas actif d'ordre 1 pour couper les hautes fréquences.

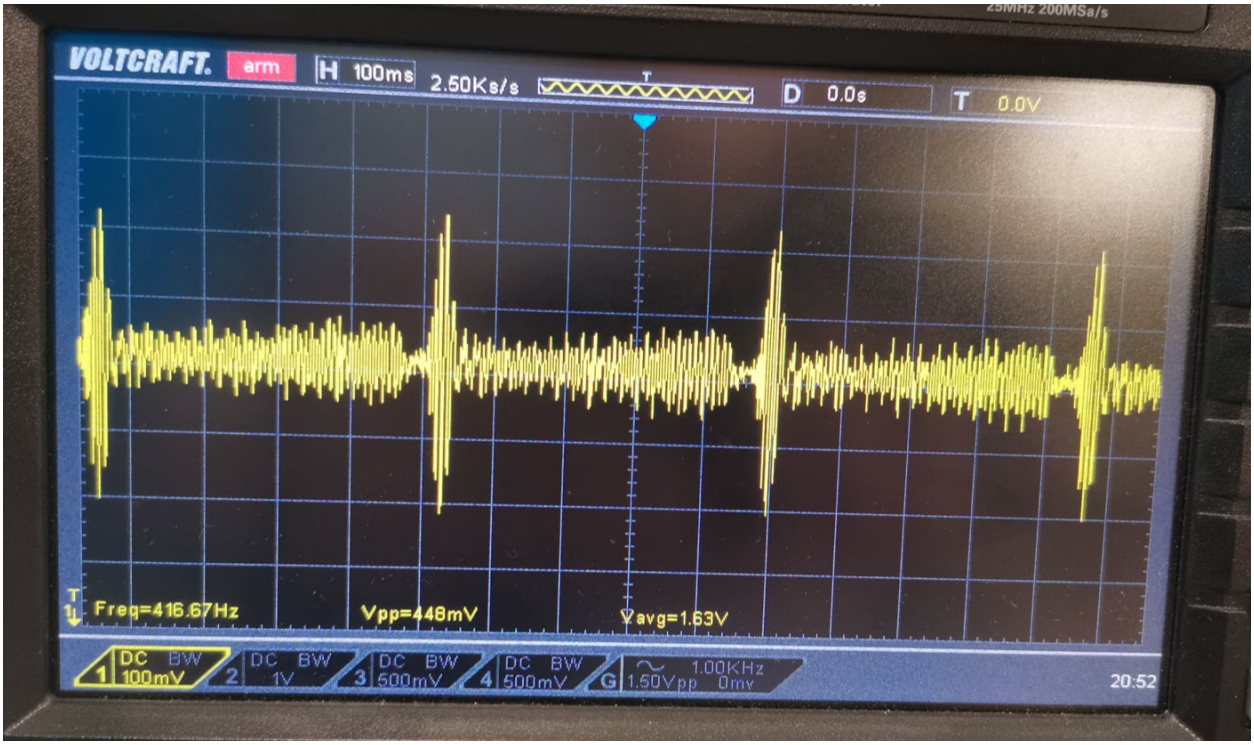
Nous avons pris une série de points de mesure de tension en sortie du filtre en fonction de la fréquence du signal d'entrée (en utilisant un GBF comme source). Nous avons obtenu le diagramme suivant :



Sur ce diagramme, on peut lire en ordonnée l'amplitude relative de tension par rapport à la tension en entrée en pourcentage. Via le filtre utilisé, on observe bien un pic d'amplitude vers 170 Hz. On atténue

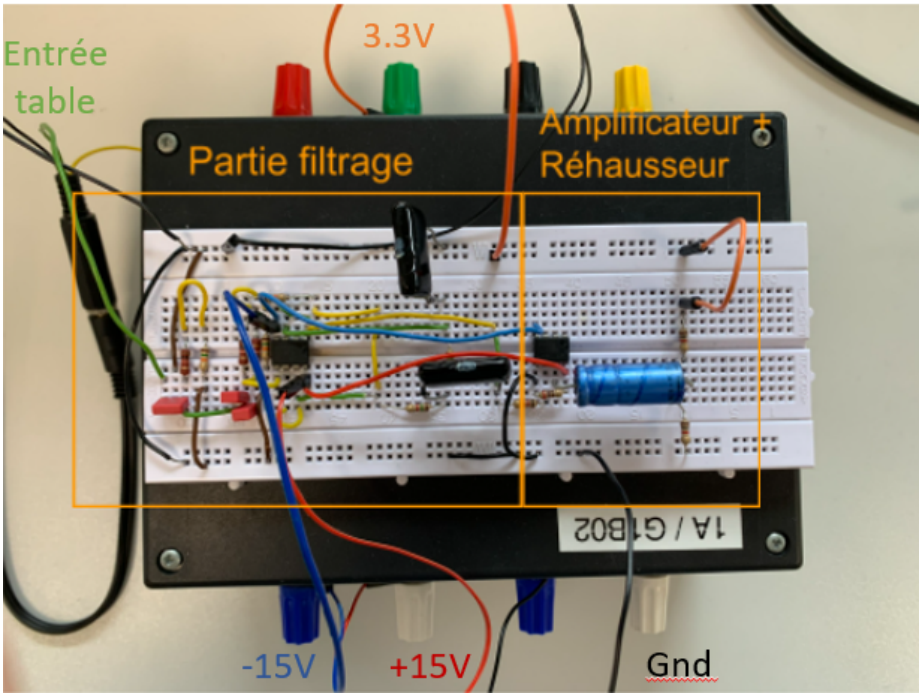
très correctement les fréquences supérieures à 500 Hz et inférieures à 40 Hz. Cependant, on note que l'on coupe uniquement environ 50% des composantes à 60 Hz, ce qui sera problématique pendant la phase de détection de tempo sur Nucléo.

Nous utilisons ensuite un amplificateur non inverseur ainsi qu'un réhausseur de manière à conditionner le signal pour la Nucléo entre 0 et 3,3 V. On notera que le système n'est pas parfaitement protégé pour la carte. En effet, notre système d'amplification est prévu pour la Pioneer DJM-700 mais pour une autre table (signal d'entrée). Une solution idéale serait l'utilisation de diode Zener.



Signal en sortie du système (Animals, Martin Garrix Drop)

En résultat, le signal est beaucoup plus clair pour être exploité numériquement. Les "kicks" de la musique testée sont bien visibles. Cependant, ce résultat était plus ou moins convaincant en fonction des musiques testées. On note la persistance d'un signal entre les "kicks". Il faudrait réaliser une analyse plus poussée des musiques et faire un traitement de signal plus adapté en conséquence pour gagner en performance.



Montage final sur plaque à trou.

3/ Partie sur le signal MIDI/DMX

Photo du matériel utilisé pour la partie signal MIDI/DMX.

