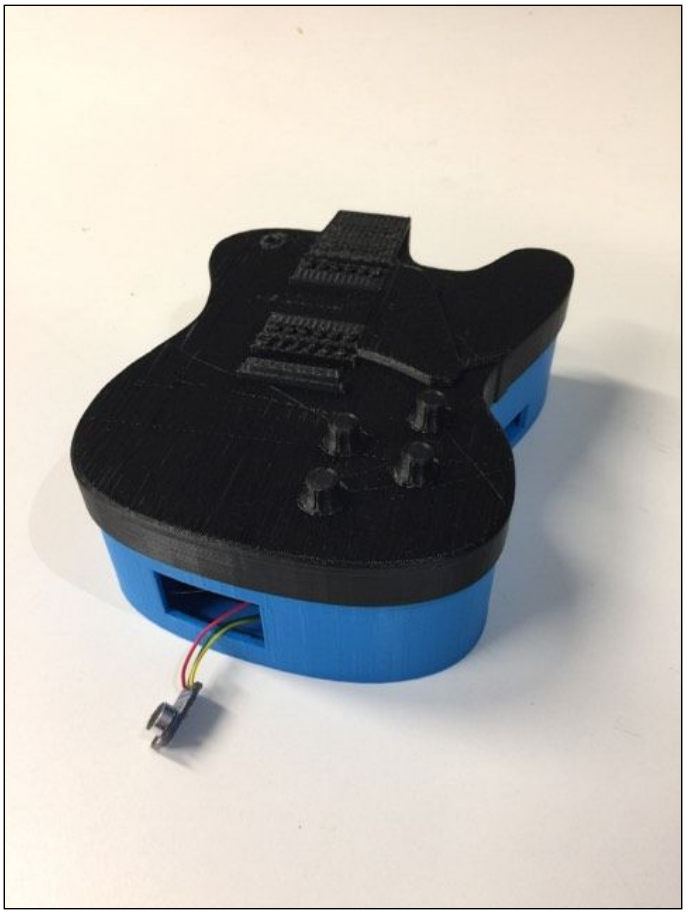
POLYTECH LILLE

Université de Lille

Ecriture automatique d'une partition musicale



Projet 73 - IMA4

Fabien RONCKIER - Hugo LEURENT

Remerciements

Nous souhaiterions tout d'abord remercier nos tuteurs de projet Monsieur Xavier Redon, Monsieur Alexandre Boé et Monsieur Thomas Vantroys pour leur aide, leur disponibilité et la mise à disposition du matériel tout au long de ce projet. Leurs conseils et explications nous ont permis d'avoir un regard nouveau sur bon nombre de situations périlleuses.

Un grand merci également à Monsieur Thierry Flamen, responsable des salles EEI, qui s'est toujours montré aimable et accueillant lorsque nous allions le harceler en quête de conseils ou de composants. Sa disponibilité et ses explications nous ont permis d'obtenir une carte électronique et des soudures de qualité.

Merci à Laurent Engels d'avoir pu se rendre disponible pour filmer notre vidéo de fin de projet malgré l'heure tardive.

Enfin, nous voudrions remercier plus globalement les enseignants du département IMA, toujours disponibles pour répondre à la moindre de nos questions et enrichir nos connaissances (Monsieur Defrance, Monsieur Carré, Monsieur Forget, Monsieur Conrard, Madame Pichonnat...). Merci également à nos camarades de promotion IMA4 qui se sont toujours montrés disposés à nous apporter leur aide.

Sommaire

Introduction

Dans le cadre de notre deuxième année de cycle ingénieur en Informatique, Microélectronique et Automatique, nous avons l'occasion de réaliser un projet nous permettant de mettre en oeuvre nos connaissances et d'en découvrir de nouvelles.

Le projet que nous avons décidé de mettre en place est un procédé de traitement de signal permettant de récupérer informatiquement les notes et rythmes jouées depuis une guitare électrique. Ces données sont ensuite automatiquement traduites en une partition de musique. L'idée de ce projet nous est venue déjà bien avant notre entrée en IMA 4. En effet, étant passionnés de musique, nous avons eu l'idée d'associer notre passion à ce projet pour tenter d'apporter quelque chose qui pourrait s'avérer utile à l'avenir. Ce projet entre ainsi dans une volonté de faciliter la vie de tout musicien en herbe souhaitant un jour écrire ses propres morceaux. Grâce à ce projet, nous espérons que composer une musique deviendra aussi simple que de brancher sa guitare.

Au fil de ce rapport, nous rappelerons dans un premier temps les objectifs du projet et le positionnement par rapport à l'existant. Dans un deuxième temps, nous détaillerons chaque partie du projet réalisé, avant de prendre du recul sur ce qui a été accompli et ce qui pourrait être amélioré dans une troisième partie.

I. Présentation du projet

a. Description

Le but de ce projet est le suivant : nous voulons traiter les signaux envoyés par un instrument de musique, et plus particulièrement par une guitare, afin d'extraire chaque note jouée par l'utilisateur. Ainsi, ces notes pourront ensuite servir à la construction d'une partition musicale, qui correspond bien à la mélodie jouée par l'instrument.

Pour ce faire, nous avons conçu une partie électronique composée d'une carte et d'une Nucleo. La carte nous permet de récupérer des signaux propres et amplifiés, afin que la Nucleo puisse extraire la fréquence de chaque note grâce à un algorithme de FFT. Ces fréquences sont ensuite envoyées vers un logiciel qui permet de leur associer une position sur la partition, et donc de leur associer une note.

b. Etat de l'art

Avant de mettre en oeuvre un projet comme celui-ci, une phase de recherche et d'analyse était nécessaire. Ainsi, en suivant les conseils de nos tuteurs, nous avons cherché à mettre en évidence les points les plus cruciaux pour mener à bien notre travail, mais également s'informer sur les inventions existantes présentant des fonctionnalités similaires.

Il est évident que tout comme notre projet, il existe de nombreux logiciels d'écriture de partition, tels Guitar Pro ou Musescore. Cependant, parmi ces logiciels, très peu ont la capacité de l'écrire automatiquement directement depuis un instrument. Ces logiciels sont en général utilisés pour éditer à la main une partition. C'est donc en ce point que notre projet pourrait se distinguer de ces programmes.

