Développement d'outils logiciels pour le pilotage d'un banc de mesures ultrasonores

Youssouf MAHAMAT YOUSSOUF

Master Informatique

15 avril 2024 - 31 août 2024

Tuteur(s) université :
Benoit DA MOTA
benoit.damota@univ-angers.fr

Tuteur(s) entreprise :
Thibaud DEVIE
thibaud.devie@univ-eiffel.fr
Pierric MORA
pierric.mora@univ-eiffel.fr
Maximilien LEHUJEUR
maximilien.lehujeur@univ-eiffel.fr

université angers

FACULTÉ DES SCIENCES
UNIVERSITÉ D'ANGERS

Engagement de non-plagiat

Je, soussigné(e),……..

déclare être pleinement conscient(e) que le plagiat de documents ou d'une partie d'un document publiés sur toutes formes de support, y compris l'internet, constitue une violation des droits d'auteur ainsi qu'une fraude caractérisée.

En conséquence, je m'engage à citer toutes les sources que j'ai utilisées pour écrire ce rapport ou mémoire.

Nom-Prénom : ……

Cet engagement de non-plagiat doit être inséré en première page de tous les rapports, dossiers, mémoires.

Table des matières

Résumé

Pour valider mon Master 1 en Informatique, j'ai effectué un stage d'une durée minimale de huit semaines. L'Université Gustave Eiffel m'a offert cette opportunité au sein de son laboratoire de recherche GeoEND, situé sur un campus de recherche au sud de l'agglomération nantaise. En tant que stagiaire ingénieur logiciel, j'ai travaillé sur un banc de mesures ultrasonores destiné à étudier les propriétés mécaniques de surface d'échantillons de béton soumis à diverses dégradations.

Ma mission principale consistait à développer une application pour contrôler ce banc de mesures ultrasonores. Pour ce faire, j'ai dû maîtriser divers instruments et outils de génération et de traitement des signaux via des interfaces graphiques modulaires. L'application comprend plusieurs modules : un gestionnaire de génération d'onde, un pilote d'un dispositif de déplacement uni-axial, un configurateur d'acquisition d'onde, et un visualiseur de traitement de données.

Resume

To validate my Master 1 in Computer Science, I completed an internship of a minimum of eight weeks. Gustave Eiffel University offered me this opportunity in its GeoEND research laboratory, located on a research campus south of the Nantes agglomeration. As a software engineer trainee, I worked on an ultrasonic measurement bench to study the mechanical surface properties of concrete samples subjected to various degradations.

My main mission was to develop an application to control this ultrasonic measurement bench. To do this, I had to master various instruments and tools for generating and processing signals via modular graphical interfaces. The application includes several modules : a wave generation manager, a motor driver for moving on an axis, a wave acquisition configurator, and a data processing viewer.

1 Introduction

1.1 Structure d'accueil et contexte

Ce stage a eu lieu au sein de l'*Université Gustave Eiffel* [1]. Cette université a vu le jour le premier janvier 2020 suite à la fusion de l'Université Paris-Est Marne-la-Vallée et de l'IFSTTAR (Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux). Trois écoles d'ingénieurs et une école d'architecture, partageant toutes les mêmes ambitions, ont été intégrées à cette nouvelle entité. L'université compte plus de trois mille membres du personnel et quinze mille étudiants répartis sur sept campus en France : Paris, Lille, Lyon, Nantes, Marne-la-Vallée, Méditerranée et Versailles. Elle comprend quinze composantes de formation, trente-trois composantes de recherche, ainsi que dix laboratoires internationaux.

Au sein de cette université, le laboratoire GéoEND [1] est une unité de recherche intégrée au département GERS [2] et implanté sur le site de Nantes. L'équipe compte actuellement douze membres permanents dont sept chercheurs, deux ingénieurs, deux techniciens et un assistant administratif. Le génie civil et l'énergie occupent une place prépondérante dans les thématiques de recherche du laboratoire qui s'attache donc à développer des techniques d'auscultation géophysique de subsurface liées aux ouvrages anthropiques et des techniques d'évaluation et contrôle non destructifs appliquées aux infrastructures de génie civil. Les activités de recherche du laboratoire s'appuient sur la modélisation numérique, l'expérimentation physique en laboratoire et l'expérimentation sur le terrain, ce qui implique de développer des outils méthodologiques et instrumentaux. Sachant que dans le domaine de la recherche, les essais en laboratoire sont en constante évolution, nécessitant ainsi des ajustements fréquents et des adaptations rapides. Les outils utilisés doivent donc être flexibles et accessibles à un large éventail d'utilisateurs.

Dans ce contexte, le laboratoire GéoEND effectue des mesures ultrasonores sur le béton afin d'estimer son état de santé. Ce stage se concentre sur un dispositif expérimental d'ondes de surfaces, destiné à sonder les 3 à 5 premiers centimètres sous la surface. La fonction de cette couche (béton d'enrobage) est de protéger les armatures métalliques des agressions de l'environnement. Les mesures sont effectuées à l'aide d'une source piézoélectrique (50 kHz à 150 kHz) appliquée au contact du béton et qui émet des ondes dans le milieu selon un signal de génération. La réception est effectuée à l'aide d'un transducteur (laser ou capteur aérien) capable de mesurer le mouvement de la surface du béton sans être au contact de celui-ci. Le récepteur est déplacé à l'aide d'un robot uni-axial afin d'estimer le temps mis par les ondes ultra-sonores pour se propager dans le béton depuis la source jusqu'au point de mesure, ce qui renseigne sur l'état interne de celui-ci.

1. Géophysique et Évaluation Non Destructive. Voir [2]
2. Géotechnique, Environnement, Risques naturels et Sciences de la Terre

Les déplacements mesurés sont numérisés et enregistrés afin d'être analysés immédiatement ou bien ultérieurement selon les besoins de l'expérience. Un exemple des travaux de recherche réalisés à l'aide des instruments présentés dans le cadre de ce stage est présenté dans Abraham et al. [3]. L'objectif de ce stage est de piloter les trois instruments en jeu : le générateur de formes d'ondes, le robot qui porte le récepteur ultrasonore, et l'oscilloscope dédié à l'acquisition du signal.

