Module OR1

Version 1.0

Patrice Freydiere

21 janv., 2024

Contents

Récupérez la documentation au format PDF : `OR1.pdf`.

CHAPTER 1

Introduction

Vous trouverez ici, une documentation que l'on souhaite exhaustive sur le module OR1 et sa partie logicielle. Cette documentation couvre :

- la mise en oeuvre (démarrage avec le logiciel et la plateforme)
- la documentation technique. (pour des informations plus précises)
- la documentation des éléments logiciels (une documentation de référence pour les fonctions logicielles)

Espérons que ce module et cette documentation aidera à construire des panoplies d'instruments et boites musicales invitant à chanter, ou mettre une ambiance sonore agréable dans les maisons, boutiques ,rues , festivals et spectacles.

Liste des ressources associées à cette documentation :

- Le site est accessible en ligne, dans sa dernière version à cette adresse : https://github.com/frett27/or1-documentation
- Le logiciel encrypter, permettant l'encryption des fichiers pour le module : https://github.com/frett27/encrypter

Note: Le logiciel utilisé dans le module OR1 n'est pas directement associé au matériel présenté. D'autres utilisations sont également possible ,le logiciel mis en place est un synthétiseur embarqué basé sur des calculs d'entiers, et spécialisé dans le jeu de fichier midi, kar, book sur une base de sons enregistrés. Sa particularité est de pouvoir être utilisé sur

des cartes économes en énergie, et propose des accès simplifiés à des écrans tactiles, et autres éléments (controlleurs rotatifs (manivelles)).

CHAPTER 2

Présentation du module OR1

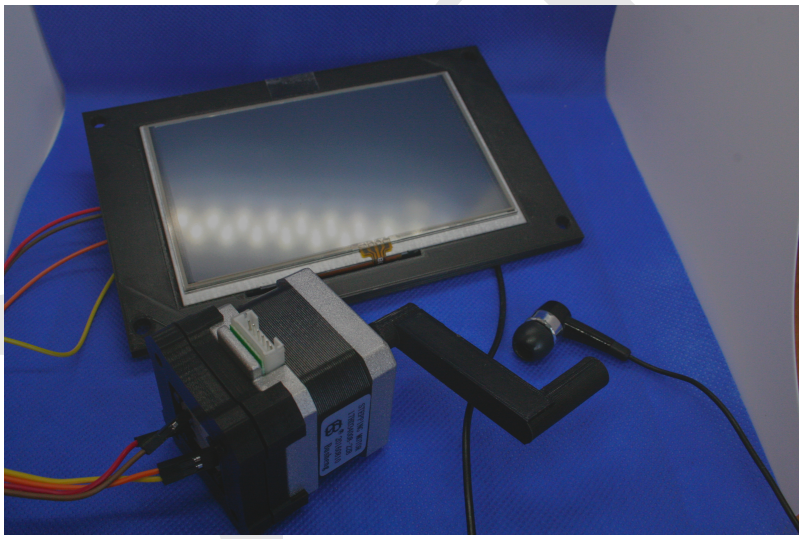


Figure1: Ensemble OR1, écran + carte électronique pour la mise en place d'instruments, pilotable par manivelle. Intégrable dans un boîtier de type Orgue, Serinette, ou Boîte à musique.

2.1 Fonctionnalités du module

Les fonctionnalités du module sont décrites ci dessous, pour la version 1 du module.

2.1.1 Lecture de fichiers interactive

- Lecture numérique de fichiers « Midi », « Kar » ou « Book » stockés sur une clef USB, (les deux formats sont proposés)
- Lecture numérique de fichiers cryptés « BookX », « MidiX » stockés sur une clef USB, fournis par les arrangeurs
- Affichage des paroles sur 4 lignes, pendant le jeu.
- Vitesse de jeu du fichier pilotée par la rotation de la manivelle, ou selection d'un tempo (jeu automatique, sans manivelle)
- Transposition par demi-tons (positifs ou négatif)
- Choix d'instrument dans une bibliothèque prédéfinie par l'utilisateur (Anche, Flute, . . .)
- Restitution sonore paramétrable (par fichiers, pour la personnalité de l'instrument) :
- possibilité de changement d'instrument, ou ajouter des registres en cours de jeu
- Suivi dans la progression du jeu. (l'utilisateur a les informations de chronométrage du jeu et de la durée du morceau de musique en cours).

2.1.2 Système de protection de fichiers arrangés ou achetés (DRM)

En complément de l'utilisation des fichiers au format MIDI, KAR ou BOOK, utilisables directement sur le module.

Ce module intègre également l'utilisation de fichiers cryptés fournis par un tier (arrangeur, auteurs). Les format d'origine, sont encryptés avec un suffix « X » (MidiX, KarX, BookX) et ne seront utilisable que sur un module donné. Ce format de fichier permettant aux arrangeurs de proposer leur travail, utilisable directement sur le module, sans donner accès au fichier d'origine.

Note: le cryptage utilisé intègre un cryptage asymétrique RSA, ce cryptage est utilisé actuellement dans les communications cryptées sur internet et également par le système bancaire.

Plus d'information dans la section « cryptage » *Encryption des oeuvres digitales*

2.1.3 Sons paramétrables

Conception et paramétrages d'instruments pour le module, en utilisant :

- des échantillons sonores au format WAV (16 bits / Stereo / 44,1Khz)
- des banques de sons utilisables au format sf2 (via outils) (version logicielle 1.1 du module)

plus d'informations dans la section *Paramétrage des sons et création d'instruments*

2.1.4 Kit de développement du module

Pour les utilisateurs ayant des compétences de développement, ou en informatique, le module peut être livré avec des outillages de développement d'applications, permettant d'ajuster et modifier les écrans et les fonctionnalités associées.

Fonctions avancées (disponibilité d'outils de personnalisation avancée, pour paramétrer les diverses actions et fonctions offertes à l'utilisateur)

- un exemple complet d'application de démonstration est fourni avec le module (pour permettre, aux intégrateurs de comprendre et construire leurs écrans).
- la possibilité de changer la banque sonore en fonction des l'effet désiré.
- utilisation du langage Python (micropython) pour la personnalisation.

Conception des fonctions et écrans en utilisant le langage python

Le module propose un interpréteur python intégrant les fonctions nécessaire à la création d'écran, de pilotage, et catalogage.

Le toolkit intègre également une bibliothèque graphique reconnue, permettant la création d'écran tactiles, permettant l'adaptation, ou la création d'une nouvelle application.

Cette capacité est proposée dans la version SDK / Facteur Orgue, et accompagnée des différents outils nécessaires pour la mise en place de programmes sur la plateforme.

2.2 Applications possibles du module

- Instrument mécanique électronique
- Pupitre de pilotage d'un instrument midifié

2.2.1 La Tourniquette

Note: Instrument crée en collaboration avec Gérard et Ludwig (le turlutain), ce nouvel instrument électronique permet de jouer des morceaux de music au rythme de la manivelle avec une grande interaction de jeu.

Votre prochain petit orgue de Barbarie électronique ?

Nous avons le plaisir de vous annoncer la sortie d'un nouvel appareil léger avec des fonctionnalités particulières, plus de cartons ou de rouleaux mais des fichiers sur une clé USB qui permettent de l'emporter partout dans un sac !

L'appareil destiné surtout à l'accompagnement du chant est doté d'un écran capacitif sur lequel vous pouvez rechercher le titre du morceau dans votre liste. Il s'affiche en entier avec la durée de défilement.

Vous choisissez sur l'écran entre un son très réaliste d'anches ou de flûtes avec réglage du volume. Pour que le morceau soit dans votre ton vous pouvez le transposer à votre guise. La manivelle répond instantanément à votre rythme, mais vous pouvez aussi la déconnecter.

Et cerise sur le gâteau les paroles s'affichent en temps réel sur l'écran.

Vous pourrez récupérer votre répertoire carton ou papier toutes gammes en fonction des accords avec les noteurs et vous procurer aussi des fichiers nouveaux auprès d'eux. Pour vos propres compositions et arrangements il y a possibilité de les transférer simplement en midi sur 68 notes chromatiques. 25 morceaux sont fournis au départ. Selon la version possibilités d'entrées micros, carte SD ou clé USB, ou Bluetooth sur l'amplificateur.

Une sangle de transport vous permet de le porter tout en jouant. Poids 6kg ou 7kg environ selon la version avec deux batteries rechargeables comme un téléphone. Pied de maintien pliant et amovible en option. Dimensions : 40 cm de large 31 cm de haut et 24 cm de profondeur. Diverses présentations : couvercle en bois massif : Hêtre, Merisier ou Noyer. Caisse en contreplaqué tout Hêtre ou Peuplier. Teintes Chêne doré, gris, Wengé, naturel. Toile de diverses couleurs au choix. Housse et charrette légère sont prévus.

Note: Et le prix ?

1900 ou 2400 euros selon la version. Contact : 03 83 48 17 82

LE TURLUTAIN Gérard DABONOT 104 Rue d'Alsace 54110 Sommerviller leturlutain@gmail.com Modèle et nom déposés

---

2.3 Utiliser le module pour la 1ère fois

Cette section présente rapidement ici, les fonctionnalités principale et le fonctionnement des 3 écrans. Cette introduction permet de démarrer rapidement dans la compréhension du module et ses fonctions principales.

Lorsqu'activé, initialement, le module propose trois écrans :

- un écran d'écoute et manipulation du jeu
- un écran de choix de fichiers
- un écran de recherche de fichier

Les trois écrans sont accessibles par pression tactile sur la dalle d'affichage, dans la zone supérieure de l'écran.

