Projet IETI
Son et lumière

François Bianchi - Rémi Garot-Jacquey - Elisa Jarry - Lucile Chanay - Corentin Lependu

1. Introduction

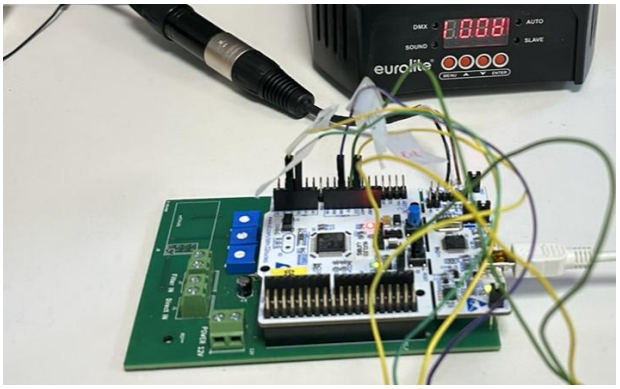
Le projet intitulé “Son et lumière” a, comme son nom l’indique, pour objectif de créer un dispositif permettant de fusionner à la fois la création de lumière et la création de son.
Pour cela, nous avons 4 éléments à notre disposition :

- Un clavier MIDI (reliable à une carte Nucléo)

- Une lyre (une LED RGB+UV pouvant tourner sur un axe de 360° en rotation et 180° en inclinaison. Elle peut être contrôlée depuis une Nucléo via un port d'entrée DMX, et reliée à d’autres lumières via un autre port DMX sortie).

- Un banc de LEDs (4 LEDs RGB, également contrôlables via Nucléo par des connexions DMX).

- Une carte Nucléo (en réalité deux comme nous avons travaillé simultanément sur les divers dispositifs) possédant un port DMX.



Notre objectif était de faire usage de ces 3 éléments (sans compter la Nucléo) afin de les combiner en 1 seul dispositif.
Ce dispositif avait au départ pour but de se baser sur le clavier MIDI. Le clavier produirait du son, et la lumière du banc LED réagirait en accord avec la note jouée. La lyre serait elle contrôlée séparément pour sa direction et sa couleur (ainsi que d'autres fonctionnalités : le stroboscope, allumage/extinction) via des joysticks.
Nous reviendrons en détail sur ces fonctionnements dans les parties correspondantes aux différents systèmes.

Au préalable, il a fallu que l'on se familiarise avec chacun des éléments à notre disposition. La répartition fut la suivante.

- Clavier MIDI + Banc de LED : Elisa Jarry, Rémi Garot-Jacquey
- Lyre : François Bianchi, Corentin Lependu, Lucile Chanay

Certains changements ont vus le jour au cours des séances:

- Le contrôle de la lumière de la lyre s'est vue passer d'un joystick à un potentiomètre
- Ajout d'une fonctionnalité Bluetooth pour la lyre : allumage/extinction de la lyre et de la fonctionnalité stroboscope (remplace l'utilisation de deux boutons analogique)
- Ajout d'une fonctionnalité "détection du BPM d'une musique et changement de couleur des LEDs en accord avec ce BPM" (sans utilisation du clavier MIDI) pour le banc de LEDs.

En conséquent, le montage "final" a changé et est devenu le suivant:

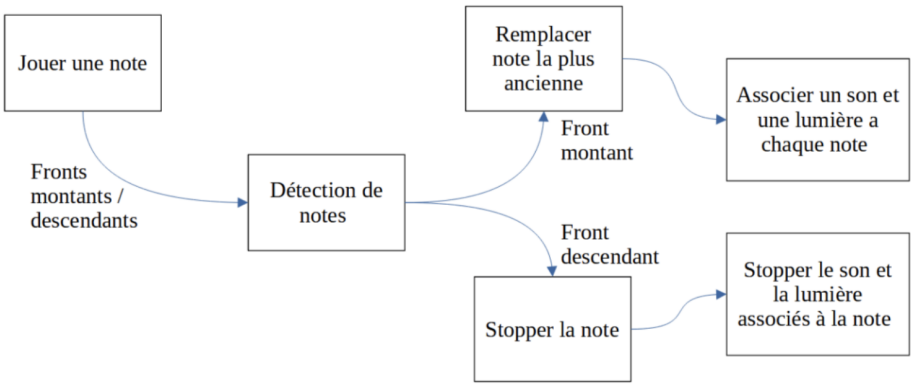
- Le banc de LEDs était relié à la Nucléo elle-même reliée à un montage de détection du BPM d'une musique via une prise Jack reliée à un smartphone. La lumière agit en rythme avec cette musique.
- La lyre est reliée à cette même Nucléo. Sa direction est contrôlée indépendamment via un joystick, mais la couleur suit le BPM de la musique (comme le banc de LEDs). La fonctionnalité Bluetooth n'est pas utilisée.
- Le clavier MIDI n'est pas utilisé.

Néanmoins, le montage "non-fusionné" conserve les fonctionnalités énoncées précédemment (Lyre contrôlable en bluetooth+joystick/potentiomètre, clavier MIDI et couleurs accordées…).

La partie qui suit s'intéresse au fonctionnement individuel de chaque dispositif, avant de s'intéresser à celui qui combine le tout. Les cahiers des charges respectifs pour chaque dispositif seront représentés par un schéma dans les parties correspondantes.

2. Description des systèmes

a. Clavier



Le but de cette partie de notre projet est de faire fonctionner un clavier MIDI et de notamment l'utiliser afin de contrôler des LED. Chaque touche du clavier sera alors associée à une note bien sûr, mais aussi à une couleur.

L'objectif que l'on se fixe est de pouvoir jouer deux notes simultanément sur le clavier. Le son des deux notes sera alors envoyé sur les haut-parleurs. Sur les quatres spot LED que nous contrôlons, deux s'allumeront de la couleur correspondant à la première note, et les deux autres à la seconde note.

i. Le montage

Le montage électrique de cette partie est très simple. L'élément central est bien sûr la carte Nucleo adaptée à la gestion d'un clavier MIDI.

