ARNAUDET Antoine
SENAFFE Vivian
IMA4SC

Orchestre électronique

Tuteurs : PRIEUX Lucas VINOT Hidéo REDON Xavier VANTROYS Thomas

Année académique 2016 – 2017

Remerciements

Nous tenons à remercier nos tuteurs école qui ont su nous fournir une aide pertinente tout au long du projet.

Nous remercions également monsieur Lucas Prieux, qui nous a accordé sa confiance et a été disponible pour répondre à nos questions et faciliter notre appréhension du projet.

Enfin, nous remercions chaleureusement monsieur Thomas Roj qui a assuré la continuité du projet, et a su nous transmettre avec passion et patience le travail déjà effectué.

Table des matières

Introduction

Vous rencontrerez en lisant ce rapport de nombreux acronymes, veuillez vous référer au glossaire en page 18 pour en avoir une brève définition.

Dans le cadre de notre projet de 4 ème année du cycle ingénieur IMA à Polytech Lille, nous avons eu l'opportunité de mettre la technologie au service de l'art, et d'allier la musique à de vieux systèmes électroniques.

Nous avons travaillé avec de vieux appareils électronique tels que des disques durs, des lecteurs de disquette, des modems et une imprimante matricielle que nous avons détournés de leur utilisation première : l'orchestre électronique est né.

Ce projet a été commandé par monsieur Lucas Prieux, directeur artistique du spectacle #Humains proposant une vision du monde de demain en plongeant le spectateur au cœur de la technologie.

Dans un premier temps, nous présenterons le contexte dans lequel s'inscrit ce projet, à savoir, le spectacle #Humains et le travail ayant déjà été accompli par Thomas Roj et Joshua Letellier. Ensuite, nous détaillerons le travail que nous avons mené. Enfin, nous expliquerons les problèmes rencontrés et les solutions envisagées.

1. Contexte

« Il y a quelque temps, un réseau social d'un genre nouveau a fait son apparition. Un réseau fonctionnant en réalité virtuelle et nécessitant un casque pour s'y connecter. Son succès a été fulgurant. Il bouleversait les interactions entre les gens et révolutionnait le Web en le spatialisant.
Mais un beau jour, les gens ont commencé à disparaître… »

Le spectacle réalisé par Lucas Prieux, est un subtil mélange de théâtre, de marionnettes, mais aussi de vidéos et d'hologrammes permettant au spectateur de s'immerger dans le monde imaginé par l'artiste, et de se questionner sur la place qu'occupe la technologie dans nos vies.

Monsieur Prieux nous a donc chargés de réaliser une partie sonore de son spectacle. En effet, nous allons réalisé un orchestre électronique composé de divers appareils obsolètes. Un compositeur sera chargé de contrôler l'orchestre à l'aide d'un clavier MIDI afin de créer l'ambiance sonore du spectacle.

Extraits du spectacle - dossier de création #Humains

Nous avons alors poursuivi le travail entrepris par Thomas Roj et Joshua Letellier. Leur réalisation finale permet de jouer en live sur des lecteurs disquette et des disques durs à l'aide d'un clavier d'ordinateur. Une interface WEB permet de gérer les instruments et offre la possibilité de lancer une musique enregistrée « à la main » *i.e.* codée par ces derniers sous forme de deux tableaux, un contenant les notes et l'autre le temps entre deux notes consécutives.

Montage final comprenant une RPI2 contrôlant un modem, les HDD, les FDD et l'imprimante matricielle

Notre travail consiste alors à améliorer ce dispositif. Nous allons donc détailler les tâches que nous avons du réaliser.

2. Cahier des charges

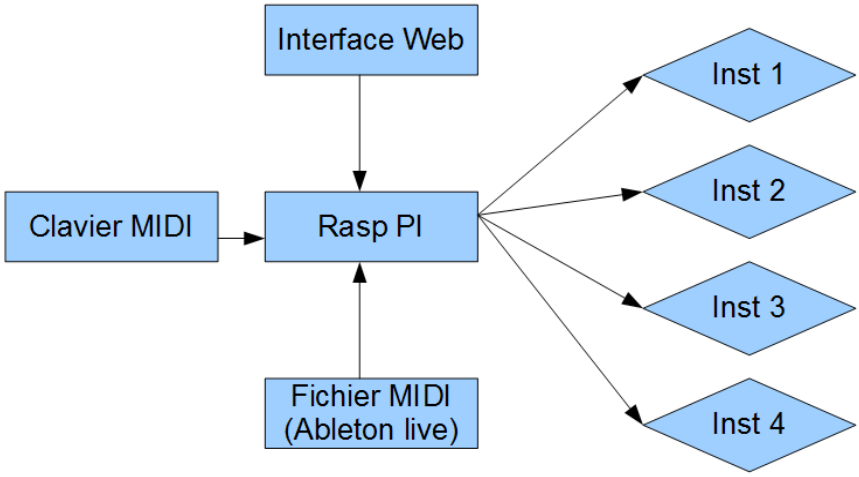
Le travail que nous devions effectuer se découpe selon deux grands axes :