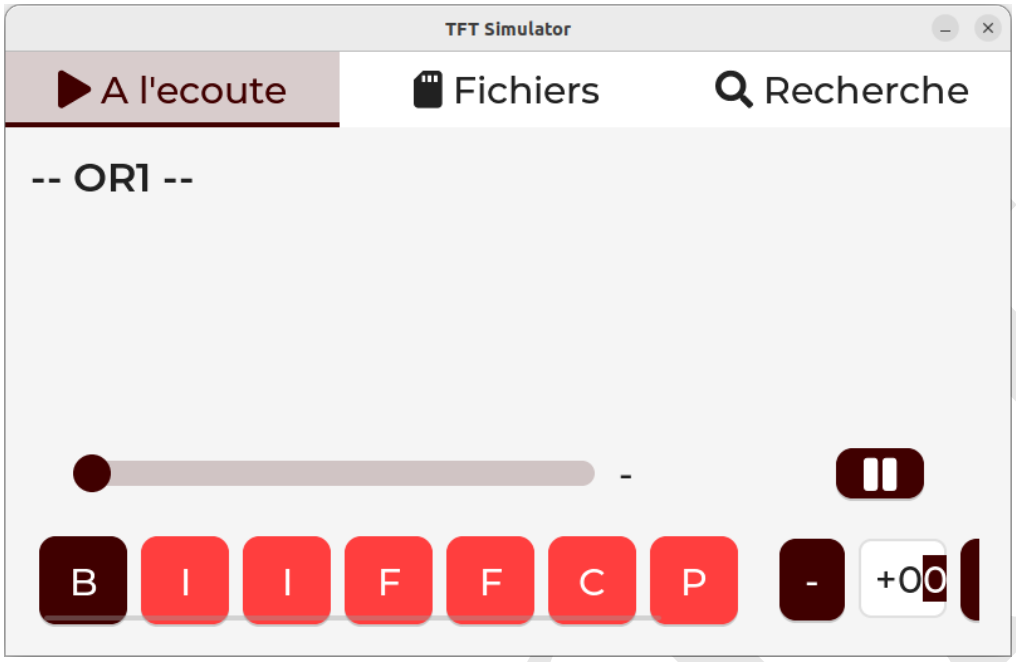


Figure2: Ecran de jeu

Figure3: Ecran de sélection de fichiers

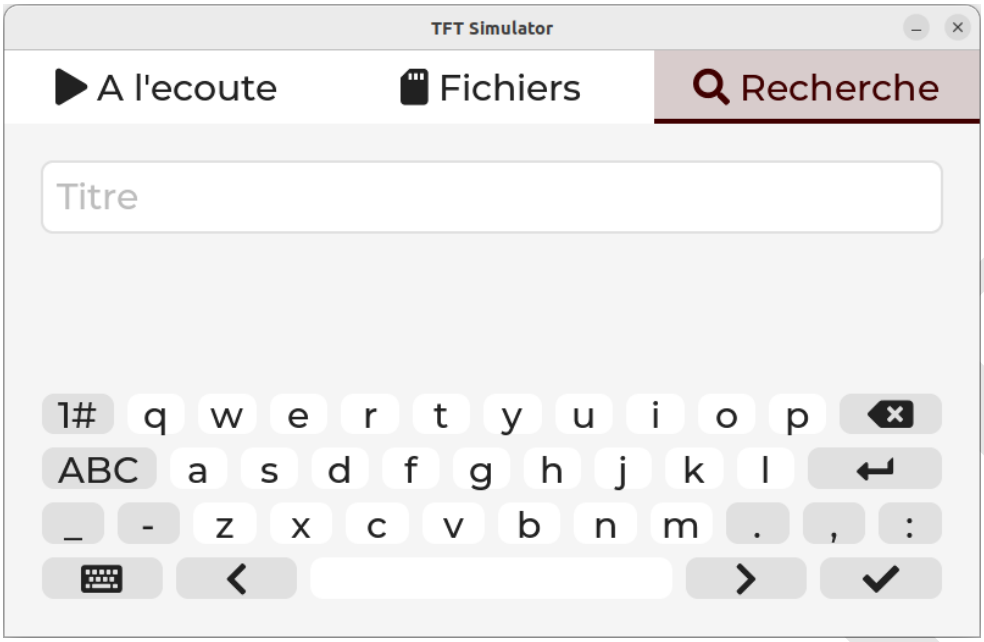


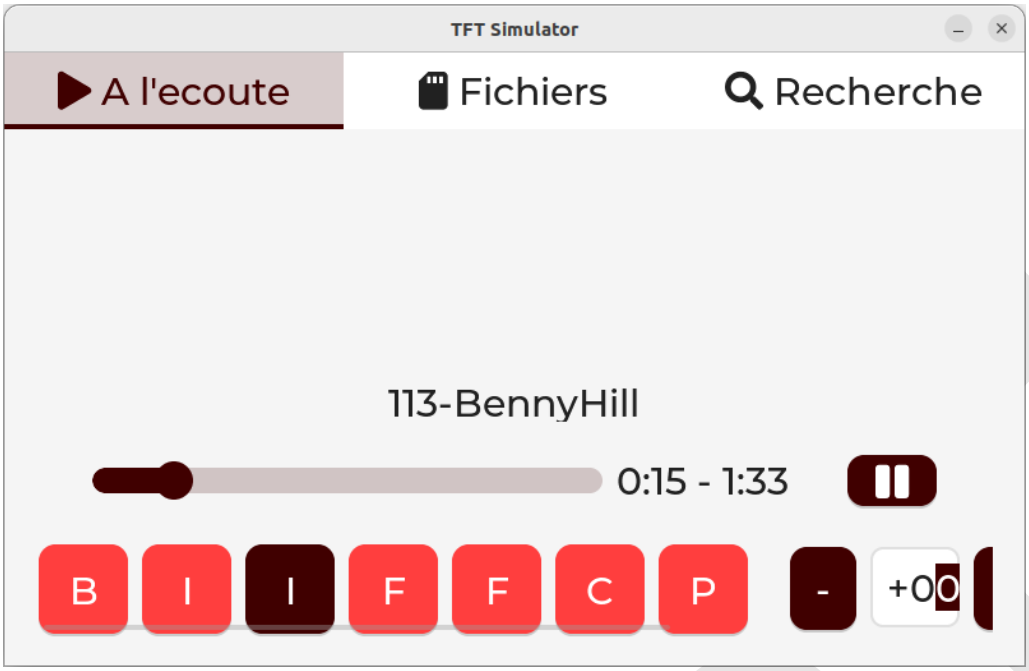
Figure4: Ecran de recherche d'un fichier

2.3.1 Jouer un fichier midi

Aller dans l'écran fichier :

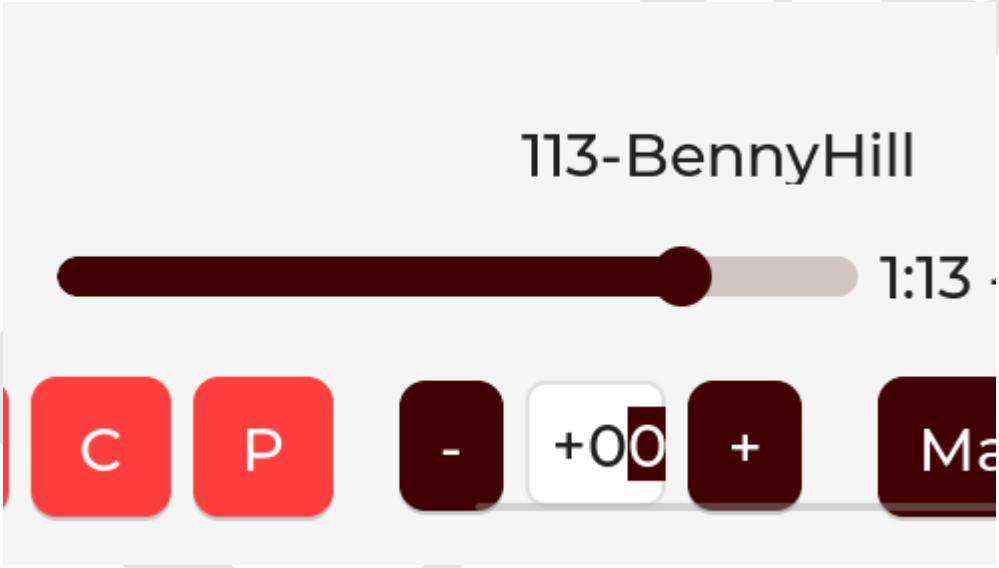
Avec les flèches situées sur la gauche de l'écran, il est possible de se déplacer dans la liste.

Lorsque le fichier est sélectionné, il est possible d'appuyer sur le bouton « play » pour lancer la lecture du fichiers. Le module bascule alors sur l'écran de jeu de fichier.

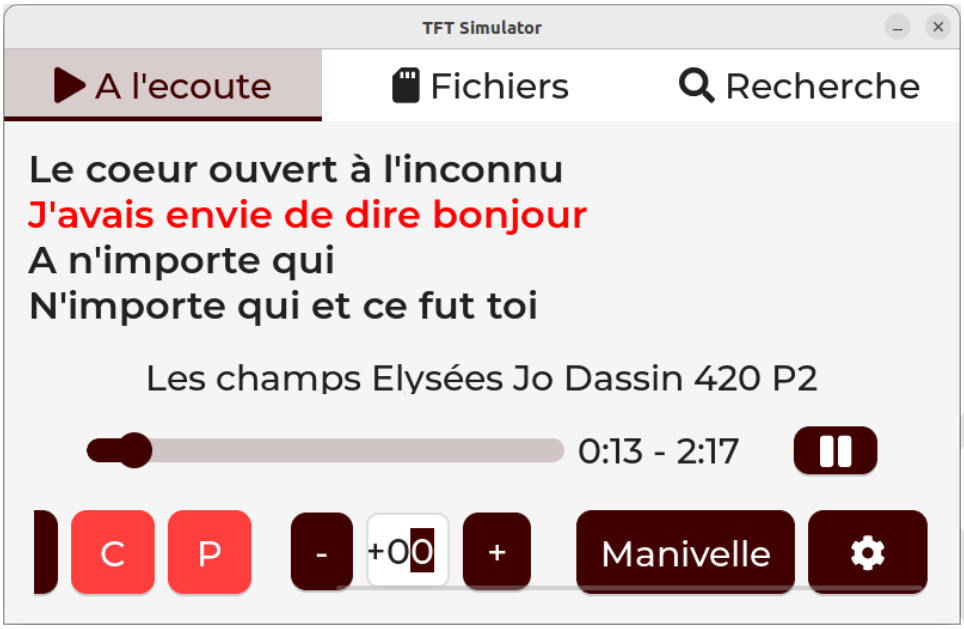


Le morceau est alors joué, des boutons situés en bas de l'écran permettent de changer l'instrument en cours de route.

Un control permet de transposer le jeu du morceau (+- 12 demis tons), permettant d'ajuster la hauteur de chant (plus haut ou plus bas).

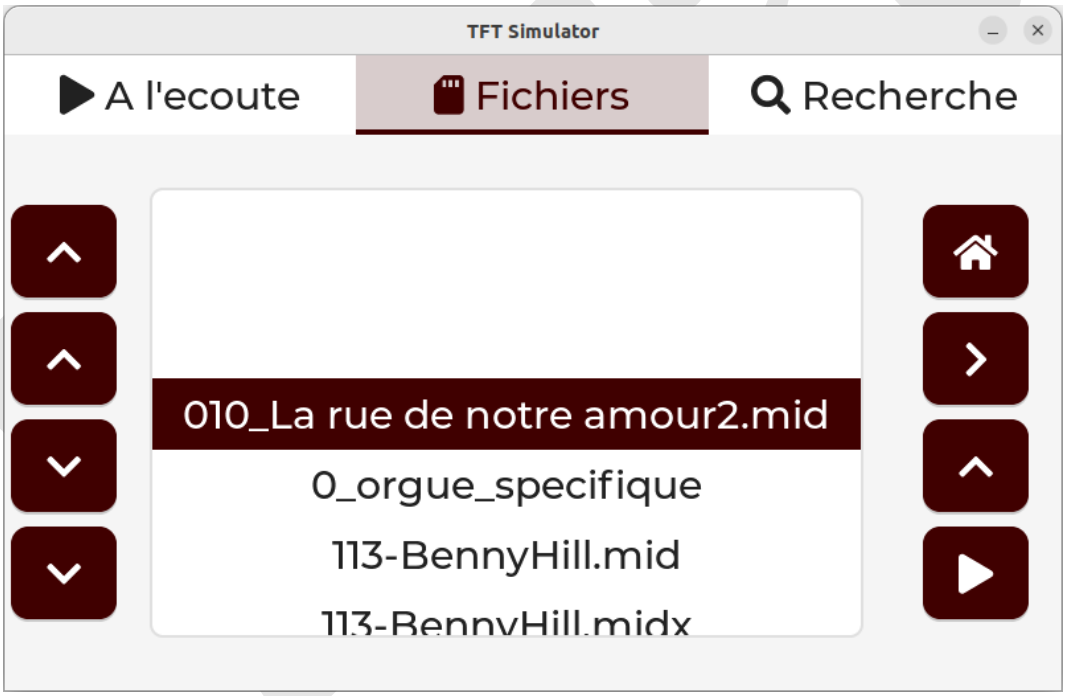


Note : certains fichiers contiennent des paroles, les paroles sont alors affichées en haut de l'écran.



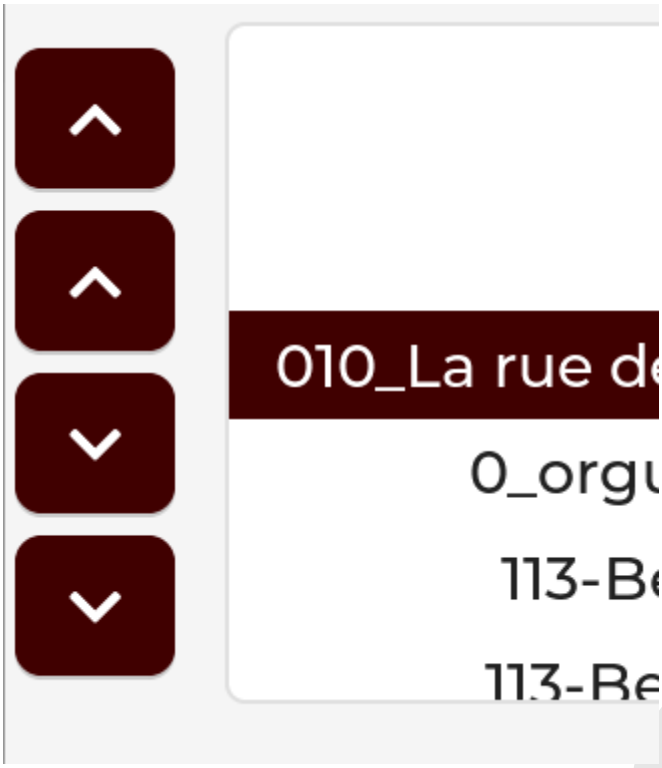
2.3.2 Sélectionner un fichier sur la clef USB

Dans le 2eme ecran, l'utilisateur peut sélectionner un fichier sur sa clef USB. L'écran permet de sélectionner un fichier, entrer dans un répertoire, remonter d'un répertoire.

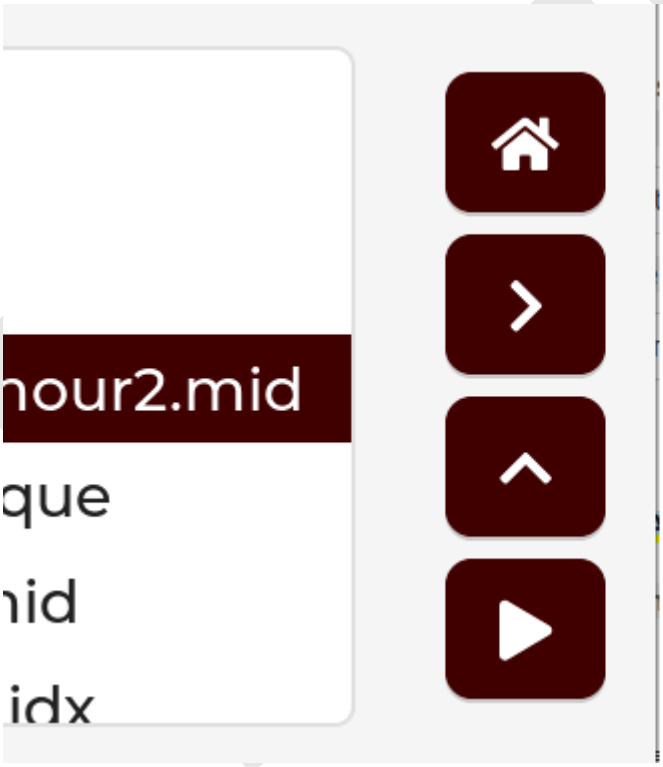


Les boutons proposent les actions suivantes :

Montée rapide ou descente rapide dans la liste des fichiers



Fonctions de navigation dans les répertoire + jeu du fichier.



2.3.3 Étapes de mise en route de la carte (paramétrage) :

Insérez la carte SD, dans l'emplacement dédié sur la carte mère. Cette carte SD contient un logiciel conçu pour les cartes :

Connectez le matériel :

- alimentation (5V)
- manivelle (I2C, SDA, SCL, GND + 3.3V)
- sortie Ampli (prise jack)

à la première mise sous tension, l'écran suivant apparaît :

Sur du matériel inconnu, cet écran apparaît car le programme n'a pas été autorisé. L'autorisation est réalisée une seule fois dans la vie du module. Cette autorisation permet également de référencer le module pour les arrangeurs, pouvant crypter les fichiers pour ce module.