Si on voulait trouver un principe se rapprochant de notre projet, ce serait celui de Rocksmith. Ce jeu vidéo utilise un câble qui relie directement la guitare au PC pour comprendre les notes et accords joués par l'utilisateur. Ce jeu est d'ailleurs l'une des sources d'inspiration pour notre projet.

Avant d'effectuer toute cette phase de recherche sur les logiciels et autres fonctionnalités existantes, nous pensions notre projet complètement novateur dans le sens où nous ne connaissions encore rien de tel. Cependant, nous sommes tombés sur AudioScore, un logiciel qui réalise exactement ce que nous avons voulu accomplir, à savoir transcrire une partition depuis un enregistrement. Bien que ce logiciel soit certainement plus avancé que ce que nous accomplirons, il coûte aussi un certain prix. Notre logiciel aura l'avantage d'être disponible gratuitement à tout un chacun sur le wiki IMA !

c. Cahier des charges

L'objectif global du projet est de pouvoir afficher les notes jouées par un utilisateur sur un logiciel. Nous souhaitons également que la capture s'exécute quasiment en temps réel, c'est à dire qu'il y ait le moins de délai possible entre l'envoi d'une note par l'instrument et son affichage sur la partition.

Pour permettre cela, nous avons décomposé notre projet en plusieurs objectifs :

- Créer une carte électronique permettant d'extraire des signaux en sortie d'un instrument (ici une guitare), et de les rendre exploitables pour un traitement du signal efficace.
- Implémenter une partie traitement du signal par FFT afin d'extraire les fréquences jouées par l'instrument.
- Concevoir un programme qui associera une note pour chaque fréquence reconnue.
- Créer un logiciel qui affichera ces notes sur la partition finale.

Nous nous sommes également fixé un objectif intermédiaire en la conception d'un accordeur de guitare, afin de fixer les bases de notre partie logicielle.

II. Réalisation

a. La carte électronique

La carte électronique est l'élément à la base de notre projet. C'est en effet elle qui permet d'extraire les signaux issus de l'instrument afin de pouvoir les exploiter de manière optimale par la suite.

A la base, nous avions prévu d'effectuer le traitement du signal par un microcontrôleur intégré à la carte. Cependant, afin de pouvoir effectuer des tests sur des signaux assez rapidement, nous avons préféré ajouter la possibilité de convertir cette carte en "shield". Ce shield de Nucleo nous permet ainsi d'utiliser les signaux filtrés et amplifiés par notre carte, sans aller jusqu'au microcontrôleur. Comme décrit par la suite, on a préféré garder ce shield dans la version finale de notre projet, afin de le garder fonctionnel.

A la fin de la phase de préparation, nous avions déjà une idée globale de ce que à quoi notre carte devrait ressembler. On peut ainsi diviser notre carte en 3 parties distinctes :

- Partie son : c'est grâce à cette partie que l'on va extraire les signaux issus de l'instrument.
- Partie microcontrôleur, qui devait à la base nous servir pour effectuer la FFT.
- Partie alimentation et USB, pour pouvoir communiquer avec l'ordinateur.

Partie son

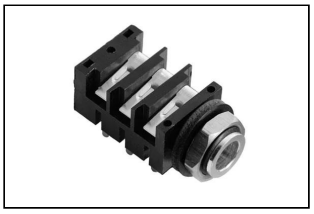
Pour extraire les signaux issus de la guitare, nous somme partis sur deux modes d'acquisition différents : par micro, ou par prise Jack. De nombreux micros comportent des amplificateurs intégrés. Cependant, les prises Jack ressortent des signaux de faible amplitude et donc difficilement exploitable. On a donc décidé d'ajouter un amplificateur en sortie Jack.

De plus, comme nous ne connaissions pas vraiment l'état des signaux en sortie de micro et en sortie du Jack, nous avons préféré ajouter un filtre passe bas,

au cas où les signaux serait trop bruités. Comme nous n'avons besoin que d'utiliser la fréquence fondamentale, nous nous servons également de ce filtre pour supprimer des harmoniques inutiles qui pourraient perturber notre FFT.

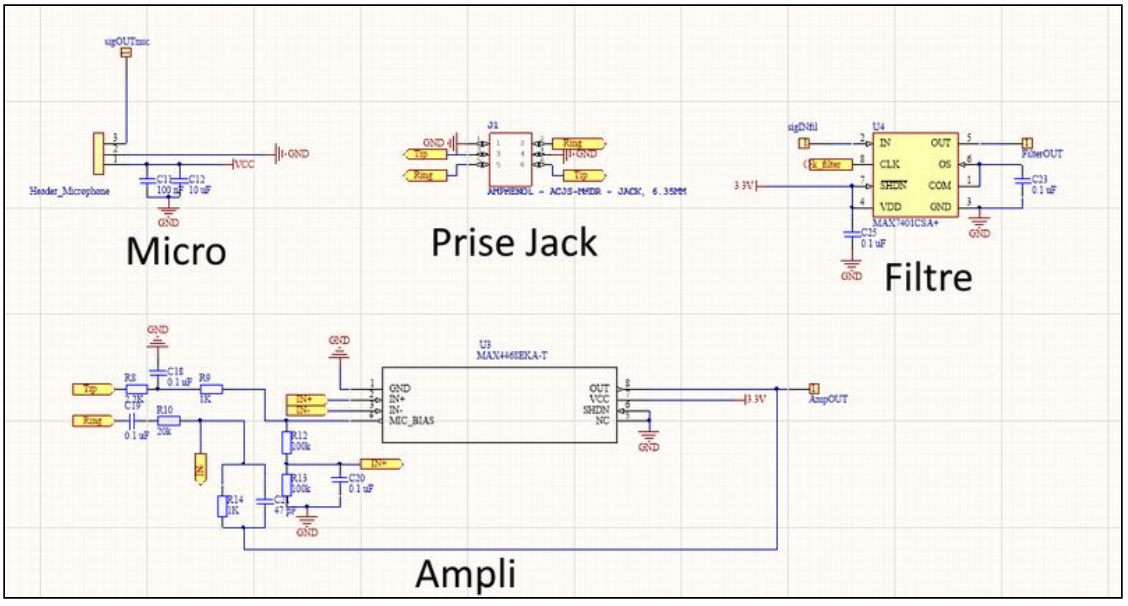
Par chance, Monsieur Redon avait déjà le micro que avons choisi dans notre liste de matériel, qui s'apparente à ceux trouvé dans les kits Arduino par exemple. L'avantage est que celui-ci comporte un ampli intégré, et qu'il peut être relié à notre carte par des câbles.

