


Traitement de signal audio embarqué temps réel sur carte STM32

Formation d'Ingénieur Par Apprentissage 2021 | Systèmes Embarqués

 ENSTA BRETAGNE	Pierre-Yves JÉZÉGOU, pierre-yves.jezegou@ensta-bretagne.org	Irvin PROBST, irvin.probst@ensta-bretagne.fr Olivier REYNET, olivier.reynet@ensta-bretagne.fr
---	---	--

Remerciements

Je tiens à remercier toutes les personnes liées, de près ou de loin, au bon déroulement de mon apprentissage ainsi que toutes celles qui m'ont accompagné pendant ce projet de fin d'études.

Je tiens particulièrement à remercier messieurs Olivier Reynet et Irvin Probst de m'avoir encadré et conseillé pendant ce projet.

Résumé

L'objectif de ce rapport est de fournir une base de travail pour la création d'un projet étudiant autour de la capture audio embarquée en temps réel. Afin de préparer ce projet, le choix de la plateforme embarquée s'est porté sur la famille de microcontrôleurs STM32 de la société ST Microelectronics. La capture se fait via un microphone numérique de type MEMS utilisant la technologie PDM pour la restitution des échantillons.

Mon travail porte sur l'étude du fonctionnement d'une chaîne d'acquisition-restitution audio utilisant un microcontrôleur STM32 et un microphone PDM.

On retrouvera donc, dans ce rapport, l'étude des entrées et des sorties de la chaîne d'acquisition-restitution, le détail des traitements appliqués sur le signal en provenance du microphone afin de les rendre compatibles de l'étage de sortie, et finalement la description des démonstrateurs que j'ai développés pour vérifier le bon fonctionnement de la chaîne dans différentes applications

Abstract

The objective of this report is to provide a work base for the creation of a student project about real-time embedded audio capture. Before the start of my work, the choice of the embedded platform was to use the microcontroller family STM32 from ST Microelectronics. The audio capture is done using a Digital MEMS microphone using PDM technology to send the sample to the microcontroller.

My work concerns the study of a capture and playback chain using an STM32 microcontroller and a PDM microphone.

In this report we will find a study of the input and the output of the capture playback chain, the details of the audio processing mandatory to adapt the microphone input to the output stage, and finally the description of the proof-of-concept programs I developed to make sure the capture playback chain works in different applications.

Sommaire

Remerciements	2
Résumé.....	2
Abstract	2
Sommaire	3
1. Objectifs du projet.....	5
2. Plateforme Embarquée	6
2.1. Carte de développement.....	6
2.2. La capture et la restitution d'un son	6
2.2.1. Le microphone.....	6
2.2.2. La sortie audio	8
3. Le traitement des échantillons.....	11
3.1. Les échantillons.	11
3.1.1. Le format de sortie : PCM.....	11
3.1.2. Le format d'entrée : PDM.....	12
3.2. Chaîne de filtrage.	12
3.3. Intégration au microcontrôleur.....	16
3.3.1. Fonctionnement.	16
3.3.2. Performances de la chaîne de filtrage.....	19
4. Les démonstrateurs.....	22
4.1. "Parrot"	22
4.2. "Digital recorder"	23
4.3. "Direct output"	24
Conclusion	26
Bibliographies.....	27
Glossaires des termes techniques	27
Table des figures.....	28
Annexes	29
Annexe 1 : Logigrammes des États du démonstrateur "Parrot"	29
Machine d'état	29
IDLE.....	29
RECORDING	30
PLAYBACK	31
TRANSMIT.....	32

Annexe 2 : Logigrammes des États du démonstrateur "Digital Recorder"	33
Machine d'état	33
WAITING_FOR_USB.....	33
IDLE.....	33
RECORDING	35
Annexe 3 : Logigrammes des États du démonstrateur "Direct Output"	36
Machine d'état	36
IDLE.....	36
RECORDING	37

1. 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'intéresse donc à la cible STM32 qui est aujourd'hui un standard de l'industrie embarquée, ainsi qu'aux microphones numériques PDM qui sont une innovation intéressante par leurs faibles bruits.

Le projet se concentre donc sur la production d'une configuration clé en main de la carte STM32. Cette configuration qui permet :

- d'acquérir un signal numérique audio en utilisant le DMA du processeur et en effectuant le filtrage des échantillons PDM en provenance du microphone,
- de générer un son préalablement acquis par le microphone PDM sur le DAC du processeur.

Pendant le projet, on s'intéresse plus particulièrement aux réglages du microphone afin de maîtriser :

- l'échantillonnage des signaux PDM et PCM ainsi que les fréquences associées,
- le filtrage PDM → PCM afin de s'affranchir de la librairie « boîte 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 chaîne d'acquisition.

Les productions techniques du projet se matérialisent par la production de démonstrateurs mettant en œuvre la chaîne d'acquisition. Parmi ces démonstrateurs, au moins un a le comportement suivant :

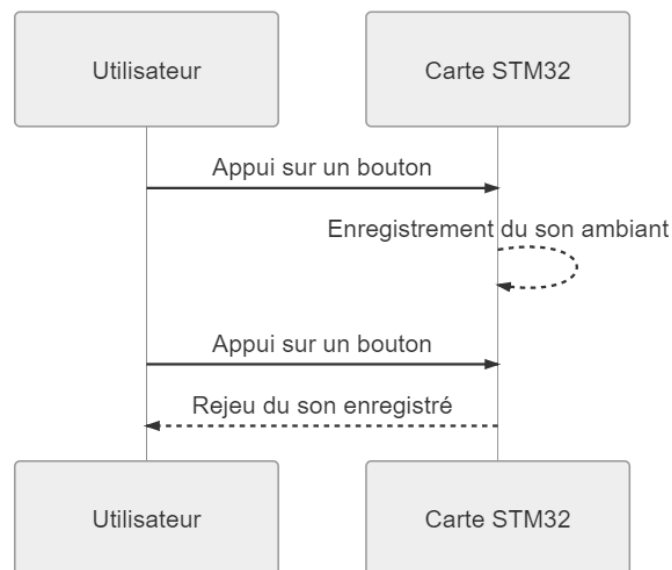


Figure 1 : Comportement du démonstrateur obligatoire

Dans la suite de ce rapport, nous étudierons dans un premier temps la plateforme embarquée qui sert de support au projet ainsi que le microphone et l'étage de restitution du son. Dans un second temps, nous étudierons la chaîne de filtrage nécessaire à la conversion des échantillons en provenance du microphone en échantillons utilisables, ainsi que l'intégration de cette chaîne sur le microcontrôleur. Finalement, nous décrirons le fonctionnement des démonstrateurs que j'ai développés afin de démontrer les capacités de la plateforme STM32 dans différentes applications centrées sur l'acquisition de signaux audio.

2. Plateforme Embarquée

2.1. 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 de la série STM32F429xx. Ce microcontrôleur fait partie de la famille 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 STM32 qui est un standard de l'industrie embarqué. Cette carte intègre¹ en plus du microcontrôleur :

- une interface de programmation ST-LINKV2,
- un écran LCD 2.4",
- plusieurs LED de débogage,
- deux boutons poussoirs,
- un port micro-USB type B compatible OTG,
- une puce de SDRAM de 64 Mbit,
- un gyroscope MEMS.



Figure 2 : Carte Cible (STM32F429I-DISC1)

2.2. La capture et la restitution d'un son

2.2.1. 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. Le transducteur MEMS permet de reproduire fidèlement un son en fournissant un encombrement minimum, de plus, la numérisation du signal au plus près du transducteur permet une forte isolation des aux bruits électromagnétiques ambiants. Un microphone PDM est généralement structuré comme suit :

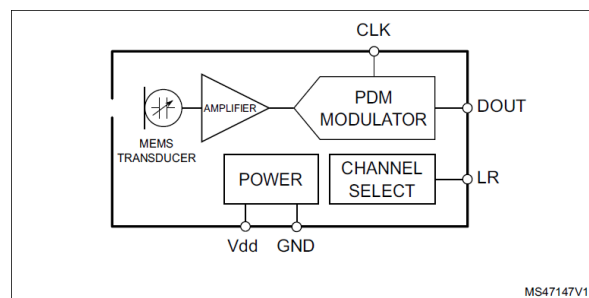


Figure 3 : Schéma bloc typique d'un microphone PDM

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

- MEMS Transducer :
Le transducteur MEMS est un condensateur dont la capacité varie en fonction des variations de pression locale de l'air, qui est caractéristique au déplacement d'une onde sonore.
- Amplifier :
L'amplificateur amplifie le signal en provenance du transducteur afin de fournir un signal adapté au modulateur PDM.
- PDM Modulator :
Le modulateur PDM convertit à chaque front de l'horloge (CLK) le signal analogique en provenance de l'amplificateur en un échantillon PDM (DOUT)
- Channel Select
Le sélecteur de canal agit sur le modulateur PDM en déterminant s'il convertit le signal sur un front montant ou descendant de l'horloge. Il définit donc si le microphone travaille sur le canal gauche ou le canal droit, quand on utilise deux microphones PDM sur un unique flux, afin de reproduire un son en stéréo
- Power :
Le bloc Power correspond à l'alimentation du microphone PDM

Pour le projet, nous travaillons dans une configuration mono pour le microphone PDM ce qui nous donne les connexions électriques suivantes :

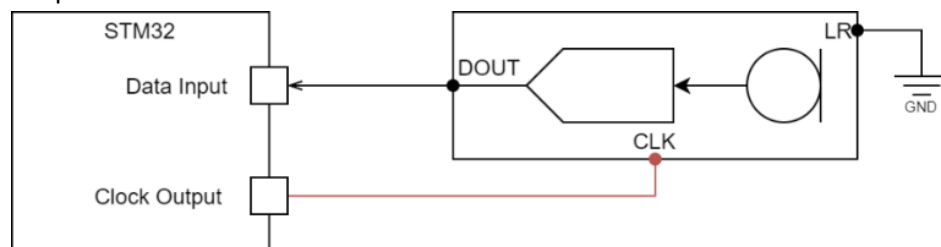


Figure 4 : Connexion du Microphone PDM en configuration mono

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). Ce périphérique génère les signaux nécessaires pour le contrôle des microphones PDM, ainsi que l'acquisition des échantillons PDM. On couple ce périphérique au DMA du microcontrôleur qui permet d'acquérir les échantillons en tâche de fond. Ce qui permet donc de filtrer le signal PDM uniquement quand le DMA nous indique que suffisamment d'échantillons sont prêts. Laissant des temps d'inactivité au microcontrôleur pour exécuter d'autres tâches. La chaîne de complète capture est représentée en Figure 5.

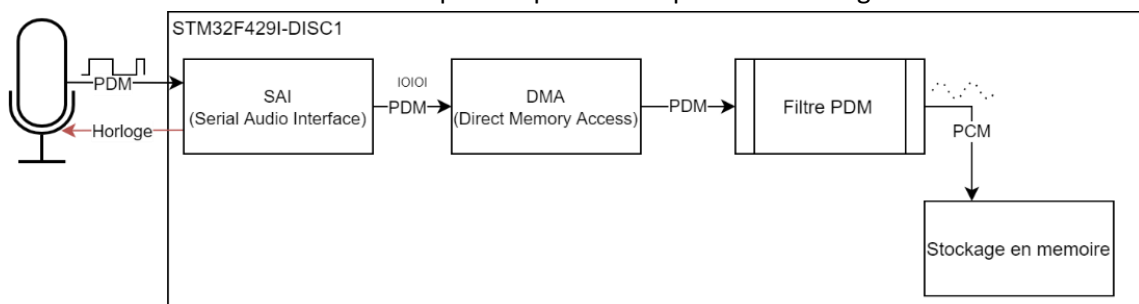


Figure 5 : Chaîne de capture

2.2.2. La sortie audio

Un deuxième objectif du projet est 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 utilise le DAC intégré au microcontrôleur afin de recréer le signal analogique correspondant aux sons captés. Le DAC intégré à notre microcontrôleur dispose de deux canaux pouvant reproduire des tensions à partir de valeurs stockées sur 12 bits. Dans la configuration utilisée par notre carte il peut générer une tension comprise en $0.2V$ et $\rightarrow VDD_{analog} - 0.2V$ avec dans notre cas $VDD_{analog} = 3.2V$ ce qui nous donne une amplitude maximale :

$$E = VDAC_{max} - VDAC_{min} = 3.0 - 0.2 = 2.8V$$

Et donc un quantum :

$$q = \frac{E}{2^n} = \frac{2.8}{2^{12}} = 683.6 * 10^{-6}V$$

Ces deux valeurs du DAC nous indiquent que pour adapter le signal PCM filtrer il faut produire des échantillons PCM sur 12 bits. Comme la sortie du DAC est unipolaire (0-3V), il faut ajouter un offset de $2047 \left(\frac{2^{12}}{2} - 1 \right)$ 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 est comprise entre 20 Hz et 20 kHz gomme cet offset nécessaire à la reproduction des valeurs négatives du signal audio.

