وزارة التعليم العالي والبحث العلمي
MINISTERE DE L'ENSEIGNEMENT SUPEREUR ET DE LA RECHERCHE SCIENTIFIQUE
جامعة عبد الحميد بن باديس مستغانم
Université Abdelhamid Ibn Badis Mostaganem
كلية العلوم والتكنولوجيا
Faculté des Sciences et de la Technologie
DEPARTEMENT DE GENIE ELECTRIQUE

N° d'ordre : M ……/GE/2019

MEMOIRE

Présenté pour obtenir le diplôme de

MASTER EN ELECTRONIQUE

Option : Électronique des systèmes embarqués

Par

- **AZZOUZI Omar Abderrezak.**
- **CHAREF AISSA Ahmed.**

INTERFACAGE D'UN INSTRUMENT DE MESURE DE PUISSANCE OPTIQUE ET D'UN CONTROLLEUR DE DIODE LASER POUR UNE COMMANDE A DISTANCE

Soutenu le 11/07/2019 devant le jury composé de :

Président :	Mr. AZZEDINE.M	MAA	Université de Mostaganem
Examinateur :	Mr. HADRI.B	Prof	Université de Mostaganem
Examinateur :	Mr. ABDELLAOUI.N	MAA	Université de Mostaganem
Rapporteur :	Mr. BENACHENHOU.A	Prof	Université de Mostaganem

Année Universitaire 2018/2019

Je dédie ce travail de mémoire
à mes chers parents Ahmed et Touatia,
à mes soeurs Sarah, Imene et Aïcha,
à ma chère fiancée Siham,
à mes frères Soufiane et Ali,
à mon grand-père et ma grand-mère,
à toute ma famille,
à mon encadreur Mr A.BENACHENHOU,
à tous les membres du laboratoire LEOG de Mostaganem,
à tous mes collègues de la promotion Électronique des systèmes embarqués 2018/2019,
à tous mes amis,
et à toutes les personnes qui m'ont aidé de près ou de loin...

Omar Abderrezak AZZOUZI

"Ô mon Seigneur, accroît mes connaissances !" Coran (20/114)

En tout premier lieu, je remercie DIEU le tout puissant de nous avoir éclairé le chemin du savoir afin de terminer ce travail,
Et je dédie ce travail
à mes chers parents,
à mes frères,
à mes sœurs,
à toute ma famille,
à mon encadreur Mr A.BENACHENHOU,
à tous les membres du laboratoire LEOG de Mostaganem,
à tous mes collègues de la promotion Électronique des systèmes embarqués 2018/2019,
à tous mes amis,
et à toutes les personnes qui m'ont aidé de près ou de loin...

Ahmed CHAREF AISSA

"Ô mon Seigneur, accroît mes connaissances !" Coran *(20/114)*

Abstract

Over the last decades, e-learning (e-labs) platforms have appeared on a global scale to replace and solve problems in relation with traditional systems, especially with face-to-face laboratories. The main objective of this thesis is to develop new system for remote interfacing of laboratory instruments, with an emphasis on the optical power and energy meter Thorlabs *PM100D*, and the laser diode controller Thorlabs *PRO800*.

The system must be characterized by its availability, accessibility, maintainability, programming flexibility and ease of use, speed, accuracy, multi-user support…etc. It is also requested to minimize the development price of the system by using free and open source resources, meanwhile We must ensure a perfect resemblance with real laboratories.

For that purpose, a very high-quality architecture is necessary for the realization of the system, using a Raspberry platform implemented with a *Flask* web server and a backend software totally programmed with Python in combination with a frontend software developed in *HTML*, *CSS* and *JavaScript*.

Finally, considering the performance of the new technique that often controls expensive instruments with a low-cost system development. The educational system and the government will achieve new features in terms of economics, education quality, efficiency, and student's performance.

Keywords: E-Learning, E-Lab, Embedded Systems, Raspberry, Web server, Graphical interface, Instruments, Thorlabs, Optics.

Résumé

Au cours des dernières décennies, des plates-formes d'e-learning (e-labs) sont apparues à l'échelle mondiale pour remplacer et résoudre les problèmes liés aux systèmes traditionnels en particulier les laboratoires. L'objectif principal de ce mémoire est de développer un nouveau système d'interfaçage à distance d'instruments de laboratoire, avec un accent particulier sur le mesureur de puissance et d'énergie optique Thorlabs *PM100D* et le contrôleur de diode laser Thorlabs *PRO800*.

Le système doit se caractériser par la disponibilité, l'accessibilité, la facilité de maintenance, la flexibilité de programmation et d'utilisation, la rapidité, la précision, la prise en charge multi-utilisateurs…etc. Il est également demandé de minimiser le prix de développement du système en utilisant des ressources libres et open source. Nous devons par ailleurs garantir la plus grande ressemblance possible au laboratoire réel.

Pour ce faire, une architecture est nécessaire pour la réalisation du système, en utilisant une plate-forme Raspberry Pi implémentée avec un serveur Web *Flask* et un logiciel back-end totalement programmer avec *Python* en combinaison avec un logiciel front-end développé en *HTML*, *CSS* et *JavaScript*.

Finalement, en prenant en compte les performances de la nouvelle technique qui contrôle souvent des instruments coûteux avec un développement de système à faible coût. Le système éducatif et le gouvernement réaliseront de nouvelles caractéristiques en termes d'économie, de qualité de l'éducation, d'efficacité et de performance des apprenants.

Mot clés : Enseignement à distance, Laboratoire distant, Systèmes embarqués, Raspberry, Serveur Web, Interfaçage graphique, Instruments, Thorlabs, Optique.

مـــلـــخّـــص

على مدار العقود الماضية، برزت منصات التعليم الإلكتروني على مستوى العالم لاستبدال وحل المشكلات المتعلقة بالأنظمة التقليدية، وخاصة المختبرات. الهدف الرئيسي من هذه الأطروحة هو تطوير نظام جديد للتحكم والتوصيل عن بعد للأدوات المخبرية عبر شبكة الأنترنت، مع التركيز بشكل خاص على مقياس الطاقة الضوئية Thorlabs *PM100D* ووحدة التحكم في الصمام الثنائي ليزر المعروفة بـ: *Thorlabs PRO800*.

يجب أن يتميز النظام بتوفره الدائم، سهولة الوصول اليه، سهولة الصيانة، مرونة البرمجة والاستخدام، السرعة، الدقة وإمكانية دعم متعدد المستخدمين ... إلخ. يتطلّب أيضًا تقليل تكلفة تطوير النظام باستخدام موارد مجانية ومفتوحة المصدر. يجب أيضا على النظام أن يتيح أكبر قدر من التشابه مع المختبر الحقيقي.

للقيام بذلك، يجب علينا تبني بنية مادية وبرمجية خاصة، وذلك باستخدام منصة Raspberry Pi التي تم ربطها بالإنترنت باستخدام خادم ويب من نوع *FLASK Web Server* وبرنامج خلفي مبرمج كليا بلغة البايثون Python مركب مع برنامج للواجهة الأمامية مطرو بـ: *JavaScript, CSS, HTML*.

أخيرًا، مع الأخذ في عين الاعتبار أداء التقنية الجديدة التي غالبًا ما تتحكم في أدوات باهظة الثمن من خلال نظام منخفض التكلفة. سوف يحقق نظام التعليم والحكومة أهداف جديدة من حيث الاقتصاد وجودة التعليم وكفاءة وأداء المتعلمين.

الكلمات المفتاحية: التعلم عن بعد، مختبر عن بعد، أنظمة مدمجة، Raspberry، خادم الويب، واجهة رسومية، أدوات مخبرية، Thorlabs، بصريات.

Table des matières

Table des figures

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Glossaire

CD	*Compact Disc.*
DVD- ROM	*Digital Versatile Disk - Read Only Memory.*
AUF	*Agence Universitaire de la Francophonie.*
LEOG	*Laboratoire d'Electromagnétisme et d'Optique Guidé.*
IOT	*Internet of Things.*
IOE	*Internet of Everything.*
SCPI	*Standard Commands for Programmable Instruments.*
TP	*Travaux Pratiques.*
USB	*Universal Serial Bus.*
RS-232	*Recommended Standard 232.*
GPIB	*General Purpose Interface Bus.*
RAM	*Random-Access Memory.*
SDRAM	*Synchronous Dynamic Random-Access Memory.*
IEEE	*Institute of Electrical and Electronics Engineers.*
VA	*Volt-Ampere.*
CG	*Cathode Grounded.*
AG	*Anode Grounded.*
CC	*Constant Current.*
CP	*Constant Power.*
TEC	*Thermal Electric Cooler.*
NTC	*Negative Temperature Coefficient.*
RTD	*Détecteurs de Température à Résistance.*
CNA	*Convertisseur Numérique Analogique.*
HTML	*HyperText Markup Language.*
CSS	*Cascading Style Sheet.*
AJAX	*Asynchronous JavaScript and XML.*
PHP	*PHP : Hypertext Preprocessor.*
WSGI	*Web Server Gateway Interface.*
JSON	*JavaScript Object Notation.*
XML	*Extensible Markup Language.*

Introduction générale

1. Généralité et état de l'art

E-learning ou l'apprentissage en ligne consiste à utiliser les technologies électroniques pour accéder à des programmes d'enseignement en dehors d'une salle de classe traditionnelle. Dans la plupart des cas, il s'agit d'un cours, d'un programme ou d'un diplôme entièrement en ligne.

Il existe de nombreux termes utilisés pour décrire l'apprentissage en ligne : l'apprentissage via Internet, l'enseignement à distance, l'apprentissage électronique informatisé et bien d'autres. Nous définissions l'apprentissage en ligne comme des cours spécifiquement disposés via internet accessible partout sans avoir besoin d'une classe où le professeur enseigne, il peut être interactif dans le sens où il peut également y avoir une communication directe avec les professeurs ou avec d'autres étudiants de la classe.

L'objectif du projet e-lab est de créer une dynamique d'excellence autour des initiatives de formation en ligne utilisant des supports d'internet. Afin d'assurer une bonne élaboration des tests et des devoirs, ainsi une manipulation concrète des travaux pratiques.

Les plates-formes d'enseignement à distance ont été développées spécialement pour faciliter l'apprentissage et suivre l'enseignement et la pédagogie des apprenants en science technique, et d'autres disciplines. L'autre but est de minimiser les moyens financiers d'enseignement et essayer de créer des formations de qualité avec un prix raisonnable.

Le *E-learning* apparaît sur deux types d'utilisation comme suit :

- L'apprentissage entièrement en ligne se déroule sans aucune interaction physique ou face à face. Pour ceux les travaux de cours et le matériel sont distribués électroniquement par courrier électronique *(Email)*, sites *Web*, forums en ligne ou plates-formes d'enseignement et CD ou *DVD-ROM*.
- L'apprentissage combiné utilise une combinaison d'instruction dirigée par Internet et d'interaction physique.
  Ce type d'*E-learning* est adopté principalement par les universités traditionnelles où les étudiants peuvent apprendre dans des classes physiques, et continuer leur enseignement par des leçons ou des travaux pratiques en ligne.

2. Contexte

Le laboratoire *d'Electromagnétisme et d'Optique Guidé (LEOG)* de l'université de Mostaganem, et avec sa collaboration avec l'*Agence Universitaire de la Francophonie (AUF)*, s'intéresse à la création d'une plateforme E-learning consacrer pour réaliser tous les types de travaux pratiques de l'électronique, l'optique, l'informatique…etc.

Notre projet de fin d'étude est dédié à la création d'un laboratoire distant complètement destiné vers une scène de travaux pratiques d'optique guidée. Ce laboratoire sera également disponible sur la plateforme *Moodle* de l'université de Mostaganem pour être exploité sous sa forme didactique.

3. Problématique et objectifs

Les laboratoires réels et surtout ceux qui sont destinés aux travaux pratiques de la science technique et la physique engendrent des obstacles difficiles à surpasser comme le coût des achats d'équipement, la disponibilité des classes, le manque du matériel…etc.

En outre les laboratoires réels distants qui existent actuellement utilisent des logiciels coûteux comme le *LabVIEW* (environ $8305 par ans [1]), de plus ces ressources sont entièrement non libres et parfois difficile à personnaliser ou à modifier pour un usage spécifique.

En revanche, les travaux pratiques dans les domaines que nous avons visés demandent une souplesse d'utilisation, une grande flexibilité de configuration et une liberté de manipulation des équipements mises à la disposition de l'étudiant l'ord de la séance de travaux pratiques.

Basant sur ces observations, nous allons réaliser un système d'interfaçage et de pilotage des instruments à distance, en tenant compte d'utiliser une architecture flexible et *Open Source (Source libre)* toute en gardant le contrôle de ces instruments par l'enseignant.

4. Organisation du mémoire

Le plan du mémoire est organisé comme suit :

Le premier chapitre : Architecture matérielle

Le premier chapitre explique la constitution de l'architecture matérielle en définissant tout le matériel utilisé, et les interfaces de communication.

Le deuxième chapitre : Architecture Logicielle

Ce chapitre présente les différentes parties de l'architecture logicielle pour la création du serveur web.

Le troisième chapitre : Implémentation et Tests de fonctionnement

Ce chapitre explique la méthode d'implémentation logiciel/matériel, et il décrit le fonctionnement final du système réalisé.

Chapitre 1 : Architecture matérielle

I.1 Introduction

De nos jours les systèmes embarqués ont subi une croissance monstrueuse à l'aide de la collaboration entreprise/université, le travail collectif et le monde open source. Cette révolution a entraîné l'apparition d'une multitude de plateformes de développement basique tel que les microcontrôleurs *(Arduino, STM32)* et de plateformes plus développées tel que *Raspberry Pi, Red Pitaya, PcDuino*…etc.

En outre, ce développement dans les systèmes embarqués joue un rôle très important dans l'expansion de l'internet des objets *(IoT),* qui forme essentiellement la base d'une constitution d'un laboratoire distant.

Le terme *IoT* est tellement utilisé dans l'industrie il est parfois appelé *Internet of Everything (IoE),* car presque tout est connecté à l'internet d'une manière ou d'une autre. L'adoption capitale du système *IoT* a également été favorisée par la montée exponentielle du marché de fabrication des systèmes embarqués. Cette dernière encourage les amateurs et les professionnels à créer leurs propres systèmes et à bricoler ceux qui existent pour trouver des solutions aux problèmes qui leur sont propres.

En effet, L'architecture *IoT* se compose en de trois couches :

- Le côté client.
- Le côté serveur.
- Une voie pour connecter le serveur aux clients.

En addition, les fonctionnalités fondamentales de laboratoire distant exigent des propriétés comme la rapidité, la justesse, l'évolutivité, la disponibilité et la maintenabilité…etc. Ces exigences nous ont guidé à choisir le Raspberry Pi V3 étant donné qu'il réponde parfaitement au cahier de charge demandé pour la réalisation de notre système.

