|  |  |  |  |
| --- | --- | --- | --- |
| Charte et Logo de l’entreprise | | | |
| **PFE : « Traitement de signal audio embarqué temps réel sur carte STM32 »**  **FIPA 2021** | | | |
|  | ENSTA Bretagne  2 rue F. Verny  29806 Brest Cedex 9, France  JÉZÉGOU Pierre-Yves,  [pierre-yves.jezegou@ensta-bretagne.org](mailto:pierre-yves.jezegou@ensta-bretagne.org) |  | Contact :  Reynet Olivier,  [olivier.reynet@ensta-bretagne.fr](mailto:olivier.reynet@ensta-bretagne.fr)  Probst Irvin,  [irvin.probst@ensta-bretagne.fr](mailto:irvin.probst@ensta-bretagne.fr) |

# Remerciements

# Résumé

# Abstract

Sommaire

[Remerciements 2](#_Toc88309054)

[Résumé 3](#_Toc88309055)

[Abstract 3](#_Toc88309056)

[Sommaire 4](#_Toc88309057)

[Introduction *(3e livrable, 2/3 de page)* 5](#_Toc88309058)

[1. Objectifs du projet 6](#_Toc88309059)

[2. Plateforme Embarquée. 7](#_Toc88309060)

[2.1. Carte de développement 7](#_Toc88309061)

[2.2. Chaine de capture 7](#_Toc88309062)

[2.2.1. Le microphone 7](#_Toc88309063)

[2.3. Sortie audio 9](#_Toc88309064)

[3. Conversion PDM → PCM 12](#_Toc88309065)

[3.1. Les échantillons 12](#_Toc88309066)

[3.1.1. PCM 12](#_Toc88309067)

[3.1.2. PDM 13](#_Toc88309068)

[3.2. Chaine de filtrage 13](#_Toc88309069)

[3.3. Intégration au microcontrôleur 16](#_Toc88309070)

[4. Les démonstrateurs 19](#_Toc88309071)

[4.1. Présentation des démonstrateurs : 19](#_Toc88309072)

[4.1.1. « Parrot » 19](#_Toc88309073)

[4.1.2. « Digital recorder » 20](#_Toc88309074)

[4.1.3. « Direct output » 21](#_Toc88309075)

[Bibliographies 23](#_Toc88309076)

[Glossaires des termes techniques 23](#_Toc88309077)

[Table des figures 24](#_Toc88309078)

[Annexes 25](#_Toc88309079)

[Annexe 1 : Logigrammes des États démonstrateur "Parrot" 25](#_Toc88309080)

[Annexe 2 : Logigrammes des États démonstrateur "Digital Recorder" 25](#_Toc88309081)

[Annexe 3 : Logigrammes des États démonstrateur "Direct Output" 25](#_Toc88309082)

# Introduction *(3e livrable, 2/3 de page)*

* Introduction générale du projet
* Présentation de la structuration du rapport

# Objectifs du projet

Ce projet de fin d’études a pour objectif de préparer de futurs projets étudiants de l’ENSTA Bretagne centrés sur l’acquisition audio sur plateforme STM32 avec des microphones PDM. On s’intéressera donc à la cible STM32 qui est aujourd’hui un standard de l’industrie embarquée. Ainsi qu’aux microphones numériques PDM qui représente une innovation intéressante par son faible bruit et donc son excellent rapport signal sur bruit.

Le projet se contrera donc sur la production d’une configuration clé en main de la carte STM32 afin :

* D’acquérir un signal numérique audio en utilisant le DMA du processeur et en effectuant le filtrage des échantillons PDM en provenance du microphone
* Générer un son préalablement acquis via le microphone PDM sur le DAC du processeur

Pendant le projet, on s’intéressera plus particulièrement aux réglages du microphone afin de maitriser :

* L’échantillonnage des signaux PDM et PCM ainsi que les fréquences associées
* Le filtrage PDM → PCM afin de s’affranchir de la librairie « boite noire » fournie par ST Microelectronics pour le filtrage
* La gestion des formats audio sur la carte cible (production de fichiers WAV)
* Le fonctionnement global de la chaine d’acquisition

Les productions techniques du projet se matérialiseront par la production de démonstrateurs mettant en œuvre la chaine d’acquisition. Dans ces démonstrateurs au moins un aura le comportement suivant :

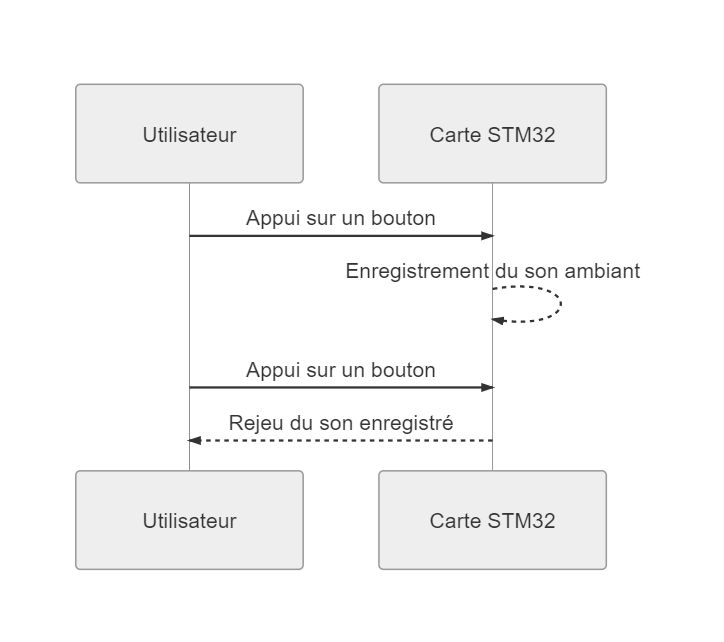
.

Figure  : Comportement du démonstrateur obligatoire

# Plateforme Embarquée.

## Carte de développement

Le Projet se base sur la carte de découverte commercialisée par ST Microelectronics : "STM32F429I-DISC1". Cette carte intègre un microcontrôleur 32 bits STM32F429xx. Ce microcontrôleur fait partie de la série de microcontrôleurs hautes performances : STM32F4. Cette série se base sur un cœur ARM Cortex M4. Cette carte permet de se familiariser avec l’architecture ainsi que de concevoir des solutions intégrant un microcontrôleur de la famille STM32 qui est un standard de l’industrie embarqué. Cette carte intègre en plus du microcontrôleur, une interface de programmation ST-LINKV2, un écran LCD 2.4", plusieurs LED de débogage, 2 boutons poussoirs, un port micro-USB type B compatible OTG, une puce de SDRAM de 64 Mbit et un Gyroscope MEMS.

Une image contenant texte, équipement électronique

Description générée automatiquement

Figure  : Carte Cible (STM32F429I-DISC1)

## Chaine de capture

### Le microphone

Le projet a pour objectif principal de lever les incertitudes sur l’utilisation de microphone PDM pour la capture audio sur une plateforme utilisant un microcontrôleur STM32. En effet, les microphones PDM sont une innovation intéressante pour la capture audio, car grâce à leur nature numérique ils sont peut influencés par les bruits électromagnétiques de leur environnement, permettant ainsi d’obtenir un rapport signal sur bruit important à moindre coût. Un microphone PDM est généralement structuré comme suit :

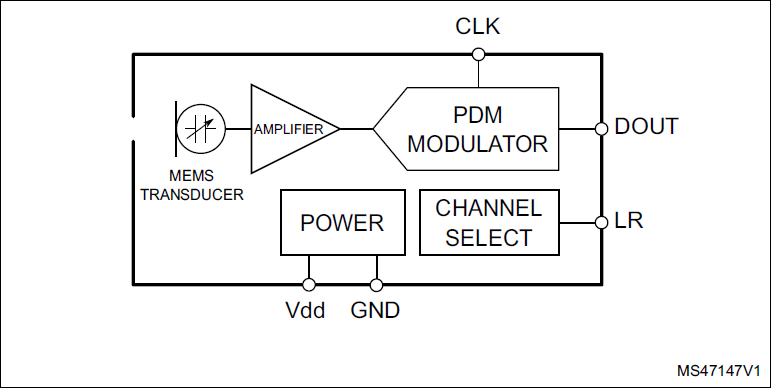


Figure  : Schéma bloc typique d’un microphone PDM

Sur la Figure 3, on a plusieurs blocs qui remplissent un rôle :

* MEMS Transducer:

Le transducteur MEMS est un condensateur dont la capacité va varier en fonction des variations de la pression locale de l’air, qui est caractéristique au déplacement d’une onde sonore.

* Amplifier

L’amplificateur va amplifier le signal en provenance du transducteur afin de fournir un signal adapté au modulateur PDM.

* PDM Modulator

Le Modulateur PDM va à chaque front de l’horloge (CLK) convertir le signal analogique en provenance de l’amplificateur en un échantillon PDM (DOUT)

* Channel Select

Le sélecteur de canal informe influe sur le fonctionnement du modulateur PDM qui indique s’il doit convertir le signal sur un front montant ou descendant de l’horloge, et donc définir si le microphone influx sur le canal gauche ou droit d’un signal stéréo, ceci permet de multiplexer deux microphones (gauche et droite) sur un unique flux PDM

* Power:

Le bloc Power correspond à l’alimentation du microphone PDM

Pour le projet nous travaillerons dans une configuration mono pour le Microphone PDM ce qui nous donne les connexions électriques suivantes :

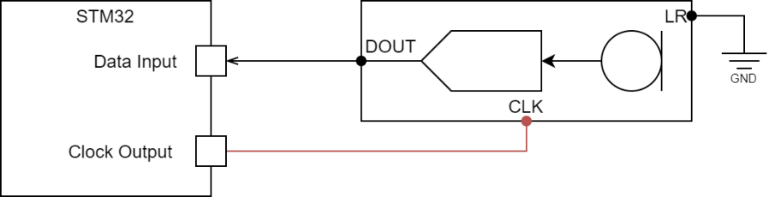


Figure  : Connexion du Microphone PDM en configuration mono

Notons que sur la Figure 4 le signal "LR" qui correspond au sélecteur gauche/droite est connecté directement à GND, indiquant que notre microphone travaille sur le canal gauche. Cependant, car nous travaillons en configuration mono, le canal utilisé par le microphone n’a pas d’importance.

