IMA4 P12 : Ostéophonie

Amine El Messaoudi et Simon Feutrier

May 15, 2018

1 Introduction

La communication osseuse - ou ostéophonie - est un type d'écoute qui permet de ne pas passer par le système auditif externe. Celui-ci n'est donc pas encombré, ce qui offre de nouvelles possibilités d'écoute. Ce dispositif s'adresse aux personnes atteintes de certaines surdités mais également à toutes celles qui travaillent dans un environnement dangereux telles que les militaires ou les ouvriers dans des usines ou des chantiers. En effet, elles peuvent ainsi recevoir des informations tout en restant attentives au bruit environnant. Il peut également être utilisé pour le loisir par les personnes pratiquant la course à pieds, le cyclisme et désirant rester attentives aux dangers de la route.

L'objectif de notre projet était de réaliser un prototype fonctionnel de système de communication par ostéophonie fonctionnelle. Dans cette optique, nous nous sommes concentrés sur la conception d'un casque sur le modèle de ceux utilisés pour la course mais bien moins coûteux. Le casque utilise une communication sans fil Bluetooth pour lui permettre de s'appareiller facilement à la plupart des appareils connectés. Il dispose d'un mode d'écoute directe ainsi que d'un mode qui lui permet de lire des musiques stockées dans le dispositif.

Ce système est pensé pour une consommation en énergie réduite grâce au choix de la communication et du microprocesseur. Enfin, la conception a pris également en compte l'esthétique et le bien-être, en essayant d'obtenir un casque design et agréable à porter.

Table des matières

2 Recherche préliminaire et choix des composants à utiliser

2.1 Recherches préliminaires

Nous nous sommes premièrement attachés à comprendre le fonctionnement global du système que nous allions mettre en place. L'idée première était d'utiliser une montre connectée

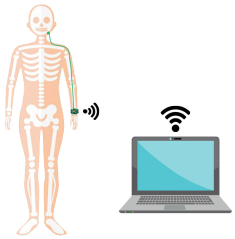


FIGURE 1 : schéma de notre projet.

qui soit reliée de manière filaire à un dispositif d'écoute discret, afin d'être utilisée pour des tours de magie. L'utilisateur aurait reçu des informations d'un compère à distance grâce à une communication sans fil.

Le problème de ce système est qu'il serait difficile d'avoir un système dissimulé si une partie est filaire (de la montre jusqu'à l'oreillette). Il serait donc plus judicieux d'utiliser un système dissimulé directement au niveau du visage. De plus, nous nous sommes rendus compte avec M. Boé que la montre n'avait pas de pins libres afin de connecter une mémoire flash dessus. Il a donc été décidé de ne pas utiliser la montre malgré l'intérêt pédagogique qu'elle aurait pu apporter.

Avec l'accord de M. Boé, nous décidons donc de nous orienter sur un casque d'ostéophonie et non sur un système à dissimuler pour la magie. Nous aurons alors besoin d'un système de communication, de préférence sans fil, d'un système d'écoute ostéophonique, et d'un système de commande à déterminer.

2.2 Transducteurs ostéophoniques

Nous avons utilisé deux types différents de système permettant de convertir le signal électrique en mécanique : le premier est piézoélectrique et le second utilise des bobines.

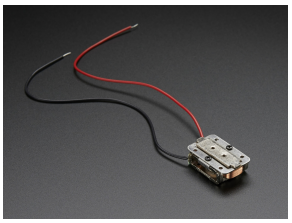
Un système piézoélectrique réagit de manière mécanique à un stimulation électrique et inversement.

Ainsi une lamelle piézoélectrique est constituée d'un disque sur lequel est placé un autre disque plus petit.

On place sur chaque disque une borne d'un système électrique et cette lamelle va se tordre en fonction de la tension qui lui est injectée. Par exemple, elle se pliera vers le haut pour une tension positive et vers le bas pour une tension négative.

Le deuxième type de transducteur osseux que nous avons utilisé est un moteur rotatif. Ce sont de simples moteurs à courant continu disposant d'un axe vertical qui peut tourner sur lui-même grâce à l'action d'une bobine à l'intérieur du boîtier.

Si ce moteur est alimenté par une tension continue, alors il tournera à la même vitesse dans le même sens. Si au contraire, on lui injecte une tension variable avec une fréquence assez élevée, l'axe va se mettre à vibrer et produira donc du son.



Pour le prototype final, nous avons utilisé un transducteur de Adafruit qui est conçu pour l'écoute osseuse. Il utilise le même système que les moteurs pour fonctionner mais est dimensionné pour l'ostéophonie.

C'est à dire que la conversion du signal électrique ne se fait pas en mouvement de rotation, mais de translation, ce qui le rend plus adapté à notre utilisation.

Il possède plusieurs avantages que nous évoquerons dans la deuxième partie.

