UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

THÈSE DOCTORALE

Supervision et instrumentation à distance, libre et interopérable, du capteur à l'utilisateur.

Auteur :
Camille LAVAYSSIÈRE

Présidente :
Pr. Nadine Couture.

Rapporteurs :
Pr. Eric Duviella
Dr. Ousmane Sow.

Superviseurs :
Pr. Franck LUTHON
Dr. Benoît LARROQUE

Cette dissertation est soumise pour un
Doctorat en informatique

au

Laboratoire Informatique de l'Université de Pau et des Pays de l'Adour
Collège Sciences et Technologies pour l'Énergie et l'Environnement

27 janvier 2023

Remerciements

Je tiens à remercier mes encadrants de thèse, Franck Luthon et Benoît Larroque qui m'ont proposé ce sujet, nous permettant de travailler ensemble, et pour m'avoir laissé libre dans ma recherche tout en répondant à mes questions.

Je tiens aussi à remercier les collègues du département Génie Industriel et Maintenance de l'IUT de Bayonne qui m'ont accueilli dans leur locaux et avec qui j'ai beaucoup échangé sur divers sujets.

Je remercie l'ensemble des étudiants, chercheurs et ingénieurs des laboratoires LIUPPA et SIAME avec qui j'ai pu travailler et qui m'ont permis, à travers leurs questions et besoins paraissant distincts, de développer ma réflexion autour d'un système unique.

Je remercie Martin Schröder, qui est à l'origine du logiciel que je développe, pour m'avoir beaucoup aider à le prendre en main et pour nous avoir reçus à Berlin afin d'échanger et de mieux se connaître.

Je remercie les rapporteurs de ma thèse pour avoir accepté de lire et de commenter mes travaux ainsi que mes examinateurs pour leur présence lors de ma soutenance de thèse.

Je remercie aussi les différentes personnes qui ont fait évoluer mon esprit critique sur le rapport du logiciel libre avec le travail.

Table des matières

Remerciements i

Table des matières ii

Table des figures v

Liste des tableaux viii

Introduction générale 1

1 Supervision et instrumentation 5
- 1.1 Systèmes physiques . . . 6
  - 1.1.1 Échange d'information . . . 7
  - 1.1.2 Télémesure et acquisition . . . 7
  - 1.1.3 Télécontrôle et manipulation . . . 9
  - 1.1.4 Stockage de données . . . 9
  - 1.1.5 Automatisation . . . 10
  - 1.1.6 Accès distant . . . 10
  - 1.1.7 Sécurité . . . 11
- 1.2 Problématiques abordées . . . 11
  - 1.2.1 Applications variées . . . 11
  - 1.2.2 Information trouvable, accessible, interopérable et réutilisable . 11
  - 1.2.3 Toile sémantique . . . 14
  - 1.2.4 Catégories de logiciels . . . 14
    - 1.2.4.1 Logiciel libre . . . 14
    - 1.2.4.2 Logiciel *open source* . . . 15
    - 1.2.4.3 *Copyleft* . . . 15
    - 1.2.4.4 Recommandations pour les logiciels . . . 16
  - 1.2.5 Interopérabilité . . . 16
  - 1.2.6 Décentralisation . . . 17
  - 1.2.7 Sécurité . . . 18
- 1.3 Propositions . . . 18
  - 1.3.1 FAIR des processus . . . 18
  - 1.3.2 Distribution d'identifiants pérennes . . . 20
  - 1.3.3 Différentes interopérabilités . . . 21
    - 1.3.3.1 Matériel hôte et système d'exploitation . . . 22
    - 1.3.3.2 Communication avec d'autres matériels . . . 22
    - 1.3.3.3 Utilisation de librairies . . . 22
    - 1.3.3.4 Différents profils d'utilisateurs . . . 23
      - 1.3.3.4.1 Configurer . . . 23
      - 1.3.3.4.2 Utiliser . . . 24
      - 1.3.3.4.3 Visualiser . . . 24
      - 1.3.3.4.4 Développer . . . 24

1.4 Bilan . . . 26

2 État de l'art des applications **27**
2.1 Secteur de l'industrie . . . 28
2.1.1 Tango Controls . . . 29
2.1.2 Scada-LTS . . . 29
2.1.3 RapidSCADA . . . 30
2.1.4 Bilan . . . 31
2.2 Secteur de la recherche . . . 31
2.2.1 National Instruments . . . 31
2.2.2 PyMoDAQ . . . 31
2.2.3 PyMeasure . . . 33
2.2.4 Contrôle d'expérimentation au National Synchrotron Light Source-II . . . 34
2.2.5 Yaq . . . 35
2.2.6 Bilan . . . 35
2.3 Secteur de l'enseignement . . . 35
2.3.1 LabsLand : une entreprise proposant d'accéder à des laboratoires distants . . . 36
2.3.2 VISIR : Virtual Instruments Systems in Reality . . . 37
2.3.3 ISES Remote Lab . . . 39
2.3.4 UNILabs . . . 39
2.3.5 RexLab . . . 40
2.3.6 VirtualRemoteLab . . . 41
2.3.7 GOLDi : The Grid of Online Laboratory Devices Ilmenau . . . 41
2.3.8 Expériences à la FEUP . . . 42
2.3.9 e-lab IST . . . 43
2.3.10 Stanford iLABS . . . 43
2.3.11 iLAB et LabShare . . . 43
2.3.12 FarLabs . . . 44
2.4 Besoins identifiés . . . 45

3 Contribution **47**
3.1 Différentes abstractions . . . 48
3.1.1 Abstraction de la connexion . . . 48
3.1.2 Abstraction de l'expérimentation . . . 48
3.1.2.1 Abstraction du transport . . . 48
3.1.2.2 Abstraction de la charge utile . . . 49
3.1.3 Abstraction de la synchronisation . . . 49
3.1.4 Abstraction des erreurs . . . 49
3.2 Architecture de PyScada . . . 50
3.2.1 Application Django . . . 51
3.2.2 Cœur de PyScada . . . 52
3.2.3 Couche d'abstraction matérielle . . . 54
3.2.3.1 Couche de la plateforme multi-matérielle . . . 55
3.2.3.2 Couche spécifique au matériel . . . 55
3.2.4 Interfaces web . . . 56
3.2.4.1 Interface d'administration . . . 56
3.2.4.2 IHM utilisateur . . . 59
3.2.5 Évènements et alertes . . . 61
3.2.6 Scripts et algorithmes . . . 62

3.2.7 Approche modulaire par *plugin* . . . 64
3.2.8 Faible besoin matériel . . . 65
3.2.9 Évaluation face aux principes du FAIR des logiciels . . . 65
3.3 Bilan . . . 68

4 Applications **69**
4.1 Enseignement hybride . . . 69
4.1.1 Laborem Box . . . 73
4.1.2 Comportement étudiant . . . 76
4.1.3 Futures études autour de la plateforme Laborem . . . 77
4.2 Supervision d'ouvrages du littoral . . . 78
4.2.1 Première station : digue de l'Artha . . . 78
4.2.2 Seconde station : fort de Socoa . . . 82
4.2.3 Troisième station : structure mobile . . . 84
4.2.4 Vers un maillage de la côte basque . . . 85
4.2.5 Études en laboratoire . . . 85
4.3 Suivi énergétique du bâtiment . . . 86
4.3.1 Première expérimentation : IUT de Bayonne . . . 87
4.3.2 Seconde expérimentation : Domolandes . . . 87
4.3.3 Troisième expérimentation : Elis . . . 88
4.4 Bilan, validation et évaluation . . . 88

Conclusion et perspectives **90**

Publications et conférences **92**
Publications en revues internationales . . . 92
Conférences internationales . . . 92

Annexe A Figures pleines pages **93**

Bibliographie **98**

Glossaire **103**

Résumé **107**

Table des figures

1.1 Supervision et instrumentation d'un système physique. 6
1.2 Exemple d'interface de télémesure. 7
1.3 Brevet du "Electric Telemeter Transmitter" de 1893. 8
1.4 La chaîne d'acquisition d'une instrumentation à distance. 9
1.5 Catégories de logiciels définies par Chao-Kuei. 15
1.6 Schéma d'un processus de supervision. 19
1.7 Exemple d'un processus de supervision. 19
1.8 Principe de contamination de licences sous copyleft. 23
1.9 Affichage de quatre variables de deux manières différentes. 25
1.10 Diagramme de Sankey représentant des flux liant des nœuds. 25

2.1 Tango Controls requiert l'utilisation de logiciels tiers pour construire une interface. 29
2.2 Résultat de l'exemple d'interface codée en HTML et proposée par Scada-LTS. 30
2.3 Exemple d'interface d'un processus industriel avec RapidSCADA. . . 30
2.4 Logiciel de test et de mesure LabVIEW disposant de deux interfaces : programmeur et utilisateur. 32
2.5 Interface d'une installation via les modules *Move* et *Viewer* de PyMoDAQ. 32
2.6 Représentations des données selon différentes dimensions. 33
2.7 Interface de PyMeasure définie par une procédure avec des données d'entrée et de sortie. 34
2.8 Organisation des librairies de NSLS-II en trois groupes. 35
2.9 Expérience Archimedes de Labsland pour l'étude du déplacement d'un objet dans un liquide. 38
2.10 Matrice de relais de VISIR permettant de connecter des éléments selon le schéma du circuit réalisé par l'étudiant. 38
2.11 Expérience d'induction électromagnétique d'ISES. 39
2.12 Expérience de caractérisation de systèmes dynamiques multivariable d'UNILabs. 40
2.13 Version virtuelle de l'expérience à quatre réservoirs contrôlés d'UNILabs. 40
2.14 Etude des réseaux à courant alternatif chez RexLab. 41
2.15 Analyse du spectre de différentes lampes sur VirtualRemoteLab. 42
2.16 Ascenseur à 4 étages programmable sur GOLDi. 42
2.17 Lampe et ventilateur contrôlés à l'aide d'interrupteurs accessibles à la FEUP. 43
2.18 Liste des expériences accessibles chez e-lab IST nécessitant Java. 44
2.19 Exemple d'un laboratoire distant accessible via Sahara. 44
2.20 Chauffe eau solaire utilisé pour les expériences à distance chez FarLabs. 45

3.1 Exécution de requêtes synchrones et asynchrones. 49
3.2 Processus de mise en place de PyScada. 50
3.3 Architecture Django Modèle-Vue-Gabarit. 51

3.4 Schéma de la structure de PyScada dans l'environnement Django. . . . 52
3.5 Fonctionnement asynchrone des modules de PyScada communiquant via la base de données. . . . 53
3.6 Contrôle du rafraîchissement des valeurs des variables dans le coin supérieur droit de l'IHM utilisateur. . . . 54
3.7 Architecture HAL de PyScada. . . . 55
3.8 Architecture de l'interface d'administration incluant certains des principaux blocs de PyScada et une liste non exhaustive de leur modèle. . . . 56
3.9 Exemple de l'interface d'administration de PyScada incluant le bloc d'authentification de Django, les blocs *Core* et IHM de PyScada et les blocs des *plugins Export*, *Scripting* et *WebService* de PyScada. . . . 57
3.10 Exemple de configuration (CHPL et HSL) d'un instrument utilisant le protocole OPC-UA dans PyScada. . . . 58
3.11 Exemple d'interface d'administration de PyScada avec des droits limités. 59
3.12 Schéma de l'IHM utilisateur de PyScada. . . . 60
3.13 Exemple d'interface utilisateur de PyScada. . . . 61
3.14 Exemple d'interface utilisateur de PyScada pour un utilisateur ne disposant pas des droits pour voir tous les éléments. . . . 62
3.15 Gestion des permissions d'affichage de certains éléments graphiques pour un groupe d'utilisateurs. Voir affichage en pleine page en annexe A. 63
3.16 Flux d'un *script* dans PyScada. . . . 63
3.17 Flux du *script* permettant de réaliser des diagrammes de Bode. . . . 64

4.1 Vue d'ensemble du banc d'essai Laborem. . . . 70
4.2 Accès aux interfaces enseignant (à gauche) et étudiant (à droite) de Laborem. . . . 71
4.3 L'interface étudiant simple à utiliser. . . . 72
4.4 Vue d'ensemble de l'architecture du système Laborem. . . . 73
4.5 Les composants de la Laborem Box. . . . 74
4.6 Vue de face de la conception 3D de la carte mère sans les composants soudés. . . . 75
4.7 *Plugs* et câbles BNC et bananes connectés à la Laborem Box. . . . 75
4.8 Scénario en cinq étapes en ligne dans Moodle. . . . 76
4.9 Baie de Saint-Jean-de-Luz, France. . . . 79
4.10 Déplacement des blocs de béton protégeant la digue de l'Artha. . . . 80
4.11 Installation technique de la digue de l'Artha. . . . 81
4.12 Interface utilisateur de PyScada sur la station de la digue de l'Artha. . . . 81
4.13 Image produite par Plotly et Datashader permettant de visualiser 18 millions de points sans perte d'information. . . . 82
4.14 Vue de la digue de l'Artha depuis la caméra installée sur le fort de Socoa. 82
4.15 Impact d'une vague filmée pendant une tempête depuis la station du fort de Socoa. . . . 83
4.16 Représentation temporelle des données de deux capteurs de l'impact de la figure 4.15. . . . 83
4.17 Vues de caméras situées au fort de Socoa et utilisées par le logiciel Sirena. 84
4.18 Structure mobile pour l'étude des phénomènes de submersion côtiers. 84
4.19 Étude en laboratoire des interactions entre vagues et structures. . . . 85
4.20 Évolution temporelle de la mesure de l'impact d'un lâché d'eau, enregistré par les quatre capteurs du banc de laboratoire. . . . 86
4.21 Interface de PyScada montrant l'intégration d'éléments graphiques dans une maquette numérique 3D. . . . 88

A.1 Gestion des permissions d'affichage de certains éléments graphiques pour un groupe d'utilisateurs. 94
A.2 Interface utilisateur de PyScada sur la station de la digue de l'Artha. . 95
A.3 Image produite par Plotly et Datashader permettant de visualiser 18 millions de points sans perte d'information. 96
A.4 Représentation temporelle des données de deux capteurs de l'impact de la figure 4.15. 96
A.5 Évolution temporelle de la mesure de l'impact d'un lâché d'eau, enregistré par les quatre capteurs du banc de laboratoire. 97

Liste des tableaux

1.1 Principes des données FAIR énoncés par Wilkinson. 12
1.2 Principes des logiciels FAIR énoncés par Lamprecht. 13

2.1 Comparaison des développements des laboratoires distants listés par Garcia-Zubia. 37
2.2 Comparaison de différentes solutions avec les critères définis en section 2.4. 46

3.1 Évaluation de PyScada pour les principes des logiciels FAIR énoncés par Lamprecht. 65
3.2 Comparaison de PyScada avant et après la thèse avec les critères définis en section 2.4. 68

Introduction générale

Cette thèse concerne les méthodes et outils destinés à l'instrumentation et à la supervision informatique de systèmes physiques distants. Ils sont illustrés ici à l'aide de procédés automatisés dans les domaines de l'éducation, l'énergie et l'environnement et dans les secteurs de l'enseignement, de la recherche ou de l'industrie. Elle s'est déroulée d'octobre 2019 à septembre 2022 au sein de deux laboratoires de l'Université de Pau et des Pays de l'Adour (UPPA) :

— le Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour (LIUPPA, EA3000), dans l'équipe Traitements des Informations pour l'adaptation de l'Interaction au contexte et à l'utilisateur (T2I),
— le laboratoire des Sciences pour l'Ingénieur Appliquées à la Mécanique et au génie Électrique (SIAME, EA4581), avec les équipes Procédés Haute Tension (PHT) et Interaction Vagues / Structures (IVS).

Génèse des travaux

Pour comprendre les propositions énoncées dans cette thèse, revenons brièvement sur la genèse de son sujet ainsi que sur les différentes thématiques abordées dont certains applicatifs sont détaillés au chapitre 4.

D'une part, j'ai travaillé comme ingénieur contractuel sur l'instrumentation à distance dans le cadre du projet Laborem porté par Franck Luthon et Benoît Larroque. Ce projet a pour but de fournir aux étudiants du Diplôme Universitaire de Technologie (DUT) du parcours Génie Industriel et Maintenance (GIM), remplacé depuis 2021 par le Bachelor Universitaire de Technologie (BUT), un accès à des instruments permettant de réaliser à distance des travaux pratiques en électronique. Afin de faciliter la diffusion de cette plate-forme, d'en réduire les coûts et d'assurer sa pérennité, j'ai cherché à remplacer les logiciels propriétaires utilisés par des logiciels libres. Cette approche m'a amené à considérer la problématique des laboratoires distants pour l'éducation comme des systèmes de supervision, de contrôle et d'acquisition en temps réel (SCADA de l'anglais *Supervisory Control And Data Acquisition*).

D'autre part, j'ai été sollicité pour des besoins d'instrumentation embarquée et de supervision d'expériences de recherche par le laboratoire SIAME dont une des équipes de recherche (IVS) travaille sur l'interaction entre les vagues et les structures côtières (digues, plages...). Cette sollicitation m'a permis de définir les problématiques et les besoins génériques de systèmes de supervision et d'instrumentation à distance. Pour définir un système agnostique du point de vue de l'application, c'est-à-dire qui est capable de fonctionner quel que soit le domaine d'utilisation, nous avons dû repenser et identifier les problématiques et besoins génériques pour chaque cas d'application.

Enfin des chercheurs et des industriels partenaires de notre université désiraient avoir une solution générique permettant de suivre et d'analyser la consommation énergétique de bâtiments, de l'intégrer dans une maquette en 3D, appelée jumeau numérique, depuis la création et durant toute la vie du bâtiment. L'approche agnostique

développée durant ma thèse nous a permis de répondre à ce besoin en développant des modules pour les problématiques spécifiques à ce domaine.

Problématique

Depuis la fin du 20ème siècle la présence de l'informatique dans l'enseignement supérieur, plus particulièrement dans les sciences et techniques de l'ingénieur, la recherche et l'industrie a permis aux travailleurs (étudiants, enseignants, chercheurs, techniciens et ingénieurs) d'avoir des outils d'acquisition de données (mesure, alarme, retour d'état de fonctionnement) et de paramétrage des processus (contrôle, configuration).

Le type, la qualité et le résultat des interactions humain-machine sont déterminés entre autres par le matériel et le logiciel utilisés. Les logiciels ont été dans un premier temps conçus pour un type d'application, ils sont parfois dédiés à un domaine particulier et ne peuvent représenter qu'une partie d'une chaîne d'acquisition, le traitement des données étant parfois réalisé ultérieurement par l'homme. Un utilisateur achetant une machine ou un outil contrôlable par ordinateur doit utiliser le logiciel fourni par le constructeur, utiliser un autre logiciel disponible ou développer son propre logiciel, par exemple à l'aide de la rétro-ingénierie (encadrée en France par la loi, voir LÉGIFRANCE [1]).

Afin de définir les lignes directrices de logiciels génériques dans le domaine de la supervision, de l'instrumentation et de l'expérimentation, il est nécessaire de définir de manière commune et en partie abstraite les besoins d'un utilisateur. Cette orientation apparaît d'autant plus nécessaire lorsque des travailleurs issus de domaines différents collaborent sur un projet commun. Certains applicatifs apportent des contraintes supplémentaires auxquelles nous tâcherons de répondre, comme par exemple de traiter de grandes quantités de données, d'avoir un faible temps de réponse ou la nécessité d'avoir un système autonome en énergie ou en prise de décision et frugal.

Le travail présenté dans ce manuscrit se concentre sur les besoins dans l'enseignement, la recherche et l'industrie pour avoir une solution pluridisciplinaire tout en permettant de l'ouvrir à d'autres champs d'applications.

Objectifs de recherche

Pour répondre aux besoins des secteurs étudiés, je définirai les principes à suivre pour disposer d'un système de supervision, de contrôle et d'acquisition de données que chacun pourra utiliser, modifier et distribuer, et je présenterai le développement d'une solution logicielle allant dans ce sens. Les travaux de notre équipe sur le sujet ont commencé en 2011 avec le projet **Laborem** de laboratoire distant pour l'enseignement (LUTHON et al. [2]). Depuis 2017 nous étudions les impacts des vagues affectant des structures côtières comme la digue de l'Artha à Saint-Jean-de-Luz à l'aide de capteurs de pression (D'AMICO [3]). Depuis 2021 nous accompagnons différentes entreprises pour les aider à suivre et réduire leurs consommations énergétiques au sein de leurs bâtiments. Dans ma thèse, mon objectif est de présenter les lignes directrices d'un système de supervision et d'instrumentation simple et accessible au travers du développement et de la mise en application d'un logiciel dans différents domaines permettant de :

— communiquer avec des instruments,
— lire, enregistrer, traiter et visualiser un grand nombre de données,

— envoyer des consignes, définir des évènements et automatiser des tâches,
— construire des interfaces correspondant aux besoins de différents utilisateurs avec une gestion de droits d'accès.

Un autre objectif est de pouvoir exécuter le logiciel en mobilisant peu de ressources matérielles, mais aussi d'être libre de le modifier et de le distribuer.

Organisation du document

Ce document se décompose en quatre chapitres :

1. le premier chapitre définit la notion de supervision et d'instrumentation, présente les problématiques rencontrées dans ces domaines, propose des définitions,
2. le second chapitre expose un état de l'art des solutions actuelles avant d'identifier des besoins,
3. le troisième chapitre présente le logiciel utilisé et maintenu par notre équipe en définissant son architecture et ses différents concepts,
4. le dernier chapitre présente les différentes applications de notre solution dans les domaines de l'éducation, l'environnement et l'énergie au sein des secteurs de l'industrie, la recherche et l'enseignement.

Enfin, je présenterai mes conclusions ainsi que les perspectives qui s'ouvrent suite à ce travail.

Chapitre 1

Supervision et instrumentation

Sommaire

1.1 Systèmes physiques 6
1.1.1 Échange d'information 7
1.1.2 Télémesure et acquisition 7
1.1.3 Télécontrôle et manipulation 9
1.1.4 Stockage de données 9
1.1.5 Automatisation 10
1.1.6 Accès distant 10
1.1.7 Sécurité 11
1.2 Problématiques abordées 11
1.2.1 Applications variées 11
1.2.2 Information trouvable, accessible, interopérable et réutilisable 11
1.2.3 Toile sémantique 14
1.2.4 Catégories de logiciels 14
1.2.4.1 Logiciel libre 14
1.2.4.2 Logiciel *open source* 15
1.2.4.3 *Copyleft* 15
1.2.4.4 Recommandations pour les logiciels 16
1.2.5 Interopérabilité 16
1.2.6 Décentralisation 17
1.2.7 Sécurité 18
1.3 Propositions 18
1.3.1 FAIR des processus 18
1.3.2 Distribution d'identifiants pérennes 20
1.3.3 Différentes interopérabilités 21
1.3.3.1 Matériel hôte et système d'exploitation 22
1.3.3.2 Communication avec d'autres matériels 22
1.3.3.3 Utilisation de librairies 22
1.3.3.4 Différents profils d'utilisateurs 23
1.3.3.4.1 Configurer 23
1.3.3.4.2 Utiliser 24
1.3.3.4.3 Visualiser 24
1.3.3.4.4 Développer 24
1.4 Bilan 26

Ce sujet de thèse utilise des notions d'informatique, d'interopérabilité, de liberté logicielle, de supervision assistée par ordinateur et d'instrumentation à distance. Dans la présentation du contexte de cette étude et pour permettre de comprendre l'objectif du travail réalisé, je commence par définir et conceptualiser plusieurs notions qui organisent la supervision et l'instrumentation à distance.