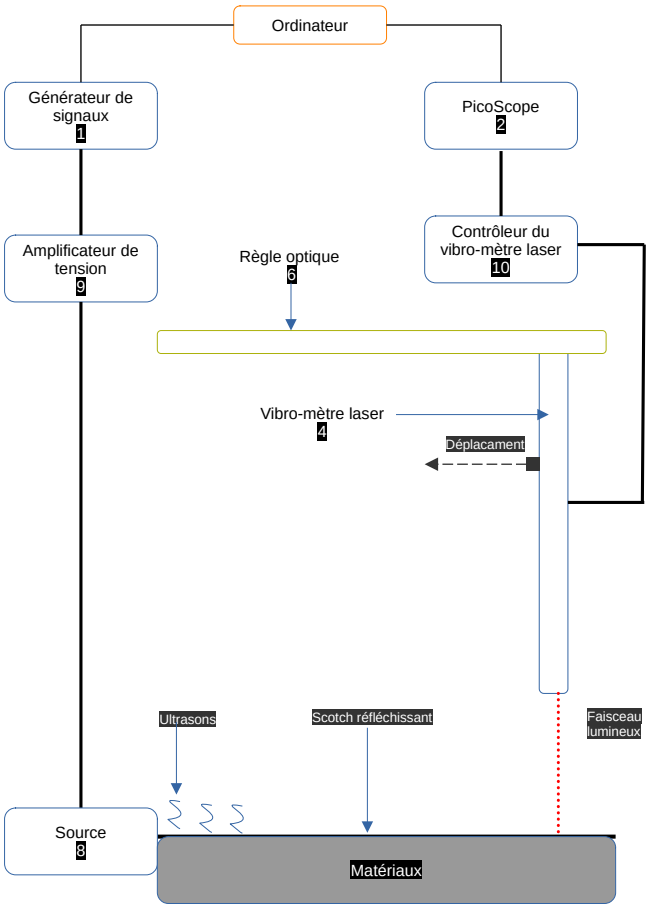


Figure 2 – Schema de la manipulation.

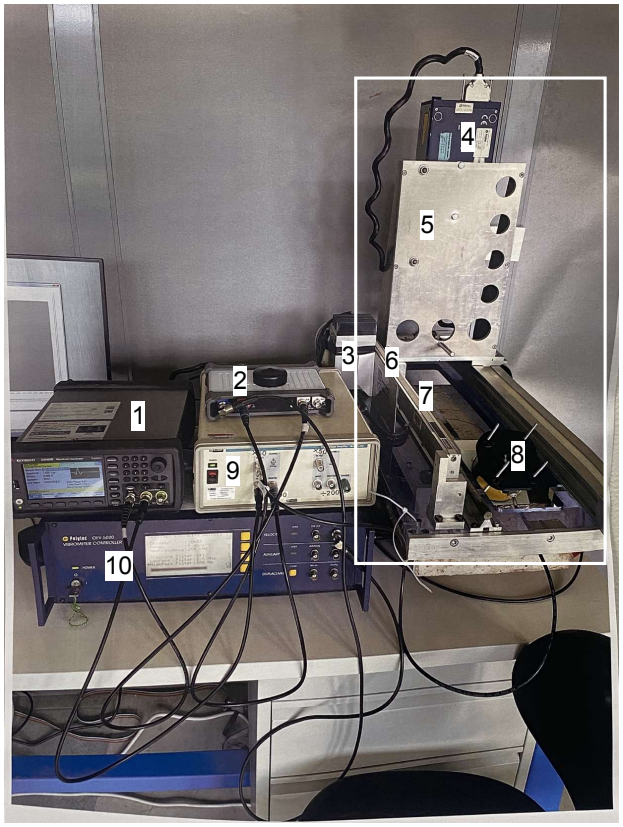


FIGURE 3 – Représentation de l'expérience.

1. Générateur de signaux numériques, connecté à l'amplificateur (9)
2. PicoScope, carte d'acquisition transformant les signaux analogiques envoyés par le récepteur (4) en signaux numériques lisibles par l'ordinateur
3. Dispositif de déplacement uniaxial
4. Vibromètre laser : émet un faisceau lumineux continu, capte le faisceau réfléchit par le scotch (7), génère un signal analogique à partir du calcul de la différence entre les faisceaux
5. Support du laser (4), permet le positionnement du laser à la verticale ou l'horizontal
6. Règle optique, système d'asservissement pour une précision au millimètre.
7. Scotch réfléchissant de faisceaux lumineux
8. Émetteur ultrasonore piézoélectrique, aussi appelé source. Créée des ondes ultrasonores à partir des signaux numériques reçus à partir du voltage appliqué
9. Amplificateur de tension : amplifie le voltage en sortie du générateur (1) pour alimenter la source (8)
10. Contrôleur du vibromètre laser (4) : réglage du calibre, autofocus, etc

FIGURE 4 – Schéma annoté de la manipulation.

1.2 Objectif et motivation du stage

La première étape cruciale de mon stage a consisté à assister à l'expérience sur le banc de mesure ultrasonore. Cette participation m'a permis d'obtenir une visualisation claire de la manipulation à orchestrer, complétée par les explications approfondies fournies par les encadrants. Par la suite, j'ai échangé avec mes encadrants afin de formuler de manière précise leurs besoins pour garantir la création d'un produit qui répond parfaitement aux attentes listées ci-dessous.

Cahier des charges :

- Programmation dans le langage Python, développement sous l'IDE PyCharm
- Développement des interfaces graphiques à l'aide de la librairie Qt, via le package python PySide6
- S'appuyer autant que possible sur les librairies pyvisa, pyvisa-py et pymeasure pour les drivers d'instruments
- Assurer la compatibilité des outils développés pour Linux et Windows, autant que possible
- Utiliser une structure modulaire dans laquelle chaque instrument a sa propre interface autonome ; les briques peuvent ensuite être assemblées dans une application maîtresse
- La demande initiale est d'avoir un projet par instrument déposé sur l'espace Gitlab du laboratoire. Cette structuration pourra éventuellement être modifiée
- Code documenté et extensible
- Prévoir une utilisation mixte des outils : via l'interface graphique et la ligne de commande
- Gestion du projet

Enfin, le développement du produit doit être en parfaite harmonie avec la stratégie du laboratoire, une étape cruciale pour sa conception optimale. En tant que chef de projet et unique développeur actif, il est de ma responsabilité d'assurer la gestion complète du projet. Cela implique la coordination de toutes les activités liées au projet, de la planification à l'exécution, en passant par le suivi et le contrôle. Par le biais de diverses réunions, je suis chargé de fournir un suivi détaillé du projet, garantissant ainsi une communication transparente et efficace avec toutes les parties prenantes.

Ma formation académique et mes expériences précédentes dans ce domaine me confèrent une position avantageuse pour assumer ce poste. Je suis extrêmement enthousiaste à l'idée de poursuivre mon apprentissage et de perfectionner mes compétences dans ce domaine.

2 Matériels

2.1 Ordinateur

L'UGE (Université Gustave Eiffel) m'a fourni un ordinateur personnel, un HP ZBook[4], pour accomplir mes missions. Cet ordinateur est équipé d'un processeur Intel Core i7 et de 15 Go de RAM, ce qui me permet de gérer efficacement plusieurs tâches simultanément et de travailler sur des projets exigeants en termes de ressources.