Dans ce chapitre nous allons présenter la Raspberry Pi et son rôle comme un contrôleur dans la globalité du système, et les différents instruments *(PM100D, PRO800),* ainsi que les protocoles de communication et le langage de commande d'instrument *« Standard Commands for Programmable Instruments » (SCPI).* Enfin nous allons voir comment mettre en place les différents composants pour pouvoir implémenter le logiciel.

I.2 Description de l'architecture matérielle

Les laboratoires présentiels de la physique et spécialement de l'optique, demandent un budget très important de la part de l'institution pour avoir des travaux pratiques de qualité. Ce point reste dans la plupart du temps irréalisable, en regardant le nombre de manipulations nécessaires, le temps consacré à la séance du *TP* et la qualité de mesure acceptée.

Pour qu'un laboratoire puisse répondre à tous ces critères, il doit être équipé de plusieurs paillasses identiques avec une moyenne d'une paillasse par étudiant. Cette solution semble d'être impossible au laboratoire traditionnel, mais elle est virtuellement réalisable par un laboratoire distant embarquant une architecture spécifique.

L'architecture matérielle du laboratoire distant devrait être flexible à utiliser et d'autre part, elle doit supporter les manipulations parallèles des étudiants, sans avoir des erreurs de mesures ou d'adressage de résultats. Elle doit aussi supporter un nombre important d'étudiant en même temps sans blocage ou de retard de réponse.

L'architecture matérielle proposée se compose d'un Raspberry Pi qui joue le rôle du contrôleur, en intégrant une interface de communication série *(RS-232 et USB)* pour établir les communications nécessaires avec les instruments utilisés au laboratoire. Et d'instruments dédiés aux travaux pratique de l'optique comme le mesureur de puissance et d'énergie optique *(PM100D)* et le contrôleur modulaire de diode laser *(PRO800)*. **La figure I.1** illustre l'architecture matérielle du système réalisé.

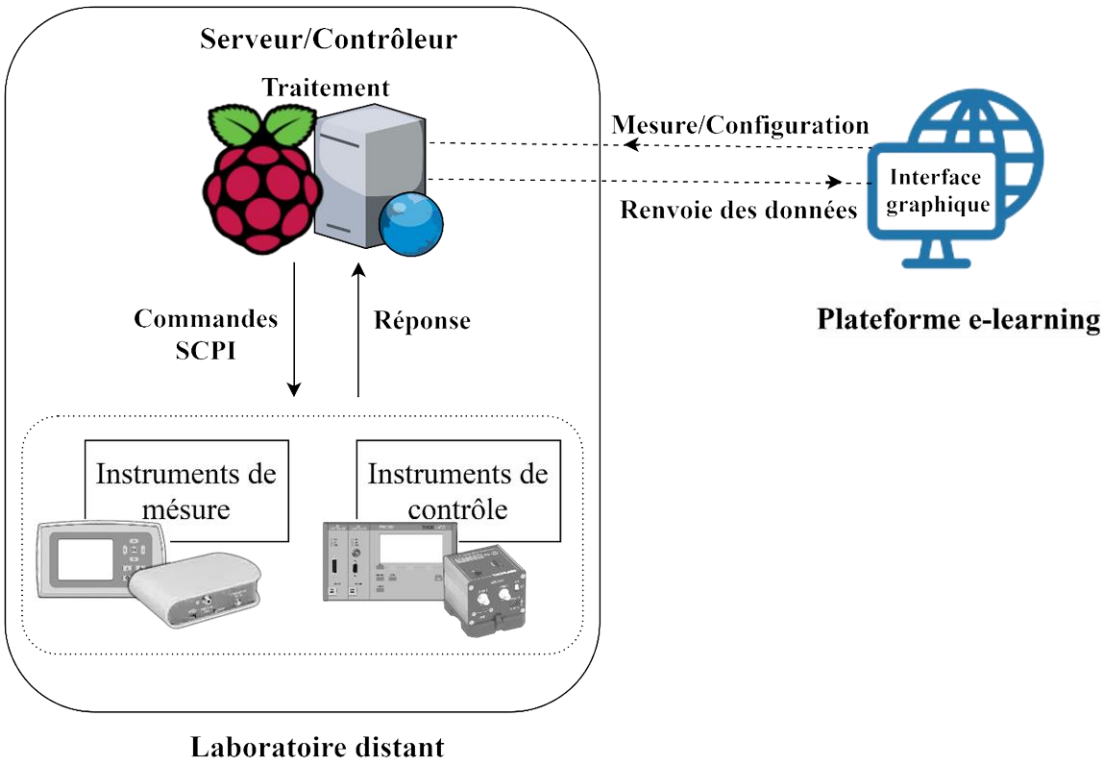

Figure I. 1 : Architecture globale du laboratoire distant.

I.2.1 Le contrôleur Raspberry Pi

Le laboratoire distant que nous allons réaliser impose une architecture matérielle de qualité et de haute gamme, Dont le but est de répondre complètement aux exigences proposées sur les échelles de coût, de performances et de disponibilité…etc.

Le marché des plateformes de développement embarqué est devenu de plus en plus large et plus complexe pour choisir la meilleure solution qui arrange nos exigences. Pour se faire nous avons créé une liste de propriétés minimales que la plateforme choisie doit respecter. Voici la liste ci-dessous :

- Type de carte : nano-ordinateur.
- L'embarquement d'une interface de communication série (*USB, RS-232*).
- La possibilité de sauvegarde des données : carte *MicroSD.*
- Connexion réseau : *Ethernet* ou *Wifi.*
- Vitesse de traitement acceptable.
- Une mémoire vive (*RAM*) acceptable.
- La possibilité d'embarquer un serveur web avec toutes ses ressources.

Après avoir Projeté un œil sur le marché des nano-ordinateurs, nous observons la présence de plusieurs plateformes différentes comme : Raspberry Pi, PcDuino, Orange Pi, Red Pitaya…etc.

Notre choix est porté sur l'utilisation du Raspberry Pi 3 Model B. Vu sa disponibilité au laboratoire *LEOG* et qu'il répond à tous nos besoins ainsi qu'à son prix raisonnable. Voici ci-après les Caractéristiques du Raspberry Pi 3 utilisé :

- Cadencement : 1,2 GHz.
- Processeur : ARM Cortex-A53 64 bits quatre cœurs.
- Mémoire (SDRAM) : 1GB LPDDR2.
- Sauvegarde des données : Carte MicroSD
- Nombre de ports USB 2.0 : 4
- Connexion réseau : 10/100 *Ethernet*, *WiFi* 802.11n et Bluetooth 4.1
- Alimentation : 5v 2.5A via micro-USB

Figure I. 2 : Caractéristiques du Raspberry Pi 3. [2]

La **Figure I. 3** suivante présente le Raspberry Pi 3 utilisé dans notre laboratoire :

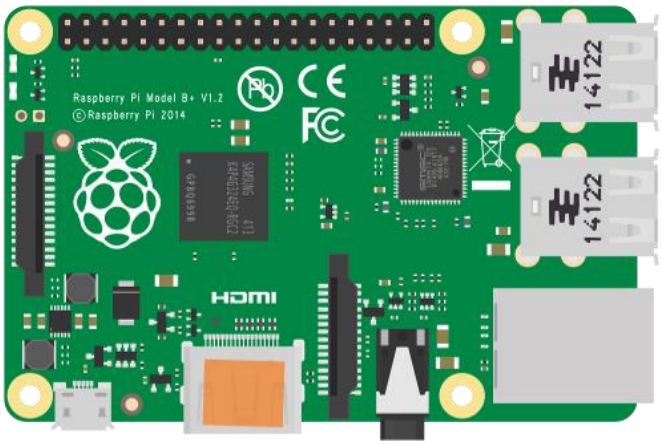

Figure I. 3 : Le Raspberry Pi 3.

L'architecture du laboratoire distant est basée sur le Raspberry Pi qui joue le rôle du cerveau en prenant en charge tous les communications et l'interfaçage avec les instruments utilisés en travaux pratiques ainsi par embarquement du serveur web qui assure l'échange d'informations entre l'étudiant et les instruments.

Remarque : Le Raspberry Pi 3 est doté uniquement de ports de communication de type *USB* donc une adaptation de protocole est envisageable pour l'établissement des liaisons avec le reste du matériel.

I.2.2 Instruments commandés

Thorlabs, un fabricant de produits de photonique, a été fondé en 1989 pour servir le marché d'électro-optique. Alors que ce marché a engendré une multitude d'innovations techniques, Thorlabs a étendu ses compétences fondamentales afin de jouer un rôle de plus en plus important au service de l'industrie de la photonique dans les domaines de la recherche, de l'industrie, des sciences de la vie, de la médecine et de la défense.

Les produits d'analyse de la lumière de *Thorlabs* sont conçus et fabriqués par un groupe divers d'ingénieurs en Amérique du Nord et en Europe. Ils ont une longue expérience dans la fourniture d'équipements photoniques clés sur le marché.

Les atouts de fabrication hautement intégrés et diversifiés de l'organisation comprennent la fabrication de semi-conducteurs, les équipements wattmètres, profileurs de faisceau, détecteurs et contrôleurs de diodes laser sont bien connus et couramment utilisés dans de nombreux laboratoires de photonique.

I.2.2.1 Puissance et énergie mètre optique (Thorlabs PM100D)

Le mesureur optique de puissance et d'énergie *PM100D* est conçu pour mesurer la puissance optique d'une lumière laser ou d'autres sources de lumière monochromatiques ou quasi monochromatiques, ainsi que l'énergie de sources de lumière exposée.

Figure I. 4 : L'instrument de mesure PM100D.

Le *PM100D* propose une conception peu encombrante, alimentée par batterie et compatible avec tous les capteurs à photodiode, thermiques, pyroélectriques, «C-Type » de Thorlabs, et aussi un certain nombre d'options d'affichage, notamment numérique, graphique, aiguille analogique simulée et des statistiques.

Il peut être utilisé manuellement ou à distance via l'interface informatique *USB 2.0*. Une fois connecté à un ordinateur, il est facile d'enregistrer des données à l'aide de l'interface graphique et des pilotes inclus sur une clé *USB.*

Une batterie lithium-polymère rechargeable permet de longs intervalles entre les cycles de charge. L'unité peut être rechargée avec l'adaptateur secteur fourni ou via une connexion *USB* à un PC ou à un ordinateur portable. Voici ci-après les caractéristiques du mesureur PM100D :

- Mesures de puissance et d'énergie optique.
- Grand écran numérique rétroéclairé de 4"
- Batterie rechargeable dure jusqu'à 8h.
- Connectivité *USB 2.0*
- Convertisseur *A/N* 16 bits.
- Slot carte *SD* pour l'enregistrement de données.

Figure I. 5 : Caractéristiques du Mesureur PM100D. [3]

Dans notre laboratoire distant nous allons associer ce mesureur à un capteur photodiode de type *« S120C »* basé sur un détecteur en Silicium à une ouverture ∅ de 9.5 mm , pour la

mesure de puissance optique d'une lumière exposée de longueur d'onde λ entre : $400 - 1100\ nm$.

Remarque : La Connectivité du mesureur *PM100D* est adaptée au support de communication utilisé par le contrôleur (Raspberry Pi). Dans ce cas un câble *USB-USB* est utilisé pour l'établissement de la liaison entre ces derniers.

I.2.2.2 Contrôleur modulaire de diode laser (Thorlabs PRO800)

La série PRO8 est une plate-forme modulaire offrant une solution flexible à presque toutes les exigences de contrôle des diodes laser. Il est disponible en deux versions :

- Une unité compacte pour deux modules *(PRO800)*.
- Une version rack 19" (pouces) pour un maximum de huit modules *(PRO8000).*

Pour une gamme étendue de modules (par exemple, courant unique ou multicanal et contrôleurs de température, commutateurs, amplificateurs de photo courant et sources laser), le système *PRO8* peut être configuré pour presque toutes les applications envisageables.

Dans ce laboratoire nous allons choisir le *PRO800* à deux modules, vu qu'il est idéal pour un système de contrôle flexible à une ou deux diodes lasers. Cet instrument dispose de deux modules de contrôle sous forme de racks (LDC8002, TED8020).

La **Figure I.6** présente le contrôleur de diode laser PRO800 :

Figure I. 6 : L'instrument de contrôle PRO800.

- Nombre maximum de modules : 2.
- Courant de sortie maximal pour tous les modules : 8 A
- Consommation électrique maximale : 220 VA
- Interface de communication : IEEE-488.2 et *RS-232*

Figure I. 7 : Caractéristiques du Contrôleur de diode laser PRO800. [4]

I.2.2.2.1 Module de contrôle de courant (Thorlabs LDC8002)

I.2.2.2.1.1 Présentation du module LDC8002

Le module *LDC8002* est un contrôleur de courant de diode capable de piloter des diodes laser avec différents schémas de connexion - *CG* (mise à la terre cathodique) et *AG* (mise à la terre anodique), ainsi qu'avec ou sans diodes de contrôle. Le module est présenté dans la figure ci-dessous :

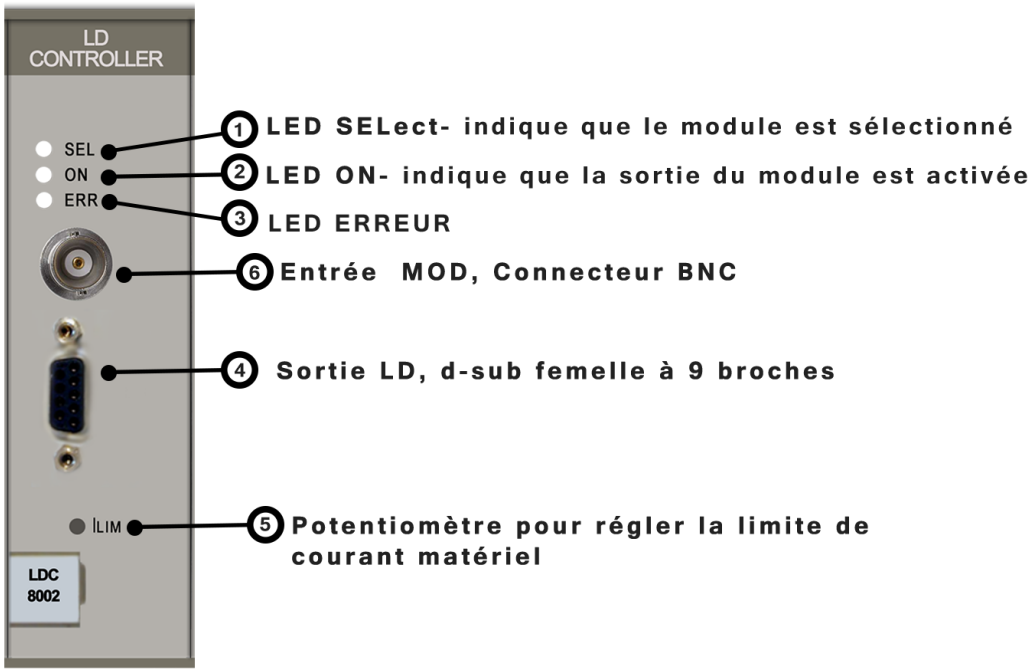

Figure I. 8 : Le module de contrôle LDC8002. [5]

I.2.2.2.1.2 Mode de fonctionnement du module LDC8002

Tous les réglages des valeurs nécessaires sont effectués par les éléments de commande situés sur la face avant du *PRO800* ou par des commandes à distance à l'aide d'un contrôleur. Le seul paramètre à configurer manuellement est la limite physique de courant de la diode laser ("limite matérielle absolue").