Ensuite nous identifions les problématiques rencontrées pour rendre la supervision et l'instrumentation distante interopérable, libre, accessible, réutilisable et capable de trouver les informations nécessaires aux utilisateurs et aux systèmes informatiques interagissant avec le système étudié.

La supervision (du latin *video*, voir) et l'instrumentation (du latin *instruo*, construire, arranger) ont comme objectifs de permettre à l'homme de surveiller, contrôler et étudier des installations techniques et des phénomènes physiques *in situ*.

L'instrumentation est la technique qui permet d'étudier des grandeurs physiques à l'aide de capteurs et d'appareils de mesure ainsi que d'interagir avec des systèmes via le contrôle d'actionneurs.

La supervision est la technique qui permet d'utiliser un système informatique dans le but de piloter et d'observer des processus physiques ou des systèmes informatiques.

1.1 Systèmes physiques

Le monde de l'instrumentation à distance présente plusieurs défis : collecter des données réelles, contrôler des systèmes, analyser et surveiller une installation, et être capable de faire abstraction de la distance physique entre l'utilisateur et l'expérience.

Afin de répondre à ces besoins, il est nécessaire de réaliser plusieurs opérations. Tout d'abord un système physique a besoin d'être observé pour être appréhendé. Certains systèmes sont dans des lieux isolés ou dangereux pour l'homme, il est alors nécessaire de les observer et d'acquérir des données à distance. Dans ce cas le système repose sur un réseau de communication bidirectionnel qui achemine les informations entre le système physique étudié et le système de supervision (voir figure 1.1).

La persistance des données liées à l'observation est réalisée en les conservant sur l'espace de stockage du système de supervision ou tout autre espace de stockage accessible depuis celui-ci. Ces données permettent aux humains et aux systèmes informatiques de prendre des décisions et d'envoyer des consignes aux actionneurs, acteurs ou unités de contrôle (automate, micro-contrôleur...) afin de modifier le comportement du système physique étudié. L'automatisation de ces tâches est rendue possible en définissant dans le système de supervision des règles réagissant à des évènements, des asservissements ou des algorithmes plus complexes.

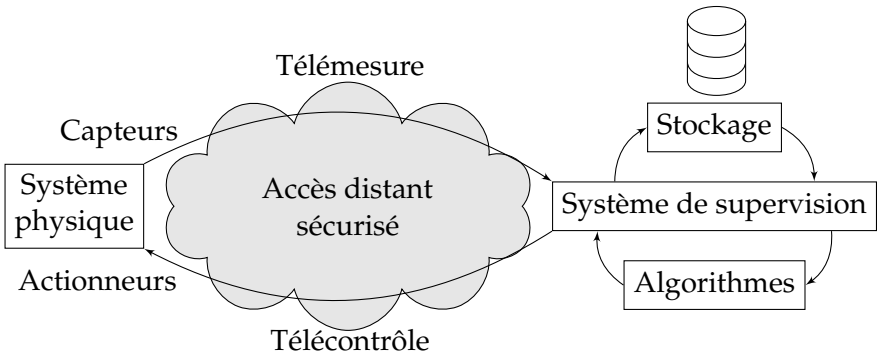


FIGURE 1.1 – Supervision et instrumentation d'un système physique.

1.1.1 Échange d'information

Un protocole de communication est l'ensemble des règles qui permettent à deux appareils ou logiciels d'échanger des informations. Chaque protocole a ses spécificités : connexion physique, conception orientée objet, relation maître-esclave, etc. Le concept d'interopérabilité au niveau de l'échange d'information définit la capacité à utiliser un grand nombre de protocoles de communication et donc de dialoguer avec différents capteurs et actionneurs.

1.1.2 Télémesure et acquisition

La télémesure représente les "**yeux**" de l'instrumentation à distance (voir figure 1.2 [1]). Elle permet de collecter des données de grandeurs physiques *in situ* et de les transmettre en vue de les stocker et de les exploiter (analyse, prédiction, décision). Elle est l'art d'utiliser un capteur de pression, un multimètre, un capteur de température, un compteur de consommation d'énergie pour fournir des informations à un opérateur distant sur l'état d'une presse hydraulique, d'un circuit électrique ou d'une station météorologique.

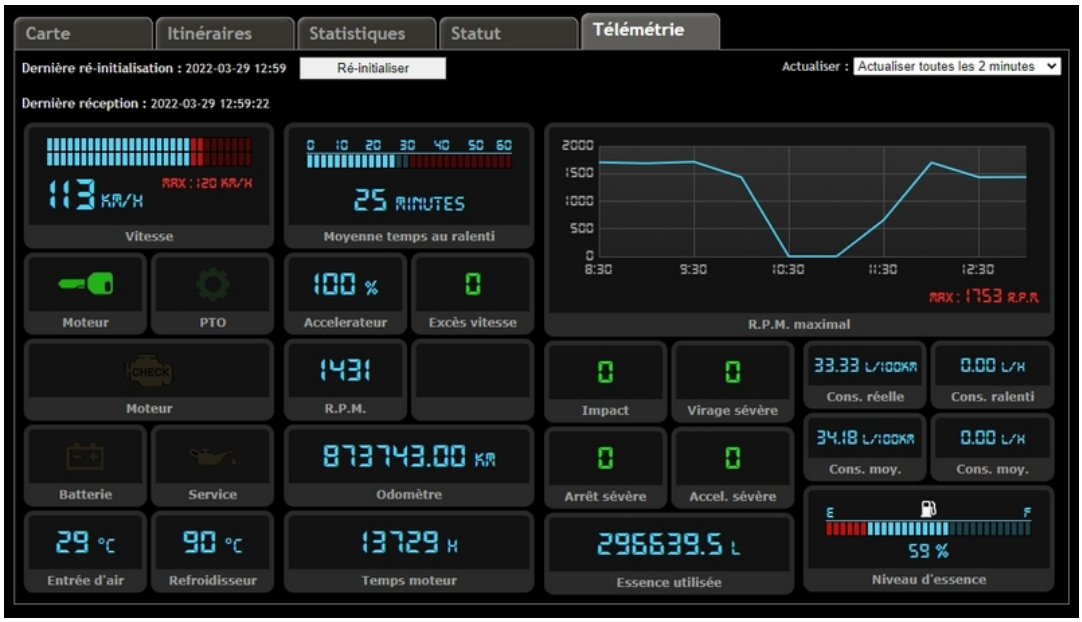


FIGURE 1.2 – Exemple d'interface de télémesure.

La télémesure est apparue avec les premiers instruments électriques développés par des ingénieurs à la fin du 19ème siècle [2]. La figure 1.3 présente le brevet d'un transmetteur de télémesure électrique datant de 1893.

Un processus d'acquisition, décrit à la figure 1.4, comprend des capteurs qui convertissent une quantité physique en un signal électrique, des conditionneurs de signaux qui préparent le signal électrique en l'amplifiant et en le pré-filtrant, des convertisseurs analogiques-numériques qui transforment les signaux électriques en données numériques. Les données numériques doivent ensuite transiter vers

1. https://www.avltrack.com/telemetrie.html
2. https://worldwide.espacenet.com/patent/search/family/002558858/publication/US490012A?q=pn%3DUS490012A

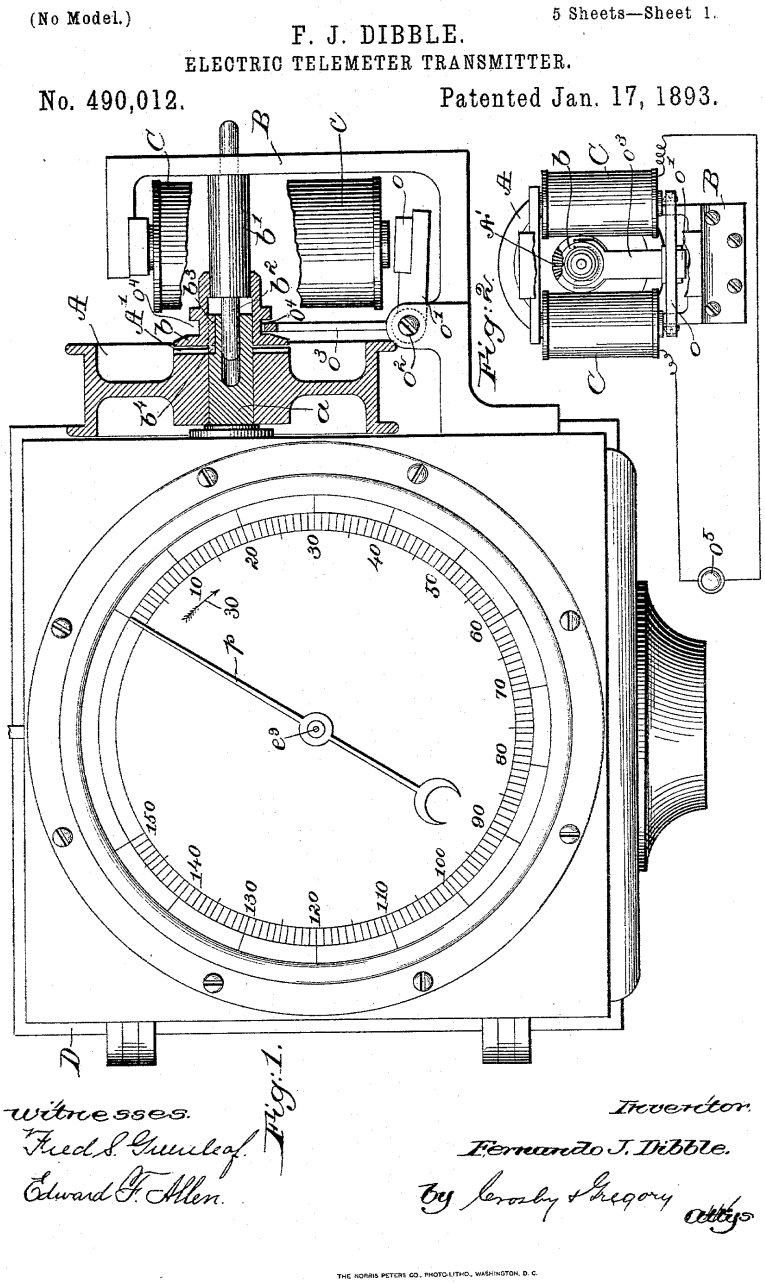


FIGURE 1.3 – Brevet du "Electric Telemeter Transmitter" de 1893.

l'ordinateur en utilisant un protocole de communication (par exemple Modbus [3] ou BacNET [4]).

3. https://modbus.org/
4. http://www.bacnet.org/

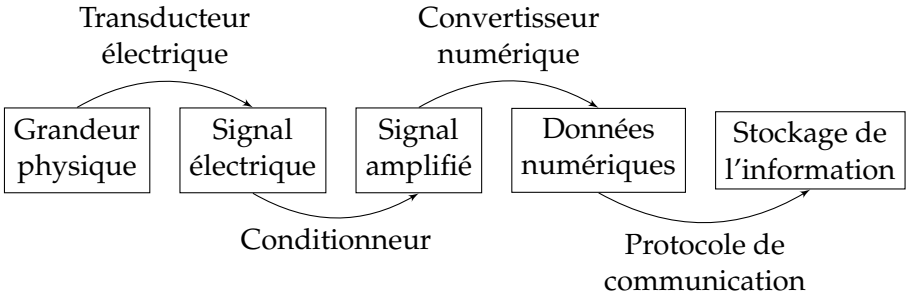


FIGURE 1.4 – La chaîne d'acquisition d'une instrumentation à distance.

1.1.3 Télécontrôle et manipulation

Le télécontrôle correspond aux "**mains**" de l'instrumentation à distance. Il consiste à envoyer un ordre à un actionneur distant et permet d'agir, par exemple, sur un interrupteur, une vanne hydraulique, un moteur électrique ou un robot. De même que la télémesure, il utilise des protocoles de communication. Le télécontrôle apporte une valeur ajoutée à l'utilisateur d'un système instrumenté par rapport à la télémesure, lui permettant de ne pas rester un spectateur passif d'un système physique mais d'être un acteur d'une expérimentation.

Ainsi, il permet d'automatiser des processus tels que l'initialisation d'une station, la transmission cyclique de données, l'envoi de données à intervalle régulier, la synchronisation de l'horloge et la configuration de la station. Il est donc prudent et nécessaire, dans le cas de systèmes automatisés, de définir des limites aux variables contrôlées automatiquement ou par l'utilisateur, d'avoir des protections physiques sur place ou d'envoyer des alertes à un opérateur afin d'éviter des comportements indésirables ou des événements imprévus, qu'il soient durables ou passagers. Ceci pose la problématique de la sécurité du système.

1.1.4 Stockage de données

L'enregistrement des données de configuration des actionneurs (télécontrôle) et des données de mesure provenant de différents capteurs (télémesure) autorise d'identifier et d'analyser, de prédire et de décider du comportement d'un système supervisé. Le suivi des données permet de reconsidérer une situation problématique, de corriger un défaut de configuration ou d'améliorer un processus. Le stockage des données et des méta-données dans une base de données permet d'utiliser *a posteriori* des algorithmes de traitement des données.

De nos jours, certains processus peuvent générer des volumes importants de données et de méta-données (voir DEMARTHON et al. [4]). Ces données massives, en anglais *Big Data*, imposent un **V**olume, une **V**ariété et une **V**élocité au traitement des données et demandent l'utilisation de méthodes particulières et d'opérations en parallèle sur plusieurs machines pour extraire la **V**aleur ajoutée de cette masse de données (les **4V** du Big Data). Les attributs du *Big Data* s'étoffent dans le temps selon les définitions proposées (voir DE MAURO et al. [5]), certains auteurs ajoutant des caractéristiques comme la **V**éracité.

Pour certains processus, la sécurité des données est critique (voir section 1.1.7) et il est nécessaire de crypter le disque de stockage (voir CHAOWEN et al. [6]). Des standards interopérables pour crypter les supports de données sont développés pour répondre à ce besoin (voir HUGHES [7]). La sécurité pour les systèmes de supervision étant un domaine de recherche à part entière, cette thèse n'approfondira pas ce sujet

et s'articulera autour des enjeux des systèmes de supervision libre, interopérable, accessible et réutilisable.

Les données enregistrées doivent pouvoir être exportées dans des formats de fichiers standard (par exemple CSV [5] ou HDF5 [6]) afin d'être traitées par un maximum de logiciels tiers. L'accès sécurisé via une interface de programmation d'applications (en anglais API pour *Application Programming Interface*) ou une connexion directe à la base de données permet un affichage en temps réel dans des applications de surveillance tierces, comme **Grafana** [7], ou une autre installation, appelée instance, de son logiciel de supervision par exemple.

1.1.5 Automatisation

L'intégration d'une multitude de capteurs et d'actionneurs dans un système de supervision à distance ainsi que le stockage des données offre la possibilité d'automatiser des tâches à l'aide d'algorithmes, de définir des alertes et des événements. Les événements peuvent réagir aux :

— variations de valeurs des variables en définissant des seuils limites ou en définissant une variation minimale requise pour une période donnée,
— conditions temporelles, par exemple déclencher une action à intervalles réguliers,
— demandes de l'utilisateur.

Les alertes peuvent être affichées sur l'interface humain-machine (IHM), déclencher une alarme sur un site ou envoyer un message électronique (mail, système de messagerie, SMS de l'anglais *Short Message Service*). Les systèmes dynamiques peuvent être étudiés et gérés en utilisant des scripts de contrôle en boucle fermée par exemple.

1.1.6 Accès distant

L'accès à distance pour l'instrumentation et la supervision donne la possibilité de se connecter à des instruments, des applications ou des données pour des systèmes ne permettant pas d'accès physique direct. L'accès à un ensemble d'instruments distants peut être utilisé pour protéger l'utilisateur d'une manipulation dangereuse ou pour lui éviter de se déplacer (par exemple dans une centrale nucléaire). La sensation d'immersion offerte à l'utilisateur cherche à remplacer les cinq sens humains, via l'interface et les données disponibles (voir LUTHON et al. [2]). Une bonne qualité d'immersion d'une IHM permettra d'appréhender le système physique et d'utiliser l'ensemble des informations disponibles.

Des contraintes peuvent être induites par l'accès distant, par exemple la synchronisation d'événements peut être difficile en raison du temps de réponse de chaque équipement, d'un débit faible d'échange de données ou d'une accessibilité intermittente. De plus, si tous les paramètres physiques d'un système ne sont pas mesurés, le manque d'information ou l'accès partiel aux informations peut fausser l'analyse de l'utilisateur.

5. https://datatracker.ietf.org/doc/html/rfc4180
6. https://portal.hdfgroup.org/display/HDF5/HDF5
7. https://grafana.com/

1.1.7 Sécurité

Dans le passé, les systèmes de surveillance autonomes n'incluaient pas de fonctions de sécurité (YADAV et al. [8]). L'évolution des systèmes de supervision monolithique (pas de réseau ni d'interconnexion de systèmes, voir section 2.1), vers des systèmes distribués dans des réseaux locaux, puis vers des systèmes en réseau via internet et maintenant des réseaux d'objets connectés, a rendu les enjeux de sécurité prépondérants (PLIATSIOS et al. [9]). Les critères de sécurité sont devenus une préoccupation majeure des industries avec l'avènement des systèmes de surveillance et des objets connectés à Internet (SAJID et al. [10]).

1.2 Problématiques abordées

L'ensemble des problématiques rencontrées lors de l'instrumentation d'un système physique et de la mise en place d'un système de supervision sera exposé dans cette partie.

Les différents domaines d'applications exposés en Introduction générale présentent des problématiques similaires qui peuvent être administrées par une solution commune et des problématiques spécifiques qui seront gérées par des ajouts de modules spécifiques.

1.2.1 Applications variées

La diversité des systèmes étudiés par les équipes avec lesquelles je travaille demande, si l'on veut y répondre de manière globale, d'avoir une approche agnostique du problème et de proposer une solution abstraite. Pour ne pas "réinventer la roue", nous avons cherché à adapter et à faire évoluer des concepts et des solutions libres existants.

1.2.2 Information trouvable, accessible, interopérable et réutilisable

Depuis 2009 (EVANS [11]) le nombre d'objets connectés a dépassé le nombre d'êtres humains sur terre. C'est ainsi que l'équipe Internet Business Solution Group (IBSG) de Cisco définit l'entrée dans l'ère de l'internet des objets (IoT, de l'anglais *Internet of Things*). Cette émergence d'objets connectés produit un grand nombre de données. Ces données sont de plus en plus en accès libre via des sites web (REICHMAN et al. [12]), comme par exemple :

- les données publiques de Météo France [8],
- les cartes de données météo : OpenWeatherMap [9],
- les données ouvertes du gouvernement américain [10],
- les données ouvertes du gouvernement français [11],
- OpenDataInception : un site regroupant plus de 2600 portails de données ouvertes [12].

Pour permettre une standardisation des méthodes et une reproductibilité des analyses, les défis des données ouvertes sont notamment la dispersion, l'hétérogénéité et la provenance (REICHMAN et al. [12]). Afin d'améliorer l'échange, l'utilisation et

8. https://donneespubliques.meteofrance.fr/
9. https://openweathermap.org/
10. https://data.gov/
11. https://www.data.gouv.fr
12. https://opendatainception.io/

l'accès aux données, les lignes directrices pour des données trouvables, accessibles, interopérables, et réutilisables (en anglais FAIR pour *Findable, Accessible, Interoperable and Reusable*) ont émergé dans la communauté scientifique depuis les principes érigés par WILKINSON et al. [13].

Notons que les principes du FAIR sont également une condition essentielle pour que le coût environnemental des données reste acceptable en regard des multiples bénéfices que l'on pourra en tirer, comme mentionné en conclusion (p.311) de l'ouvrage récent sur les impacts du numérique dans le monde PITRON [14].

Ces principes s'appliquent dans un premier temps aux données et sont énoncés dans la table 1.1.

TABLE 1.1 – Principes des données FAIR énoncés par Wilkinson.

Qualificatif	Sigle	Définition
Trouvable	F1	Les (méta)données se voient attribuer un identifiant unique et permanent au niveau mondial.
	F2	Les données sont décrites avec des méta-données riches (définies par R1 ci-dessous).
	F3	Les méta-données incluent clairement et explicitement l'identifiant des données qu'elles décrivent.
	F4	Les (méta)données sont enregistrées ou indexées dans une ressource consultable.
Accessible	A1	Les (méta)données sont récupérables par leur identifiant en utilisant un protocole de communication standardisé.
	A1.1	le protocole est ouvert, gratuit et universellement réalisable.
	A1.2	Le protocole prévoit une procédure d'authentification et d'autorisation, le cas échéant.
	A2	Les méta-données sont accessibles, même lorsque les données ne sont plus disponibles.
Interopérable	I1	Les (méta)données utilisent un langage formel, accessible, partagé et largement applicable pour la représentation des connaissances.
	I2	Les (méta)données utilisent des vocabulaires qui suivent les principes FAIR.
	I3	Les (méta)données incluent des références à d'autres (méta)données.
Réutilisable	R1	Les méta(données) sont richement décrites avec une pluralité d'attributs précis et pertinents.
	R1.1	Les (méta)données sont publiées avec une licence d'utilisation des données claire et accessible.
	R1.2	Les (méta)données sont associées à une provenance détaillée.
	R1.3	Les (méta)données répondent à des normes communautaires pertinentes pour le domaine.

Ces propositions ont été élargies et ne concernent désormais plus seulement les données. En effet, les discussions autour de FAIR incluent les algorithmes, les logiciels et d'autres ressources de recherche (LAMPRECHT et al. [15]). Chaque installation, appelée *instance*, d'un logiciel doit être identifiable.

Afin de différencier les principes érigés pour les données par WILKINSON et al. [13], des principes proposés pour les logiciels de recherche par LAMPRECHT et al. [15], nous les précéderons d'une astérisque (*) (voir table 1.2).

TABLE 1.2 – Principes des logiciels FAIR énoncés par Lamprecht.

