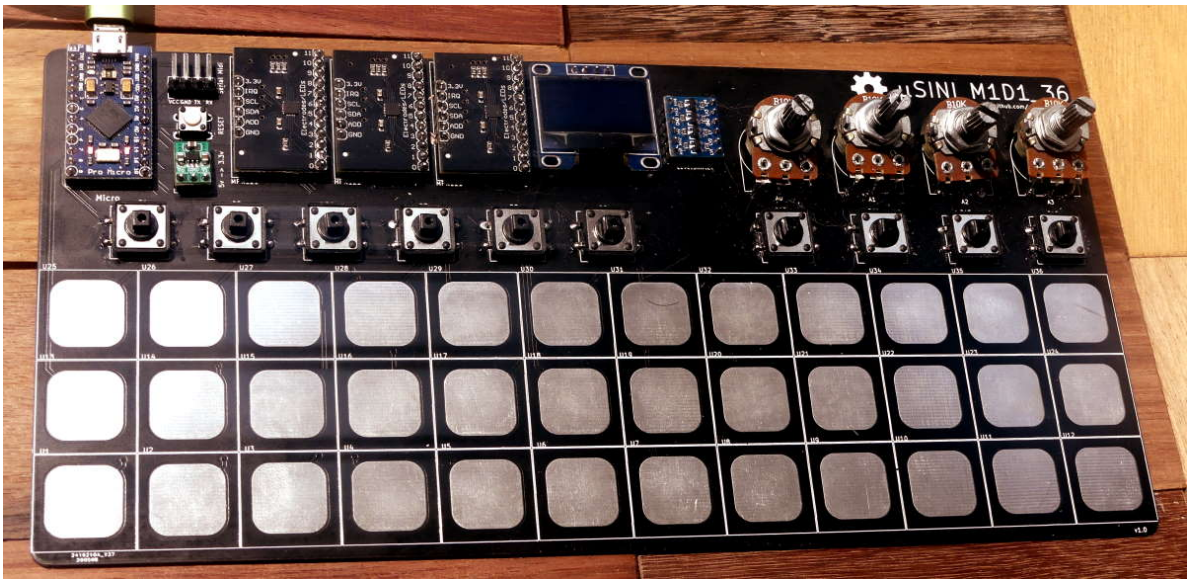
Pads Midi Customisable

Guide complet pour fabriquer un instrument midi customisable avec des pads, des boutons, des potentiomètres et un écran.

- Boutons (buttons.h)
- Midi USB et série (midi.h)
- Pads (mpr121.h)

Introduction

Dans ce guide je vais vous expliquer en détail comment j'ai fabriqué cette instrument midi et comment je l'utilise **pour faire de la musique**,avec un **ordinateur** , un **Raspberry Pi** et même à l'aide d'un **Arduino**



Ce guide est en trois parties :

Dans la première partie, je vais vous expliquer comment créer ce kit

- Les **composants utilisés** et où les **commander**
- Comment **commander un PCB** à partir d'un gerber
- **Assemblage du PCB**
- Comment **programmer** notre instrument

Dans la **seconde partie**, je vais vous montrer comment on peut utiliser cette instrument

- Utilisation dans LMMS avec un plugin de guitare / batterie / basse gratuit
- Utilisation dans Reason avec un plugin de guitare (métal) / batterie / basse payant
- Utilisation avec Patchbox OS

Dans une troisème partie, je vais vous expliquer comment le modifier

- Comment changer la forme et le placement des pads sur kicad
- Explication du code et comment customiser le fonctionnement
- Les améliorations et alternatives possibles

Fabrication

Fabrication

Liste des composants

Sparkfun Pro Micro □□

□□ Officiel : https://www.sparkfun.com/products/12640

□□ Clone (Amazon) : https://www.amazon.fr/KeeYees-ATmega32U4-Développement-Leonardo-Bootloader/dp/B07FQBQ4Z6

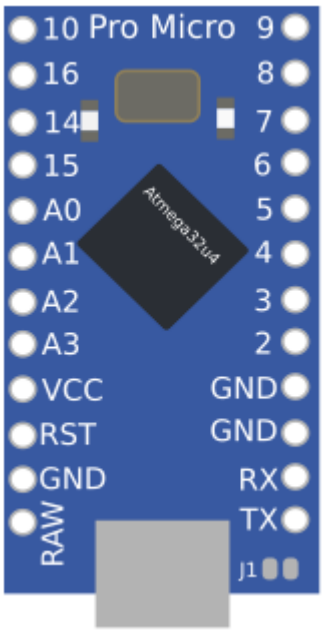
□□ Clone (Aliexpress) :

https://fr.aliexpress.com/item/1871481789.html?spm=a2g0s.9042311.0.0.20aa6c37z4qLog

Voici le **"cerveau"** de notre instrument, pour cette tâche nous allons utiliser un clone de **Sparkfun Micro.**

Souvent confondu avec l'Arduino Micro, le **Sparkfun Pro Micro est plus petit mais n'a pas les broches A4 / A5 / D11 / D12 / D13 / D17 (SS)**

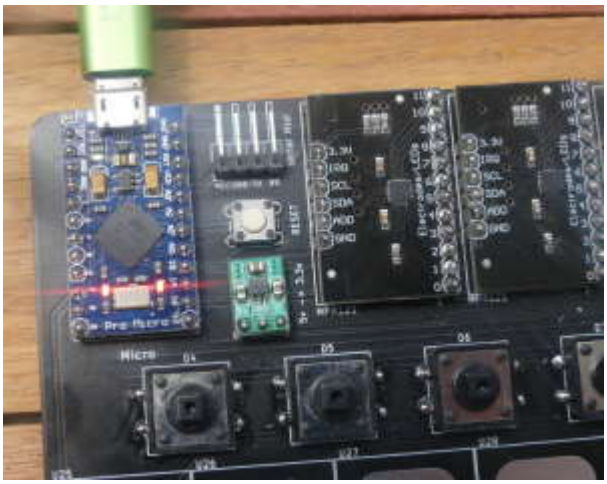
> □□ Les plans sous Eagle (et fusion 360) sont disponibles sous licence open source.



Cette carte se programme comme un Arduino Uno, mais elle est capable d'émuler le fonctionnement d'un **périphérique USB** (Un **clavier** / Une **souris** / Un **joystick** et un **instrument midi**) car elle utilise un

atmega32u4 (au lieu du atmega328 sur l'arduino uno)

> ⚡ J'ai utilisé la version 5V, les capteurs capacitifs fonctionnant en 3.3V, on pourrait simplifier le circuit en utilisant la version 3.3v.



Capteurs Capacitifs MPR121 □□

□□ Officiel : https://www.sparkfun.com/products/retired/9695

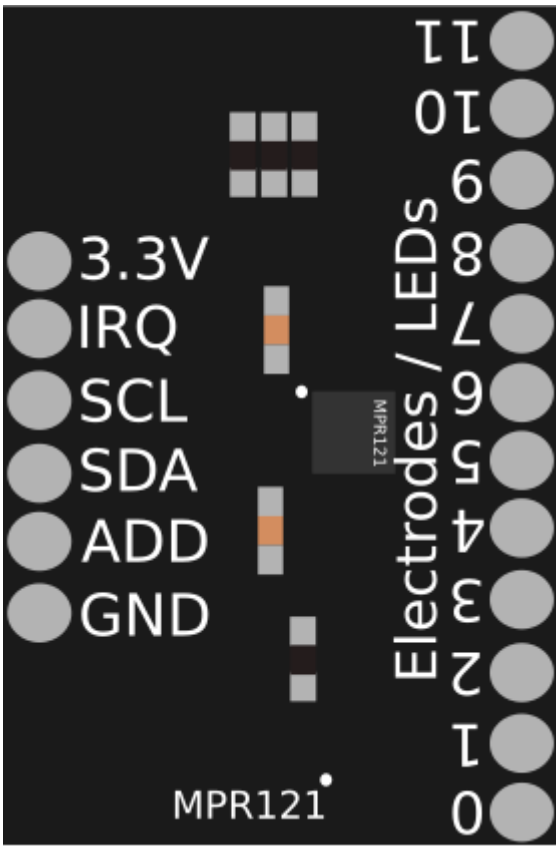
□□ Clone (Amazon) : https://www.amazon.fr/TECNOIOT-Breakout-Capacitive-Controller-Keyboard/dp/B084BVLXCB

□□ Clone (Aliexpress) : https://fr.aliexpress.com/item/32821362153.html

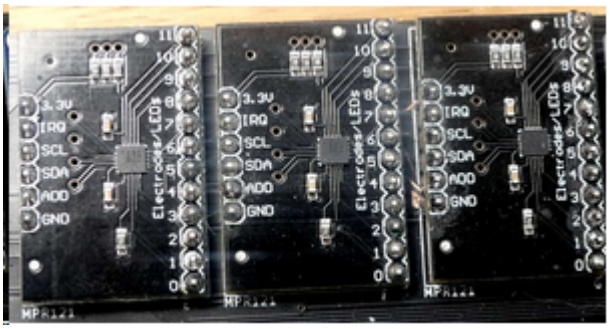
Les capteurs MPR121, réagissent quand on les touchent **avec la main** □□, ou **avec un objet conducteur** □□**.**
Il nous suffit donc d'exposer la broche à l'aide de **pastilles de soudures,** ce qui nous permet de créer les formes que nous voulons.

> □□ Les plans sous Eagle (et fusion 360) sont disponibles sous licence open source.

> Chaque module est capable de gérer **12 pads** et il est possible d'en utiliser **4 à la fois** (**46 Pads**)



> Ces modules communiquent en i²c, il ne nécessite que 2 broches (SDA / SCL) pour pouvoir l'utiliser, peut importe le nombre de MPR121 utilisés.



Boutons □□

□□ Sparkfun : https://www.sparkfun.com/products/15326

□□ Amazon : https://www.amazon.fr/POPESQ®-Interrupteur-instantane-Momentary-A2105/dp/B07DRRX4P3

□□ Aliexpress : https://fr.aliexpress.com/item/32834276752.html

Grâce à l'économie de broches que les capteurs capacitifs nous offrent, il est possible d'ajouter 10 boutons sur les broches restantes sur le Sparkfun micro.

>

> Les boutons sont reliés sans résistances à l'Arduino grâce auxrésistances de rappel (pull-up resistors) intégrées à celui-ci

4 des boutons sont placés au niveau des potentiomètres et les 6 autres à coté.

Potentiomètres •

□□ Amazon : https://www.amazon.fr/potentiometre-Simple-lineaire-Conique-rotatif/dp/B018S9GUKI

□□ Aliexpress : https://fr.aliexpress.com/item/33051479190.html

Le Sparkfun micro a 4 sorties analogiques, parfait pour gérer 4 potentiomètres.
Ceux ci vont nous permettre de régler des effets / synthétiseurs au sein des logiciels de musique assisté par ordinateur.

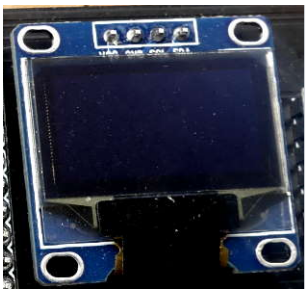
> La résistance mécanique des potentiomètres est uniquement soutenu par les soudures.

Afficheur OLED 128x64 □□

□□ Aliexpress : https://fr.aliexpress.com/item/32957159545.html

Ces afficheurs de 128x64 utilise la technologie OLED, à la différence des afficheurs LCD, le contraste est plus élevé , l'image très clair (car chaque pixel sont éclairés individuellement).
Ils fonctionnent en i²c et utilise donc les mêmes broches que pour les capteurs capacitifs.

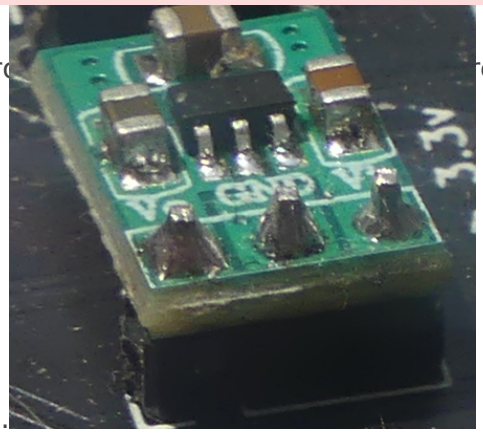
> Certains modules inversent les broches VCC / GND et d'autres ont plus de broches car ils utilisent le protocoles SPI, faites très attention en les commandant.



Régulateur de tension 5v vers 3.3v ⚡

◫ Aliexpress : https://fr.aliexpress.com/item/32718499724.html

> Tout les modules de régulation de tension n'ont pas les broches aux même endroit.
> Il faut qu'il est Vo / GND / Vi

Le sparkfun Micro 5v n'a pas de br 

éduire la tension à 3.3v afin d'alimenter les capteurs capacitifs.

Convertisseur de niveau logique ⚡

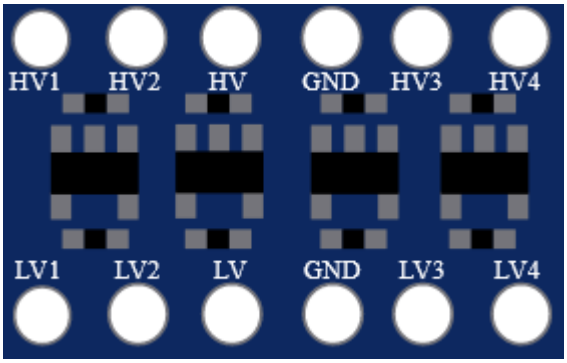
◫ Sparkfun : https://www.sparkfun.com/products/12009

◫ Amazon : https://www.amazon.fr/Hiletgo-modules-convertisseur-bidirectionnel-canaux/dp/B07F7W91LC

◫ Aliexpress : https://fr.aliexpress.com/item/4000587260340.html

Afin que les capteurs capacitif communique avec le Sparkfun micro il nous faut aussi un convertisseur de niveau (Logic Level Converter)

Celui-ci est alimenté d'un coté en 5v (HV) et de l'autre en 3.3v (LV) est converti le signal i²c (SDA/SCL) entre ces deux tensions (HV1 / HV2 <--> LV1 / LV2)



Fabrication

Commande du PCB

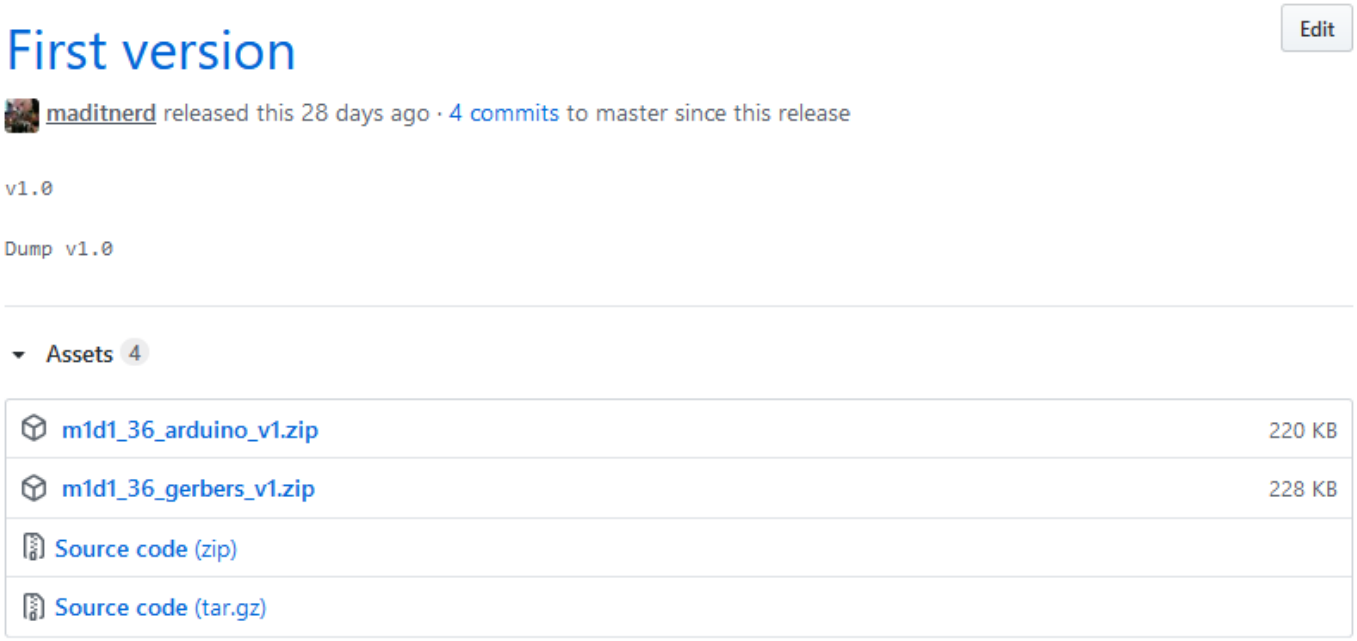
Téléchargement du gerber □□

Afin de commander les PCB, nous avons besoins du **gerber** de celui-ci

> Le format de fichier **Gerber** est le standard de-facto utilisé pour transmettre des informations concernant la fabrication des circuits imprimés. Il contient la description des diverses couches de connexions électriques (les pistes, les pastilles, les plages CMS, les vias...).

