

ENS Paris Saclay : Département EEA



école —————
normale —————
supérieure —————
paris–saclay —————

Stage de recherche : Master

Intégration d'une solution de Beamforming sur
microcontrôleur, conception du PCB.

Mise en place d'une POC fonctionnelle avec les
techniques de prototypages rapides.

Réalisé par :

Bouteiller Rémi

Encadrant :

Manuel Pariente

Sommaire

1. INTRODUCTION GENERALE	3
a. <i>Problématique</i>	3
b. <i>Objectifs du stage</i>	4
c. <i>Travail préliminaire</i>	4
2. CHOIX TECHNOLOGIQUES	5
a. <i>Définition des directions principales</i>	5
a. Unité de calcul	5
b. Système d'acquisition, microphone	6
c. Fonctionnalités annexes	6
3. CONCEPTION MECANIQUE	7
a. <i>Dessin 3D sur Fusion 360</i>	7
4. CONCEPTION DU CIRCUIT ELECTRONIQUE	9
a. <i>Choix des composants</i>	9
a. Les microphones	9
b. Le microcontrôleur	11
b. <i>Réalisation du schéma électrique</i>	12
a. Partie microcontrôleur	12
b. Partie alimentation	14
c. Partie périphériques annexes et microphones	15
c. <i>Routage du circuit électronique</i>	16
a. Placement des composants	17
b. Plans de cuivre	17
c. Routage des pistes	18
d. <i>Fabrication et montage du circuit imprimé</i>	18
5. TESTS ET PROGRAMMATION DE LA CARTE	20
a. <i>Projet CubeIDE</i>	20
b. <i>Tests programmation</i>	20
c. <i>Retour sonore par écouteur</i>	21
d. <i>Utilisation de FatFS pour la carte SD</i>	21
e. <i>Mise en œuvre du DFSDM</i>	22
6. CONCLUSION ET SUITE DU PROJET	24
BIBLIOGRAPHIE	25
ANNEXE	26

1. Introduction générale

Mon stage s'est déroulé au sein de l'équipe Multispeech de l'INRIA Nancy, laboratoire lorrain de recherche en informatique et ses applications pour un projet porté par la start-up Pulse. J'ai été encadré par Manuel Pariente en thèse au sein du LORIA et supporté par Antoine Deleforge directeur de thèse de Manuel.

Pulse est un projet de lunettes auditives intelligentes. Les lunettes Pulse permettent une audition focalisée dans les environnements bruyants. La source sonore d'intérêt est localisée à partir de la position des yeux, puis rehaussé à l'aide d'algorithmes d'intelligence artificielle et transmise à l'utilisateur par voie osseuse.

L'équipe porteuse est composée de:

- Manuel Pariente, doctorant à l'INRIA, spécialisé dans le rehaussement de la parole, le deep learning, le traitement du signal et la perception auditive. Créateur d'Asteroid (The PyTorch-based audio source separation toolkit for researchers). Ancien élève de l'ENS Cachan.
- Thibaud Moufle, consultant en BioTech et MedTech pour des startups. Expertise en développement de dispositif médicaux, en valorisation de l'innovation et en science réglementaire.

L'équipe entamera un programme de maturation à **l'INRIA Sophia-Antipolis** en Septembre 2021.

Le LORIA, *Laboratoire Lorrain de Recherche en Informatique et ses Applications* n'as servi que de structure d'accueil pour ce stage. Le LORIA est une Unité Mixte de Recherche commune à plusieurs établissements : le CNRS, l'Université de Lorraine et Inria Nancy.

L'équipe Multispeech dont je faisais partie pendant mon stage était commune avec l'INRIA Nancy. Le stage a cependant été intégralement réalisé à distance à cause des limitations de présence imposées par l'épidémie de COVID-19. J'ai donc travaillé principalement en autonomie avec des retour récurrent de Manuel sur Slack ou via réunion en visio.

a. Problématique

La conception de lunettes auditives intelligentes introduit plusieurs problématiques pour le développement de l'électronique intégrée. En effet, la taille réduite d'une paire de lunettes nous oblige à mettre en place une architecture électronique minimaliste et avec des composants de taille optimisée. De plus, l'utilisation *intrinsèquement* nomade d'une paire de lunette induit un système autonome en énergie.



b. Objectifs du stage

Le but principal est donc d'avancer dans le design du prototype de prothèses auditives intelligentes sous le format de lunettes, sur la partie logicielle et matérielle, le projet était au commencement du stage à l'état d'idée, aucun choix technologique n'avait été arrêté mais les lignes directrices du produit étaient fixées. La problématique énoncée précédemment nous permet de cibler les points d'attention qui seront cruciaux pour le développement des lunettes.

Le but du stage est donc de développer l'électronique d'une paire de lunettes d'assistance auditive intelligentes qui soit autonome en énergie et conforme aux standards de format de taille des paires de lunettes.

Le stage se décompose donc en deux parties :

- Conception d'un PCB de prototypage pour valider nos différents choix techniques
- Validation et écriture du code de test pour le PCB

Si ces deux premiers points sont finalisés, le travail se poursuivra sur l'implémentation d'un algorithme de beamforming.

c. Travail préliminaire

En amont de la période de stage, un premier travail de défrichage à été réalisé dans le cadre d'un mini-projet d'informatique industrielle. Ce projet portant sur la conception d'un code pour carte STM32F746G-DISCO avait pour objectif de nous faire prendre en main un périphérique particulier du STM32. Manuel a proposé un sujet de mini-projet portant sur l'implémentation d'un algorithme de traitement audio simple sur le stm32 de la carte.

C'est donc ce projet que j'ai réalisé, il m'a permis de me familiariser avec les concepts de base du Beamforming et de prendre en main le périphérique de transmission de signaux audio.

Le code développé dans ce mini projet pose les bases du code écrit pour le stage, il sera donc explicité dans la partie associée.

Comme indiqué au début de cette introduction, Manuel est doctorant en Intelligence Artificielle appliqué au rehaussement de la parole, par conséquent il ne dispose pas de compétences approfondies en développement électronique. Ce rapport aura donc principalement une visée explicative sur mon travail sans rentrer dans le détail des calculs effectués à chacune des étapes. L'objectif étant ainsi que Manuel puisse comprendre et reprendre si nécessaire l'électronique conçue pendant ce stage.

2. Choix technologiques

a. Définition des directions principales

Seul le cadre du projet étant fixé au commencement du stage, j'ai dû, en collaboration avec Manuel, choisir les points principaux qui guideront le reste du développement bien que toujours modifiables par la suite si cela s'avère nécessaire.

Les points principaux à définir étaient les suivants :

- Unité de calcul
- Système d'acquisition

a. Unité de calcul

La finalité du projet étant de réaliser des prothèses auditives intelligentes, il est nécessaire d'embarquer un composant programmable qui permettra de mettre en œuvre l'algorithme de traitement audio.

Pour cela, deux options ont été envisagée pour faire tourner le programme principal:

- Un microcontrôleur ou PSoC monté directement dans les lunettes
- Un mini-ordinateur déporté dans un boîtier porté à la ceinture par exemple

La finalité du projet est de mettre en œuvre un algorithme à base d'intelligence artificielle développé par Manuel directement sur les lunettes. Dans ce cadre, l'utilisation d'un mini-ordinateur permettrait l'utilisation de code directement issu du PC de développement alors qu'un microcontrôleur nécessite de porter le code vers du langage C, que ce soit un code simple ou un modèle Tensorflow. Néanmoins, la faible taille d'un microcontrôleur permet de l'intégrer directement dans la paire de lunettes tandis que son architecture interne lui permet d'être économe en énergie ce qui est un point crucial du projet si l'on souhaite réaliser des lunettes visuellement anodines mais pleinement fonctionnelles et autonomes. C'est pour ces deux raisons que nous avons choisi d'utiliser un microcontrôleur pour la suite du projet. La gamme STM32 de STMicroelectronics qui correspond au microcontrôleur utilisé dans le mini-projet préliminaire nous ayant donné satisfaction, nous avons décidé de poursuivre avec ces modèles.

b. Système d'acquisition, microphone

Il existe deux principaux types de microphones que l'on peut monter sur circuit imprimé, ECM *Electret Condenser Microphones* et MEMS *Micro-Electro-Mechanical System*.

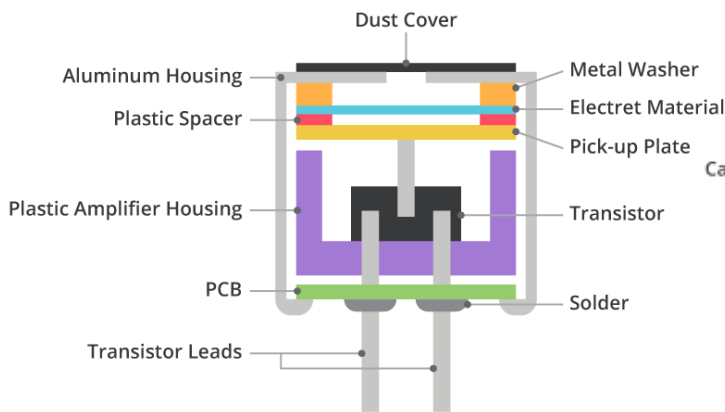


Figure 2 : Schéma de construction d'un microphone ECM

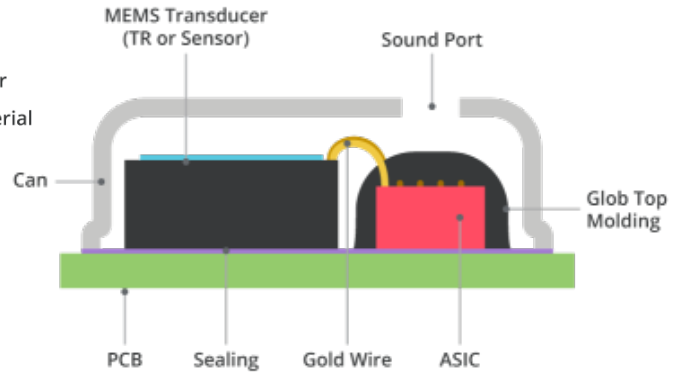


Figure 1 : Schéma de construction d'un microphone MEMS

On observe que les microphones MEMS ne sont pas constitués avec des parties mobiles contrairement aux micros ECM, ce qui permet d'obtenir des boîtiers plus compacts et plus robustes aux vibrations ou chocs. On observe également que le MEMS contient un **ASIC** qui permet de réaliser un conditionnement du signal émis sur la patte de sortie qui peut ainsi être analogique comme les microphones ECM ou digital.

J'ai donc décidé d'utiliser un microphone MEMS pour la suite du projet, cela correspond à notre cahier des charges et est recommandé par STMicroelectronics pour notre cas d'usage.

c. Fonctionnalités annexes

Les deux autres points importants pris en compte lors du développement de ce prototype étaient la gestion de l'alimentation et la communication entre les deux branches de la paire de lunettes. J'ai choisi d'utiliser un accumulateur lithium-ion pour sa forte densité énergétique et leur grande disponibilité. Néanmoins ce type d'accumulateur nécessite une gestion de la charge et de la décharge spécifique si l'on souhaite écarter tout danger lors de l'utilisation.

La communication entre les branches de lunettes est un point essentiel du projet si l'on souhaite mettre en place un algorithme de beamforming performant afin de pouvoir mutualiser les sons perçus par les six microphones. Pour la première partie du projet, j'ai décidé de mettre en place une communication par bus **SPI** qui permet une très haute fréquence de communication *maximum 10Mbits/s*. Par la suite, ce canal de communication sera soit intégré à la structure des lunettes soit changé pour une communication sans fil.

