

*Rapport de stage de quatrième année du cycle ingénieur*

*Charrière Clément*

## **Développement d'un module RED PITAYA pour la caractérisation de capteurs MEMS**



Tome principal

Filière Informatique et Électronique des Systèmes Embarqués  
Année universitaire 2023/2024  
Période du 15 avril 2024 au 30 août 2024



## I. Remerciements

Tout d'abord je souhaite remercier l'ensemble des personnes présentes pendant la recherche, la réalisation et l'aboutissement de ce stage.

Plus particulièrement, je souhaite en premier lieu remercier **Martial Defoort**, mon tuteur, pour m'avoir guidé et épaulé durant ce stage mais également pour toutes les connaissances qu'il a pu me transmettre. Je remercie aussi **Viviana Giordano** pour m'avoir aidé dans les démarches administratives.

Également grand merci à **Skandar Basrour** pour son conseil et sa bonne humeur tout au long de ce stage.

Un grand merci à mon collègue stagiaire, **Léo Matokwong** de Phelma avec lequel j'ai eu le plaisir de travailler en équipe pendant ces quatre mois.

Et enfin merci à **Sylvain Toru** et plus généralement à Polytech Grenoble pour m'avoir donnée l'opportunité de réaliser un stage aussi enrichissant et motivant que celui-ci.

# Table des matières

I. Remerciements.....	3
II. Glossaire.....	7
III. Introduction et présentation.....	9
1. Introduction.....	9
2. Présentation de la structure d'accueil.....	9
3. Présentation du sujet.....	11
4. Outils.....	12
IV. Travaux réalisés.....	13
1. Description de la carte d'acquisition.....	13
1.1 Produits.....	13
1.2 Caractéristiques Techniques.....	14
1.3 Implémentation d'une application avec une interface web.....	16
1.4 Frontend.....	18
1.5 Backend.....	19
1.6 Application web.....	20
2. Génération et acquisition de signaux.....	23
2.1 Conversion numérique/analogique.....	23
2.2 Génération d'un signal multi-tons avec un seul oscillateur.....	25
2.3 Acquisition de signaux.....	28
2.3.a La fréquence d'échantillonnage.....	29
2.3.b Le niveau de déclenchement du trigger.....	30
2.3.c Le délai de déclenchement du trigger.....	30
2.3.d Définir un canal de référence.....	30
2.4 Caractérisation du temps d'exécution du programme entre deux acquisitions.....	31
3. Implémentation d'une fonctionnalité de démodulation.....	33
3.1 Calcul d'une transformée de Fourier rapide.....	33
3.2 Le Fenêtrage.....	36
3.3 Filtre numérique.....	39
3.3.a Filtre à réponse impulsionnelle finie.....	39
3.3.b Filtre à réponse impulsionnelle infinie.....	40
3.3.c Définir les coefficients du filtre IIR.....	40
3.3.d Fonction de transfert analogique.....	41
3.4 La démodulation par détection synchrone.....	43
3.5 Filtre numérique à réponse impulsionnelle infinie.....	46
3.6 Test de l'application.....	46
4. Test du filtre avec la démodulation.....	47
5. Test sur dispositif.....	49
V. Bilan.....	54
1. Perspective.....	54
2. Compétences professionnelles.....	55
3. Compétences transversales et diagramme de Gantt.....	55
4. Conclusion.....	57
VI. Annexe.....	58
1. Concepts de base.....	58
1.1 Classification des signaux.....	58
1.2 Traitement des signaux.....	59
1.3 Numérisation des signaux.....	59

1.4 Échantillonnage et théorème.....	59
1.5 La quantification.....	60
1.6 Transformée de Fourier.....	60

## Index des figures

Figure 1 : Instrument de mesure HF2LI de Zurich Instrument.....	11
Figure 2 : Red Pitaya STEMlab 125-14.....	11
Figure 3 : Caractéristiques techniques de la Red Pitaya STEMlab 125-14.....	14
Figure 4 : Positionnement des cavaliers sur le Red Pitaya STEMlab 125-14.....	14
Figure 5 : Schéma fonctionnel de la Red Pitaya STEMlab 125-14.....	15
Figure 6 : Intégration du terminal de la Red Pitaya via un protocole SSH, dans le terminal d'un ordinateur utilisant le système d'exploitation Ubuntu 22.04.....	16
Figure 7 : Interaction entre la partie Frontend et Backend d'une application web Red Pitaya.....	17
Figure 8 : Architecture 3 tiers d'une application web.....	17
Figure 9 : Fonctionnement de la partie Frontend d'une application web.....	18
Figure 10 : Diagramme décrivant le rôle du contrôleur.....	19
Figure 11 : Capture du menu de l'interface web de la Red Pitaya.....	20
Figure 12 : Capture de l'interface web de l'application.....	21
Figure 13 : Description du fonctionnement de l'application par un diagramme bloc.....	22
Figure 14 : Schéma de câblage de la Red Pitaya et d'un oscilloscope.....	24
Figure 15 : Visualisation sur un oscilloscope MSOX2024A du signal sinusoïdale généré sur la voie 1 de la Red Pitaya STEMlab 125-14 en temporelle à gauche et en fréquentielle à droite.....	24
Figure 16 : Mauvaise génération d'un signal périodique avec un ton à 12 kHz et un ton à 13,1 kHz .....	26
Figure 17 : Génération d'un signal périodique avec un ton à 10 kHz et un ton à 100 kHz.....	26
Figure 18 : Signaux sinusoïdaux de fréquence proche.....	27
Figure 19 : Génération d'un signal périodique constitué de deux fréquences à 12 kHz et 13,1 kHz.	28
Figure 20 : Écriture de chaque échantillon dans le buffer à partir de la position n.....	29
Figure 21 : Procédure normale d'acquisition.....	31
Figure 22 : Mesure du temps d'exécution avec 1 campagne de 1000 acquisitions.....	32
Figure 23 : Mesure du temps d'exécution avec 1000 campagnes de 1 acquisition.....	32
Figure 24 : Distribution du temps d'exécution pour 1000 campagnes de 1 acquisition.....	32
Figure 25 : Distribution du temps d'exécution pour 1 campagne de 1000 acquisitions.....	32
Figure 26 : Algorithme de Cooley-Tukey Radix-2 DIT.....	34
Figure 27 : Signal arbitraire composé de la somme d'une fréquence à 12 kHz et d'une fréquence à 13,1 kHz.....	35
Figure 28 : FFT du signal arbitraire composé de la somme d'une fréquence à 12 kHz et d'une fréquence à 13,1 kHz.....	36
Figure 29 : Peigne de Dirac.....	36
Figure 30 : Sinus cardinal (tracé temporel en bleu, tracé discret en rouge).....	37
Figure 31 : FFT du signal tronqué.....	37
Figure 32 : Fenêtre de Blackman.....	38
Figure 33 : FFT de la fenêtre de Blackman.....	38
Figure 34 : FFT du signal tronqué multiplié par une fenêtre de Blackman.....	38
Figure 35 : Amplitudes du signal $S_1$ démodulé avec une fréquence de coupure à 5 kHz et à 1 kHz.	45
Figure 36 : Module de la Transformée de Fourier des amplitudes calculées avec un filtre de fréquence de coupure 5 kHz (courbe bleue) et 1 kHz (courbe orange).....	45

Figure 37 : Amplitudes du signal $S_2$ démodulé avec une fréquence de coupure à 5 kHz (courbe bleue) et à 1 kHz (courbe orange).....	46
Figure 38 : Module de la Transformée de Fourier des amplitudes calculées avec un filtre de fréquence de coupure 5 kHz et 1 kHz.....	47
Figure 39 : Vérification de l'ordre du filtre par démodulation.....	48
Figure 40 : Photo du dispositif de test.....	49
Figure 41 : Amplitude de déplacement du dispositif calculé par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz.....	50
Figure 42 : Déphasage du dispositif avec la référence calculée par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz.....	50
Figure 43 : Diagramme du processus de calcul de l'amplitude et du déphasage du MEMS pour une fréquence du balayage.....	51
Figure 44 : Amplitude de déplacement du dispositif calculé par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz après application du filtre moyenneur.....	51
Figure 45 : Phase de déplacement du dispositif calculé par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz.....	52
Figure 46 : Amplitude de déplacement du dispositif calculé par la STEMlab 125-14 et la HF2LI sur une plage de fréquence de 255 kHz à 265 kHz.....	52
Figure 47 : Phase de déplacement du dispositif calculé par la STEMlab 125-14 et la HF2LI sur une plage de fréquence de 255 kHz à 265 kHz.....	52
Figure 48 : Diagramme de Gantt.....	56
Figure 49 : Exemple de signal discret numérique.....	58

## Index des tableaux

Tableau 1 : Caractéristique du convertisseur numérique analogique de la Red Pitaya STEMlab 125-14.....	25
Tableau 2 : Caractéristique du convertisseur analogique de la Red Pitaya STEMlab 125-14.....	28
Tableau 3 : Pseudo-code du calcul des pôles du filtre de Butterworth.....	42
Tableau 4 : Code démonstratif de l'implémentation de la démodulation.....	44
Tableau 5: Répartition des tâches effectuées par jours.....	56

## II. Glossaire

**API bas niveau** : C'est une interface de programmation d'application qui permet l'interaction directe avec le matériel d'un système informatique, tels que les périphériques et les registres de processeur. Ces API fournissent des fonctions et des méthodes pour lire et écrire directement dans les registres matériels, configurer les périphériques, et gérer les interruptions et autres opérations spécifiques au matériel. Elles sont essentielles pour le développement de pilotes de périphériques, les systèmes embarqués, et d'autres applications nécessitant un contrôle fin et direct du matériel.

**Backend** : Étape de production d'une puce électronique dans laquelle on va découper toutes les puces d'un wafer pour les encapsuler dans des boîtiers. Dans le développement web, la notion de Backend se réfère à la partie invisible de l'utilisateur qui va permettre le bon fonctionnement d'un site internet et qui échange avec la mémoire du dispositif.

**Compilateur** : Un compilateur est un programme informatique qui traduit un langage, le langage source, en un autre, appelé le langage cible (ou langage objet), en préservant la signification du texte source. Dans notre étude le compilateur convertira le code C++ et C en code assembleur directement lisible par le processeur.

**Debugger** : Un débugueur (de l'anglais debugger) est un logiciel qui aide un développeur à analyser les bugs (erreurs) d'un programme. Il est possible avec son utilisation d'exécuter le code instruction par instruction pour déterminer d'où provient un bug.

**FIR** (Finite Impulse Response) : Filtre à réponse impulsionnelle finie

**FPGA** (Field-programmable gate array) : est un circuit intégré fait pour être (re)programmé par l'utilisateur après sa fabrication en utilisant un langage informatique spécifique, donc sans modifier le matériel.

**Frontend** : Etape de production d'une puce électronique dans laquelle on va travailler une plaquette de silicium pour y introduire des transistors par différentes techniques.

**Hardware** : désigne les parties physiques d'un ordinateur et des dispositifs connexes (portes logiques, accumulateurs...).

**Harmonique** : Un harmonique est une composante d'un son périodique, dont la fréquence est un multiple entier d'une fréquence fondamentale. Si on appelle «  $f$  » la fréquence fondamentale, les tons ont des fréquences égales à :  $f, 2f, 3f, 4f\dots$

**IDE** (Integrated développement environnement) : L'IDE est un outil pour un programmeur pour consolider les différents aspects d'écriture d'un programme. L'IDE est un outil qui intègre un éditeur, la chaîne de production de code, et le debugger.

**IIR** (Infinite Impulse Response) : Filtre à réponse impulsionnelle infinie

**MEMS** : Les MEMS, acronyme de Micro Electro Mechanical Systems, sont des dispositifs miniaturisés combinant plusieurs principes physiques. Ils intègrent généralement des éléments mécaniques couplés à de l'électronique et sont réalisés par des procédés de fabrication issus de la microélectronique.

**Microcontrôleur** : Un microcontrôleur (noté µc, uc ou encore MCU en anglais) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties.

**Open Source** : Un logiciel Open Source est un code conçu pour être accessible au public. N'importe qui peut accéder au code, le modifier et le redistribuer à sa convenance.

**Systèmes embarqués** : assemblage de logiciels et de matériel informatique conçus pour une fonction particulière. Il peut également fonctionner au sein d'un système plus large. Ce dernier peut disposer d'une fonctionnalité fixe tout en offrant la possibilité d'être programmable. Ces systèmes doivent généralement respecter des contraintes de basse consommation et des temps d'exécution courts.

**Timer** : type d'horloge utilisé pour mesurer des intervalles de temps.

## **III. Introduction et présentation**

### **1. Introduction**

Le contenu de ce rapport de stage a pour objectif de présenter les travaux réalisés pendant les 17 semaines de mon stage de quatrième année d'études du cycle d'ingénieur à Polytech Grenoble. J'ai intégré le laboratoire TIMA et son équipe CDSI (Circuit, Devices and System Integration). J'ai effectué ce stage du 15/04/2024 au 30/08/2024 exclusivement en présentiel.

Ce rapport sera partagé en trois grandes parties, la première concernera la présentation du laboratoire et du sujet de mon stage. Une deuxième partie traitera l'aspect plus technique de mon projet qui se divisera en plusieurs sous parties.

La première sous-partie exposera le fonctionnement de cette carte et de sa première utilisation, tout en soulignant les caractéristiques qu'elle offre. Une seconde sous-partie expliquera le fonctionnement du générateur de tension et du convertisseur analogique numérique interne de la carte d'acquisition, et les améliorations qui lui ont été apportées. Dans une troisième partie j'expliquerai, les caractéristiques développées pour effectuer une détection synchrone, illustrées de plusieurs développements mathématiques. Dans une dernière sous-partie, nous découvrirons le test de la carte sur un dispositif MEMS.

Enfin une dernière partie conclura le rapport et me permettra de faire mon retour personnel sur le stage et sur les compétences que j'ai pu développer. De plus, nous y aborderons la méthodologie et la gestion de projet.

Ce rapport est suivi d'un rapport plus technique destiné au laboratoire, afin d'approfondir certains détails plus techniques. Dans ce rapport technique une annexe présentera la totalité du code créé lors de ce projet.

### **2. Présentation de la structure d'accueil**

Le laboratoire TIMA est un organisme de recherche public du CNRS (Centre national de la recherche scientifique), de Grenoble-INP et de l'UGA (Université Grenoble Alpes). Son nom est l'abréviation de "Techniques de l'Informatique et de la Microélectronique pour l'Architecture des systèmes intégrés". Les équipes de TIMA regroupent des membres et des stagiaires du monde entier. Une grande partie de la recherche est menée dans le cadre de projets de coopération avec des partenaires industriels et universitaires, soutenus par des subventions régionales, nationales et européennes.

Le laboratoire TIMA se concentre sur une gamme étendue de domaines de recherche répartie en plusieurs équipes, allant de la spécification à la conception, en passant par la vérification, les tests, les outils de conception assistée par ordinateur (CAO) et les méthodes de conception pour les systèmes intégrés. Ces domaines englobent divers aspects, depuis les composants analogiques et numériques jusqu'aux systèmes multiprocesseurs sur puce, avec leur système d'exploitation de base.

Les activités et thèmes de recherche de TIMA incluent :

1. Robustesse, fiabilité et essais : Cette recherche vise à améliorer la durabilité et la fiabilité des systèmes intégrés tout en développant des méthodes de test efficaces.
2. Co-conception matériel/logiciel : TIMA explore la collaboration entre la conception matérielle et logicielle pour optimiser les performances et l'efficacité des systèmes intégrés.
3. Simulation et vérification : Le laboratoire développe des outils et des méthodes pour simuler et vérifier la fonctionnalité des systèmes intégrés, garantissant ainsi leur bon fonctionnement.
4. Conception de faible puissance : TIMA se penche sur les techniques visant à réduire la consommation d'énergie des systèmes intégrés, contribuant ainsi à une meilleure efficacité énergétique.
5. Sécurité du matériel et confiance intégrée : Cette recherche vise à renforcer la sécurité des systèmes intégrés contre les cybermenaces et à garantir la confiance dans leur fonctionnement.
6. Conception asynchrone : TIMA explore les avantages de la conception asynchrone dans les systèmes intégrés pour améliorer la performance et la flexibilité.
7. Nouvelles techniques d'échantillonnage et de traitement des données : Le laboratoire développe des méthodes innovantes pour l'acquisition et le traitement efficaces des données dans les systèmes intégrés.
8. MEMS, capteurs intelligents et actionneurs : L'équipe qui m'a engagée se concentre sur la conception et l'optimisation des microsystèmes électromécaniques, des capteurs intelligents et des actionneurs pour diverses applications.
9. Conception des dispositifs, circuits et systèmes AMS/RF/mmW : Une équipe du laboratoire ce concentre sur la conception avancée de dispositifs, circuits et systèmes analogiques, radiofréquences et millimétriques.
10. Modélisation, commande et étalonnage des dispositifs, circuits et systèmes AMS/RF : TIMA développe des techniques de modélisation, de commande et d'étalonnage pour améliorer les performances et la précision des systèmes analogiques et RF.

### 3. Présentation du sujet

L'objectif de ce stage était de piloter une carte d'acquisition Red Pitaya et d'optimiser son utilisation afin de réaliser une détection synchrone, dans le but de mesurer les performances d'un capteur MEMS et de caractériser précisément ses figures de mérite. La réalisation de ces fonctionnalités permettra d'utiliser le système comme un outil de démonstration facilement transportable et manipulable.

Actuellement, l'équipe utilise un instrument de mesure prêt à l'emploi, HF2LI de Zurich Instrument, afin de réaliser une détection synchrone (voir la figure 1 ci-dessous).



Figure 1 : Instrument de mesure HF2LI de Zurich Instrument

C'est l'instrument de choix est idéal pour la caractérisation de dispositifs MEMS tels que des gyroscopes. Malgré sa haute performance, cet HF2LI reste surdimensionné pour l'application visée, et son coût élevé (aux alentours de 15 000€) suggère de trouver d'autres alternatives, notamment en utilisant des cartes d'acquisitions autonomes.

Ainsi, la plateforme Red Pitaya propose une bonne alternative à cet instrument de mesure, d'une part en proposant des prix plus avantageux (aux alentours de 1 000 € pour la carte sélectionnée dans le cadre de ce stage), et d'autre part en proposant des articles clé en main et reproductible, avec un environnement open-source développé spécialement pour la recherche. Un article proposé par la plateforme à retenue l'attention de mon équipe puisqu'elle connaît actuellement un grand succès dans la recherche, la Red Pitaya STEMlab 125-14 (voir figure 2 ci-dessous).



Figure 2 : Red Pitaya STEMlab 125-14

Étant donné que l'équipe actuelle n'est pas spécialisée dans le domaine des systèmes embarqués et que cette carte d'acquisition leur est complètement inconnue, mon stage leur offrira la possibilité d'obtenir un document sur lequel ils pourront se baser pour s'initier à ce nouveau système.

## 4. Outils

Pour le déroulement de mon stage j'ai utilisé plusieurs outils pour mener à bien mon projet :

- **Ubuntu Linux** : Système d'exploitation GNU/Linux basé sur la distribution Debian. Il est libre, gratuit, et simple d'utilisation. La version 22.04 a été utilisée pour ce projet.
- **Visual Studio Code** : Éditeur de code gratuit et extensible développé par Microsoft pour Windows, Linux et MacOS. Ces fonctionnalités incluent la prise en charge de plusieurs terminaux et du débogage, la mise en évidence de la syntaxe pour tout type de langage, la complétion intelligente de code, les snippets (code réutilisable incorporable dans des modules plus larges), la refactorisation de code (améliorer le code source d'une application informatique en le modifiant sans ajouter de nouvelles fonctionnalités) et l'intégration de Git pour les plateformes Github et Gitlab.
- **Python** : Langage de programmation interprété, multi-paradigme et multiplate-formes. Lors de ce projet sa principale utilisation fut d'afficher des courbes avec l'utilisation des modules 'SciPy' et 'NumPy' pour le traitement des signaux, 'PyQt5' qui simplifie la création et la gestion d'une interface utilisateur et 'Matplotlib' destinée à tracer et visualiser des données sous forme de graphiques.
- **GNU compiler** : GNU Compiler Collection, abrégé en GCC, est un ensemble de compilateurs créés par le projet GNU. GCC est un logiciel libre capable de compiler divers langages de programmation, dont C, C++, Objective-C, Java, Ada, Fortran et Go.
- **Git** : logiciel de gestion de version qui m'a permis de partager et d'entreposer mes codes. J'ai modifié plusieurs fois les codes pour les rendre plus facilement compréhensibles et intégrables.
- **Google Scholar** : moteur de recherche d'articles scientifiques sur lesquels j'ai réalisé toutes mes recherches pour l'état de l'art de mon sujet.
- **Firefox** : Navigateur web libre et gratuit disponible pour PC et mobiles, développé et distribué par "Mozilla Foundation" depuis 2003. Lors de ce stage, ce moteur de recherche fut utilisé pour afficher l'interface web de la Red Pitaya.
- **Libre Office** : LibreOffice est une suite bureautique libre et gratuite et est un logiciel libre. Utilisé pour rédiger ce rapport.

## **IV. Travaux réalisés**

### **1. Description de la carte d'acquisition**

Afin de concevoir les fonctionnalités demandées sur la carte d'acquisition Red Pitaya STEMlab 125-14, une phase préliminaire de prise en main du système fut cruciale pour connaître les performances de la carte et les fonctionnalités proposées.

#### **1.1     Produits**

La carte d'acquisition Red Pitaya STEMlab 125-14 fait partie d'une série de plusieurs cartes d'acquisition développées par la fondation "Red Pitaya" à Ljubljana en Slovénie. La mission de Red Pitaya est de rendre les outils de laboratoire professionnels accessibles à un large public, y compris les ingénieurs, les chercheurs, les universitaires et les amateurs de technologie. Ils visent à démocratiser l'accès à l'instrumentation électronique et aux solutions de mesure grâce à des dispositifs compacts, abordables et polyvalents, dans les domaines de l'électronique, l'acquisition de données analogique pour le traitement du signal et le traitement d'information via un serveur web intégré. Le produit phare de Red Pitaya est la série STEMlab, qui inclut des cartes d'acquisitions de données et de traitement du signal STEMlab 125-14 utilisées lors de ce stage.