Peu de documentations sont disponibles concernant les prises Jack 6.35mm sur le web. Nous avons donc commandé une prise standard et avons nous même réalisé son empreinte sur Altium.



Après diverses recherches de schematic de projets similaires en rapport avec le son, nous avons choisi notre filtre et notre ampli. Ce sont respectivement le MAX7401 et un MAX4468, qui sont tous deux assez souvent utilisé pour ce genre de traitement.

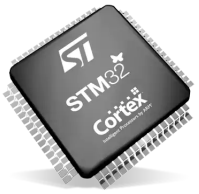
Une fois les composants choisis, nous avons pu commencer à établir le câblage de notre circuit. Pour cela, nous avons dans un premier temps fait un gros travail de recherche sur les montages existants. Ensuite, avec l'aide de M. Flamen et M. Boé, nous avons affiné ce circuit afin de limiter les éventuelles dégradations pouvant altérer nos signaux. Nous sommes ainsi arrivés au schéma de câblage suivant :



Ici, il faut prendre en compte qu'on aura un seul signal en sortie de notre partie son : le signal filtré. Ce signal en sortie de filtre provient directement du micro, ou alors il provient de l'ampli, relié à notre jack. Pour effectuer le choix entre prise Jack ou micro, nous avons décidé de relier nos sorties à des headers, afin de pouvoir choisir facilement quel signal filtrer.

La sortie du filtre est également relié à un header. En effet, en accord avec les conseils de M. Boé, cela nous permettra d'utiliser le signal sur la Nucleo, dans le cas où notre microcontrôleur n'est pas utilisable.

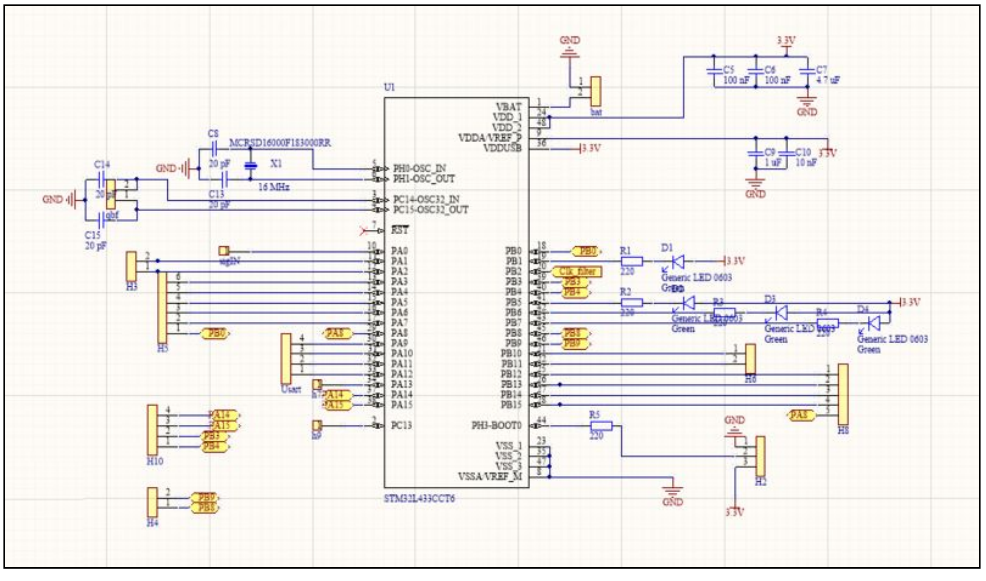
Partie Microcontrôleur



Pour le cas où le signal devait être traité sur la carte, il nous fallait un microcontrôleur capable d'effectuer une FFT. Après quelques recherches et concertation avec M. Boé, nous sommes parti sur un microcontrôleur du type STM32, celui-ci présentant l'avantage de posséder des librairie FFT.

Après de plus ample recherches, nous pensions utiliser un STM32F411RET6, qui semblait assez pratique car programmable à partir d'un port USB, et souvent utilisé pour des projets de conversion analogique-numérique. Cependant, M. Redon possédant déjà un microcontrôleur similaire (STM32L433CCT6), nous avons préféré intégrer ce dernier à notre circuit.

Après quelques modifications en s'aidant de la datasheet du microcontrôleur, nous sommes arrivé au schéma de circuit suivant :



Sur le schéma, et suite aux conseils de nos encadrants, nous avons préféré relier chaque PIN non utilisable à des headers, aux cas où l'un de ceux-ci deviendrait utile par la suite. Nous avons également relié deux Quartz (1 haute fréquence et 1 basse fréquence), même si il n'est normalement pas prévu d'utiliser le quartz basse fréquence. Tous ces ajouts non nécessaires pour le moment, au même titre que nos LEDs, devraient nous être utile pour déboguer facilement notre carte en cas de dysfonctionnement. Au niveau de la conception en elle-même, cela n'ajoute pas vraiment de complexité.