Dans notre application, le microcontrôleur intègre un périphérique SAI (Serial Audio Interface) afin de générer les signaux nécessaires pour le contrôle des microphones PDM ainsi que l’acquisition en continu (sans influer sur la charge du processeur) des échantillons PDM quand il est couplé au DMA. Ce qui permet d’appliquer un filtre sur le signal PDM pour le convertir en PCM uniquement quand le DMA nous indique que suffisamment d’échantillons PDM ont été chargés en mémoire, permettant au microcontrôleur d’exécuter le reste du temps d’autres instructions. Ce qui nous donne la chaine de capture suivante :

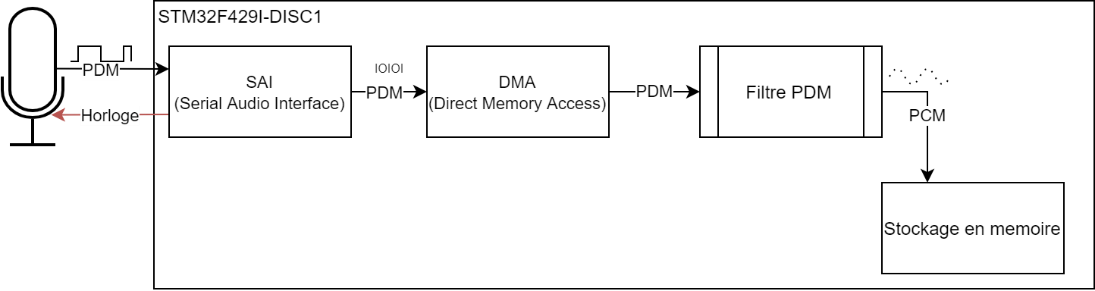


Figure  : Chaine de capture

## Sortie audio

Un deuxième objectif de projet et de reproduire les sons captés sur un haut-parleur/casque. On a étudié précédemment l’aspect capture des données du son, pour l’aspect reproduction on va utiliser le DAC intégrer au microcontrôleur afin de recréer le signal analogique correspondant aux sons captés. Le DAC intégrer à notre microcontrôleur à deux canaux de 12 bits, dans la configuration qu’il utilise sur notre carte il peut générer une tension comprise en 0.2 et → avec dans notre cas ce qui nous donne une amplitude maximale :

Et donc un quantum :

Ces deux valeurs du DAC nous indiquent que pour adapter le signal PCM filtrer il faudra produire des échantillons sur 12 bits et comme la sortie du DAC est unipolaire (0-3V) il faudra ajouter un offset de 2047 () afin de positionner le « zéro » de notre signal au milieu de l’excursion du DAC, ainsi, le haut-parleur agissant comme un filtre passe-bande dont la bande passante et comprise entre 20 Hz et 20 kHz cet offset est gommé, ce qui nous nous permet de conserver l’ensemble des valeurs négatives qui aurait été perdu sinon.

Maintenant que l’on a généré un signal électrique se pose la question de l’alimentation en courant du haut-parleur, en effet les haut-parleurs/casque ont une impédance très faible () ce qui fait que si on les connecter simplement ainsi :

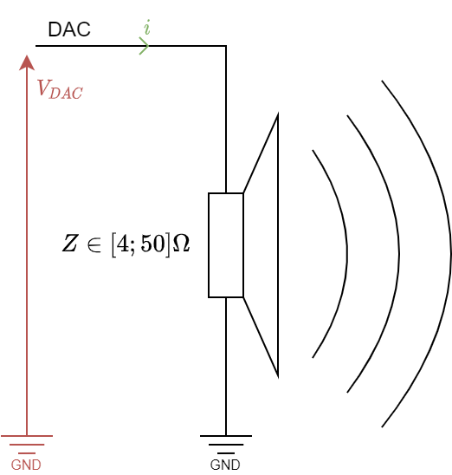


Figure  : Montage direct d’un haut-parleur sur le DAC

Dans ce montage on obtient avec  et une impédance du haut-parleur :

Or la documentation du microcontrôleur nous indique que le courant maximum qui peut être délivré par une sortie est et que l'ensemble des sorties peut fournir .

Donc pour pouvoir connecter un haut-parleur au DAC sans avoir à ajouter un étage d’amplification il faut augmenter l’impédance minimum du haut-parleur c’est pourquoi j’ai décidé de réaliser le montage suivant :

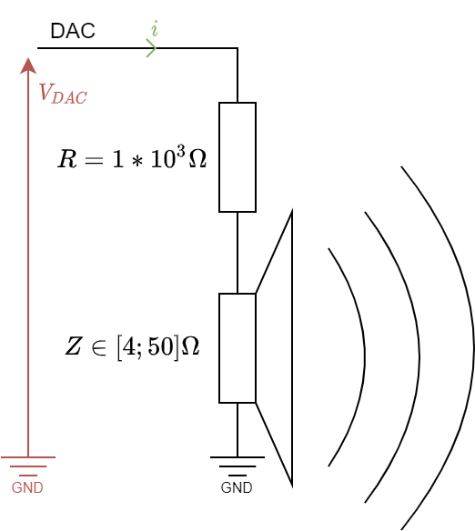


Figure : Branchement d’un haut-parleur en série avec une résistance de 1 kΩ sur le DAC

Dans ce montage on obtient avec   une impédance du haut-parleur :

Ce courant maximum donne une marge suffisante par rapport aux capacités du microcontrôleur pour s’assurer que le risque de destruction des ports du DAC est minimum.

Pour simplifier les connexions/déconnexions d’un haut-parleur/casque j’ai donc fabriqué un câble que l’on connecte sur les connecteurs Dupont de la carte et fournit une prise jack 3,5 mm stéréo préchargée d’une résistance de sur chacun des deux canaux. Ce câble suit le schéma suivant :

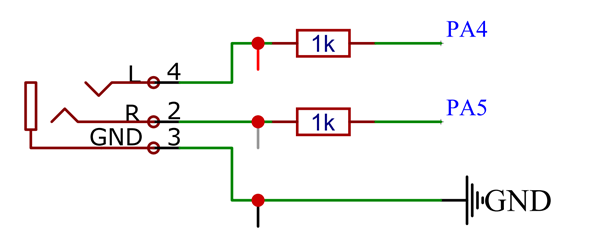
.

Figure : Schéma Électrique du câble DAC → Jack 3,5 mm

On remarque sur le schéma que j’ai fait sortir des fils pour connecter des sondes côté haut-parleur afin de mesurer les signaux reproduits par le DAC.

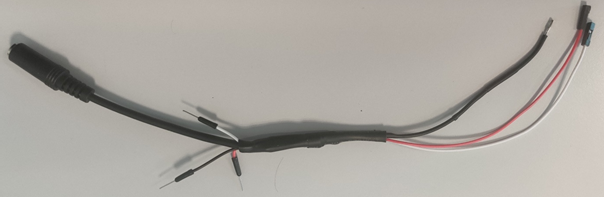


Figure : Câble DAC → Jack 3.5mm

Le câble ainsi fabriqué utilise le code couleur des connecteurs RCA (Blanc → Gauche ; Rouge → Droit) pour les fils transportant le signal et le fil Noir correspondant à la masse

Du point de vue du logiciel, le DAC est configuré pour convertir l’échantillon suivant de chaque canal quand il reçoit l’évènement de remplissage du conteur lié au Timer 2 du microcontrôleur. Dans le Timer 2, on peut configurer la fréquence qui correspond à la fréquence à laquelle l’évènement de remplissage du conteur apparait en configure deux valeurs :

* "Prescaler" (PSC)
* "Counter Period" (ARR)

On peut calculer ces deux valeurs grâce à l’expression suivante :

Avec qui correspond à la fréquence d’entrée du Timer.

Par exemple si l’on veut que le DAC ait une fréquence d’échantillonnage de 48 kHz () avec une fréquence . On fixe et donc on obtient :

Pour fournir les échantillons aux deux canaux du DAC, on va utiliser le DMA qui à partir d’un buffer circulaire pour chaque canal fournit les échantillons au DAC en continu, Le DMA levant deux interruptions, une première quand il a lu la moitié du buffer et une seconde quand il atteint la fin. Interruptions qui sont gérées par le programme principal de façon qu’il y ait toujours une moitié de buffer complète prête à être transmise au DAC.

# Conversion PDM → PCM

## Les échantillons

### PCM

Le PCM (Pulse Code Modulation) est une représentation numérique d’un signal analogique résultant de sa numérisation par échantillonnage. En effet, pour construire un Signal PCM à partir d’un signal analogique, un convertisseur analogique numérique (ADC) mesure la valeur du signal analogique à une fréquence appelée fréquence d’échantillonnage. Le résultat de cette mesure est alors stocké numériquement. Un signal PCM est donc défini par deux valeurs :

* Sa fréquence d’échantillonnage
* Le nombre de bits utilisé pour stocker la mesure

Cette deuxième valeur influe sur la précision de la valeur mesurée, en effet, plus N est important plus le signal échantillonné aura de valeur possible et donc plus la mesure sera précise.