Avec un téléphone portable , il est alors possible de scanner le QR code pour envoyer une demande par mail. Une fois le mail envoyé, le programme est alors compilé spécifiquement et mis en place sur une plateforme d'échange.

Lorsque l'accusé réception de la création du programme est réalisé, une manipulation est nécessaire sur le module pour permettre sa réception.

La réception du programme est réalisée par le cable ethernet (sur la carte).

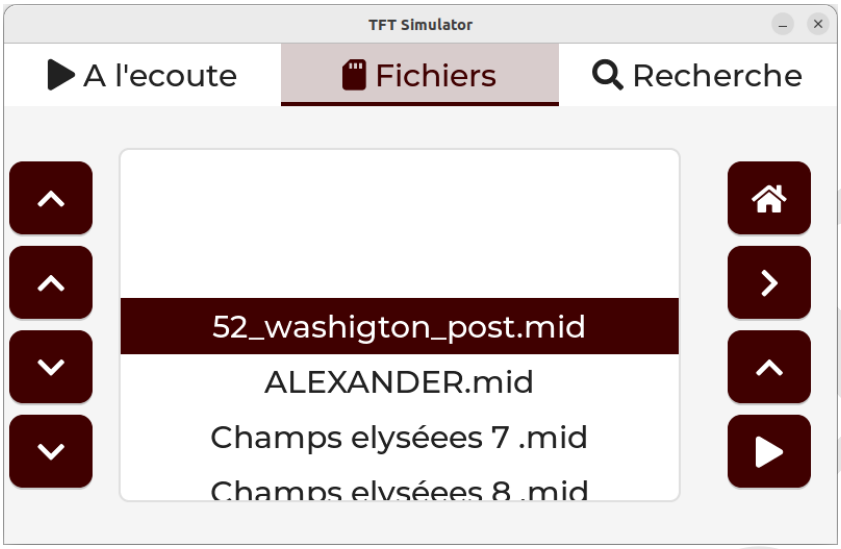
Le branchement sur internet se fait sur par un cable reseau connecté à une box internet.

Une fois celle ci branchée sur le module, il faut procéder comme suit:

- Eteindre et allumer le module
- Lorsque l'écran apparaît, il faut appuyer sur le bouton « Recuperer le fichier automatiquement ».

- Quelques secondes plus tard l'écran principal est affichées

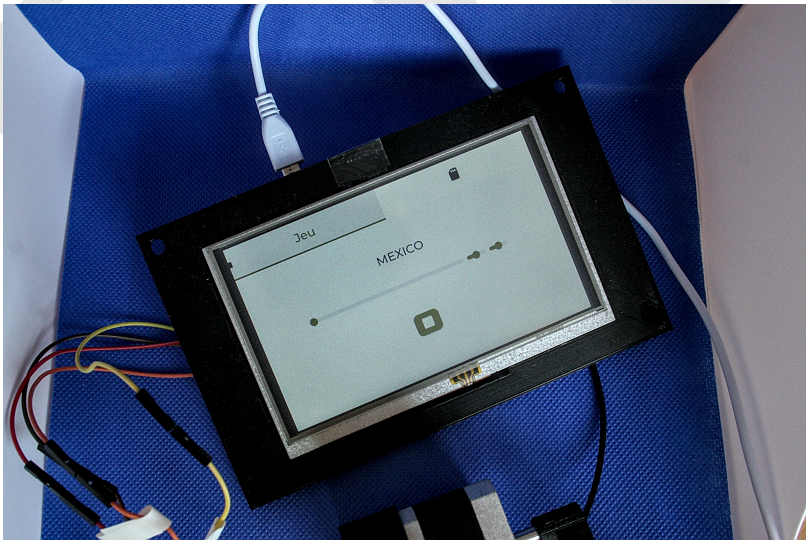
L'opération réalisée avec succès ne sera plus effectuée au redémarrage du matériel.

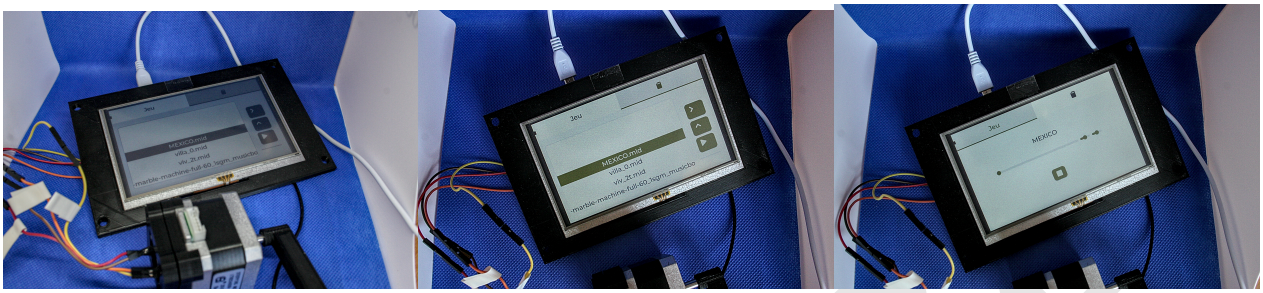


2.4 Électronique et matériel

Le module se présente en différents composants , une carte électronique, un écran, et une carte encodeuse de rotation montée sur un moteur pas à pas (pour le guidage mécanique).

Ecran tactile , 5 pouces (13 cm en diagonale), Résolution : 800 x 480.



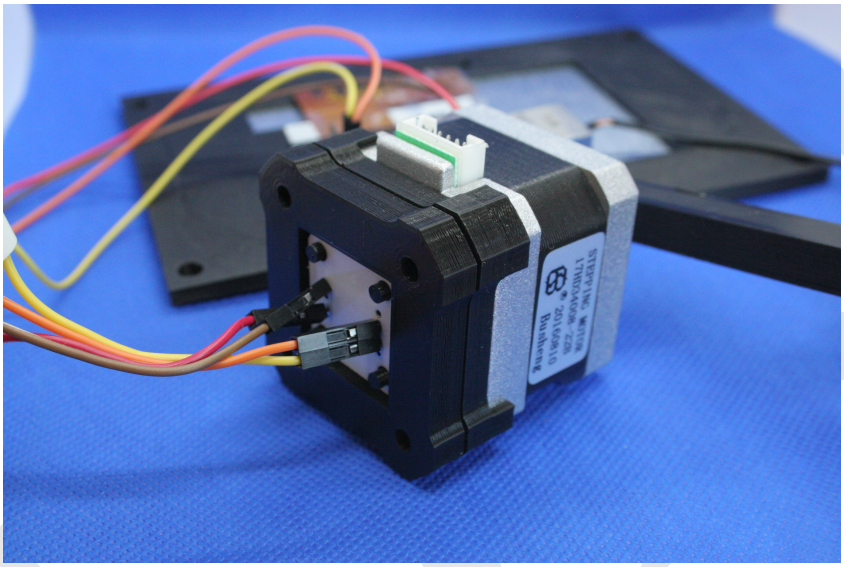


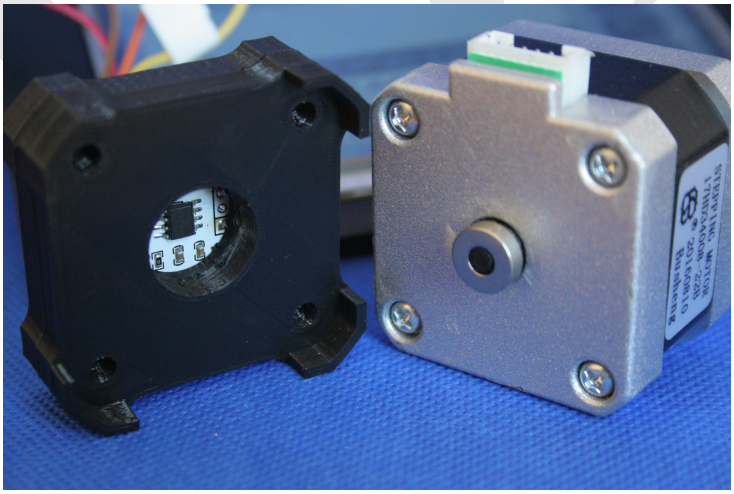
2.4.1 Module manivelle

Le module manivelle proposé est construit sur un moteur NEMA, proposant une simplification mécanisque du guidage de la manivelle.

Ce module , en fonction des besoins, peut être remplacé par un autre système, proposant la connexion avec l'encodeur de gestion de la rotation et de l'avancement du jeu musical.

Ci dessous une illustration du module proposé :





Encodeur magnétique , utilisant une liaison **sans contacts**, et intégrant 4096 incréments par tours. Guidage par roulements, intégré dans le moteur NEMA

Ci dessous, le détail de la pièce de fixation de l'encodeur sur le moteur (position arrière, avec l'aimant)

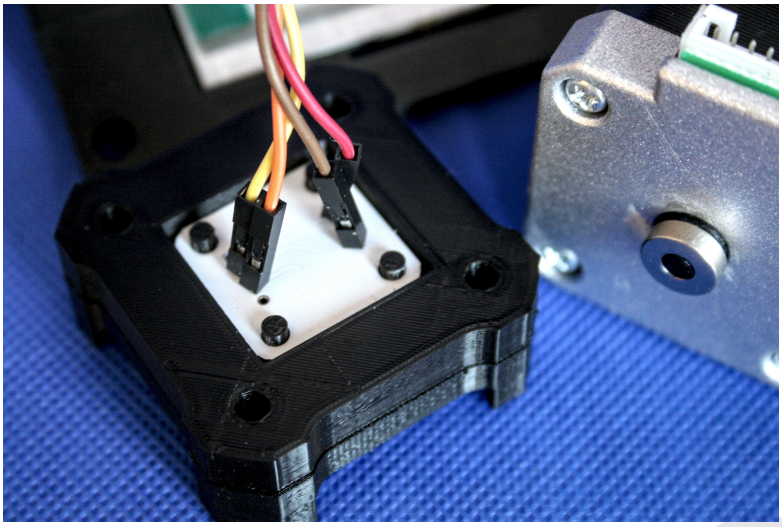
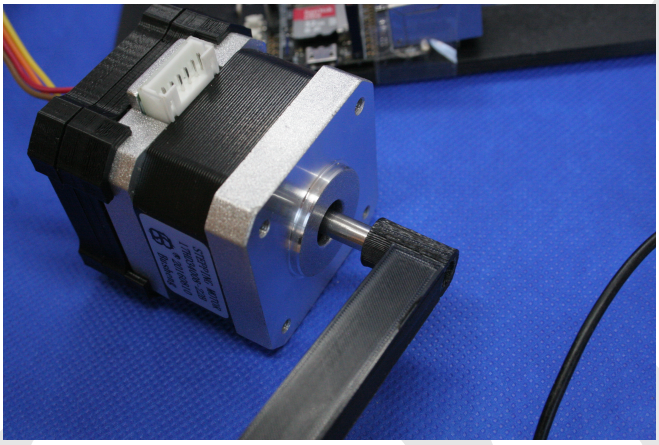
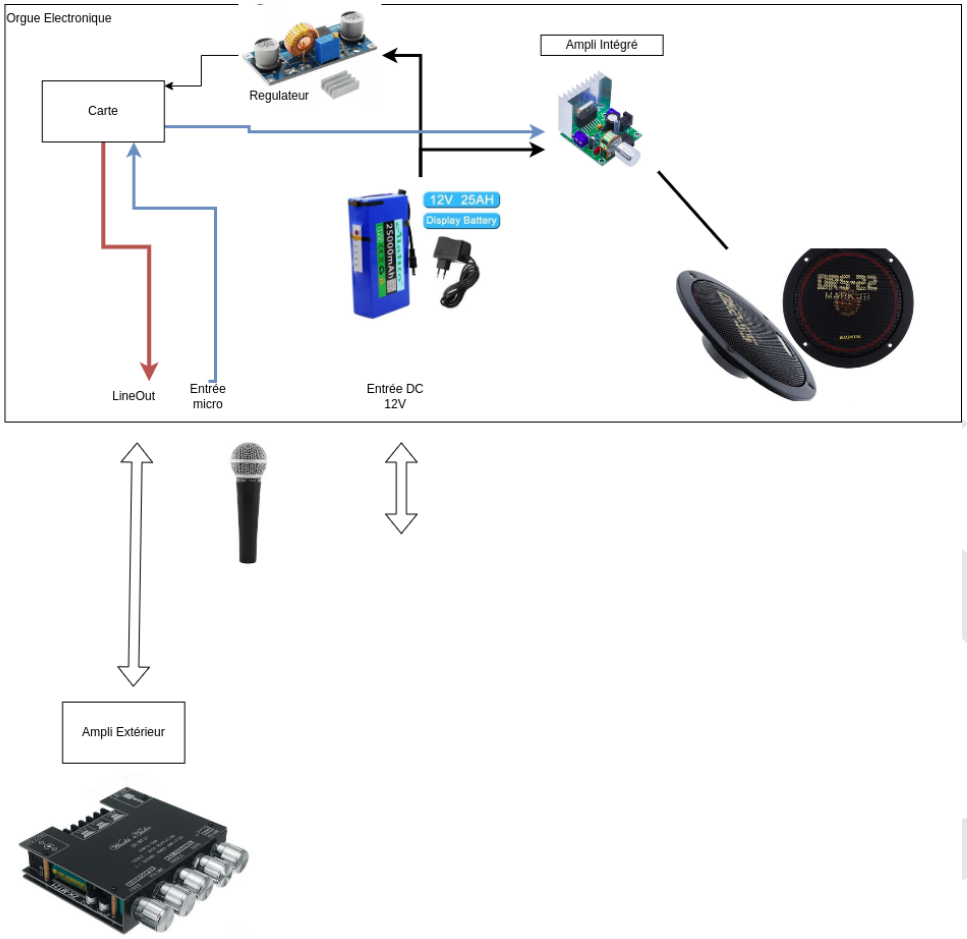


Illustration d'une manivelle fixée sur l'arbre moteur, d'un diamètre de 5 mm.