Pour générer un son, on utilise un haut-parleur dont l'impédance est très faible ($Z \in [4; 50] \Omega$), c'est pourquoi il faut s'assurer que le courant fourni par le DAC ne peut pas excéder le courant maximum qui peut être fourni par le microcontrôleur² :

$$I_{IO} = 25mA \text{ et } \Sigma I_{IO} = 120mA$$

Une solution simple serait de connecter le haut-parleur entre la sortie du DAC et la masse comme sur le montage suivant :

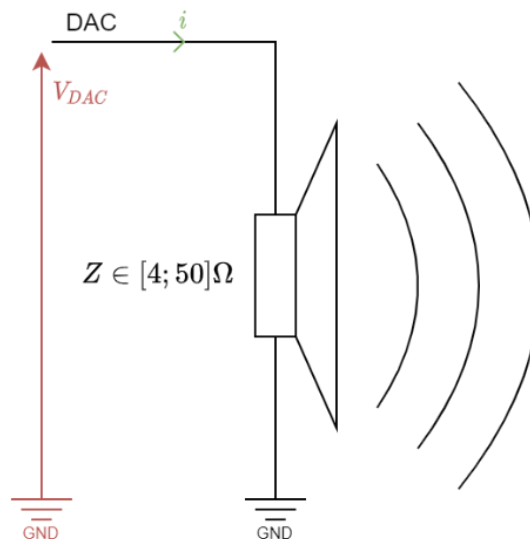


Figure 6 : Montage direct d'un haut-parleur sur le DAC

Dans ce montage on a pour la tension maximale de sortie du DAC $V_{DAC} = 3V$ et une impédance du haut-parleur $Z = 4\Omega$:

$$i = \frac{V_{DAC}}{Z} = \frac{3}{4} = 750mA$$

Dans ce cas, le courant à fournir est très supérieur au maximum spécifié. Il faut donc trouver une solution où $i < 25mA$. J'ai donc fait le choix de placer en série du haut-parleur une résistance $R = 1k\Omega$. Créant ainsi le montage suivant :

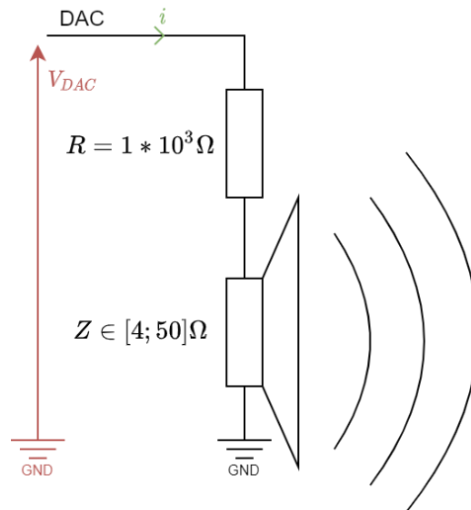


Figure 7 : Branchement d'un haut-parleur en série avec une résistance de 1 kΩ sur le DAC

Dans ce montage on obtient pour $V_{DAC} = 3V$ une impédance du haut-parleur $Z = 4 \Omega$:

$$i_{max} = \frac{V_{DAC}}{R + Z} = \frac{3}{4 + 1000} = 2,99 \text{ mA}$$

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 fabriqué un câble qui se connecte sur les connecteurs dupont de la carte et qui fournit une prise Jack stéréo de 3,5 mm. Ce câble précharge chacun des canaux stéréo par une résistance de 1kΩ en série avec chacun de ses deux canaux, et suit le schéma suivant :

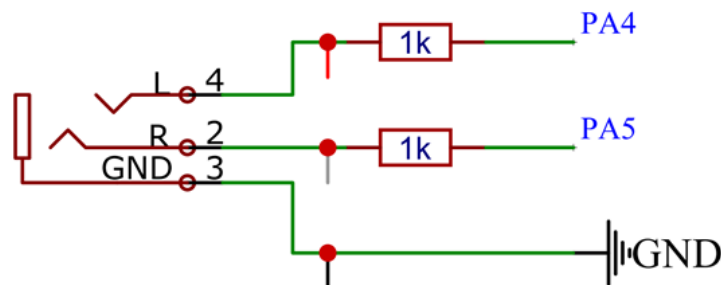


Figure 8 : Schéma électrique du câble DAC → Jack 3,5 mm

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



Figure 9 : Câble DAC → Jack 3,5 mm

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 correspond à la masse.

Du point de vue du logiciel, le DAC est configuré pour convertir l'échantillon suivant de chaque canal quand il reçoit un évènement en provenance Timer 2 du microcontrôleur. On peut configurer la fréquence de cet évènement via deux valeurs :

- "Prescaler" (PSC),
- "Counter Period" (ARR)

On calcule ces deux valeurs grâce à l'expression suivante :

$$(PSC + 1) * (ARR + 1) = \frac{f_{SYS}}{f_{TIM2}}$$

Où f_{SYS} 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 (f_{TIM2}) avec une fréquence $f_{SYS} = 72 * 10^6 \text{ Hz}$. On fixe $PSC = 0$ et donc on obtient :

$$ARR = \frac{f_{SYS}}{f_{TIM2}} - 1 = \frac{72 * 10^6}{48 * 10^3} - 1 = 1499$$

Pour fournir les échantillons à reproduire au DAC on utilise un buffer circulaire par canal du DAC. Les échantillons de ces buffers sont ensuite fournis au DAC à la fréquence f_{TIM2} par le DMA du microcontrôleur. Le DMA informe le programme principal de sa position dans les buffers par deux interruptions, une première qu'il lève quand il atteint la moitié du buffer et une seconde quand il atteint la fin de celui-ci. Ces interruptions sont ensuite traitées par le programme principal afin d'y charger de nouveaux échantillons, faisant ainsi en sorte qu'au moins une moitié du buffer soit toujours prête à être reproduite.

¹ Discovery Kit with STM32F429ZI MCU (ST Microelectronics, 2020)

² STM32F427xx STM32F429xx (ST Microelectronics, 2018)

3. Le traitement des échantillons.

3.1. Les échantillons.

3.1.1. Le format de sortie : 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 f_s appelée fréquence d'échantillonnage. Le résultat de cette mesure est stocké numériquement. On définit un signal PCM grâce aux deux valeurs suivantes :

- sa fréquence d'échantillonnage f_s ,
- le nombre de bits utilisés pour stocker la mesure n .

Cette deuxième valeur influe sur la précision de la valeur mesurée. En effet, plus n est important, plus le signal échantillonné aura de valeurs possibles et donc plus la mesure sera précise.

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

$$x(t) = A * \sin(2 * \pi * f * t) + B$$

Avec $f = 1 \text{ Hz}$, $A = \frac{1}{2}$ et $B = \frac{1}{2}$

Notre ADC peut mesurer une valeur appartenant à l'intervalle $[0; 1]$ 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 $f_s = 32 \text{ Hz}$. Cela nous donne donc les signaux suivants :

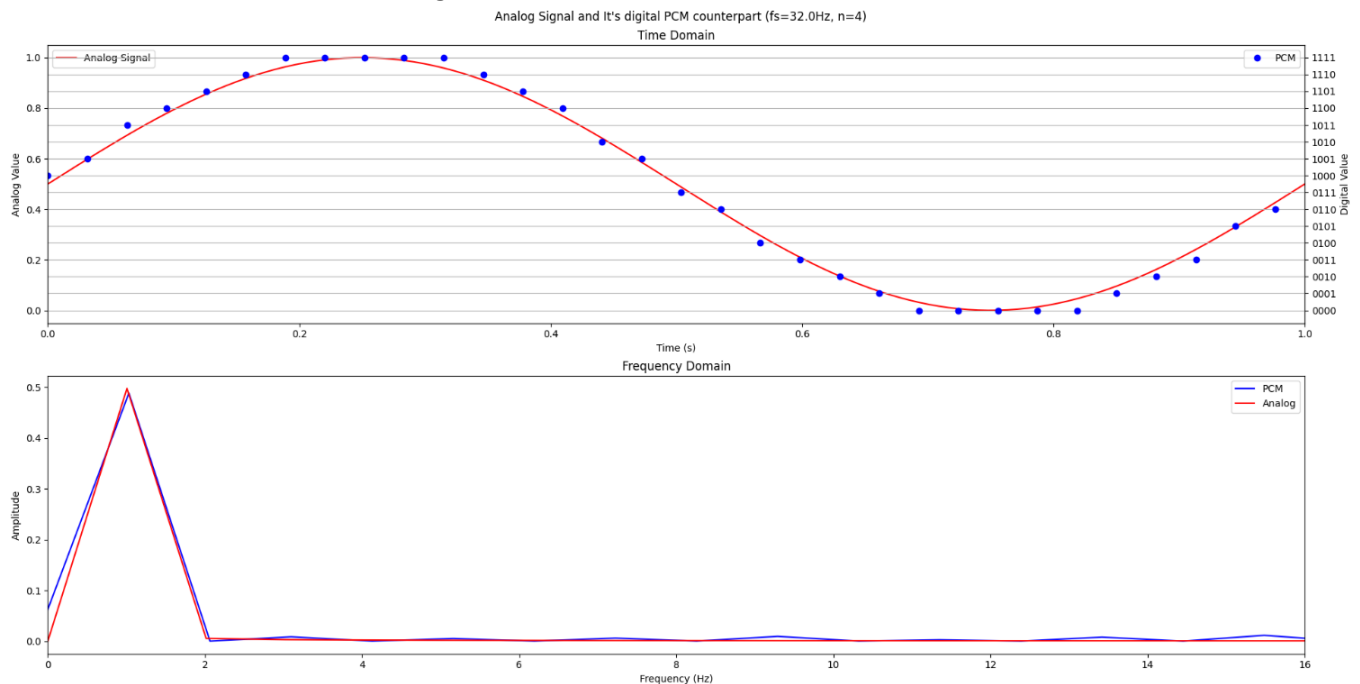


Figure 10 : Signal Analogique et sa conversion numérique en PCM ($f_s = 32 \text{ Hz}$, $n = 4$)

Les sons PCM sont généralement encodés sur 16 bits (65 536 valeurs possibles) avec une fréquence d'échantillonnage de 44,1 kHz pour les CD audio et 48 kHz pour la bande sonore d'un fichier vidéo.

3.1.2. Le format d'entrée : PDM.

Le PDM (Pulse Density Modulation) est 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 est déterminée à partir de la concentration de bit à 1 ou à 0. Si on utilise le signal analogique suivant que l'on observe pendant 1 seconde :

$$x(t) = A * \sin(2 * \pi * f * t) + B$$

Avec $f = 1 \text{ Hz}$, $A = 1$ et $B = 0$

Le signal PDM sera alors le suivant :

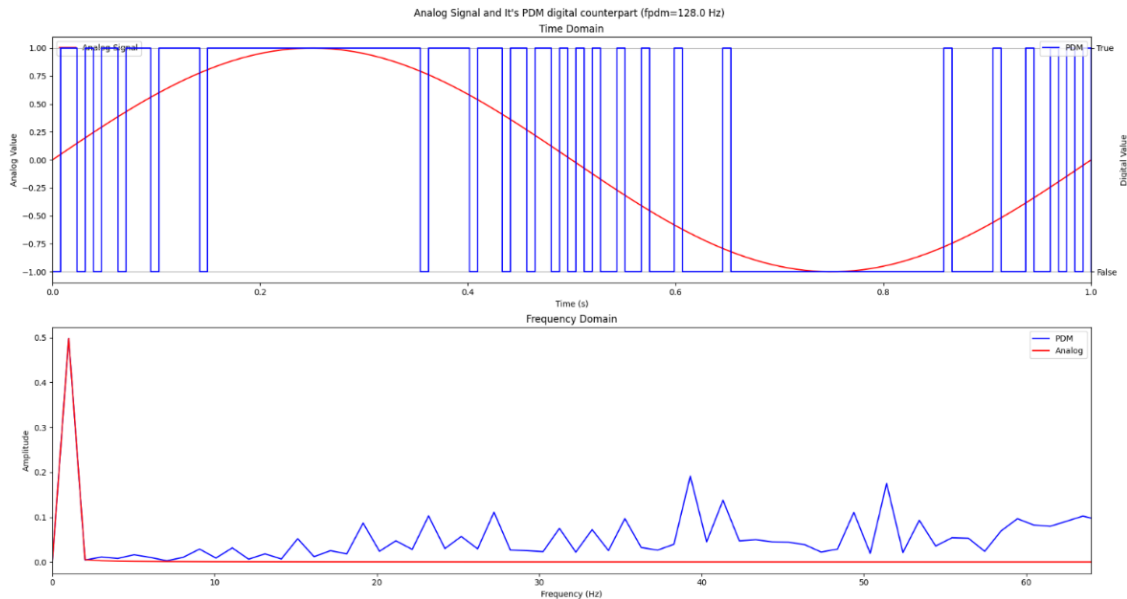


Figure 11 : Signal Analogique et sa conversion numérique en PDM ($f_{PDM} = 128\text{Hz}$)

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 caractérisée par une forte concentration locale d'échantillons PDM à 0. Un signal PDM est défini 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 pour qualité similaire. Généralement, la fréquence du flux PDM est entre 48 et 128 fois supérieure à la fréquence d'échantillonnage PCM.

