
Ecole Nationale
Supérieure
de l'Electronique
et de ses Applications

ENSEA

Rapport de Stage de Deuxième année

# Skydrum

***Elève :*** BRAZ Luiz Gustavo
***Conseiller : MOUNET Jean-François***

Cergy - juillet 2020

# Table des matières

## Abstract

The main objective of this work is to update the Skydrum project with the latest hardware and software, reducing the size of the instrument, improving portability and usability. We also want to add new functionalities to facilitate the integration of the instrument with the protocols and instruments already established on the market.


# 1 Objectif

Le but du stage est de mettre à jour le projet Skydrum avec le matériel et les logiciels les plus récents, en réduisant la taille de l'instrument, en améliorant la portabilité et la convivialité. Nous souhaitons également ajouter de nouvelles fonctionnalités pour faciliter l'intégration de l'instrument dans un contexte plus virtuel utilisant des protocoles et instruments notamment virtuels déjà établis sur le marché.

# 2 Introduction

## 2.1 Historique

Le concept Skydrum a été inventé par le musicien et batteur Jean-François Mounet dans les années 90 en France. L'objectif était de créer une batterie en utilisant des faisceaux lumineux pour déclencher des sons.

Ce concept unique au monde utilise des colonnes géantes qui s'illuminent a leur tour en traduisant en lumière les sons déclenchés.

La version originale du concept Skydrum utilise des colonnes lumineuses comme on peut le voir ci-dessous :

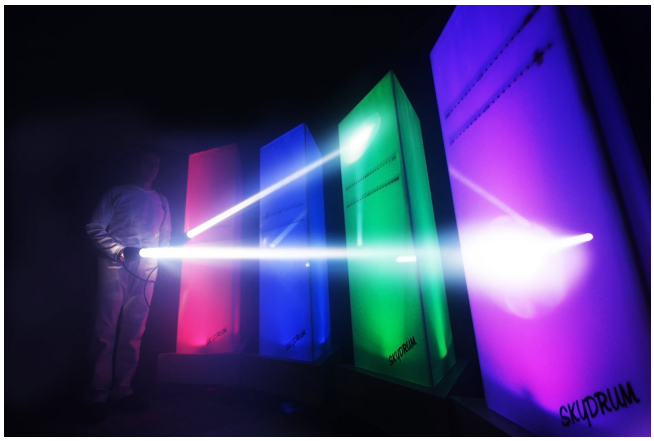


FIGURE 1 – Skydrum original

Le Skydrum a été invité sur France 2 à la fin des années 90 sur l'émission "La France m'étonne" réalisée en collaboration avec Valérie Anne Giscard d'Estaing, auteur du livre mondial des inventions dans lequel figure Skydrum dans deux éditions. Skydrum dans cette émission est utilisé en mode déclenchement de mélodie.(`https://www.youtube.com/watch?time_continue=42&v=KtIppJV7E-k&feature=emb_logo`)

FIGURE 2 – Skydrum en France 2 1992

## 2.2 Protocole MIDI

Le langage MIDI est utilisé pour transmettre des informations en temps réel pour déclencher des notes de musique.

≪Temps réel≫ signifie que chaque message est envoyé exactement au moment où il doit être interprété par le synthétiseur cible (qui peut être un synthétiseur matériel ou un synthétiseur logiciel).

Différents messages sont définis pour transmettre les informations nécessaires au déclenchement de la musique.

Le point important est que le langage MIDI ne définit pas le son lui-même, mais uniquement la séquence d'instructions pour créer le son dans le synthétiseur cible.

Les messages MIDI sont envoyés sous la forme d'une séquence temporelle d'un ou plusieurs octets (8 bits), normalement un message de trois octets. Le premier octet est un octet **STATUS**, souvent suivi d'octets **DATA** avec des paramètres supplémentaires. Un octet **STATUS** a le bit 7 mis à 1 et un octet DATA a le bit 7 mis à 0 ce qui permet de les différencier.

L'octet **STATUS** détermine le type du message. Le nombre d'octets **DATA** qui suivent dépend du type de message.

Pour les messages **NOTE ON** et **NOTE OFF**, par exemple, nous avons 3 octets comme indiqué ci-dessous.

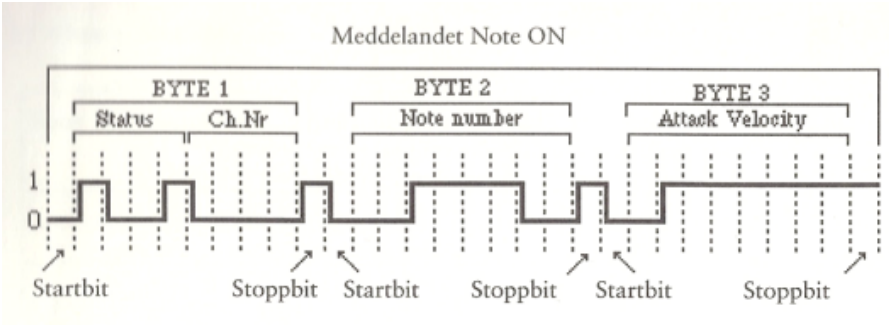


FIGURE 3 – NOTE ON

Une partie extrêmement importante pour le fonctionnement de la communication MIDI est la vitesse de transmission, Baudrate. Le protocole MIDI a un baudrate de 31250 bauds, c'est-à-dire que chaque bit a une période de 32us, donc un message avec trois octets plus trois start bits et trois stop bits a 3x320us de temps, donc 960us.

Le signal lors d'une communication MIDI, lorsqu'il est au repos, reste à 5v (HIGH) et passe à 0v lors du démarrage d'un message.

### 2.2.1 NOTE ON et NOTE OFF

Les principaux messages sont les messages NOTE ON et NOTE OFF.

Le message NOTE ON est envoyé lorsque l'interprète frappe une touche du clavier musical. Il contient des paramètres pour spécifier la hauteur de la note ainsi que la vélocité (intensité de la note lorsqu'elle est frappée).

Lorsqu'un synthétiseur reçoit ce message, il commence à jouer cette note avec la hauteur et le niveau de force corrects.

Lorsque le message NOTE OFF est reçu, la note correspondante est désactivée par le synthétiseur.

NB : Chaque message NOTE ON nécessite son message NOTE OFF correspondant, sinon la note sera jouée indéfiniment. La seule exception concerne les instruments à percussion, où il peut arriver que seule la NOTE ON soit envoyée, car la note de percussion s'arrête automatiquement. Mais il est préférable d'envoyer la NOTE OFF dans tous les cas, car elle pourrait être faussement interprétée par le synthétiseur qui la reçoit ou bloquer celui-ci.

Le message **NOTE ON** est structuré comme suit :
— Status byte : 1001 CCCC
— Data byte 1 : 0PPP PPPP
— Data byte 2 : 0VVV VVVV

≪CCCC≫ est le numéro de canal MIDI (de 0 à 15)
≪PPP PPPP≫ est la valeur du pas (de 0 à 127)
≪VVV VVVV≫ est la valeur de la vitesse (de 0 à 127)

La valeur de hauteur détermine la fréquence de la note à jouer. Elle va de 0 à 127, la note C du milieu étant représentée par la valeur 60 :

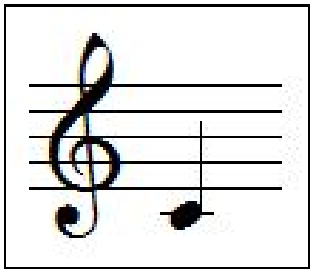

FIGURE 4 – Note C

La valeur de hauteur de note est représentée par demi-tons, de sorte que C sera 61, D sera 62, ...