Notant aussi que la sortie peut fonctionner en mode *CC (courant constant)* ou *CP (puissance constante)*.

Le courant de la diode laser (mode de courant constant) et le courant de la diode de contrôle (mode de puissance constante) du module *LDC8002* sont réglés avec une résolution de 16 bits entre $\pm$ 4 A .

Les valeurs limites du courant de diode laser (limite logicielle) et du courant de diode de contrôle (limitant la puissance de sortie optique) sont définies avec une résolution de 12 bits.

Le courant de diode de contrôle est relu avec 16 bits, le courant de diode laser, la tension de diode laser et la limite du courant de diode laser (limite matérielle) avec le signe plus 15 bits. Le filtre intégré dans le châssis principal et le blindage du transformateur, du microprocesseur et du module lui-même fourniront une excellente suppression du bruit et des ondulations. [6]

I.2.2.2.2 Module de contrôle de température (Thorlabs TED8020)

I.2.2.2.2.1 Présentation du module TED8020

Le module *TED8020* c'est un régulateur de température capable de contrôler des éléments *TEC*, afin de maintenir une température constante.

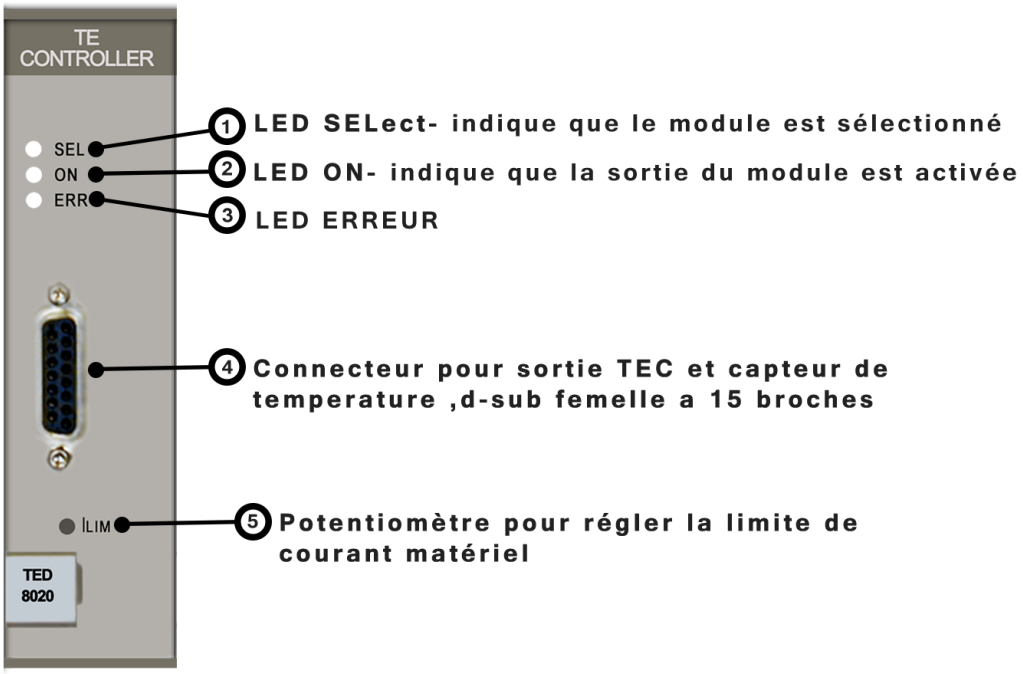

Figure I. 9 : Le module de contrôle TED8020. [7]

Remarque : un élément *TEC* est défini comme étant un dispositif thermoélectrique de refroidissement à semi-conducteurs qui utilise l'effet Peltier [8] pour créer un flux de chaleur entre deux surfaces.

I.2.2.2.2.2 Mode de fonctionnement du module TED8020

Le module *TED8020* contient un amplificateur en boucle fermée avec des paramètres ajustables : *P (proportionnel), I (intégrateur) et D (différentielle).*

Il dispose aussi de trois types de capteurs de température :

- Thermistances standard (*NTC* - Thermistance à coefficient de température négatif) avec deux plages max. 20 KΩ et max. 200 KΩ.
- Capteurs de température à circuit intégré (AD590, AD592, LM335).
- Thermomètre à résistance de platine Pt-100 et Pt-1000 (*RTD* - Détecteurs de température à résistance).

Les valeurs de consigne de température et de résistance du module TED8020 sont définies avec une résolution de 16 bits.

Les valeurs limites du courant *TEC* (limite logicielle) sont définies avec une résolution de 12 bits.

La température réelle (résistance) est relue avec 16 bits, le courant *TEC*, la tension *TEC* et la limite du courant TEC (limite matérielle) avec signe plus 15 bits.

Les paramètres *P*, *I* et *D* de la boucle de commande analogique sont définies par trois convertisseurs numériques / analogique indépendants (*CNA*) par 12 bits. [9]

I.3 Mise en place des différents composants

I.3.1 Interface de communication série

Les ports d'entrée-sortie sont des éléments matériels de l'ordinateur, permettant au système de communiquer avec des éléments extérieurs, c'est-à-dire d'échanger des données, d'où l'appellation d'interface d'entrée-sortie (notée parfois interface d'E/S)

RS-232 est une norme standardisant une voie de communication de type série. Disponible sur presque tous les ordinateurs. Il est de plus en plus remplacé par le port *USB*.

I.3.2 Mise en connexion du matériel

Après avoir cité les différents éléments de notre laboratoire distant et leurs caractéristiques, il est fortement demandé de trouver une méthode de connectivité pour lier les instruments au contrôleur, et pour assurer une communication rapide et fiable.

Le Raspberry Pi 3 possède une communication sous interface *USB,* chose qui nous oblige à convertir les interfaces non adaptées en interface *USB.*

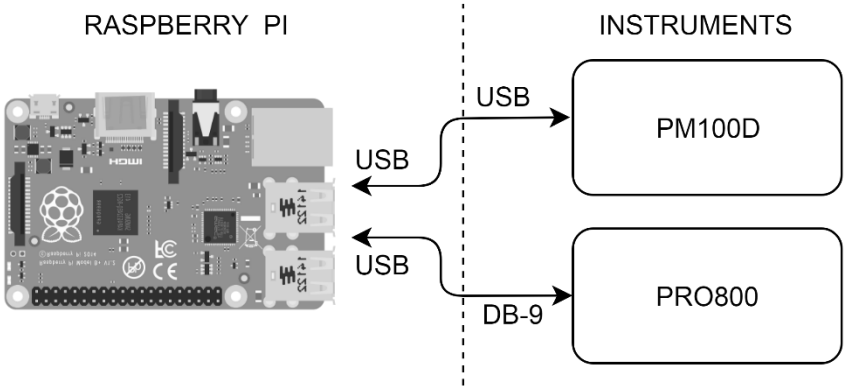

Figure I. 10 : La méthode de liaison des différents équipements.

D'ailleurs l'instrument mesureur *PM100D* possède une interface *USB,* donc un câble *USB*-to-*USB* est suffisant pour échanger les informations avec le Raspberry Pi.

D'autre part, nous pouvons remarquer que l'interface de communication du contrôleur de diode laser *PRO800* n'est pas adaptée à celle du contrôleur (Raspberry Pi). Ce qui nous oblige à apporter une adaptation entre *RS-232* et *USB* via un convertisseur *RS232-to-USB,* avec la configuration de lignes imposée par l'instrument PRO800 illustré dans la figure suivante :

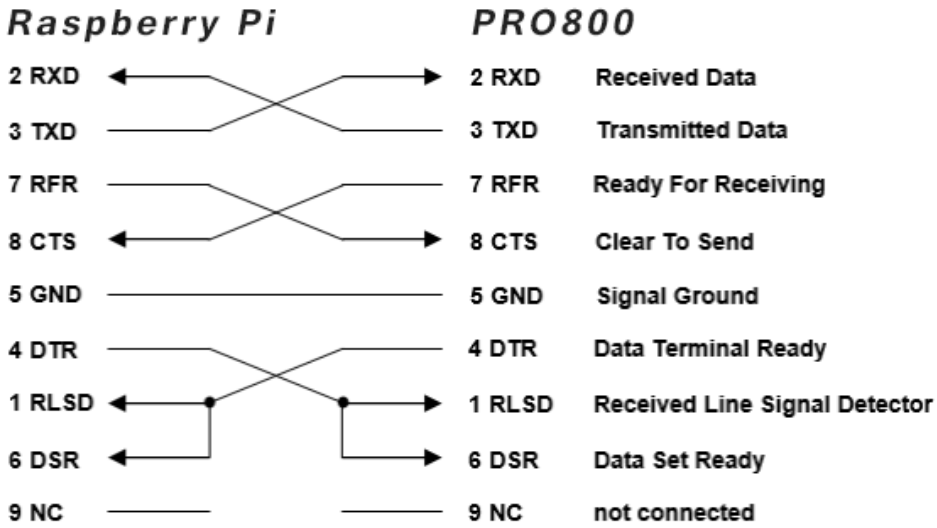

Figure I. 11 : Configuration des lignes de communication PRO800/Raspberry Pi. [10]

I.4 Conclusion

Ce chapitre a été destiné à la présentation de l'architecture matérielle du laboratoire distant. Cette architecture a été choisie d'une façon rigoureuse, afin de répondre parfaitement aux exigences souhaitées qui assurent la réalisation des travaux pratiques de hautes qualités.

Cette architecture possède plusieurs avantages :

- La facilité de la mise en œuvre : l'architecture est basée sur du matériel Open-Source qui offre des solutions rapides et efficaces.
- La facilité de la maintenance : une architecture simple, ordinaire et facile à comprendre.
- La flexibilité et la disponibilité : l'architecture assure une manipulation identique avec la réalité et la possibilité de partage des instruments par plusieurs utilisateurs.
- Un faible coût : une architecture simple et minimale avec un faible coût.

Chapitre 2 : Architecture Logicielle

II.1 Introduction

Dans ces dernières années le monde de développement web a subi un saut exceptionnel avec l'apparition des nouvelles technologies de développement et des plateformes *Open Source* comme *Flask, node.js, Django…etc.*

Cependant, la plupart des applications web sont conçu pour avoir deux parties différentes :

- Le développement Web d'arrière-plan *Back end :* Cette partie est écrite par de nombreux langages courants comme : *Ruby, PHP, Python, Java* et *JavaScript*. Les travaux de développement Web d'arrière-plan comprennent souvent ce que l'on pourrait appeler des tâches « administrateur système »,
- Le développement Web du plan-avant *Front end :* Cette partie est écrite par des langages comme : *HTML, CSS, JavaScript, jQuery et React.*

Cette diversité des ressources et des langages de programmations crée une liberté dans le choix de l'architecture, d'où vient la possibilité de réaliser n'importe quel système avec les performances souhaitées, sans avoir la nécessité de dépenser beaucoup d'argent.

D'autre part, Les laboratoires distant qui existent actuellement utilisent des logiciels coûteux et moins flexibles comme le *LabVIEW* et les logiciels propriétaires livrés par le producteur du matériel.

Cependant, le laboratoire distant que nous visons à réaliser doit répondre parfaitement aux exigences proposées, toute en le gardant *Open Source.*
Donc un développement logiciel en deux échelles est probable, le premier pour l'établissement de la communication entre le contrôleur et les instruments, et le deuxième pour la création d'un serveur qui assure la connexion entre les clients et le serveur.

Dans ce chapitre nous allons présenter l'architecture logicielle ainsi que les techniques de développement web utilisées pour la mise en œuvre du système de contrôle à distance.

II.2 Description de l'architecture logicielle

L'architecture logicielle du laboratoire distant devrait être flexible à utiliser ainsi qu'elle doit supporter les manipulations parallèles des étudiants, sans avoir des erreurs de mesures ou d'adressage de résultats. Elle doit aussi supporter un nombre important d'étudiants en même temps sans blocage ou de retard de réponse.

L'interface graphique de l'application web doit également être rapprochée le plus possible à la réalité pour faciliter la manipulation et la configuration des instruments, et aussi pour laisser les étudiants ressentir la même chose comme s'ils réalisent des travaux pratique en présentiel.

Nous allons adopter une architecture basée sur des modules séparables de telle sorte que nous puissions intervenir très facilement pour une éventuelle maintenance ou une modification partielle envisageable.

L'architecture de notre laboratoire distant est basée sur le transfert des requêtes entre des clients (étudiants) et un serveur (système), ce mode de transfert de requêtes est dit serveur à multi-clients. Donc ce serveur apparaît unique pour chaque utilisateur mais il est réellement partagé entre tout le monde. Il est nécessaire aussi de citer que le serveur doit répondre en temps réel avec un temps de retard minimal.

Prenant en compte toutes ces exigences, nous avons choisi une architecture logicielle simple, efficace, détachable sous forme de module, rapide et *Open Source.*

Cette architecture est basée sur la création d'un serveur web qui prend en charge toutes les opérations allant de la réception des requêtes jusqu'à la communication avec le matériel. Le serveur est divisé en deux parties comme suit :

- Le côté serveur : cette partie est réalisée à l'aide de la Framework *Web FLASK* et des modules et classes écrits en langage *Python.*
- Le côté client : l'interface graphique est réalisée par le fameux *HTML, CSS, JavaScript.*

En outre, nous avons profité de plusieurs concepts de la programmation orientée objets, en héritant des classes déjà écrites en *Python* par des développeurs pour l'établissement de la communication avec les instruments comme la bibliothèque *PySerial,* et aussi le concept de classe par la création de classes comme *PRO800* et *PM100D* ainsi que l'utilisation des modules séparable pour faciliter la maintenance.

Nous avons aussi utilisé plusieurs d'autre ressources comme : *AJAX, JSON* et autres bibliothèques comme *USBTMC.* La **Figure II. 1** suivante présente un schéma synoptique qui résume les ressources utilisées pour le développement de l'architecture logicielle.

