

**Rapport technique**

**Projet BitScope**

Par

Nicolas Morisseau

Abdellah Ben Rahmoun

Valentin Beau

Étudiants en 5ème année

À l’INSA Centre Val de Loire (Promotion 2018)

Filière GSI option SA3I – ISI

3 rues de la chocolaterie, 41000 Blois, France

Janvier 2018

# RÉSUMÉ

Dans le cadre de notre cursus scolaire au sein de l’INSA Centre Val de Loire, nous avons pour projet de développer un programme en langage python capable de recevoir, traiter et afficher les signaux reçus par un BitScope Micro à travers un Raspberry PI.

Ce document retrace le travail effectué durant 3 mois sur le projet BitScope. Il regroupe notamment les détails du programme, les étapes du projet, les difficultés rencontrées, le manuel d’utilisation et les pistes d’amélioration.

# SOMMAIRE

[RÉSUMÉ 2](#_Toc503979652)

[SOMMAIRE 3](#_Toc503979653)

[INTRODUCTION 4](#_Toc503979654)

[1. Étude du besoin 4](#_Toc503979655)

[2. But et Objectifs 4](#_Toc503979656)

[I. Développement du projet 5](#_Toc503979657)

[I.1. Ressources 5](#_Toc503979658)

[I.2. Travail effectué 5](#_Toc503979659)

[I.3. Difficultés rencontrées 8](#_Toc503979660)

[II. Manuel d’utilisation 9](#_Toc503979661)

[II.1. Exécution du programme avec paramètres 9](#_Toc503979662)

[II.2. Fonctions secondaires 10](#_Toc503979663)

[II.2.1. Aide 10](#_Toc503979664)

[II.2.2. Réglage de la durée d’acquisition 10](#_Toc503979665)

[II.2.3. Choix du mode 11](#_Toc503979666)

[II.2.4. Enregistrer 11](#_Toc503979667)

[II.2.5. Affichage des valeurs 11](#_Toc503979668)

[II.2.6. Affichage des temps 11](#_Toc503979669)

[II.2.7. Affichage d’informations 12](#_Toc503979670)

[III. Organisation du code 13](#_Toc503979671)

[IV. Restes à faire 13](#_Toc503979672)

[BILAN 14](#_Toc503979673)

# INTRODUCTION

## 1. Étude du besoin

Pour nos études à l’INSA Centre Val de Loire, afin d’apprendre à maitriser le domaine de l’instrumentation virtuelle, on nous demande de travailler sur des projets précis. Nous avons donc choisi ce sujet et c’est donc principalement pour cela que le projet existe. Cependant, en dehors de ce contexte, le projet a une vraie valeur ajoutée.

Cette étude s’inscrit effectivement dans une problématique de portabilité des oscilloscopes souvent trop encombrants. En effet, l’idée est de faire fonctionner l’oscilloscope embarqué BitScope construit par l’entreprise du même nom avec le mini-ordinateur Raspberry PI. Peu chères, très légers, très petits et donc portables, ces appareils sont utilisables n’importe où et accessibles par tous. L’intérêt du Raspberry est aussi qu’il fonctionne sous l’OS Raspbian (basé sur Debian) et offre donc plus de souplesse et de possibilités qu’un ordinateur classique avec Windows.

Par exemple, avec un tel matériel, les techniciens, chercheurs ou encore ingénieurs pourront se déplacer plus facilement et travailler chez le client avec leurs propres outils.

L’intérêt est aussi de fournir une base de code qui pourra être réutilisée et réécrit par d’autres développeurs pour leur propre besoin.

## 2. But et Objectifs

Les programmes et librairies fournis avec le BitScope étant génériques et souvent compliqués d’utilisation (voir même inexistante pour l’environnement Debian), notre mission est donc d’écrire nous-mêmes les scripts. Ces scripts sont codés en langage python, car ce langage est interprété, orienté objet, rapide et facile à manipuler.

L’objectif principal est de créer un programme capable de recevoir les signaux en sorties du BitScope, les traiter et les afficher. Ce programme ne comprend pas d’interface graphique, il fonctionne uniquement par un terminal.

# I. Développement du projet

## I.1. Ressources

La toute première tâche réalisée est de recenser les ressources à notre disposition.

* Matériels

Fournis par l’INSA Centre Val de Loire, nous avons un kit BitScope, un kit Raspberry et un écran portable. Ce dernier n’est finalement pas utilisé, car pour des raisons pratiques on préfère utiliser nos propres écrans. Nous utilisons aussi les générateurs de basses fréquences (GBF) de l’école pour générer des signaux.