- D'une part, l'analyse de fichiers et flux MIDI *i.e.* fournir un module permettant de traduire les informations contenues dans un fichier MIDI en une commande adaptée à chaque instrument, mais aussi d'interpréter « à la volée » les informations en provenance du clavier maître.
- Explorer plus en détail la commande de l'imprimante matricielle afin d'offrir une large gamme de sons et sonorités (sons percussifs du chariot,…). Trouver un moyen de boucler la communication des modems, c'est-à-dire simuler un serveur vocal pour pouvoir les contrôler avec des commandes Hayes.

Le dispositif doit être modulaire en offrant la possibilité au compositeur de sélectionner, grâce à l'interface WEB, sur quels instruments il veut jouer. Il aura aussi la possibilité d'envoyer, au dispositif, un fichier MIDI, écrit à l'aide d'un logiciel tiers tel que Ableton, qui sera traduit et joué.

Nous poursuivrons avec les choix techniques faits par nos prédécesseurs, soit :

- Une Raspberry Pi 2 : faisant office de serveur WEB et sur laquelle les processus de traitement MIDI et de gestion des instruments sont implantés. Ses GPIOs nous permettent de gérer les HDD et les FD. Ses ports USB, l'imprimante matricielle, les modems et le clavier MIDI.
- Disques durs, lecteurs disquette, imprimante matricielle, modem 56k, clavier MIDI et alimentation.



Synoptique du dispositif employé

3. Imprimante matricielle

L'imprimante utilisée est une Epson LQ-570+. Sa tête d'impression est composée d'un moteur sur lequel sont montées 24 aiguilles. En fonction du caractère à imprimer, l'imprimante actionne le moteur et un certain nombre d'aiguilles. Son connecteur DB – 25 offre une liaison série par laquelle nous pouvons envoyer des commandes ESC / P2, langage propriétaire décrivant les pages utilisé par les imprimantes Epson.

Nous avons expérimenté les trois modes d'impression offerts sur ce modèle d'imprimante, en faisant varier le type de caractère, le nombre, la taille, la résolution et les attributs de caractère (gras, italique, souligné). Ces modes sont :

- Le multipoint, mode « classique » permettant d'écrire du texte et de jouer sur la police de caractère.
- Le mode Compressed Raster Graphics RLE (Run Length Encoding), offrant un format compressé des caractères à imprimer. Il permet d'augmenter la vitesse d'impression mais dégrade la qualité.
- Le mode bit-image, sans doute le mode le plus intéressant dans notre cas puisqu'il nous permet de contrôler (presque) chaque aiguille de la tête d'impression. Après avoir choisi une densité de point par colonnes, les bits de la donnée envoyée qui sont à « 1 » activeront une aiguille. Ci-dessus la densité est de 8 bits par colonnes et la donnée envoyée vaut 01011100 :

Comme aucun de ces modes n'offre la possibilité de contrôler la vitesse d'impression, la palette de sons obtenus reste très restreinte. Elle peut se résumer au bruit :

- du chariot se déplaçant et arrivant en butée
- du moteur faisant avancer la page
- de l'écriture d'une série de caractères utilisant plus ou moins d'aiguilles

Nous détaillerons dans un paragraphe ci-après les solutions qui sont envisageables pour obtenir une plus large gamme de sonorités.

