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

SOLEC

Pilotage et conversion MIDI/DMX

Rapport technique

Mouttaki Ali
Ellafi Ilian
Pilloix Meili
Minnaert Étienne

Table des matières

1 Introduction **2**

2 Cahier des charges **2**

3 Pilotage des Projecteurs **2**

3.1 Branchements 2

3.2 Structures des données et notations 3

3.3 Programme de conversion MIDI/DMX 5

3.4 Programme d'écriture et lecture de la carte micro-SD 5

3.5 Interface Homme-Machine 6

4 Travail effectué et tests de validation **6**

5 Difficultés rencontrées **7**

6 Conclusion **8**

7 Annexes **8**

7.1 Codes 8

7.2 Sources 8

1 Introduction

La création de spectacles lumineux nécessite le contrôle de projecteur. La majorité de ceux-ci sont pilotable via un protocole normalisé appelé DMX512. Pour programmer et piloter les jeux de lumières, il existe des logiciels dédiés qui convertissent soient directement en DMX soient paramètre des contrôleurs. Parmis ces contrôleurs certains utilise le protocole MIDI initialement développé pour le musique numérique. L'utilisation de piano ou des interfaces utilisées par le beat maker pour créer des rythmiques offre un grand grand nombre de touches facilitant des scénographies dynamiques.
Notre projet vise à réaliser un tel dispositif pour une utilisation par exemple dans les évènements festifs ou les spectacles de l'école.Il est composé d'une interface matérielle de conversion MIDI vers DMX ainsi qu'une interface logicielle pour la programmation de spectacle, dans l'optique de s'affranchir de l'ordinateur pour le pilotage des jeux de lumière.

2 Cahier des charges

Nous avons reçu de la part de la société SOLEC un cahier des charges précis à réaliser. Le but est de pouvoir piloter jusqu'à 16 groupes de projecteurs indépendamment à l'aide d'une interface matérielle qui est l'interface du contrôleur MIDI. L'utilisateur doit pouvoir modifier la couleur, l'orientation selon deux axes de rotations, et l'intensité de chaque projecteur grâce à une liaison DMX. Il faut aussi pouvoir accéder aux différents modes suivants :

- Mode Scriabin : une couleur correspond à un bouton
- Mode Contrôleur MIDI : un bouton déclenche un motif spécifique pour tout les spots
- Mode Séquenceur : un bouton déclenche une séquence sur un ou plusieurs projecteurs

Nous nous sommes concentrer sur le Mode Séquenceur, en plus de la réalisation d'une interface logicielle Homme-Machine permettant de configurer les projecteurs (adresses DMX et configurations des canaux associées à une note du contrôleur MIDI. Les modes Scriabin et Contrôleur MIDI peuvent être interprété comme des cas particuliers du mode séquenceur. Le mode Scriabin est une séquence dans laquelle le motif est identique et le mode Contrôleur MIDI correspond au cas où des séquences de différentes lampes sont associées à une même note MIDI.

La solution proposée compte tenu du cahier des charge peut être résumé dans le schéma fonctionnel en **Figure 1** :

Le contrôleur MIDI et les projecteurs sont paramétrés par un logiciel. Les informations sont mises en forme et stockées sur une carte micro-SD. Le micro-contrôleur accède aux informations et initialise ses variables. Dès lors le microcontrôleur capte puis convertie un évènement sur le contrôleur MIDI en une action sur la chaine DMX qui contrôlent les projecteurs.

