Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts
Western Switzerland

Filière Systèmes industriels

Orientation Infotronics

Travail de bachelor
Diplôme 2017

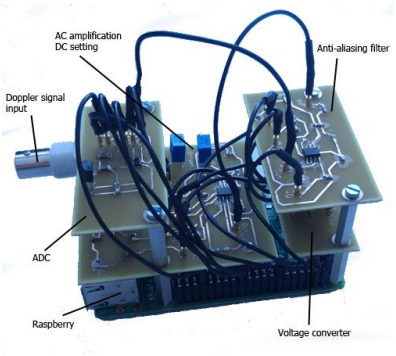
David Comte

Implémentation embarquée d'un signal Doppler

Professeur
Django Kandaswamy

Expert
Charles Riva

Date de la remise du rapport
18.08.2017



Implémentation embarquée d'un signal Doppler

Diplômant David Comte

Objectif du projet

Développer un système d'acquisition et de traitement d'un signal Doppler

Méthodes | Expériences | Résultats

Y-YBar conçoit des appareils médicaux utiles dans la recherche en ophtalmologie. Les appareils qu'elle développe sont basés sur la LDF (Laser Doppler Flowmetry). La LDF est une technique de vélocimétrie utilisant l'effet Doppler pour mesurer le flux sanguin dans l'œil.

En ophtalmologie, la LDF permet, entre autres, de déterminer des paramètres hémodynamiques utiles pour la détection de maladies dans l'œil. Les paramètres hémodynamiques sont : ChBVel qui est la vitesse du sang, ChBVol qui est le volume de sang et ChBF qui est le flux du sang.

Le système développé permet l'acquisition et le traitement d'un signal Doppler sur une carte embarquée. Une interface web pilote le système. Elle permet : De faire des mesures, de calculer les paramètres hémodynamiques et de récupérer les résultats.

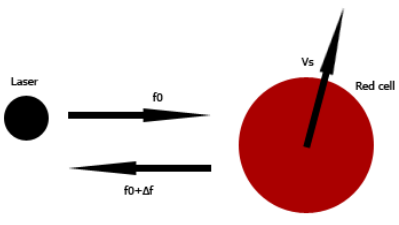
Travail de diplôme
| édition 2017 |

Filière
Systèmes industriels

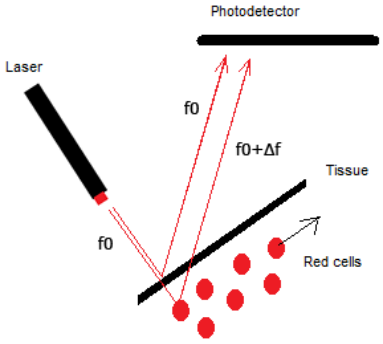
Domaine d'application
Infotronics

Professeur responsable
Djano Kandaswamy
django.kandaswamy@hevs.ch

Partenaire
Y-YBar
Martial Geiser
martial.geiser@hevs.ch



Effet Doppler - Un signal lumineux réfléchi par un objet en mouvement est décalé en fréquence. Le décalage est fonction de la vitesse de l'objet



LDF - Le signal Doppler est généré par un photodétecteur. Il contient plusieurs fréquences dépendant de la vitesse des cellules rouges

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mon travail de diplôme est mandaté par l'entreprise Y-YBar. Y-YBar produit des appareils médicaux utiles dans la recherche en ophtalmologie. Les appareils qu'elle produit sont basés sur la technique de fluxmétrie Doppler par laser, appelée LDF (Laser Doppler Flowmetry) en anglais.

La LDF est, entre autre, utilisée dans la fluxmétrie dans l'oeil. La fluxmétrie dans l'oeil permet de déduire des paramètres hémodynamiques. Ces paramètres sont utiles pour la détection de maladies dans l'oeil. Ils sont également utiles dans la recherche scientifique.

M. Francesco Marazzi, un élève de l'université de Modène et de Reggio d'Émilie, a développé un système permettant d'acquérir et de traiter un signal provenant de la LDF, appelé signal Doppler. L'acquisition du signal est fait sur une carte embarquée tandis que le traitement de ce dernier, calculant les paramètres hémodynamiques, est fait sur un ordinateur.

Mon travail consiste à créer un **nouveau** système d'acquisition et de traitement d'un signal Doppler. Les fonctionnalités du système devront être proches du système développé par M. Francesco Marazzi, à l'exception que le calcul des paramètres hémodynamiques devra se faire directement sur la carte embarquée.

1.1 Laser Doppler Flowmetry

La LDF est une technique basée sur l'effet Doppler. Elle est, entre autre, utilisée dans la fluxmétrie dans l'oeil. Le signal Doppler à acquérir et à traiter, pour obtenir les paramètres hémodynamiques, provient directement de cette technique.

Un laser émet un signal optique avec une fréquence de base, f0 (Figure 1.1). Le signal, réfléchi par l'oeil, est décalé d'une fréquence Δf. Lorsque le signal est réfléchi sur le tissu de l'oeil, la fréquence Δf vaut zéro, ce n'est pas le cas lorsqu'il est réfléchi par les cellules rouges. Ces décalages en fréquence sont directement explicables par l'effet Doppler (Chapitre 1.2). Le signal est intercepté sur un photodétecteur qui le convertit en un signal électrique. C'est ce signal électrique qui appelé signal Doppler.

1.2 Effet Doppler

L'effet Doppler permet d'expliquer les décalages en fréquence que subit le signal optique lorsqu'il est réfléchi (Figure 1.1). Le signal optique est réfléchi sur une cellule rouge, considérée comme la source. Du fait de la vitesse de déplacement de cette dernière, le signal perçu par le récepteur, dans notre cas, un photodétecteur, est décalé d'une fréquence Δf.

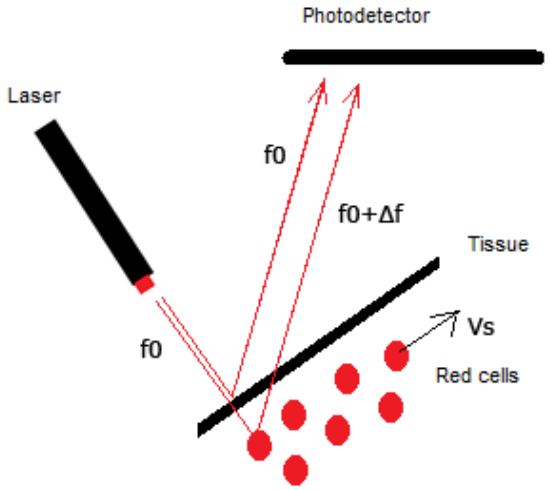


FIGURE 1.1: Principe de la LDF

La fréquence perçue par le récepteur est :

$$f = \frac{c + Vr}{c + Vs} * f0 \tag{1.1}$$

Avec c : Vitesse de la lumière, Vr : Vitesse du récepteur, Vs : Vitesse de la source et f0 : Fréquence initiale

Le décalage en fréquence s'exprime par :

$$\Delta f = f - f0 = \frac{\Delta V}{c} * f0 = \frac{Vr - Vs}{c} * f0 \tag{1.2}$$

Comme la vitesse du récepteur est nulle, le décalage en fréquence devient :

$$\Delta f = \frac{-Vs}{c} * f0 \tag{1.3}$$

1.3 Traitement du signal Doppler

Le photodétecteur récolte plusieurs signaux optiques en même temps. Ces signaux optiques n'ont pas tous la même fréquence. De ce fait, le signal Doppler généré par le photodétecteur possède plusieurs fréquences. Sa bande de fréquence **utile**, dépendant de la vitesse des cellules rouges (donc du sang), se situe entre 30Hz et 30kHz (Source [4]).

Les plus grandes fréquences du signal Doppler sont dues aux cellules rouges. tandis que les plus basses sont dues au tissu (Figure 1.2).

Un signal Doppler issu du photodétecteur a typiquement une composante DC qui se situe entre 0V et 1V. Sa composante AC est d'environ 10mVRMS (Figure 1.3).

Le spectre de puissance d'un signal Doppler, P(f), à une forme décroissante (Figure 1.4).

Il permet de calculer les paramètres hémodynamiques :

- **ChBVel** est la vitesse du sang. Cette valeur est proportionnelle à la moyenne des vitesses des cellules rouges, qui elle, est proportionnelle à la moyenne des

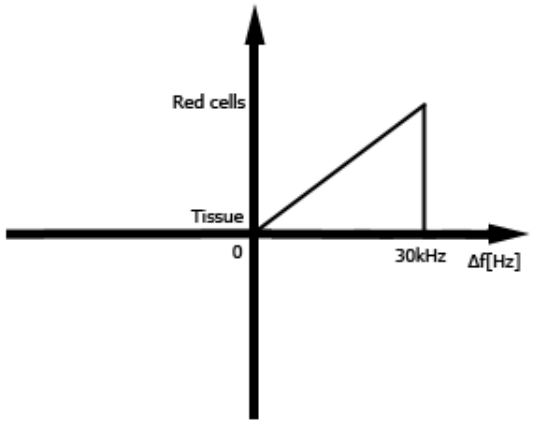


FIGURE 1.2: Bande de fréquence du signal Doppler

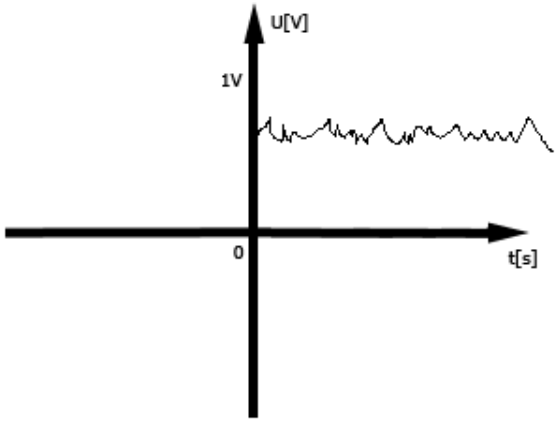


FIGURE 1.3: Signal Doppler

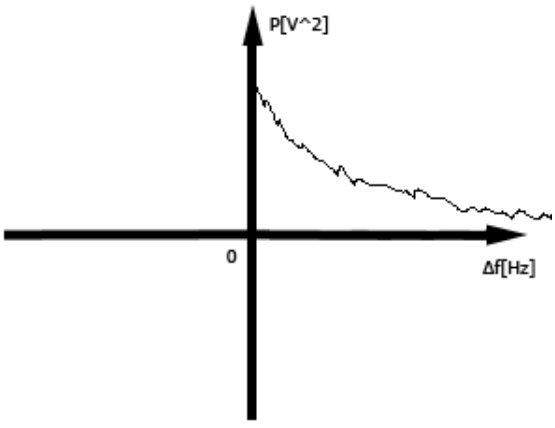


FIGURE 1.4: Spectre de puissance du signal Doppler

décalages en fréquences :

$$ChBVel = \frac{\int fP(f)df}{\int P(f)df} \tag{1.4}$$

- **ChBVol** est le volume de sang. Cette valeur est proportionnelle au nombre de cellules rouges, qui lui, est proportionnel aux amplitudes et au nombre de fréquences :

$$ChBVol = \frac{1}{DC^2} * \int P(f)df \tag{1.5}$$

- **ChBF** est le flux du sang. Cette valeur est proportionnelle à la multiplication des deux précédents paramètres :

$$ChBF = ChBVel * ChBVol \tag{1.6}$$

Chapter 2

Système existant

Ce chapitre décrit le système existant, développé par M. Francesco Marazzi. Il permet de comprendre quelles fonctionnalités son système contient, et quelles sont les fonctionnalités souhaitées, en plus de celles existantes, pour le nouveau système.

2.1 Schéma bloc

Son système (Figure 2.1) comporte quatre étages distincts :

- **Une électronique** qui amplifie la composante AC du signal Doppler et coupe les fréquences supérieures à 40kHz (la bande de fréquence utile du signal va de 30Hz à 30kHz, mais les fréquences entre 30kHz et 40kHz sont utiles pour calculer le bruit)
- **Une Discovery board** qui échantillonne le signal et sauvegarde temporairement ses valeurs dans un buffer
- **Un Raspberry** qui met à disposition une interface web de pilotage du système et de récupération des valeurs
- **Un ordinateur client** qui calcule les paramètres hémodynamiques

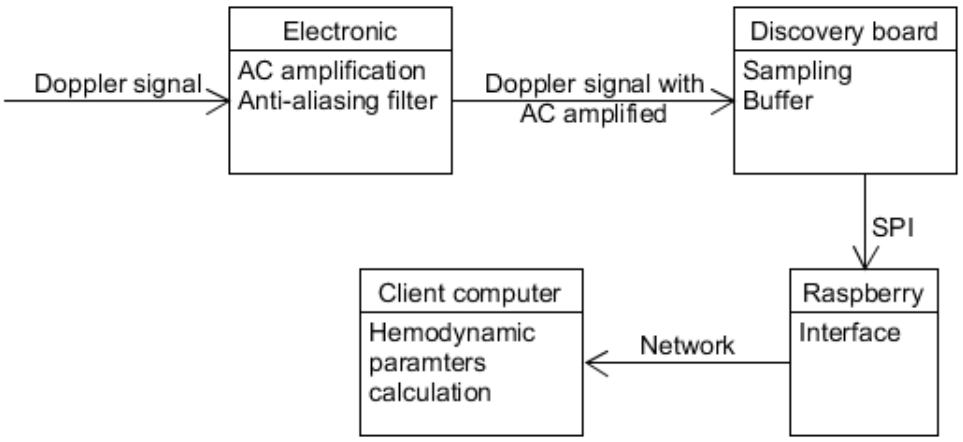


FIGURE 2.1: Schéma bloc du système existant

Notes : L'amplification de la composante AC permet d'améliorer l'erreur de quantification (Source [3]). Les fréquences supérieures à 40kHz sont coupées afin d'éviter des problèmes de repliement.

2.2 Electronique

L'électronique est alimentée en +5V et +-12V. Elle permet (Figure 2.2) :

- D'inverser le signal Doppler
- De choisir entre le signal inversé ou non inversé grâce à un jumper
- D'amplifier la composante AC du signal d'un facteur dépendant d'une résistance interchangeable
- De régler sa composante DC grâce à un potentiomètre
- De couper les fréquences du signal supérieures à 40kHz à l'aide d'un filtre passe-bas numérique

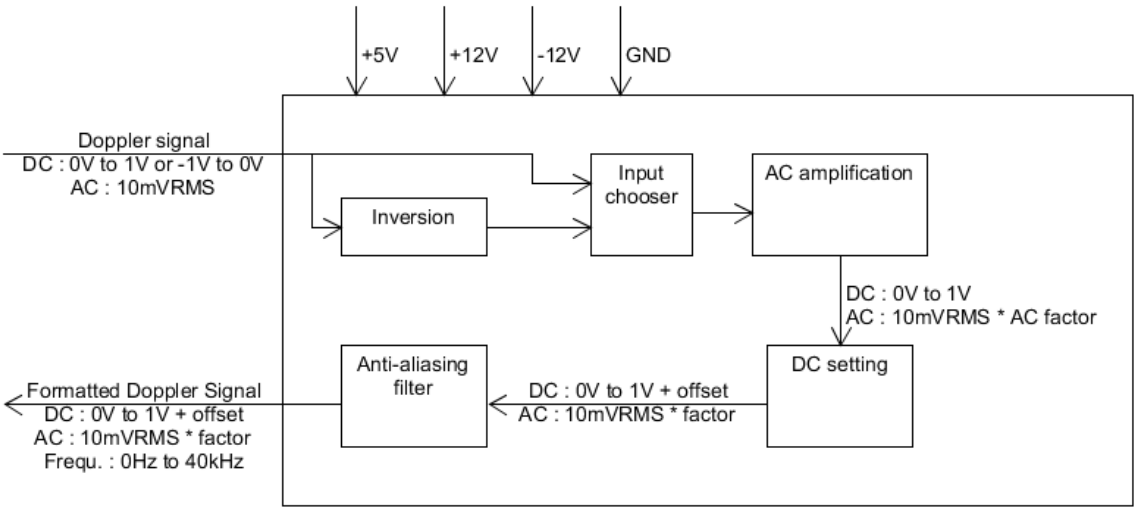


FIGURE 2.2: Schéma bloc de l'électronique

Améliorations souhaitées pour le nouveau système :

- Alimentation unique en +5V (utilisable depuis le Raspberry) de l'électronique
- Réglage du facteur d'amplification grâce à un potentiomètre
- Coupure des fréquences avec un filtre analogique (un filtre numérique utilise une horloge qui peut apporter des problèmes de fréquences parasites)

Notes : L'étage d'inversion permet de réinverser le signal Doppler lorsqu'il est inversé en entrée, se situant entre -1V et 0V. Le réglage de la composante DC permet de supprimer le bruit de fond.