2.5 Connectique et intégration

Ci dessous un exemple de mise en œuvre dans un instrument :



Vous trouverez plus de détails dans la rubrique Dimension et Encombrement située en annexe.

CHAPTER 3

Utilisation des paroles dans les fichiers MIDI

Le module OR1 propose un affichage des paroles pouvant être insérés dans les fichiers midi. Ces paroles sont enregistrées dans le fichier midi, avec des timings d'affichage pour faire du karaoke, ou apprendre les paroles d'une chanson. Nous regardons ici les différentes méthodes pour ajouter les paroles dans le fichier midi.

Plusieures méthodes existent dans l'encodage des paroles au sein des fichiers MIDI, et cette rubrique vise à donner plus de détails sur l'utilisation et la mise en place dans un fichier Midi, avec le minimum d'outillage pour le module OR1.

3.1 Encodage des paroles pour les fichiers midi

Tout comme les évènements « note » dans les fichiers MIDI, des évènements de type « texte » peuvent également y être insérés. Ces évènements sont utilisés dans les fichier .kar pour le karaoké.

Deux type d'évènements de type texte sont disponibles dans la norme midi, OR1 n'exploite que les évènements de type « Lyric ». Les évènement « Text » sont ignorés.

Les textes dans les évènements Lyric son encodés en utilisant le character set « ISO-8859-1 » (iso latin). Cet encodage est celui utilisé par défaut en europe. Si l'encodage n'est pas correctement utilisé, les accents peuvent ne pas s'afficher correctement sur le module.

3.2 Lecture et utilisation des évènements TEXT Midi, dans le Module OR1

Lors de l'ouverture d'un fichier midi, OR1 effectue deux lectures du fichier. La première lecture extrait les évènements texte et reconstitue des lignes de texte.

Ces lignes sont créée avec le contenu du texte des évènements, et sont découpées en utilisant le caractère de retour chariot (»n ») ou line feed (»c ») à la fin de l'évènement.

ATTENTION, si un évènement texte contient plusieures lignes, uniquement la fin de ligne est évaluée pour le découpage en ligne.

3.2.1 Modification du midi pour insérer les paroles

Il est possible de mettre les paroles dans un fichier midi, en utilisant différents programmes. Ici dans un but illustratif, le programme https://github.com/pietvo/midi2text est utilisé. Celui ci converti au format texte les évènements midi encodés dans un fichier .mid.

exemple :

```
mf2t p1.mid > p1.txt
```

le resultat de l'opération de transcription du fichier midi en texte est le suivant :

```
MFile 0 1 480
MTrk
0 Tempo 500000
0 Meta Text "created with MidiSwing"
488 On ch=1 n=72 v=100
968 Off ch=1 n=72 v=100
1475 On ch=1 n=73 v=100
1955 Off ch=1 n=73 v=100
2413 On ch=1 n=74 v=100
2893 Off ch=1 n=74 v=100
3363 On ch=1 n=75 v=100
3843 Off ch=1 n=75 v=100
4313 On ch=1 n=76 v=100
4793 Off ch=1 n=76 v=100
5300 On ch=1 n=77 v=100
5780 Off ch=1 n=77 v=100
6263 On ch=1 n=78 v=100
6743 Off ch=1 n=78 v=100
7225 On ch=1 n=79 v=100
7705 Off ch=1 n=79 v=100
7705 Meta TrkEnd
TrkEnd
```

Il est remarqué dans le resultat qu'il n'y a aucun évènement de type *Meta Lyric*, En ajoutant nouvellement ces éléments, avec les bons timings, nous pouvons ajouter les paroles :

```
MFile 0 1 480
MTrk
0 Tempo 500000
0 Meta Text "created with MidiSwing"
488 On ch=1 n=72 v=100
968 Off ch=1 n=72 v=100
968 Meta Lyric "ligne 1\r"
1475 On ch=1 n=73 v=100
1955 Off ch=1 n=73 v=100
2413 On ch=1 n=74 v=100
2893 Off ch=1 n=74 v=100
2893 Meta Lyric "ligne 2\r"
3363 On ch=1 n=75 v=100
3843 Off ch=1 n=75 v=100
4313 On ch=1 n=76 v=100
4793 Off ch=1 n=76 v=100
```

(suite sur la page suivante)

(suite de la page précédente)

```
4793 Meta Lyric "ligne 3\r"
5300 On ch=1 n=77 v=100
5780 Off ch=1 n=77 v=100
5780 Meta Lyric "ligne 4\r"
6263 On ch=1 n=78 v=100
6743 Off ch=1 n=78 v=100
6743 Meta Lyric "ligne 5\r"
7225 On ch=1 n=79 v=100
7705 Off ch=1 n=79 v=100
7705 Meta TrkEnd
TrkEnd
```

Puis, reconvertir le fichier texte en fichier midi :

```
t2mf p1.txt > p1.mid
```

Une fois ouvert avec le module nous obtenons le resultat, suivant :

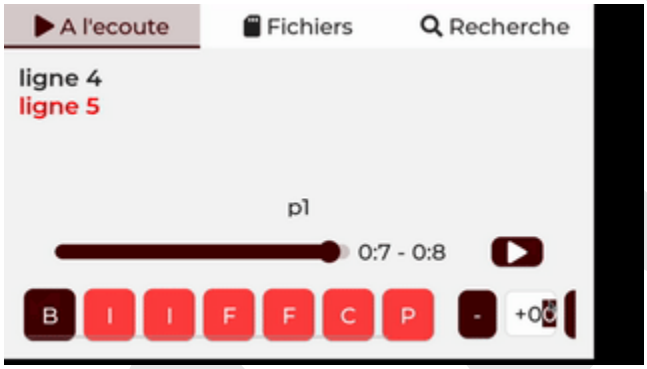


Figure1: Module affichage les paroles nouvellement insérées dans le fichier midi

CHAPTER 4

Paramétrage des sons et création d'instruments

Le module OR1 permet la définition d'instrument qui pourront être lus et sélectionnés dans le module. De base, le module OR1 effectue une restitution basée sur des fichiers WAV qui sont mélangés en fonction de la note demandée. Chaque note est associés à un son qui sera joué.

4.1 Organisation des instruments

Le module gère plusieurs instrument en même temps, ceci permet de pouvoir facilement et rapidement passer d'un instrument à un autre. Une répertoire est donc mis en place pour héberger tous les instruments utilisables par le module.

```
├── Instrument-Anche
│   └── Standard+Orch
└── Instrument-Flute
    └── Standard+Orch
```

Ce répertoire, analysé au démarrage de l'application, doit contenir des fichiers descripteurs pour informer du format de l'instrument ainsi que les éléments associés (fichier wav, registres , ..)

Chaque instrument a sa description stockée dans le fichier YAML : MANIFEST.yml

4.2 Définition d'instrument avec le synthétiseur OR1

Dans la version OR1 , le module utilise des fichiers WAV pour la restitution des sons. Chaque fichier WAV est associé à une note. Le nom du fichier WAV est composé de la note et de l'octave. Par exemple, le fichier « Anche_0.WAV » est associé à la note 0 et au registre « Anche ».

Le synthetiseur nécessite également un fichier de description de l'instrument. Ce fichier est au format YAML et doit être nommé « MANIFEST.yml ». Ce fichier contient les informations suivantes :

« name » : nom de l'instrument « registers » : liste des registres additionels « percussions » : répertoire contenant les fichiers WAV des percussions (canal 10 en Midi), en respectant l'ordre GM (general Midi pour la spécification des percussions)

exemple de fichier MANIFEST.yml

```
instrument:
  name: "Instrument Generique Anche et registre Flutes"
  registers: ["Flute", "Anche"]
  percussions: "Standard+Orch"
```

Dans cet exemple ci dessus, l'instrument est nommé avec la balise « name », ce nom est celui présenté à l'API et dans l'application. la balise « registers » indique les registres additionels pouvant être activés avec cet instrument.

On retrouve à l'intérieur du répertoire, les fichiers sonores associés. Les fichiers wav sont des fichiers non compressés, stéréos, et encodés en 16bits signés.

exemple de contenu d'un instrument :

```
Instrument1/
├── MANIFEST.yaml
├── Anche_0.WAV
├── Anche_100.WAV
├── Anche_101.WAV
├── Anche_37.WAV
...
├── Anche_38.WAV
├── Anche_39.WAV
├── Anche_3.WAV
├── Anche_9.WAV
├── DEFAULT_0.WAV
...
├── DEFAULT_100.WAV
├── DEFAULT_99.WAV
├── DEFAULT_9.WAV
├── Flute_48.WAV
...
├── Flute_49.WAV
├── Flute_84.WAV
├── Flute_85.WAV
├── Flute_86.WAV
├── Flute_87.WAV
```

4.3 Extension de définition pour la lecture des fichiers BOOK

Les fichier book, sont des fichiers XML contenant la description d'un carton. Dans le fichier book, on retrouve la description de l'instrument utilisé pour la lecture du carton. Il faut donc ajouter à la définition de l'instrument, pour chaque piste, la note midi utilisée pour la restitution sonore.

cette possibilité est réalisée par le champ « midimapping », comme suit :

```
instrument:
  type: "MIDI"
  name: "Flutes2"
  registers: []
  midimapping:
    0: [0, 57]
```

(suite sur la page suivante)

(suite de la page précédente)

```
    1: [0, 48]
    2: [0, 50]
    3: [0, 53]
    4: [0, 55]
    5: [0, 60]
    6: [0, 62]
    7: [0, 64]
    8: [0, 65]
    9: [0, 66]
    10: [0, 67]
    11: [0, 69]
    12: [0, 70]
    13: [0, 71]
    14: [0, 72]
    15: [0, 73]
    16: [0, 74]
    17: [0, 75]
    18: [0, 76]
    19: [0, 77]
    20: [0, 78]
    21: [0, 79]
    22: [0, 80]
    23: [0, 81]
    24: [0, 82]
    25: [0, 83]
    26: [0, 84]
    27: [0, 86]
    28: [0, 68]
```

Dans cet exemple, on retrouve la définition de l'instrument « Flutes2 » qui est de type « MIDI ». Le champ « midimapping » contient la liste des canaux et notes midi à utilisées pour la restitution sonore, pour chaque piste (ici la définition d'un carton 29t est utilisée).

4.4 Description des instruments utilisant des soundfont sf2 (module OR1.1, avec support synthétiseur)

Le module logiciel OR2 permet l'utilisation de bank de sons soundfont. https://fr.wikipedia.org/wiki/SoundFont Ce sont des ficheirs sf2 permettant l'encodage de sons d'instrument et éditables avec le logiciel polyphone https://www.polyphone-soundfonts.com/

La définition d'un instrument utilisant cette définition contient des configuration associée dans le fichier MANIFEST.yml

exemple de définition :

```
instrument:
  type: "Synth"
  name: "B"
  registers: ["DEFAULT"]
sf2: "/usr/share/sounds/sf2/default-GM.sf2"
sf2registers:
```

(suite sur la page suivante)

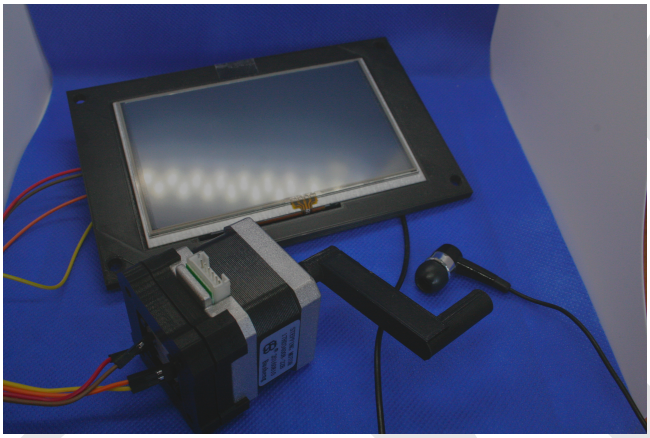
(suite de la page précédente)

```
- channel: 0
  banknum: 0
  presetnum: 22
  register: "DEFAULT"
```

Le répertoire ne contient pas de fichier wav, mais un fichier sf2 référencé. Pour chaque registre, un champ banknum et preset indique le numéro de la banque et du preset à utiliser pour la restitution du son. Le champ register indique le registre associé à ce son.

CHAPTER 5

Encryption des oeuvres digitales



Le module OR1 permettant la création d'instrument de musique mécanique digitaux intègre une protection des œuvres des arrangeurs et auteurs. Cet article explique le fonctionnement du système de protection, avec les détails techniques d'implémentation. Ces informations sont publiques, et visent à expliquer le système de protection utilisé.

Le module OR1 intègre une clef de cryptage privée, avec des éléments matériels et logiciels. Une phase d'enregistrement (ou « enrollment ») et de compilation permet de créer un logiciel spécifique pour chaque plateforme OR1, en fonction des éléments matériels.