On rappelle que le but de cette partie est de changer la couleur des projecteurs quand on appuie sur une touche de clavier. Autrement dit, il faut récupérer un signal MIDI du piano, l'analyser, et envoyer la couleur associée à cette note vers les projecteurs en la convertissant en DMX.

Sachant que dans cette partie on veut piloter des LED en format DMX à partir d'un clavier qui est en format MIDI, il est nécessaire de relier les deux formats à l'aide d'une carte d'extension que l'on a ajoutée à la carte Nucléo. Cette carte nécessite d'avoir deux embouts femelle (un MIDI et un DMX) qui sont respectivement reliés au clavier avec un câble MIDI mâle-mâle et aux lumières avec un câble femelle-femelle (avec un adaptateur femelle-mâle). Chaque lumière d'une rampe est reliée en série par un câble DMX (fourni).

La première étape de cette partie a été de comprendre le fonctionnement de chaque élément. Nous avons donc effectué des recherches sur les projecteurs, le schéma de câblage, les pattes de la carte qui nous intéressent pour le projet, et sur les protocoles MIDI et DMX.

Comme on recherche à récupérer un signal MIDI et envoyer un signal DMX, il faut :

- récupérer une entrée MIDI, qui correspond à MIDI_in sur la carte d'extension et D2 sur la Nucléo,

- se servir d'une sortie DMX qui correspond à DMX_Tx sur la carte d'extension et D5 sur la Nucléo.

La carte d'extension utilise un protocole DMX512 qui est une norme de transmission de données pour pilotage d'éclairages. Elle peut contrôler 512 canaux. Sa trame est la suivante : un break à l'état bas de 88 µs, une marque de break à l'état haut de 8 µs, un start code et une succession de 512 données (1 bit de start, 8 bits de données, 2 bits de stop). Chaque bit dure 4 µs et a un débit de 250000 bauds.

L'idée est d'attribuer un nombre de canaux à chaque projecteur, et c'est à travers ces canaux que l'on va transmettre l'information (changement de couleur, d'intensité). La norme utilisée pour la rampe de 4 projecteurs est l'attribution de 7 canaux par projecteur.

L'attribution des canaux se fait donc sur les projecteurs, et nous avons choisi les canaux suivants : 1er projecteur à canal de 001 à 007, 2^{e} projecteur à canal de 009 à 015, 3^{e} projecteur à canal de 017 à 023, 4^{e} projecteur à canal de 025 à 032. Il est important de retenir ces canaux car il faut les préciser dans le programme, c'est ce que nous verrons plus loin.

Sur le LENSE, nous avons récupéré un programme de test pour MBED pour les protocoles DMX et MIDI, pour comprendre comment ces deux formats sont reliés et exploités. Nous avons simplement compilé le programme tel quel, puis en changeant successivement un élément du code, pour comprendre ce que chaque partie du code effectue.

Le code est présenté et documenté en annexe, mais nous allons décrire les problèmes que nous avons eu ainsi que les principales fonctions utiles à connaître pour pouvoir manier le programme.

Nous avons distingué le mode scriabin qui permet d'attribuer à chaque note du clavier une couleur. Pour chaque projecteur, 3 scriabins doivent être définis pour le rouge, le vert et le bleu. Scriabin est un tableau dont les valeurs vont de 0 à 255. Au début, la taille du tableau était 12, et nous avons modifié le programme de sorte qu'il soit égal à 37, comme le nombre de touches du clavier. De cette façon, une couleur est associée à chaque touche. Si on veut une couleur différente sur chaque projecteur, il faut définir 3 scriabin par projecteur.

La façon dont le programme convertit le signal MIDI est la suivante : lorsqu'on appuie sur une touche, on récupère un MIDI number, un entier entre 23 et 59. Cette valeur est convertie en un nombre entre 1 et 37. L'information est ensuite envoyée via DMX vers les LED.

Une fonction updateDMX permet d'actualiser/mettre à jour le programme tous les x ms pour envoyer l'information aux LED.

Dans la boucle principale, on envoie les informations de couleur et d'intensité aux canaux associés via dmx_data[n° du canal], sachant qu'un canal correspond à l'intensité minimale de la LED, un autre à l'intensité maximale, et 3 autres aux couleurs.

On peut créer des scénarios originaux en modifiant l'intensité et les couleurs de chaque LED : dégradés de couleurs, LED clignotantes, variations progressives d'intensité…
Nous avons créé 6 programmes de pilotage des LEDs par les notes du clavier. Par exemple, un programme utilise un ticker pour faire clignoter les LED avec une période définie. Des simples boucles ''if, else'' permettent d'effectuer une variation d'intensité. Cela dit, il pourrait être intéressant d'étendre le champ des possibilités en réalisant plusieurs scénarios et en utilisant TeraTerm pour les proposer à l'utilisateur. De cette façon, il n'y aurait pas besoin de recompiler à chaque fois qu'on veut changer de scénario.

Nous avons rencontré des difficultés par rapport au matériel, parfois certaines LED se mettaient à clignoter intempestivement et être incontrôlables, ce qui nous a forcé à changer certains projecteurs. Dans le programme, nous avons rencontré des difficultés à mettre une couleur différente pour chaque note et pour qu'elle corresponde bien à ce qu'on envoie. Pour cela, nous avons fait des tests où on mettait toutes les notes en rouge et une seule note en vert par exemple. Il y avait un décalage avec la correspondance de la 1ère note avec la première valeur du tableau de couleurs. Comme le nombre des notes fonctionne en modulo 12, il fallait soustraire la valeur de la note obtenue par 12.

4/ Fusion des parties analogique/DMX

L'objectif du projet était de pouvoir utiliser simplement à la fois la partie analogique du projet, et à la fois la partie contrôlée par clavier MIDI. La partie analogique a dû récupérer plusieurs principes de code de la partie DMX, car en effet, une fois que le montage électronique permettait de récupérer les impulsions des basses des musiques, ce signal régulier peut être envoyé de la carte Nucléo aux lampes, sous format DMX. Comme le groupe DMX avait déjà mis en place les protocoles de communication en DMX, il était simple de faire le lien entre le signal en sortie du groupe analogique et les principes des codes du groupe DMX.

En pratique, nous nous sommes servis du premier code présenté en annexe. Ce code permet dans un premier temps d'initialiser les couleurs et intensités des lampes à des valeurs aléatoires. Nous implantons un ticker associé à la fonction funct_detect. Cette fonction regarde périodiquement (ici, toutes les 75 ms) la valeur de la tension ; si elle est supérieure à un certain seuil, ici dans ce code 1 V, alors les quatre lampes changent de couleur, de nouveau aléatoirement. Si le seuil de tension n'est pas dépassé, le code attend et le ticker s'exécute de nouveau, 75 ms après. Nous choisissons une période de déclenchement du ticker de l'ordre de 75 ms, car les signaux musicaux ont souvent un tempo de 150 bpm environ. Cela correspond à environ 2 battements par seconde, soit 2 Hz, donc une période de 500 ms. Il faut donc se placer à une période de déclenchement plus faible que 500 ms ; nous choisissons une période plusieurs fois plus faible pour être certains de détecter chaque kick du signal audio, même si une musique possède un tempo plus rapide que 150 bpm (ce qui peut arriver fréquemment).