- En entrée de la carte: le clavier. Celui-ci doit être relié via le câble MIDI.
- En sortie de la carte : Les spots LED reliés par un câble DMX d'une part, les deux sorties D3 et D5 reliées aux deux entrées des hauts-parleurs d'autre part.

ii. Fonctionnement général

La partie la plus importante de cet aspect de notre projet est l'écriture du programme de gestion des différents éléments du montage.

Le cœur de ce système est l'association de chacune des touches du clavier à un son et une couleur. Ceci est réalisé via deux listes que l'on définies :

- Une liste qui contient autant de fréquences sonores qu'il y a de touches sur le clavier. On les choisit comme correspondant typiquement aux fréquences des touches d'un piano classique.
- Une liste contenant une code RGB pour chacune des touches du clavier. La liste est donc définie comme suit: [R_1,G_1,B_1,R_2,G_2,B_2,...]

Notre programme s'appuie sur un certain nombre de fonctions prédéfinies qui permettent de communiquer avec le clavier MIDI d'une part et avec les LED d'autre part.

De manière générale, on définit une fonction "`clavier()`" qui détecte les notes jouées au clavier et agit en conséquence sur les LED et les haut-parleurs. Dans le "`main()`", après l'initialisation des sorties et variables, cette fonction de détection est appelée en continu afin de jouer les notes en temps réel.

iii. Détail de la gestion du son et de la lumière

Dans cette partie, nous allons voir dans le détail comment est détectée l'information d'une note et comment on la traduit en une information pour les LED et haut-parleurs.

La fonction qui nous permet de détecter les notes est une fonction prédéfinie: "`isNoteMIDIdetected()`" Cette fonction renvoie TRUE si une touche est enfoncée ou relevée, et FALSE sinon.

La fonction qui nous permet de lire la touche qui à été jouée est également une fonction prédéfinie:
"`playNoteMIDI2(note_data, velocity_data)`" Cette fonction nous permet de connaître le 'numéro' de la touche jouée et si elle à été enfoncée ou relevée.

Afin de jouer un son, on écrit simplement:

```
speaker1.period(1.0/liste_freq[index]);
speaker1.write(0.5);
```

Cela permet de choisir la fréquence à laquelle la tension sur les sorties digitales (entrées des haut-parleurs) va varier. Pour permettre une variation de cette tension, on définit un rapport cyclique de 0.5.

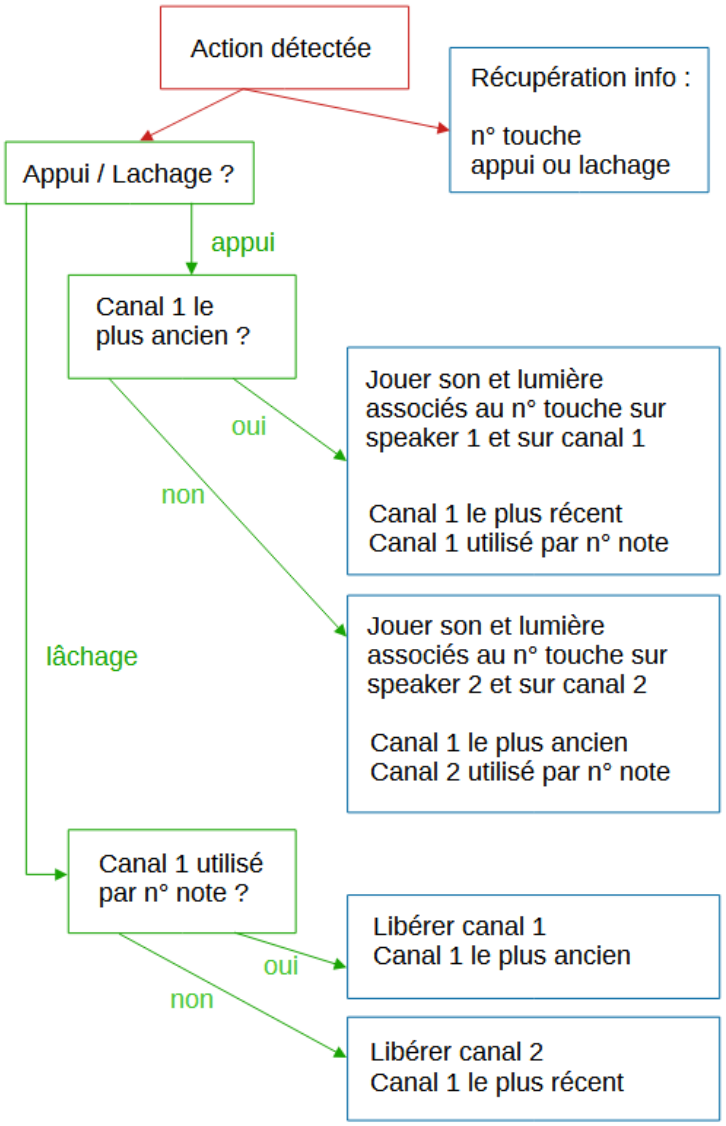
Afin de créer une couleur sur les LED, on écrit:

```
dmx_data[1]=liste_color[3*index];
dmx_data[2]=liste_color[3*index+1];
dmx_data[3]=liste_color[3*index+2];
updateDMX();
```

La fonction prédéfinie "`updateDMX()`" indique au LED l'intensité du rouge, vert et bleu à afficher respectivement à partir des valeurs qui sont stockées dans dmx_data[1], dmx_data[2] et dmx_data[3].

iv. Gestion des channels

La dernière partie importante de notre programme est l'organisation de la fonction "`clavier()`" et sa gestion des deux canaux disponibles pour jouer les notes détectées. Le fonctionnement de cette fonction peut être bien résumé par le diagramme ci-joint.



Pour mettre en pratique les conditions présentées dans le diagramme précédent, on s'appuie sur trois variables: `isChannel1Old`, `channel2UsedBy` et `channel1UsedBy`.

L'information stockée par ces variables est changée à chaque fois qu'une note est jouée. La première variable permet de savoir sur quel canal est la note la plus anciennement jouée. Les deux autres permettent de savoir qu'elle note est jouée par chacun des canaux afin de pouvoir les libérer si le "relevage" de la touche correspondante est détectée.

Pour produire deux notes simultanément sur les haut-parleurs, on utilise les variables "speaker1" et "speaker2" qui correspondent aux sorties digitales D3 et D5 respectivement.