2.3 Discovery board

Les fonctionnalités de la Discovery board sont (Figure 2.3) :

- Echantillonner le signal Doppler mis en forme par l'électronique
- Sauvegarder les valeurs dans un buffer

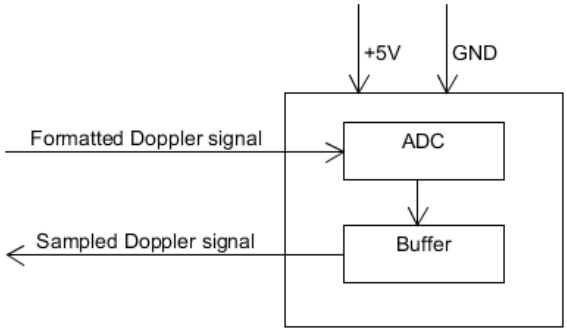


FIGURE 2.3: Schéma bloc de la Discovery board

- Envoyer les valeurs via SPI (Serial Peripheral Interface) au Raspberry

Une Discovery board est utilisée parce que le système d'exploitation du Raspberry, Raspbian, n'est pas temps réel. Ainsi, il est difficile de garantir une fréquence d'échantillonnage constante lorsqu'un ADC (Analog to Digital Converter) est directement branché sur le Raspberry. L'utilisation du microcontrôleur de la Discovery board, en amont du Raspberry, pallie à ce problème. Pour ce qui est du buffer, il est nécessaire, car la vitesse d'échantillonnage du microcontrôleur et la vitesse de réception du Raspberry ne sont pas les mêmes.

Le buffer est rempli de la manière suivante (Figure 2.4) : Les valeurs y sont sauvegardées une à une. Une fois le buffer rempli à moitié, son contenu est envoyé via SPI au Raspberry. Durant cette étape, le buffer continue de se remplir. Et ainsi de suite.

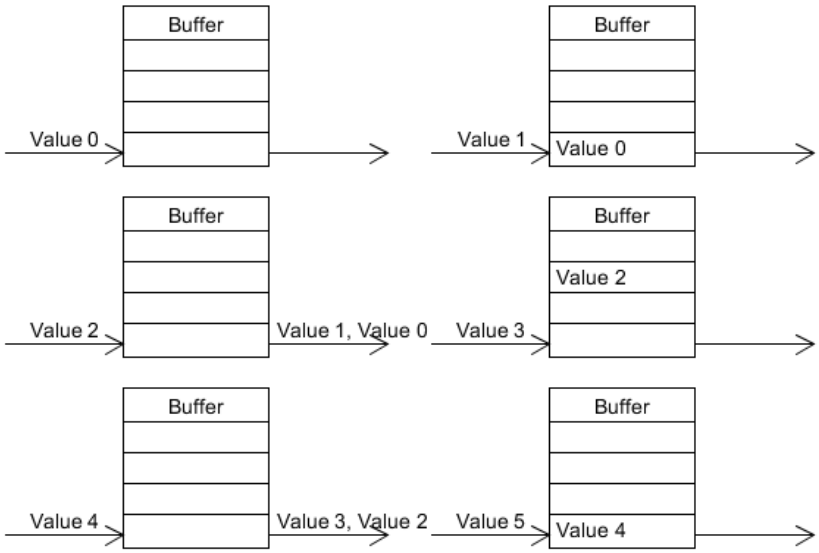


FIGURE 2.4: Buffer de la Discovery board

Améliorations souhaitées pour le nouveau système :

- Ne plus utiliser de Discovery board afin d'améliorer la compacité et simplifier la partie électronique
- Brancher un ADC directement sur le Raspberry

2.4 Raspberry

Le Raspberry permet (Figure 2.5) :

- De recevoir les valeurs de la Discovery board via SPI
- D'enregistrer les valeurs dans un fichier CSV (Comma Separated Values)
- De démarrer une mesure depuis une interface web
- De récupérer des mesures depuis une interface web

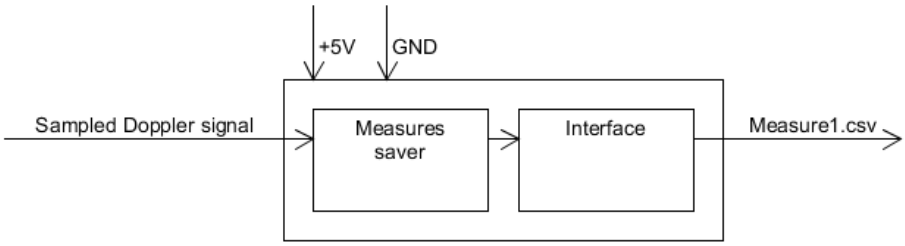


FIGURE 2.5: Schéma bloc du Raspberry

Amélioration souhaitées pour le nouveau système :

- Recevoir les valeurs depuis un ADC via SPI
- Envoyer en direct les valeurs sur l'interface web (ce qui permet d'avoir une **idée** du signal afin de régler la composante DC et le facteur d'amplification)
- Déclencher des mesures sous forme d'expérience, en renseignant différents paramètres à l'aide d'une interface web comme le nom du patient, son prénom, etc.
- Calculer les paramètres hémodynamiques directement sur le Raspberry
- Avoir la possibilité de déclencher une mesure sur plusieurs Raspberry en simultané

2.5 Ordinateur client

L'ordinateur client permet (Figure 2.6) :

- De calculer les paramètres hémodynamiques

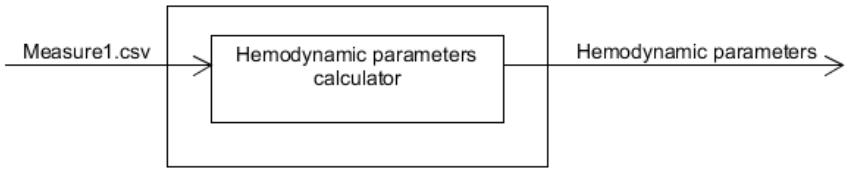


FIGURE 2.6: Schéma bloc de l'ordinateur client

Améliorations souhaitées :

- Suppression du programme de calcul des paramètres hémodynamiques de l'ordinateur client (ce dernier devant se faire sur le Raspberry)

Chapter 3

Nouveau système

Ce chapitre présente le nouveau système développé. Il explique le travail effectué, permet de comprendre quelles sont les fonctionnalités du nouveau système, et comment elles ont été implémentées.

3.1 Schéma bloc

Le nouveau système (Annexe A.1) comprend une électronique qui amplifie la composante AC du signal Doppler et coupe les fréquences supérieures à 40kHz (Figure 3.1). Un ADC qui échantillonne le signal mis en forme et transmet les valeurs via SPI à un Raspberry. Ce même Raspberry embarque une interface web permettant de déclencher des mesures et de calculer les paramètres hémodynamiques. Lesquels sont récupérables par un ordinateur client depuis l'interface web.

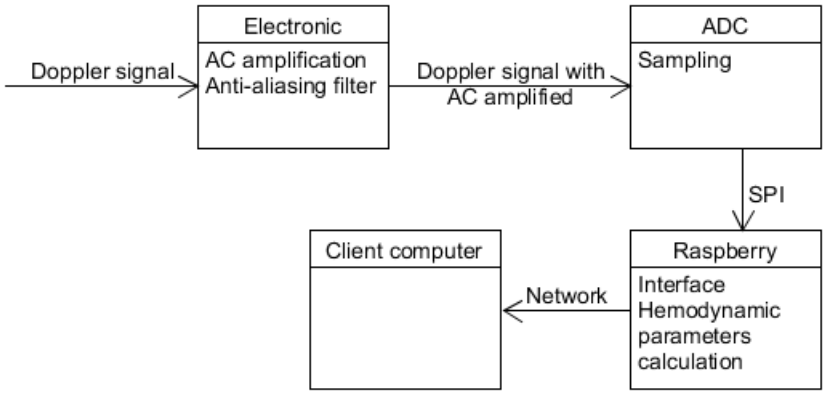


FIGURE 3.1: Schéma bloc du nouveau système

3.2 Electronique

Dans le cadre de la nouvelle électronique (Figure 3.2), un convertisseur de tension permet de n'avoir qu'une seule alimentation pour tout le circuit. Il génère des tensions de +-12V à partir d'une tension de +5V. Un circuit inverse le signal d'entrée. Un jumper permet, ensuite, de choisir entre l'entrée inversée ou non inversée. La composante AC du signal est amplifiée, et sa composante DC est réglée. De plus, les fréquences supérieures à 40kHz sont coupées.

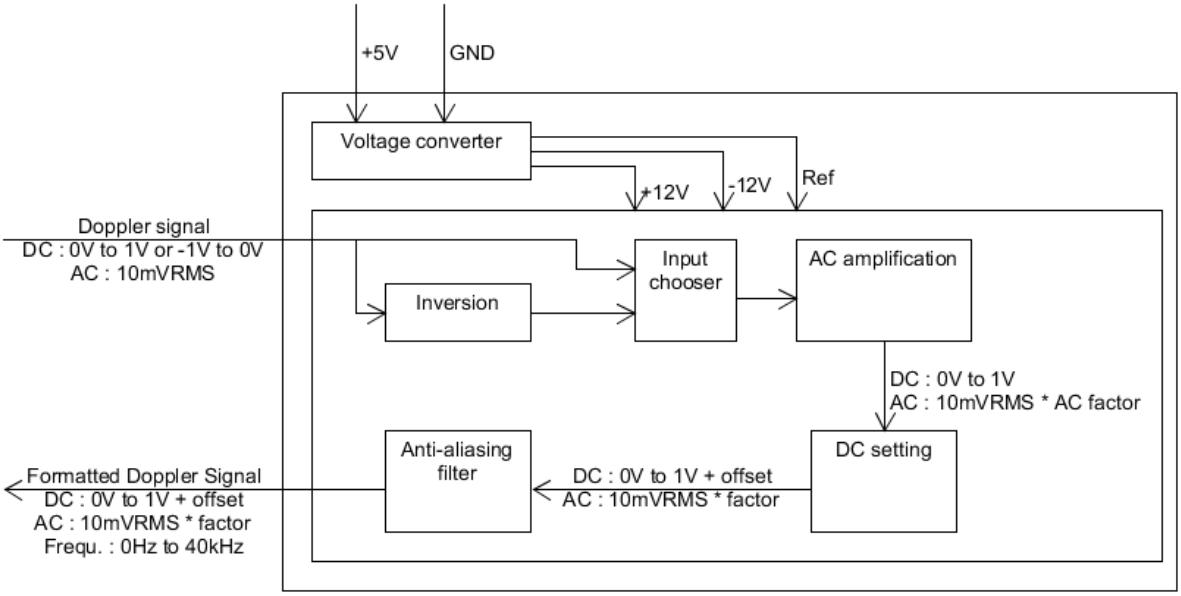


FIGURE 3.2: Schéma bloc de l'électronique

3.2.1 Convertisseur de tension

Le convertisseur de tension simplifie les branchements. Ainsi, l'électronique est alimentée directement avec une tension de +5V prise sur une des pins du Raspberry. Le montage a été fait selon la note d'application du fabricant (Annexe A.2).

3.2.2 Inverseur

L'inverseur trouve son utilité lorsque le signal Doppler est inversé en entrée et se situant entre -1V et 0V. Il est réalisé grâce à un AOP (Amplificateur Opérationnel) monté en inverseur (Figure 3.3).

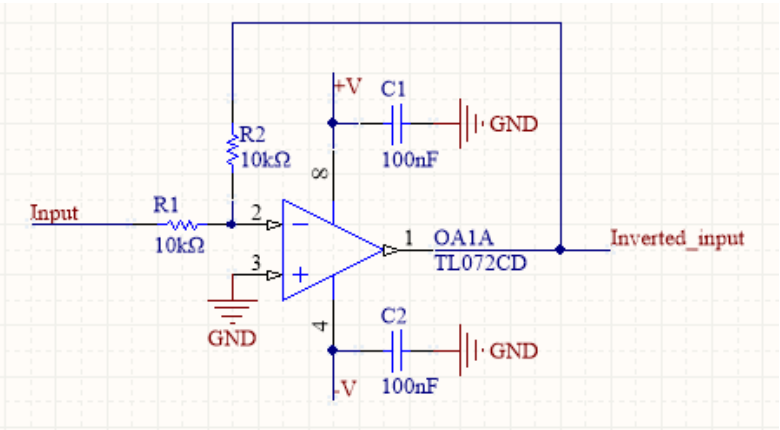


FIGURE 3.3: Montage de l'amplificateur inverseur

La tension de sortie est :

$$Inverted\_input = \frac{-R2}{R1} * Input = \frac{-10k}{10k} * Input = -Input \tag{3.1}$$

Notes : Des condensateurs de découplage de 100nF ont été utilisés pour chaque composant actif. Ceci permet de stabiliser la tension d'alimentation de ces derniers. L'AOP utilisé est un TL072. C'est un modèle populaire dans le domaine de l'audio. Il a un faible bruit et possède une haute vitesse de balayage.

3.2.3 Amplificateur AC

La composante AC du signal Doppler est de l'ordre d'une dizaine de mVRMS. L'amplificateur de la composante AC (Figure 3.4) permet de diminuer l'erreur de quantification.

Un filtre passe-haut isole la partie AC du signal. Celle-ci est amplifiée par un AOP monté en non inverseur. Le facteur d'amplification est réglé par un potentiomètre.

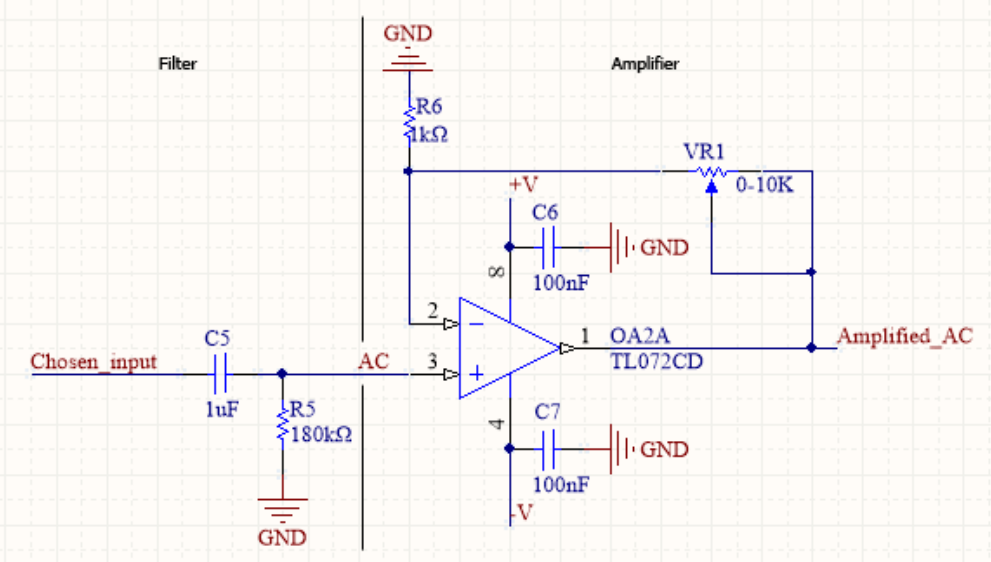


FIGURE 3.4: Montage de l'amplificateur AC

La fréquence de coupure du filtre est :

$$fc = \frac{1}{2 * \pi * R5 * C5} = \frac{1}{2 * \pi * 180k * 1u} = 0.88Hz \tag{3.2}$$

La sortie de l'amplificateur vaut :

$$Amplified\_AC = AC * (1 + \frac{VR1}{R6}) \tag{3.3}$$

Dans notre cas, la sortie minimale et maximale vaut :

$$Amplified\_ACmin = AC * (1 + \frac{VR1min}{1k}) = AC * (1 + \frac{0}{1k}) = AC \tag{3.4}$$

$$Amplified\_ACmax = AC * (1 + \frac{VR1max}{1k}) = AC * (1 + \frac{10k}{1k}) = AC * 11 \tag{3.5}$$

Afin de se rendre compte de l'utilité de l'amplificateur AC, il est intéressant d'estimer la précision gagnée grâce à ce dernier. Le pas de quantification de l'ADC est :

$$step = \frac{range}{2^n} = \frac{3.3}{2^{12}} = 0.8mV \tag{3.6}$$

Avec range : La plage d'entrée de l'ADC et n : Le nombre de bits de l'ADC

Lorsque le signal n'est pas amplifié, sa composante AC est d'environ 10 mvRMS. Admettons 20mV pour les calculs. Cela fait un nombre de points équivalent à :

$$Npoints = \frac{2 * amplitude}{step} = \frac{2 * 20m}{0.08m} = 50 \tag{3.7}$$

Lorsque le signal est amplifié d'un facteur 11, le nombre de points devient :

$$Npoints = \frac{2 * amplitude * factor}{step} = \frac{2 * 20m * 11}{0.08m} = 550 \tag{3.8}$$

Une fois échantillonné, le signal est divisé par 11 de façon logicielle. Le pas de quantification théorique est amélioré d'un facteur 11 :

$$theorical\_step = \frac{2 * amplitude}{Npoints} = \frac{2 * 20m}{550} \tag{3.9}$$

$$= \frac{step}{factor} = \frac{0.8m}{11} \tag{3.10}$$

$$= 0.073mV \tag{3.11}$$

Ce pas théorique correspond à un ADC avec un nombre théorique de bits de :

$$theorical\_n = log_2(\frac{range}{theorical\_step}) = log_2(\frac{3.3}{0.073m}) = 15.46 \tag{3.12}$$

Le nombre théorique de bits gagnés est :

$$gain\_n = log_2(\frac{range}{theorical\_step}) - n = log_2(\frac{3.3}{0.073m}) - 12 \tag{3.13}$$

$$= log_2(\frac{range}{step/factor}) - n = log_2(\frac{3.3}{0.8m/11}) - 12 \tag{3.14}$$

$$= 3.46 \tag{3.15}$$

3.2.4 Régleur DC

Un bruit de fond est possible au niveau de l'électronique. Typiquement, l'utilisation d'amplificateurs opérationnels introduit des offsets DC. Le régleur DC (Figure 3.5) permet de supprimer ce genre d'offsets.