Qualificatif	Sigle	Définition
Trouvable	*F1	Le logiciel et ses méta-données associées ont un identifiant global, unique et persistant pour chaque version publiée.
	*F2	Les logiciels sont décrits avec des méta-données riches.
	*F3	Les méta-données incluent clairement et explicitement les identifiants de toutes les versions du logiciel qu'elles décrivent.
	*F4	Les logiciels et leurs méta-données sont inclus dans un registre de logiciels consultable.
Accessible	*A1	Les logiciels et leurs méta-données associées sont accessibles par leur identifiant au moyen d'un protocole de communication normalisé.
	*A1.1	Le protocole est ouvert, gratuit et universellement implémentable.
	*A1.2	Le protocole permet une procédure d'authentification et d'autorisation, si nécessaire.
	*A2	Les méta-données des logiciels sont accessibles, même lorsque le logiciel n'est plus disponible.
Interopérable	*I1	Le logiciel et ses méta-données associées utilisent un langage formel, accessible, partagé et largement applicable pour faciliter la lisibilité par la machine et l'échange de données.
	*I2S.1	Les logiciels et leurs méta-données sont formellement décrits à l'aide de vocabulaires contrôlés qui respectent les principes FAIR.
	*I2S.2	Les logiciels utilisent et produisent des types et des formats de données qui sont formellement décrits à l'aide de vocabulaires contrôlés conformes aux principes du FAIR.
	*I4S	Les dépendances des logiciels sont documentées et il existe des mécanismes pour y accéder.
Réutilisable	*R1	Les logiciels et leurs méta-données sont décrits de manière riche avec une pluralité d'attributs précis et pertinents.
	*R1.1	Le logiciel et ses méta-données associées ont des licences d'utilisation indépendantes, claires et accessibles, compatibles avec les dépendances du logiciel.
	*R1.2	Les méta-données du logiciel incluent une provenance détaillée, le niveau de détail doit être convenu par la communauté.
	*R1.3	Les méta-données et la documentation du logiciel sont conformes aux normes communautaires pertinentes pour le domaine.

1.2.3 Toile sémantique

La notion de toile sémantique (en anglais *semantic web*) est apparue en 2001 dans l'article de BERNERS-LEE et al. [16] où il présente une nouvelle orientation d'internet. Le web a d'abord été créé pour être utilisé, lu et conçu par et pour les humains. Il était orienté autour de documents écrits par exemple en langage de balisage hypertexte (HTML, de l'anglais *HyperText Markup Language*).

Avec le web sémantique, aussi appelé web 3.0, internet sera orienté autour de données, de méta-données et d'informations qui pourront être traitées automatiquement par des algorithmes. Le web sémantique rejoint les recommandations FAIR, c'est-à-dire que chaque ressource doit être identifiable par un identifiant uniforme de ressource (en anglais URI pour *Uniform Resource Identifier*) et accessible par une adresse web (en anglais URL pour *Uniform Resource Locator*, qui est un sous-groupe des URI).

1.2.4 Catégories de logiciels

L'organisation à but non lucratif Free Software Foundation (FSF) [13], dont la mission est de promouvoir la liberté des utilisateurs d'ordinateurs, ainsi que le système d'exploitation GNU (acronyme récursif pour *GNU's Not Unix*), tous deux fondés par Richard Stallman, proposent plusieurs catégories de logiciel.

Ces catégories sont représentées par Chao-Kuei de la fondation GNU [14] dans la figure 1.5. Il définit 3 catégories :

— à gauche les logiciels libres,
— à droite les logiciels privateurs (ou privés),
— et l'ensemble des logiciels à téléchargement gratuit regroupant une partie des logiciels libres et privateurs.

Certaines catégories peuvent sembler similaires. Les distinctions, non moins importantes pour la FSF et GNU, peuvent être d'ordre politique ou philosophique. Elles sont explicitées ci-après.

1.2.4.1 Logiciel libre

La fondation GNU définit [15] les logiciels libres comme étant ceux dont "les utilisateurs ont la liberté d'exécuter, copier, distribuer, étudier, modifier et améliorer ces logiciels". Elle définit ainsi quatre libertés :

0. faire fonctionner le programme,
1. étudier et modifier le fonctionnement du programme,
2. distribuer des copies,
3. distribuer ses modifications.

Ces libertés limitent fortement les droits issus de la propriété, sans pour autant interdire la propriété que Proudhon définit comme le vol (PROUDHON [17]). Chacun est bien propriétaire de ce qu'il développe mais dès lors que quelqu'un possède une application libre, il possède les mêmes droits.

13. https://www.fsf.org/
14. https://www.gnu.org/philosophy/categories.fr.html
15. https://www.gnu.org/philosophy/free-sw.fr.html

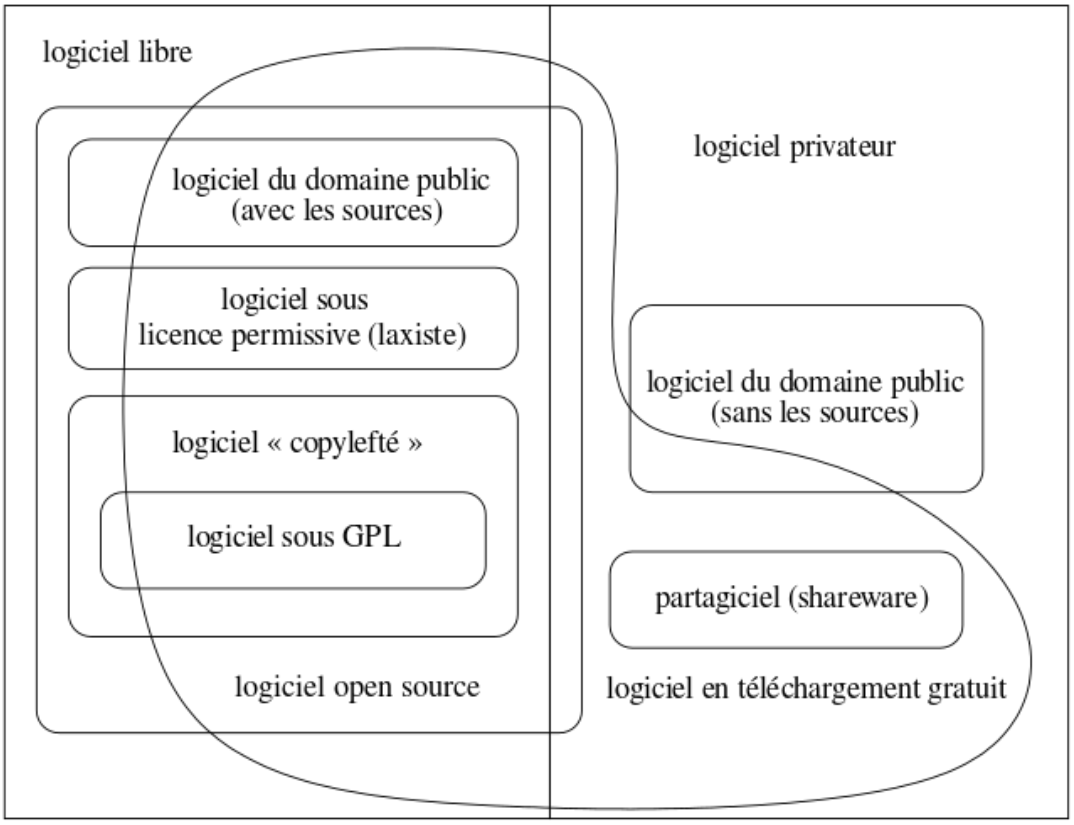


FIGURE 1.5 – Catégories de logiciels définies par Chao-Kuei.

1.2.4.2 Logiciel *open source*

Les logiciels open source se différencient peu des logiciels libres. La définition légale en France d'un standard ouvert, dans la loi (LÉGIFRANCE [18]), est :

> *On entend par standard ouvert tout protocole de communication, d'interconnexion ou d'échange et tout format de données interopérable et dont les spécifications techniques sont publiques et sans restriction d'accès ni de mise en œuvre.*

Le standard ouvert propose une interopérabilité dans les échanges et les communications mais ne suppose pas que le logiciel soit libre. Il ne parle pas de liberté de modification ni de liberté de distribution. La notion d'*open source* ne s'intéresse pas à l'éthique d'après Richard Stallman, alors que le mouvement du logiciel libre promeut la liberté des utilisateurs de l'informatique (voir section 1.2.4.1).

1.2.4.3 *Copyleft*

La notion de *copyleft*, traduit par *gauche d'auteur* ou *copie laissée* en opposition au droit d'auteur, a pour but de protéger une œuvre tout en la rendant libre ainsi que toutes les versions modifiées de l'œuvre [16]. L'auteur protège ainsi son œuvre de tout changement de licence. En effet quiconque modifie le logiciel conserve la même licence et donc les mêmes droits.

16. https://www.gnu.org/licenses/copyleft.html

La plus connue des licences sous copyleft est sans doute la licence publique générale GNU (GNU GPL, en anglais GNU *General Public Licence*[17]). Elle répond favorablement aux quatre libertés nécessaires aux logiciels libres définies en 1.2.4.1.

1.2.4.4 Recommandations pour les logiciels

Des recommandations émergent dans la communauté scientifique dans le but d'encourager les développeurs de logiciels au service de la recherche et mettre ainsi en œuvre certaines bonnes pratiques. Les recommandations proposées par JIMÉNEZ et al. [19] sont au nombre de quatre :

1. rendre accessible publiquement le code source d'une application dès sa création,
2. faciliter l'accès au code en enregistrant le logiciel et toutes les données nécessaires à son utilisation (licence, version, identifiant pérenne : PID de l'anglais *Persistent Identifier*) sur des moteurs de recherche et des dépôts de logiciels populaires,
3. choisir une licence logicielle qui permette d'utiliser l'ensemble des librairies tierces,
4. définir des règles de contribution, de gérance et de communication pour la communauté souhaitant utiliser ou participer au développement du logiciel.

Ces recommandations rejoignent les principes de **trouver**, **accéder** et **réutiliser** du FAIR.

1.2.5 Interopérabilité

Le gouvernement français publie depuis 2005[18] un référentiel général d'interopérabilité afin de dialoguer avec ses systèmes d'information, qu'il décompose en plusieurs niveaux (voir DINUM [20]) :

— le niveau juridique qui définit le cadre légal dans lequel des données sont transmises et échangées,
— le niveau organisationnel qui définit qui envoie la donnée, à quel moment et suite à quel évènement,
— le niveau technique qui définit le transport des données,
— le niveau syntaxique qui définit le format dans lequel les données sont transmises,
— le niveau sémantique qui définit la signification des données.

Dans notre cas, le niveau juridique est représenté par la licence logicielle, le niveau organisationnel par la définition d'un processus de supervision (voir la section 1.3.1), le niveau technique par la télémesure et le télécontrôle, le niveau syntaxique par les protocoles de communication mis en œuvre, et le niveau sémantique par la présentation des données via une interface ou en donnant de la **valeur** (voir DE MAURO et al. [5]) aux données.

L'interopérabilité doit se retrouver à toutes les étapes de la mise en place du système de supervision, listées ci-après :

— installation logicielle,
— connexion physique des instruments,
— configuration du matériel,
— création d'une interface d'utilisation,

17. https://www.gnu.org/licenses/gpl.html
18. https://www.numerique.gouv.fr/publications/interoperabilite/

— utilisation et interaction avec le matériel à distance.

Par exemple, lorsque deux systèmes veulent échanger des données, ils doivent partager un même protocole de communication et format de données. Ils mettent alors en place une interopérabilité au niveau de la communication. L'interopérabilité est donc un choix au niveau de la conception du matériel ou du logiciel permettant de l'intégrer dans un groupe plus complexe d'éléments. Elle doit permettre d'utiliser un système dans sa globalité et sans limitation.

Des normes permettent de rendre les systèmes interopérables, par exemple le World Wide Web Consortium (W3C) [19] se définit comme une communauté internationale développant des standards ouverts pour encourager la croissance à long terme de l'internet.

S'attaquer à l'interopérabilité en la limitant peut être utilisé par certains constructeurs en position de monopole pour se renforcer (voir la victoire de la Commission européenne sur Microsoft [21]).

Les différentes notions d'interopérabilité seront détaillées dans les propositions en section 1.3.

1.2.6 Décentralisation

Tim Berners-Lee, qui est considéré comme l'un des inventeurs du *World-Wide-Web* (WWW), explique qu'il est urgent de revenir aux fondements de l'internet et de décentraliser son fonctionnement et ses applications afin de garantir son ouverture à tous [20]. Certains états, comme l'Iran ou la Chine, disent se protéger des surveillances de masse, comme celle de la NSA révélée par Edward Snowden [21], et ferment leur internet. Ils peuvent néanmoins réaliser les mêmes opérations de surveillance au sein de leur intranet.

Le principe d'application décentralisée existe au moins depuis les années 1980 (BALKOVICH et al. [22]). Une application décentralisée est une application qui est accessible par différentes adresses. Par exemple, *YouTube*, qui est une application centralisée, est accessible uniquement via une adresse [22], redirigeant vers un groupe de serveurs de *Google*. Alors que *PeerTube* est accessible via une liste d'adresses [23] en constante évolution, chacun pouvant installer l'application et la rendre accessible au public s'il le désire.

Pour les applications décentralisées, l'information est aussi décentralisée et se transmet via des protocoles d'échange tels que l'*ActivityPub* [24] à l'aide de notifications (ILIK et al. [23]).

Les applications libres et décentralisées permettent d'éviter les prises de monopole et de contourner les fermetures de certains réseaux par des états ou des opérateurs de télécommunications, comme c'est le cas pour le site d'accès libre aux publications scientifiques Sci-Hub [25]. Chacun pouvant créer une copie d'une application, appelée instance, et la rendre accessible sur internet.

La décentralisation permet de répartir le nombre de connexions sur le serveur, de diminuer le coût des serveurs, par rapport à une gestion centralisée, lorsque le nombre de connexions et la quantité de données transmises sont importants (KAUR

19. https://www.w3.org/
20. https://www.wired.co.uk/article/tim-berners-lee-reclaim-the-web
21. https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data
22. https://www.youtube.com/
23. https://joinpeertube.org/instances#instances-list
24. https://activitypub.rocks/
25. https://www.nextinpact.com/article/29285/107689-les-principaux-fai-francais-doivent-bloquer-sci-hub-et-libgen

et al. [24]). Elle permet aussi de diminuer les risques de perte totale de connexion, car si un serveur n'est plus accessible, l'application reste disponible via les autres instances, avec potentiellement des données manquantes.

La méthode de l'informatique en périphérie de réseau (***Edge computing*** en anglais) consiste à pré-traiter les données à la sortie ou proche des capteurs avant de les transférer à un ou plusieurs serveurs. Cette méthode réduit les besoins en bande passante entre le capteur et le serveur en ne transférant pas les données brutes ou en sélectionnant les données à envoyer. Cette technique utilise par exemple des téléphones portables pour traiter les données (AHMED et al. [25]).

1.2.7 Sécurité

Les systèmes de supervision sont utilisés dans des systèmes de *monitoring* (surveillance) pour des infrastructures critiques (voir PLIATSIOS et al. [9]). Les systèmes récents sont inter-connectés via internet et rendent plus faciles les cyber-attaques. Ces attaques peuvent avoir des conséquences catastrophiques sur des systèmes publics en compromettant des infrastructures cruciales qui acheminent l'électricité, l'eau ou le gaz. Elles peuvent aussi s'attaquer à des infrastructures hospitalières ou militaires critiques et les détruire.

Comme expliqué en sections 1.1.4 et 1.1.7, et bien que la sécurité soit un enjeu crucial pour les systèmes de supervision et les utilisateurs, nous n'approfondirons pas cette partie dans ce mémoire de thèse.

1.3 Propositions

Je détaille dans cette section des propositions permettant d'étendre les directives du FAIR aux logiciels de supervision et d'instrumentation, et de concevoir un système générique et agnostique du domaine d'application.

1.3.1 FAIR des processus

J'appellerai par la suite **processus de supervision**, représenté à la figure 1.6, toute expérimentation réalisée à l'aide d'un système de supervision ou d'instrumentation pouvant impliquer :

— des variables d'entrée définies par l'utilisateur,
— des variables d'entrée non définies par l'utilisateur, par exemple la date ou la valeur d'un capteur,
— des fonctions et des algorithmes manipulant ces données d'entrée,
— des variables de sortie générées par les fonctions et algorithmes utilisés, qui seront stockées ou serviront à contrôler des actionneurs.

Considérons un processus de supervision simple représenté à la figure 1.7 : toutes les heures le processus lit la valeur d'un capteur de luminosité, si la valeur est inférieure à un seuil défini par l'utilisateur, une lumière est allumée. Les variables d'entrée définies par l'utilisateur sont le seuil ainsi que l'exécution du processus toute les heures. La variable d'entrée indépendante de l'utilisateur est la valeur envoyée par le capteur de luminosité. L'algorithme contient la règle suivante, si la luminosité lue est inférieure au seuil la lampe s'allume, sinon la lampe s'éteint. La variable de sortie est l'interrupteur contrôlant l'allumage de la lampe.

Afin de rendre un processus de supervision compatible avec les principes du FAIR, je propose de pouvoir :

— l'identifier à l'aide d'un PID (trouver),

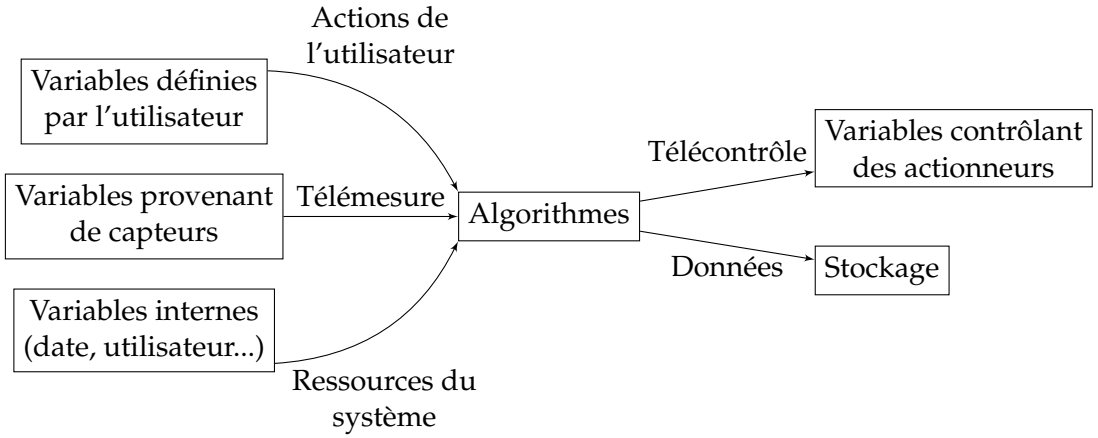


FIGURE 1.6 – Schéma d'un processus de supervision.

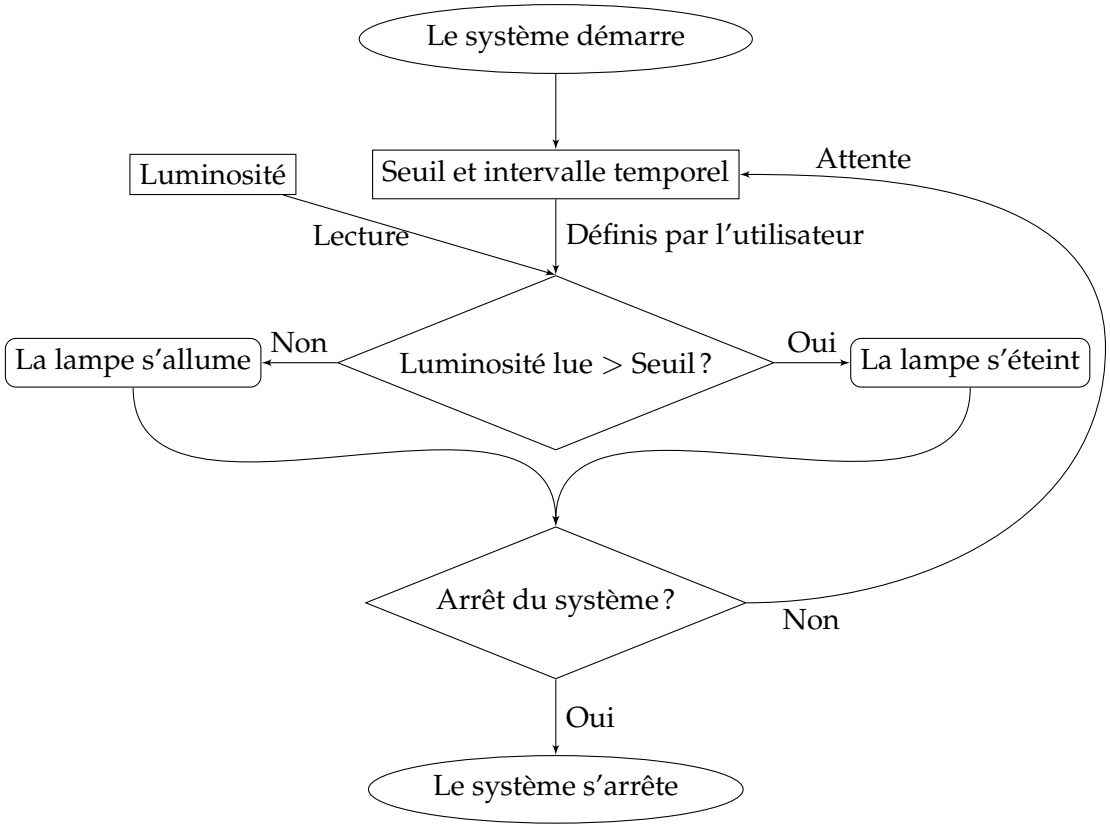


FIGURE 1.7 – Exemple d'un processus de supervision.

— l'ajouter à une instance (réutiliser),
— modifier les capteurs d'entrée et les actionneurs de sortie (interopérable),
— le publier sur une plateforme publique ou privée (accessible).

Sachant qu'un système de supervision ou d'instrumentation permettant de définir des processus doit aussi définir des paramètres (entrée) et des scripts (fonctions et algorithmes) produisant des données (sortie), le PID de ce processus doit permettre d'importer dans une nouvelle installation l'ensemble des éléments nécessaires à son fonctionnement, par exemple :

— les variables d'entrée et de sortie,

— les instruments permettant de lire les variables d'entrée (capteurs) et de contrôler les variables de sorties (actionneurs),
— les unités des variables,
— les valeurs des variables d'entrée définies par l'utilisateur sur une installation donnée,
— l'algorithme.

L'ensemble des éléments en relation avec un processus devra respecter les principes du FAIR. Par exemple une variable d'entrée :
— possédera un PID pour la trouver,
— contiendra l'ensemble de sa configuration nécessaire à sa réutilisation, liée via des PID à la configurations d'un instrument, d'un protocole, d'une unité, etc.,
— sera modifiable et interopérable,
— existera sur une plateforme accessible à l'utilisateur.

De même une donnée de sortie :
— possédera un PID pour la trouver,
— indiquera à l'aide de méta-données sa provenance afin d'être réutilisée :
  — PID du processus,
  — date de création,
  — configuration du processus au moment de sa création,
— utilisera une description claire et comprise de tous dans le domaine étudié,
— existera sur une plateforme accessible à l'utilisateur.

Ainsi toute donnée générée par ce processus de supervision pour un processus physique donné :
— possédera un PID pour la trouver,
— contiendra l'ensemble de la configuration nécessaire à sa réutilisation, liée via des PID. Par exemple le lieu de l'instrumentation ou la personne responsable,
— sera comparable à d'autres données, par exemple pour des installations similaires,
— existera sur une plateforme accessible à l'utilisateur.

Reprenons notre exemple, si un utilisateur utilise le processus de supervision en remplaçant la lecture de la luminosité venant d'un capteur par l'action d'un utilisateur qui décide ou non d'allumer la lampe. Dans ce cas la variable d'entrée lue via un capteur sera remplacée par une variable d'entrée contrôlée par l'interface utilisateur. Un nouveau PID sera créé pour cette variante du processus de base et une nouvelle version de la configuration du processus possédera un PID.