Ainsi il est possible pour ce module OR1, d'encrypter des fichiers spécifiquement pour une carte, ne pouvant être utilisés sur d'autres matériels.

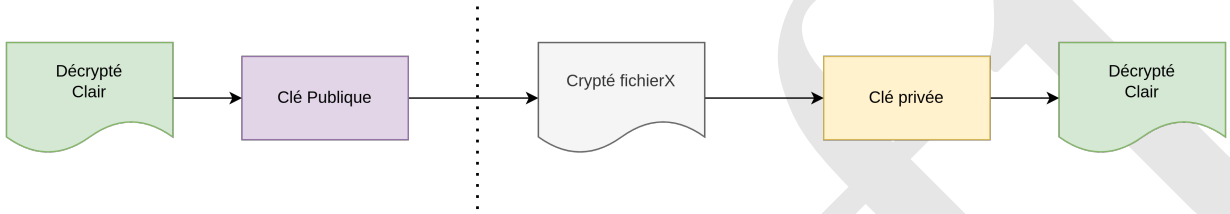
5.1 Vue générale du système de protection utilisé

Le cryptage utilisé est asymétrique, et repose principalement sur une paire de clefs RSA (utilisé dans les systèmes bancaires, et). Une clef permet le cryptage et une autre clef permet le décryptage.

Ainsi, un fichier décrypté (midi, ou book), peut être crypté en utilisant la **clef publique**, disponible pour tous.

Ce fichier crypté, pourra être décodé uniquement avec la clef privée.

On peut voir le processus sur le schéma ci dessous :



plus d'information sur le cryptage, dans l'article ci joint : https://cryptobook.nakov.com/asymmetric-key-ciphers/the-rsa-cryptosystem-concepts

La clefs privée est générée à la déclaration du module et est compilée en partie dans le logiciel. Cette clef n'est pas directement accessible, et dépends du matériel (protection sur la lecture de la clef privée).

Les fichiers midi sont décryptés en mémoire lors de la lecture du fichier, le module ne stocke pas de version décryptée, la lecture est réalisée en flux.

5.2 Utilisation du cryptage par les arrangeurs

Dans le cadre de l'utilisation du module OR1, les auteurs de fichiers midi ont un accès à la clef publique de cryptage, leur permettant de pouvoir crypter les fichiers, sans donner accès aux fichiers midi. (les fichiers midi et leur contenu reste sur l'ordinateur de l'arrangeur).

Un logiciel de cryptage est proposé pour permettre le cryptage des fichiers pour une carte OR1. Ce logiciel permet :

- la récupération de la clef publique d'un instrument, pour le cryptage
- la sélection d'un ou plusieurs fichiers midi
- la création d'un répertoire contenant les fichiers encryptés

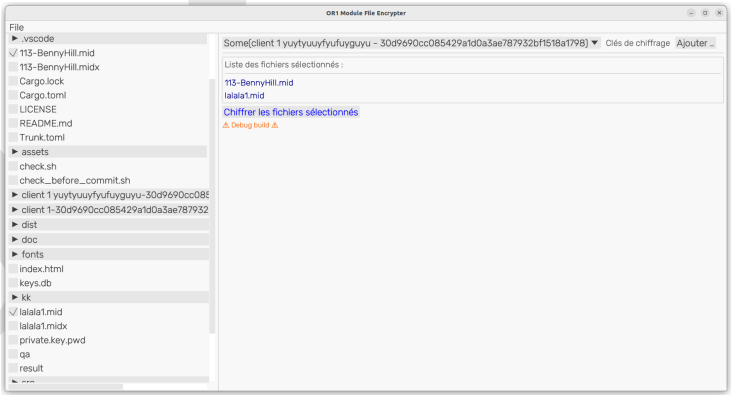
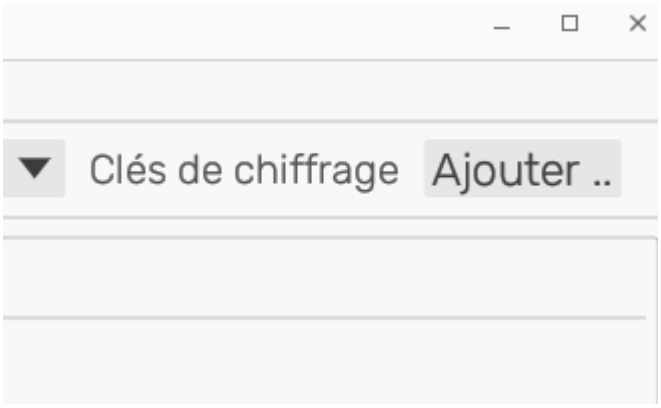


Figure1: Copie d'écran du logiciel d'encryptage

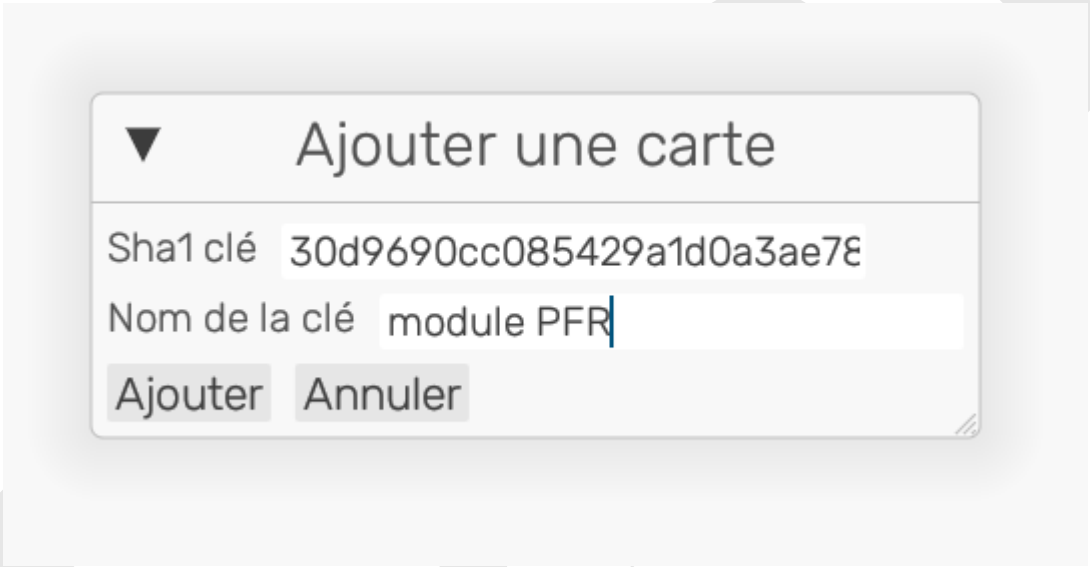
5.3 Ajout d'une nouvelle clef publique dans le logiciel

La récupération de la clef publique d'un module OR1 est réalisée en quelque secondes à partir du numéro du module.

- en appuyant sur le bouton « Ajouter »



Une fenêtre de saisie apparaît, pour y saisir le n° du module.

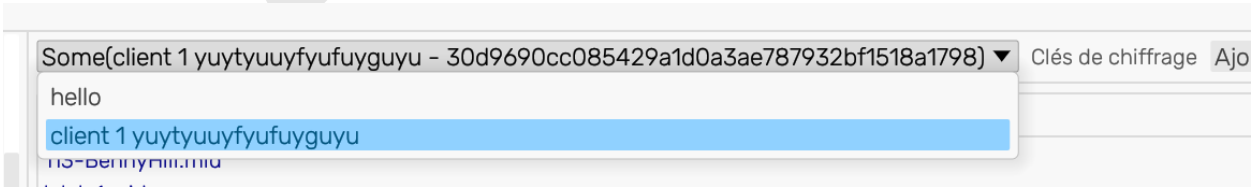


On entre la clef matérielle du module (SHA1 clé), on indique également un libellé propre à la clé pour simplifier l'utilisation.

En appuyant sur « Ajouter », le logiciel récupère la clef publique stockée sur un site de référence, et qui est insérée dans une base de donnée locale au logiciel.

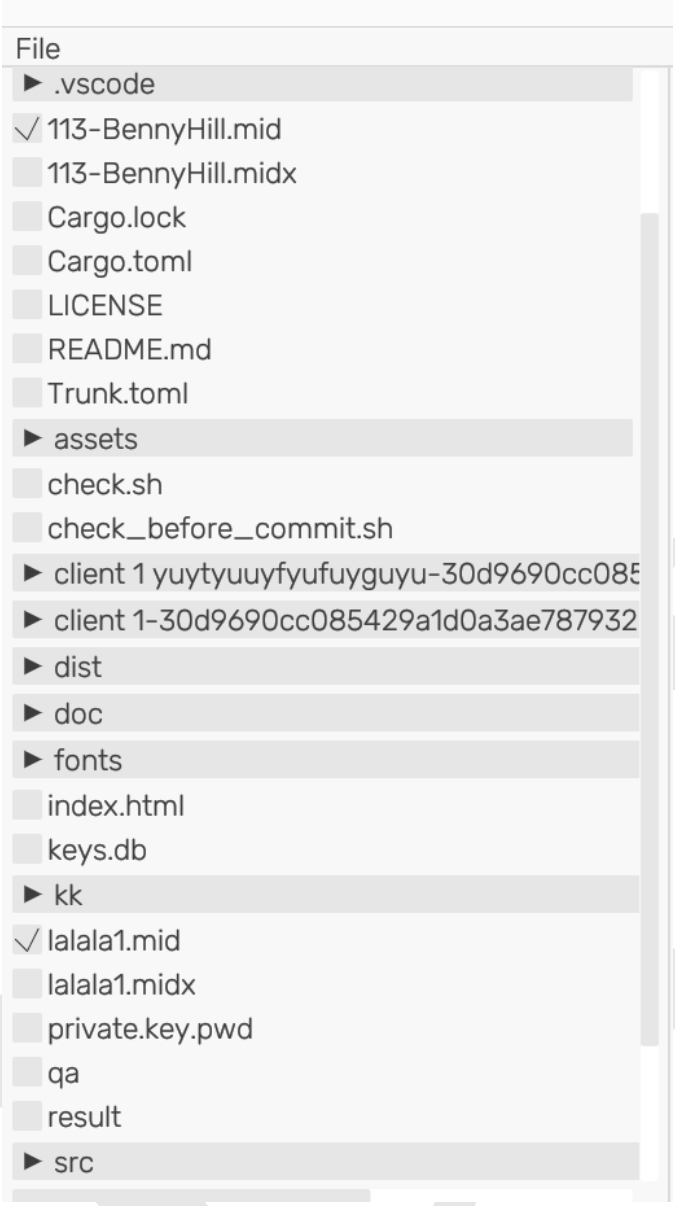
Nota : si la clef est disponible sous forme de fichier, il est possible de l'ajouter manuellement dans le logiciel en utilisant un client SQLite.

La nouvelle clé apparaît alors dans le choix de clef et peut être utilisée pour le cryptage des fichiers.



5.3.1 Cryptage de fichiers

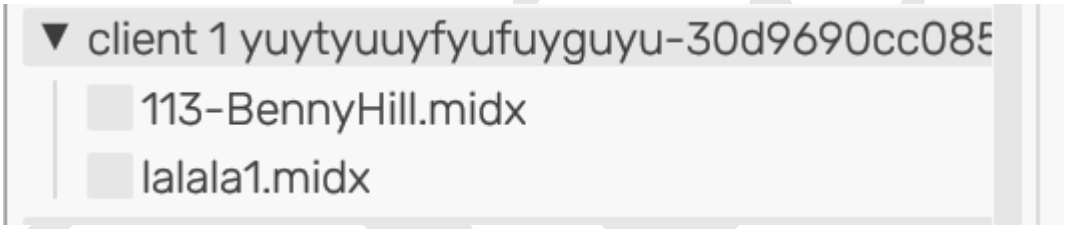
Une fois la clé publique ajoutée, il est possible de la sélectionnée, puis de sélectionner sur le volet gauche du logiciel les fichiers à crypter pour ce module.



Le panneau central récapitule alors la liste des fichiers sélectionnés, et propose un bouton « Chiffrer les fichier sélectionnés » pour crypter les fichiers.

Une fois le cryptage réalisé, un message informe du succès de l'opération de cryptage

Les fichiers cryptés résultants sont créés dans un répertoire spécifique, associé à la clé et reprenant le libellé et le n° du module



On note le « x » à la fin du fichier mentionnant que le fichier est crypté, et non utilisable en dehors du module.

Ces fichiers peuvent être alors transmis sur une clef USB ou par mail, pour utilisation dans le module OR1.

5.4 FAQ : Questions fréquentes sur le cryptage

Q : Je suis arrangeur, est ce que mes fichiers midi sortent de mon ordinateur pour être utilisés sur le module OR1 ?

R : non, l'utilisation du programme d'encryptage est local à la machine de l'arrangeur et les fichiers ne sont pas envoyés.

Q : Je suis arrangeur, comment puis-je vérifier le fonctionnement du cryptage sur le module OR1 du client ?

R : Vous ne pouvez pas directement, la clef privée est propre au module, et vous ne pouvez pas la récupérer, si vous n'êtes pas confiant du principe, vous pouvez faire un essai en direct avec la personne en présence, sur son module OR1.

Q : Qu'est ce qui garantit que quelqu'un ne piratera pas le système ?

R : La sécurité informatique repose sur une notion de « temps de décryptage », notamment pour le cassage des clefs ou cracking du programme. Donc il n'y a pas de sécurité absolue, toute l'informatique repose sur ce principe. Néanmoins,

il est utilisé ici des systèmes standard de cryptage, implémentés par des professionnels, permettant d'assurer que la personne ayant un fichier midx, mettra des années à récupérer le contenu en dehors du module OR1

Q : est-ce que les fichiers décryptés peuvent être envoyés sur une sortie midi ?