Pour transposer une note une octave plus haut, on ajoute 12 à sa valeur de hauteur. En utilisant le MIDI, la transposition est très simple car elle se fait simplement en ajoutant ou en soustrayant une valeur fixe.

| Note | Octave | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| C | 0 | 12 | 24 | 36 | 48 | 60 | 72 | 84 | 96 | 108 | 120 |
| C# | 1 | 13 | 25 | 37 | 49 | 61 | 73 | 85 | 97 | 109 | 121 |
| D | 2 | 14 | 26 | 38 | 50 | 62 | 74 | 86 | 98 | 110 | 122 |
| D# | 3 | 15 | 27 | 39 | 51 | 63 | 75 | 87 | 99 | 111 | 123 |
| E | 4 | 16 | 28 | 40 | 52 | 64 | 76 | 88 | 100 | 112 | 124 |
| F | 5 | 17 | 29 | 41 | 53 | 65 | 77 | 89 | 101 | 113 | 125 |
| F# | 6 | 18 | 30 | 42 | 54 | 66 | 78 | 90 | 102 | 114 | 126 |
| G | 7 | 19 | 31 | 43 | 55 | 67 | 79 | 91 | 103 | 115 | 127 |
| G# | 8 | 20 | 32 | 44 | 56 | 68 | 80 | 92 | 104 | 116 | --- |
| A | 9 | 21 | 33 | 45 | 57 | 69 | 81 | 93 | 105 | 117 | --- |
| A# | 10 | 22 | 34 | 46 | 58 | 70 | 82 | 94 | 106 | 118 | --- |
| B | 11 | 23 | 35 | 47 | 59 | 71 | 83 | 95 | 107 | 119 | --- |

FIGURE 5 – Valeurs notes MIDI

La valeur de vélocité va normalement de 1 à 127, couvrant la plage allant d'une note pratiquement inaudible jusqu'au niveau de note maximum. Il correspond essentiellement à l'échelle des nuances trouvées dans la notation musicale, comme suit (c'est plus indicatif que les valeurs exactes) :

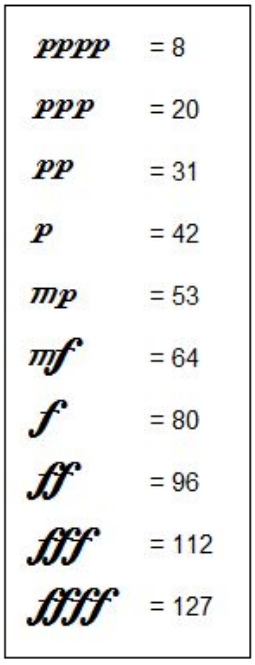


FIGURE 6 – Vitesse des Notes

### 2.2.2 Control Change

Pour modifier les paramètres et les réglages à l'aide de la communication MIDI, nous utilisons la commande ≪Control Change≫ (CC).

Dans le CC, il y a à nouveau trois octets, le premier octet (0x0b + numéro de canal), le second correspond au numéro de paramètre et le troisième correspond à la valeur du paramètre (0 à 127).

| STATUS | DATA 1 | DATA 2 |
|---|---|---|
| 1011cccc (BcH) | 0xxxxxxx | 0vvvvvvv |

cccc: channel Number
xxxxxxx: controller number
vvvvvvv: controller value

FIGURE 7 – Messages MIDI control change

NB. Les valeurs de 0x14 à 0x1F pour le deuxième octet (numéro de paramètre) sont indéfinies et utilisables par l'utilisateur et donc nous les utiliserons dans notre projet.

# 3 Développement

## 3.1 Projet Général

Le "Skydrum" se composera de 5 boîtiers interconnectés contenant chacun des rampes de capteurs de luminosité qui détectent un faisceau de lumière et émettent ainsi une note via un message MIDI.

Le signal MIDI est ensuite interprété par un logiciel qui émule un synthétiseur puis le signal audio du synthétiseur est amplifié par une sonorisation.

Les boîtiers étant chaînés l'un après l'autre par le bus MIDI, les signaux MIDI traversent donc tous les boîtiers (Figure 11).

Chaque boîtier reporte sur le suivant (par son MIDI out) les messages MIDI qu'il reçoit et ajoute un message NOTE ON à chaque passage du faisceau dans les rampes associées au boîtier. Donc chaque boîtier possède une entrée et une sortie MIDI.

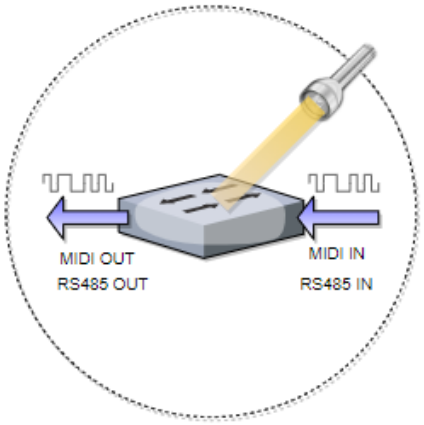


FIGURE 8 – Schématique détail boite

Pour effectuer le contrôle, nous utilisons deux sorties numériques (I01 et I02) du PIC qui entrent dans le multiplexeur CD45042 (A et B), donc pour BA :

— BA = 00 : Config1 ;
— BA = 01 : Config2 ;
— BA = 10 : Config3 ;
— BA = 11 : Config4.

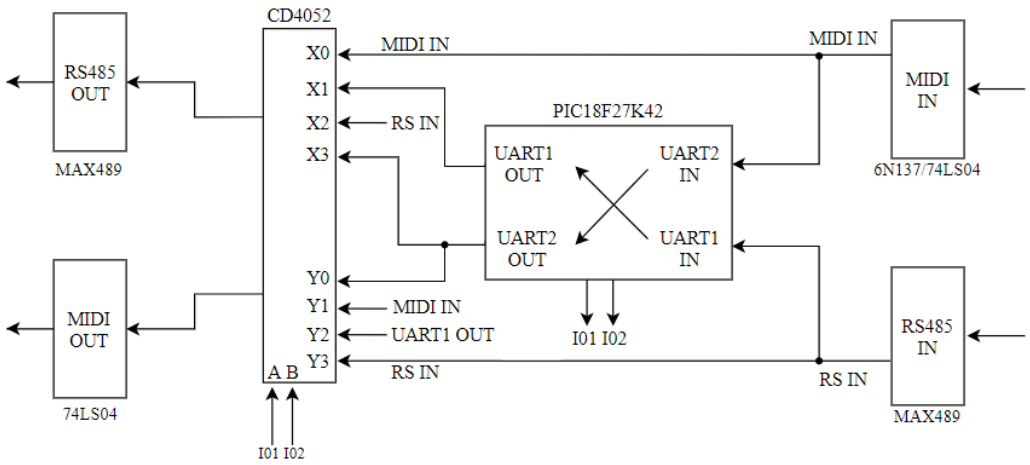


FIGURE 9 – Schématique de configuration

Chaque configuration aura les paramètres suivants :

— Config1 : MIDI ⟶ RS Thru ; RS ⟶ MIDI Thru ;
— Config2 : RS IN ⟶ UART1 ⟶ MIDI + RS ;
— Config3 : MIDI Thru ; RS Thru ;
— Config4 : MIDI ⟶ UART2 ⟶ MIDI + RS.

Architecture du système utilisant un bus MIDI :

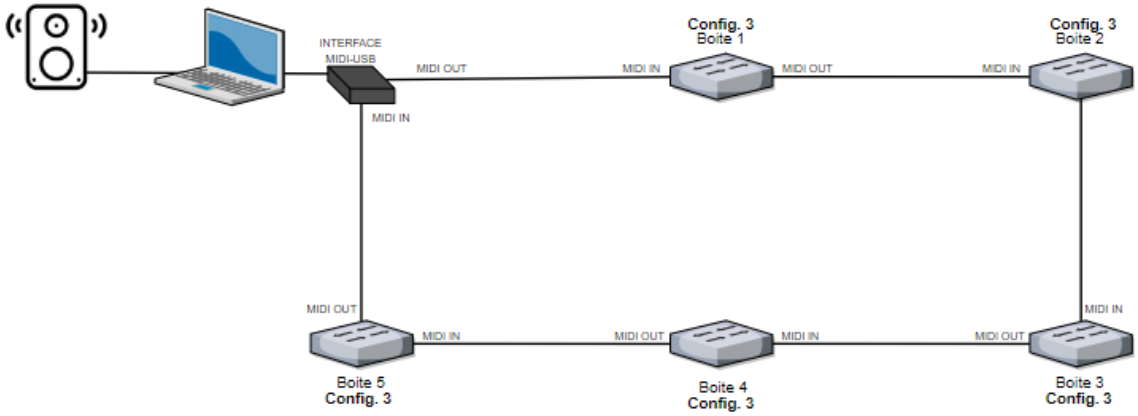


FIGURE 10 – Schématique générale avec connecteur MIDI

Architecture du système utilisant un bus RS485 :