Red Pitaya soutient une communauté active d'utilisateurs et de développeurs. L'entreprise encourage l'innovation collaborative et la diffusion des connaissances à travers des forums en ligne, des tutoriels, des projets open-source et des webinaires. Le code source est disponible sur le service en ligne Github au lien suivant : <https://github.com/RedPitaya/RedPitaya>. De plus une API bas niveau, pré-compilé sur le système d'exploitation propose des fonctionnalités simplifiant la manipulation des registres et périphériques disponibles pour toutes les versions de cartes d'acquisition proposées par la plateforme.

## 1.2 Caractéristiques Techniques

Le noyau de la carte est régi par la coalition entre un processeur ARM Cortex-A9 double cœur et d'un FPGA programmable intégré dans un même composant SoC Xilinx 7010. Le FPGA permet une personnalisation et une gestion efficace des tâches de traitement du signal temps réel (voir figure 3).

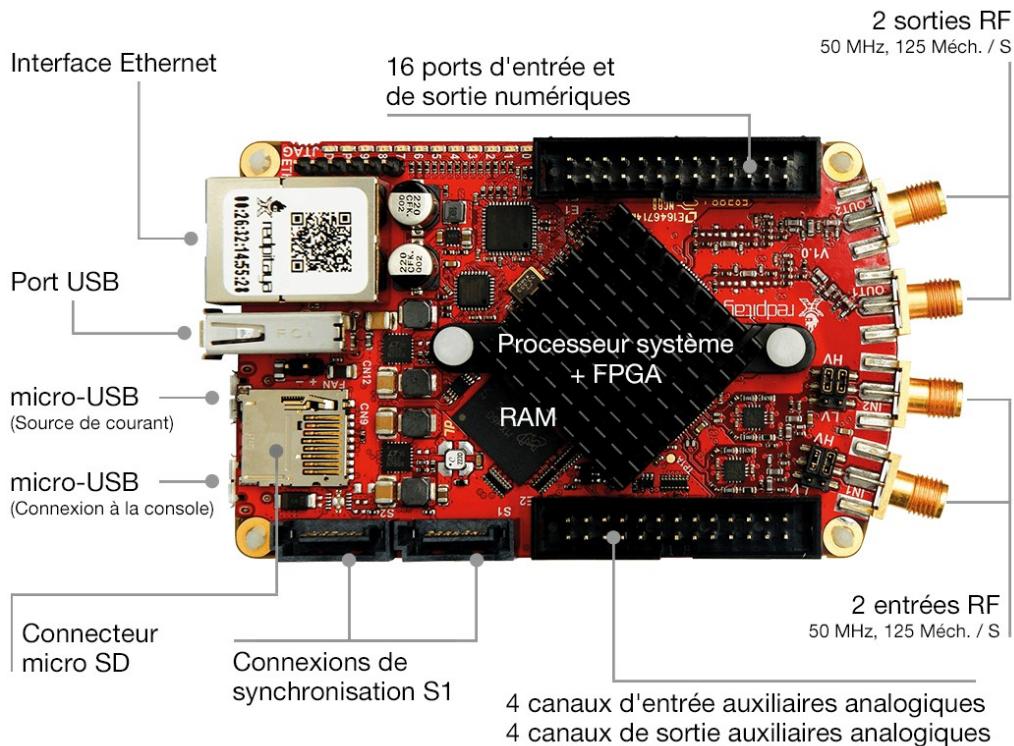


Figure 3 : Caractéristiques techniques de la Red Pitaya STEMlab 125-14

Deux canaux entrés et deux canaux de sortie analogique haute fréquence, avec une fréquence d'échantillonnage maximale de 125 MS/s (méga-échantillons par seconde) et une résolution de 14 bits permettant d'acquérir et de générer des signaux continues avec une haute précision. De plus la gamme de tension des signaux d'entrées est configurable à  $\pm 1$  V ou  $\pm 20$  V par l'utilisation de jumpers (cavaliers) déplaçable sur la carte (voir figure 4).

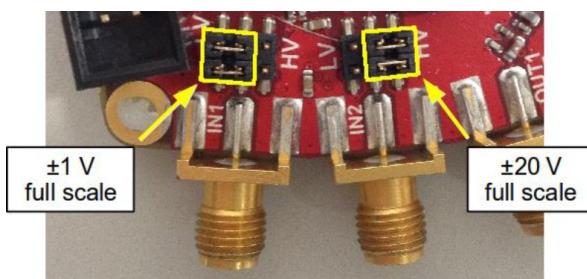


Figure 4 : Positionnement des cavaliers sur le Red Pitaya STEMlab 125-14

La carte propose 16 broches GPIO (General Purpose Input/Output) numérique configurable en entré et sortie et quatre canaux d'entrées et quatre canaux de sorties analogiques, ce qui permet la communication avec d'autre composant via des bus de communications I2C, SPI et CAN RS232 et d'effectuer des mesures analogiques qui ne nécessitent pas d'avoir une grande précision. Deux connecteurs SATA permettent de communiquer avec des supports de mémoire externe. Un port USB 2.0 permet de communiquer avec un autre système et ou lire les données d'une clé. Enfin un module Wifi et un port Ethernet d'une vitesse de 1 Gigabit par seconde permet d'établir une connexion réseau rapide. La figure 5 illustre bien les interconnexions entre les principaux composants de la carte.

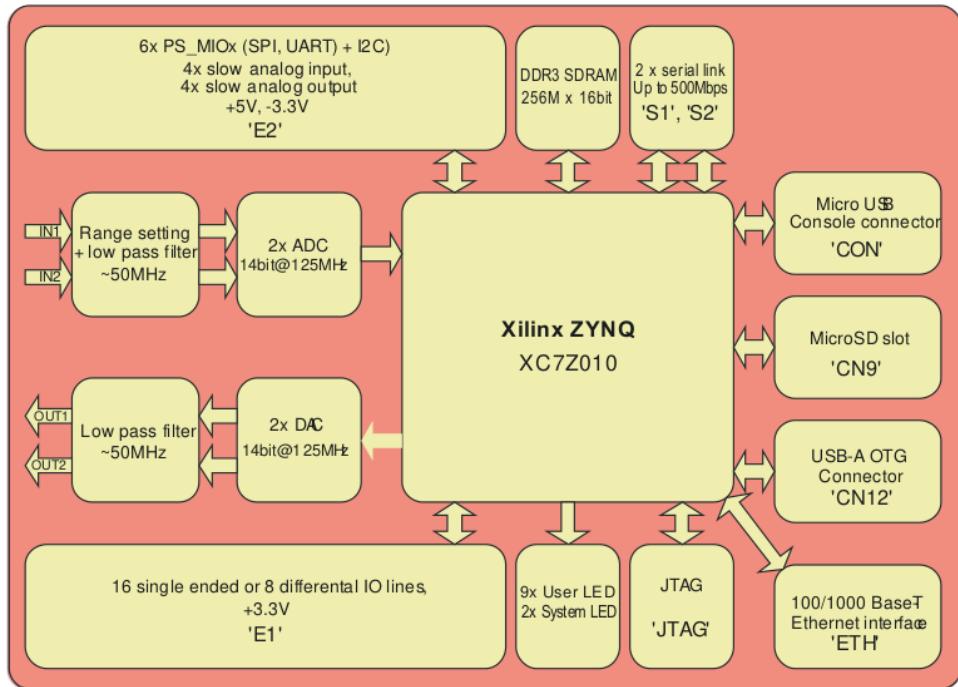


Figure 5 : Schéma fonctionnel de la Red Pitaya STEMlab 125-14

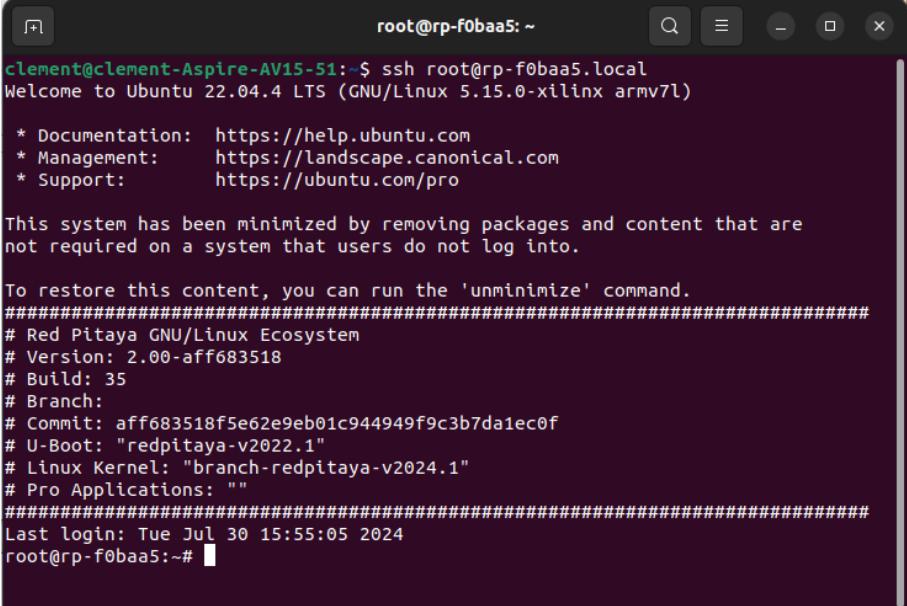
Le fonctionnement de la carte est assuré par un système d'exploitation Linux de la distribution Ubuntu version 18.0, installé sur une carte SD. L'utilisation de Linux permet de supporter divers environnements de développement et langage de programmation comme le C/C++, Python, Matlab, et bien plus, facilitant le développement d'application sur mesure. Ainsi l'OS permet la gestion d'un serveur NGINX (<https://nginx.org/en/>) ce qui contribue à la mise en œuvre d'une interface web utilisatrice intuitive pour l'accès à des applications d'instrumentations virtuels comme un oscilloscope, un générateur de signaux, et un analyseur de spectre. En utilisant un tel système d'exploitation, le port Ethernet et le module Wifi de la carte sont automatiquement gérés, ce qui permet de la connecter à un réseau pour la contrôler en local ou à distance, à partir d'un ordinateur ou d'un autre système.

### 1.3 Implémentation d'une application avec une interface web

Au début du stage, j'avais entrepris de développer une application avec une interface web hébergée sur la Red Pitaya. Comme expliqué précédemment, la Red Pitaya offre un serveur web intégré permettant la gestion d'une interface utilisateur web ainsi qu'un port Ethernet pour ce connecter sur un autre système, ou un réseau internet. Comme la plateforme est open-source, il est possible de développer des applications en ayant accès au code source des librairies de l'API Red Pitaya. Cela permet de concevoir des applications complètes sans avoir à se soucier de la gestion des périphériques de la carte.

Tout d'abord, j'ai choisi de connecter la carte d'acquisition via un câble Ethernet sur mon ordinateur, pour établir un réseau local, ce qui permet de profiter de la rapidité de l'Ethernet.

Afin d'envoyer des fichiers et des commandes à la Red Pitaya, la connexion Ethernet propose d'établir une connexion SSH entre l'ordinateur et la carte d'acquisition afin d'intégrer le terminal de celle-ci dans le terminal de l'ordinateur, voir la capture d'écran suivante 6.



```
clement@clement-Aspire-AV15-51:~$ ssh root@rp-f0baa5.local
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-xilinx armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

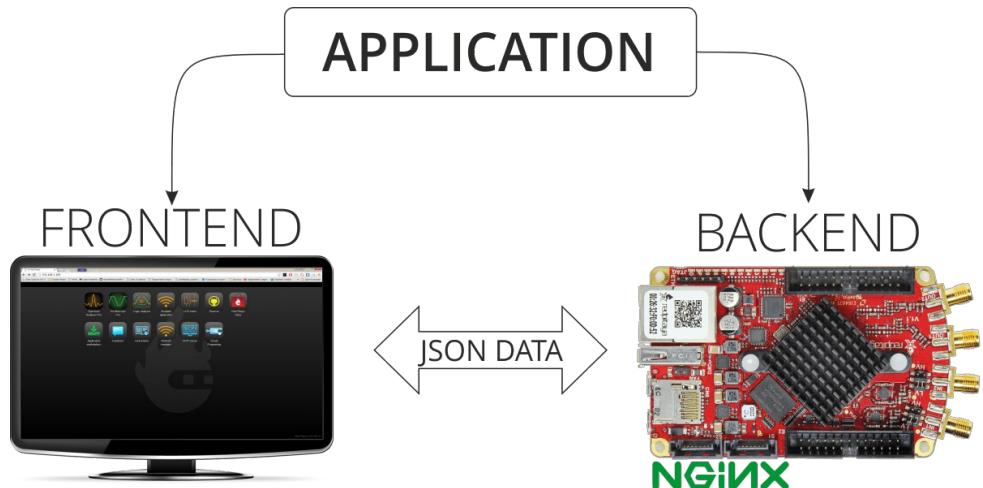
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
#####
# Red Pitaya GNU/Linux Ecosystem
# Version: 2.00-aff683518
# Build: 35
# Branch:
# Commit: aff683518f5e62e9eb01c944949f9c3b7da1ec0f
# U-Boot: "redpitaya-v2022.1"
# Linux Kernel: "branch-redpitaya-v2024.1"
# Pro Applications: ""
#####
Last login: Tue Jul 30 15:55:05 2024
root@rp-f0baa5:~#
```

Figure 6 : Intégration du terminal de la Red Pitaya via un protocole SSH, dans le terminal d'un ordinateur utilisant le système d'exploitation Ubuntu 22.04

L'application web développée, se décompose en deux parties distinctes. La première partie invisible de l'utilisateur se nomme la partie "Backend", compilable à même la carte. J'ai choisi de programmer cette partie en langage informatique C et C++ de la distribution GNU. La puissance de ces langages permet de créer des applications légères, fiables et robustes, tout en ayant accès au plus profond de la machine. Grâce au système d'exploitation Linux, ces deux langages permettent de configurer les périphériques de la carte (commander les LED, commander les entrées et sorties analogiques, changer l'image FPGA, etc.), via les fonctions C pré-compilé de l'API.

La seconde partie nommée "Frontend", est transmise par le serveur au navigateur internet de l'utilisateur pour afficher et gérer la partie visible de l'application. Comme imaginé sur la figure ci-dessous, le "Frontend" et le "Backend" nécessite de communiquer entre eux pour fonctionner l'un dans l'autre. Cela se fait principalement avec les API du réseau Red Pitaya qui sont techniquement basées sur une connexion Web étendue. Cela permet de créer une demande au serveur, sans ce soucier des difficultés de communication et de transfert de données. Les API de réseau prennent soin du transfert de données.



*Figure 7 : Interaction entre la partie Frontend et Backend d'une application web Red Pitaya*

Ce système fonctionne sur une architecture 3 tiers. Le premier tiers est l'ordinateur qui navigue sur le web, c'est-à-dire le client. Le deuxième tiers, le serveur, va réceptionner les demandes de pages web du client afin de lui renvoyer le résultat de sa recherche. Le dernier tiers, le serveur de base des données, va contenir toutes les données exploitées par le serveur web.



*Figure 8 : Architecture 3 tiers d'une application web*

Cette architecture est le modèle fondamental qui explique la manière dont fonctionne le web. Cela permet de séparer les différents acteurs du web mais surtout de pouvoir suivre à la trace tout le processus entre le moment où l'utilisateur demande une page web et le moment où il la reçoit dans son navigateur. Généralement, le second tiers et troisième tiers sont séparés matériellement chez les fournisseurs de services web. Pour un système embarqué contraint en taille et en consommation et limité par un seul processeur multi-cœurs, un logiciel installé manège ces deux processus. Sur la Red Pitaya le serveur logiciel utilisé est NGINX.

Lorsque le client demande une ressource au serveur, une requête HTTP est envoyée au serveur. C'est à ce moment-là que la deuxième couche de l'architecture 3 tiers devient intéressante, à savoir le serveur web (inclus dans le service NGINX).

Lorsque le serveur reçoit la requête HTTP, il va d'abord la lire et récupérer les informations qui l'intéresse, ensuite il effectue une action et exécute des scripts en fonction des ressources demandées par le client. Une fois cette action terminée, le serveur construit une réponse HTTP. Le corps de la réponse contiendra le code de statut, la réponse des scripts exécutés et les actions résultantes de l'action effectuée. Ce script répond à une URL bien spécifique.

Pour construire une page web, le concepteur est plus ou moins forcée d'utiliser certains langages. Pour le premier tiers, le langage HTML est utilisé pour la structure et le contenu, et le langage CSS est utilisé pour la mise en forme. Pour les deux autres tiers le langage JavaScript est utilisé pour définir le comportement de la page web.

Puisque que le serveur web de la carte fonctionne sous Ubuntu, les pages HTML peuvent directement être implémentées dans le répertoire du serveur web de la Red Pitaya (/opt/redpitaya/www/apps). Puisqu'il est assez malin, juste en lisant les URL, il peut déterminer la position des sources de la page web sur la mémoire interne de la carte.

Ces technologies standardisées, sont les seuls qui garantissent le bon fonctionnement de la page web avec les navigateurs (Firefox, Chromium, Edge,...). Mais en Backend absolument n'importe quel langage de programmation peut faire l'affaire, vu que la seule chose à effectuer est de déclencher des scripts sur commande. Seulement la compatibilité entre le serveur web et le langage utilisé est important. Le serveur NGINX est compatible avec les langages C et C++.

Grâce à Nginx chaque application est interprétée comme des modules, ce qui permet de les modifier, d'en ajouter ou d'en supprimer, sans avoir à redémarrer le système d'exploitation.

## 1.4 Frontend

Le Frontend est la partie visuelle de l'application dans un navigateur web. Elle utilise la technologie HTML5 pour la mise en page, CSS3 pour définir le style graphique des éléments et JavaScript pour définir le comportement de l'application web. L'image 9 ci-dessous montre le flux d'informations échangés dans l'interface utilisateur web.

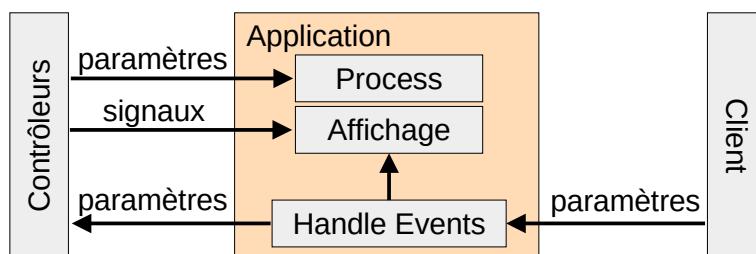


Figure 9 : Fonctionnement de la partie Frontend d'une application web

L'utilisateur (à droite) modifie certains paramètres de l'application dans l'interface utilisateur. Ceci active des fonctions événementielles (*handler*) qui notifient les modifications de l'environnement du navigateur. L'interface utilisateur peut directement traiter ces évènements et modifier l'état de l'écran, ou bien, envoyer ces évènements sous forme de paramètres au contrôleur, afin d'effectuer des calculs spécifiques sur l'appareil et changer l'état du dispositif.

Le contrôleur (la partie Backend) applique les paramètres à ordre interne sur le matériel et effectue des calculs. Suivant les algorithmes, le contrôleur peut envoyer les nouveaux signaux et changer certains paramètres de l'interface web pour les appliquer à l'écran. Les échanges entre ces deux processus s'effectue avec des données au format JSON.

## 1.5 Backend

Le Backend a la main sur le matériel, il nécessite d'un composant logiciel, nommé "contrôleur", qui gère la communication avec le matériel et permet l'exécution des fonctions spécifique de l'appareil. Ces contrôleurs sont souvent implémentés sous forme de bibliothèques partagées. Ils sont réunis dans un fichier reconnaissable par son extension '.so' (*shared object*) pour le système d'exploitation de type Unix/Linux. Un contrôleur permet de contrôler les périphériques matériels, d'acquérir des signaux, de configurer des paramètres, et d'exécuter d'autres opérations matérielles. Les contrôleurs sont aussi accessibles par le serveur web, voir figure 10.

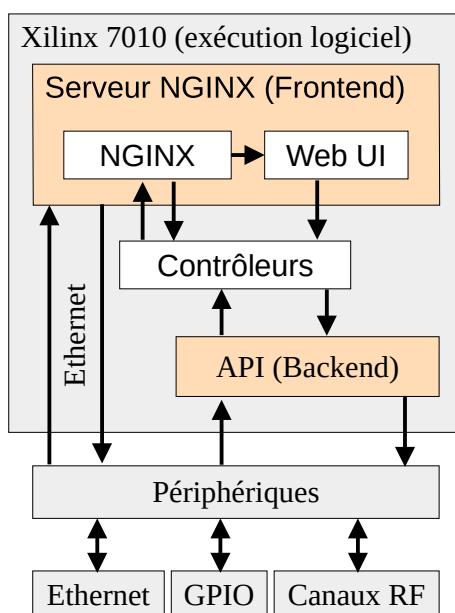


Figure 10 : Diagramme décrivant  
le rôle du contrôleur

Même si la présence d'un tel fichier est obligatoire pour l'exécution d'une application. Les contrôleurs restent invisibles lors du développement de la partie Backend (C/C++) de l'application. Ils sont intégrés à l'application au moment de la compilation de la partie Backend de l'application.

## 1.6 Application web

Afin de développer une application complète, je m'étais fixé comme mission de concevoir un oscillateur numérique avec une interface web temps réel. Le résultat obtenu permettait partir de l'interface web, d'activer la génération de signaux analogiques sur les deux sorties analogiques haute fréquence, dont leur forme, leur amplitude, leur phase et leur offset (tension de décalage) sont configurables. L'interface permet aussi d'activer l'acquisition de signaux en temps sur les deux entrées analogiques haute fréquence. Pour mes expériences, j'ai rebouclé la sortie analogique 1 sur l'entrée 1 et la sortie analogique 2 sur l'entrée 2 avec des câbles coaxiaux. Après acquisition, les signaux sont affichés dans un graphique temporel de l'interface web. Le diagramme figure 13, décrit le fonctionnement de l'application. Celle-ci nécessite d'être lancé par l'utilisateur en cliquant sur une icône disponible sur l'interface web de la Red Pitaya, voir figure 11. L'application web envoie un message à la red pitaya afin de démarrer l'exécution de l'application, d'initialiser ses contrôleurs et d'allouer les ressources nécessaires. Si l'application a démarré avec succès, la partie Backend de l'application envoie les fichiers HTML, CSS et Javascript au navigateur web du client, pour réaliser le rendu graphique de l'interface, voir figure 12.