3.2. Chaîne de filtrage.

La conversion d'un signal PDM échantillonné à haute fréquence (f_{SPDM}) vers un signal PCM échantillonné à plus basse fréquence (f_{SPCM}) se fait par le filtrage du signal PDM par un filtre passe-bas ayant une fréquence de coupure maximale $f_c = \frac{f_{SPCM}}{2}$. Une fois le signal PDM filtré, on le sous-échantillonne afin de faire correspondre sa fréquence d'échantillonnage à celle recherchée pour le signal audio (f_{SPCM}). Ce sous-échantillonnage est réalisé avec un facteur $D = \frac{f_{SPDM}}{f_{SPCM}}$, généralement $D \in [48; 128]$.

On applique ensuite au signal un offset et un gain afin de centrer sa valeur moyenne et adapter le niveau du signal au médium de sortie. La chaîne de filtrage correspondante est représentée en Figure 12.

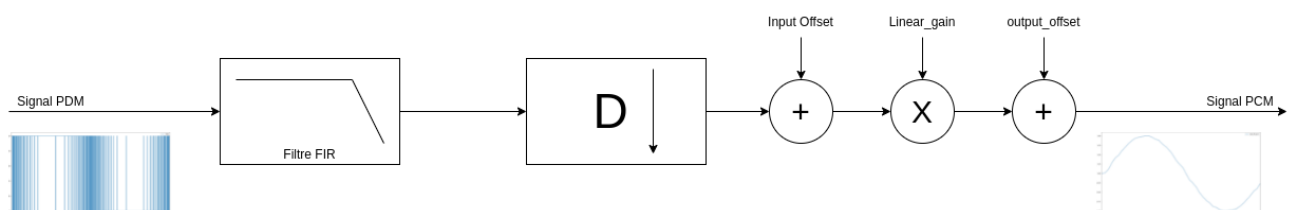


Figure 12 : Chaîne de filtrage PDM \rightarrow PCM

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

$$x(t) = \frac{2}{\pi} * \sum_{k=1}^{N_f} \frac{(-1)^{k+1}}{k} * \sin(2 * \pi * f_0 * k * t)$$

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

Paramètre	Valeur
Fréquence PDM	$f_{PDM} = 3.072 * 10^6 Hz$
Fréquence d'échantillonnage PCM	$f_{SPCM} = 48 * 10^3 Hz$
Fréquence fondamentale	$f_0 = 1 * 10^3 Hz$
Ordre de la série de Fourier	$N_f = 10$
Durée du signal	$T = 10 ms$

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

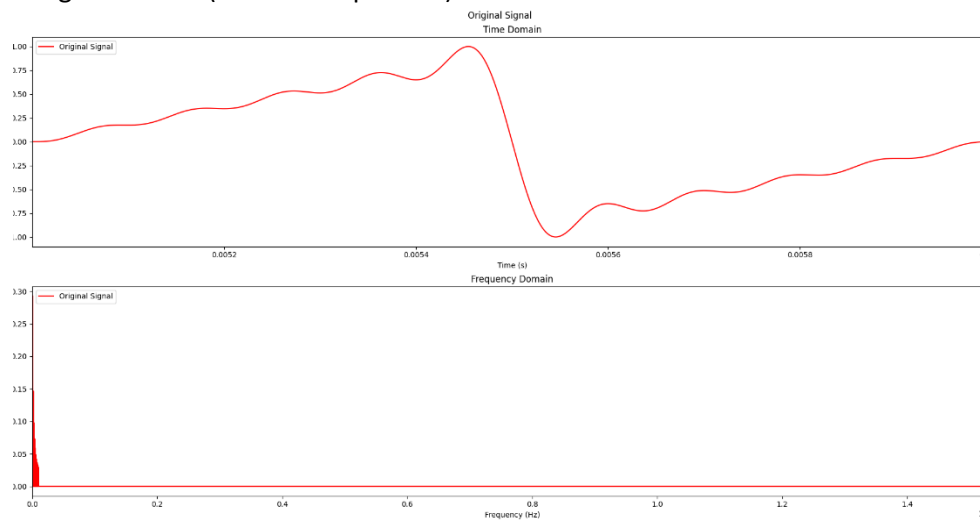


Figure 13 : Signal original

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

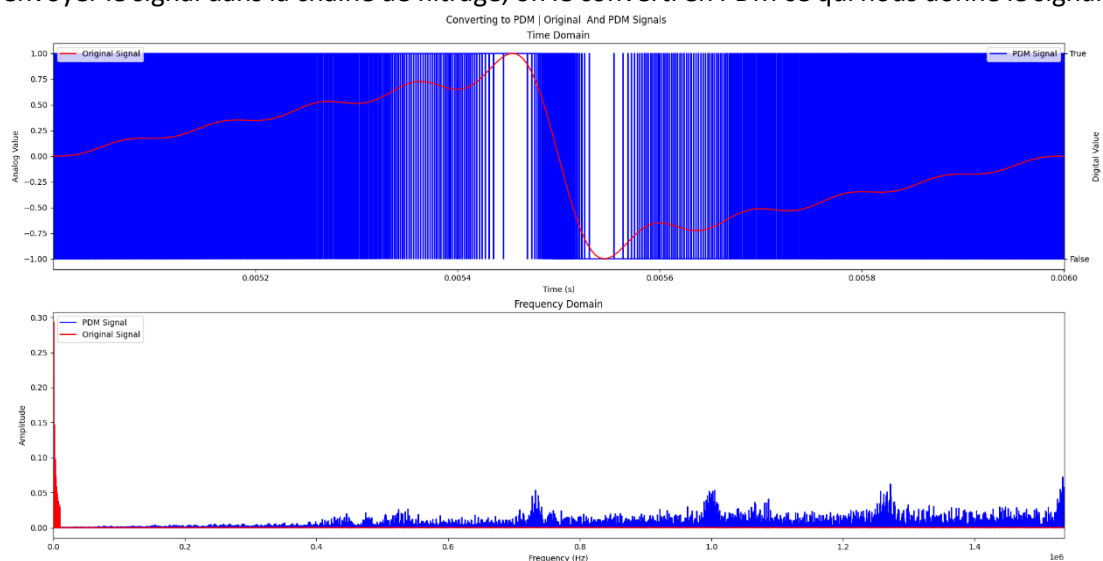


Figure 14 : Conversion PCM (rouge) → PDM (bleu)

La première étape de la chaîne est d'appliquer un filtre passe-bas au signal. Dans notre cas, on utilise un filtre numérique FIR (Finite Impulse Response) d'ordre N qui nous permet d'éliminer les hautes fréquences caractéristiques d'un signal PDM pour ne laisser que les fréquences utiles. Ici, l'ordre du filtre FIR est $N = 16$. À la suite du filtrage, on obtient le signal suivant :

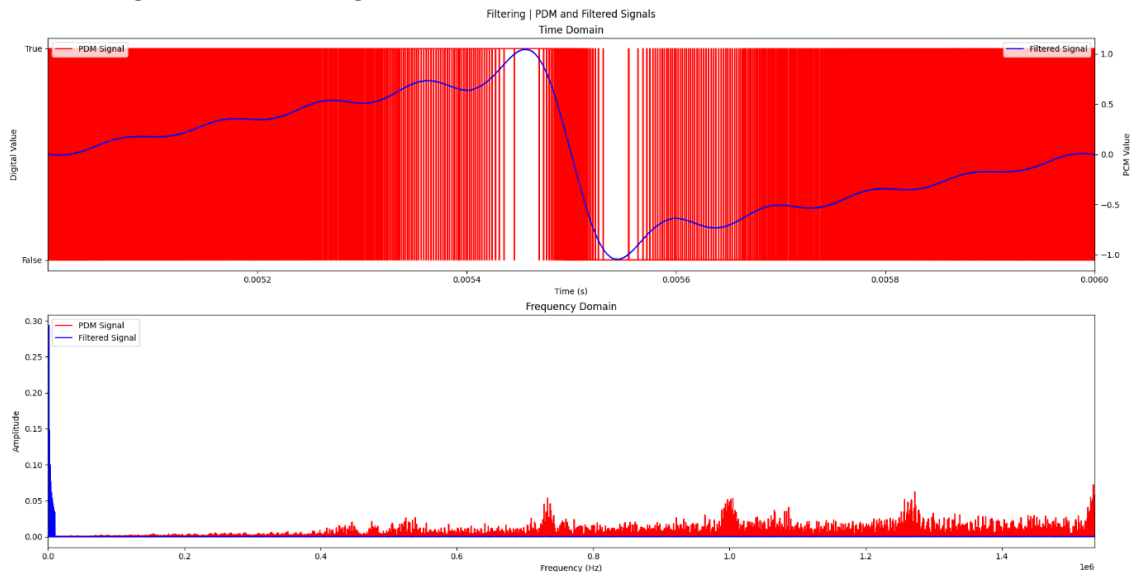


Figure 15 : Conversion PDM (Rouge) → PCM (Bleu)

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. Le filtrage du signal réduit donc fortement la bande passante nécessaire à la reproduction du signal. On va donc sous-échantillonner le signal filtré à la fréquence recherchée pour le signal PCM par un facteur D (on sélectionne un échantillon sur D) :

$$D = \frac{f_{PDM}}{f_{SPCM}}$$

Sous-échantillonnage qui nous donne le signal suivant :

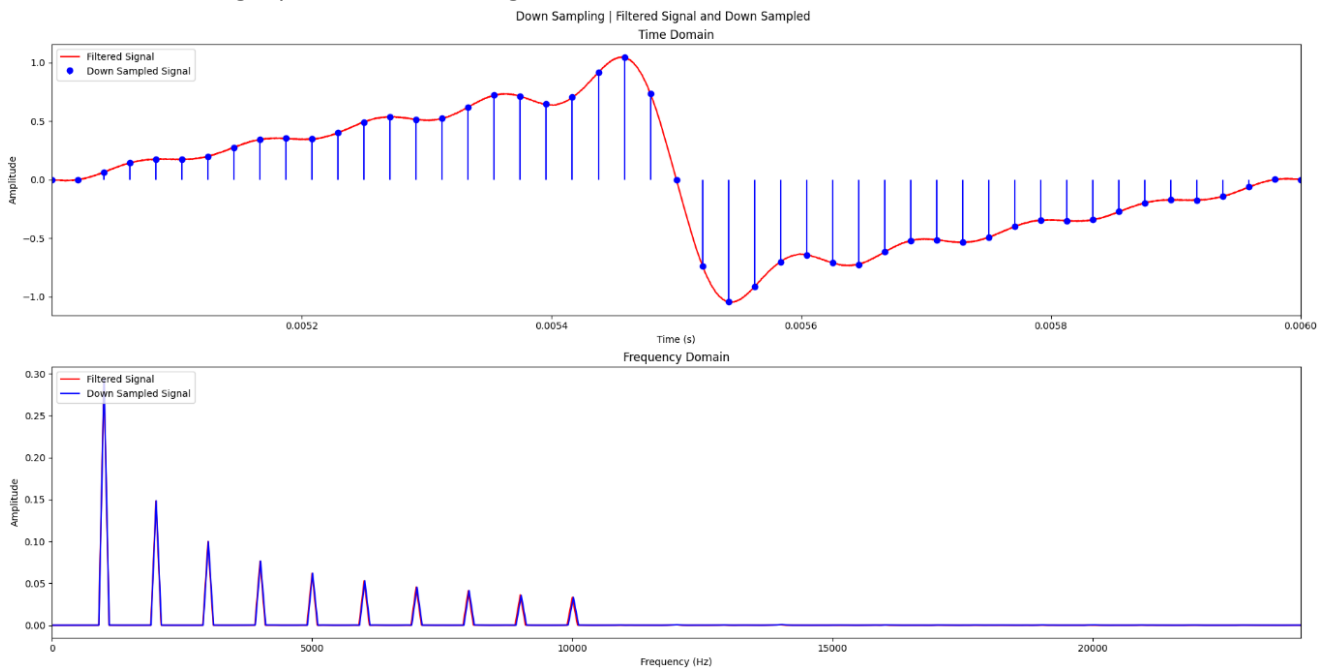


Figure 16 : Sous-échantillonnage du signal filtré

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 en sorti de chaine :