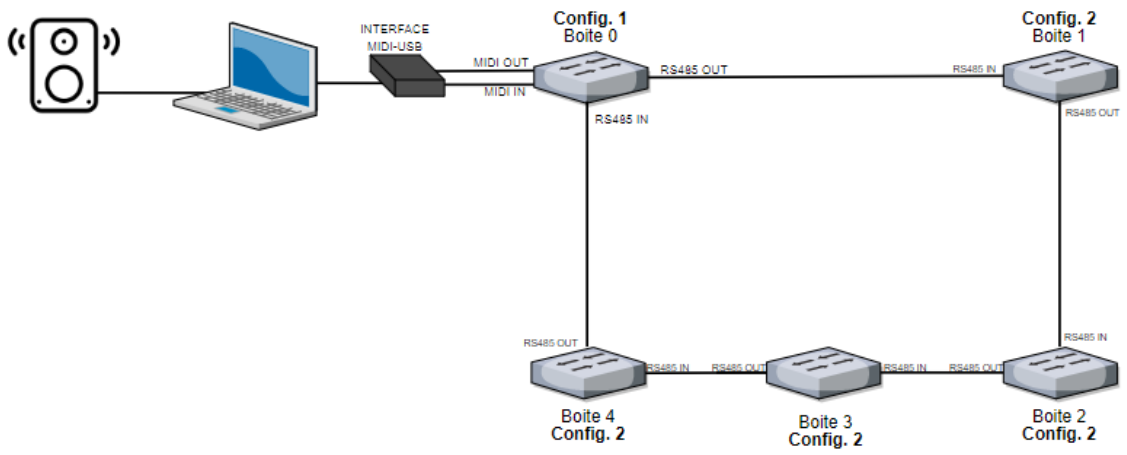


FIGURE 11 – Schématique générale avec connecteur RS-485

Afin de communiquer avec le logiciel de l'ordinateur, qui dispose, par exemple, d'un synthétiseur, nous utilisons une interface USB-MIDI (boîtier noir sur le schéma ci-dessus) qui convertit les signaux MIDI IN et MIDI OUT pour le port USB.

La maquette de test utilise seulement une chaîne MIDI sans la RS-485.

### 3.1.1 Fonctionnement du boîtier

Chaque boîtier a deux rampes de capteurs de lumière, comme indiqué dans l'image. Ainsi, il est possible de déterminer s'il y a eu un balayage de faisceau (frappe).


FIGURE 12 – Repos

Il existe deux types de frappe possible, de bas en haut ("bas-haut") et de haut en bas ("haut-bas"). Notre système est capable de détecter les deux suivant la valeur de la variable "typefrappe" (typefrappe = 0 pour une simple frappe de haut en bas, typefrappe = 1 pour une frappe dans les deux sens).

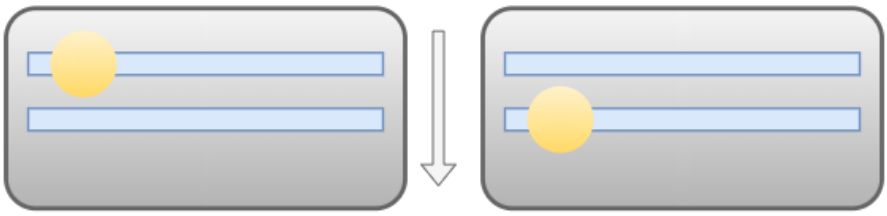

FIGURE 13 – Frappe haut-bas

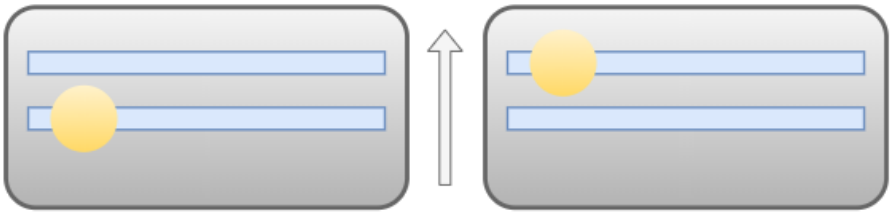

FIGURE 14 – Frappe bas-haut

## 3.2 Hardware

### 3.2.1 Composants

#### PIC microcontroller - PIC18F27K42

Les microcontrôleurs PIC (*Peripheral Interface Controller*) forment une famille de microcontrôleurs de la société Microchip. Ils ont une unité de traitement et d'exécution de l'information à laquelle on a ajouté des périphériques internes permettant de réaliser des montages sans nécessiter l'ajout de composants annexes. Un microcontrôleur PIC peut donc fonctionner de façon autonome après programmation.

Les PIC intègrent une mémoire programme non volatile (FLASH), une mémoire de données volatile, une mémoire de donnée non volatile (E2PROM), des ports d'entrée-sortie (numériques, analogiques, MLI, UART, bus I2C, Timers, SPI, etc.), et même une horloge, bien que des bases de temps externes puissent être employées. Certains modèles disposent de ports et unités de traitement de l'USB et Ethernet.

Ils sont populaires en raison : de leur faible coût, de la disponibilité d'outils de développement peu coûteux, ainsi que de leurs capacités de programmation en série et de reprogrammation de la mémoire FLASH. Ils comportent aussi une vitesse de traitement élevée en raison de l'architecture Harvard.

Après vérification, nous avons choisi le PIC18F27K42 pour ce projet. Il va traiter, recevoir et renvoyer l'information.

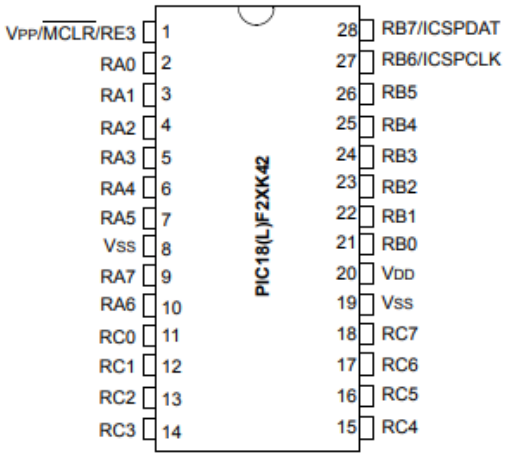


Figure 15 – PIC18F2X42

Caractéristiques principales :
— Up to 64 MHz clock input (on utilise 8MHz)
— Up to 1 KB Data EEPROM
— 2.3V to 5.5V (PIC18F26/27/45/46/47/55/56/57K42)
— Four 16-Bit Timers (TMR0/1/3/5)
— Two UART Modules
— Analog-to-Digital Converter with Computation

**Connecteurs MIDI (midi-din) et RS485 DB9**

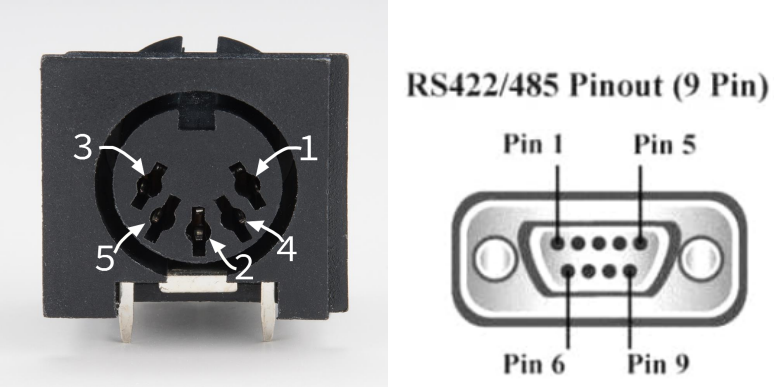


FIGURE 16 – Connectors MIDI (midi-din) et RS485 DB9

**74LS04 - Hex Inverter Gates Logic IC**

Nous utilisons un 74LS04 pour propager le signal MIDI et éviter la perte de puissance du signal et les distorsions.