3. Conception mécanique

Le produit final de Pulse sera vraisemblablement une paire de lunettes, Manuel m'as donc dirigé vers un dessin 3d open-source de lunettes adaptées pour l'intégration d'un micro-PCB. Je me suis donc servi de cette base de travail pour modifier l'espace disponible et intégrer des trous pour les sons des micros. Cette partie de dessin 3d nous permettra d'avoir une bonne idée de la place disponible pour le circuit imprimé pour la suite du projet. Les contraintes mécaniques n'étant pas la préoccupation principale, celles-ci ont été prises en compte de manière empirique.

a. Dessin 3D sur Fusion 360

Fusion 360 est le logiciel de dessin 3D développé par autodesk, il permet de réaliser des schémas et modèles 3D. J'ai donc importé le modèle open source évoqué précédemment pour pouvoir m'en inspirer et le modifier afin de correspondre à nos besoins. L'une des principales contraintes du design 3D réside dans le placement des microphones qui doivent être parfaitement équidistants les uns des autres pour faciliter le traitement intégré au microcontrôleur. Un des microphones doit également être placé au plus proche du pavillon de l'oreille pour que l'on puisse disposer du son perçu normalement par le porteur des lunettes. Voici donc les modifications effectuées sur le design.

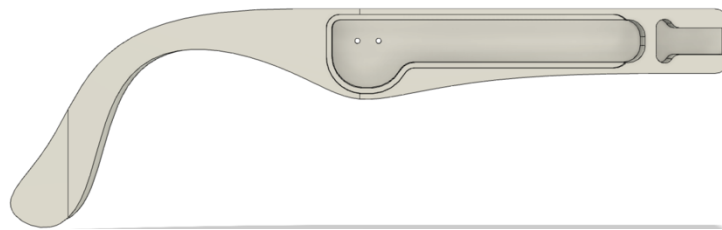


Figure 3 : Dessin de branche de lunettes initial

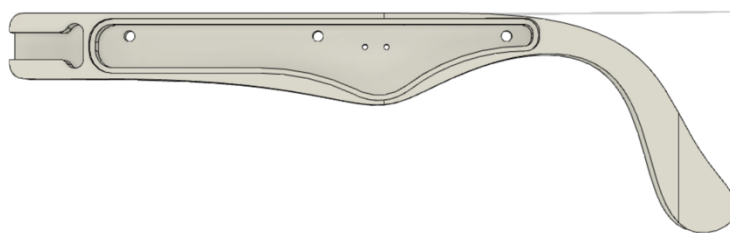
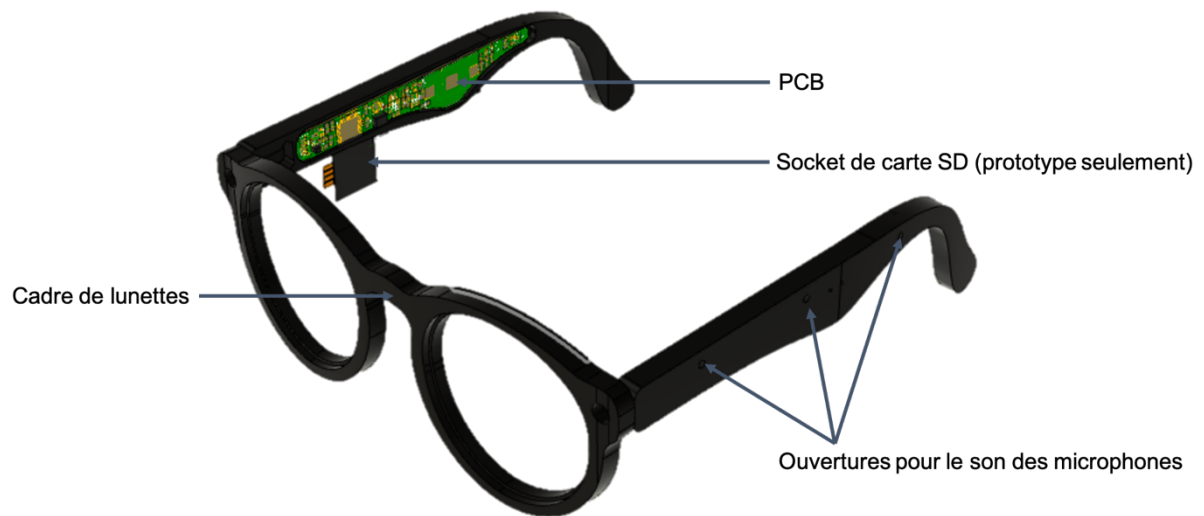


Figure 4 : Dessin de branche de lunettes modifié

La forme de la branche a été reprise, mais l'emplacement creux à l'intérieur de la branche a été retravaillé pour permettre d'y intégrer les trous pour le son des microphones et d'agrandir l'emplacement afin de pouvoir placer le troisième microphone proche de l'oreille.

A partir de ce dessin 3D, j'ai pu extraire la forme et les dimensions voulues pour la carte électronique et les réutiliser dans le logiciel de CAO électronique afin de définir les contours du circuit imprimé.

Voici un dessin 3D des lunettes complètes réaliser à la fin du stage, il permet de mieux comprendre le produit imaginé.



Le retour audio non représenté ici est réalisé avec un écouteur directement soudé sur le circuit imprimé *PCB*.

4. Conception du circuit électronique

Pour valider le prototype, j'ai décidé de concevoir directement un PCB avec les périphériques nécessaires pour obtenir une première version capable de capter du son avec les microphones MEMS, de traiter ces données audio avec un microcontrôleur puis de retransmettre le résultat via une oreillette. J'ai également prévu de mettre des interfaces de programmation, débogage et de communication avec l'extérieur (PC via ST Link ou FTDI, un autre microcontrôleur, ...).

Je vais donc dans cette partie vous présenter les options choisies et leur implémentation sur la CAO du PCB réalisée avec le logiciel Eagle d'autodesk.

a. Choix des composants

La première partie de la conception d'une carte électronique réside dans l'établissement du schéma électrique du circuit que l'on souhaite imprimer.

a. Les microphones

Les microphones utilisés pour ce projet doivent permettre de capter l'intégralité du spectre audible, être de taille réduite pour loger dans une branche de lunettes et pouvoir être interfacés avec le microcontrôleur choisi par la suite.

Il existe 3 types d'interfaces de microphones qui sont largement répandues :

- Sortie analogique, le signal électrique est l'image du signal sonore, il est nécessaire de le convertir en informations numériques avec un CAN *Convertisseur Analogique Numérique* du microcontrôleur.
- Sortie PCM / SAI, le microphone contient le CAN qui échantillonne les données sur n bits à une fréquence donnée souvent proche de $48kHz$.

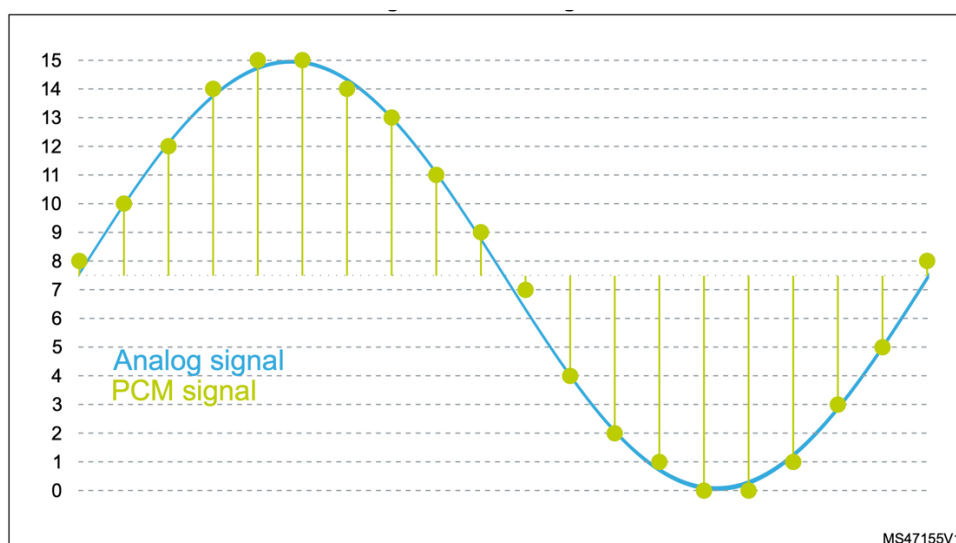


Figure 5 : Schéma d'un relevé au format PCM

On observe que le format PCM correspond bien à un échantillonnage classique du signal analogique.

Les données de sortie sont ensuite transmises sous formes de trames de données codées sur 8 bits via un bus **i2s** ou **SPI**.

- Sortie PDM, le microphone contient le CAN, les données sont émises sous forme de signal binaire haute fréquence 1-5MHz, la valeur moyenne de ce signal correspond au signal analogique.

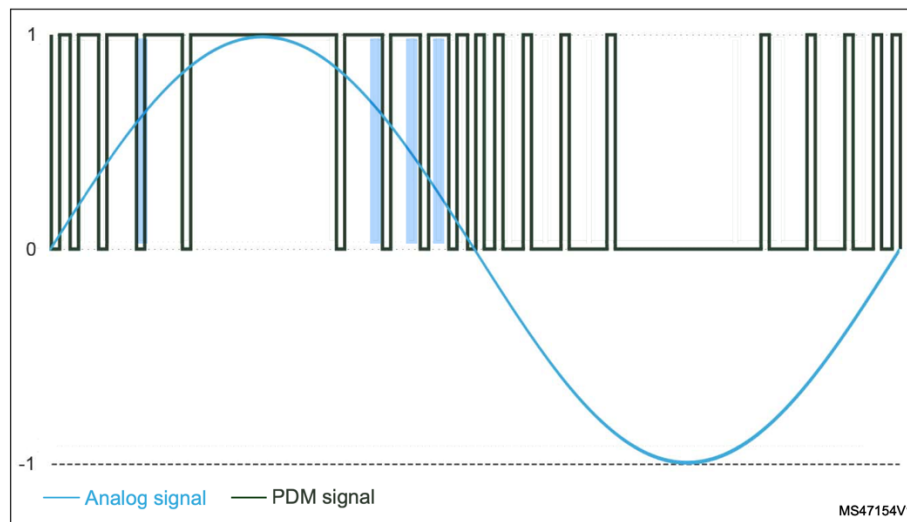


Figure 6 : Schéma d'un relevé au format PDM

Afin de convertir le flux PDM en échantillons PCM, le flux PDM doit être filtré et décimé.

Dans l'étape de décimation, la fréquence d'échantillonnage du signal PDM est réduite à la fréquence d'échantillonnage audio visée (48 kHz par exemple). En sélectionnant 1 de chaque M échantillons, la fréquence d'échantillonnage est réduite d'un facteur M. Par conséquent, la fréquence des données PDM (qui est la fréquence de l'horloge du microphone) est M fois la fréquence d'échantillonnage audio cible nécessaire dans une application, où M est le facteur de décimation.

$$\text{Fréquence PDM} = \text{fréquence d'échantillonnage audio} \times \text{facteur de décimation}$$

L'étape de décimation est précédé d'un filtre passe-bas pour éviter la distorsion due au repliement.