R : non, les fichiers encryptés midi, ne peuvent pas être envoyés sur une prise midi. La synthèse et la génération du son est directement faite pas le module et envoyée sur la sortie sonore. Il est possible de faire une reconnaissance IA, cependant l'attaquant perd une partie des informations (timing, tempo, paroles, ..)

CHAPTER 6

Versions du logiciel

6.1 Encrypter - Version 1.0.16 - 2024-01-21

Téléchargement de la version :

https://github.com/frett27/encrypter/releases/tag/prerelease-v1.0.16

Changements et améliorations :

- Amélioration d'ergonomie (affichage des erreurs dans les panneaux concernés)
- Indication sur les étapes de la démarche.
- Ajout du support de clefs publiques ajoutée par fichier texte ou texte libre.

6.2 Version 2023-12-14 - Module OR1

Améliorations et corrections :

- Prise en charge de l'arrêt du jeu si la manivelle est arrêtée
- Ajout de la clef publique dans les panneaux d'informations

6.3 Version 2023-09 - Module OR1

Améliorations et corrections :

- Correction sur des fuites mémoire pouvant gêner après 4 heures de jeu en continu (sans redémarrages)

CHAPTER 7

SDK

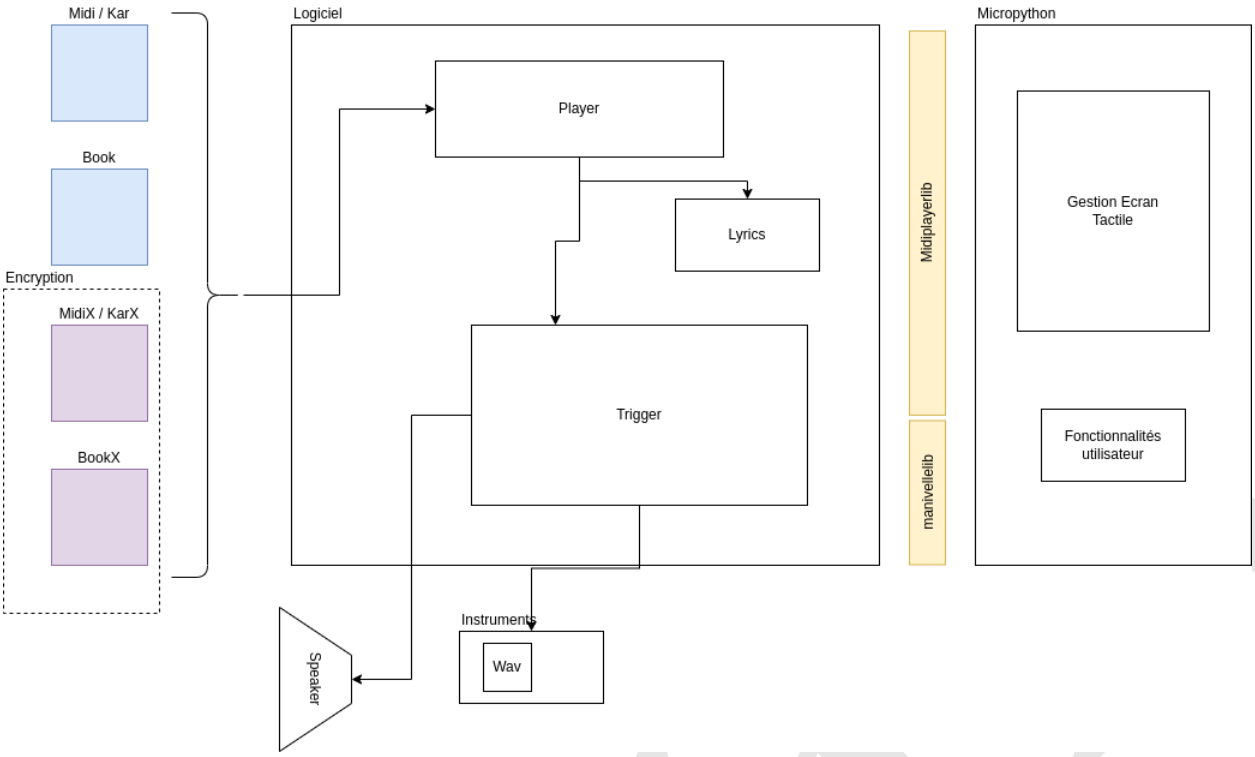
Le SDK propose les informations nécessaires pour créer des logiciels utilisant le module OR1, ou adapter les fonctionnalités du logiciel existant.

7.1 Organisation logicielle générale

La partie logicielle du module OR1 comprends les éléments suivants:

- un système linux avec les différents drivers matériels
- un interpreteur micropython, compilé spécifiquement pour cette plateforme
- une bibliothèque libmidiplayer, exposée sur l'interpreteur micropython

Cette organisation est représentée comme ci dessous :



La partie logicielle est au centre du fonctionnement du module, cette partie implémentée en C/C++/rust fourni l'ensemble des fonctions nécessaires à la création d'écran et la partie sonore.

Cette partie est exposée à l'interpreteur python pour l'orchestration.

Plusieurs composants sont présents dans le logiciel : le module *trigger* réalisant la synthèse sonore à partir des fichiers wav, permettant de mixer au rythme souhaité, les différentes voies, avec les registres et récalages sonores. Un composant *player* prenant en charge la lecture des fichiers, l'envoi des notes au trigger, la gestion du temps et tempo.

7.2 Utilisation l'API micropython

Micropython est un interpreteur python spécialisé pour les petites plateformes, certains points sont à noter sur cette version, si le lecteur est familié avec le développement en python :

- l'allocation de mémoire est réalisé différemment (on ne passe pas par le système, et est alloué dans une arena)
- la bibliothèque de fonction est la même pour le noyau, cependant les modules sont spécifique à micropython (en fonction de la plateforme)

La documentation complète du language est disponible sur internet à cette adresse : https://docs.micropython.org/en/latest/

7.2.1 Gestion de l'écran

Le module python LGVL est intégré à l'interpreteur, il vient avec un grand nombre de fonctionnalités et bibliotheques de widgets, permettant la construction d'écrans.

La documentation complète de la librairie LGVL est disponible sur internet à cette adresse : https://docs.lvgl.io/latest/en/html/index.html

La librairie est accessible via le module *lvgl*, il est possible de l'importer dans un script python et l'initialiser pour le module OR1, avec le script suivant :

```
import usys as sys
sys.path.append('') # See: https://github.com/micropython/micropython/issues/6419

import lvgl as lv
lv.init()

import fb # Frame buffer driver

disp_buf1 = lv.disp_draw_buf_t()
buf1_1 = bytearray(WIDTH_SCREEN*10)
buf1_2 = bytearray(WIDTH_SCREEN*10)
disp_buf1.init(buf1_1, buf1_2, len(buf1_1)//4)
disp_drv = lv.disp_drv_t()
disp_drv.init()
disp_drv.draw_buf = disp_buf1
disp_drv.flush_cb = conf.cb
disp_drv.hor_res = WIDTH_SCREEN
disp_drv.ver_res = HEIGHT_SCREEN

disp = disp_drv.register() # Register the driver and save the created display object

import ev # Event interface (touchpad, encoder, keyboard, mouse)

fb.init() # Initialize the frame buffer device
ev.init() # Initialize the input device driver

def timercb(timer): # Timer callback
    lv.tick_inc(5)
    lv.task_handler()

player_state_task = lv.timer_create(timercb, 20, None) # every 20ms


if __name__ == '__main__':
    while True:
        utime.sleep(1)
```

7.2.2 Utilisation de la manivelle

Un module *manivelle* permet de se connecter à la manivelle et lire la position absolue de l'axe. Ci dessous un exemple d'utilisation de cette librairie pour afficher en continue la position de l'axe.

```
import manivelle
manivelle.init()

while True:
    print(manivelle.get_pos())
```

7.2.3 Fonctions de référence de la librairie midiplayer

Le module midiplayer contient les fonctions nécessaires à l'utilisation de OR1. il contient les fonctions suivantes :

Fonctions d'initialisation de la bibliothèque

```
"""
Module contenant les fonctionnalités du module OR1
"""

def init():
    """
        initialise la librairie midiplayer
    """
    pass


def feature_set(feature_name, feature_value):
"""
    définit la valeur d'une feature
"""
    pass

def property_set(property_name, property_value):
"""
    définit la valeur d'une propriété de paramétrage
"""
    pass
```

L'initialisation de la librairie est nécessaire au démarrage, pour pouvoir utiliser les fonctions associée.

Fonctions d'utilisation des instruments

Les fonction ci dessous proposent une gestion des bank de sons et instruments.

```
def loadsoundbank(filename):
    """
        charge une bank de sons, filename contient le nom de l'instrument
        cette fonction monte en mémoire la connaissance de la bank de son
        pour l'utiliser dans le module il est nécessaire d'utiliser "definesoundbank"
    """
    pass*



def definesoundbank():
    """
        applique la bank de sons chargée précédemment
    """
    pass


def activebank(bank):
"""
active le jeu de registre "bank"

@exception  une exception est levée en cas d'erreur
"""
    pass

def deactivatebank():
"""
desactive le jeu de registre "bank"

@exception  une exception est levée en cas d'erreur
"""
    pass

def instruments():
"""
    @return retourne une liste de noms d'instrument disponibles pour le jeu
"""
    pass

def get_current_instrument_registers():
"""
    pour l'instrument en cours, donne la liste des registres
"""
    pass

def current_instrument_name():
"""
    donne le nom de l'instrument actif actuellement
"""
    pass
```

Gestion du jeu du fichier, et interaction avec le player

Une fois initialisée, le player peut être piloté avec les quelques fonctions proposées ci dessous.

```
def play(filename):
"""
    Démarrage du jeu en lisant le fichier dont le nom est passé en paramètres

    @exception une exception est retournée en cas d'erreur (lecture de fichier)

"""
    pass

def stop():
"""
    Arrête le player et le jeu du fichier courant
"""
    pass


def isplaying():
"""

    @return un booleen indiquant si le player est en cours de jeu
"""
    pass

def pauseresume():
"""
    effectue une pause / reprise du jeu en cours
"""
    pass
```

Ci dessous les fonctions permettant d'interragir avec le player, et le jeu. On retourne un nombre de fonction permettant de savoir où en est le jeu.

```
def playstreamlength():
 """
     retourne la longueur du fichier en cours de jeu
 """
     pass

 def playstreamposition():
 """
     retourne la position actuelle de jeu dans le fichier
 """
     pass
```

```
def changetempofactor(tempofactor):
"""
change le tempo de jeu
@param tempofactor : un nombre définissant le facteur de temp (1.0 -> vitesse originale,␣
↪0.5 -> vitesse doublée)
```

(suite sur la page suivante)

(suite de la page précédente)

```
"""
    pass


def change_pitch(pitch_int):
"""
@param pitch_int
    nombre de demi ton pour la transposition
    (ce nombre peut être négatif)
"""

    pass


def current_pitch():
"""
@return integer
    retourne le pitch positionné
"""

    pass


def stopallvoices():
"""
    arrete le jeu de toutes les voix (faire silence)
    en cas de mise en pause du player, certaines voix peuvent
    rester active.
    Cette fonction permet d'arreter les voix en cours
"""
    pass

def lyrics_clock():
"""
    @return retourne l'horloge de changement associée aux paroles.
    lorsque les paroles change (associée au morceau),
    le nombre detrourné est
"""
    pass

def lyrics_current():
"""
    @return recupère les paroles affichée en ce moment
"""
    pass
```

Encryption

```
def encrypt(filename):
"""
    encrypte le fichier donné en paramètre, en utilisant la clef cryptée
"""
    pass
```

Gestion de l'autorisation et de la license

```
def license_serial():
"""
    retourne l'identifiant de license
"""
    pass

def license_query():
"""
    retourne l'état d'autorisation de la license, pour pouvoir utiliser les fonctions
"""
    pass
```

7.3 Utilisation des API C

L'api C propose un accès directe aux fonctionnalités du player, sans interpreteur. Cette librairie est utilisable dans des languages natifs ou autres contexte que le micropython.