Un potentiomètre ajuste une tension de réglage, nommé offset. Cette tension est additionnée grâce à un AOP monté en additionneur.

La tension de réglage, ou tension d'offset, vaut :

$$Offset = [-V, +V] = [-12V, +12V] \tag{3.16}$$

La sortie de l'additionneur est :

$$Ouput = \frac{R11 + R10}{n\_inputs * R10} * (DC + Amplified\_AC + Offset) \tag{3.17}$$

$$= \frac{20k + 10k}{3 * 10k} * (DC + Amplified\_AC + Offset) \tag{3.18}$$

$$= DC + Amplified\_AC + Offset \tag{3.19}$$

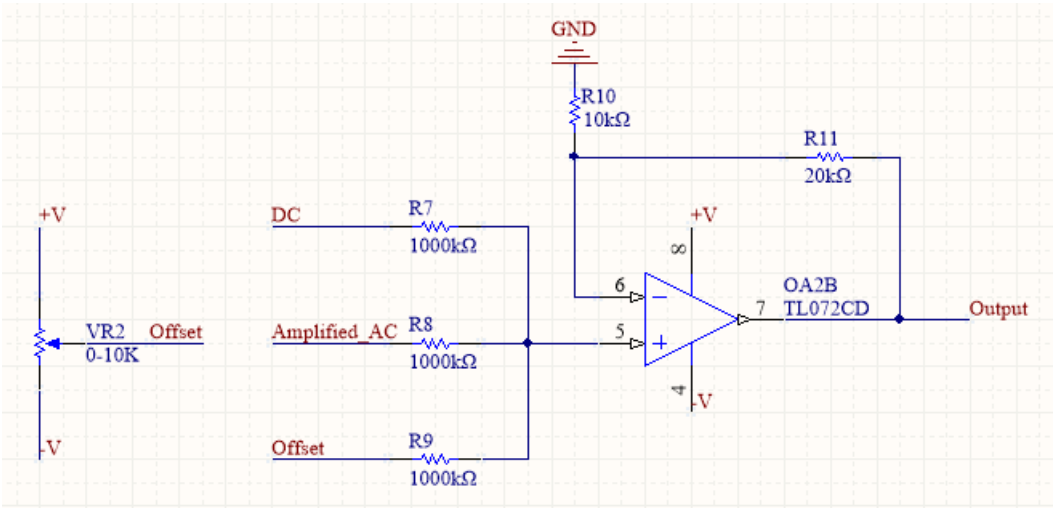


FIGURE 3.5: Montage du régleur DC

L'isolation de la partie DC du signal Doppler se fait grâce à un filtre passe-bas de deuxième ordre. Il est réalisé à l'aide d'un AOP monté en topologie Sallen et Key (Figure 3.6), choisie pour sa simplicité.

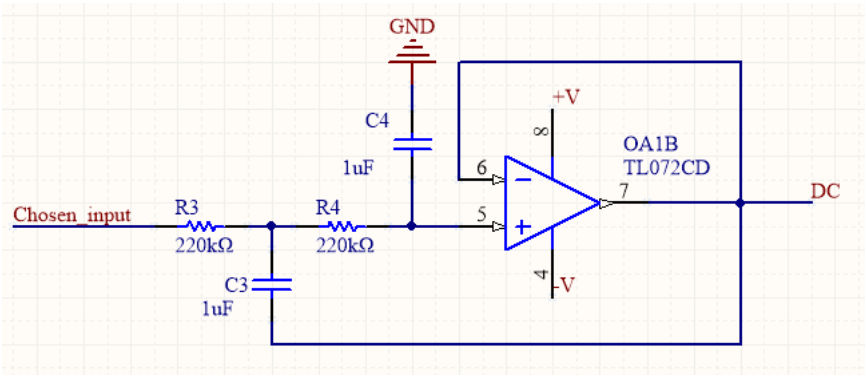


FIGURE 3.6: Montage de l'isolateur DC

La fréquence de coupure du filtre est :

$$fc = \frac{1}{2 * \pi * \sqrt{R3 * R4 * C3 * C4}} = \frac{1}{2 * \pi * \sqrt{220k * 220k * 1u * 1u}} = 0.72Hz \quad (3.20)$$

3.2.5 Filtre anti-repliement

Des problèmes de repliement sont possibles lors de l'échantillonnage, un filtre anti-repliement est inséré pour palier à ceux-ci (Figure 3.7).

Les fréquences utiles du signal Doppler vont de 30Hz à 30kHz. La fréquence de coupure du filtre est fixée à 40kHz. Admettons que le signal ait des fréquences allant au delà de 40kHz et que la fréquence d'échantillonnage soit fixée à 80kHz. Comme l'échantillonnage déplace le spectre du signal autour de multiples entiers de la fréquence d'échantillonnage, il en résulte un chevauchement lorsqu'aucun filtre d'anti-repliement n'est utilisé. Ce n'est pas le cas lorsqu'un filtre d'anti-repliement est utilisé, le théorème de Nyquist-Shannon étant réspecté :

$$fsmin = 2 * fmax \quad (3.21)$$

Où fsmin : Fréquence minimale d'échantillonnage et fmax : Fréquence maximale du signal à échantillonner.

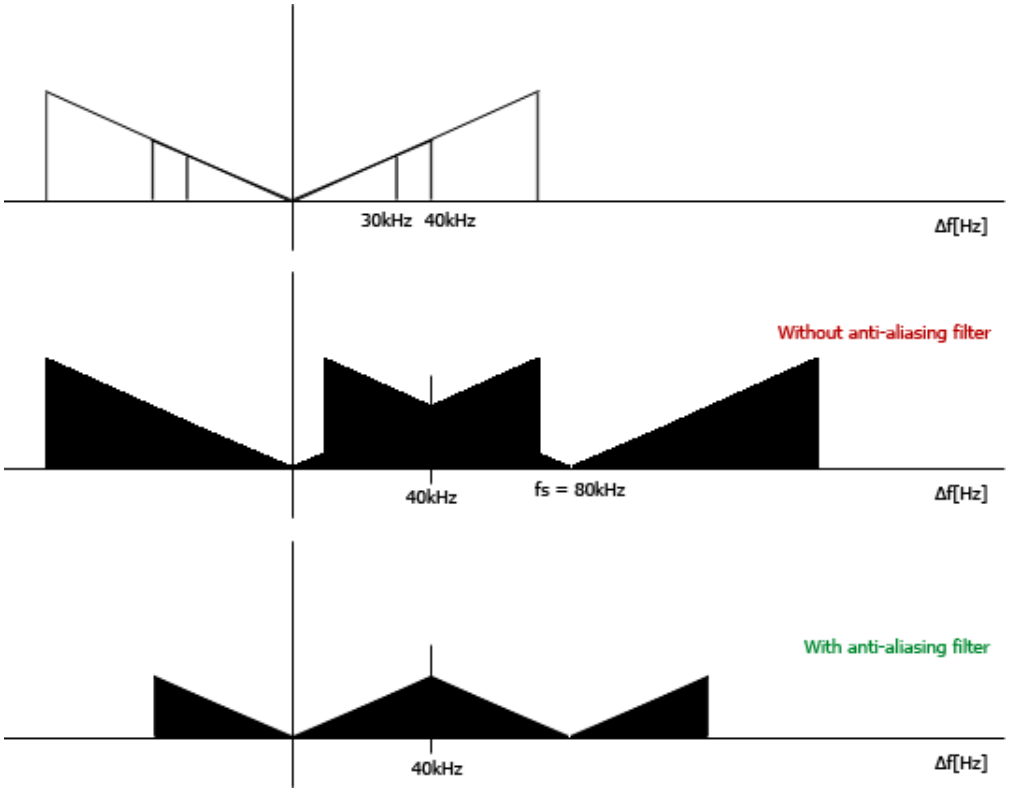


FIGURE 3.7: But d'un filtre anti-repliement

Un filtre passe-bas d'ordre 4 est utilisé pour éviter ces problèmes. Il coupe les fréquences supérieures à 40kHz. Il est réalisé en cascadant deux étages d'AOP monté en topologie de type Sallen et Key (Annexe A.3). La réponse en fréquence choisie est de type Butterworth afin de posséder un gain aussi constant que possible dans la bande passante. Un des deux étages est illustré en figure 3.8.

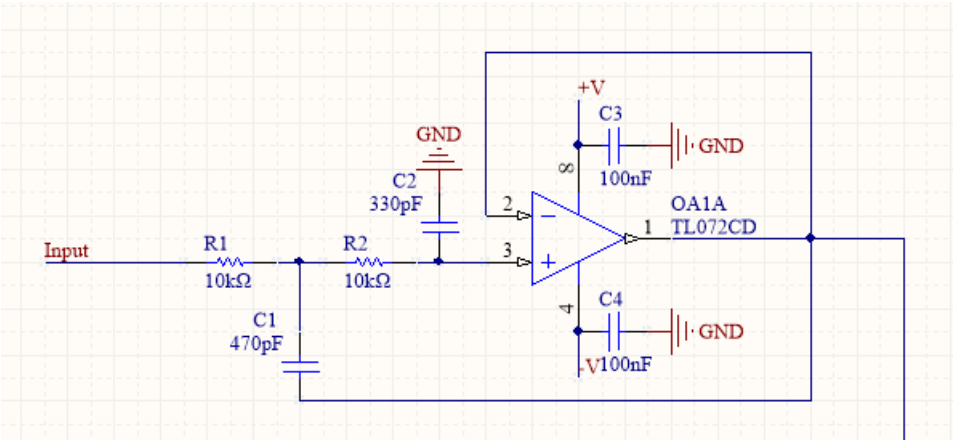


FIGURE 3.8: Montage du premier étage du filtre anti-repliement

Les composants ont été calculés grâce à un outil en ligne (Source [8]).

3.3 Convertisseur analogique-numérique

Le Raspberry ne possède pas de convertisseur analogique-numérique. Il est nécessaire d'utiliser un ADC externe pour échantillonner le signal Doppler. L'ADC choisi a été monté selon la fiche technique du fabricant (Annexe A.4).

L'ADC choisi est un ADC121S021. C'est un ADC 12bits permettant des vitesses d'échantillonnage jusqu'à 200kSPS (Kilo Samples Per Second). C'est suffisant, sachant que le signal Doppler mis en forme a une bande de fréquence allant de 0Hz à 40kHz, et donc, que la fréquence d'échantillonnage **minimale** doit être de 80kHz.

Le Raspberry, en mode SPI (Figure 3.9), choisit quand échantillonner le signal en activant la pin CS (Chip Select). Puis, l'ADC envoie la valeur échantillonnée grâce à la pin MISO (Master In Slave Out). La cadence de réception des bits (6.250MHz) de la valeur est donnée par la pin SCLK (Serial Clock).

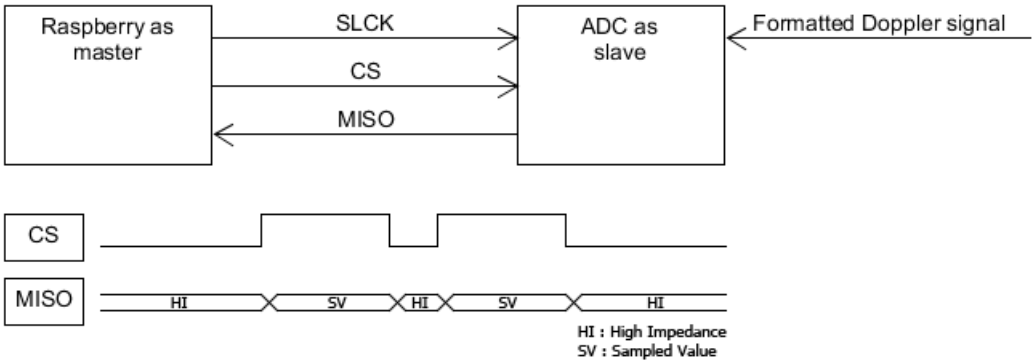


FIGURE 3.9: Fonctionnement de l'ADC

3.4 Raspberry

L'architecture du Raspberry (Figure 3.10) contient un serveur Apache qui fournit une interface web à un ordinateur client, via le réseau LAN (Local Area Network). L'interface web se connecte au serveur WebSocket dont les fonctionnalités sont implémentées dans ws_server, basé sur Node.js. Le client peut envoyer différentes requêtes pour recevoir, par exemple, les valeurs échantillonnées en direct. Suivant la requête du client, le serveur WebSocket démarre, soit GetVoltage, soit GetExperiment. Ces programmes retournent des résultats au serveur WebSocket, qui les envoie au client.

Le Raspberry possède ainsi 4 programmes distincts :

- **GetVoltage** échantillonne le signal Doppler à une fréquence de 2Hz (le but de ce programme est de donner une idée du signal au client. Les valeurs échantillonnées sont transmises en direct sur l'interface web, voilà pourquoi la fréquence d'échantillonnage est basse)
- **GetExperiment** échantillonne le signal Doppler à une fréquence de 100kHz, calcule les paramètres hémodynamiques et sauvegarde les résultats dans des fichiers
- **ws_client** contient l'interface web qui est fournie au client par le serveur Apache

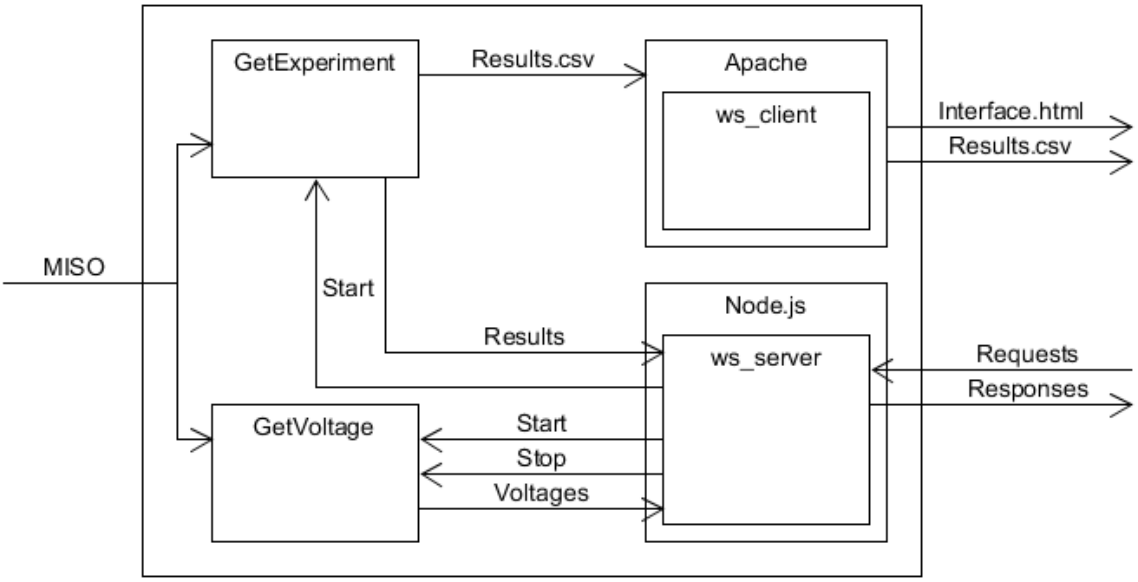


FIGURE 3.10: Architecture du Raspberry

- **ws_server** contient le serveur WebSocket, basé sur Node.js

Les librairies utilisées par les programmes sont :

- bcm2835 pour l'utilisation du SPI sur le Raspberry, écrite en C
- FFTW pour le calcul de la FFT (Fast Fourrier Transform), nécessaire pour calculer le spectre de puissance, écrite en C
- Bootstrap pour le style de l'interface web, écrite en CSS (Cascading Style Sheets) et JavaScript
- jQuery pour les fonctionnalités de l'interface web, écrite en JavaScript
- Chart.js pour l'implémentation de graphiques dans l'interface web, écrite en CSS et JavaScript
- ws pour l'implémentation d'un serveur WebSocket, écrite pour Node.js, en JavaScript

Notes: Apache est un serveur HTTP (HyperText Transfer Protocol). Node.js est une librairie serveur JavaScript. Un WebSocket est un canal de communication bidirectionnel entre un client et un serveur.

3.4.1 GetVoltage

Ce programme a été écrit en C et il est exécutable de cette manière :

```
sudo ./GetVoltage <Interval in ms between each sampling>
```

Le but de ce programme est d'échantillonner les données qui vont être transmises en direct au client, lui donnant un aperçu du signal. La fréquence d'échantillonnage est basse (2Hz). Elle est fixée par des contraintes de l'interface web.

A chaque échantillonnage, le programme écrit dans la console la valeur échantillonnée en mV. Le programme est démarré par le server WebSocket.

La console est un moyen de communiation entre l'application GetVoltage (en C) et le serveur WebSocket (en Javascript).