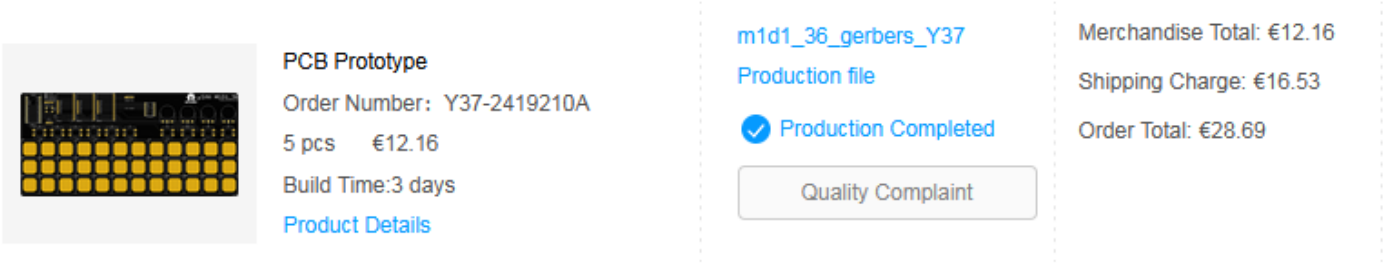
Vous pouvez les télécharger sur github

□□ Gerbers: https://github.com/usini/m1d1_36/releases



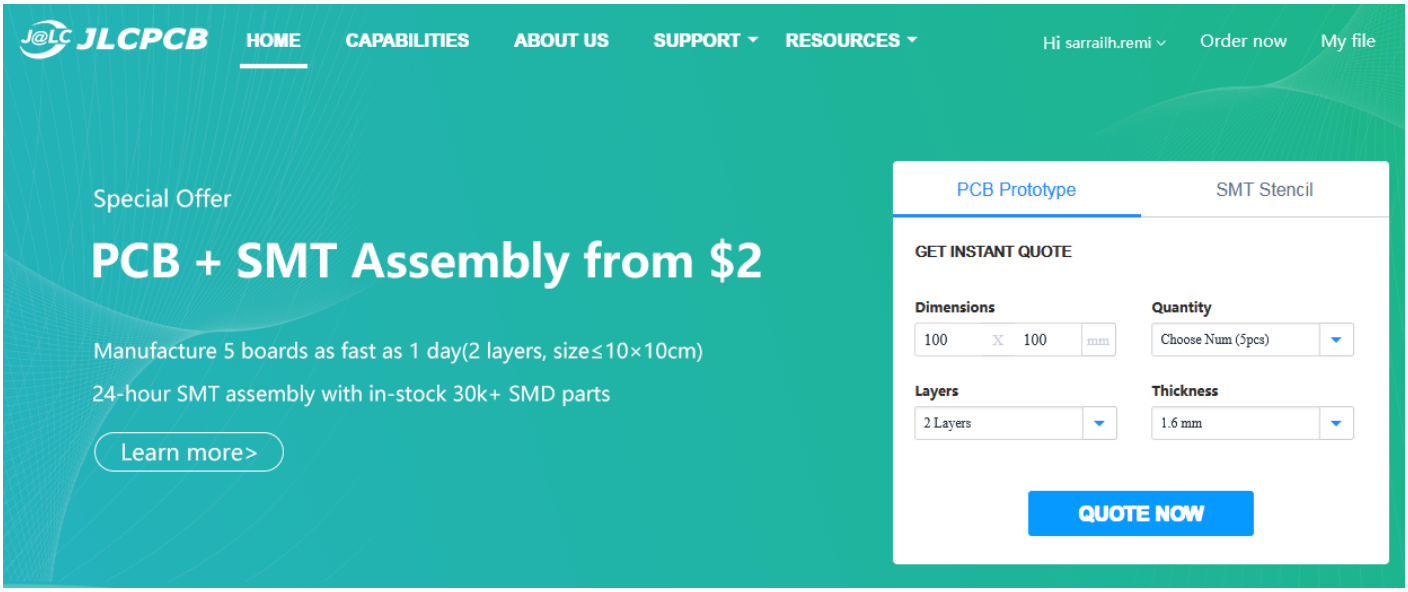
Commande du PCB □□

Après avoir essayer plusieurs fournisseurs, le seul a avoir un prix raisonnable pour un PCB de cette taille est JLCPCB.
5 exemplaires du PCB coûtent 12.16€, et en livraison rapide le tout revient à 28.69€ soit 5,7€ le PCB.

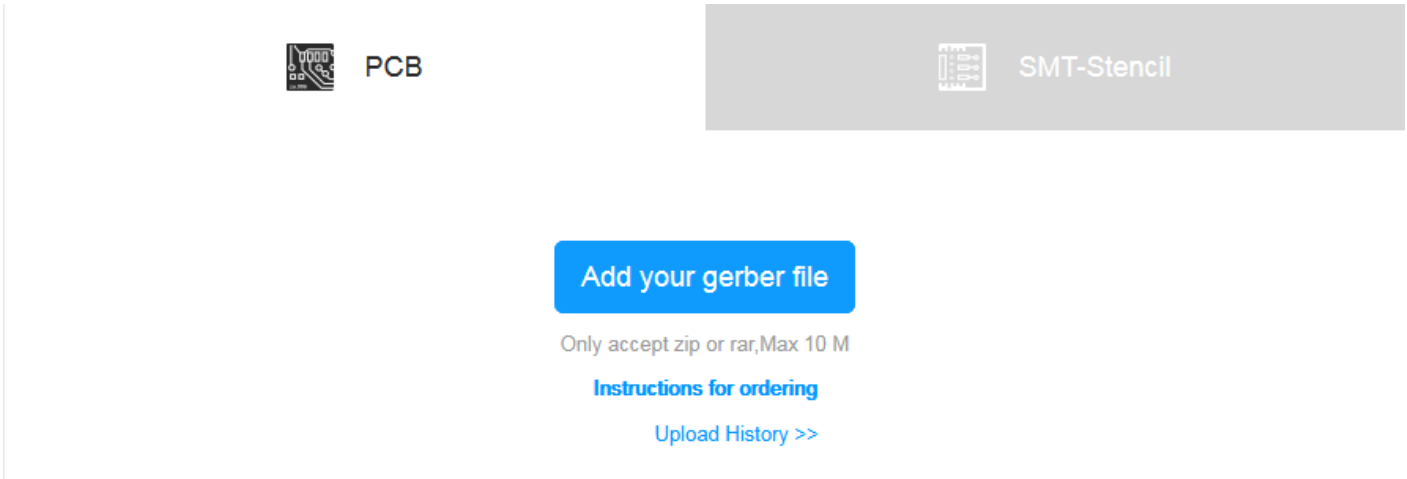


Toutefois la commande d'un PCB est a peu près identique sur chaque plateforme, voici les étapes pour commander le PCB

Aller sur https://jlcpcb.com/ puis cliquer sur **QUOTE NOW**

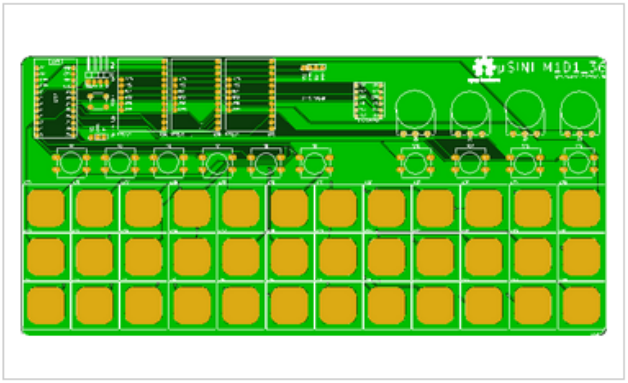


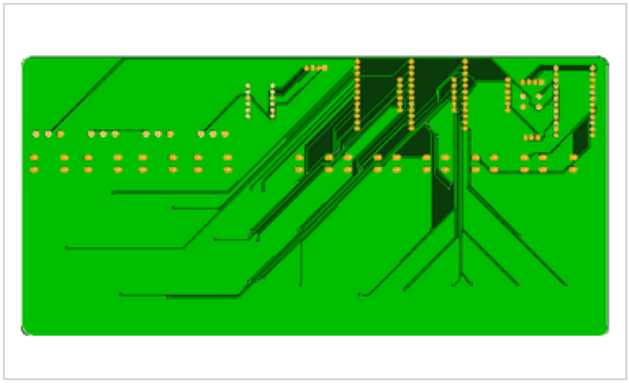
Cliquer sur le bouton **Add your gerbers** et ajouter le fichier **m1d1_36_gerbers_v1.zip**



Detected 2 layer board of 117x244mm(4.6x9.62 inches) .

Your upload has finished processing. Enter the project details below and we'll move on to checking all the individual layers to make sure that they're correct.





The gerber viewer is for reference purpose only and may differ from the actual PCB product.

Gerber Viewer

Success,this file has been saved to your File Manager

Vous n'avez pas besoin de changer la moindre option, mais si vous voulez la couleur noir (par ex.) vous pouvez la changer ici (**cela augmentera la durée de fabrication**)

- Cliquer sur **Save to Cart**
- Cliquer sur **Checkout securely**

Assemblage du PCB

Modification sur le MPR121 ✂

> Attention si vous ne faites pas cette modification, les MPR121 dont les adresses sont modifiés physiquement bloqueront l'exécution du programme.
> https://electronics.stackexchange.com/questions/325702/how-to-cut-the-add-to-gnd-trace-on-a-mpr121-capacitive-touch-sensor#325714

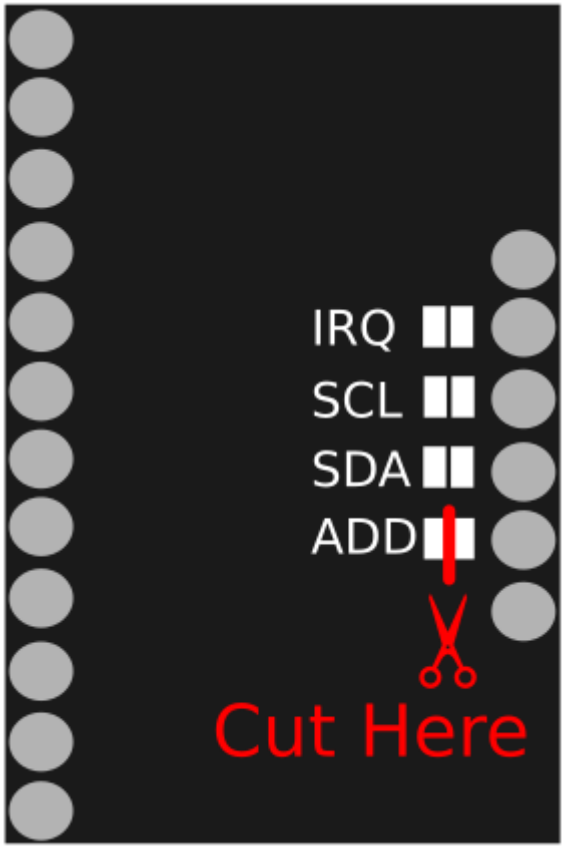
✖ Les clones de MPR121 ont un défaut, on ne peut pas changer leur adresse i²c, et donc en utiliser plusieurs.

Pour changer l'adresse d'un MPR121, il suffit de relier la broche **ADD** soit au 3V, au SDA ou au SCL

- **ADD non connecté : 0x5A**
- **ADD sur 3V : 0x5B**
- **ADD sur SDA : 0x5C**
- **ADD sur SCL : 0x5D**

Pas de panique, même si cela semble un peu risqué, c'est une modification relativement simple (avec les bons outils)
Pour que cela marche, **il faut couper une piste** (par ex à l'aide d'un petit tournevis plat)

Grattez la surface entre les deux pads de soudures jusqu'à ce que le contact soit rompu (vérifier ceci à l'aide d'**un multimètre**)



https://www.youtube.com/embed/VhU-bXgKgZ0

Soudure des composants

Ici rien de bien compliquer, le sens de chaque module est indiqué sur le PCB.

> Vous pouvez utiliser des broches 2.54mm femelles si vous voulez pouvoir retirer les modules (mais cela augmentera la hauteur des modules)
> Pour l'afficheur c'est plus embêtant vu qu'il ne sera pas soutenu correctement.

> J'attends de recevoir les composants pour souder les PCB restants et documenter la fabrication

Programmation

La programmation de la carte est relativement simple,

Vous pouvez télécharger la dernier version ici

Code : https://github.com/usini/m1d1_36/releases

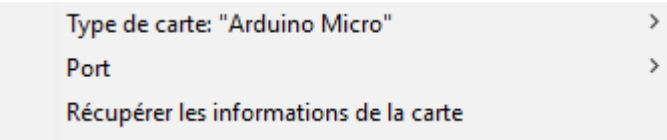
Bibliothèques

Toutes les bibliothèques peuvent s'installer à l'aide du gestionnaire de bibliothèques

- **AceButton** (Brian Park - MIT https://github.com/bxparks/AceButton)
- **MIDIUSB** (Arduino, Gary Grewa - GNU v2.1 https://www.arduino.cc/en/Reference/MIDIUSB)
- **Adafruit SSD1306** (Adafruit Industries - BSD Licence https://github.com/adafruit/Adafruit_SSD1306)
- **Adafruit MPR121** (Adafruit Industries - BSD Licence https://github.com/adafruit/Adafruit_MPR121)

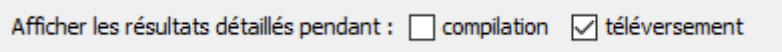
Téléversement

Choisissez la carte Arduino Micro, **le port correspondant** à votre carte et appuyez sur **téléverser** .



> Les Arduino Micro ont parfois du mal à être programmer car elle change de port série (COM sous Windows) avant que le programme se copie.
> Si c'est le cas appuyez sur le bouton RESET après avoir lancer le téléversement.