1.3.2 Distribution d'identifiants pérennes

Afin de trouver et d'accéder à une instance, un objet ou une donnée liés à un système de supervision, il est proposé l'utilisation de PID pour tout logiciel de supervision et d'instrumentation à distance. Afin de rendre durable la relation entre un identifiant et la ressource liée, le PID doit :
— fournir une organisation offrant un accès permanent aux PID,
— être opaque afin d'éviter la modification de celui-ci si une caractéristique change, dans ce cas un nouveau PID doit être publié,
— ne pas publier un identifiant déjà existant.

Chaque logiciel possède sa structure propre et devra donc être accessible via un PID, comme par exemple le Digital Object Identifier [26] (DOI) ou le Archival Resource

26. https://www.iso.org/obp/ui/#iso:std:iso:26324:ed-1:v1:en

Key [27] (ARK). Dans les exemples suivants, le ARK sera utilisé car il permet de créer sa propre autorité de gestion des PID.

L'auteur de chaque logiciel devra passer par une autorité d'identification qui permettra à son logiciel de posséder un identifiant unique. Dans le cas de l'ARK il est nécessaire de déployer une autorité d'adressage (NMA, de l'anglais *Name Mapping Authority*), et une autorité d'assignement de nom (NAAN, de l'anglais *Name Assigning Authority Number*). Un formulaire en ligne est disponible pour enregistrer son autorité sur le site de l'ARK Alliance.

L'identifiant ARK se présente sous la forme d'un URI suivant :

`ark:/NAAN/ID/PART.TYPE`

Les éléments constituant un ARK sont :

— NAAN : autorité d'assignement attribuant un identifiant à une ressource,
— ID : identifiant d'une ressource,
— PART (optionnel) : identifiant d'une partie de la ressource, par exemple la page d'un livre,
— TYPE (optionnel) : variante de la ressource ou de la partie de la ressource, par exemple la version ou le format.

En passant par un résolveur, le NMA, l'ARK est intégrée dans un URI pour rediriger les identifiants ARK vers la NAAN correspondante. L'URI prend la forme de :

`PROTOCOLE://NMA/ark:/NAAN/ID/PART.TYPE`

Les éléments constituant l'URI d'un ARK sont :

— PROTOCOLE : protocole d'accès, par exemple **https**,
— NMA : autorité d'adressage permettant de résoudre l'URI vers la ressource ARK.

Par exemple, l'ouvrage **Les filles du feu** par Gérard de Nerval, hébergé à la Bibliothèque Nationale de France (BNF), possède plusieurs URI :

— le livre : `https://gallica.bnf.fr/ark:/12148/bpt6k107371t`,
— un chapitre : `https://gallica.bnf.fr/ark:/12148/bpt6k107371t/f134`,
— la miniature du chapitre : `https://gallica.bnf.fr/ark:/12148/bpt6k107371t/f134.thumbnail`,

où les éléments sont :

— PROTOCOLE : *https*,
— NMA de la BNF : *gallica.bnf.fr*,
— NAAN : *12148*,
— ID : *bpt6k107371t*,
— PART : aucun ou *f134*,
— TYPE : aucun ou *thumbnail*.

Dans le cas d'un système de supervision, chaque URI renverra par exemple vers une ressource de l'installation, un historique des versions utilisées, une personne à contacter ainsi qu'un lien permettant d'accéder à tous les paramètres et les données pour cette instance. Pour des raisons de confidentialité, l'accès à un URI peut être protégé et demander des droits d'accès.

1.3.3 Différentes interopérabilités

D'après GOPSTEIN et al. [26], un moyen pratique de caractériser l'interopérabilité d'un système est d'étudier sa complexité d'intégration. L'intégration d'un équipement va de la **simple** connexion, connue sous le terme anglais de *Plug-and-Play* (PnP),

27. `https://arks.org/`

ne nécessitant aucun développement ni configuration, à l'intégration **complexe** demandant le développement d'interfaces spécialisées, en passant par l'intégration dite **moyenne**, demandant seulement d'entrer la configuration d'un équipement. L'intégration complexe est souvent utilisée lors de l'intégration d'un composant unique, utilisant un protocole propriétaire ou obsolète, alors que l'intégration moyenne est souvent utilisée pour intégrer un ensemble de ressources partageant un même protocole.

Lors d'un échange d'informations entre deux composants, il est nécessaire de s'assurer que les deux parties donnent le même sens à l'information échangée, ce qu'on appelle l'interopérabilité sémantique. L'interopérabilité sémantique n'est pas évidente à obtenir lors d'un échange de données anciennes ne comportant pas ou peu de méta-données. Le résultat d'une erreur de sens attribué à une donnée peut mener à de faux résultats et peut être catastrophique (voir HEILER [27]).

J'expliciterai dans cette section l'ensemble des domaines pour lesquels un logiciel de supervision doit préférer une approche interopérable.

1.3.3.1 Matériel hôte et système d'exploitation

Un système d'exploitation (OS de l'anglais *Operating System*) est l'ensemble des programmes qui permettent l'utilisation d'un ordinateur, d'un téléphone, d'une tablette ou d'un serveur. Chaque OS fonctionne sur une liste de matériels donnés. Afin de faire fonctionner un système de supervision sur un grand nombre de matériels hôtes, appelés serveurs, il est nécessaire de le faire fonctionner sur différents OS, soit en utilisant un langage de programmation compatible sur un grand nombre d'OS soit en utilisant une technologie comme les conteneurs qui permet d'exécuter le logiciel et ses librairies nécessaires sur un OS.

1.3.3.2 Communication avec d'autres matériels

La communication entre le serveur et des instruments permet de lire des valeurs de capteurs et de contrôler des actionneurs (voir 1.1.2 et 1.1.3). Il existe plusieurs dizaines de protocoles de communication pour les systèmes automatisés[28], leur intégration permet de communiquer avec un ensemble d'objets (voir 1.2.5). Cette liste n'étant pas figée, une approche évolutive et modulaire du logiciel de supervision permettra d'ajouter des protocoles au fur et à mesure des besoins ou de leur apparition.

1.3.3.3 Utilisation de librairies

Les licences logicielles ne sont pas toutes compatibles entre elles. Par exemple un programme utilisant une librairie sous GPL doit être sous licence GPL ou toute licence compatible avec la GPL[29]. On parle alors de contamination de la licence GPL. En revanche dans un programme sous licence GPL, une librairie sous une autre licence compatible avec la GPL conserve sa licence comme présenté dans la figure 1.8. Cette contrainte peut limiter l'utilisation de libraires ou de programmes à inclure dans un système de supervision.

Afin de satisfaire les recommandations N°1, N°3 et N°4 de JIMÉNEZ et al. [19] (voir section 1.2.4.4), de permettre à tous de modifier et d'ajouter de nouvelles fonctionnalités, protocoles de communication ou interfaces de visualisation en suivant les

28. https://en.wikipedia.org/wiki/List_of_automation_protocols
29. https://www.gnu.org/licenses/license-list.fr.html

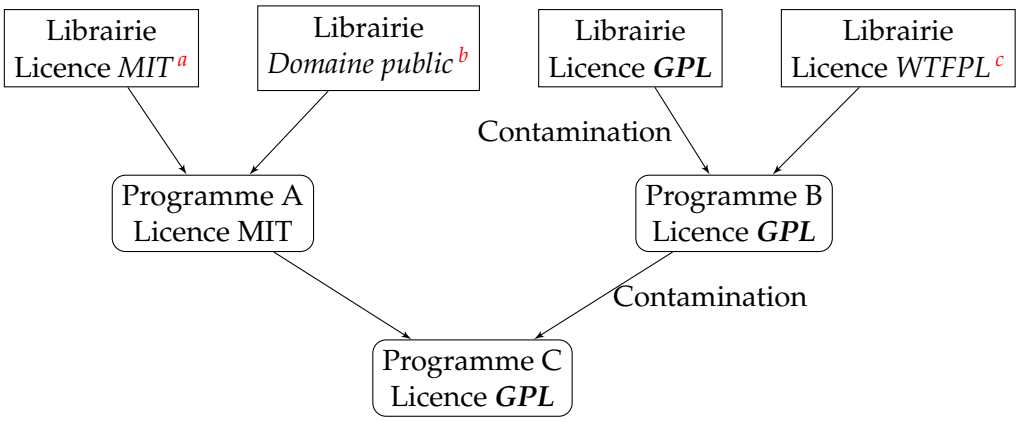


FIGURE 1.8 – Principe de contamination de licences sous copyleft.

a. https://directory.fsf.org/wiki/License:Expat
b. https://directory.fsf.org/wiki/License:PublicDomain
c. https://directory.fsf.org/wiki/License:WTFPL-2

quatre règles de la fondation GNU (voir section 1.2.4.1), je recommande d'utiliser une des deux licences suivantes :

— GNU GPL v3 ou ultérieure, qui permet à toute personne achetant un logiciel utilisant votre logiciel de supervision d'en obtenir le code source et de le modifier, le distribuer ou le vendre,
— AGPL [30] v3 ou ultérieure, donnant les mêmes droits que la GPL à une personne par le seul fait d'accéder à un service proposé par le logiciel sans même l'avoir acheté ou avoir payé pour ce service. L'AGPL est utile par exemple pour les applications web.

1.3.3.4 Différents profils d'utilisateurs

L'interopérabilité d'une interface humain-machine se caractérise par l'ensemble des différents profils d'utilisateurs accédant à un même système distant. Ces profils peuvent se caractériser par la fonction qu'ils exercent sur le système (configurer, utiliser, visualiser, développer) et qui sont décrits aux paragraphes suivants. Quel que soit le type d'utilisateur, l'interopérabilité humaine doit permettre à tous de comprendre et d'interagir avec le système en proposant par exemple plusieurs niveaux d'interaction : novice et expert, scientifique et non scientifique.

1.3.3.4.1 Configurer

Lors de la configuration d'un système de supervision, les caractéristiques à prendre en compte peuvent être très variées. Par exemple :

— stocker une date ou une température,
— récupérer la consommation énergétique d'un bâtiment et afficher la consommation énergétique totale par jour, mois ou année,
— afficher l'évolution de plusieurs variables selon plusieurs dimension (temps, espace ou paramètre physique).

En partant des mêmes données, il est possible de présenter le même résultat visuel en le décrivant différemment. Pour un utilisateur lambda, un graphique montrant l'évolution d'une température d'une ville dans le temps sera décrit par un utilisateur scientifique comme l'évolution d'une fonction définie par un paramètre (la valeur),

30. https://www.gnu.org/licenses/agpl.html

une unité (degré Celsius), une position géographique (latitude et longitude) et une dimension (le temps).

L'interopérabilité pour la configuration d'un système nécessite d'être capable de proposer différentes approches afin de s'adapter à la sémantique de l'utilisateur. Ainsi il est possible d'obtenir le même résultat à partir des mêmes données en utilisant deux formulaires de configuration différents.

Cette proposition rejoint la description des niveaux d'interopérabilité proposée par GOPSTEIN et al. [26] (PnP, moyenne et complexe, voir section 1.3.3) en proposant différents niveaux de configuration donnant accès à plus ou moins de caractéristiques ou en proposant différentes conceptualisations d'un même problème.

1.3.3.4.2 Utiliser

Afin de pouvoir utiliser un système de supervision, il est nécessaire de disposer d'un matériel informatique, appelé client, interagissant avec le serveur. De nos jours tous les logiciels de supervision sont accessibles via un ordinateur. Certains sont aussi accessibles via un téléphone ou une tablette tactile.

L'interopérabilité dans l'utilisation d'un système de supervision repose sur la capacité à utiliser une grande variété de matériels informatiques de manière similaire à l'interopérabilité côté serveur, présentée en section 1.3.3.1. L'accès à une application web ne nécessite pas de logiciel supplémentaire, les ordinateurs, téléphones et tablettes possédant tous un navigateur web par défaut, dès lors que le site internet utilise les langages standardisés comme HTML [31], JavaScript [32] ou CSS [33]. Pour répondre aux attentes des développeurs et des utilisateurs, les standards sont en constante évolution et le nombre d'applications web évolue rapidement (voir WAKIL et al. [28]).

1.3.3.4.3 Visualiser

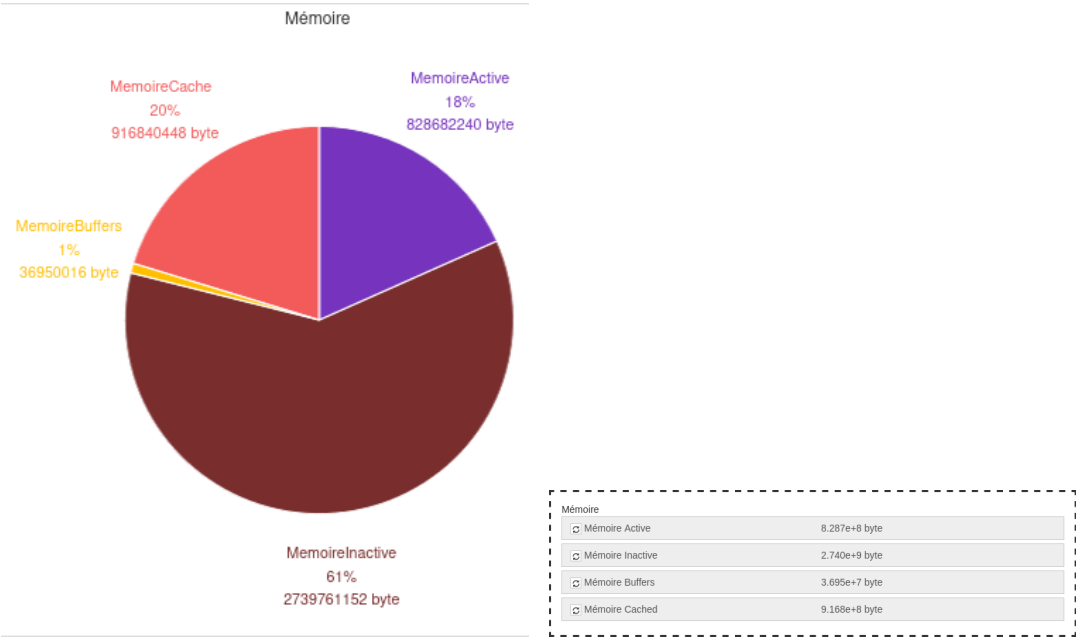
La compréhension des données peut être influencée par la manière de visualiser ces données [34] [35] [36]. Par exemple la figure 1.9 affiche les mêmes variables à gauche dans le camembert et à droite dans le tableau. Le camembert indique en plus la proportion de chaque valeur par rapport à leur somme.

L'interopérabilité pour la visualisation doit permettre d'ajouter de nouveaux visuels et librairies graphiques afin de s'adapter au besoin d'un utilisateur ou d'un secteur d'activité. Par exemple le secteur de l'énergie utilise le diagramme de Sankey afin de mieux comprendre les répartitions de consommations énergétiques (voir figure 1.10).

1.3.3.4.4 Développer

La recommandation N°4 de JIMÉNEZ et al. [19] des logiciels libres pour la recherche demande de définir des règles claires et transparentes de contribution, de communication entre développeurs ainsi que de lister les gestionnaires du logiciel et leurs méthodes de développement du logiciel. Cette recommandation permet à la communauté d'un logiciel libre de travailler ensemble à son amélioration en définissant une stratégie de collaboration.

31. https://html.spec.whatwg.org/multipage/
32. https://262.ecma-international.org/12.0/
33. https://www.w3.org/Style/CSS/current-work
34. https://everydayanalytics.ca/2014/08/stacked-area-graphs-are-not-your-friend.html
35. https://leeoniya.github.io/uPlot/demos/stacked-series.html
36. https://www.vizwiz.com/2011/12/when-you-use-smoothed-line-chart-your.html



(A) Camembert (B) Tableau

FIGURE 1.9 – Affichage de quatre variables de deux manières différentes.

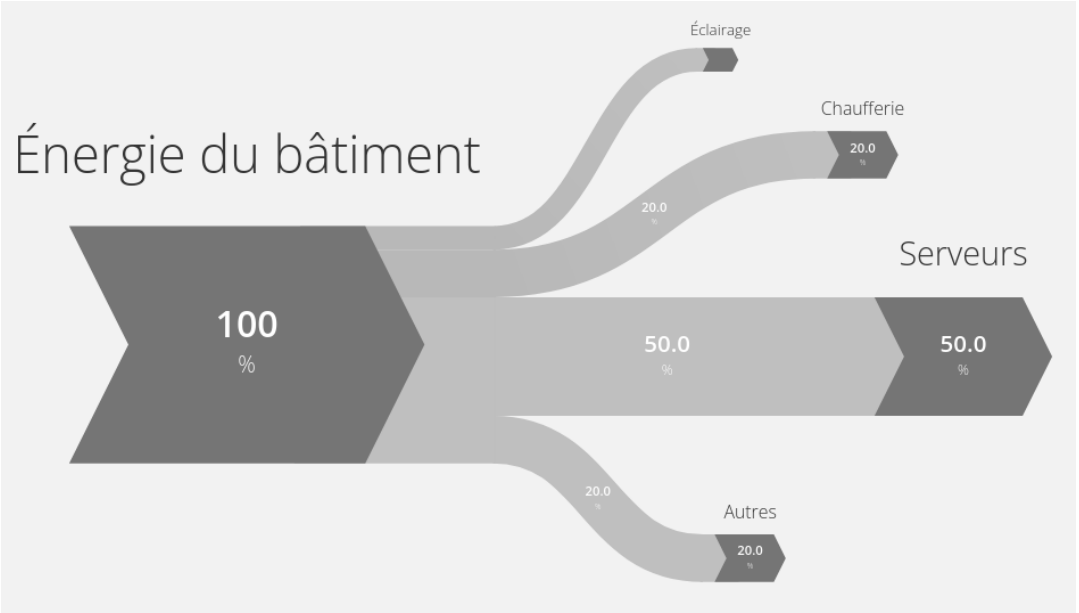


FIGURE 1.10 – Diagramme de Sankey représentant des flux liant des nœuds.

Pour répondre aux différents besoins de visualisation, d'utilisation ou de configuration, l'approche modulaire (en anglais, *plugin*) permet à chaque développeur d'ajouter le module qu'il désire à l'édifice. Avec cette approche, le système doit définir un cadre **fixe**, que chaque module doit prendre en compte et qui ne sera pas modifiable, ainsi qu'un cadre **mobile**, que chacun pourra remplacer ou faire évoluer.

Dans le cas d'un logiciel de supervision et d'instrumentation, le cadre **fixe** peut être l'organisation et la hiérarchie entre protocoles, instruments, variables et unités, alors que le cadre **mobile** inclura les types de graphiques, la liste des protocoles

disponibles et l'intégration à d'autres logiciels.

1.4 Bilan

Dans ce chapitre nous avons présenté les principes de la supervision et de l'instrumentation à distance pour en dégager des problématiques afin de permettre une utilisation par tous, pour tout type de système, évolutive et interopérable. Ces problématiques s'appuient notamment sur les principes du FAIR, qui ont été étendus aux processus de supervision, sur l'évolution du web à travers la généralisation d'identifiants pérennes, sur l'approche philosophique et politique du logiciel libre et sur une abstraction de l'interopérabilité en plusieurs catégories.

Dans le chapitre 2 nous allons présenter les applications existantes pour répondre à ces problématiques en définissant leurs limites selon des besoins identifiés.

Chapitre 2

État de l'art des applications

Sommaire

2.1 Secteur de l'industrie 28
2.1.1 Tango Controls 29
2.1.2 Scada-LTS 29
2.1.3 RapidSCADA 30
2.1.4 Bilan 31
2.2 Secteur de la recherche 31
2.2.1 National Instruments 31
2.2.2 PyMoDAQ 31
2.2.3 PyMeasure 33
2.2.4 Contrôle d'expérimentation au National Synchrotron Light Source-II 34
2.2.5 Yaq 35
2.2.6 Bilan 35
2.3 Secteur de l'enseignement 35
2.3.1 LabsLand : une entreprise proposant d'accéder à des laboratoires distants 36
2.3.2 VISIR : Virtual Instruments Systems in Reality 37
2.3.3 ISES Remote Lab 39
2.3.4 UNILabs 39
2.3.5 RexLab 40
2.3.6 VirtualRemoteLab 41
2.3.7 GOLDi : The Grid of Online Laboratory Devices Ilmenau . . 41
2.3.8 Expériences à la FEUP 42
2.3.9 e-lab IST 43
2.3.10 Stanford iLABS 43
2.3.11 iLAB et LabShare 43
2.3.12 FarLabs 44
2.4 Besoins identifiés 45

Je présente dans ce chapitre différentes applications utilisant des logiciels de supervision ou d'instrumentation dans les secteurs de l'industrie, de la recherche ou de l'enseignement, qu'ils se situent dans les domaines de l'énergie, de l'environnement ou de l'éducation.

Certaines solutions logicielles utilisent un *framework*, qui a pour but d'offrir un cadre de travail aux développeurs en proposant un tronc commun de bibliothèques génériques et personnalisables (aussi appelées librairies). Un *framework* permet des développements rapides, possède une communauté active avec des mises à jour régulières et offre une architecture robuste ainsi que des composants et librairies réutilisables.

Enfin nous exposons les besoins identifiés, ce qui pose la problématique avec les verrous adressés dans cette thèse. L'ensemble des solutions présentées sera comparé selon une liste de critères disponibles dans le tableau 2.2, p. 46.

2.1 Secteur de l'industrie

L'industrie utilise des systèmes informatiques pour piloter et superviser des installations techniques. Ces systèmes peuvent contrôler directement des processus industriels ou se présenter comme une couche intermédiaire entre les systèmes de contrôle et l'homme, ils se concentrent alors sur la supervision. On parle dans ce cas de système de SCADA. Ils sont actuellement capables de gérer jusqu'à un million d'entrées et sorties en parallèle (DANEELS et al. [29]).

Ils existent depuis les années 1960 et la **première génération** se composait de systèmes isolés et non connectés entre eux ; on parle de l'ère **monolithique** (*Technical Information Bulletin* [30]). Les protocoles de communication et les connecteurs développés par les fabricants étaient souvent propriétaires et avaient pour unique but de communiquer avec les terminaux distants (RTU de l'anglais *Remote Terminal Unit*).

La **seconde génération** de systèmes SCADA est apparue avec l'émergence des réseaux locaux (LAN de l'anglais *Local Area Network*). Cette architecture distribuée consistait en plusieurs ordinateurs communiquant ensemble et remplissant la fonction de :

— communication avec les RTU,
— station fournissant une IHM à des opérateurs,
— station de calculs ou de stockage.

La répartition des tâches entre plusieurs machines permettait d'accroître la puissance de calcul mais ces machines restaient accessibles uniquement dans le réseau local de l'installation. Certains fabricants développaient leurs protocoles propriétaires LAN afin d'optimiser les échanges en temps réel mais limitaient ou éliminaient la possibilité de se connecter au matériel d'autres fabricants. La distribution des fonctionnalités sur plusieurs ordinateurs augmentait la redondance et la stabilité du système. La communication avec les RTU n'avait pas changé et reposait toujours sur des protocoles dédiés. Comme pour la première génération, le matériel et le logiciel étaient limités à ceux fournis par un fabricant.

