Laboratoire TIMA – UGA - Grenoble

Rapport de stage de quatrième année du cycle ingénieur

Charrière Clément

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

Rapport technique

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

Table des matières

I. Introduction

Le contenu de ce rapport de rapport technique vient en complément d'un rapport de stage de 4ème année d'études d'ingénieur. Il explique les fonctionnalités implémentées permettant le pilotage d'une carte d'acquisition Red Pitaya STEMlab 125-14. Les fonctionnalités ont pour but de fournir une librairie de traitement du signal 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.

Dans une première partie, un prérequis expliquera comment démarrer correctement la Red Pitaya STEMlab 125-14. Dans une deuxième partie le rapport technique exposera une liste des fonctionnalités de l'API C Red Pitaya avec leurs explications. Dans une dernière partie, les fonctionnalités de l'application développées en C++ seront listés et décrites.

Le code du projet est disponible sur un dépôt public de Github accessible avec le lien suivant :
https://github.com/ClemtoClem/RedPitaya_SignalProcessing.git

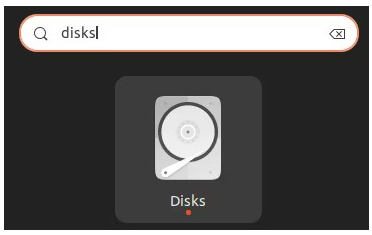
II. Prérequis

Ce prérequis explique comment démarrer pour la première fois une carte Red Pitaya STEMlab 125-14, afin de ne pas faire d'erreur. De plus, cette partie donnera quelles astuces pour entretenir le système d'exploitation et calibrer la carte. Plusieurs choix sont offerts à la manière de la mise en place de la chaîne de développement avec la plateforme Red Pitaya.

Dans cet ouvrage, l'installation des équipements nécessaire au démarrage de l'OS Red Pitaya s'effectue sur un ordinateur sur lequel est installé le système d'exploitation Linux Ubuntu version 22.04 et dont vous êtes le super utilisateur.

1. Préparer la carte SD

Afin de fonctionner la Red Pitaya STEMlab utilise une mémoire flash sur lequel est installé son système d'exploitation. Cette mémoire flash et une carte SD. Le système d'exploitation doit être installé via un ordinateur utilisant un support de lecture de carte SD. La documentation en ligne de la plateforme disponible au lien "https://Red_Pitaya.readthedocs.io/" propose à la rubrique "*2.3. Prepare SD card*" le dernier OS disponible.
Sur un ordinateur Ubuntu, ouvrez le site et cliquez sur le lien pour télécharger l'archive contenant l'image. Après téléchargement du le fichier zippé, décompressez-le dans un dossier, vous devrez y trouver un fichier avec l'extension '.iso' d'environ 8Go.
Ouvrez l'application Udisks.



Sélectionnez, la carte SD sur la liste de gauche. La carte SD peut déjà avoir été formaté préalablement, cliquez sur l'icon avec 3 petits point en haut à gauche de l'interface. Dans le menu sélectionnez "Formatez le disque". Une petite fenêtre s'ouvre, cliquez sur le bouton "Formater". Faites attention de sélectionner le bon périphérique de stockage, car vous pourriez par mégarde supprimer la partition hôte de votre système d'exploitation.
Maintenant, toutes les partitions ont été supprimées de la carte SD.
Pour installer l'OS sur celle-ci, sélectionnez à nouveau l'icon avec les 3 petits points, puis lancer un test de performance pour vérifier si la carte SD n'est pas endommagée.
Après cette étape, retournez dans le répertoire dans lequel vous avez dézippé l'image. Faite un cilc droit sur l'image ISO et cliqué sur "*Ouvrir avec Enregistreur d'images disque*", l'application Udisks devrez réapparaître au premier plan. Sélectionnez à nouveau la carte SD, puis lancez la récupération. Après la phase de chargement, votre carte sera enfin prête.