Nous devons également utiliser les fonctions d'initialisation et d'actualisation du protocole DMX, initDMX et updateDMX, étant donné que nous utilisons aussi ce protocole.

Nous avons rencontré plusieurs problèmes lors de la fusion de ces deux parties. Initialement, nous souhaitions mettre en place une fonction d'interruption qui se déclencherait avec la détection des fronts montants des kicks. Cependant, une fonction de détection de fronts montants ne fonctionne qu'avec un signal numérique en entrée ; or notre signal était analogique en entrée, il n'était donc pas possible de faire cela. Après la recherche de plusieurs solutions de contournement de ce problème, nous avons donc changé de méthode, et tenté de lancer un ticker à intervalles réguliers, pour être certains de détecter chaque kick. Malheureusement, cette technique n'est pas aussi précise que la précédente, car dans tous les cas, un décalage aura lieu, et tous les kicks ne seront pas détectés ; il peut aussi arriver le cas où le même kick est détecté deux fois, s'il est un peu plus large temporellement que les autres.

Ces situations peuvent être illustrées sur la figure page suivante.

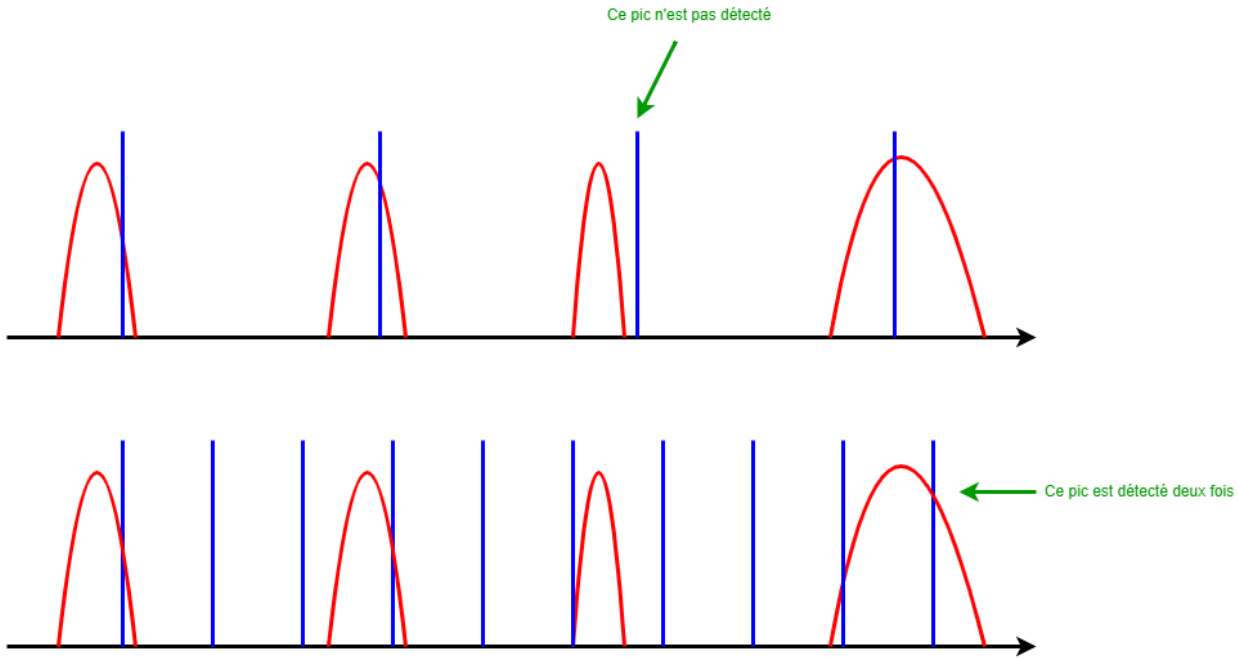


Schéma des problèmes d'échantillonnage et de détection des pics de kicks de la musique.

Le code fourni en annexe fonctionne donc approximativement ; sur certaines musiques, nous étions capables de faire se déclencher l'allumage des lampes sur presque tous les kicks, mais sur d'autres musiques, le système s'emballait ou était très lent, et faisait clignoter les lampes soit trop rapidement, soit trop lentement.

Une autre problématique, moindre pendant notre projet, mais qui pouvait occasionnellement être contraignante, est la tension de seuil utilisée comme condition de changement de couleur. En effet, selon le moment dans la musique, les kicks peuvent être plus ou moins prononcés, et donc être très distinguables du bruit de signal, ou au contraire être proche du bruit. Même si nous avions pu utiliser la détection de front montant, cet effet aurait été limitant selon les situations. Étant donné que cet effet était moins prononcé que la problématique d'échantillonnage évoquée précédemment, nous avons consacré moins de temps à sa résolution optimale.

Les deux limitations à ce code sont donc le seuil en tension, qui est potentiellement trop élevé ou trop bas selon les musiques, et l'échantillonnage des pics de signal. Cependant, ces limitations sont celles que nous avons pu identifier durant notre projet. Il est probable que premièrement, elles soient en réalité mal identifiées et que d'autres sources aux problèmes soient à considérer ; et deuxièmement, que d'autres problèmes apparaissent une fois ces deux-ci améliorés et réglés.
Idéalement, il aurait été intéressant de se pencher davantage sur ces deux problématiques, afin de les optimiser, de manière à ce que, pour n'importe quelle musique, le programme ne soit pas limité par un seuil mal calculé ou une mauvaise détection des pics.