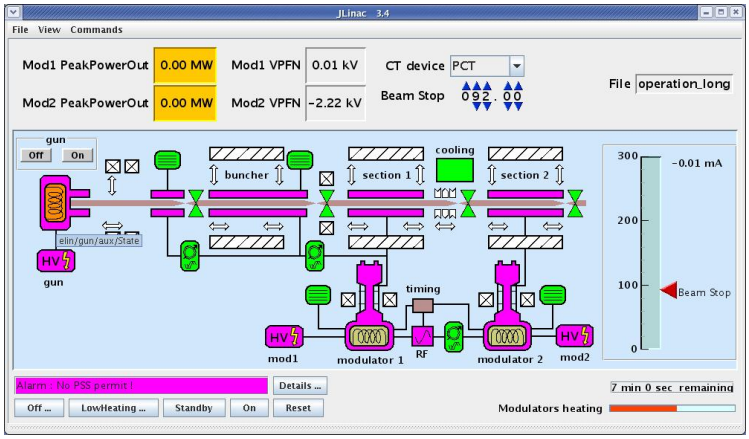
La **troisième génération**, dite en réseau, se différencie de la seconde par une architecture de systèmes ouverts plutôt que propriétaires et contrôlés par un fabricant. Cette ouverture facilite la connexion de dispositifs tiers au système et au réseau, comme des imprimantes, des espaces de stockage, des écrans. La principale nouveauté est l'apparition du protocole internet (IP de l'anglais *Internet Protocol*) dans la communication entre le serveur et les équipements, permettant de communiquer au travers de connections Ethernet. La distribution des équipements à travers divers réseaux locaux améliore la robustesse d'un système, la chute d'un réseau local n'entraînant pas la chute de tout le système.

La **génération actuelle** tend à remplacer les réseaux locaux distribués d'une entreprise par des objets tous connectés à internet (IoT) et où la gestion est délocalisée sur des serveurs accessibles par internet (informatique en nuage, *cloud computing* en anglais).

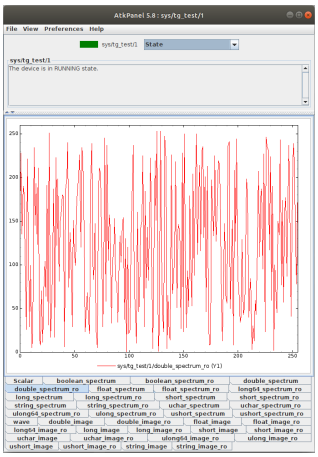
Je présente dans la suite des systèmes SCADA libres et ouverts disponibles et actuellement en développement.

2.1.1 Tango Controls

Tango Controls [1] est un outil permettant de construire un réseau distribué de systèmes contrôlés. Il peut être utilisé comme un protocole pour contrôler des instruments à distance. Il nécessite Java [2] pour être utilisé et requiert l'utilisation de logiciels tels que *Jive*, *Astor* ou *ATKPanel* pour créer un instrument ou une interface de contrôle [3] (voir figure 2.1). Il est recommandé de l'utiliser avec une base de données. Tango Control ne possède pas d'IHM web accessible facilement et permettant la configuration et l'utilisation des instruments connectés. Il est possible de passer par des applications tierces pour créer des interfaces web communiquant avec Tango Controls. Il n'utilise pas d'infrastructure logicielle (en anglais *framework*).



(A) Interface synoptique construite via l'éditeur graphique JDraw de ATKPanel et connectée à des instruments de Tango Controls.



(B) Visualisation de variables de Tango Controls avec ATKPanel.

FIGURE 2.1 – Tango Controls requiert l'utilisation de logiciels tiers pour construire une interface.

2.1.2 Scada-LTS

Scada-LTS [4] est un logiciel de SCADA *open source* permettant de construire des interfaces web pour dialoguer avec des instruments. Il est écrit en Java et ne nécessite pas d'installation sur les machines clientes accédant à l'IHM. L'exemple fourni par Scada-LTS [5] demande des notions en HTML et CSS afin de construire une interface web (voir figure 2.2). Il est difficile de construire des interfaces complexes et la configuration est inaccessible aux personnes n'ayant pas des bases de programmation.

Scada-LTS n'étant pas developpé avec un *framework* web tel que Spring [6], il apparaît difficile d'intégrer des applications extérieures, comme des applications d'authentification par exemple (CAS [7], OAuth [8]).

1. https://www.tango-controls.org
2. https://www.java.com/fr/download/help/whatis_java.html
3. https://tango-controls.readthedocs.io/en/latest/getting-started/index.html
4. http://scada-lts.com
5. https://github.com/SCADA-LTS/Scada-LTS/tree/examples/doc/examples/home
6. https://spring.io
7. https://docs.spring.io/spring-security/reference/servlet/authentication/cas.html
8. https://docs.spring.io/spring-security/reference/servlet/oauth2/index.html

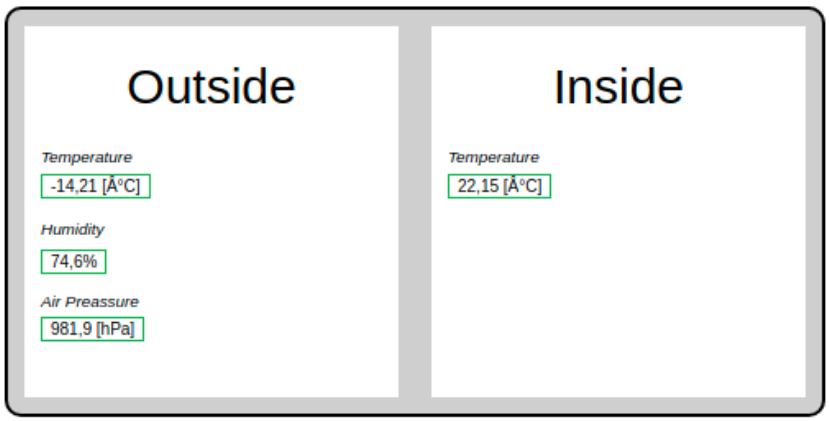


FIGURE 2.2 – Résultat de l'exemple d'interface codée en HTML et proposée par Scada-LTS.

2.1.3 RapidSCADA

RapidSCADA [9] est un logiciel ouvert écrit en *C#* et proposant une interface web pour l'automatisation de processus industriels (voir figure 2.3). Il est originellement fait pour communiquer avec des instruments utilisant le protocole **Modbus** ou **OPC**.

Des *plugins* permettent de communiquer dans d'autres protocoles. Ils ne sont pas tous libres et certains sont payants [10].

RapidSCADA n'utilise pas de *framework* logiciel, ne possède pas de système de gestion de contenu pour la création d'IHM et ne permet pas la création de scripts.

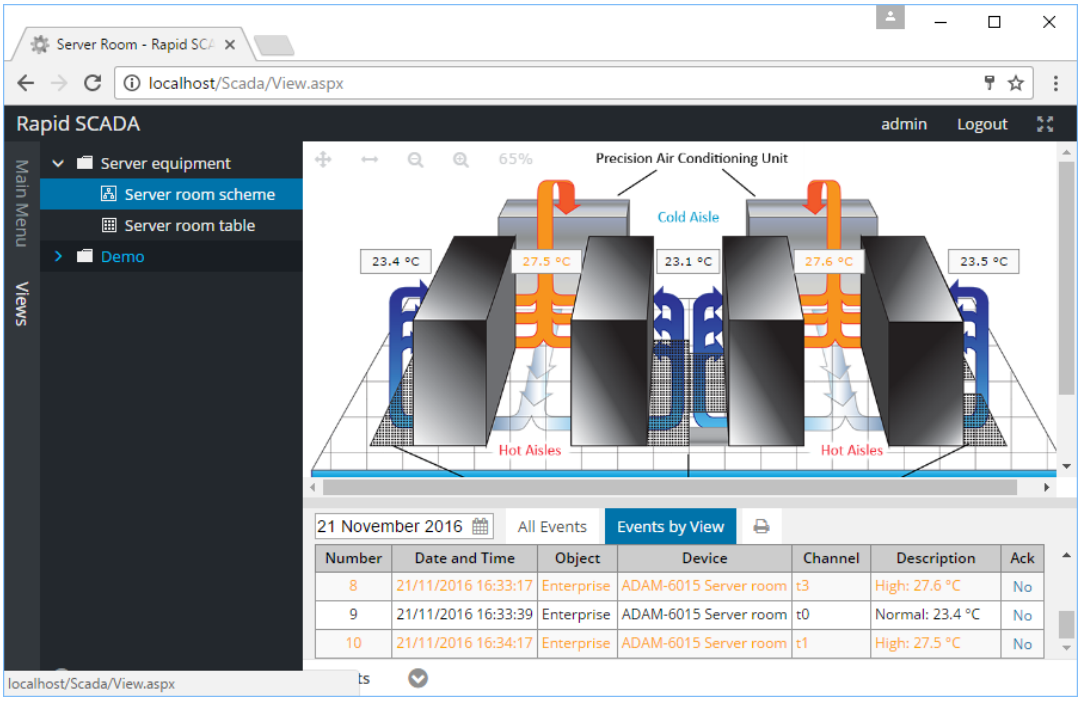


FIGURE 2.3 – Exemple d'interface d'un processus industriel avec RapidSCADA. Chaque page affiche un seul type de visualisation.

9. https://rapidscada.org
10. http://rapidscada.net/scada/plugins/Store/StorePublic.aspx

2.1.4 Bilan

Aucun des logiciels présentés n'utilise de *framework* logiciel permettant de faire évoluer simplement et rapidement la solution proposée. La création d'interfaces utilisateur est complexe ou limitée à certains types de contenu et peut nécessiter de passer par des applications tierces. Les protocoles supportés sont limités et requièrent dans le cas de RapidSCADA l'achat de modules supplémentaires.

2.2 Secteur de la recherche

Les laboratoires de recherche utilisent des instruments pour collecter des données sur les expérimentations qu'ils réalisent. Pour faciliter et automatiser la prise de mesure ainsi que stocker les mesures afin de les analyser, les chercheurs ont développé des solutions en utilisant des logiciels payants ou en collaborant autour de librairies libres.

2.2.1 National Instruments

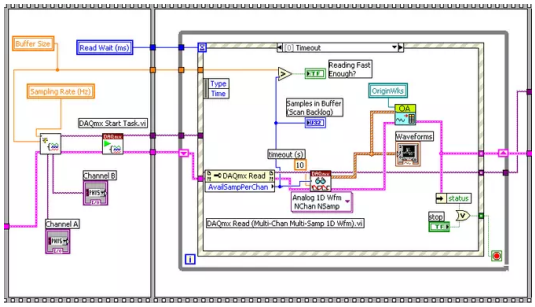
Dans le domaine de la recherche, de nombreux laboratoires utilisent le matériel proposé par *National Instruments* [11] (NI) et son logiciel de programmation graphique *LabVIEW* [12] pour *Laboratory Virtual Instrument Engineering Workbench* (CHACÓN et al. [31], GRÖBER et al. [32], TAWFIK et al. [33], BHUTADA et al. [34], GANGREKAR et al. [35] et SÁENZ et al. [36]). Le matériel de NI permet de connecter un grand nombre de capteurs et d'actionneurs. LabVIEW n'est pas un logiciel libre et ouvert. Il a besoin d'être installé sur un ordinateur, demande des ressources importantes pour s'exécuter et se définit comme **une programmation graphique pour le test et la mesure**. Son prix pour l'installation complète s'élève à plusieurs milliers d'euros. Il permet de réaliser rapidement des interfaces pour un usage particulier mais il est difficile de générer des applications complexes et génériques (voir figure 2.4). LabVIEW propose une interface web pour ses applications qui peut demander l'installation de *plugins* et qui nécessite la dernière version pour être compatible sur tous les navigateurs web récents. L'utilisation de LabVIEW dans les laboratoires de recherche amène souvent les utilisateurs à concevoir des applications nouvelles dédiées à leurs besoins et difficilement évolutives ou adaptables à d'autres publics.

2.2.2 PyMoDAQ

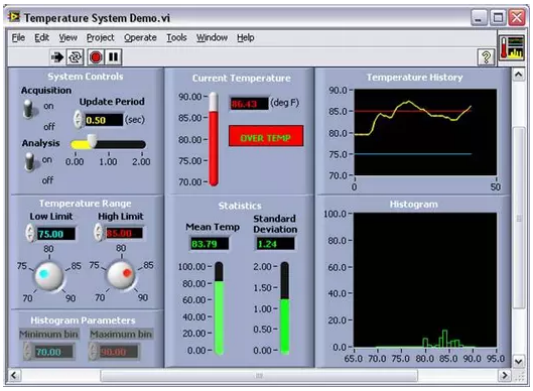
PyMoDAQ [13], pour *Modular Data Acquisition with Python*, est un logiciel développé par Sébastien Weber du laboratoire CEMES (Toulouse) au CNRS qui permet de connecter simplement divers instruments (détecteurs et actionneurs) pour réaliser de l'acquisition de données via des modules en Python. PyMoDAQ se compose de divers modules (voir figure 2.5) :

- DAQ Move pour contrôler l'ensemble des variables d'une expérimentation,
- DAQ Viewer pour visualiser l'ensemble des paramètres d'une expérimentation ainsi que les données enregistrées,
- DAQ Scan pour automatiser l'acquisition de données en définissant l'ensemble des paramètres d'un enregistrement ainsi qu'un fichier de sortie des données au format HDF5,

11. https://www.ni.com
12. https://www.ni.com/fr-fr/shop/labview.html
13. http://pymodaq.cnrs.fr



(A) Interface de programmation graphique de LabVIEW (*Diagram*) parfois difficile à comprendre dans le cas de grandes interfaces.



(B) Interface d'utilisation de LabVIEW (*Front panel*) à reconstruire pour chaque projet.

FIGURE 2.4 – Logiciel de test et de mesure LabVIEW disposant de deux interfaces : programmeur et utilisateur.

— DAQ Logger pour enregistrer des données temporelles de plusieurs détecteurs.

Il est possible de rajouter de nouvelles fonctionnalités via des *plugins*.

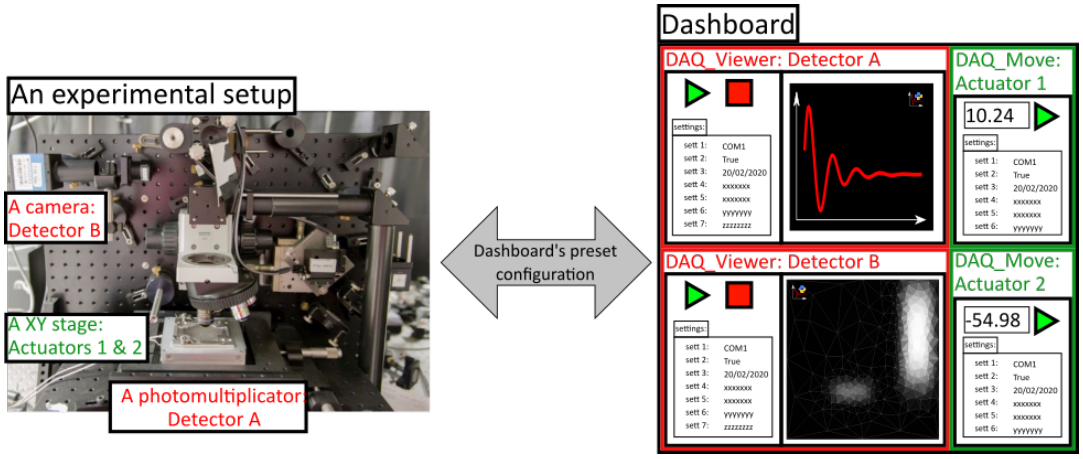


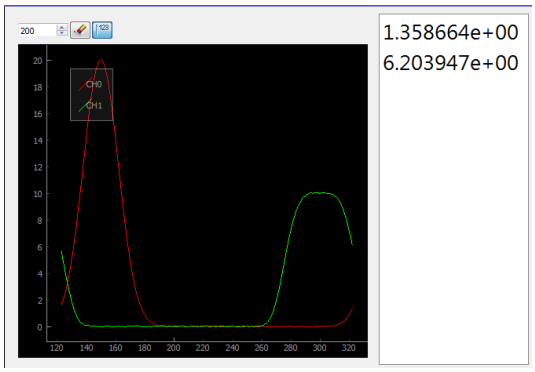
FIGURE 2.5 – Interface d'une installation via les modules *Move* et *Viewer* de PyMoDAQ.

La figure 2.6 montre l'interface de PyMoDAQ permettant de visualiser des données en plusieurs dimensions, allant de zéro à quatre. L'affichage en *0D* consiste à afficher une ou plusieurs variables évoluant avec le temps. Les affichages en *1D*, *2D* ou *ND* permettent d'afficher des données figées dans le temps et variant selon une ou plusieurs dimensions (par exemple la fréquence ou la longueur d'onde).

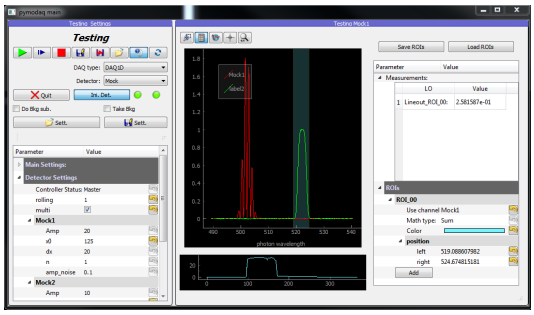
PyMoDAQ ne permet pas de construire simplement une interface à l'aide d'un système de gestion de contenu (SGC). Il ne propose pas d'interface web et nécessite l'installation du logiciel sur chaque machine voulant l'utiliser.

Bien qu'il propose une liste de *plugins* permettant l'utilisation de divers instruments [14], il ne propose pas d'ajouter un instrument connaissant son protocole et la configuration requise pour communiquer avec lui. PyMoDAQ nécessite la création de *plugins* pour réaliser une connexion PnP pour chaque nouvel instrument.

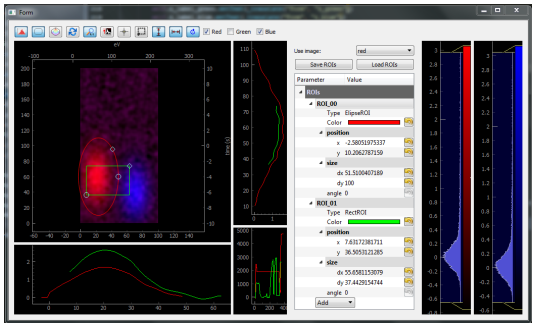
14. https://github.com/CEMES-CNRS/pymodaq_plugin_manager



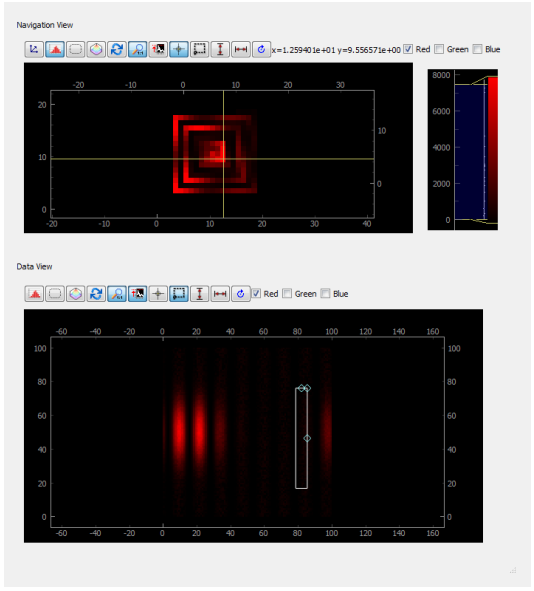
(A) Vue temporelle *0D*.



(B) Vue *1D*.



(C) Vue *2D*.



(D) Vue *ND*.

FIGURE 2.6 – Représentations des données selon différentes dimensions.

2.2.3 PyMeasure

PyMeasure est un projet créé par l'Université de Cornell en 2013 [15] ayant pour but de communiquer avec des instruments scientifiques respectant l'API Virtual instrument software architecture (VISA). Il permet de communiquer avec des instruments connectés en liaison série (RS232), GPIB, Ethernet ou USB. PyMeasure fournit une interface de haut niveau permettant de cacher les commandes de bas niveau utilisant la syntaxe *Standard Commands for Programmable Instruments* (SCPI). Par exemple la commande SCPI **MEASure :VOLTage** demande à l'instrument de lire une tension. Cette commande est remplacée dans PyMeasure par la fonction **measure_voltage**.

L'automatisation d'enregistrement se fait en programmant en Python des **procédures** détaillant les étapes d'une expérimentation à l'aide de paramètres nécessaires à sa réalisation. La figure 2.7 présente l'interface graphique qui n'est pas directement accessible à distance.

15. https://pymeasure.readthedocs.io

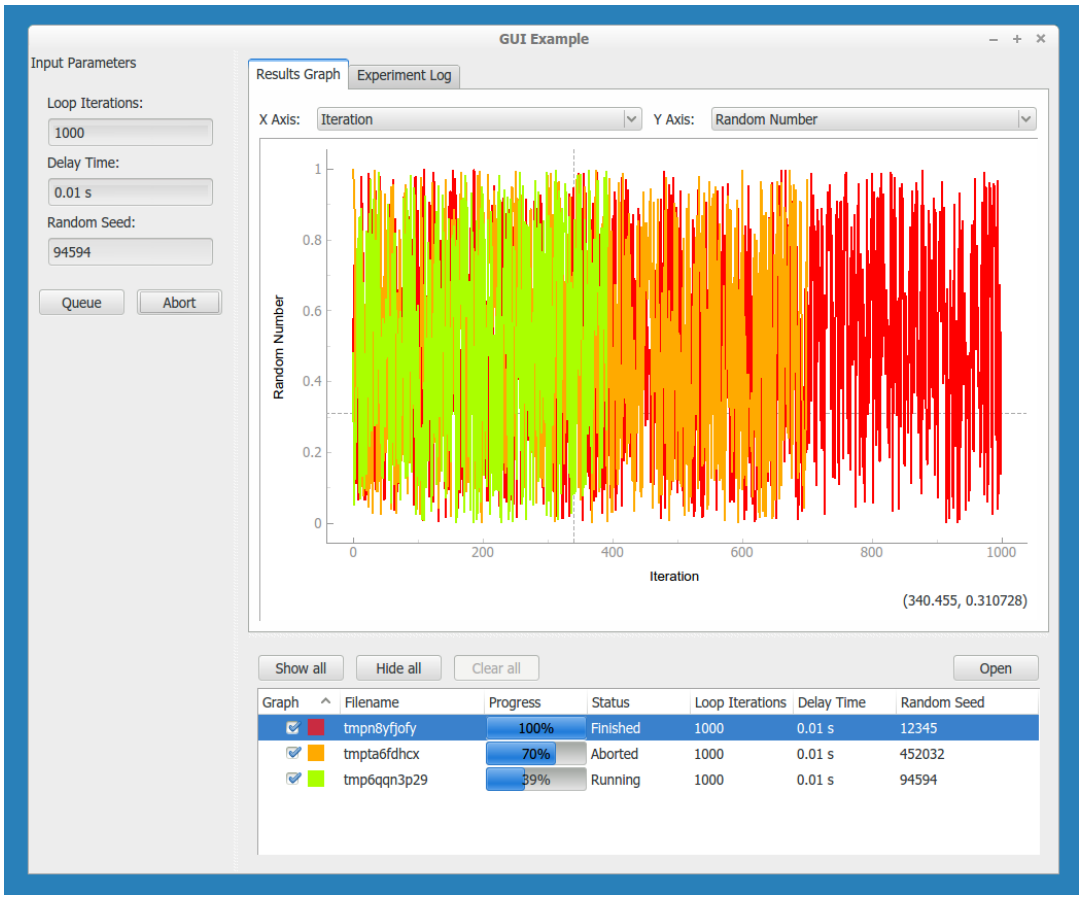


FIGURE 2.7 – Interface de PyMeasure définie par une procédure avec des données d'entrée et de sortie.