Si vous n'arrivez pas à la programmer, vous pouvez vérifier que la programmation se passe bien en changeant ce paramètre dans **les préférences**



Configuration

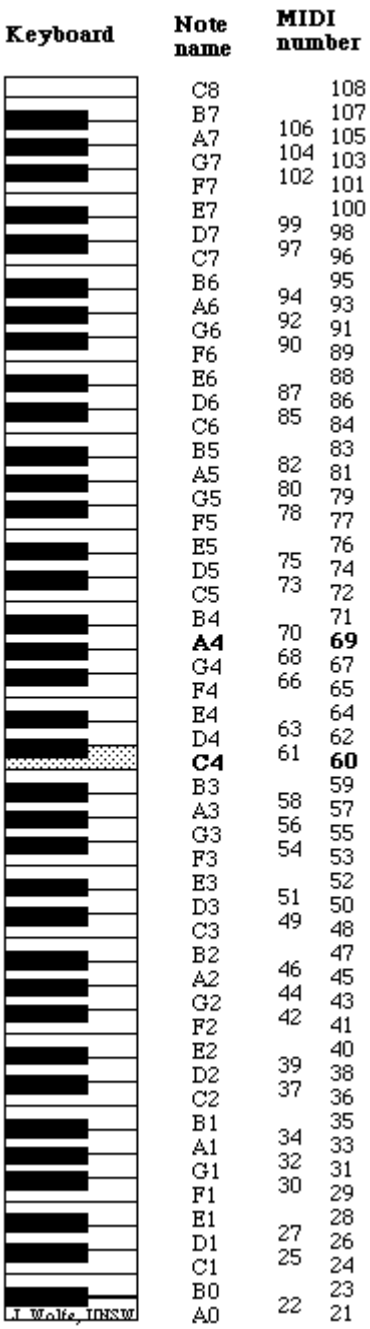
Il est possible de changer les notes des pads (**capNote**) dans l'onglet **settings.h** ainsi que les valeurs CC (Control Change) des boutons et des potentiomètres.

- https://newt.phys.unsw.edu.au/jw/notes.html
- https://www.midi.org/specifications-old/item/table-3-control-change-messages-data-bytes-2

m1d1_36 | OLED.h | buttons.h | midi.h | mpr121.h | pots.h | settings.h

```
/*!
 * @file settings.h
 *
 * Documentations \n
 * Control Change : https://www.midi.org/specifications-old/item/table-3-control-change-messages-
 * Midi Note      : https://newt.phys.unsw.edu.au/jw/notes.html
*/

uint8_t transpose = 1; ///< Default Transposition (by step of an octave, 12 notes)
uint8_t channel = 1; ///< Default Channel
byte btn_cc[10] = {0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19}; ///< Buttons Cont
byte pots_cc[4] = {0x01, 0x02, 0x03, 0x04}; ///< Potentiometers Control Change parameters
uint8_t capNote[36] = {28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
                       33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44,
                       38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49
                      }; ///< Note from pad (before transposition)
const byte SCREEN_ADDRESS = 0x3C; ///< Screen I²C address
const bool SERIAL_ACTIVE = true; ///< Enable Serial debug
```



00	Bank Select
01	Modulation Wheel or Lever
02	Breath Controller
03	Undefined
04	Foot Controller
05	Portamento Time
06	Data Entry MSB
07	Channel Volume (formerly Main Volume)
08	Balance
09	Undefined
0A	Pan
0B	Expression Controller
0C	Effect Control 1
0D	Effect Control 2

Points à améliorer

Utilisation

Guitare Acoustique

Dans cette exemple, nous allons voir comment utiliser mon instrument avec des plugins gratuits (mais **pas open source** par contre) afin de créer un ensemble **Guitare Acoustique** / **Basse** et **Batterie**.

> Malheureusement, les plugins guitares / basses ne fonctionnent pas sous Linux

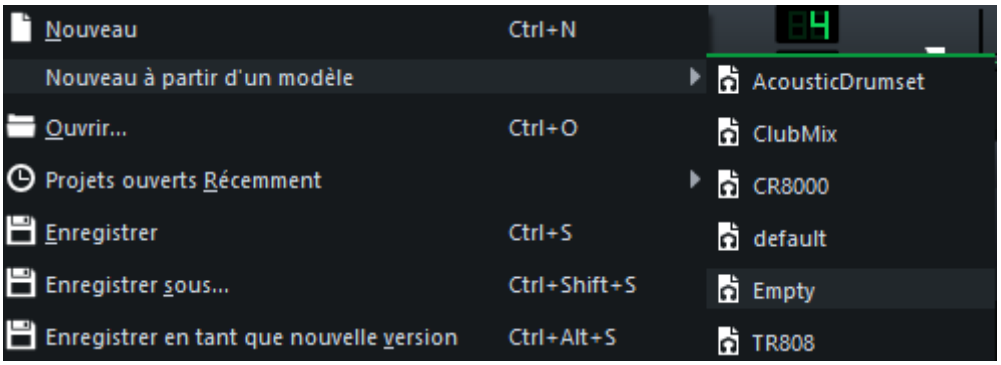
Installation

Pour cela il va nous falloir installer 4 programmes

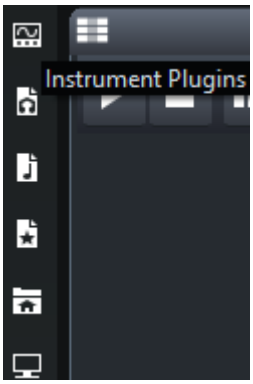
- Ample Guitar M II Lite : Guitare
- Ample Bass P Lite II : Basse
- MT-PowerDrumKit : Batterie
- LMMS : Logiciel de Musique Assisté par Ordinateur

Utiliser des VST dans LMMS

Par défaut LMMS s'ouvre avec un modèle, faisons le vide en allant dans **Fichier** --> **Nouveau à partir d'un modèle** --> **Empty**



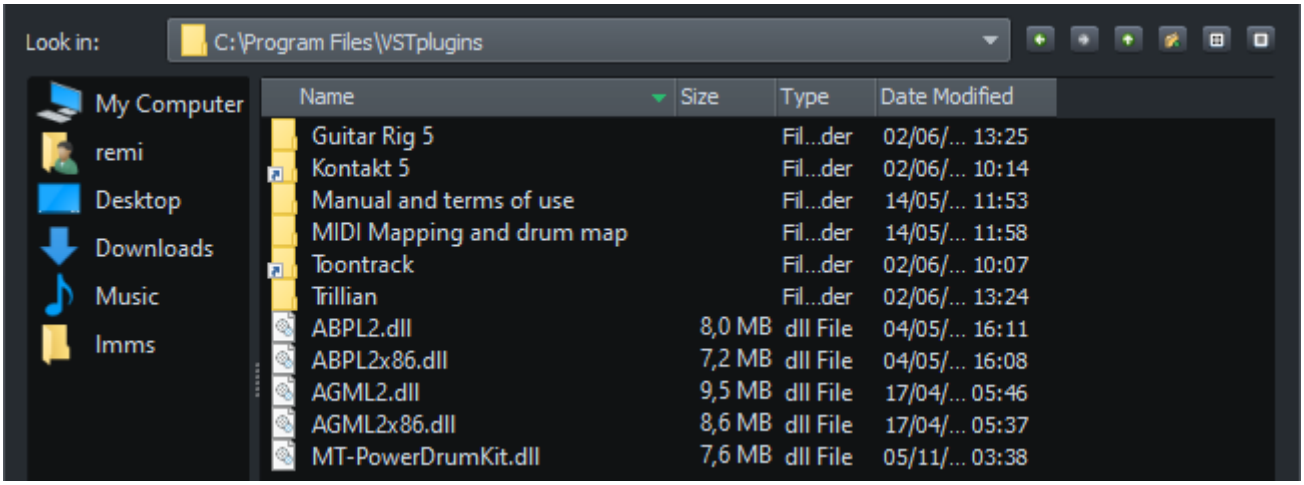
Cliquer sur Instruments Plugin et choisissez (en bas de la liste), **VeSTige**



> A la différence des autres logiciels de M.A.O, qui demande à ce que les plugins VST soit installer dans un dossier précis, dans LMMS, nous pouvons ouvrir les plugins directement.

Par défaut, les plugins VST s'installe dans **Program Files\VstPlugins,** les VST ont comme extension **.DLL**
MT-PowerDrumKit peut être installer où vous voulez mais je recommande de l'installer au même endroit afin qu'il marche avec les autres logiciels de M.A.O

- **ABPL2.dll** : Basse
- **AGML2.dll** : Guitare
- **MT-PowerDrumKit.dll** : Batterie

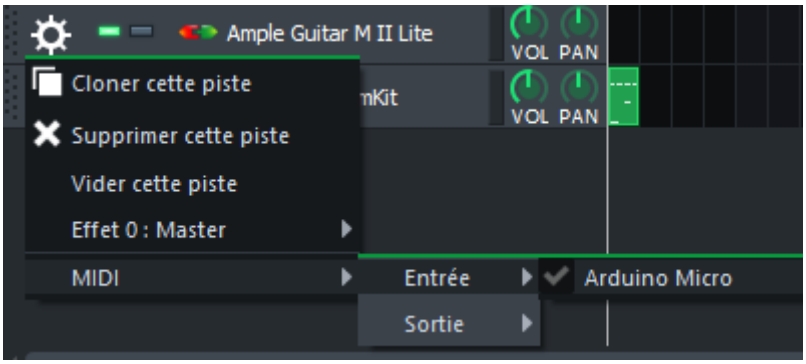


Au final, vous devriez avoir 3 **VeSTige** dans votre **éditeur de morceau**

> MT-PowerDrumKit est gratuit, mais nécessite (à l'instar d'un WinRaR) de passer un écran de démarrage, il vous suffit de faire une donation pour ne plus avoir cette écran.

Contrôler un instrument en MIDI

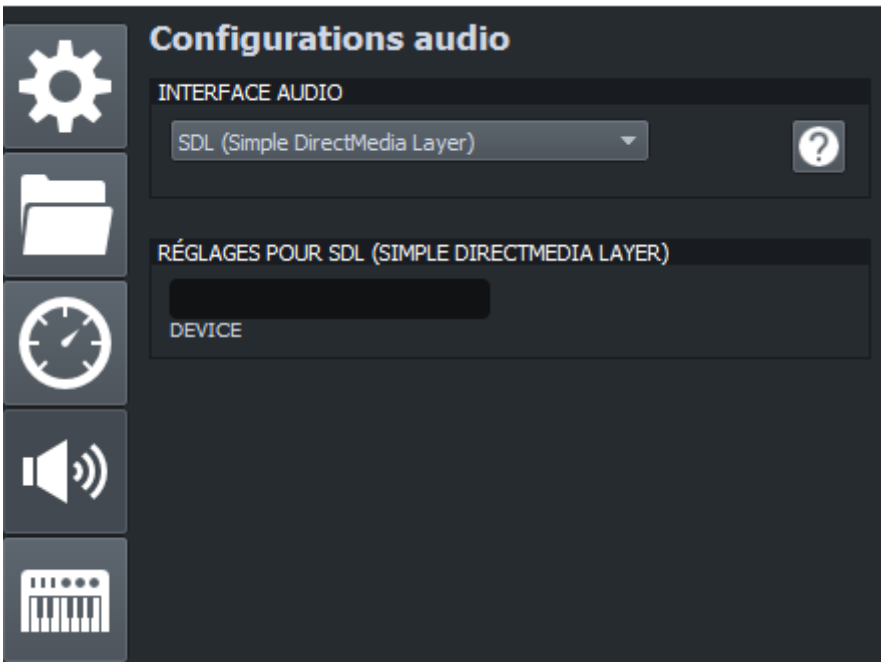
Cliquer sur l'**engrenage** de la piste que vous voulez contrôler et cliquez sur **MIDI** --> **Entrée** --> **Arduino Micro**



> La gestion du MIDI dans LMMS impose de le **redémarrer** pour **détecter des nouveaux périphériques MIDI**.
> Si vous n'avez pas brancher votre instrument MIDI avant le démarrage ou qu'il a été débranché, vous devez redémarrer le logiciel.

Pas de son ?

Vous pouvez changer le réglage du son dans **Éditer --> Configuration --> Configuration Audio**



Utilisation

Guitare électrique

Voici un exemple d'utilisation avec Prominy-V Metal / Guitar Rig

> J'ai utilisé Propellerhead Reason pour cette exemple, mais n'importe quel logiciel capable de faire fonctionner des VST peut faire l'affaire.

https://www.youtube.com/embed/bJb0MLeT7dg

https://www.youtube.com/watch?v=bJb0MLeT7dg

Prominy-V Metal est un instrument VST dont le but est de simuler une guitare électrique.

Pour cela il nous faut simuler différents "effets"

- Les cordes étouffés (PALM MUTE)
- Les harmoniques
- Les différents accords (En mode Lite, les accords sont joués en une seul note)
- Les blocages des cordes

Le fonctionnement de ces effets est très simple, la plupart des changements se font en appuyant **sur une note précise**, ou en **changeant la vélocité**.

Prominy-V est très complet (vu que le plugin coûte 299$ heureusement), et il y a aussi beaucoup de possibilités que je n'ai pas encore exploré.

Fonctionnement

Nous allons utiliser la partie la plus à gauche de l'instrument pour gérer ces différents effets.
Le reste de l'instrument simule les trois premiers cordes d'une guitare.

> **Palm Mute** et **Harmonics** doivent être laisser appuyer pour fonctionner

Distortion

Prominy-V Metal n'a pas d'effets ou de simulateur d'ampli, il nous faut donc, en plus, un simulateur d'effets.

Au point où on en est niveau tarif autant prendre le top pour ça : Guitar Rig 5

Comme point de départ, nous allons utiliser le Preset [94 Rock Solo], afin d'avoir un son correct, il va nous falloir faire quelques réglages, combiner deux sons de guitares et une bonne stratégie pour avoir un son plus recherchés.

> Le secret pour avoir un bon son de guitare métal (si on a pas un ingénieur du son particulièrement doué) et d'ajouter une basse et de régler le son de la guitare de façon que seul elle ne sonne pas si bien que ça mais combiner à la basse, elle donne un son complet.

Règlages de Promini-V Metal

Afin de bien utiliser cette instrument VST, il est important de bien comprendre comment il fonctionne, regardons un peu les paramètres de ce plugin.

Utilisation

Patchbox OS

Introduction

La première idée que j'ai eue lorsque j'ai reçu mon Raspberry Pi, ça a été de vouloir faire un synthé avec.
J'ai un clavier Midi USB, du coup, ça paraissait logique de pouvoir faire de la musique avec.

Mais après avoir testé plein de solutions, c'est assez laborieux, et on finit par revenir assez rapidement à utiliser un PC à la place.

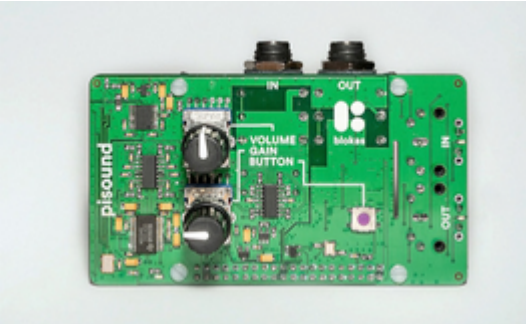
Ça , c'était avant que je découvre Patchbox OS, un OS pour Raspberry Pi pensé pour le transformer en synthétiseur mais aussi comme pédale à effet.

Patchbox OS est vraiment complet, je vais donc principalement parler de Modep que j'ai testé.

À première vue lorsque l'on regarde le site de Blokas.io ont pourrait penser que leur logiciel ne fonctionne qu'avec leur matériel, mais en vrai Patchbox OS marche très bien sans !

Pour autant pour profiter pleinement du logiciel, leur carte son le PiSound permet d'avoir une entrée et une sortie audio de qualité

https://blokas.io/pisound/



Présentation

Patchbox OS est une distribution Linux pour le Raspberry Pi destiné aux projets audio. Il est pré configuré pour avoir une faible latence et incorpore un paquet de logiciels audio.