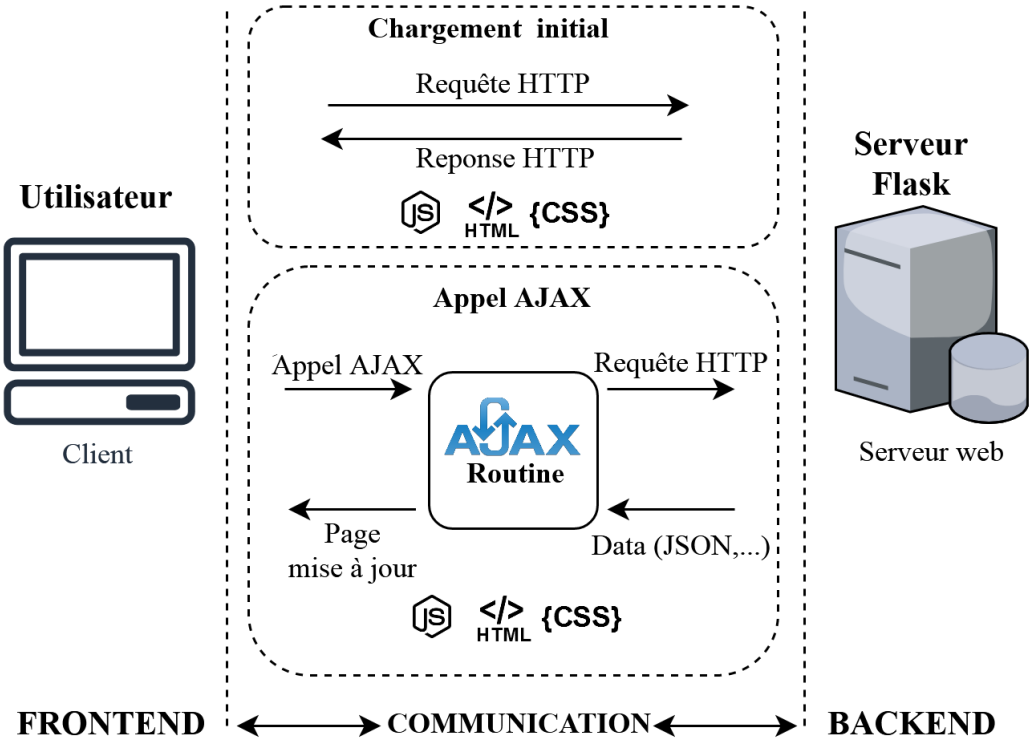

Figure II. 1 : L'architecture logicielle du laboratoire distant.

II.2.1 Environnement Client/serveur

L'environnement client/serveur désigne un mode de communication organisé par l'intermédiaire d'un réseau et d'une interface *Web* entre plusieurs ordinateurs.

Cela signifie qu'une relation client/serveur se résume sur un programme (le client) qui demande un service ou une ressource à un autre programme (le serveur).

Des machines clientes contactent un serveur qui fournit des services. Ces derniers sont exploités par des programmes (*Frontends*), ainsi que ces requêtes s'exécutent sur des serveurs (*backend*),

Le concept en général s'effectue sous un réseau partagé. Dans ce cas, le client doit établir une connexion au serveur en passant par un réseau local *(LAN)* ou étendu *(WAN)*, tel que l'internet.

II.2.1.1 Le protocole HTTP

L'acronyme *HTTP (protocole de transfert hypertexte).* C'est un protocole qui définit la communication entre un client (sous navigateur) et un serveur sur le *World Wide Web (WWW).*

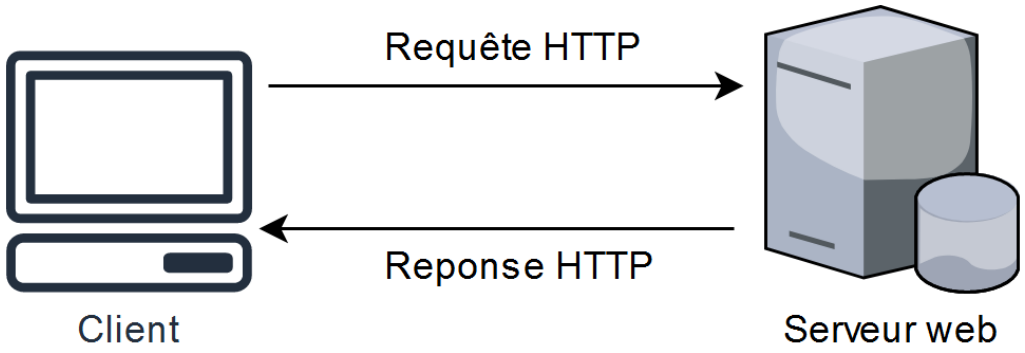

Figure II. 2 : Description du protocole HTTP.

Le principe "requête-réponse" sert à une communication entre un navigateur *Web* (le client) et un serveur *Web*, cela déroule d'une manière à envoyer une requête à un serveur web. Cette requête demande un document (exemple : page *HTML*, image, fichier *CSS* ...). Le serveur cherche les informations, puis il est parfois amené à interpréter les résultats sous des formes normalisées, pour finalement envoyer la réponse. Cette réponse contient les entêtes du protocole *HTTP* en combinaison avec le contenu demandé.

II.2.2 Côté client

Le coté client *(Frontend)* est parmi les points les plus importants pour l'utilisateur, car il donne une impression sur le système par sa représentation de l'interface graphique de l'application web. Plus cette interface est rapprochée à la réalité plus la satisfaction des utilisateurs est meilleure.

La réalisation du côté client est principalement sur l'*HTML*, *CSS* et *JavaScript*.

II.2.2.1 Création de l'interface graphique

Dans cette étape nous allons parler sur les méthodes adoptées pour la création de la page *Web*.

Pour faire nous avons découpé le travail en deux parties :

- Une partie statique (*HTML*/*CSS*) : Créer le contenu de la page *Web*.
- Une partie dynamique (*JavaScript*) : Donner la vie au contenu de la page.

II.2.2.1.1 Structure de la page web (HTML/CSS)

II.2.2.1.1.1 HTML

L'*HTML (HyperText Markup Language*) ou « *langage de balisage d'hypertexte* » en français, est un langage utilisé pour la création des pages web. Cette signification porte bien son nom puisqu'effectivement ce langage permet de réaliser de l'hypertexte à base d'une structure de balisage.

HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web.

Les blocs de construction des pages *HTML* sont souvent appelés des éléments *HTML*. Avec ces éléments, des images et d'autres objets tels que des formulaires interactifs peuvent être incorporés dans la page rendue.

HTML fournit un moyen pour créer des documents structurés en indiquant une sémantique structurelle pour un texte tel que des en-têtes, des paragraphes, des listes, des liens, des guillemets et d'autres éléments. Les éléments *HTML* sont délimités par des balises, écrites à l'aide de chevrons. Des balises telles que <*img* /> et <*input* /> introduisent directement le contenu dans la page. D'autres balises telles que <*p*> fournissent des informations sur le texte du document et peuvent inclure d'autres balises en tant que sous-éléments. La **Figure** suivante illustre la structure minimale d'une page *HTML*.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1"/>
<title> Titre de la page </title>
<!-- Référence à une feuille de style externe -->
<link rel="stylesheet" type="text/css" href="styles.css" />
<style>
/* Styles CSS globaux spécifiques à la page */
</style>
</head>
<body>
<p> Ceci est un paragraphe ***</p>
<script>
Script écrit JavaScript
</script>
</body>
</html>
```

Figure II. 3 : la structure minimale d'un page HTML.

Les navigateurs n'affichent pas les balises *HTML*, mais ils les utilisent pour interpréter le contenu de la page.

HTML peut aussi intégrer des programmes écrits dans un langage de script tel que JavaScript, ce qui affecte le comportement et le contenu des pages *Web*.

Dans notre interface nous allons créer d'abord les corps de la page à l'aide de l'*HTML*, en utilisant des images, des boutons, des paragraphes…etc.

II.2.2.1.1.2 CSS

Le *CSS* « *Cascading Style Sheets* » ou « feuilles de style en cascade ». Ces feuilles de style sont utilisées pour décrire la présentation d'un document écrit dans un langage de balisage tel que *HTML*.

CSS est conçu pour permettre la séparation entre la présentation et le contenu, y compris la mise en page, les couleurs et les polices. Cette séparation peut améliorer l'accessibilité au contenu, offrir plus de flexibilité et de contrôle dans la spécification des caractéristiques de présentation, permettre à plusieurs pages *Web* de partager le formatage en spécifiant le *CSS* dans un fichier .css séparé, et aussi réduire la complexité et la répétition du contenu structurel.

La séparation du formatage et du contenu permet également de présenter la même page de balisage dans différents styles pour différentes méthodes d'interprétation.

Le nom en cascade provient du schéma de priorités spécifiées pour déterminer quelle règle de style s'applique si plusieurs règles correspondent à un élément particulier. Ce schéma de priorité en cascade est prévisible. La figure ci-dessous présente la structure d'une feuille *CSS.*

```
#para1 {
  text-align: center;
  color: red;
}

.center {
  text-align: center;
  color: red;
}

h1 {
  text-align: center;
  color: red;
}
p {
  text-align: center;
  color: red;
}
```

Figure II. 4 : la structure d'une feuille *CSS*.

II.2.2.1.2 Interaction de la page web (Javascript)

JavaScript, souvent abrégé en *JS*, est un langage de programmation haut niveau pour le *Web*, il est basé sur les concepts de la programmation orientation objet. Il est utilisé pour modifier le code *HTML* et le *CSS* d'une page web, en plus *JavaScript* interprète les pages *Web* de manière interactive et dynamique. Cela permet aux pages de réagir aux événements, d'afficher des effets spéciaux, d'accepter des textes variables, de faire des calculs, de valider des données, de créer des cookies, de détecter le navigateur de l'utilisateur…etc.

Dans notre projet le JavaScript est aussi utilisé pour donner à l'étudiant l'impression qu'il travaille sur l'instrument lui-même, en ajoutant des couleurs, des mouvements, des boutons interactifs…etc.

Dans le but aussi de rendre réelles les manipulations faites sur le côté client, nous avons insisté à créer une page web qui s'actualise sans la recharger (presque automatique). Pour cette nouvelle fonctionnalité nous avons utilisé une technique appelée *AJAX*.

AJAX (*Asynchronous Javascript and XML*) est un ensemble de techniques de développement *Web* utilisées pour créer des applications *Web* asynchrones. Avec *AJAX*, les applications *Web* peuvent envoyer et récupérer des données d'un serveur de manière asynchrone (en arrière-plan) sans interférer avec l'affichage et le comportement de la page existante. En séparant la couche d'échange de données de la couche de présentation, *AJAX* permet aux pages *Web* de modifier le contenu de manière dynamique, sans qu'il soit nécessaire de recharger l'intégralité de la page. En pratique, les implémentations modernes utilisent généralement *JSON* au lieu de *XML*.

L'objet *XMLHttpRequest* intégré ou la nouvelle fonction « *fetch ()* » de JavaScript sont couramment utilisés pour exécuter *AJAX* sur des pages *Web*, permettant aux sites *Web* de charger du contenu à l'écran sans actualiser la page. *AJAX* n'est pas une nouvelle technologie, ni un langage différent, mais uniquement les technologies existantes utilisent de nouvelles manières. **La Figure II.5** suivante présente le concept de la technique *AJAX*.

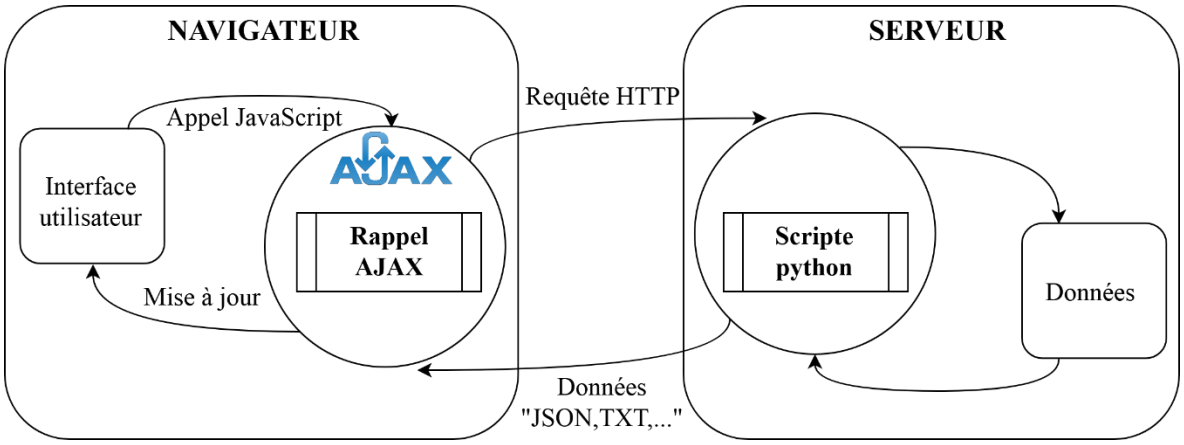

Figure II. 5 : Représentation de la technique AJAX [11].

Dans le but de réaliser une interface client riche et complète nous avons utilisé d'autres ressources comme le *LocalStorage*, *Chart.js*.

II.2.2.1.2.1 Stockage local (LocalStorage)

LocalStorage est un type de stockage *Web* qui permet aux sites *Web* et aux applications *JavaScript* de stocker et d'accéder aux données directement dans le navigateur, sans date d'expiration. Cela signifie que les données stockées dans le navigateur persisteront même après la fermeture de la fenêtre du navigateur.

Cette solution de stockage des données dans le côté client apporte plusieurs avantages aux étudiants, dans le cas où il oublie de prendre ou il veut consulter ses résultats. Avec une capacité de stockage de 60 mesures, qui s'actualisent chaque manipulation.

II.2.2.1.2.2 Traçage de graphe (Chart.js)

Chart.js est une bibliothèque *JavaScript* open source qui permet de dessiner et de visualiser les données en différents types de graphiques à l'aide de l'élément « *canvas* » de *HTML5*.

Cet outil est flexible et facile dans sa programmation et son fonctionnement, ainsi il est capable de créer un contenu lisible et significatif.

L'étudiant peut visualiser le graphe d'évolution de ses mesures en temps réel, et d'utiliser le graphe sans l'avoir retracé.

II.2.3 Côté serveur

Le coté serveur ou *backend* représente une partie extrêmement sensible dans la réalisation du laboratoire distant, cette partie assure en premier lieu l'échange d'informations entre le contrôleur et les instruments contrôlés puis elle communique ces informations (sous forme de données étiquetées) avec les clients selon les demandes.

Le côté serveur est réalisé principalement à l'aide du langage *Python*.

Nous avons divisé la mise en œuvre du serveur en deux parties essentielles :

- Communication en échelle du laboratoire (Communication physique contrôleur-instruments).
- Echange des données à l'aide du serveur web via internet.

II.2.3.1 Protocole de Communication

La communication sous l'échelle contrôleur-instruments affecte d'une façon directe la qualité de l'application *Web* et le temps de réponse du système globale. Donc nous devons assurer une bonne communication entre ces derniers.

II.2.3.1.1 Le protocole SCPI

Standard Commands for Programmable Instruments (SCPI) définissent une norme pour la syntaxe et les commandes à utiliser pour contrôler les dispositifs de contrôle et de mesure programmables, tels que les équipements de contrôle automatique et les équipements de test électronique.

SCPI a été défini comme une couche supplémentaire au-dessus de la spécification *IEEE 488.2*. La norme spécifie une syntaxe, une structure de commande et des formats de données communs. Ces commandes sont regroupées dans des sous-systèmes.

La liaison de communication matérielle physique pour l'*SCPI* peut être utilisée sous le bus *IEEE-488.1* (*GPIB*), *RS-232*, RS-422, Ethernet, *USB*…etc.

Les commandes *SCPI* sont des chaînes de texte *ASCII* envoyées à l'instrument par la couche physique (*USB, RS-232…etc.*). Les commandes sont sous forme d'une série d'un ou plusieurs mots-clés, dont beaucoup prennent des paramètres. Les mots clés complets peuvent être utilisés ou abrégés uniquement en partie majuscule comme (*CORRection, CORR*). Les réponses aux commandes de requête sont généralement des chaînes *ASCII*.

La communication avec les instruments est réalisée avec ce protocole, en créant des classes en Python *(PM100D.py et PRO800.py)* qui regroupent toutes les fonctions intégrées dans chaque instrument. Ces classes sont utilisées pour faciliter la programmation, et minimiser la taille du code source. **La Figure II.6** suivante montre un petit morceau de la classe créée *PRO800*.