3.4.2 GetExperiment

Ce programme a été écrit en orienté objet, en C++. Le programme est exécuté de la manière suivante :

```
sudo ./GetExperiment <DC offset> <Amplification factor> <Etc.>
```

Tous les paramètres sont indiqués dans le tableau 3.1.

DC offset	Offset DC réglé par l'électronique
Amplification factor	Facteur d'amplification de l'AC réglé par l'électronique
Save data	Indique si les données échantillonnées doivent être sauvés
Recording time	Temps de l'expérience en s
DC precision	Précision DC pour le tri des données valides et invalides
Low frequency	Fréquence de départ pour le calcul des paramètres hémodynamiques
High frequency	Fréquence de fin pour le calcul des paramètres hémodynamiques
Firstname	Prénom du patient
Lastname	Nom du patient
Birthdate year	Année de naissance du patient
Birthdate month	Mois de naissance du patient
Birthdate day	Jour de naissance patient
Gender	Genre du patient
Experiment	Nom de l'expérience
Step	Etape de l'expérience
Comments	Commentaires sur l'expérience

TABLE 3.1: Paramètres du programme GetExperiment

Le but de ce programme est d'échantillonner le signal Doppler, de calculer les paramètres hémodynamiques ainsi que la FFT et le spectre de puissance.

A chaque expérience, le programme écrit dans la console des informations sur l'avancée de cette dernière. Les informations écrites dans la console sont récupérées par le serveur WebSocket afin d'être envoyées au client.

Il est découpé en 5 classes (Figure 3.11) :

- **Main** est le point d'entrée du programme
- **AdcData** échantillonne le signal Doppler
- **FramesSelection** découpe le signal échantillonné en cadres de données
- **HemodynamicParameters** calcule les paramètres hémodynamiques pour chaque cadre de données
- **DataProcessing** calcule la FFT et le spectre de puissance pour chaque cadre de données

Notes : Le diagramme de séquence du programme GetExperiment est visible en annexe A.5.

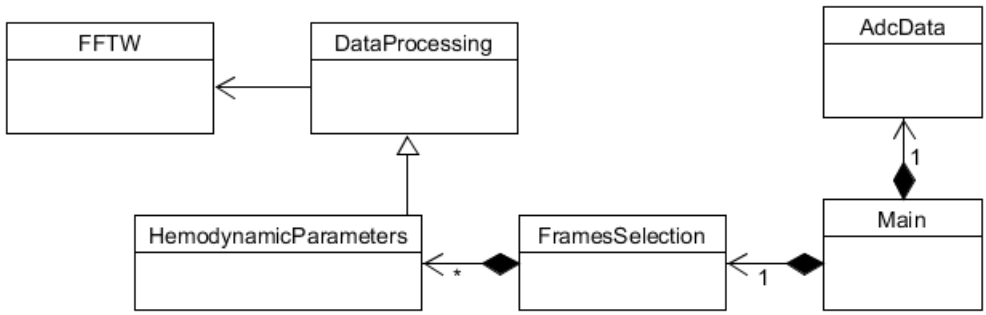


FIGURE 3.11: Diagramme de classe du programme GetExperiment

Main

C'est le point d'entrée du programme. Elle joue le rôle de factory (Figure 3.12). Elle s'occupe de récupérer les données échantillonnées. Elle les fournit ensuite à la classe FramesSelection, laquelle calcule les résultats de l'expérience et les sauvegarde dans des fichiers CSV.

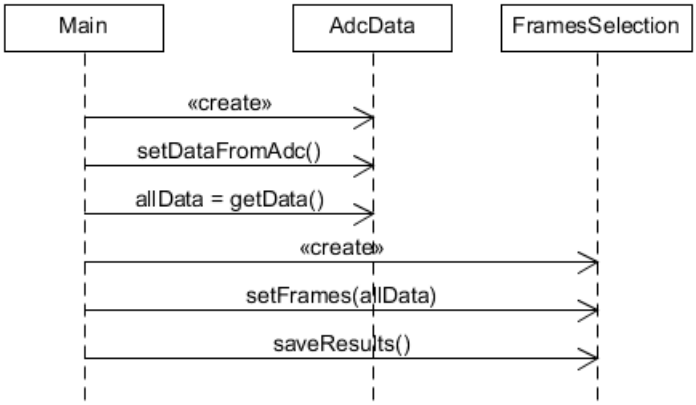


FIGURE 3.12: Diagramme de séquence de la fonction Main

AdcData

Le système d'exploitation du Raspberry n'est pas temps réel. Le programme GetExperiment est exécuté dans un processus, lequel peut être mis en pause par l'ordonnanceur. La partie échantillonnage des données devient problématique car la vitesse d'échantillonnage peut varier au cours de la mesure.

La classe AdcData a été développée en prenant compte de ces considérations. Afin d'obtenir les résultats les plus satisfaisants, un minuteur temps réel système a été utilisé. Le programme est aussi exécuté en indiquant à l'ordonnanceur que celui-ci doit être traité en temps réel, avec la plus haute priorité.

Le minuteur a été programmé de façon à exécuter le code qui échantillonne le signal toutes les 10us. La fréquence d'échantillonnage est donc :

$$fs = \frac{1}{Interval} = \frac{1}{10u} = 100kHz \tag{3.22}$$

FramesSelection

En ophtalmologie, lorsqu'une mesure est faite sur un patient, il arrive que celui-ci ferme les yeux. Les valeurs ainsi récoltées ne sont pas toutes correctes. Dans ce sens, la classe FramesSelection trouve son utilité. Elle permet de faire le tri entre les valeurs valides et invalides. Elles fonctionne en découpant toutes les valeurs récoltées en plusieurs cadres (Figure 3.13).

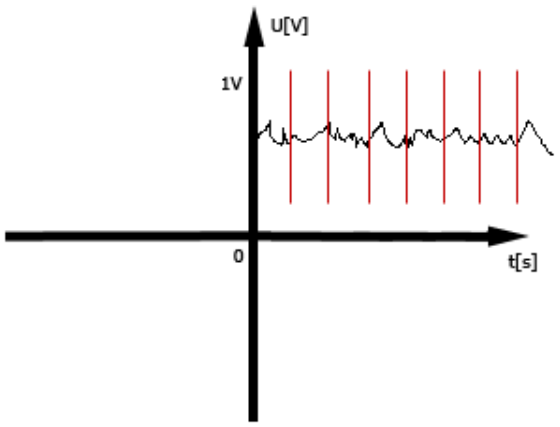


FIGURE 3.13: Découpage des valeurs en cadres

Chaque cadre de données correspond à une instance de la classe HemodynamicParameters. La classe FramesSelection a donc un tableau d'objets HemodynamicParameters. Chaque cadre possède notamment sa propre valeur DC et son propre paramètre hémodynamique ChBVol, utiles pour le tri. Des moyennes de ces valeurs sont faites en prenant en compte tous les cadres.

Un cadre de données est gardé si (Source [4]) :

- Sa valeur DC ne dévie pas de plus de 20% de la valeur DC moyenne de tous les cadres
- Son paramètre hémodynamique ChBVol ne dévie pas plus de 30% du paramètre hémodynamique moyen de tous les cadres
- La puissance spectrale est plus grande dans les basses fréquences que dans les hautes fréquences, avec un facteur minimum de 10

Sur les cadres restants (ceux ayant passé le test ci-dessus), les paramètres hémodynamiques moyens, la FFT moyenne et le spectre de puissance moyen sont calculés. Ces résultats, en plus des données échantillonnées sont sauvegardées dans des fichiers CSV. Ces fichiers se trouvent dans le répertoire d'Apache. Ils sont ainsi téléchargeables depuis l'interface.

DataProcessing

C'est une classe générale utile pour le traitement de données. Elle calcule la moyenne des données (qui est la valeur DC), la FFT et le spectre de puissance.

Le calcule de la FFT prend un temps de calcul non négligeable. Pour optimiser le temps de calcul, la propriété suivante a été utilisée (valable que lorsque les valeurs

sont réelles) :

$$FFT[i] = FFT[i - ndata] \tag{3.23}$$

Avec ndata : Le nombre de données

Ce principe a été utilisé en indiquant à la libraire FFTW que les données étaient purement réelles. Ainsi, le temps de calcul de la FFT se trouve réduit d'un facteur 2.

HemodynamicParamters

La classe HemodynamicParamters hérite de la classe DataProcessing. Elle s'appuie sur ses fonctionnalités pour calculer les paramètres hémodynamiques (car ceux-ci dépendent du spectre de puissance).

3.4.3 ws_client

C'est l'interface web (Annexe A.6) de pilotage du Raspberry. Elle est fournie par Apache lorsqu'un ordinateur client tape l'adresse IP du Raspberry dans son navigateur.

En haut à droite dans l'interface se trouvent les appareils de mesure (les Raspberry). Ceux qui sont utilisables sont écrits en vert, les autres étant écrits en rouge. Elle comporte trois onglets :

- **Préférences** pour saisir la façon dont va se dérouler l'expérience (durée, fréquence de départ et de fin, etc.)
- **Subject** pour saisir les informations sur la patient (nom, prénom, etc.)
- **Results** pour visualiser le signal échantillonné en direct, démarrer l'expérience et récupérer les résultats

L'interface se connecte au serveur WebSocket. Une requête du client est dans le format JSON (JavaScript Object Notation). Elle doit contenir un nom de commande ainsi que des paramètres. Une réponse du serveur est aussi dans ce format. Par exemple, le message envoyé par le client pour recevoir les valeurs échantillonnées en direct toutes les 500ms est :

```
"command":"getVoltage",
"params":{
    "refreshTime":500
}
```

Un diagramme de séquence (Figure 3.14) montre les messages échangés. L'interface demande au serveur de recevoir les données en direct, avec un interval de 500ms entre chaque réception. L'utilisateur renseigne les données de l'expérience et la démarre. Une nouvelle requête est envoyée au serveur avec les données saisies. Les résultats de l'expérience sont retournés et l'utilisateur peut récupérer ces derniers.

Un contrôle des données de l'expérience est fait côté serveur. Si l'utilisateur essaie de démarrer une expérience sans avoir renseigné correctement les données de cette dernière, il obtient des messages d'erreur (Figure 3.15).

Lorsque l'expérience est finie, les résultats sont retournés à l'utilisateur (Figure 3.16). L'utilisateur dispose d'un lien où il peut télécharger les fichiers de résultats (Figure 3.17).

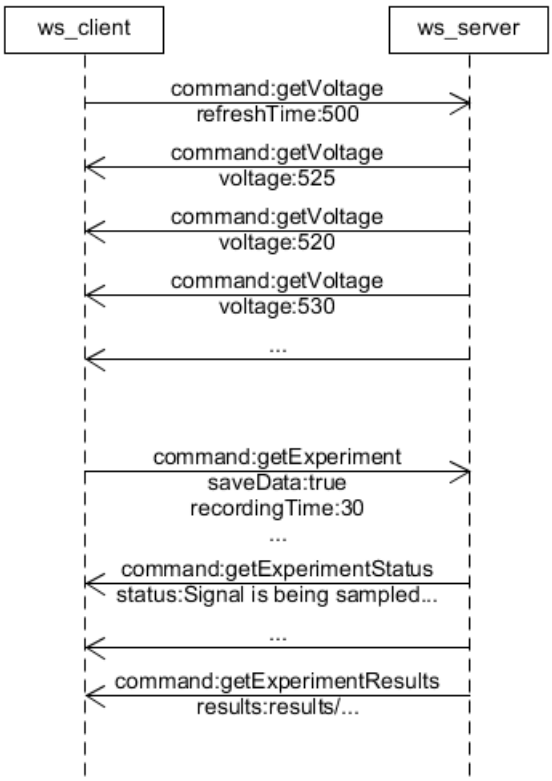


FIGURE 3.14: Messages WebSocket échangés

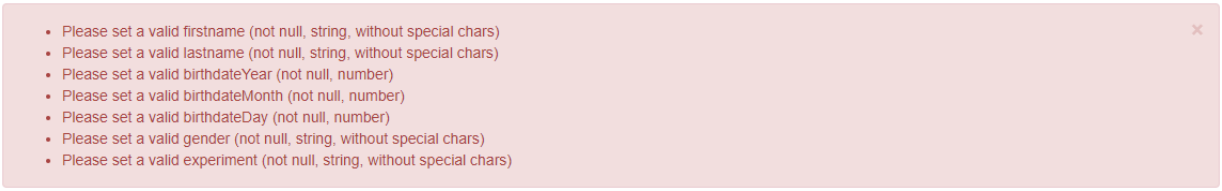


FIGURE 3.15: Messages d'erreur possibles lors du démarrage d'une expérience

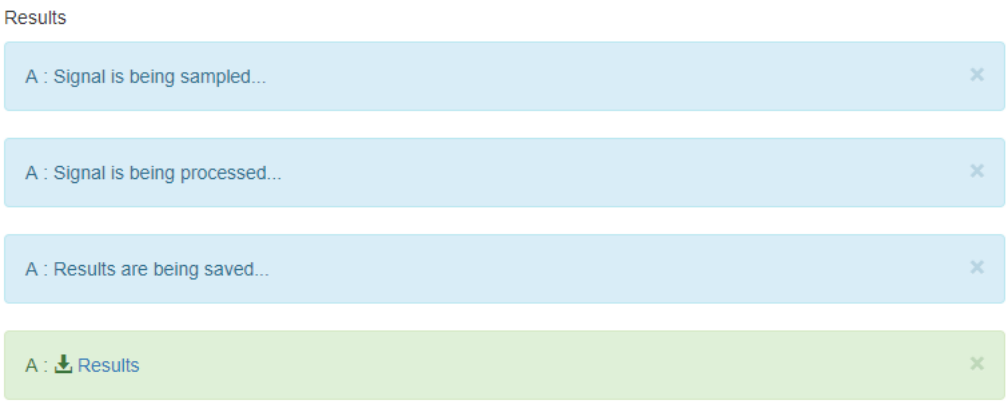


FIGURE 3.16: Résultats d'une expérience

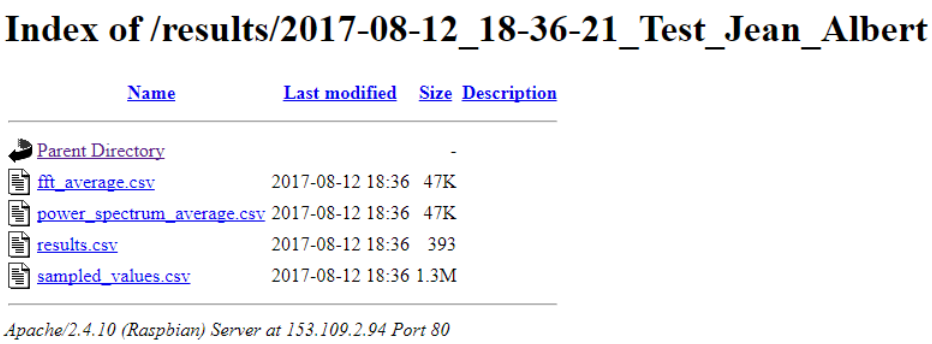


FIGURE 3.17: Fichiers de résultats d'une expérience

Notes : Il est possible d'effectuer une mesure sur plusieurs Raspberry en même temps. Pour cela, ils doivent être sélectionnés dans le premier onglet. L'option de sauvegarde des données permet, quant à elle, d'indiquer si les valeurs échantillonnées doivent être sauvegardées dans un fichier CSV ou non.

3.4.4 ws_server

Le serveur WebSocket a été implémenté dans Node.js. Il se lance de la manière suivante :

```
sudo node server.js
```

Comme le Raspberry peut être éteint, il a fallu trouver une manière de lancer automatiquement le serveur au démarrage. Ainsi, la commande de lancement du serveur a été ajoutée dans le fichier rc.local du Raspberry.

Les WebSockets ont été utilisés afin d'avoir une connexion bidirectionnelle entre le client et le serveur. Ceci permet de développer une interface qui devient très fluide, sans besoin de faire des recharges de page, comme c'est le cas pour une application web standard.

Son fonctionnement (Figure 3.18) est le suivant : Lorsqu'il reçoit une requête du client, il détermine quel programme lancer. Il lance le programme par le biais de la console. Le programme s'exécute ensuite et les informations qu'il écrit dans la console sont récupérées par ce dernier. Ces informations sont envoyées au client.

Le serveur n'accepte qu'un seul client à la fois. Ceci pour éviter que plusieurs expériences soient lancées en même temps sur le Raspberry, celui-ci ne possédant qu'un ADC.

Notes : Le fichier rc.local est utile pour l'exécution de commandes au démarrage du Raspberry.