Parfait pour fabriquer son propre synthétiseur, une pédale à effet, ou même pour créer des installations artistiques (il fonctionne super bien avec TouchOSC ou tout logiciel utilisant le protocole OSC)

Concept Modulaire

Patchbox est construit autour de module, le but des modules est de simplifier la configuration de chaque logiciel, ce qui permet de passer d’un projet à un autre sans avoir à modifier fondamentalement votre système.

Les 3 modules

Orac

Orac est système modulaire basé sur PureData crée par Mark Harris. Ce système est utilisé dans l'instrument Organelle.

https://www.youtube.com/embed/-m8p9E-WGWE

https://www.youtube.com/watch?v=-m8p9E-WGWE

PureData

Je ne connais pas bien PureData, du coup je n'ai pas vraiment réussi à l'utiliser.

Modep

Ce qui a attirait mon attention c'est plus Modep, depuis une interface web on peut configurer ses synthés et effet comme des pédales à effet de guitare.

En gros c'est une interface pour utiliser les effets et synthé **Calfs**, **Guitarix**, **fluidsynth** et pleins d'autres.

Matériels nécessaires

- Un Raspberry Pi (Minimum 2B)
- Un instrument Midi (ou TouchOSC sur Android/Iphone)
- Recommandé : Une bonne carte son USB / shield audio

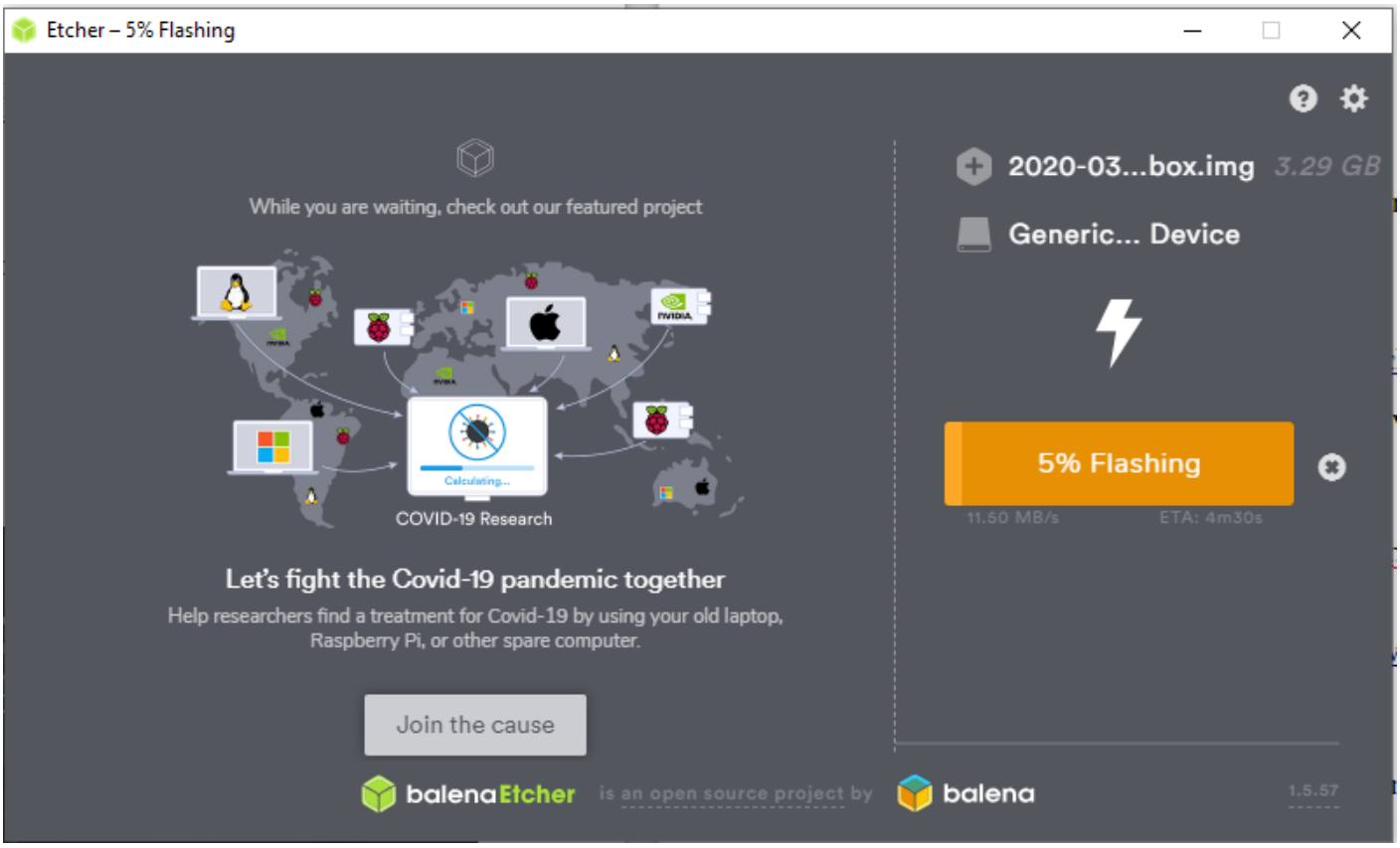
Installation

Documentation officielle : https://blokas.io/patchbox-os/docs/
L'installation est vraiment simple, on sent le travail bien fait derrière.

Flashage de la carte SD

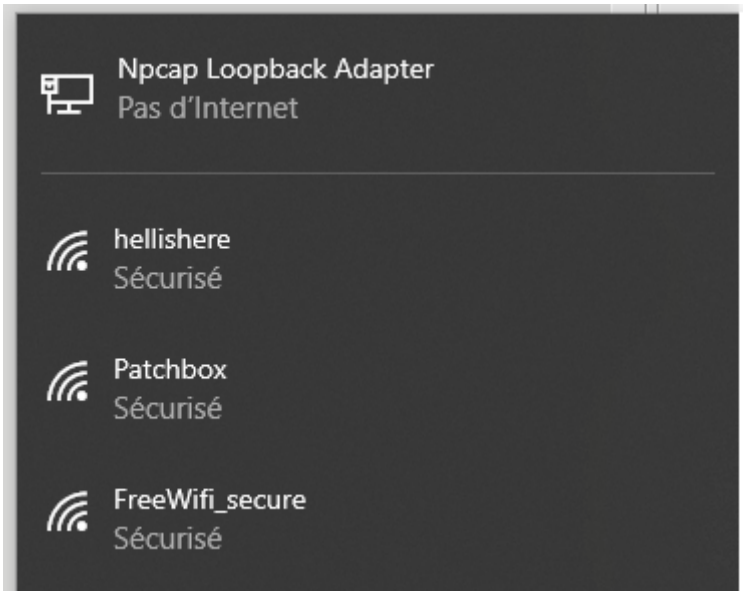
Pisound est une image disque (comme Raspbian), il va donc falloir la télécharger puis la copier sur une carte micro SD.

https://blokas.io/patchbox-os/#patchbox-os-download

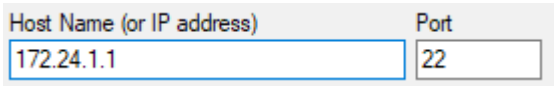


Premier démarrage

Il va falloir se connecter en SSH sur le Raspberry Pi, heureusement par défaut PatchBox OS se met en mode point d'accès Wifi (vous pouvez évidemment aussi le brancher en Ethernet)

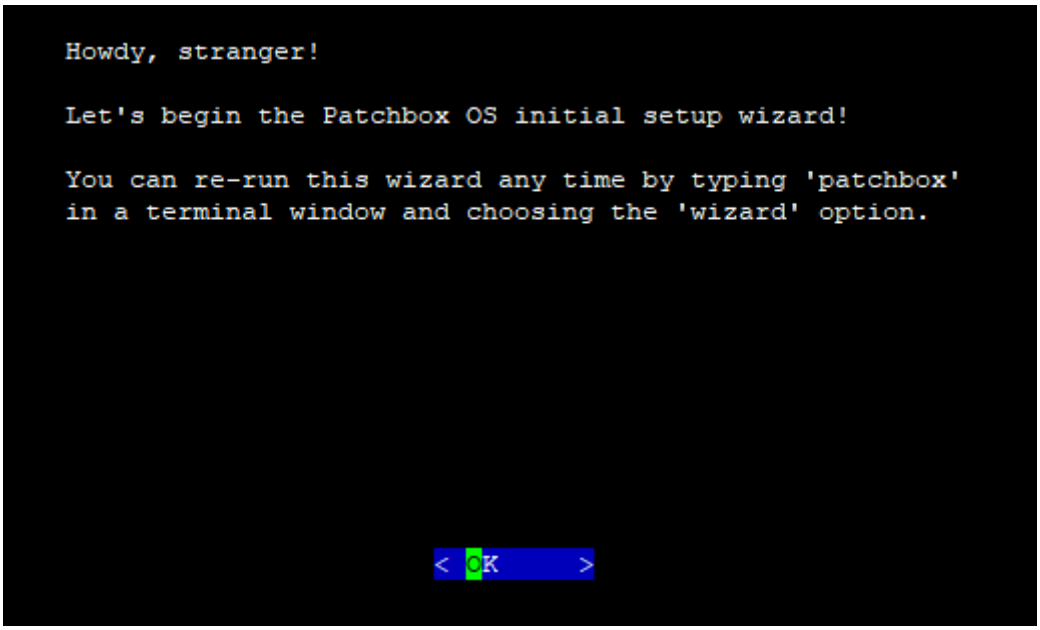


- Connectez-vous au point d'accès Wifi, le mot de passe est **blokaslabs**
- Puis connectez-vous en SSH, identifiant : **patch** mot de passe **blokaslabs**



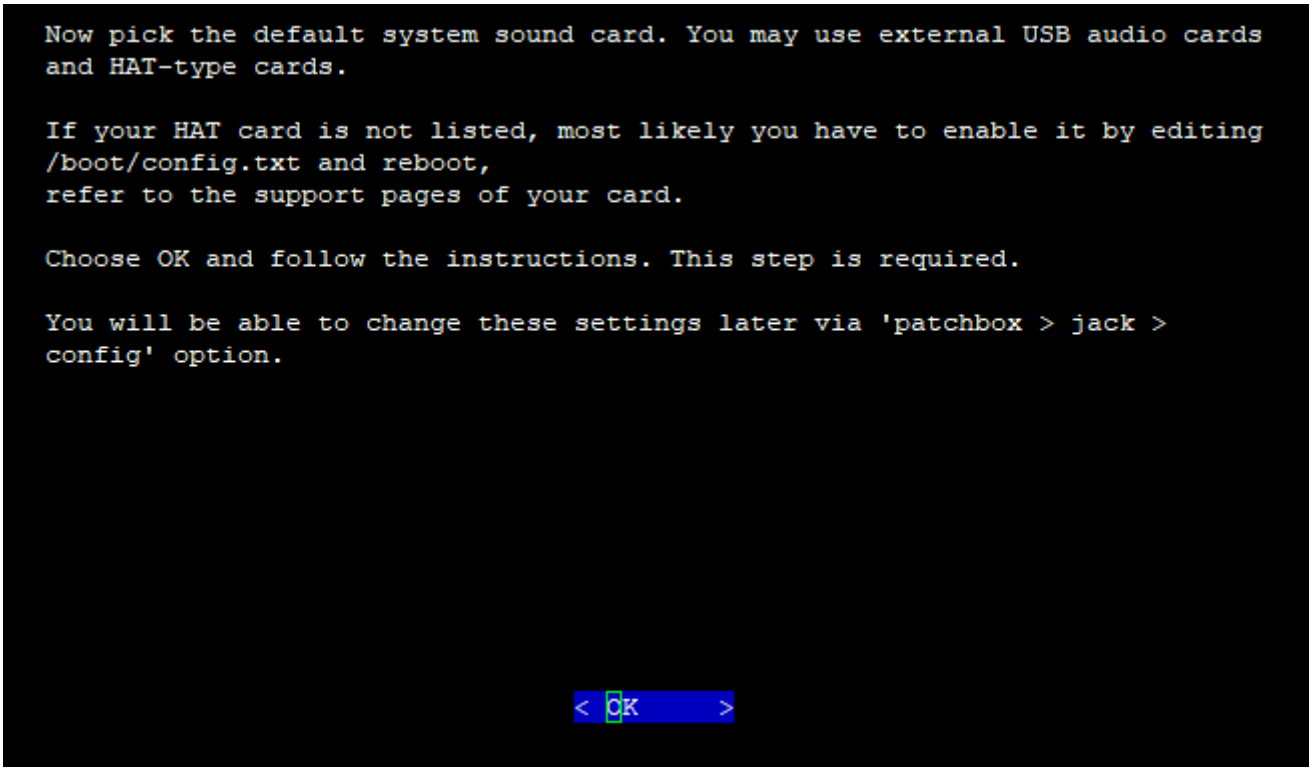
Assistant d'installation

Surprise, vous n'arrivez pas sur un terminal linux, mais sur un assistant.
Si vous avez fait une erreur pas de panique, il vous suffira de tapez **patchbox** pour le redémarrer.



- Dites non pour les mises à jour (vous n'êtes pas connecté pour le moment)
- Changer le mot de passe

Choix de la carte son



Il est maintenant temps de choisir la carte son, voici un tableau récapitulatif pour le PiSound, la sortie intégrée et une carte son USB générique.
Si vous avez paramétré une carte son, sous DOS, ça devrait vous rendre un peu nostalgique ;-)

	Pisound	Sortie intégrée (bcm2835_alsa)	Carte son USB
Sampling rate (-r)	48000	44100	44100
Buffer size (-p)	128	512	512
Period (-n)	2	3	3

Choix de l’environnement graphique

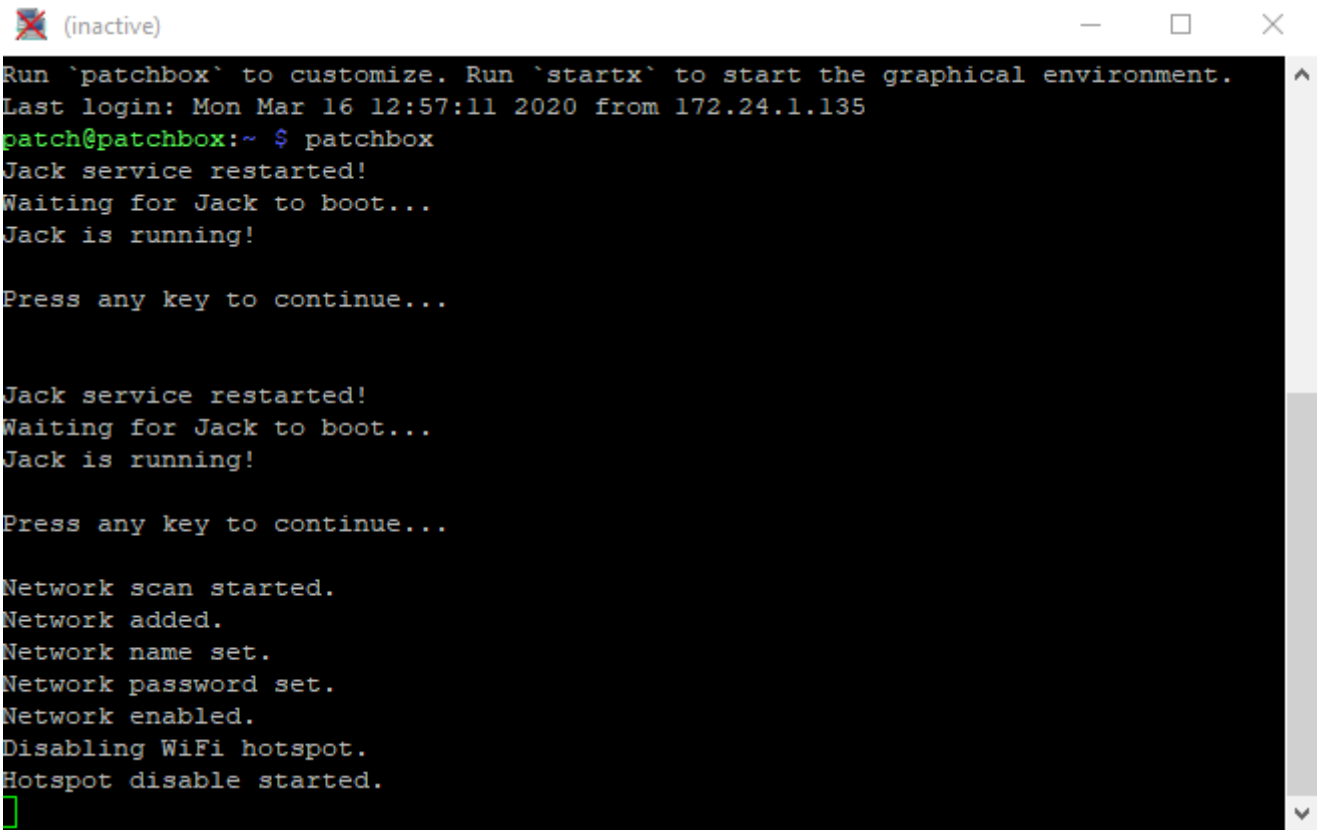
Patchbox OS est avant tout basé sur Raspbian, vous pouvez ainsi démarrer sur le bureau, en soit à moins de vouloir utiliser pure-data ce n’est pas utile, en effet nous allons configurer Modep depuis notre ordinateur.