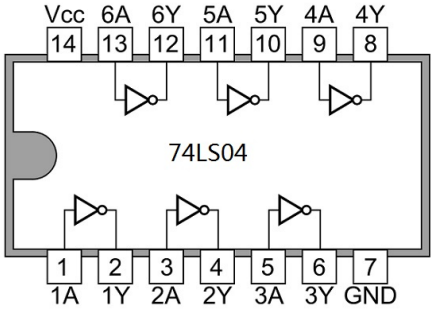


FIGURE 17 – Diagram 74LS04

Caractéristiques principales :
— Number of inverters, 6 inverters
— Logic Type, Bipolar
— Supply Voltage – Max, 5.25 V
— Supply Voltage – Min, 4.75 V
— High Level Output Current, – 0.4 mA
— Low Level Output Current, 16 mA
— Operating Supply Voltage, 5 V

### 6N137 - High Speed Optocoupler

Nous utilisons un 6N137 pour recevoir le signal MIDI, plus de détails sur le circuit utilisant ce composant dans la section «Schématique» ensuite.

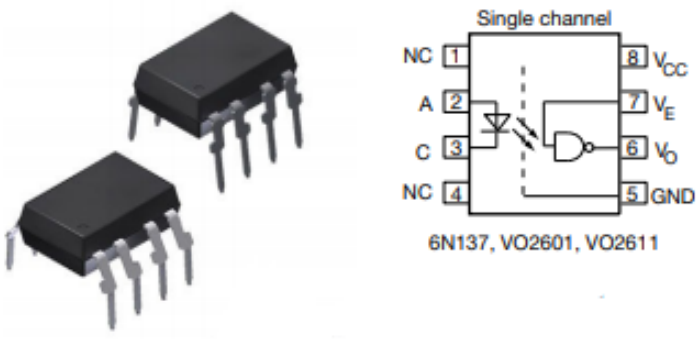


FIGURE 18 – Diagram 6N137

Caractéristiques principales :
— High speed : 10 MBd typical
— +5 V CMOS compatibility
— Low input current capability of 5 mA
— Choice of CMR performance of 15 kV/us, 5 kV/us, and 1000 V/us

### MAX489 - RS-485/RS-422 Transceiver

Nous utilisons le MAX489 pour recevoir et envoyer des signaux via les connecteurs RS-485, ils ont des émetteurs et des récepteurs intégrés.

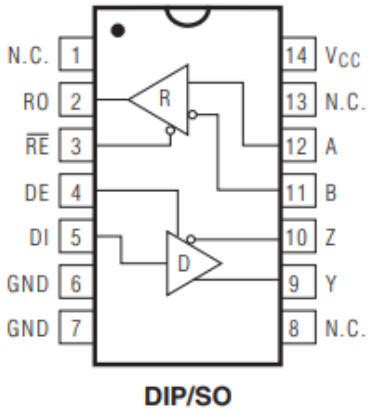


FIGURE 19 – MAX489 - RS-485/RS-422 Transceiver

Caractéristiques principales :
— Low Power Consumption Minimizes Thermal Dissipation, Reducing System Cost
— Slew-Rate-Limited Drivers up to 250kbps Data Rate
— Integrated Protection Enhances System Robustness

### CD4052B - Analog Multiplexer

On utilise le CD4052B pour basculer entre les sorties et entrées MIDI et RS-485, c'est un multiplexeur avec 8 entrées et 2 sorties, on le contrôle à l'aide de deux entrées numériques simples.

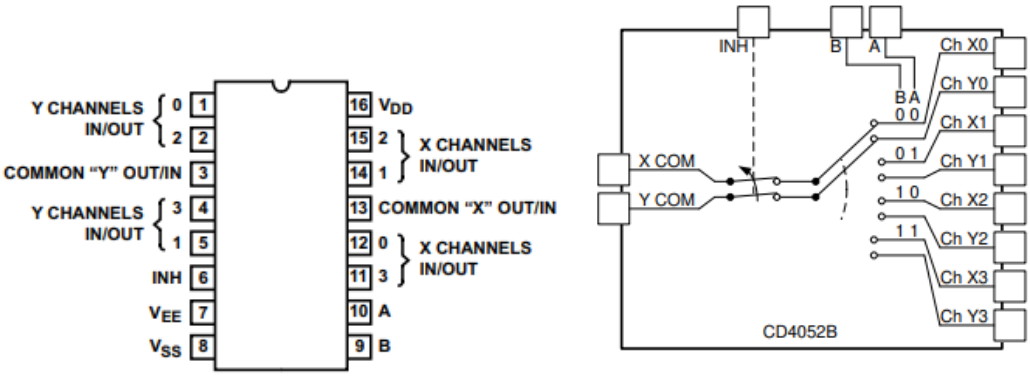


Figure 20 – CD4052B - Analog Multiplexer

Caractéristiques principales :
— Wide Range of Digital and Analog Signal Levels (3v to 20v)
— Low ON Resistance, 125 ohms
— High OFF Resistance
— Maximum Input Current of 1 uA at 18 V

### TSR 1-2450 - Convertisseur DC/DC

Nous utilisons le TSR 1-2450 pour convertir une tension d'entrée de 12V à 5v, tension de fonctionnement de tous les composants et capteurs de l'équipement.


Figure 21 – TSR 1-2450 - Convertisseur DC/DC

Caractéristiques principales :
— max input voltage of 36v and minimum of 6,5v
— output voltage of 5v
— max current of 1A

### 3.2.2 Schématique

#### Communication MIDI

La communication MIDI nécessite un circuit spécifique pour un fonctionnement correct.

Pour recevoir les messages, il faut utiliser un circuit avec le 6N137 pour convertir le signal en haut ou en bas (+5v ou 0v) d'une bonne manière pour que le micro-contrôleur puisse l'interpréter. Le 6N137 dispose également d'une diode de protection pour limiter une éventuelle tension inverse.

Pour l'envoi, nous devons utiliser des «buffers» afin de ne pas avoir de perte de puissance dans le signal même avec de longues distances de câble, garantissant ainsi un système final plus fiable.

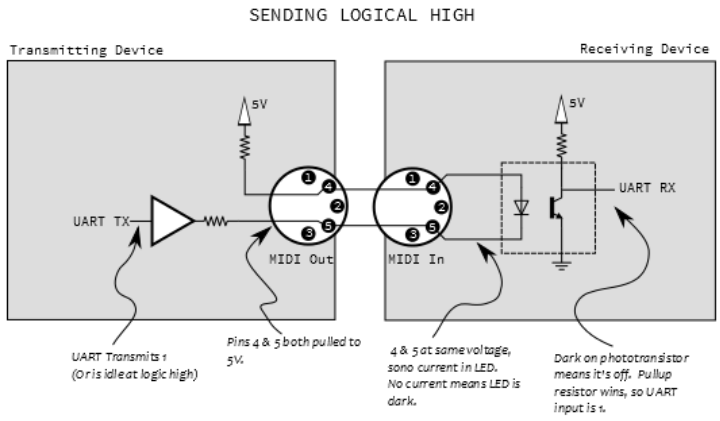


FIGURE 22 – Diagram MIDI IN et MIDI OUT

En utilisant le logiciel Eagle, nous pouvons générer les schémas ci-dessous :

Pour le circuit d'entrée, comme nous l'avons dit précédemment, nous utilisons un 6N137, une diode de protection et, sur la sortie 6, nous utilisons une résistance de pull-up. L'entrée 7 est filtrée par une capacité de 10pF.

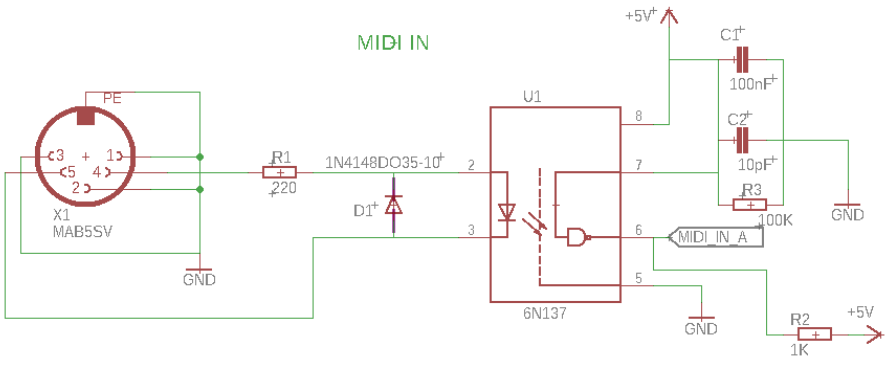


FIGURE 23 – Schéma MIDI IN

Pour le circuit de sortie, comme nous l'avons dit précédemment, nous avons ajouté les «buffers» 74LS04 avant le connecteur de sortie.