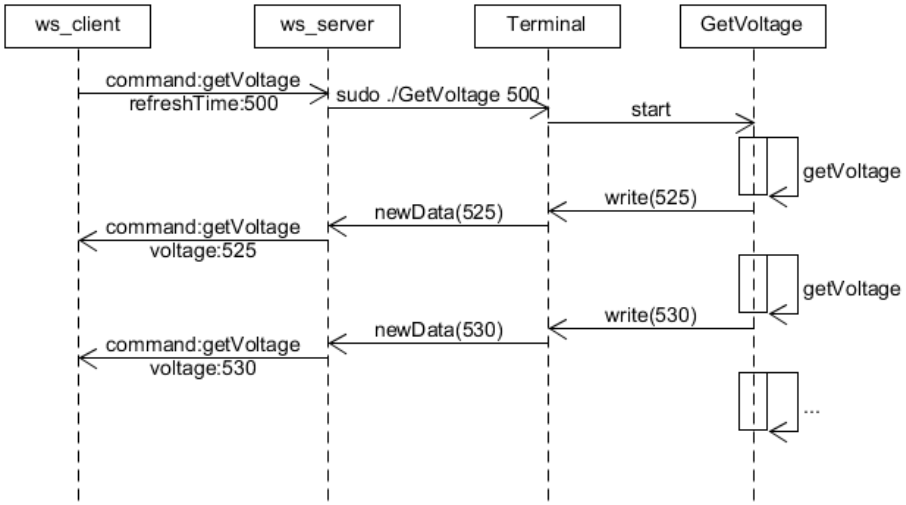


FIGURE 3.18: Fonctionnement du serveur WebSocket

Chapter 4

Tests

Ce chapitre sert à expliquer les tests effectués sur chacune des parties du système de manière indépendante. De plus, la procédure de validation des fonctions est explicitée. Enfin, le système complet a été testé.

4.1 Electronique

Chaque partie de l'électronique a été testée individuellement. Les résultats espérés sont écrits dans le tableau 4.1 :

Convertisseur de tension	Les tensions de sortie sont de +-12V
Inverseur	Le signal est inversé
Amplificateur AC	La composante AC peut être amplifiée d'un facteur entre 1 et 11
Régleur DC	La composante DC peut être ajustée
FIltre anti-repliement	La fréquence de coupure est à 40kHz et la pente est de -80dB/dec

TABLE 4.1: Tests effectués sur l'électronique

4.1.1 Convertisseur de tension

La tension d'entrée de +5V est bien convertie en deux tensions de sortie d'environ +12V et -12V (Figure 4.1).

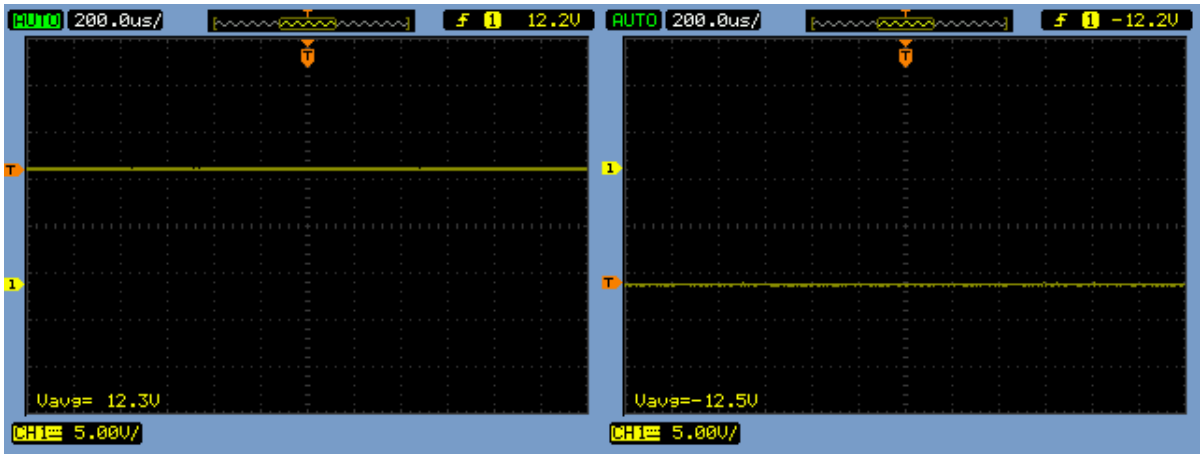


FIGURE 4.1: Test du convertisseur de tension

4.1.2 Inverseur

En appliquant un signal sinusoïdal en entrée, c'est bien un signal sinusoïdal inversé qui est obtenu en sortie (Figure 4.2).

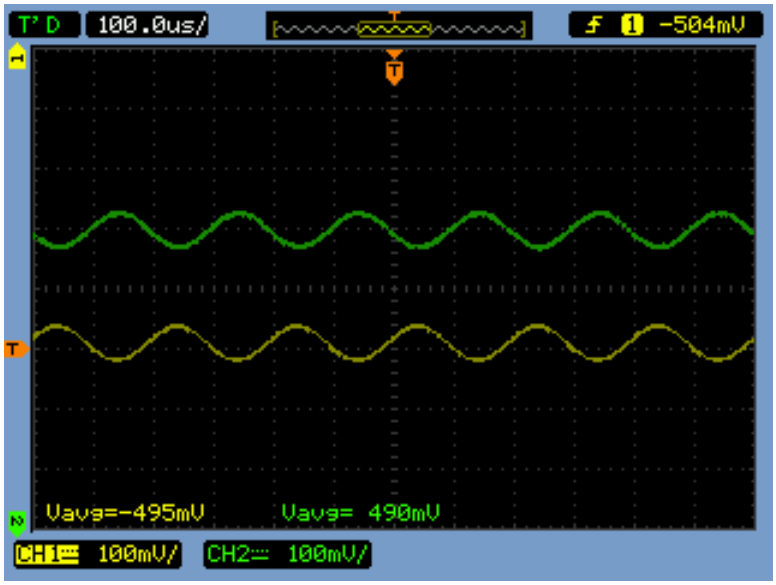


FIGURE 4.2: Test de l'inverseur

4.1.3 Amplificateur AC

La composante AC du signal peut être amplifiée d'un facteur allant de 1.1 à 10 (Figure 4.3). L'amplitude du signal de sortie varie de 72mVpp à 640mVpp à partir d'un signal dont l'amplitude est 64mVpp. Le facteur d'amplification maximal est légèrement inférieur à la valeur théorique qui est de 11.

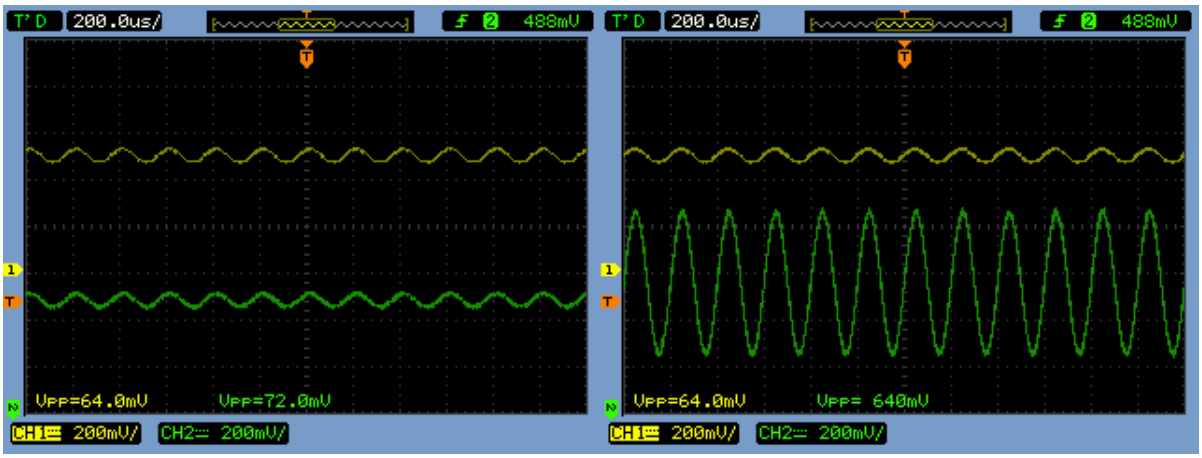


FIGURE 4.3: Test de l'amplificateur AC (à droite : g=1.1, à gauche : g=10)

4.1.4 Régleur DC

A l'aide du potentiomètre, la composante DC a été variée. Le signal en sortie est bien décalé d'un offset positif ou négatif (Figure 4.4).

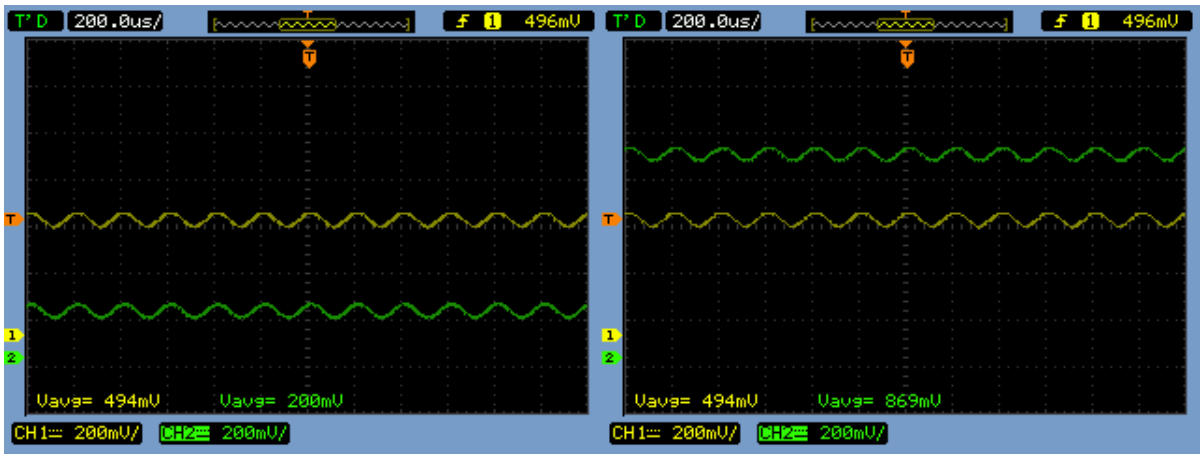


FIGURE 4.4: Test du régleur DC

4.1.5 Filtre anti-repliement

Les tests concernant le filtre ont été effectués en injectant un signal dont l'amplitude a été fixée. L'amplitude du signal en sortie du filtre a été mesurée tout en balayant la fréquence du signal d'entrée.

La fréquence de coupure, relevée à -3dB, est proche de 40kHz. La pente est légèrement moins bonne que celle attendue pour un filtre de quatrième ordre, qui est de -80dB/dec.

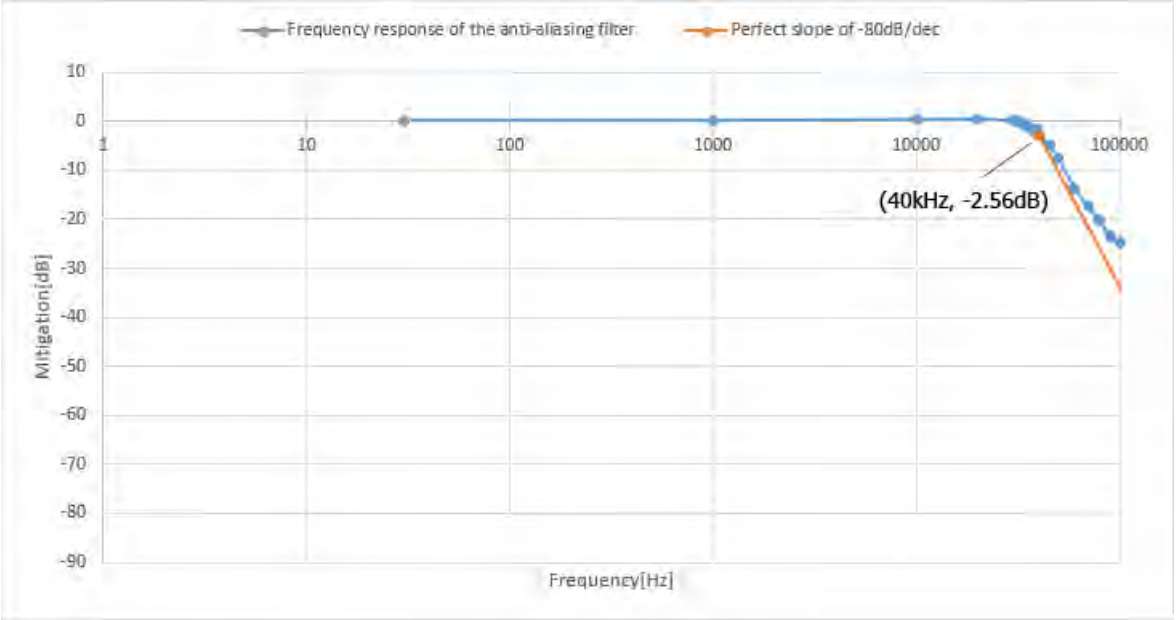


FIGURE 4.5: Test du filtre anti-repliement

4.2 GetVoltage

Afin de tester ce programme, un signal carré d'une fréquence de 1Hz a été échantillonné à une fréquence de 2Hz. Ce signal carré a une valeur DC de 500mV et une amplitude de 100mV (Figure 4.6).

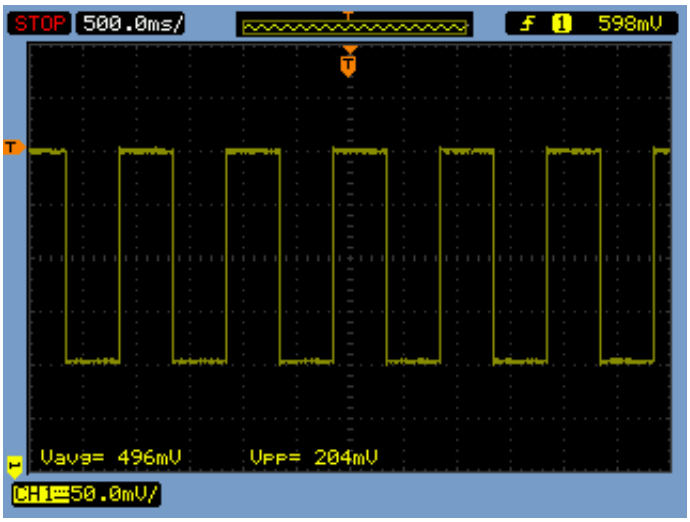


FIGURE 4.6: Test du programme GetVoltage avec un signal carré

Les valeurs échantillonnées vont bien de 400mV à 600mV comme attendus (Figure 4.7). De petites variations sont notables car le signal carré injecté n'est pas parfait.

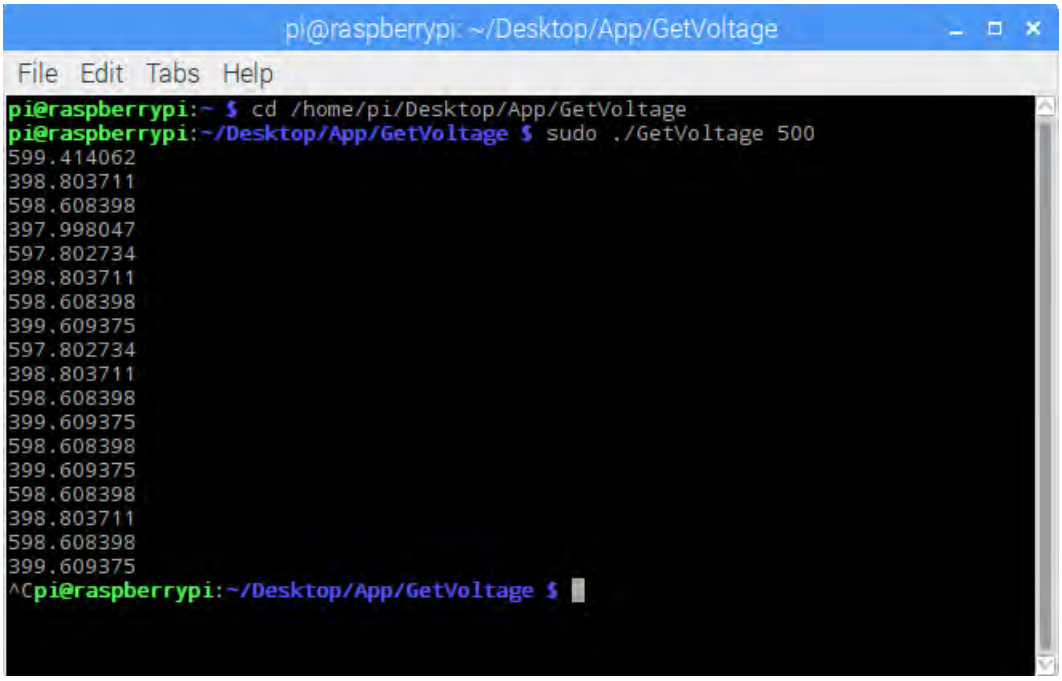


FIGURE 4.7: Test du programme GetVoltage

4.3 GetExperiment

Les tests fait sur le programme GetExperiment ont été découpés en trois. D'abord, la véracité du calcul de la FFT et du spectre de puissance a été contrôlée. Ensuite c'est l'échantillonnage à une fréquence de 100kHz qui a été testé. Finalement, c'est le calcul des paramètres hémodynamiques qui a été validé.