3 Pilotage des Projecteurs

3.1 Branchements

Le schéma électrique en **Figure 2** résume l'ensemble des branchements à réaliser. Il suffit de brancher le contrôleur MIDI sur une alimentation secteur et de relier le contrôleur via un câble

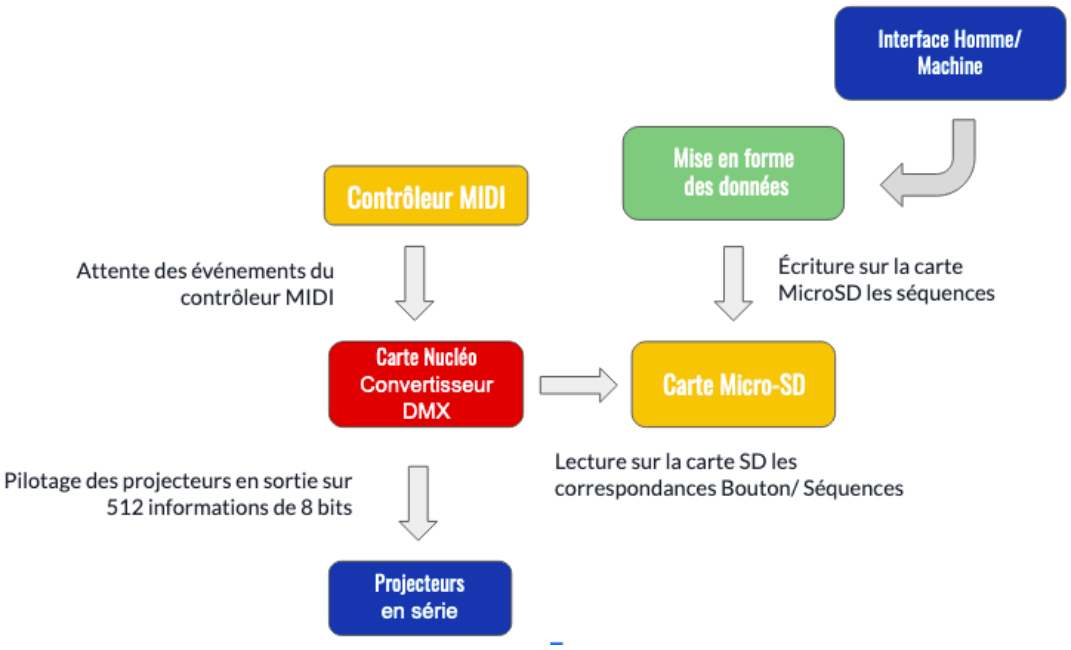

FIGURE 1 – Schéma fonctionnel

MIDI 5 broches à une carte Nucléo à l'aide d'un adaptateur. On branche alors ce câble sur la prise MIDI-OUT du contrôleur MIDI.

Le dispositif utilisé sur la carte Nucléo est une carte extension fournie par le LEnsE permettant des branchements MIDI-IN et MIDI-OUT. On branche alors une prise femelle sur l'entrée MIDI-IN que l'on relie donc au câble à 5 broches du contrôleurs MIDI.

Une autre carte extension du LEnsE est également reliée à la carte Nucléo. Elle a été conçu spécialement pour les liaisons DMX. On relie alors cette carte extension à une prise femelle DMX, elle-même lié à un câble DMX qui relie en série tout les projecteurs souhaités. À la carte Nucléo, nous relions un lecteur de carte micro-SD selon les branchements de la **Figure 2**.

Nous effectuons également le bon adressage des projecteurs. En effet, ces projecteurs recevront une liste DMX de 512 valeurs et selon les projecteurs et leur condition d'utilisation ils traitent plus ou moins d'information. Par exemple pour contrôler indépendamment 4 projecteurs, utilisant 8 canaux L'adressage sera le suivant :
- le premier projecteur lit les informations à partir de la première valeur
- le second projecteur lit les informations à partir de la 9ème valeur
- le troisième projecteur lit les informations à partir de la 17ème valeur
- le quatrième projecteur lit les informations à partir de la 25ème valeur

3.2 Structures des données et notations

Le programme utilise une structure imbriquée qui assure l'indépendance des différentes lampes les unes par aux autres. Chaque lampe comporte un plusieurs séquences, elles-même composées d'états (**(Figure 3)**. La lampe est définie par sa position dans la chaine DMX, nous gardons en mémoire la première position. À chaque séquence on associe une note du contrôleur MIDI qui permettra de l'enclencher. Chaque séquence est composée de plusieurs états. Les états contiennent les