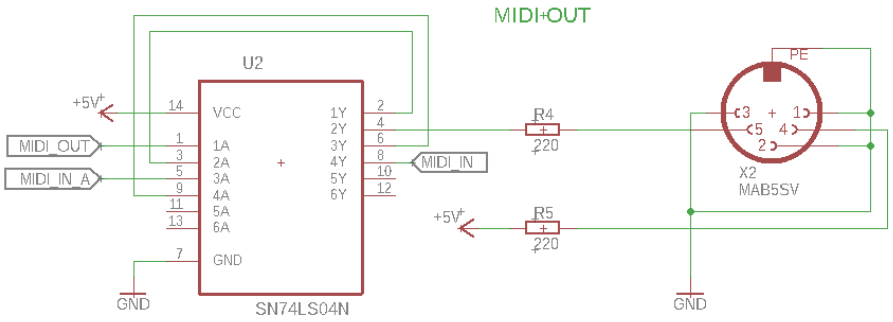


FIGURE 24 – Schéma MIDI OUT

Self1 et Self2 sont des filtres CEM de ligne afin d'éliminer les parasites des lignes RS-485.

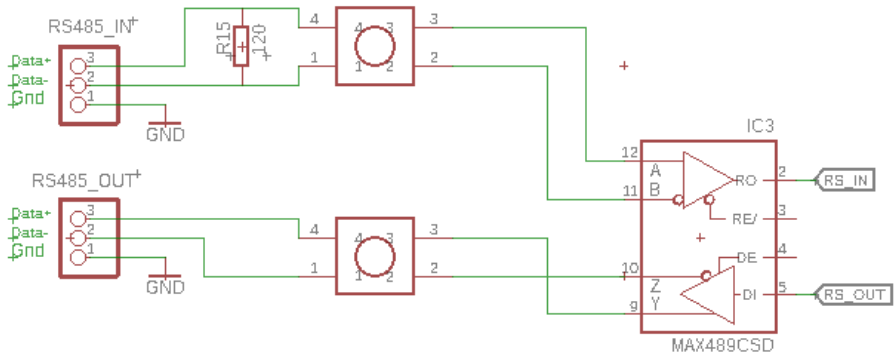


FIGURE 25 – Schéma RS-485 IN et OUT

Pour basculer entre les entrées et sorties MIDI et RS485, nous utilisons le multiplexeur CD4052D, dans le schéma EAGLE, c'est comme suit :

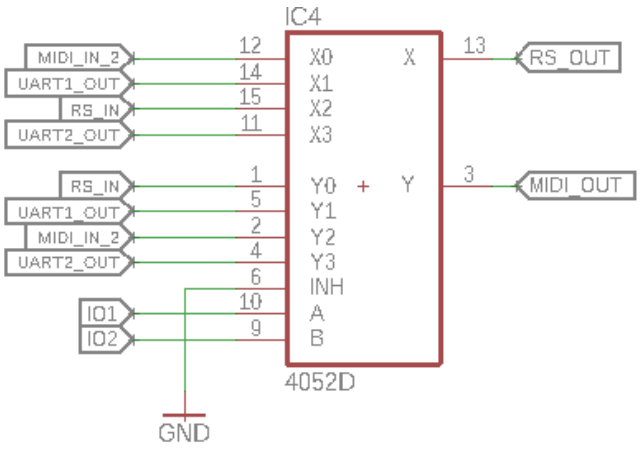


FIGURE 26 – Schéma Multiplexer CD4052D

### LEDs de vérification

Les LEDs permettent de visualiser certains paramètres comme l'envoi et la réception des messages, les erreurs de communication et la détection du faisceau lumineux par les capteurs.

Les LEDs sont pilotées par un circuit ULN2803 afin de limiter la puissance fournie par le micro-controleur sur ses sorties.

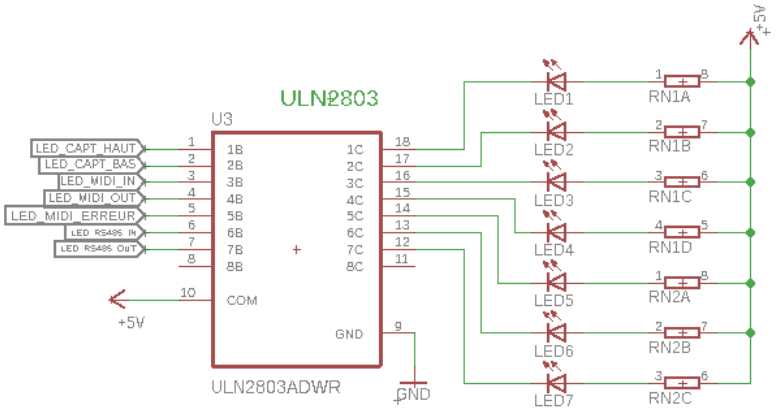


FIGURE 27 – Schéma activation des LEDs

### Alimentation

Le circuit d'alimentation ci-dessous convertit une tension d'entrée de 12v en tension de fonctionnement du circuit (+ 5v).Pour cela, nous utilisons le TSR 1-2450, une diode de protection et un condensateur de découplage.

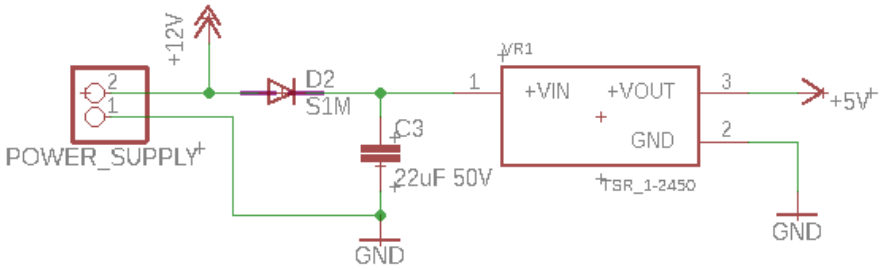


FIGURE 28 – Schéma Alimentation

### 3.2.3 PCB

Tous les tests et prototypes ont été réalisés à l'aide d'un «Breadboard», mais pour l'instrument final, nous avons besoin de quelque chose de plus définitif, pour cela nous allons fabriquer un PCB.

Un PCB sera beaucoup plus fiable et permet une miniaturisation.

En utilisant le logiciel EAGLE, nous avons fabriqué la carte ci-dessous, nous avons décidé d'utiliser un maximum de composants CMS pour réduire encore la taille de la carte.

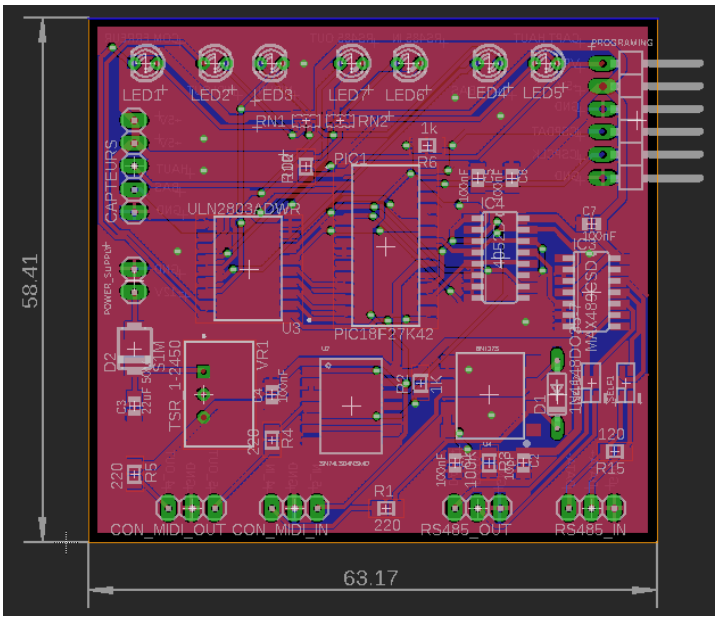


FIGURE 29 – PCB

A la fin nous obtenons la plaque suivante, elle a les dimensions de 6,3cm par 5,8cm Nous mettons également des écritures qui facilitent le sondage et son utilisation.