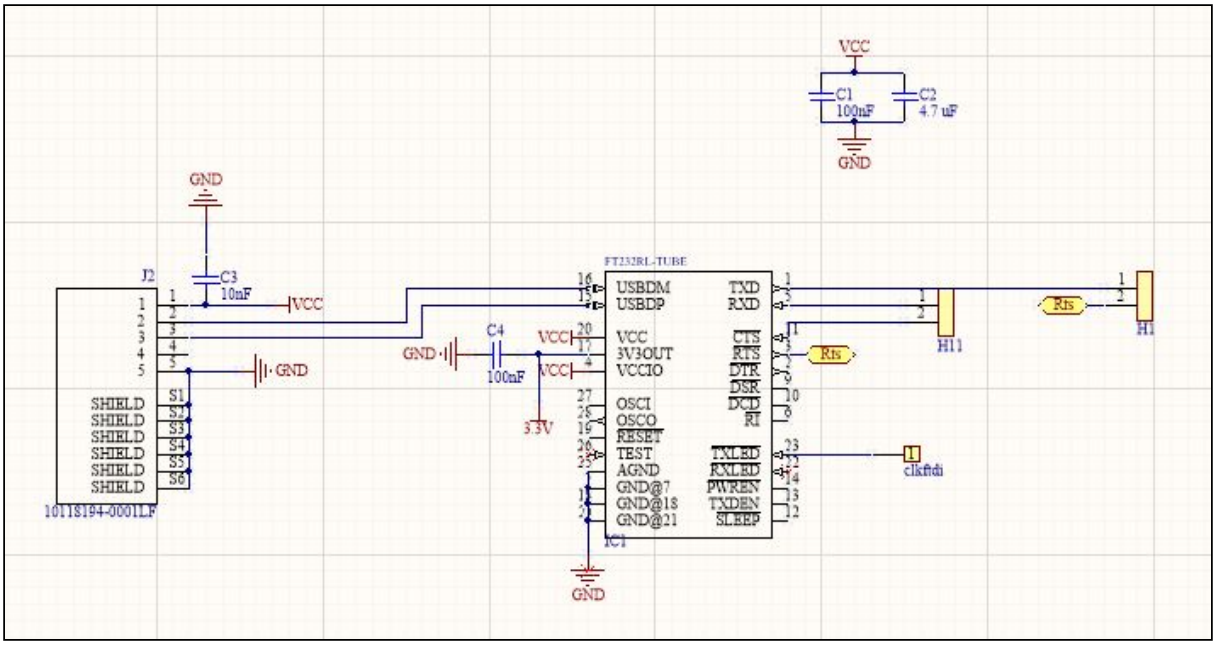
Partie alimentation

Au départ, nous hésitions entre une communication série avec l'ordinateur, et une communication directement via USB. En effet, nous souhaitions dans un premier temps passer par une communication via un ATMega16u2 pour réaliser une conversion USB-Série, afin de se rapprocher de ce que nous avions vu en tutorat système au semestre précédent, avec la conception d'une manette de jeu. Cependant, comme notre microcontrôleur peut directement communiquer via USB, nous avons décidé de supprimer cette partie 16u2.

Retournement de situation ! Les documentations sur la partie communication via USB étant assez faibles, et celles concernant l'utilisation de l'UART étant plus riche en informations, nous sommes finalement partis sur une communication série.

Après réflexion, nous avons décidé d'utiliser un FTDI. En effet, étant donné que la majorité que notre circuit est alimenté en 3.3V, dans le cas de l'utilisation d'une 16u2, il aurait fallu ajouter un Voltage Regulator. Notre FTDI comporte quant à lui une sortie 3.3V et est alimenté en 5V, soit directement via l'USB. En plus de se charger de la communication Série-USB, c'est donc lui qui jouera le rôle de Voltage Regulator pour alimenter notre circuit.

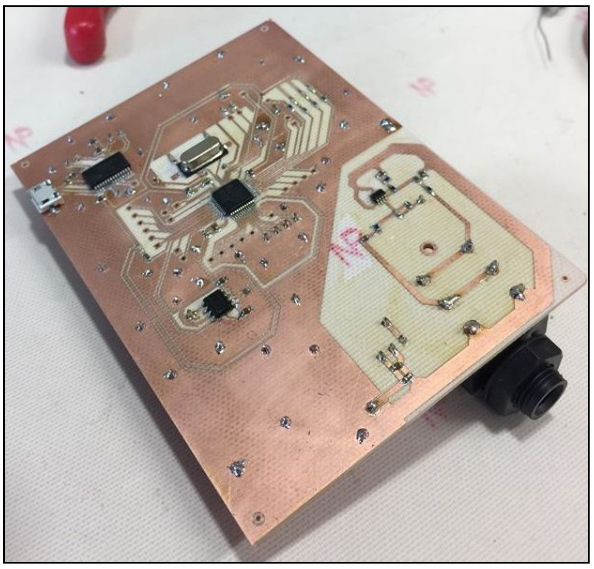
Nous avons ainsi pu établir la dernière partie de notre circuit :



Une fois nos schéma établis, il ne nous restait plus qu'à router et souder notre carte. Pour cette partie, nous remercions encore une fois M. Flamen et M. Boé pour leurs précieux conseils.

Lors de la conception de notre carte, et ne maîtrisant pas totalement toutes les technique de routage, nous avons préféré essayer de mettre le maximum d'élément sur une seule couche. La partie routage est assez rapide une fois les principes de bases maîtrisés. Ainsi, malgré quelques débuts difficiles nous avons su terminer le PCB sans encombres (voir Figure 1 de l'annexe). Finalement, notre carte comporte 2 faces, mais la face basse est essentiellement constitué d'une masse commune afin de disposer d'un plan de masse uniforme. C'est également la raison pour laquelle nous avons tant de via.

Après cela, nous avons rapidement pu souder les composants sur la carte (voir ci-contre). Ici, on a pu se rendre compte de l'importance de prévoir des pins de secours. En effet, nous nous sommes rendus compte un peu tard que notre ampli n'avait pas été relié à l'alim sur notre PCB. On a donc eu quelques difficultés à souder celui-ci à la piste, ce qui aurait pu être évité si nous avions été plus vigilants.



Notre carte imprimée et soudée

Mis à part cela, après avoir vérifié que tout était bien connecté, nous avons pu commencer à effectuer quelques essais avec le micro, avant de la connecter à la Nucleo. Comme expliqué précédemment, nous avons fait le choix de laisser de côté la partie microcontrôleur de cette carte, par manque de temps.

Nous avons également réalisé d'autres montages tests à côté, notamment un montage sur breadboard d'un Jack associé à un ampli. Il nous a permis d'entamer des essais de capture de notes avant d'avoir notre carte soudée entre les mains, et nous aurait servi de solution de secours dans le cas d'un dysfonctionnement de la partie son de la carte.

b. La carte Nucleo

La carte Nucleo dotée d'un microcontrôleur STM32F401 nous était prêtée au départ pour faire de simples tests et nous servir en cas de non-fonctionnement de la carte électronique. Cependant, elle s'est finalement retrouvée dans notre version finale du projet, n'ayant pas eu le temps d'implanter un code dans notre carte principale.

Son rôle est donc de remplacer le microcontrôleur de la carte électronique. Nous souhaitons l'utiliser pour réaliser les fonctions de traitement de signal qui nous permettent de renvoyer la fréquence fondamentale de la note vers la liaison série.