Pour faire deux couleurs simultanément, on décide de faire une couleur sur deux des LED et une autre sur les deux autres. Pour cela, on place deux des LED sur le canal 1 et deux autres sur le canal 11.

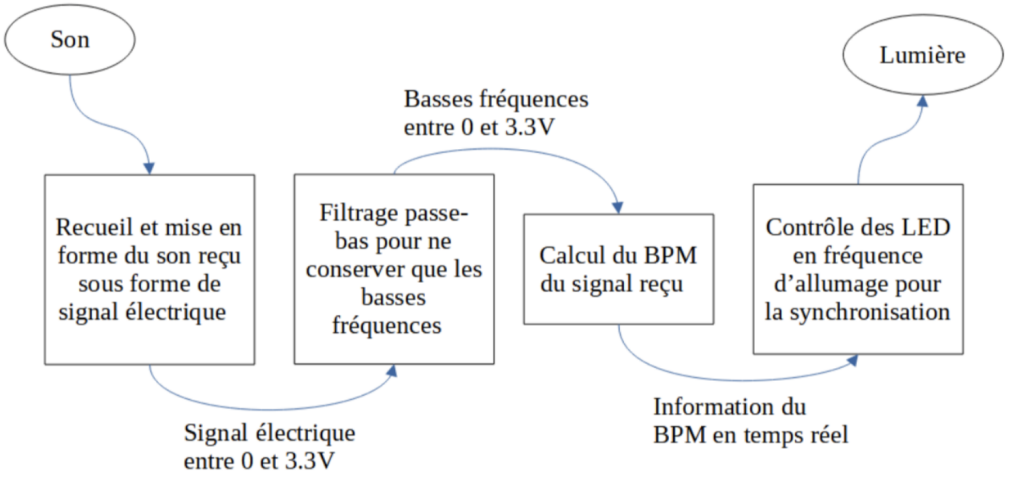
Ainsi, la commande suivante sera lue comme l'intensité du rouge par les sur le canal 1 et ignorée par les autres :

```
dmx_data[1]=liste_color[3*index];
```

A l'inverse, la commande suivante sera lue comme l'intensité du rouge par les sur le canal 11 et ignorée par les autres:

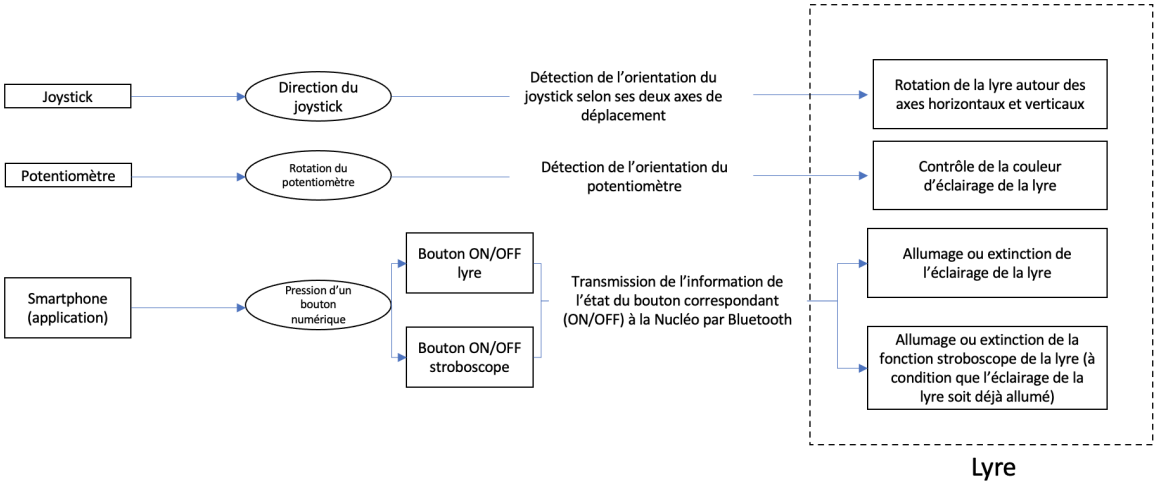
```
dmx_data[11]=liste_color[3*index];
```

b. BPM



Le système BPM avait pour but de contrôler la couleur des LED de façon synchronisée avec les BPM de la musique. Pour se familiariser avec la détection de BPM, nous avons choisi de faire un premier code avec une carte Nucléo et une entrée de son remis en forme adapté pour la Nucléo. Une acquisition de la tension correspondant à l'intensite sonore est effectuée à intervalles réguliers, toutes les 40us, puis moyennée sur une petite suite de 64 mesures ("séquence"). Au bout d'une seconde, un calcul de la moyenne et de l'écart-type de la série des séquences est réalisée. Chaque nouvelle valeur de séquence sera comparée avec la moyenne et l'écart-type pour déterminer si on constate un pic d'intensité sonore, qui conduira à un changement aléatoire de la lumière.
Si un changement de lumière doit avoir lieu, la couleur (Rouge, Vert ou Bleu) est choisie aléatoirement, et son intensité (de 0 à 255) variera de 70 ou 10 selon si le pic sonore est important ou pas.

c. *Lyre*



En parallèle du clavier et du système de contrôle des LEDs à partir du BPM de la musique, nous avons travaillé sur une Lyre. La lyre reçoit des informations numériques via un câble DMX (entrée “in”). Elle possède une adresse (un nombre entre 0 et 512) qui servira de référence pour l’information DMX, ainsi qu’un nombre de channels qui correspond au nombre de paramètres modifiables (Pour la lyre, on a par exemple 16 channels, avec le premier correspondant au mouvement horizontal, le deuxième à celui vertical…).
Le câble DMX permet d’envoyer des informations sur 512 différents canaux. Quand on envoie une information à la lyre par le câble DMX, il faut donc

- Envoyer l’information au bon dispositif (d’où l’adresse de la lyre)
- Changer le paramètre qui nous intéresse (d’où les différents channels) sur une échelle de 0 à 255.