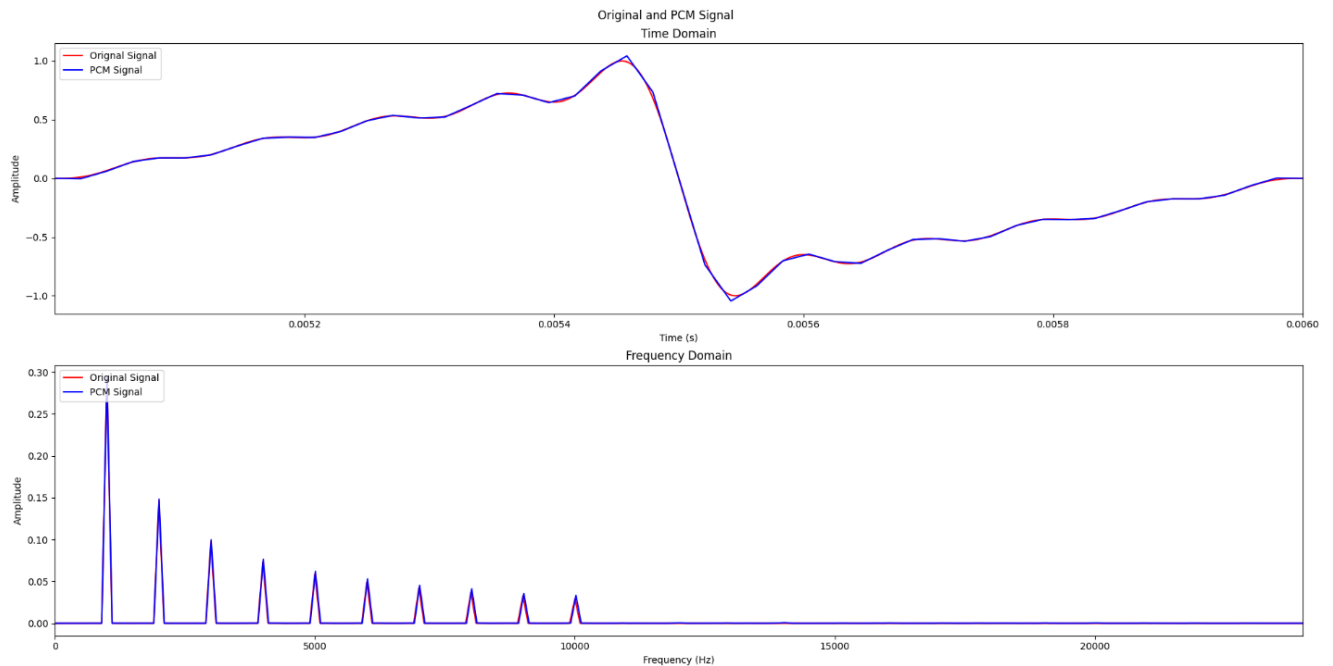


Figure 17 : Signal original (Bleu) et signal en sortie de chaine de filtrage (Rouge)

Afin de comparer le signal original au signal filtré, ce dernier n'a pas été amplifié et son offset est nul. Sur la Figure 17, on voit que forme du signal de sortie est très proche du signal original dans le domaine temporel, dans le domaine fréquentiel l'ensemble des raies du signal original sont présentes avec une amplitude similaire. Cette chaine de filtrage nous permet donc de produire un signal PCM fidèle au son original à partir d'échantillons PDM.

3.3. Intégration au microcontrôleur.

3.3.1. Fonctionnement.

Les échantillons PDM sont chargés en RAM par le DMA dès qu'un bloc de 16 échantillons PDM (16 bits) est mis à disposition par le périphérique SAI. 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 levé 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. Une fois, ces données récupérées depuis le buffer du DMA on les passe dans la chaîne de filtrage présenté précédemment. Ce qui nous donne le processus suivant que l'on intègre au programme principal :

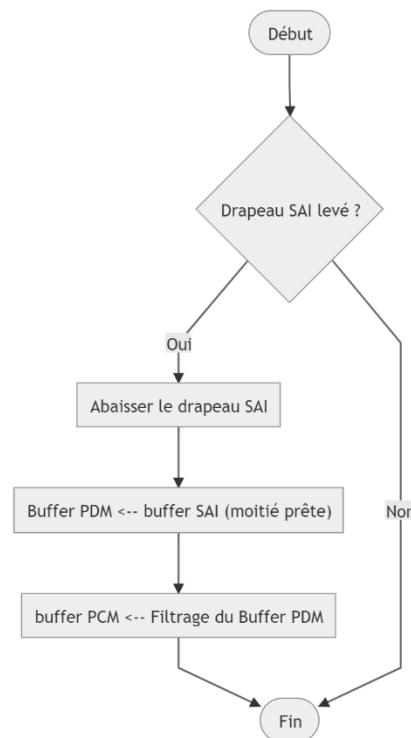


Figure 18 : Logigramme de la gestion des interruptions SAI

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

Le bloc de filtrage utilise une bibliothèque C³ que j'ai retravaillée pour l'adapter à nos besoins. La bibliothèque utilise la fenêtre FIR d'ordre N fournie par la bibliothèque Python "Scipy". En effet, un script Python fourni avec la bibliothèque permet de générer une LUT correspondant à la fenêtre FIR sous forme du tableau constant à deux dimensions $i = 2 * N ; j = 256$. L'axe i est deux fois plus grand que l'ordre du filtre, car la bibliothèque applique le filtre sur les groupes d'échantillons PDM de 16 bits en deux fois 8 bits MSB en premier. On retrouve donc sur l'axe i le positionnement de l'échantillon de 8 bits dans le buffer, et sur l'axe j le résultat de la convolution de l'échantillon avec la fenêtre. Ainsi sur la cible il suffit de sommer le résultat pour chaque échantillon du buffer de filtrage. L'amplitude de la LUT est amplifiée de façon occuper au maximum le conteneur de sortie dont on configure la taille à la génération de la LUT.

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 et la fenêtre FIR, et donc qui produit un échantillon PCM en appliquant l'algorithme suivant :

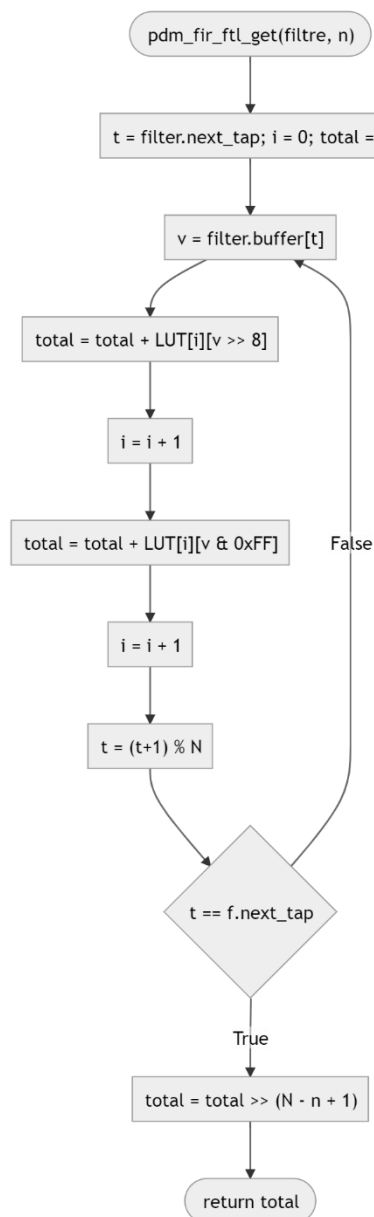


Figure 19 : Logigramme de la fonction "pdm_fir_ftl_get"

La bibliothèque nous permet donc d'implémenter le bloc de filtrage et une partie du bloc de sous-échantillonnage de la chaîne de filtrage PDM (cf. Figure 12). Pour compléter la chaîne, j'ai donc ajouté à la bibliothèque une structure de donnée contenant la configuration de la chaîne de filtrage (facteur de sous-échantillonnage, facteur d'amplification, offset...) ainsi que la fonction :

- "pdm_fir_ftl_chunk" :

Cette fonction produit le signal PCM correspondant au buffer PDM qu'on lui fournit en lui appliquant les différentes étapes de la chaîne 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.

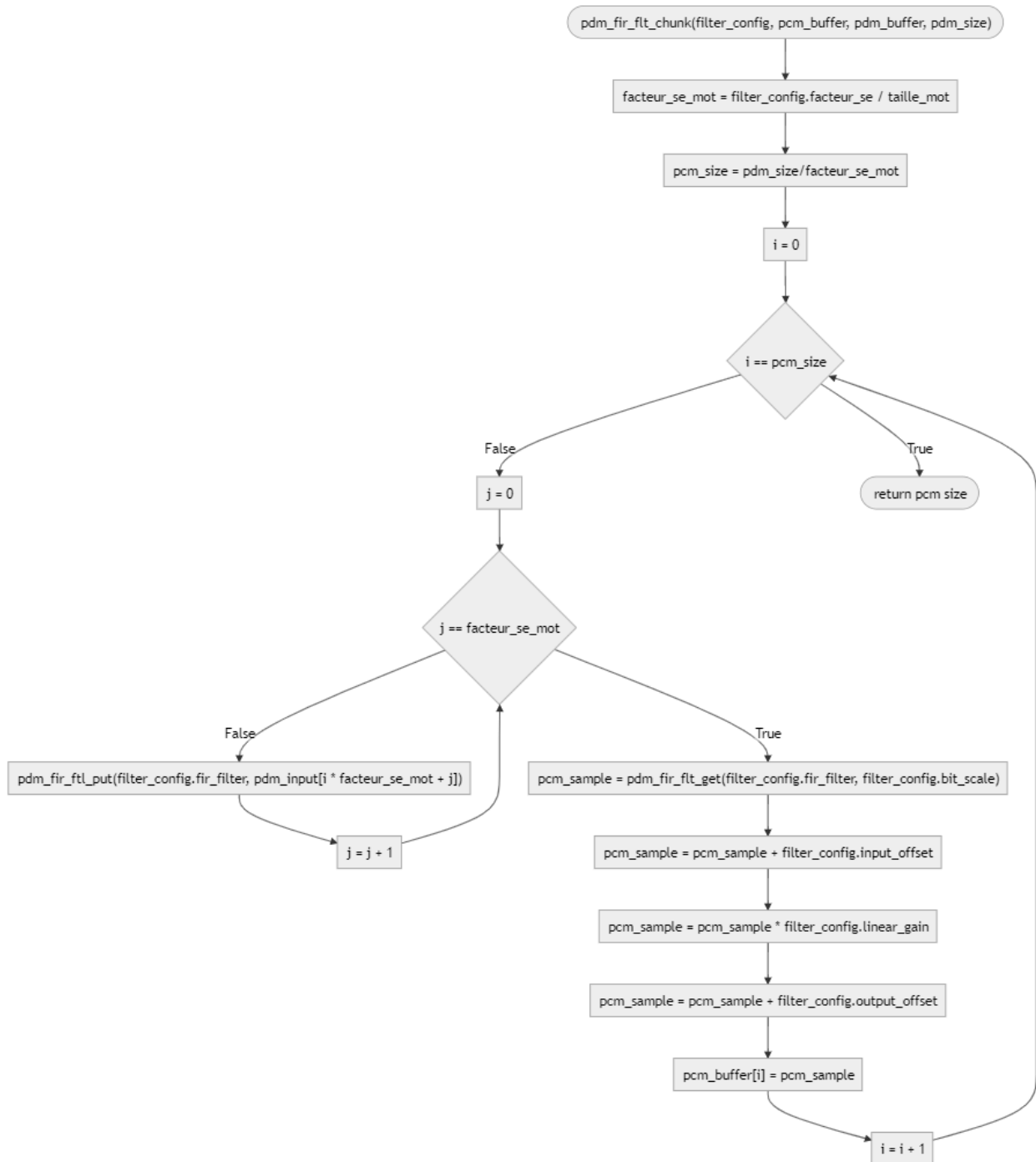


Figure 20 : Logigramme de la fonction "pdm_fir_ftl_chunk"

En conclusion, les données du microphone PDM sont recueillies par le périphérique SAI qui est couplé au DMA du microcontrôleur pour charger les échantillons PDM en RAM. Une fois ces échantillons chargés, le programme principal les fait passer dans la chaîne de filtrage pour produire les échantillons PCM correspondant. Cette chaîne de filtrage provient d'une bibliothèque C qui nous fournit le filtre FIR, le reste de la chaîne étant fournie par des sources que j'ai développées pour compléter la bibliothèque.