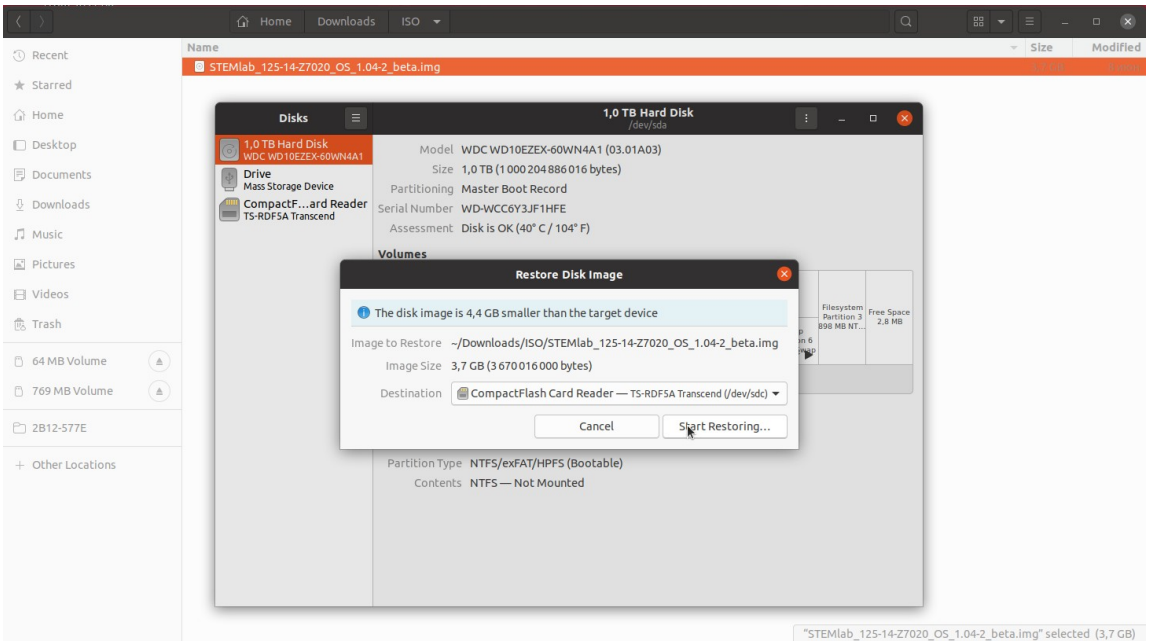
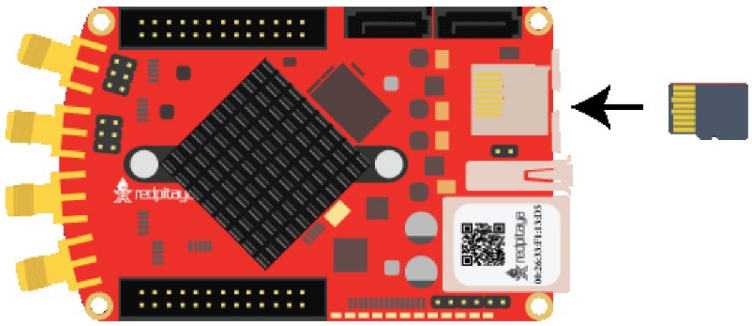


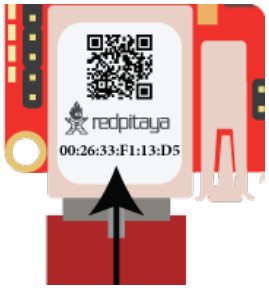
Figure 1: Sélection du disque et restauration de l'image

Si vous êtes sous Windows, utilisez l'application Rufus.

Maintenant éjecter la carte SD et insérez la carte SD dans l'emplacement prévu de la Red Pitaya, et connecté l'alimentation sur le connecteur PWR.



La led située sur le dessus doit s'allumer, attendez 2 minutes, le temps que le système d'exploitation se met en place puis branchez un câble internet de la Red Pitaya à votre machine ou à un réseau local, puis attendez quelques minutes.



Maintenant ouvrez un navigateur internet puis entrez l'adresse DHTC local de la Red Pitaya inscrite sur le haut de l'autocollant fourni dans le pack.
Une interface web comme Firefox, si la carte est connectée sur un rooter tapez votre IP Red Pitaya (exemple `10.42.0.172`) ou sinon si votre ordinateur est directement relié à la carte tapez l'URL Red Pitaya (exemple http://rp-f0baa5.local) et mettez le lien en favori pour ne pas avoir à le réécrire tout le temps.

2. Première connexion sur un terminal SSH

Afin de pouvoir échanger des fichiers entre votre ordinateur et contrôler la carte, puisque quelle ne possède pas d'écran, la création d'une connexion SSH sur un terminal est une méthode simple et sécurisé communiquer avec la carte.

Dans un terminal Ubuntu, saisissez la commande suivante pour établir la connexion à la carte avec le nom de votre carte. Dans notre exemple se sera :

```
ssh root@rp-f0baa5.local
```

Puisque c'est votre première connexion SSH, vous devez vous identifier. Le message suivant devrait s'afficher :

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:u9Em1vGF8MWTNDboKIz0Ae/HiMiwNVj6ln2QP0UH9ug.
Please contact your system administrator.
Add correct host key in /home/clement/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/clement/.ssh/known_hosts:3
  remove with:
  ssh-keygen -f "/home/clement/.ssh/known_hosts" -R "rp-f0baa5.local"
Host key for rp-f0baa5.local has changed and you have requested strict checking.
Host key verification failed.
```

Votre ordinateur doit mémoriser la clé d'identification de la Red Pitaya. Tapez le code suivant en indiquant le nom de votre ordinateur et de la Red Pitaya.

```
ssh-keygen -f "/home/monordinateur/.ssh/known_hosts" -R "rp-f0baa5.local"
```

Exécutez la commende puis écrivez "*yes*"

Pour ne pas à avoir à rentrer le mot de passe pour chaque commande, vous avez la possibilité d'enregistrer la clé publique SSH sur votre ordinateur, en indiquant l'IP de votre Red Pitaya.

```
ssh-copy-id root@10.42.0.172
```

Il vous sera demandé d'inscrire un mot de passe, tapez "*root*"

Et essayez d'établir à nouveau la connexion SSH.

```
ssh root@rp-f0baa5.local
```

Et voilà vous êtes connecté à la carte.