2.2 Générateur de signaux

Le laboratoire est équipé de plusieurs générateurs d'ondes. Le modèle envisagé dans le cadre de cette expérience est Agilent33500B[5] (maintenant Keysight Technologies). Cet appareil est un générateur de formes d'onde arbitraires et de fonctions à deux voies qui offre une flexibilité et une performance de génération d'ondes avec une fréquence atteignant 30 MHz. L'amplitude, l'offset et la phase sont aussi des paramètres ajustables d'un signal. Il peut émettre des signaux de type : Sinusoïdaux, Carrés, Triangle, DC, Rampe, Impulsions, Bruit et Fonctions Arbitraires. D'habitude, on génère divers types de signaux tels que des impulsions ("Ricker wavelet"), des "burst" et des "chirps". Il est également possible de générer des formes d'onde plus complexes, comme des résultats de mesure lus à l'envers ("retournement temporel"). Sa capacité à générer des signaux précis et de haute qualité a été essentielle pour la réalisation de mes missions.

FIGURE 5 – Générateur d'ondes de marque Keysight, de modele Agilent 33500B

Le fait de contrôler le générateur de signaux Agilent 33500B depuis l'ordinateur offre plusieurs avantages, notamment :

— *Automatisation des tests :* Cela accroît l'efficacité et permet d'exécuter des tests complexes de manière séquentielle ou simultanée.
— *Flexibilité et complexité :* Des tâches difficiles à réaliser manuellement deviennent faciles comme créer et ajuster dynamiquement des formes d'onde complexes, ou encore programmer des séquences de test. Par exemple dans le cas d'une rétroaction où l'on souhaiterait réémettre un signal que l'on a mesuré.
— *Gestion de plusieurs instruments :* Depuis une interface graphique, on peut coordonner le fonctionnement de multiples instruments de mesure, mais aussi configurer des tests plus sophistiqués.
— *Facilité l'utilisation :* Une interface graphique conviviale peut simplifier l'utilisation de l'instrument, surtout pour des utilisateurs moins expérimentés.

2.3 Dispositif de déplacement uni-axial

Le moteur de déplacement sur un axe est un équipement unique, développé spécialement par le laboratoire. Son système d'asservissement sur une règle optique permet un positionnement très précis. Le contrôle du moteur par ordinateur permet d'automatiser les déplacements du robot, par exemple pour un déplacement à pas constant sur de longues durées. Un point de vigilance est de s'assurer que le robot n'est pas envoyé à une position qui dépasse ses limites de déplacements, comprises entre 0 et 400 mm. J'ai reçu l'aide d'un de mes encadrants pour connecter le robot à l'ordinateur afin de procéder aux tests.

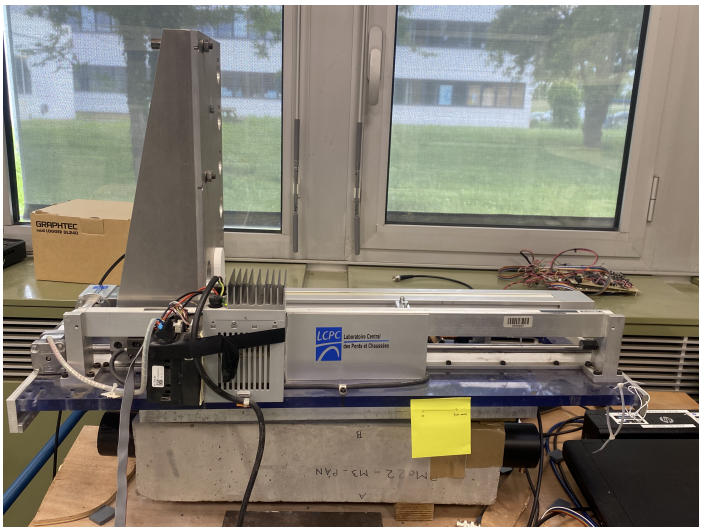


FIGURE 6 – Dispositif de déplacement uni-axial.

2.4 Oscilloscope

Le dernier instrument que l'on souhaite piloter par ordinateur dans le cadre de cette expérience est le PicoScope 5243D [6] de la famille des *PicoScope 5000D series*. Ce PC Oscilloscope peut acquérir des signaux sur deux voies. Sa fonction est de numériser des signaux analogiques afin de pouvoir les traiter par ordinateur. Les paramètres ajustables et que l'on souhaite pouvoir modifier depuis l'ordinateur sont :

- le nombre de points acquis à chaque mesure,
- la fréquence d'échantillonnage en Hertz, qui contrôle le nombre de points mesurés par seconde,
- la résolution, qui contrôle le nombre de bits avec lesquels les valeurs sont encodées. les valeurs possibles sont 8, 12, 14, 15 et 16 bits.
- Le calibre, qui contrôle la gamme d'amplitude des valeurs mesurables. Une gamme trop large entraîne une mauvaise résolution des amplitudes mesurées, et un gamme trop étroite ne permet pas de mesurer les amplitudes les plus fortes et risque de fournir des signaux saturés.
- les paramètres permettant de contrôler le déclenchement (trigger) de la mesure.
- le pre-trigger qui définit la durée du signal qui doit être préservée avant le déclenchement.

La mémoire de capture de forme d'ondes est de 256 MS (million samples). Il comprend une option de génération d'onde arbitraire.

Figure 7 – PicoScope 5000A

3 Outils logiciels

3.1 Système d'exploitation

L'ordinateur fourni par l'université fonctionne sous **Ubuntu 22**, un système d'exploitation robuste et fiable offrant une grande flexibilité pour le développement de logiciels. Tous les modules sont développés et testés sur cet appareil, ainsi que sur un autre poste de travail sous **Windows**, afin de garantir leur compatibilité et performance sur les deux plateformes.

3.2 Langage de programmation : Python

Python [7] est un langage de programmation polyvalent et populaire, connu pour sa simplicité et sa lisibilité. Créée par Guido van Rossum, sa popularité est due à sa syntaxe claire et concise, qui facilite l'apprentissage et l'utilisation par les développeurs de tous niveaux. L'une des forces majeures de Python réside dans sa communauté active et engagée. De plus, Python fourni des librairies efficaces aussi bien pour la physique que pour l'instrumentation. La version 3.10 de Python, récemment publiée (4 oct. 2021) et stable, a été choisie pour ce projet.

3.3 Environnement de développement : PyCharm

PyCharm est un environnement de développement intégré (IDE) puissant et polyvalent, spécialement conçu pour Python. Développé par JetBrains, PyCharm facilite le codage, le test et le débogage avec des outils complets. Son intégration transparente avec des outils de gestion de versionnement comme Git m'a été particulièrement utile. Avec une interface intuitive et personnalisable, PyCharm est un IDE incontournable pour maximiser la productivité et la qualité du code en Python.