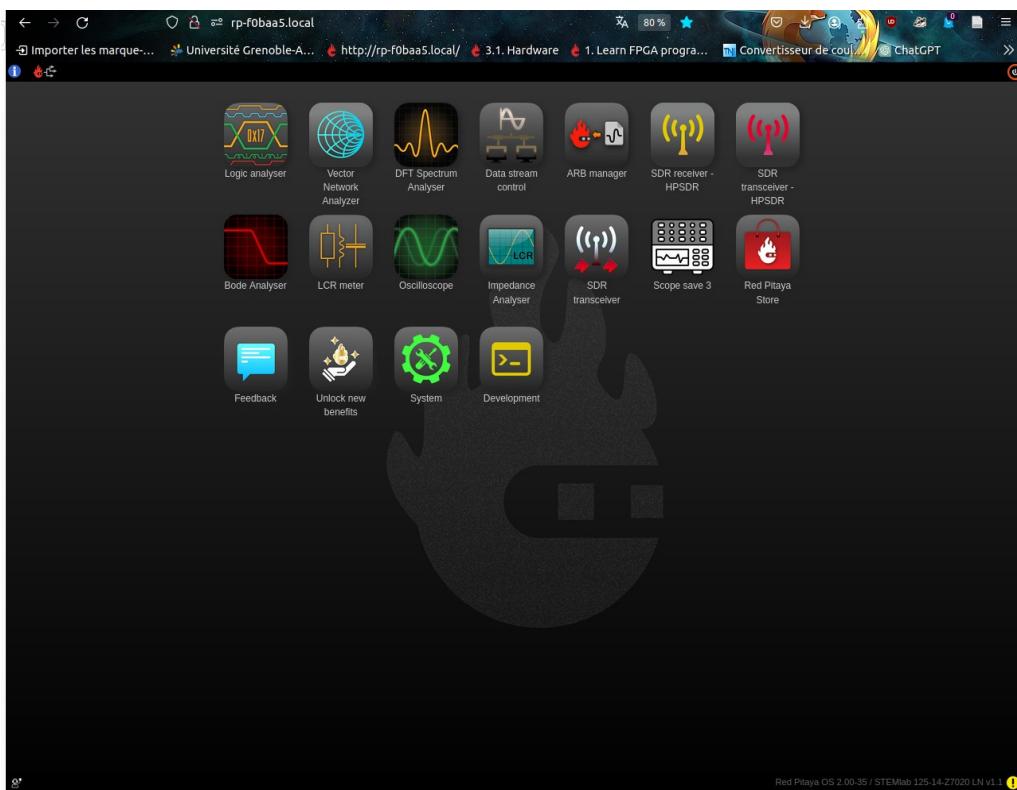


Figure 11 : Capture du menu de l'interface web de la Red Pitaya

Pour le design de la page web, j'ai choisi une structuration, où les fonctionnalités de configuration et l'affichage des courbes sont visibles sur la même page. Comme vous pouvez l'observer sur la droite de la figure 12, un panneau vertical permet de choisir la voie à configurer et

de modifier leurs paramètres spécifiques. La configuration du convertisseur numérique analogique (CNA) et du convertisseur analogique numérique (CAN) peut s'effectuer à n'importe quel moment de l'exécution de l'application. Pour cela, la partie Backend de l'application doit toujours être réceptive à de nouvelle modification. Après la configuration d'un signal de sortie, un calcul arbitraire de la forme du signal généré est effectué, afin de l'observer sur l'interface.

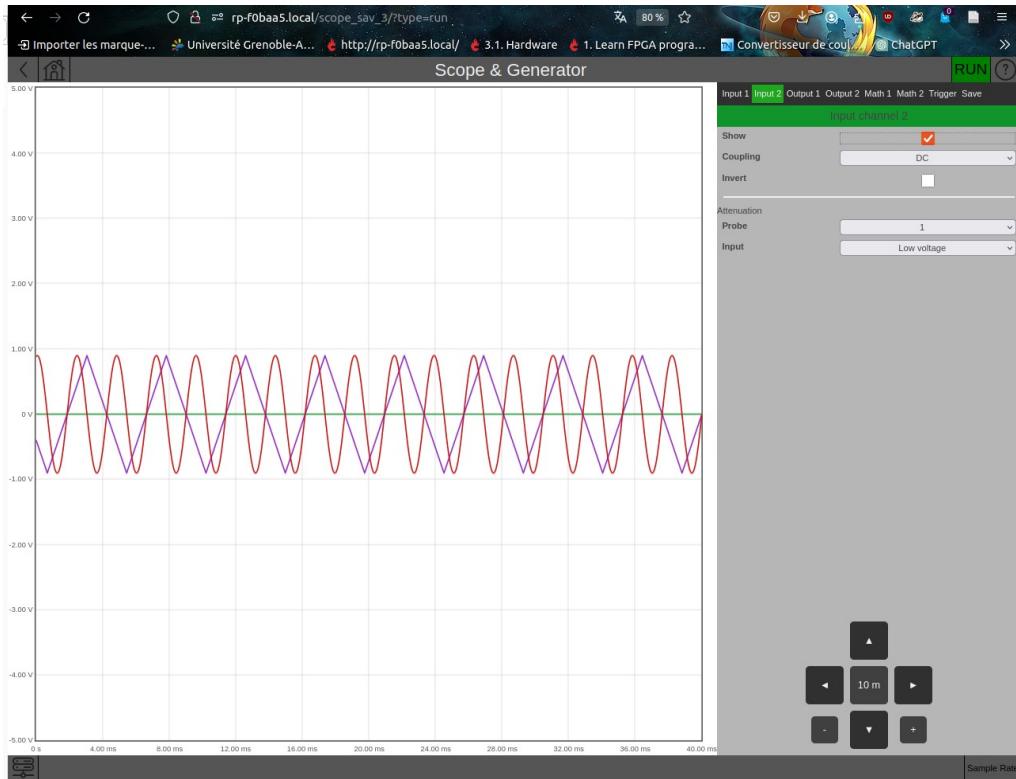


Figure 12 : Capture de l'interface web de l'application

Le développement d'une telle application suscite plusieurs difficultés.

Les deux processus "Backend" à droite et "Font-end" à gauche de la figure 13, présentent une première difficulté. Ces processus requièrent une structure temps réel, en raison du fonctionnement asynchrone du serveur web Nginx. Ce qui implique une conception asynchrone de l'application "Backend" programmée en C++, qui requiert l'utilisation de plusieurs threads (exécution successives de plusieurs parties du programme indépendantes, partageant des ressources logicielles communes). Comme vous pouvez l'observer sur cette même figure, il est important d'attente la fin d'une acquisition, avant lire le contenu du buffer du convertisseur analogique numérique et de l'envoyer au client. Cette pause, provoque une rupture dans l'exécution du programme. Pendant cette pause les signaux logiciels et les paramètres s'entassent dans une pile, attendant la libération du programme. Comme le traitement des signaux et des paramètres est prioritaire sur l'exécution du programme et que les paramètres n'ont pas d'ordre particulier. Ce manque de régularité peut provoquer des problèmes de synchronisation entre le lancement d'une acquisition, et l'attente de la fin de celle-ci, puisque que le temps d'exécution séparant ces deux instructions peut-être plus longues que le temps d'acquisition. Ainsi la génération du signal de fin

d'acquisition par le CAN peut ne pas être détecté et le programme se bloque dans une boucle infinie.

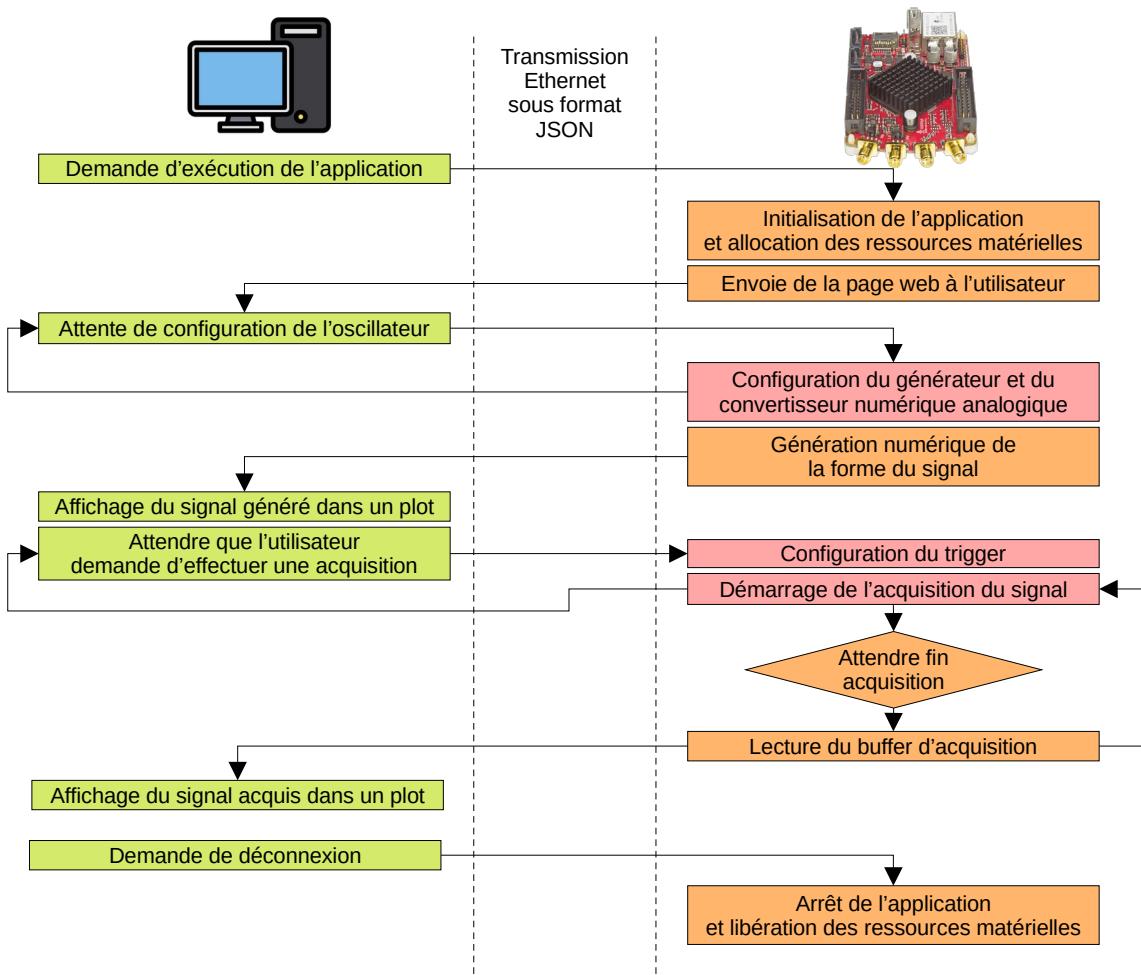


Figure 13 : Description du fonctionnement de l'application par un diagramme bloc

La reproduction du signal généré pose un second problème, car il est difficile de synchroniser la génération réelle du signal avec le signal calculé numériquement.

La dernière difficulté réside dans la variété des langages de programmation utilisés pour programmer une seule application (C, C++, JavaScript, CSS, HTML), ce qui rend difficile son débogage, par la recherche et la correction d'erreurs. Après avoir consacré trop de temps à résoudre des problèmes de communication, j'ai donc pris la décision de créer une application en C++ qui peut être exécutée sur la Red Pitaya en ligne de commande, ne nécessitant plus d'interface graphique. Les signaux acquis seront quant à eux inscrits dans des fichiers textes et seront affichés par des scripts Python indépendamment de l'application (post-processing). Avec cette nouvelle application l'aspect temps réel est perdu, mais ne plus avoir d'interface utilisateur améliore sa rapidité exécution. Dans les prochaines sections de ce rapport, nous aborderons les fonctionnalités développées sur cette application unique.

## 2. Génération et acquisition de signaux

La conception de cette nouvelle application est une opportunité pour gagner en rigueur sur le fonctionnement de l'acquisition et de la génération de signaux analogiques. Les sections suivantes expliqueront comment ces fonctionnalités fonctionnent et quelles problématiques impliquent-elles. Une analyse des fonctionnalités déjà présentent dans l'API permet d'avoir une compréhension fine des méthodes de génération et d'acquisition de signaux analogiques hautes fréquences avec la carte. Afin d'avoir la plus haute précision sur toute la totalité du stage, la carte a été configurée sur une plaine échelle de  $\pm 1$  V.

### 2.1 Conversion numérique/analogique

La génération de signaux analogiques s'effectue par l'utilisation d'un composant configurable capable de générer indéfiniment deux signaux analogiques périodiques sur une plaine échelle configurable. Ce composant fait partie de la famille des convertisseurs numériques analogiques haute performance. Le convertisseur est cadencé par horloge externe générée par un quartz avec une fréquence de cadencement de 125 MHz, ce qui permet de générer une valeur tous les 8 nanosecondes.

Toutes les voies sont configurables indépendamment les unes des autres. Pour chaque voie, les configurations permettent de modifier la forme du signal généré (sinusoïdale, carré, triangulaire, sciage montante, sciage descendante, signal continue, signal PWM qui est un signal carré avec une modulation sur son rapport cyclique), la fréquence entre 1 Hz et 50 MHz (voir tableau 1), l'amplitude et l'offset sur une plage de fréquence de  $\pm 1$  V, la phase en radian entre  $-2\pi$  et  $2\pi$  et le temps de montée et de descente pour un signal carré et PWM en microsecondes. Il est aussi possible de configurer la source de déclenchement du générateur, qui par défaut est déclenché par le FPGA. Une partie de la mémoire vive DDR3 de la Red Pitaya est réservé pour le composant. Une partie de cette mémoire réservée permet au composant de sauvegarder les configurations effectuées lors de l'exécution de l'application, il n'est donc pas nécessaire de renvoyer toutes les configurations lorsque l'on souhaite changer la forme du signal émis.

Chaque valeur du signal généré a une précision de 14 bits, ce qui signifie que sur une pleine échelle de  $\pm 1$  V, le composant peu générer 16 384 valeurs entre -1 et +1 volt avec quantum (voir annexe VI.1.5) de :

$$q = \frac{2}{2^{14}} = 122,07 \mu V$$

Pour que le signal soit généré correctement, il ne reste plus qu'à activer les sorties souhaitées, puis déclencher leur génération (si elle est contrôlée par le FPGA, dans notre cas, et non par un déclenchement externe).

Cependant, il ne faut pas oublier que chaque valeur du signal sont générées toutes les 8ns. Si nous voulons maintenir une bonne qualité du signal, il est conseillé de ne pas descendre en dessous de 10

points par périodes, ce qui fixe une fréquence maximale du signal de sortie à 12,5 MKz. De plus, le CNA ne propose pas de récupérer la forme des signaux générés, chacun devra être calculé par l'application afin d'être visualisée.

Après avoir compris les différentes fonctionnalités, il s'ensuit une première caractérisation de la carte. Ces étapes sont cruciales dans le développement de l'application, car elles permettent de savoir si le programme engendre des erreurs et permettent de tester les performances de la carte.

Pour cette première caractérisation un signal sinusoïdal de 10 kHz et d'amplitude 0,8 volt est généré sur la voie de sortie 1 de la Red Pitaya (OUT1). Le signal émis est observé sur un instrument de mesure fiable, un oscilloscope numérique MSOX2024A de Keysight Technologies, comme le montre le schéma de câblage de la figure 14. Le convertisseur numérique analogique de la Red Pitaya est imaginé par un oscillateur, suivie d'un interrupteur commandé par le quartz de 125 MHz, que l'on indiquera par la variable  $f_{quartz}$ . Le signal analogique généré est transmis par un câble coaxial à l'oscilloscope numérique MSOX2024A de Keysight Technologies.

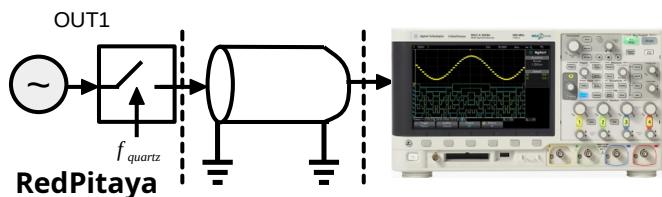


Figure 14 : Schéma de câblage de la Red Pitaya et d'un oscilloscope

Sur la figure 15, nous pouvons observer des captures de l'oscilloscope affichant la représentation temporelle et fréquentielle du signal généré. Afin de ne pas avoir de valeur erronée en tension, l'impédance d'entrée de l'oscilloscope a été adaptée en fonction de l'impédance de sortie de la Red Pitaya de 50 ohms, voir tableau 1.

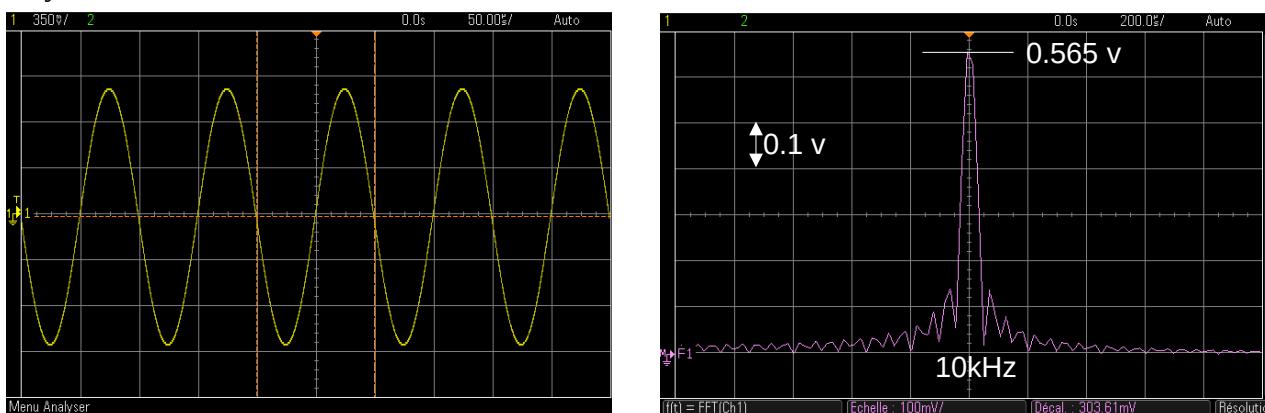


Figure 15 : Visualisation sur un oscilloscope MSOX2024A du signal sinusoïdal généré sur la voie 1 de la Red Pitaya STEMlab 125-14 en temporelle à gauche et en fréquentielle à droite

Sur la représentation temporelle, nous retrouvons bien l'amplitude maximale du signal de 0,8 volts et sur le spectre fréquentiel la fréquence fondamentale du signal à 10 kHz à la position du pic sur l'axe des abscisses.

Avec cette première caractérisation nous pouvons valider le fonctionnement de la carte en tant que générateur périodique.

Caractéristiques du CNA	
Canaux de sortie	2
Plage de fréquence	1 à 50 MHz
Résolution	14 bits
Signal buffer	16 384 samples
Plage de tension en sortie	$\pm 1$ V
Couplage de sortie	DC
Résistance de sortie	50 $\Omega$

Tableau 1 : Caractéristique du convertisseur numérique analogique de la Red Pitaya STEMlab 125-14

## 2.2 Génération d'un signal multi-tons avec un seul oscillateur

Pour les utilisations futures, il est nécessaire de produire un signal sinusoïdal, qui se compose de plusieurs fréquences. Un tel signal peut être obtenu par la combinaison de plusieurs signaux sinusoïdaux, que nous appellerons ton.

Dans le domaine analogique, la création d'un signal à plusieurs tons utilise plusieurs oscillateurs sinusoïdaux de fréquences et d'amplitudes réglables. Les signaux de sortie sont successivement additionnés par un amplificateur additionneur pour voir le signal souhaité.

Toutefois, la carte utilisée ne possède qu'un seul oscillateur pour générer notre signal. Pour répondre à ce problème, le convertisseur numérique analogique (CNA) offre la possibilité de générer un signal arbitraire, constitué d'une forme d'onde périodique personnalisable de 16 384 échantillons. Ces valeurs sont inscrites dans un tampon de la mémoire vive. Le CNA se chargera de lire ce tampon afin de reproduire le signal périodique.

Cependant, les niveaux de tension doivent respecter la limite de la tension de sortie de  $\pm 1$  V, sinon, la Red Pitaya normalisera l'ensemble du tampon.

De plus, ce signal arbitraire est soumis aux mêmes paramètres de base que les formes d'ondes prédéfinies (amplitude, phase, offset et fréquence). Cette configuration impose à l'utilisateur de n'inscrire qu'une seule période du signal normalisé de 16 384 échantillons et d'adapter ces paramètres pour modifier la forme du signal.

L'utilisateur à la possibilité d'inscrire plusieurs périodes du signal dans ce tampon, mais cela engendre des erreurs sur la fréquence du signal. Prenons l'exemple, l'utilisateur inscrit deux périodes sinusoïdales de fréquence de 10 kHz, et une amplitude de 1 V dans le tampon. Lorsque le signal généré est mesuré, la fréquence de sortie réelle aura changé. L'utilisateur observera une onde

sinusoïdale de 20 kHz, car la Red Pitaya aura estimé la fréquence de lecture du buffer en fonction de la configuration de la fréquence. Ainsi, pour calculer la fréquence de sortie réelle, nous pouvons utiliser la formule suivante :

$$f_{output} = f_{buff} \times \frac{N_{buff}}{N_{periode}} \text{ avec}$$

$f_{output}$	est la fréquence de sortie réelle
$f_{buff}$	est la fréquence de lecture du buffer
$N_{period}$	est le nombre de period du signal inscrite dans le tampon

Bien que cette solution reste envisageable, il est plus simple et plus précis d'inscrire une seule période du signal complet et de normaliser son amplitude.

De cette manière, cette règle doit être respectée, pour générer un signal avec plusieurs fréquences combinées. La formule suivante indique comment le générer :

$$buffer[k] = \sum_{i=0}^n A_i \sin\left(\frac{2\pi f_i k}{N_{buff} f_{min}}\right) \text{ avec}$$

$A_i$	l'amplitude de chaque signaux
$f_i$	la fréquence de chaque signaux
$f_{min}$	la fréquence minimale des $f_i$
$n$	le nombre d ' harmonique du signal
$N_{buff}$	la taille du buffer de 16384
$k$	indice de la valeur calculée sur l'intervalle $[0, N-1]$

Nous effectuons un premier test avec cette formule. Comme vous pouvez l'observer sur la figure 17, il n'y a aucun problème de génération lorsque les fréquences sont multiples (avec 10 kHz d'amplitude 0,8 volt et 100 kHz d'amplitude 0,2 volt) ; Cependant, lorsque les signaux ne sont pas multiples, par exemple avec 12 kHz et 13,1 kHz nous pouvons observer un saut d'amplitude lorsque le CNA revient à la première valeur du buffer, dû à une discontinuité entre la fin et le début du buffer, voir figure 16.