```
The authenticity of host 'rp-f0baa5.local (10.42.0.172)' can't be established.
ED25519 key fingerprint is SHA256:u9Em1vGF8MWTNDboKIz0Ae/HiMiwNVj6ln2QP0UH9ug.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'rp-f0baa5.local' (ED25519) to the list of known hosts.
root@rp-f0baa5.local's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-xilinx armv7l)

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

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: Wed Apr 17 11:50:12 2024
root@rp-f0baa5:~#
```

Maintenant, exécuter la commande ‘rw’ afin de rendre le système de fichiers exécutable.

On peut s’assurer que l’image FPGA est chargée, en exécutant la commande suivante :

```
overlay.sh v0.94
```

```
root@rp-f0baa5:~# overlay.sh v0.94
Time taken to load BIN is 169.000000 Milli Seconds
BIN FILE loaded through FPGA manager successfully
[1]+  Done                    systemctl start redpitaya_scpi
```

3. Redimensionner le système de fichiers

Lorsque l’image a été écrite sur la carte SD vous avez pu lire que la taille de la partition fait le même nombre de bit que l’image iso. Afin d'augmenter l'espace libre disponible pour utiliser pleinement l’espace de la carte SD, ouvrez un terminal SSH et exécutez le script suivant:

```
/opt/Red Pitaya/sbin/resize.sh
```

Une fois le script terminé, le système vous demandera de redémarrer votre Red Pitaya. Vérifiez la nouvelle taille de la partition en executant la commande suivante:

```
df -h
```

Si la taille de l’image n’a pas été modifiée vous pouvez essayer la commande suivante :

```
sudo resize2fs /dev/mmcblk0p2
```

En réinscrivant la commande df -h vous devrez avoir un résultat semblable :

```
root@rp-f0baa5:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        29G  5.5G   23G  20% /
tmpfs           231M     0  231M   0% /dev/shm
tmpfs            93M  4.0M   89M   5% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
none            5.0M  8.0K  5.0M   1% /var/log
/dev/mmcblk0p1  484M  365M  119M  76% /boot
```

Dans cet exemple, on peut observer que le la partition /dev/root fait maintenant 29 Go sur une SD de 32Go.

4. Mise à jour de l’OS

Pour mettre à jour le système d’exploitation, ouvrez l’interface web sur votre navigateur internet et ouvrez le menu "System". Sélectionnez l’application "Red Pitaya OS Update", et suivez les étapes.

A la fin des étapes, le système d'exploitation est toutes les applications sont maintenant à jour. Un message vous indiquera certainement que la version de l'OS n'est pas adaptée à la carte, ignorez-le.

Revenez sur le menu principal et éteigne la carte avec le bouton power off. Détranché la connexion Ethernet et l'alimentation, puis rebranché la carte pour vérifier si la connexion s'établit sans problème dans le navigateur web.

Un petit message d'avertissement "Red Pitaya OS x.x / STEMlab 125-14" a certainement dû apparaître au bas de l'écran. Cliquez sur le message. Le gestionnaire de mise à jour va s'ouvrir, suivez les étapes (10 minutes environ).

Dans un second temps, ouvrez un terminal SSH, et entrez les commandes suivantes pour mettre à jour les librairies installées sur l'OS :

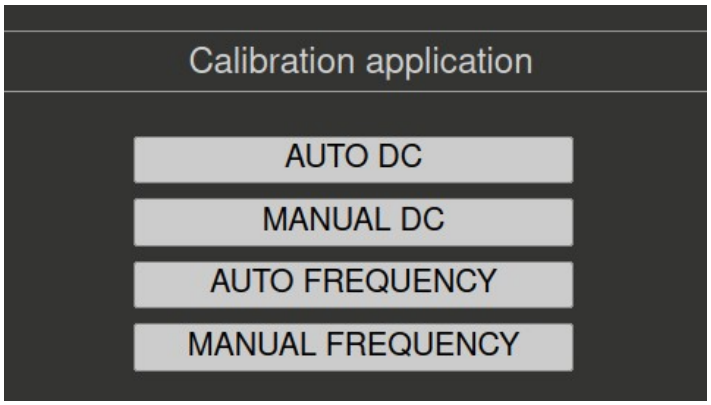
```
sudo apt-get update;
sudo apt-get upgrade
```

Réglage de l'heure :