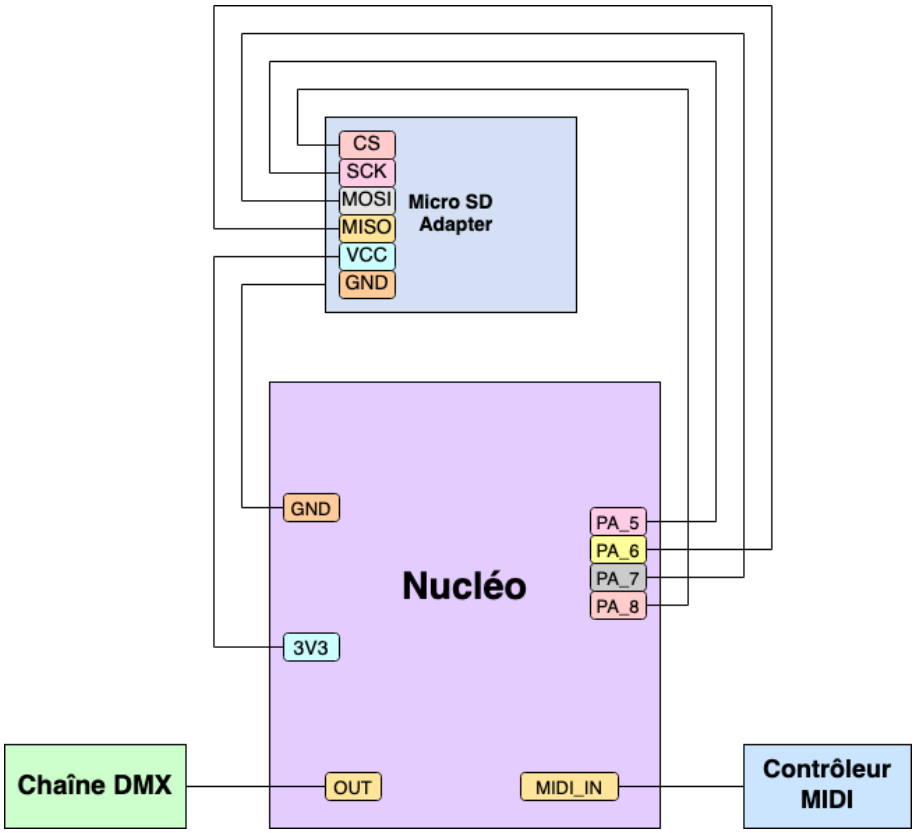

FIGURE 2 – Schéma électrique

informations de la lampe (l'intensité lumineuse, la position angulaire, la présence de stroboscopie, la quantité de rouge, bleu, vert, ambre, blanc...). Les états dépendent des possibilités offertes par chacune des lampes. Il est nécessaire d'utiliser la fiche technique.