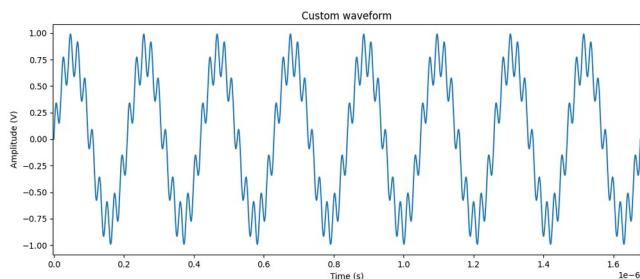


Figure 17 : Génération d'un signal périodique avec un ton à 10 kHz et un ton à 100 kHz

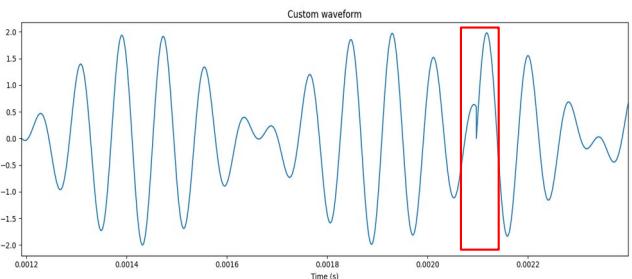


Figure 16 : Mauvaise génération d'un signal périodique avec un ton à 12 kHz et un ton à 13,1 kHz

Ce problème peut-être corrigé en calculant préalablement la fréquence de répétition du signal. Par exemple si nous observons les deux signaux sinusoïdaux de 12 kHz et 13,1 kHz sur le même graphique (figure 18), nous pouvons voir qu'ils démarrent en phase. Le signal avec la plus grande fréquence se déphase petit à petit jusqu'à revenir en phase avec l'autre signal, ce temps entre deux périodes de phase permet de définir la fréquence de répétition du signal.

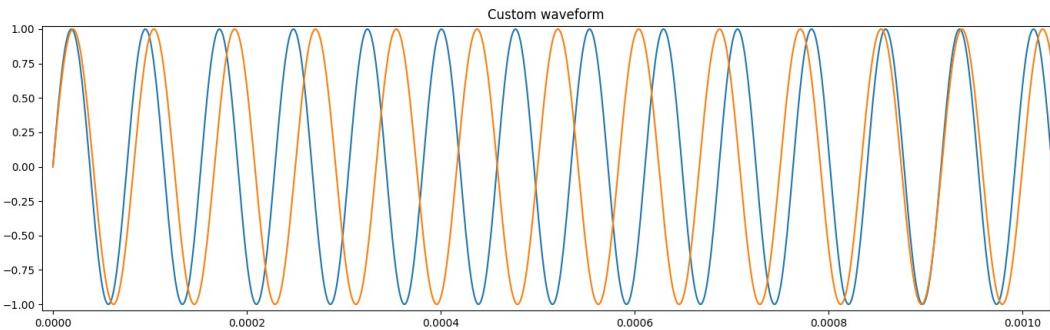


Figure 18 : Signaux sinusoïdaux de fréquence proche

Par un calcul de PGCD, nous pouvons déterminer le dénominateur commun entre toutes les fréquences du signal à générer. La valeur résultante est la fréquence du signal complet lorsque tous les tons sont en phase. En inversant la valeur nous pouvons connaître sa période.

$$T_{output} = \frac{1}{GCD}(\{f_1, f_2, \dots\})$$

Afin de n'inscrire qu'une seule période du signal arbitraire, il faut calculer le pas temporel entre chaque valeur :

$$\Delta t = \frac{T_{output}}{N_{buff}}$$

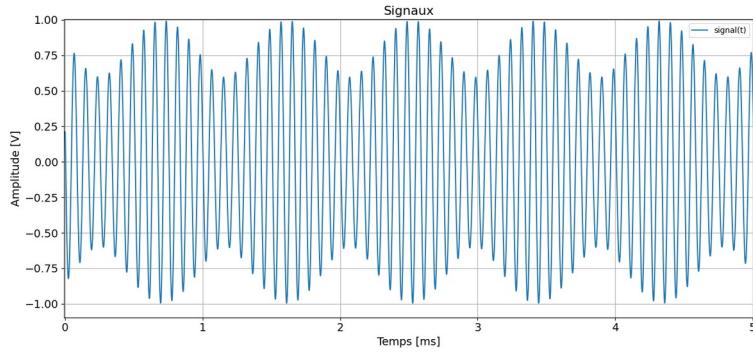
Pour calculer les 16 384 valeurs, il ne manque plus qu'à multiplier ce pas par une variable entière incrémentée de 0 à  $N_{buff}-1$  :

Le temps calculé est inséré dans les fonctions sinus pour calculer la valeur du signal :

$$t_k = k \Delta t \text{ avec } k \in [0, N_{buff}-1]$$

$$buffer[k] = \sum_{j=1}^{N_{tons}} A_j \sin(2\pi f_j k t_k)$$

Avec cette nouvelle fonctionnalité, nous pouvons générer un signal à plusieurs tons avec un seul oscillateur. Une autre contrainte s'ajoute, puisque la taille du tampon est fixe (16 384), la période de répétition du signal résultant ne doit pas être trop grande. Plus la période du signal est grande, plus le pas temporel entre deux valeurs successives est augmenté pour tenir dans le buffer. Certaines valeurs essentielles du signal ne seront pas inscrites, ce qui peut détériorer sa qualité. Pour déterminer si le signal peut être inscrit dans le buffer, le théorème de Nyquist impose qu'un signal doit être représenté par un minimum de deux points sur sa plus grande tonalité. Cependant en pratique, il faut bien une dizaine de points par période pour que le signal généré soit suffisamment propre pour être exploité. En respectant toutes ces contraintes le signal peut être correctement généré, voir figure 19.



*Figure 19 : Génération d'un signal périodique constitué de deux fréquences à 12 kHz et 13,1 kHz*

## 2.3 Acquisition de signaux

L'acquisition de signaux analogiques est l'étape qui doit s'effectuer avec le plus de rigueur. Grâce à la Red Pitaya nous avons la possibilité d'acquérir deux signaux simultanément sur une plaine échelle de  $\pm 1$  V ou de  $\pm 20$  V. La pleine échelle est configurable physiquement par des cavaliers situés sur la carte (voir tableau 2). Comme pour le CNA, ce composant fait partie de la famille des convertisseurs analogiques numériques (CAN) haute performance.

Une partie de la mémoire vive de la Red Pitaya lui est attribué pour sauvegarder les signaux acquis et les paramètres. Le convertisseur est cadencé par la même horloge externe que le CNA de 125 MHz, ce qui lui permet de faire au maximum une acquisition tous les 8 nanosecondes.

Caractéristiques de CAN	
Canaux d'entrée	2
Bande passante	50 MHz
Résolution	14 bits
Taille de la mémoire d'échantillonnage	16 384 samples
Plage d'entrée	$\pm 1$ V (LV) ou $\pm 20$ V (HV) <sup>1</sup>
Couplage d'entrée	Courant Continu (DC)
Sensibilité minimale à la tension	$\pm 0,122$ mV ou $\pm 2.44$ mV <sup>1</sup>
Trigger extérieur	Via le connecteur d'extension
Impédance d'entrée	$1M\Omega$

*Tableau 2 : Caractéristique du convertisseur analogique de la Red Pitaya STEMlab 125-14*

Pour effectuer une acquisition correcte de données, une condition de déclenchement doit être spécifiée par une commande de l'API C (*Rp\_AcqStart*), sinon les données retournées peuvent être

1 configuration avec un cavalier sur la carte

invalides ou corrompues, car le FPGA peut acquérir des données beaucoup plus rapidement que le système d'exploitation Linux ne peut les lire à partir des registres FPGA.

Les 16 384 échantillons capturés sur chacun des canaux d'entrée de la Red Pitaya sont stockés dans un buffer de mémoire parcourue de manière circulaire jusqu'à la prochaine collecte (figure 20). La position d'inscription du premier échantillon dans la mémoire est définie aléatoirement. Afin de lire les données acquises, il est nécessaire de récupérer la position du premier échantillon (`rp_AcqGetWritePointer(&pos)`).

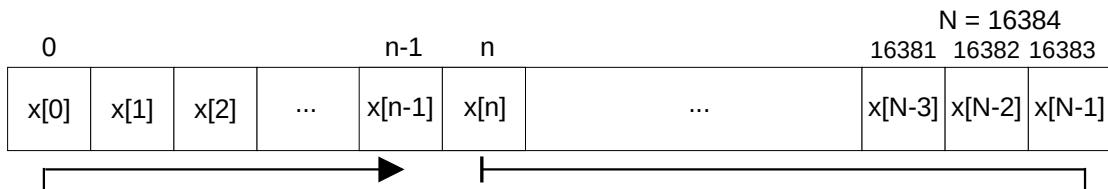


Figure 20 : Écriture de chaque échantillon dans le buffer à partir de la position  $n$

Un échantillon est une valeur entière non signé de 32 bits. Lors de la lecture du tampon, l'API Red Pitaya se chargera de convertir chaque valeur en virgule flottant signé 32bits.

Des paramètres configurables permettent de régler les conditions à partir duquel l'échantillonnage doit démarrer après l'appel de la fonction de l'API `Rp_AcqStart`, comme sur un oscilloscope standard. Quatre de ces paramètres sont cruciaux : la fréquence d'échantillonnage, le niveau de déclenchement du trigger, le délai de déclenchement du trigger et le canal d'entrée de référence.

### 2.3.a La fréquence d'échantillonnage

La fréquence d'échantillonnage est un paramètre crucial. Elle définit le nombre de fois par seconde qu'une valeur analogique est convertie en échantillon numérique. Sur la Red Pitaya, une haute fréquence d'échantillonnage permet de capturer des signaux à haute fréquence avec une grande précision. Une fréquence d'échantillonnage élevée permet de capturer plus de détails du signal, résultant en une meilleure représentation numérique du signal analogique original. Selon le théorème de Nyquist, la fréquence d'échantillonnage doit être au moins deux fois supérieures à la fréquence maximale du signal à mesurer pour éviter l'aliasing. Plus la fréquence d'échantillonnage est élevée, plus la précision temporelle des échantillons est grande, ce qui est crucial pour des analyses détaillées du signal. Cependant pour effectuer des analyses fréquentielles avec une haute précision, il serait utile de réduire cette fréquence d'échantillonnage afin d'acquérir plus de périodes du signal à analyser ou pour faciliter le traitement des signaux à des fréquences plus basses sans perdre les informations importantes du signal.

Le processus de réduction de la fréquence d'échantillonnage s'effectue sur la Red Pitaya en divisant la fréquence du quartz de 125 MHz par une puissance de 2 sur 14 bits, appelé la décimation. Les 15 valeurs de configuration de la décimation sont : 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192 et 65 536.

Les exigences de l'application à réaliser impose de déterminer le facteur de décimation optimal pour acquérir un signal dont nous connaissons sa fréquence. Pour acquérir le plus de période du signal, j'ai ajouté une option permettant de choisir le nombre de points par période. À partir de cette valeur le programme cherche la décimation la plus proche, puis indique à l'utilisateur le nombre de points par période réellement acquis.

### **2.3.b Le niveau de déclenchement du trigger**

Le niveau de déclenchement de l'anglais *trigger level* est un paramètre crucial dans les systèmes d'acquisition de signaux. Il définit le point de tension (ou courant) spécifique à partir duquel le système d'acquisition commencera à capturer les données. Sur la figure suivante, le niveau de déclenchement est affiché sous la forme d'une ligne horizontale. Cette ligne est la valeur spécifique de la tension (ou du signal) qui, une fois atteinte par le signal d'entrée, déclenche le système d'acquisition pour commencer l'enregistrement des données. Pour notre application, cela permet de capturer un signal périodique de manière stable chaque fois que le signal atteint un certain niveau.

### **2.3.c Le délai de déclenchement du trigger**

Le délai de déclenchement, de l'anglais *trigger delay*, est un paramètre qui permet de décaler le point de déclenchement du trigger en nombre d'échantillons. C'est le nombre d'échantillons qui sépare le point de déclenchement de l'acquisition effective des données, et le moment à partir duquel le trigger commence à comparer la valeur au niveau de déclenchement. Cela est utile lorsque la première valeur acquise est déjà au-dessus du niveau de déclenchement, jusqu'à ce que le nombre d'échantillons acquis dépasse le délai de déclenchement, le système calcul la moyenne du signal, puis adapte la condition de déclenchement en fonction de la moyenne calculée, ce qui certifie que la valeur précédente le premier échantillon a une valeur inférieure au niveau de déclenchement fixé.

Le déclenchement de l'acquisition sur les deux voies d'entrées ne sont pas reliés, même si les paramètres sont définis globalement. Chaque voie à son propre trigger, la capture des signaux peut démarrer à des temps différents, mais l'API attendra que les deux signaux soient acquis totalement.

### **2.3.d Définir un canal de référence**

Ce dernier paramètre est utile lorsque deux signaux sont acquis simultanément, et que l'un d'entre eux peut être considéré comme une référence. Cela permet de synchroniser le déclenchement de l'acquisition des deux signaux sur le même trigger. Cela permet une comparaison directe et précise des deux signaux. En définissant un signal de référence, nous pouvons observer la variation de l'autre signal par rapport à l'autre.

## 2.4 Caractérisation du temps d'exécution du programme entre deux acquisitions

Dans la section précédente, il a été évoqué que le temps écoulé entre deux acquisitions n'est pas négligeable. Il doit être caractérisé afin de connaître le délai entre deux acquisitions successives. Cette caractérisation s'est répartie en deux tests. Le premier effectue 1 campagne de 1000 acquisitions en mesurant le temps d'exécution entre deux acquisitions successives. Le deuxième test effectue 1000 campagnes de 1 acquisition.

Une campagne est un regroupement d'instructions, effectuant dans un premier temps, une initialisation de l'API Red Pitaya, dans un second temps l'activation et la configuration du générateur de tension sur les deux voies, dans un troisième temps l'acquisition des signaux sur les deux voies et dans un dernier temps l'arrêt du générateur de tension et la libération des ressources de l'API (voir figure 21). Le premier test consiste donc à boucler uniquement sur la partie acquisition, alors que le deuxième test boucle sur l'ensemble de la procédure décrite.

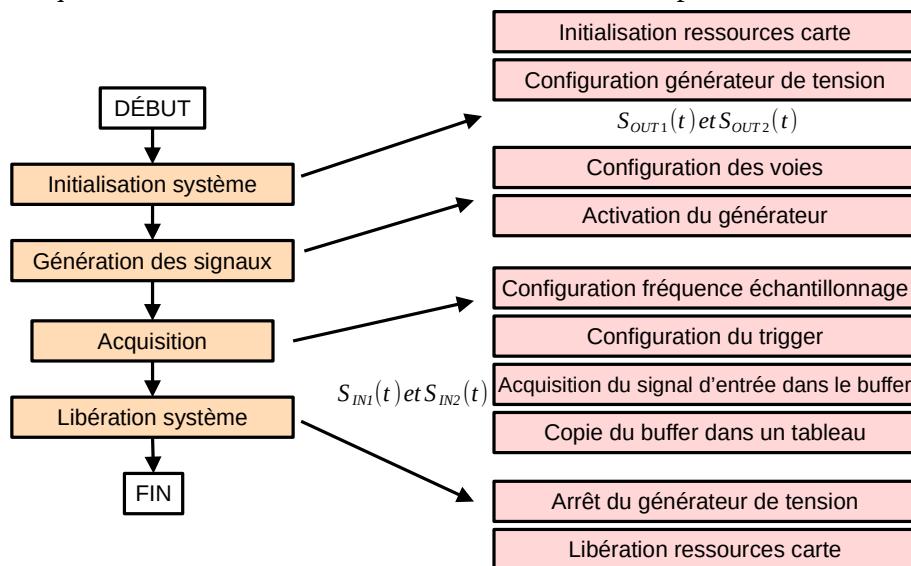


Figure 21 : Procédure normale d'acquisition

Les valeurs mesurées sont inscrites dans un fichier textuel pour les 1000 répétitions. À partir des données récupérées nous pouvons les afficher dans un graphique à l'aide de la librairie 'Matplotlib' de l'environnement de programmation 'Python'.

En comparant les figures 22 et 23, nous constatons que le temps d'acquisition moyen est proche. Nous pouvons lire que le temps d'exécution du programme entre deux acquisitions pour le premier test avoisine les 143 µs et 145 µs pour le second. Cependant en comparant les temps maximums mesurés, nous observons que la seconde méthode peut atteindre un temps d'exécution jusqu'à 580 µs.

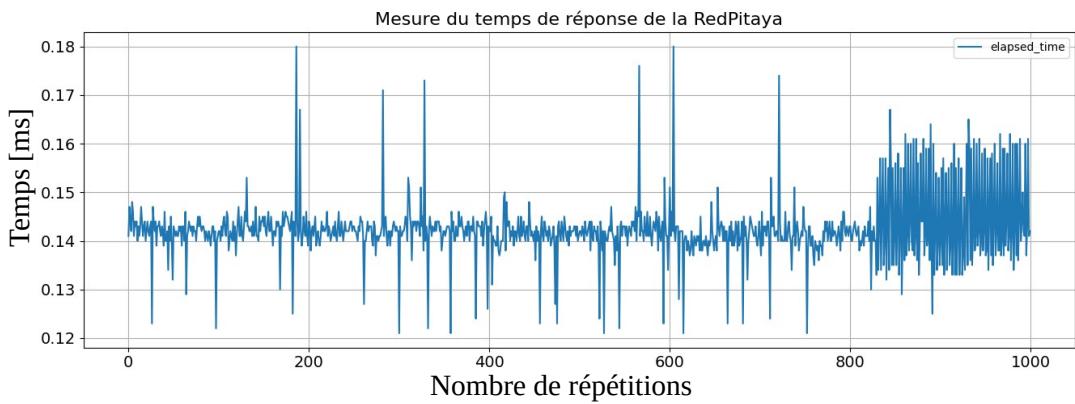


Figure 22 : Mesure du temps d'exécution avec 1 campagne de 1000 acquisitions

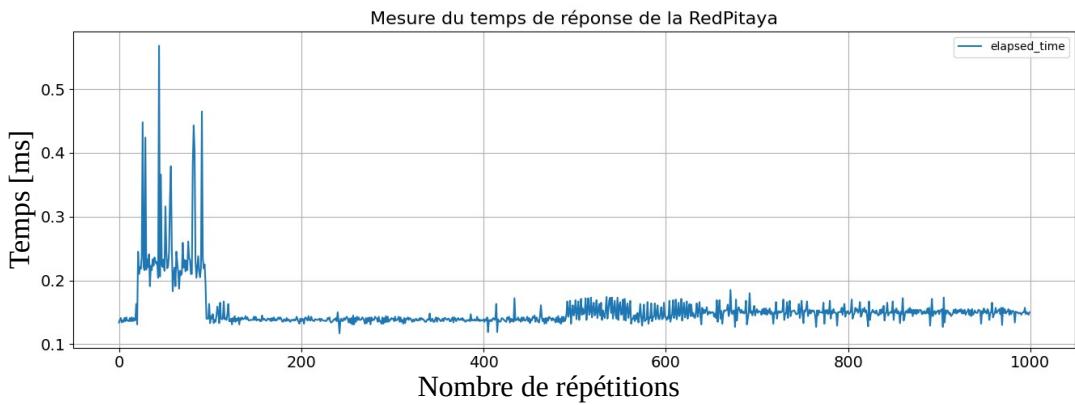


Figure 23 : Mesure du temps d'exécution avec 1000 campagnes de 1 acquisition

Une distribution de ces valeurs sous forme d'histogramme permet d'observer différemment ces valeurs :

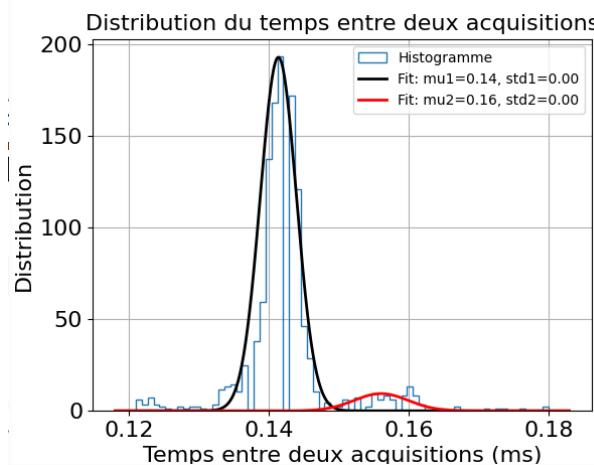


Figure 25 : Distribution du temps d'exécution pour 1 campagne de 1000 acquisitions

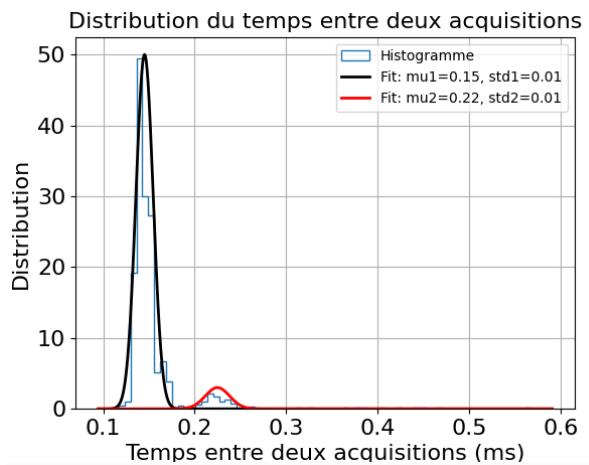


Figure 24 : Distribution du temps d'exécution pour 1000 campagnes de 1 acquisition

### 3. Implémentation d'une fonctionnalité de démodulation

Toutes les fonctionnalités exposées dans ce chapitre s'appliquent une fois que le signal est acquis. Le chapitre utilise des concepts du traitement du signal numérique avec des valeurs échantillonnées. Toutes les caractéristiques exposées ne sont pas disponibles sur la Red Pitaya, et ont été entièrement développées pour le projet en C++. Le travail fourni peut être assimilé à une librairie C++ proposant la plupart des outils indispensables pour effectuer du traitement du signal numérique sur Red Pitaya.