Les microphones analogiques ont rapidement été écartés car impliquant plusieurs points négatifs sur le design du PCB.

- Par design, un signal analogique est plus sensible aux interférences électromagnétiques *IEM*. Sur le PCB cela nécessite donc de faire particulièrement attention aux pistes de ces signaux avec une attention portée sur le plan de masse et sur l'empilement des couches.

- Chaque microphone nécessite une entrée sur le microcontrôleur et un canal de *CAN Convertisseur Analogique -> Numérique*.
- Un conditionnement du signal est souvent nécessaire si l'on ne veut pas être limité par les capacités du *CAN* interne du microcontrôleur qui n'est pas spécifiquement désigné pour les signaux audio. La mise en place de ce conditionnement serait alors coûteuse en place sur le PCB mais ajouterait également de la complexité sur le design final et un coût sur la BOM *Bill of Material/Liste d'achat*.

Ces points ne sont pas retrouvés avec un microphone à sortie numérique qui offre en effet conditionnement standardisé du signal.

Le type de microphone digital le plus simple à utiliser est la sortie i2s, les données sont directement interprétables dans le code il n'y a pas de post-traitement nécessaire. C'est ce type de sortie que nous utiliserions dans un cas générique, cependant l'un des objectifs des lunettes est de fournir un algorithme de beamforming angulaire. Dans ce cas, la fréquence d'échantillonnage a un impact fort sur l'écart angulaire minimum avec lequel on peut travailler.

L'avantage des systèmes PDM est que la fréquence d'échantillonnage élevée permet un petit délai (multiple entier de l'intervalle d'échantillonnage), une petite distance entre les microphones (4-8 mm) ce qui correspond à nos contraintes d'intégration. Pour atteindre les mêmes objectifs, les systèmes PCM nécessiteraient un filtre à retard fractionnaire, ce qui est très complexe et coûteux à mettre en œuvre. Dans notre cas d'implémentation sur microcontrôleur cette solution n'est donc pas envisageable.

Le PDM est particulièrement pratique sur les microcontrôleurs STM32 car il existe un périphérique matériel dédié appelé DFSDM. DFSDM signifie Digital Filter for Sigma-Delta-Modulated signaux. DFSDM est le filtre dédié à la conversion PDM vers PCM. Le délai du PDM traité par DFSDM peut être contrôlé dynamiquement au moment de l'exécution. L'utilisation de ce périphérique permet de décaler un signal PDM de 1 échantillon avant d'effectuer la décimation, ce qui permet une grande précision d'utilisation dans le cas d'un beamforming angulaire. J'ai donc choisi d'utiliser un microphone MEMS à sortie PDM, Il faudra veiller à utiliser un microcontrôleur STM32 comprenant le périphérique DFSDM.

b. Le microcontrôleur

Le microcontrôleur est le composant principal du circuit envisagé, c'est également le composant le plus complexe. Les points importants pour choisir le modèle adapté dans la gamme STM32 sont :

- La consommation électrique la plus faible possible
- La puissance de calcul haute pour pouvoir tester un maximum d'algorithmes
- La possibilité d'acquérir les signaux de plusieurs micros simultanément via le DFSDM
- Un package compatible avec le dessin 3d de la carte vu précédemment

Avec ces points en tête, j'ai en premier lieu regardé les différents modèles de microcontrôleur comprenant un DFSDM. En se référant à la documentation fournie par STMicroelectronics, on peut constater que le DFSDM est présent sur les lignes STM32L4, STM32L4+, STM32H7 ainsi que sur certains microcontrôleurs STM32F4 et STM32F7. En regardant les datasheets de ces microcontrôleurs, on constate que les *STM32LX* sont des microcontrôleurs Ultra Basse Consommation. Le design des lunettes permettant uniquement d'intégrer un accumulateur de faible taille, la consommation d'énergie est un point important du système final. J'ai donc choisi d'utiliser un microcontrôleur STM32L4+, sa faible consommation d'énergie et sa puissance de calcul comparable au microcontrôleurs haute performance en font un compromis idéal. Le modèle exact est choisi pour répondre aux contraintes de taille et de pinout.

b. Réalisation du schéma électrique

Le schéma électrique permet de faire les connexions entre les différentes pattes des périphériques afin de simplifier la partie routage. On y place donc tous les composants qui seront montés sur le circuit imprimé.

a. Partie microcontrôleur

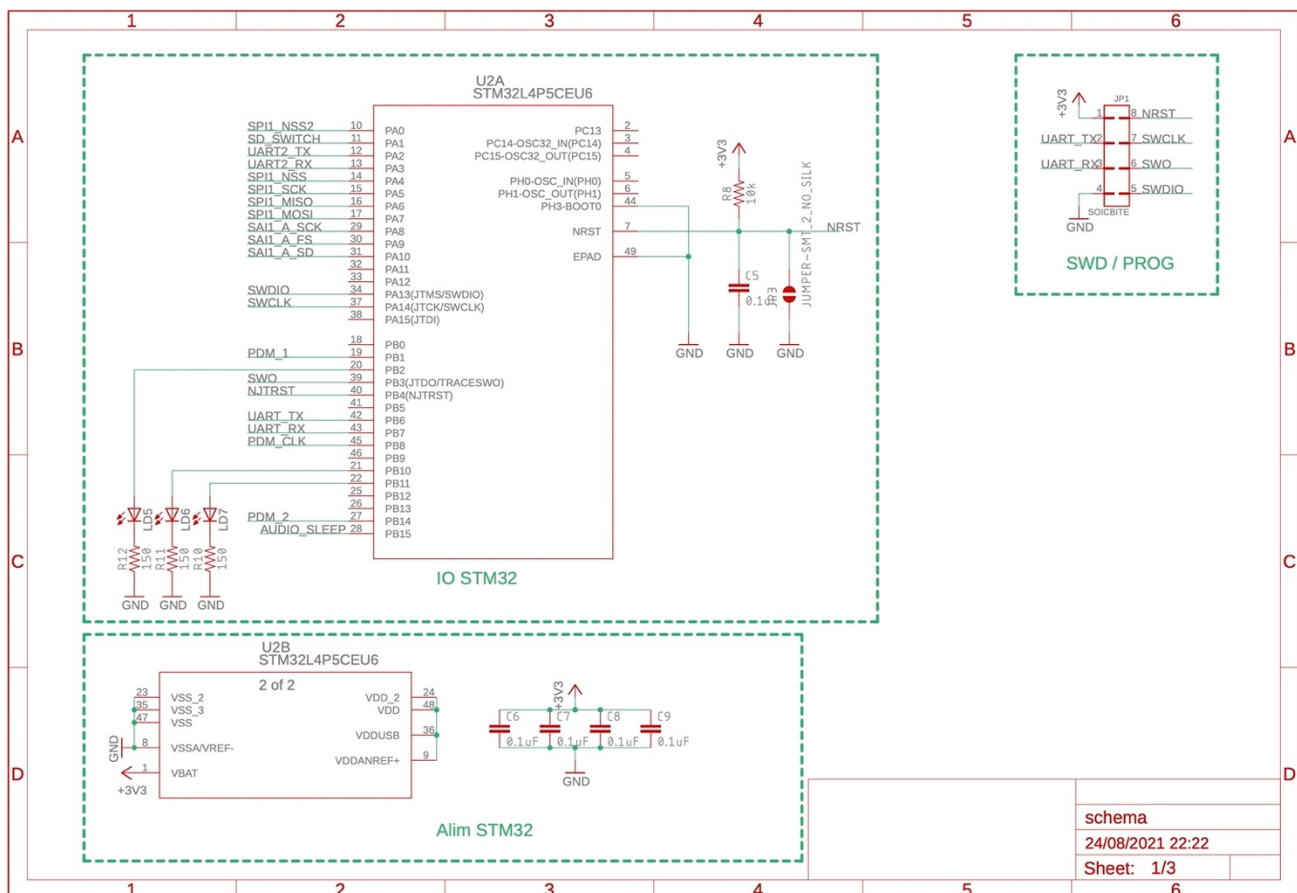


Figure 7 : Schématique, microcontrôleur

Sur cette première page, j'ai placé uniquement le microcontrôleur et les composants passifs associés.

Le microcontrôleur choisi est le modèle de la série STM32L4+ permettant d'embarquer le plus de puissance dans le package le plus petit, soit le STM32L4P5CEU6.

Sur la partie alimentation, on observe la présence de capacités de découplage, un condensateur de découplage est un condensateur destiné à réduire le couplage entre signal et alimentation. Le condensateur relie le canal d'alimentation à la masse pour abaisser son impédance dans la bande passante du signal. Sans ce découplage, les pics de courant transitoires demandés par le microcontrôleur provoqueraient une courte chute de tension qui altère le fonctionnement de ce dernier.

Sur la partie IO *Entrées/Sorties*, j'ai connecté des DELs de débogage, ces leds sont connectées directement aux sorties du microcontrôleur, cela permet de changer leur état en changeant uniquement l'état du pin associé. Lorsque l'on utilise une liaison série pour retourner des informations de la carte vers l'utilisateur, la durée de transmission et les délais associés ne sont pas négligeables. De fait, cela nous interdit d'envoyer des caractères à n'importe quel moment de l'exécution du code sous peine de ne pas respecter la temporalité d'exécution. L'utilisation des LEDs nous permet de s'affranchir de cette limitation afin de disposer d'une interface de retour limitée mais utilisable à chaque instant. Sur la partie IO, j'ai connecté le pin BOOT0 à la masse pour que le microcontrôleur soit programmé par SWD. Pour le reset, j'ai choisi de garder la possibilité de l'activer manuellement en utilisant un jumper sous forme de pastilles sur le PCB.

Enfin, comme pour tout microcontrôleur, il est nécessaire de pouvoir le programmer, pour cela on utilise le protocole *SWD Serial Wire Debug*. Ce protocole a l'avantage de nécessiter uniquement 2 signaux contrairement au JTAG plus ancien mais nécessitant 4 signaux. Le circuit imprimé devant loger dans un espace le plus réduit possible, nous ne pouvions pas utiliser la prise de programmation classique de STMicroelectronics. J'ai donc décidé d'utiliser le projet opensource *SOICBite* (Merrett, 2020) qui repart d'une prise de programmation de mémoire au format SOIC afin de l'utiliser pour la programmation de microcontrôleur. J'ai donc connecté à cette prise les deux signaux *SWD*, les deux signaux d'alimentation, ainsi que le signal de reset et le signal de debug. En plus de cela j'ai ajouté une communication UART pour recevoir des données de débogage plus précises que les LEDs si nécessaire.