3.3.2. Performances de la chaîne de filtrage

Nous avons une bibliothèque C qui gère le filtrage du signal PDM, mais avant de pouvoir développer les démonstrateurs il faut s'assurer que les contraintes temps réel liées à l'acquisition du signal sont respectées par celle-ci. Pour ce faire, nous utiliserons le démonstrateur "Direct Output", qui rejoue en temps réel les sons captés. Le démonstrateur a été instrumenté de façon à fournir les signaux suivants sur le GPIO :

- Un signal à 3 V entre la levée du drapeau SAI et le début du filtrage et à 0 V le reste du temps
- Un signal à 3 V pendant le filtrage et à 0 V le reste du temps

Ces deux signaux nous donnant les chronogrammes suivants :

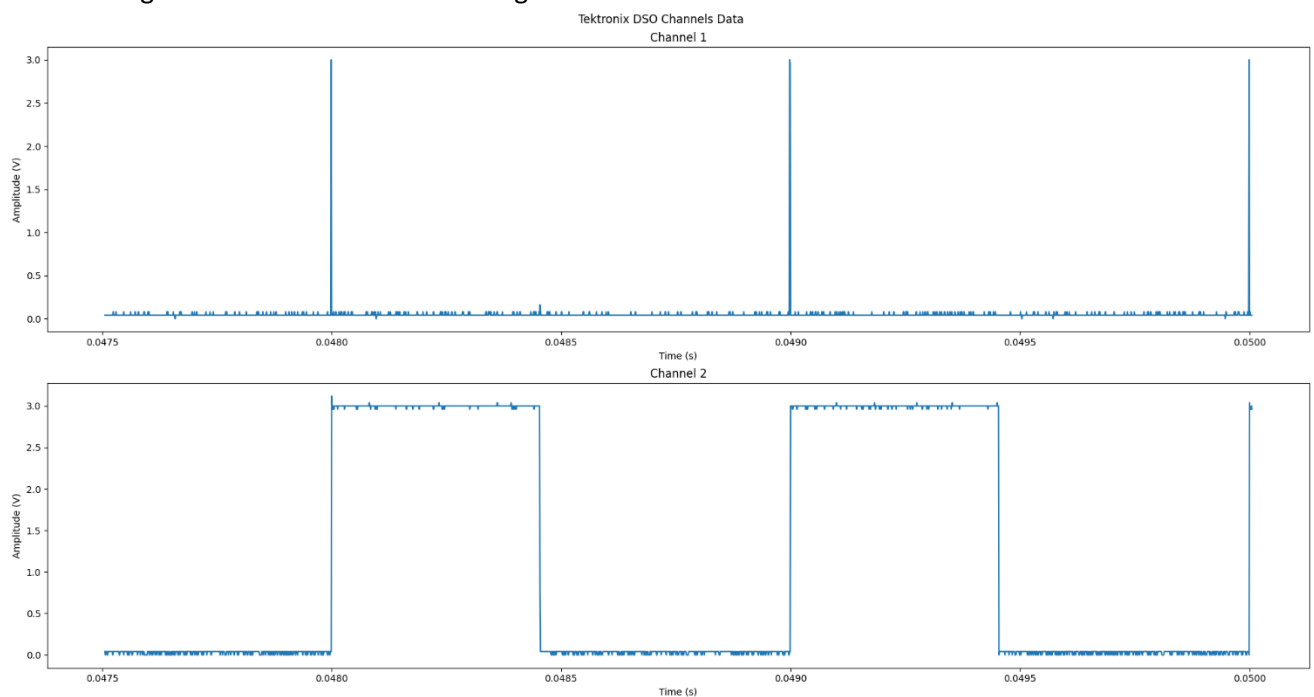


Figure 21 : Signaux « Drapeau SAI » (Channel 1) et « filtrage » (Channel 2)

Sur la Figure 21, le programme utilise les paramètres suivants : $f_{SPCM} = 48kHz$; $f_{system} = 72MHz$; $t_{SAI} = 1ms$.

Dans ces mesures, nous nous intéresserons plus particulièrement au temps de filtrage par rapport au temps entre deux drapeaux SAI. En effet quand l'on enregistre les sons ambiants l'ensemble des traitements effectués sur le signal doit être compris dans le cycle entre deux drapeaux SAI pour maintenir l'intégrité du signal capté. Notons que ces traitements comprennent le filtrage du signal, la recopie des échantillons sur le DAC, l'enregistrement des échantillons sur un périphérique... Notre objectif pour ces mesures est de déterminer le profil de l'occupation du cycle SAI par le filtrage en fonction des fréquences d'échantillonnage et de l'horloge système.

Dans le démonstrateur "Direct Output" qui nous sert de base, la chaîne de filtrage a été générée avec les paramètres suivants :

Paramètre	Valeur
Fréquence du signal PDM	$f_{PDM} = 3.072MHz$
Ordre de la fenêtre FIR	$N = 16$

Fréquence de coupure	$f_c = 20\text{kHz}$
Facteur de sous échantillonnage	$D = 64$
Offset avant amplification	$input_offset = -50$
Facteur d'amplification linéaire	$linear_gain = 10$
Offset après amplification	$output_offset = 127$
Taille des échantillons PCM (bits)	$n = 12$

Nous ferons donc trois séries de mesures à trois fréquences de l'horloge système différente. Pour chacune de ses séries, on a fixé le buffer de filtrage à 1 ms, c'est à dire que le cycle SAI dure 1 ms.

Nous ferons donc varier la fréquence d'échantillonnage PCM jusqu'à atteindre le point où le temps nécessaire pour filtrer les données est supérieur à 1 ms. À partir de ce point, le filtrage du signal (seul) ne nous permet plus de respecter les contraintes temps réels fixées par le cycle SAI.

L'abaque et la figure suivante présentent le résultat des mesures réalisées à trois fréquences d'horloge système (72, 108 et 144 MHz), pour les fréquences d'échantillonnage PCM entre 8 et 192 kHz avec un pas de 8 kHz :

$t_{SAI} = 1\text{ms}$	$f_{system} = 72 * 10^6\text{Hz}$		$f_{system} = 108 * 10^6\text{Hz}$		$f_{system} = 144 * 10^6\text{Hz}$	
f_{SPCM}	$t_{filtrage}$	Occupation	$t_{filtrage}$	Occupation	$t_{filtrage}$	Occupation
8 kHz	80 μs	8,00 %	52 μs	5,20 %	44 μs	4,40 %
16 kHz	150 μs	15,0 %	108 μs	10,8 %	88 μs	8,80 %
24 kHz	230 μs	23,0 %	164 μs	16,4 %	128 μs	12,8 %
32 kHz	310 μs	32,0 %	212 μs	21,2 %	166 μs	16,6 %
40 kHz	390 μs	39,0 %	264 μs	26,4 %	204 μs	20,4 %
48 kHz	450 μs	45,0 %	316 μs	31,6 %	248 μs	24,8 %
56 kHz	530 μs	53,0 %	372 μs	37,2 %	284 μs	28,4 %
64 kHz	600 μs	60,0 %	420 μs	42,0 %	328 μs	32,8 %
72 kHz	670 μs	67,0 %	476 μs	47,6 %	368 μs	36,8 %
80 kHz	750 μs	75,0 %	524 μs	52,4 %	408 μs	40,8 %
88 kHz	820 μs	82,0 %	576 μs	57,6 %	448 μs	44,8 %
96 kHz	900 μs	90,0 %	628 μs	62,8 %	488 μs	48,8 %
104 kHz	980 μs	98,0 %	680 μs	68,0 %	532 μs	53,2 %
112 kHz	1060 μs	106,0 %	732 μs	73,2 %	572 μs	57,2 %
120 kHz	Non mesuré	Non mesuré	784 μs	78,4 %	612 μs	61,2 %
128 kHz	Non mesuré	Non mesuré	840 μs	84,0 %	652 μs	65,2 %
136 kHz	Non mesuré	Non mesuré	892 μs	89,2 %	696 μs	69,6 %
144 kHz	Non mesuré	Non mesuré	940 μs	94,0 %	736 μs	73,6 %
152 kHz	Non mesuré	Non mesuré	995 μs	99,5 %	776 μs	77,6 %
160 kHz	Non mesuré	Non mesuré	1060 μs	106,0 %	816 μs	81,6 %
168 kHz	Non mesuré	Non mesuré	Non mesuré	Non mesuré	860 μs	86,0 %
176 kHz	Non mesuré	Non mesuré	Non mesuré	Non mesuré	896 μs	89,6 %
184 kHz	Non mesuré	Non mesuré	Non mesuré	Non mesuré	940 μs	94,0 %
192 kHz	Non mesuré	Non mesuré	Non mesuré	Non mesuré	985 μs	98,5 %

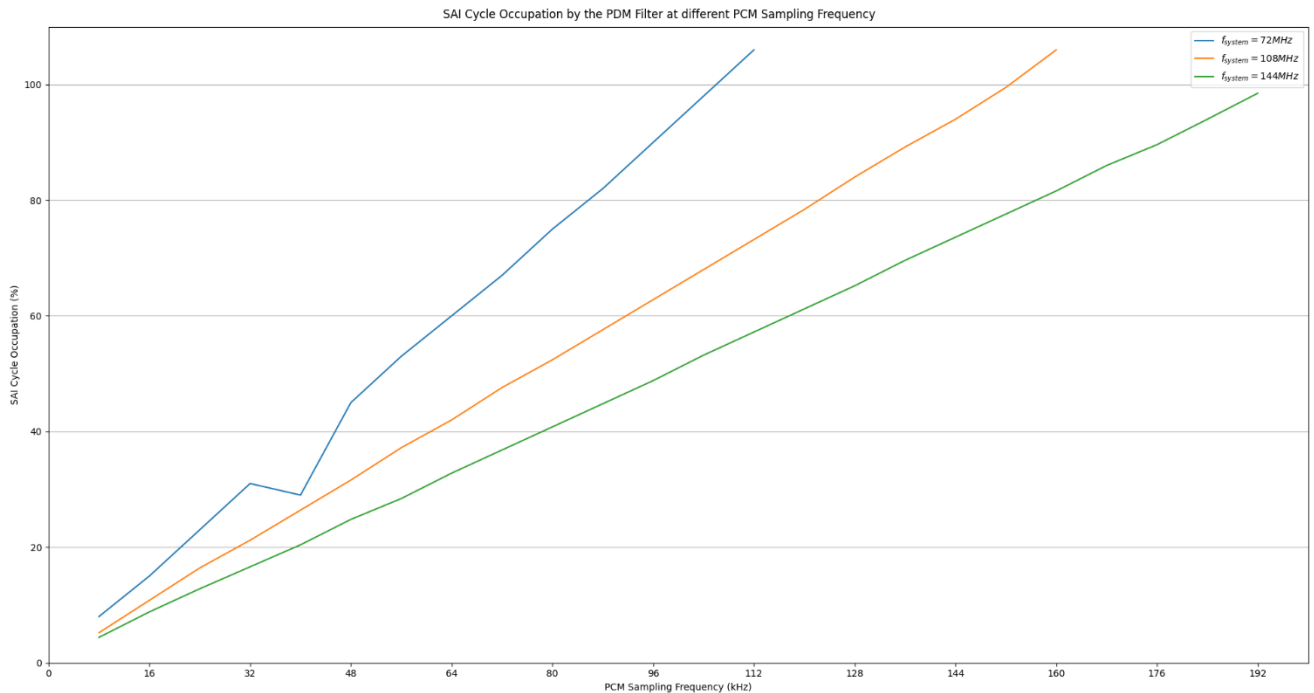


Figure 22 : Occupation du cycle SAI par le filtre PDM selon la fréquence d'échantillonnage PCM

On remarque sur les deux courbes que l'occupation du cycle SAI par le filtre est une fonction linéaire de la fréquence d'échantillonnage PCM. On remarque aussi que le coefficient directeur de cette fonction semble être proportionnel avec la fréquence de l'horloge système. Ce qui semble cohérent avec le fait que l'horloge système influe directement sur le nombre d'instructions par seconde qui sont exécutées par le microcontrôleur.

Dans le cas du démonstrateur "Direct Output" où l'on recherche une reproduction en temps réel des sons ambiants avec une bonne qualité audio, les paramètres :

- $f_{system} = 72 * 10^6 Hz$
- $t_{SAI} = 1 * 10^{-3} s$
- $f_{SPCM} = 48 * 10^3 Hz$

Permettent un taux d'occupation du cycle SAI de 45 % laissant ainsi une grande marge pour la reproduction du signal sur le DAC en offrant l'ensemble du spectre audible ($[20Hz; 20kHz]$).

³ PDM bitstream FIR filter (olegv142, 2017)

4. Les démonstrateurs

Afin de maîtriser les différents aspects de la chaîne 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.