L'objectif du code réalisé était de déterminer de manière précise les figures de mérite d'un composant MEMS (Micro ElectroMechanical System) en utilisant la démodulation. L'utilisation de la démodulation est un moyen efficace d'extraire la partie continue d'un signal périodique. Il est donc essentiel de travailler avec des outils spécifiques au domaine des fréquenciel. En raison du caractère temporel des signaux obtenus, il existe un outil mathématique performant pour les convertir dans le domaine fréquentiel, la Transformée de Fourier. En général comme les signaux se composent de plusieurs fréquences indésirables, un autre outil, le filtre, permet de les rejeter afin de ne garder que la partie continue du signal.

Dans une première section, nous exposerons les principaux outils développés pour réaliser une démodulation, tels que le fenêtrage, le calcul de transformée de Fourier rapide et le filtre à réponse impulsionnelle infini. Dans une seconde section, nous aborderons le fonctionnement et la mise en œuvre d'une détection synchrone. Enfin, une dernière partie traitera de la conception d'un test pratique.

#### 3.1 Calcul d'une transformée de Fourier rapide

En traitement du signal numérique, la Transformée de Fourier Rapide ou *Fast Fourier Transform* (FFT) en anglais, est un algorithme efficace pour calculer la Transformée de Fourier Discrète (DFT de *Discrete Fourier Transform*) d'un signal échantillonné. Dans le domaine complexe, la DFT convertit un signal temporel échantillonné en ses composantes fréquentielles permettant une analyse en termes de fréquence plutôt qu'en termes de temps.

La DFT comme son nom l'indique est l'adaptation du théorème de Fourier (TF) du domaine continu, pour l'utilisation sur des signaux échantillonnés (voir VI.1.6 pour un rappel de base sur les transformées de Fourier).

Un signal échantillonné est un signal fini, le calcul de la DFT s'effectue sur un nombre d'échantillon fixe. Sur la Red Pitaya le calcul de la DFT s'effectuera au maximum sur 16 384 échantillons. Ainsi, comme décrit dans l'expression suivante, l'intégrale de la TF est remplacée par une sommation qui s'effectue sur l'intervalle  $[0, N-1] \in \mathbb{N}$  :

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i \frac{2\pi}{N} kn} \text{ avec } \forall k \in [0, N-1], \mathbb{N}$$

$x[n]$  est le signal échantillonné et  $X[k]$  est le résultat de la DFT.

Une autre méthode plus rapide permet d'obtenir le même résultat, en utilisant l'algorithme de Cooley-Tukey. Cet algorithme décompose le calcul de la DFT en une séquence de DFT plus petits. Pour le stage, il est intéressant de développer une telle optimisation, car cela permet de diminuer le temps de calcul et donc le temps mort entre deux acquisitions.

Cette méthode, dite algorithme de Cooley-Tukey Radix-2 DIT, divise une DFT de taille N en deux DFT entrelacées de taille N/2 (figure 26). La première partie une contenant les échantillons d'index pair et l'autre contenant les échantillons d'index impair. Pour que le séquencement s'effectue symétriquement, la taille du signal N doit être une puissance de deux. Nous pouvons donc diviser l'équation en deux parties. L'algorithme calcule d'abord les DFT des entrées à index pairs (2m), puis les entrées aux index impairs (2m+1), et combine ces deux résultats pour produire la FFT de toute la séquence :

$$X[k] = \sum_{n=0}^{\left(\frac{N}{2}\right)-1} X[2n] e^{-\frac{2\pi i}{N}(2n)k} + \sum_{m=0}^{\left(\frac{N}{2}\right)-1} X[2m+1] e^{-\frac{2\pi i}{N}(2m+1)k}$$

Pour chaque valeur entière de k, l'équation calcule le nouvel échantillon avec les échantillons calculés précédemment, c'est un algorithme récursif.

Cette méthode réduit le nombre d'instruction de  $2^N$  à  $N \log(N)$  (figure 26).

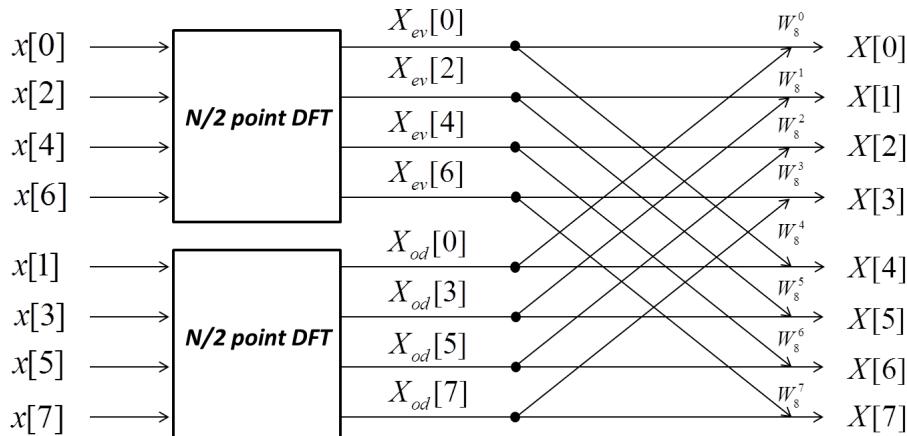


Figure 26 : Algorithme de Cooley-Tukey Radix-2 DIT

Cependant, cet algorithme récursif utilise beaucoup de mémoire vive. En algorithmique, un processus est dit récursif lorsqu'il s'appelle lui-même, directement ou indirectement, jusqu'à atteindre la solution souhaitée. Chaque processus demande un emplacement mémoire réservé afin de sauvegarder ses variables locales. Pour chaque appel une nouvelle partie de la mémoire sera réservée. Toute la mémoire réservée récursivement par l'ensemble de ce processus sera libérée seulement lorsque la dernière valeur sera calculée. Dans notre cas (N=) 16 384 tampons d'une taille minimum de 64 octets (8bits) seront réservés lors de l'appel du processus ce qui fait une taille intolérable de 1 048 576 octets. Pour limiter l'utilisation de la mémoire, tout en gardant le même fonctionnement, les étapes récursives sont reproduites par une méthode itérative effectuant le calcul par permutation inversée. Cette permutation effectue un renversement des bits des indices de chaque échantillon, pour former une famille paire et une famille impaire. De la sorte, les étapes de l'algorithme effectueront des combinaisons d'éléments pairs et impairs.

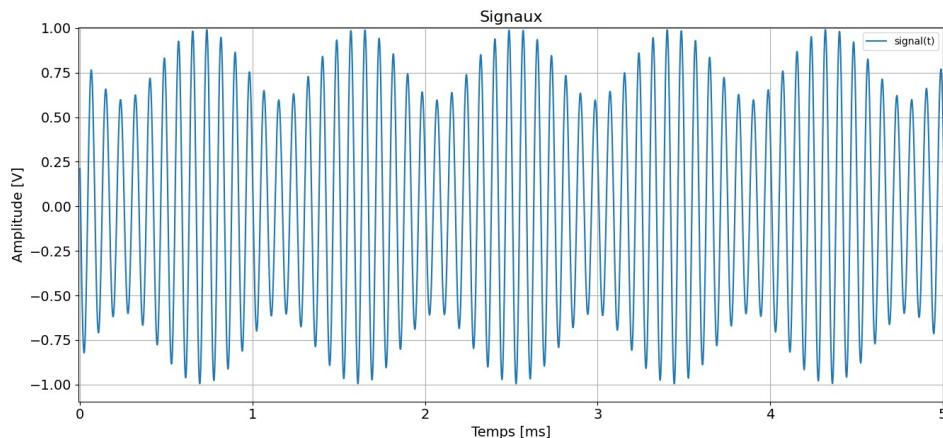
Par exemple, si nous considérons une séquence de longueur  $N=8$  sur 3 bits, nous pouvons écrire leurs indices de 0 à 7 en binaires :

- 0 : 000
- 1 : 001
- 2 : 010
- 3 : 011
- 4 : 100
- 5 : 101
- 6 : 110
- 7 : 111

En renversant les bits de ces indices, nous obtenons la séquence suivante :

- 0 : 000 (0) pair
- 1 : 100 (4) pair
- 2 : 010 (2) pair
- 3 : 110 (6) pair
- 4 : 001 (1) impair
- 5 : 101 (5) impair
- 6 : 011 (3) impair
- 7 : 111 (7) impair

Voici un exemple de calcul de FFT. La figure 27 montre un signal échantillonné composé de la somme d'un sinus de fréquence 12 kHz ( $f_1$ ) et d'amplitude 0,8 V et d'un sinus de fréquence 13,1 kHz ( $f_2$ ) et d'amplitude 0,2 V généré sur la voie 1.



*Figure 27 : Signal arbitraire composé de la somme d'une fréquence à 12 kHz et d'une fréquence à 13,1 kHz*

Le signal généré est rebouclé sur l'entrée 1 de la Red Pitaya afin d'y être échantillonné. Par la suite sa FFT est calculée afin d'avoir son spectre. Le spectre est par la suite récupéré sur un ordinateur afin d'y être affiché. Comme le spectre est du domaine complexe, nous ne pouvons pas le représenter tel quel sur un graphe. Nous devons récupérer son module et la phase en utilisant la formule suivante :

$$module[k] = \frac{|spectre[k]|}{N}$$

Et on peut aussi récupérer sa phase en calculant son argument. Le module de la FFT est observable sur la figure 28.

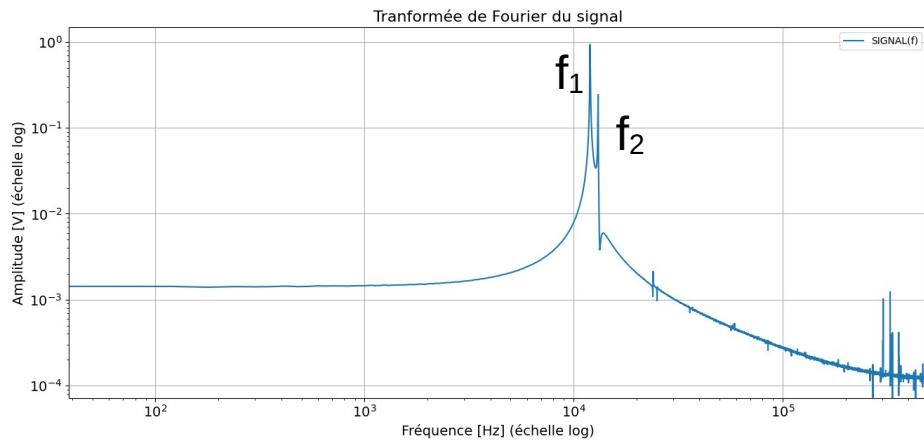


Figure 28 : FFT du signal arbitraire composé de la somme d'une fréquence à 12 kHz et d'une fréquence à 13,1 kHz

### 3.2 Le Fenêtrage

En traitement du signal, le fenêtrage est utilisé dès lors que l'on s'intéresse à la partie d'un signal volontairement limitée. En théorie, pour limiter un signal infini, nous le multiplions par une fenêtre de pondération (rectangulaire ou porte) avant d'effectuer le calcul de FFT. Cette fenêtre vaut 1 dans l'intervalle des valeurs à récupérer, zéro à l'extérieur. Puis chaque échantillon est retardé pour que le premier commence à  $t=0$ . Sa fonction de transfert en temporel est la suivante :

$$h(t) = \begin{cases} 1 & \text{si } t \in [T_1, T_2] \\ 0 & \text{sinon} \end{cases}$$

Nous obtenons alors une expression permettant de calculer le signal tronqué  $S_h(t)$  :  
 $S_h(t) = s(t)h(t)$

Pour notre utilisation, le signal est déjà tronqué par la taille du buffer. Cette troncature peut s'apparenter, à un fenêtrage rectangulaire. En discret la fenêtre se définit comme un peigne de Dirac de taille N, où la distance entre deux pics est la période d'échantillonnage  $dt$  (figure 29).

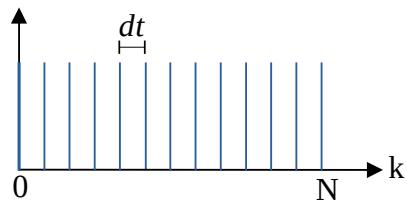


Figure 29 : Peigne de Dirac

En passant le signal temporel dans le domaine fréquentiel, nous obtenons un produit de convolution :  $S_h(f) = (S * H)(f)$  où  $H(f)$  est la transformée de Fourier de la fenêtre H. Sa transformée de Fourier donne alors un sinus cardinal, voir figure 30.

$$S_{peigne}(f) = S(f) * \tau \operatorname{sinc}(\pi f)$$

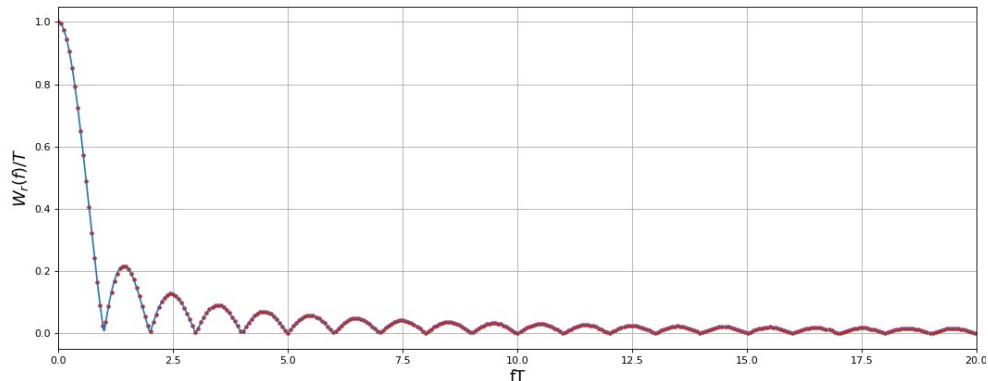


Figure 30 : Sinus cardinal (tracé temporel en bleu, tracé discret en rouge)

La forme cardinale de la Transformée de la porte, se répercute sur celle du signal, engendrant un étalement spectral, et des fuites spectrales sur toutes les fréquences de la FFT, voir figure 31. L'étalement spectral cause une augmentation de la largeur des pics des tonalités du signal. Ce résultat engendre une perte de précision. Les pics de plus basses amplitudes seront masqués par la fuite spectrale. Sur une échelle logarithmique, les pics avec des fréquences les plus proches, à hautes fréquences, seront regroupés en un seul pic large.

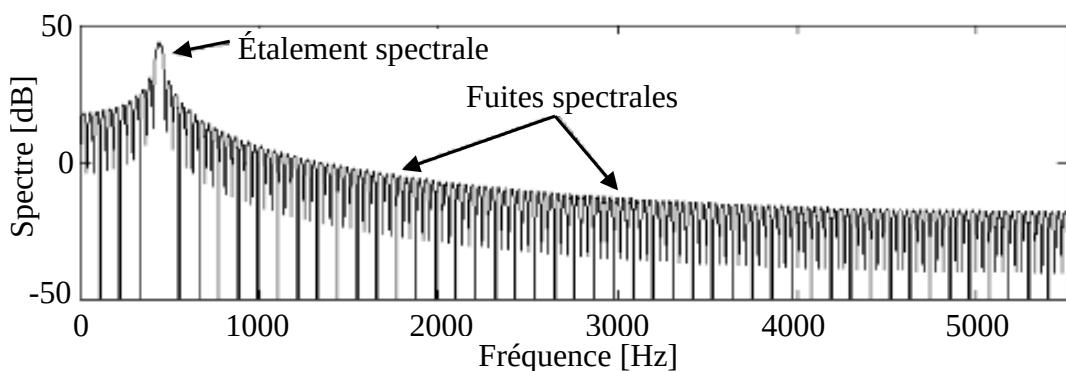


Figure 31 : FFT du signal tronqué

Après échantillonnage, il sera obligatoire de filtrer le signal par une fenêtre pour restreindre l'étalement spectral. Pour cela, il y a une multitude de types de porte de formes différentes, dont la majorité ont été implémentées et testées en C++. La fenêtre de Blackman (figure 32) a été retenue, car elle entraîne moins d'oscillation.

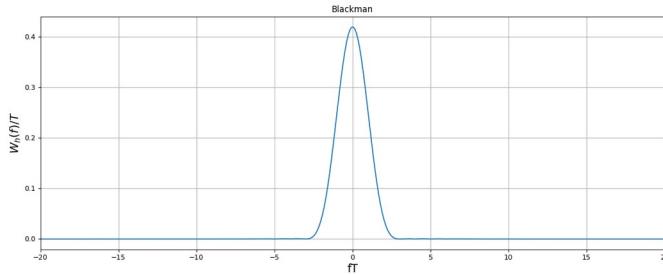


Figure 32 : Fenêtre de Blackman

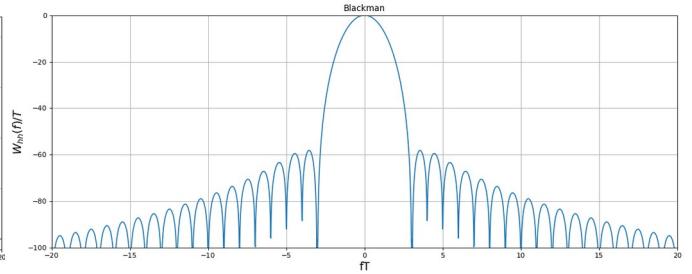


Figure 33 : FFT de la fenêtre de Blackman

L'utilisation d'une fenêtre de Blackman permet d'obtenir un FFT plus propre, même si nous percevons toujours la diminution d'amplitude causé par la fenêtre en fréquentiel, voire figure 34.

L'algorithme développé dans le cadre de ce stage propose une classe C++, nommée ‘Window’, permettant de sélectionner le type de fenêtre à utiliser et ses paramètres. La classe se comporte comme un composant. Nous envoyons les paramètres à la classe via une méthode nommée ‘set’. Et une méthode ‘apply’ qui filtre le signal ce qui nous permet d'obtenir la figure 34.

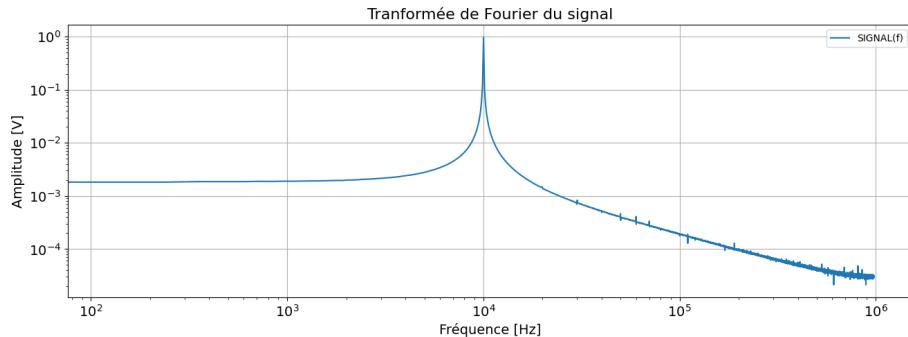


Figure 34 : FFT du signal tronqué multiplié par une fenêtre de Blackman

### 3.3 Filtre numérique

Le filtrage est une opération linéaire qui consiste à sélectionner certaines parties du spectre d'un signal. C'est un élément de base du traitement du signal très utilisé en électronique rejetant une partie de son spectre fréquentiel en laissant passer certaines de ces composantes. Aujourd'hui, nous préférions effectuer le filtrage dans le domaine numérique, car les algorithmes complexes sont beaucoup plus faciles à mettre en œuvre. Les filtres sont généralement des systèmes linéaires invariants, qui ont un comportement et des caractéristiques linéaires n'évoluant pas dans le temps. Il existe différents gabarits de filtre en fonction de la partie du spectre à filtrer : passe-bas, passe-haut, coupe-bande et passe-bande.

Puisque que l'objectif de ce stage était aussi de fournir un outil complet de traitement du signal sur Red Pitaya, j'ai développé une fonctionnalité permettant de choisir le type de filtre utilisé, avec la forme du gabarit, l'ordre et les fréquences de coupures configurables. Pour synthétiser un filtre numérique, nous considérons comme connu le gabarit du filtre analogique et nous cherchons un système numérique caractérisé par la fonction de transfert  $H(z)$  d'un filtre analogique permettant de satisfaire le gabarit analogique.

Il existe deux types de filtre numérique, le filtre à réponse impulsionnelle finie (FIR pour *Finite Impulse Response*) et le filtre à réponse impulsionnelle infinie (IIR pour *Infinite Impulse Response*).

#### 3.3.a Filtre à réponse impulsionnelle finie

Le filtre FIR est défini par la combinaison linéaire entre les valeurs du signal d'entrée  $x$  et les valeurs des coefficients de la fonction de transfert du filtre  $b_k$ , pour donner les valeurs du signal de sortie  $y$ . Le nombre de coefficient utilisé définit l'ordre  $M$  du filtre. Voici la fonction de transfert discrète du filtre :

$$y[k] = \sum_{m=0}^{M-1} b_k x[k-m] \text{ avec } k \in [0; N-1] \text{ et } N = 16384$$

Pour ce filtre, chaque échantillon de sortie est la moyenne pondérée de l'échantillon d'entrée le plus récent. La sortie est la somme de toutes les réponses impulsionales de chaque échantillon d'entrée (non récursif). En algorithmique, la somme de l'expression mathématique sera implémentée comme une boucle effectuant plusieurs sommes. Le filtrage de chaque échantillon utilise les valeurs filtrées précédemment de retard  $k-m$ . Ce filtre est donc un filtre à mémoire.

### **3.3.b Filtre à réponse impulsionnelle infinie**

L'équation suivante montre la fonction de transfert discrète de ce type de filtre. Il est défini par l'équation aux différences où  $x$  représente les valeurs du signal d'entrée et  $y$  les valeurs du signal de sortie. Le filtre a besoin des coefficients  $a_k$  et  $b_k$ . Si  $a_k$  ne comporte aucun coefficient, il s'agira d'un filtre FIR.

$$y[k] = \sum_{m=0}^{M-1} b_m x[k-m] - \sum_{p=0}^{P-1} a_p x[k-p] \text{ avec } k \in [0; N-1] \text{ et } N = 16384$$

Ce filtre fonctionne de la même façon que le filtre à réponse impulsionnelle finie.

Le filtre IIR est intéressant, car il est plus rapide que son frère en termes de vitesse de calcul. A la différence du filtre FIR qui dépend des entrées actuelles et passées, le filtre IIR dépend des entrées actuelles et passées et aussi des sorties passées. Avec ses nouveaux paramètres, le filtre IIR nécessite un ordre plus faible qu'un filtre FIR pour un même résultat.