Prenons par exemple le signal suivant que l’on observe pendant 1 seconde :

Avec  Hz, et

Notre ADC peut mesurer une valeur appartement à l’intervalle Qu’il stocke dans un entier encodé sur 4 bits (0000 correspond à la valeur 0 et 1111 à la valeur 1). L’ADC effectue ses mesures à une fréquence . Cela nous donne donc les signaux suivants :

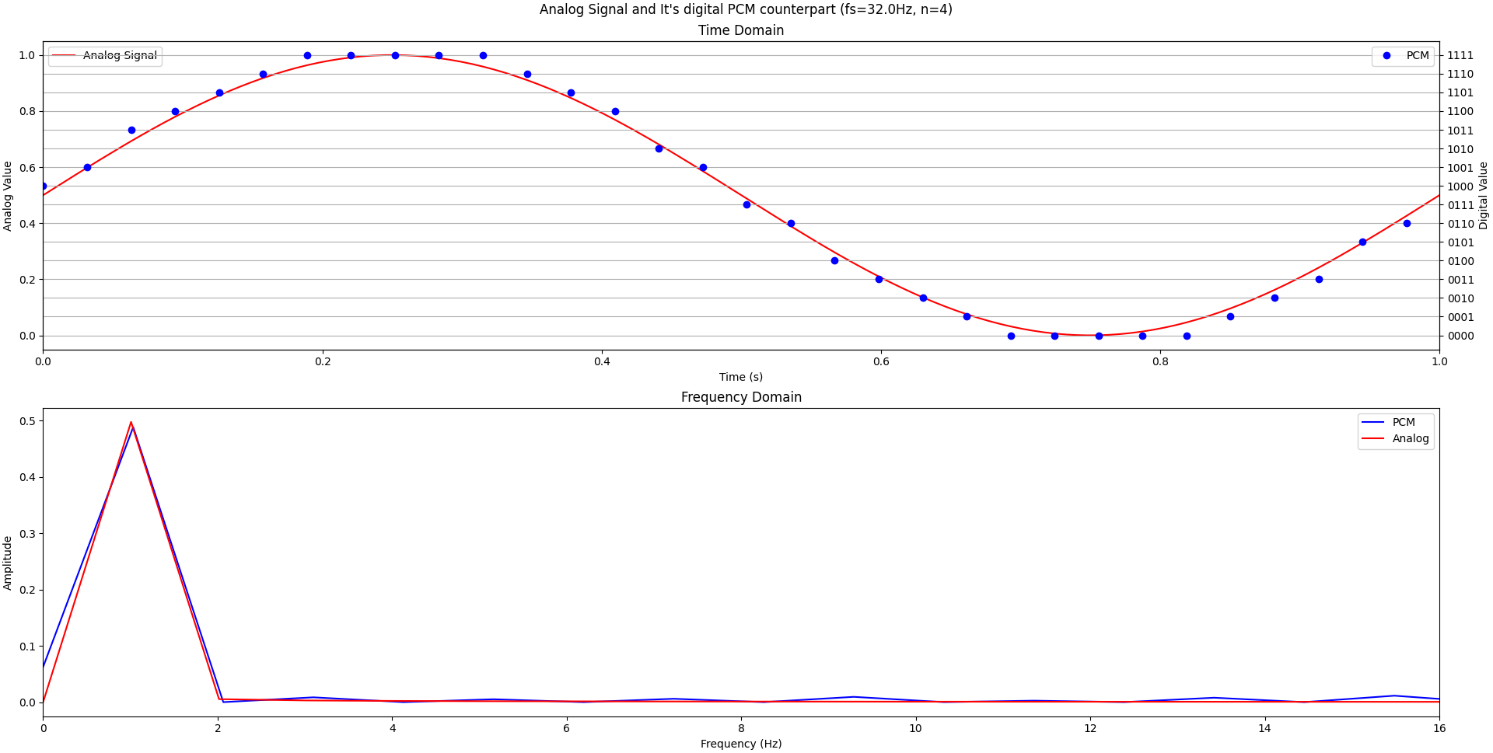


Figure  : Signal Analogique et sa conversion numérique en PCM (fs=32 Hz, n=4)

Les sons PCM sont généralement encodés sur 16 bits (65 536 valeurs possibles) et échantillonnés à pour les CD audio et pour la bande sonore d’un fichier vidéo.

### PDM

Le PDM (Pulse Density Modulation) et une représentation numérique d’un signal analogique. Cette représentation se caractérise par un flux de bits à haute fréquence. Flux dans lequel l’amplitude du signal analogique et déterminer à partir de la concentration de bit a 1 ou à 0.

Reprenons notre signal analogique que l’on observe pendant 1 seconde :

Avec  Hz, et

Le signal PDM sera alors le suivant :

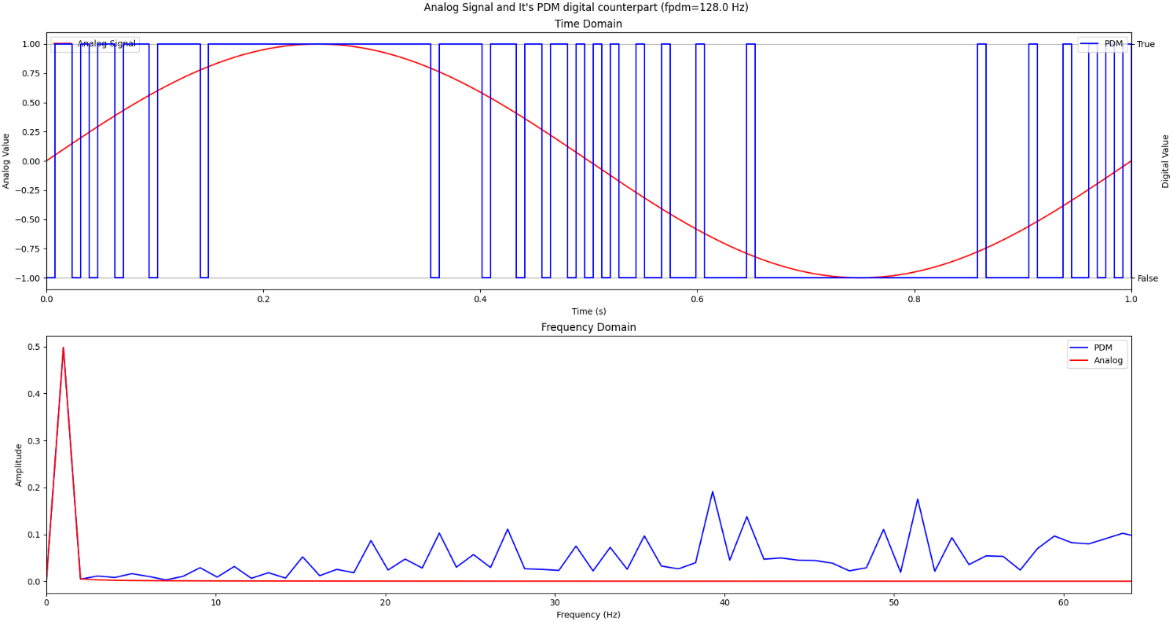


Figure 11 : Signal Analogique et sa conversion numérique en PDM ()

On distingue qu’une grande valeur positive est caractérisée par une forte concentration locale d’échantillons PDM à 1 et une grande valeur négative est caractérisé par une forte concentration locale d’échantillons PDM à 0. Un signal PDM est caractérisé par la fréquence du flux qui lui est associé. Notons que par rapport à la fréquence d’échantillonnage d’un signal PCM celle d’un signal PDM doit être bien plus élevée que (entre 48 et 128 fois plus importante) pour qualité similaire pour le signal.

## Chaine de filtrage

Le passage d’un signal PDM échantillonné à haute fréquence () vers un signal PCM échantillonné à plus basse fréquence () se fait par le filtrage du signal PDM par un filtre Passe Bas ayant une fréquence de coupure maximale , une fois le signal PDM filtré, on le sous-échantillonne afin de faire correspondre sa fréquence d’échantillonnage à celle recherché pour le signal audio (). Ce sous-échantillonnage est fait avec un facteur , généralement .

On applique ensuite au signal un offset et un gain pour l’adapter l’amplitude désirée pour le signal PCM. Ce qui nous donne la chaine de filtrage suivante :

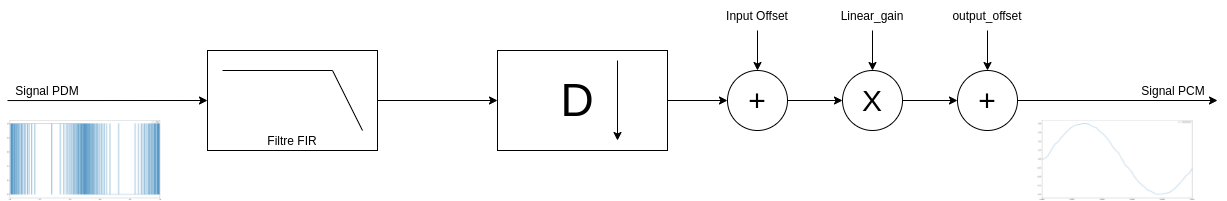


Figure  : Chaine de filtrage PDM → PCM

Pour mieux comprendre le fonctionnement du filtre, nous allons étudier un signal pendant son passage au travers de cette chaine de filtrage. Nous utiliserons un signal en dent de scie suivant l’expression suivante :

Le signal que nous utiliserons dans cet exemple a les paramètres suivants :

|  |  |
| --- | --- |
| Paramètre | Valeur |
| Fréquence PDM |  |
| Fréquence d’échantillonnage PCM |  |
| Fréquence fondamentale |  |
| Ordre de la série de Fourier |  |
| Durée du signal |  |

Ce qui donne le signal suivant (centré une période) :