4.1. "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 :

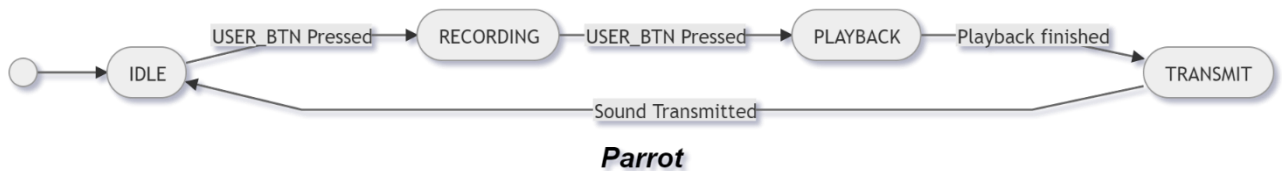


Figure 23 : Machine d'état du démonstrateur "Parrot"

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 est disponible en Annexe 1 : Logigrammes des États du démonstrateur "Parrot"

L'état "TRANSMIT" peut être désactivé en commentant la définition de la constante 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 :

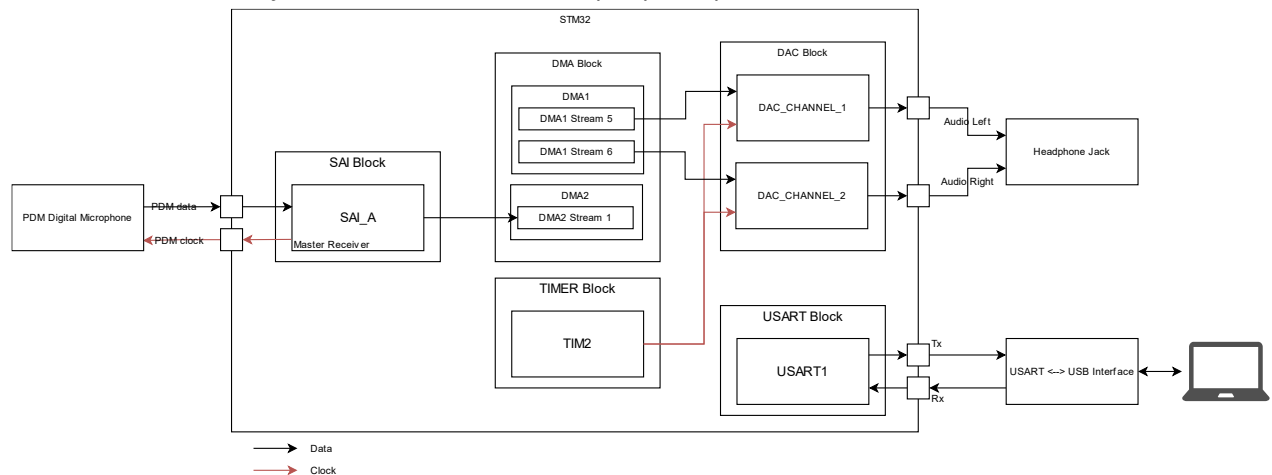


Figure 24 : Schéma Bloc des périphériques utilisés par le démonstrateur "Parrot"

Pour ce démonstrateur, j'ai fait le choix d'échantillonner le signal PCM à 32 kHz afin de pouvoir stocker 3 secondes de son ($bytes_per_sample * sample_rate * record_time = 2 * 32000 * 3 = 187.5 Kio$) en saturant au maximum la mémoire RAM (192 Kio) du microcontrôleur. Si l'on utilise une fréquence d'échantillonnage PCM de 48 kHz, le temps maximum d'enregistrement descend à 2 secondes ($\frac{32*10^3}{48*10^3} * 3$) ce qui peut paraître 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 format WAV

4.2. "Digital recorder"

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

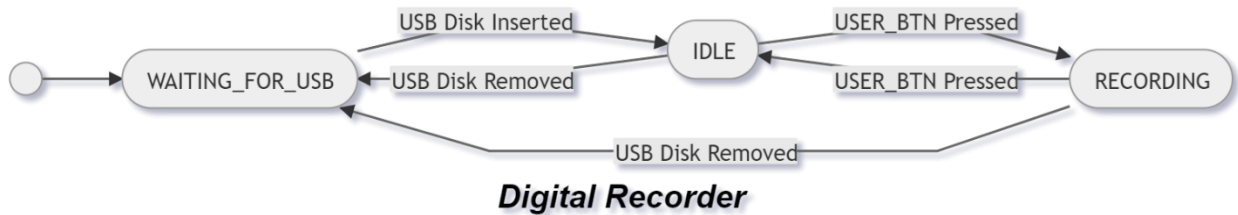


Figure 25 : Machine d'état du démonstrateur "Digital Recorder"

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 est disponible en Annexe 2 : Logigrammes des États du démonstrateur "Digital Recorder"

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

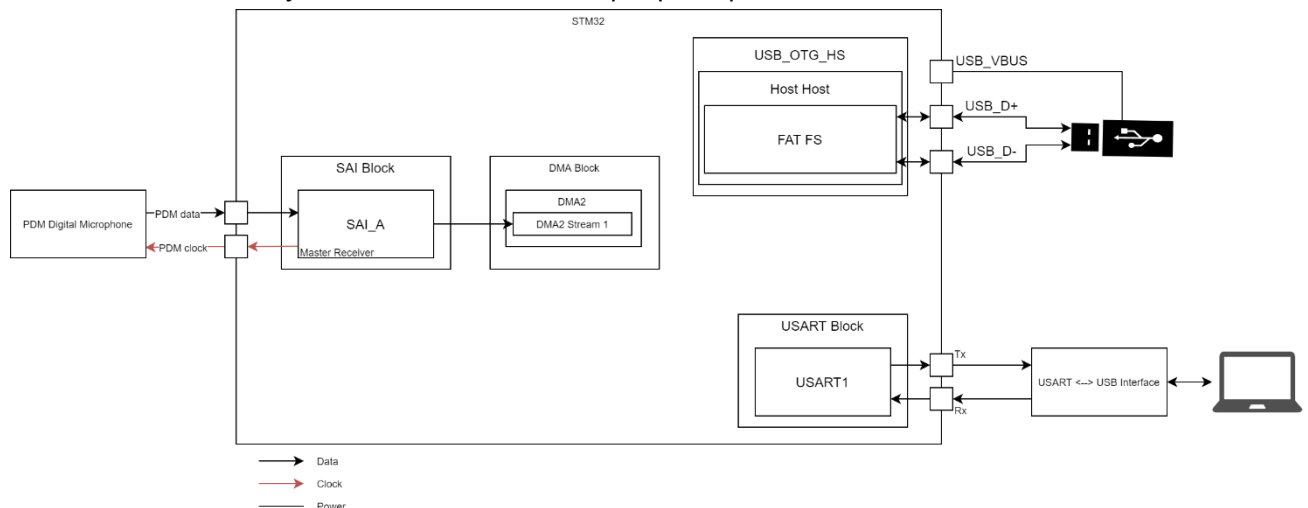


Figure 26 : Schéma Bloc des périphériques utilisés par le démonstrateur "Digital Recorder"

4.3. "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 :



Figure 27 : Machine d'états du démonstrateur "Direct Output"

- "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 est disponible en Annexe 3 : Logigrammes des États du 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 chaîne de capture.

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

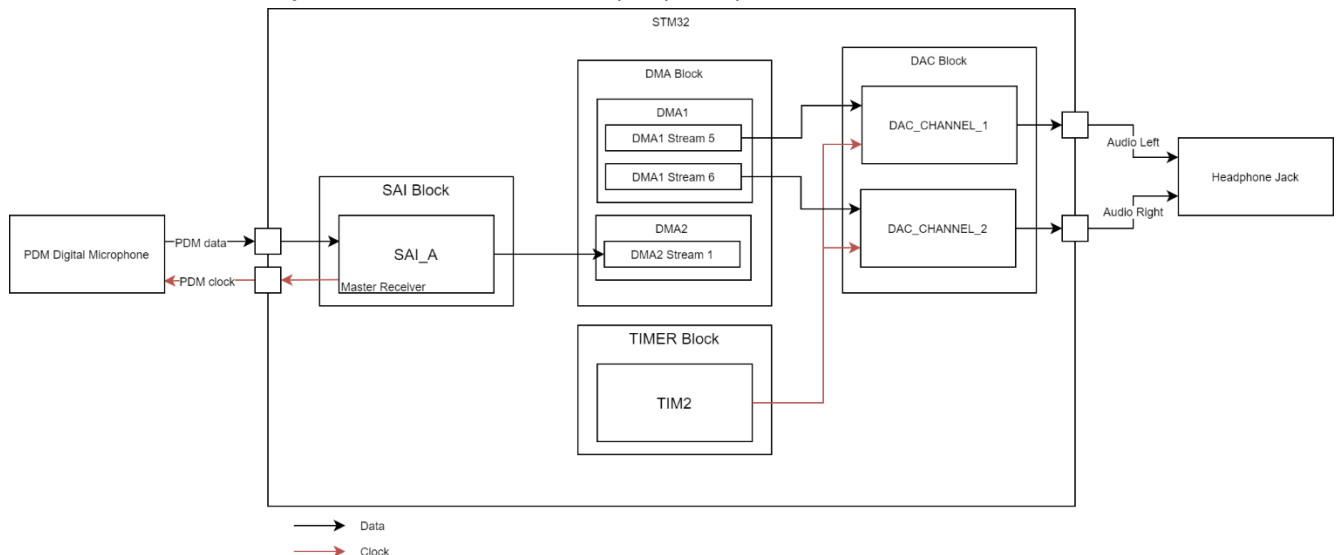


Figure 28 : Schéma Bloc des périphériques utilisés par le démonstrateur "Direct Output"

Les paramètres de la chaîne de capture pour chacun des démonstrateurs sont résumés dans le tableau suivant :

		Parrot	Digital Recorder	Direct Output
Fréquence du flux PDM	f_{PDM}	2.048 MHz	3.072 MHz	3.072 MHz
Facteur de sous échantillonnage	D	64	64	64
Fréquence d'échantillonnage PCM	f_{SPCM}	32 kHz	48 kHz	48 kHz
Durée du cycle SAI	t_{SAI}	1ms	10 ms	1 ms
Fréquence système	f_{system}	72 MHz	168 MHz	72 MHz
Fréquence de coupure du filtre	f_c	16kHz	20 kHz	20 kHz
Ordre du filtre	N	16	16	16
Taille des échantillons PCM	n	12 bits	16 bits	12 bits

Conclusion

En conclusion, l'étude et le développement d'un programme permettant l'acquisition et la restitution en temps réel des sons ambiants par un microphone PDM sont des thèmes intéressants pour un projet étudiant. En effet, ils permettent d'aborder les différentes étapes de la conception d'un système embarqué. Aborder un tel sujet nécessite de commencer par l'étude du microcontrôleur/microprocesseur cible, puis des périphériques que l'on utilise pour la capture et la restitution des sons. Une fois la plateforme étudiée, on travaille sur le format des données en provenance du microphone, pour prototyper le traitement de ces données grâce à des langages de haut niveau. Une fois l'algorithme développé, on peut l'embarquer en y ajoutant les problématiques de capture en temps réel via le DMA et donc la gestion d'interruptions. De plus, l'embarquement d'un code permet de mieux appréhender les limitations d'un microcontrôleur en termes de capacité mémoire et en puissance de calcul. Ces limitations qui conduisent souvent à la transformation de l'algorithme intuitif à haut niveau en un algorithme plus complexe à comprendre, mais qui est plus adapté à la cible.

Cependant, le filtrage d'un signal PDM tels qu'étudier dans ce rapport implique l'utilisation d'une fenêtre FIR dans le domaine temporel et donc la convolution d'un signal. Or l'algorithme qui permet la convolution prend beaucoup de temps à comprendre puis à développer et optimiser. Ce qui risque pour des projets où le temps est limité de soit décourager les étudiants, soit monopoliser leurs temps et donc ne pas leur permettre d'atteindre les objectifs fixés. Une solution pourrait être de fournir la librairie de filtrage aux étudiants une fois qu'ils ont prototypé leurs filtres sur un langage de haut niveau. Une deuxième solution serait d'utiliser une cible équipée d'un DFSDM (Digital Filter For Sigma Delta Modulators) qui réaliser le filtrage des échantillons PDM de manière matérielle. Rendant ainsi le logiciel plus simple, car il s'agirait uniquement de traiter les entrées sorties ainsi que les états du programme.

D'un point de vue personnel, j'ai trouvé ce projet très enrichissant, car il m'a permis de découvrir la famille des microcontrôleurs STM32. Ce projet m'a aussi permis de parfaire mes connaissances dans la capture de son et qui plus est de l'embarquement de celle-ci. Un point très intéressant de mon point de vue est que ce projet ne se concentrait pas uniquement sur l'embarquement d'une fonctionnalité, mais surtout sur la création de celle-ci puis son embarquement. Ce qui permet de prendre du recul sur la fonctionnalité en elle-même puis de prototyper via des langages de haut niveau sur PC. Pour finalement embarquer le prototype en prenant en compte les contraintes liées à la cible (puissance de calcul et espace mémoire limité, notion de temps réel et gestion d'interruption).