```
sudo timedatectl set-timezone Europe/Paris
```

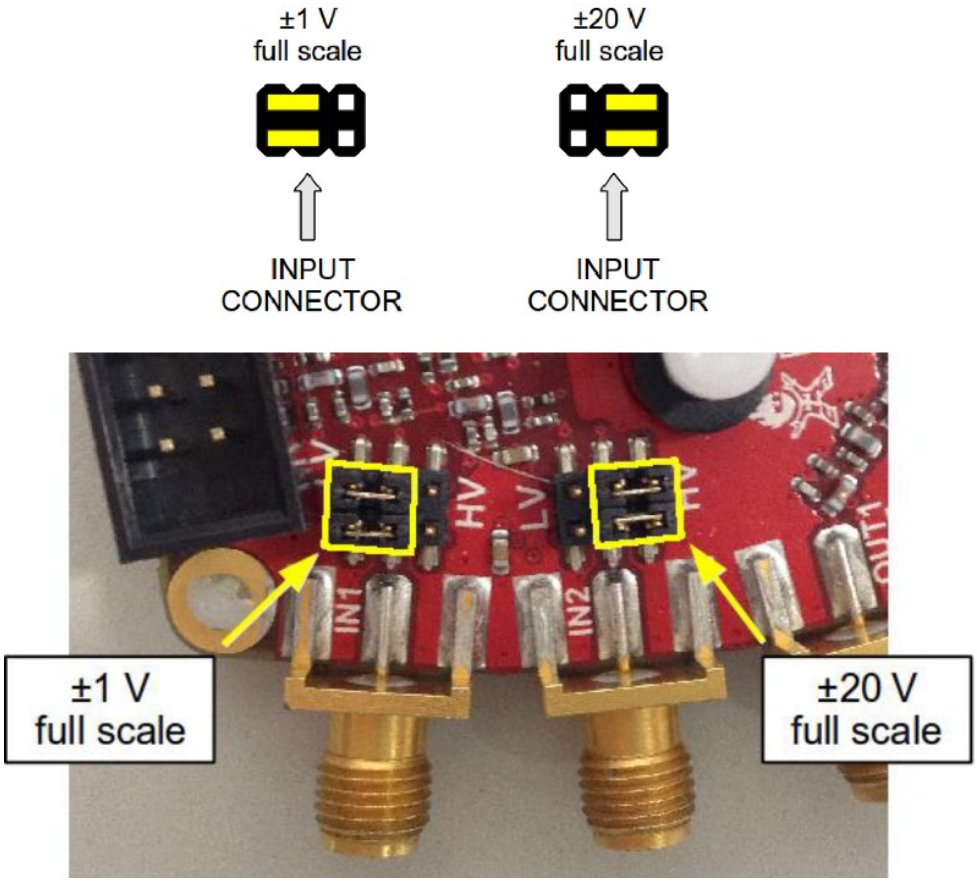
5. Calibration de la carte

Pour calibrer la carte, ouvrez l'interface web et exécutez l'application "Calibration" dans le menu système. Lorsque l'application s'ouvre vous verrez quatre options.



Pour cette étape, vous auriez besoin de deux câbles SMA ou BNC, deux adaptateurs SMA vers BNC si besoin, d'une liaison en T SMA ou BNC, et d'un générateur de tension avec une bonne précision.

Cliquez sur le bouton Auto DC pour commencer l'étalonnage en tension.
Cliquez sur le bouton Start, puis suivez les instructions, recommencer ce processus pour valider toutes les étapes de calibration.
Pour la calibration avec le 5 volts et le 0,5 volt, il est conseillé d'utiliser un générateur avec une bonne précision de lors de 10^{-4} volts, car il sera utilisé comme source étalon en tension.



Pour finir faîte une calibration en fréquence avec l'onglet 'Auto Frequency'.

III. Description des fonctionnalités de l'API C Red Pitaya

Dans les tableaux suivant vous retrouverait une liste des fonctions de l'API C Red Pitaya utilisées pour le projet.

1. Commandes d'initialisation

Fonction	**Description**
rp_Init()	Initialise et active l'interface de commande.
rp_IsApiInit()	Vérifiez si l'interface API est initialisée.
rp_Release()	Libérez les ressources de l'interface de commande.
rp_Reset()	Réinitialise les paramètres des broches numériques et analogiques ainsi que paramètres de génération et d'acquisition aux valeurs par défaut.

2. Commandes de versionnage

Fonction	Description
rp_IdGetID(uint32_t *id)	Renvoie l'ID de la carte Red Pitaya.
const char* rp_GetVersion()	Renvoie la version du tableau Red Pitaya.

3. Commande du générateur de tension

Fonction	Description
rp_GenReset()	Arrêter la génération et définit tous les paramètres du générateur aux valeurs par défaut.
rp_GenSynchronise()	Déclencher de manière synchrone la génération immédiate des deux sorties analogiques rapides. Les phases des signaux sont alignées.
rp_GenOutEnable(rp_channel_t channel) rp_GenOutDisable(rp_channel_t channel)	Activer/désactiver l'alimentation en tension de la sortie analogique rapide spécifiée. Lorsqu'il est activé, le signal ne commence à la valeur souhaitez, mais à une valeur de tension initiale.
rp_GenSetInitGenValue(rp_channel_t channel, float amplitude)	Configuration de la valeur initiale (0 par défaut).
rp_GenOutIsEnabled(rp_channel_t channel, bool *value)	Obtenir l'état d'activation/désactivation de la tension d'alimentation de la sortie analogique rapide spécifiée.
rp_GenTriggerSource(rp_channel_t channel, rp_trig_src_t src)	Définir la source de déclenchement pour le signal sélectionné. La source de déclenchement est le FPGA de la Red Pitaya par défaut.
rp_GenGetTriggerSource(rp_channel_t channel, rp_trig_src_t *src)	Obtenir la source de déclenchement.
rp_GenWaveform(rp_channel_t channel, rp_waveform_t type)	Définissez la forme d'onde d'une sortie analogique rapide. RP_WAVEFORM_SINE, RP_WAVEFORM_SQUARE, RP_WAVEFORM_TRIANGLE, RP_WAVEFORM_RAMP_UP, RP_WAVEFORM_RAMP_DOWN, RP_WAVEFORM_DC, RP_WAVEFORM_PWM, RP_WAVEFORM_ARBITRARY, RP_WAVEFORM_DC_NEG, RP_WAVEFORM_SWEEP
rp_GenGetWaveform(rp_channel_t channel, rp_waveform_t *type)	Obtenir la forme d'onde d'une sortie analogique rapide.
rp_GenFreq(rp_channel_t channel, float frequency)	Définir la fréquence du signal d'une sortie analogique rapide. Pour la forme d'onde arbitraire

Fonction	Description
	(RP_WAVEFORM_ARBITRARY), il s'agit de la fréquence d'une période de signal (un tampon de 16 384 échantillons).
rp_GenAmp(rp_channel_t channel, float amplitude)	Définir l'amplitude d'une sortie analogique rapide en volts. L'amplitude doit être inférieure à la plage de tension de sortie maximale (± 1 V).
rp_GenOffset(rp_channel_t channel, float offset)	Définir la tension de décalage CC d'une sortie analogique rapide en volts. L'amplitude de l'offset doit être inférieure à la plage de tension de sortie maximale (± 1 V).
rp_GenPhase(rp_channel_t channel, float phase)	Définir la phase d'une sortie analogique rapide en degrés. Le signal commence à être généré avec la phase spécifiée.
rp_GenDutyCycle(rp_channel_t channel, float ratio)	Définir le rapport cyclique d'une onde PWM.
rp_GenArbWaveform(rp_channel_t channel, float *waveform, uint32_t length)	Importez des données pour une période d'une forme d'onde arbitraire (doit contenir exactement 16 384 échantillons). Si moins d'échantillons sont fournis, la fréquence de sortie sera plus élevée.

4. Commandes pour l'acquisition

Fonction	Description