b. Partie alimentation

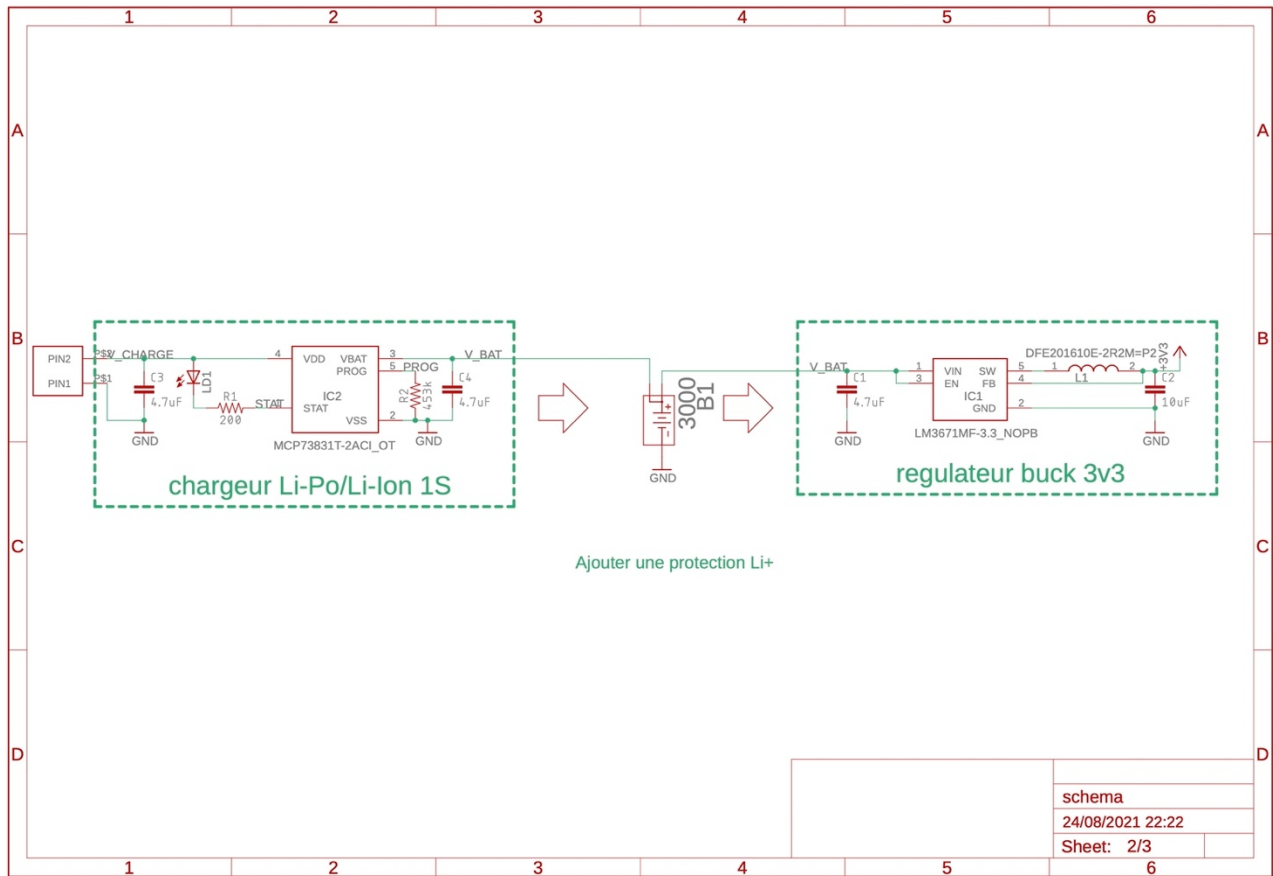


Figure 8 : Schématique, gestion de l'accumulateur

La partie alimentation est composée de deux sous-parties, la partie chargeur adapté aux accumulateurs lithium-ion et le régulateur de tension.

L'accumulateur que j'ai choisi d'utiliser est une cellule lithium-ion **LIR1220** avec une tension de charge de 4.2V, une tension nominale de 3.7V et une capacité de 10mAh. La capacité de la batterie est très faible mais cela nous permet de disposer d'un format réduit intégrable directement dans la branche de lunettes. Pour connecter cet accumulateur au circuit imprimé, j'ai décidé d'intégrer un socket au format **LIR1220** qui loge dans la pièce 3D dessinée.

La charge d'un accumulateur Li-Ion nécessite une gestion particulière du courant de charge pour éviter de détruire l'accumulateur. Le chargeur Li-Ion est basé sur le composant **MCP73831T-2ACI** de Microchip qui est spécifiquement conçu pour cet usage et permet de s'adapter aux caractéristiques de l'accumulateur que l'on souhaite recharger. Le courant de charge est notamment défini par une résistance connectée entre la patte **PROG** et la masse, dans notre cas, l'accumulateur peut se recharger à maximum 0.2C ce qui correspond à une fois sa capacité soit un courant de charge maximal de 2mA. La datasheet nous indique qu'il est nécessaire de respecter la règle $I_{charge} = \frac{1000V}{R_{PROG}}$ avec R_{PROG} en kΩ soit $R_{PROG} = \frac{1000}{0,2} \approx 453k\Omega$. Les condensateurs placés

de part et d'autre de l'IC de charge permettent de filtrer les tensions pour minimiser les transients de tension.

Enfin, la tension de l'accumulateur étant variable entre 4.2V et 3V, elle n'est pas directement utilisable pour alimenter le microcontrôleur qui nécessite une tension stable. C'est pour cette raison qu'il est indispensable de réguler cette tension. La tension d'alimentation choisie pour la carte est le 3,3volts en raison du fait que c'est une valeur très répandue et compatible avec les microphones choisis. Les types de régulateurs les plus répandus pour ce type de problématique sont les régulateurs *LDO low drop out*, régulateurs *Buck abaisseur de tension* et les régulateur *Buck-Boost abaisseur élévateur de tension*. Les régulateurs LDO dissipent l'énergie liée au différentiel de tension en chaleur tandis que les régulateurs Buck hachent la tension d'entrée afin d'obtenir un rendement optimal. Un régulateur de type Buck/Boost permet d'être en mode Buck lorsque la tension de l'accumulateur est supérieure à 3,3V et en mode Boost sinon, ce qui permet d'utiliser la pleine capacité de l'accumulateur. Cependant, la fonctionnalité Boost demande des composants supplémentaires (inductance, diode,...) qui prennent de la place sur le circuit imprimé. Ne disposant pas de cette place mais souhaitant un rendement optimal, j'ai choisi d'utiliser un convertisseur Buck **LM3671** de Texas Instruments.

c. Partie périphériques annexes et microphones

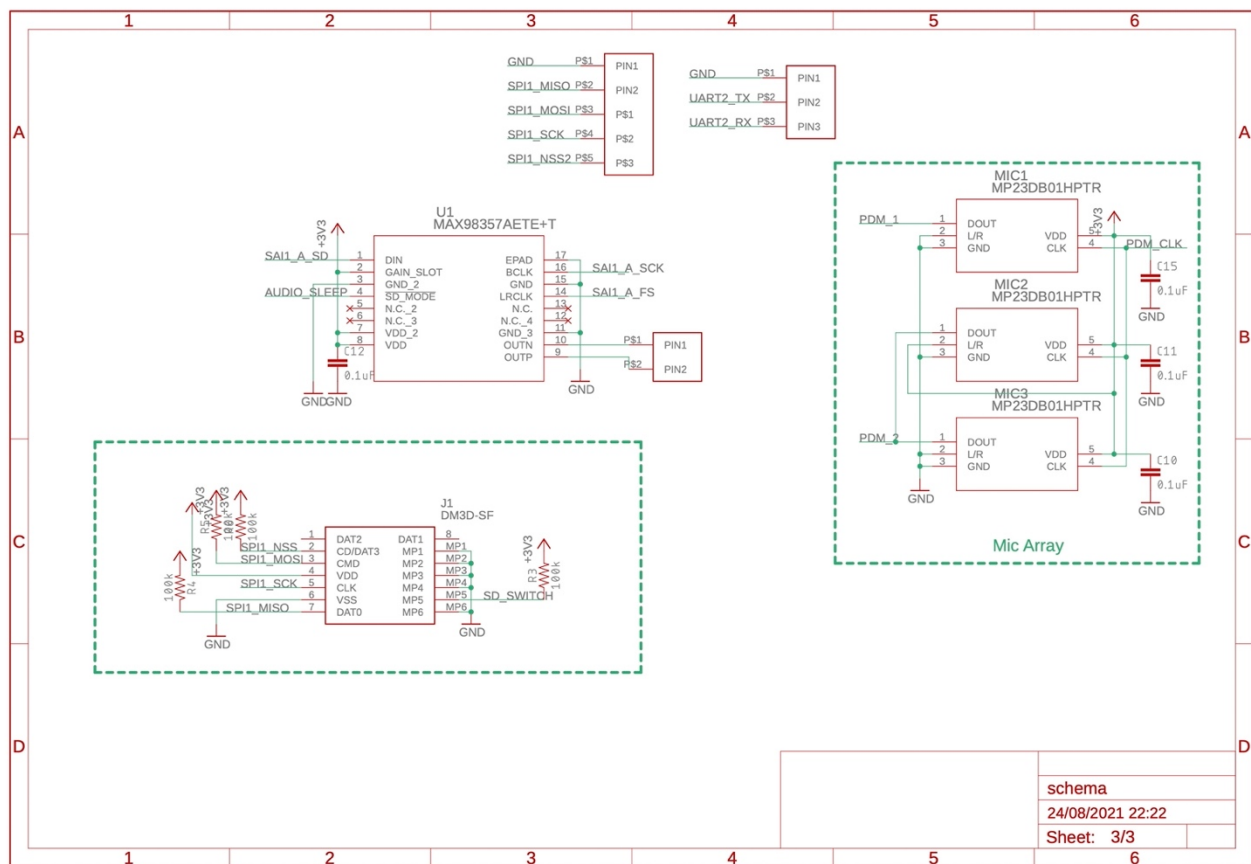


Figure 9 : Schématique, périphériques annexes

Les composants réunis sur cette troisième page du schéma sont les microphones, un socket de carte microsd, un codec audio et des prises pour une communication **SPI** ou **UART**.

Les microphones choisis sont donc des microphones MEMS de chez STMicroelectronics avec une sortie au format PDM. Pour des facilités de montage du circuit imprimé dans la branche de lunettes, nous avons décidé de capter le son par le dessous des micros via un trou dans le circuit imprimé, ces caractéristiques correspondent au modèle de micro **MP23DB01HPTR**.

Le socket de carte microsd est un socket générique trouvé sur Mouser, cependant la communication entre la carte sd et le microcontrôleur diffère des méthodes classiques. En effet le STM32L4P5CEU6 ne dispose pas d'interface **QuadSPI** utilisée normalement, j'ai donc dû utiliser un simple bus de communication **SPI**.

Et enfin, j'ai choisi d'utiliser un codec audio **MAX98357** de Maxim Integrated pour réaliser la conversion Digital -> Analogique des sons envoyés vers l'écouteur. J'ai choisi ce composant pour son faible encombrement et nombre de pin réduit.

c. Routage du circuit électronique

Le routage d'un circuit électronique consiste à placer les différents composants dans une disposition permettant d'optimiser le nombre de pistes et leurs longueurs. Il faut entre autres faire attention à la disposition des microphones qui doivent être bien équidistants. Pour loger le socket de carte sd, j'ai ajouté une extension sur la forme originale issue du dessin 3D. Ce socket n'étant pas prévu pour la version finale, cette extension sera supprimée par la suite.