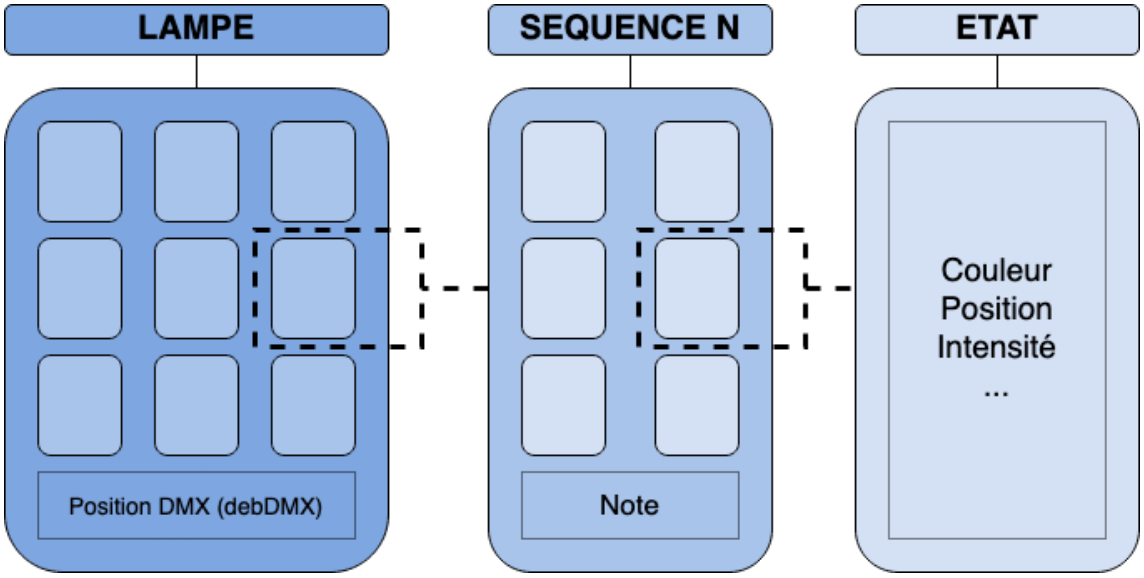

FIGURE 3 – Schéma d'une lampe comportant 9 séquences de 6 états

Dans la partie du programme menant à l'initialisation des variables, le code étant écrit en Python, nous avons utilisons une stratégie orienté objet en définissant les classe Lampe, Sequence et État, puis par héritage on spécifie on crée des états spécifiques à chaque lampe.

La partie de gestion des lampes par la carte Nucléo en C++ doit être systématique quelque soit le modèle utilisé. En effet ou souhaite compiler et insérer une seule fois le code dans le micro-contrôleur puis qu'il s'adapte en fonction du fichier insérer dans l'espace mémoire. Ceci nous a amené à définir en amont un grand nombre de variable et de fonction dont le comportement est identique. Le tableau suivant introduit les notations :

Notation du programme en C (les valeurs entre parenthèse sont remplacées par des nombres entiers)	
L(k)debDMX	début de la séquence de la lampe n°k
L(k)tick_S(p)	ticker contrôlant la lampe k et déclenchant la séquence n°p
L(k)fonctionS(n)	fonction appelé par le ticker précédent
L(k)NoteS(p)	note associé au ticker
L(k)S(p)	liste des états concaténées de la séquence p de la lampe k
L(k)S(p)E(q)	état n°q de la séquence n°p de la lampe n°k

3.3 Programme de conversion MIDI/DMX

Le coeur de cette section est de convertir un évènement sur le contrôleur MIDI en une action sur les projecteurs. La **Figure 4** schématise les différentes étapes. Les codes des fonctions présentées sont fournis en annexe.

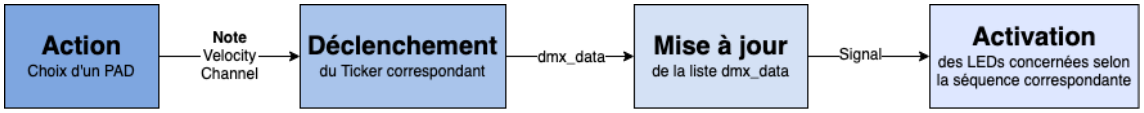

FIGURE 4 – Activation d'une séquence par contrôle MIDI

Lorsque nous appuyons sur une touche du contrôleur MIDI, le contrôleur envoie 3 informations : le channel (channel_data), la note (note_data) et la puissance d'appuie (velocity_data). Le programme enregistre ces valeurs et utilise uniquement la note de la touche en question. Les tests effectués prenaient uniquement en compte les boutons de type Beat bien que la fonction de détection (ISR_midi_in) permet de détecter les évènements issus par exemple d'un potentiomètre ou d'un fader.

La note caractérise un bouton et déclenche une fonction Ticker associée. qui modifie la liste dmx_data périodiquement. Chaque fonction Ticker est associée à un lampe, et une séquence. L'état de lampe à activer est déterminé à l'aide d'un compteur incrémenté après chaque passage. La fonction modifie alors localement la chaîne DMX en plaçant les information de l'état (**Figure 5**).