Autrement dit, l’’adresse de la lyre sert de “point de référence” pour l’information DMX, et les channels à savoir quel paramètre modifier.
Par exemple, si on veut changer le mouvement horizontal de la lyre, il faudra envoyer l’information DMX sur le canal numéro “Adresse de la lyre” + “Channel correspondant”.
Imaginons que la lyre est adressée à 40, et qu’on veut mettre de la couleur rouge à son intensité maximale (channel 6), il faut donc que le câble DMX envoie l’information “255” au canal 40 + 6 donc 46.
Nous y reviendrons juste après dans le code, mais à titre d’exemple, cela revient à envoyer “dmx_data[46] = 255” à la Nucléo.

Toutes les fonctionnalités que nous avons tenté de mettre en place nécessitent une carte Nucléo et donc un code (écrit sous Mbed, accessible en annexe). Dans ce code, il existe deux fonctions indispensables : initDMX, qui permet d’initialiser la Lyre dans un état imposé par le code, et la fonction updateDMX, qui permet de mettre à jour les différents paramètres de la Lyre. La fonction initDMX est appelée une seule fois dans le main. En revanche, la fonction updateDMX est placée dans la boucle while(1) de la fonction main et est donc

appelée à de nombreuses reprises (ce qui est logique, puisqu'on veut adapter les paramètres de la Lyre en continu en fonction des interactions avec des joysticks, des potentiomètres…). Cette fonction est toujours appelée à la fin des programmes qui seront présentés ci-dessous.

i. Direction

Le premier objectif concernant la Lyre était de contrôler sa direction à l'aide d'un joystick. Nous avons choisi de faire en sorte que bouger le joystick selon son axe « x » permette de faire tourner la Lyre autour de l'axe vertical (i.e. on fait tourner la Lyre vers la droite ou vers la gauche). Bouger le joystick selon son axe « y » permet ensuite de faire tourner la Lyre autour de l'axe horizontal (i.e. on fait tourner la Lyre vers le haut ou vers le bas).

Pour arriver à ce fonctionnement, on passe par une carte Nucléo. Elle permet d'envoyer des instructions à la Lyre grâce à un connecteur DMX en temps réel à partir des informations qu'elle reçoit. Plus précisément, ce sont les channels 1 et 2 (la lyre est en mode 16 channels) qui transmettent ces informations. On branche également le joystick aux différentes broches de la carte, de sorte que la carte reçoit les données du joystick. Dans le code, on associe les informations sur la position du joystick reçues par la carte Nucléo à des ordres contrôlant la direction de la Lyre. Le code impose à la carte de lire deux entrées analogiques, qui sont reliées aux broches notées « x » et « y » du joystick. On alimente également le joystick avec la sortie de la carte fournissant 3,3V et on le relie à la masse.

Pour plus de précision sur le fonctionnement du système, on peut commencer par regarder le code plus en détail. Celui-ci est disponible en annexe. On voit qu'on associe les broches A2 et A3 (deux entrées analogiques) de la carte Nucléo respectivement aux axes "x" et "y" du joystick à l'aide des variables x1_axis et y1_axis. Plus tard dans le code, dans la fonction main, on accède à la valeur de la position du joystick à l'aide de la fonction read. Cette fonction renvoie une valeur comprise entre 0 et 1 en fonction de la position du joystick sur l'axe (on stocke ces valeurs dans les variables x1_data et y1_data). Pour commander à la Lyre d'être dans une position particulière, on doit donc lui fournir deux valeurs entre 0 et 255, une correspondant au pan, l'autre au tilt. Dans le code, dmx_data[0] représente le pan et dmx_data[1] représente le tilt. On dit alors que la valeur du pan est donnée par x1_data*255. De cette manière, on a bien une valeur variant entre 0 et 255 en fonction de la position du joystick selon son axe "x". De même, la valeur du tilt est donnée par y1_data*255.

On note que la position de référence de la Lyre (i.e. la position qu'elle prend lorsqu'on lâche le joystick) est la position verticale.

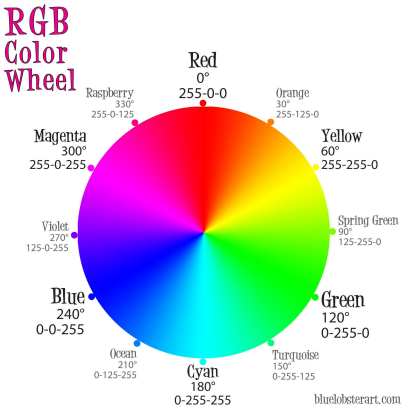
Les tests de cette fonction étaient concluants : en bougeant le joystick, la Lyre tournait de manière contrôlée. Pendant l'écriture du code, pour comprendre quelle information la carte Nucléo recevait du joystick, nous avons affiché sur TeraTerm les variables x1_data et y1_data.

Pour intégrer cette fonctionnalité au système final, on intègre juste les parties du code correspondantes dans le code global. On veillera à changer l'adresse de la lyre (et donc le numéro du DMX_data à changer) afin de s'assurer que les adresses correspondant au fonctionnement de la bande LED ne se superposent pas à celles pour le contrôle de la lyre.

ii. Couleurs

Dans un second temps, nous avons cherché à contrôler les couleurs de la Lyre. Nous pensions au départ ajouter un second joystick et faire en sorte que se déplacer le long de l'un des axes du joystick changerait la couleur de la Lyre. Néanmoins, nous avons abandonné cette idée, trouvant plus pratique d'utiliser un potentiomètre pour cette fonctionnalité. Le principe global est alors le suivant : on branche le potentiomètre à une entrée analogique de la carte Nucléo (en plus de l'alimenter en tension et de le relier à la masse du circuit). En fonction de la position du potentiomètre, la tension lue par la carte sur l'entrée analogique est différente. La sortie DMX de la carte Nucléo permet alors de renseigner à la Lyre la couleur correspondant à la tension lue.

Il a donc fallu trouver un moyen de relier la tension à la couleur. Nous avons choisi de faire en sorte que toutes les couleurs du cercle chromatique soient accessibles en tournant le potentiomètre. On sait que la tension lue par l'entrée analogique de la carte Nucléo est comprise entre 0 et 1.
De plus, chaque couleur est représentée sur le cercle chromatique : on peut alors associer à chaque couleur un angle de 0 à 360°. En multipliant la tension lue par 360, on obtient un nombre entre 0 et 360, que l'on associe à la couleur correspondante sur le cercle chromatique.