2.3 Module de communication sans fil

Pour notre application, nous cherchons à transmettre de la voix ainsi que de la musique. Un fichier mp3 est souvent codé avec un débit de 128kbit/s (avec un MPEG-1 Layer III) car c'est un bon compromis : il permet d'avoir une bonne qualité par rapport au poids du fichier. Il y a différent types de protocoles réseaux qui peuvent être utilisés : Le WiFi est un protocole qui consomme énormément d'énergie et qui est utile pour des débits très rapides sur de grandes distances, ce qui n'est pas notre cas. Le Zigbee quant à lui peut théoriquement permettre du transfert à 250 kbits/s mais cette valeur ne représente pas le payload (les données utiles). En effet, il prendre en compte tous les bits de protocoles comme le CRC ou l'entête. Ainsi avec Zigbee, on atteint au maximum un throughput de 912 bits/s (le throughput représente le payload divisé par le temps complet pour l'échange de données, ie transfert et réception). Le Bluetooth v4.0/4.1 permet un throughput de 305Kbits/s et le v4.2 permet quant à lui d'atteindre 803 kbits/s (remarque : on parle ici du Bluetooth 4 Classic et non du Bluetooth 4 Low Energy qui n'est pas du tout conçu pour ça).

Étant donné le débit correspondant au mp3 nous nous tournons donc vers cette dernière alternative car le débit est suffisant pour notre utilisation.

Nous avons passé beaucoup de temps lors de cette séance sur le Bluetooth. En effet, comme précisé ci-dessus, les données intéressantes pour le Bluetooth sont le data throughput et le payload. Le problème est qu'il n'existe visiblement aucune norme sur les informations données dans les datasheets.

Sur la plupart de celle-ci, seul le débit est indiqué et il ne correspond en fait qu'au débit théorique de chaque version du Bluetooth. Pour le peu de modules que nous avons trouvés contenant des informations supplémentaires, les indications ne sont pas cohérentes puisque pour un modèle, le data throughput n'est que de 32 kbps (ce qui n'est pas clair car il n'est

pas précisé si l'on parle ici de bits ou de bytes), ce qui est très faible comparé au débit du Bluetooth. Cela signifierait que sur 1 Mb pour une seconde, seulement 32 kb sont "utiles", ce qui n'est pas cohérent. Sur un autre modèle enfin, le data throughput est indiqué en kilobyte (ce n'est donc pas un débit mais une valeur) et dépend du système d'exploitation utilisé. Ces valeurs restent très faibles même si on suppose que l'on donne cette valeur pour une seconde.

Après concertation avec M.Redon puis avec M.Boé, qui nous ont confirmé que ces valeurs étaient effectivement très faibles et/ou pas précises, incohérentes, M.Boé nous a proposé de trouver un modèle de dernière génération et de le commander, quitte à faire des tests plus tard quant à son data throughput réel.

Finalement, nous avons trouvé un composant qui contient à la fois un module Bluetooth de quatrième génération permettant le Bluetooth Classic et le BLE, et un module de conversion audio : ce composant est en effet pensé pour être utilisé sans même un contrôleur si nécessaire et d'assurer la réception Bluetooth et la conversion audio.

Comme précisé précédemment, le Bluetooth LE 4.2 étant conçu pour être utilisé pour de longs moments de veille et un faible volume de données, il ne permet pas d'avoir un débit net (troughput) supérieur à 128Kbits/s. Nous avons donc préféré utiliser un Bluetooth Classic 4.1 IS2020 intégré dans un module BM20 qui contient également un DAC 12 bits intégré et une mémoire de programme EEPROM , permettant d'avoir plus de fonctionnalités au niveau du son comme de la stéréo par exemple. Le module Bluetooth que nous avons choisi est de classe 2, ce qui veut dire qu'il a une puissance de 2.5W et une portée d'une dizaine de mètres.

2.4 Microprocesseur et mémoire flash

Pour notre système nous avons également besoin d'un microprocesseur, afin de pouvoir gérer les informations stockées sur la mémoire flash et les utiliser pour être lues par l'utilisateur. Le microcontrôleur compris dans le module BM20 n'est pas assez modulable, nous avons donc prévu d'utiliser le nôtre.

Etant donné que nous sommes ici sur la conception d'un système embarqué, d'un objet connecté, la consommation de la batterie est importante. Nous avons recherché le microprocesseur qui pourrait suffir à nos attentes : c'est à dire une vitesse de traitement suffisante pour la musique ainsi qu'une consommation en énergie la plus basse possible.

Après quelques recherches, nous nous sommes rendus compte que les microcontrôleurs ATMega ne sont pas adaptés à cet usage étant donné leur consommation, mais la famille de microcontrôleurs STM32L (dit Low power) sont un bon compromis entre performances et consommation. Nous choisissons donc le STM32L4 qui propose l'une des plus basses consommations d'électricité ; de plus, il intègre un DAC qui sera utile pour la musique. Nous reviendrons sur STM32 dans la troisième partie du rapport, afin d'évoquer les difficultés rencontrées.

Enfin, pour la mémoire flash, nous avions besoin d'un taille de 4 Mo soit 32 Mbits pour un fichier mp3 de taille moyenne. C'est pourquoi nous avons choisi une mémoire de 64 Mbits afin de pouvoir stocker 2 musiques dessus pour les tests.