4.3.1 FFT et spectre de puissance

Dans le document d'implémentation des paramètres hémodynamiques (Source [4]), plusieurs façons ont été décrites pour obtenir la valeur DC et la puissance moyenne à partir de la FFT et du spectre de puissance.

$$DC = \frac{1}{ndata} \sum_{i=0}^{ndata-1} data[i] = FFT[0] = \sqrt{PSD[0] * \Delta f} \tag{4.1}$$

Avec DC : La valeur DC du signal échantillonné, ndata : Le nombre de données, data : Les données, FFT : La transformée de Fourier, PSD : La puissance spectrale et Δf : Le pas de la fréquence (fs/ndata)

$$power\_average = \frac{1}{ndata} \sum_{i=0}^{ndata-1} data[i] * data[i] = \sum_{i=0}^{\frac{ndata}{2}} PSD[i] * \Delta f \tag{4.2}$$

Avec power_average : La puissance moyenne du signal échantillonné

Ces différentes équations ont été utilisées pour valider le calcul de la FFT et du spectre de puissance (Figure 4.8). Les différentes équations du calcul de la valeur DC ont donné le même résultat. Ceci est pareil avec les différentes équations de calcul de la puissance moyenne.

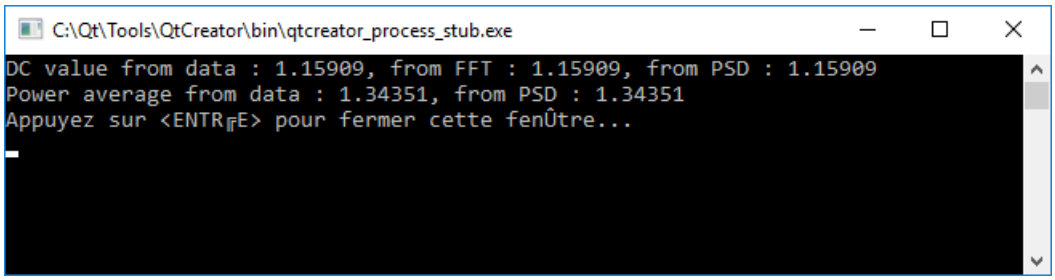


FIGURE 4.8: Test de la FFT et du spectre de puissance du programme GetExperiment

4.3.2 Echantillonnage à 100kHz

Afin de valider l'échantillonnage à 100kHz, un signal sinusoïdal à une fréquence de 15kHz a été généré. La FFT sur le signal échantillonné a été comparé à une FFT calculée pour un signal de même caractéristique, qui constitue la référence.

La valeur DC dévie de 0.4%, tandis que la valeur à la fréquence du sinus dévie de 12.72% (Figure 4.9).

Ces variations sont explicables par la problématique temps réel expliquée précédemment. Comme le système d'exploitation n'est pas temps réel, l'interval entre chaque échantillonnage peut varier dans une moindre mesure, malgré l'utilisation d'un timer dit temps réel (Chapitre 3.4.2).

4.3.3 Calcul des paramètres hémodynamiques

Pour rappel, les trois paramètres hémodynamiques sont :

- **ChBVel** qui donne une information sur la vitesse du sang

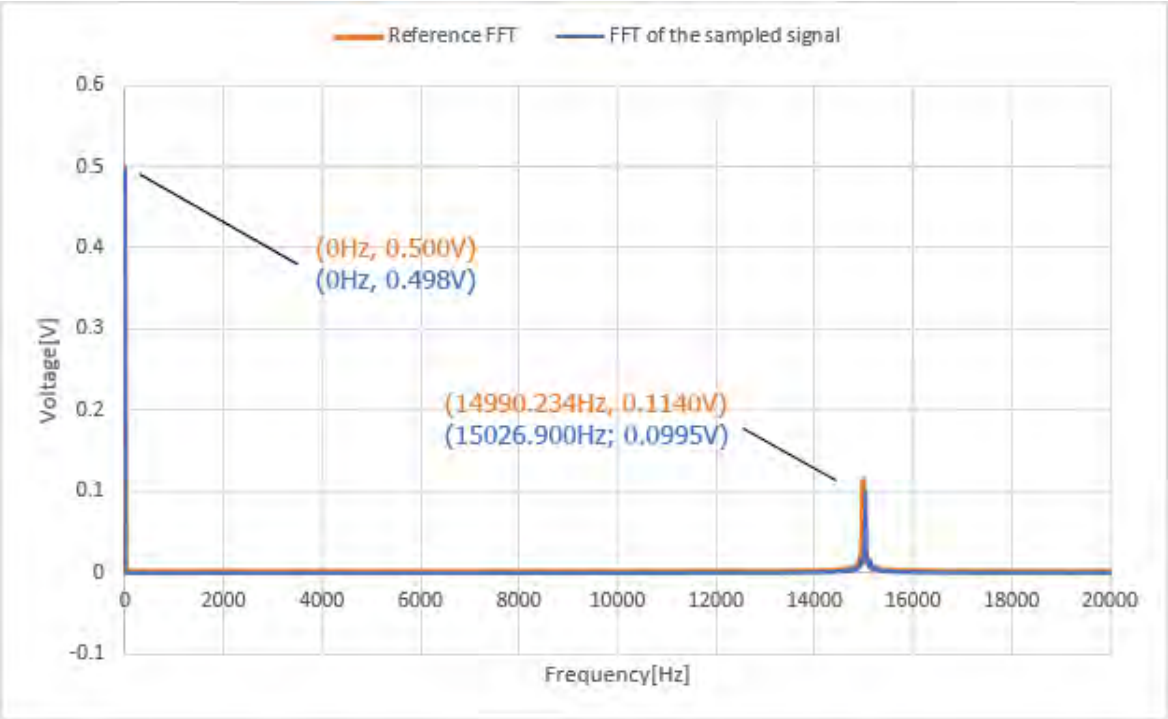


FIGURE 4.9: Test de la fréquence d'échantillonnage du programme GetExperiment

- **ChBVol** qui donne une information sur le volume de sang
- **ChBF** qui donne une information sur le flux du sang

Si les paramètres hémodynamiques sont calculés à partir d'un signal sinusoïdal, ChBVel doit varier linéairement en fonction de la fréquence du signal, tandis que ChBVol doit varier linéairement en fonction de l'amplitude du signal. ChBF, quant à lui, est la multiplication des deux précédents paramètres, il varie en fonction de ceux-ci.

En fonction de la fréquence d'un signal sinusoïdal

Un sinus à différentes fréquences (dans la bande de fréquence utile du signal) a été échantillonné. L'amplitude et la valeur DC de ce dernier n'ont pas été changés. Les résultats (Figures 4.10, 4.11) sont : ChBVel varie de façon linéaire. ChBVol, quant à lui, varie peu avec une différence de 3.85% entre la plus grande et la plus petite valeur.

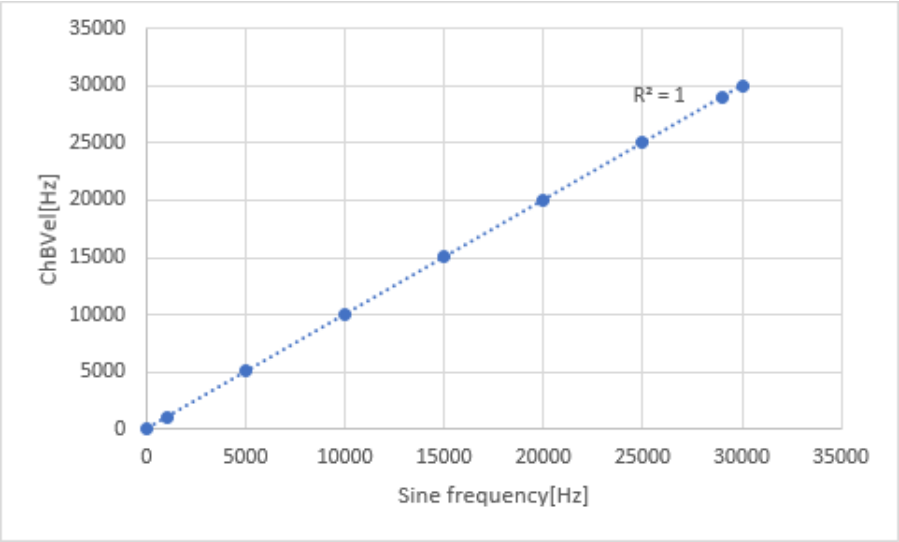


FIGURE 4.10: ChBVel en fonction de la fréquence d'un signal sinusoïdal

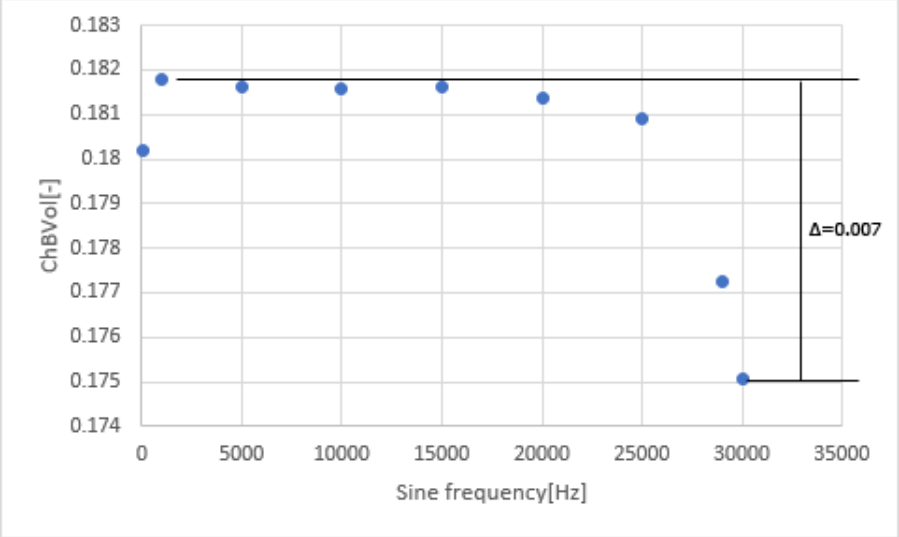


FIGURE 4.11: ChBVol en fonction de la fréquence d'un signal sinusoïdal

En fonction de l'amplitude d'un signal sinusoïdal

Les mêmes tests ont été faits avec un sinus à différentes amplitudes. La fréquence et la valeur DC ont été fixées. Les résultats (Figures 4.12, 4.13) sont ceux espérés : Pour le paramètres ChBVol, il varie linéairement en fonction de l'amplitude du sinus. ChBVel, quant à lui, ne varie que peu avec une différence entre la plus grande et la plus petite valeur de 0.06%.

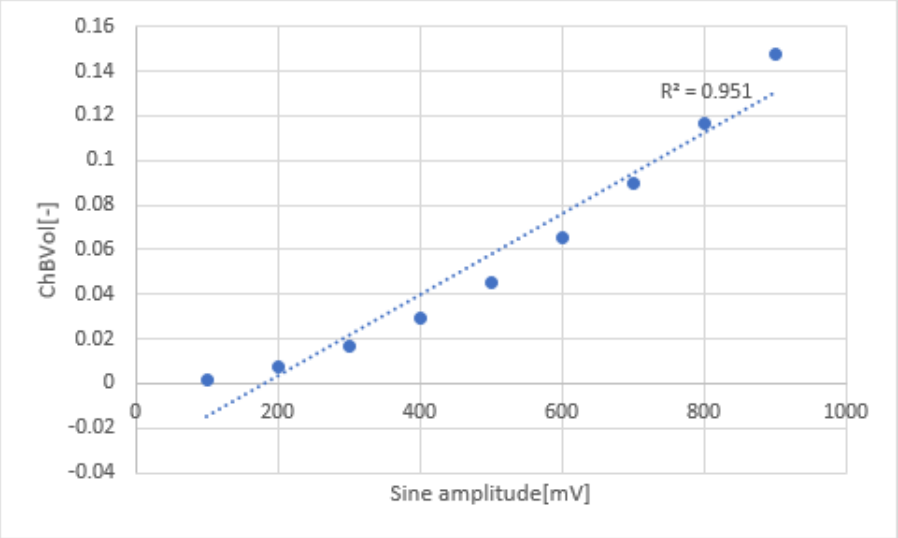


FIGURE 4.12: ChBVol en fonction de l'amplitude d'un signal sinusoïdal

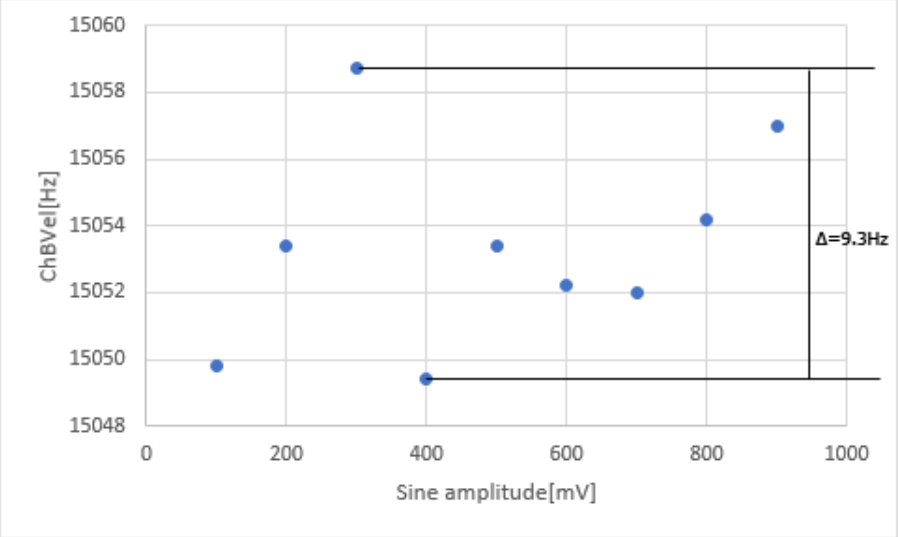


FIGURE 4.13: ChBVel en fonction de l'amplitude d'un signal sinusoïdal

4.4 Système complet

Il a été vu que la FFT obtenue n'est pas parfaite. Des tests ont été fait afin de voir les répercussions sur les paramètres hémodynamiques avec un signal Doppler.

Pour générer ce dernier, un laser, un photodétecteur, une roue en téflon et un moteur ont été utilisés (Figure 4.14). Le moteur DC permet de faire tourner la roue à un vitesse de rotation qui est proportionnelle à la tension qui lui est apppliquée.

Le moteur a été alimenté avec différentes tensions. Les résultats attendus sont que le paramètre ChBVel varie linéairement en fonction de la tension appliquée sur le moteur. Le paramètre ChBVol ne doit pas varier.

Le paramètre ChBVel varie linéairement en fonction de la vitesse de la roue (Figure 4.15). La paramètre ChBVol varie peu avec une différence entre la plus grande et la plus petite valeur de 4.55% (Figure 4.16).

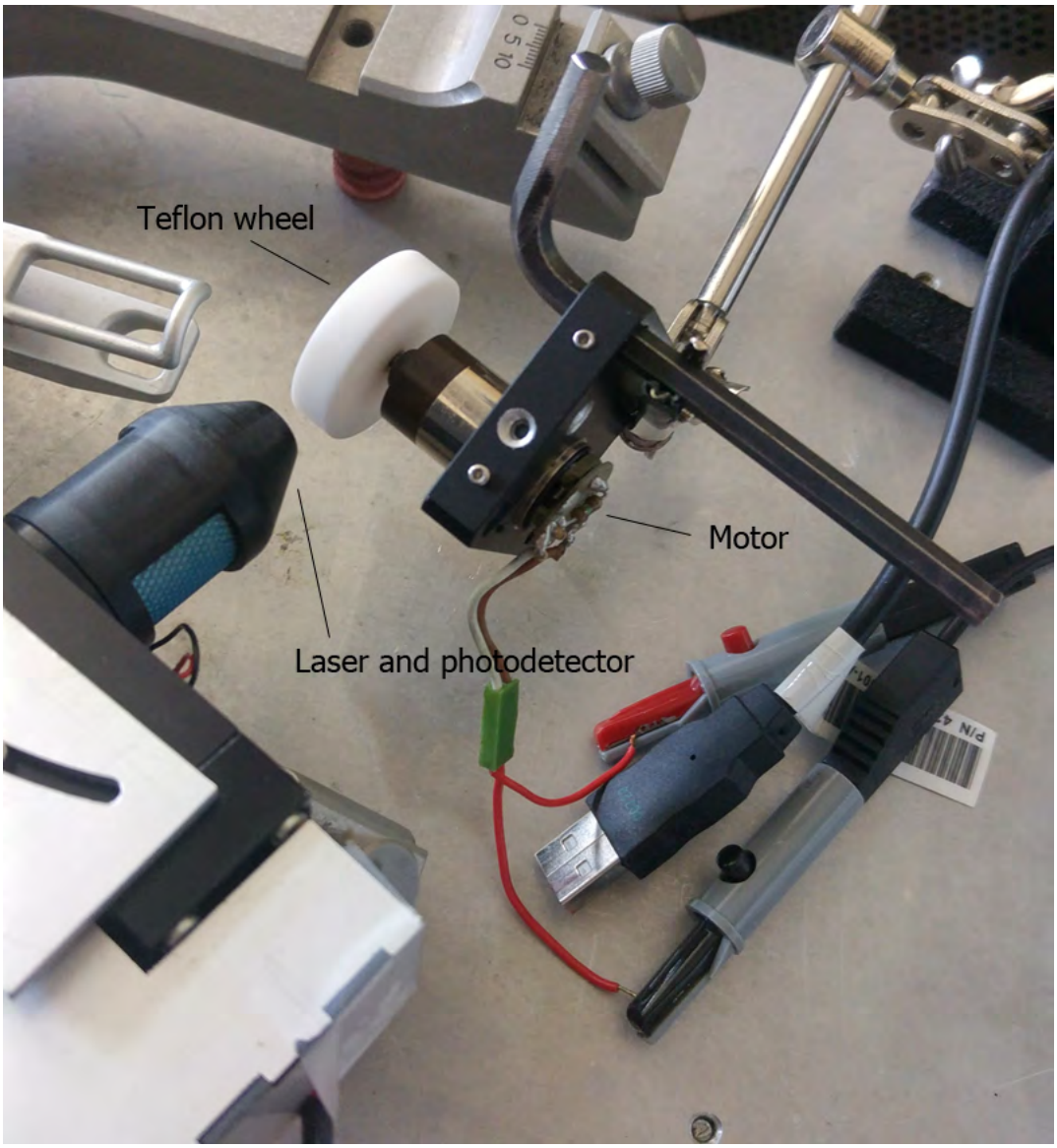


FIGURE 4.14: Génération d'un signal Doppler avec une roue en téflon et un moteur

Le spectre de puissance du signal Doppler (Figure 4.17) a été fait avec deux tensions différentes appliquées sur le moteur. Les formes des spectres de puissance obtenues sont bien décroissantes. Il également intéressant de voir que lorsque le moteur est alimenté en 2.6V, un pic de puissance est visible vers 110Hz, et lorsqu'il est alimenté en 5.28V, le pic est présent aux alentours des 230Hz.