L'information est ensuite délivrée aux projecteur grâce à la fonction updateDMX(). Le protocole de communication DMX est décrit plus largement au lien suivant `https://ww2.ac-poitiers.fr/electronique/IMG/pdf/Le_DMX_512.pdf`. Pour une chaine DMX de 512 informations, le temps minimal de mise de rafraîchissement est de 23 ms.

3.4 Programme d'écriture et lecture de la carte micro-SD

L'initialisation des variables des lampes et est effectuée au début du programme grâce à la lecture d'un fichier texte (data.txt). Cette partie est réalisée par la carte Nucléo et codée en

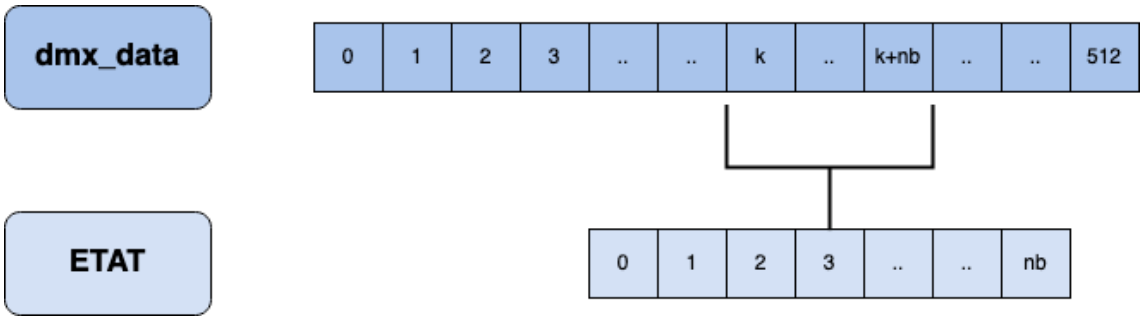

FIGURE 5 – Modification de la chaîne DMX par activation d'un ticker (k = debDMX, nb = NB_INFO)

C++. Ce fichier est généré en amont par un programme en Python à l'aide des informations issus du logiciel de paramétrage. La structure du fichier est présentée sur la **Figure 6** pour deux lampes comportant respectivement trois et une séquence. Les codes sont fournis en annexe. Leur complexité est linéaire avec le nombre de lampe : Le fichier est généré et lu de la première à la dernière ligne sans retour en arrière.

La lecture du fichier et l'initialisation des variables sont effectuées en amont de l'utilisation pour ne pas ralentir la dynamique de la carte Nucléo par des requêtes multiples sur le fichier.

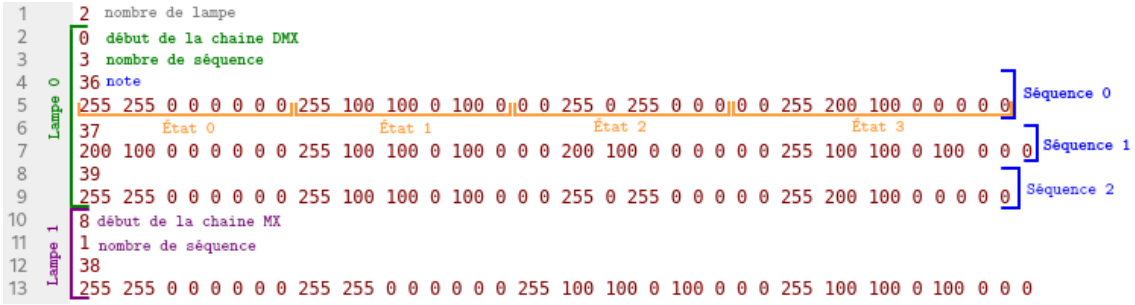

FIGURE 6 – Exemple d'un fichier d'initialisation pour deux lampes dont l'une contient trois séquences et le deuxième une

3.5 Interface Homme-Machine

L'interface Homme-Machine est un logiciel permettant le paramétrage des lampes en spécifiant l'ensemble des paramètres. Les informations données sont utilisées pour générer le fichier texte. Le logiciel est développé en Python : il renvoie une liste d'objet de la classe Lampe. C'est à ce niveau du programme que sont enregistrées les caractéristiques de chaque type de lampe en particulier l'ordre des informations d'un état (couleur, intensité...).