3 Électronique : étude de l'ostéophonie, système d'amplification, PCBs et soudure

3.1 Tests sur les transducteurs

Nous avons d'abord réalisé des tests avec un générateur de signaux. Nous avons utilisé un signal carré car plus la différence de tension est importante, plus le matériau va vibrer et donc produire du son.

Nous avons donc branché la lamelle au générateur qui nous permet de contrôler le type de signal (carré ici donc), la fréquence et la tension.

En augmentant la tension progressivement et en se plaçant à une fréquence de 440Hz (ce qui correspond à un La), nous entendons la note sans utiliser notre oreille externe.

Nous avons donc progressivement augmenté la tension du générateur. A partir de 0.3V, le son devient audible mais est très faible. A partir de 2V, le son devient audible pour une écoute normale.

Nous avons ensuite réalisé les même tests avec cette fois des moteurs à courant continu (MCC). Nous avons pu récupéré dans la C201 différentes tailles de moteurs et nous avons donc dû injecter plus ou moins de tension en fonction de la taille. Pour faire fonctionner l'ostéophonie avec les moteurs, il est nécessaire que l'on injecte suffisamment de tension pour que la bobine se mette à tourner.

L'avantage d'un moteur est qu'il produit moins de son audible directement par les oreilles qu'une lamelle, ce qui est un gros avantage. Pour le plus petit moteur testé, nous avons pour une même tension (0.3V) un son légèrement moins audible que pour un lamelle.

On sait également qu'un jack dispose d'une tension de 2.2V crête-à-crête maximum. Cependant, dans la réalité, lorsque l'on mesure à l'oscilloscope la tension circulant en sortie d'une prise jack, la tension maximale que nous obtenons est de 200mV, ce qui n'est pas suffisamment fort pour une bonne qualité d'écoute avec nos transducteurs.

Le dernier point à préciser est que le son est différent en fonction du composant utilisé. En effet, il peut être plus grave ou plus aigu d'un composant à un autre, ce qui est important pour l'écoute de musique mais l'est moins pour écouter une voix. Pour la lamelle, lorsqu'on augmente le volume, un son strident est émis en permanence, ce qui pose de vrais problèmes pour l'écoute.

Dans chacun de nos tests, nous avons pu entendre du son par ostéophonie, les tests sont donc concluants mais à nuancer : Premièrement, le volume sonore est très faible comparé à des écouteurs classiques. Ensuite, le volume sonore dépend beaucoup de la partie du corps en contact : il est le plus fort lorsque la lamelle est mordue entre les dents, il est également assez élevé au dessus de la tempe. Contrairement à nos attentes, il est parfaitement possible d'entendre le son directement par l'oreille externe, c'est un problème car l'objectif est de laisser le champ auditif libre pour d'autres sons.

Après quelques recherches, nous avons trouvé un composant Adafruit qui permet justement de conduire le son à travers les os. Ce composant fonctionne exactement comme nous

l'envisagions : il y a trois bobines qui entourent une bobine centrale. Cette dernière bobine se déplace en fonction du champs du moteur. Ce composant est spécialement conçu pour cette utilisation et possède la taille d'une pièce de 1 euro, ce qui n'est pas trop imposant pour un écouteur.

C'est ce composant que nous utiliserons pour le prototype final. Néanmoins il faudra intégrer au système un circuit d'amplification, et si besoin de filtrage.

3.2 Amplificateur

Pour pouvoir avoir un son correct avec la tension d'entrée de 3.6V de la batterie, nous avons décidé de mettre en place un circuit d'amplification. À partir d'un cours de CCE de l'année dernière, nous avons réalisé un montage à base de LM386N.

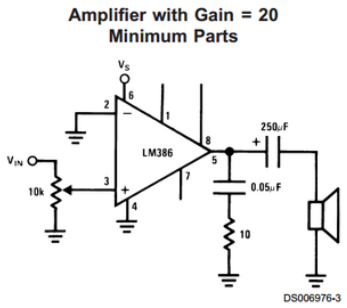


FIGURE 2 : Amplificateur LM386.

Ce circuit nous permettra d'avoir un gain allant de 20 à 200 avec une alimentation entre 4V et 12V, le gain étant réglable par la valeur de la capacité mise entre le pin 1 et 8 du LM386.

Nous disposons d'une batterie de 3.6V, nous avons donc mis en place un autre circuit d'amplification avec l'amplificateur TDA2822 fonctionnant avec une tension entre 1.8V et 15V.

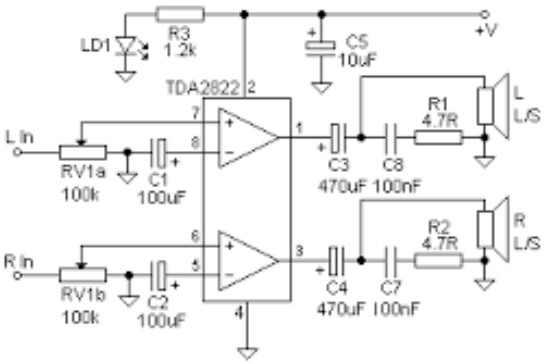


FIGURE 3 : TDA2822.