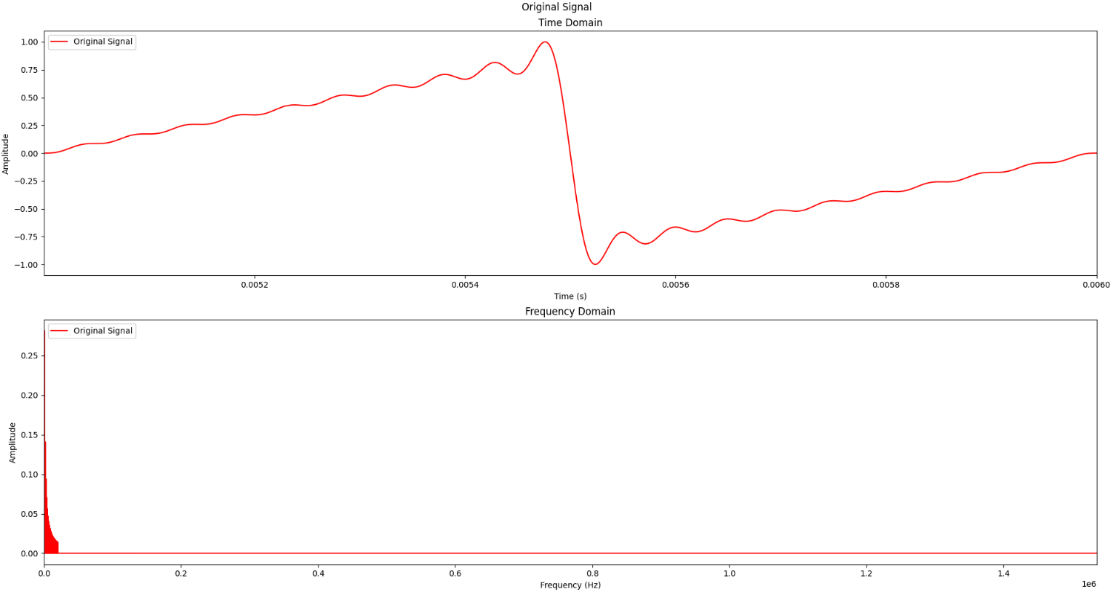


Figure  : Signal Original

Avant d’envoyer le signal dans la chaine de filtrage, on le converti en PDM ce qui nous donne le signal suivant :

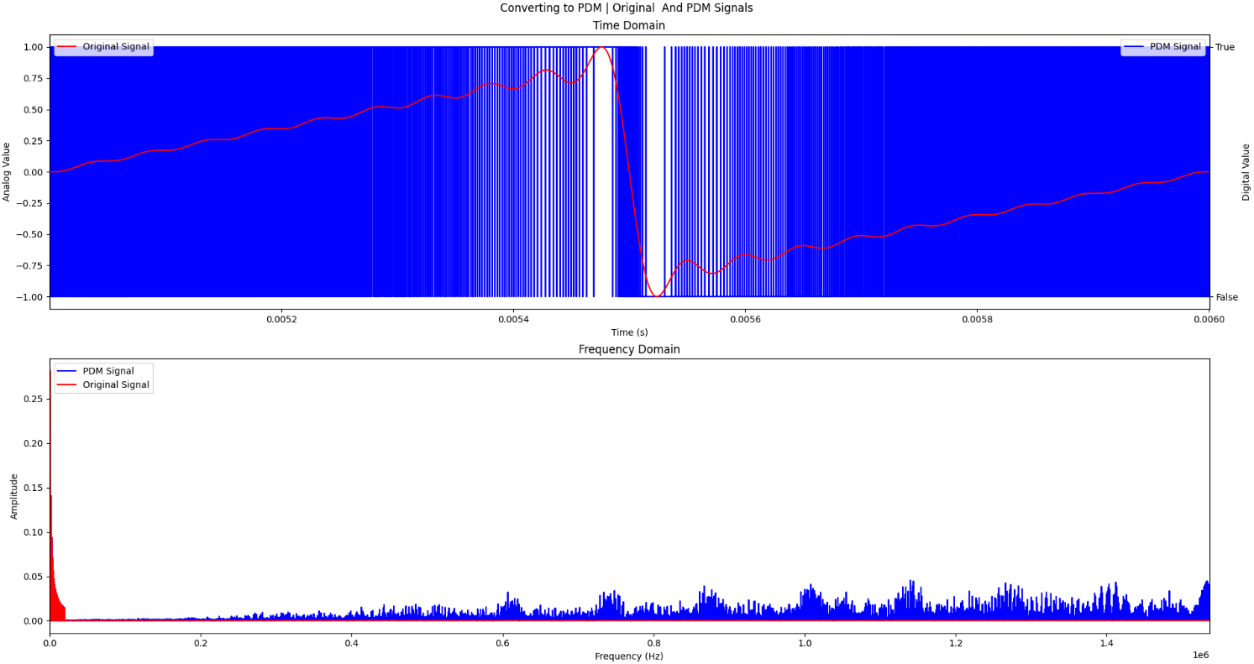


Figure  : Conversion PCM (rouge) → PDM (bleu)

La première étape de la chaine est d’appliquer un filtre passe-bas au signal, dans notre cas, c’est un filtre numérique FIR (Finite Impulse Response) qui nous permet d’éliminer les hautes fréquences caractéristiques d’un signale PDM pour ne laisser que les fréquences utiles ce qui nous donne le signal suivant :

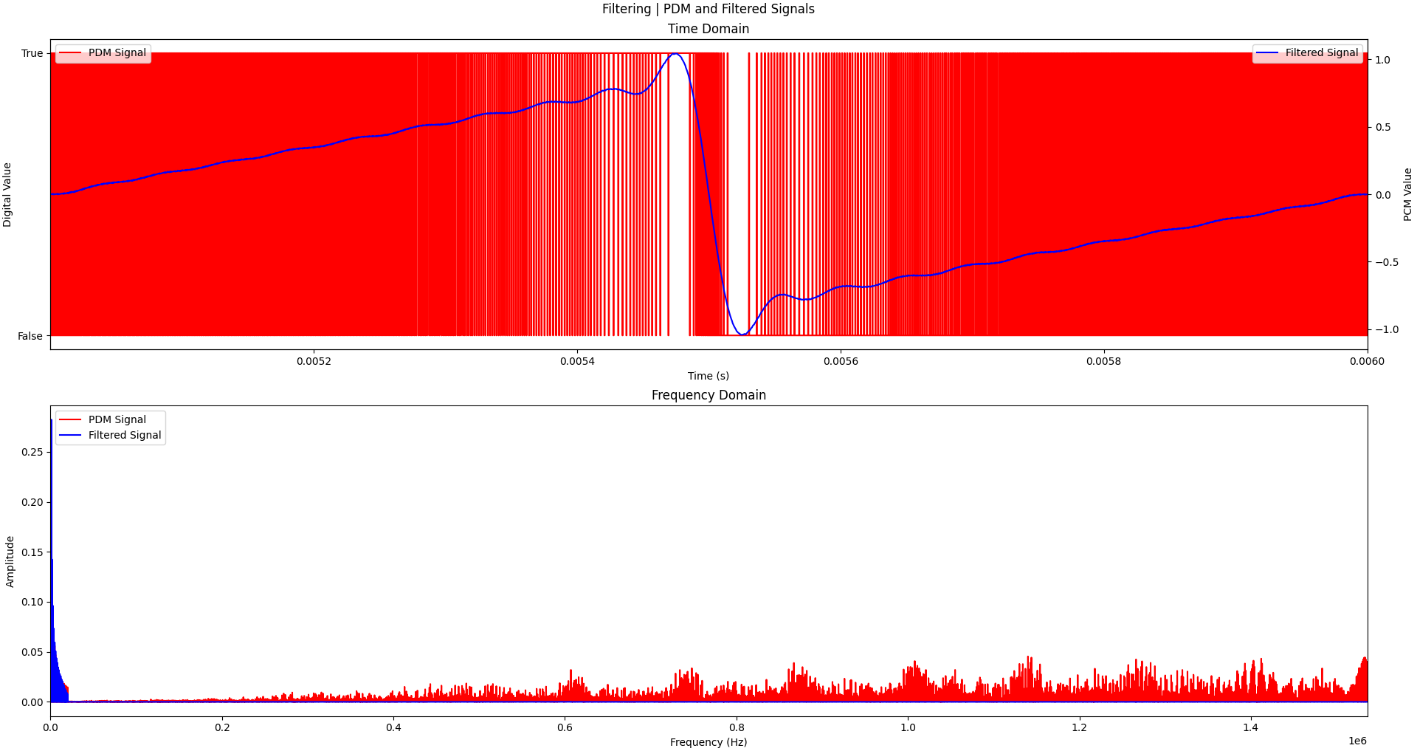


Figure 15 : Conversion PDM (Rouge) → PCM (Bleu)

Une fois le signal filtré, il a la même fréquence d’échantillonnage que le signal PDM. Or le signal PDM est échantillonné à haute fréquence pour éloigner le plus possible les bruits générés par sa nature de la bande utile, maintenant que le signal a été filtrer la bande nécessaire est très inférieur à celle du signal PDM. On va donc sous-échantillonner le signal filtré à la fréquence recherchée pour le signal PCM. Pour se faire un va sous-échantillonner le signal filtré par un facteur (on sélectionne un échantillon sur D) :

Sous-échantillonnage qui nous donne le signal suivant :