Pour cette première version, j'ai décidé de faire deux cartes électroniques, une pour la branche droite et une pour la branche gauche, ainsi pour simplifier la conception j'ai utilisé le même schéma électronique pour les deux cartes. Cependant les composants utilisés n'étant pas symétriques, les dispositions sur les circuits électroniques ne peuvent être les mêmes.

a. Placement des composants

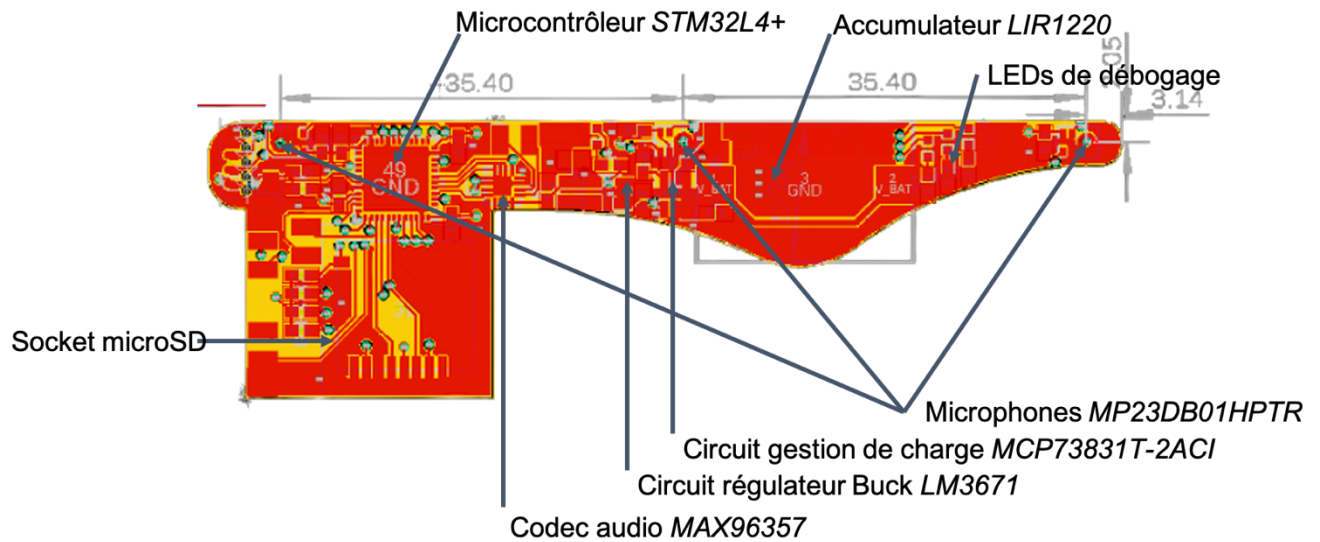


Figure 10 : Placement des composants sur le circuit imprimé

Les composants sont disposés par ordre d'impérativité de placement, avec en premier lieu les microphones et l'accumulateur. Les circuits de charge et de régulation devant être proche de l'accumulateur ils sont positionnés juste à côté avec des pistes très larges pour éviter les pertes en ligne et l'impact des perturbations.

Enfin j'ai placé le microcontrôleur, le codec audio et les LEDs là où de la place était disponible.

b. Plans de cuivre

Un PCB est constitué d'un empilement de couches de cuivre et de substrat de fibre de verre *FR4*. Les PCBs les plus simples sont souvent constitués de deux couches de cuivre qui prennent le substrat en sandwich, cependant cette conception simple n'est pas très robuste aux perturbations. Pour optimiser cet aspect on place deux plans de cuivre supplémentaires pour transmettre la puissance, ces deux plans étant séparés par du substrat, on obtient un effet capacitif qui permet de découpler légèrement l'alimentation.

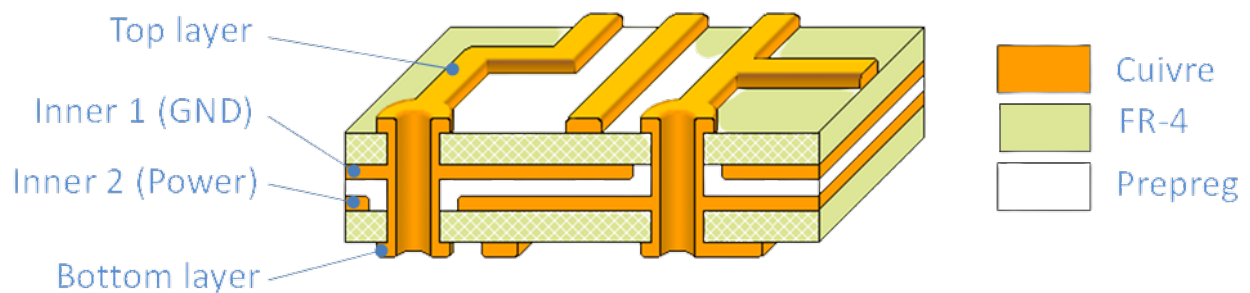


Figure 11 : Disposition des couches du PCB

Voici donc la disposition retenue avec le nom des couches utilisées sous Eagle.

c. Routage des pistes

Le routage des pistes consiste à définir la disposition des connexions entre les composants sur le circuit imprimé. Si le schéma électronique a été correctement réalisé, seules les connexions du schéma sont possibles lors du routage. Les points d'attention lors de cette étape sont donc la largeur et la longueur des pistes qui influent fortement sur le bruit, la *CEM Compatibilité ElectroMagnétique*, ainsi que la chauffe à cause de la puissance dans les pistes.

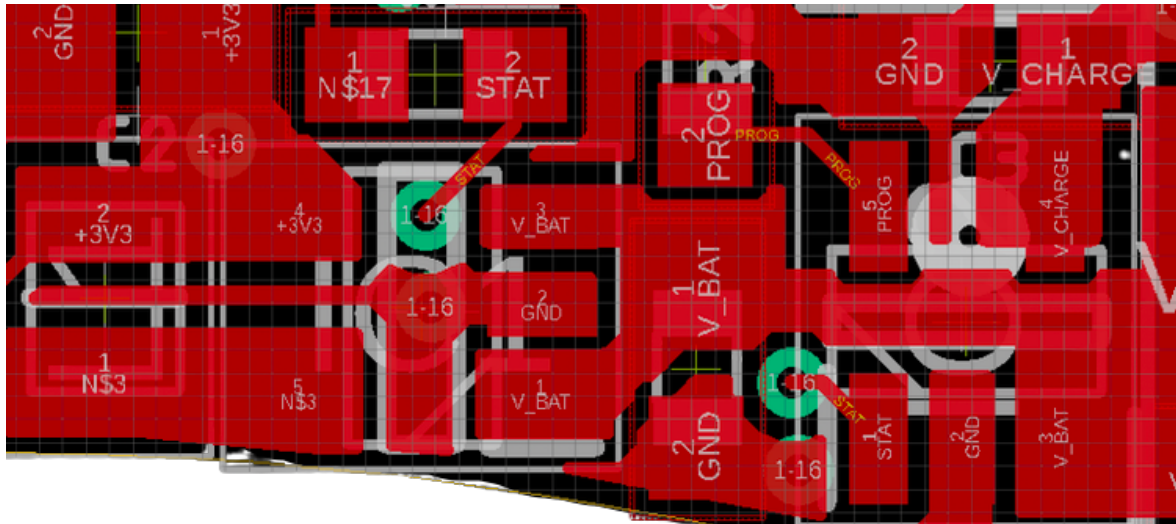


Figure 12 : Routage de la partie alimentation

Ainsi, la partie alimentation est critique de ce point de vue, les sources de tensions doivent être filtrées et les composants placés les plus proches les uns des autres pour éviter la propagation du bruit.

d. Fabrication et montage du circuit imprimé

Pour des raisons de rapidité de production, nous avons choisi de faire fabriquer le circuit imprimé *PCB* d'un côté et de commander les différents composants de l'autre.

Pour des soucis de compacité dans la branche de lunette, j'ai choisi de réaliser le PCB en 0.8mm d'épaisseur contrairement aux 1.6mm habituels, c'est le seul point qui diffère de la fabrication habituelle de PCB.

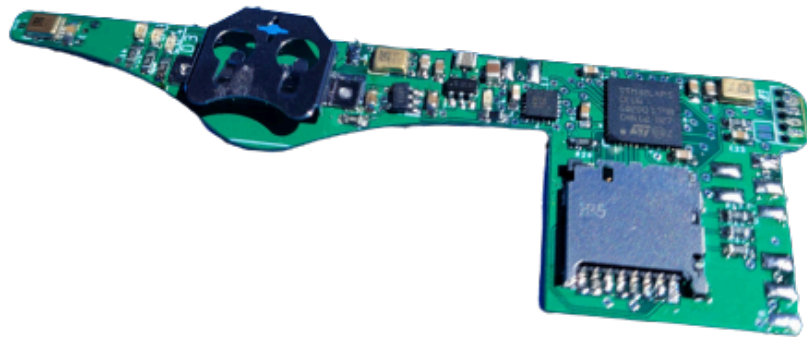


Figure 13 : Circuit imprimé monté et soudé

Une fois les PCB reçus, ceux-ci ont été montés manuellement par mes soins.

5. Tests et programmation de la carte

Une fois la carte montée, on peut passer à l'écriture du code puis à la programmation. La programmation est effectuée avec CubeIDE.

a. Projet CubeIDE

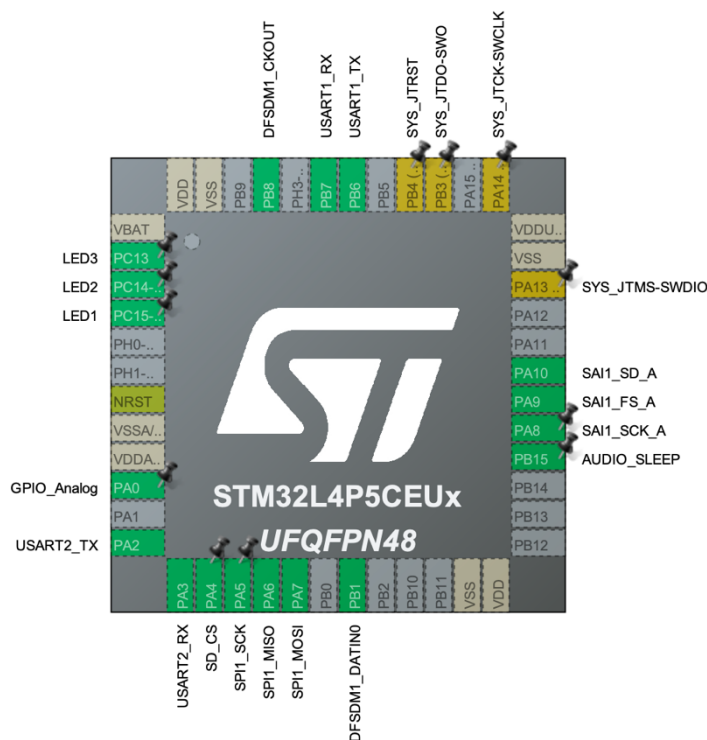


Figure 14 : Entrées/Sorties du STM32 dans CubeMX

Le code est écrit sur cubeMX, un environnement de développement développé par ST et qui dispose d'une interface graphique nommée **Device Configuration Tool** pour la configuration bas-niveau via un fichier *.ioc*. Le code de configuration est alors automatiquement écrit dans le fichier *main.c* du projet.

b. Tests programmation

Une fois cette première version de PCB montée, j'ai pu procéder à différents tests unitaires afin de valider les différentes fonctionnalités de la carte. Le premier test effectué est celui de la programmation, pour cela j'ai écrit un code très simple qui allume une LED. J'ai donc réalisé un programmeur basé sur le ST-Link V2 et une pince SOIC.

Le pinout suivant a été utilisé pour connecter la pince SOIC au programmeur et à une puce FTDI.