Nous en connaissions peu sur la programmation de cartes Nucleo. Nous savions bien sûr qu'elles étaient à programmer en langage C, mais sa méthode de programmation s'avère bien différente d'un programme C classique ou d'un programme d'Arduino, par exemple. Après avoir demandé des conseils et suivi quelques tutoriels, nous avons vite pu implémenter nos premiers programmes, et nous atteler aux choses sérieuses.

Le programme final est issu d'un programme d'accordeur réalisé par un membre de la communauté. Il faut savoir que la programmation de la carte se fait sur l'application en ligne Mbed, qui comporte notamment une communauté de codeurs qui mettent à disposition leurs réalisations. Ainsi, nous sommes partis d'un programme de conversion analogique-numérique puis numérique analogique pour comprendre et appréhender la bibliothèque de FFT. Le programme s'avère finalement assez court, malgré plusieurs séances de recherches.

La bibliothèque FFT s'est révélée assez simple à comprendre et à utiliser. De plus, elle nous donne des résultats très fiable, comme nous avons pu le constater lors de nos différents tests. Il nous suffit de modifier une ou deux valeurs pour facilement varier la précision de la capture. En stabilisant la valeur de la fréquence d'échantillonnage à 3kHz, nous pouvons jouer sur la longueur du tableau qui stockera les valeurs de fréquences. Ainsi, nous pouvons aller jusqu'à stocker 4096 valeurs dans un tableau, et donc avoir une précision de 3000/4096 = 0.73 Hz.

Le principe de ce tableau est que pour chaque indice, il contient la valeur d'amplitude de la fréquence captée. Ainsi, il suffit ensuite de récupérer l'indice de la case possédant la valeur d'amplitude la plus haute pour connaître l'emplacement de la fréquence fondamentale sur le spectre. Nous avons constaté un souci pour les notes les plus graves de la guitare : la fréquence fondamentale

affichée était deux fois supérieure à la réelle fréquence. Techniquement, cela ne change pas la note, mais la considère comme plus aigüe d'un octave, ce qui posera des problèmes au moment de l'écriture de la partition.

Nous avons constaté, assez tardivement malheureusement, que, bien que précis, un tableau trop grand nous handicapait du fait de sa lenteur à se remplir et à envoyer ses données vers le PC. Pour l'acquisition de la partition, il faut être le plus proche possible des rythmes voulus, et ainsi recevoir des informations le plus souvent possible. Avec un tableau de 1024 éléments, nous nous rapprochons d'un débit d'envoi correct sans perdre trop de précision.

```
int main() {

    pc.printf("Debut de programme\n");
    //arm_cfft_instance_f32 S;   // ARM CFFT module
    float maxValue;            // Max FFT value is stored here
    uint32_t maxIndex;         // Index in Output array where max value is
    pc.baud(9600);
    pc.printf("Starting FFT\r\n");
    while(1) {
            timer.attach_us(&sample,333); //333µs = 3kHz sampling rate
            for (int i = 0; i < SAMPLES; i += 2) {
                while (trig==0){}
                trig=0;
                Input[i] = myADC.read()- 0.5f; //Real part NB removing DC offset
                Input[i + 1] = 0;              //Imaginary Part set to zero
                }
            timer.detach();
        // Init the Complex FFT module, intFlag = 0, doBitReverse = 1
        //NB using predefined arm_cfft_sR_f32_lenXXX, in this case XXX is 1024
        arm_cfft_f32(&arm_cfft_sR_f32_len1024, Input, 0, 1);

        // Complex Magniture Module put results into Output(Half size of the Input)
        arm_cmplx_mag_f32(Input, Output, FFT_SIZE);

        //Calculates maxValue and returns corresponding value
        arm_max_f32(Output, FFT_SIZE, &maxValue, &maxIndex);

        float freq_recherchee = maxIndex*(3000/1024.0);
        if (maxValue > 10 && freq_recherchee < 2000 && freq_recherchee > 50){
            pc.printf("%f\r\n",freq_recherchee);
```

Fonction main de notre programme d'envoi de fréquence fondamentale

Comme l'accordeur et la partition nécessitent deux caractéristiques différentes (l'un privilégie la précision, l'autre la vitesse), nous avons conclu qu'il serait mieux de proposer deux programmes, chacun optimisé pour une tâche. Nous sommes conscients qu'il peut être embêtant de changer de programme à

téléverser après avoir accordé son instrument, mais il faut savoir que, si besoin, l'autre programme peut faire l'affaire, il sera juste moins performant.

Justement, détaillons maintenant la conception et les fonctionnalités de ces fameux modules d'accordage et d'écriture de partition dans la partie suivante, consacrée au logiciel.

c. Le logiciel

Le logiciel est bien sûr une partie fondamentale de notre projet. Il permet de servir d'interface homme-machine et va interpréter les données envoyées par la carte Nucleo en informations compréhensibles par tout un chacun.

Nous avons choisi de le programmer en langage Java. Ayant eu l'occasion d'en étudier les bases au cours de ce semestre en filière SC, nous nous sommes dit qu'il s'agissait là d'un bon moyen d'approfondir nos connaissances et de mettre à profit celles que nous avions acquises.

Effectivement, comme nous allons le préciser dans chacune des sous-parties à suivre, ce logiciel nous a donné l'occasion d'apprendre non seulement à concevoir une interface graphique, mais aussi à ouvrir un port série en Java, en exploiter les données, naviguer entre les images et les superposer, faire une capture d'écran… En bref, explorer une partie de l'océan de fonctionnalités proposé par ce langage.

On peut voir à quoi ressemble l'interface graphique en figure 2 de l'annexe.