5/ Bilan de l'équipe

Dans cette partie, nous allons faire le bilan de notre projet ProTIS. Nous verrons dans une première partie le bilan technique c'est-à-dire ce que l'on a développé comme fonctionnalité et celle que l'on aurait pu faire. Dans une 2nde partie, nous aborderons le bilan des compétences acquises et notre ressenti vis-à-vis de ce projet.

a) Bilan technique

Tout d'abord, les objectifs (rappelés dans la partie 1/a)) qui nous étaient fixés ont été atteints dans leur ensemble.
Pour ce qui est de la partie analogique, nous avons vu que nous avions des résultats assez mitigés. En effet, le clignotement des lampes LED n'arrivait pas à suivre toutes les basses. Pour éviter ce problème, la 1ère solution qui a été envisagée était de faire un traitement en transformée de Fourier sur le signal analogique pour obtenir les fréquences et ainsi filtrer beaucoup plus simplement. Le programme que nous voulions utiliser pour faire cette transformée de fourier sur le signal était issu d'un TP de 1ère année. Cependant, nous n'avons pas réussi à déboguer cet algorithme pour l'adapter à notre situation et c'est pour cette raison que nous nous sommes orientés vers une nouvelle solution. Un autre moyen d'améliorer cette partie analogique est de faire une détection de front montant au lieu de faire un simple seuillage. En effet, dans la partie algorithme nous n'avons utilisé qu'une comparaison simple entre le signal et une valeur seuil sans se préoccuper du "passé" (savoir si on est déjà sur le plateau ou non). Enfin, une dernière amélioration possible est d'ajouter des diodes Zener dans le montage pour bien s'assurer que l'on ne dépasse pas une tension critique qui pourrait faire claquer les ASO ou la carte Nucléo. Ce dispositif n'est pas non plus indispensable car on s'est occupé d'avoir de la marge au cas où nous aurions une surcharge en entrée (>20V).
Pour la partie numérique, nous avons obtenu de meilleurs résultats (cf. 3/). Cependant nous aurions voulu implémenter plus de motifs et de scénarios réalisables par notre programme. De plus, nous sommes arrivés à contrôler "seulement" l'ensemble de 4 lampes mais nous aurions voulu contrôler plus de lampes. Par exemple, contrôler toutes les lampes lors d'une soirée. Enfin, une autre amélioration que nous aurions voulu faire est une sorte d'interface graphique via TeraTerm pour pouvoir contrôler les différents scénarios.

b) Bilan des compétences acquises

Dans cette partie, nous allons faire un bilan humain de ce projet c'est-à-dire des compétences acquises (savoir-être et savoir-faire). Pour ce qui est des savoir-être, nous avons eu assez de mal à nous répartir les différentes tâches au sein du groupe car nous ne savions pas quel était le domaine de prédilection de chacun. Nous avons donc dû nous adapter et nous avons pu apprendre à travailler en groupe. Nous avons aussi dû apprendre à expliquer notre travail le plus simplement et succinctement pour avancer dans ce projet. Une autre compétence que nous ne soupçonnions pas d'être très utile dans ce projet est l'utilisation et l'adaptation d'un code déjà existant. Nous avons développé cette compétence tant dans la

partie numérique avec l'utilisation d'un code pré-existant pour le clavier que dans la partie analogique avec le code de la transformée de Fourier. D'un point de vue savoir-faire, nous avons amélioré nos compétences sur MBED et en électronique en utilisant par exemple des logiciels comme KiCad pour faire des schémas électroniques clairs. En électronique nous avons revu des éléments indispensables à la conception électronique comme des circuits amplificateurs ou rehausseurs.

Nous avons donc vu au travers de ces bilans que ce projet, même s'il a été compliqué à finir, reste une expérience pédagogique très intéressante qui nous permet d'avoir un 1er rapport au travail d'ingénieur au sein d'un groupe.

6/ Conclusion

Durant les séances de ce projet, nous avons été amenés à mettre en place un certain nombre de solutions techniques de manière à répondre aux exigences initiales. Malgré des problèmes récurrents et difficiles à résoudre, nous avons été capables de répondre à une partie non négligeable des attentes.

7/ Annexes

a) Montages électriques

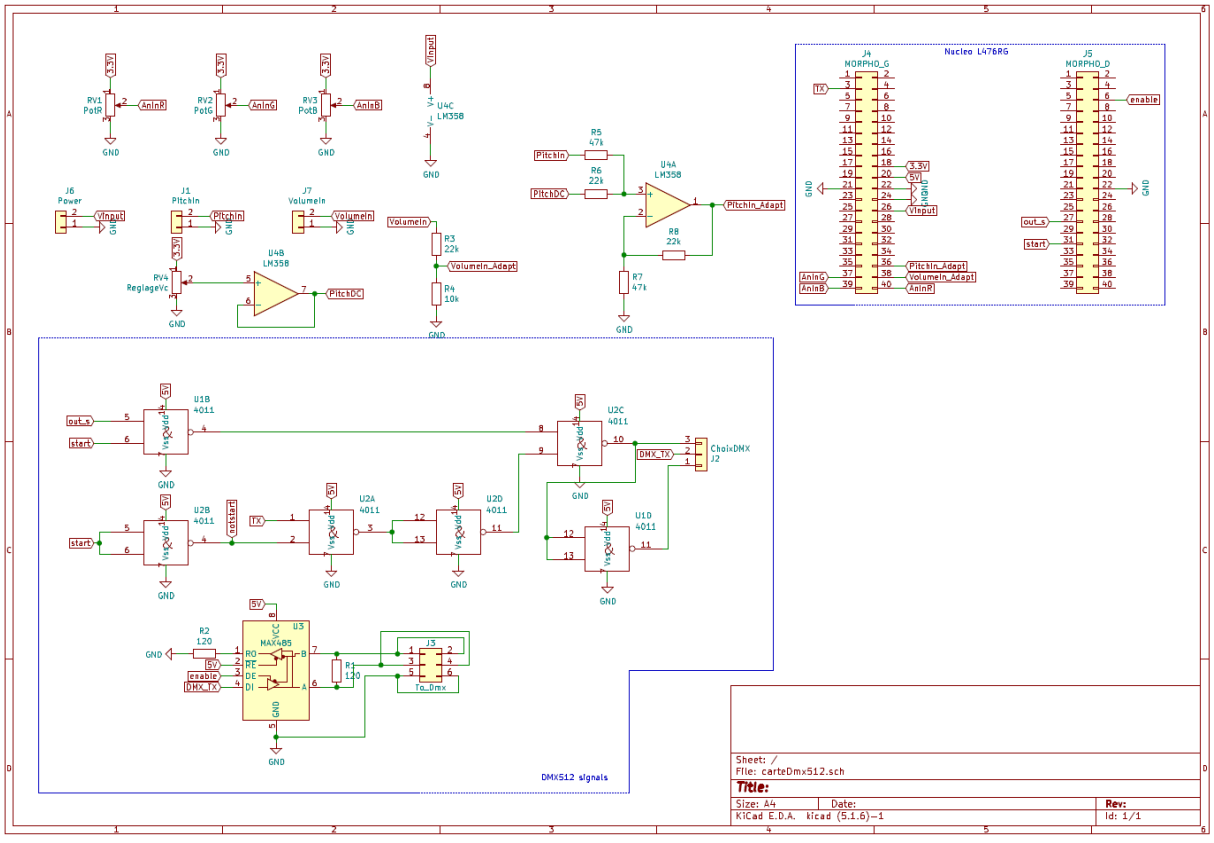


Schéma de câblage de la carte d'extension DMX512 (lien carte d'extension/carte Nucléo).

b) Fonctions et codes MBed

Les fonctions suivantes sont utilisées dans (presque) tous les codes : dans une volonté d'optimisation de place, nous les déposons ici, et leurs emplacements seront simplement indiqués dans les autres programmes.