2.2.4 Contrôle d'expérimentation au National Synchrotron Light Source-II

Les logiciels développés par NSLS-II [16] sont une suite d'outils pour l'acquisition, la gestion et l'analyse de données scientifiques. Cette suite logicielle se décline en 3 parties représentées en figure 2.8 :

— Bluesky permet de définir des procédures (similaires à PyMeasure) et d'acquérir les données à l'aide de fonctions de haut niveau. Il communique avec Ophyd et Databroker.
— Ophyd configure les instruments permettant à Bluesky de communiquer avec eux. Il crée une couche d'abstraction des fonctions de bas niveau.
— Databroker sauvegarde et donne l'accès aux données des enregistrements.

Bluesky nécessite de coder en Python des procédures qui déclarent les capteurs et les actionneurs, définissent les séquences d'acquisition, utilisent une base de données pour sauvegarder les données et déclarent des fonctions pour traiter les données (affichage, stockage...). Bluesky ne posséde pas d'interface d'utilisation graphique par défaut, chacun devant se créer sa propre interface.

Ophyd est la couche d'abstraction matérielle offrant une approche générique permettant de communiquer avec une grande variété d'instruments, principalement en utilisant la librairie EPICS [17] qui propose une infrastructure pour les systèmes de contrôle distribués.

16. https://nsls-ii.github.io
17. https://epics-controls.org

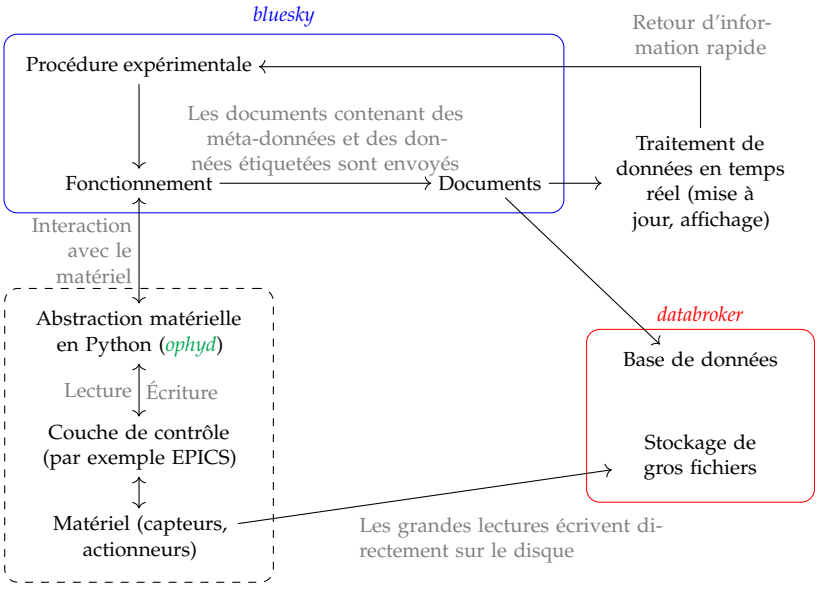


FIGURE 2.8 – Organisation des librairies de NSLS-II en trois groupes.

2.2.5 Yaq

Yaq [18] est un autre logiciel créant une couche d'abstraction logicielle permettant d'offrir des fonctions de haut niveau pour communiquer avec des instruments scientifiques. Yaq se veut **minimaliste** et offre une interface en ligne de commandes. Chaque instrument est configuré à l'aide d'un fichier.

Yaq possède des passerelles vers des clients tels que Bluesky et s'organise autour d'un protocole permettant à chaque passerelle ou processus gérant un instrument de dialoguer ensemble. Les données transmises via ce protocole utilisent le format de codage des données Avro RPC [19]. Tout programme implémentant le protocole Yaq peut dialoguer avec d'autres instruments ou clients de l'écosystème Yaq.

2.2.6 Bilan

Les besoins recensés dans le secteur de la recherche sont d'avoir un système *open source*, de permettre un accès distant aux systèmes instrumentés et de fournir une interface évolutive et adaptable aux besoins de chaque utilisateur et expérimentation. Les anciens instruments utilisés communiquent très souvent via les ports série et GPIB et pour les plus récents via les ports USB ou Ethernet. Certains logiciels proposent des fonctions de haut niveau pour cacher la complexité et les différences entre les instruments communiquant via le protocole VISA et utilisant les commandes SCPI.

2.3 Secteur de l'enseignement

Les laboratoires à distance pour l'éducation sont en développement sur tous les continents (GARCIA-ZUBIA [37]). L'un des principaux objectifs des laboratoires à distance est d'étendre les possibilités d'expérimentation pratique dans les formations universitaires en sciences, technologie, ingénierie et mathématiques (STEM de l'anglais *Science, Technology, Engineering, and Mathematics*). En effet, le profil des

18. https://yaq.fyi
19. https://avro.apache.org/

étudiants de l'enseignement supérieur est diversifié et est désormais bien adapté aux nouvelles technologies de l'information. De plus, comme le montre TIKHONOVA et al. [38], la massification des étudiants dans l'enseignement supérieur a entraîné des changements tels que les techniques d'apprentissage à distance par l'utilisation de cours en ligne ouverts et massifs (MOOC de l'anglais *Massive Open Online Course*), l'apprentissage en ligne et la simulation de systèmes physiques. Pour résoudre ce problème, il est essentiel de créer de nouveaux outils pour les enseignants et les apprenants, tels que des laboratoires à distance intégrés dans des scénarios de type jeu sérieux (LUTHON et al. [2]).

L'accessibilité 24 heures sur 24, l'amélioration des processus d'apprentissage et les exemples pratiques pour l'enseignement en classe font partie des nombreux avantages que les laboratoires à distance apportent aux enseignants et aux étudiants. Les laboratoires à distance sont donc bien adaptés à l'expérience dynamique en ligne des étudiants d'aujourd'hui mais il peut aussi être utilisé en classe pour illustrer le cours. Un laboratoire à distance est différent d'un laboratoire virtuel ; le laboratoire virtuel n'étant qu'un modèle informatique d'un système physique, moins coûteux et plus facile à mettre en œuvre et à partager puisqu'il n'utilise que la simulation (SÁENZ et al. [36]). Depuis plus de vingt ans, de nombreuses publications attestent de l'utilisation massive des laboratoires à distance comme technologie d'apprentissage alternative mais non exclusive (AKTAN et al. [39]).

Parmi tous les laboratoires et expériences à distance répertoriés dans GARCIA-ZUBIA [37], seul un petit nombre utilise des logiciels libres (voir tableau 2.1). Cependant, il a été observé qu'il existe une tendance à utiliser des logiciels à code source ouvert pour améliorer la normalisation et la simplicité d'utilisation (VILLAR-MARTÍNEZ et al. [40] et BERMÚDEZ-ORTEGA et al. [41]).

Les logiciels développés répondent aux mêmes besoins que tout logiciel de supervision et d'instrumentation à distance avec certaines fonctionnalités spécifiques à l'enseignement. Il est par exemple nécessaire de gérer une **file d'attente** afin de séparer les données de chaque étudiant. Certains laboratoires peuvent autoriser plusieurs étudiants à se connecter simultanément pour **travailler en groupe** ; dans ce cas un seul utilisateur doit avoir la main sur l'expérience, les autres utilisateurs restant spectateurs. En dehors du laboratoire distant pour l'éducation que nous avons développé, **Laborem**, aucun autre laboratoire distant présenté ci-dessous n'utilise un logiciel libre et générique de supervision ou d'instrumentation à distance. Ils ne proposent pas différents niveaux de configuration adaptable à l'utilisateur (enseignant ou étudiant). Ils ne possèdent pas de SGC, n'utilisent pas de *framework* logiciel et ne permettent pas d'ajouter des scripts, des évènements ou des alertes via une interface d'administration (voir tableau 2.2).

2.3.1 LabsLand : une entreprise proposant d'accéder à des laboratoires distants

Labsland [20] est issue de WebLab Deusto [21] de l'université de Deusto en Espagne. Il regroupe maintenant plus de 30 expériences situées dans 24 universités dans le monde. C'est le projet le plus abouti en terme de fédération de laboratoires distants pour l'éducation (ORDUÑA et al. [42]). Il propose entre autres des expériences différées (donc pas en temps réel) où l'étudiant voit un enregistrement de l'expérience, pensant qu'il est en train de manipuler des équipements. Cette approche permet de ne pas

20. www.labsland.com
21. https://weblab.deusto.es

TABLE 2.1 – Comparaison des développements des laboratoires distants listés par Garcia-Zubia.

Laboratoire distant	**Logiciel *open source***	**Matériel *open source***
UNILabs	Oui [a]	Non
WebLab-Deusto LabsLand	Oui [b]	Non
RexLab	Oui [c]	Non
ISES RemLabNet	Non	Non
VirtualRemoteLab	Non	Non
GOLDi	Non	Non
OE@FEUP HTML/LabVIEW	Non	Non
e-lab IST	Non	Non
Stanford iLABS	Non	Non
iLAB LabShare	Non	Non
FarLabs	Non	Non
Laborem	Oui [d]	Oui [e]

a. https://github.com/UNEDLabs
b. https://github.com/weblabdeusto
c. https://github.com/RExLab
d. https://github.com/clavay/pyscada-laborem
e. https://doi.org/10.5281/zenodo.5564369

mettre en place de file d'attente et augmente les capacités de connexions simultanées d'étudiants.

Par exemple l'expérience **Archimedes** permet d'analyser les caractéristiques d'un objet et d'un liquide en lançant un objet dans un tube de test et en regardant s'il flotte (voir figure 2.9). Les variables de flottabilité sont contrôlées par l'étudiant.

WebLab Deusto propose un outil [22] libre pour administrer des laboratoires distants (gestion et suivi des utilisateurs, des permissions, des calendriers...) mais il ne propose pas d'outil de création de laboratoire distant.

2.3.2 VISIR : Virtual Instruments Systems in Reality

VISIR est un projet issu de Blekinge Institute of Technology (BTH) en Suède (TAWFIK et al. [33]). Il offre la possibilité à un étudiant d'assembler un circuit électrique comprenant par exemple des résistances et des condensateurs, de le soumettre à des signaux fournis par un générateur de fonctions basse fréquence (GBF) et d'analyser le comportement du circuit à l'aide d'un multimètre et d'un oscilloscope. L'étudiant peut changer les paramètres des différents instruments à l'aide d'une photo représentant les boutons des instruments et mimant leur état.

VISIR est un des laboratoires distants les plus utilisés (GARCIA-ZUBIA [37]). Il autorise 60 étudiants à se connecter en simultané grâce à sa matrice de relais qui permet de multiplexer les signaux et de connecter les éléments du circuit désiré sans intervention humaine (voir figure 2.10).

VISIR est depuis sa création lié au matériel de National Instruments (NI) et son logiciel LabVIEW. Il ne propose pas de logiciel ni de matériel libre.

22. https://github.com/weblabdeusto/weblabdeusto

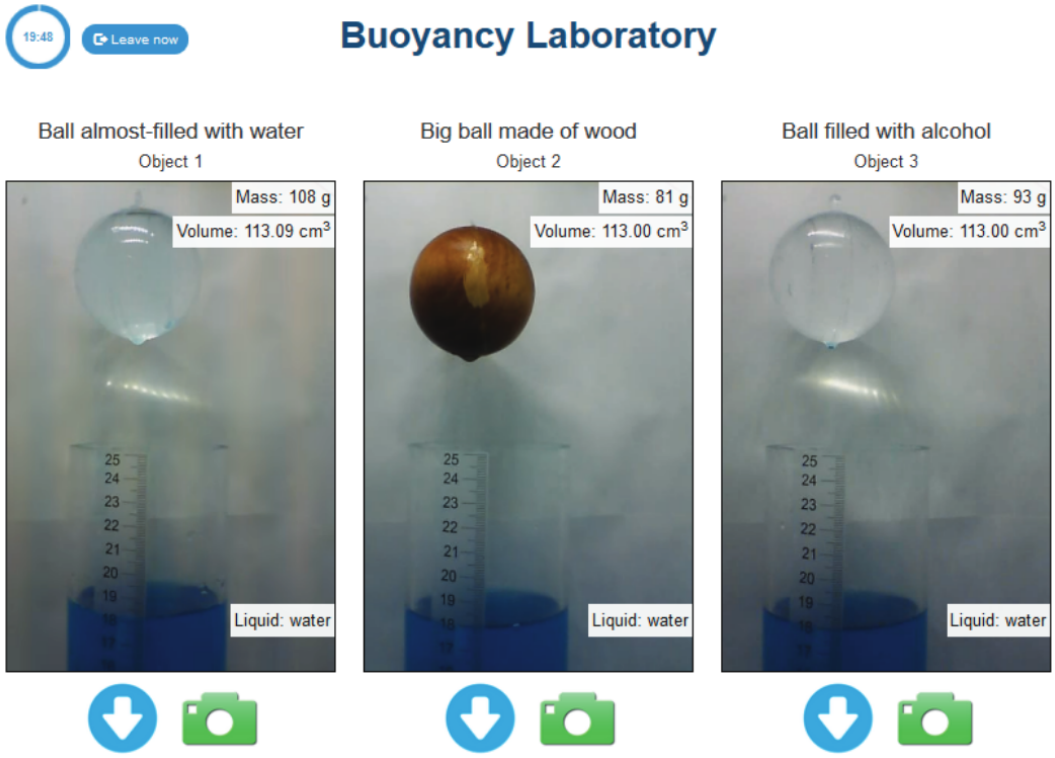


FIGURE 2.9 – Expérience Archimedes de Labsland pour l'étude du déplacement d'un objet dans un liquide. Ces images sont des enregistrements, l'étudiant ne manipulant pas directement de système physique.

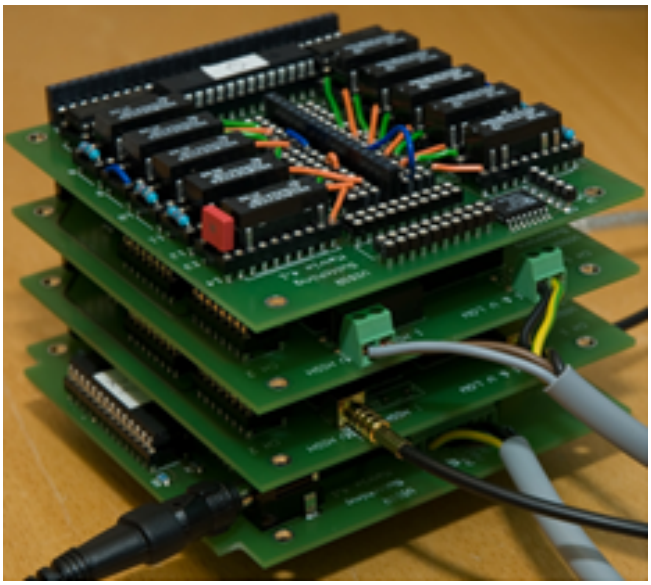


FIGURE 2.10 – Matrice de relais de VISIR permettant de connecter des éléments selon le schéma du circuit réalisé par l'étudiant.

2.3.3 ISES Remote Lab

ISES est géré par la faculté de mathématiques et de physique de Charles University en République Tchèque. Il propose environ 20 expériences de différents domaines de la physique.

Par exemple il propose une expérience d'induction électromagnétique [23] permettant à l'étudiant de contrôler la tension d'entrée et la bobine utilisée. L'expérience se présente à l'aide d'un retour vidéo et d'un graphique (voir figure 2.11). Les données sont exportables sous forme de fichier CSV.

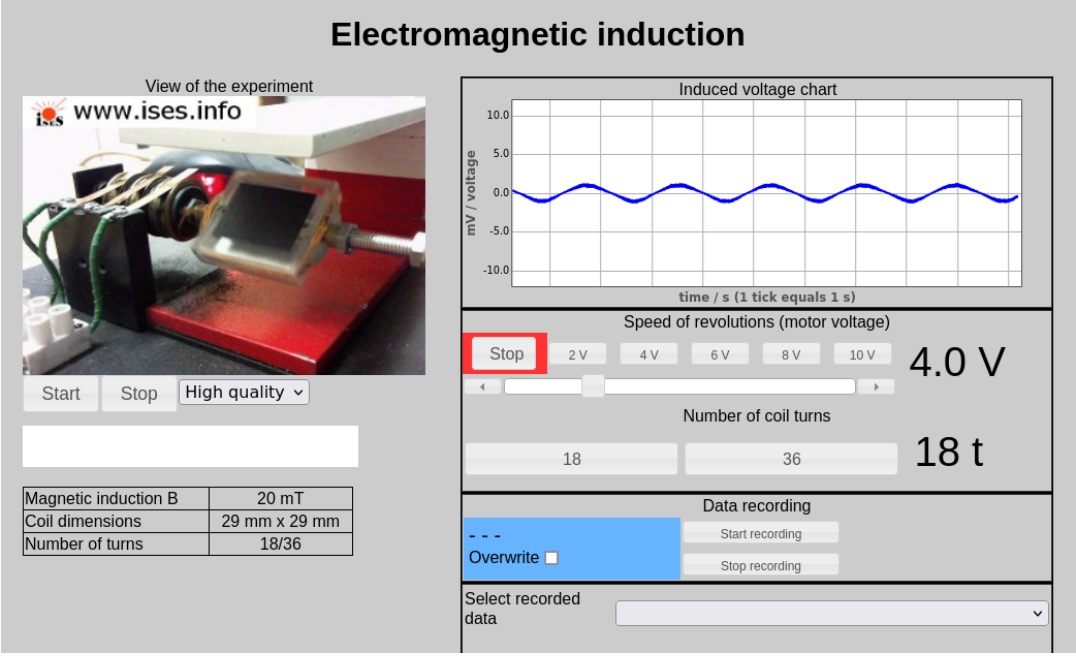


FIGURE 2.11 – Expérience d'induction électromagnétique d'ISES.

2.3.4 UNILabs

UNILabs propose 14 expériences [24]. Ces expériences de contrôle de systèmes et de processus sont complexes et coûteuses. Le laboratoire distant est souvent accompagné par un simulateur qui permet de s'entraîner en étant hors ligne.

La figure 2.12 présente un système de quatre réservoirs utilisés pour l'apprentissage de la commande de processus multivariable. Chaque réservoir possède une sortie de section connue et une autre de section inconnue, régulée au moyen d'une vanne. Le système peut également réguler le débit de liquide provenant de deux pompes entrant dans chacun des réservoirs à l'aide de deux vannes à trois voies. Ce laboratoire permet d'étudier des caractéristiques dynamiques du système, de modéliser le système par des techniques d'identification et de concevoir un contrôleur multivariable pour réguler le niveau d'un réservoir.

La figure 2.13 présente la version simulée de ce laboratoire.

UNILabs propose des logiciels libres [25] pour intégrer les laboratoires distants au système de gestion de l'apprentissage libre **Moodle** [26] (LMS en anglais pour *Learning*

23. http://kdt-39.karlov.mff.cuni.cz/index_en.html
24. https://unilabs.dia.uned.es/blog/index.php?entryid=3
25. https://github.com/UNEDLabs
26. https://moodle.org/

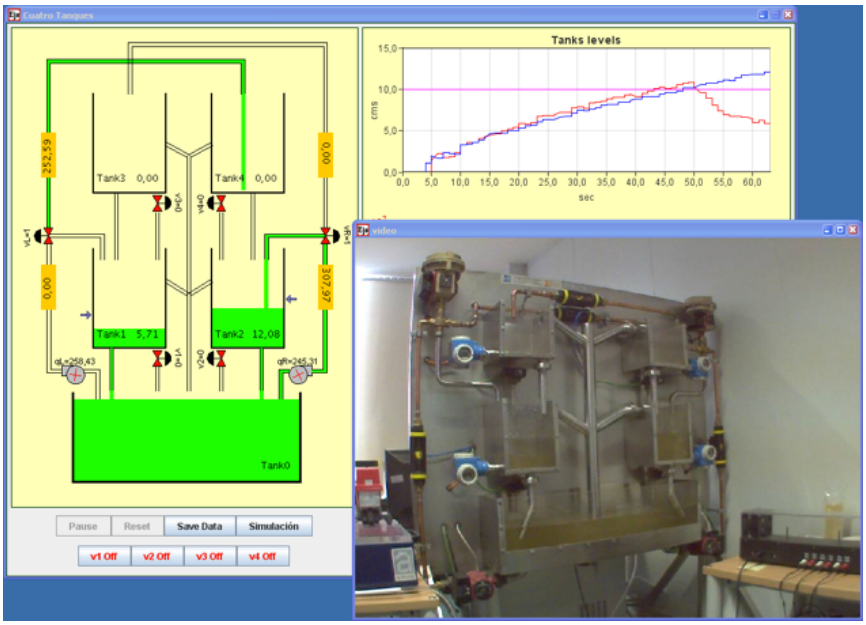


FIGURE 2.12 – Expérience de caractérisation de systèmes dynamiques multivariable d'UNILabs.

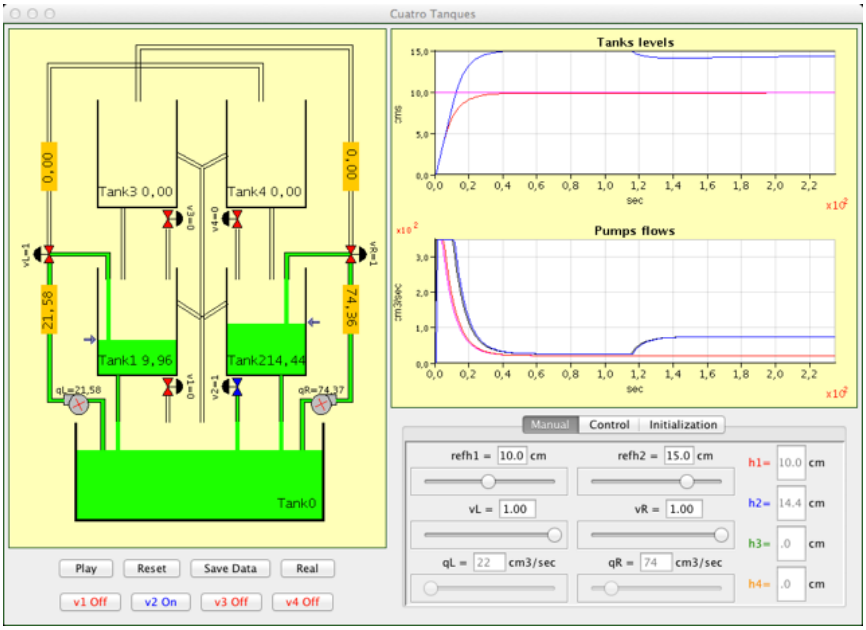


FIGURE 2.13 – Version virtuelle de l'expérience à quatre réservoirs contrôlés d'UNILabs.