Il reste finalement à relier l'angle et la couleur correspondante sur le cercle chromatique. Pour cela, on applique les règle suivantes :

- Entre 0° et 60°, l'intensité dans le rouge est maximale et on augmente progressivement l'intensité dans le vert jusqu'à l'intensité maximale (aucune intensité dans le bleu)
- Entre 60° et 120°, on garde le vert à son intensité maximale et on diminue l'intensité dans le rouge jusqu'à 0
- Entre 120° et 180°, l'intensité dans le vert est maximale et on augmente progressivement l'intensité dans le bleu jusqu'à l'intensité maximale
- Entre 180° et 240°, on garde le bleu à son intensité maximale et on diminue l'intensité dans le vert jusqu'à 0
- Entre 240° et 300°, l'intensité dans le bleu est maximale et on augmente progressivement l'intensité dans le rouge jusqu'à l'intensité maximale
- Entre 300° et 360°, on garde le rouge à son intensité maximale et on diminue l'intensité dans le bleu jusqu'à 0

On applique le principe décrit ci-dessus dans le code (disponible en annexe). La broche branchée au potentiomètre est référencée par l'AnalogIn nommée color. La tension en sortie du potentiomètre est alors référencée par la variable color_data. Enfin, l'angle correspondant à une couleur du cercle chromatique est représenté par angle_couleur. Les valeurs des intensités des couleurs rouge, vert et bleu correspondent respectivement aux varaiables R, G et B. On transmet ensuite la valeur de ces intensités (comprises pour chaque couleur entre 0 et 255) à la Lyre via la variable dmx_data.

De même que pour la direction, nous avons pu tester le fonctionnement en observant directement le résultat. En tournant le potentiomètre, la couleur changeait bien de manière graduelle, en passant par toutes les couleurs du cercle, comme cela était attendu. Durant la réalisation du circuit, nous avons eu besoin de vérifier que la tension récupérée par la sortie analogique reliée au potentiomètre était bien comprise entre 0 et 1. Nous avons donc une fois de plus affiché cette tension sur TeraTerm et vérifié que nous avions bien ce que nous voulions.

Ce système d'évolution des couleurs avec le potentiomètre n'a cependant pas été conservé dans le montage final. Nous avons finalement préféré faire en sorte que les lumières de la Lyre soient synchronisées avec les lumières des LEDs.

iii. Boutons allumer / éteindre

L'un des objectifs initiaux était de faire en sorte qu'appuyer sur le bouton du joystick permette d'allumer et éteindre la Lyre.
Le principe global de fonctionnement est simple : on relie la sortie correspondant au bouton du joystick à une broche de la carte Nucléo. Quand la carte détecte un front montant (i.e. quand on a appuyé sur le bouton), on change dans le code la valeur de dmx_data[3] qui contrôle l'intensité de la lumière émise par la Lyre. Si la Lyre était précédemment allumée, on met 0 pour cette variable. Dans le cas inverse, on met 255 (qui correspond à la valeur maximale de l'intensité).

Nous avons néanmoins rencontré un problème. En effet, lorsqu'on appuyait sur le bouton, il arrivait que les lumières ne s'éteignent ou ne s'allument pas : le système semblait marcher qu'une fois de temps à autre, de manière irrégulière. De plus, nous observions parfois que la Lyre clignotait, comme si elle détectait deux fronts pour un seul appui du bouton. Nous avons observé ce qu'il se passait lorsqu'on appuyait sur le bouton à l'oscilloscope pour comprendre le signal envoyé à la carte Nucléo lorsque l'on appuie sur le bouton du joystick. Nous avons alors observé un signal en créneau. Nous nous sommes dit que la carte lisait peut-être mal le signal envoyé par rapport à ce que nous souhaitions et qu'elle considérait un front (montant ou descendant) comme une instruction "allumer/éteindre la Lyre". Ainsi, lorsque la carte captait les deux fronts du créneau, elle changeait une première fois la valeur de l'intensité au front montant puis une deuxième fois au front descendant, ce qui entraînait le clignotement des lumières. Nous avons alors tenté de modifier le code en prenant ce principe en compte (en indiquant de changer l'intensité seulement si deux fronts étaient détectés) et le système marchait un peu mieux. Malgré tout, appuyer sur le bouton ne permettait pas à chaque fois d'allumer ou éteindre la lumière. Nous avons supposé que la détection des fronts montants et descendants n'était pas optimale. Nous n'avons pas réussi à trouver de solution satisfaisante pour pallier au problème.

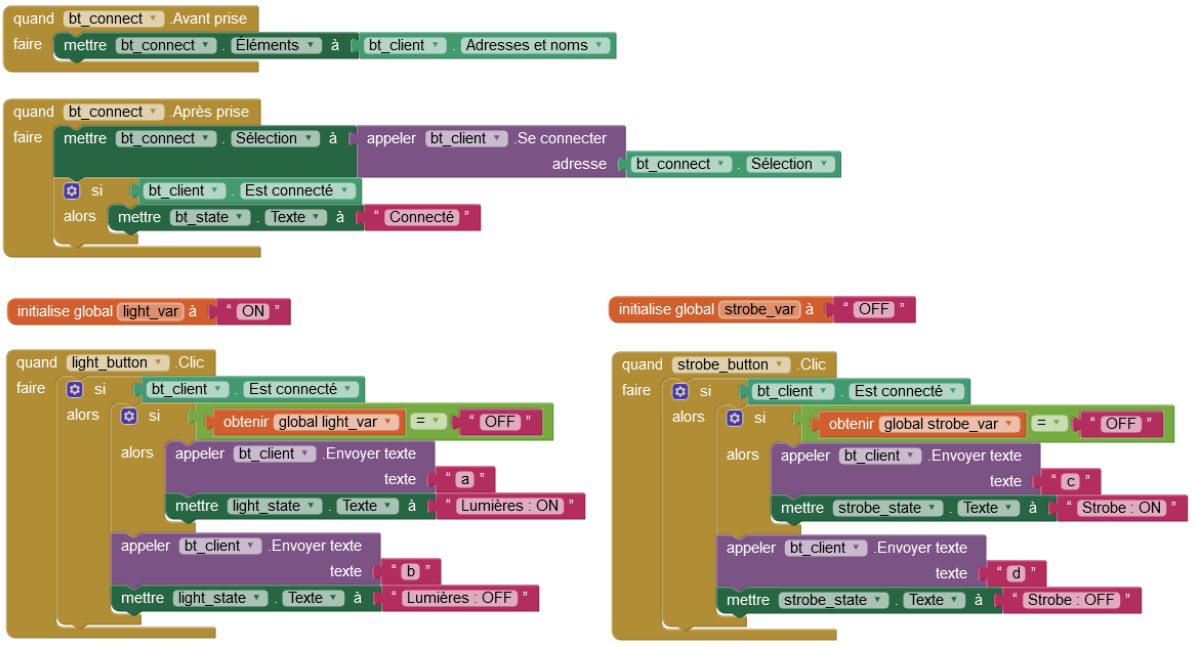
De fait, ce système n'a finalement pas été retenu dans le montage final et n'apparaît plus dans le code rendu en pièce jointe. Comme nous avions remplacé le joystick qui devait initialement contrôler les couleurs par un potentiomètre, nous n'avons pas non plus de code correspondant au démarrage ou à l'arrêt de la fonction stroboscope. Nous avons finalement décidé d'implémenter les fonctionnalités "allumer et éteindre la Lyre" ainsi que "allumer et éteindre le stroboscope" dans le système sans fil présenté ci-dessous, qui ne comportait pas de bugs. Ainsi, les fonctionnalités restent accessibles à l'utilisateur.