rp_AcqStart()	Démarrer le système d'acquisition.
rp_AcqStartCh(rp_channel_t channel)	Démarrer le système d'acquisition pour la voie spécifiée.
rp_AcqStop()	Arrêter le système d'acquisition.
rp_AcqStopCh(rp_channel_t channel)	Arrêter le système d'acquisition pour la voie spécifiée.
rp_AcqReset()	Arrêter le système d'acquisition et réinitialiser les paramètres avec les valeurs par défaut.
rp_AcqResetCh(rp_channel_t channel)	Réinitialiser la voie spécifiée.
rp_AcqSetDecimation(rp_acq_decimation_t decimation)	Choisir la décimation du système d'acquisition avec les énumérations : RP_DEC_1, RP_DEC_2, RP_DEC_4, RP_DEC_8, RP_DEC_16, RP_DEC_32, RP_DEC_64, RP_DEC_128, RP_DEC_256, RP_DEC_512, RP_DEC_1024, RP_DEC_2048, RP_DEC_4096, RP_DEC_8192, RP_DEC_16384, RP_DEC_32768, RP_DEC_65536
rp_AcqSetDecimationCh(rp_channel_t channel, rp_acq_decimation_t decimation)	Choisir la décimation pour la voie spécifiée.
rp_AcqGetDecimation(rp_acq_decimat	Obtenir la valeur de la décimation.

Fonction	Description
ion_t* decimation)	
rp_AcqGetDecimationCh(rp_channel_t channel, rp_acq_decimation_t* decimation)	Obtenir la valeur de la décimation pour la voie spécifiée.
rp_AcqSetDecimationFactor(uint32_t decimation)	Définir la décimation avec une valeur numérique entre 1 et 65536.
rp_AcqSetDecimationFactorCh(rp_channel_t channel, uint32_t decimation)	Définir la décimation avec une valeur numérique entre 1 et 65536 pour la voie spécifiée.
rp_AcqGetDecimationFactor(uint32_t* decimation)	Obtenir la décimation avec un valeur numérique.
rp_AcqGetDecimationFactorCh(rp_channel_t channel, uint32_t* decimation)	Obtenir la décimation avec une valeur numérique pour la voie spécifiée.
rp_AcqGetSamplingRateHz(float* sampling_rate)	Obtenir la fréquence d'échantillonnage du quartz en hertz.
rp_AcqSetGain(rp_channel_t channel, rp_pinState_t state)	Régler le gain du canal spécifié sur HIGH ou LOW.
rp_AcqGetGain(rp_channel_t channel, rp_pinState_t *state)	Obtenir le gain de la voie spécifiée.
rp_AcqGetGainV(rp_channel_t channel, float *voltage)	Obtenir le gain de la voie spécifiée en volt.
rp_AcqGetBufSize(uint32_t *size)	Obtenir la taille des buffers (16384).
rp_createBuffer(<maxChannels>, <length>, <initInt16>, <initDouble>, <initFloat>)	Effectue une allocation de mémoire et renvoie le tampon demandé. – <maxChannels> : combien de chaînes seront acquises – <lenght> : longueur du tampon en échantillons (max 16384) – <initInt16>, <initDouble>, <initFloat> : type des échantillons
rp_deleteBuffer(<buffer>)	Libére les ressources allouées. – <buffer> : tampon à libérer/libérer
rp_AcqGetBufferFillState(bool* state)	Renvoie 1 si tous les tampons d'acquisition sont complets, sinon 0.
rp_AcqGetBufferFillStateCh(rp_channel_t channel, bool* state)	Renvoie 1 si le tampon d'acquisition de la voie spécifiée est complet, sinon 0.
rp_AcqSetTriggerHyst(float voltage)	Définissez la valeur du seuil d'hystérésis de déclenchement en Volts.
rp_AcqGetTriggerHyst(float* voltage)	Obtenir la valeur du seuil d'hystérésis de déclenchement en Volts.
rp_AcqSetTriggerLevel(rp_channel_trigger_t channel, float voltage)	Régler le niveau de déclenchement en V.

Fonction	Description
rp_AcqGetTriggerLevel(rp_channel_trigger_t channel, float* voltage)	Obtenir le niveau de déclenchement en V.
rp_AcqGetWritePointer(uint32_t* pos)	Obtenir la position actuelle de l'échantillon le plus récent dans le tampon.
rp_AcqGetWritePointerAtTrig(uint32_t * pos)	Obtenir la position où l'événement déclencheur est apparu.
rp_AcqGetWritePointerCh(rp_channel_t channel, uint32_t* pos)	Obtenir la position actuelle de l'échantillon le plus récent dans le tampon pour la voie spécifiée.
rp_AcqGetWritePointerAtTrigCh(rp_channel_t channel, uint32_t* pos)	Obtenir la position où l'événement déclencheur est apparu pour la voie spécifiée.
rp_AcqGetDataV(rp_channel_t channel, uint32_t pos, uint32_t* size, float* buffer)	Lire les 'size' échantillons du tampon à partir de la position 'start_pos' en volts

IV.Description des fonctionnalités développées

Le code du projet est disponible sur un dépôt public de Github accessible avec le lien suivant : https://github.com/ClemtoClem/RedPitaya_SignalProcessing.git

1. 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 définie 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 impulsionnelles 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.

Si nous souhaitons réutiliser le filtre avec des valeurs venant d'un autre signal, le filtre devra être précédemment réinitialisé. L'algorithme suivant indique comment est implémentée la classe C++ du filtre FIR.