```
from serial import *

class PRO800(Serial):

    def __init__(self,PortAdd, Baudrate):
      self = Serial.__init__(self,PortAdd,
      Baudrate,timeout=1,bytesize=8,stopbits=1,rtscts=False,
      dsrdtr=True)

    # Getting the device IDN
    def get_idn(self):
        self.write("*IDN?\n".encode())
        return self.readline().decode('ascii').rstrip()

    # Switching device on Local Mode
    def go_to_local(self):
        self.write("&GTL\n".encode())

    def clear_device(self):
        self.write("&DCL\n".encode())

    # Getting the plug-in Modules configuration
    def ask_config(self):
        self.write(":CONFIG:PLUG?\n".encode())
        return self.readline().decode('ascii').rstrip()
```

Figure II. 6 : Un morceau de la classe PRO800.py.

II.2.3.2 Création du serveur web

Le serveur Web c'est l'outil qui prend en charge les échanges entre les clients et le matériel du laboratoire distant. Ces échanges sont divisés en deux niveaux :

- Routage des requêtes faites par les clients (serveur *FLASK*).
- Traitement et exécution des différentes demandes (programme *Python* relié au server).

II.2.3.2.1 Serveur FLASK

FLASK est un Framework d'application *Web Server Gateway Interface (WSGI)* léger. Il est conçu pour permettre une mise en route rapide et facile du développement *Web*, avec la possibilité d'évoluer vers des applications complexes. Il a commencé comme une simple enveloppe autour de *Werkzeug*[1] et *Jinja*[2] puis il est devenu l'un des cadres d'application *Web Python* les plus populaires.

Flask propose des suggestions, mais n'impose aucune dépendance ni structure de projet. Il appartient au développeur de choisir les outils et les bibliothèques qu'il souhaite utiliser, grâce

[1] Système de routage sous forme d'une bibliothèque complète d'applications Web *WSGI*.
[2] Un langage de modélisation moderne et convivial pour les environnements web écrits en Python.

aux nombreuses extensions fournies par la communauté qui facilitent l'ajout de nouvelles fonctionnalités.

Le serveur est réalisé à l'aide d'un programme écrit en *Python (Server.py).* C'est un programme qui englobe et utilise dans son exécution tous les autres modules, comme le module *Instruments.py* qui effectue le transfert des ordres et d'informations entre le contrôleur et le matériel. Il embarque aussi une fonction réservée à la traçabilité des évènements et des requêtes effectuées par les étudiants.

II.2.4 Traitement des requêtes (client/serveur)

Le laboratoire distant doit prendre en charge un nombre intéressant d'étudiants qui peuvent manipuler les instruments en même temps et par conséquent nous avons adopté une méthode de traitement de requêtes qui consiste à :

- Traiter chaque demande individuellement : chaque client obtient les réponses demandées.
- N'exécuter qu'une seule demande à la fois : les instruments ne peuvent pas effectuer plusieurs instructions simultanément.
- Créer une queue (buffer) pour les clients qui rentrent en même temps : le premier arrivé est le premier servi.
- Bloquer la rentrée en état de communication avec un instrument s'il y'est déjà : Donner le temps nécessaire à l'instrument pour qu'il soit libre.
- Utiliser un algorithme qui détecte les paramètres modifiés et non modifiés lors d'une requête : mettre à jour que les paramètres modifiables.

La Figure suivante représente un organigramme qui résume la méthode utilisée pour le traitement des requêtes.

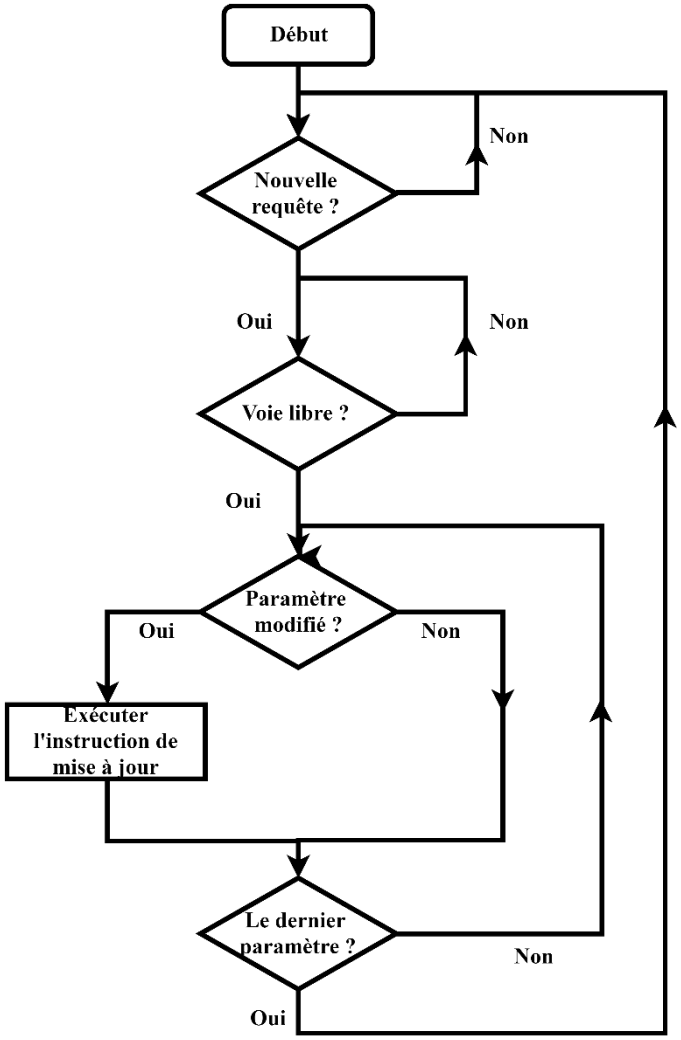

Figure II. 7 : Organigramme de traitement des requêtes.

Remarque : les informations échangées entre le client et le serveur viennent sous forme de données notées en *JavaScript Object Notation (JSON),* cette notation est utilisée pour faciliter la collection, l'analyse, et le traitement des données.

JSON est construit sur deux structures :

- Une collection de paires nom / valeur.
- Une liste ordonnée de valeurs.

II.3 Mise en connexion du logiciel

Après avoir choisi l'architecture logicielle adaptée à notre laboratoire distant, en présentant chaque technique utilisée, il est donc nécessaire de trouver une technique pour relier tous ces programmes, sans changer dans le fonctionnement global souhaité.

Les serveur *FLASK* choisi pour notre application nous oblige à utiliser une méthode pour regrouper les différents programmes qui consiste à :

- Créer un répertoire pour mettre tous les programmes des deux côtés serveur et client.
- Créer un dossier appelé *« templates »* qui contient les fichiers « .html » de la page Web.
- Créer un dossier appelé *« static »* qui contient les fichiers statiques comme les images, le code *« .js »*, le code *« .css »* et les bibliothèques *JavaScript* utilisées
- Déposer les programmes *Python* dans le répertoire principal.

Voir la figure ci-dessous pour plus de détail.

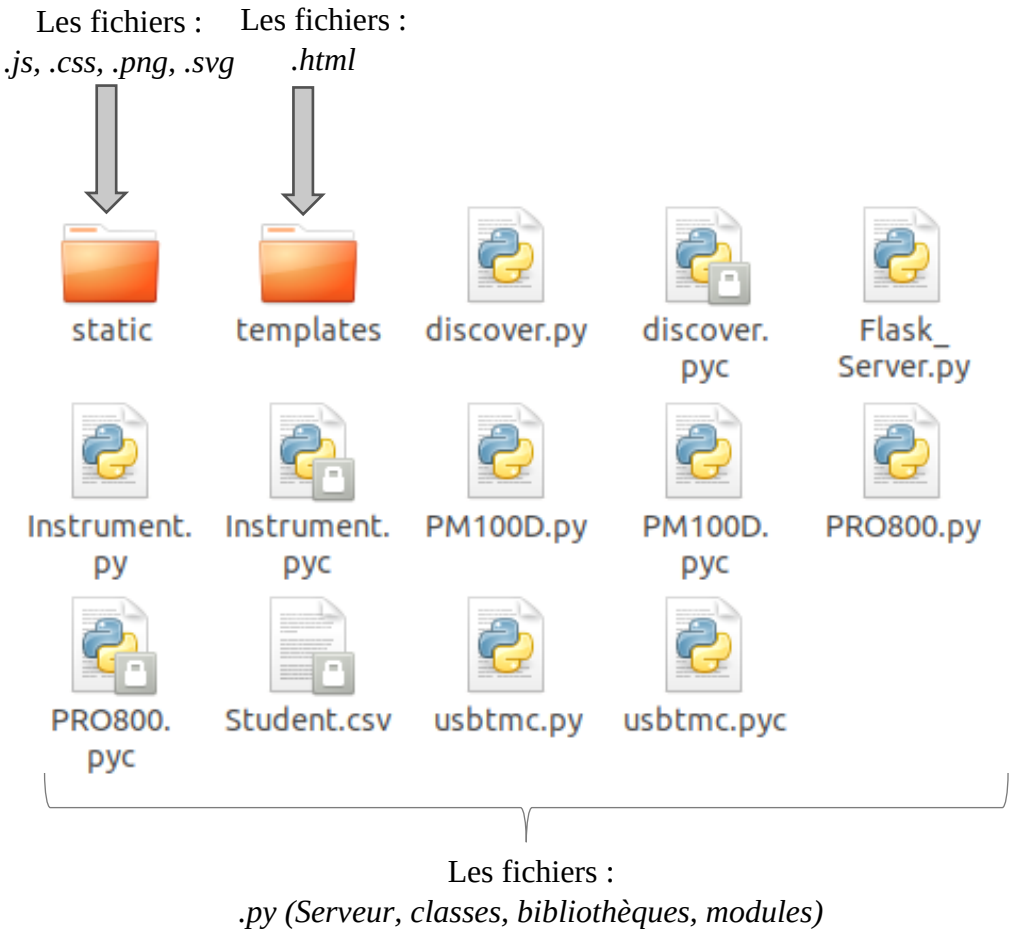

Figure II. 8 : Le regroupement des différents programmes de l'application Web.

II.4 Conclusion

Ce chapitre a été consacré à la présentation de l'architecture logicielle du laboratoire distant. Durant la conception logicielle du système, nous avons utilisé des techniques de programmation qui s'adaptent avec le matériel utilisé, et qui répondent également aux exigences du laboratoire fixées auparavant.

Cette architecture apporte plusieurs avantages :

- La facilité de programmation : Les langages de programmation utilisés possèdent une très grande communauté sur internet.
- Modification facile de programme : La méthode utilisée est basée sur les classes et les modules (Parties de code séparées).
- Coût de conception presque nul : Une architecture basée sur des ressources *open source*.

Chapitre 3 : Implémentation et Tests de fonctionnement

III.1 Introduction

Après la réalisation des architectures matérielle et logicielle séparément, il est donc le temps de procéder à l'étape finale qui consiste à conjoindre le matériel et le logiciel ensemble. Donc une implémentation du logiciel dans le matériel est indispensable.

Dans ce chapitre nous allons présenter la méthode suivie pour effectuer l'implémentation logiciel/matériel, puis nous allons tester les fonctionnalités du système final.

III.2 Implémentation logiciel /matériel

III.2.1 Installation préalable requise

L'architecture logicielle utilisée impose une préparation préalable du matériel, qui consiste à :

- Installer l'environnement *Python 3.7* (recommander) au Raspberry Pi :
  Par l'exécution de la commande suivante dans la ligne de commande :
  ```
  $ sudo apt install python3.7
  ```
- Installer le gestionnaire de paquets de *Python3.7 « pip3 » :*
  Par l'exécution des commandes suivantes :
  ```
  $ sudo apt update
  $ sudo apt install python3-pip
  ```
- Installer la bibliothèque de communication série *usbtmc* pour l'instrument *PM100D* :
  Par l'exécution de la commande suivante :
  ```
  $ sudo pip3 install python-usbtmc
  ```
- Installer la bibliothèque de communication série *pyserial* pour l'instrument *PRO800* :
  Par l'exécution de la commande suivante :
  ```
  $ sudo pip3 install pyserial
  ```
- Installer le Framework du serveur *FLASK* :
  Par l'exécution de la commande suivante :
  ```
  $ sudo pip3 install flask
  ```
- Déposer les fichiers des programmes dans un dossier sous une structure :
  Nous avons déjà abordé cette structure dans la partie **Mise en connexion du logiciel** du **chapitre 2.**

III.2.2 Reconnaissance du matériel

Après l'installation du logiciel requis nous allons parler sur les méthodes utilisées pour la reconnaissance des instruments connectés au Raspberry Pi, étant donné que ces instruments

se connectent chaque fois dans le port *USB* sur des adresses différentes, par suite nous avons adopté deux méthodes de connexion :

III.2.2.1 Reconnaissance par identificateur

Cette méthode est utilisée pour le mesureur *PM100D*, elle consiste à :

- Connecter l'instrument au Raspberry.
- Lancer la ligne de commande et exécuter la commande suivante :
  `$ lsusb`
- Chercher le nom de l'instrument et prendre ses identificateurs (idVendor, idProduct) : ces identificateurs sont uniques au mesureur *PM100D*.

La deuxième et la troisième étape sont entamées uniquement une seule fois, et tout le reste devient automatique.

- Ouvrir la communication série à l'aide des codes suivants :

Fichier de la classe *PM100D.py :*