Pin	1	2	3	4	5	6	7	8
SWD	VCC	UART TX	UART RX	GND	SWDIO	SWO	SWCLK	RESET

Une fois que le fonctionnement des périphériques de base est validé on peut tester plus en profondeur les différentes fonctionnalités que l'on souhaite utiliser pour ce projet.

c. Retour sonore par écouteur

Dans le mini projet effectué en amont j'ai eu l'occasion d'utiliser un codec et c'est donc pour cette raison que j'ai choisi d'en réutiliser un dans ce montage. De plus l'utilisation d'un codec permet normalement d'obtenir une meilleure qualité sonore. Sur le composant, il y'a également un signal d'activation *ENABLE* que j'ai relié au microcontrôleur afin de pouvoir désactiver le codec et ainsi être économe en énergie. Cependant lors de l'activation du signal *ENABLE* toute la carte s'éteignait, le problème peut être dû à une soudure mal réalisée ou au composant en lui-même qui serait trop énergivore.

J'ai donc choisi d'utiliser le *CNA Convertisseur Numérique Analogique* intégré au microcontrôleur en connectant directement l'écouteur dessus.

▼ DAC Out1 Settings

Output Buffer	Enable
Trigger	Timer 6 Trigger Out event
Wave generation m...	Disabled
User Trimming	Factory trimming
Sample And Hold	Sampleandhold Disable

Figure 15 : Paramètres retenus pour le DAC (CNA)

On observe que j'ai utilisé le Timer 6 pour déclencher une conversion ainsi, on peut initialiser un tableau avec les valeurs que l'on souhaite en sortie, le *CNA* génère alors une tension analogique en passant à la valeur suivante à chaque événement du Timer. Cela nous permet ainsi de choisir la fréquence des échantillons et donc en la réglant correctement on peut retrouver la même que celle des échantillons en sortie du DFSDM.

d. Utilisation de FatFS pour la carte SD

L'un des objectifs de ce prototype est de pouvoir réaliser des enregistrements de son sur une carte microSD pour que Manuel puisse travailler dessus en *hors-ligne* avec son ordinateur.

Comme expliqué précédemment le socket de la carte SD est connecté avec le microcontrôleur en SPI. Cette utilisation n'étant pas conventionnelle, je me suis inspiré du travail (Kiwih, 2019) sur GitHub pour effectuer les modifications nécessaires. En effet, la gestion de carte SD nécessite d'avoir une communication SPI fonctionnelle via le *HAL Hardware Abstract Layer* fourni par STMicroelectronics.

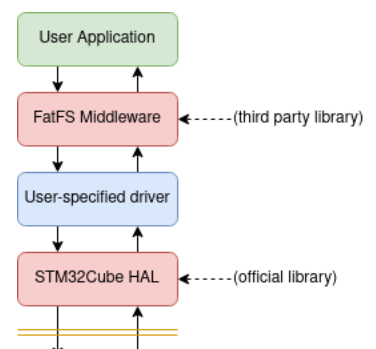


Figure 16 : Pile logicielle pour la carte SD

Il est également nécessaire de prendre en charge la gestion du système de fichier *FAT*. Pour cela, on utilise le Middleware *FatFS* associé avec des fonctions définies par moi-même pour faire le lien entre *FatFS* et le *HAL* de ST.

e. Mise en œuvre du DFSDM

Nous avons précédemment vu que le périphérique *DFSDM* du STM32 permet de convertir les signaux PDM en sortie des microphones vers des signaux PCM utilisables dans les algorithmes classiques.

Le DFSDM permet de connecter jusqu'à deux microphones par pin en alternant les données d'un micro et de l'autre.

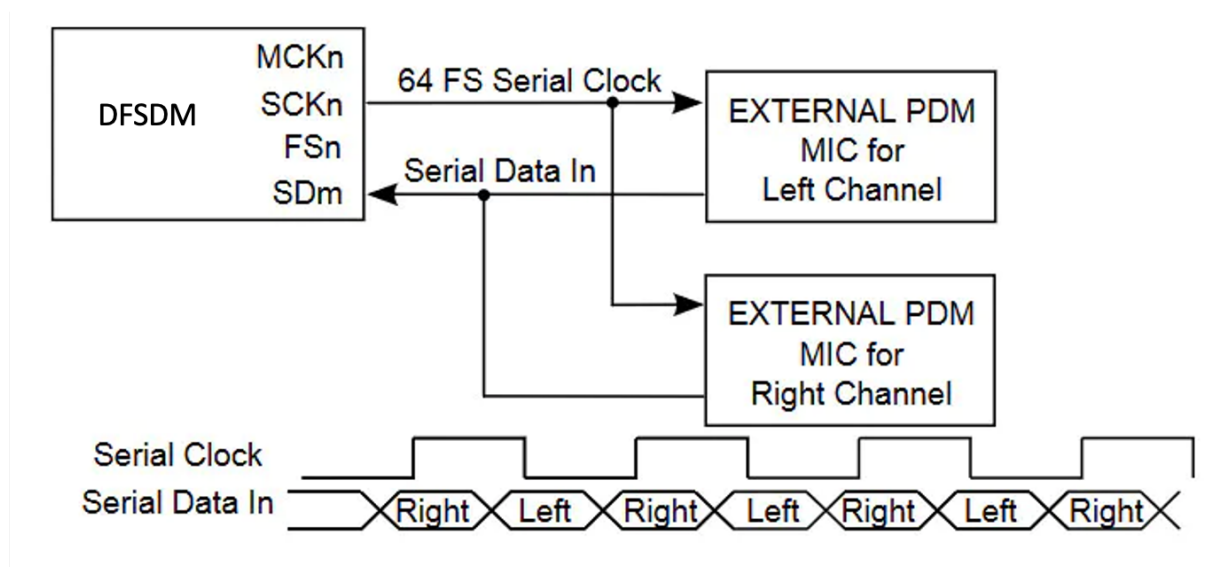


Figure 17 : Alternance des données en provenance des microphones (JayV, 2019)

Cette facilité d'usage nous permet de router moins de pistes jusqu'au microcontrôleur et d'utiliser moins de pins du microcontrôleur, dans notre cas c'est le premier avantage qui nous intéresse, tous les pins du microcontrôleur n'étant pas utilisés.

L'horloge est partagée entre tous les microphones et les deux micros avant partagent le même signal de retour.

Ainsi, l'interface PDM fournit des données brutes. Dans ce projet, j'utilise un filtre FIR Sinc avec un ratio de 64 comme décimateur pour reconstruire les données PDM en données PCM. Le filtre est instancié dans une boucle d'opération de décalage de 16 bits, où chaque 64-bits est filtré. Les données PCM sont ainsi enregistrées dans un tableau de 2 valeurs (stéréo). On peut ensuite utiliser ces valeurs comme un flux PCM classique pour le traitement audio.

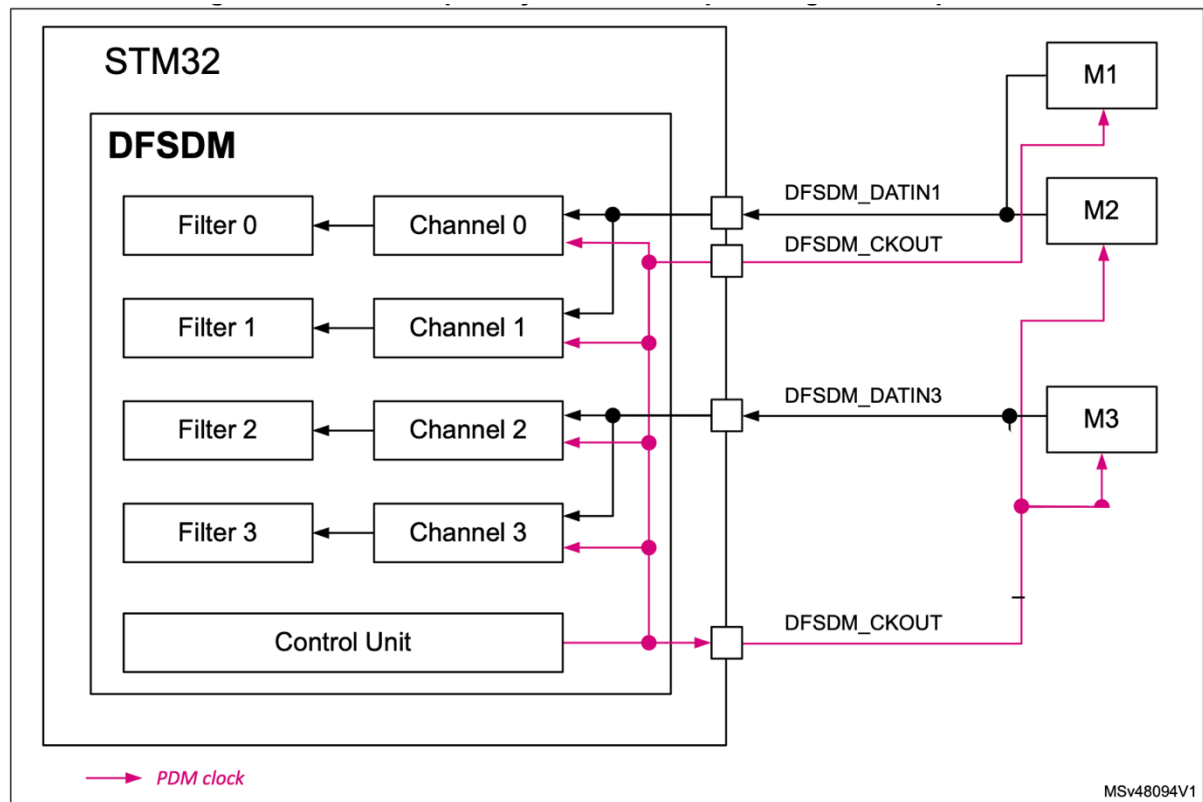


Figure 18 : Schéma du DFSDM et des connexions avec les microphones (STMicroelectronics, Interfacing PDM digital microphones using STM32 MCUs and MPUs, 2019)

6. Conclusion et suite du projet

Ce stage m'a permis d'appliquer à une problématique concrète les outils acquis lors de mes études à l'ENS, principalement en électronique et informatique industrielle. La grande liberté offerte par Manuel sur les choix technologiques du projet m'a permis de mettre en application mes nombreuses idées parfois inappropriées au problème rencontré mais avec un résultat toujours instructif. La variété des sujets abordés m'a permis de mieux réaliser l'ampleur d'un projet mené de bout en bout, cette diversité a été motivante, les différents axes de travail permettant de rester investi.

Le développement des lunettes Pulse est aujourd'hui loin d'être terminé, la partie électronique a été dégrossie mais les premières cartes fabriquées pendant le stage ne m'ont pas donné satisfaction. La majorité des fonctionnalités ont été validées individuellement mais je n'ai pas pu sauvegarder des sons enregistrés avec les microphones de la carte. Cet objectif sera sans doute atteint avec la version 2. Le développement a sans doute été ralenti par une implémentation laborieuse de certaines fonctionnalités, nous avons fait le choix de souder nous même les composants sur les cartes électroniques mais cela a introduit une incertitude supplémentaire lors du débogage des fonctionnalités posant problème en plus des erreurs de code ou de routage électronique.

Le projet engagé avec Manuel au début du stage avait pour vocation d'être poursuivi au-delà des simples dates du stage. Nous avons donc décidé que je continuerai de travailler avec Pulse sur mon temps libre pour l'année qui arrive dans la mesure du possible. La problématique sur laquelle nous allons travailler en premier lieu est l'enregistrement de données audio avec un prototype v2 afin que Manuel puisse essayer différents algorithmes en *hors-ligne* sur son ordinateur. Ensuite nous tenterons d'implémenter un traitement audio temps réel sur la carte (STMicroelectronics, Microphone array beamforming in the PCM and PDM domain, 2018).