Connexion au WiFi

Afin de pouvoir finir l’installation, il va nous falloir une connexion internet, vous perdrez la connexion actuelle et la possibilité d’utiliser patchbox OS en point d’accès WiFi (utile si vous voulez le configurer sans être chez vous)

Vous pourrez de toute façon changer ce paramétrage ultérieurement en tapant **patchbox**

Reconnexion

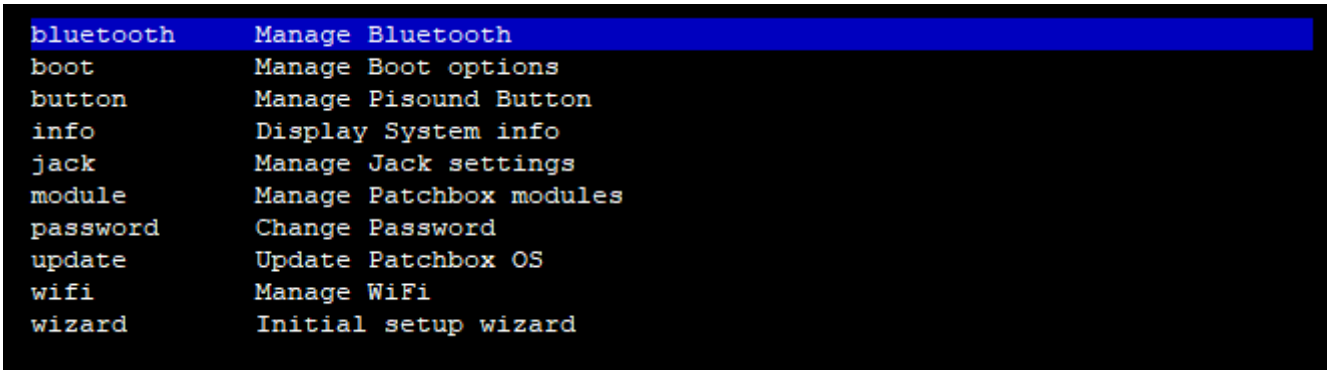


Oh non tout à planter ! Pas de panique c'est normal, votre Raspberry Pi est maintenant connecté sur votre box.
Reconnectez vous, l'avantage c'est que vous pouvez utiliser **patchbox.local** plutôt que l'IP à partir de maintenant.

(rappel : identifiant : **patch** mot de passe **blokaslabs** (si vous ne l'avez pas changé)

Tapez patchbox pour finir la configuration

- Aller sur update pour mettre à jour Patchbox
- Puis aller dans module –> modep pour l'installer (et l'activer)



Quand j'ai testé Patchbox OS, leurs serveurs étaient très lents, mais apparemment le problème est résolu, il ne reste plus qu'à attendre la fin de l'installation.

Let's rock !

J'ai eu des soucis de CPU à 100 %, je vous conseille de redémarrer avant de lancer modep.

Depuis votre navigateur web aller sur http://patchbox.local/

Montez le son !

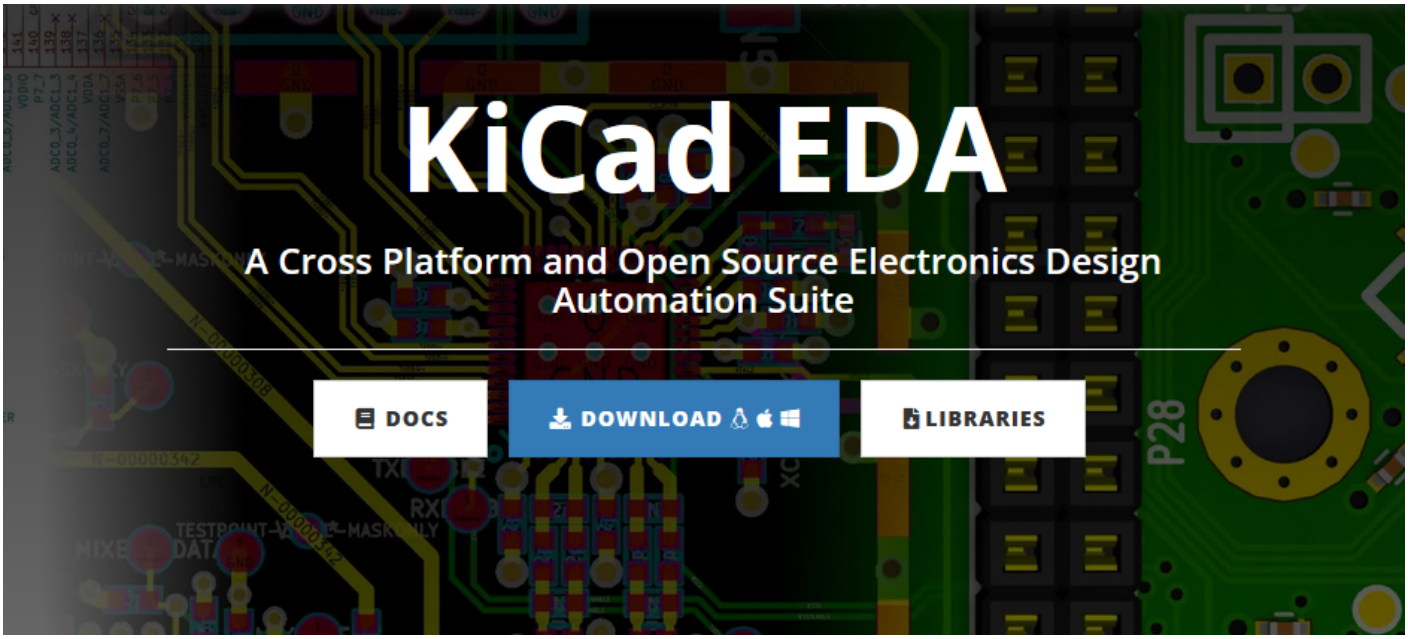
Dans la ligne de commande tapez alsamixer, F6 pour choisir la carte son.

Arduino (AY-3-8910)

Customisation (Kicad)

Schéma sur Kicad

Kicad n'est pas comme on pourrait le penser, une seule application mais un ensemble d'applications, qui communique entre eux.



https://kicad-pcb.org/

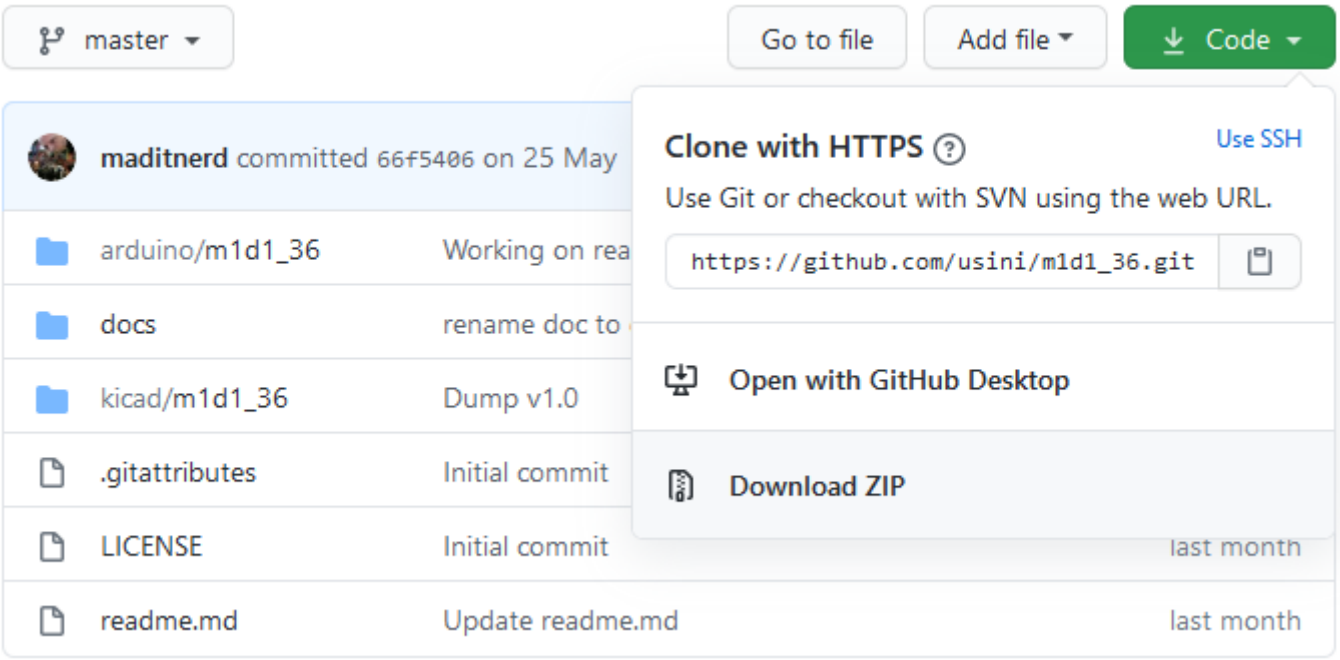
Créer un PCB se fait en 2 étapes, **le schéma** puis le **PCB**.
Le schéma **n'est pas obligatoire**, mais il nous permet d'être sûr **de notre routage**, et **d'avoir une vision clair des branchements**.

- Pour le schéma, nous utilisons le programme **EEschema** (le premier icône de la liste)
- Pour le PCB, nous utiliserons **PCBNew** (le troisième icône)

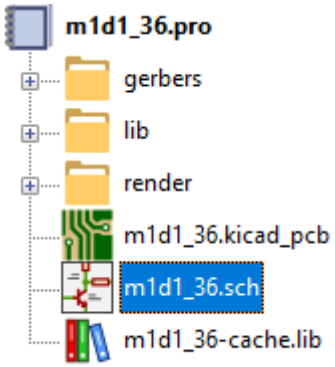
> EEschema et PCBNew est accompagné de deux autres programmes, l'éditeur de symbole et l'éditeur d'empreintes qui nous permet de créer nos propres composants.

Téléchargement

Les fichiers Kicad sont disponibles dans les sources dans le dossier Kicad, vous pouvez le télécharger ici : https://github.com/usini/m1d1_36



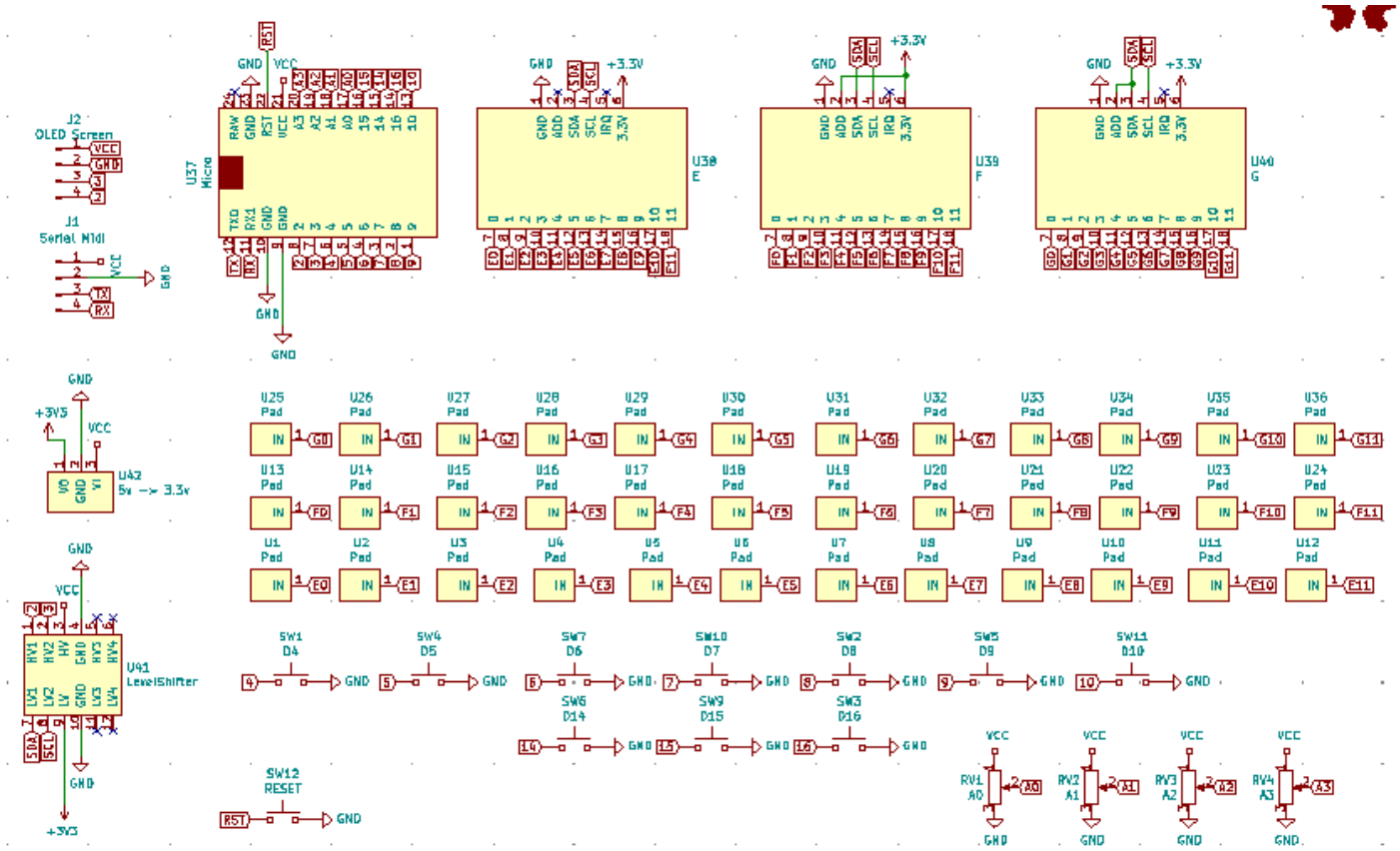
Ouvrez le fichier **m1d1_36.pro** puis ouvrez **m1d1_36.sch**



Le schéma

Voici le schéma de l'instrument, habituellement il est recommandé de router les entrées / sorties à l'aide de **fils verts**, ici vu que nous utilisons des modules tout fait, j'ai pris la liberté d'utiliser des **labels**


Si un label a le même nom, Kicad considère qu'ils sont reliés entre eux.