```
from usbtmc import *

class PM100D(Instrument, UsbtmcException):

    def __init__(self, idVendor, idProduct):
        self = Instrument.__init__(self,idVendor, idProduct)
      .
      .
      .
```

Fichier du module *Instrument.py :*

```
from PM100D import *

def PM100D_Begin():
    os.system("sudo modprobe usbserial vendor=0x1313 product=0x8078")
    return PM100D(4883,32888)
```

Fichier *Serveur.py :*

```
from Instrument import *

PM100D = PM100D_Begin()
.
.
.
```

Figure III. 1 : Un morceau du programme pour l'ouverture de la communication avec PM100D.

Remarque : La dernière partie se fait automatiquement lors du lancement du serveur.

III.2.2.2 Reconnaissance par adresse

Cette méthode est adaptée au contrôleur de diode laser *PRO800*, elle consiste à :

- Brancher l'instrument au Raspberry.
- Reprendre la deuxième et la troisième étape comme l'instrument *PM100D*.
- Chercher l'adresse de l'instrument au port série à l'aide du module *discover.py* dans la figure suivante :

```
def discover_port():
    import usb
    import os
    from serial.tools.list_ports import comports
    import platform

    serial_ports = [(x[0], x[1], dict(y.split('=', 1) for y in
x[2].split(' ') if '=' in y)) for x in comports()]

    for dev in usb.core.find(find_all=True, custom_match= lambda x:
x.bDeviceClass != 9):
        try:
            dev._langids = (1033, )
            if not (dev.manufacturer == 'Thorlabs'):
                continue
        except usb.core.USBError:
            pass

        port_candidates = [x[0] for x in serial_ports if
x[2].get('VID:PID', None) == '067B:2303']

    assert len(port_candidates) == 1
    return port_candidates[0]
```

Figure III. 2 : Le programme qui cherche l'adresse d'un instrument. [12]

Ce programme prend les identificateurs (idVendor, idProduct) de l'instrument PRO800 comme arguments, et il retourne son adresse sur le port *USB*.

- Ouvrir la communication série à l'aide des scripts suivants :

Fichier de la classe *PRO800.py :*

```
from serial import *

class PRO800(Serial):

    def __init__(self,PortAdd, Baudrate):
        self = Serial.__init__(self,PortAdd,
Baudrate,timeout=1,bytesize=8,stopbits=1,rtscts=False, dsrdtr=True)
```

Fichier du module *Instrument.py :*

```
from PRO800 import *

def PRO800_Begin():
    return PRO800(discover_port(), 19200)
```

Fichier *Serveur.py:*

```
from Instrument import *

PRO800 = PRO800_Begin()
```

Figure III. 3 : Un morceau du programme pour l'ouverture de la communication avec PRO800.

III.3 Tests et fonctionnalité

Dans cette partie nous allons parler sur le fonctionnement du laboratoire distant en présentant toutes les fonctions intégrées dans le système.

III.3.1 Présentation de l'interface web

La constitution de l'interface client du laboratoire distant propose plusieurs fonctionnalités aux étudiants à travers une plateforme e-learning, parmi lesquelles on citera : la manipulation libre est instantanée des instruments, la représentation simplifiée et significative des mesures, la semblance graphique entre la réalité et l'application web, et encore d'autres fonctionnalités.

L'accès distant au laboratoire peut avoir lieu sur n'importe quel navigateur web comme (Google *Chrome*, *Mozilla Firefox*, *Opera*…etc.)

Pour accéder au laboratoire il est nécessaire que l'enseignant lance le serveur web, et l'étudiant doit avoir une connexion internet ainsi qu'un compte sur la plateforme Moodle de l'université.

III.3.1.1 L'interfaçage du mesureur PM100D

L'interfaçage de l'instrument PM100D est réalisé par une *GUI (html, css, JavaScript, images, icons…etc.)* qui permet l'interaction avec ce dispositif de mesure.

D'autre part, l'étudiant peut effectuer des manipulations de contrôle et rentrer des paramètres de mesures différents. Ces paramètres de configuration sont envoyés au serveur à chaque fois l'étudiant décide de les appliquer sur l'appareil.

Cette configuration est communiquée avec l'instrument via une requête *AJAX* sous le protocole *http*, qui consiste à envoyer des paramètres sous forme de données *JSON* comme suit :

```
var Data_to_send = {
                                wavelength :Wavelength,
                                attenuation : Attenuation,
                                beamdiameter : BeamDiameter,
                                bandwidth_HI_LO : Bandwidth_HI_LO,
                                zeronig: Zeroing,
                                auto_range : Auto_Range,
                                current_range_send : Current_Range_Send
                              };
```

Figure III. 4 : Les paramètres communiqués au PM100D chaque requête.

En retour le serveur envoie de la même manière tous les paramètres nécessaires pour actualiser l'affichage à l'aide d'un dictionnaire *Python* converti en données *JSON*. La figure suivante montre la structure de la réponse obtenue par le serveur.

```
OUT = dict();
 OUT['Power_W'] = PM100D.get_meas("POW")
 OUT['Current_A'] = PM100D.get_meas("CURR")
 OUT['Attenuation'] = PM100D.get_attenuation_factor("")
 OUT['WaveLength'] = PM100D.get_wavelength("")
 OUT['BeamDiameter'] = PM100D.get_beam_diameter("")
 OUT['Power_Range'] = PM100D.get_meas_range("POW","")
 OUT['Current_Range'] = PM100D.get_meas_range("CURR","")
 OUT['Auto_Range'] = PM100D.get_meas_autorang("POW")
 OUT['BandWidth_HI_LO'] = PM100D.get_pdiode_band()
 OUT['Max_Power_Range'] = PM100D.get_meas_range("POW","MAX")
 OUT['Max_Current_Range'] = PM100D.get_meas_range("CURR","MAX")
 OUT['Zero_Value'] = PM100D.get_zero_value()
 OUT['Frequency'] = PM100D.get_meas("FREQ")
 OUT['Temperature'] = PM100D.get_meas("TEMP")
 OUT['Power_Dens'] = PM100D.get_meas("PDEN")
```

Figure III. 5 : Les paramètres de réponse pour l'instrument PM100D.

Dans le coté client tous ces paramètres vont être traités et adaptés par un programme *.js* pour à la fin les présenter au client.

III.3.1.1.1 Les méthodes de représentation des mesures (PM100D)

Dans cette partie nous allons procéder à la description de quelques méthodes d'affichage des résultats sur le mesureur PM100D.

- **Le menu principal du PM100D** :
  Un affichage numérique qui présente presque toutes les informations sur l'appareil.

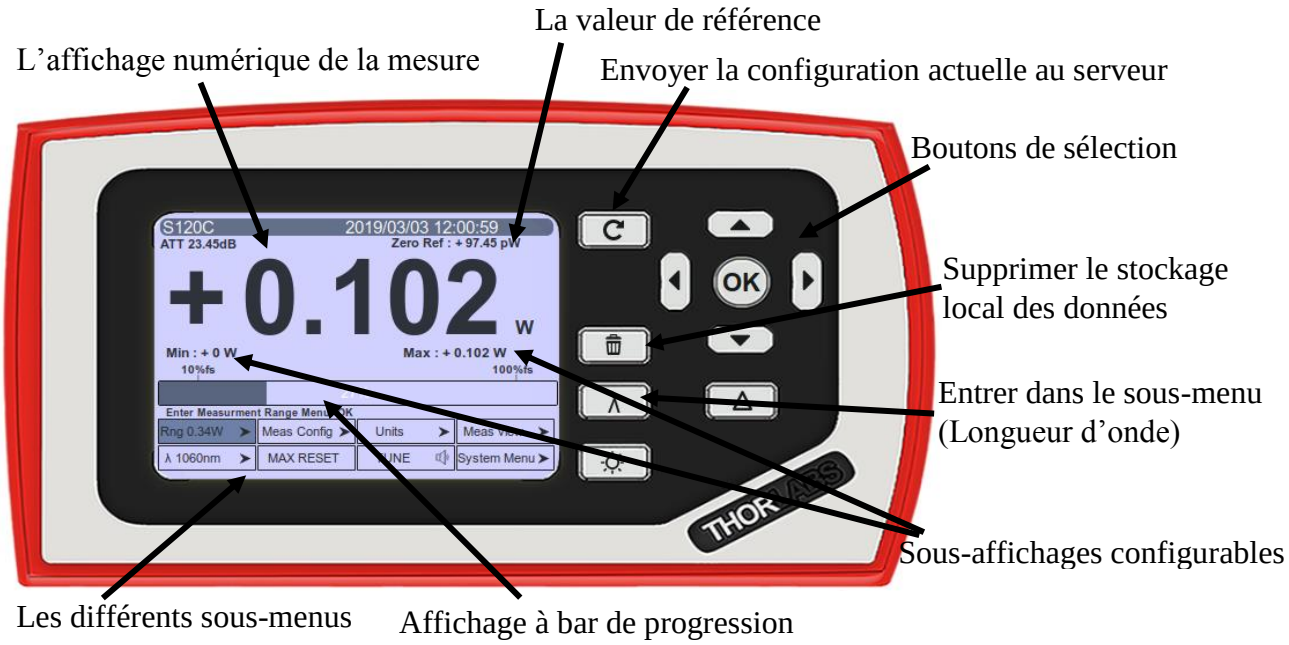

Figure III. 6 : L'aspect général du menu principal de l'instrument PM100D.

- **Les trois modes d'affichage numérique de mesure** :
  Le graphisme que nous avons réalisé propose trois modes d'affichage de résultat selon l'unité choisie (puissance en Watt *(W)*, puissance en décibel *(dBm)* et courant en ampère *(A)*. Les trois (3) figures suivantes montrent les modes d'affichage.

Figure III. 7 : Affichage de la mesure en dBm (PM100D).

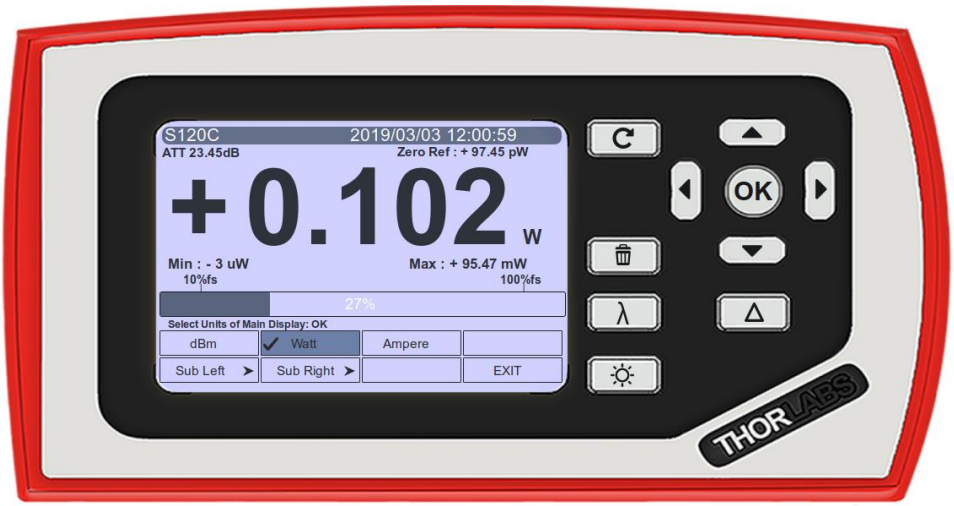

Figure III. 8 : Affichage de la mesure en Watt (PM100D).

Figure III. 9 : Affichage de la mesure en Ampère (PM100D).

- **Le type d'affichage de mesure en graphe :**

  En combinaison avec l'affichage numérique, nous avons ajouté un affichage en graphe qui introduit une nouvelle valeur à chaque actualisation, cet affichage supporte un nombre maximal de soixante (60) valeurs.

  Les valeurs mesurées sont restées stockées dans le coté client, même si l'étudiant ferme son navigateur pour y revenir ultérieurement dans le but de compléter son travail pratique, il peut trouver son ancien graphe présent avec les valeurs trouvées auparavant. La figure ci-dessous présente l'affichage de mesure sous un graphe.

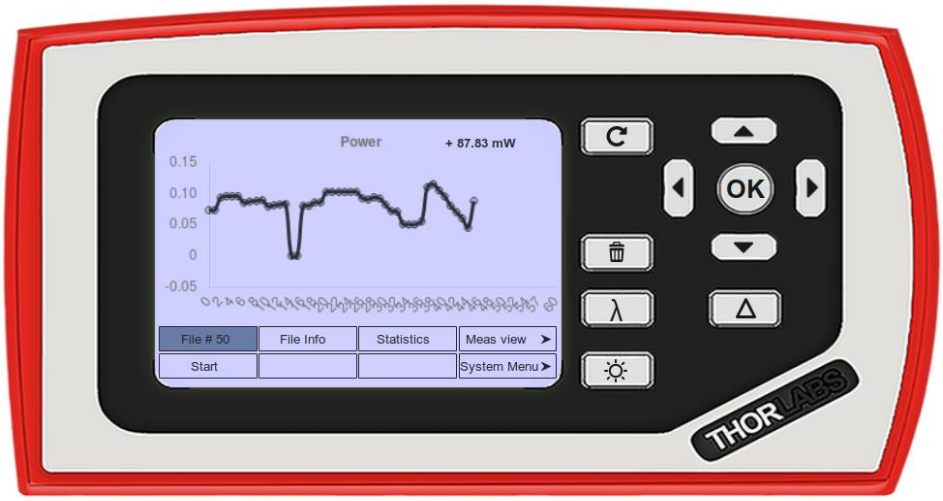

Figure III. 10 : Affichage des mesures sous forme de graphe (PM100D).

- **L'exportation des mesures sous forme de fichier (.csv) :**

  Pour ajouter plus de performances à notre système, nous avons aussi intégré une autre fonction qui consiste à lancer une routine qui collecte des mesures de puissance en fonction de temps *P(t)* pendant deux (2) minutes (cinq cents (500) valeurs en deux minutes), avec une fréquence d'échantillonnage égale à $240\ ms$.

  Cette collection sera ensuite envoyée au côté client pour être transformer en fichier *.csv* téléchargeable. L'étudiant peut ensuite utiliser ce fichier pour un traçage de graphe à l'aide d'un traceur comme (*Excel),* ou pour l'utiliser dans la rédaction du compte rendu du travail pratique.

  Pour accéder à cette fonctionnalité l'étudiant doit rentrer dans le menu « Meas view », puis choisir « Tune Graph » et ensuite sortir une nouvelle fois au menu principal et cliquer sur « Start », une fois la collection terminée un nouvel élément « Download » apparait pour télécharger le fichier dans son ordinateur personnel.

Voir dans la figure suivante un exemple de traçage d'un fichier *.csv à* l'aide de l'outil *Plotly*. [13]

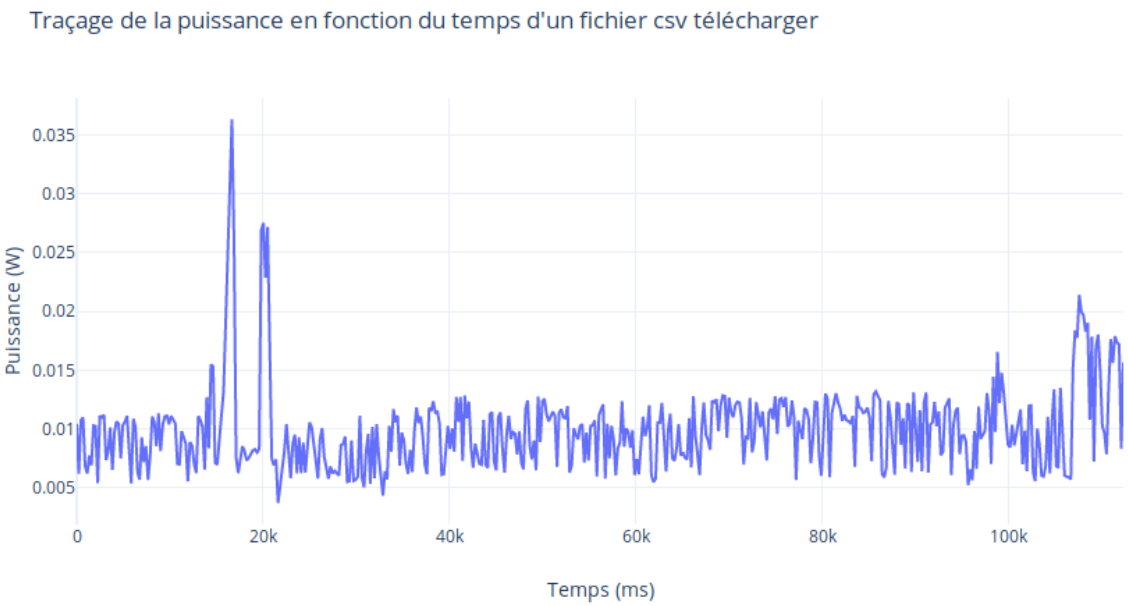

Figure III. 11 : Traçage du graphe d'un fichier .csv

Remarque : Dans les deux minutes de collection des données, l'instrument reste en état d'utilisation (accès bloqué), donc aucun autre client ne peut accéder à cet appareil dans cette période.

Il existe aussi d'autre options d'affichage comme :

- Le graphe progressif à bar dans le menu principal, qui cherche la relation entre la valeur mesure actuelle et les deux extrémités (maximum, minimum), et il l'affiche sous forme de graphe progressif pour aider l'étudiant à repérer ces résultats.
- Les sous-affichages droit et gauche qui acceptent plusieurs paramètres à être affichés (fréquence, température, surface, valeur maximal mesurée, valeur minimal, rapport max/min…etc.). Les deux figures suivantes présentent les deux sous-menus d'affichage.

Figure III. 12 : Le menu de choix du sous-affichage côté gauche.

Figure III. 13 : Le menu de choix du sous-affichage côté droit.

III.3.1.1.2 La modification des paramètres de mesures sur le PM100D

Dans cette partie nous allons présenter quelques fonctions de contrôle dédiées au mesureur PM100D.

- **Le choix de la plage de mesure :** cette fonction permet de changer la plage de mesure de l'instrument, elle accepte le mode automatique ou le mode manuel à l'aide d'un menu de six valeurs qui varient selon d'autres paramètres comme l'atténuation, la longueur d'onde…etc.

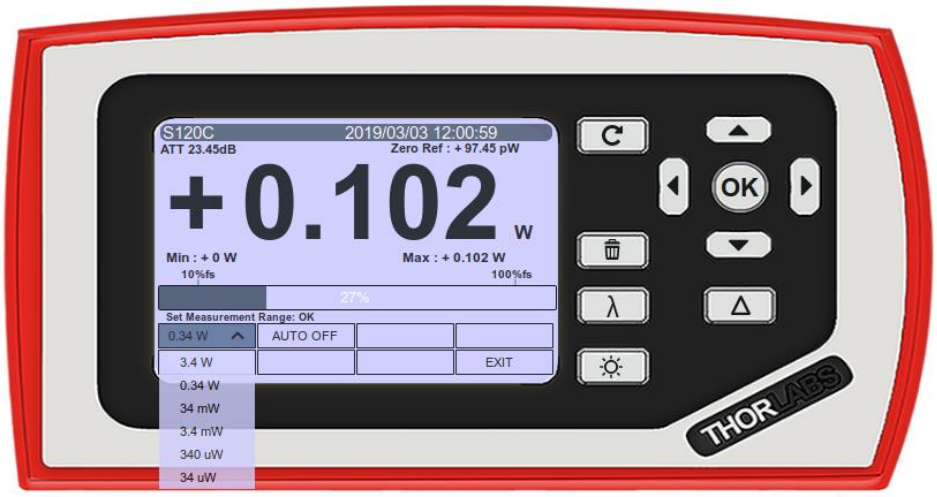

Figure III. 14 : Le menu de choix de la plage de mesure (PM100D).

- **Le choix de la longueur d'onde :**

  Cette fonction modifie la longueur d'onde**,** elle fonctionne soit par un double clic sur la valeur souhaitée par les choix présents ou par l'insérassions d'une valeur à l'aide d'un menu qui apparait en maintenant le bouton *« OK »* appuyé. La figure suivante montre le menu du choix de la longueur d'onde.

Figure III. 15 : Le menu de choix de la longueur d'onde (PM100D).

- **Le choix de l'atténuation et le diamètre du faisceau lumineux :**

  Ces deux paramètres sont configurés à l'aide d'un menu contextuel qui apparait lorsque l'étudiant clique sur le bon bouton. La figure ci-dessous présente la méthode de modification du diamètre du faisceau lumineux.

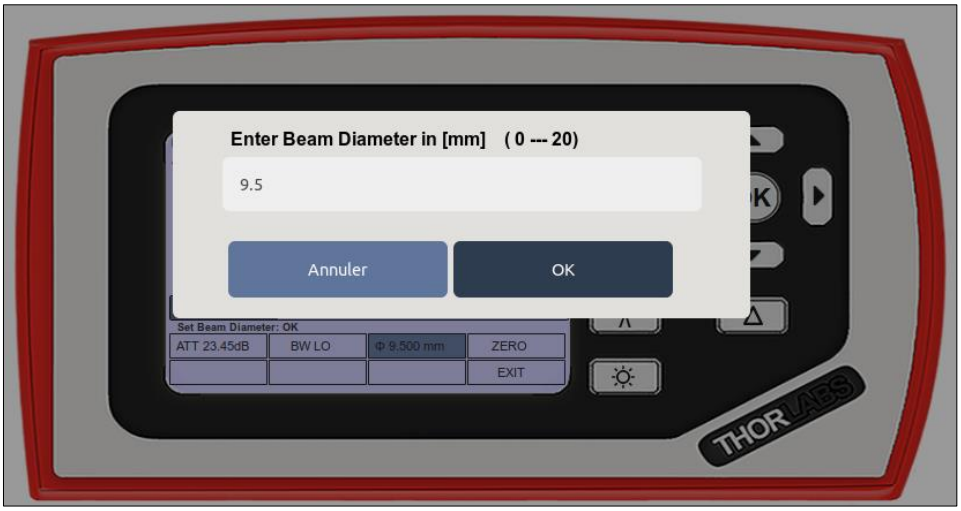

Figure III. 16 : Le menu de choix du diamètre du faisceau lumineux (PM100D).

- **La modification de d'autres paramètres :**

  Il existe aussi un type de paramètre ou fonction qui n'accepte que deux valeur *(true, false)*, comme le bande passante *(BW)* qui accepte deux états *(Low, High)*. Donc ces paramètres peuvent être modifiés juste par un double clic.

III.3.1.2 L'interfaçage du contrôleur PRO800

L'interfaçage de l'instrument *PRO800* est réalisé par la même méthode comme le *PM100D*, est aussi inspiré de l'instrument lui-même pour avoir une impression positive de la part des étudiants (clients).

Dans ce cas, l'étudiant effectue des commandes à distance via internet qui seront appliqués sur l'appareil comme la commutation de la diode laser ON et OFF, les contrôles de courant de la diode…etc.

Tout comme l'autre instrument, les clients communiquent avec l'instrument via des requêtes *AJAX* sous le protocole *http*, par l'envoie d'un message sous forme de donnée *JSON*. Ces données sont collectées par le code suivant :