4. Analyse MIDI

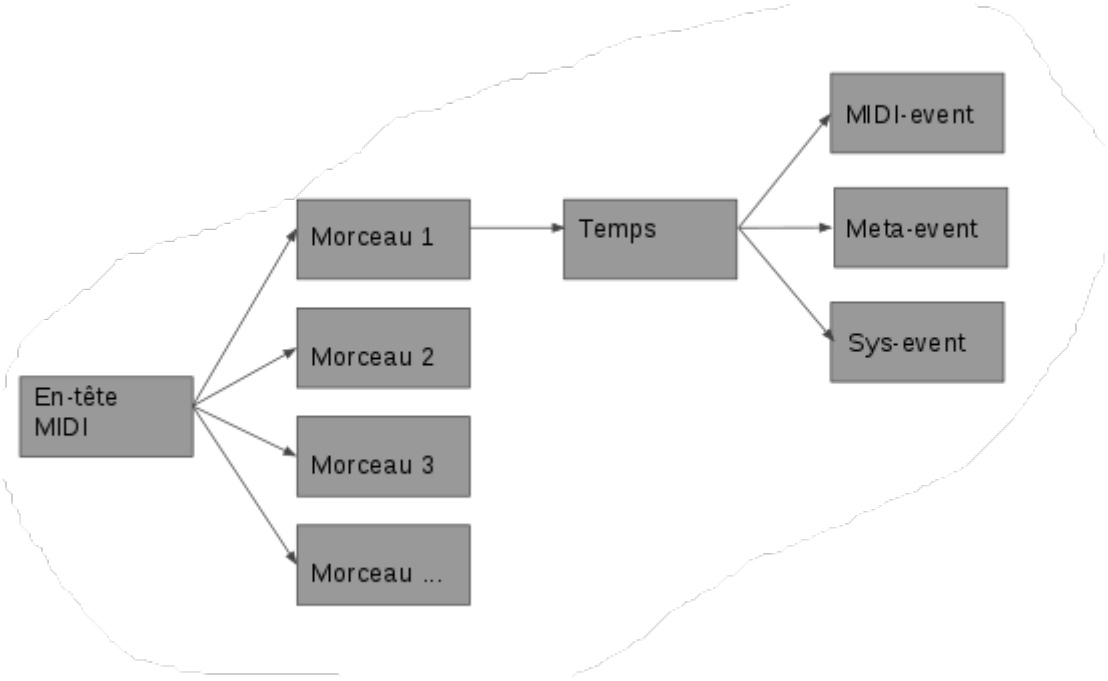
La norme MIDI (Musical Instrument Digital Interface) est utilisée dans le monde de la musique. A la fois protocole et format de fichier, il permet le transport des informations musicales de façon universelle entre instrument électronique, logiciel, contrôleur et séquenceur tels qu'une boîte à effet ou console de mixage. Il délivre un message de contrôle et non sonore.

La réalisation matérielle se fait grâce à une fiche DIN à 5 branches. Le code envoyé n'endommage en aucun cas les appareils reliés. En théorie, le débit est de 31,25 kilo-octets par seconde.

Câble midi

Exemple branchement MIDI sur clavier

Concernant le fichier MIDI, il est standardisé, ainsi il peut être transcrit par n'importe quel logiciel / matériel utilisant le protocole MIDI.



Synoptique de la formation d'un fichier MIDI

L’en-tête (commençant toujours par 0x4d54) nous permet de connaître :

- sa longueur (codé sur 4 octets)
- son format (codé sur 1 octet), c'est-à-dire s'il y a une seule piste ou plusieurs pistes ou s'il y a plusieurs chansons
- le nombre de morceaux (codé sur 2 octets)
- le nombre de ticks/beat (codé sur 2 octets), il permet de régir le temps.

L’en-tête du morceau (commençant par 0x4d54726b) nous renseigne sur sa longueur.

Le timing est divisé en 2 parties selon sa taille. En effet, si le premier octet est supérieur à 128, il faut prendre en compte l'octet suivant. Pour calculer le « timing », il faut alors enlever le bit de poids fort du premier octet, de décaler cet octet de 8 et de l’additionner à l'octet suivant.

La norme MIDI utilise ces contractions afin de réduire la taille de ses fichiers.

Après avoir récupéré le timing, le fichier MIDI comporte 3 sortes d'événements :

- les métas événements commençant par 0xff

- les événements MIDI commençant par 0xXY avec X allant de 8 à e (représentant le type d'événement) et Y allant de 0 à f (représentant le numéro du canal utilisé)

- les événements systèmes commençant par 0xfX (X allant de 0 à e)

Dans le cadre du projet, les événements systèmes et métas ne sont pas nécessaires. Il suffit de connaître le moment où il s'arrête. Dans le cas des événements systèmes, il se finit par 0xf7. Dans l'autre cas, la taille de l'événement est donné sur son 3 octets.