```
Classe FIRFilter
    Variables:
        - coefficients: Liste de nombres réels
        - input_buffer: Liste de nombres réels
        - order: Entier
        - isSetup: Booléen

    Méthodes:
        - FIRFilter()
            Initialiser isSetup à faux

        - FIRFilter(coefficients: Liste de nombres réels)
            Initialiser coefficients avec les coefficients donnés
            Initialiser order avec la taille des coefficients
            Initialiser isSetup à vrai

        - setCoefficients(coefficients: Liste de nombres réels)
            Mettre à jour coefficients avec les nouveaux coefficients
            Mettre à jour order avec la taille du tableau coefficients
            Appeler la méthode reset()
            Initialiser isSetup à vrai

        - reset()
            Redimensionner input_buffer à la taille de order, initialiser avec des zéros

        - apply(x: Réel) -> Réel
            Si isSetup est faux
                Afficher "Erreur: FIR filter is not setup."
                Retourner x

            Si order est 0
                Retourner x

            Décaler les valeurs de input_buffer vers la droite
            Placer x en première position de input_buffer

            Initialiser y à 0.0
            Pour chaque coefficient dans coefficients
                Ajouter coefficient * valeur correspondante de input_buffer à y

            Retourner y
```

Tableau 1: Pseudo-code du filtre à réponse impulsionnelle finie

2. Filtre à réponse impulsionnelle infinie

L'algorithme suivant indique comment est implémentée la classe C++ du filtre IIR.

```
Classe IIRFilter
    Variables:
        - coefficients_a: Liste de nombres réels
        - coefficients_b: Liste de nombres réels
        - input_buffer: Liste de nombres réels
        - output_buffer: Liste de nombres réels
        - order_a: Entier
        - order_b: Entier
        - isSetup: Booléen

    Méthodes:
        - IIRFilter()
            Initialiser isSetup à faux

          - IIRFilter(coefficients_a: Liste de nombres réels, coefficients_b: Liste de nombres
réels)
            Initialiser coefficients_a avec les coefficients_a donnés
            Initialiser coefficients_b avec les coefficients_b donnés
            Initialiser order_a avec la taille des coefficients_a
            Initialiser order_b avec la taille des coefficients_b
            Initialiser isSetup à vrai

          - setCoefficients(coefficients_a: Liste de nombres réels, coefficients_b: Liste de
nombres réels)
            Mettre à jour coefficients_a avec les nouveaux coefficients_a
            Mettre à jour coefficients_b avec les nouveaux coefficients_b
            Mettre à jour order_a avec la taille du tableau coefficients_a
            Mettre à jour order_b avec la taille du tableau coefficients_b
            Appeler la méthode reset()
            Initialiser isSetup à vrai