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. Ce qui lui permet de 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 ralentir l'exécution du programme principal
GPIO	"General Purpose Input Output". Entrées/Sortie du microcontrôleur qui ne sont pas affectées à des périphériques spécifiques
LookUp Table (LUT)	« Table de Correspondance » Structure de données qui contient des données précalculer pour afin de réduire 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.
LSB	"Least Significant Byte" Octet de poids faible, dans un mot binaire dont la taille est supérieure à un octet le LSB qualifie l'octet dont le poids est plus faible dans l'évaluation du nombre
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.
MSB	"Most Significant Byte" Octet de poids fort, dans un mot binaire dont la taille est supérieure à un octet le MSB qualifie l'octet dont le poids est plus important dans l'évaluation du nombre
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 est 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 : $q = \frac{E}{2^n}$ où E correspond à la plage de tension de travail du convertisseur en Volt, et n 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 On The Go" 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)

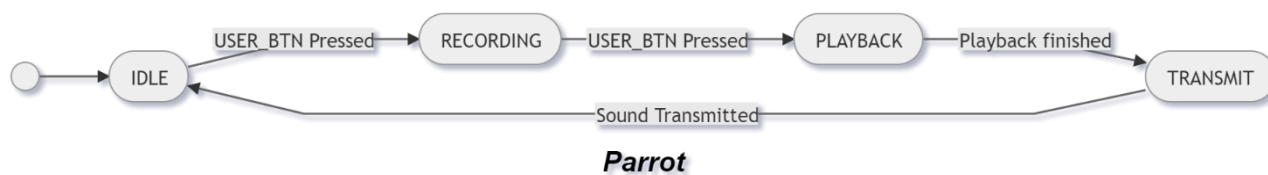
Table des figures

Figure 1 : Comportement du démonstrateur obligatoire	5
Figure 2 : Carte Cible (STM32F429I-DISC1)	6
Figure 3 : Schéma bloc typique d'un microphone PDM	6
Figure 4 : Connexion du Microphone PDM en configuration mono	7
Figure 5 : Chaîne de capture	7
Figure 6 : Montage direct d'un haut-parleur sur le DAC	8
Figure 7 : Branchement d'un haut-parleur en série avec une résistance de 1 kΩ sur le DAC	9
Figure 8 : Schéma électrique du câble DAC → Jack 3,5 mm	9
Figure 9 : Câble DAC → Jack 3,5 mm	9
Figure 10 : Signal Analogique et sa conversion numérique en PCM ($f_s = 32\text{ Hz}$, $n = 4$)	11
Figure 11 : Signal Analogique et sa conversion numérique en PDM ($f_{PDM} = 128\text{ Hz}$)	12
Figure 12 : Chaîne de filtrage PDM → PCM	12
Figure 13 : Signal original	13
Figure 14 : Conversion PCM (rouge) → PDM (bleu)	13
Figure 15 : Conversion PDM (Rouge) → PCM (Bleu)	14
Figure 16 : Sous-échantillonnage du signal filtré	14
Figure 17 : Signal original (Bleu) et signal en sortie de chaîne de filtrage (Rouge)	15
Figure 18 : Logigramme de la gestion des interruptions SAI	16
Figure 19 : Logigramme de la fonction "pdm_fir_ftl_get"	17
Figure 20 : Logigramme de la fonction "pdm_fir_ftl_chunck"	18
Figure 21 : Signaux « Drapeau SAI » (Channel 1) et « filtrage » (Channel 2)	19
Figure 22 : Occupation du cycle SAI par le filtre PDM selon la fréquence d'échantillonnage PCM	21
Figure 23 : Machine d'état du démonstrateur "Parrot"	22
Figure 24 : Schéma Bloc des périphériques utilisés par le démonstrateur "Parrot"	22
Figure 25 : Machine d'état du démonstrateur "Digital Recorder"	23
Figure 26 : Schéma Bloc des périphériques utilisés par le démonstrateur "Digital Recorder"	23
Figure 27 : Machine d'états du démonstrateur "Direct Output"	24
Figure 28 : Schéma Bloc des périphériques utilisés par le démonstrateur "Direct Output"	24

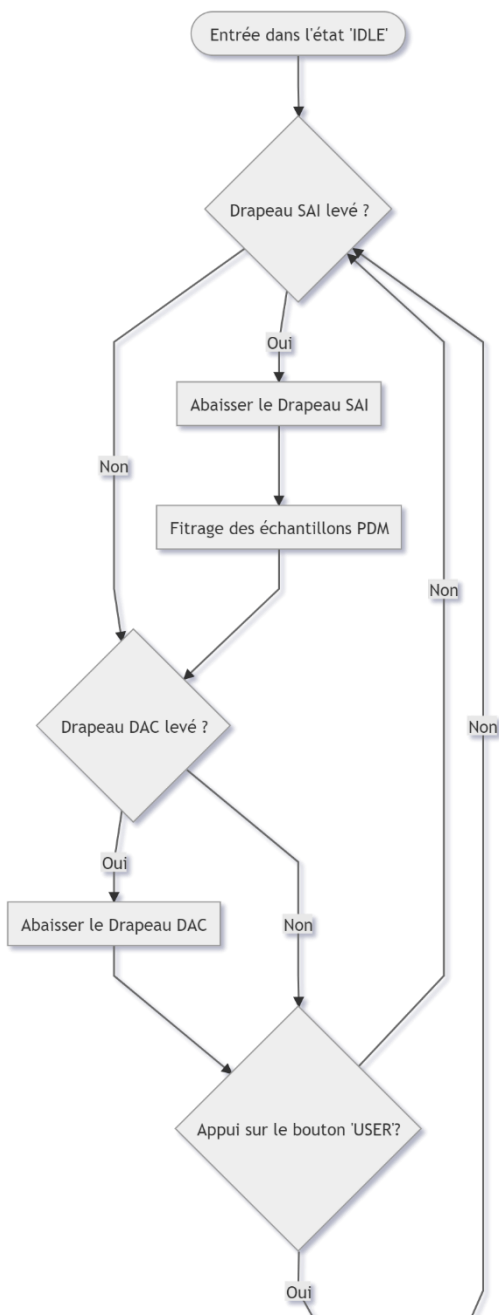
Annexes

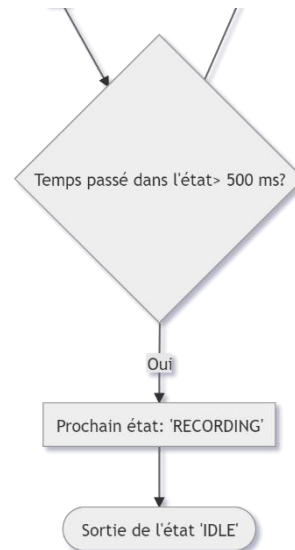
Annexe 1 : Logigrammes des États du démonstrateur "Parrot"

