**Rapport technique**

**Projet BitScope**

Par

Nicolas Morisseau

Abdellah Ben Rahmoun

Valentin Beau

Etudiants en 5ème année 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

# RESUME

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 code source, les difficultés rencontrés, la documentation technique et le manuel d’utilisation.

# SOMMAIRE

[RESUME 2](#_Toc503657793)

[SOMMAIRE 3](#_Toc503657794)

[INTRODUCTION 4](#_Toc503657795)

[1. Etude du besoin 4](#_Toc503657796)

[2. But et Objectifs 4](#_Toc503657797)

[I. Développement du projet 4](#_Toc503657798)

[I.1. Ressources 4](#_Toc503657799)

[I.2. Travail effectué 5](#_Toc503657800)

[I.3. Difficultés rencontrés 6](#_Toc503657801)

[II. Manuel d’utilisation 6](#_Toc503657802)

[II.1. Appel de la fonction avec paramètres 6](#_Toc503657803)

[II.2. Paramétrage 7](#_Toc503657804)

[II.3. Section help 8](#_Toc503657805)

[III. Organisation du code 8](#_Toc503657806)

[BILAN 9](#_Toc503657807)

[REMERCIEMENTS 10](#_Toc503657808)

# INTRODUCTION

## 1. Etude 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 choisis 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.

## 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 même les scripts. Ces scripts sont codé en langage python car ce langage est interprété, orienté objet, rapide et facile à manipuler.

L’objectif principale est de créer un programme capable de recevoir les signaux en sortis 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.

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 utilise 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 conçu 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 technique 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 limité, celle-ci est très importante puisque c’était notre seul moyen de communiquer avec l’appareil. Après avoir bien saisie 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. A 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 donnés
* Enregistrement des donnés
* Affichage des données

Avec ce code, on obtient une graphique du signal enregistré par le BitScope au moment de l’exécution du code. Soit,

(insert screen signal)

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

On se répartir les tâches pour développer ces 3 nouvelles partis :

* Réception en temps réel
* 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 alros en temps réel via un thread les signaux reçu par le Biscope.

## I.3. Difficultés rencontrés

# II. Manuel d’utilisation

## II.1. Appel de la fonction avec paramètres

Pour lancer le programme sous un Raspberry, il suffit d’ouvrir le terminal et d’exécuter la commande :

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

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

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

* « Voie » Valeur attendu : String

Cela désigne le 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 possible pour ce paramètre.

* « NbPoints » Valeur attendu : int

Ce paramètre représente la précision de la mesure demandé. Plus la valeur est grande, plus la mesure est précise mais plus le temps d’exécution sera long. Cela peut donc provoquer un décalage en temps réel.

* « Rate » Valeur attendu : int

C’est la fréquence d’échantillonnage.

## II.2. Paramétrage

Pour paramétrer une mesure, il existe plusieurs commandes.

* Choix du mode

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

Blablabla……………..

* Enregistrer

**?**

Blablabla……………..

## II.3. Section help

Pour obtenir des informations sur le BitScope et sur le programme, l’utilisateur peut taper la ligne de commande suivante dans le terminale :

**Help**

(insert screen section help)

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

# III. Organisation du code

Mmmh pas grand-chose à dire

# BILAN

# REMERCIEMENTS