        - reset()
            Redimensionner input_buffer à la taille de order_a, initialiser avec des zéros
            Redimensionner output_buffer à la taille de order_b, initialiser avec des zéros

        - apply(x: Réel) -> Réel
            Si isSetup est faux
                Afficher "Erreur: IIR filter is not setup."
                Retourner x

            Si order_a est 0 et order_b est 0
                Retourner x

            Décaler les valeurs de input_buffer vers la droite
            Placer x en première position de input_buffer

            Initialiser result1 à 0.0
            Pour chaque coefficient dans coefficients_a
                Ajouter coefficient * valeur correspondante de input_buffer à result1

            Initialiser result2 à 0.0
            Pour chaque coefficient dans coefficients_b
                Ajouter coefficient * valeur correspondante de output_buffer à result2

            Calculer y comme étant result1 - result2

            Décaler les valeurs de output_buffer vers la droite
            Placer y en première position de output_buffer
```

Retourner y

Tableau 2: Filtre à réponse impulsionnelle infinie

V. Code

1. Procédure de compilation et d'exécution du programme

Comme détaillé précédemment, le code a besoin d'être transvasé sur la Red Pitaya afin d'être compilé avec toutes les librairies de l'PI C, déjà pré compilé sur la carte. Pour cela vous devez vous connecter en SSH sur un terminal, créer un répertoire du nom du projet dans le répertoire racine 'root' de la Red Pitaya. L'exécution du script shell ' send_all.sh' sur votre ordinateur se chargera de copier tous le code source vers la Red Pitaya avec les fichiers de compilations.

Pour compiler le programme, vous avez la possibilité de taper la commande 'make' qui fait appel au fichier Makefile présent à la racine du projet. Cependant, l'exécution de cette commende peut parfois remplir votre terminal de ligne de log. Puisque, votre terminal est limité en nombre de ligne visible, vous pouvez perdre les logs afficher en premier, qui sont souvent les plus communicatifs de l'erreur de compilation. Pour faire face à cela le script shell './compile.sh' créer un fichier 'output.log' dans lequel sont inscrit tous les logs.

Après succès de la compilation, vous pouvez à présent exécuter le code dans le terminal SSH.
Pour cela vous écrivez le nom du programme crée que vous pouvez vérifier en écrivant la commande 'ls' qui liste le contenu du répertoire.
Si vous exécutez directement le code, il s'affichera le message 'Write 'help' to see available tests or modules.', qui vous indique d'entrer la commande help pour lister toutes les fonctionnalités proposées.

Les fonctionnalités proposées se séparent en deux catégories, les fonctionnalités de test et les modules.
Au moment de l'écriture du programme, 6 fonctionnalités de tests ont été développées, qui sont :

- acquire
- spectrum
- demodulation
- demodulation2
- realTimeAcquisition
- realTimeAcquisition2

Et un module est disponible :

- frequencyScanning

Une seule des fonctionnalités peut être exécutée à la fois. Il vous suffit d'écrire le nom du programme suivi du nom de la fonctionnalité et d'une liste de paramètres. Afin d'être reconnue tous les arguments du programme doivent être séparés par un espace. Comme précédemment afin de connaître les paramètres disponibles pour une fonctionnalité, vous pouvez écrire un seul paramètres

‘help’ et exécuter le programme. Il s’affichera dans le terminal, une petite description de la fonctionnalité suivie de la liste des paramètres utilisable avec leur description.
Afin de personnaliser l’exécution d’une fonctionnalité, comme choisir l’amplitude ou la fréquence, vous pouvez inscrire seulement les paramètres qui vous semblent utilise, les autres seront initialisés avec une valeur par défaut.

À la fin de son exécution, le programme créer dans le répertoire ‘,/data’ un nouveau répertoire pour contenir les fichiers de données. Le répertoire a pour nom, la date et l’heure à lequel la première donnée est écrite. Ce répertoire contiendra aussi un fichier de traçabilité, qui n’est rien d’autre qu’un fichier de configuration méconnaissable par son extension ‘.cfg’. Il indique le nom du test ou du module qui a été choisie avec les valeurs de tous les paramètres et les valeurs de mesure.

Afin d’afficher les données, vous devez les importer sur votre ordinateur en exécutant le script ‘./recup_data.sh’. Afin de pas prendre trop de mémoire sur la Red Pitaya, la commande videra le répertoire ‘./data’ après les avoir transvasées sur votre ordinateur.

Afin de simplifier l’affichage, un script python ‘display.py’ vous permet d’afficher vos données. Il vous sera demandé de sélectionner le répertoire avec les données que vous voulez afficher, et le programme se chargera automatiquement de choisir les bons axes des graphes.

2. Arborescence du projet

Voici l’arborescence nécessaire au bon fonctionnement du projet :

```
├── compile.sh
├── connect.sh
├── python
│   └── display.py
├── recup_data.sh
├── recup_output_make.sh
├── send_all.sh
├── send_code.sh
├── send_main.sh
├── send_modules.sh
├── send_tests.sh
└── src
    ├── CSVFile.cpp
    ├── CSVFile.hpp
    ├── Demodulator.cpp
    ├── Demodulator.hpp
    ├── Filter.cpp
    ├── Filter.hpp
    ├── Noise.cpp
    ├── Noise.hpp
    ├── PID.cpp
    ├── PID.hpp
    ├── Signal.cpp
    ├── Signal.hpp
    ├── Spectrum.cpp
```