Management System) et via le protocole d'interopérabilité distante pour les laboratoires distants défini par CHACÓN et al. [43]. Il nécessite l'utilisation de Java [27].

2.3.5 RexLab

RexLab [28] propose quinze laboratoires distants, dont **VISIR**. Huit nouvelles expériences sont actuellement en phase de préparation. RexLab propose des expériences à

27. https://unilabs.dia.uned.es/mod/data/view.php?d=3&advanced=0&paging&page=0
28. https://rexlab.ufsc.br/en/

distance depuis 2008 et fait partie des références internationales. Les expériences sont déployées à l'Université Fédérale de Santa Catarina au Brésil.

Rexlab met à disposition son logiciel [29] de gestion de laboratoires distants ainsi qu'une documentation permettant de mettre en place par exemple l'expérience de l'étude des associations série, parallèle et mixte dans les réseaux électriques à courant alternatif (voir figure 2.14).

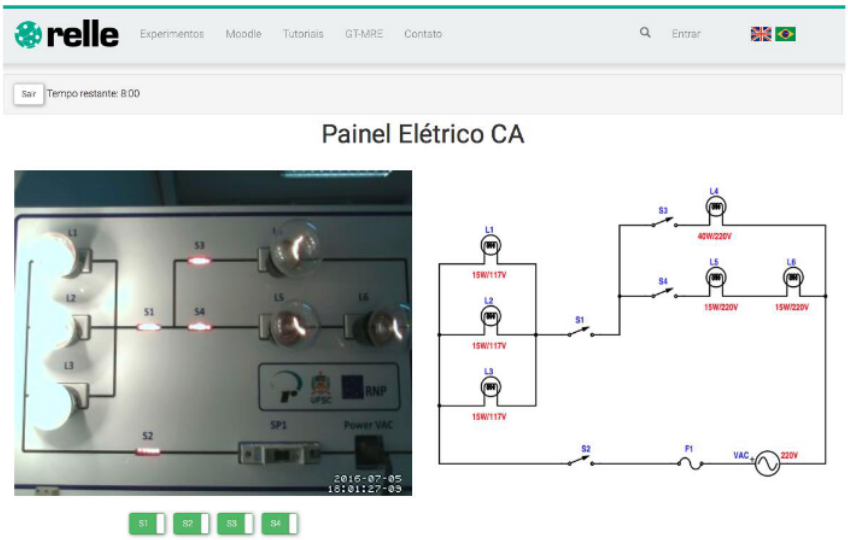


FIGURE 2.14 – Etude des réseaux à courant alternatif chez RexLab.

2.3.6 VirtualRemoteLab

VirtualRemoteLab est hébergé à l'Université de Munich et propose des laboratoires distants depuis plus de quinze ans (GRÖBER et al. [32]), notamment dans le domaine de la spectrométrie [30]. Sur l'exemple de la figure 2.15, il permet de caractériser différentes lampes en étudiant le spectre lumineux mesuré. Son interface web est accessible depuis tout ordinateur, mais elle est dédiée aux expériences proposées par VirtualRemoteLab et ne peut pas s'adapter facilement à tout type d'expériences.

2.3.7 GOLDi : The Grid of Online Laboratory Devices Ilmenau

GOLDi est un projet de l'Université Technique de Ilmenau [31] en Allemagne. Il offre des laboratoires distants et des laboratoires de simulation. L'étudiant peut programmer en C, assembleur ou VHDL des micro-contrôleurs, des réseaux de portes programmables in situ (FPGA en anglais pour *Field-Programmable Gate Array*) et des automates programmables industriels (API). Il connecte ensuite ces instruments programmables à des systèmes d'ascenseurs (voir figure 2.16) ou des systèmes de contrôle de niveau d'eau pour que l'étudiant puisse vérifier son code et l'améliorer. GOLDi est une référence dans les expériences distantes de systèmes programmables. Cependant son interface ancienne et non sécurisée n'a pas évolué avec les standards du web.

29. https://github.com/RExLab/relle
30. https://www.didaktik.physik.uni-muenchen.de/remotelab/index.php?lang=en&page=1
31. https://www.goldi-labs.net/

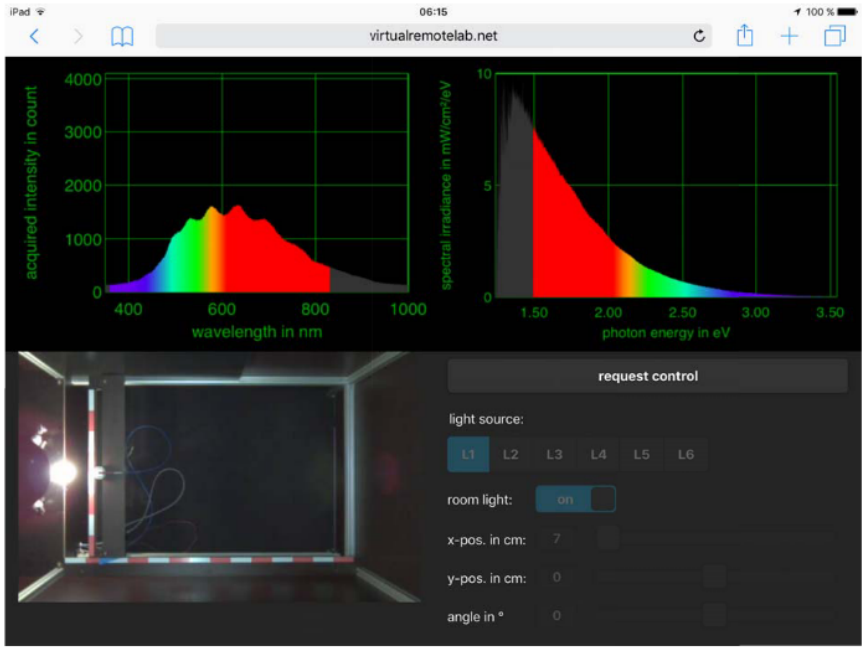


FIGURE 2.15 – Analyse du spectre de différentes lampes sur Virtual-RemoteLab.

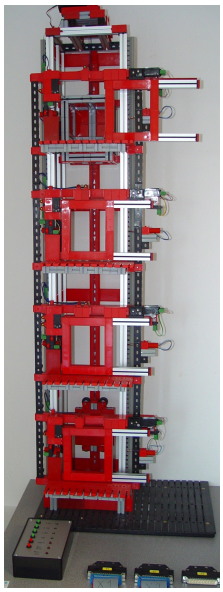


FIGURE 2.16 – Ascenseur à 4 étages programmable sur GOLDi.

2.3.8 Expériences à la FEUP

La Faculté d'Ingénierie de l'Université de Porto (FEUP) propose des expériences à distance dont certaines utilisent LabVIEW et d'autres sont accessibles en HTML [32]. Les expériences utilisant LabVIEW nécessitent un *plugin* et ne fonctionnent pas parfaitement avec tous les navigateurs web. Sur la figure 2.17 l'étudiant peut contrôler une lampe et un ventilateur à l'aide d'interrupteurs pour étudier l'évolution de la température, de l'humidité et de la luminosité.

32. https://remotelab.fe.up.pt

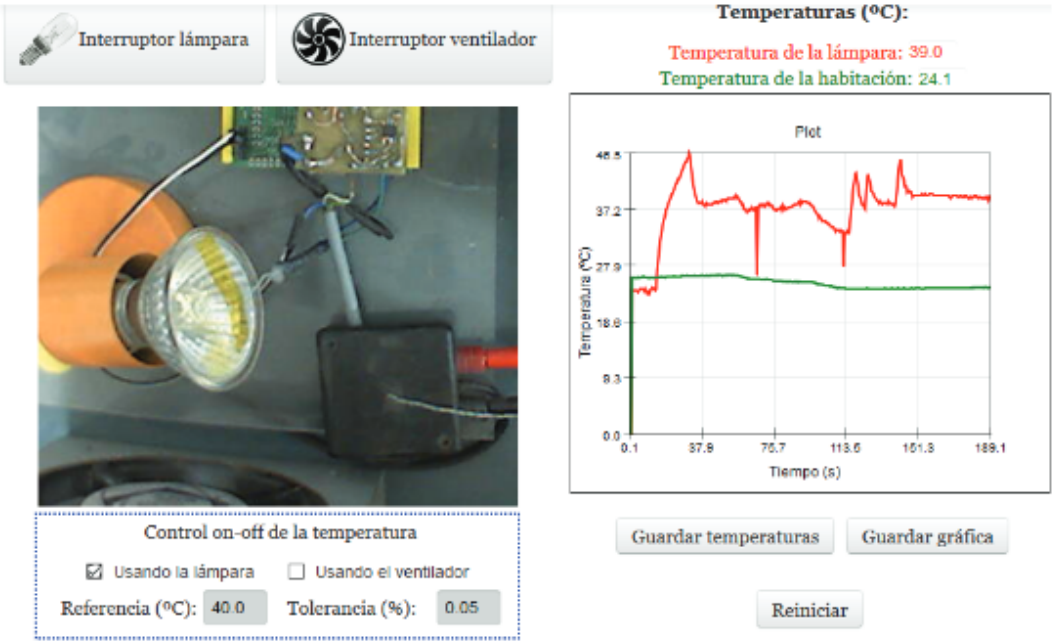


FIGURE 2.17 – Lampe et ventilateur contrôlés à l'aide d'interrupteurs accessibles à la FEUP.

2.3.9 e-lab IST

L'Institut Technique de l'Université de Lisbone [33] a developpé le système e-lab IST. Il nécessite l'installation de Java coté client pour accéder à ses laboratoires distants (voir figure 2.18). Cette contrainte, qui n'utilise pas les standards web, peut demander d'ouvrir des ports sur la machine du client ou le réseau informatique et d'ajouter des règles au pare-feu. Les services informatiques des universités sont rarement enclins à de telles mesures, ce qui peut rendre inaccessible ce laboratoire pour d'autres institutions.

2.3.10 Stanford iLABS

L'Université de Stanford aux États Unis offre un choix de cinq laboratoires distants dans divers domaines de la physique [34]. Ces laboratoires n'offrent qu'un accès à des enregistrements d'expérience, l'étudiant ne pouvant pas réellement manipuler des instruments à distance.

2.3.11 iLAB et LabShare

iLAB est un projet du MIT dirigé par Judson Harward. LabShare est un projet dirigé par David Lowe de l'Université de Technologie de Sydney (UTS). Ces deux projets sont maintenant plus ou moins inactifs bien qu'ils aient été sponsorisés par Microsoft ou le gouvernement australien. L'UTS a développé jusqu'en 2017 un logiciel [35] en Java nommé Sahara permettant l'accès distant à un ordinateur connecté à des intruments permettant de réaliser des expériences (voir figure 2.19).

33. http://elab.ist.utl.pt
34. http://ilabs.education/listing
35. https://github.com/sahara-labs

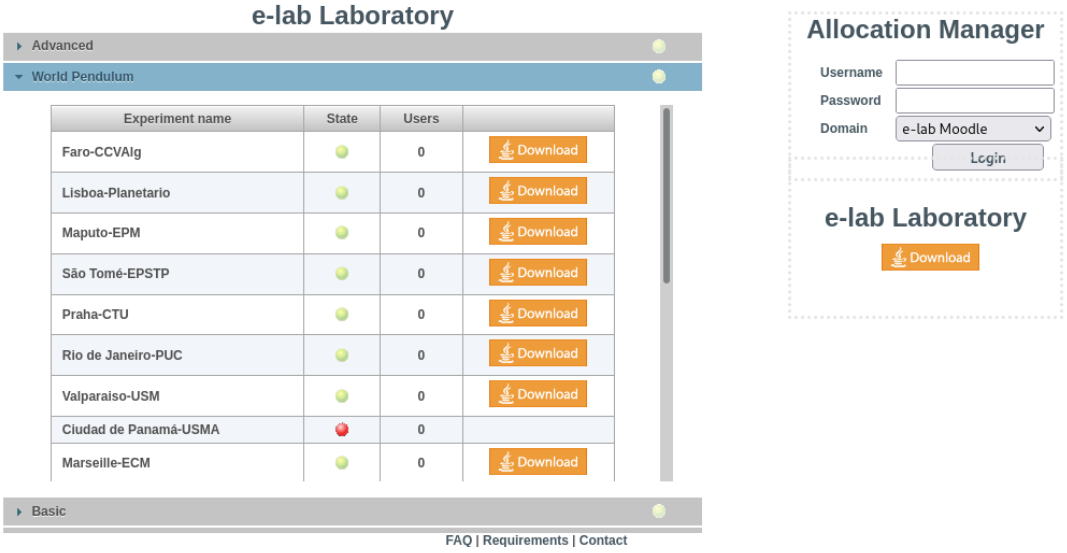


FIGURE 2.18 – Liste des expériences accessibles chez e-lab IST nécessitant Java.

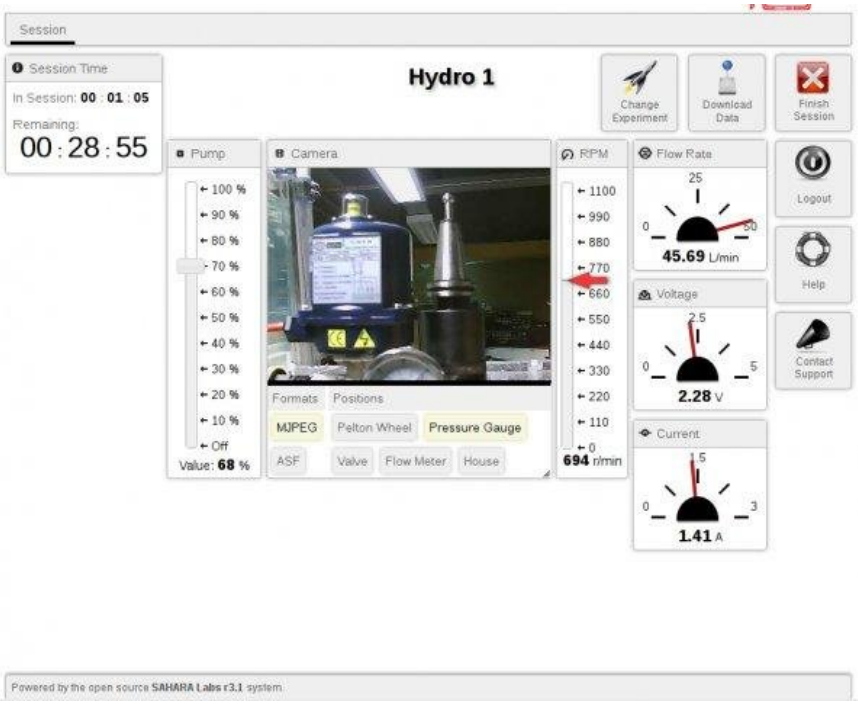


FIGURE 2.19 – Exemple d'un laboratoire distant accessible via Sahara.

2.3.12 FarLabs

FarLabs (Freely Accessible Remote Laboratories) [36] est un réseau de laboratoires distants de plusieurs universités australiennes et dirigé par l'Université de La Trobe. Il propose plusieurs laboratoires dans les domaines du nucléaire, de l'environnement (voir figure 2.20) et de l'optique.

36. https://www.farlabs.edu.au/



FIGURE 2.20 – Chauffe eau solaire utilisé pour les expériences à distance chez FarLabs.

2.4 Besoins identifiés

Après avoir étudié des systèmes de supervision et d'instrumentation à distance dans les secteurs de l'industrie, de la recherche et de l'enseignement, différents critères permettent de classer ces solutions dans différentes catégories :

1. Logiciels entièrement libres et ouverts. Les solutions propriétaires (ou partiellement) sont moins facilement distribuables.
2. Intégration de différents types d'architectures matérielles, de protocoles de communication et de niveaux de configuration (voir section 1.3.3.4.1).
3. Pas de nécessité d'installer un logiciel ou un *plugin* pour accéder à l'application. Cela simplifie l'utilisation côté client.
4. Configuration du système accessible sans code. Il n'est pas nécessaire d'avoir des connaissances en informatique pour configurer le système.
5. Système de gestion de contenu pour la création de l'IHM utilisateur. Chaque projet peut créer l'interface qui lui convient.
6. Développement et évolution du système simple via l'utilisation d'un *framework* et d'une architecture logicielle claire.
7. Possibilité d'ajouter des scripts côté serveur, des évènements et des alertes.
8. Gestion multi-sites.

Le tableau 2.2 compare les différentes solutions présentées dans ce chapitre aux huit critères listés ci-dessus. Les meilleurs résultats sont obtenus pour les logiciels du secteur de l'industrie ou pour LabVIEW qui n'est pas libre. Très peu de solutions utilisent un SGC pour s'adapter aux types d'utilisateurs finaux ou un *framework* permettant de faciliter l'évolution du logiciel nécessaire afin de répondre aux nouveaux besoins techniques.

Les besoins identifiés pour définir un système générique sont :

— échanger des informations avec des capteurs et des actionneurs permettant de connaître l'état d'un système physique et d'agir sur celui-ci,
— conserver les données afin de les traiter et de les présenter,
— donner du sens aux données en utilisant des algorithmes et en réagissant automatiquement à leurs variations,

— abstraire la distance entre l'utilisateur et le système étudié,
— être agnostique du point de vue du domaine scientifique étudié,
— être simple et intuitif quel que soit l'utilisateur et son rôle,
— rendre les données, la configuration d'un système ainsi que le logiciel faciles à copier, modifier et installer,
— laisser la liberté à chacun d'utiliser, modifier et distribuer le logiciel.

TABLE 2.2 – Comparaison de différentes solutions avec les critères définis en section 2.4 (O = Oui ; N = Non ; ? = Inconnu).

		Critères								
Secteur	Solution	1	2	3	4	5	6	7	8	Score
Industrie	Tango Controls	O	O	N	O	N	N	O	O	5
	Scada-LTS	O	O	O	O	O	N	O	O	7
	RapidSCADA	N	O	O	O	N	N	N	O	4
Recherche	National Instruments	N	O	N	N	O	O	O	O	5
	PyMoDAQ	O	N	N	O	N	N	O	N	3
	PyMeasure	O	N	N	N	N	N	O	N	2
	NSLS-II	O	N	N	N	N	N	O	N	2
	Yaq	O	N	N	N	N	N	O	N	2
Enseignement	LabsLand	N	N	O	O	N	N	N	O	3
	VISIR	N	N	O	O	N	O	N	O	4
	ISES	N	N	O	?	N	N	N	O	2
	UniLabs	O	N	N	O	N	N	N	O	3
	RexLab	O	N	O	O	N	N	N	O	4
	VirtualRemoteLab	N	N	O	?	N	N	N	O	2
	GOLDi	N	N	O	?	N	N	N	O	2
	FEUP	N	N	N	O	N	N	N	O	2
	e-lab IST	N	N	N	?	N	N	N	O	1
	iLABS	N	N	N	?	N	N	N	O	1
	iLAB & LabShare	N	N	N	?	N	N	N	O	1
	FarLabs	N	N	O	?	N	N	N	O	2

Je présente dans le chapitre 3 la solution logicielle proposée pour répondre à ces besoins et satisfaire les critères présentés ci-dessus (voir 3.2).

Chapitre 3

Contribution

Sommaire

3.1 Différentes abstractions 48
3.1.1 Abstraction de la connexion 48
3.1.2 Abstraction de l'expérimentation 48
3.1.2.1 Abstraction du transport 48
3.1.2.2 Abstraction de la charge utile 49
3.1.3 Abstraction de la synchronisation 49
3.1.4 Abstraction des erreurs 49
3.2 Architecture de PyScada 50
3.2.1 Application Django 51
3.2.2 Cœur de PyScada 52
3.2.3 Couche d'abstraction matérielle 54
3.2.3.1 Couche de la plateforme multi-matérielle 55
3.2.3.2 Couche spécifique au matériel 55
3.2.4 Interfaces web 56
3.2.4.1 Interface d'administration 56
3.2.4.2 IHM utilisateur 59
3.2.5 Évènements et alertes 61
3.2.6 Scripts et algorithmes 62
3.2.7 Approche modulaire par *plugin* 64
3.2.8 Faible besoin matériel 65
3.2.9 Évaluation face aux principes du FAIR des logiciels 65
3.3 Bilan 68

Afin de répondre aux besoins identifiés au chapitre précédent et de présenter une application capable de gérer les différents niveaux d'interopérabilité énoncés en section 1.2.5, je présenterai le logiciel **PyScada** développé à l'origine par Martin Schröder de l'Université Technique de Berlin, logiciel dont je suis désormais le principal développeur. PyScada se définit comme :

> *Un système SCADA open source avec une IHM HTML5, construit à l'aide du framework Django.*

PyScada se présente donc sous la forme d'une page web utilisant les standards du web accessible sur tous les dispositifs récents. Le code source est disponible sur *GitHub* [1] et la documentation sur *ReadTheDocs* [2].

Dans ce chapitre nous présentons dans un premier temps les différentes abstractions implémentées pour mettre en place les interopérabilités proposées au chapitre 1. Puis j'exposerai l'architecture de PyScada ainsi que son fonctionnement répondant aux besoins identifiés en section 2.4.

1. https://github.com/pyscada/PyScada
2. http://pyscada.rtfd.io

3.1 Différentes abstractions

Les caractéristiques architecturales proposées par JOHN JOSEPH ROETS [44] sont transposées à la télé-instrumentation dans cette section. Elles se décomposent en différents niveaux d'abstraction et sont destinées à faciliter la mise en œuvre d'un tel système, l'ajout de nouvelles fonctionnalités (évolutions et modifications) et la configuration d'une installation.

3.1.1 Abstraction de la connexion

La première fonctionnalité est l'abstraction de la connexion qui consiste, pour l'utilisateur qui configure le système, à connecter différents instruments via l'interface web sans ajouter ou modifier le code de l'application. Cette abstraction peut être plus ou moins forte. Voici l'exemple de trois niveaux d'abstraction réalisant la même tâche :

— abstraction forte : un *plugin* installé dans PyScada peut automatiquement créer des variables météorologiques (température, humidité, prévisions) en géo-localisant le serveur où est installée l'application (via son adresse IP ou ses coordonnées géographiques : latitude et longitude). Aucune configuration n'est requise, il s'agit d'interopérabilité PnP (voir section 1.3.3),

— abstraction moyenne : un *plugin* peut demander d'entrer des coordonnées géographiques, choisir un site de prévisions météorologiques et sélectionner les données à afficher,

— abstraction faible : un *plugin* permet de lire des documents formatés en *JavaScript Object Notation* (JSON) et accessibles via un URI. L'utilisateur configurant le système doit créer un instrument représentant le site web hébergeant les données et définir des variables pour chaque donnée lue dans le fichier JSON en lui affectant les propriétés nécessaires (nom, unité, changement d'échelle...).