3.4 Outils de versionnage : GitLab

GitLab est une plateforme gérée par GIT pour la gestion de code source et le développement logiciel collaboratif. L'UGE possède une plateforme Gitlab hebergée sur un serveur interne permettant aux agents de créer des projets collaboratifs et de bénéficier de services adaptés à leurs besoins. Durant mon stage, j'ai rencontré des difficultés pour prendre en main cet outil en raison des erreurs liées aux droits de lecture et d'écriture difficiles à cerner et à résoudre. De plus, les problèmes de branches parallèles entraînent souvent des conflits lors des mises à jour. Enfin, j'ai été confronté à des bugs liés à l'IDE Pycharm qui compliquent la gestion de plusieurs projets sur le même compte GitLab.

3.5 Api : Qt

Qt(cute) est un framework [3] multiplateforme pour le développement d'applications et d'interfaces utilisateur. Il est largement utilisé en raison de sa robustesse, de sa flexibilité et de sa capacité à fonctionner sur diverses plateformes, y compris Windows, macOS, Linux, Android et iOS. Les projets s'appuient sur la version 6.7 de Qt, actuellement la dernière version disponible.

3.6 Bibliothèque graphique : PySide6

PySide6 [9] est un ensemble de liaisons Python pour la bibliothèque Qt6. Partie intégrante du projet Qt for Python, successeur de PyQt5, PySide6 permet aux développeurs de créer des interfaces utilisateur multiplateformes en utilisant le langage Python, offre la possibilité d'utiliser les widgets et les outils de conception graphique fournis par Qt et bénéficie de la vaste communauté de Qt.

4 Développement

4.1 Communications avec les équipements

4.1.1 Libraire PyMeasure & PyVisa-py

Dans la mesure du possible, on souhaite communiquer avec les appareils cités précédemment, en s'appuyant sur la librairie python **PyMeasure** [10]. PyMeasure est conçue pour simplifier le contrôle des instruments de mesure et d'acquisition de données dans les laboratoires et les environnements industriels. C'est un outil précieux si bien que les chercheurs et ingénieurs sont en mesure de se concentrer sur l'analyse et l'interprétation des données plutôt que sur la gestion des communications avec les instruments. PyMeasure est basé sur **PyVISA**, agissant comme une surcouche qui étend ses fonctionnalités. PyVISA est une bibliothèque Python essentielle pour l'interfaçage et le contrôle des instruments de mesure via le standard VISA (Virtual Instrument Software Architecture). Elle permet de communiquer avec divers équipements de test et de mesure, en utilisant des protocoles couramment utilisés comme GPIB, USB, Ethernet et série. PyVISA peut s'appuyer sur des pilotes propriétaires tels que NI-VISA (National Instruments VISA) ou Keysight VISA. En revanche, PyVISA-py est une implémentation de PyVISA utilisant uniquement Python et sans dépendance vis-à-vis de pilotes VISA externes. Dans ce projet, nous nous sommes limités à PyVISA-py pour établir une communication fiable et cohérente avec tous les instruments. Un exemple d'utilisation de la librairie PyVISA-py est représenté sur la figure 8. La syntaxe PyVISA est très proche de PyVISA-py mais les résultats sont parfois légèrement différents (voir la figure 23).

3. Infrastructure de développement et/ou outils de structuration d'application. Voir [8]

Dans ce qui suit, nous détaillerons les démarches accomplies lors de la prise en main du module de contrôle du générateur de signaux afin de mieux comprendre comment communiquer efficacement avec les instruments nécessaires à l'expérience.

Tout d'abord, nous devons vérifier que l'appareil est bien connecté à l'ordinateur, c'est possible grâce à la méthode liste_resources() de PyVISA-py (Figure 8).

```
1 from pyvisa import ResourceManager
2 pyvisa.ResourceManager('@py').list_resources()
3 ('USB0::2391::11271::MY59003222::0::INSTR',)
```

FIGURE 8 – Établissement de la liste des instruments connectés (ressources) à l'aide de PyVisa-py.

Nous avons bien une adresse de ressource disponible sur le port USB0, examinons de quel appareil, s'agit-il. Pour cela, il existe une commande '*IDN ?' demandant l'identité d'un instrument (figure 9). Attention cette commande n'est pas universelle à tous les équipements, mais limité.

```
1 from pyvisa import ResourceManager
2 inst =
     rm.open_resource('USB0::2391::11271::MY59003222::0::INSTR')
3 inst.query('*IDN?')
4 'Agilent Technologies,33522B,MY59003222,5.02-3.15-2.00-58-00\n'
```

FIGURE 9 – Instructions permettant de collecter l'identité d'un instrument.

Sur la figue 9, '*IDN ?' renvoie une chaîne de caractères séparée par des virgules, on peut lire que l'appareil connecté est Agilent de modèle 33522B, ce qui correspond bien à notre générateur de signaux.

La librairie PyMeasure intègre un certain nombre d'instruments préconfigurés dont l'Agilent33500 correspondant au générateur de signaux utilisé dans ce projet. On peut dès à présent se connecter à l'aide de PyMeasure.instruments et tester quelques commandes (Figure 10).

```
from pymeasure.instruments.agilent.agilent33500 import
   Agilent33500
generator =
   Agilent33500('USB0::2391::11271::MY59003222::0::INSTR')
generator.shape
'SIN'
#### Value shape in ['SIN', 'SQU', 'TRI', 'RAMP', 'PULS',
   'PRBS', 'NOIS', 'ARB', 'DC']####
generator.shape = 'SQU'
generator.shape = 'SIN'
generator.frequency
1000.0
generator.ask('*IDN?')
'Agilent Technologies,33522B,MY59003222,5.02-3.15-2.00-58-00\n'
```

FIGURE 10 – Contrôle du générateur de signaux.

Remarques : generator.shape fait appel à l'accesseur de la valeur shape de l'appareil, représentant la forme d'onde courante à générer. generator.shape = 'SIN' appelle le mutateur de la valeur, ici, on sélectionne la forme d'onde sinusoïdale.

Consultez la figure 24 pour un aperçu de la manière dont PyMeasure procède.