Après l'assemblage du circuit d'amplification avec le module Bluetooth, nous nous somme

rendus compte que le BM20, et plus précisément le microcontrôleur intégré du BM20, prenait en compte dans son programme un retour des valeurs des pins de sortie.

Nous avons donc décidé de simplement placer un condensateur de grande capacité après la sortie pour pouvoir filtrer les basses fréquences et supprimer la composante continue du signal. En faisant cela nous éliminons une partie du bruit et le BM20 compense en amplifiant le signal de sortie.

3.3 Conception des PCBs

Nous avons réalisé durant le projet trois différents PCBs : les deux premiers PCBs servaient à tester nos composants. Le premier était destiné au module Bluetooth BM20 : nous avons réalisé le symbole et le footprint nous mêmes. Malheureusement, nous n'avions prévu qu'un seul module, nous avons donc préféré en commander un autre plutôt que d'essayer de réaliser un PCB shield qui pourra se fixer sur notre PCB final. Les soudures ont été

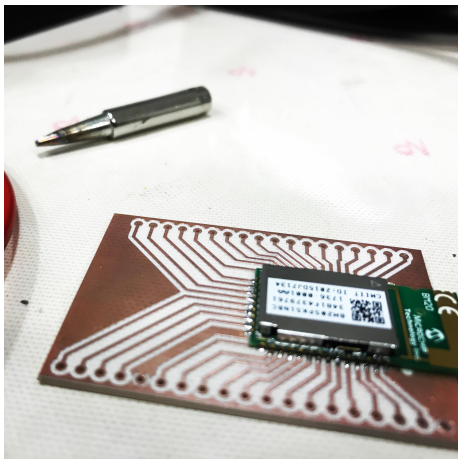


FIGURE 4 : BM20 de tests.

réalisées à la main pour ne pas abîmer le module avec le four. En effet, au cours d'une discussion avec M. Flamen, nous avons appris que les composants Bluetooth ne résistent pas bien à la chaleur.

Le second PCB servait à tester le microcontrôleur STM32L : lors de la réalisation, nous nous sommes rendus compte que nous nous n'avions pas prévu toutes les sorties nécessaires pour notre circuit et notamment une sortie pour mettre le Bootloader. Par conséquent nous avons utilisé un autre microcontrôleur, pour le premier prototype, avec lequel nous étions plus à l'aise. Les symbole et footprint de ce microcontrôleur ont été récupérés sur internet, et les soudures ont été faites au four.

Le dernier PCB correspond au montage final du prototype : il contient le module Bluetooth BM20, la mémoire flash et l'ATMega328p, ainsi que les composants passifs nécessaires à leur fonctionnement.

Contrairement au STM32L, nous avons mis en place des interrupteurs, et bouton, ainsi que des sorties pour le RX, TX, RESET, et SPI, qui nous permettront de mettre le bootloader à l'ATMega328p, et de pouvoir programmer l'ATMega328p et le BM20. Nous pourrons donc facilement accéder au deux modes de fonctionnement du BM20 et modifier nos programmes.

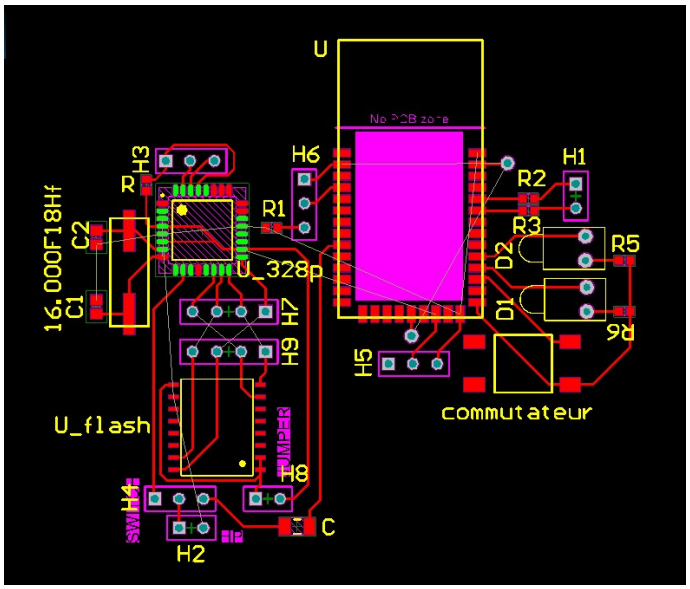


FIGURE 5 : PCB complet.

4 Informatique : code entre le processeur et la mémoire flash, paramétrage du BM20

4.1 Paramétrage du BM20

Pour paramétrer ce module, nous avons tout d'abord essayé de nous connecter au module en liaison série en utilisant un FTDI mais peu importe si le module était en mode test ou en mode appairage, la communication était impossible.