Sont incluses dans chaque programme les bibliothèques suivantes :

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

Nous avons également à chaque fois :

```
// Blinking rate in milliseconds
#define SAMPLES 512
#define RAND_MAX 255
#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);

/* 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(100);                   // 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);
}
/* Détection d'une note reçue en MIDI */
bool isNoteMIDIdetected(void){
	if(new_note_midi == 1)
	return true;
	else
	return false;
}
/* Fonction d'interruption sur MIDI */
void ISR_midi_in(void){
	char data = midi.getc();
	if(data >= 128) {
		cpt_midi = 0;}
	else {
		cpt_midi++;}
	midi_data[cpt_midi] = data;
	if(cpt_midi == 2){
		cpt_midi = 0;
		if(((midi_data[0] & 0xF0) == MIDI_NOTE_ON) || ((midi_data[0] & 0xF0) == MIDI_NOTE_OFF)){
			new_note_midi = 1;
			channel_data = midi_data[0] & 0x0F;
			note_data = midi_data[1];
			velocity_data = midi_data[2];
		}
		else{
			if(midi_data[0] == MIDI_CC){
				new_data_midi = 1;
				control_ch = midi_data[1];
				control_value = midi_data[2];
			}
		}
	}
}


// Définition des matrices de couleurs RGB pour chaque LED
// 37 valeurs dans chaque liste = 37 touches sur le clavier avec un dégradé de couleurs
const	uint8_t	scriabin_r1[37] = {191, 191, 225, 255, 255, 240, 255, 255, 255, 255, 255, 255, 255, 255, 226, 194, 175, 126, 46, 51, 51, 51, 41, 41, 48, 43, 108,138,161, 192, 224, 255, 255, 255, 255, 255, 255};
const	uint8_t	scriabin_g1[37] = {0, 0, 0, 60, 85, 105, 112, 145, 159, 163, 198, 210, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 234, 205, 110, 65, 23, 20,20, 18, 23, 15, 20, 23, 20, 20, 20};
const	uint8_t	scriabin_b1[37] = {0, 0, 0, 0, 0, 2,2, 0,15,  5,  10,  31,  43,  31,  36,  25,  38,  41,  95, 160, 228, 255, 255, 255, 255, 255, 255, 255, 255, 255, 247, 208, 170, 138, 138,138};

const	uint8_t	scriabin_r2[37] = { 240, 255, 255, 255, 255, 255, 255, 255, 255, 226, 194, 175, 126, 46, 51, 51, 51, 41, 41, 48, 43, 108,138,161, 192, 224, 255, 255, 255, 255, 255, 255,191, 191, 225, 255, 255};
const	uint8_t	scriabin_g2[37] = { 105, 112, 145, 159, 163, 198, 210, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 234, 205, 110, 65, 23, 20,20, 18, 23, 15, 20, 23, 20, 20, 20, 0, 0, 0, 60, 85};
const	uint8_t	scriabin_b2[37] = { 2,2, 0,15,  5,  10,  31,  43,  31,  36,  25,  38,  41,  95, 160, 228, 255, 255, 255, 255, 255, 255, 255, 255, 255, 247, 208, 170, 138, 138,138, 0, 0, 0, 0, 0};

const	uint8_t	scriabin_r3[37] = {  255, 255, 255, 255, 226, 194, 175, 126, 46, 51, 51, 51, 41, 41, 48, 43, 108,138,161, 192, 224, 255, 255, 255, 255, 255, 255,191, 191, 225, 255, 255, 240, 255, 255, 255, 255};
const	uint8_t	scriabin_g3[37] = {  198, 210, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 234, 205, 110, 65, 23, 20,20, 18, 23, 15, 20, 23, 20, 20, 20, 0, 0, 0, 60, 85, 105, 112, 145, 159, 163};
```

```
const uint8_t scriabin_b3[37] = { 10, 31, 43, 31, 36, 25, 38, 41, 95, 160, 228, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 247, 208, 170, 138, 138,138, 0, 0, 0, 0, 0, 2,2, 0,15, 5};

const uint8_t scriabin_r4[37] = { 194, 175, 126, 46, 51, 51, 51, 41, 41, 48, 43, 108,138,161, 192, 224, 255, 255, 255, 255, 255, 255,191, 191, 225, 255, 255, 240, 255, 255, 255, 255, 255, 255, 255, 255, 226};
const uint8_t scriabin_g4[37] = { 255, 255, 255, 255, 255, 255, 255, 234, 205, 110, 65, 23, 20,20, 18, 23, 15, 20, 23, 20, 20, 20, 0, 0, 0, 60, 85, 105, 112, 145, 159, 163, 198, 210, 255, 255, 255};
const uint8_t scriabin_b4[37] = { 25, 38, 41, 95, 160, 228, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 247, 208, 170, 138, 138,138, 0, 0, 0, 0, 0, 2,2, 0,15, 5, 10, 31, 43, 31, 36};
```

A partir du programme blinkfast, les lignes suivantes sont également présentes avant le Main :

```
DigitalOut  out_tx(D5);
DigitalOut  start(D4);
DigitalOut  enableDMX(D6);
Serial      dmx(A0, A1);
AnalogIn    myInput(A2);

void initDMX();
void updateDMX();

char dmx_data[SAMPLES] = {0};
int delay=100;
double meas;
double tension;
int curseurFantome;

Ticker detect;                  // Déclaration du ticker
void funct_detect(void);        // Déclaration de la fonction d'interruption du ticker
```