7.3.1 libmidiplayer.so.1

Cette librairie expose les différentes fonctionnalités principales du module.

```
/**
* init the runtime
*/
void midiplayerlibcinit()

/**
* init the library
*/
int midiplayerlib_init(void)


// properties
static const char *LICENSE_PROPERTY = "lic";
static const char *DISABLE_PERCUSSIONS = "mute_percussions";
static const char *INTEGRATED_MANIVELLE = "integrated_manivelle";
static const char *INTEGRATED_MANIVELLE_MODE = "integrated_auto_manivelle";
static const char *INTEGRATED_MANIVELLE_FACTEUR = "integrated_auto_manivelle_decifacteur
```

(suite sur la page suivante)

(suite de la page précédente)

```
↪";
static const char *OUTPUT_SERIAL_DEVICE = "output_serial_device";
static const char *SOUND_PATH_PROPERTY = "sound_path";
static const char *PLAY_COMMAND_LINE_TEMPLATE = "play_command_line_template";

int midiplayerlib_defineproperty(const char *property, const char *value)


/**
* load a sound bank / instrument
*/
void *midiplayerlib_loadsoundbank(const char *filename)

/**
* encrypt file using the public internal key
*/
int encrypt_file(const char *filepath)

/**
* play file, using it's filename
*/
int midiplayerlib_play(const char *filename)

int midiplayerlib_pauseresume(void) // return -1, 0: false, 1:true

int midiplayerlib_stop(void)

int midiplayerlib_stopallvoices(void)


////////////////////////////////////////////////////////////////

int midiplayerlib_definesoundbank(const void *soundbank)

int midiplayerlib_activatebank(const char *bank)

int midiplayerlib_deactivatebank(const char *bank)

int midiplayerlib_changetempofactor(const double f)

int midiplayerlib_changepitchfactor(const double f)

int midiplayerlib_change_pitch(const int pitch)

int midiplayerlib_current_pitch(int *pitch)

float midiplayerlib_play_currentstreamposition()

float midiplayerlib_play_currentstreamlength()

int midiplayerlib_isplaying() // 1 true , 0 false
```

(suite sur la page suivante)

(suite de la page précédente)

```
int midiplayerlib_activatefeature(const char *feature, int activated)


int midiplayerlib_instrument_count()

int midiplayerlib_instrument_name(int index, char *buffer, int *maxchars)

int midiplayerlib_instrument_current(char *buffer, int *maxchars)

int midiplayerlib_instrument_register_count()

int midiplayerlib_instrument_register_name(int index, char *buffer, int *maxchars)


int midiplayerlib_lyrics_simple_get_lamport(int *clock)

int midiplayerlib_lyrics_simple_get_content(char *buffer, int *maxchar)


int midiplayerlib_encrypt_file(const char *filename)


int midiplayerlib_serial(char *outserial, int *outchars, int *valid)
```

CHAPTER 8

Annexes

8.1 Caractéristiques Techniques détaillées

Caractéristiques sonores

- Restitution en qualité CD (44,1 Khz, 16 bits, Stéréo)
- Polyphonie : plus de 20 voies jouées simultanément
- Le Volume est fixé en sortie de la carte, une possibilité d'ajouter un ampli audio, pour augmenter le volume ou ajuster et la balance des aiguës et des graves.

Electrique / Connexions :

- Alimentation 5V par connexion sur la carte (régulé), ou prise USB
- 200 mA de consommation courant (Avec écran), hors amplification
- Prise Jack 3.5mm pour sortie amplificateur

Caractéristiques du Processeur central :

SoC	Allwinner V3S
Processeur	ARM Cortex-A7
Fréquence de fonctionnem	1.2GHz
RAM	64MB DDR2
Stockage	Flash SPI/Micro SD
	* FPC LCD RVB 40P universel : * Résolutions prises en charge : 272 × 480, 480 × 800,1024, 600 × XNUMX * Puce RTP embarquée, prend en charge un écran tactile
	*SDIO x2 * SPI x1 * I2C x2 * UARTx3 * 100M Ether x1 (inclut EPHY) * USB OTG x1 * MIPI CSI x1 * PWM x2 * LRADC x1 * Haut-parleurx2 + Micro x1
	Micro USB 5 V, broches 2.54 mm Alimentation 3.3 V ~ 5 V ; Alimentation de trou de timbre de 1.27 mm. 1 GHz linux IDLE fonctionne 90 ~ 100 mA ; cpu-burn run ~180mA Température de stockage -40 ~ 125 Température de fonctionnement -20 ~ 70

8.2 Dimensions – Encombrement

8.2.1 Support écran + Carte CPU

Pièce 3D support, facilitant la fixation de l'écran sur l'instrument

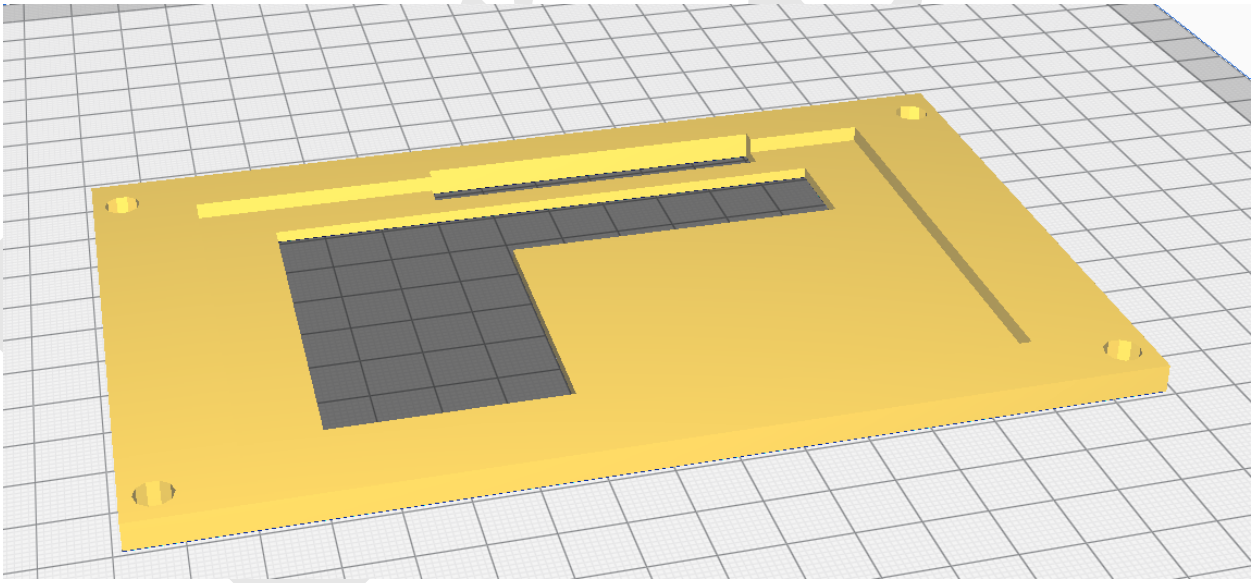


Figure1: Support 3D de l'écran

Fichier STL de la pièce :

`Boitier_Fixation.stl`

Fichier Openscad de la pièce :

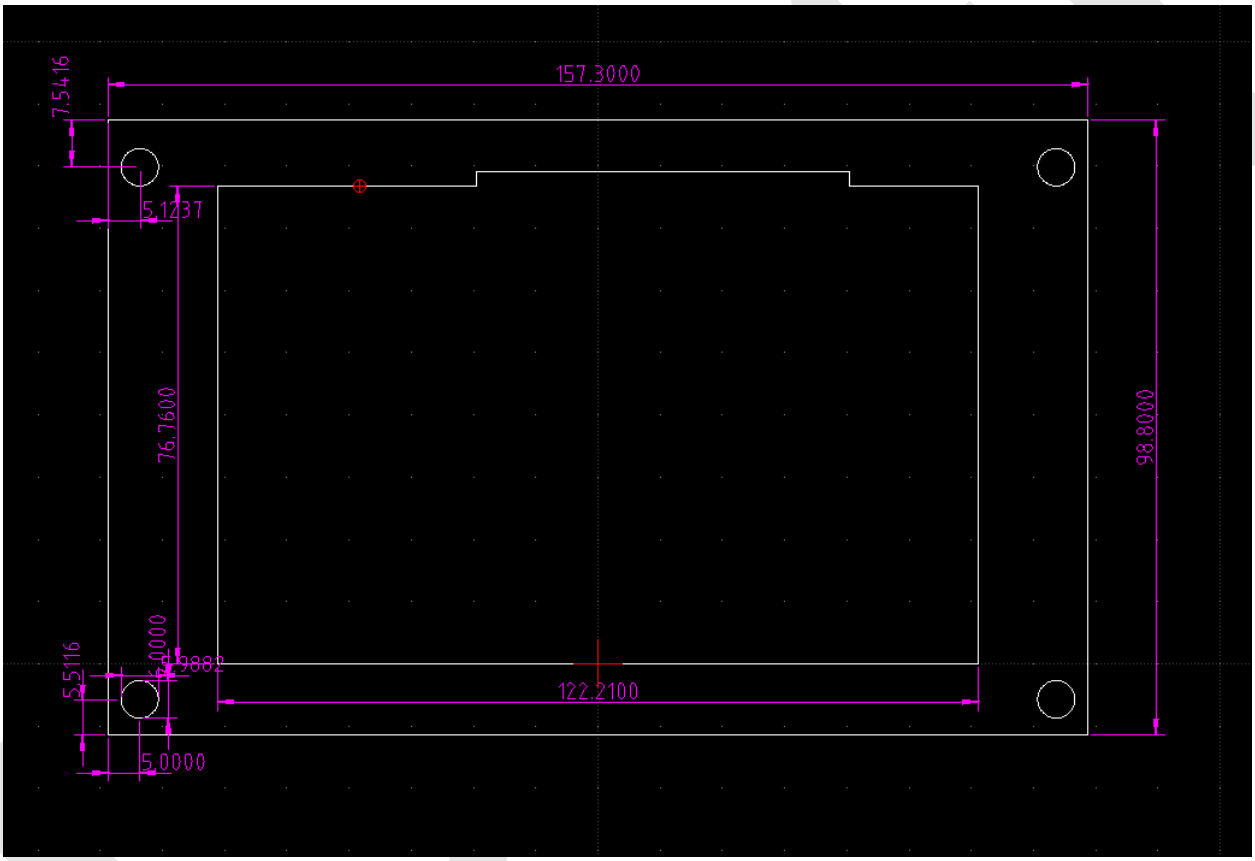


Figure2: Dimensions du support 3D Ecran

`Boitier_Fixation.scad`

Pièce additionelle pour la fixation de la carte mère derrière l'écran

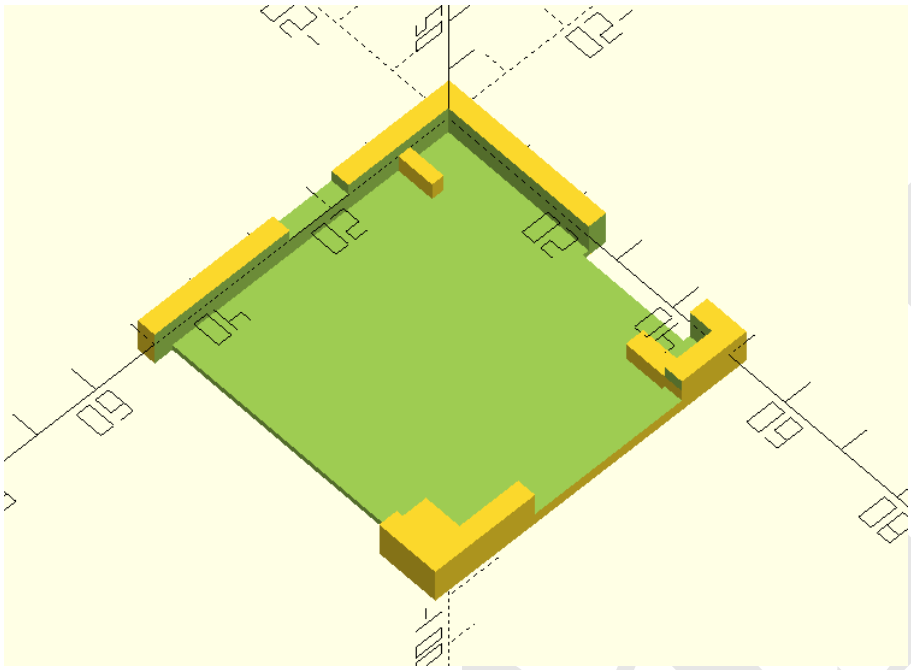


Figure3: Support 3D de fixation de la carte CPU

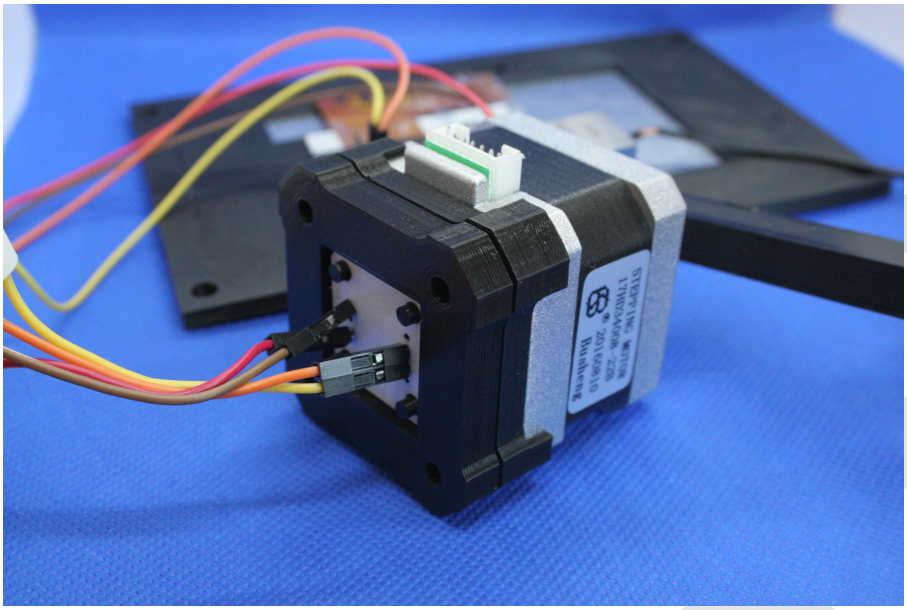
Fichier STL de la pièce :

`fixation_carte_lichee_pi.stl`

Fichier Openscad de la pièce :