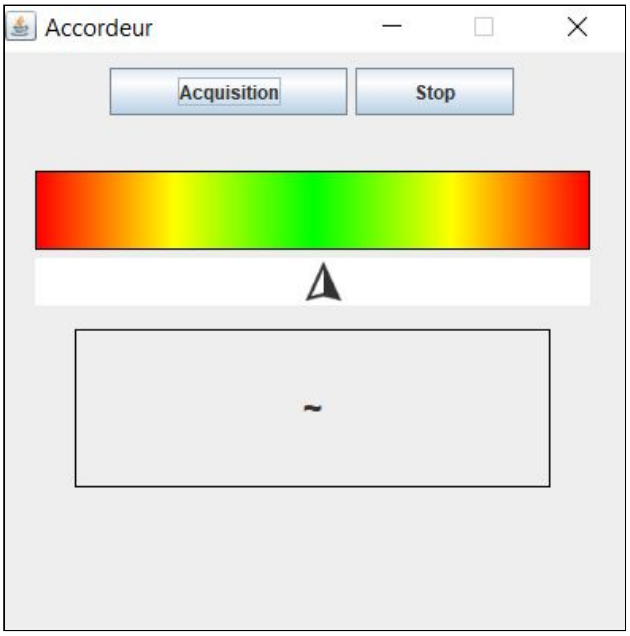
L'acquisition des ports

Comme on peut le constater sur l'interface, le logiciel dispose d'un menu permettant de sélectionner le port série voulu. Cela signifie que nous avons dû en premier lieu implanter une fonction de reconnaissance des appareils connectés aux ports COM. La détection se fait au lancement de l'application. Cela fait qu'elle peut parfois prendre un certain temps à se lancer (quelques secondes) et qu'on

ne peut pas connecter notre carte et espérer qu'elle soit reconnue une fois l'application lancée.

Si on essaie de lancer l'acquisition du signal pour l'accordeur ou la partition alors qu'aucun port n'a été sélectionné au préalable, un message d'erreur s'affiche.

L'accordeur



Fenêtre de l'accordeur

Un bouton permet d'accéder à l'accordeur. Comme précisé sur le Wiki, c'est la première fonctionnalité que nous avons implanté sur ce logiciel, afin de nous familiariser avec les fonctions d'interface graphique et de récupération série. Lorsque l'on clique sur "acquisition", le port COM choisit ouvre l'accès aux données transmises. Là, on récupère la fréquence que l'on compare à la note la plus proche. Elle s'affiche en vert si on est à +-2 Hz, on se considère alors comme accordé. La note s'affiche en rouge sinon. Comme expliqué dans la partie Nucleo, la réception en série ne se fait ici pas très rapidement (environ une note toutes les 500-600ms) mais la fréquence reçue est précise.

Il est important d'appuyer sur le bouton "stop" avant de quitter la fenêtre, puisque c'est ce bouton qui fermera la communication. Il faut savoir qu'un seul processus peut demander l'accès au port COM à la fois, il est donc important de le refermer une fois son utilisation terminée.

Le métronome

Un autre point sur lequel nous nous étions penché plus tôt dans le projet était la réalisation d'un métronome. Notre premier métronome n'était pas fiable puisqu'il affichait les temps avec un drôle de décalage inexpliqué. Ici, au lieu de peindre des cercles, action qui était sûrement la cause du décalage, nous passons à un métronome sonore.

Bien que nous retrouvons des similitudes et utilisons certaines fonctions du premier métronome que nous avions créé il y a quelques semaines, nous devons nous attaquer à une nouvelle tâche : générer un son pour chaque temps. Après s'être renseigné sur l'utilisation de fichiers audio, nous trouvons une solution bien plus simple et plus efficace sur internet, dont nous nous inspirons : le package *javax.sound.midi*. Il possède les méthodes permettant de générer un son MIDI directement, parmi une grande banque de données. Parfait donc pour l'application que nous souhaitons en faire.

Pour continuer à entendre le métronome en jouant notre morceau, nous l'avons fait s'exécuter dans un thread. C'est alors l'occasion, après les avoir étudié en C, d'apprendre à se servir des thread en Java.

La partition

Voici le plus gros morceau du logiciel. Remplir cette partition est la partie qui a pris le plus de temps à programmer. Il nous a fallu trouver le moyen de superposer des images à cette partition, placer chaque note au bon endroit, combiner ceci avec une captation via le port COM, ajouter différents rythmes... Programmer cette acquisition était une partie importante de notre projet, une quête dans laquelle nous avons mis du temps à nous lancer, par peur de l'échec principalement...

Finalement, le résultat obtenu est loin d'être désastreux. Lorsque nous lançons l'acquisition, le programme calcule la durée correspondant à un temps (grâce à la valeur entrée dans le métronome) pour venir la comparer à la durée de

la note jouée et en déterminer le rythme. Ensuite, on placera correctement l'image sur la partition en fonction de la hauteur de la note.

Des problèmes se sont posés au cours de la programmation, dont certains restent encore non résolus. Principalement, le programme ne peut pas reconnaître deux fois la même note jouée d'affilée, puisque nous n'avons pas intégré la reconnaissance des amplitudes. Ainsi, tant qu'il reconnaît la même fréquence, il continuera d'attendre qu'une note de fréquence différente soit jouée pour évaluer le rythme correspondant. Pour résoudre ce problème, il faudrait dans notre message envoyer l'amplitude du signal. Nous pourrions ainsi reconnaître quand la même note est rejouée si la nouvelle amplitude reçue est plus importante que la valeur précédente.

La deuxième amélioration dont pourrait se pourvoir notre programme est l'ajout des silences. En effet, une partition est composée de notes mais aussi de silences, qui permettent de préciser pendant de temps l'utilisateur ne joue pas. Ici nous n'avons pas eu le temps d'intégrer les silences à notre logiciel.

Enfin, nous avions tenté de rajouter plus d'octaves sur notre programme et ainsi pouvoir voir plus de notes sur notre partition, mais l'ajout de cette fonctionnalité l'a révélée assez dysfonctionnelle et faisait pire que mieux au moment d'afficher nos notes. Nous l'avons alors désactivée pour l'instant.

Un bouton permet en plus de recréer une nouvelle partition vide. Pour cela il faut que l'acquisition soit terminée (donc que l'on ait appuyé sur le bouton "stop").

La sauvegarde de la partition

Enfin, un dernier bouton permet une sauvegarde de la partition. Le programme va en réalité réaliser une capture d'écran de la zone de la partition puis la sauvegarder dans un fichier. Cela est possible grâce à un autre package que nous avons découvert : *java.awt.Robot*. Nous voulions au départ faire une sauvegarde de l'image contenue dans le panneau de la partition, mais étonnamment l'image enregistrée ne comportait que la partition et la dernière

note jouée. C'est pour cela que nous avons préféré opter pour la capture d'écran. Méthode certainement un peu moins réglementaire que l'autre, mais fonctionnelle. Seul bémol, il a fallu régler la capture de l'écran au pixel près sur notre ordinateur, et donc cette fonctionnalité risque de ne pas capturer correctement la partition sur une autre machine, dont la taille de l'écran serait différente du nôtre.