Changer le schéma

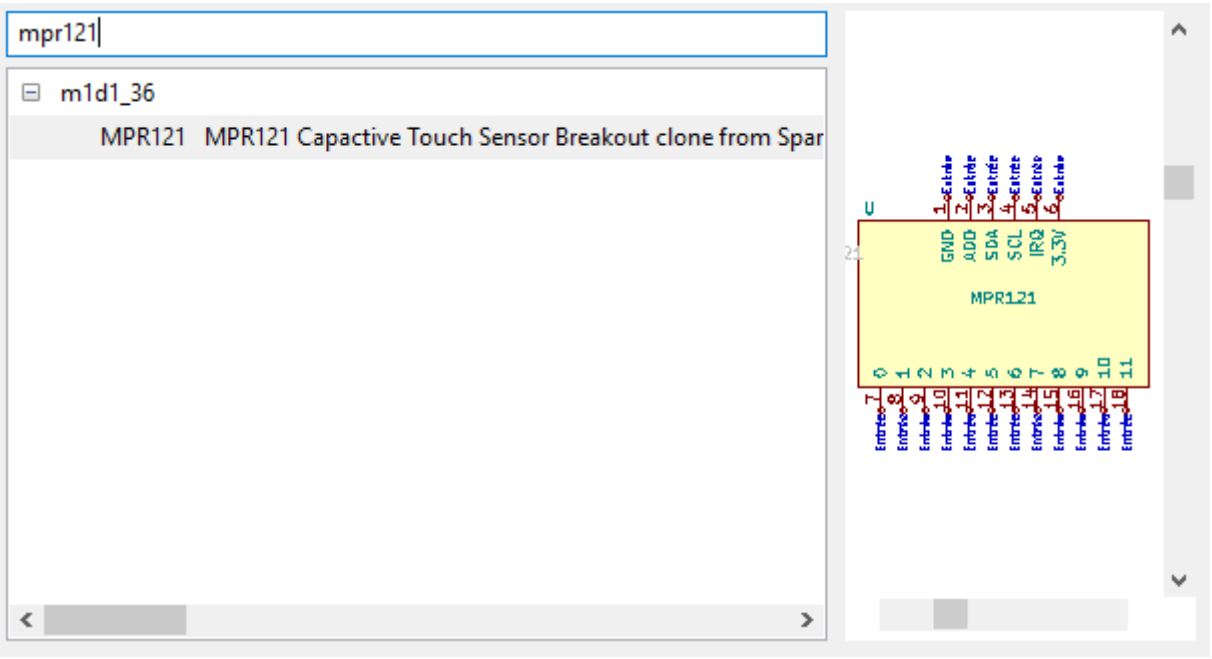
> A l'exception des boutons et potentiomètres, tout les symboles ont été créent et sont disponible dans **lib/m1d1_36.lib**

Imaginons que nous ne voulons ajouter un capteur capacitif dans notre schéma, il va falloir :

- Ajouter un MPR121
- Router le MPR121 vers le level shifter

Cliquer sur **Placer Symbole** (ou **SHIFT-A**)

Cliquer n'importe où sur le schéma
Chercher **MPR121** et placer le sur le schéma


mpr121
m1d1_36
MPR121 MPR121 Capactive Touch Sensor Breakout clone from Spar
U
GND
ADD
SDA
SCL
IRQ
3,3V
MPR121

Vue PCB sur Kicad

Modifier un PAD

Les pads que j'ai fait sont basés sur la taille d'une touche de clavier, mais il est possible d'utiliser d'autres formes.

Voici quelques exemples tirés du datasheet

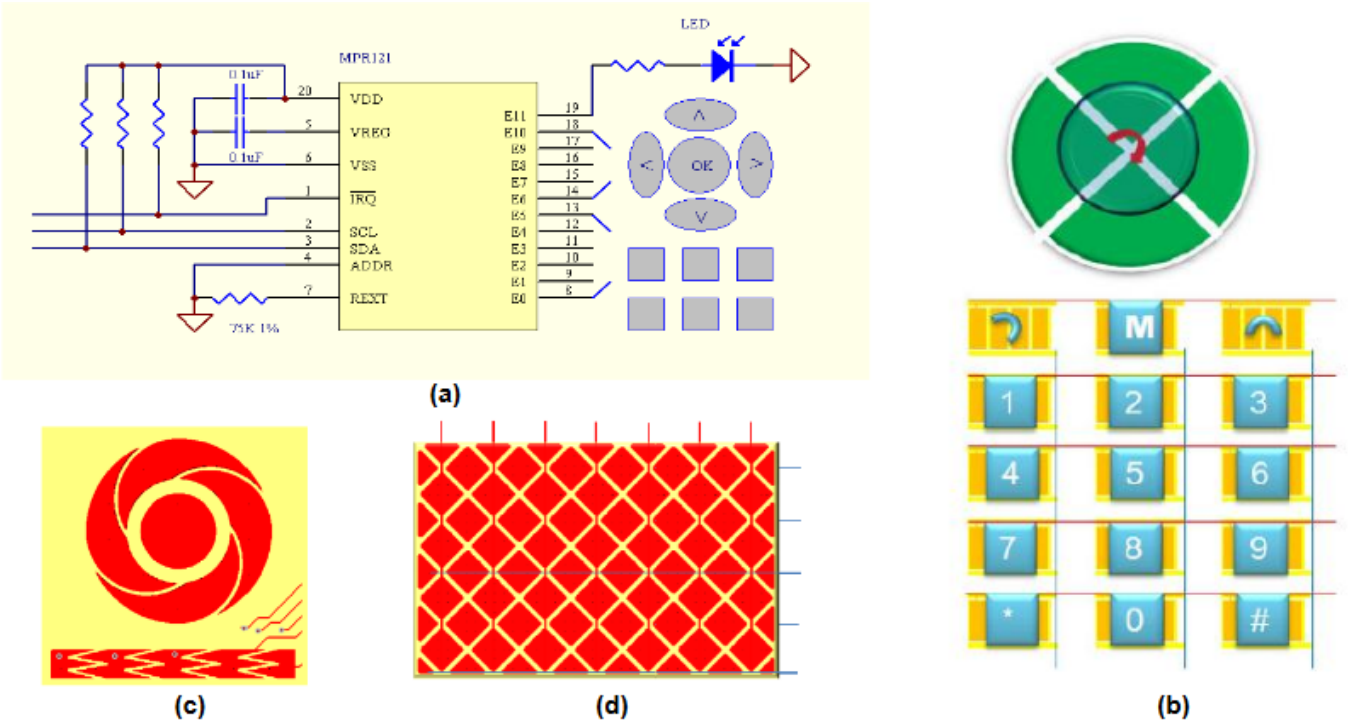


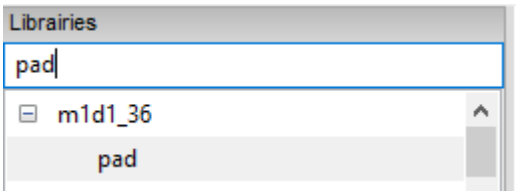
Figure 1. (a) Typical application circuit
(b) Button matrix pattern using 12 channels for 20 touch buttons
(c) Slide wheel and slide bar pattern using 10 channels
(d) Touchpad 5x7 pattern using 12 channels

Aller dans l'éditeur d'empreinte

Afin de modifier la forme d'un Pad, il nous suffit de changer l'empreinte (footprint) de celui-ci.

Dans Kicad aller dans **l'éditeur d'empreinte**

Dans Librairies, cherchez **pad**

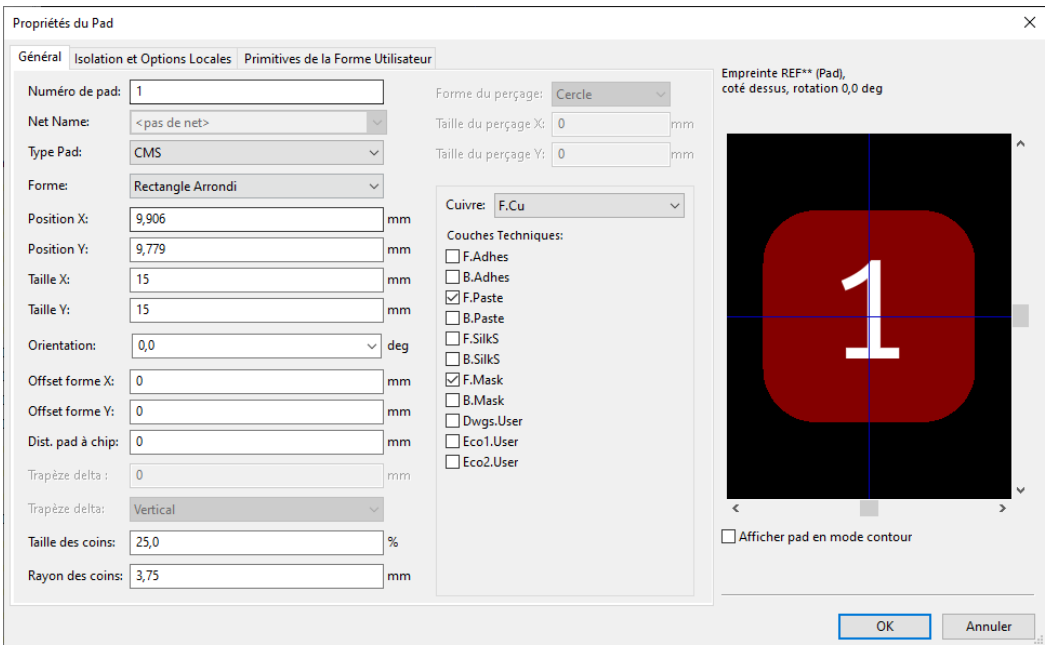


> Vous pouvez avoir un visuel en 3D du résultat en allant dans **Affichage** --> **3D visualisateur**

Modifier les propriétés du PAD

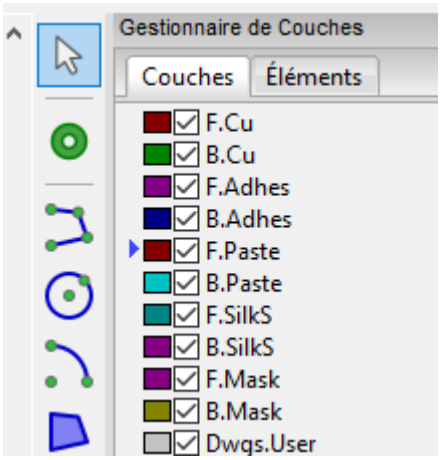
Vous pouvez modifier la forme en **double cliquant** simplement dessus, cela vous permet de faire:

- Rectangle
- Rectangle arrondi
- Ovale
- Cercle
- Trapézoïdale (Triangle)

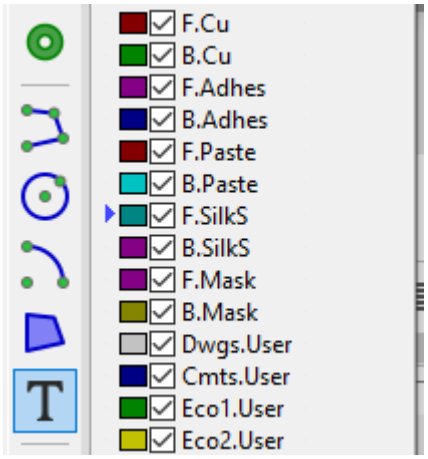


Modification Avancés

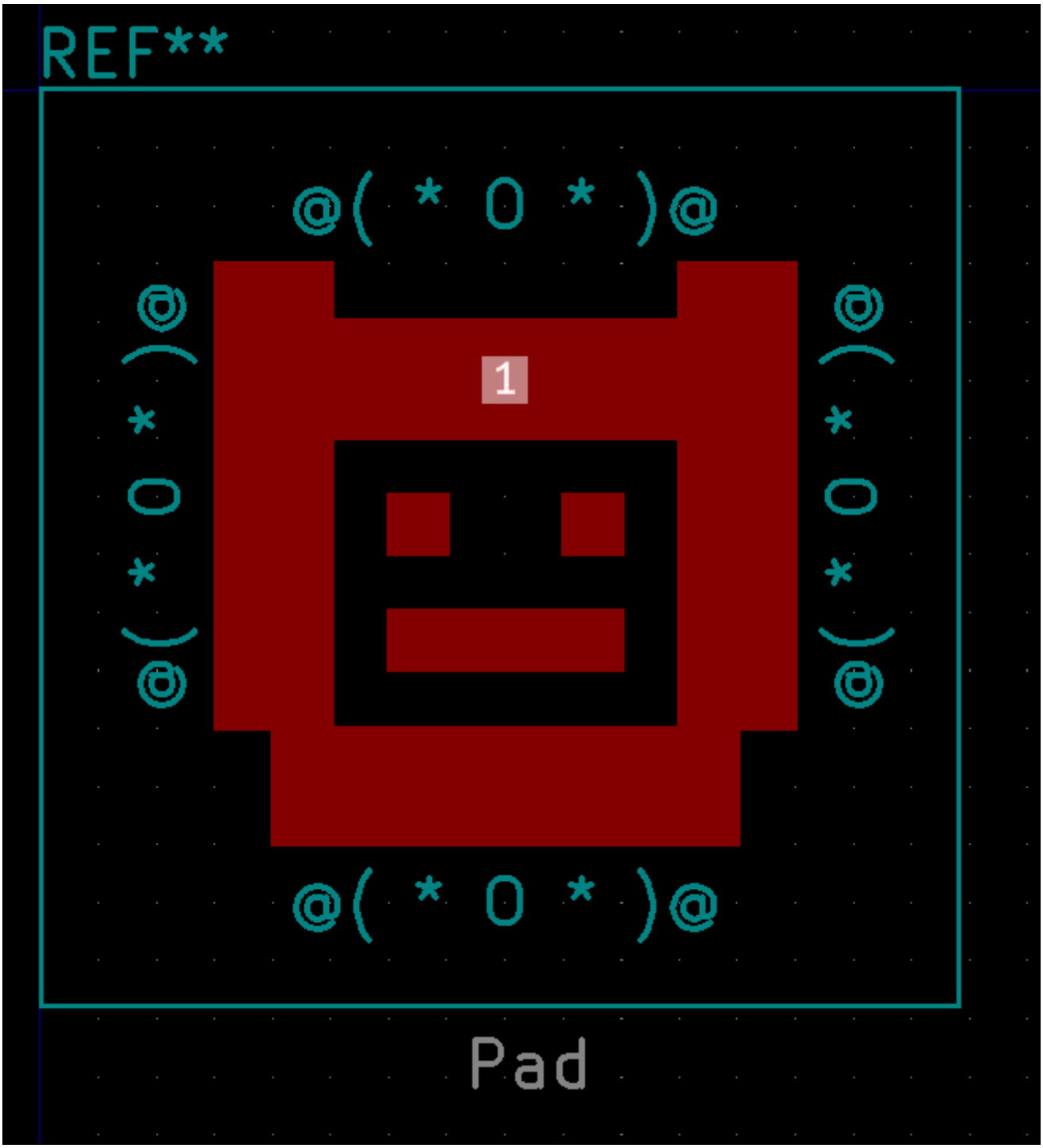
Si vous cherchez à faire des formes plus complexe, c'est possible à l'aide des outils de dessin.
Sélectionner **F.PASTE**



Vous pouvez aussi ajouter du texte avec **F.SilkS** (Sérigraphie)



> N'oublier pas de garder le Pad de référence et qu'il soit en contact avec les zones de soudure ajoutés

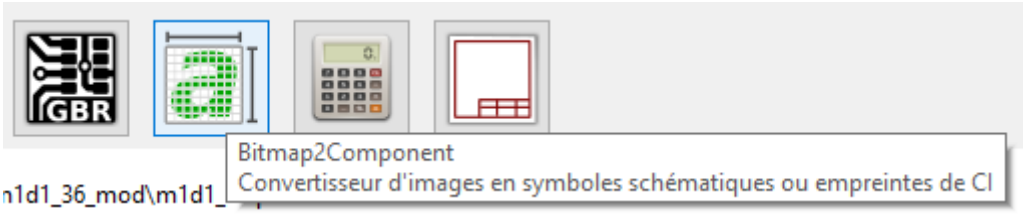


Modification à partir d'une image

Il est aussi possible d'utiliser une image, il est par contre **un peu compliqué de la dimensionner correctement.**

- Il vous faut donc **connaître la taille du pad** que vous voulez créer.
- Ainsi qu'une **image monochrome**, pour notre exemple nous allons utiliser le logo de Fear Factory.