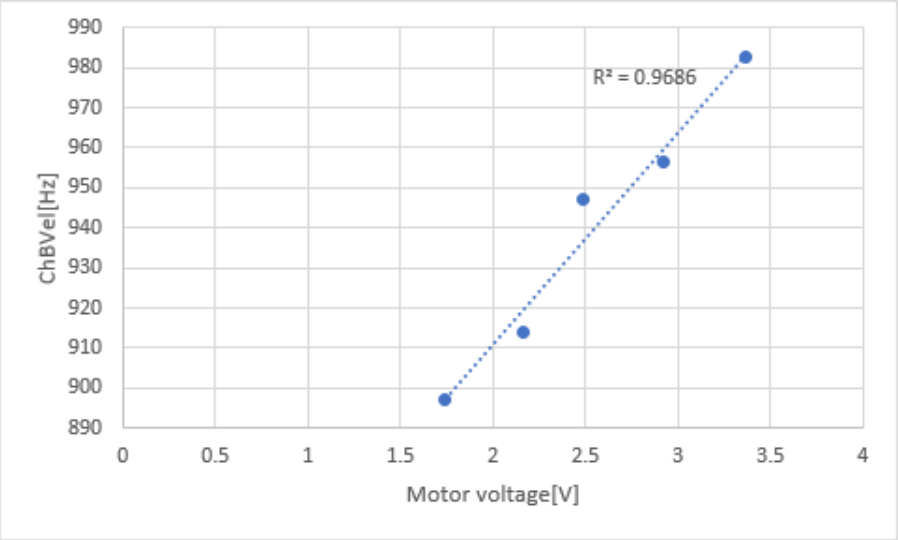


FIGURE 4.15: ChBVel en fonction de la tension du moteur

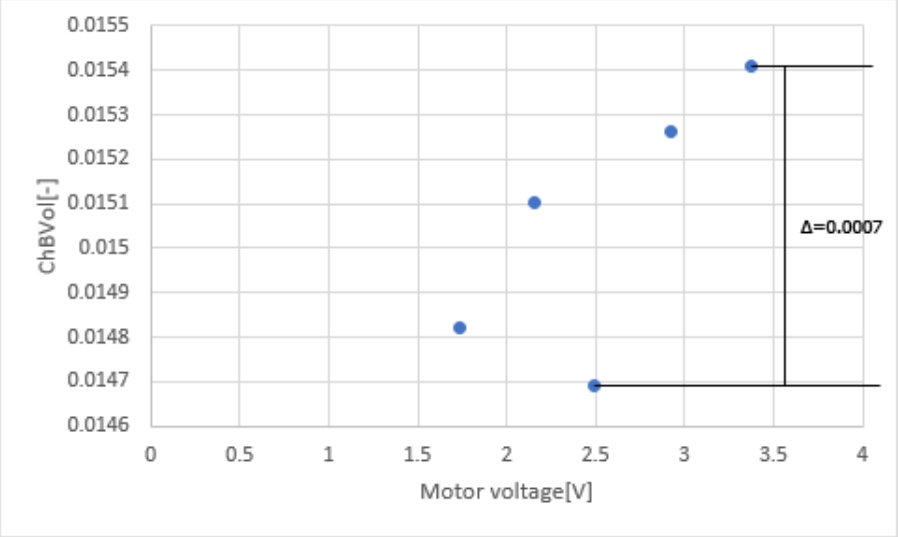


FIGURE 4.16: ChBVol en fonction de la tension du moteur

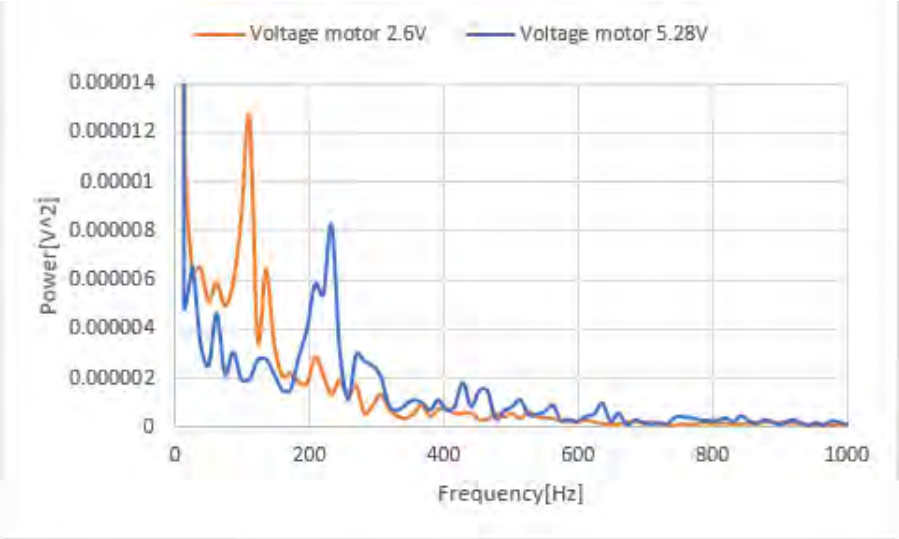


FIGURE 4.17: Spectres de puissances avec différentes tensions appliquées sur le moteur

Chapter 5

Conclusion

Le système développé par M. Francesco Marazzi, basé sur la Fluxmétrie Doppler par Laser (LDF), permet d'extraire les paramètres hémodynamiques d'un signal Doppler. Le signal est échantillonné sur une carte embarquée et le calcul des paramètres hémodynamiques est fait sur un ordinateur.

Dans le cadre de mon travail, l'objectif principal était de créer un nouveau système dont le calcul des paramètres hémodynamiques soit fait directement sur la carte embarquée. Ce travail était aussi l'occasion d'apporter des améliorations pour le nouveau système.

La nouvelle électronique est maintenant plus simple au niveau des branchements : Une seul tension d'alimentation est requise, prise sur le Raspberry. Le facteur d'amplification de la composante AC peut être réglé à l'aide d'un potentiomètre. Le filtre anti-repliement numérique a été changé par un analogique, évitant les problèmes d'horloge du filtre numérique qui ajoutaient du bruit sur le signal.

La Discovery board a été remplacée par un ADC, simplifiant le système et le rendant plus compact. En contrepartie, des problématiques de temps réel sont apparues. Ces considérations ont été prises en compte dans le développement de l'application en utilisant un minuteur système temps réel.

Malgré ces précautions, les tests ont révélé que la transformée de Fourier calculée à partir des valeurs échantillonnées d'un sinus de référence n'était pas parfaite. A partir de cela, des tests complémentaires ont été réalisés afin de vérifier l'impact de ces problèmes sur le calcul des paramètres hémodynamiques.

Lors de ces tests complémentaires, un sinus dont la fréquence a été variée, a démontré que les résultats obtenus étaient satisfaisants. Effectivement, le paramètre ChBVel, correspondant à la vitesse de déplacement du sang, varie linéairement en fonction de la fréquence avec un coefficient de détermination R^2 de 1. Le paramètre ChBVol, quant à lui, associée au volume du sang reste quasi constant avec une variation maximale de 3.85%.

Le cas où l'amplitude d'un sinus a été variée montre également de bons résultats. Le paramètres ChBVol varie linéairement en fonction de l'amplitude avec un coefficient de détermination R^2 de 0.951. Le paramètre ChBVel est constant avec un variation maximale de 0.06%.

Ainsi, ces tests ont donnés des resultats prometteurs. Pour valider ce nouveau système, celui-ci a été testé dans son intégralité en incluant un laser et un photodetecteur pour réaliser une vraie mesure de signal Doppler. Naturellement, il fallait un système en mouvement permettant d'engendrer l'effet Doppler, un disque tournant en téflon a donc été utilisé.

Les résultats ont également été satisfaisants. Effectivement, le paramètre ChBVel varie linéairement en fonction de la tension d'alimentation du moteur. Le coefficiant de détermination R^2 est 0.9686. Pour ce qui est du paramètre ChBVol, il reste constant avec une variation maximale de 4.55%.

Au niveau du portage du calcul des paramètres hémodynamiques sur le Raspberry, les optimisations faites donnent de bons résultats : Pour 30s de données, un temps de calcul d'environ 2s est nécessaire.

5.1 Perspectives

Le nouveau système montre qu'il est possible d'obtenir des résultats correctes sans utiliser de microcontrôleur. Bien entendu, l'utilisation d'un microcontrôleur règle totalement les problèmes de temps réel. Un système intégrant un microcrontrôleur serait donc plus précis.

Afin d'avoir quand même un système compact, il serait possible d'interfacer un microcontrôleur seul (sans Discovery board) avec le Raspberry. Un système de buffer, comme utilisé dans le système de M. Franceso Marazzi est idéal. Cette façon de faire nécessite cependant d'avoir un buffer d'assez grande taille, afin de permettre au Raspberry de réagir dans les temps lorsque le buffer du microcontrôleur doit être vidé. Si tel n'est pas le cas, les valeurs risquent d'être écrasées. La problématique de temps réel est ainsi toujours présente mais à un autre niveau.

Sur un principe similaire, un microcontrôleur pourrait écrire via SPI dans une mémoire flash les valeurs échantillonnées. Le microcontrôleur indiquerait au Raspberry lorsque le signal a été **entièrement** échantillonné, celui-ci réagirait ensuite en récupérant les valeurs dans la mémoire (Figure 5.1). L'avantage de cette solution par rapport à la précédente est qu'il existe des mémoire de très grande taille (1Gbits), et donc que la mémoire ne joue plus le rôle de buffer. Par exemple, une mémoire de 1Gbits donne, en MBytes :

$$1Gbits = 1024Mbits = 1024/8Mytes = 128MBytes \tag{5.1}$$

Les valeurs échantillonnées sont sauvegardées dans 2 bytes. Il est possible de sauvegarder un nombre de valeurs équivalent à :

$$128MBytes/2bytes = 64M \tag{5.2}$$

La fréquence d'échantillonage est de 100kHz, cela permet de sauvegarder un signal d'une durée de :

$$64M/100000 = 640s \tag{5.3}$$

Soit plus de 10minutes.

Selon discussions avec M. Alberto Dassatti, Professeur à la HEIG-VD, la solution la plus optimale serait de développer un pilote linux directement installable dans le noyau de Raspbian. Le pilote, n'étant plus dans l'espace utilisateur de Raspbian, serait éxécuté en temps réel. Une DMA (Direct Memory Access) serait utilisée pour copier les valeurs du SPI vers la mémoire. La DMA, après avoir copié un certain nombre de données, déclencherait une interruption. Cette interruption exécuterait le code de calcul de la FFT sur les valeurs disponibles en mémoire. Tout ceci est illustré en figure 5.2.

5.2 Ressenti personnel

Le fait que ce travail englobe différents domaines m'a beaucoup plu. Durant ce travail, j'ai eu la possibilité d'utiliser des compétences dans les domaines de la physique, de l'électronique et de l'informatique.

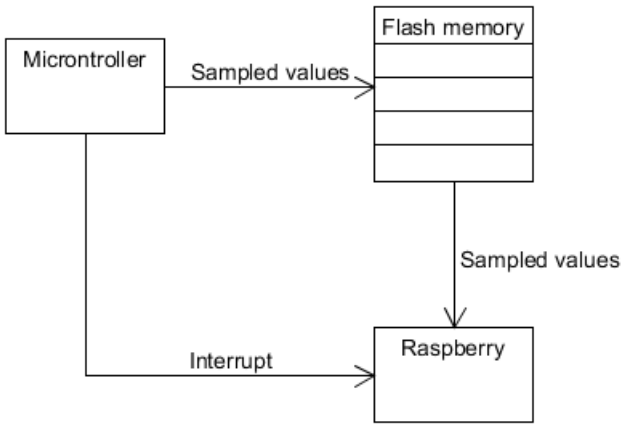


FIGURE 5.1: Microcontroleur avec une mémoire flash pour un échantillonnage en temps réel

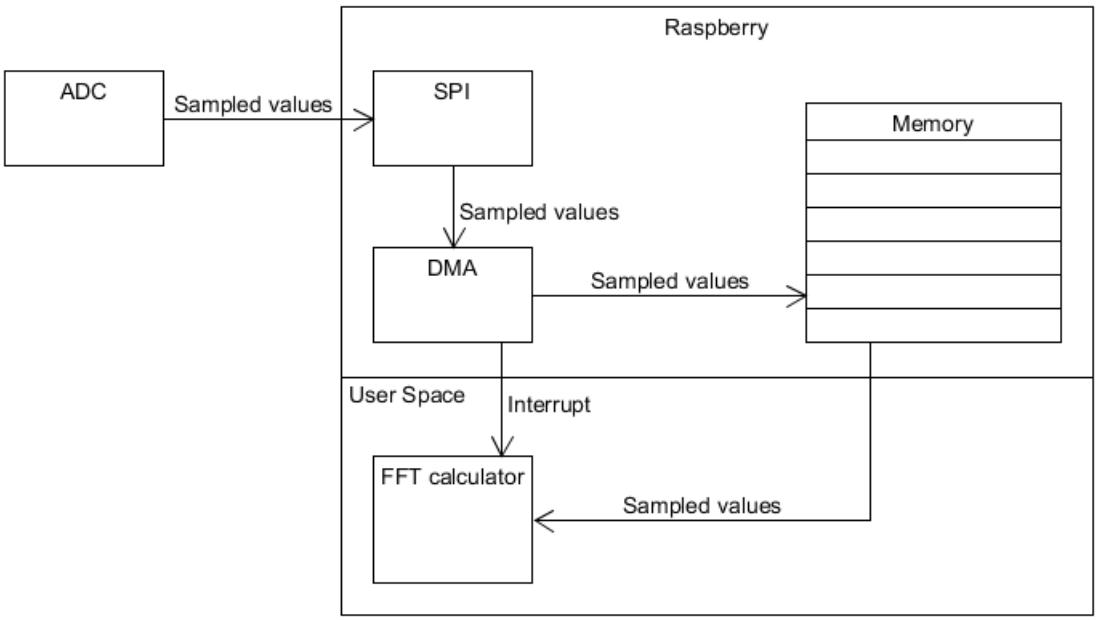


FIGURE 5.2: Pilote Linux pour un échantillonnage en temps réel

Ayant effectué une formation d'informaticien, la partie électronique aurait pu me poser certains problèmes. C'est pour cela que j'ai utilisé beaucoup d'énergie pour celle-ci. Ceci m'a permis d'apprendre beaucoup de choses. Je me sens maintenant à l'aise dans ce domaine.

Concernant la partie informatique, j'ai acquis beaucoup d'expérience. Par exemple, le mécanisme de compilation de librairies et de liage de celles-ci lors de la compilation d'un programme est maintenant beaucoup plus clair pour moi.

De plus, j'ai eu l'opportunité d'éveiller ma curiosité sur une thématique médicale en me confrontant à la partie physique mise en jeu. Et, le fait d'avoir pu travailler sur un projet concret dans la recherche médicale a été très excitant et m'a motivé tout au long de ce travail.