Je tiens au terme de ce stage à remercier toutes les personnes qui m'ont accompagnées dans ce travail. En premier lieu, merci à Manuel Pariente de m'avoir encadré tout au long de ces 3 mois, tes conseils et tes idées sont le moteur de ce projet, tu as su me recadrer lorsque je m'éloignais des objectifs principaux tout en me laissant une autonomie certaine. Ta réactivité et ton envie de comprendre chaque aspect du projet sont inspirantes. Je te remercie également de ta confiance pour poursuivre le développement commencé ensemble. Je remercie aussi Ayoub Zahir qui m'as suivi tout au long du projet et m'as apporté son expertise pour me débloquer et m'aider à avancer, nos réunions ont toujours été des moments très instructifs. Enfin, je remercie le LORIA qui a hébergé et financé ce stage, les moyens mis à ma disposition m'ont permis de réaliser les premiers prototypes de Pulse.

Bibliographie

- JayV. (2019). *DFSDM*. Récupéré sur Hackster.io: <https://www.hackster.io/voske65/arduino-zero-pdm-microphone-9136cb>
- Kiwih, H. (2019). *An SD card over SPI using STM32CubeIDE and FatFS*. Récupéré sur 01001000 Projects by Kiwih: <https://01001000.xyz/2020-08-09-Tutorial-STM32CubeIDE-SD-card/>
- Merrett, S. (2020). *SOICbite as programming interface*. Récupéré sur GitHub: <https://github.com/SimonMerrett/SOICbite>
- STMicroelectronics. (2018). *Microphone array beamforming in the PCM and PDM domain*. Récupéré sur Application Note: https://www.st.com/resource/en/design_tip/dm00528068-microphone-array-beamforming-in-the-pcm-and-pdm-domain-stmicroelectronics.pdf
- STMicroelectronics. (2019). *Interfacing PDM digital microphones using STM32 MCUs and MPUs*. Récupéré sur Application Note: https://www.st.com/resource/en/application_note/dm00380469-interfacing-pdm-digital-microphones-using-stm32-mcus-and-mpus-stmicroelectronics.pdf

Annexe

Mini Projet STM32 audio

Introduction

Le but du projet est de mettre en place une solution de traitement audio embarquée sur microcontrôleur (noté uC).

Le code est écrit sur cubeMX, un environnement de développement développé par ST et qui dispose d'une interface graphique nommée **Device Configuration Tool** pour la configuration bas-niveau via un fichier *.ioc*. Le code de configuration est alors automatiquement écrit dans le fichier *main.c* du projet.

Pour mieux comprendre le code et les différentes parties du projet, il est important de prendre en compte les différentes *strates* logicielles.

1. raw, on modifie les registre directement, trop complexe au vue des possibilités offertes par le uC.
2. HAL driver, basée directement sur la couche raw, elle met à disposition des fonctions simples qui permettent d'écrire un code plus lisible. Ces librairies sont différentes pour chaque famille de uC (F4XX,F7XX,...). Ces librairies sont générées par le **Device Configuration Tool** et utilisées par le code générée par celui-ci.
3. BSP driver, code plus haut niveau mais également plus spécifique, il est différent pour chaque uC. Les fonctions de ces librairies se basent sur les librairies HAL et les caractéristiques raw du uC. Ces librairies permettent d'utiliser facilement des périphériques complexes facilement. Les fonctionnements sous-jacents sont donc inconnus, l'utilisation de ces librairies peut-être bloquante pour une utilisation poussée des périphériques(cf ci-dessous).



Decryptage demo ST

La première partie du projet a été de prendre en main l'exemple fourni par ST, celui-ci permet de mettre en place un loopback audio. Après vérification du fonctionnement, on observe son fonctionnement.

```
/* Initialize Audio Recorder */
BSP_AUDIO_IN_OUT_Init(INPUT_DEVICE_DIGITAL_MICROPHONE_2, OUTPUT_DEVICE_HEADPHONE);

/* Initialize SDRAM buffers */
memset((uint16_t*)AUDIO_BUFFER_IN, 0, AUDIO_BLOCK_SIZE*2);
memset((uint16_t*)AUDIO_BUFFER_OUT, 0, AUDIO_BLOCK_SIZE*2);
audio_rec_buffer_state = BUFFER_OFFSET_NONE;

/* Start Recording */
BSP_AUDIO_IN_Record((uint16_t*)AUDIO_BUFFER_IN, AUDIO_BLOCK_SIZE);

/* Start Playback */
BSP_AUDIO_OUT_SetAudioFrameSlot(CODEC_AUDIOFRAME_SLOT_02);
BSP_AUDIO_OUT_Play((uint16_t*)AUDIO_BUFFER_OUT, AUDIO_BLOCK_SIZE * 2);
```

On constate donc qu'en premier lieu la démo utilise les bibliothèques BSP pour la gestion des flux audio. Ces bibliothèques reposent ensuite sur le composant de communication SAI du uC.

Sur le stm32f746, le périphérique SAI 2 est connecté au driver audio wm8994 la voie A est utilisée pour les sorties (jack et HP), la voie B est utilisée pour les micros.

En étudiant plus précisément la fonction d'init,

```
BSP_AUDIO_IN_OUT_Init(INPUT_DEVICE_DIGITAL_MICROPHONE_2, OUTPUT_DEVICE_HEADPHONE,
```

on comprend mieux les différents périphériques mis en œuvre.

Cette fonction initialise d'abord l'horloge du bus SAI en fonction de la fréquence d'échantillonnage souhaitée. Puis le bus SAI est initialisé avec les différents paramètres, et en utilisant les deux voies simultanément pour les micros et les sorties son.

On retiendra les éléments suivants

```
/* Only MICROPHONE_2 input supported */

/* Configure SAI in master mode :
 * - SAI2_block_A in master TX mode
 * - SAI2_block_B in slave RX mode synchronous from SAI2_block_A
 */
```

On peut également noter que les données du bus SAI sont transmises via DMA. (Direct Access Memory, le périphérique SAI du uC a directement accès à la mémoire (SDRAM dans notre cas) sans intervention du CPU.)

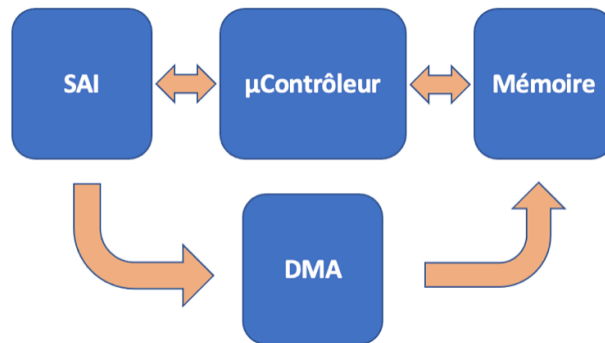


Schéma du fonctionnement du DMA

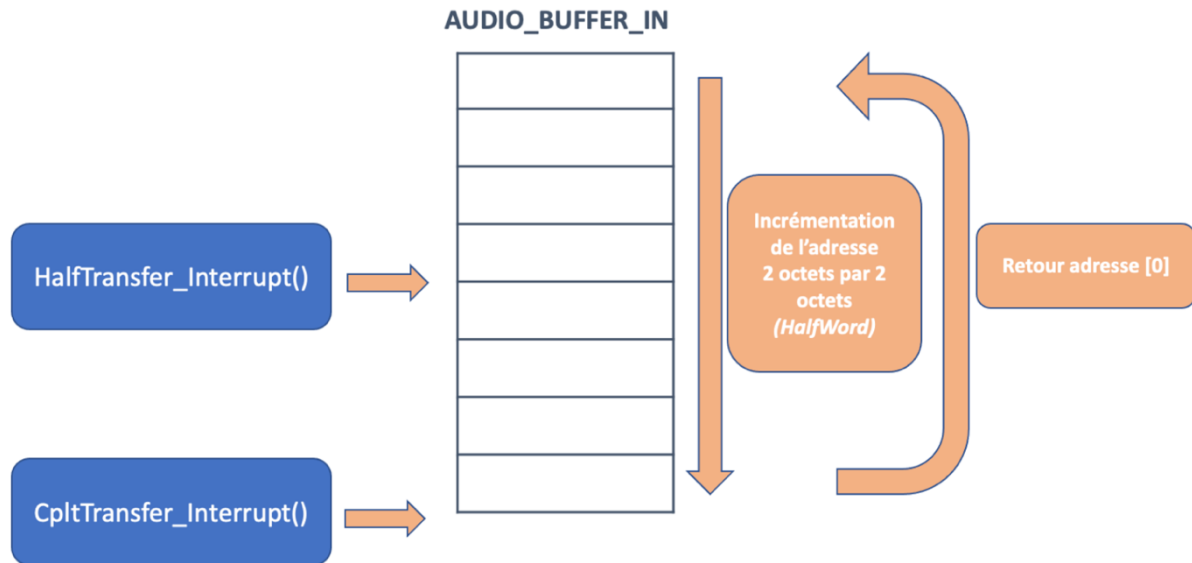
Les fonctions `BSP_AUDIO_IN_Record` et `BSP_AUDIO_OUT_Play` ont pour unique effet d'activer le DMA.

Implementation dans un projet CubeMX

La demo loopback ST était une partie d'un programme avec de multiples fonctionnalités, la deuxième partie du travail a donc été d'isoler les fonctionnalités utiles au sein du projet initial pour obtenir le code le plus simple possible. Cette partie s'est déroulée sans encombre, mais le code en résultant avait certaines spécificités étranges. Il ne fonctionnait que si il y'avait une lecture d'un bouton dans la boucle while d'attente des DMA (On attend que le buffer du micro soit plein et on le copie dans le buffer des HP).

La suite logique a donc été de mettre en place un projet propre tout neuf sous cubeMX afin de ne pas être encombré par les restes du précédent projet et de pouvoir connaître et maîtriser l'intégralité du code (hors librairies).

Le problème auquel j'ai été confronté est que la librairie BSP audio configure presque tout mais il a néanmoins fallu que, dans le Device Configuration Tool, j'active le périphérique SAI2 avec le channel A en maître et le B en esclave synchrone. IL faut également que les DMA soient correctement réglés (Circular, Half Word).



Fonctionnement du DMA en mode circular HalfWord

Finalement, on obtient une base de travail fonctionnelle et robuste pour tester diverses implementations. Une branche Git sera créée pour les différents essais.