Pour plus de détails, voir l’annexe Nomenclature.pdf.

* Logiciels

Pour le coté programmation en langage python, on utilise l’IDE gratuit *PyCharm*.

Le Raspberry fonctionnant sous le système d’exploitation GNU/Linux-Debian, on implémente nos codes sous ce même système. Pour cela, certains d’entre nous utilisent l’application *Ubuntu* sous Windows (machine virtuelle) et d’autres font un Dual Boot (cohabitation Windows/Ubuntu).

Pour connecter nos ordinateurs au Raspberry, on utilise le programme *PuTTY* pour gérer les protocoles SSH.

Enfin, pour mieux travailler en groupe, on enregistre nos travaux sur *GitHub* qui est une plateforme en ligne spécialement conçue pour la programmation.

## I.2. Travail effectué

Notre première approche du sujet est d’étudier le BitScope et le Raspberry Pi. En se répartissant les tâches, on parcourt les documentations techniques et la librairie du BitScope et, d’une autre part, on branche et test l’interface du Raspberry.

L’entreprise BitScope ne fournit qu’une seule librairie, Bitlib. Bien que limitée, celle-ci est très importante puisque c’était notre seul moyen de communiquer avec l’appareil. Après avoir bien saisi son fonctionnement, on mène une première réflexion de groupe afin de déterminer ce qu’on devait développer. On convient alors qu’il faut développer un premier bout de code, le plus important, qui permet de détecter le BitScope, de l’activer et de récupérer les données. On travaille tous les 3 dessus chacun de son côté pour arriver à une première version. À la fin, on arrive alors récupérer un tableau de point sur une plage donnée.

Une fois cela fonctionnel on se répartit les tâches pour développer ces 3 parties :

* Traitement des données
* Enregistrement des données
* Affichage des données
* Générer/simuler un signal (permets de pouvoir travailler chez soi sans dépendre des générateurs de l’école)

Avec ce code, on obtient des informations sur le BitScope utilisé et un tableau des points mesurées au moment de l’exécution du code.

Avec le générateur, on envoie un signal dont les caractéristiques sont les suivantes :

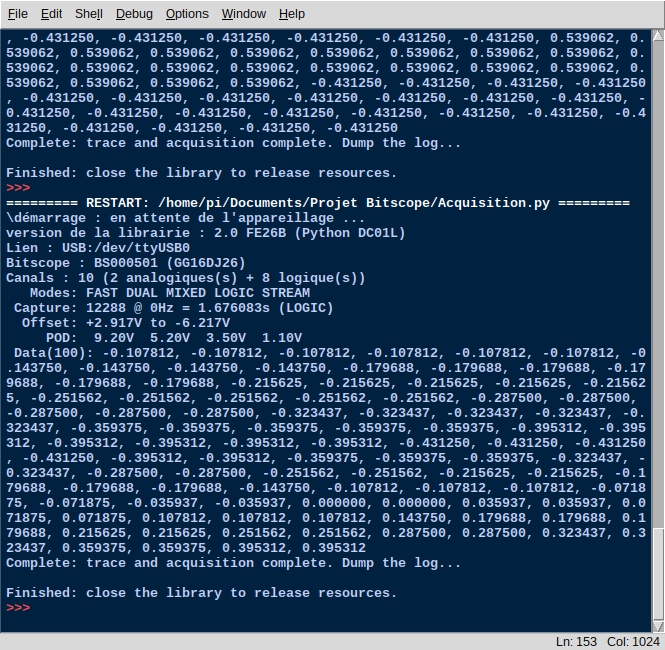
• Signal Triangle

• Amplitude: 1V peak to peak

• Duty: 10 %

• Fréquence: 5kHz

• Offset: 0V

Soit avec la commande Acquisition(100,5000) dans le terminal,

Les résultats sont conformes au fonctionnement attendu, c’est-à-dire que le programme récupère la valeur de la tension à une fréquence donnée, et ces valeurs sont en accord avec celles envoyées par le GBF.

La 4eme partie est finalement abandonnée, car trop compliquée. Nous nous sommes rendu compte que cela sortait du cadre du projet et qu’il valait mieux concentrer nos efforts sur le reste. Nous restons donc sur l’utilisation des générateurs.