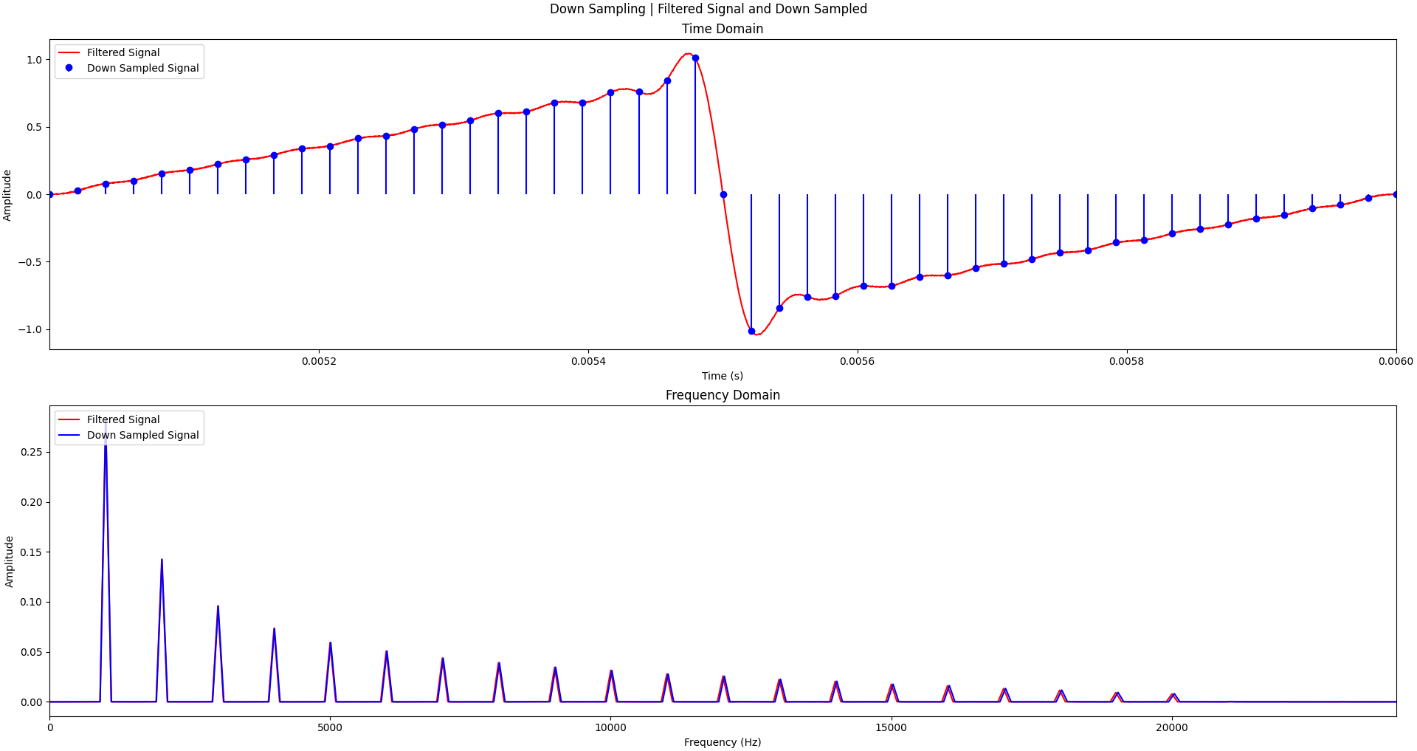


Figure 16 : Sous-échantillonnage du signal filtré

Une fois le signal sous-échantillonné on applique un gain et un offset sur le signal pour l’adapter à la sortie PCM nous obtenons alors le signal suivant[[1]](#footnote-1) en sorti de chaine :

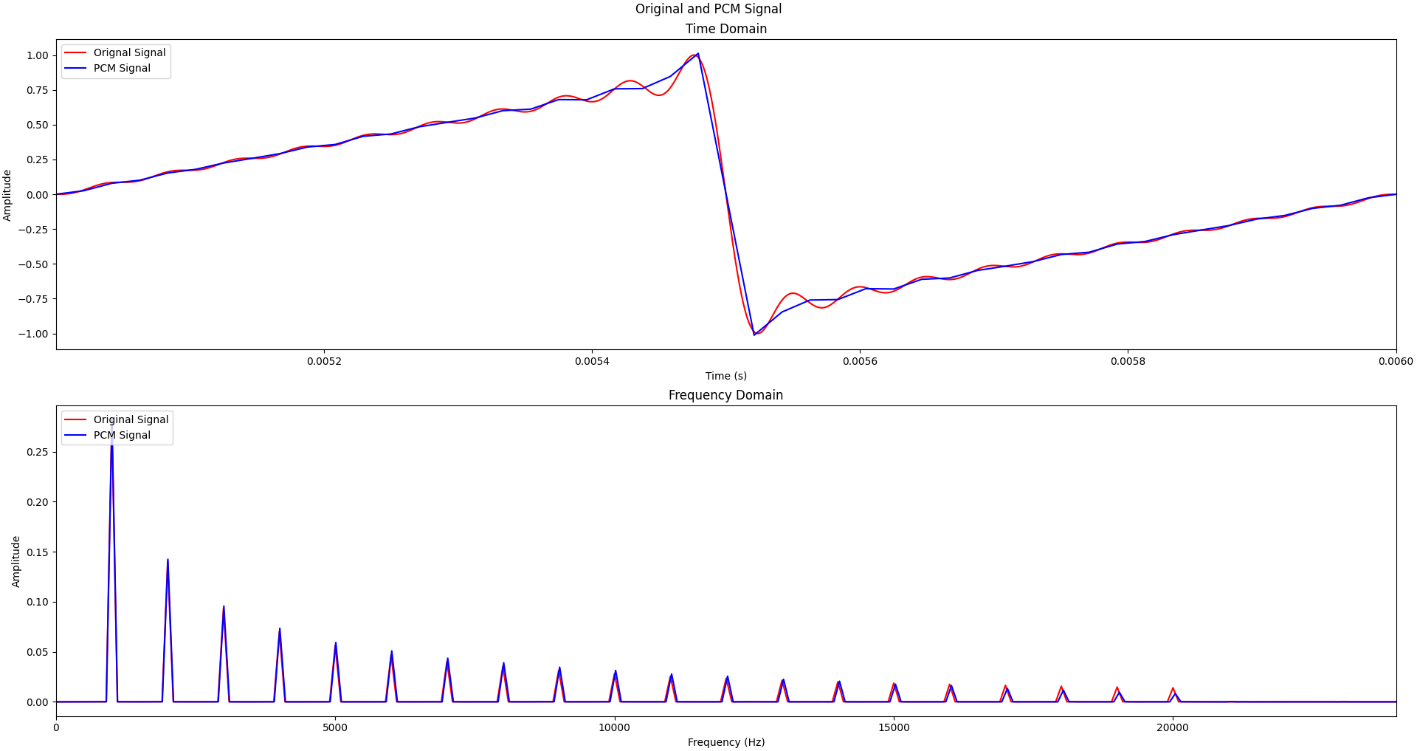


Figure 17 : Signal original (Bleu) et signal en sortie de chaine de filtrage (Rouge)

## Intégration au microcontrôleur

Les échantillons PDM arrivent dans la mémoire RAM via le DMA qui charge en continu les données en provenance du périphérique SAI dans un buffer accessible par le programme. Ce buffer est circulaire, c’est-à-dire qu’une fois la fin de celui-ci atteinte le DMA revient au début pour stocker les échantillons suivants. Afin de notifier le programme de sa position dans le buffer, le DMA lève 2 interruptions, une première quand il passe la moitié du buffer la seconde à la fin de celui-ci. Ces interruptions sont traitées par le programme dès que le lever via l’appel de fonction de "Callbacks". Ces fonctions agissent alors sur deux variables, une première qui notifie le programme principal de la présence de nouvelles données dans le buffer et une seconde qui indique dans quelle moitié du buffer lire les données. Ce qui nous donne le processus suivant qui est intégré au programme principal :

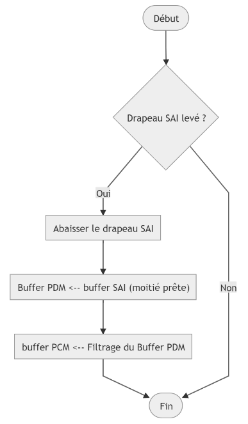


Figure  : Logigramme, Gestion des interruptions SAI

Dans ce processus, on vérifie si le drapeau SAI est levé, si oui on charge dans le buffer PDM la moitié dont le DMA vient de terminer l’écriture, puis on filtre le buffer PDM et l’on stocke le résultat dans le buffer PCM à la disposition du reste du programme.

Le bloc de filtrage utilise une librairie C que j’ai retravaillée pour l’adapter à nos besoins. Cette librairie utilise la fenêtre FIR fournie par la bibliothèque Python "Scipy" dont les coefficients sont exportés sous forme de tableau constant C, fournissant une lookup table pour la convolution de la fenêtre FIR avec le signal PDM.

Pour gagner en performance et en simplicité plutôt que travailler sur des échantillons PDM distinct, la librairie utilise des blocs de 16 échantillons PDM qui correspondent au format des données transmises par le périphérique SAI. Cette utilisation de blocs de 16 bits applique donc un premier sous-échantillonnage au signal PDM d’un facteur 16.

Pour filtrer notre signal PDM, cette librairie nous fournit deux fonctions :

* "pdm\_fir\_flt\_put" :

Cette fonction ajoute un mot PDM au buffer du filtre sur lequel on applique la fenêtre FIR. La taille de ce buffer est fixée par l’ordre du filtre FIR. Il a un fonctionnement circulaire qui permet de reproduire l’aspect glissant de la convolution d’un signal et d’une fenêtre.

* "pdm\_fir\_ftl\_get" :

Cette fonction calcule la convolution entre le buffer du filtre est la fenêtre FIR, et donc qui produit un échantillon PCM

La librairie telle que fournie nous permet donc d’implémenter dans le bloc de filtrage et une partie du bloc de sous-échantillonnage de la chaine de filtrage PDM (cf. Figure 12). Pour compléter la chaine j’ai don ajouter à la libraire une structure de donnée contenant la configuration de la chaine de filtrage (facteur de sous-échantillonnage, facteur d’amplification, offset…) ainsi que la fonction :

* "pdm\_fir\_flt\_chunk" :

Cette fonction produit le signal PCM correspondant au buffer PDM qu’on lui fournit en lui appliquant les différentes étapes de la chaine de filtrage, c’est cette fonction que l’on appelle en Figure 18 dans le bloc "buffer PCM <-- Filtrage du Buffer PDM" pour filtrer le signal PDM.