L’événement MIDI permet de connaître :

- la nature de l’événement (Note on, off etc)
- son canal
- 1 ou 2 paramètres selon sa nature

- sa note
- son intensité

Commande	Sens	Nbre paramètres	Paramètre 1	Paramètre 2
0x8X	Note-off	2	Note	Velocity
0x9X	Note-on	2	Note	Velocity
0xAX	Aftertouch	2	Note	Effect
0xBX	Continuous controller	2	Controller number	New value
0xCX	Patch change	1	Instrument number	
0xDX	Channel Pressure	1	Single greatest pressure value	
0xEX	Pitch bend	2	lsb(7bits)	msb(7bits)

Evénements MIDI

Le travail effectué était donc de lire ces fichiers MIDI en hexadécimal et de récupérer les informations nécessaires au déclenchement/arrêt des instruments en respectant son format. C'est-à-dire récupérer les événements de note et leur timing.

5. Clavier MIDI

Clavier MIDI utilisé pour le projet

Tout d'abord, nous utilisons la commande « aconnect -i » afin de rechercher le numéro du port du clavier MIDI sur la RPI.

Ensuite, afin de pouvoir recevoir les informations du clavier MIDI, nous avons utilisé la bibliothèque libasound (ALSA sequencer port) et le code source de *aseqdump* afin de récupérer les informations utiles, qui sont en l'occurrence les événements Note-on et Note-off ainsi que leur hauteur et vélocité.

Ainsi le code se découpe en plusieurs parties :

- initialisation du sequencer (ouverture et naming du sequencer)

  snd_seq_open et *snd_seq_set_client_name*

- création du port

  snd_seq_create_simple_port

- connexion du port

  snd_seq_connect_from

- récupération des événements dans une structure « snd_seq_event »

  snd_seq_event_input

- boucle attendant l'apparition d'un événement

  snd_seq_poll_descriptors (permet de savoir si un événement a été effectué)

- fermeture du port

  snd_seq_close

Avant de détailler comment nous avons implanter sur la RPI le code de traitement des événements MIDI, il semble important de rappeler brièvement le principe de fonctionnement des lecteurs de disquette et des disques durs. Leur commande est la même puisqu'il s'agit simplement de commander le moteur de leur tête de lecture avec une modulation à largeur d'impulsion (PWM). La fréquence choisie correspondra à la note jouée sur l'appareil.

6. Raspberry Pi

Sur la RPI sont implantés un serveur WEB lighttpd, plus léger qu'un Apache, et une Common Gateway Interface (CGI) permettant de lancer les scripts python via une page WEB. Cette interface propose de jouer un morceau enregistré « à la main » i.e. que les notes et le temps sont codés en dur dans le fichier à jouer. Nous pouvons aussi choisir de jouer en live sur les lecteurs disquette et disques durs avec le clavier MIDI.

Pour faire le lien entre la bibliothèque de traitement des événements MIDI qui permet d'acquérir les informations en provenance du clavier, et le code Python permettant de contrôler les PWM, donc les appareils, nous utilisons le module ctypes de Python. Il permet d'utiliser les fonctions définies dans une bibliothèque dynamique C dans du code Python. L'avantage du Python est que c'est le langage « par défaut » de la RPI et qu'il offre un module de contrôle des GPIO. Le langage C nous permet de récupérer et traiter les événements MIDI facilement puisque nous avons à notre disposition le code source de l'utilitaire aseqdump.

Détaillons le déroulement général du code Python permettant de jouer en live :

- On associe à chaque instrument un thread gérant la fréquence de la PWM associée au GPIO concerné
- Ensuite le programme principal attend un événement MIDI puis envoie un signal aux threads afin de leur préciser que la variable globale contenant les informations MIDI (note On-Off, hauteur de note,vélocité,…) a été mise à jour
- Les threads actualisent la fréquence de leur PWM en conséquence
- Pour terminer le programme on envoie un signal aux threads afin de les arrêter proprement, puis on ferme le port associé au clavier MIDI et on ré-initialise les GPIO

7. Problèmes rencontrés

Le chemin emprunté lors de ce projet a été pour le moins semé d’embûches. Dans un premier temps, il faut considérer que la première tentative n'est pas forcément la bonne, il faut donc sans cesse rechercher des informations complémentaires ou d'autres moyens d'aboutir. Ce qui nous amène aux problèmes de gestion de projet, nous n'aurions pas du passer autant de temps sur l'imprimante matricielle. Il faut être plus réactif : des recherches en amont aurait permis de nous rendre compte de l'impossibilité d'agir sur la vitesse du chariot par exemple.