La seconde étape est maintenant d’obtenir le même résultat, mais en temps réel. À ce stade, on décide de se réunir pour discuter de la suite des évènements.

On se répartit les tâches pour développer ces 3 nouvelles parties :

* Réception en temps réel
* Affichage des données amélioré (graphique)
* Paramétrage de la mesure par l’utilisateur (auparavant le paramétrage s’effectuait directement dans le code)
* Section help (informations)

Une fois implémenté, on obtient alors le programme complet. Le programme affiche en temps réel via un thread les signaux reçus par le BitScope dans un graphique.

Soit,

**(insert screen signal)**

## I.3. Difficultés rencontrées

La première difficulté rencontrée fut de maitriser le langage python. Nous sommes pour la plupart débutants et nous nous sommes donc renseignés sur internet. Heureusement le langage est relativement facile à comprendre et d’autant plus grâce aux connaissances acquises à l’INSA Centre Val de Loire (apprentissage de d’autres langages comme le C, C++ et Java).

La compréhension de la documentation de la librairie Bitlib est aussi un challenge. Entièrement rédigé en anglais et assez conséquente avec beaucoup de termes techniques nous avons passé du temps ensemble pour la comprendre et la maitriser. Cependant, nous n’avons pas réussi à utiliser cette librairie pour générer un signal (cf I.2. Travail effectué). Nous voulions récupérer d’autres libraires développées par l’entreprise BitScope elle-même qui permet cela (BitGen). Mais après avoir contacté la compagnie située en Australie, nous avons appris que seul Bitlib est fonctionnel avec le BitScope que nous avons. Cela réduit donc considérablement nos ressources.

Pour plus de confort, nous avons aussi mis en place des procédures pour travailler chez soi à distance. Après quelques problèmes au début, cela fut vite résolu avec beaucoup de communication et avec la maitrise du logiciel GitHub.

Une autre difficulté est d’appréhender l’environnement Debian du Rasberry. Une fois de plus, la plupart d’entre nous sont débutants avec ce système d’exploitation. Une période d’apprentissage et d’installation a donc été nécessaire.

Enfin, la partie sur le développement du code en temps réel nous a aussi pris du temps. Cela requiert l’utilisation d’un thread qui est difficile à manipuler. Nous n’avons pas réussi à réaliser tout ce que nous voulions, mais l’objectif principal est rempli.

# II. Manuel d’utilisation

## II.1. Exécution du programme avec paramètres

Le programme possède 2 fonctions principales, une première fonction permettant d’acquérir le signal sur une durée précise et une deuxième qui récupère les données en temps réel.

Pour utiliser ces programmes sous un Raspberry, il suffit d’ouvrir le terminal et d’exécuter les commandes suivantes,

1er fonction :

**Acquisition(Voie,NbPoints,Rate)**

2ème fonction :

**TR\_Acquisition(Voie ,Rate)**

**(insert screen de l’appel du code avec paramètres)**

Cet appel ouvrira une fenêtre qui contiendra un graphique du signal en temps réel.

* « Voie » Valeur attendue : String

Canal utilisé. Sur le BitScope, seuls 2 sont fonctionnels avec le programme; les connecteurs BNC A et B. A et B sont donc les seuls choix possibles pour ce paramètre.

* « NbPoints » Valeur attendue : int

Nombre de point à acquérir. Plus la valeur est grande, plus la mesure est complète, mais plus le temps d’exécution sera long. Ce paramètre n’existe pas pour la 2eme fonction puisqu’en temps réel, elle acquière constamment des points.

* « Rate » Valeur attendue : int

Fréquence d’échantillonnage.

## II.2. Fonctions secondaires

Pour paramétrer une mesure, afficher des informations ou encore obtenir de l’aide, nous avons implémenté plusieurs commandes.

Les mots en rouge désignent la commande à taper dans la console pour exécuter la fonction.

### II.2.1. Aide

**TR\_Acquisition.Help** ou **Acquisition.Help**

Avec cette commande, l’utilisateur obtient un menu interactif. Il peut alors demander différentes informations en tapant le chiffre adéquat.

**(insert screen section help)**

Toutes les fonctions à suivre se retrouvent dans ce menu.

### II.2.2. Réglage de la durée d’acquisition

**Acquisition.setDuration(NewDuration)**

*Attention pour la 1ère fonction uniquement.*

* « NewDuration » Valeur attendue : int

Durée d’acquisition en seconde.