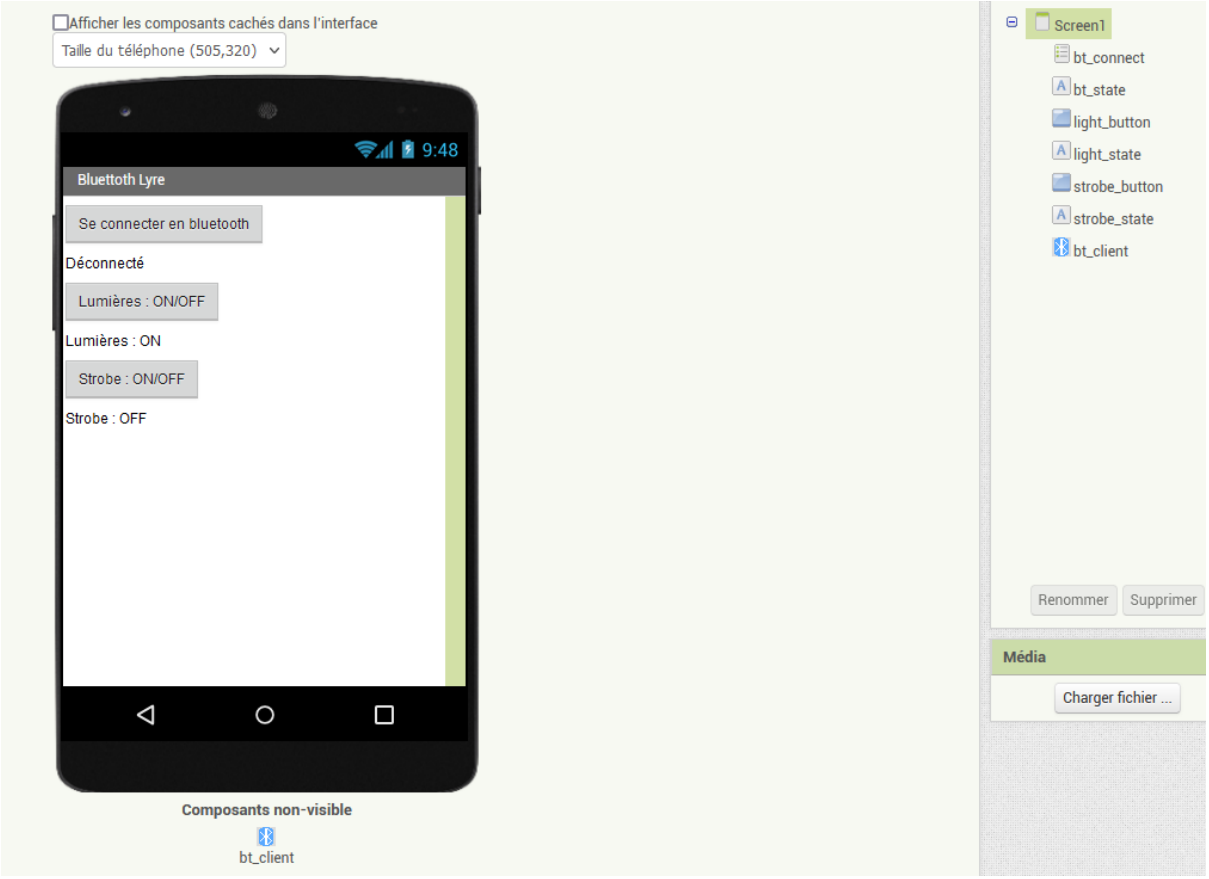
iv. Bluetooth

Le dernier objectif, qui nous apparaissait comme l'objectif ultime pour la lyre, était d'implémenter dans une application sur smartphone l'ensemble des fonctions que nous avons créées, pour ensuite tout contrôler à distance à l'aide d'un module bluetooth. Cela comprenait au début les joysticks pour gérer la direction et l'inclinaison ainsi que les différents boutons et sliders pour gérer la couleur ou le strobe par exemple. Nous utilisons un module bluetooth HC-05 pour recevoir les données bluetooth venant du téléphone et le site web https://appinventor.mit.edu/explore/get-started pour pouvoir créer l'application.

Les branchements pour le module HC-05 sont les suivants : 5V de la carte nucléo sur la patte VCC, GND relié au ground de la carte nucléo et TXD relié à une entrée digitalin de la carte nucléo. ATTENTION dans le code, l'entrée assignée à RX (receive) est bien reliée à la broche TX (transfer) du module HC-05. Ce petit problème nous a coincé pendant deux séances.