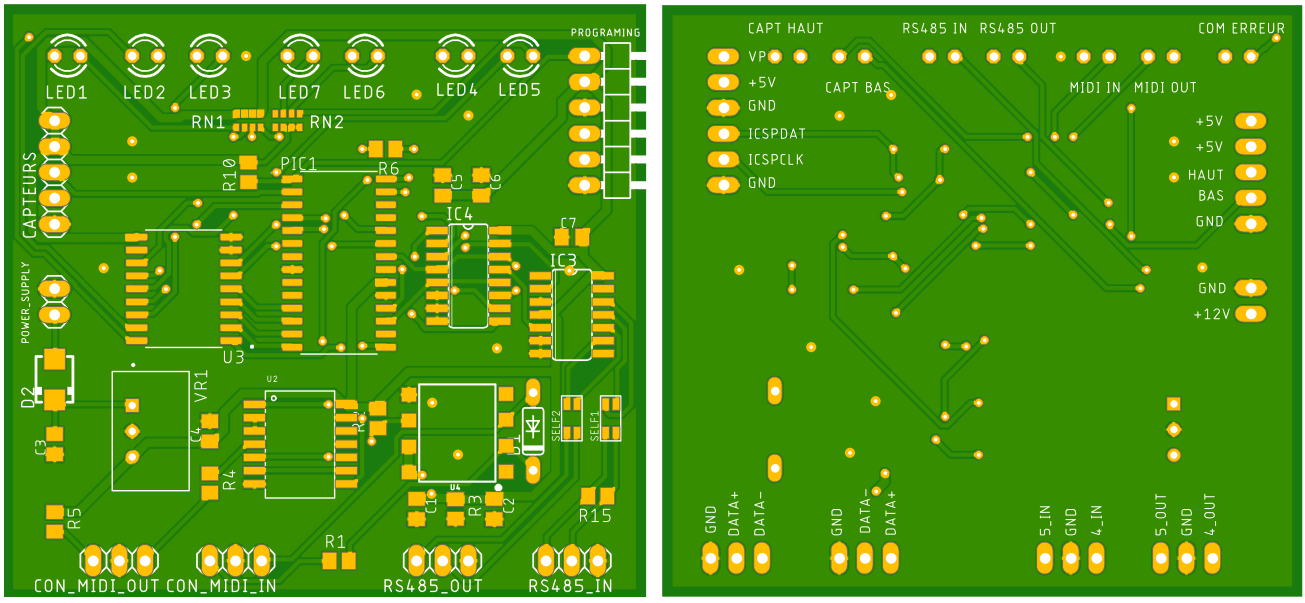


FIGURE 30 – PCB face avant et face arrière

## 3.3 Software

### 3.3.1 Protocole de communication

Pour envoyer et recevoir des notes, **NOTE ON et NOTE OFF**, nous utiliserons exactement le protocole MIDI décrit dans l'introduction, chaque note est composée de trois octets, le premier étant l'état ("NOTE ON" ou "NOTE OFF" + CHANNEL NUMBER) , le second la valeur de la note (0 à 127) de la plus basse à la plus haute, et le troisième et dernier la vitesse de la note (0 à 127).

Pour le **CONTROL CHANGE**, nous utiliserons des numéros de contrôle indéfinis dans la norme MIDI (de 20 à 31 en décimal) soit :

**-CC 20 (octet 2 = 20 en décimal)** : Type de frappe :

-si valeur du contrôle (octet 3) = 0 : mode simple frappe (par défaut) ;

-si valeur du contrôle (octet 3) = 1 : mode double frappe.

Le numéro de canal (1 à 15) correspond au numéro du boîtier pour le CC simple ou double frappe afin d'individualiser les boîtiers en simple ou double frappe. Le canal 16 permet de configurer simultanément tous les boîtiers en simple ou double frappe.

**-CC 21 (octet 2 = 21 en décimal)** : Numéro (adresse) du boîtier.

Octet 3 = le numéro du boîtier (de la colonne) de 1 à 5 ou plus. Cette valeur est mémorisée en EEPROM du PIC.

**-CC 22 (octet 2 = 22 en décimal)** : Note du boîtier. Octet 3 = la note envoyée par le boîtier à chaque passage du faisceau de 1 à 127. Cette valeur est mémorisée en EEPROM du PIC.

### 3.3.2 Réception et envoi de messages MIDI

Les messages NOTE ON et CC entrant sont mémorisés temporairement dans un buffer avant d'être retransmis sur la sortie MIDI de la maquette. Lorsqu'un message NOTE ON résultant du balayage de faisceau (appelé Beam NOTE ON) est à envoyer, alors deux cas se présentent :

Cas 1 : le buffer MIDI est vide, alors le Beam Note ON est envoyé directement sur la sortie MIDI.

Cas 2 : Le buffer MIDI n'est pas vide, alors le Beam Note ON est envoyé directement sur la sortie MIDI puis le buffer est vidé vers la sortie.

Cas 3 : Il n'y a pas eu de balayage de faisceau et le buffer n'est pas vide, alors le buffer est vidé vers la sortie MIDI.

Cas 4 : Le buffer est vide et il n'y a pas eu de balayage de faisceau, alors rien ne se passe et la boucle main continue.

Les cas 1 à 4 sont gérés dans le main du programme.

Le buffer est une FIFO (first in first out) qui se remplit à chaque entrée de message et se vide vers la sortie MIDI. Dans le buffer les octets sont groupés par 3 car chaque message (NOTE ON ou CC) tient sur 3 octets.

Les messages MIDI entrant sont analysés à la réception (dans le programme d'interruption).Lorsqu'il s'agit d'un message CC, l'action à exécuter est positionnée au sein du programme de réception (en interruption) sous forme de positionnement de flag ou modification de variable et l'action positionnée sera ensuite exécutée dans le main du programme. Après l'analyse du message MIDI entrant, celui-ci est stocké à la suite dans le buffer MIDI par le programme de réception en interruption. Le buffer MIDI est vidé vers la sortie MIDI au niveau du main en tenant compte des cas 1 à 4 ci-dessus.

### 3.3.3 Détection balayage de faisceau

Le système de détection d'un balayage de faisceau se compose d'une machine à 5 états comme indiqué ci-dessous, afin que nous puissions identifier si nous avons un balayage haut-bas ou bas-haut.

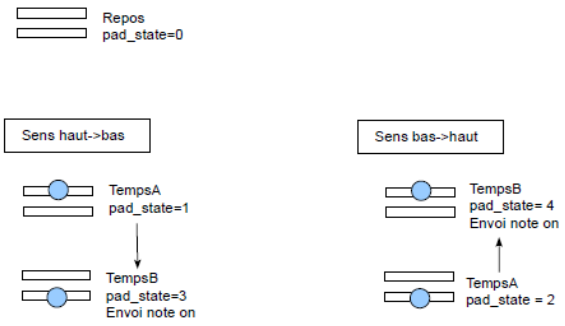


FIGURE 31

Nous avons créé deux fonctions principales de détection, "*detect_strokeHB*" et "*detect_strokeBH*", ces fonctions renvoient une variable de temps entre la détection de la rampe supérieure et de la rampe inférieure. Puis, avec cette valeur, nous calculons la valeur de vitesse qui sera envoyée par le message MIDI (NOTE ON).