Nous avons donc ensuite téléchargé la suite de logiciels qui sont fournis par MicroChip pour ce composant. Nous avons ensuite trouvé une datasheet qui s'utilise pour la carte d'évaluation BM-20-EVB. Celle-ci explique justement succinctement la démarche à suivre pour modifier le code du composant.

La première chose à faire est d'utiliser l'UI tool auquel on demande de lire un fichier qui contient en réalité la configuration que le module doit adopter. On récupère donc un fichier base dont le système de communication IS20 correspond à notre module. On peut ensuite modifier de nombreux paramètres tels que le nom ou le codage des LEDs pour connaître des informations. De manière générale, ces options seront pour l'interface utilisateur.

Ensuite on utilise le DSP tool qui permet de régler les paramètres du module audio intégré dans le BM20 ; on peut ainsi changer les paramètres du CODEC, du traitement et de la conversion de la musique, de la voix.

Puis, nous utilisons le MPET tool qui permet quand à lui de réunir les différents fichiers afin d'être téléchargés dans le module Bluetooth. Il faut tout d'abord sélectionner un fichier de type .bin qui est donné et contient déjà un ensemble de trois fichiers qui correspondent

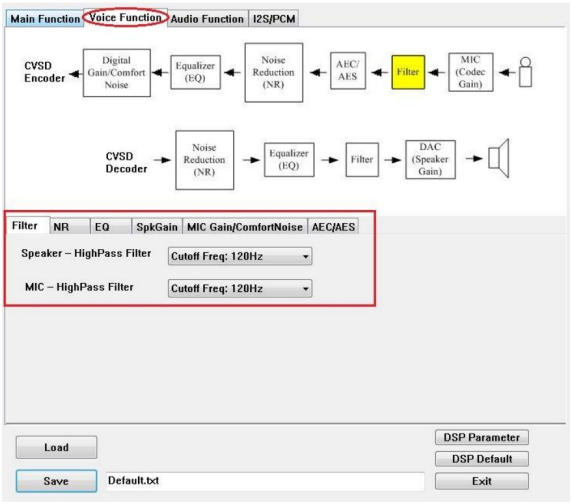


FIGURE 6 : UI tool.

à la configuration basique du composant.

On peut ensuite choisir d'échanger chaque fichier avec un autre correspondant, il doit bien sûr toujours y avoir un fichier correspondant à l'interface utilisateur (UI) et au module audio.

Enfin, on lance l'E2PROM tool qui va permettre d'écrire sur la mémoire du BM20. On se branche toujours en FTDI entre le composant et l'ordinateur et on passe le module en mode test. Il nous faut ensuite indiquer le port sur lequel nous l'avons branché et le logiciel nous indique le nom du composant si il le reconnaît. On peut enfin écrire notre programmation.

Remarque : le fait que l'E2PROM tool reconnaisse le composant montre bien que l'on peut communiquer mais nous ne parvenons tout de même pas à lire directement sur la liaison série par Minicom ou PuTTY.

4.2 Communication avec le STM32L

La famille des microprocesseurs STM32 de ST Microelectronics fournit une vaste gamme de périphériques autour d'un cœur d'ARM Cortex-M4. Le nôtre , le STM32L433CC, opère sur 32 bits, il dispose de GPIO et d'interfaces de communication CAN, I2C, LPUART, SAI, SPI, USART, et USB. Il possède deux DAC sur 12 bits chacun et 10 channel de ADC cadencés à 80MHz.

Nous avons choisi ce microcontrôleur parce qu'il intègre un DAC, ce qui va nous permettre d'envoyer directement notre signal au transducteur. Nous utiliserons le bus SPI pour communiquer avec la mémoire flash. Le STM32L433CC propose un compromis entre une puissance de calcul appréciable et une consommation réduite offrant des modes de veille en vue de réduire la consommation moyenne d'une application.

Pour la compilation, nous sommes partis sur arm-none-eabi-gcc, une chaîne de compilation basée sur l'habituelle génération des binaires issus de 'binutils' et 'gcc' sur architecture x86 à destination du processeur ARM. Nous nous somme basés sur le script

git ://github.com/esden/summon-arm-toolchain.git pour l'installation de la toolchain avec la version Vanilla de GCC au lieu de linaro GCC ; par défaut, seule la bibliothèque libre 'libopencm3' est installée. Nous avons rédigé un Makefile, pour la compilation et la génération du fichier binaire que nous avons testé sur une carte de développement STM32 Nucleo-64 boards avec un programme pour faire clignoter une led.

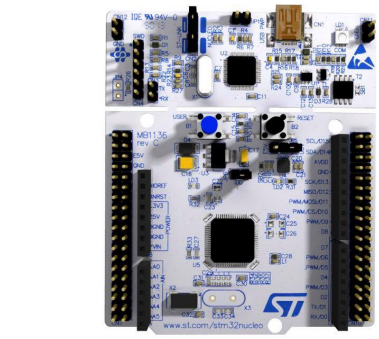


FIGURE 7 : Carte de développement Nucleo-64.