4.1.2 Spécificité du dispositif de déplacement uni-axial

Le dispositif de déplacement uni-axial a un câble USB indépendant, cette connexion est reconnue directement et utilisable sans plus de démarche sur Windows. Sur Ubuntu, nous avons fait face à des problèmes de droits d'accès de l'utilisateur aux ports séries que nous avons résoutes avec les commandes suivantes :

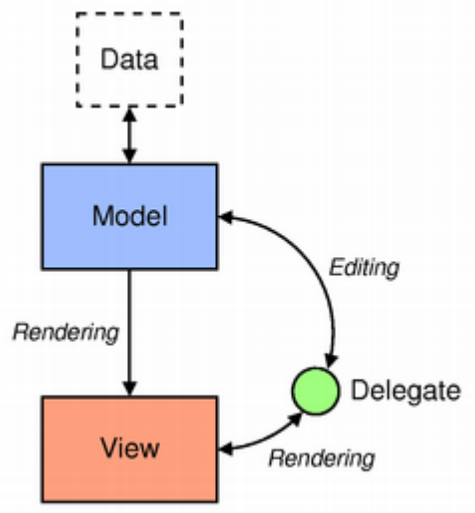
```
sudo usermod -aG dialout $USER
sudo chmod a+rw /dev/ttyUSB0
```

FIGURE 11 – Commandes léguant l'accès root sur /dev/ttyUSB0

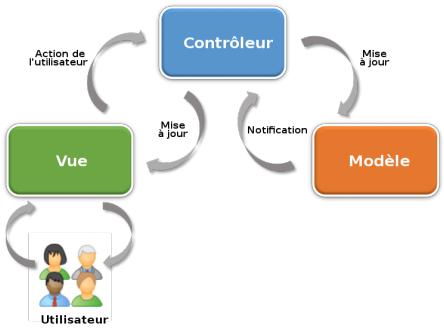
4.2 Interface Graphique

4.2.1 L'architecture

L'architecture **Model-View (MV)** [4] en Python se concentre sur la séparation des responsabilités entre les données (Model) et l'interface utilisateur (View). Contrairement à l'architecture traditionnelle Model-View-Controller (MVC), où le Controller joue un rôle intermédiaire en gérant les interactions entre le Model et la View, l'architecture MV simplifie cette interaction en éliminant le besoin d'un Controller distinct. Cela permet une communication plus directe entre les composants, réduisant la complexité du code et facilitant la maintenance et les mises à jour. Les principaux avantages de l'architecture MV incluent une meilleure modularité, une plus grande facilité de test et une réduction des couplages entre les composants, ce qui rend le développement plus agile et adaptable. C'est cette architecture qui a été retenue pour les développements effectués durant le stage.



(a) Schéma de l'architecture MV.



(b) Schéma de l'architecture MVC.

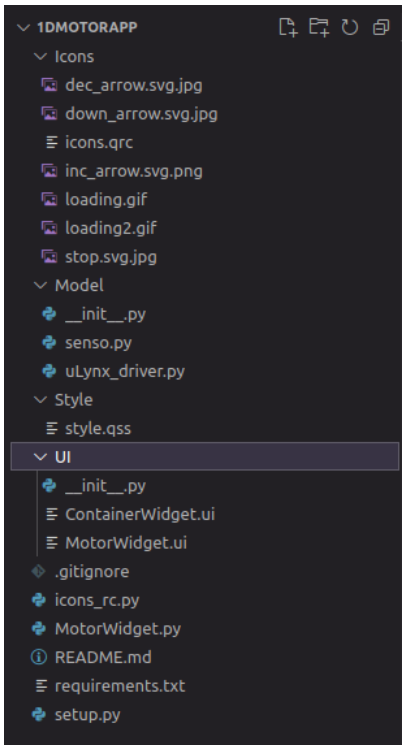
FIGURE 12 – Différence entre MV (tiré de [11]) et MVC (tiré de [12]).

Afin de présenter la procédure employée pour développer un module indépendant, mais aussi articulable avec les attendus définis dans le cahier des charges, nous allons étudier le module du dispositif de déplacement uni-axial. Ce choix s'explique par le fait qu'il s'agit non seulement du module le plus récent a validé les tests, mais également du plus simple, donc facile à comprendre, offrant de nombreuses fonctionnalités. On fera un parallèle avec les autres modules en même temps pour signifier leurs points communs et différences.

4. Voir *Architecture Model View de PySide6* [11]

4.3 Vue d'ensemble

Le package python développé pour le contrôle du moteur (dénommé 1DMotorApp) inclut plusieurs modules python : Les modules Model et View sont dédiés à l'architecture Model-View décrite précédemment. En plus de ces derniers, le projet inclut également les modules UI, Icons et Style.



- Icons est le module des icônes et image de l'application
- Model rassemble tous les fichiers nécessaires à la communication avec notre instrument
- Style sert à rendre l'application plus esthétique
- UI regroupe tous les fichiers d'extension '.UI' créées avec pyside6-designer
- View contient des classes Python, des sous-fenêtres de l'application final
- MotorWidget.py est la racine du projet, peut être déplacée dans View mais sa position actuelle permet de simplifier l'indexation de l'application en appelant la méthode __main__
- REAME.md est une brève introduction au projet, une suite d'instructions à suivre afin de lancer le programme
- requierments.txt est le fichier qui liste l'ensemble des paquets requis dans l'environnement de travail.
- setup.py est le fichier d'installation du paquet dans l'environnement courant. Il est destiné au gestionnaire de paquet "pip". Il définit la version de l'interpréteur python requise, les plateformes sur lesquelles le code peut s'installer ou encore une liste de dépendances.

FIGURE 13 – Organisation du paquet python 1DMotorApp dédié au contrôle du moteur de déplacement uni-axial.

4.4 Design Pattern composite : Découpage en blocs, sous-blocs etc

Les **Design Patterns** [13] (patrons de conception) sont des solutions éprouvées pour résoudre des problèmes récurrents dans le développement logiciel, rendant le code plus maintenable, extensible et réutilisable. Le pattern **Composite**, utile pour le découpage en blocs et sous-blocs, permet de structurer des objets en arborescence

et de gérer des hiérarchies complexes. Dans ce projet, nous l'utilisons pour gérer dynamiquement l'affichage de blocs.

L'interface du module est divisée en deux blocs : un pour la connexion, ensuite pour la configuration Home et enfin un pour le contenu, géré de manière dynamique. Cette séparation facilite l'organisation et la gestion des fonctionnalités, avec un bloc de connexion pour établir et gérer les connexions, et un bloc de contenu pour charger, modifier et afficher les informations de manière flexible.

4.5 Connexions

Tout d'abord, lors de l'ouverture de l'application, un 'QTimer' est lancé. La méthode '_loadDevices' de la classe 'MotorWidget' est appelée toutes les secondes par ce timer, pour vérifier si le processus de chargement des appareils connectés est terminé. Si le thread est terminé, la méthode créée et lance un nouveau 'QThread' pour une nouvelle vérification, sinon, elle attend et n'effectue aucune action. Le processus vérifie la validité des connexions, filtre les équipements avortés, récupère les nouvelles connexions possibles et les ajoute à la liste des appareils détectés. Lorsqu'une connexion est perdue avec l'appareil, autrement dit, au moment où l'adresse de notre instrument n'apparaît pas dans la liste d'appareils détectée par PyMeasure, on émet un Signal de perte de connexion, dans notre cas connectionLostSig. Ce signal sera intercepté par un Slot, ici connectionLost une fonction de la classe, dans laquelle on met à jour la vue en chargeant la vue par défaut, c'est-à-dire avant la connexion. C'est le principe des signaux et slots qui est un mécanisme de communication proposée par Qt. On l'utilisera à plusieurs reprises dans les projets, comme pour ajouter un nouvel appareil détecté à la vue, etc. Sinon ces instructions sont dans le corps de la méthode 'loadDevices' de la classe. Avant de cliquer sur le bouton de connexion de type QPushButton l'utilisateur doit spécifier le driver à utiliser. Si un driver non conforme est sélectionné, l'utilisateur ne pourra pas contrôler l'appareil. Les drivers et les appareils détectés sont proposés sous forme de listes déroulantes(QComboBox en Qt). Les options incluent senso et uLynx, avec senso sélectionné par défaut. uLynx est défini comme un template générique pour divers moteurs, tandis que senso offre des précisions spécifiques et convient parfaitement au dispositif de déplacement uniaxial.