Pour l'application, le code que nous avons réalisé sur le site appinventor est le suivant :

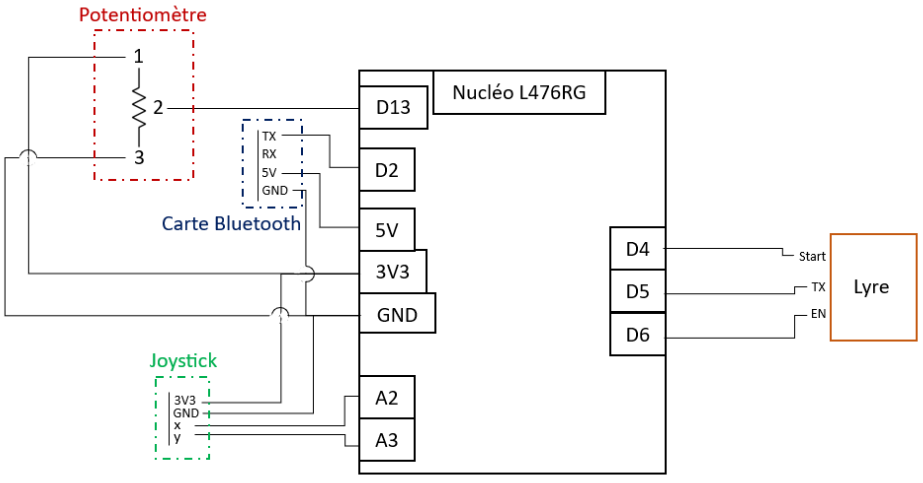




Les fonctions sont assez simples mais ne permettent d'envoyer que des lettres ou des chiffres de 0 à 9 au module bluetooth, du moins nous n'avons pas réussi à faire mieux. Nous avons donc abandonné la gestion de l'orientation et de l'inclinaison de la lyre à l'aide du bluetooth et avons gardé uniquement une fonction ON/OFF pour l'éclairage et le strobe.

v. Branchements du montage complet

On représente en figure ci-dessous le schéma des branchements réalisés ainsi qu'une photo du montage final comprenant toutes les fonctionnalités de la Lyre (joystick contrôlant la direction, potentiomètre permettant de faire varier la couleur, et système Bluetooth).



d. BPM avec Lyre (assemblage du système complet)

3. Bilan technique et moral

a. Partie clavier - BPM

Le fonctionnement du clavier respecte de manière globale les attendu du cahier des charges:

- chaque touche est associée à une note et une couleur
- les notes peuvent être jouées en temps réel
- deux notes peuvent être jouées simultanément

En revanche, lorsque deux actions sont réalisées trop rapidement (appuyer sur une touche et en relâcher une autre simultanément), le programme n'a pas le temps de détecter ces actions séparément et une note peut continuer d'être jouée alors que la touche à été relâchée. Améliorer l'efficacité du programme serait une bonne piste pour obtenir un fonctionnement plus fluide du piano.

b. Partie Lyre

Le fonctionnement de la lyre respecte les grandes lignes du cahier des charges, néanmoins, comme expliqué en introduction, des modifications ont été apportées au dispositif final… Récapitulons cela à travers un tableau :

	Direction
Etat	Fonctionnel
Commentaires et points d'amélioration	- La lyre est capable de tourner selon un angle de 360° en rotation et 180° en inclinaison. - La lyre a tendance à "trembler" quand on ne touche pas le joystick (sensibilité trop élevée du dispositif qui interprète sans doute du bruit électrique comme un déplacement du joystick) - Braquer le joystick dans une direction de rotation la fait tourner au-delà de 360°, ce qui rend parfois le contrôle un peu difficile (on tourne "plus que l'on voudrait"). - La lyre reprend sa position de référence dès lors que l'on lâche le joystick (au lieu de garder en mémoire celle juste avant).

	Couleurs
Etat	Fonctionnel mais manque de certaines fonctionnalités

Commentaires et points d'amélioration	- Le potentiomètre est capable d'afficher toutes les couleurs du cercle chromatique lorsqu'on le fait tourner - L'usage du potentiomètre au lieu d'un joystick a fait perdre la possibilité d'avoir un second bouton dédié à l'allumage/éteignage de la fonctionnalité stroboscope. Elle a néanmoins été implémentée grâce au Bluetooth (bouton sur le téléphone). - La fonctionnalité allumage/éteignage des lumières sur le joystick ne fonctionnait pas de manière satisfaisante : appuyer sur le bouton allumait parfois la lumière, parfois non. Pareil lorsqu'il s'agissait de l'éteindre. Malgré plusieurs tentatives de debogage, aucune n'a marché, et la "solution" trouvée était encore une fois d'introduire un bouton numérique via Bluetooth

	Bluetooth
Etat	Fonctionnel mais manque de certaines fonctionnalités
Commentaires et points d'amélioration	- La lyre peut s'allumer/éteindre à distance via bluetooth à l'aide d'une application sur Androïd. - Aucune solution n'a été trouvée pour gérer la direction et la couleur de la lyre. En effet, la carte Nucléo n'arrivait à lire qu'un seul caractère (lettre abcd… ou chiffre de 0 à 9), ce qui empêchait de lui transmettre des informations de 0 à 255 nécessaires pour contrôler la direction de la lyre. - La fonctionnalité Bluetooth nous a pris beaucoup de temps (2 séances entières dédiées) suite au problème mentionné ci-dessus.

Globalement, le système respecte donc le cahier des charges. Il y a néanmoins eu des changements (joystick 2 qui devient un potentiomètre), ainsi que des fonctionnalités améliorables.

c. Assemblage du système complet

Rassembler les deux systèmes (synchronisation avec le BPM + contrôle de la Lyre) ne faisait pas partie du cahier des charges initial. Le système fonctionne correctement. Il aurait été intéressant d'implanter le système Bluetooth au système final pour pouvoir ajouter des fonctionnalités.

d. Retour d'expérience de l'équipe

Le travail en groupe a été efficace tout au long du projet. La division en deux sous-groupes (l'un travaillant sur la Lyre, l'autre sur le clavier puis sur le système de synchronisation des LED avec le BPM) a permis à chaque personne de s'investir activement dans le projet. L'emploi du temps réalisé au début du projet a globalement été tenu, malgré quelques difficultés (réussir à faire marcher le Bluetooth nous a finalement pris deux séances entières par exemple) ou quelques évolutions.