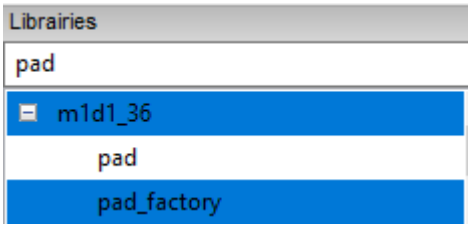
- Cliquer sur le **Convertisseur d'images en symboles schématiques ou empreintes de CI** (Bitmap2Component)



- Charger une image Bitmap
- Dans Format cliquez sur Pcbnew

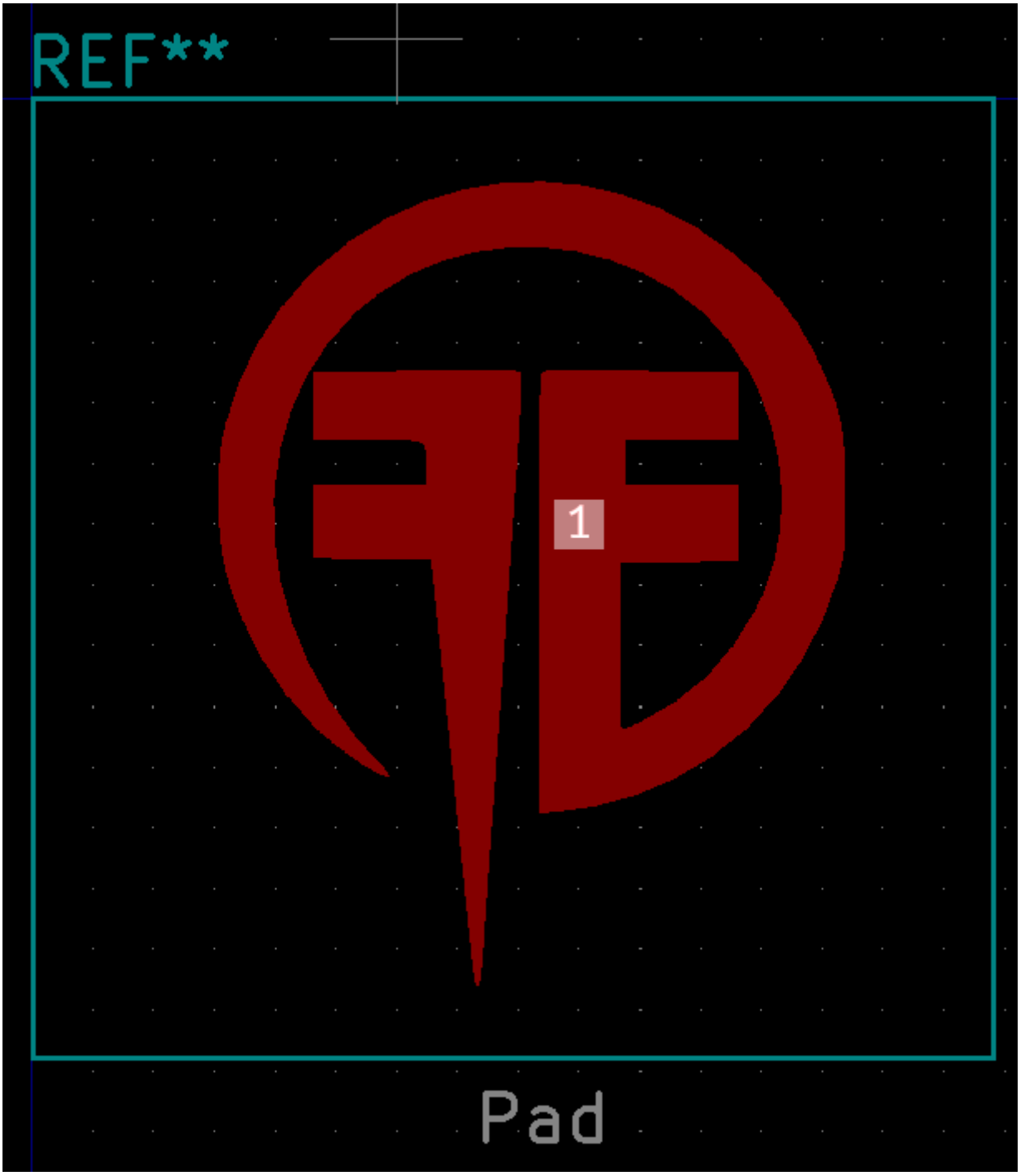
Le seul paramètre pour dimensionner notre image est la résolution, plus **celle-ci est elevé, plus l'image est petite.**

- Modifier la **résolution DPI** jusqu'à obtenir la bonne taille.
- Cliquer sur exporter et sauvegarder le fichier dans **lib/m1d1_36.pretty/**
- **Dans** l'éditeur d'empreinte ouvrez le fichier que vous avez crée (ici pad_factory)



- Copier la forme (CTRL-C)
- Sélectionner un **point de référence pour la copie** (par exemple **un point en haut à gauche**)
- Ouvrez **Pad** et coller la forme
- Double-cliquez sur la forme et changer la couche en **F.PASTE**

- N'oubliez pas de garder le pad d'origine et de faire qu'il soit en contact avec la forme

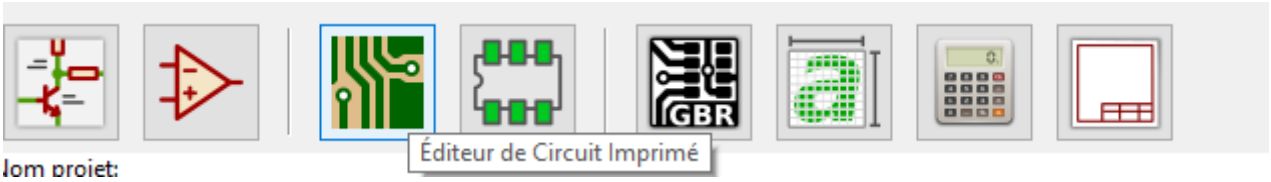


Ré-association de l'empreinte

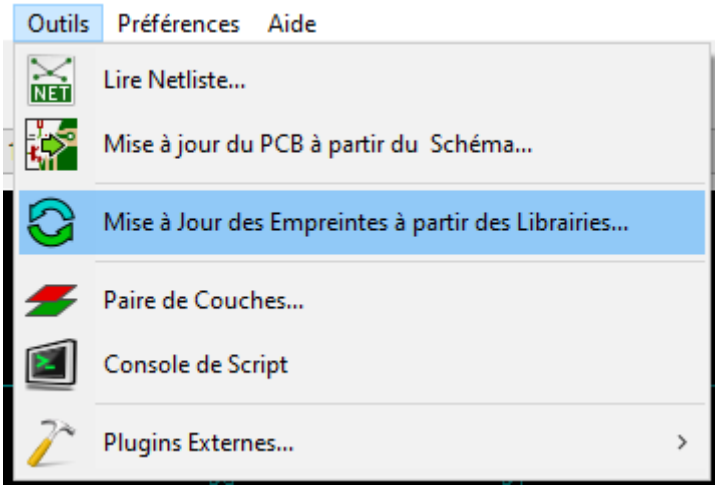
Notre empreinte à été modifié, pour information, les empreintes se trouvent dans **/lib/m1d1_36.pretty** comme vous pouvez le voir dans **Préférences / Configurer les librairies d'empreintes / Librairies Spécifiques au Projet**

Il nous faut maintenant mettre à jour l'empreinte dans notre circuit

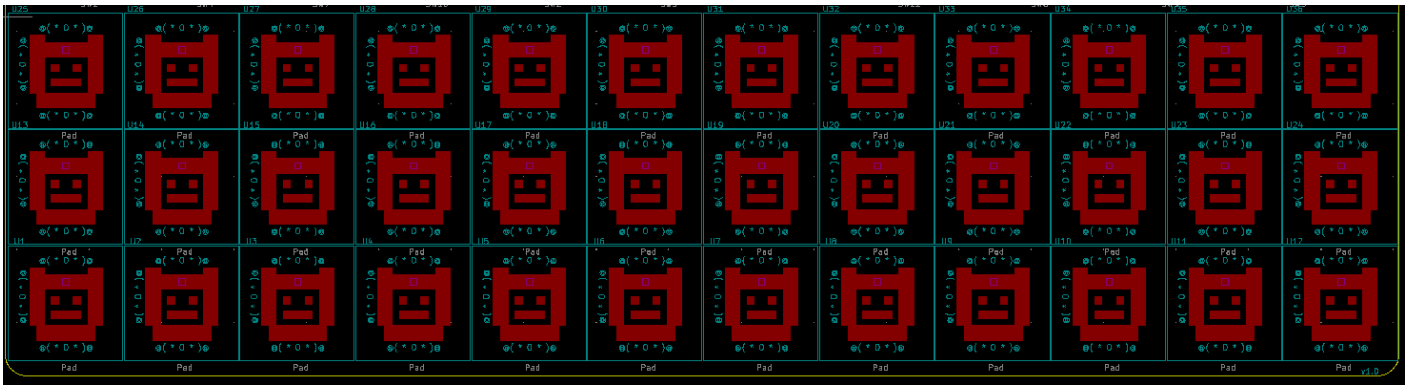
- Sortez de **l'éditeur d'empreinte** (après avoir sauvegarder)
- Aller dans **l'éditeur de circuit**



- Aller dans **Outils** / **Mise à jour des empreintes à partir des librairies** et cliquez **Mise à jour toutes les empreintes du PCB**



Et voilà nous avons toutes nos empreintes, ici nous n'avons pas besoin de repenser la carte ou déplacer les pads donc cela reste relativement simple.



Toutefois, il nous faut refaire les pistes, pas de panique nous allons utiliser un outil d'autorouting afin de simplifier cette tâche.

Autoroutage avec FreeRouting

Bien que router des pistes manuellement sur Kicad soit un plaisir pour moi, ici vu que nous avons 36 pads, cela risque d'être très long. heureusement il est possible de le faire automatiquement avec freerouting.

Installer Java

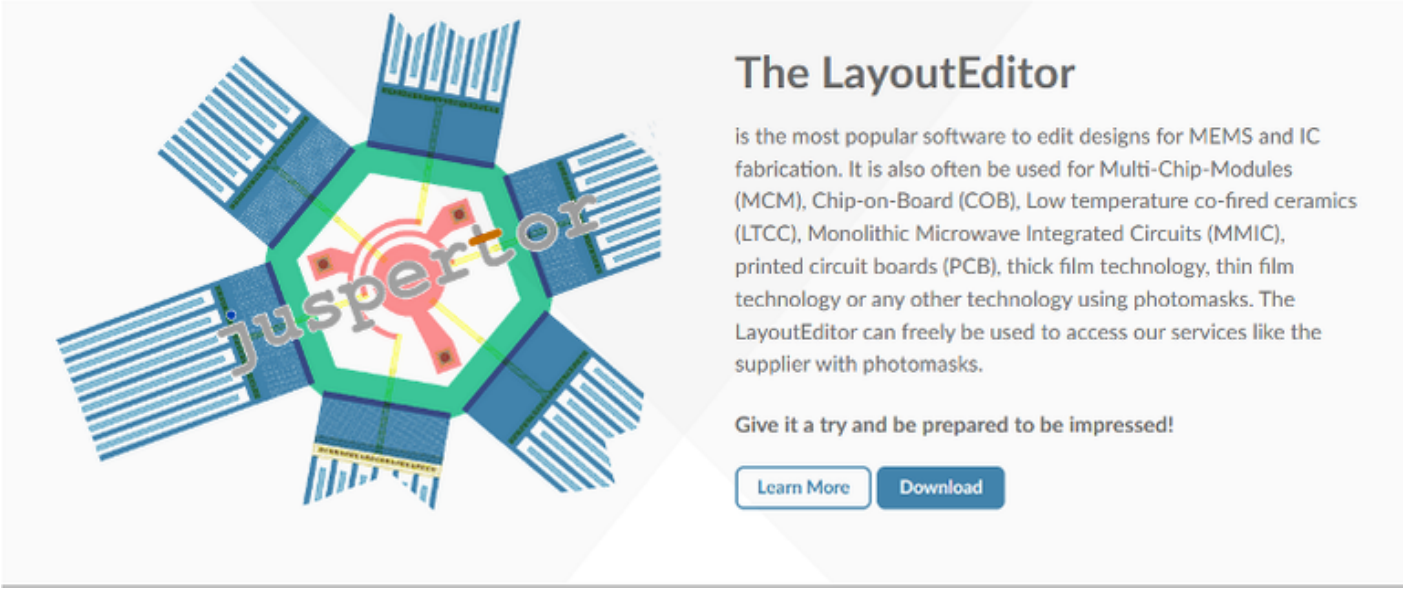
Freerouting utilise Java, il va donc nous falloir l'installer.

https://www.java.com/fr/

Installer Layout Editor

Pour une raison assez bizarre, freerouting n'est pas disponible seul, il faut installer **Layout Editor** pour récupérer freerouting.