```
var Data_to_send = {          ch1_STATE : CH1_STATE,
                              ch1_TS : CH1_TS,
                              ch1_TWIN : CH1_TWIN,
                              ch1_PSH : CH1_PSH,
                              ch1_ISH : CH1_ISH,
                              ch1_DSH : CH1_DSH,
                              ch1_ISHARE : CH1_ISHARE,
                              ch1_IMAX : CH1_IMAX,
                              ch1_SENSOR : CH1_SENSOR,
                               ch1_RS : CH1_RS,
                              ch1_RWIN : CH1_RWIN,
                              ch1_R0 : CH1_R0,
                              ch1_B : CH1_B,
                              ch1_T0 : CH1_T0,
                              ch1_C1 : CH1_C1,
                              ch1_C2 : CH1_C2,
                              ch1_C3 : CH1_C3

                              ch2_STATE : CH2_STATE,
                              ch2_IM : CH2_IM,
                              ch2_ILD : CH2_ILD,
                              ch2_IMAX : CH2_IMAX,
                              ch2_MLIM : CH2_MLIM,
                              ch2_PLD : CH2_PLD,
                              ch2_PLIM : CH2_PLIM,
                              ch2_CAL : CH2_CAL,
                              ch2_MODE : CH2_MODE,
                              ch2_LDPOL : CH2_LDPOL,
                              ch2_PDPOL : CH2_PDPOL,
                              ch2_UBIAS : CH2_UBIAS,
                              ch2_TWIN  : CH2_TWI
                             };
```

Figure III. 17 : Les paramètres communiqués au PRO800 chaque requête.

Le serveur exécute ces instructions et revoie une donnée *JSON* avec les mêmes paramètres que le client, il crée cette donnée à l'aide d'un dictionnaire *Python*.

III.3.1.2.1 La présentation des différents paramètres du PRO800

Dans cette partie nous allons décrire comment notre interface graphique présente les différents paramètres liés à l'appareil du contrôle *PRO800.*

- **Le menu principal du PRO800 :**

  Le menu principal contient des informations sur l'état des modules, les mesures de paramètre comme la température pour le module TED8020, le courant et la puissance du module LDC8002, plus de boutons de contrôle. Voici ci-dessous une figure qui présente le menu principal.

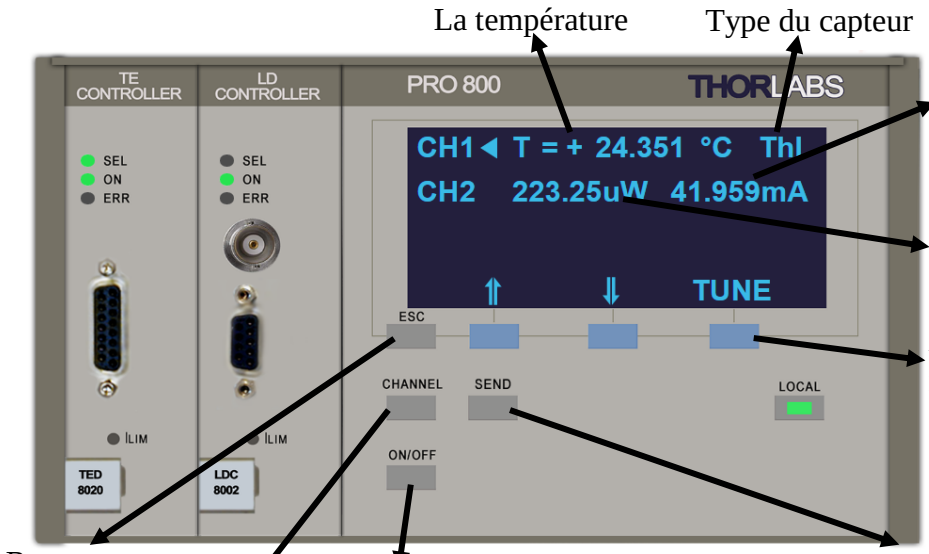

Figure III. 18 : L'aspect général du menu principal de l'instrument PRO800.

- **Le menu principal du module TED8020 :**

  Le menu de ce module contient vingt-trois (23) éléments, rongés dans une liste. Cette liste est affichée par un nombre de trois (3) éléments à la fois, il est donc nécessaire d'utiliser les boutons (haut, bas) pour passer d'un élément à un autre. Voici ci-après quelques figures qui présentent ce module.

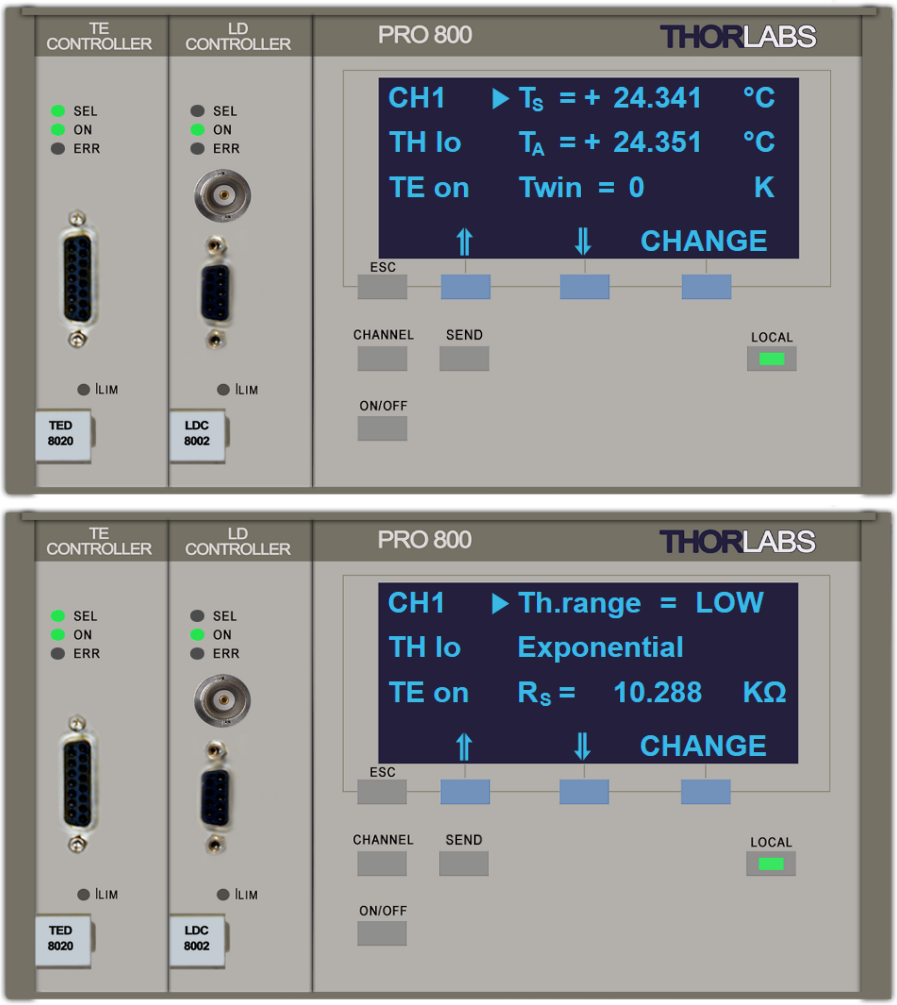

Figure III. 19 : Le menu du module TED8020.

- **Le menu principal du module LDC8002 :**

  Le menu de ce module contient quinze (15) éléments, rongés dans une liste. Cette liste est affichée par le même principe que le module *TED8020*. Voici ci-après quelques figures qui présentent ce module.

Figure III. 20 : Le menu du module LDC8002.

III.3.1.2.2 La modification des paramètres de l'instrument

Dans cette partie nous allons procéder aux méthodes suivaient pour changer les différents paramètres de l'instrument.

Nous observons la présence de deux types de paramètres, ceux qui acceptent une entrée sous forme numérique et ceux qui commutent entre deux valeurs possibles. Pour chaque type nous avons créé une méthode de changement.

- **Changement d'un paramètre qui accepte une entrée numérique :**

L'étudiant peut changer et introduire des valeurs aux différents paramètres comme le courant de la diode, la limite logicielle, la température…etc. Pour le faire l'étudiant doit juste placer le curseur sur le paramètre désiré et ensuite cliquer sur *« CHANGE »* ou un double clic pour ouvrir une entrée de texte où l'étudiant peut écrire, et en fin cliquer sur *« Entrer »* ou sur le bouton *« ESC »*. Voici ci-après une figure qui montre comment ça fonctionne.

Figure III. 21 : La méthode de modification de paramètre numérique.

- **Changement de paramètre qui commute entre deux valeurs :**

Comme il existe des paramètres numériques, il existe aussi d'autres qui acceptent uniquement des valeurs prédéfinies par le constructeur, donc nous avons utilisé une méthode adaptée pour que les étudiants puissent accéder et modifier ces paramètres. Les deux figures ci-dessous montrent cette méthode.

Figure III. 22 : Le changement du LDC8002 au mode puissance constante.

Figure III. 23 : Le changement du LDC8002 au mode puissance constante.

III.3.2 Autre fonctionnalité du serveur

Le serveur est un élément essentiel dans une application *Web,* c'est la partie qui assure la communication entre les clients et les objets contrôlés, en d'autres termes c'est une passerelle qui relie deux extrémités différentes.

Tous les serveurs web du monde intègrent une fonction de sauvegarde d'historique des évènements produits sur leurs terrains. De la même manière nous avons ajouté cette fonctionnalité à notre système du laboratoire distant qui consiste à :

- Sauvegarder toutes les requêtes produites par les clients en spécifiant, l'identifiant de l'étudiant, son adresse IP, le temps et la date du requête et toute la configuration des paramètres modifiables de chaque instrument.
- Stocker ces données sous un fichier *.csv* pour faciliter sa lecture.
- Sécuriser l'accès à ce fichier.

Cette fonctionnalité aide à apporter plusieurs avantages pour notre laboratoire distant :

- Faciliter à l'enseignant la correction et l'évaluation des étudiants.
- Assurer un travail unique pour chaque étudiant à l'aide de la traçabilité par adresse IP.
- Utiliser ces historiques pour effectuer des études de statistique, par exemple une étude sur l'utilisation de l'outil distant du laboratoire.

La création et la manipulation des historiques sont réalisées à l'aide d'un programme écrit en *Python* qui crée un fichier *.csv* et ajoute une ligne de paramètre à chaque réception de requêtes.

La figure suivante contient la fonction utilisée pour la création de l'historique des évènements.