Programme éclairage des lampes avec une couleur aléatoire pour chaque LED à chaque fois qu'une basse est détectée (fonctionne moyennement) :

```
/*****************************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* But du programme : piloter 4 LEDs avec les basses d'un signal sonore */
/* couleur aléatoire à chaque LED à chaque battement de basse */
/*****************************************************************************/
/* Code fonctionnel pour un signal issu d'un GBF en pulses de largeur 80ms, de période 500ms et d'amplitude P to P 2V*/
/* Quelques bugs de double détection et de non-détection */

char dmx_data[SAMPLES] = {0};
int delay=100;
double meas;
double tension;
int curseurFantome;

int main() {
        initDMX();
        debug_pc.baud(115200);
        dmx_data[0] = 0;
        dmx_data[3]= 100;
        dmx_data[4] = 200;
        dmx_data[5] = 0;
        dmx_data[6] = 0;
```

```
            dmx_data[8] = 0;
            dmx_data[11] = 100;
            dmx_data[12] = int(rand()/2)+128;
            dmx_data[13] = int(rand()/2)+128;
            dmx_data[14] = int(rand()/2)+128;

            dmx_data[16] = 0;
            dmx_data[19] = 100;
            dmx_data[20] = int(rand()/2)+128;
            dmx_data[21] = int(rand()/2)+128;
            dmx_data[22] = int(rand()/2)+128;

            dmx_data[24] = 0;
            dmx_data[27] = 100;
            dmx_data[28] = int(rand()/2)+128;
            dmx_data[29] = int(rand()/2)+128;
            dmx_data[30] = int(rand()/2)+128;

            detect.attach(&funct_detect, 0.075);

            while (1) {
            curseurFantome=myInput.read_u16();
            tension = curseurFantome / 65536.0 * 3.3;          // en V
            updateDMX();
            }
}

void initDMX()                          // voir ci-dessus
void updateDMX()                        // voir ci-dessus
void initMIDI(void)                     // voir ci-dessus
bool isNoteMIDIdetected(void)           // voir ci-dessus
void ISR_midi_in(void)                  // voir ci-dessus

void funct_detect()
{
            if (tension>1)
            {
            dmx_data[0] = 0;
            dmx_data[3] = 100;
            dmx_data[4] = int(rand()/2)+128;
            dmx_data[5] = int(rand()/2)+128;
            dmx_data[6] = int(rand()/2)+128;

            dmx_data[8] = 0;
            dmx_data[11] = 100;
            dmx_data[12] = int(rand()/2)+128;
            dmx_data[13] = int(rand()/2)+128;
            dmx_data[14] = int(rand()/2)+128;

            dmx_data[16] = 0;
            dmx_data[19] = 100;
            dmx_data[20] = int(rand()/2)+128;
            dmx_data[21] = int(rand()/2)+128;
            dmx_data[22] = int(rand()/2)+128;

            dmx_data[24] = 0;
```

```
            dmx_data[27] = 100;
            dmx_data[28] = int(rand()/2)+128;
            dmx_data[29] = int(rand()/2)+128;
            dmx_data[30] = int(rand()/2)+128;
            wait_us(5000);
            }
}
```

Programme blinkfast

```
/*****************************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* Application pour pilotage DMX/MIDI */
/* Utilisation : piloter 4 LEDs avec un clavier MIDI, clignotement rapide en continu */
/* Clignotement se réinitialise à chaque note jouée */
/* chaque touche du clavier donne une couleur différente à chaque LED */
/*****************************************************************************/

// Définition des matrices de couleurs RGB pour chaque LED

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

        // Initialisation des liaisons DMX et MIDI
        initDMX();
        initMIDI();

        blink.attach(&funct_blink, 0.08);          // Période de clignotement : 0.08 s

        dmx_data[0] = 0;
        dmx_data[3] = 100;
        dmx_data[8] = 0;
        dmx_data[11] = 5;
        dmx_data[16] = 0;
        dmx_data[19] = 100;
        dmx_data[24] = 0;
        dmx_data[27] = 5;

        while(1) {
                if(isNoteMIDIdetected()){
                        debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
                        char note = (note_data-17)%37-1;          // on retranche 17 ici car c'est le décalage induit par le clavier utilisé dans les tests (cette valeur peut varier)
                        debug_pc.printf("N=%d\r\n", note);
                        // Renkforce LPT12 - AD 1
                if (dmx_data[3]==100){                    // si l'intensité de la première lampe est maximale, on la rend plus faible, sinon si l'intensité est minimale, on la rend maximale. Le canal 3 commande l'intensité de la première lampe. On répète pour les autres lampes, les canaux 11, 19, 27 codant les intensités des trois autres lampes.
                        dmx_data[3] = 5;}
                else{
                        dmx_data[3]=100;}
                dmx_data[4] = scriabin_r1[note];
                dmx_data[5] = scriabin_g1[note];
                dmx_data[6] = scriabin_b1[note];
```

```
                        if (dmx_data[11]==100){
                                dmx_data[11] = 5;}
                        else{
                                dmx_data[11]=100;}
                        dmx_data[12] = scriabin_r2[note];
                        dmx_data[13] = scriabin_g2[note];
                        dmx_data[14] = scriabin_b2[note];

                        if (dmx_data[19]==100){
                                dmx_data[19] = 5;}
                        else{
                                dmx_data[19]=100;}
                        dmx_data[20] = scriabin_r3[note];
                        dmx_data[21] = scriabin_g3[note];
                        dmx_data[22] = scriabin_b3[note];

                        if (dmx_data[27]==100){
                                dmx_data[27] = 5;}
                        else{
                                dmx_data[27]=100;}
                        dmx_data[28] = scriabin_r4[note];
                        dmx_data[29] = scriabin_g4[note];
                        dmx_data[30] = scriabin_b4[note];

                        new_note_midi = 0;
                        }
                        updateDMX();
                        wait_us(10000);
                }
}

void initDMX()                          // voir ci-dessus
void updateDMX()                        // voir ci-dessus
void initMIDI(void)                     // voir ci-dessus
bool isNoteMIDIdetected(void)           // voir ci-dessus
void ISR_midi_in(void)                  // voir ci-dessus

void funct_blink() {                    // Fonction d'interruption du ticker
        if (dmx_data[27]==100) {
                dmx_data[27]=5; }
        else {
                if (dmx_data[27]==5) {
                        dmx_data[27]=100; }
                if (dmx_data[19]==100) {
                        dmx_data[19]=5; } }
        else {
                if (dmx_data[19]==5) {
                        dmx_data[19]=100; }
        }
        if (dmx_data[3]==5) {
                dmx_data[3]=100; }
        else{
                if (dmx_data[3]==100) {
                        dmx_data[3]=5; }
        }
        if (dmx_data[11]==5) {
                dmx_data[11]=100; }
```

```
            else{
                if (dmx_data[11]==100) {
                    dmx_data[11]=5; }
            }
            updateDMX();
}
```