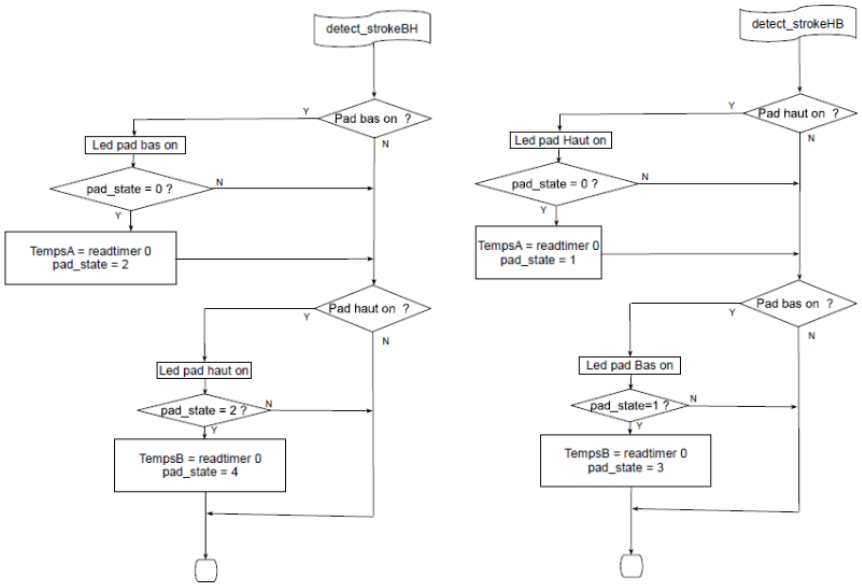


FIGURE 32

### 3.3.4 Calcul de la vélocité

Comme nous l'avons mentionné dans la partie "Protocole de Communication", la valeur de vitesse de la note doit être comprise entre 0 et 127, donc pour obtenir cette valeur nous utilisons le temps de détection entre les deux rampes calculé dans les fonctions "detectstrokeHB" et "detectstrokeBH".

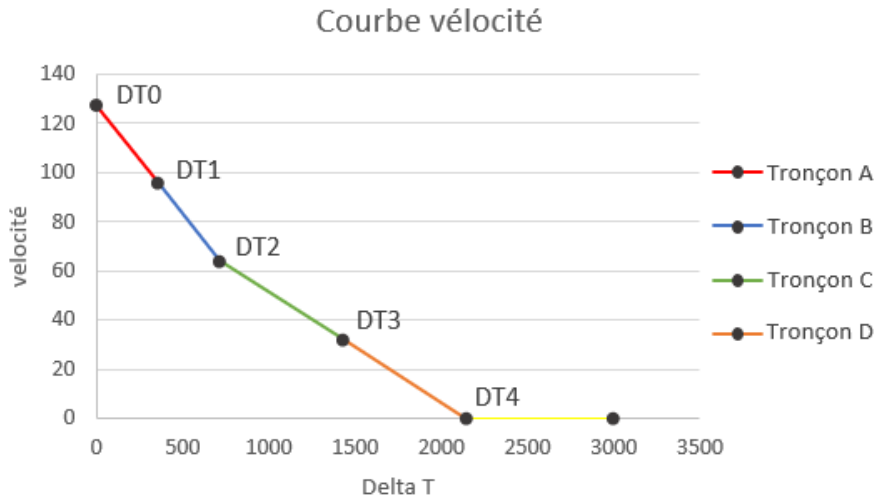


FIGURE 33 – Courbe vélocité

Pour la conversion, nous avons divisé la courbe en 4 tronçons, A, B, C et D. La ligne A a une vitesse maximale de 127, B est 90, C est 60 et D est 30.

Alors on fait le calcul de v (vélocité) pour chaque tronçon :

**Tronçon A**

A :

$$v = \frac{32 * (DT1 - \Delta T}{DT1 - DT0} + 96$$

**Tronçon B**

B :

$$v = \frac{32 * (DT2 - \Delta T}{DT2 - DT1} + 64$$

**Tronçon C**

C :

$$v = \frac{32 * (DT3 - \Delta T}{DT3 - DT2} + 32$$

**Tronçon D**

D :

$$v = \frac{32 * (DT4 - \Delta T}{DT4 - DT3} + 0$$

## 3.4 Test de communication avec Ableton Live

### 3.4.1 Test faisceau de lumière

Le premier test que nous avons effectué a été l'envoi d'une note après la détection d'un balayage du faiseau de lumière. À l'aide d'un oscilloscope, nous pouvons parfaitement vérifier le message envoyé.

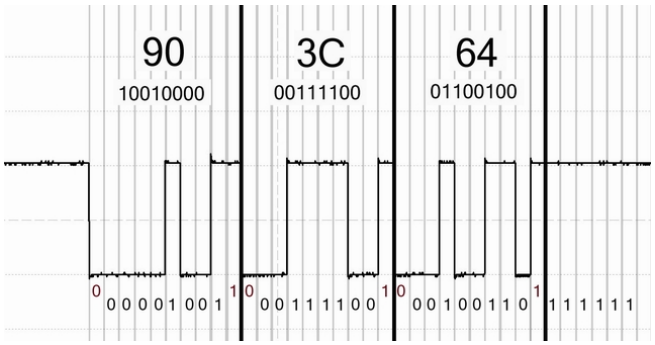


FIGURE 34

Le message ci-dessus est un NOTE ON, avec note C3 et vitesse de 64 ($mf$).

On peut observer que les bits du message sont inversés dans le temps c'est parce que le PIC interprète les octets comme "Little endian", c'est-à-dire que les bits les moins significatifs apparaissent en premier.

### 3.4.2 Test MIDI IN avec clavier

Pour tester la bonne réception et le renvoi des messages MIDI arrivant au PIC, nous avons créé un système "echo" utilisant "ableton live" et une interface USB-MIDI.

Pour envoyer des notes, nous utilisons le clavier de l'ordinateur et les envoyons sur le canal 1, puis les notes sont reçues par le PIC et renvoyées au logiciel où nous pouvons écouter et visualiser la bonne réception.

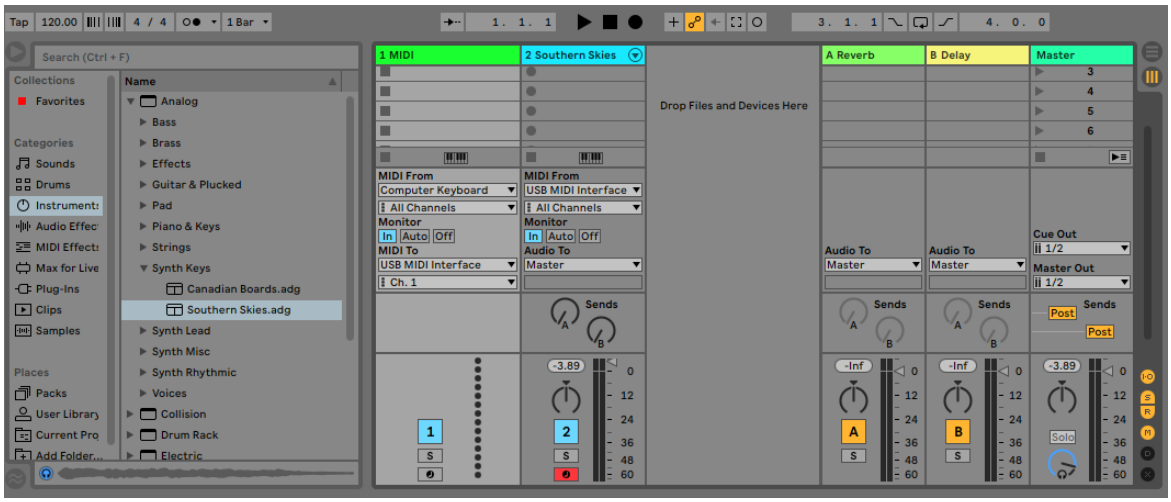


FIGURE 35

Pour un meilleur traitement des notes reçues et pour éviter tout type de problème de synchronisation, nous utilisons un "buffer" qui stocke tous les messages reçus (plus de détails sur le fonctionnement du buffer dans le partie "Réception et envoi de messages MIDI").

### 3.4.3 Test Control Change

Pour tester le fonctionnement de Control Change, nous utilisons également le logiciel "Ableton Live", avec lui nous pouvons envoyer des messages "Control Change" facilement et nous pouvons personnaliser les messages exactement comme nous le voulions.