Fin de la prise en main

Il est à retenir de cette partie que la librairie BSP est spécifique au uC utilisé et les réglages étant épars, il est à craindre des problèmes de portabilités lors d'un probable passage sur un uC plus adapté. *Il est possible que les librairies BSP ne soient pas disponibles/compatibles avec un format différent(plus de micros, driver différent...)*

Néanmoins, le code est propre et le format de données compris, en effet les buffer d'entrées sorties sont sous la forme :

```
| int16 R[0] | int16 L[0] | int16 R[1] | int16 L[1] | ... | int16 R[size-2] | int16 L[size-2] | int16 R[size-1] | int16 L[size-1] |
```

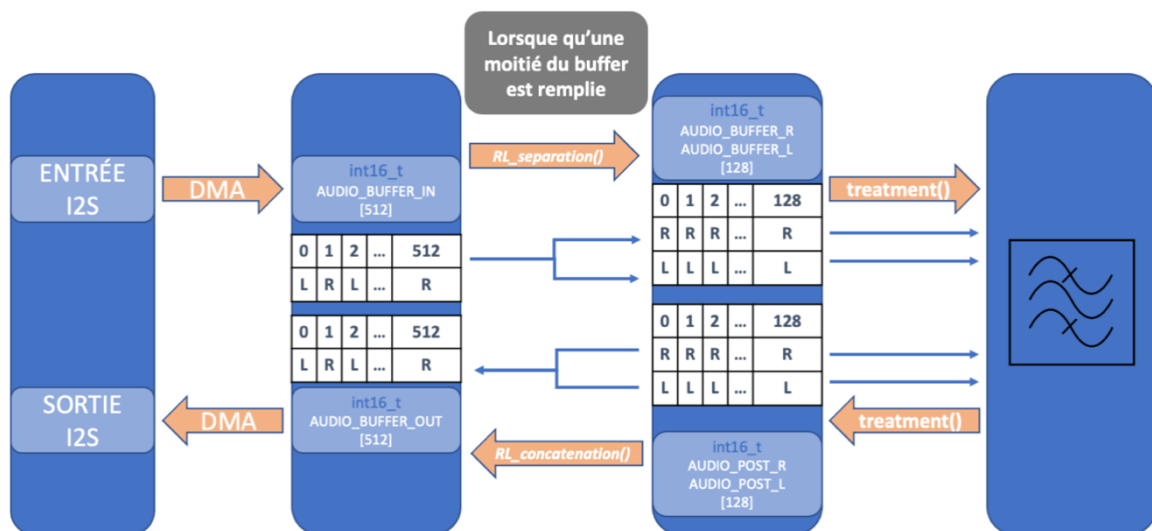
avec size la taille choisie pour les buffer.

Premiers traitements

Le but du projet est de mettre en place un algorithme de beamforming, le beamforming est un filtrage spatial. Dans notre cas d'usage, le but est de récupérer le son provenant d'une direction uniquement. Pour cela il nous faut séparer les signaux audio droits et gauches afin de pouvoir effectuer des opérations dessus.

```
void RL_sep(int16_t* buffer, int16_t size){
    uint16_t j=0,k=0;
    for(size_t i=0;i<size;i++){
        if(i%2){
            ((int16_t*)AUDIO_BUFFER_R)[j] = buffer[i];
            j++;
        }else{
            ((int16_t*)AUDIO_BUFFER_L)[k] = buffer[i];
            k++;
        }
    }
}
```

Voici donc un schéma du fonctionnement global



Chaîne d'information du système

Comme on peut le voir, il y'a de nombreuses étapes de copie de tableau, si le beamforming est coûteux, il sera intéressant de minimiser ces étapes.

Avant de travailler les algorithmes de traitement potentiels, voici un schéma du positionnement des micros et la convention angulaire choisie.

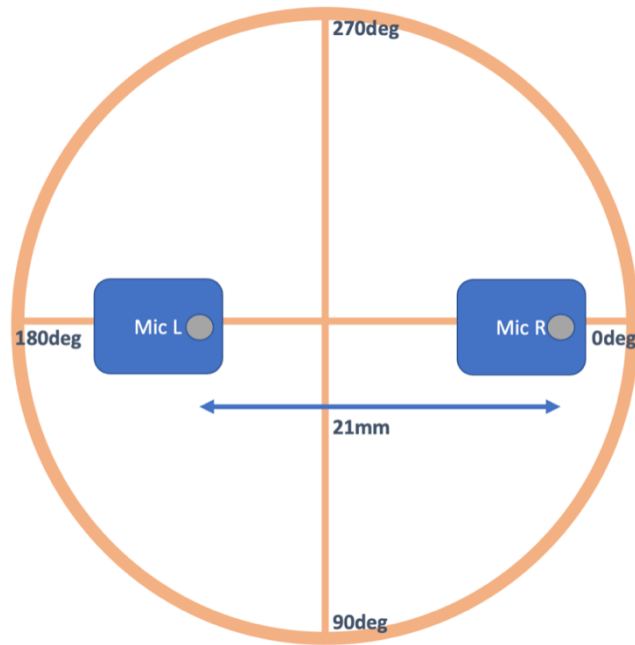


Schéma des micro de la carte Discovery

Delay and Sum Beamforming

J'ai en premier lieu essayé d'implémenter une forme simple de beamforming qu'est le delay and sum.

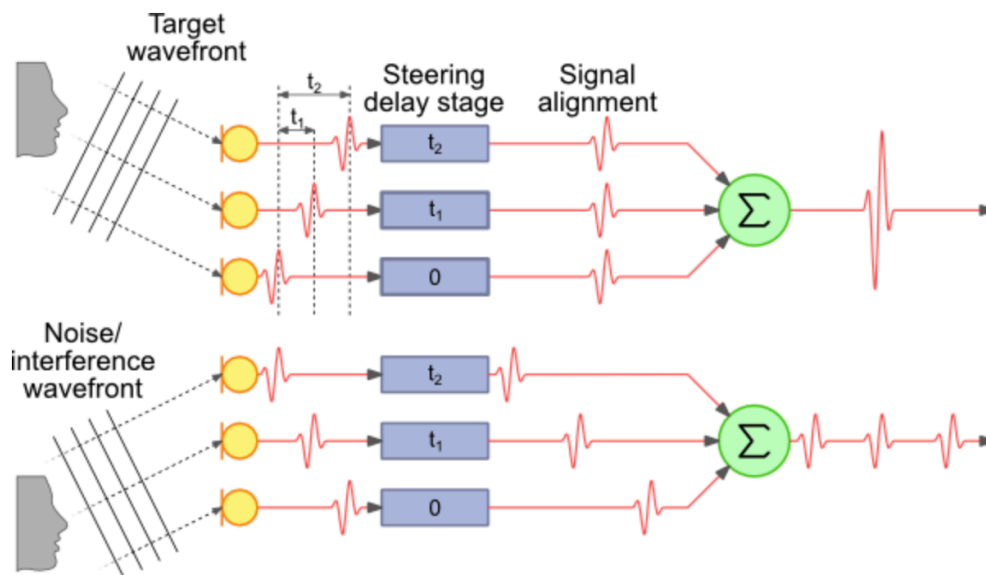
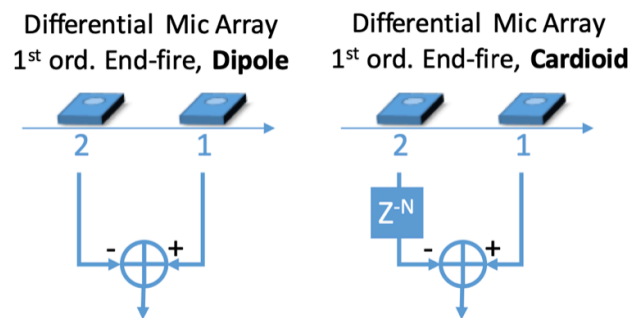


Schéma du delay and sum beamforming

Cependant comme on peut le voir sur le schéma, l'atténuation maximale dépend du nombre de micro dont on dispose ainsi que de la fréquence. La carte *STM32F746G* utilisée ne dispose que de deux micros et le résultat était donc quasiment imperceptible. Dans le but de vérifier le bon fonctionnement des étapes précédentes, j'ai décidé d'implémenter une solution de suppression des bruits en provenance d'un endroit donné.

Differential Microphone Arrays (DMA)

Le differential microphone arrays est une technique de filtrage qui permet de supprimer une source sonore en fonction de sa direction.

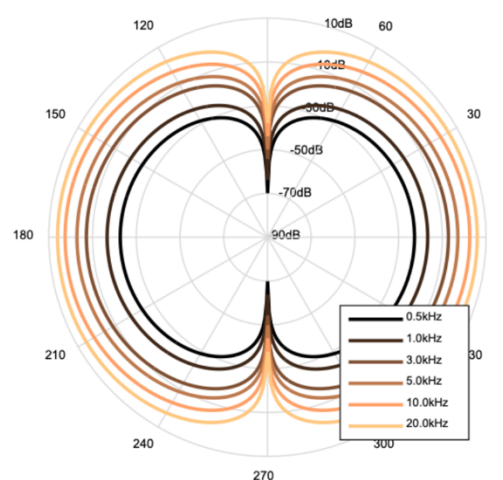


Exemples de réseau de microphones différentiels

L'inconvénient de ce type de système pour la suite du projet est sa très haute selectivité angulaire pour la suppression des sons. Mais on doit pouvoir inverser le phénomène je pense.

Dipole DMA

J'ai en premier lieu essayé le mode Dipole, très simple à implémenter, celui-ci a donné de bons résultats permettant efficacement d'éliminer une source en face des micros. Avec un très faible coût en calcul, puisqu'il s'agit d'une simple soustraction entre les données du micro droit et les données du micro gauche.

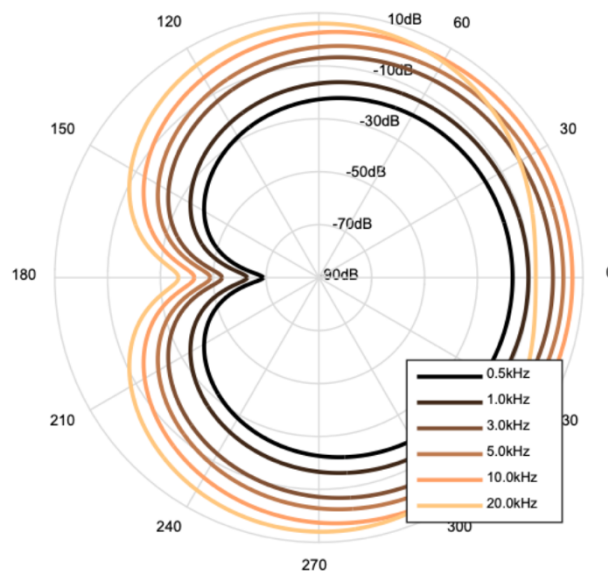


Tracé logarithmique de l'intensité en fonction de la fréquence à un angle donné

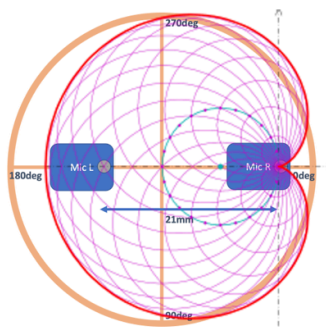
Comme on peut le voir sur le graphique, le filtrage est purement symétrique, en effet si on se reporte au schéma de placement des micros, rien ne différencie un son émis à 90 degrés d'un son émis à 270 degrés.

DMA Cardioid

Puis j'ai implémenté le mode Cardioid qui consiste à delayer le signal d'un des micros pour supprimer le son en provenance du 0 degrés.



Tracé logarithmique de l'intensité en fonction de la fréquence à un angle donné



Placement des micros et gain en réception en fonction de l'angle

Les résultats obtenus avec ce BSS sont impressionnants

https://cnl.salk.edu/~tewon/Blind/blind_audio.html

if $F_s = 16 \text{ kHz}$, $T_s = 62.5 \mu\text{sec}$; with $N = 1$, $T = 62.5 \mu\text{sec}$, $D = 21.4 \text{ mm}$ and $f = 8 \text{ kHz}$ (bandwidth of interest centered around 4 kHz).