Pour conclure cette deuxième partie, nous avons expliqué en quoi notre projet avait eu l'occasion de couvrir une multitude de sujets d'IMA, et comment nous l'avons mené à bien. Dans un troisième temps, nous allons maintenant nous consacrer à une rétrospective sur la réalisation de notre projet. Nous allons expliquer notre organisation au sein de ce semestre en mettant en avant ce que nous avons accompli, mais également ce qui nous a fait défaut.

III. Rétrospective

a. Organisation au cours du semestre

Tout au long de ce huitième semestre, et même depuis la fin du S7, ce projet est resté notre source majeure d'attention et de préoccupations. Motivés dès Décembre, nous avions demandé des conseils pour notre carte à Mr.Boé avant les vacances de Noël, pour entamer notre schematic durant les deux semaines à suivre. Pourtant, une fois arrivés au S8, l'organisation et la planification nous ont fait défaut. Présents, bien sûr, à toutes les séances de projet, notre travail de la semaine s'arrêtait le mercredi à 18h et nous ne nous remettions pas dedans avant le mercredi suivant. Peu efficaces en début de semestre, nous n'avions pas mesuré toute l'ampleur du travail qui nous attendait et sommes tombés dans le piège du "on a le temps".

7 semaines sur un schematic, avec tout ce qui nous reste à faire derrière, c'est trop. C'est évident et nous en sommes conscients. Le temps d'apprivoiser le logiciel Altium et de concevoir notre carte en partant de rien est passé vite, et nous avons eu du mal à évoluer rapidement. C'est à peu près à mi-projet et au moment où Mr.Redon nous a conseillé de mettre un coup d'accélérateur que nous avons pu passer à la vitesse supérieure. Le schematic était enfin terminé et

nous avons alors pu nous répartir de nouvelles tâches pour travailler en parallèle et optimiser notre temps.

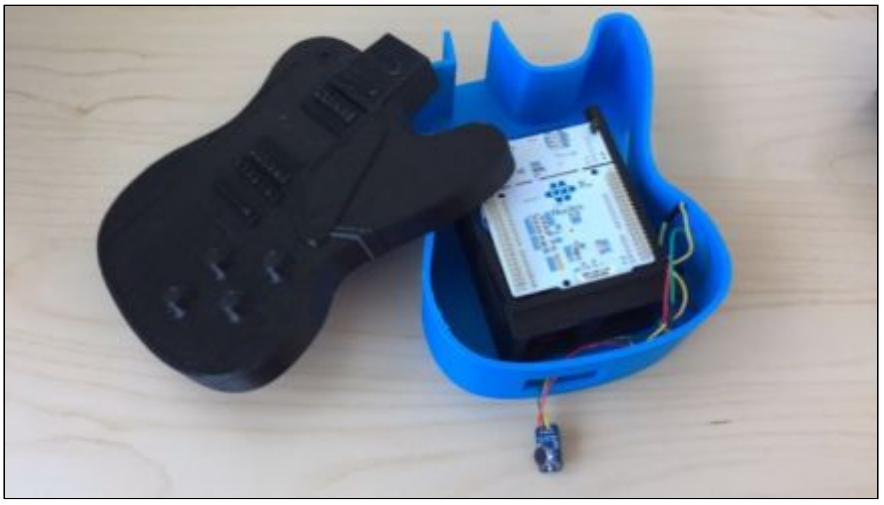
Effectivement, pendant les 5 premières semaines, nous avions choisi de travailler à deux sur le schematic. Pouvant paraître comme une perte de temps au premier abord, nous avons décidé de travailler ainsi car la conception d'une carte aussi remplie était nouveau pour nous et nous avions besoin de nos deux visions de la carte pour arriver à bout de ce schematic. Ainsi, lorsque l'un ne comprenait pas pourquoi faire d'une certaine façon où l'intérêt d'un composant, l'autre pouvait lui expliquer. Cette façon de travailler nous a permis d'éviter, nous pensons, bon nombre d'erreurs d'inattention sur la carte, et de bien avoir en tête la façon dont nous procéderions à la capture des notes.

Dans un deuxième temps, nous avons fait le choix de nous séparer les tâches. Alors que l'un s'occupait de concevoir tout le PCB, l'autre faisait des recherches sur la carte Nucleo, et lorsque l'un concevait le boîtier 3D, l'autre réalisait le logiciel. Bien sûr, chacun avait toujours un oeil sur ce que l'autre faisait et n'hésitait pas à donner son avis ou à demander à comprendre certains points du projet. Lorsque l'un faisait face à un obstacle, nous nous y mettions à deux pour le franchir. C'est notamment à deux que nous avons pu déboguer et comprendre le fonctionnement du programme à implanter sur la Nucleo.

Nous avons finalement passé énormément d'heures pour arriver au bout de ce projet, plus d'une centaine si on en croit la feuille d'heures (sans compter la la rédaction du wiki et du rapport). Bien que nous soyons fiers du travail accompli, nous sommes conscients que nous aurions dû répartir le travail de façon plus uniforme au sein du semestre pour ne pas avoir à faire plus de 50% des heures dans le dernier mois. C'est une leçon que nous devrons garder précieusement en tête lors de l'organisation de projets futurs.

b. Etat du projet et amélioration possibles

En résumé, notre projet se compose donc à l'heure actuelle d'un boîtier imprimé en 3D, de deux cartes électroniques (une Nucleo et une conçue par nos soins), et d'un logiciel Java. Nous utilisons notre carte électronique pour capter les sons soit depuis un micro, soit directement en prise Jack, que nous envoyons sur le port A0 de la carte Nucleo. Le programme implanté dans le microcontrôleur STM32F401 réalise une FFT qui renvoie la fréquence fondamentale vers le logiciel, qui se charge ensuite de placer les notes sur une partition. Comparativement au cahier des charges, la mission est remplie. Chacune des trois principales parties du projet est réalisée et sont ensuite mises en lien afin d'obtenir le résultat final.