Si l’on résume le DMA rempli automatiquement le buffer SAI à partir des données transmises par le périphérique SAI, dès que le DMA termine l’écriture d’une moitié du buffer SAI un drapeau est levé pour que le programme principal entame le filtrage de cette moitié pour un usage ultérieur des échantillons PCM par le programme.

Pour estimer les performances de la chaine de filtrage, prenons le démonstrateur "Direct Output". À partir de ce démonstrateur, on va mesurer à l’oscilloscope deux signaux :

* Un signal à 3V quand le programme transfère les données dans le buffer du DAC (Channel 1)
* Un signal à 3V quand le programme passe le signal PDM dans la chaine de filtrage (Channel 2)

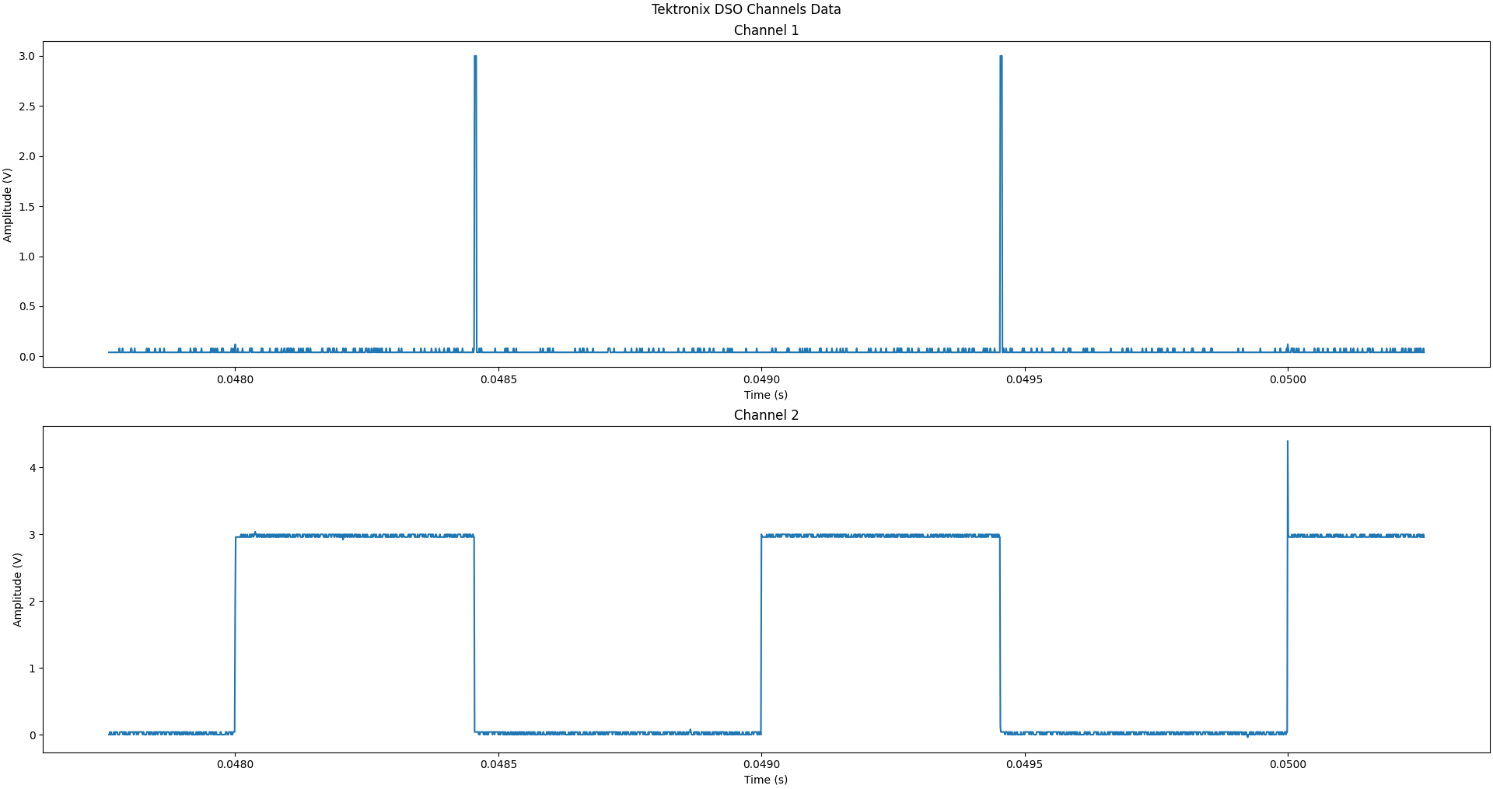


Figure 19 : Signaux de copie sur le DAC et de filtrage pour le démonstrateur "Direct Output"

Le démonstrateur est configuré pour que les drapeaux propres au DMA (SAI et DAC) se lèvent toutes les 1 ms, ce que l’on peut voir sur le "Channel 1" avec la fréquence à laquelle le programme copie les données PCM sur le DAC. Sur le "Channel 2" on peut mesurer un temps à 3V de soit environ 45 % du temps, laissant 55 % de temps pour d’autres traitements (écriture sur une clé USB…).

# Les démonstrateurs

## Présentation des démonstrateurs :

Afin de maitriser les différents aspects de la chaine de capture et de la carte cible, j’ai développé trois démonstrateurs, permettant de mettre en lumière différentes applications que l’on peut avoir pour ce type de microphones.

### « Parrot »

Le démonstrateur Parrot est un programme qui enregistre un son puis le rejoue le son sur un haut-parleur. Pour remplir ses fonctions, le programme s’appuie sur la machine de 4 états suivante :

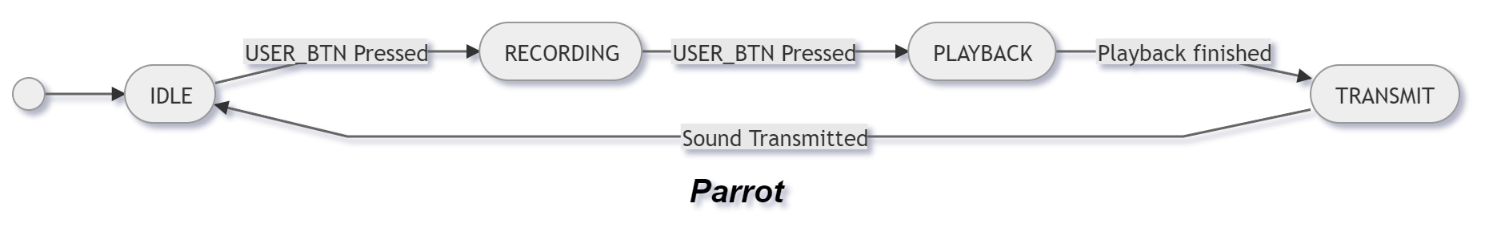


Figure 20 : machine d’état du démonstrateur "Parrot"

Le programme passe donc dans les états suivants :

* "IDLE" :

Le programme est en attente d’un appui sur le bouton "USER" (bouton bleu) pour passer dans l’état "RECORDING"

* "RECORDING" :

Le programme enregistre les sons ambiants et conserve jusqu’à 3 secondes en mémoire RAM. Il passe dans l’état "PLAYBACK" lors d’un appui sur le bouton "USER"

* "PLAYBACK" :

Le programme rejoue le son enregistré en mémoire sur les 2 canaux du DAC. Il passe dans l’état "TRANSMIT" une fois que la totalité du son enregistré a été rejouée.

* "TRANSMIT" :

Le programme transmet le son enregistré sous forme de données WAV sur la ligne série du l’UART1, une fois l’ensemble des données transmises, le programme passe à l’état "IDLE"

Le logigramme de chacun des états et disponible en Annexe 1 : Logigrammes des États démonstrateur "Parrot"

L’état "TRANSMIT" peut être désactivé en commentant la définition de la macro C "#define TRANSMIT" (Core/Src/main. c l.80). Quand cet état est désactivé, le programme passe directement de l’état "PLAYBACK" à l’état "IDLE" sans transmettre les données sur la ligne série.