À l'occasion de ce projet, nous avons acquis ou approfondi certaines compétences techniques. En particulier, nous avons pu nous améliorer en codage, nous avons découvert les systèmes MIDI / DMX, comment fonctionnait une carte Bluetooth… Le travail en groupe a également été une expérience enrichissante, qui a permis à chaque personne d'utiliser ses compétences pour le bien de l'équipe.

4. Conclusion

Malgré certaines difficultés techniques au cours du projet, nous avons réussi à le mener à son terme. Les systèmes finaux respectent globalement les cahiers des charges posés au début du projet. Le projet a évolué au cours des séances et est allé plus loin que ce que nous prévoyions au départ. Finalement, les systèmes mis en place permettent de contrôler la lumière des LED et / ou de la Lyre de différentes manière (en appuyant sur les touches du clavier MIDI, à l'aide d'un montage {carte Bluetooth, potentiomètre, joystick} et par une synchronisation du BPM). Ce projet a également permis un travail autour du son et de l'association son et lumière.

Annexe

Code de la partie clavier:

```
#include    "mbed.h"
#include    "DMX_MIDI.h"
#include    "projecteurs.h"
#include    "controleurs.h"
#include    <cstdint>
#include    <stdbool.h>

#define SAMPLES     512


Serial          debug_pc(USBTX, USBRX);
DigitalOut      debug_out(D13);

//Clavier
PwmOut          speaker1(D3);
PwmOut          speaker2(D5);

//Définition de la fonction qui fait jouer les notes
void clavier(void);

//On initialise les différents canaux sonores comme étant libres
```

```
char channel1UsedBy=0x00;
char channel2UsedBy=0x00;
\\On initialise le premier canal comme étant celui à utiliser
bool isChannel1Old=true;

int debut_clavier = (int)0x2C; //On définit la tonalité du clavier

// On définit une liste des fréquences qui seront associées à chaque
note
float
liste_freq[]={130.8,138.6,146.8,155.6,164.8,174.6,185,196,207.6,220,233
,246.9,262,277,294.6,311.2,330,349,370,392,415,440,466,494,523,554,589,
622,660,698,740,784,831,880,932,968,1046};

// On définit une liste des codes R,G et B des couleurs associée à
chaque note
int
liste_color[]={175,50,60,180,52,29,202,71,19,174,134,8,75,139,40,41,141
,57,49,95,105,53,100,164,52,70,123,56,63,118,108,55,73156,34,107,207,60
,55,230,64,35,235,76,13,205,155,3,105,171,30,30,158,60,40,111,108,50,11
5,207,59,82,178,60,62,154,115,53,101,183,35,135,236,61,68,234,65,40,230
,83,10,244,187,4,159,216,17,29,154,57,35,125,107,55,123,210,54,89,179,6
2,67,159,115,49,119,219,37,150,231,59,68};
int index; // Variable qui contient l'indice de chaque note qui nous
permettra de récupérer le son et la couleur dans les listes freq et
color

// Main
int main(){
      //Initialisation
      debug_pc.baud(9600);
      initDMX();
      initMIDI();

      speaker1.write(0);
      speaker2.write(0); //On ne joue aucun son au début
      srand(time(NULL));

      dmx_data[0] = 255; dmx_data[1] = 0; dmx_data[2] = 0;
      dmx_data[3] = 0; dmx_data[4] = 0; dmx_data[5] = 0;
      dmx_data[6] = 0; //Initialisation des DMX des 1ere LED
      dmx_data[10] = 255; dmx_data[11] = 0; dmx_data[12] = 0;
      dmx_data[13] = 0; dmx_data[14] = 0; dmx_data[15] = 0;
```

```
      dmx_data[16] = 0; // Initialisation des DMX des 2emes LED
      updateDMX();

      //on fait les mesures tous les 40us
      clavier();
      while(1){
      }
}

void clavier(void){
      if(isNoteMIDIdetected()){
            playNoteMIDI2(note_data, velocity_data); \\ Le "numéro" de
la touche et l'information de si elle est enfoncée ou non sont
récupérés
            if(velocity_data){ // Si on enfonce sur la note, on va
mettre le son et les lumières
                index=(int)note_data-debut_clavier;
                if(isChannel1Old){ //Si le canal 1 est libre OU si on
occupe déjà le 1 et le 2, alors on utilise le 1


             // Son à la fréquence selectionée dans la liste
                    speaker1.write(0);
                    speaker1.period(1.0/liste_freq[index]);
                    speaker1.write(0.5);
             // Lumière dont les codes RGB sont récupérés dans la liste
                    dmx_data[1]=liste_color[3*index];
                    dmx_data[2]=liste_color[3*index+1];
                    dmx_data[3]=liste_color[3*index+2];
             //On indique qu'on occupe le channel 1
                    channel1UsedBy=note_data;
                    isChannel1Old=!isChannel1Old;
                    updateDMX();
                }
            //Sinon on force l'utilisation du channel 2
                else{
                //On met le son
                    speaker2.write(0);
                    speaker2.period(1.0/liste_freq[index]);
                    speaker2.write(0.5);
                //On met la lumière
                    dmx_data[11]=liste_color[3*index];
                    dmx_data[12]=liste_color[3*index+1];
```

```
                    dmx_data[13]=liste_color[3*index+2];
                //On indique qu'on occupe le channel 2
                    channel2UsedBy=note_data;
                    isChannel1Old=!isChannel1Old;
                    updateDMX();
                }
            }
        //Si on relève la note, on va enlever le son et les lumières
            else{
            //Si on utilisait le channel 1, c'est lui qu'on libère
                if(channel1UsedBy==note_data){
                    //On reset le son
                    speaker1.write(0.0);
                    //On reset les lumières
                    dmx_data[1]=0;
                    dmx_data[2]=0;
                    dmx_data[3]=0;
                    //On indique que le channel est libre
                    channel1UsedBy=0x00;
                    isChannel1Old=true;
                    updateDMX();
                }
                //Si on utilisait le channel 2, c'est lui qu'on libère
                else{
                    //On reset le son
                    speaker2.write(0.0);
                    //On reset les lumières
                    dmx_data[11]=0;
                    dmx_data[12]=0;
                    dmx_data[13]=0;
                    //On indique que le channel est libre
                    channel2UsedBy=0x00;
                    isChannel1Old=false;
                    updateDMX();
                }
            }
            resetNoteMIDI();
        }
}
```