Boîtier final comprenant la carte Nucleo, et notre carte électronique en dessous

Pourtant, des améliorations sont bien évidemment possibles. Déjà, l'utilisation de la Nucleo n'était pas prévue au départ et devait servir de plan B. La raison est que nous n'avons pas eu le temps d'effectuer tous les tests voulus sur la partie microcontrôleur de notre carte électronique. De ce fait, comme nous avions déjà un programme fonctionnel sur la Nucleo, nous avons fait le choix de n'utiliser que la partie son de notre carte que nous renvoyons directement sur la Nucleo. Ainsi, nous avons passé 7 semaines à faire un schematic et 3 semaines à faire un PCB et souder une carte pour n'utiliser qu'un quart de ses composants. Nous aurions dû partir sur l'idée de la Nucleo dès le début afin de concevoir une carte plus petite, sans perdre tant de temps, et pouvoir ainsi le mettre à profit

ailleurs. Autre solution, nous aurions pu (ou dû) accélérer la phase de conception de schematic pour mettre le temps gagné dans des tests et débugs plus poussés de la carte électronique et ainsi pouvoir la programmer.

Concernant la partie logiciel, la majorité des améliorations possibles a déjà été listée dans la partie II.c. Correction des octaves, ajout des silences et prise en compte de l'amplitude des signaux seraient les principales fonctions à implanter dans une potentielle prochaine version du projet. Il nous faudrait enfin upgrader le microcontrôleur utilisé pour lui permettre de faire ses opérations plus précisément et plus rapidement.

Conclusion

Réaliser ce projet fût pour nous très formateur. Passionnés par la musique et motivés par le sujet, nous avons pris plaisir à voir les différentes pièces du projet s'assembler pour obtenir le résultat final que nous présentons ici. Bien qu'ayant mis du temps à réaliser l'ampleur de la tâche, nous avons comblé ce retard en nous impliquant corps et âme dans la deuxième moitié du projet, pour en obtenir un prototype fonctionnel.

L'état du projet actuel conviendrait parfaitement à un débutant qui souhaiterait s'initier à la guitare et au solfège par la même occasion. Simple d'utilisation, le logiciel est conçu pour être rapide à prendre en main et il y a peu de branchements à réaliser pour l'utilisateur.

Nous sommes heureux du résultat obtenu, majoritairement conforme au cahier des charges établi en début de projet. Bien évidemment, il s'agit d'une première version et des améliorations seraient à prévoir s'il fallait continuer sa conception.

Annexes

Figure 1 : PCB de la carte électronique, faces haut et bas

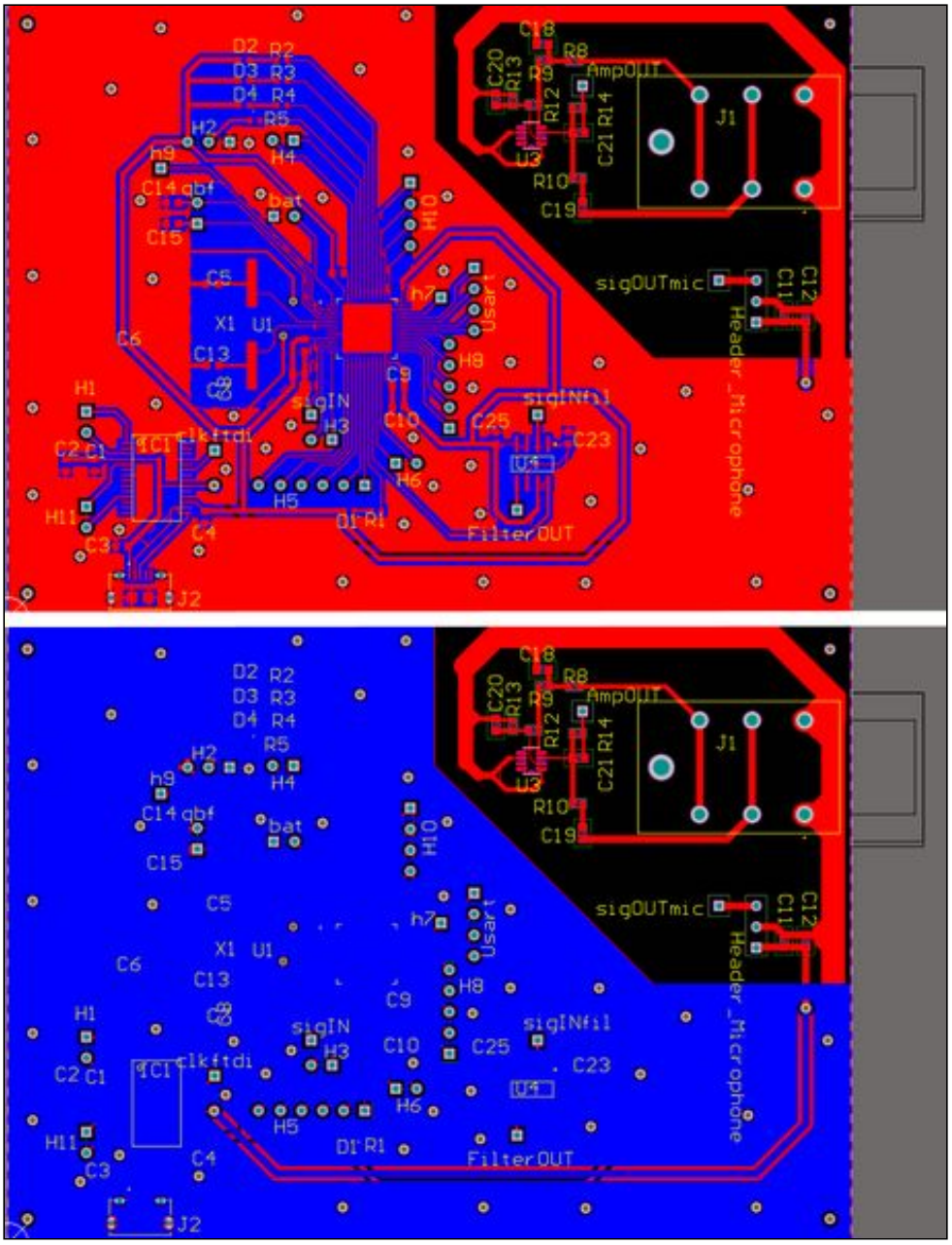


Figure 2 : Interface graphique du logiciel