Machine d'état

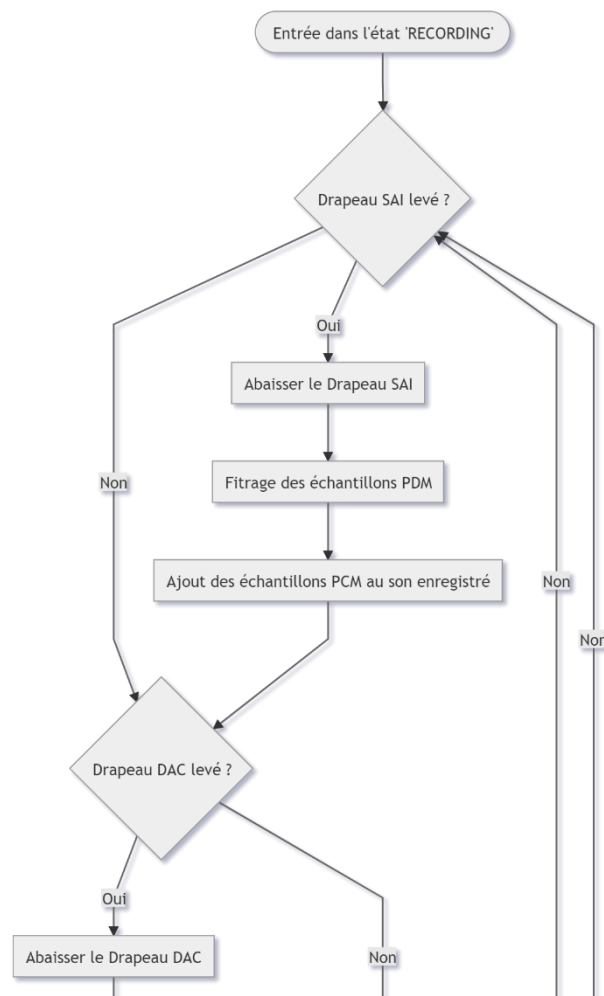


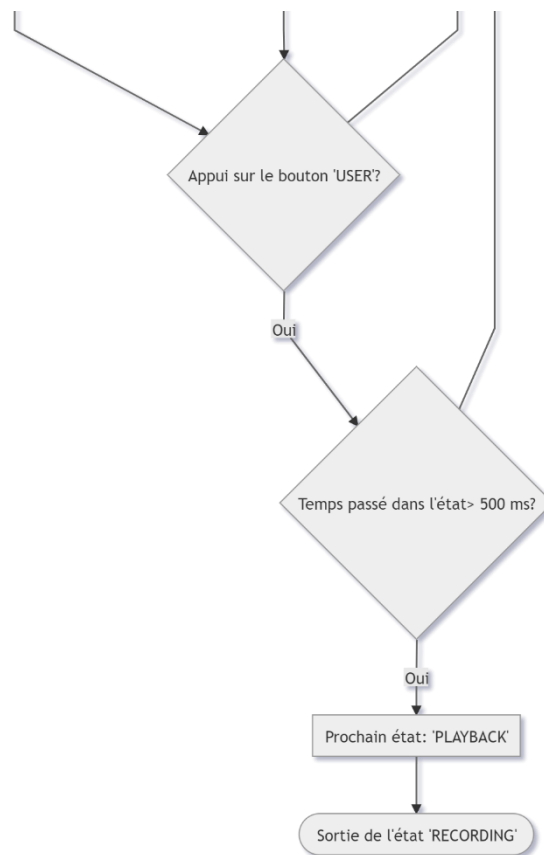
IDLE



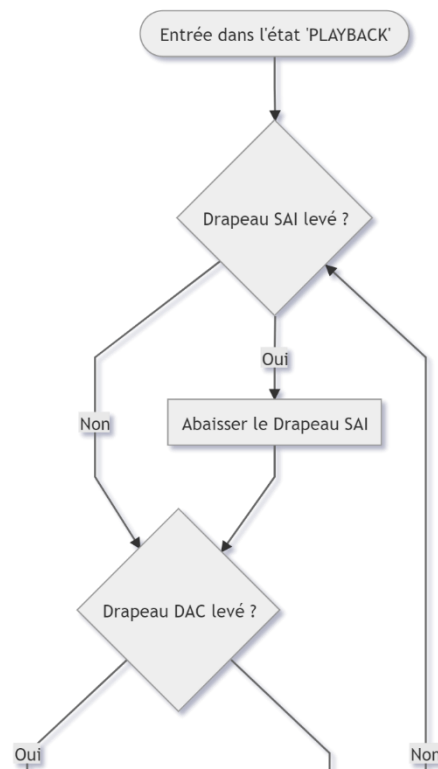


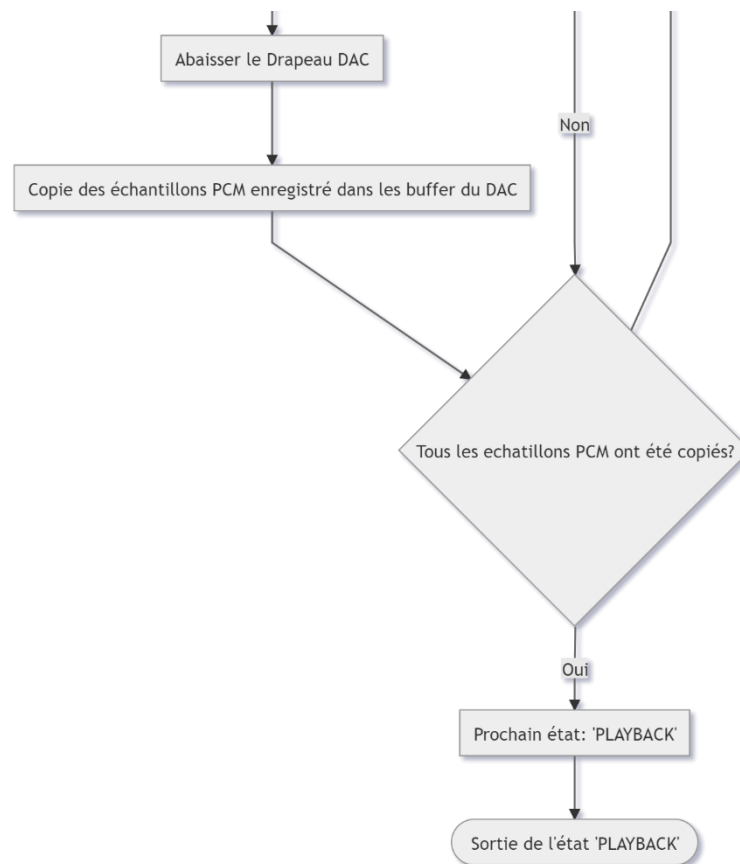
RECORDING



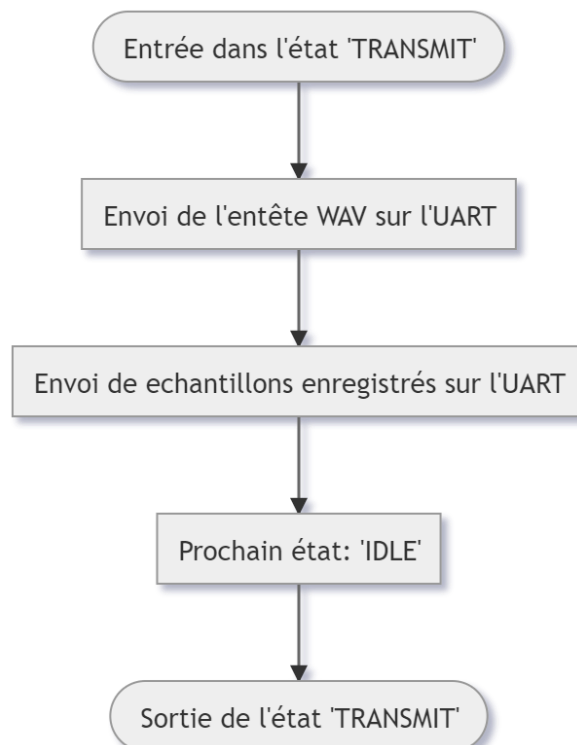


PLAYBACK



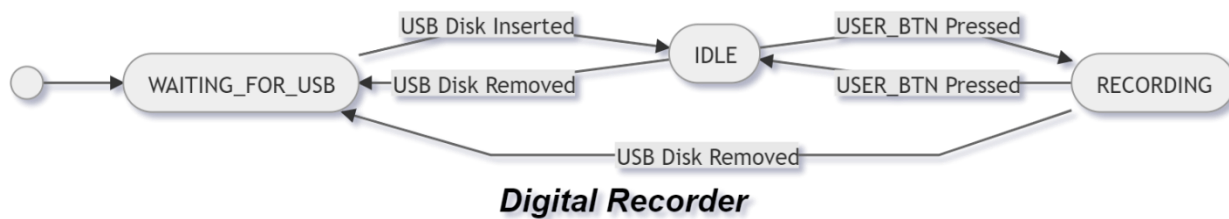


TRANSMIT

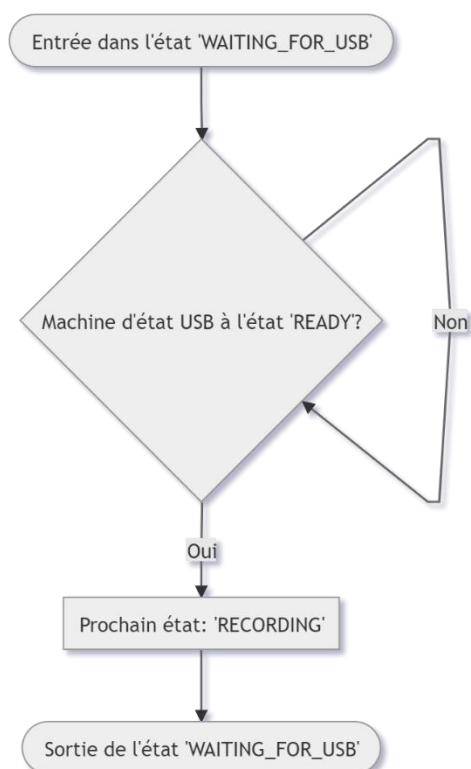


Annexe 2 : Logigrammes des États du démonstrateur "Digital Recorder"

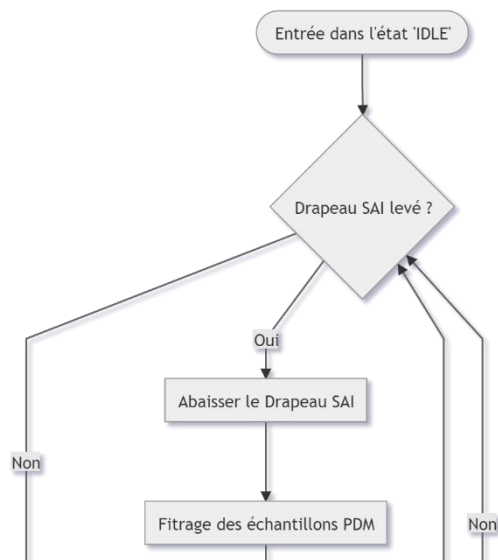
Machine d'état

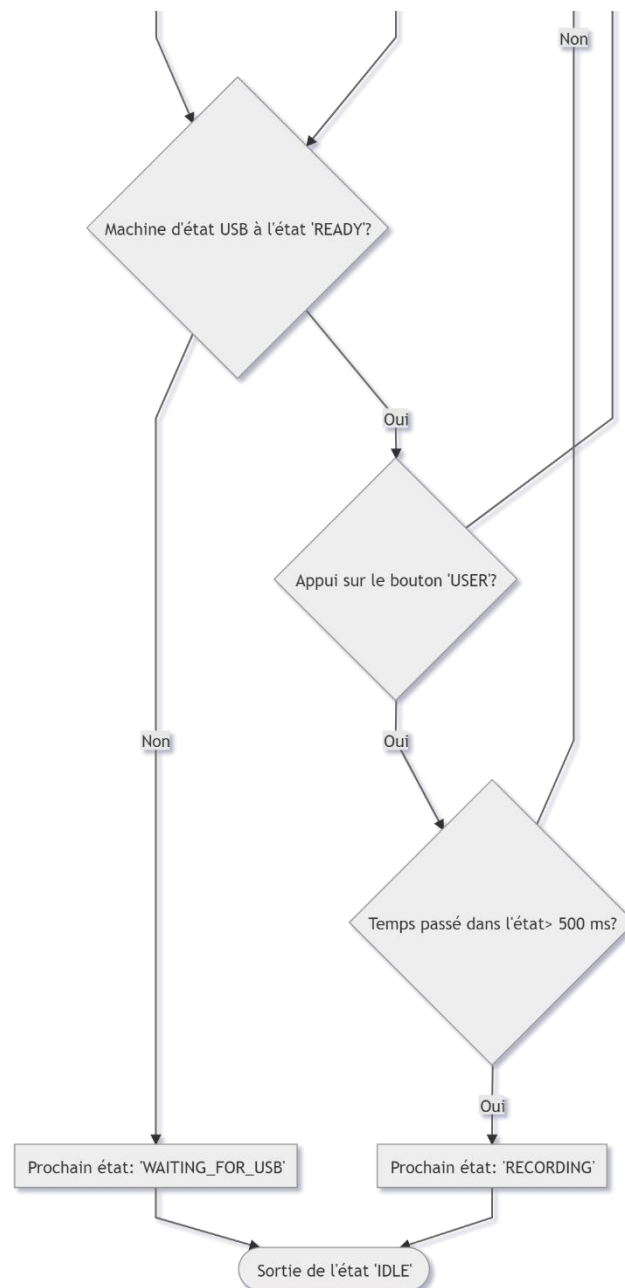


WAITING_FOR_USB

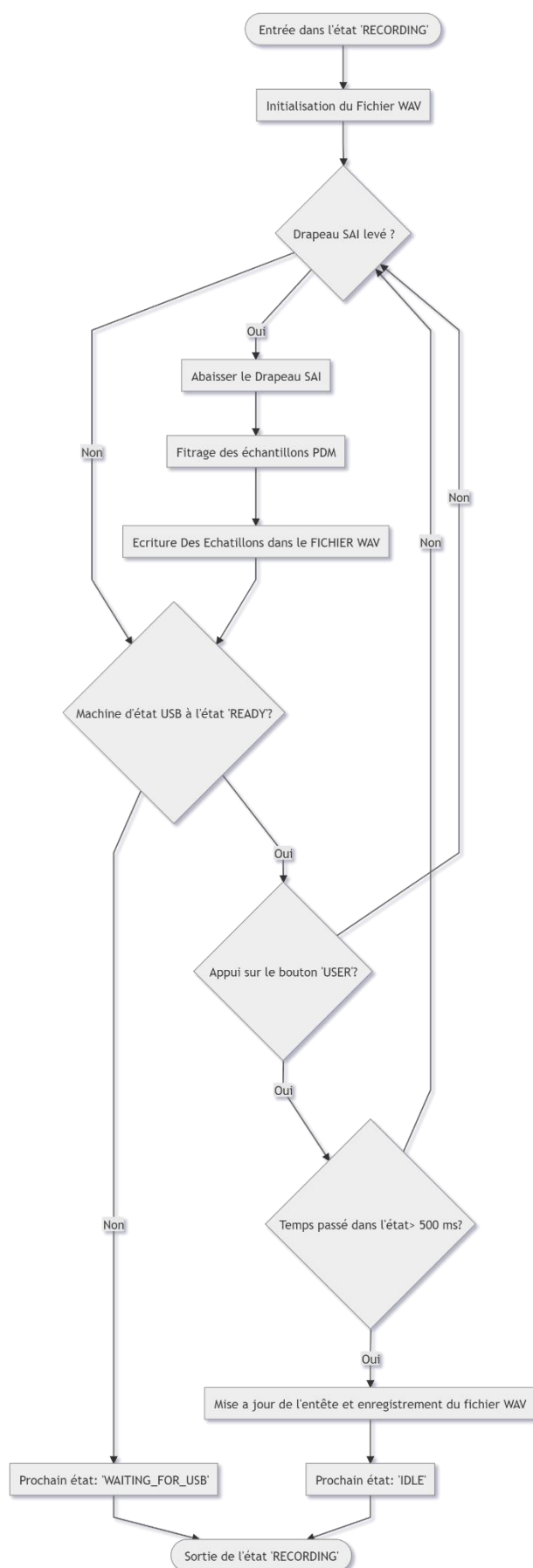


IDLE



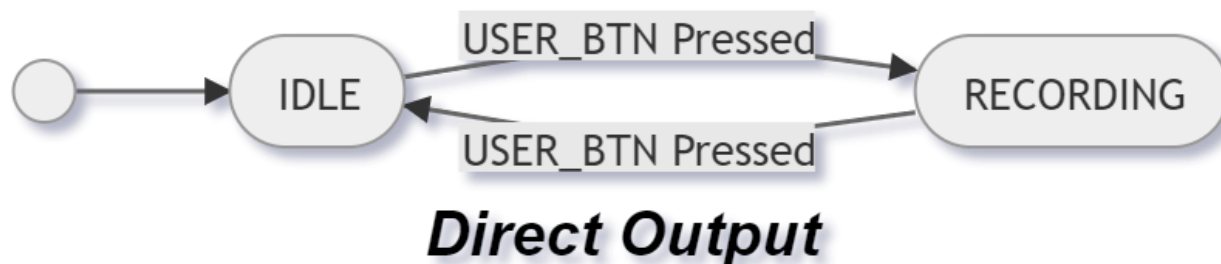


RECORDING

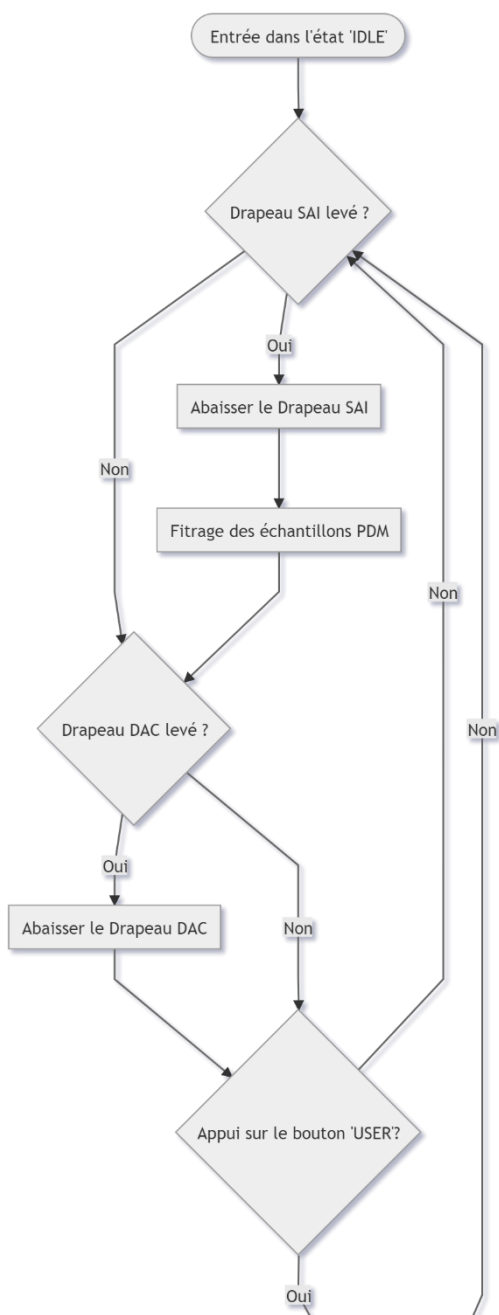


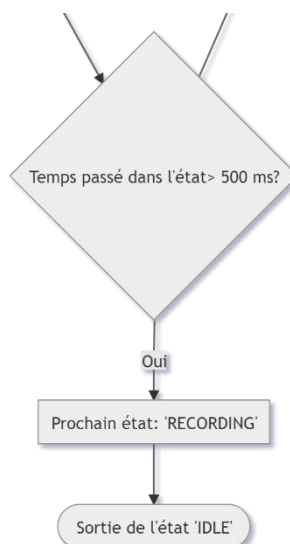
Annexe 3 : Logigrammes des États du démonstrateur "Direct Output"

Machine d'état



IDLE





RECORDING