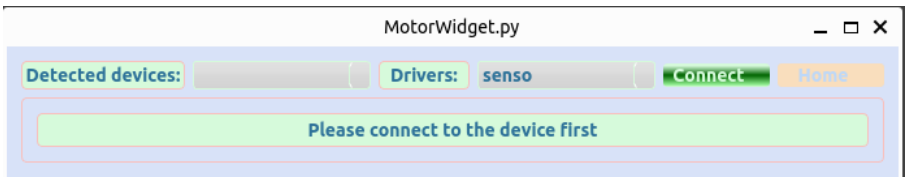


FIGURE 14 – Interface graphique du dispositif de déplacement uni-axial avant connection.

L'interface ci-dessus est identique à l'interface de connexion du module de configuration d'acquisition de signaux. Et plus complète que celle du module de génération de signaux. Car elle permet de choisir un driver, ce qui n'est pas le cas dans l'autre. Les interfaces du picoscope et du module de génération sont présentées dans les annexes.

4.6 Conteneurs

Une fois que la connexion est établie, on désactive le choix des drivers et des appareils détectés et au contraire, on active le bouton de configuration Home. Cette opération permet au dispositif de fixer son origine, sa position de départ (voir la méthode find_home du driver choisi). Ensuite, on donne à l'utilisateur le contrôle de l'appareil depuis l'interface.

Le contenu affiché est le suivant :

— Saisie : une position absolue est récupérée depuis le QLabel lors d'un clic sur le QPushButton ou à la fin de l'édition du label
— Limites : labels de positions max et min modifiables sur activation du QCheckBox
— État de mouvement : un bouton clignotant si le dispositif est en mouvement et rouge fixe sinon
— Pas de mouvement : un QDoubleSpinBox définie un pas d'incrémentation (décrémentation) de la position et est liée aux deux boutons sur sa gauche et son droit
— Barre de progression : une QProgressBar indique la position actuelle relativement au min et max
— Bouton stop : Bouton d'arrêt immédiat/urgence du mouvement en cours, ne fait rien à l'arrêt

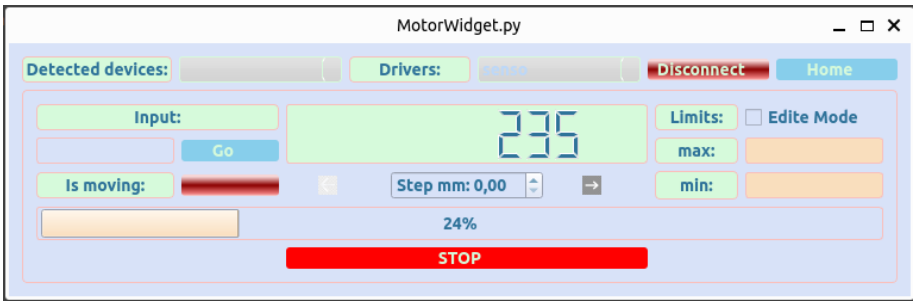


FIGURE 15 – Interface graphique du dispositif de déplacement uni-axial après connection.

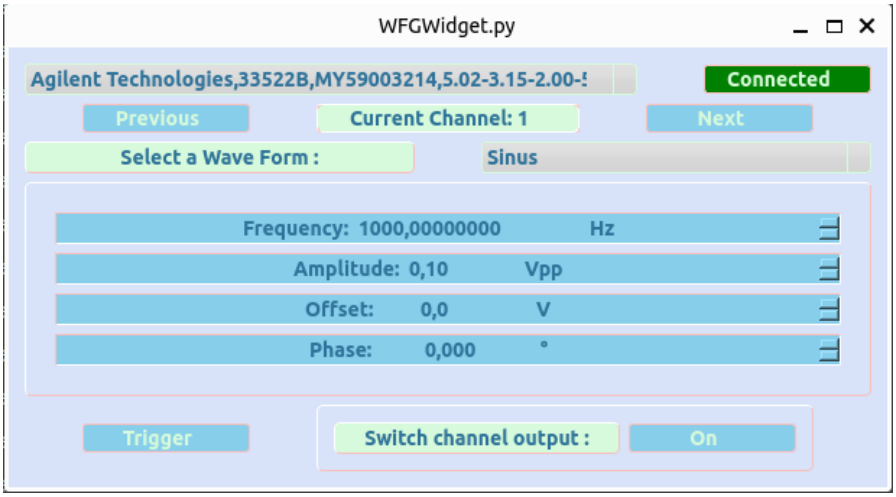


FIGURE 16 – Interface graphique du dispositif de déplacement uni-axial après connection.

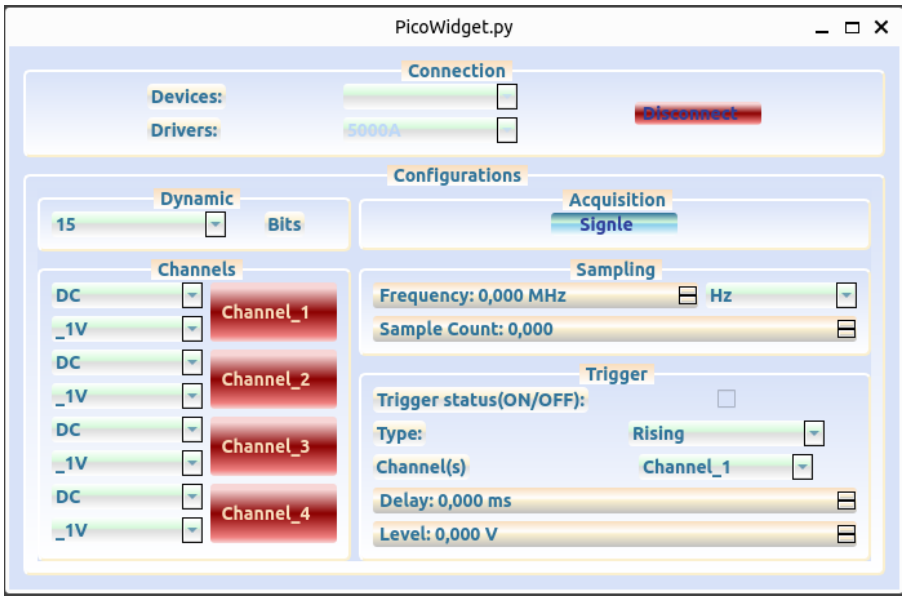


FIGURE 17 – Interface graphique du dispositif de déplacement uni-axial après connection.