Une prémisse a été codé pour la bibliothèque Tkinter : un visuel du logiciel est proposé **Figure 7**.

4 Travail effectué et tests de validation

Le travail présenté se découpe en sous-systèmes (**Figure 1**). Chacun de ces sous-systèmes a pu être développé et testé séparément pour s'assurer de leur fonctionnement.

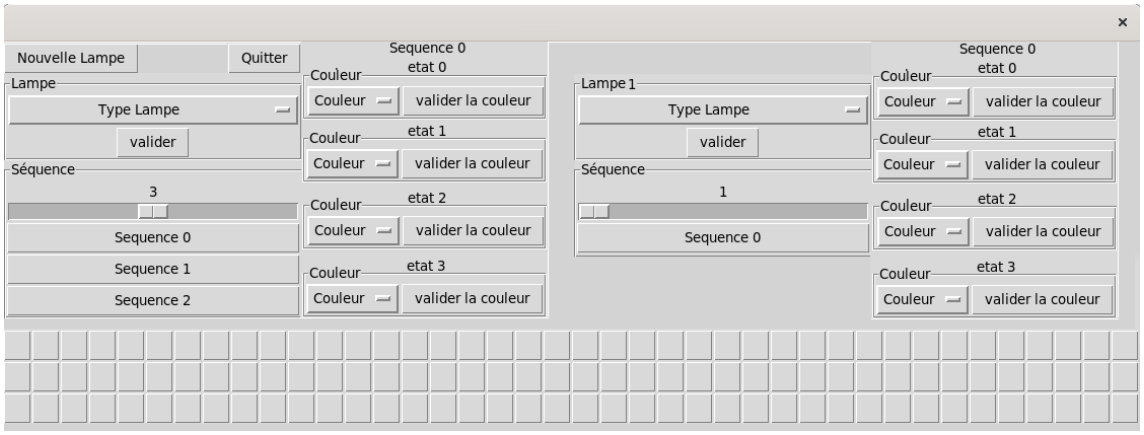

FIGURE 7 – Capture d'une première interface : les carrées permettent de placer les lampes dans la chaine DMX

Les premiers tests ont été effectués pour mettre à jours à intervalle régulier avec des Ticker la chaîne DMX et modifier l'état des lampes. Nous avons testé les fonctions de gestions des évènements du contrôleur MIDI. Puis nous avons associé les deux blocs pour déclencher la modification de la chaîne DMX par un évènement du contrôleur MIDI. Dans ces premiers temps nous devions écrire à la main des états pour les lampes et les ajouter directement dans le programme du microcontrôleur.
Parallèlement, nous avons écrit les fonctions générant le fichier texte et permettant la lecture et l'initialisation des variables. Les tests ont été réalisé sur des compilateurs GCC. L'intégration à la carte Nucléo fut une étape supplémentaire car il fallait ajouter un espace mémoire et réussir la communication pour ouvrir le fichier. Le programme accessible dans l'annexe est adapté à deux lampes contenants au maximum trois séquences de quatre états définit sur huit canaux. Les lampes sont des EuroliteLED Party Hybrid Spot commandées par sept canaux dont on commande l'intensité et les niveaux RVB et de blanc.

L'interface Homme-Machine n'étant pas terminé nous ne l'avons jamais intégré au reste du projet. Aussi nous avons créée à la main quelques États à intégrer aux classes Séquence et Lampe du programme Python. Des démonstrations ont été réalisées lors des deux dernières séances de travail.

5 Difficultés rencontrées

Les difficultés rencontrées ont été de différents types.

Dans un premier temps il fallait prendre en main la matériel et se familiariser avec les protocoles MIDI et DMX dont nous avions aucune connaissance préalable. On trouve plusieurs ressources en ligne mais elles sont souvent trop exhaustif. En particulier au début de projet où nos objectifs n'étaient pas clarifier, il fallait à la fois cibler les informations importantes et définir une stratégie qui dépend elle même des résultats de la recherche d'informations.