Le second prérequis, après l'obtention d'une chaîne de compilation fonctionnelle, concerne l'outil pour programmer le microcontrôleur. Deux solutions sont possibles : la programmation par JTAG grâce à OpenOCD et à une sonde, et la programmation par RS232 avec un outil tel que stm32flash.

Nous ne disposons pas d'une sonde JTAG, mais la carte de dev STM32 Nucleo-64 boards dispose d'un programmer ST-LINK/V2 que nous pouvons utiliser avec l'utilitaire 'OpenOCD' ou bien 'st-util'. En déplaçant des jumpers, nous pouvons court-circuiter le microcontrôleur présent sur la carte, et utiliser le nôtre avec les pins SWDIO et SWCLK.

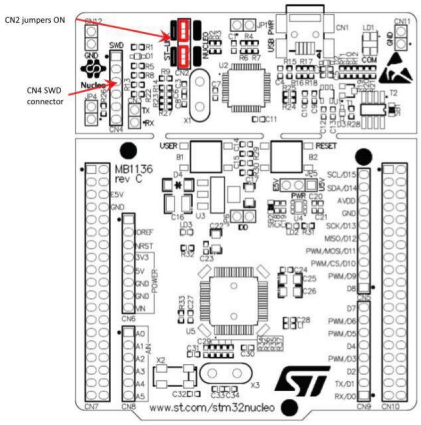


FIGURE 8 : branchements bootloader Nucleo.

Après quelque tests nous nous sommes rendus compte que le circuit que nous avions réalisé n'était pas correct, et qu'il fallait modifier le PCB pour y intégrer des sorties permettant de mettre le Bootloader, ainsi que les pins choisis pour l'horloge.

Nous avons décidé de réaliser notre premier prototype avec un ATMega328p avec lequel nous étions plus familiarisés.

4.3 Code entre la mémoire flash et le microprocesseur

La première étape était de trouver le moyen de lire de la musique grâce à ce processeur. Nous avons pour cela utilisé la bibliothèque PCM qui permet de lire les musiques au format .wav en stockant les valeurs dans un tableau. Chaque case du tableau contient une valeur entre 0 et 255.

La bibliothèque s'occupe ensuite seule de lire la musique sur une sortie PWM. Il est important de garder en tête que le tableau est ici stocké de manière statique dans le programme et donc dans la mémoire. Or, la mémoire de l'ATMega est très limitée, c'est pourquoi nous avons besoin de la mémoire flash.

Notre mémoire flash utilise le protocole SPI pour communiquer. Celui-ci est également disponible sur l'ATMega328p. Le protocole interne à l'Arduino IDE ne permet que de communiquer dans un seul sens ou alors entre deux modules programmables. Or ici, on ne peut écrire de fonctions et de programme que dans l'ATMega.

Le protocole SPI utilisent 4 PINs : DO, DI, SCK, SS. SS ou CS est le PIN qui permet d'indiquer si le module est sélectionné par l'Arduino (dans notre cas). Les PINs DO et DI permettent l'envoi et la réception de messages. Enfin, SCK permet la synchronisation des deux côtés : mémoire flash et Arduino dans notre cas.

Nous utilisons donc la bibliothèque SPIFlash qui permet à la base de communiquer avec des mémoires flash du constructeur Winbond mais, forte de son succès, cette bibliothèque a été très développée et peut ainsi permettre de gérer de nombreuses cartes et disposent de plusieurs fonctions de haut niveau qui permettent de ne pas gérer les durées des échanges du protocole SPI.

En effet, sans bibliothèque, nous aurions dû prendre en compte la taille de chaque envoi ou réception et du traitement. Ces informations sont d'ailleurs précisées dans la datasheet de chaque mémoire flash pour communiquer correctement.

La SPIFlash nous permet d'écrire sur la mémoire ainsi que de lire comme nous le souhaitons. En effet, nous pouvons écrire par octet, par chaînes de caractère, par entiers. Elle implémente également des fonctions d'effacements de zone mémoire.

La librairie PCM demande un tableau de valeurs entre 0 et 255. Après plusieurs tests nous avons calculé que nous pouvions à peut prêt stocker 528 valeurs dans un tableau avant que la mémoire ne sature. Nous récupérons un fichier de données pour notre musique qui contient environ 35 fois plusieurs valeurs (pour quelques secondes). C'est pourquoi nous avions besoin d'une mémoire flash. Nous devons donc dans un premier temps stocker toutes les valeurs (par plusieurs téléchargements) dans la mémoire flash.

Ensuite on lit octet par octet pour stocker un tronçon de musique dans un tableau. Puis on peut utiliser la bibliothèque PCM pour utiliser ce tableau et jouer la musique. Si on lit suffisamment rapide la mémoire flash, on entendera qu'une musique et non pas des morceaux.