Les trois interfaces présentent des contenus variés. Jetons donc un coup d'œil aux deux autres conteneurs.

4.7 Dépendances et moyen d'utilisation des paquets.

La procédure suivante doit être appliquée afin de pouvoir utiliser les modules 1DMotorApp (contrôle du moteur), WFGApp (génération de signaux) et PicoApp (contrôle de l'oscilloscope). Ouvrir un terminal dans le répertoire du projet. Ensuite, pour installer les dépendances du paquet, entrer la commande :

```
pip install -r requirements.txt
```

FIGURE 18 – Installation des dépendances du paquet dans l'environnement courant.

La librairie PySide6 nécessite quelques paquets supplémentaires pour fonctionner normalement : Assurez-vous d'avoir la bibliothèque C++ `libstdcxx-ng` dans votre environnement. Cette bibliothèque est essentielle pour créer les outils de contrôle d'une fenêtre. Notamment les boutons d'extension, de réduction et de fermeture de la fenêtre. Pour l'installer, vous pouvez utiliser des gestionnaires de paquets appropriés en fonction de votre système d'exploitation.

```
sudo apt-get update
sudo apt-get install libstdc++6
```

FIGURE 19 – Pour les systèmes basés sur Linux

```
xcode-select --install
brew install gcc
```

FIGURE 20 – Pour les systèmes basés sur Mac Os

```
conda install libstdcxx-ng
```

FIGURE 21 – Possibilité d'installation avec **Anaconda** sur toutes les plateformes

```
python nomdumudeule.py
```

FIGURE 22 – commande pour l'exécution / lancement des différents modules.

5 Conclusion

5.1 Bilan du stage

Au cours de ce stage, j'ai été amené à développer des outils logiciels pour le pilotage de plusieurs instruments de laboratoire en vue d'une expérience visant à la fois à générer et à enregistrer des signaux ultra-sonores. La complexité de ce travail vient du fait que les appareils doivent fonctionner ensemble bien qu'ils aient chacun des caractéristiques très différentes. Actuellement, j'ai réussi à poser les fondements du projet en créant les modules suivants : gestion du générateur d'ondes (WFGApp), pilotage du moteur de déplacement uni-axial (1DMotorApp) et pilotage du système d'acquisition des signaux (PicoApp). Ces différents composants ont été développés séparément afin de pouvoir être testés et maintenus individuellement conformément au cahier des charges qui m'a été donné. Chacun de ces modules peut être lancé séparément. J'ai testé le premier paquet (WFGApp) au fur et à mesure de son développement, tandis que les deux autres (1DMotorApp et PicoApp) ont été testés après avoir été partiellement achevés. Je n'interviens pas sur la section concernant la visualisation et le traitement des données, ni sur la partie qui gère la communication bas niveau avec le moteur de déplacement sur un axe. C'est en fusionnant ces modules que l'expérience prendra sa forme finale. Afin de faciliter le travail de mise en commun de ces briques élémentaires ainsi que leur développement continu, j'ai proposé une structuration commune basée sur l'architecture Modèle-Vue. J'ai également veillé à uniformiser les procédures d'installation des différentes librairies définies dans les objectifs du stage afin que les trois paquets puissent être déployés dans un environnement commun. Enfin, j'ai proposé des interfaces graphiques basées sur des schémas similaires afin qu'un utilisateur non expert puisse prendre en main l'expérience plus facilement. Cette expérience m'a non seulement permis d'acquérir une connaissance pratique des équipements, mais aussi de comprendre l'importance de la collaboration et de la communication dans la résolution de problèmes techniques complexes.

5.2 Perspectives

Ce rapport porte sur la partie préliminaire de ce stage dédiée au développement des outils de contrôle des instruments (effectuée entre le 15/04/2024 et le 10/06/2024). Ma convention prévoit une prolongation de mon stage jusqu'au 31/08/2024 durant laquelle je vais me concentrer sur la finalisation des tests, notamment pour le contrôle du PicoScope, et leur automatisation via des tests unitaires. Je vais également travailler sur la mise en commun des différents paquets développés, en développant une application maîtresse pour intégrer et coordonner tous les modules.

5.3 Remerciements

Je souhaite exprimer ma profonde gratitude envers mes tuteurs de stage pour leur soutien indéfectible et leur mentorat précieux tout au long de mon expérience. En premier lieu, je tiens à remercier Thibaud DEVIE, qui m'a guidé et accompagné de manière constante et efficace tout au long de ce parcours. De plus, je suis extrêmement reconnaissant envers Pierric MORA et Maximilien LEHUJEUR pour leur engagement sans faille et leur attention minutieuse dans la rédaction et la correction de mon rapport de stage. Enfin, je ne saurai passer sous silence l'importance de la confiance que m'a accordée monsieur Benoit DA MOTA, me permettant ainsi de représenter dignement notre université.

6 Annexes

```
from pyvisa import ResourceManager
pyvisa.ResourceManager().list_resources()
('ASRL1::INSTR', 'ASRL2::INSTR', 'ASRL3::INSTR',
   'ASRL4::INSTR', 'ASRL5::INSTR', 'ASRL6::INSTR',
   'ASRL7::INSTR', 'ASRL8::INSTR', 'ASRL9::INSTR',
   'ASRL10::INSTR', 'ASRL11::INSTR', 'ASRL12::INSTR',
   'ASRL13::INSTR', 'ASRL14::INSTR', 'ASRL15::INSTR',
   'ASRL16::INSTR', 'ASRL17::INSTR', 'ASRL18::INSTR',
   'ASRL19::INSTR', 'ASRL20::INSTR', 'ASRL21::INSTR',
   'ASRL22::INSTR', 'ASRL23::INSTR', 'ASRL24::INSTR',
   'ASRL25::INSTR', 'ASRL26::INSTR', 'ASRL27::INSTR',
   'ASRL28::INSTR', 'ASRL29::INSTR', 'ASRL30::INSTR',
   'ASRL31::INSTR', 'ASRL32::INSTR')
```

FIGURE 23 – Liste de ressources disponibles avec PyVisa.

```
shape = Instrument.control(
    get_command: "FUNC?",
    set_command: "FUNC %s",
    docs: """ A string property that controls the output waveform. Can be set to:
    SIN<USOID>, SQU<ARE>, TRI<ANGLE>, RAMP, PULS<E>, PRBS,  NOIS<E>, ARB, DC. """,
    validator=strict_discrete_set,
    values=["SIN", "SQU", "TRI", "RAMP", "PULS", "PRBS", "NOIS", "ARB", "DC"],
)
```