Pour ce démonstrateur j’utilise le schéma bloc des périphériques suivants :

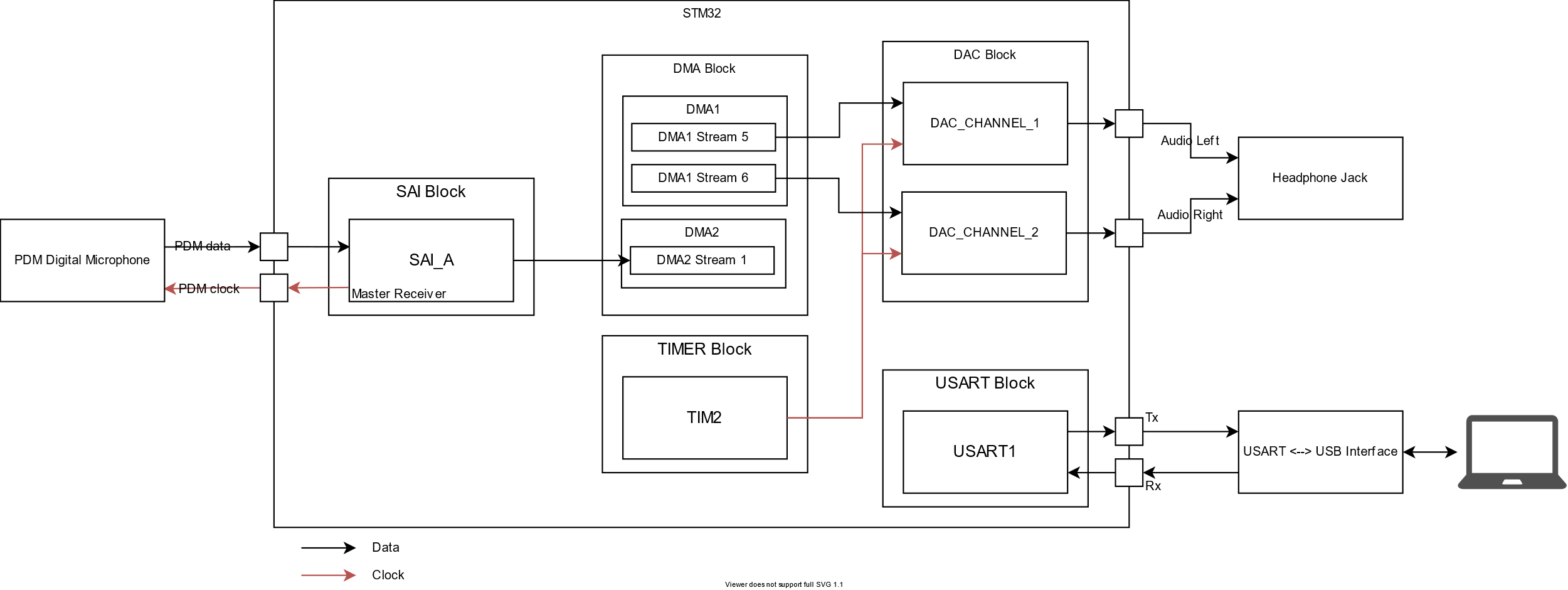


Figure 21 : Schéma Bloc des périphériques utilisés par le démonstrateur "Parrot"

Ce démonstrateur produit un signal PCM échantillonné sur 12 bits à 32 kHz. Pour ce démonstrateur, j’ai choisi un facteur de sous-échantillonnage de 64 donc . Pour le filtre j’ai généré la LUT pour une fenêtre FIR d’ordre 16 avec une fréquence de coupure

Dans ce démonstrateur, j’ai fait le choix d’échantillonner le signal PCM à 32 kHz afin de pouvoir stocker 3 secondes de son () en saturant au maximum la mémoire RAM () du microcontrôleur, si l’on utilise une fréquence d’échantillonnage PCM de 48 kHz le temps maximum d’enregistrement descend à 2 secondes () ce qui peut paraitre court quand on utilise le démonstrateur.

Lors de la transmission des échantillons PCM, le programme amplifie le signal pour utiliser l’ensemble des 16 bits par échantillons mis à disposition par le fichier WAV

### « Digital recorder »

Le démonstrateur "Digital recorder" est un programme qui reproduit les fonctions enregistreuses audio, en sauvegardant le son capté par le microphone sous forme de fichiers WAV sur la clé USB connectée sur l’interface USB\_HS. Pour remplir ses fonctions, le programme s’appuie sur la machine de 3 états suivante :

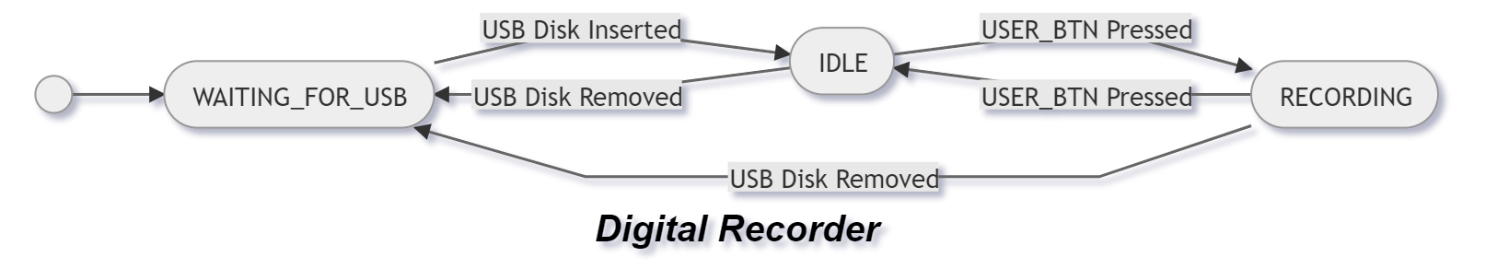


Figure : Machine d’état du démonstrateur "Digital Recorder"

Le programme passe donc dans les états suivants :

* "WAITING\_FOR\_USB" :

Le programme est en attente de l’insertion d’une clé USB qui le fera passer dans l’état "IDLE". Si la clé USB est retirée, le programme reviendra directement dans cet état, peu importe son état courant.

* "IDLE" :

Le programme est en attente d’un appui sur le bouton "USER" (bouton bleu) pour passer dans l’état "RECORDING"

* "RECORDING" :

Le programme enregistre les sons ambiants dans un fichier WAV sur la clé USB, lors de l’appui sur le bouton "USER" le programme finalise l’enregistrement et passe dans l’état "IDLE"

Le logigramme de chacun des états et disponible en Annexe 2 : Logigrammes des États démonstrateur "Digital Recorder"

Pour ce démonstrateur j’utilise le schéma bloc des périphériques suivants :

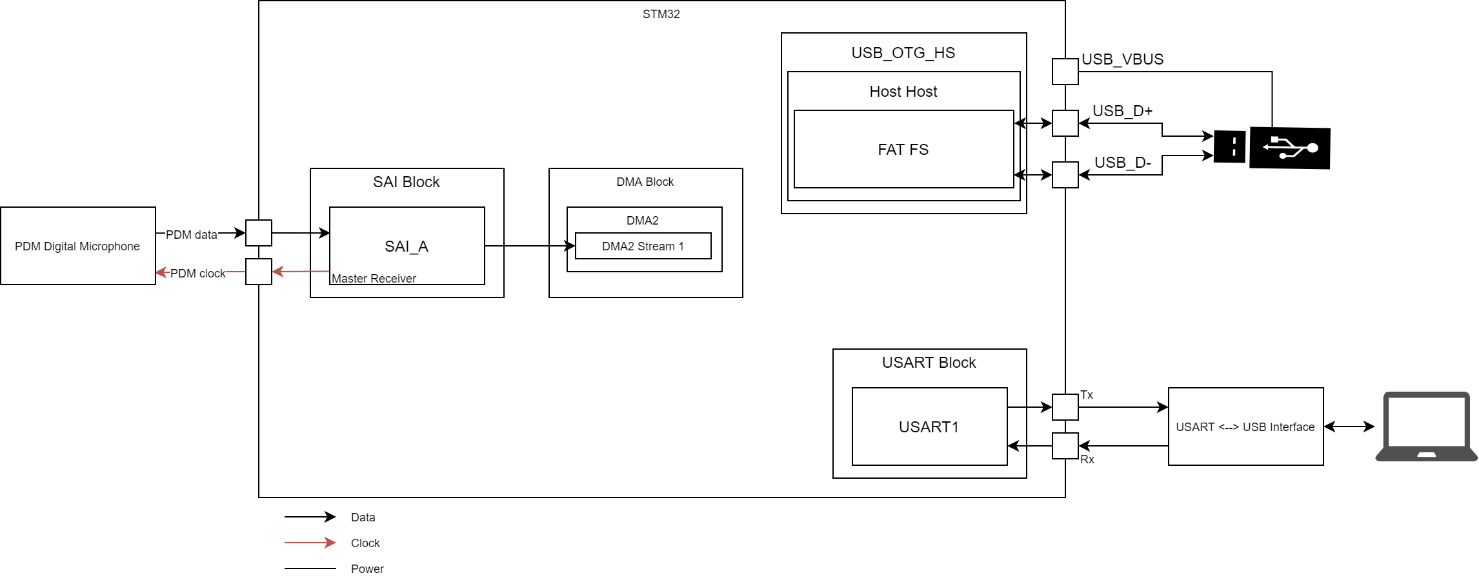


Figure 23 : Schéma Bloc des périphériques utilisés par le démonstrateur "Digital Recorder"

Ce démonstrateur produit un signal PCM échantillonné sur 12 bits à 48 kHz. Pour ce démonstrateur, j’ai choisi un facteur de sous-échantillonnage de 64 donc . Pour le filtre j’ai généré la LUT pour une fenêtre FIR d’ordre 16 avec une fréquence de coupure

### « Direct output »

Le démonstrateur "Direct Output" est un programme qui enregistre un son puis le rejoue le son sur un haut-parleur. Pour remplir ses fonctions, le programme s’appuie sur la machine de 2 états suivante :

Une image contenant texte, périphérique, sombre, mètre

Description générée automatiquement

* "IDLE" :

Le programme est en attente d’un appui sur le bouton "USER" (bouton bleu) pour passer dans l’état "RECORDING"

* "RECORDING" :

Le programme enregistre les sons ambiants et les rejoue en temps réel sur le DAC. Il retourne dans l’état "IDLE" lors d’un appui sur le bouton "USER".

Le logigramme de chacun des états et disponible en Annexe 3 : Logigrammes des États démonstrateur "Direct Output"

Ce démonstrateur quand il est dans l’état "RECORDING" permet si l’on branche un casque relativement bien isolé des sons ambiants d’offrir une démonstration relativement impressionnante des capacités de notre chaine de capture.