Supposons que nous voulions un filtre passe-bas avec une certaine bande passante et atténuation. Un filtre FIR peut nécessiter 100 coefficients pour obtenir une réponse en fréquence adéquate, alors qu'un filtre IIR peut obtenir une réponse similaire avec seulement 4 coefficients (2 pour les  $a$  et 2 pour les  $b$ ).

Ainsi pour chaque échantillon, FIR effectuera 100 multiplications et 99 additions, alors que son frère n'effectuera que 8 multiplications, 6 additions et une soustraction.

### **3.3.c Définir les coefficients du filtre IIR**

Afin de réaliser la démodulation, le filtre IIR est donc le plus adapté car il nécessite d'ordre plus petit. Cependant une nouvelle contrainte s'ajoute, comme nous voulons pouvoir choisir l'ordre, le gabarit (passe-haut, passe-bas, passe-bande et coupe-bande) et la ou les fréquences de coupures du filtre, nous devons trouver une méthode afin de déterminer ses deux tableaux de coefficients.

La façon la plus répandue de calculer les coefficients d'un filtre IIR est de définir la fonction de transfert d'un filtre analogique en fonction du gabarit et de le numériser en effectuant une transformation bilinéaire.

La définition d'un filtre analogique se définit en Laplace comme la division de deux polynômes.

$$H(s) = \frac{B(s)}{A(s)}$$

où  $s$  est la variable complexe de Laplace.

Par la suite la transformation bilinéaire substitue la variable  $s$  par l'expression suivante :

$$s = \frac{1}{T} \ln(z) \approx \frac{2}{T_e} \frac{1-z^{-1}}{1+z^{-1}}$$

où  $T_e$  est l'intervalle d'échantillonnage ( $T_e = \frac{1}{f_e}$ , avec  $f_e$  la fréquence d'échantillonnage).

Afin d'obtenir la fonction de transfert numérique, il faut substituer  $s$  dans la fonction de transfert analogique  $H(s)$  avec l'expression de la transformation bilinéaire. Cela permet d'obtenir une nouvelle fonction de transfert discrète en termes de  $z$ .

$$H(z) = H\left(\frac{2}{T_e} \frac{1-z^{-1}}{1+z^{-1}}\right)$$

Enfin, en développant et simplifiant l'équation avec la nouvelle valeur de  $s$ , nous obtenons la forme standard de la fonction de transfert d'un système linéaire invariant :

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_M z^M}{a_0 + a_1 z^{-1} + \dots + a_P z^P}$$

Avec  $b_i$  et  $a_i$  les coefficients du filtre IIR.

Maintenant, il faut synthétiser la forme du filtre. Le filtre le plus simple à implémenter est le filtre de Butterworth. Une première étape doit synthétiser le filtre passe-bas analogique en fonction de l'ordre que nous souhaitons lui donner et une deuxième étape doit transformer celui-ci en fonction du gabarit souhaité.

### 3.3.d Fonction de transfert analogique

La fonction de transfert d'un filtre de Butterworth d'ordre  $n$  est définie pour avoir une réponse en fréquence la plus plate possible dans la bande passante. La fonction de transfert normalisée de Butterworth dans le domaine fréquentiel est donnée par l'équation suivante :

$$H_n(j\omega) = \frac{1}{B_n(j\omega)} = \frac{1}{\sqrt{1+\omega^{2n}}}$$

où  $B_n(j\omega)$  le polynôme de Butterworth d'ordre  $n$ .

(voire: [https://en.wikipedia.org/wiki/Butterworth\\_filter](https://en.wikipedia.org/wiki/Butterworth_filter))

Le polynôme de Butterworth est construit de manière à ce que tous ses pôles se trouvent sur un cercle unitaire dans le plan  $s$ , ce qui assure une atténuation monotone de la réponse en fréquence et une stabilité du filtre. Les pôles du filtre de Butterworth sont donnés par :

$$S_k = e^{j2\pi \frac{2k+n+1}{4n}}, k = \{0, 1, \dots, n-1\}$$

Pour que le système soit stable, la sortie du filtre pour une entrée bornée doit également être bornée. Dans le plan  $Z$ , la condition de stabilité est remplie si tous les pôles du système sont à l'intérieur de la moitié gauche du cercle unité dans le plan complexe (c'est-à-dire que leur module est strictement inférieur à 1). Si un pôle se trouve à l'intérieur du cercle unité, la réponse du système à une

impulsion va décroître exponentiellement avec le temps, car le terme correspondant dans la réponse impulsionale aura une amplitude qui diminue au fil des itérations (en raison de la multiplication répétée par une valeur inférieure à 1). En revanche, si un pôle se trouve à l'extérieur du cercle unité, la réponse du système croîtra de manière exponentielle, menant à une instabilité, car la sortie pourrait devenir infiniment grande pour une entrée finie. On peut savoir si un pôle est stable en vérifiant si son argument complexe  $j2\pi \frac{2k+n+1}{4n}$  est compris entre 90 et 270 degrés.

Le filtre de Butterworth normalisé est donc définie comme suit :

$$B_n(s) = \prod_{k=0}^{n-1} \left( s - e^{j2\pi \frac{2k+n+1}{4n}} \right)$$

En réorganisant l'équation pour regrouper chaque pôle avec son conjugué complexe et en utilisant les formules d'Euler, nous pouvons simplifier davantage cette expression :

$$B_n(s) = \prod_{k=0}^{n-1} \left( s - 2 \cos\left(2\pi \frac{2k+n+1}{4n}\right) s + 1 \right)$$

Il nous reste un nombre pair de pôles conjugués complexes, ainsi le polynôme de degré n de Butterworth normalisé est donné par l'équation suivante :

$$B_n(s) = \begin{cases} \prod_{k=0}^{\frac{n}{2}-1} \left( s^2 - 2 \cos\left(2\pi \frac{2k+n+1}{4n}\right) s + 1 \right) & \text{Si } n \text{ est pair} \\ (s+1) \prod_{k=0}^{\frac{n-1}{2}-1} \left( s^2 - 2 \cos\left(2\pi \frac{2k+n+1}{4n}\right) s + 1 \right) & \text{Si } n \text{ est impair} \end{cases}$$

Voici l'implémentation de cette fonction en pseudo-code :

```

PROC CalculerPôlesButterworth(ordre: ENTIER, pôles: LISTE_COMPLEXE)
    SI ordre <= 0 ALORS
        AFFICHER "Erreur : L'ordre doit être positif."
        RETOURNER
    FIN SI

    // Effacer la liste des pôles
    VIDE(pôles)

    // Calcul des pôles pour l'ordre donné
    moitiéOrdre <- ordre DIV 2

    POUR k DE 0 À moitiéOrdre - 1 FAIRE
        theta <- PI * (2 * k + 1 + ordre) / (2 * ordre)
        pôle <- EXP(IMAGE_CIRCONFRENTIELLE(theta))
        AJOUTER(pôles, pôle)
    FIN POUR

    // Ajouter un pôle supplémentaire pour les ordres impairs
    SI ordre MOD 2 ≠ 0 ALORS
        AJOUTER(pôles, (-1.0, 0.0))
    FIN SI
FIN PROC

```

Tableau 3 : Pseudo-code du calcul des pôles du filtre de Butterworth

En fonction du gabarit choisi, une transformation doit être effectué pour dénormaliser le filtre avec le bon gabarit et les bonnes fréquences de coupures.

Enfin pour calculer ses coefficients, nous effectuons une transformation bilinéaire comme expliqué précédemment.

### 3.4 La démodulation par détection synchrone

Une technique essentielle dans le domaine des communications et du traitement du signal est la démodulation de signaux. Que ce soit dans les réseaux filaires, les réseaux hertziens ou les capteurs, il est crucial de pouvoir extraire des informations pertinentes dans un signal ondulant. L'un des éléments essentiels de cette démodulation consiste à récupérer la composante continue d'un signal pour évaluer son amplitude et sa phase.

Une technique efficace consiste à multiplier le signal acquis, par une sinusoïde, nommée la composante en phase (I) et par une autre sinusoïde déphasée de  $\pi/2$ , nommée la composante en quadrature (Q) de même fréquence que la plus petite fréquence composant le signal. Les deux composantes résultantes, seront filtrées parallèlement par deux filtres passe-bas, afin d'isoler leur composante continue (fréquence=0) des autres composantes (fréquence>0). Le module des composantes filtrées permettront de calculer l'amplitude pour chaque échantillon. La tangente inverse de la composante en phase filtré sur la composante en quadrature filtrée permet de calculer la phase pour chaque échantillon. Nous pouvons démontrer ce processus à partir d'un signal avec une fréquence et une amplitude fixe :

$$S_N = A \sin(\omega t + \phi) \text{ avec } \omega = 2\pi f t, t = \frac{k}{f_s}, k \in [0; N-1] \text{ et } N = 16384$$

Dans un premier temps pour un  $k$  donné, le processus calcule la composante en phase en effectuant la multiplication de l'échantillon  $k$  signal par un sinus avec la même fréquence que le signal d'entrée :

$$I[k] = A \sin(\omega t + \phi) \sin(\omega t) = \frac{A}{2} (\cos(\phi) - \cos(2\omega t))$$

Puis, le processus calcule la composante en quadrature en effectuant la même multiplication avec un cosinus :

$$Q[k] = A \sin(\omega t + \phi) \cos(\omega t) = \frac{A}{2} (\sin(\phi) + \sin(2\omega t))$$

Dans un second temps les composantes sont filtrées par un filtre passe bas avec une fréquence de coupure définie par l'utilisateur, inférieur à la plus petite fréquence du signal. Nous observons dans les équations suivantes que la pulsation a été retirée et qu'il ne reste plus que la phase :

$$Q_{filtered}[k] = LPF(Q[k]) \Rightarrow \frac{A}{2} \sin(\phi)$$

$$I_{filtered}[k] = LPF(I[k]) \Rightarrow \frac{A}{2} \cos(\phi)$$

Dans un troisième temps, le module est calculé avec les expressions suivantes. Grâce à la formule trigonométrique  $\cos^2(a)+\sin^2(a)=1$ , nous pouvions facilement retirer la phase de l'expression :

$$I_{\text{filtered}}[k]^2 + Q_{\text{filtered}}[k]^2 = \left(\frac{A}{2}\cos(\phi)\right)^2 + \left(\frac{A}{2}\sin(\phi)\right)^2 = \frac{A^2}{2}(\cos^2(\phi) + \sin^2(\phi)) = \frac{A^2}{2}$$

$$\text{Amplitude}[k] = \sqrt{\frac{A^2}{2}}$$

Pour finir, la phase est calculée par la tangente inverse de la division des deux composantes :

$$\frac{Q_{\text{filtered}}[k]}{I_{\text{filtered}}[k]} = \frac{\left(\frac{A}{2}\sin(\phi)\right)}{\left(\frac{A}{2}\cos(\phi)\right)} = \tan(\phi)$$

$$\text{Phase}[k] = \tan^{-1}(\tan(\phi)) [2\pi]$$

Un modulo  $2\pi$  est calculé sur chaque valeur de phase, afin qu'elles restent dans l'intervalle  $[-2\pi, 2\pi]$ .

Toutes ces étapes sont répétées pour chaque valeur de k.

Dans l'application, une classe C++ nommée ‘Demodulator’ utilise une méthode ‘set’ afin de configurer l'ordre, la fréquence de coupure des filtres et la fréquence de démodulation. La démodulation est initialisée avec la méthode ‘setup’. Enfin, la méthode ‘apply’ démodule le signal passé en paramètre. La phase et l'amplitude sont inscrites dans deux autres buffers de même taille que le signal. Le tableau suivant expose un exemple de code résumant les expressions indiquées ci-dessus.

```
// Multiplication par les sinusoïdes
double I_component = signal[k] * cos(omega * t);
double Q_component = signal[k] * sin(omega * t);

// Filtrage passe-bas avec un filtre de Butterworth d'ordre 2
double filtered_I = butterworthLowPassFilter(2, I_component);
double filtered_Q = butterworthLowPassFilter(2, Q_component);

// Calcul de l'amplitude et de la phase
double amplitude[k] = 2 * sqrt(filtered_I * filtered_I + filtered_Q * filtered_Q);
double phase[k] = modulo(atan2(filtered_Q, filtered_I), 2*pi);
```

Tableau 4 : Code démonstratif de l'implémentation de la démodulation

Une première caractérisation du processus est effectuée en démodulant un signal sinusoïdal  $s_1(t) = \sin(2\pi f t)$  avec  $f=10\text{kHz}$  pour deux fréquences de coupures différentes du filtre de la démodulation.

Sur la figure 35, nous pouvons observer la démodulation du signal avec une fréquence de coupure de 5 kHz en bleu et de 1 kHz en orange. Cela nous permet de voir l'efficacité du filtre en observant la diminution d'amplitude des oscillations après le régime transitoire du calcul de l'amplitude du signal. Ce graphique nous permet aussi de voir que le temps de réponse de l'amplitude est lié au temps de réponse du filtre  $\tau=1/f_c$ . Ainsi, plus la fréquence de coupure sera petite plus le temps de

réponse du filtre sera grand. Cette causalité est à prendre en compte lorsque nous souhaitons observer un signal de faible fréquence, car le nombre d'échantillon du buffer peut-être insuffisant pour observer la partie stable du signal (régime permanent).

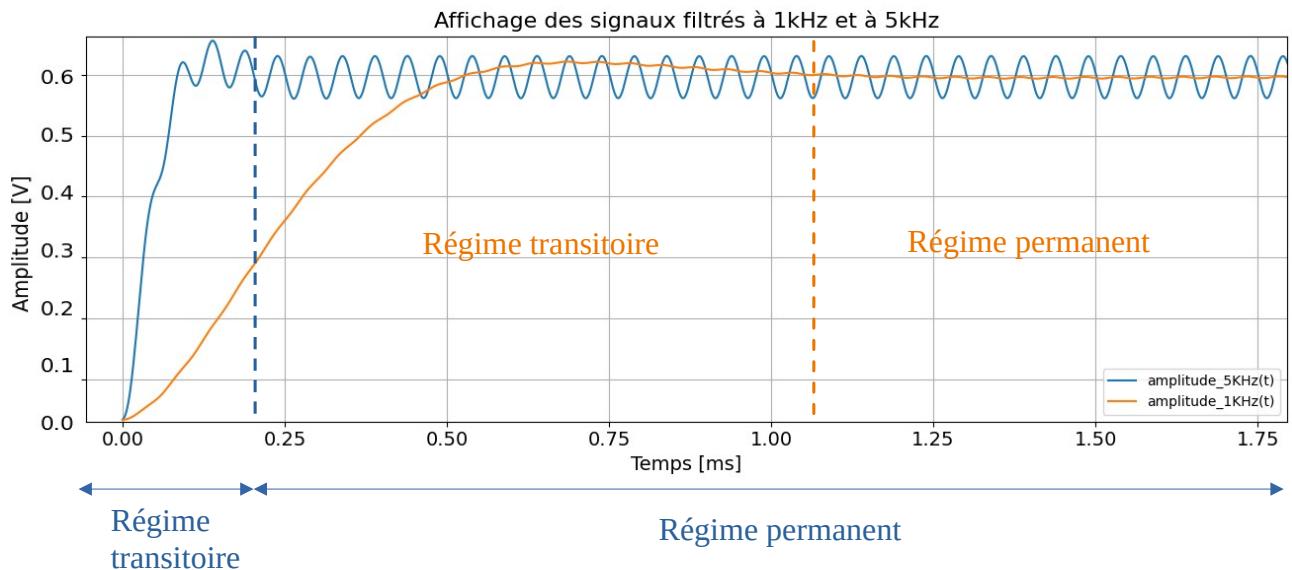


Figure 35 : Amplitudes du signal  $S_1$  démodulé avec une fréquence de coupure à 5 kHz et à 1 kHz

Nous pouvons aussi afficher le module de la Transformée de Fourier de l'amplitude pour visualiser ses composantes. Nous observons que l'amplitude de la composante du signal de fréquence  $f=10$  kHz est bien filtrée, mais que la composante à  $2f$  résultant de la multiplication du signal par un sinus de même fréquence est visible.

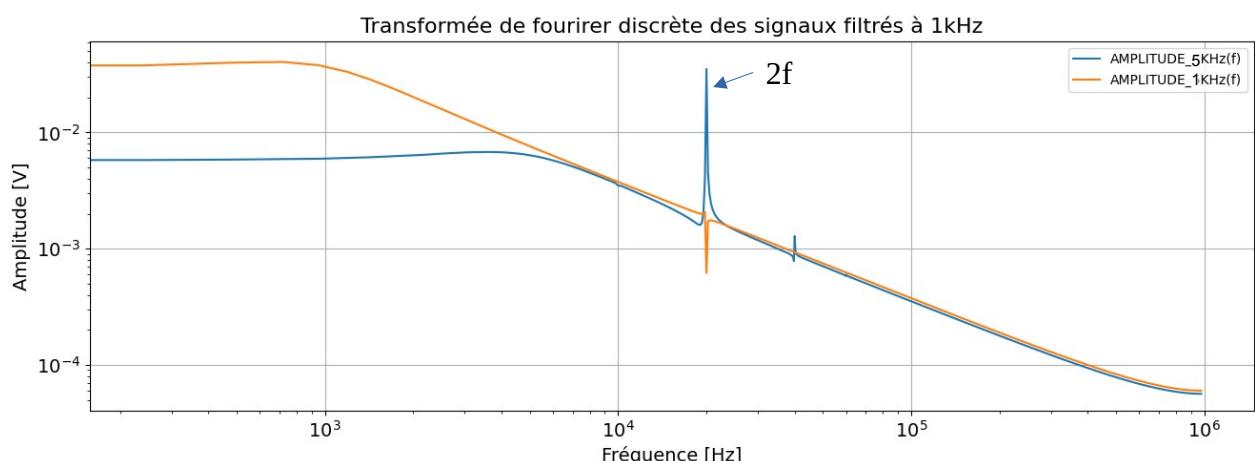


Figure 36 : Module de la Transformée de Fourier des amplitudes calculées avec un filtre de fréquence de coupure 5 kHz (courbe bleue) et 1 kHz (courbe orange)

La même caractérisation est effectuée avec un signal composé de deux sinusoïdes :

$$s_2(t) = 0.8 \sin(2\pi f_1 t) + 0.2 \sin(2\pi f_2 t) \text{ avec } f_1 = 10 \text{ kHz et } f_2 = 100 \text{ kHz} .$$

La démodulation est effectuée à la fréquence du signal avec la plus petite fréquence, dans notre cas se sera 10 kHz.

Nous pouvons observer l'amplitude calculée en temporel et le module de sa FFT pour deux fréquences de coupures de 1 kHz et 5 kHz (figures 37 et 38). Sur la première figure nous pouvons observer que les oscillations visibles sur la courbe bleue ont été filtrées sur la courbe orange pour une fréquence de coupure plus petite. De plus, nous retrouvons bien la valeur efficace de amplitude du signal de 0,565 volt qui équivaut bien à  $\frac{8}{\sqrt{(2)}}$ .

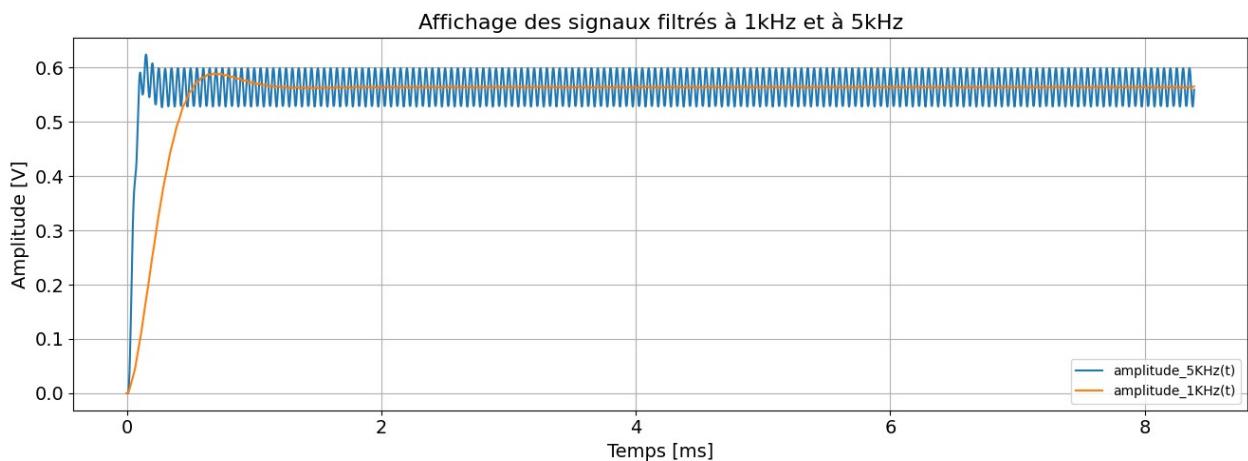


Figure 37 : Amplitudes du signal  $S_2$  démodulé avec une fréquence de coupure à 5 kHz (courbe bleue) et à 1 kHz (courbe orange)

Sur le module de la FFT du signal pour une fréquence de 5kHz, nous observons l'apparition de nouveaux pics qui résultent des équations trigonométriques de la démodulation avec le signal  $s_2$  .

Le développement de la composante de phase donne :

$$I[k] = [A \sin(\omega_1 t + \phi) + B \sin(\omega_2 t + \phi)] \sin(\omega_1 t)$$

$$I[k] = \frac{A}{2} [\cos(2\pi(2f_1)t + \phi) + \cos(\phi)] + \frac{B}{2} [\cos(2\pi(f_2 - f_1)t + \phi) - \cos(2\pi(f_2 + f_1)t + \phi)]$$

Et le développement de la composante en quadrature donne :

$$Q[k] = [A \sin(\omega_1 t + \phi) + B \sin(\omega_2 t + \phi)] \cos(\omega_1 t)$$

$$Q[k] = \frac{A}{2} [\sin(2\pi(2f_1)t + \phi) + \sin(\phi)] + \frac{B}{2} [\sin(2\pi(f_2 - f_1)t + \phi) - \sin(2\pi(f_2 + f_1)t + \phi)]$$

Les pics visibles sur le graphique sont les trois fréquences colorées en rouge sur les équations. De plus, nous observons sur la figure 38 que pour une fréquence de coupure de 1 kHz, la majorité des oscillations ont été filtrées, car les pics ne sont plus visibles sur le module de la FFT.