Une autre source de difficulté est arrivée pour définir la structure du programme et les variables importantes. Dans nos premiers tests nous travaillons sur la chaîne DMX en contrôlant systématiquement les projecteurs par groupe de quatre. Nous rencontrions des difficultés pour

rendre indépendant les projecteur. Une solution aurait pu être de travailler sur des groupes de quatre projecteurs et rendre indépendant la gestion des uns des autres.
Cette solution était plus embarrassante techniquement car elle nécessitaient au moins deux groupes de lampes (de préférence de même marque pour simplifier la définition des états). Par ailleurs elle complexifiait le code car nous réalisions des extractions dans des listes. Nous avons effectué un virage en redéfinissant la structure et en introduisant le formalisme présentée précédemment.
Ce formalisme est typique d'une programmation orientée objet. Or le langage C n'appartient pas à ce paradigme de programmation. La création de l'ensemble de variables alourdie fortement le programme. Pour notre exemple de lampe à trois séquences de quatre états nous créons une quarantaine de variable et les fonctions des tickers prennent chacune une vingtaines de lignes. Pour un plus grand nombre de lampe et de séquence le fichier principale comporte rapidement un très grand nombre de ligne et le rend illisible. Nous avons essayé de créer des bibliothèques spécifiques à chaque lampe sans parvenir à un résultat dans le temps qui nous était imparti pour ce projet.

6 Conclusion

Le projet proposé est une prototype fonctionnel d'un produit commercialisable. La solution proposée est conforme au cahier des charges fixée par l'entreprise. Il est possible de piloter indépendamment des groupes de projecteur différents selon des plusieurs modes. Une application de programmation a été amorcée permettant à terme la création des scénographies. La solution sera compacte et ne nécessite aucune compilation sur le micro-contrôleur. La modification de la scénographie est réalisable par une simple modification du fichier présent sur la carte mémoire.

Des perspectives sont envisageables comme la prise en compte d'autres phénomènes ou d'option en utilisant d'autres capacités des contrôleur MIDI ou des projecteurs. On peut par exemple imaginer contrôler l'intensité lumineuse par la pression sur les touches ou à l'aide des potentiomètres.
Le développement de ces perspectives sera grandement simplifier par une simplification du programme en résolvant les problèmes liées aux bibliothèques et en s'intéressant plus en détail aux possibilités du C++.

7 Annexes

7.1 Codes

L'ensemble des codes commentées ainsi que des test de fonctionnement sont disponibles à l'adresse suivante : `https://owncloud.institutoptique.fr/s/xA2qgs4x7oeJTsZ`

7.2 Sources

Les ressources disponibles sur le LEnsE pour le prototypage avec les carte Nucléo nous ont été utiles. En particulier les tutoriaux liés *Gestion du temps / Interruptions* pour le traitement de interuptions MIDI et les Tickers et la configuration d'un système de stockage microSD *Ajouter de la mémoire de données (SRAM) en SPI*. `http://lense.institutoptique.`

fr/prototyper-avec-nucleo-et-mbed-tutoriels-a-la-carte/

Les fonctions liées aux interfaces MIDI et DMX ont été inspirées par celles proposées par Julien Villemjane et disponibles en ligne sur son répertoire Mbed : https://os.mbed.com/users/villemejane/

Le cours de langage C de F. Goudail, S.Lebrun, P.Lugan, F. Kroeger, C. Garrido-Alzar disponible à l'Institut d'Optique a permis de raviver des notions de langage C vu en première année, notamment l'utilisation de fichier et des pointeurs. http://paristech.institutoptique.fr/cours.php?id=114

La partie en langage Python c'est appuyé principalement sur deux ouvrages : *Introduction à la programmation Python pour la biologie* de P Fuchs et P. Poulain disponible en ligne https://python.sdv.univ-paris-diderot.fr/ pour la partie de programmation orientée objet et le livre très complet de G Swinnen *Apprendre à programmer en Python 3* librement téléchargeable : http://inforef.be/swi/python.htm pour l'utilisation de la bibliothèque Tkinter.