Pour ce démonstrateur j’utilise le schéma bloc des périphériques suivants :

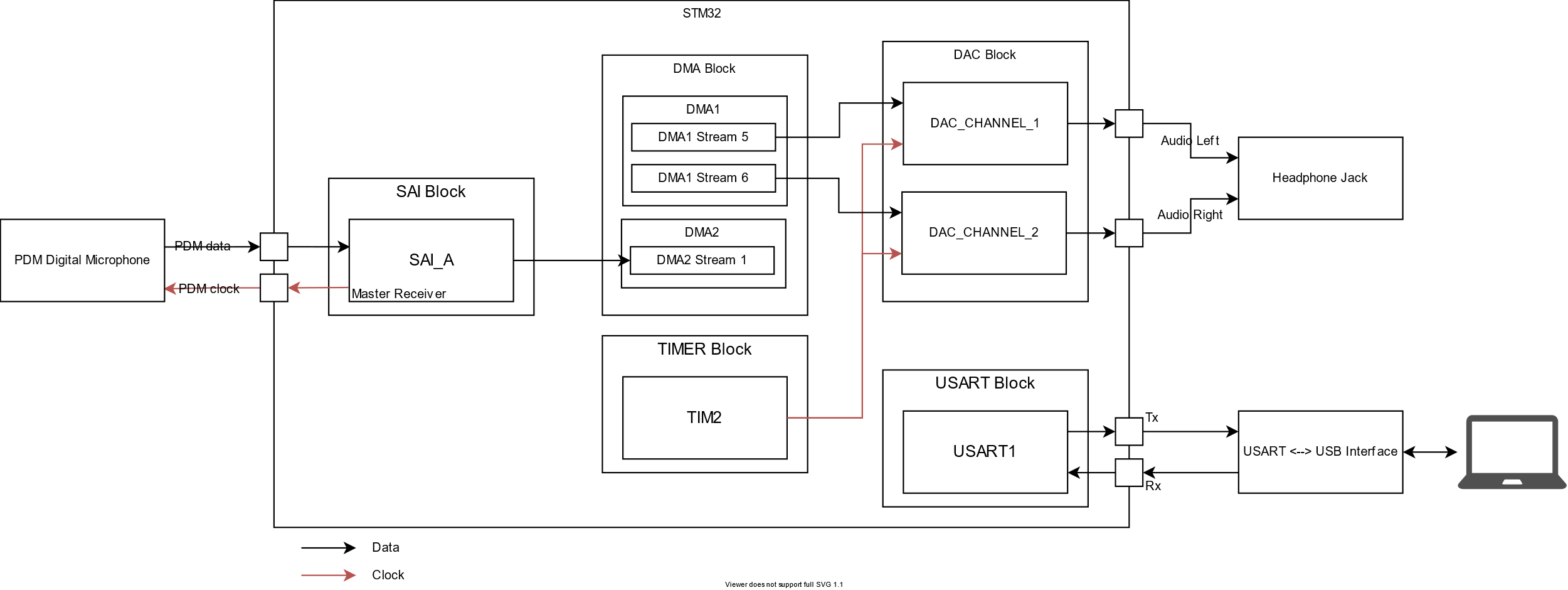


Figure : Schéma Bloc des périphériques utilisés par le démonstrateur "Direct Output"

Ce démonstrateur produit un signal PCM échantillonné sur 12 bits à 48 kHz. Pour ce démonstrateur, j’ai choisi un facteur de sous-échantillonnage de 64 donc . Pour le filtre j’ai généré la LUT pour une fenêtre FIR d’ordre 16 avec une fréquence de coupure

# Bibliographies

olegv142. (2017). *PDM bitstream FIR filter*. Retrieved from Github.com: https://github.com/olegv142/pdm\_fir

ST Microelectronics. (2018, January). STM32F427xx STM32F429xx. *Datasheet | STM32F427xx STM32F429xx*.

ST Microelectronics. (2019, July). Interfacing PDM digital microphones using STM32 MCUs and MPUs. *AN5027 | Interfacing PDM digital microphones using STM32 MCUs and MPUs*.

ST Microelectronics. (2020, August). Discovery kit with STM32F429ZI MCU. *UM1670 | Discovery kit with STM32F429ZI MCU*.

# Glossaires des termes techniques

|  |  |
| --- | --- |
| **Terme ou acronyme** | **Définition** |
| ADC | "Analog to Digital Converter". Convertisseur analogique vers Numérique. |
| Buffer | Espace mémoire tampon |
| Buffer Circulaire | Un buffer circulaire est un buffer de taille fixe auquel on a rejoint le début et la fin de façon qu’il puisse recevoir des valeurs de façon infinies, les nouvelles valeurs remplaçant les anciennes au fur et à mesure que l’on en ajoute |
| DAC | "Digital to Analog Converter". Convertisseur numérique vers Analogique. |
| DMA | "Direct Memory Access". Composant du microcontrôleur permettant l’échange de données entre la RAM et les périphériques sans impacter l’exécution du programme principal |
| LookUp Table (LUT) | « Table de Correspondance » Structure de données qui contient des données précalculer pour afin de réduit le temps nécessaire au programme pour effectuer une opération complexe en la remplaçant par une consultation de la table. Cependant, une LUT peut prendre beaucoup d’espace en mémoire. |
| MEMS | "MicroElectroMechanical Systems", un MEMS est un système dont la taille est de l’ordre de quelques micromètres, cette taille très réduite permet d’associer les propriétés électriques d’un semi-conducteur à celle d’un capteur mécanique en ayant un encombrement minimum |
| PCM | "Pulse Code Modulation". Modulation d’un signal numérique où chaque échantillon stocke le niveau du signal à un instant T |
| PDM | "Pulse Density Modulation". Modulation d’un signal numérique où le niveau du signal et définie par la densité d’échantillons à "1" |
| Quantum | Le Quantum d’un ADC/DAC correspond à la plus petite variation de tension qu’il peut reproduire/distinguer, il se calcule grâce à la formule suivante : où correspond à la plage de tension de travail du convertisseur en Volt, et correspond au nombre de bits du convertisseur |
| SAI | "Serial Audio Interface". Interface numérique de transfert de signaux audio en série |
| USB OTG | "Universal Serial Bus" Extension du protocole USB qui permet l’échange de données entre deux périphériques USB sans avoir à passer par un ordinateur Hôte (ex. Téléphone → Clé USB) |
|  |  |
|  |  |

# Table des figures

[Figure 1 : Comportement du démonstrateur obligatoire 6](#_Toc88309083)

[Figure 2 : Carte Cible (STM32F429I-DISC1) 7](#_Toc88309084)

[Figure 3 : Schéma bloc typique d’un microphone PDM 7](#_Toc88309085)

[Figure 4 : Connexion du Microphone PDM en configuration mono 8](#_Toc88309086)

[Figure 5 : Chaine de capture 8](#_Toc88309087)

[Figure 6 : Montage direct d’un haut-parleur sur le DAC 9](#_Toc88309088)

[Figure 7 : Branchement d’un haut-parleur en série avec une résistance de 1 kΩ sur le DAC 10](#_Toc88309089)

[Figure 8 : Schéma Électrique du câble DAC → Jack 3,5 mm 10](#_Toc88309090)

[Figure 9 : Câble DAC → Jack 3.5mm 10](#_Toc88309091)

[Figure 10 : Signal Analogique et sa conversion numérique en PCM (fs=32 Hz, n=4) 12](#_Toc88309092)

[Figure 11 : Signal Analogique et sa conversion numérique en PDM () 13](#_Toc88309093)

[Figure 12 : Chaine de filtrage PDM → PCM 13](#_Toc88309094)

[Figure 13 : Signal Original 14](#_Toc88309095)

[Figure 14 : Conversion PCM (rouge) → PDM (bleu) 14](#_Toc88309096)

[Figure 15 : Conversion PDM (Rouge) → PCM (Bleu) 15](#_Toc88309097)

[Figure 16 : Sous-échantillonnage du signal filtré 15](#_Toc88309098)

[Figure 17 : Signal original (Bleu) et signal en sortie de chaine de filtrage (Rouge) 16](#_Toc88309099)

[Figure 18 : Logigramme, Gestion des interruptions SAI 16](#_Toc88309100)

[Figure 19 : Signaux de copie sur le DAC et de filtrage pour le démonstrateur "Direct Output" 18](#_Toc88309101)

[Figure 20 : machine d’état du démonstrateur "Parrot" 19](#_Toc88309102)

[Figure 21 : Schéma Bloc des périphériques utilisés par le démonstrateur "Parrot" 20](#_Toc88309103)

[Figure 22: Machine d’état du démonstrateur "Digital Recorder" 20](#_Toc88309104)

[Figure 23 : Schéma Bloc des périphériques utilisés par le démonstrateur "Digital Recorder" 21](#_Toc88309105)

[Figure 24 : Schéma Bloc des périphériques utilisés par le démonstrateur "Direct Output" 22](#_Toc88309106)

# Annexes

## Annexe 1 : Logigrammes des États démonstrateur "Parrot"

Annexe disponible dans le fichier « [Annexe\_1\_logigrammes\_parrot.html](https://piwithy.fr/files/Public/ensta/pfe_stm32/report/Annexe_1_logigrammes_parrot.html)»

## Annexe 2 : Logigrammes des États démonstrateur "Digital Recorder"

Annexe disponible dans le fichier « [Annexe\_2\_logigrammes\_digital\_recorder.html](https://piwithy.fr/files/Public/ensta/pfe_stm32/report/Annexe_2_logigramme_digital_recorder.html)»

## Annexe 3 : Logigrammes des États démonstrateur "Direct Output"

Annexe disponible dans le fichier « [Annexe\_3\_logigrammes\_direct\_output.html](https://piwithy.fr/files/Public/ensta/pfe_stm32/report/Annexe_3_logigramme_direct_output.html)»

1. Il a été appliqué sur ce signal un gain linéaire de 1 et un offset de 0 (signal identique au signal sous-échantillonné) [↑](#footnote-ref-1)