Programme blinkmedium

```
/*****************************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* But du programme : piloter 4 LEDs avec un clavier, clignotement moyen */
/* chaque touche du clavier donne une couleur différente à chaque LED */
/*****************************************************************************/

// Définition des matrices de couleurs RGB pour chaque LED

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

    initDMX();
    initMIDI();

    blink.attach(&funct_blink, 0.8); // Période de clignotement : 0.8s

    dmx_data[0] = 0;
    dmx_data[3] = 100;

    dmx_data[8] = 0;
    dmx_data[11] = 5;

    dmx_data[16] = 0;
    dmx_data[19] = 100;

    dmx_data[24] = 0;
    dmx_data[27] = 5;

    while(1) {
        if(isNoteMIDIdetected()){
            debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
            char note = (note_data-17)%37-1;        // on retranche 17 ici car c'est le décalage induit par le clavier utilisé dans les tests (cette valeur peut varier)
            debug_pc.printf("N=%d\r\n", note);
            // Renkforce LPT12 - AD 1
        if (dmx_data[3]==100){                      // si l'intensité de la première lampe est maximale, on la rend plus faible, sinon si l'intensité est minimale, on la rend maximale. Le canal 3 commande l'intensité de la première lampe. On répète pour les autres lampes, les canaux 11, 19, 27 codant les intensités des trois autres lampes.
            dmx_data[3] = 5;}
        else{
            dmx_data[3]=100;}
        dmx_data[4] = scriabin_r1[note];
        dmx_data[5] = scriabin_g1[note];
        dmx_data[6] = scriabin_b1[note];

        if (dmx_data[11]==100){
```

```
                        dmx_data[11] = 5;}
                    else{
                        dmx_data[11]=100;}
                    dmx_data[12] = scriabin_r2[note];
                    dmx_data[13] = scriabin_g2[note];
                    dmx_data[14] = scriabin_b2[note];

                    if (dmx_data[19]==100){
                        dmx_data[19] = 5;}
                    else{
                        dmx_data[19]=100;}
                    dmx_data[20] = scriabin_r3[note];
                    dmx_data[21] = scriabin_g3[note];
                    dmx_data[22] = scriabin_b3[note];

                    if (dmx_data[27]==100){
                        dmx_data[27] = 5;}
                    else{
                        dmx_data[27]=100;}
                    dmx_data[28] = scriabin_r4[note];
                    dmx_data[29] = scriabin_g4[note];
                    dmx_data[30] = scriabin_b4[note];

                    new_note_midi = 0;
                    }
                    updateDMX();
                    wait_us(10000);
            }
}

void initDMX()                          // voir ci-dessus
void updateDMX()                        // voir ci-dessus
void initMIDI(void)                     // voir ci-dessus
bool isNoteMIDIdetected(void)           // voir ci-dessus
void ISR_midi_in(void)                  // voir ci-dessus

void funct_blink() {                    // Fonction d'interruption du ticker
            if (dmx_data[27]==100) {
                    dmx_data[27]=5; }
            else {
                    if (dmx_data[27]==5) {
                        dmx_data[27]=100; }
                    if (dmx_data[19]==100) {
                        dmx_data[19]=5; } }
            else {
                    if (dmx_data[19]==5) {
                        dmx_data[19]=100; }
            }
            if (dmx_data[3]==5) {
                    dmx_data[3]=100; }
            else{
                    if (dmx_data[3]==100) {
                        dmx_data[3]=5; }
            }
            if (dmx_data[11]==5) {
                    dmx_data[11]=100; }
            else{
```

```
                if (dmx_data[11]==100) {
                        dmx_data[11]=5; }
        }
        updateDMX();
}
```

Programme blinkslow

```
/*****************************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* But du programme : piloter 4 LEDs avec un clavier, clignotement lent */
/* chaque touche du clavier donne une couleur différente à chaque LED */
/*****************************************************************************/

// Définition des matrices de couleurs RGB pour chaque LED

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

        initDMX();
        initMIDI();

        blink.attach(&funct_blink, 3); // Période de clignotement : 3s

        dmx_data[0] = 0;
        dmx_data[3] = 100;

        dmx_data[8] = 0;
        dmx_data[11] = 5;

        dmx_data[16] = 0;
        dmx_data[19] = 100;

        dmx_data[24] = 0;
        dmx_data[27] = 5;

        while(1) {
                if(isNoteMIDIdetected()){
                        debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
                        char note = (note_data-17)%37-1;        // on retranche 17 ici car c'est le décalage induit par le
                clavier utilisé dans les tests (cette valeur peut varier)
                        debug_pc.printf("N=%d\r\n", note);
                        // Renkforce LPT12 - AD 1
                if (dmx_data[3]==100){                  // si l'intensité de la première lampe est maximale, on la rend plus
        faible, sinon si l'intensité est minimale, on la rend maximale. Le canal 3 commande l'intensité de la première lampe. On
        répète pour les autres lampes, les canaux 11, 19, 27 codant les intensités des trois autres lampes.
                        dmx_data[3] = 5;}
                else{
                        dmx_data[3]=100;}
                dmx_data[4] = scriabin_r1[note];
                dmx_data[5] = scriabin_g1[note];
                dmx_data[6] = scriabin_b1[note];

                if (dmx_data[11]==100){
                        dmx_data[11] = 5;}
```

```
                    else{
                            dmx_data[11]=100;}
                    dmx_data[12] = scriabin_r2[note];
                    dmx_data[13] = scriabin_g2[note];
                    dmx_data[14] = scriabin_b2[note];

                    if (dmx_data[19]==100){
                            dmx_data[19] = 5;}
                    else{
                            dmx_data[19]=100;}
                    dmx_data[20] = scriabin_r3[note];
                    dmx_data[21] = scriabin_g3[note];
                    dmx_data[22] = scriabin_b3[note];

                    if (dmx_data[27]==100){
                            dmx_data[27] = 5;}
                    else{
                            dmx_data[27]=100;}
                    dmx_data[28] = scriabin_r4[note];
                    dmx_data[29] = scriabin_g4[note];
                    dmx_data[30] = scriabin_b4[note];

                    new_note_midi = 0;
                    }
                    updateDMX();
                    wait_us(10000);
            }
}

void initDMX()                          // voir ci-dessus
void updateDMX()                        // voir ci-dessus
void initMIDI(void)                     // voir ci-dessus
bool isNoteMIDIdetected(void)           // voir ci-dessus
void ISR_midi_in(void)                  // voir ci-dessus

void funct_blink() {                    // Fonction d'interruption du ticker
            if (dmx_data[27]==100) {
                    dmx_data[27]=5; }
            else {
                    if (dmx_data[27]==5) {
                            dmx_data[27]=100; }
                    if (dmx_data[19]==100) {
                            dmx_data[19]=5; } }
            else {
                    if (dmx_data[19]==5) {
                            dmx_data[19]=100; }
            }
            if (dmx_data[3]==5) {
                    dmx_data[3]=100; }
            else{
                    if (dmx_data[3]==100) {
                            dmx_data[3]=5; }
            }
            if (dmx_data[11]==5) {
                    dmx_data[11]=100; }
            else{
                    if (dmx_data[11]==100) {
```

```
                                        dmx_data[11]=5; }
                }
                updateDMX();
}
```