Il y a toutefois un problème pour l'utilisation de ces deux bibliothèques : ces deux librairies utilisent le PIN PWM 11. Ce PIN est utilisé pour la communication SPI comme MOSI. Les autres pins utilisaient par cette bibliothèques sont le 10,12 et 13. Ces PINs sont défini

dans l'électronique du système et ne peuvent donc pas être changé. Le PIN 11 est quant à lui utilisé par la bibliothèque PCM comme sortie pour la musique. Ce PIN est choisi car il correspond au registre OCR2A du Timer 2.

Le fonctionnement interne de la bibliothèque est l'utilisation du Timer 2 et du Timer 1. Le timer 2 utilise le mode Fast-PWM et permet ainsi de donner en sortie de la musique. La solution est donc de changer de Timer dans le code source de la bibliothèque. On change donc tous les registres nécessaires mais il y a en fait un problème à cette technique : le timer 0 et 2 ne sont pas tout à fait identiques. En effet, le Timer 2 permet de passer en mode asynchrone ce qui n'est pas possible avec le Timer 0. Le résultat est donc qu'une seule même piste musicale peut être joué et ne changera pas avec le Timer 0 mais en effectuant de nombreuses modifications dans le code. Il est donc nécessaire de faire d'autres modifications dans la bibliothèque PCM pour faire lire tous les morceaux de la musique que nous devrons récupérer.

5 Conception : modélisation 3D du casque

Le dernier aspect de notre projet reposait sur la conception d'un casque 3D. En effet, étant donné que nous travaillons sur un système embarqué destiné à l'utilisateur, il était nécessaire, en plus d'un code et de l'électronique fonctionnelle, d'avoir un objet design.

Afin d'obtenir un objet peu coûteux avec les caractéristiques exactes souhaitées, nous nous sommes rapidement tournés vers la solution de l'impression 3D.

Cette solution nécessite de prendre en compte de nombreux éléments. Premièrement, nous ne pouvons pas imprimer aussi finement qu'on le désire car le résultat dépend de la taille de la tête d'impression Ensuite, l'impression 3D que nous utilisons est une impression additive, c'est-à-dire couche par couche. Lorsque c'est nécessaire, les logiciels des imprimantes 3D rajouteront donc des structures pour faire tenir des objets arrondis, etc. Nous devons donc réfléchir à un casque qui soit relativement plat afin d'être imprimé assez simplement. Il faut éviter les objets trop hauts, les arrondis et les courbes trop prononcées. Enfin, il faut prendre en compte que les impressions 3D sont rarement parfaites : la tête d'impression subit des décalages, du fil peut s'accumuler. Il est donc important de créer un prototype assez simple.

Ci-dessus, on peut voir les différents prototypes qui ont été conçus durant ce projet. Pour la conception, nous avons utilisé le logiciel Open Source Blender que maîtrisait déjà Simon Feutrier. Ce logiciel n'est pas orienté conception industrielle. C'est à la fois un atout et un inconvénient. En effet, il donne plus de liberté et nous pouvons faire tout ce que nous voulons plus facilement, ce qui est d'autant plus vrai pour des formes complexes. Toutefois, il est plus difficile d'avoir des meshs (formes 3D) qui sont parfaitement prévues pour l'impression 3D et donc de connaître leur dimensions précisément.

Nous avons ainsi créé un premier prototype qui correspondait en fait à un modèle trouvé sur internet. Ce modèle présente donc une bonne ergonomie, un design élégant et est minimaliste. A contrario, il ne prend pas en compte les contraintes de l'impression 3D puisque le casque est très arrondi sur tous les axes et est réalisé en une seule pièce : sa taille et les structures qu'il va nécessiter rendront donc très ardue son impression. En outre, il y a trop peu d'espace pour y stocker un système prototypique avec une batterie et un

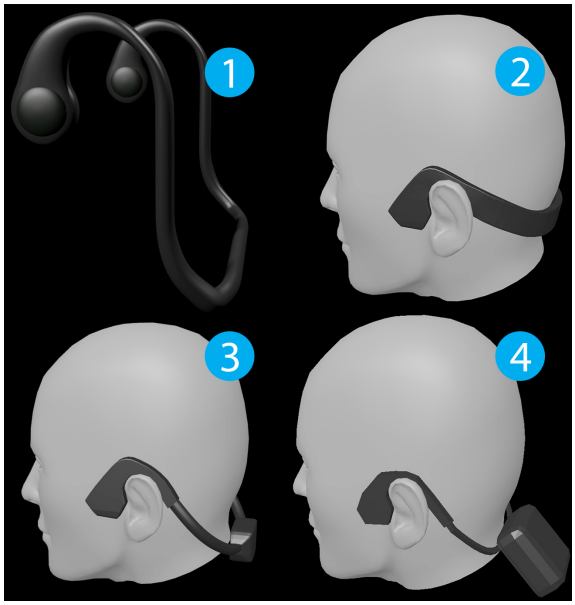


FIGURE 9 : Images des différents prototypes.

PCB.