```
├── Spectrum.hpp
├── Window.cpp
├── Window.hpp
├── acquisition.cpp
├── acquisition.hpp
├── globals.hpp
├── main.cpp
├── modules.cpp
├── modules.hpp
├── tests.cpp
├── tests.hpp
├── utils.cpp
└── utils.hpp
```

3. Description des scripts shell

Les scripts shell (‘.sh’) on était écris dans le but de simplifier la communication avec la Red Pitaya. Voici leurs descriptions :

- ‘connect.sh’ permet de se connecter à la Red Pitaya (changer l’ID en fonction de votre carte)
- ‘recup_data.sh’ permet de copier dans le répertoire ‘./data’ sur l’ordinateur le contenu du fichier ‘./data’ du projet sur la Red Pitaya
- ‘recup_output_make,sh’ permet de récupérer le fichier de débogage générée par le script ‘compile.sh’ lorsqu’il est exécuté sur la Red Pitaya
- ‘send_all.sh’ envoie le code source du répertoire ‘./src’, le fichier Makefile de compilation et le script ‘compile.sh’ dans le répertoire du projet sur la Red Pitaya
- ‘send_code.sh’ envoie le code source du répertoire ‘./src’ dans le répertoire du projet sur la Red Pitaya
- ‘send_main.sh’ envoie seulement le fichier ‘./src/main.cpp’ sur la Red Pitaya.readthedocs.io
- ‘send_modules.sh’ envoie seulement les fichiers ‘./src/modules.hpp’ et ‘./src/modules.cpp’ sur la Red Pitaya
- ‘send_tests.sh’ envoie seulement les fichiers ‘./src/tests.hpp’ et ‘./src/tests.cpp’ sur la Red Pitaya

4. Descriptions des fichiers sources c++

Par convention, j’ai choisi d’inclure tous les fichiers sources dans le répertoire ‘./src’. Deplsu tous les fichiers source décrivant une classe spécifique possède le nom de la classe avec une

majuscule en première lettre, les autres fichiers ne définissant pas de classe commencent par une minuscule.

Dans le projet vous pourrez apercevoir un répertoire ‘./include’, il n’est pas utile pour le projet, mais permet à l’éditeur de code vscode de ne pas surligner toutes les fonctions prédéfinies de la Red Pitaya inexistante sur votre ordinateur. De plus dans le répertoire ‘.src/’ se trouvent un répertoire ‘Base’, c’était un autre projet qui visé à réécrire toutes les fonctionnalités de la Red Pitaya en C++. Comme ce projet n’est pas terminé, les fonctionnalités de l’application ne l’utilisent pas. Vous pouvez ignorer ce répertoire.

Les premières instructions exécutées par l’application sont disponibles dans le fichier ‘main.cpp’ qui possède la fonction d’entrée de l’application, c’est elle qui récupéra tous les paramètres de l’application et se chargera de la redistribuer à fonctionnalité choisie.

En tant que tel, le nom des fichiers décrivent bien leur utilité.

Le fichier ‘globals.hpp’ créer toutes les constantes nécessaires au fonctionnement du programme.

Les fichiers ‘Signal.hpp’ et ‘Signal.hpp’ implémente une classe Signal offrant tous les outils pour créer un signal, et effectuer des opérations sur ses données ou entre plusieurs signaux.

De même, les fichiers ‘Spectrum.hpp’ et ‘Spectrum.hpp’ implémente une classe Spectrum offrant tous les outils pour créer un spectre, et effectuer des opérations sur ses données ou entre plusieurs spectres.

Voici un exemple d’utilisation :

```
Signal s1(16384, "s1(t)") ;
s1.generateWeveform(RP_WAVEFORM_SINE, 0.8, 1e3) ;
Signal s2(16384, "s2(t)") ;
s1.generateWeveform(RP_WAVEFORM_SINE, 0.2, 10e3) ;
Signal s3("s3(t)") ;
s3 = s1+s2 ; // additionne les deux signaux
Spectrum S1("S1(f)") ;
s1.FFT(S1) ; // calculer la transformée de Fourier du signal s1(t)
Spectrum S2("S1(f)") ;
s2.FFT(S2) ; // calculer la transformée de Fourier du signal s2(t)
Spectrum S3("S1(f)") ;
s3.FFT(S3) ; // calculer la transformée de Fourier du signal s3(t)
```

Pour enregistrer les signaux et spectres chez choisie le format CSV. La classe CSVFile permet d’inscrire dans le fichier en axes (temps, fréquence…) en fonction du type de données inscrite. Sur les lignes suivantes est inscrit la liste de signaux avec leur nom et leurs valeurs espacées par ne virgule.

Si vous souhaitez plus de détail sur le code en tant que tel, des commentaires décrivent l’utilisation des fonctions. La dernière version du code avec toutes les annotations sera le travail de la dernière semaine de stage.

Merci.