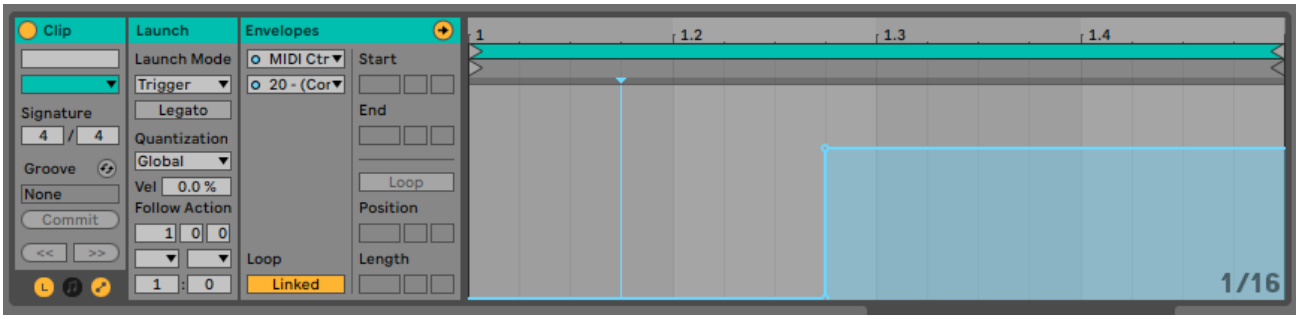


FIGURE 36

Ci-dessus, nous pouvons voir la fenêtre d'envoi du message "Ableton Live". Dans ce cas précis nous envoyons "22" dans le deuxième octet, c'est-à-dire que nous voulons modifier la note envoyée par la boîtier, et dans le troisième octet nous avons envoyé deux notes différentes, C3 (60) et C4 (72).

Pour tester, nous avons fait balayer la lumière à travers les capteurs et avons vu clairement que toutes les 3 secondes la note émise changeait avec les messages control change entrants.

# 4 Organisation

Afin de mener à bien notre travail de manière efficace et avec un temps bien réparti, il est important de procéder avant tout à une bonne planification. Dans notre cas, nous avons eu neuf semaines de travail et avons décidé de le répartir comme suit :

### Semaine 24 (08/06-12/06)

- Finalisation du schéma ;
- Codage initial.

### Semaine 25 (15/06-19/06)

- Réception du matériel ;
- Cablage du véroboard ;
- Test sur les leds (allumage-extinction) ;
- Tests des 2 ADC (oscillo, PWM).

### Semaine 26 (22/06-26/06)

- Calcul de la différence de tension entre les deux rampes et seuillages par rapport à un seuil constant (constante) ;
- Eclairage de la led A si différence positive et supérieure au seuil ;

— Eclairage de la led B si différence négative et supérieure au seuil en valeur absolue ;
— Test midi out avec logiciel MAO (declencht instrument) ;
— Test midi out avec logiciel MAO (declencht instrument) ;
— Eclairage led Midi out (à voir extinction).

### Semaine 27 (29/06-03/07)

— Calcul de la vélocité de note en balayage de faisceau (utilisation de la courbe à tronçons) ;
— Envoie de la note avec vélocité variable.

### Semaine 28 (06/07-10/07)

— Créer buffer Fifo (variable globale tableau) et gestion du pointeur de Fifo ;
— Programme interruption qui reçoit les notes on (envoyés par Ableton live ou un deuxième PIC) et les range dans le buffer ;
— Eclairage led midi in (quand octet reçu) (à voir extinction) ;
— Vidange du buffer vers midi out (dans le main).

### Semaine 29 (13/07-17/07)

— Réception des messages midi et envoie vers midi out (via Fifo) mergé avec traitement du faisceau et envoi du note on du faisceau ;
— Déclenchement instrument de Ableton live dans ces conditions avec une certaine note en provenance du midi in et une autre note quand balayage du faisceau ;
— Led com erreur (hard + soft).

### Semaine 30 (20/07-24/07)

— Option 1 : Ajustage de la fonction de frappe/double frappe ;
— Option 2 : Mise du type de frappe, numéro boitier et numéro de note (en EEPROM et RAM) avec réception control change depuis un deuxième pic.

### Semaine 31 (27/07-31/07)

— Option 3 : Ajout du max 489 (fonction RS485 sur uart2) dans le schéma.

### Semaine 32 (03/08-07/08)

— Routage carte sur eagle (avec boitiers CMS sauf le PIC en DIP).

## 5 Conclusion

Avec la fin du projet, nous voyons que l'un des points les plus positifs était la bonne organisation et la répartition des tâches depuis le début du stage, il a donc été possible de finaliser l'ensemble des parties du projet dans le temps imparti (options incluses) grâce à une bonne organisation et exécution des différentes tâches programmées.

Techniquement et d'après les différents tests, le projet s'avère fiable avec l'avantage de comporter des composants bon marché permettant sa fabrication à un coût raisonnable.

## Références


[1] Datasheet PIC18(L)F26/27/45/46/47/55/56/57K42 - http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18LF26-27-45-46-47-55-56-57K42-Data-Sheet-40001919E.pdf

[2] Summary of MIDI Messages - https://www.midi.org/specifications-old/item/table-1-summary-of-midi-message

[3] Expanded Messages List (Status Bytes) - https://www.midi.org/specifications-old/item/table-2-expanded-messages-list-status-bytes

[4] Control Change Messages (Data Bytes) - https://www.midi.org/specifications-old/item/table-3-control-change-messages-data-bytes-2

[5] The Standard MIDI File Format - https://www.music.mcgill.ca/ gary/306/week9/smf.html: :text=For%20example%2C%20audio%20WAV%20files,supposed%20to%20be%20big%20endian

[6] Hex to Binary converter - https://www.rapidtables.com/convert/number/hex-to-binary.html

[7] Universal Asynchronous Receiver Transmitter (UART) on 8-Bit PIC® Microcontrollers - http://ww1.microchip.com/downloads/en/AppNotes/TB3156-Univ-Asynchronous-Receiver-Transmitter-on-8bit-PIC-90003156A.pdf

[8] Timers (Microchip) - https://ww1.microchip.com/downloads/en/DeviceDoc/70205D.pdf

[9] Microchip Developer Help - https://microchipdeveloper.com/mcc:interrupts: :text=When%20a%20developer%20uses%20MCC,user%2Dsupplied%20files%20and%20main

[10] Using the internal PIC's AUSART to send MIDI data - http://www.barrysoft.it/blog/midi-with-pic-ausart.html

[11] MIDI Tutorial - http://www.music-software-development.com/midi-tutorial.html: :text=The%20NOTE%20ON%20message%20is,note%20when%20it%20is%20hit

[12] SN74LS04 Texas instruments - https://www.ti.com/lit/ds/symlink/sn74ls04.pdf

[13] 6N137 High Speed Optocoupler - https://www.vishay.com/docs/84732/6n137.pdf

[14] UART Overrun Error - https://community.cypress.com/docs/DOC-12028

[15] CD405xB CMOS Single 8-Channel Analog Multiplexer with Logic-Level Conversion - https://www.ti.com/lit/ds/symlink/cd4052b.pdf?ts=1595967370698&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FCD4052B

[16] MAX489 - Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers - https://www.maximintegrated.com/en/products/interface/transceivers/MAX489.html

[17] TSR 1-2450 - Convertisseur DC/DC - http://www.farnell.com/datasheets/2648940.pdf?_ga=2.180616622.805355247.1596203450-1667595797.1596203450

[18] 744232161 - Filtre de ligne de puissance - http://www.farnell.com/datasheets/1735625.pdf?_ga=2.109220236.805355247.1596203450-1667595797.1596203450

[19] CAT/CAY 16 Series - Chip Resistor Arrays - http://www.farnell.com/datasheets/1885635.pdf?$_g a = 2.237609536.1743469988.1596464236-1667595797.1596203450

[20] ULN2803ADWR Footprint - https://www.snapeda.com/parts/ULN2803ADWR/Texas%20Instruments/view-part?ref=searcht=ULN2803

[21] MAX489CSD+T Footprint - https://www.snapeda.com/parts/MAX489CSD%2BT/Maxim%20Integrated/view-part/?ref=searcht=max489c

[22] PIC18F27K42-I/SO Footprint - https://www.snapeda.com/parts/PIC18F27K42-I/SO/Microchip/view-part/?ref=searcht=PIC18F27K42