Programme degrade1

```
/*****************************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* But du programme : piloter 4 LEDs avec un clavier, dégradé de couleurs */
/* chaque touche du clavier donne une couleur différente à chaque LED */
/*****************************************************************************/

// Définition des matrices de couleurs RGB pour chaque LED

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

        initDMX();
        initMIDI();

        while(1) {
                if(isNoteMIDIdetected()){
                        debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
                        char note = (note_data-15)%37;
                        debug_pc.printf("N=%d\r\n", note);
                        // Renkforce LPT12 - AD 1
                        dmx_data[0] = 0;
                        dmx_data[3] = 100;
                        dmx_data[4] = scriabin_r1[note];
                        dmx_data[5] = scriabin_g1[note];
                        dmx_data[6] = scriabin_b1[note];

                        dmx_data[8] = 0;
                        dmx_data[11] = 100;
                        dmx_data[12] = scriabin_r2[note];
                        dmx_data[13] = scriabin_g2[note];
                        dmx_data[14] = scriabin_b2[note];

                        dmx_data[16] = 0;
                        dmx_data[19] = 100;
                        dmx_data[20] = scriabin_r3[note];
                        dmx_data[21] = scriabin_g3[note];
                        dmx_data[22] = scriabin_b3[note];

                        dmx_data[24] = 0;
                        dmx_data[27] = 100;
                        dmx_data[28] = scriabin_r4[note];
                        dmx_data[29] = scriabin_g4[note];
                        dmx_data[30] = scriabin_b4[note];

                        new_note_midi = 0;
                }
                updateDMX();
                wait_us(10000);
```

```
        }
}

void initDMX()                          // voir ci-dessus
void updateDMX()                        // voir ci-dessus
void initMIDI(void)                     // voir ci-dessus
bool isNoteMIDIdetected(void)           // voir ci-dessus
void ISR_midi_in(void)                  // voir ci-dessus
```

Programme degrade2

```
/*****************************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* But du programme : piloter 4 LEDs avec un clavier, dégradé 2 par 2 */
/* chaque touche du clavier donne une couleur différente à chaque LED 2 à 2 (LEDs 1 et 3, LEDs 2 et 4) */
/*****************************************************************************/

// Définition des matrices de couleurs RGB pour chaque LED

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

        initDMX();
        initMIDI();

        while(1) {
                if(isNoteMIDIdetected()){
                        debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
                        char note = (note_data-15)%37;
                        debug_pc.printf("N=%d\r\n", note);
                        // Renkforce LPT12 - AD 1
                        dmx_data[0] = 0;
                        dmx_data[3] = 100;
                        dmx_data[4] = scriabin_r1[note];
                        dmx_data[5] = scriabin_g1[note];
                        dmx_data[6] = scriabin_b1[note];

                        dmx_data[8] = 0;
                        dmx_data[11] = 100;
                        dmx_data[12] = scriabin_r2[note];
                        dmx_data[13] = scriabin_g2[note];
                        dmx_data[14] = scriabin_b2[note];

                        dmx_data[16] = 0;
                        dmx_data[19] = 100;
                        dmx_data[20] = scriabin_r3[note];
                        dmx_data[21] = scriabin_g3[note];
                        dmx_data[22] = scriabin_b3[note];

                        dmx_data[24] = 0;
                        dmx_data[27] = 100;
                        dmx_data[28] = scriabin_r4[note];
                        dmx_data[29] = scriabin_g4[note];
                        dmx_data[30] = scriabin_b4[note];
```

```
                    new_note_midi = 0;
                }
                updateDMX();
                wait_us(10000);
            }
}

void initDMX()                      // voir ci-dessus
void updateDMX()                    // voir ci-dessus
void initMIDI(void)                 // voir ci-dessus
bool isNoteMIDIdetected(void)       // voir ci-dessus
void ISR_midi_in(void)              // voir ci-dessus
```

Programme intensitevariable

```
/*********************************************************************/
/* Projet ProTIS 2022 */
/* Groupe 1 */
/* But du programme : piloter 4 LEDs avec un clavier, dégradé sur tout le spectre */
/* chaque touche du clavier donne une couleur différente aux 4 LEDs (même couleur pour toutes les LEDs) */
/* variation de l'intensité en continu (4 niveaux) */
/*********************************************************************/

// Définition des matrices de couleurs RGB pour chaque LED
// Main
int main() {
    debug_pc.baud(115200);
    debug_pc.printf("Essai DMX512\r\n");
    initDMX();
    initMIDI();

    intensite.attach(&funct_intensite, 0.2);

    while(1) {
        if(isNoteMIDIdetected()){
            debug_pc.printf("C=%d,N=%d,V=%d\r\n", channel_data, note_data, velocity_data);
            char note = (note_data-15)%37;
            debug_pc.printf("N=%d\r\n", note);
            // Renkforce LPT12 - AD 1
            dmx_data[0] = 0;
            dmx_data[3] = 100;
            dmx_data[4] = scriabin_r1[note];
            dmx_data[5] = scriabin_g1[note];
            dmx_data[6] = scriabin_b1[note];

            dmx_data[8] = 0;
            dmx_data[11] = 50;
            dmx_data[12] = scriabin_r2[note];
            dmx_data[13] = scriabin_g2[note];
            dmx_data[14] = scriabin_b2[note];

            dmx_data[16] = 0;
            dmx_data[19] = 20;
            dmx_data[20] = scriabin_r3[note];
            dmx_data[21] = scriabin_g3[note];
            dmx_data[22] = scriabin_b3[note];

            dmx_data[24] = 0;
```

```
                dmx_data[27] = 5;
                dmx_data[28] = scriabin_r4[note];
                dmx_data[29] = scriabin_g4[note];
                dmx_data[30] = scriabin_b4[note];

                new_note_midi = 0;
            }
            updateDMX();
            wait_us(10000);
        }
}

void initDMX()                          // voir ci-dessus
void updateDMX()                        // voir ci-dessus
void initMIDI(void)                     // voir ci-dessus
bool isNoteMIDIdetected(void)           // voir ci-dessus
void ISR_midi_in(void)                  // voir ci-dessus

void funct_intensite() {        // Fonction d'interruption du ticker
    if (dmx_data[3]==100) {
    dmx_data[3]=50; }
    else {
        if (dmx_data[3]==50) {
            dmx_data[3]=20;}
        else{
            if (dmx_data[3]==20) {
                dmx_data[3]=5; }
            else {
                if (dmx_data[3]==5) {
                    dmx_data[3]=100; }
                }
            }
        }
    if (dmx_data[11]==50) {
        dmx_data[11]=20; }
    else {
        if (dmx_data[11]==20) {
            dmx_data[11]=5; }
        else{
            if (dmx_data[11]==5) {
                dmx_data[11]=100; }
            else {
                if (dmx_data[11]==100) {
                    dmx_data[11]=50; }
                }
            }
        }
    if (dmx_data[19]==20) {
        dmx_data[19]=5; }
    else {
        if (dmx_data[19]==5) {
            dmx_data[19]=100;}
        else{
            if (dmx_data[19]==100) {
                dmx_data[19]=50; }
            else {
                if (dmx_data[19]==50) {
```

```
                                        dmx_data[19]=20; }
                                }
                        }
                }
if (dmx_data[27]==5) {
                dmx_data[27]=100;}
else {
                if (dmx_data[27]==100) {
                        dmx_data[27]=50; }
                else{
                        if (dmx_data[27]==50) {
                                dmx_data[27]=20; }
                        else {
                                if (dmx_data[27]==20) {
                                        dmx_data[27]=5; }
                                }
                        }
                }
updateDMX();
}
```