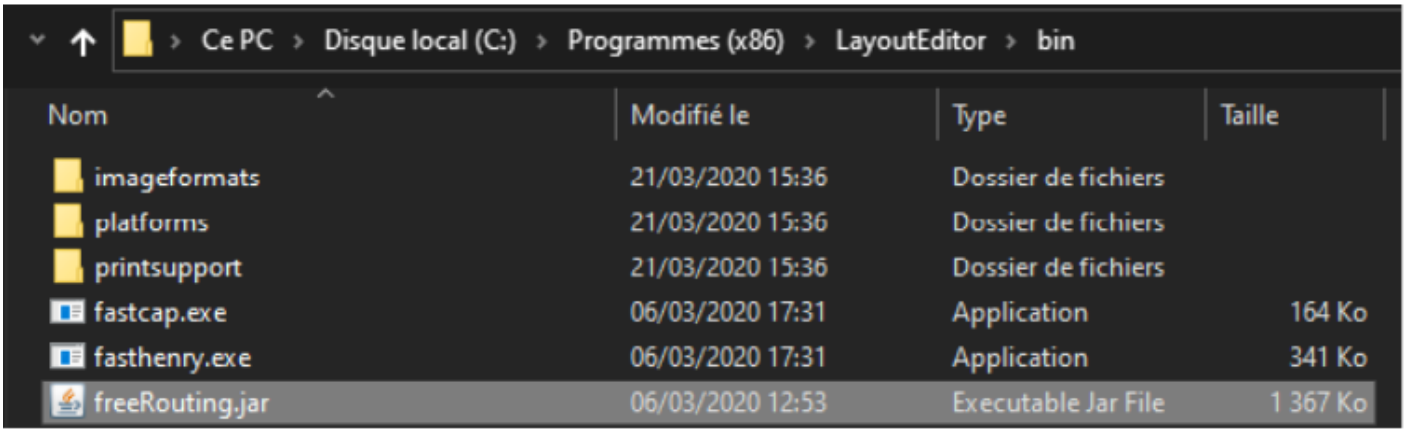
https://layouteditor.com/



Créer un raccourci vers freerouting

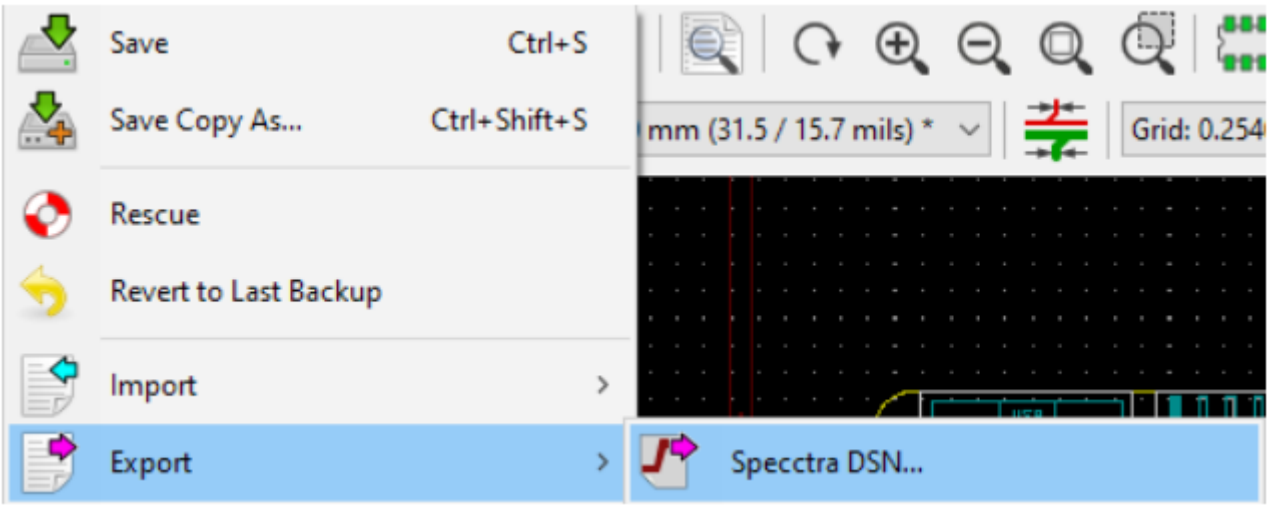
Freerouting est disponible dans le dossier bin de LayoutEditor, si vous êtes sous Windows il devrait être dans **"C:\Program Files (x86)\LayoutEditor\bin\freeRouting.jar"**

Créer un raccourci sur votre bureau vers ce fichier.



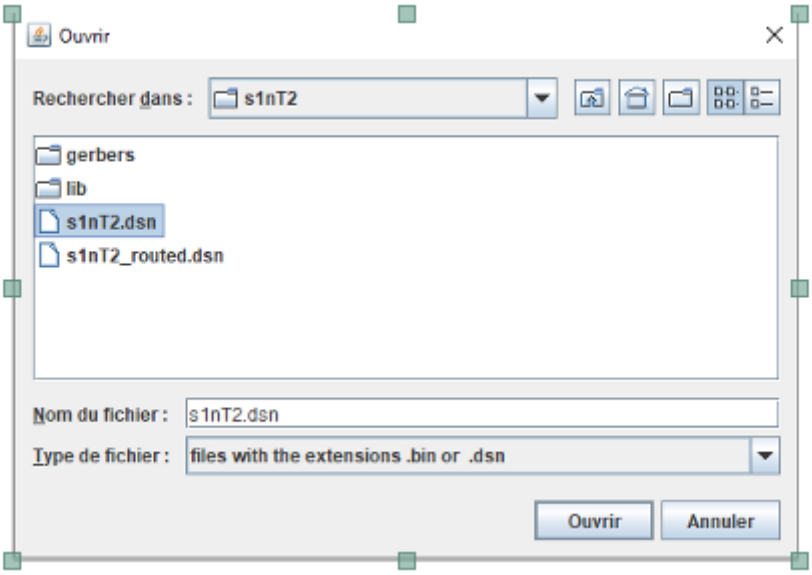
Créer un fichier Specctra DSN

Dans **Kicad / PCBNew,** exporter votre carte au format DSN



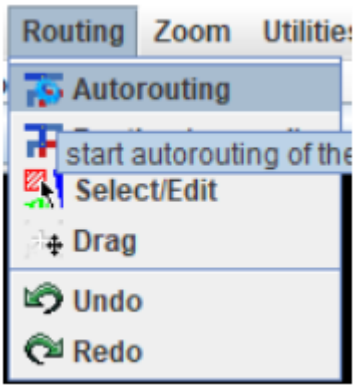
Importer le fichier Specctra DSN

Ouvrez Freerouting, puis ouvrer le fichier Specctra DSN



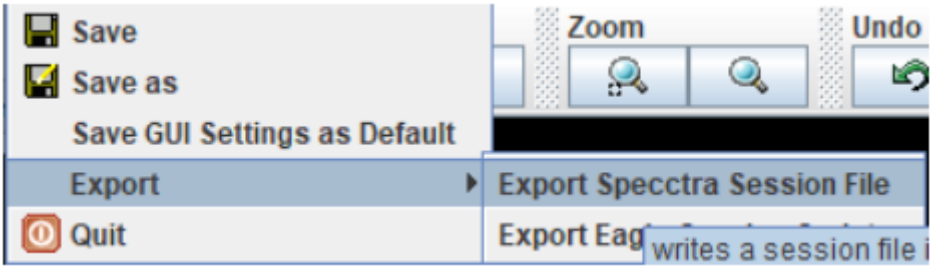
Lancer l’autorouting

Vous n’avez plus qu’à lancer l’autorouting, cela va prendre un petit moment, vu que le logiciel va faire plusieurs passes avant de décider du meilleur routage.

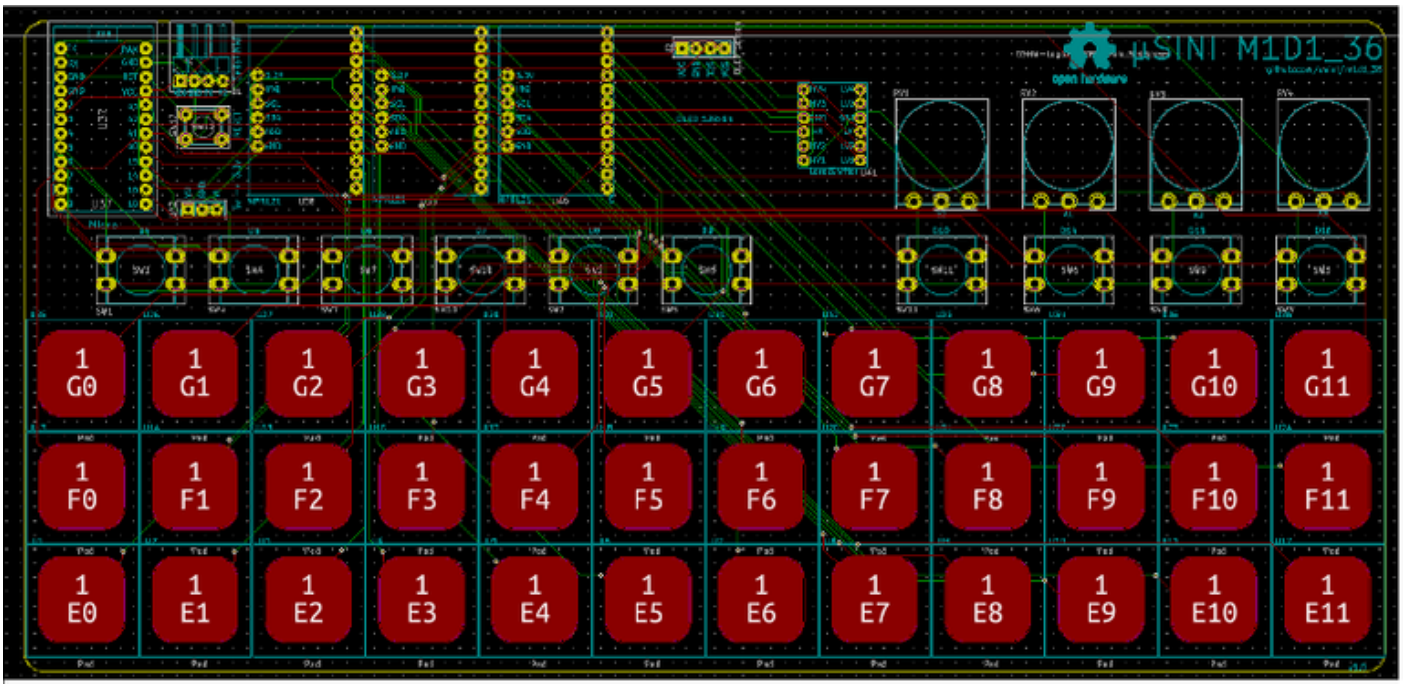


Exporter le fichier Specctra

Une fois fini, il ne nous reste plus qu'à **exporter le fichier Specctra** puis le **réimporter dans Kicad**



Et voilà votre routage est fait, c'est un sacré gain de temps ! Toutefois ne faites pas confiance aveuglement au routage ! N'oubliez pas de revérifier les pistes.



Bonus

J'avais filmé une partie la fabrication de cette carte, ce n'est pas très passionnant à regarder mais ça peut être utile

https://youtu.be/buCY1lT0xS4

https://www.youtube.com/embed/buCY1lT0xS4

Customisation (Arduino)

Code Principale

> Documentation développeur : https://usini.github.io/m1d1_36/docs/index.html

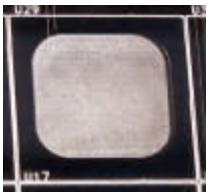
Afin de simplifier la modification du fonctionnement de l'instrument, tout le code est basé sur un système événementielle
Chaque fonctionnalité est rangé dans un fichier.h

- OLED.h (Afficheur)
- buttons.h (Boutons)
- midi.h (Midi)
- mpr121.h (Module capacitif)
- pots.h (Potentiomètre)
- settings.h (Paramètres)

Voici les 3 événements disponibles dans ces 3 fonctions

- MPREvent
- potsEvent
- buttonsEvent

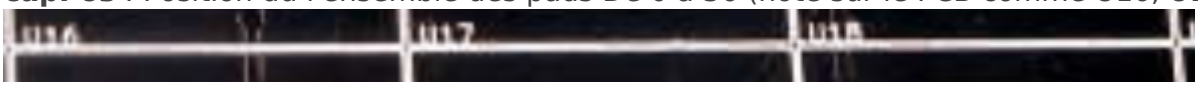
MPREvent (PAD)



Lorsqu'un pad est appuyé ou relâché, cette fonction devient active.

Voici les paramètres de cette fonction:

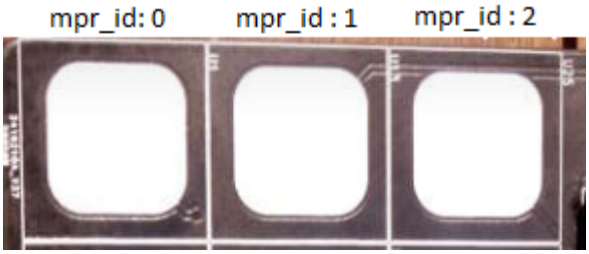
capPos : Position du l'ensemble des pads De 0 à 36 (noté sur le PCB comme U16, U17, U18 par ex)



note : La note correspondant à la position du pad comme noté dans settings.h (voir capNote[36])

```
uint8_t capNote[36] = {28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
                       40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
                       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63
                      }; ///< Note from pad (when transpose = 0)
```

state : **true** si appuyé et **false** si relaché
mpr_id : id du module capacitif, chaque ligne correspond à un module et commence par la ligne du bas



cap_id : id du pad par rapport au module (de 0 à 12)

Exemple, jouer une note du tableau capNote[36]

```
if (state) { //Pad appuyé
 noteOn(note, 120, 1) //Note midion (équivalent à capNote[capPos]), velocité, canal)
} else { // Pad Relaché
 noteOff(note, 0, 1) //Note midioff (équivalent à capNote[capPos]), velocité, canal)
}
```

potsEvent (Potentiomètre)

Lorsqu'un potentiomètre est tourné, cette fonction s'active

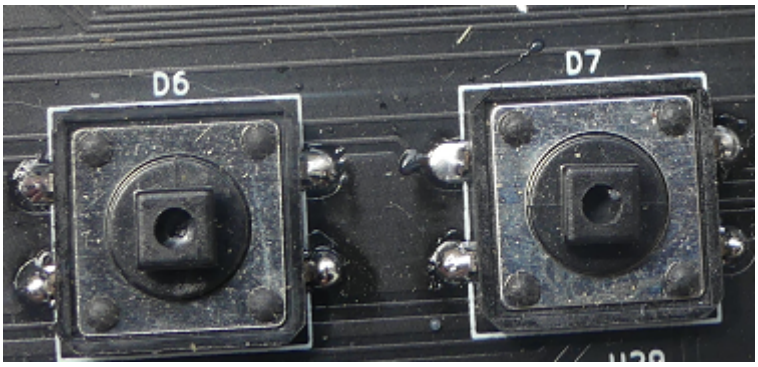
Voici les paramètres de cette fonction:
pot_id : Le numéro correspondant au potentiomètre **(noté A0 à A3 sur le PCB)**
pots_cc[] : Le paramètre lié au potentiomètre **sous la forme de 0x01 à 0x7F (voir settings.h)**
pots_val[] : La valeur du potentiomètre de 0 à 127

> pots_cc et pots_val sont des variables globales

Exemple

```
switch (pot_id) {
    case 0:
      controlChange(0x01, pots_val[pot_id], 1);
      break;
    case 1:
      controlChange(0x02, pots_val[pot_id], 1);
      break;
    case 2:
      controlChange(0x03, pots_val[pot_id], 1);
      break;
    case 3:
      controlChange(0x04, pots_val[pot_id], 1);
      break;
}
```

buttonsEvent (Boutons)



Lorsqu'un bouton est **appuyé**, **relaché**, **appuyé deux fois**, **laissé appuyer** et **appuyé plusieurs fois**

Pour gérer les boutons, je suis passé par la bibliothèque AceButton qui permet de facilement pouvoir gérer plusieurs évènements.

Voici les paramètres de cette fonction:
button : Objet bouton, ici nous ne l'utilisons uniquement pour récupérez son identifiant
button->getid() : Identifiant du bouton
eventType : L'évènement qui a déclenché cette fonction
buttonState : État numérique du bouton (LOW/HIGH)
btn[] : État sauvegardé du bouton

btn_cc[] : Le paramètre lié au potentiomètre **sous la forme de 0x01 à 0x7F (voir settings.h)**

>

> btn[] et btn_cc sont des variables globales

Exemple

```
uint8_t id = button->getId();
switch (eventType) {
case AceButton::kEventPressed:
  btn[id] = !btn[id]; //Inversement de l'état du bouton
  if(btn_id){
    controlChange(btn_cc[id], 0, 1);
  } else {
    controlChange(btn_cc[id], 127,1);
  }
  break;
}
```

Afficheur (OLED.h)

Le fichier **OLED.h** gère l'afficheur OLED : **SSD1306**

Nous allons utiliser pour cela, la bibliothèque d'Adafruit SSD1306.

Celle-ci utilise une bibliothèque générique pour les afficheurs : Adafruit GFX

Initialisation de l'afficheur

L'afficheur s'in

Boutons (buttons.h)

Midi USB et série (midi.h)

Pads (mpr121.h)