```
def Create_CSV(request,response):

  REQ = json.loads(request.data)
  if not os.path.exists("/home/omar/Desktop/PM100D_Server/Student.csv"):
    CSV_File =open("Student.csv", "a+")
    CSV_File.write("Time,IP Address,Wavelength(Req),"
    +"Attenuation(Req),BeamDiameter(Req),"
    +"Bandwidth(HI/LO)(Req),Current Range(Req),"
    +"Zeroing(1/0)(Req),AUTO Range(ON/OFF)(Req),"
    +"Power(Res),Current(Res),Frequency(Res),"
    +"Temperature(Res),Power Density(Res),"
    +"Power Range(Res),Current Range(Res),"
    +"Zero Value(Res),AUTO Range(Res),"
    +"Bandwidth(HI/LO)(Res)"+"\r\n")

  CSV_File =open("Student.csv", "a+")

  CSV_File.write( time.ctime() + ","+request.remote_addr
  +","+str(REQ['wavelength'])+','+str(REQ['attenuation'])
  +","+str(REQ['beamdiameter'])+","+str(REQ['bandwidth_HI_LO'])+
  +","+str(REQ['current_range_send'])+","+str(REQ['zeronig'])
  +","+str(REQ['auto_range'])+","+ str(response['Power_W'])
  +","+str(response['Current_A'])+","+str(response['Frequency'])
  +","+str(response['Temperature'])+","+str(response['Power_Dens'])
  +","+str(response['Power_Range'])+","+str(response['Current_Range'])
  +","+str(response['Zero_Value'])+","+str(response['Auto_Range'])
  +","+str(response['BandWidth_HI_LO'])  +"\r\n")
  CSV_File.close()
```

Figure III. 24 : la fonction utilisée pour la création de l'historique des évènements.

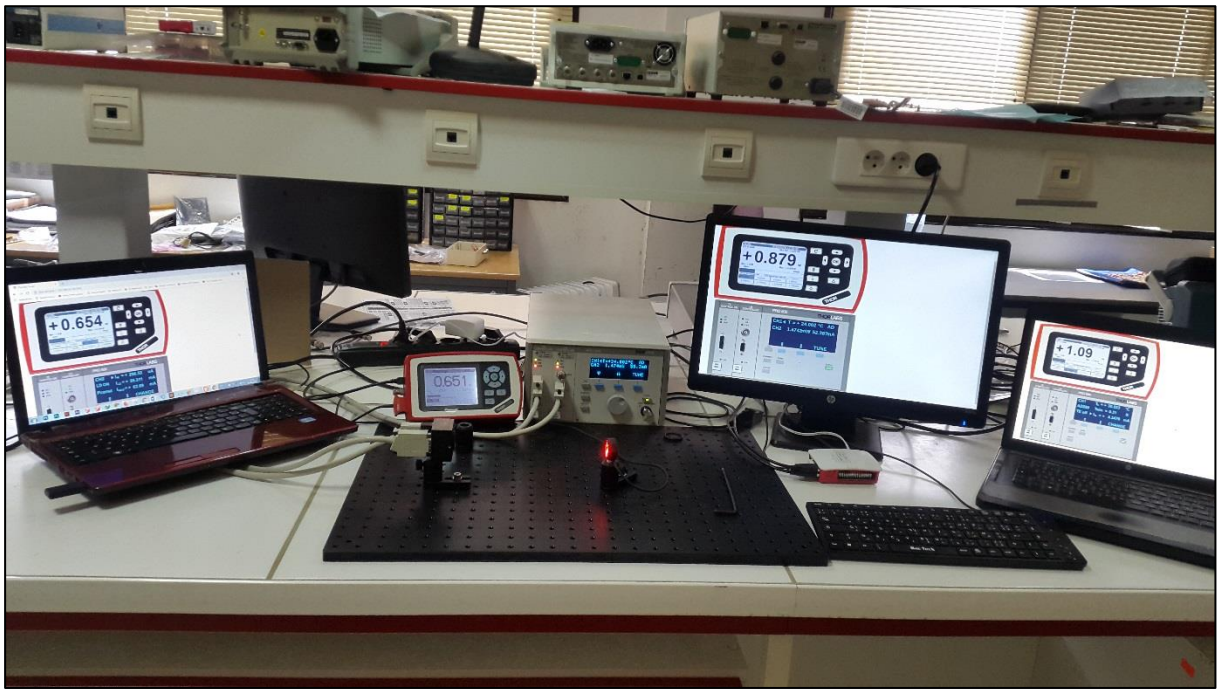

Figure III. 25 : Le laboratoire distant en réalité.

III.4 Conclusion

Dans ce chapitre nous avons entamé l'implémentation logiciel/matériel et par suite nous avons expliqué le principe de fonctionnement du laboratoire distant, ainsi que toutes les fonctionnalités proposées par les systèmes conçus.

Selon les tests effectués au cours de la réalisation du système, nous pouvons dire que ce laboratoire distant répond exactement à nos exigences en respectant les termes de rapidité, la facilité et la flexibilité d'utilisation, la disponibilité et la maintenabilité.

Conclusion générale

Dans ce mémoire et à travers ce projet de fin d'étude, plusieurs avantages du laboratoire distant sont présentés et interprétés réellement par la réalisation d'un laboratoire entièrement contrôlé à distance via internet.

Les techniques d'information, de communication et de l'électronique se propagent d'une façon incorporée vers le monde académique, implique l'apparition des nouvelles solutions, afin de sortir de la domination des laboratoires présentiels qui restent toujours incomplets sur des échelles très importantes comme la disponibilité, l'accessibilité, la suffisance en termes de disponibilité des places, le coût de la réalisation et bien d'autres problèmes.

Grâce aux études profondes sur les laboratoires distants, et à l'aide des ressources *open source* qui existent actuellement sur internet, nous avons pu réaliser un système concourant qui élimine d'une façon effective les problèmes majeurs des laboratoires traditionnels. En spécifiant que ce système est basé sur une architecture matérielle *open source* et réutilisable, en combinaison avec une architecture logicielle adaptable, puissante, flexible et facile à implémenter.

D'ailleurs, ce système propose à l'étudiant d'effectuer ces travaux pratiques toute en restant à la maison via un ordinateur et une connexion internet, avec la possibilité de manipuler des appareils de mesures et de contrôles 24h/24 et 7jr/7. Il est aussi probable d'effectuer un nombre infini de manipulation contrairement au laboratoire traditionnel qui est limité dans un lapse de temps très court. De plus le système peut intégrer des fonctions de correction et de notation automatique des travaux pratiques.

Ce système est conçu sur des mesures afin d'avoir un coût de développement faible par rapport à son utilisation, qui consiste à partager des instruments coûteux sur un nombre important d'étudiants en même temps. Cette pertinence peut apporter plusieurs avantages en termes économique et éducatif.

L'architecture du système est inspirée de la réalité pour assurer une meilleure ressemblance avec un laboratoire présentiel, tout en gardant une flexibilité et une simplicité d'utilisation de cet outil, à l'aide d'une interface graphique presque réelle.

Comme projection dans le futur, ce système est encore en exposition pour plus de développement en essayant de regrouper un ensemble complet d'instruments à contrôler comme l'analyseur de spectre et le contrôleur de moteur à courant continu de Thorlabs, pour pouvoir réaliser des travaux pratiques complets avec un bon rendement.

Références bibliographiques

[1] Coût d'achat du logiciel « LabVIEW Professional Edition » par ans, à partir de : https://www.ni.com/en-lb/shop/labview/labview-details.html

[2] Caractéristique du Raspberry Pi 3, à partir de : https://www.generationrobots.com/fr/402366-raspberry-pi-3-modele-b.html , page consulté le 10/06/2019.

[3] Caractéristiques du mesureur PM100D, depuis le catalogue de Thorlabs, le 25/03/2019, lien du catalogue : https://www.thorlabs.com/images/Catalog/V21/V21_7_LightAnalysis.pdf .

[4] Caractéristiques du Contrôleur PRO800, depuis le catalogue de Thorlabs, le 31/03/2019, lien du catalogue : https://www.thorlabs.com/catalogpages/V21/1161.pdf .

[5] Représentation de la face avant du module LDC8002, depuis le manuel du module V3.3, date de sortie : https://www.thorlabs.com/_sd.cfm?fileName=7154-D02.pdf&partNumber=LDC8002

[6] Mode de fonctionnement du module LDC8002, depuis le manuel du module V3.3, date de sortie : 08/09/2016, lien : https://www.thorlabs.com/_sd.cfm?fileName=7154-D02.pdf&partNumber=LDC8002 .

[7] Représentation de la face avant du module TED8020, depuis le manuel du module, V3.2, date de sortie : 08/09/2016, lien : https://www.thorlabs.com/_sd.cfm?fileName=7158-d02.pdf&partNumber=TED8020 .

[8] L'effet Peltier est un effet thermoélectrique consistant en un phénomène physique de déplacement de chaleur en présence d'un courant électrique. Source : https://fr.wikipedia.org/wiki/Effet_Peltier .

[9] Principe de fonctionnement du module TED8020, depuis le manuel du module, V3.2, date de sortie : 08/09/2016, lien : https://www.thorlabs.com/_sd.cfm?fileName=7158-d02.pdf&partNumber=TED8020 .

[10] Configuration des lignes de communication PRO800/PC, depuis le manuel de l'instrument PRO800 V3.6, Date de sortie 25/07/2017, lien : https://www.thorlabs.com/_sd.cfm?fileName=7146-D02.pdf&partNumber=PRO800 .

[11] La présentation du concept de la technique *AJAX*, à partir de : http://javascript-coder.com/tutorials/re-introduction-to-ajax.phtml ,Page consulté le : 20/04/2019.

[12] Scripte inspiré de la fonction discover_stages () sous le lien : https://github.com/UniNE-CHYN/thorpy/blob/master/thorpy/comm/discovery.py .

[13] Une technique qui fournit des outils de graphique, d'analyse et de statistique en ligne, site web : https://plot.ly/create/#/ .

Annexes

Annexe 1 : Code source de l'application (Coté client et serveur)

Le code source utilisé pour le développement du système est trop volumineux, pour cela vous trouverez la totalité du code dans notre répertoire GitHub sous le nom : « Source-Code-AZZOUZI-CHAREF-AISSA ».

Consulter le lien : https://github.com/OmarAbde/Source-Code-AZZOUZI-CHAREF-AISSA