De plus, la reprise de la Raspberry PI fut laborieuse car la carte SD était corrompue, nous avons du procéder à une réinstallation totale. A l’avenir il faudra avoir le réflexe de vérifier immédiatement le matériel utilisé. Nous avons aussi remarqué que les scripts python ne se lançaient pas selon le réseau depuis lequel nous les lancions. Enfin, certains appareils étaient défectueux : certains lecteurs disquette ne marchent pas et l'alimentation est aussi tombée en panne en cours de projet. Cependant, nous avons mis du temps à nous rendre compte de ces problèmes, cherchant une origine logicielle.

8. Tâches à poursuivre et solutions envisagées

Il reste donc des tâches à effectuer :

- S'occuper de l'imprimante matricielle afin d'offrir une plus large gamme sonore, c'est-à-dire soit reprogrammer le firmware, soit contrôler directement les moteurs qui gérent les aiguilles et le chariot de celle-ci. Ce qui impliquerait probablement la réalisation d'une carte électronique de puissance.
- Se pencher sur la communication des modems afin que ceux-ci réagissent aux commandes Hayes et produisent les sons souhaités. Pour cela, il faudrait trouver le moyen de simuler un serveur vocal.
- Implanter la possibilité de charger un fichier MIDI sur l'interface WEB (fichier MIDI créé par le compositeur)
- Finir le code de traduction du fichier MIDI : remplir les tableaux (notes / timing) afin de les envoyer dans le script python pour lancer les commandes sur les instruments souhaités.

Conclusion

Le route pour arriver au bout de notre projet aura été longue et imprévisible. Mais nous en tirons de nombreux enseignements. La première solution envisagée n'est souvent pas la bonne ou la meilleure. Il nous semble important d'accorder du temps à une recherche approfondie, voire un état de l'art, avant de se lancer dans le développement d'une idée. Cela évite de perdre du temps *a posteriori*, en empêchant de revenir au point de départ.

Ce projet nous aura permis de découvrir des appareils électroniques qui ne sont – presque – plus utilisés tels que l'imprimante matricielle ou les lecteurs de disquette. Nous avons pu mettre en application, et avoir des exemples concrets, de ce que nous avons appris durant notre formation : la mise en place de processus légers, signaux, PWM, etc. La dimension artistique de ce projet fut réellement agréable pour nous, et nous a montré que arts et sciences ne sont pas aussi éloignés que l'on pourrait le croire.

Nous espérons que la relève sera assurée pour mener le projet à son terme et entendre le doux bruit de l'imprimante résonner dans des salles de spectacle.

Glossaire

MIDI : Le Musical Instrument Digital Interface est un protocole de communication et un format de fichier dédiés à la musique, et utilisés pour la communication entre instruments électroniques, contrôleurs, séquenceurs, et logiciels de musique.

RPI : Raspberry PI, nano-ordinateur

GPIO : Les General Purpose Input/Output sont les ports d'entrées/sorties d'usage général utilisés sur la Raspberry PI.

HDD : Hard Drive Disk ou disque dur pour les amis de la langue de Molière.

FDD : Floppy (Drive) Disk ou lecteur de disquette.

Bibliographie

MIDI :

https://www.midi.org/specifications/item/table-1-summary-of-midi-message
http://www.ccarh.org/courses/253/handout/smf/
https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html
http://ehess.modelisationsavoirs.fr/marc/ens/projets/cours/midifile/midifile.html
https://www.pianoweb.fr/comprendre-le-midi.php

Imprimante matricielle :

http://www.qotile.net/dotmatrix.html
https://files.support.epson.com/pdf/general/escp2ref.pdf (Manuel de référence ESC / P2)

Librairie Alsasound :

http://www.alsa-project.org/alsa-doc/alsa-lib/index.html

Python :

https://docs.python.org/3/library/ctypes.html (module ctypes)
https://docs.python.org/3/library/signal.html (signaux)
https://docs.python.org/2/library/threading.html (threads)

RPI :

https://pinout.xyz/# (diagramme interactif des gpios de la rpi)

Lecteur de disquette :

http://hxc2001.com/download/datasheet/floppy/thirdparty/Teac/TEAC-FD235HF-A291.PDF