Nous avons donc ensuite créé un deuxième modèle plus épais et qui conserve une forme plus rectangulaire. Le résultat est satisfaisant mais l'arceau est toujours arrondi. Par ailleurs, il est toujours réalisé en une seule pièce et c'est là qu'intervient une autre contrainte de l'impression 3D. A l'interieur du casque, nous devons y insérer le transducteur à l'avant, ainsi que les fils qui vont du PCB jusqu'à l'écouteur et donc évidemment la carte électronique. Nous devons faire un système qui permette d'ouvrir le casque.

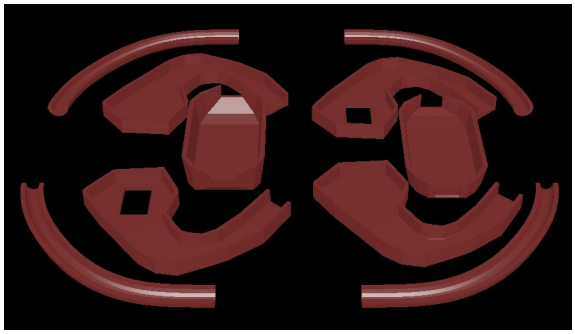


FIGURE 10 : PCB complet.

Sur le troisième modèle on arrive donc sur un casque dont l'oreillette est séparée de l'arceau. L'oreillette est plate, ce qui respecte moins la physiologie humaine mais permet une impression 3D. Il y a également une coque à l'arrière qui intégrera le PCB.

Chaque partie (il y en a 3) est séparée en deux afin d'être imprimée comme on peut le voir sur la photographie ci-dessus. Notre idée était ensuite de coller chaque partie. On peut voir que nous avons dû rajouter de l'épaisseur sur chaque partie car sinon ce n'est pas cohérent d'un point de vue physique.

Cependant sur les conseils de M. Redon, nous avons décidé de créer une quatrième version qui permet justement d'ouvrir et de refermer les parties avec un système de fixation.

Le quatrième prototype est le dernier que nous ayons réalisé. L'oreillette et la coque possèdent un système qui leur permet de s'ouvrir et de fermer. L'oreillette à été drastiquement réduite car après la première impression, nous nous sommes rendus compte que le casque était trop volumineux. Toutes les pièces ont été réduites au maximum, ce qui laisse donc peu de place au système de fixation.

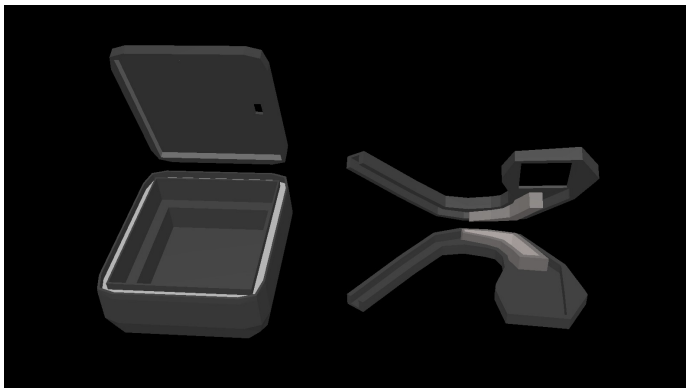


FIGURE 11 : Systèmes de fixations.

Au milieu de la branche de l'oreillette est disposé un système d'insertion entre les deux parties : une partie s'imbrique dans une autre.

La coque arrière possède un système de couvercle avec le même procédé que celui décrit ci-avant. On peut remarquer que la coque est bien plus épaisse. Elle est en effet trop volumineuse pour un produit final. Malheureusement, la batterie étant composée de fonctionnant à l'aide de piles, elle est assez épaisse et même si le PCB ne contient que des composants CMS, il comporte tout de même une certaine épaisseur. On pourra noter que les branches du casque sont bien plus fines et s'insèrent dans les oreillettes afin de les retirer si nécessaire.

En fin de compte le dernier prototype est certes moins ergonomique mais possède l'avantage d'être réalisable et fonctionnel. Dans le cadre d'un produit industriel, d'autres techniques d'impression nous pourraient être utilisées, qui permettraient d'obtenir un résultat plus esthétique.

6 Conclusion

Lors de ce projet nous avons abordé de nombreuses notions liées aux systèmes embarqués. Nous avons dû prendre en compte l'économie d'énergie, la miniaturisation du système, le choix de la communication ainsi qu'utiliser les ressources disponibles pour obtenir un objet final prototypique mais complet.

De plus, durant nos séances nous avons fait appel à des notions dans de nombreux domaines tel que la conception de cartes électroniques, la création de circuits électroniques, la modélisation de pièces complexes en 3D ou encore l'utilisation de bibliothèques C et C++.

Pour finir ce projet nous a permis de travailler avec un objectif précis sur une longue période et nous avons donc utilisé nos capacités de travail en équipe, d'organisation ainsi que nos compétences acquises durant notre scolarité. D'autre part nous avons développés nos connaissance notamment sur la conception de carte, les contraintes des systèmes embarqués, de communication et l'implémentation de fonctions complexes à bas niveau.