`fixation_carte_lichee_pi.scad`

8.2.2 Dimensions fixation manivelle



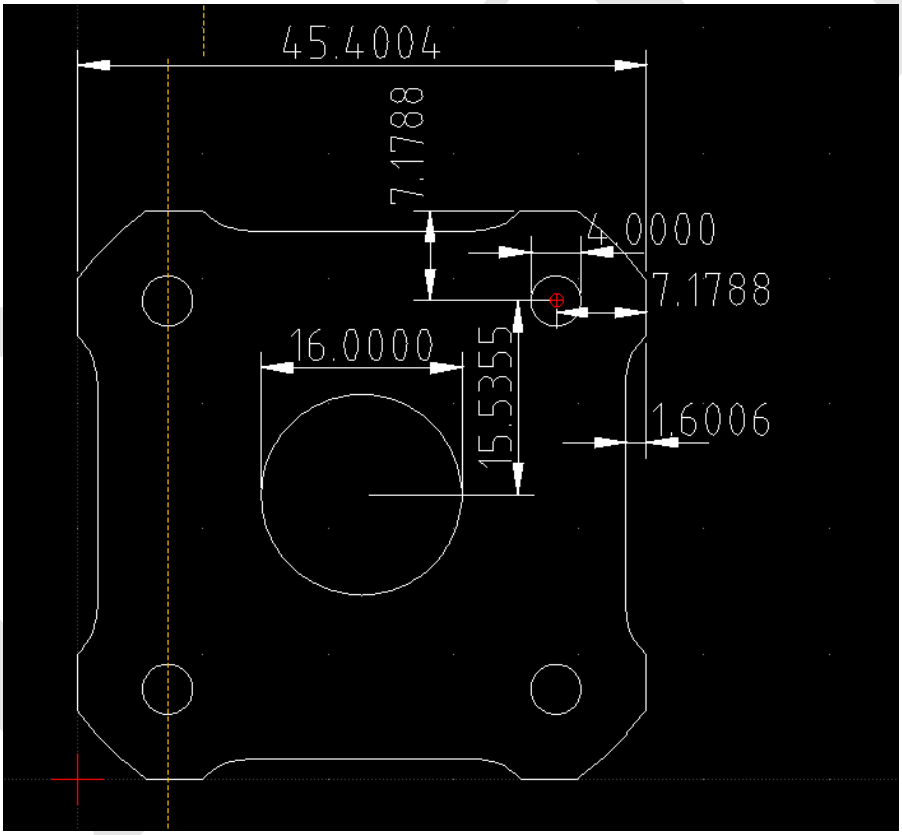


Figure4: Dimension support de manivelle

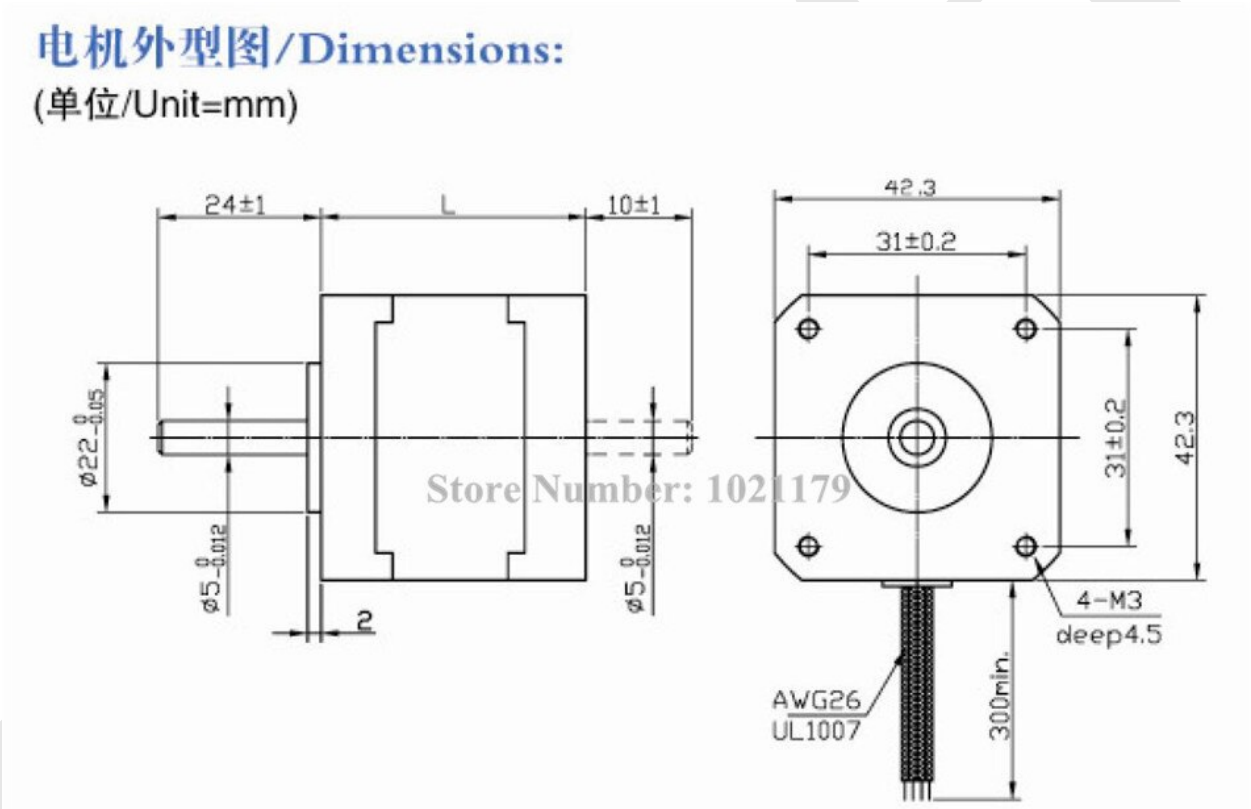
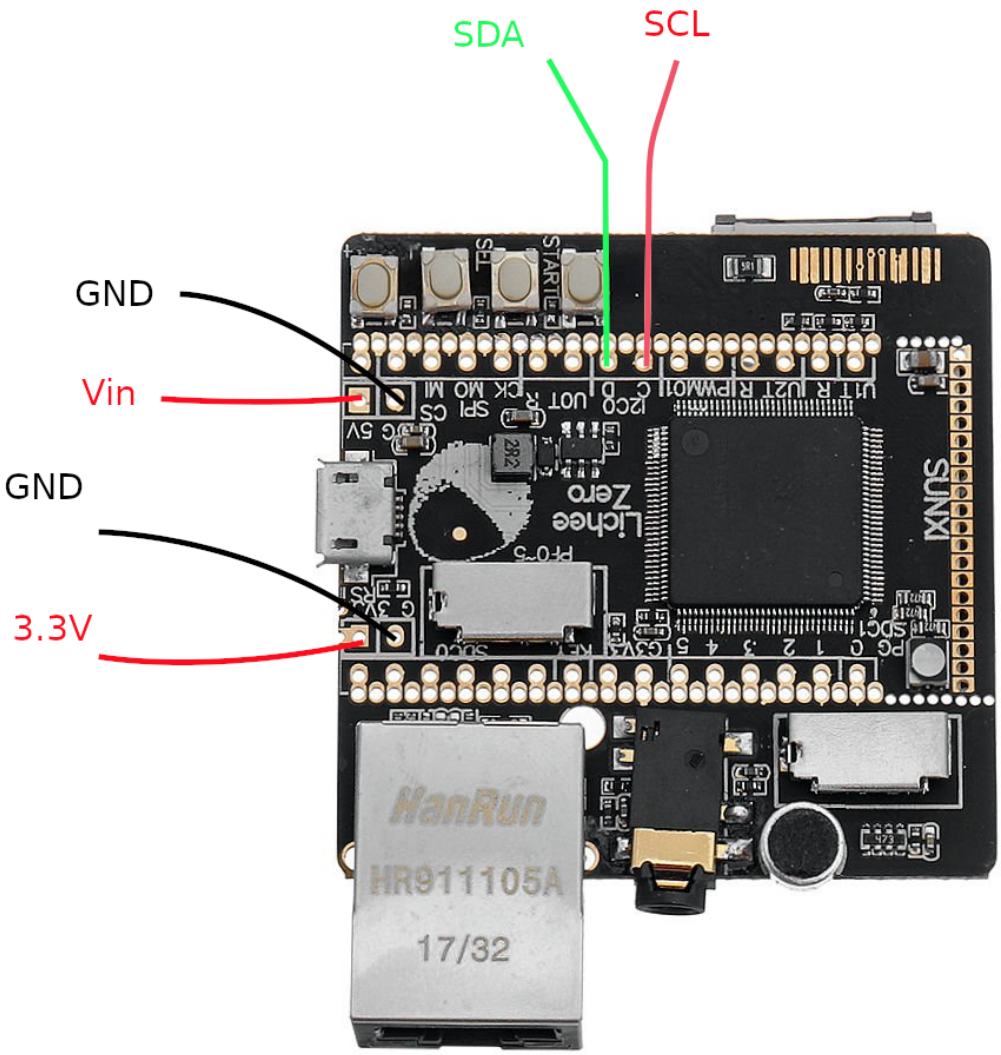


Figure5: Dimension Moteur NEMA Standard

8.2.3 Connexions électroniques

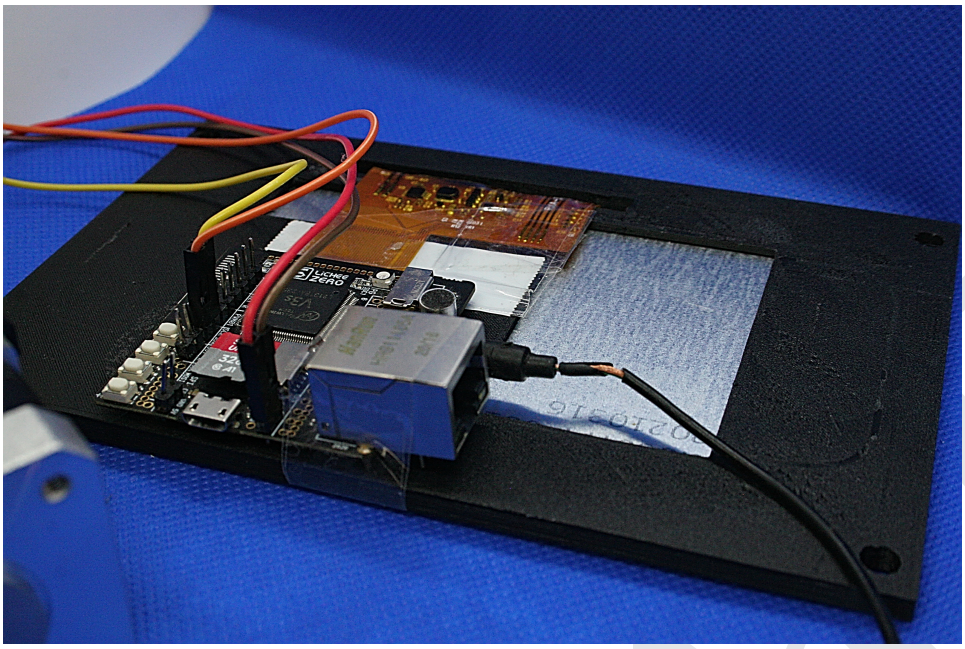
La carte principal est connectée aux différents élément comme suit :

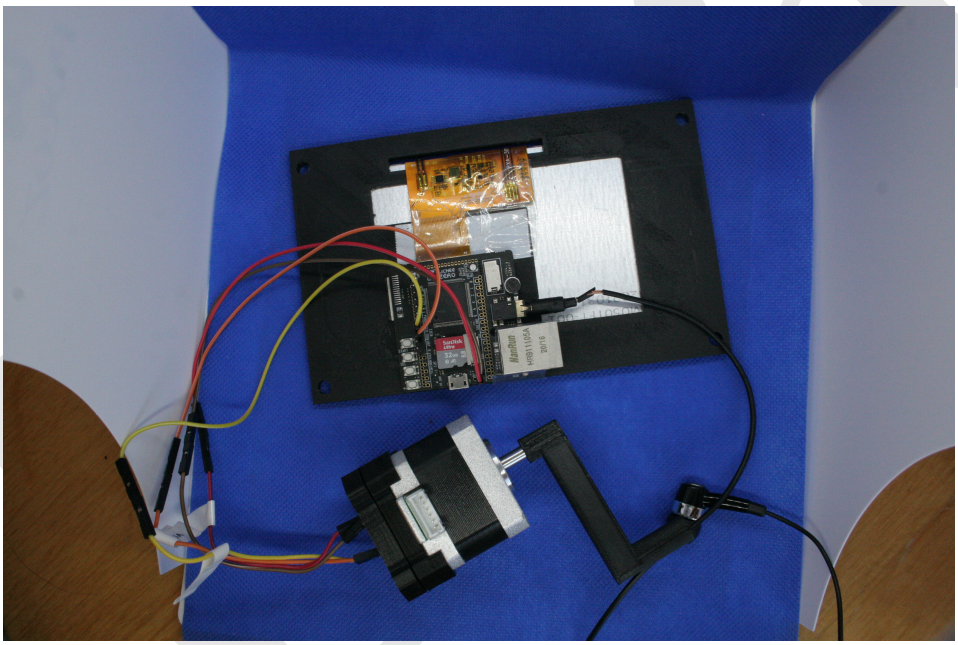


La connexion SDA/SCL/3V et GND est en liaison avec le capteur de rotation de la manivelle.

La connexion Vin / GND est soudée sur l'alimentation

Avertissement: Attention, à ne pas dépasser 5V pour l'alimentation, même si la carte possède un regulateur, celui ci ne pourra dissiper un large différence de tension. Le module peut être déterioré avec une mauvaise alimentation.





CHAPTER 9

Tables et Indexes

- genindex
- modindex
- search