Chapter 6

Sources

- [1] Gert E. Nilsson, E. Göran Salerud, N.O Tomas Strömberg, Karin Wardell. *Laser Doppler Perfusion Monitoring and Imaging*
- [2] Francesco Marazzi. *Laser Doppler acquisition system with low quantization noise*
- [3] Francesco Marazzi, Frederic Truffer, Martial Geiser. *Laser Doppler Flowmetry acquisition system with low quantization error*
- [4] Martial Geiser, Gilbert Maître, Hugo Evéquoz. *Laser Doppler flowmetry of the foveal choroidal human eye : Experimental power spectrum analysic*
- [5] Charles E. Riva. *Basic principles of laser Doppler flowmetry and application to the ocular circulation*
- [6] http://www.omegawave.co.jp/en/products/oz/principle.shtml, Laser Blood Flow Imager
- [7] https://www.sonelecmusique.com/electronique_theorie_aop_audio.html, AOP et audio
- [8] http://www.daycounter.com/Filters/SallenKeyLPCalculator.phtml, Sallen-Key Low Pass Filter Calculator

Appendix A

Nouveau système

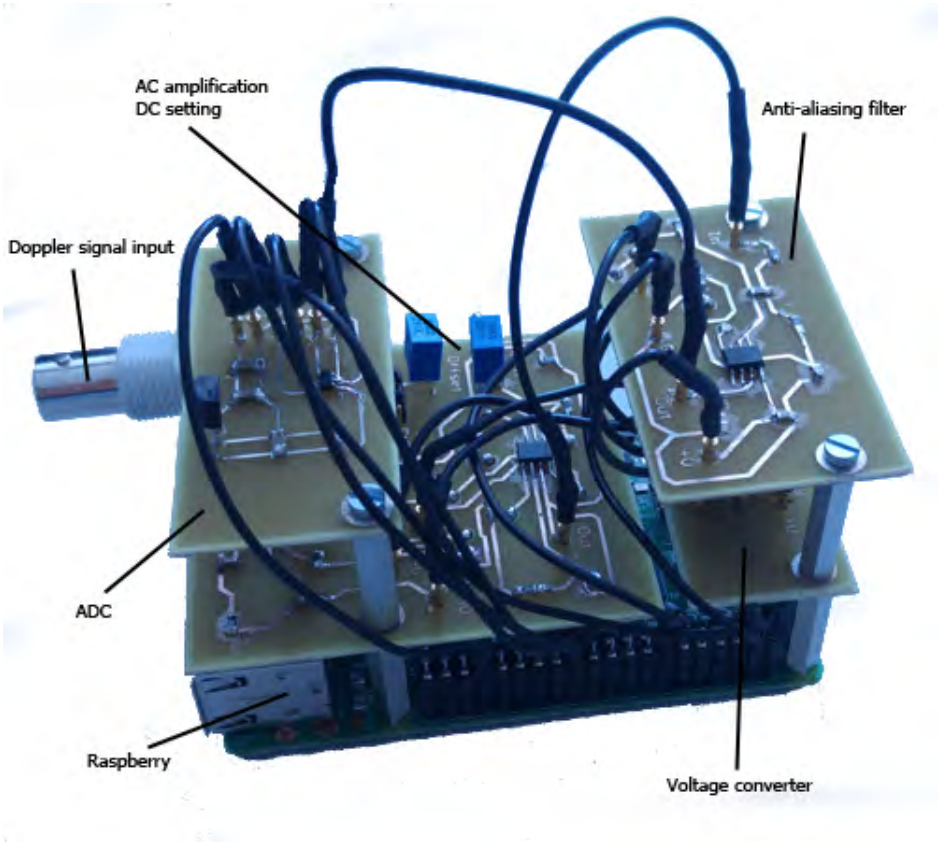


FIGURE A.1: Nouveau système

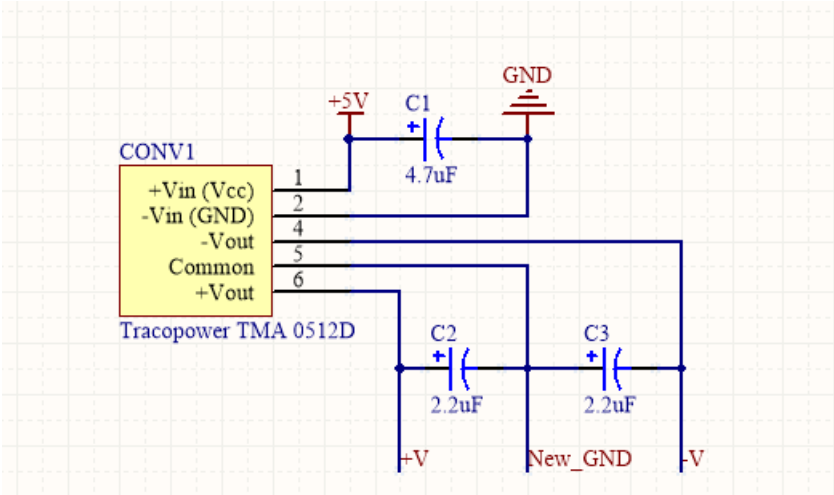


FIGURE A.2: Montage du convertisseur

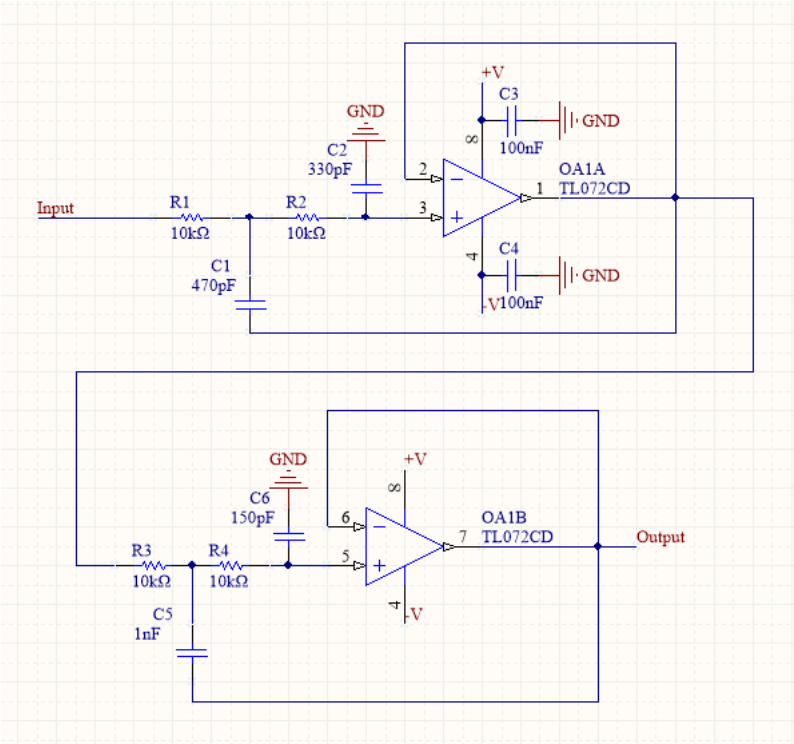


FIGURE A.3: Montage du filtre anti-repliement

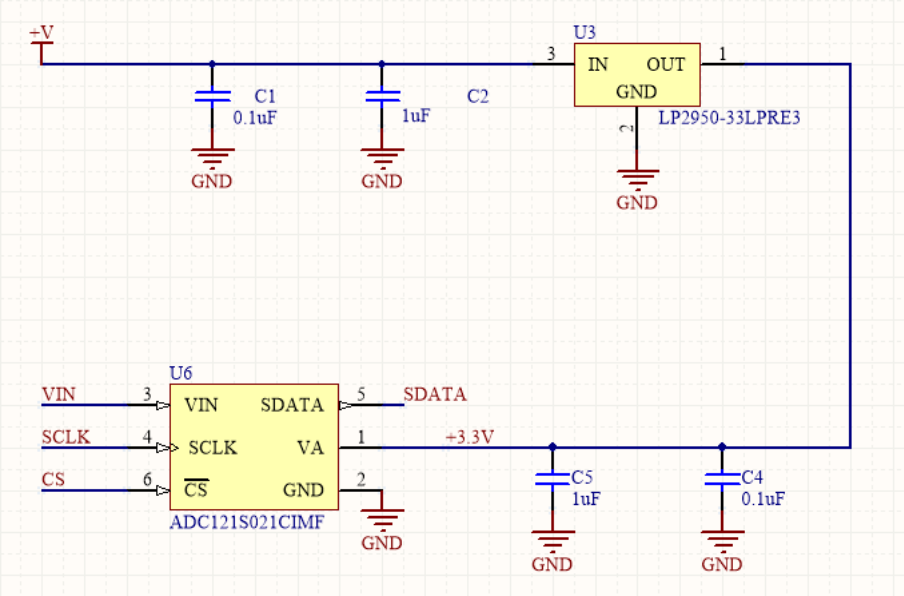


FIGURE A.4: Montage de l'ADC

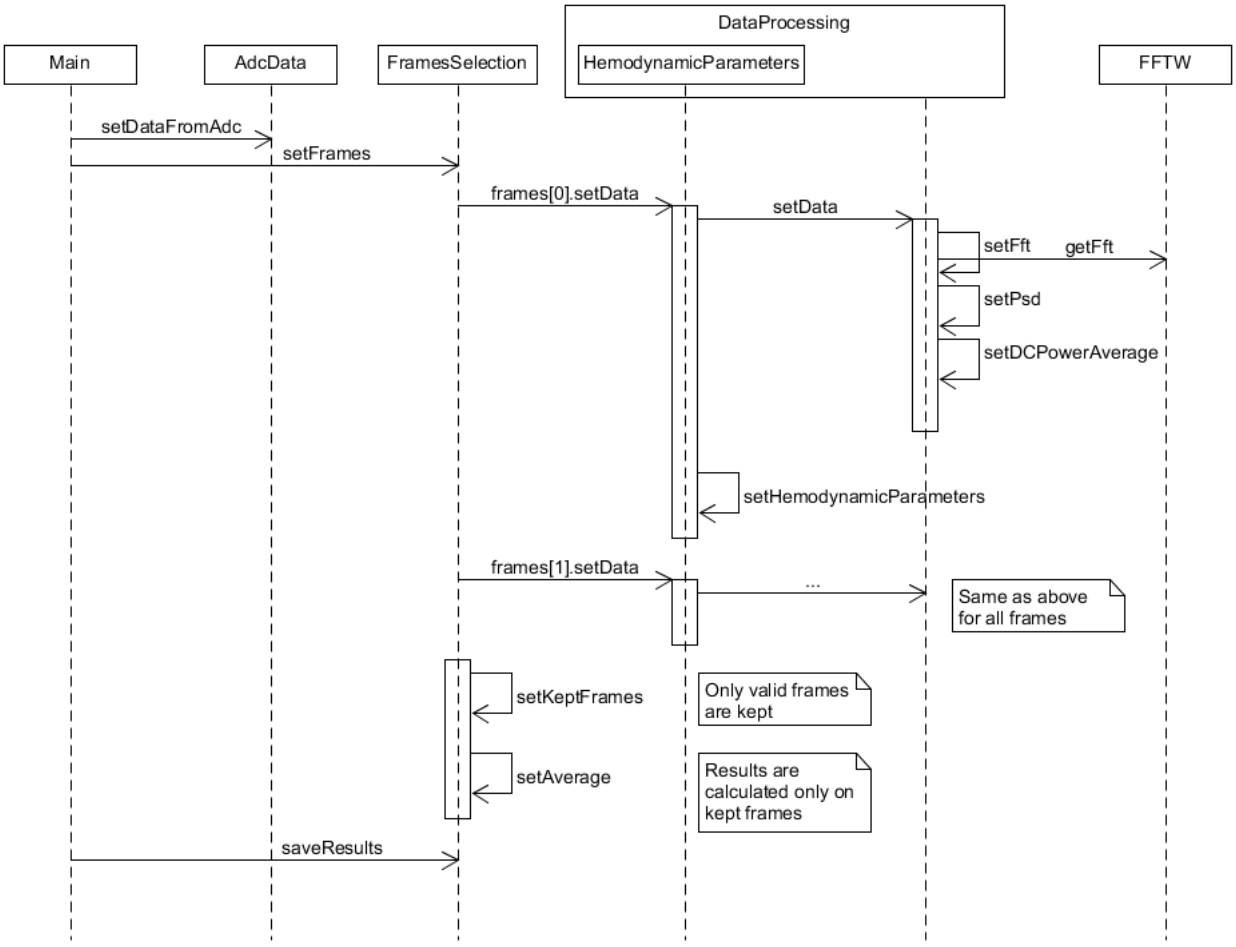


FIGURE A.5: Diagramme de séquence du programme GetExperiment

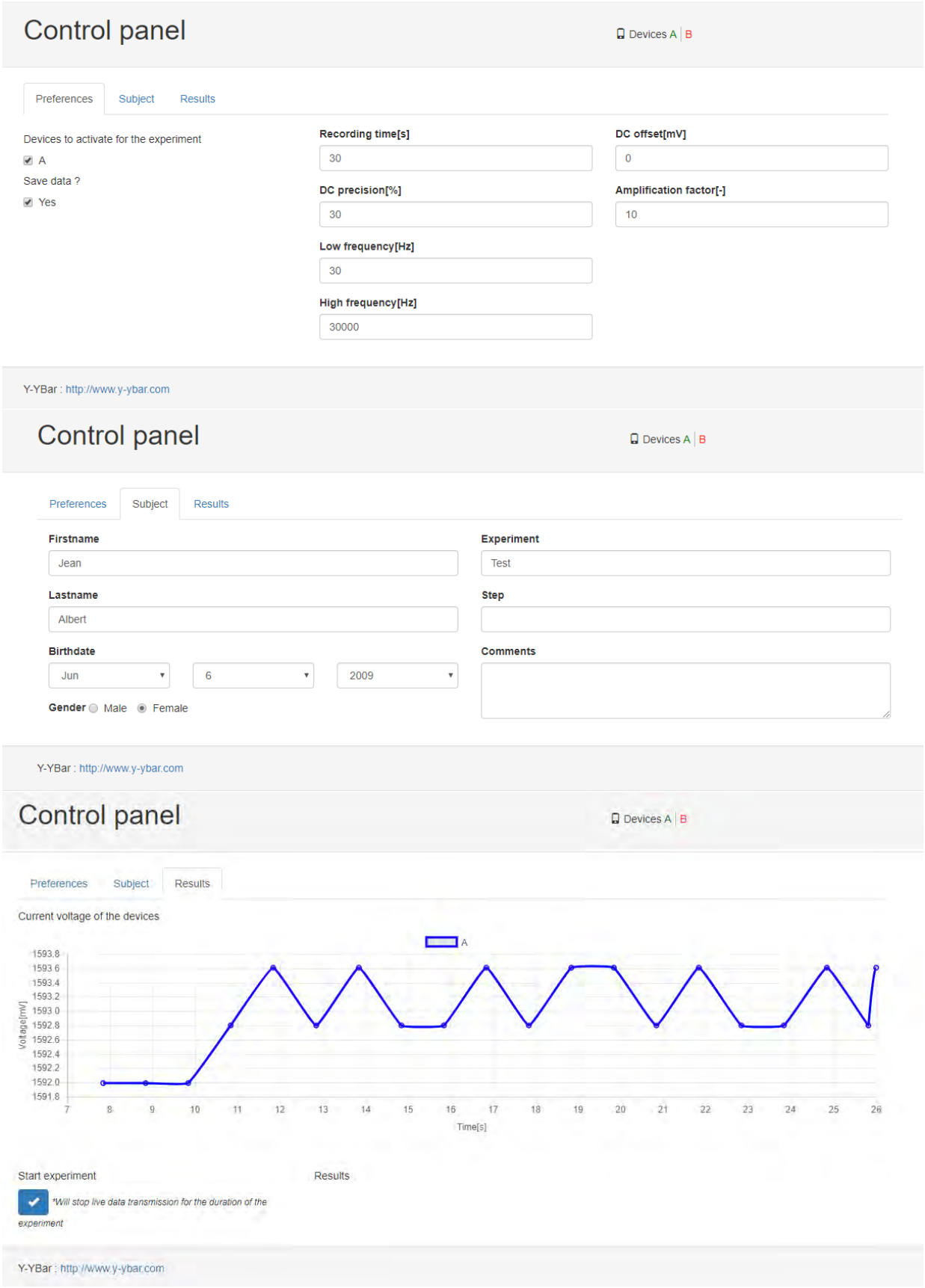


FIGURE A.6: Interface web

Appendix B

Dossier de projet

- **Divers** : Divers documents utiles pour le développement du système
- **Hardware** : Documents de l'électronique
  - **Choix_composants** : Document du choix de la carte embarquée
  - **Commandes** : Fichiers de commande des composants électroniques
  - **Schemas_electronique** : Schémas électroniques (CircuitMaker et Altium)
  - **Simulation_electronique** : Fichiers de simulation de l'électronique (LTspice)
- **Rapport** : Documents du rapport final
  - **annexes** : Annexes du rapport final
  - **documentation** : Documentation pour l'installation et l'utilisation du système
  - **Images** : Images utilisées dans le rapport final
  - **poster** : Documents du poster
  - **presentation** : Documents de la présentation
  - **Sauvegardes** : Anciennces version du rapport final
  - **Template** : Template LaTex du rapport final
- **Rapport_documents** : Documents qui ont été utiles pour l'élaboration du rapport
  - **Rapport_intermediaire_corrige** : Rapport intermédiaire corrigé et struture pour le rapport final
  - **Systeme_complet** : Documents de test du système complet avec le signal Doppler
  - **Tests_electronique** : Documents de test de la partie électronique
  - **Tests_GetExperiment** : Documents de test du programme GetExperiment
  - **_old** : Rapport intermédiaire et anciens documents de tests
- **Software** : Programmes
  - **App** : Programmes finaux
  - **_old** : Anciens programmes et programmes de test
  - **_Sauvegarde_App** : Anciennes versions des programmes finaux