FIGURE 24 – Accesseur et mutateur de la valeur shape du générateur de signaux

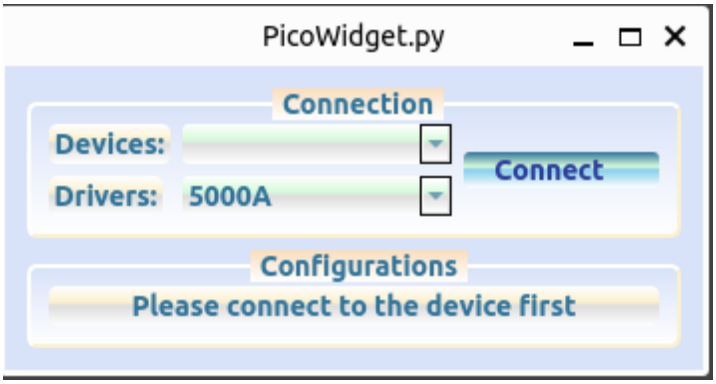


FIGURE 25 – Interface graphique du configurateur d'acquisition de signaux avant connection.

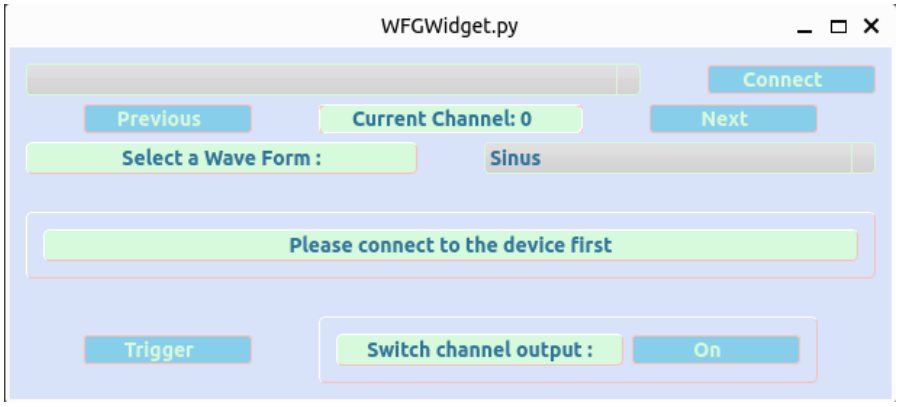


FIGURE 26 – Interface graphique du générateur de signaux avant connection.

Appel d'offre

Université Gustave Eiffel
LABORATOIRE GEOEND
GÉOPHYSIQUE ET
ÉVALUATION NON
DESTRUCTIVE

Université Gustave Eiffel,
Campus de Nantes,
Allée des Ponts et Chaussées,
44340 Bouguenais

Offre de stage – M1 Informatique – 2024

Développement d'outils logiciels pour le pilotage d'un banc de mesures ultrasonores

Mots clés : Python, Qt, Git, Pyside6, Instrumentation, Ultrasons, Évaluation non-destructive

Contexte : Dans le domaine de la recherche, les essais en laboratoire sont en constante évolution et nécessitent donc d'être fréquemment modifiés et adaptés. Les outils utilisés se doivent donc d'être flexibles et maîtrisés par le plus grand nombre. C'est dans cette optique que le langage Python prend une part croissante dans les développements logiciels au sein du laboratoire GéoEND. Il bénéficie d'une grande popularité et communauté très active qui fournie des librairies efficaces aussi bien pour la physique que pour l'instrumentation. Ce stage se concentre sur un banc de mesures ultrasonores conçu pour sonder les propriétés mécaniques de surface d'échantillons de béton qui ont été exposés à différentes dégradations. Il est constitué d'un générateur de signaux, qui alimente un émetteur ultrasonore, d'un récepteur ultrasonore monté sur un axe de déplacement, et d'une carte d'acquisition.

Objectifs : Le travail consistera à développer en python un logiciel complet de contrôle d'un banc de mesures ultrasonores. Il faudra pour cela piloter les différents instruments, maîtriser les outils de génération et traitement des signaux et concevoir une interface graphique modulaire. Les développement devront être réalisés en accord avec la stratégie du laboratoire (librairies PyMeasure et PySide6, dépôt GitLab ...).

Profil & compétences : Master en Informatique. Compétences en programmation : Python, Qt, Git.

Lieu du stage : Le stage se déroulera au laboratoire GeoEND, situé sur un campus de recherche au sud de l'agglomération nantaise, et constitué d'une équipe multi-disciplinaire à l'intersection des géosciences, de l'évaluation non-destructive et des matériaux du génie civil.
https://geoend.univ-gustave-eiffel.fr/

Rémunération : Selon réglementation en vigueur (4.05 €/h en 2023, soit environ 600€/mois).

Contact : Thibaud DEVIE (thibaud.devie@univ-eiffel.fr), Pierric MORA (pierric.mora@univ-eiffel.fr), Maximilien LEHUJEUR (maximilien.lehujeur@univ-eiffel.fr).

7 Sommaire des schémas

Table des figures

8 Sources et bibliographie

Références

[1] *Université Gustave Eiffel*. URL : https://www.univ-gustave-eiffel.fr/.

[2] *GéoEnd*. URL : https://geoend.univ-gustave-eiffel.fr/presentation/le-laboratoire.

[3] O. ABRAHAM et al. « Non-contact, automated surface wave measurements for the mechanical characterisation of concrete ». In : *Construction and Building Materials*. Non Destructive Techniques for Assessment of Concrete 37 (déc. 2012), p. 904-915. ISSN : 0950-0618. DOI : 10.1016/j.conbuildmat.2012.03.015.

[4] *Hewlett-Packard HP ZBook 14 G2*. URL : https://support.hp.com/fr-fr/document/c04541393.

[5] *33522B Waveform Generator*. URL : https://www.keysight.com/us/en/support/33522B/waveform-generator-30-mhz-2-channel-arb.html.

[6] *PicoScope® 5000 Series*. URL : https://www.picotech.com/oscilloscope/5000/picoscope-5000-features.

[7] *Python*. URL : https://www.python.org/about/.

[8] *QT*. URL : https://doc.qt.io/qt-6/.

[9] *Bibliothèque grahique : PySide6*. URL : https://pypi.org/project/PySide6/.

[10] *PyMeasure*. URL : https://pymeasure.readthedocs.io/en/latest/#api-docs.

[11] *Architecture Model View de PySide6*. URL : https://doc.qt.io/qtforpython-6/overviews/model-view-programming.html.

[12] *Architecture Modèle-Vue-Contrôleur*. URL : https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur.

[13] *Design Patterns(Patrons de conception)*. URL : https://refactoring.guru/fr/design-patterns.

[14] *CSS*. URL : https://developer.mozilla.org/fr/docs/Web/CSS/.