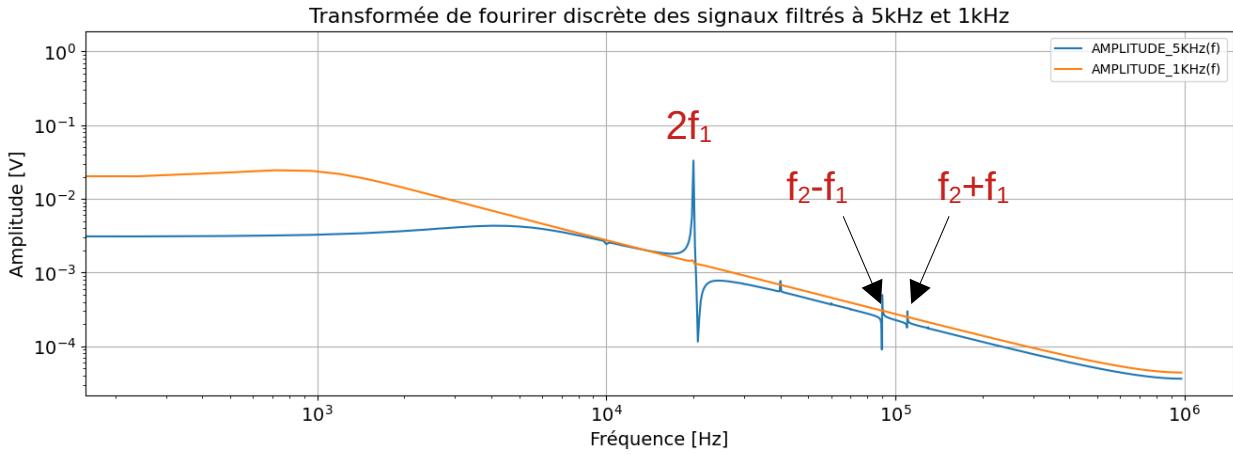


Figure 38 : Module de la Transformée de Fourier des amplitudes calculées avec un filtre de fréquence de coupure 5 kHz et 1 kHz

Grâce à cette caractérisation, nous avons pu démontrer que la démodulation parvient à récupérer de manière adéquate la composante continue d'un signal périodique de forme sinusoïdale. De plus, il a été démontré que la fréquence de coupure du filtre doit être strictement inférieure à la moitié du ton le plus petit et que le temps de réponse du filtre doit être inférieur à la taille temporelle du tampon.

#### 4. Test du filtre avec la démodulation

Afin de tester l'efficacité du filtrage, une méthode simple consiste à démoduler plusieurs signaux sinusoïdaux que nous faisons varier sur une plage de fréquence. La forme du filtre sera observable sur les amplitudes des modules de la FFT de chaque signal démodulé.

Pour le test effectué, il a été choisi de faire évoluer la fréquence du signal à deux composantes avec une première composante fixe à 10 kHz ( $f_1$ ) et une seconde composante avec une fréquence mobile ( $f_2$ ) variant de  $f_1+100$  Hz à  $f_1+10$  kHz. Le signal est filtré par un filtre d'ordre 2 de fréquence de coupure du filtre de 1 kHz. Nous nous attendons que l'amplitude des modules restent constantes avant le filtre et décroissent après le filtre suivant une forme analytique d'un filtre du second ordre (voir figure 39).

Pour la comparaison un filtre analytique a été ajouté indépendamment de la Red Pitaya, sur l'affichage des courbes. L'équation de ce filtre analytique est la suivante :

$$|H(\omega)| = \frac{1}{\sqrt{\left(1 - \left(\frac{\omega}{\omega_0}\right)^2\right)^2 + \left(\frac{\omega}{Q}\right)^2}} \text{ avec } \omega = 2\pi f, \omega_0 = 2\pi f_0 \text{ et } f_0 = 1 \text{ kHz}$$

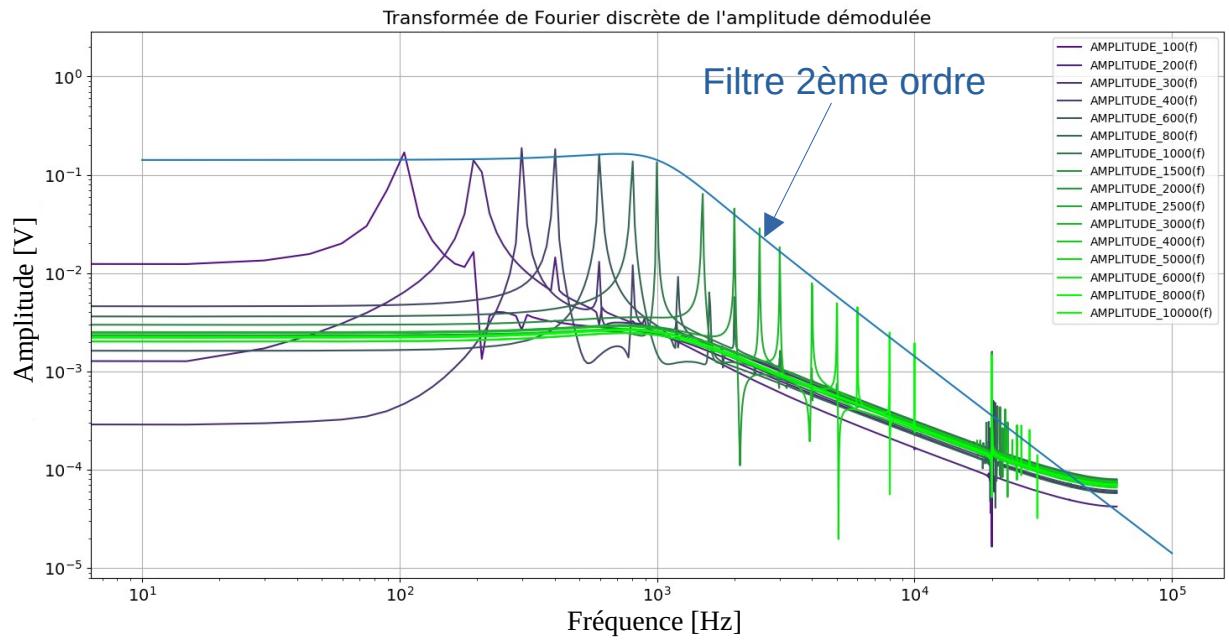


Figure 39 : Vérification de l'ordre du filtre par démodulation

Nous pouvons observer sur la figure 39 que les amplitudes décroissent comme prévu après la fréquence de coupure du filtre de 1 kHz ne suivant la courbe de l'équation analytique du filtre passe bas d'ordre 2 (courbe bleue).

Grâce à cette comparaison nous pouvons valider l'ordre et l'efficacité du filtre.

## 5. Test sur dispositif

Dans cette dernière section, il vous sera présenté la conception un test pour un dispositif MEMS (Micro ElectroMechanical System) afin d'évaluer ses figures de mérite.

Voici à quoi ressemble le MEMS (voir photo figure 40).

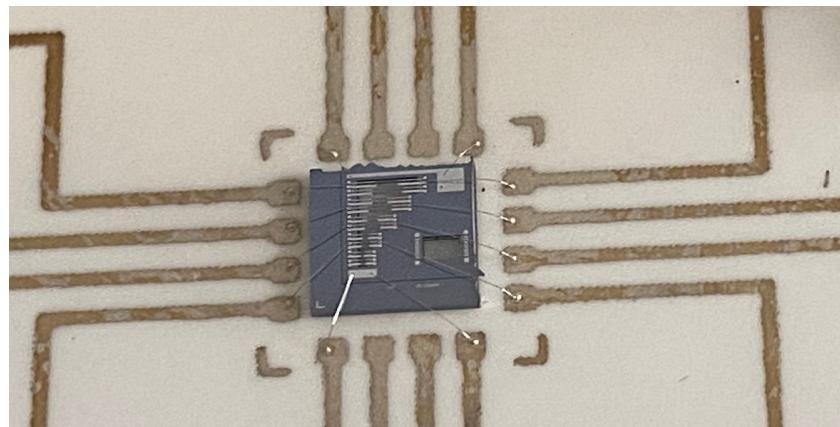


Figure 40 : Photo du dispositif de test

Ce système a été conçu pour des essais en laboratoire, il n'est donc pas référencé par un fabricant. Il se constitue de plusieurs poutres en silicium de différentes tailles et largeurs. Leurs extrémités sont encastrées dans une couche piézo-électrique. Lorsqu'un signal est appliqué sur une entrée du dispositif, celui-ci excite la couche piézo-électrique d'une des l'extrémité d'une poutre. Cette vibration engendrée est transmise par la poutre, puis mesurée sur l'autre extrémité par une autre couche piézo-électrique qui convertie la vibration en signal électrique. En fonction de la fréquence d'excitation, la poutre va osciller différemment et peut se mettre en résonance.

Afin de connaître la fréquence de résonance d'une poutre, il faut faire évoluer progressivement une fréquence sur une plage, démoduler le signal généré par le piézo-électrique avec un signal de référence puis calculer l'amplitude et le déphasage pour chaque fréquence.

Tout d'abord, afin de représenter approximativement la position de la résonance, on a effectué un balayage grossier en fréquence sur une plage de fréquence très étendue et un pas de 500 Hz. Puis, nous avons réalisé un balayage minutieux autour de la position de l'amplitude la plus élevée, sur une plage de fréquence allant de 255 kHz à 265 kHz, avec une variation de 20 Hz, pour observer la résonance.

Pour chaque fréquence, la Red Pitaya génère un signal sinusoïdal d'amplitude 1 volt, sur la première sortie analogique. Le signal analogique est rebouclé sur la voie d'entrée 1 afin d'être utilisé comme référence. Dans le même temps, ce signal est transmis à une entrée du dispositif pour stimuler une de ses poutres. Le signal de sortie du dispositif est analysée sur la voie d'entrée 2 de la Red Pitaya. La Red Pitaya a été configuré pour acquisition des signaux d'entrée soient synchronisés sur le trigger de l'acquisition du signal de référence, permettant ainsi leur échantillonnage simultané.

Après échantillonnage, la démodulation est appliquée sur les deux signaux, nous permettant d'avoir leur amplitude et leur phase. Pour avoir n'avoir qu'une seule valeur d'amplitude et de phase par mesure, l'algorithme effectue un moyennage des 16 384 valeurs de l'amplitude calculées et effectue un moyennage de la différence de la phase du dispositif avec la phase de la référence. Voici les formules :

$$\text{Phase}[f_i] = \left( \frac{1}{N_{\text{freq}}} \right) \sum_{k=k_{\text{filter}}}^{16383} \phi_{\text{demod } 1[k]} - \phi_{\text{demod } 2[k]}$$

$$\text{Amplitude}[f_i] = \left( \frac{1}{N_{\text{freq}}} \right) \sum_{k=k_{\text{filter}}}^{16383} A_{\text{demod } 2[k]}$$

Voici les résultats obtenus en amplitude et en phase sur les figures 41 et 42 :

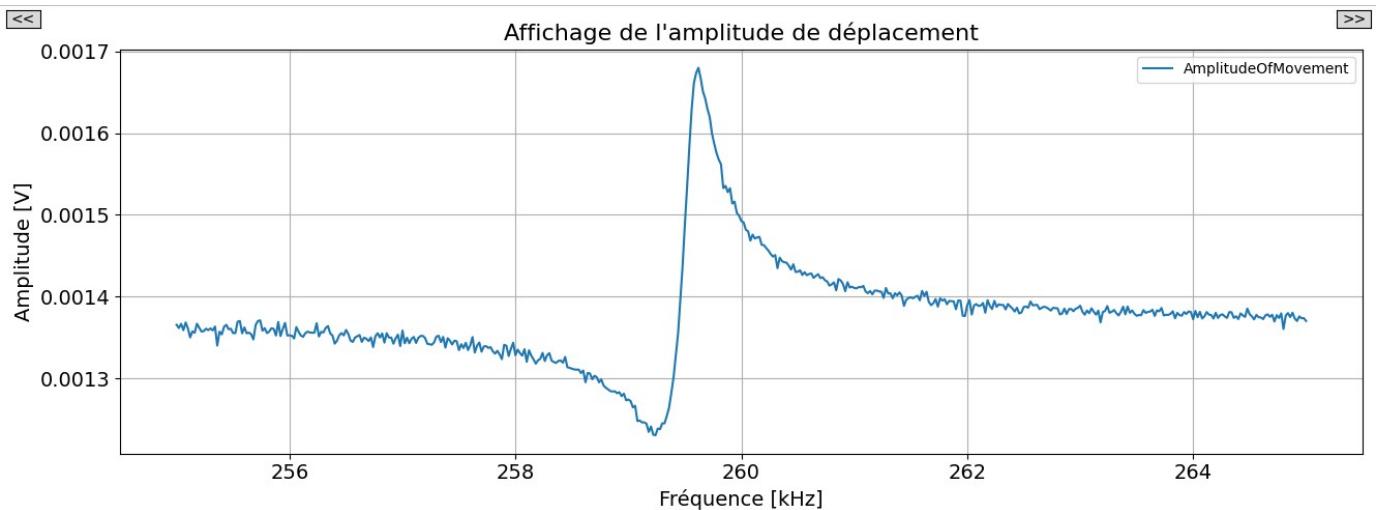


Figure 41 : Amplitude de déplacement du dispositif calculé par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz

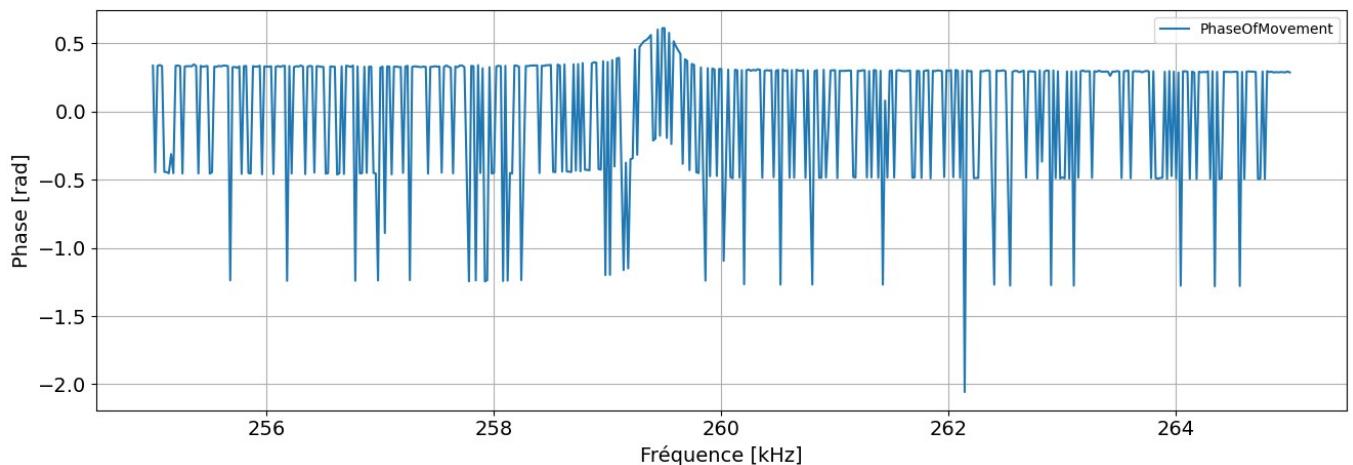


Figure 42 : Déphasage du dispositif avec la référence calculée par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz

Comme nous pouvons l'observer le déphasage entre les deux signaux est très bruité. Nous pouvons visualiser des variations de  $\pi/4$  résultant probablement de l'échantillonnage.

Pour supprimer ces oscillations indésirables, un filtre moyenneur glissant d'ordre 10 est appliqué en fin de processus sur toutes les valeurs d'amplitude et de déphasage. Le diagramme suivant image le fonctionnement du processus de balayage pour une fréquence du balayage.

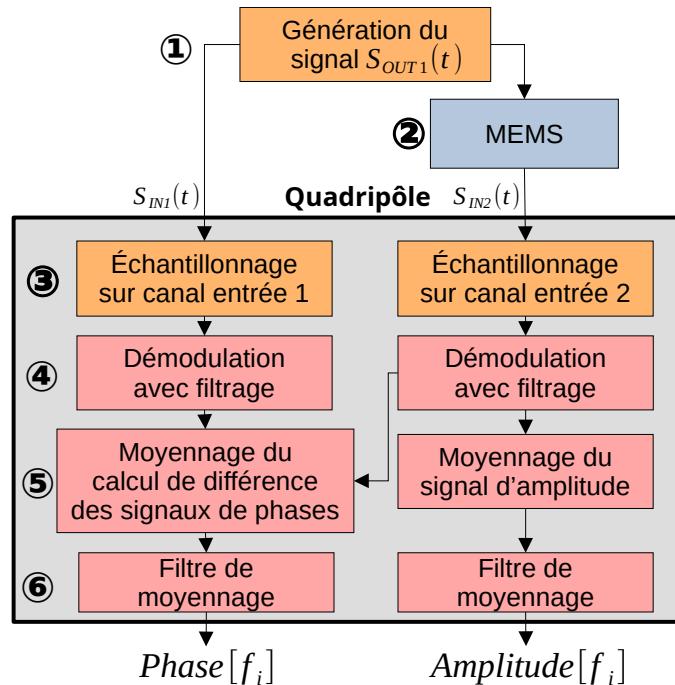


Figure 43 : Diagramme du processus de calcul de l'amplitude et du déphasage du MEMS pour une fréquence du balayage

Sur les figures 44 et 45, nous pouvons visualiser les mérites du composant sur une plage de fréquence de 255 kHz à 265 kHz afin de connaître précisément la forme du pic de résonance en amplitude et en phase. Grâce au résultat généré par l'application, nous pouvons lire un pic de résonance à environ 260 kHz sur l'amplitude et la phase.

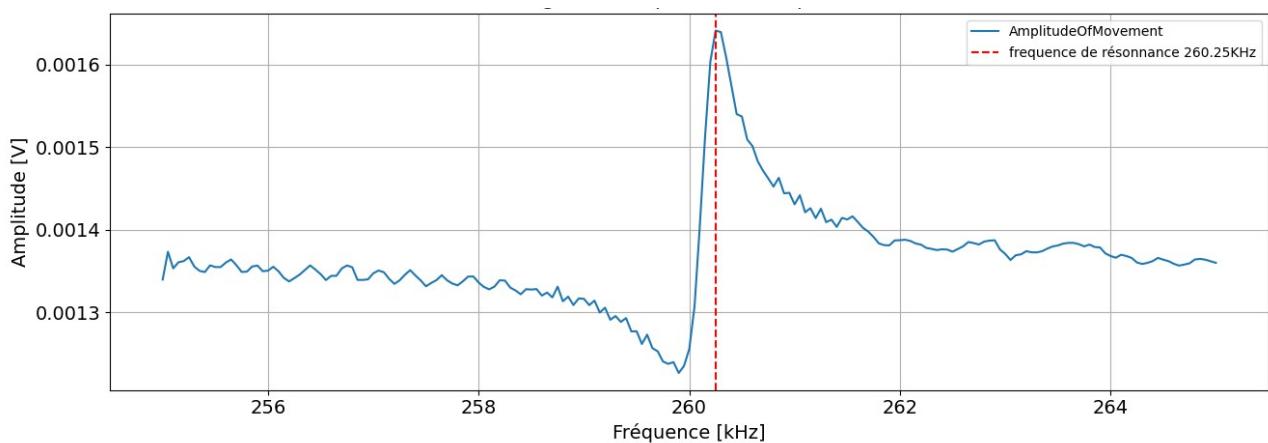


Figure 44 : Amplitude de déplacement du dispositif calculé par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz après application du filtre moyenneur

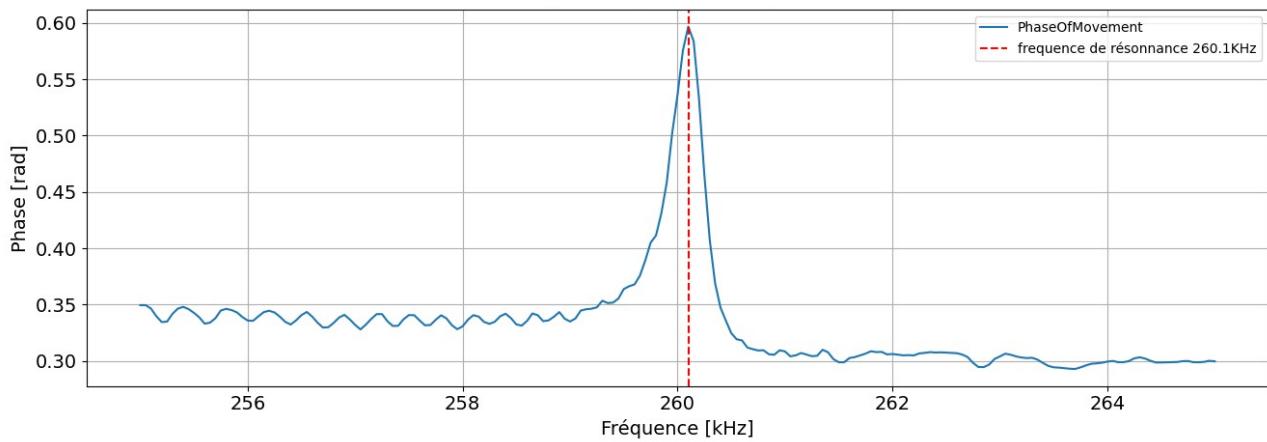


Figure 45 : Phase de déplacement du dispositif calculé par la Red Pitaya sur une plage de fréquence de 255 kHz à 265 kHz

Pour confirmer l'efficacité du programme, nous avons effectué une comparaison entre les résultats précédents et les graphiques obtenus grâce à l'instrument de démodulation HF2LI (figures 46 et 47).