### II.2.3. Choix du mode

**TR\_Acquisition.setMode(mode)** ou **Acquisition.setMode(mode)**

* « mode » Valeur attendu : String

Sélection du mode d'acquisition de la sonde. Les choix possibles sont :

FAST : Acquisition rapide d'une seule voie.

DUAL : Acquisition des 2 voies simultanément.

MIXED : Acquisition de la voie analogique et logique simultanément.

LOGIC : Acquisition sur les voies logiques.

STREAM : Mode pour l'utilisation du logiciel de visualisation fourni avec le BitScope.

*Attention les modes, LOGIC, STREAM et DUAL ne sont pas encore fonctionnels.*

### II.2.4. Enregistrer

**TR\_Acquisition.Enregistrer** ou **Acquisition.Enregistrer**

Permets de lancer l'acquisition préalablement paramétrée.

### II.2.5. Affichage des valeurs

**Acquisition.DisplayAcq**

*Attention pour la 1ère fonction uniquement.*

Permets d'afficher le tableau des valeurs acquises dans la console. Si l'acquisition n'a pas été lancée avec Enregistrer, le tableau est initialement rempli de 0.

### II.2.6. Affichage des temps

**Acquisition.DisplayTime**

*Attention pour la 1ère fonction uniquement.*

Permets d'afficher le tableau des temps d'acquisition dans la console. Si l'acquisition n'a pas été lancée avec Enregistrer, le tableau est initialement rempli de 0.

### II.2.7. Affichage d’informations

**TR\_Acquisition.Infos** ou **Acquisition.Infos**

Permets d'afficher les informations de configuration du BitScope et de l'acquisition en cours.

# III. Organisation du code

Le code est réparti en 2 fichiers d’extension python (py). Un pour acquérir le signal sur une durée précise, Acquisition.py, et une deuxième qui récupère et affiche les données en temps réel, TR\_Acquisition.py.

Ces 2 fichiers possèdent leurs propres fonctions et fonctionnent donc indépendamment.

# IV. Restes à faire

Bien que complet, le programme est loin d’être parfait. Voici quelques idées que nous avons développé ensemble que nous aurions voulu développer avec du temps.

* **Implémentation des différents modes**

Cf II.2.3. Choix du mode page10

Bitlib propose différents modes d’acquisition, ces modes offriraient plus de souplesse au programme.

Pour l’instant seuls les modes FAST et MIXED sont opérationnels.

* **Interface graphique**

Pour plus de confort pour l’utilisateur, nous pourrions coder une interface graphique avec des boutons et différents graphiques.

* **Implémentation avec les autres canaux**

Ici, seuls les connecteurs BNC fonctionnent avec le programme. Cependant, le BitScope propose aussi des connecteurs logiques.

* **Sauvegarde**

Pour pouvoir vraiment exploiter les capacités du programme, l’utilisateur doit pouvoir récupérer les valeurs ainsi qu’un « screen » du graphique pour d’éventuels traitements. La fonction **save** est un début de code qui permet ce processus, mais elle n’est pas fonctionnelle.

# BILAN

Bien que semés d’embuche, nous avons réussi à mener ce projet à terme et produire un programme convenable et fonctionnel. À l’état actuel, le code est capable de lancer d’afficher un graphique des points mesurés à un temps t ou en temps réel. C’est donc assez rudimentaire, mais il répond à la problématique initiale du projet.

Comme tout nouveau programme, il reste évidemment de nombreuses améliorations. Avec du temps et le matériel de l’école, nous pourrions continuer à travailler sur ce projet et y apporter toutes ces améliorations afin d’affuter nos compétences.

Ce projet nous a permis de bien mieux percevoir les mécaniques de l’instrumentation virtuelle et du langage python. Cela nous a permis d’acquérir une première expérience concrète qui nous servira pour notre carrière. Notamment pour l’un d’entre nous qui compte débuter sa carrière professionnelle dans le domaine de l’instrumentation. Nous avons maintenant une meilleure vision de ce que nous pouvons réaliser avec du matériel innovant comme le BitScope.

En dehors de cela, ce projet nous a surtout appris à développer en groupe. En effet, on a pu voir que ce n’est pas chose facile et qu’il faut de l’organisation et de l’implication. En plus de cela, on a aussi appris à travailler en groupe à distance. Sur une sujet aussi complexe, il nous fallut énormément de communication. Une fois de plus nous sera fort utile pour nos futurs parcours.