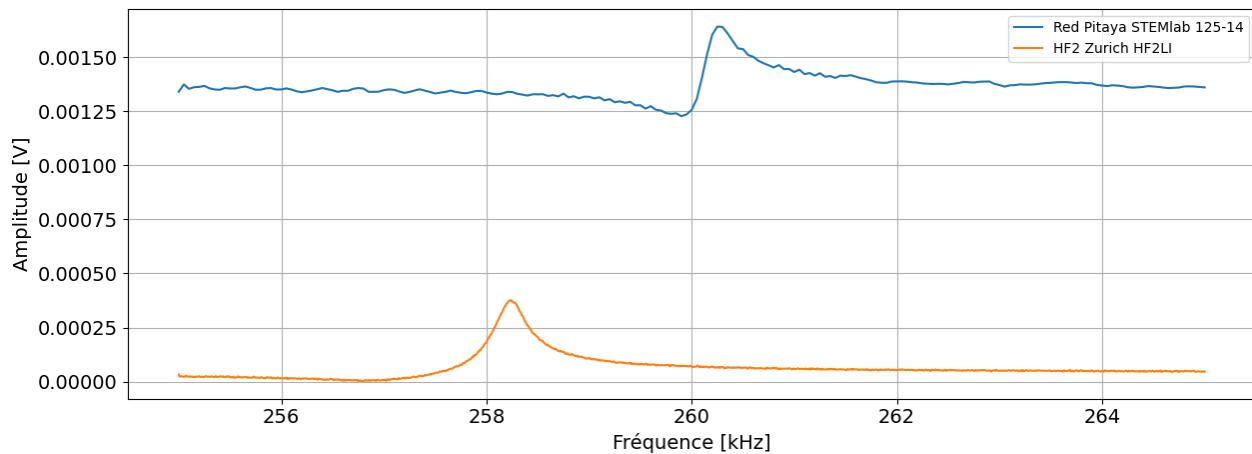


Figure 46 : Amplitude de déplacement du dispositif calculé par la STEMlab 125-14 et la HF2LI sur une plage de fréquence de 255 kHz à 265 kHz

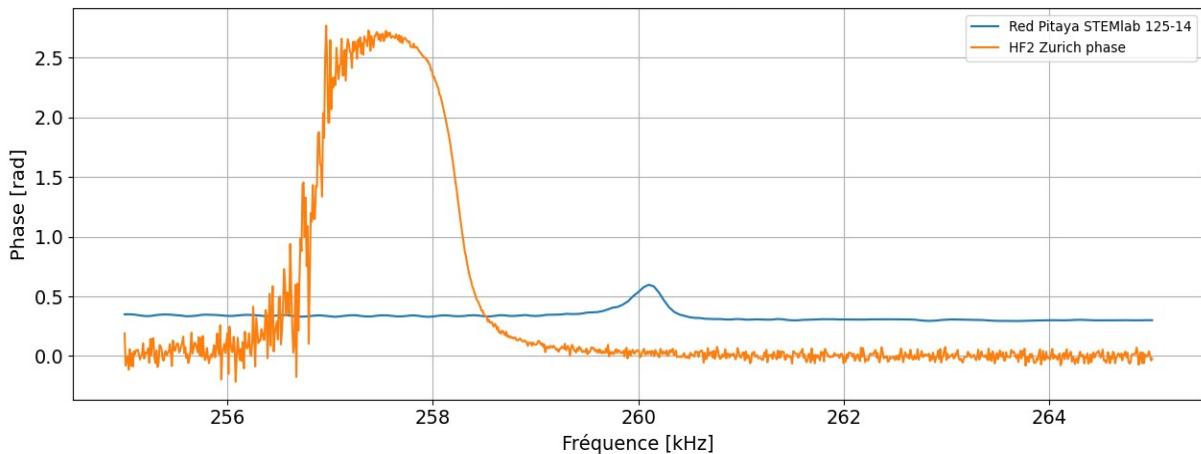


Figure 47 : Phase de déplacement du dispositif calculé par la STEMlab 125-14 et la HF2LI sur une plage de fréquence de 255 kHz à 265 kHz

En superposant les deux résultats, nous observons des différences. Nous pouvons faire quelques hypothèses sur ce phénomène, mais elles demandent de réaliser d'une étude approfondie. Cependant cette étude dépasse le cadre de ce stage. Les hypothèses que nous avons pu en tirer sont les suivantes. Sur les deux graphes il est observable un décalage en fréquence de 2 kHz, résultant probablement de la différence entre les deux dispositifs, dû fait que les mesures n'ont pas été faites le même jour (dérive thermale) et que les instruments de mesure ont besoin d'être synchronisés si l'on veut qu'ils aient la même référence de temps (et donc de fréquence). De plus, on peut observer une différence en amplitude et phase. Cela peut provenir d'une variation en amplitude et phase plus faible de la Red Pitaya, ce qui est probablement dû à la transduction piézoélectrique.

## V. Bilan

L'application développée fournit la base des outils nécessaires pour faire du traitement du signal sur carte d'acquisition Red Pitaya. Plusieurs fonctionnalités de tests ont été implémentées afin de tester les différentes fonctionnalités de l'application. Afin qu'elle s'adapte aux besoins de l'utilisateur, tous les paramètres sont configurables en lignes de commande et nous pouvons accéder à leurs descriptions en entrant la commande 'help'. Pour avoir une traçabilité, chaque test génèrent des fichiers dans le répertoire 'data' de projet sur la Red Pitaya. L'utilisateur peut récupérer facilement les fichiers sur son ordinateur, en exécutant une commande 'shell' que j'ai programmé.

Les fonctionnalités principales de l'application, sont nommés 'modules' afin de faire une distinction avec les fonctionnalités de test. Chaque module génère aussi un répertoire de traçabilité.

Dans le répertoire de traçabilité, chaque module crée un nouveau répertoire avec pour nom la date et l'heure d'exécution du programme. Dans ce nouveau répertoire un fichier de configuration est créé indiquant de nom du test qui a été effectué avec les valeurs de tous les paramètres et les valeurs de mesure. En fonction du type du test ou du module choisi, différents fichiers de donnée au format CSV seront créés, incluant en première ligne, le nom et les valeurs de l'axe utilisé, et sur les lignes suivantes, le nom et les valeurs de chacun des signaux (valeurs réelles) ou spectres (valeurs complexes au format  $a+bj$ ). Pour afficher les données, un script Python permet de sélectionner via un menu en ligne de commande le répertoire de données à afficher. Le script sélectionnera automatiquement le type de graphique à afficher en fonction des informations présentent dans le fichier de configuration.

Si vous voulez plus d'information un rapport technique complète certaines informations. Le code développé lors de ce stage est lisible sur un dépôt public accessible au lien suivant : [https://github.com/ClemtoClem/RedPitaya\\_SignalProcessing.git](https://github.com/ClemtoClem/RedPitaya_SignalProcessing.git)

## 1. Perspective

Les fonctionnalités qui restent à développer sont encore nombreuses. Si un prochain étudiant est amené à continuer le projet, il pourra dans un premier temps implémenter un module capable de réaliser une boucle à verrouillage de phase avec ses différentes fonctionnalités de test. Dans un second temps, il serait envisageable de revenir sur l'interface web afin de programmer un environnement utilisateur plus compréhensible, ce qui faciliterait la configuration des paramètres d'un test ou d'un module et ce qui permettrait d'avoir une traçabilité visuelle de l'historique des mesures effectuées avec leurs descriptions.

Dans un dernier temps, si l'étudiant possède des compétences en programmation FPGA, l'application en ligne de commande et l'application web nécessiterait d'une optimisation avec la rapidité d'exécution du FPGA. Cette optimisation permettrait de diminuer le temps d'exécution du programme et éliminer le temps mort entre chaque acquisition.

## **2. Compétences professionnelles**

Durant ces quatre mois de stage à TIMA, j'ai eu l'opportunité d'enrichir mes connaissances et développer des compétences nouvelles. Les compétences professionnelles que j'ai le plus développées sont les suivantes : intégrer un groupe et interagir dans l'entreprise.

En effet, ayant fait preuve de beaucoup d'autonomie pour réaliser la tâche qui m'était confiée, j'ai dû très vite m'intégrer au sein du service pour connaître les personnes qui travaillent autour de moi. La communication, qu'elle soit orale ou écrite entre mon tuteur, ses collègues, mon collègue stagiaire et moi furent essentielle au bon déroulement du stage. J'ai organisé assez régulièrement des points avec mon tuteur, par le biais de plusieurs réunions orales, pour leur faire part de mes avancées et de me questionner en retour sur les points à faire évoluer. Je suis finalement parvenu à faire des présentations les plus claires possibles, dont je suis fier, car c'était un travail long et assez fastidieux. De mon point de vue de stagiaire, cet outil me paraissait comme une boîte noire verrouillée qui nécessitait de résoudre des énigmes afin de l'ouvrir.

Par ailleurs, la difficulté principale du projet résidait sur la compréhension d'un système inconnue dont les informations sont disséminées dans des dizaines de publications scientifiques pour la grande majorité en anglais et des milliers de lignes de code. De plus, la réalisation d'un premier programme d'essai m'a fait perdre beaucoup de temps (un mois), vu l'ampleur des informations qu'il fallait recueillir et trier.

Des difficultés supplémentaires résidaient sur la résolution de problèmes mathématiques et physiques qui m'ont demandé d'ajouter des tâches à réaliser sur mon diagramme de Gantt.

## **3. Compétences transversales et diagramme de Gantt**

Comme expliqué précédemment, mon stage s'est déroulé en grande autonomie. Pour réussir mon stage et avancer dans les temps tout en suivant une démarche de gestion de projet, j'ai établi un diagramme de Gantt (figure 48). Celui-ci récapitule l'ensemble des tâches à accomplir, leur date de commencement et le nombre de jours que je prévoyais (tableau 5). Cela m'a servi de fil conducteur et m'a permis de gérer les retards ou problèmes rencontrés avec sérénité sans avoir peur de manquer de temps pour les tâches qui me restaient à accomplir.

Sur la figure 48, je présente le temps réellement passé sur chacune des tâches. Nous constatons que j'ai passé beaucoup plus de temps que prévu sur certaines tâches dans la première moitié de mon stage. Elles correspondent notamment au temps passé en recherches d'informations et correction de bugs.

Ce stage de quatrième année, m'a permis de découvrir le domaine de la recherche, ce qui ne m'a pas déplu. Ayant eu plusieurs expériences professionnelles en job d'été depuis mes 17 ans dans diverses entreprises, j'ai pu observer la différence entre le domaine associatif, le métier d'opérateur, de technicien et de chercheur. Cela m'a permis de développer une compréhension approfondie des

dynamiques variées de chaque secteur. J'ai acquis des compétences polyvalentes, telles que la gestion de projet, la résolution de problèmes techniques et la conduite de recherches approfondies. Ces expériences m'ont également aidé à améliorer mes capacités d'adaptation, mon sens de l'organisation et ma communication interpersonnelle, des atouts indispensables pour réussir dans un environnement professionnel diversifié et en constante évolution.

<b>Diagramme de Gantt</b>			
	<b>Tâches</b>	<b>Date de début</b>	<b>Date de fin</b>
Installation de l'OS sur la RedPitaya		15/04/2024	19/04/2024
Prise en main de la RedPitaya		15/04/2024	21/04/2024
Développement d'une application avec interface web		21/04/2024	19/05/2024
Développement de la partie font-end		21/04/2024	19/05/2024
Développement de la partie back-end		21/04/2024	24/05/2024
Programmation script d'acquisition		24/05/2024	06/06/2024
Implémentation Filtre RII		28/05/2024	09/06/2024
Implémentation de la démodulation		01/06/2024	16/06/2024
Implémentation FFT discrète		09/06/2024	02/07/2024
Implémentation fenêtrage		10/06/2024	20/06/2024
Implémentation PID		11/06/2024	15/06/2024
Caractérisation et test FFT		08/06/2024	04/07/2024
Caractérisation et test du filtre RII		20/06/2024	11/07/2024
Caractérisation et test de la démodulation		14/06/2024	11/07/2024
Caractérisation temps d'exécution		24/06/2024	11/07/2024
Implémentation d'un module de détermination des figures de mérites d'un dispositif		11/07/24	25/07/2024
Test sur dispositif		18/07/24	26/07/2024
Écriture du rapport		26/07/24	03/08/2024

Tableau 5 : Répartition des tâches effectuées par jours

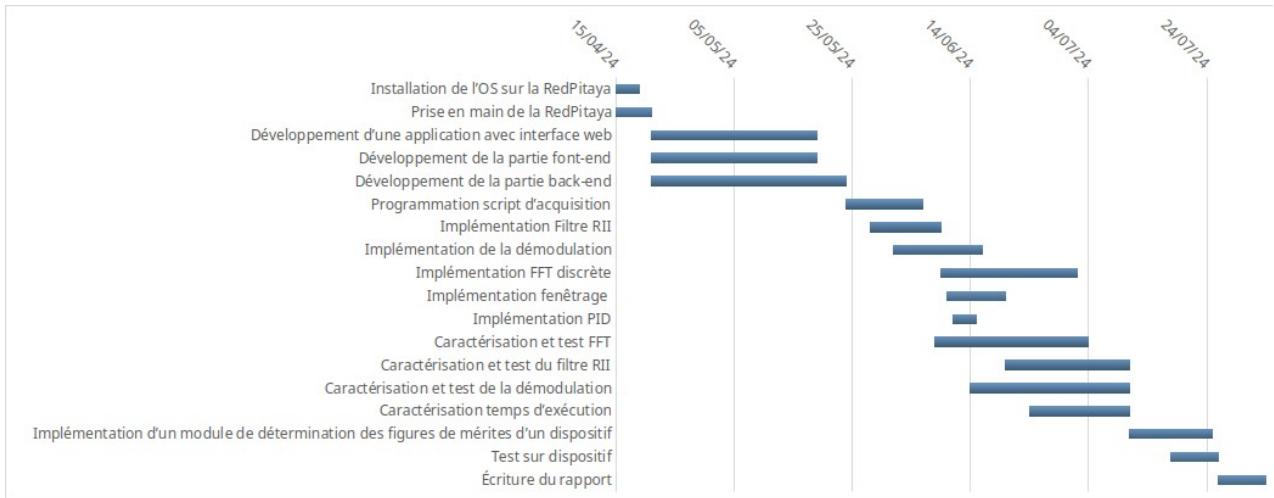


Figure 48 : Diagramme de Gantt

## 4. Conclusion

L'objectif de mon stage était la mise en opération d'un système de démodulation en temps réel sur une carte d'acquisition programmable Red Pitaya STEMlab 125-14 afin de caractériser les figures de mérite d'un capteur MEMS. Les contraintes techniques liées au temps réel n'ayant pu être résolue, cause de connaissance et de temps pour implémenter du FPGA. J'ai tout de même réussi à mettre en place un bloc algorithmique capable de réaliser cette tâche, qui s'est avéré suffisante pour le test sur dispositif.

## VI. Annexe

### 1. Concepts de base

Le but de cette partie n'est pas d'être un cours ou une étude mathématique du signal, mais un simple rappel des notions fondamentales des outils de base du traitement numérique du signal. Ce chapitre sera centré sur les éléments de base de représentation des signaux en relation avec les fonctionnalités développées dans ce projet.

#### 1.1 Classification des signaux

D'une façon générale, un signal est défini comme toute manifestation sous forme physiquement observable et mesurable d'un phénomène réel. Par exemple, les signaux visuels sont des ondes de lumière apportant une information à nos yeux. Les signaux sonores sont des fluctuations de la pression de l'air transportant un message à nos oreilles. Dans un sens plus limité, un signal est la représentation physique de l'information qu'une source émet vers un récepteur. Dans un sens mathématique, le signal est représenté par une fonction réelle ou complexe à une ou plusieurs dimensions d'une ou plusieurs variables réelles ou entières. Ainsi, il existe plusieurs types de signaux dont les plus utilisés en traitement des signaux sont les suivants:

- **Signal analogique :** le signal analogique est un signal dont les variables indépendantes de la représentation mathématique, sont continues. Souvent, ces signaux sont modélisés par des fonctions de la variable  $t$ . Le support temporel du signal  $x(t)$  peut être borné dans le temps.
- **Signal discret :** un signal discret appelé aussi signal échantillonné est un signal dont les variables indépendantes de la représentation mathématique sont discrètes.
- **Signal numérique :** c'est un signal discret, dont l'amplitude est aussi discrète. Un exemple de signal numérique est représenté sur la figure 49.

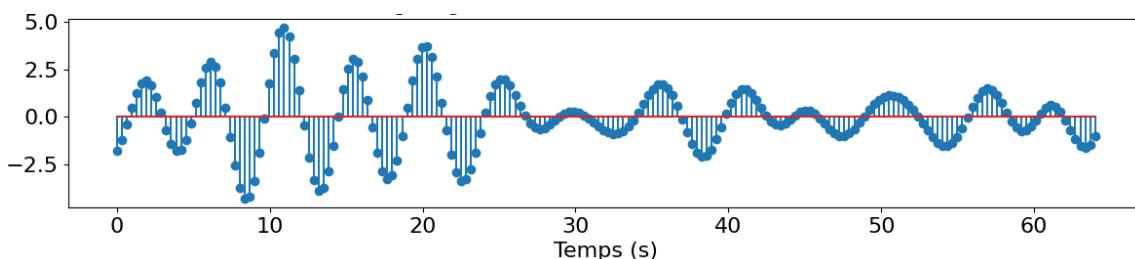


Figure 49 : Exemple de signal discret numérique

## 1.2 Traitement des signaux

La théorie du signal ainsi que son traitement sont étroitement liés : la théorie du signal modélise et décrit les signaux, tandis que le traitement des signaux interprète et élabore d'autres signaux à l'aide de dispositifs ou systèmes de traitement des signaux d'entrée. Un système de traitement de signaux agit sur un signal d'entrée et produit à sa sortie un autre signal sous une forme plus appropriée pour satisfaire certaines exigences techniques requises par une application donnée. Les systèmes de traitement peuvent être classés de la même façon que les signaux. Les systèmes échantillonnés opèrent sur des signaux échantillonnés et produisent des signaux échantillonnés. Les systèmes numériques opèrent sur des signaux numériques et produisent des signaux numériques. Le traitement numérique d'un signal implique une représentation appropriée, sous une représentation en virgule fixe ou en virgule flottante. Dans l'application développée les données sont représentées en virgule flottante 64 bits (appelé 'double' en langage informatique C et C++).

## 1.3 Numérisation des signaux

Tous les signaux analysés dans ce projet sont de type analogique, si on veut les analyser sous forme numérique, il faut d'abord convertir ces signaux en une suite de nombres codés sous forme binaire. Cette transformation effectuée une **conversion analogique à numérique**. La conversion d'un signal analogique à un signal numérique s'appuie sur une double approximation. La première approximation s'effectue dans l'espace des temps: le signal  $x(t)$  est remplacé par ses valeurs  $x(nT)$  à des instants multiples entiers d'une durée  $T$ . C'est ce qu'on appelle l'**échantillonnage**. La deuxième approximation s'effectue au niveau des amplitudes. Chaque valeur  $x(nT)$  est approchée par un multiple entier d'un pas élémentaire noté  $q$ , c'est ce qu'on appelle la **quantification**.

## 1.4 Échantillonnage et théorème

L'échantillonnage est défini comme le prélèvement régulier de valeurs  $x(n T_e)$  à des instants multiples entiers d'une durée  $T_e$ . Cette durée représente la période d'échantillonnage d'un signal analogique fonction du temps  $x(t)$ . Quand un signal analogique quelconque  $x(t)$  est échantillonné, il est essentiel de savoir s'il est possible de le reconstituer au besoin, à partir des échantillons  $x(n T_e)$ . L'intérêt principal du théorème d'échantillonnage est d'établir la condition que doit satisfaire la fréquence d'échantillonnage  $f_e$  pour avoir la possibilité de reconstitution du signal d'origine sans perte d'information. Un théorème énonce les conditions que doit respecter la suite des échantillons  $x(n T_e)$  d'un signal pour représenter correctement ce signal. Un signal est considéré correctement représenté par les échantillons  $x(n T_e)$  s'il est possible de reconstruire le signal d'origine  $x(t)$  à partir de ses échantillons  $x(n T_e)$  prélevés avec la période  $T_e$ . Pour cela, il faut que la fréquence de répétition  $f_e$  soit supérieure ou égale à deux fois la fréquence maximale. C'est le **théorème de Nyquist**.

## 1.5 La quantification

Le traitement numérique des signaux nécessite une représentation d'un nombre fini de valeurs discrètes, il faut donc quantifier le signal. Quantifier un signal  $x(t)$  revient à une approximation de chaque valeur de ce signal par un multiple entier d'une quantité élémentaire, notée  $q$  et appelée pas de quantification. Lorsque  $q$  est constant et indépendant de l'amplitude, on parle d'une quantification uniforme; c'est le cas de la représentation des nombres en virgule fixe. Par contre, quand  $q$  est approximativement proportionnel à l'amplitude du signal, la représentation numérique se fait en virgule flottante. Pour un système numérique le quantum est l'inverse de la puissance de 2 du nombre de bits utilisées pour coder l'information analogique.

## 1.6 Transformée de Fourier

La transformation de Fourier, inventée par le physicien et mathématicien français Joseph Fourier au 18e siècle, est un outil mathématique révolutionnaire permettant de représenter les signaux dans le domaine fréquentiel afin de connaître les fréquences qui le compose. Caractériser un signal par son spectre de fréquence permet notamment de mettre en évidence l'importance de l'harmonique fondamental ainsi que la décroissance plus ou moins rapide de l'amplitude des tons de rang plus élevé. Comme le montre la formule suivante la transformée de Fourier transforme un signal réel  $\mathbb{R}$  en un signal à valeur du domaine complexe  $\mathbb{C}$  :

$$F(f): s \rightarrow \hat{f}(s) = \int_{-\infty}^{+\infty} f(x) e^{-isx} dx$$

Étudiant :	Charrière Clément
Courriel :	<a href="mailto:clement.charriere@etu.univ-grenoble-alpes.fr">clement.charriere@etu.univ-grenoble-alpes.fr</a>
Année d'étude dans la spécialité :	IESE4 (2023-2024)
Entreprise :	TIMA
Adresse complète :	46 Avenue Félix Viallet 38 000 GRENOBLE FRANCE
Téléphone :	+33 4 76 57 50 79
Responsable administratif :	Mr CAMART Jean-Christophe
Fonction :	Administrateur provisoire
Téléphone :	04 57 42 21 42
Tuteur de stage :	DEFOORT Martial
Fonction :	chercheur CNRS
Téléphone :	+33 4 76 57 46 12
Courriel :	<a href="mailto:martial.defoort@univ-grenoble-alpes.fr">martial.defoort@univ-grenoble-alpes.fr</a>
Enseignant-référent :	TORU Sylvain
Téléphone :	+33 4 76 82 79 15
Courriel :	<a href="mailto:sylvain.toru@univ-grenoble-alpes.fr">sylvain.toru@univ-grenoble-alpes.fr</a>
Titre : Développement d'un module RED PITAYA pour la caractérisation de capteurs MEMS	
Résumé : Implémentation de fonctionnalités de traitement du signal afin de réaliser un système de caractérisation des figures de mérite d'un dispositif MEMS à l'aide d'un module d'acquisition RED Pitaya STEM 125-14.	