Pour l'utilisateur final, qui n'est pas forcément celui qui a construit l'IHM, l'abstraction de la connexion lui permet de ne pas se rendre compte du nombre d'instruments avec lesquels le système communique. L'utilisateur interagit avec des variables représentant de manière identique des données (valeur, unité, date d'acquisition), indépendamment de leurs configurations ou des instruments et protocoles utilisés.

3.1.2 Abstraction de l'expérimentation

L'abstraction fonctionnelle, ou abstraction de l'expérimentation, est la capacité de séparer la logique d'une instrumentation (interface, algorithmes, événements) des instruments utilisés. Dans ce cas il est possible, pour l'utilisateur qui configure l'expérimentation, de créer d'abord l'interface utilisateur avec des variables fictives (non configurées et sans données enregistrées) en définissant par exemple le texte à afficher, les courbes temporelles, les boutons ou les images. Dans un second temps, il peut configurer les instruments et leurs variables afin de récolter des données et d'envoyer des consignes. L'abstraction fonctionnelle utilise deux types d'abstractions.

3.1.2.1 Abstraction du transport

L'abstraction du transport est la capacité à changer le protocole de communication d'un instrument sans influencer l'expérimentation. Par exemple, un instrument peut avoir plusieurs protocoles de communication ou un changement de version d'un instrument nécessite de changer le protocole utilisé.

3.1.2.2 Abstraction de la charge utile

L'abstraction de la charge utile est la capacité d'envoyer un message ou une requête de deux manières, soit en envoyant des types de requêtes simples que chaque protocole accepte (lecture et écriture d'une variable), soit en implémentant une couche d'abstraction permettant d'envoyer des messages plus complexes.

Par exemple une variable correspondant à l'énergie consommée par un compteur électrique est stockée dans deux mots de 16 bits, c'est-à-dire dans deux endroits de la mémoire du compteur, la valeur supérieure à un MegaWatt-heure (MWh) dans un mot (mot de poids fort) et la valeur inférieure dans l'autre mot (mot de poids faible). Dans ce cas la couche d'abstraction de cet instrument doit permettre de lire les deux mots, à l'aide d'une ou de deux requêtes, et de les additionner afin de présenter à l'utilisateur final une valeur unique de la consommation énergétique à une date donnée. Cette opération est transparente pour l'utilisateur final qui perçoit la valeur stockée dans une seule variable.

Chaque instrument possède son fichier de configuration permettant de réaliser des échanges plus complexes que la simple lecture et écriture d'une variable. Dans PyScada ce fichier s'appelle un *handler* (traduit de l'anglais par gestionnaire).

3.1.3 Abstraction de la synchronisation

L'abstraction de la synchronisation est la capacité d'appeler les fonctions d'une bibliothèque logicielle, permettant par exemple d'utiliser un protocole, indépendamment du fait que ses fonctions soient synchrones ou asynchrones (voir figure 3.1). Une fonction asynchrone ne bloque pas le déroulement du logiciel durant son exécution. Il doit être possible d'intégrer des fonctions asynchrones dans des fonctions synchrones et *vice-versa*.

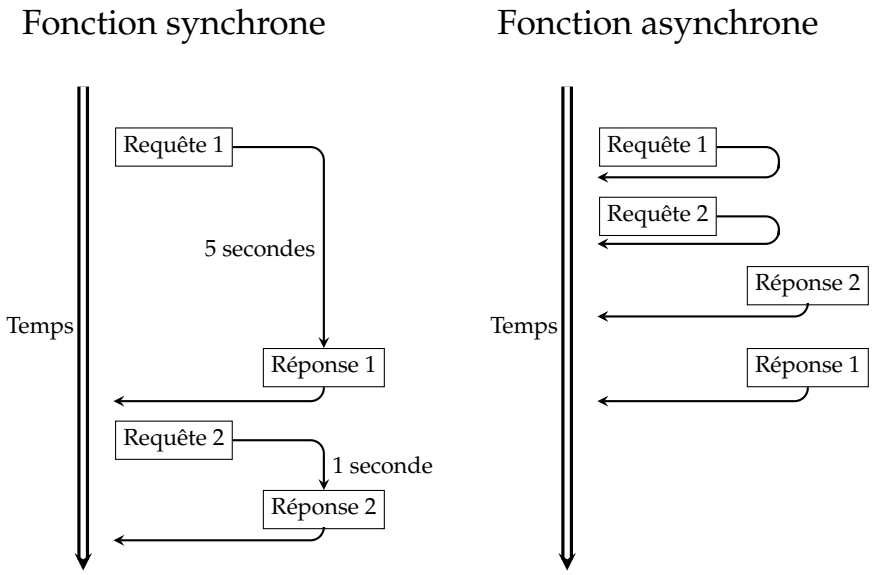


FIGURE 3.1 – Exécution de requêtes synchrones et asynchrones.

3.1.4 Abstraction des erreurs

L'abstraction des erreurs, ou du basculement, est la capacité de gérer un message d'erreur ou une perte de connexion avec un instrument sans changement fonctionnel

de l'application. Il peut être nécessaire d'informer l'utilisateur de l'incident ou de conserver sa trace afin de l'analyser.

3.2 Architecture de PyScada

PyScada est un logiciel *open source* et libre (voir section 1.2.4) depuis sa création (première recommandation de la section 1.2.4.4), sous licence GPL-3.0 (troisième recommandation de la section 1.2.4.4) et fonctionne sur toute plateforme UNIX et sur Windows à l'aide de conteneurs Docker [3] (cela répond à la problématique d'interopérabilité de la section 1.3.3.1). Le dépôt officiel de PyScada [4] contient le code source et les méta-données du logiciel (deuxième recommandation de la section 1.2.4.4). PyScada est également disponible sur le Python Package Index (PyPI) [5]. Un canal de discussion est disponible sur Matrix [6] et les questions ou problèmes sont traités au fur et à mesure qu'ils se présentent (dernière recommandation de la section 1.2.4.4). PyScada respecte donc l'ensemble des recommandations énoncées par JIMÉNEZ et al. [19].

L'architecture de PyScada est développée autour de Django [7], un *framework* Python, et vise à être simple à déployer (figure 3.2). Django fournit deux interfaces web lui permettant de configurer et d'interagir avec des instruments utilisant l'un des protocoles suivants : Modbus [8], BACNet [9], VISA [10], OneWire [11], SMBus/I2C [12], GPIO [13], PT104 [14], OPC-UA [15], SML [16], Serial [17], WebService [18]. L'ajout de nouveaux protocoles est facile grâce à l'architecture modulaire de PyScada (voir section 3.2.7), répondant à la problématique d'interopérabilité de la section 1.3.3.2. Pour générer les pages web, PyScada utilise le serveur web NGINX [19] et la passerelle Python WSGI (*Web Server Interface Gateway*) Gunicorn [20].

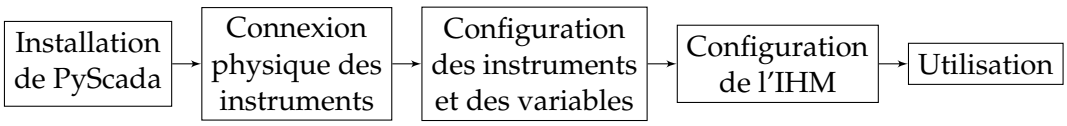


FIGURE 3.2 – Processus de mise en place de PyScada.

3. https://www.docker.com/
4. https://github.com/pyscada/PyScada
5. https://pypi.org/project/PyScada/
6. https://matrix.to/#/#pyscada:matrix.org
7. https://www.djangoproject.com/
8. https://modbus.org/
9. http://www.bacnet.org/
10. https://www.ivifoundation.org/
11. https://www.maximintegrated.com/en/design/technical-documents/app-notes/7/74.html
12. https://www.i2c-bus.org/
13. https://www.kernel.org/doc/html/latest/driver-api/gpio/
14. PT104 est un enregistreur de mesure de température de chez PicoTech. https://www.picotech.com/data-logger/pt-104/high-accuracy-temperature-daq
15. https://opcfoundation.org/
16. https://de.wikipedia.org/wiki/Smart_Message_Language
17. Accéder à des périphériques via un port série : https://en.wikipedia.org/wiki/Serial_port
18. https://www.w3.org/TR/ws-arch/
19. https://www.nginx.com/
20. https://gunicorn.org/

Le matériel et les variables sont configurés dans PyScada par une couche abstraite et une couche spécifique du protocole utilisé (voir la section 3.2.3). Cette interopérabilité protocolaire et matérielle de PyScada permet de traiter toutes les variables de manière uniforme en lecture ou en écriture. Ces données sont utilisées pour créer des événements, des alertes ou des scripts afin d'exécuter des algorithmes complexes, interagir avec l'utilisateur et automatiser des processus de supervision (voir section 1.3.1).

3.2.1 Application Django

Django se définit lui-même comme un cadre web Python de haut niveau qui encourage le développement rapide et une conception propre et pragmatique. Il est gratuit, *open source*, rapide, sécurisé et évolutif. Sa grande communauté, ses mises à jour régulières et ses applications disponibles permettent de créer une application web sans "réinventer la roue".

Les données et la configuration sont stockées dans une base de données. Django prend en charge PostgreSQL [21], MySQL [22], MariaDB [23], Oracle [24], SQLite [25].

Django utilise trois couches d'abstractions : le modèle, la vue et le gabarit (MVT de l'anglais *Model-View-Template*, voir figure 3.3).

Le **modèle** est utilisé pour structurer et manipuler les données. Il contient ses champs essentiels, définit comment créer, lire, mettre à jour et supprimer des données (C.R.U.D de l'anglais *Create, Read, Update, and Delete*) et est lié à une base de données par une liaison objet-relationnel (ORM de l'anglais *Object-Relational Mapping*).

La **vue** traite les demandes des utilisateurs (via un distributeur d'URL) et renvoie une réponse après avoir organisé les données à l'aide de gabarits.

Les **gabarits** sont utilisés pour générer des pages web HTML de manière dynamique. La production des gabarits consiste pour chaque requête à trouver le code statique correspondant et interpoler le contenu provenant de la base de données à l'aide d'une syntaxe particulière aux gabarits.

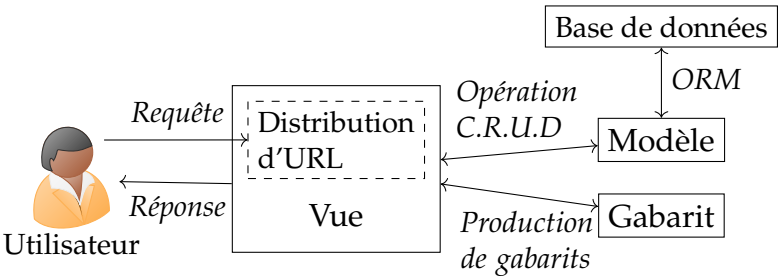


FIGURE 3.3 – Architecture Django Modèle-Vue-Gabarit.

L'architecture MVT, séparant la logique de création de site web en trois parties, a plusieurs avantages :

— elle rend facilement évolutif le site web, les développeurs pouvant travailler simultanément sur les différentes parties de l'application,
— un modèle peut avoir différentes représentations (vues),

21. https://www.postgresql.org/
22. https://www.mysql.com/
23. https://mariadb.com/
24. https://www.oracle.com/database/
25. https://www.sqlite.org/

— la logique de l'application est séparée de l'intégration visuelle et du stockage de données,
— la maintenance est plus aisée.

A l'aide de cette architecture, PyScada peut répondre aux besoins de supervision et d'instrumentation pour des applications variées (voir section 1.2.1) via l'abstraction de l'expérimentation (voir section 3.1.2).

La figure 3.4 présente les relations entre les différents éléments de PyScada dans l'environnement géré par Django (base de données et serveur web). Les modules de PyScada, en bleu foncé, sont définis dans l'application par défaut ou ajoutés via un *plugin*. Ils peuvent communiquer avec des instruments, des capteurs, des actionneurs ou le système hôte en implémentant des librairies *open source* utilisant un protocole de communication. La création de l'interface utilisateur est générée via le module d'IHM de PyScada et par l'intermédiaire de Gunicorn et NGINX. Un *plugin* peut proposer de nouveau composants en ajoutant de nouvelles **vues** et de nouveaux **gabarits**, comme par exemple un thème pour l'interface utilisateur ou des types de graphiques à l'aide des librairies (voir figure 1.10). Il peut aussi permettre de se connecter à de nouveaux instruments en ajoutant un protocole à la liste des protocoles acceptés par PyScada.

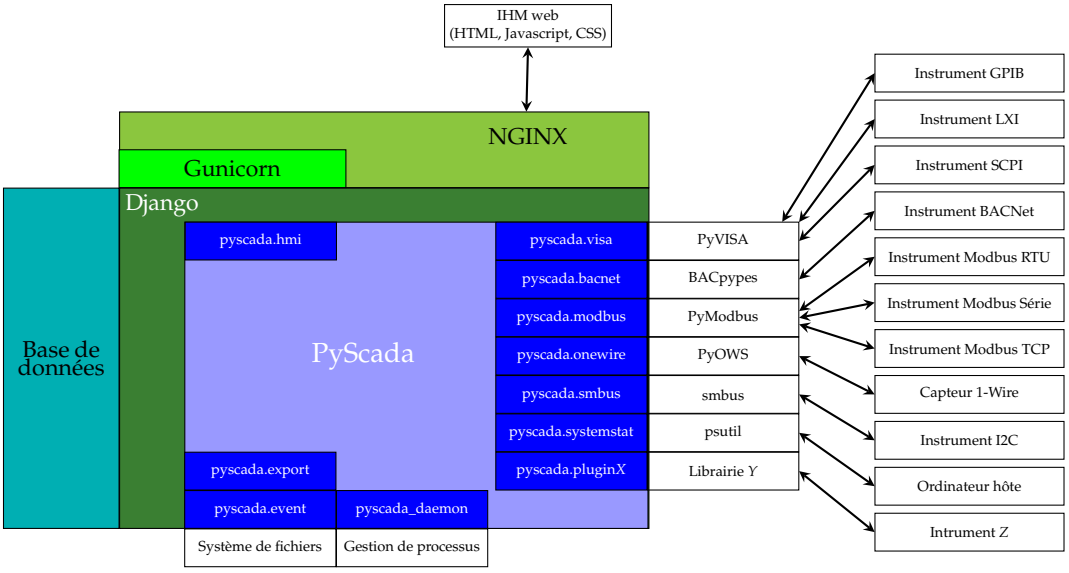


FIGURE 3.4 – Schéma de la structure de PyScada dans l'environnement Django.

3.2.2 Cœur de PyScada

L'ensemble des modules de PyScada s'échange des données de manière asynchrone via la base de données. La figure 3.5 présente le fonctionnement des différents modules.

Les instruments sont définis par des champs génériques et communs à tous :
— un nom,
— un protocole de communication,
— un intervalle de rafraîchissement.

Chaque variable créée doit être attachée à un instrument. Elle peut être accessible en lecture (télémesure) ou en écriture (télé-contrôle). Les valeurs de l'ensemble des variables d'un instrument sont actualisées à intervalle régulier sauf si une demande d'**action de lecture** pour cet instrument (*Device Read Task*) est solicitée. Dans ce cas

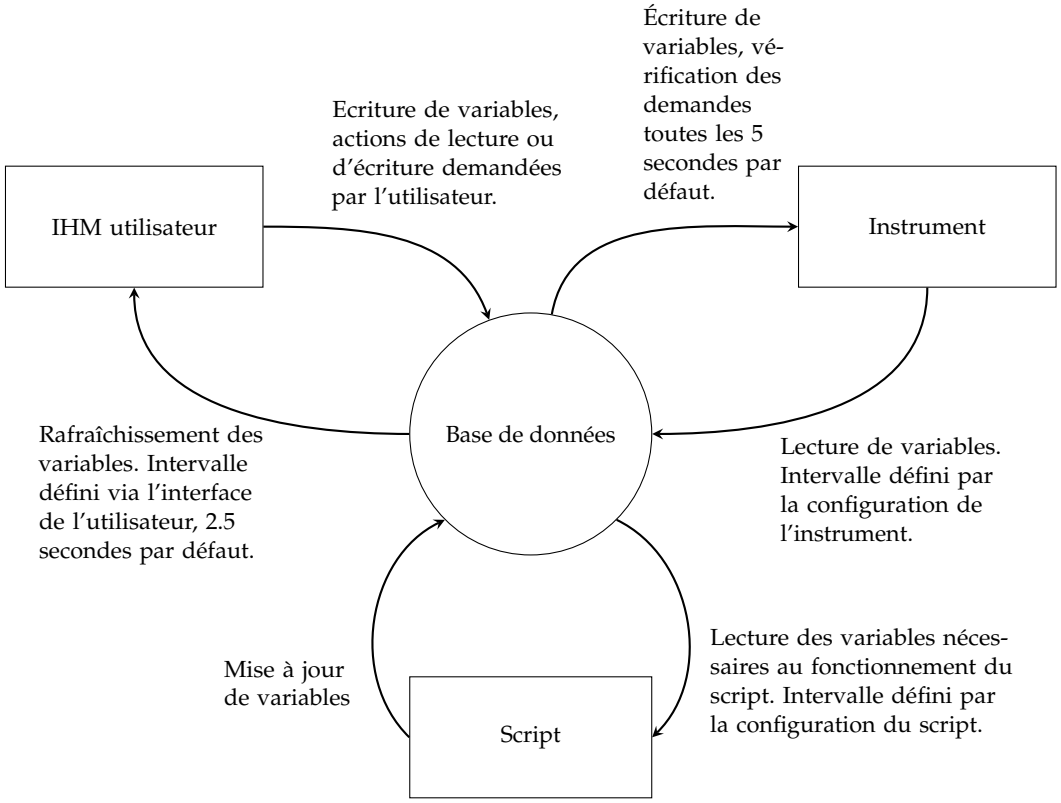


FIGURE 3.5 – Fonctionnement asynchrone des modules de PyScada communiquant via la base de données.

l'instrument procède à une mise à jour anticipée des valeurs de ses variables. Pour les variables définies en écriture, chaque instrument scrute par défaut toutes les cinq secondes si une **action d'écriture** (*Device Write Task*) est demandée pour chacune de ses variables et envoie les consignes correspondantes.

Par défaut, l'IHM utilisateur rafraîchit toutes les deux secondes et demie l'ensemble des valeurs des variables affichées à l'utilisateur. Le taux de rafraîchissement est modifiable par l'utilisateur et il est possible d'arrêter le rafraîchissement automatique des variables affichées (voir figure 3.6). L'IHM envoie les **actions d'écriture** pour une ou plusieurs variables sur demande de l'utilisateur en réagissant à ses interactions via des boutons pour les variables booléennes, des champs, des listes déroulantes ou des formulaires pour les autres types de variables : entiers, décimaux, textes (voir figure 3.13).

Il est possible d'exécuter des scripts en Python (voir section 3.2.6) qui s'exécutent à un intervalle configuré dans l'interface d'administration. Les scripts peuvent lire et écrire les valeurs de variables et déclencher des **actions de lecture** ou **d'écriture** pour des instruments. D'autres *plugins* tels que ceux permettant de définir des alertes et des évènements ou de programmer l'exportation de données fonctionnent sur le même principe ; ils lisent des variables en base de données à intervalle régulier et créent des **demandes d'écriture** ou de **lecture** si besoin.

Une option lors de l'installation de PyScada permet de notifier instantanément un instrument ou un script qui reçoit une **demande de lecture** ou **d'écriture**, rendant le

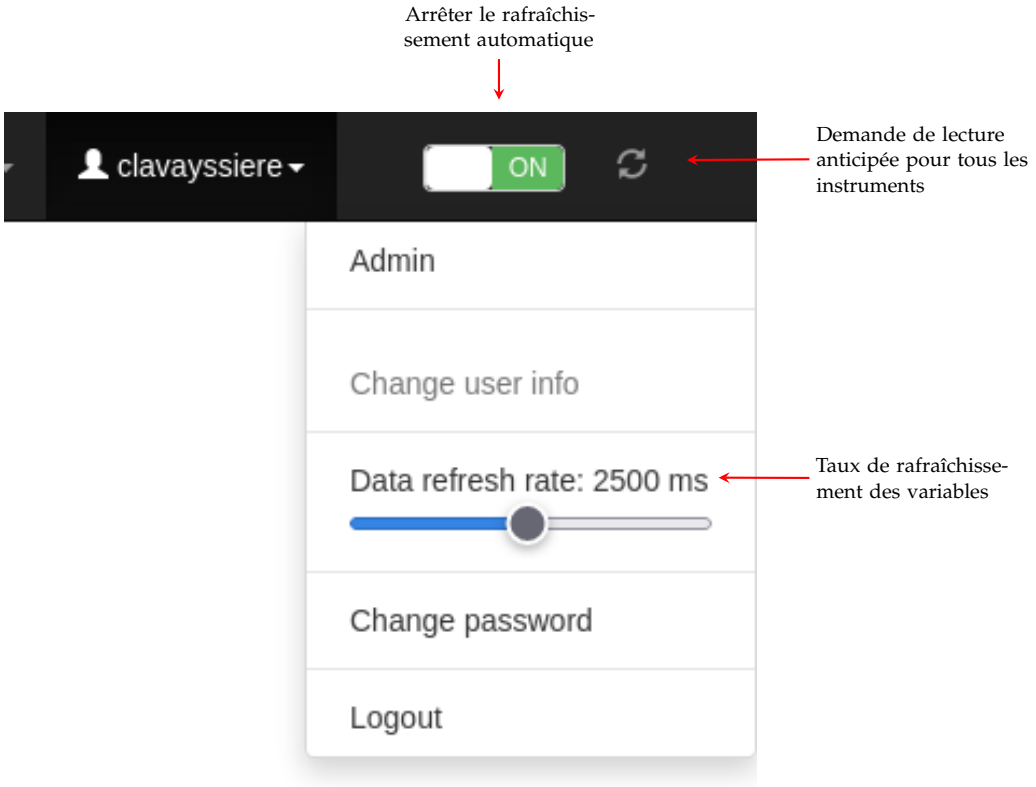


FIGURE 3.6 – Contrôle du rafraîchissement des valeurs des variables dans le coin supérieur droit de l'IHM utilisateur.

système plus réactif en réduisant les intervalles de lecture et d'écriture. Cette technique utilise Redis [26] et l'application Django Channels [27] pour passer des messages via la mémoire vive entre les processus systèmes.

3.2.3 Couche d'abstraction matérielle

La couche d'abstraction matérielle (HAL de l'anglais *Hardware Abstraction Layer*) est la partie du code qui permet la communication entre le matériel et le logiciel d'une manière abstraite et uniforme. L'objectif de la HAL est d'améliorer l'interopérabilité du matériel et de faciliter l'écriture de logiciels (voir section 1.3.3.4.4). La HAL est utilisée dans différents domaines de l'informatique, elle est connue au niveau des systèmes d'exploitation (OS) permettant de communiquer avec un composant d'un ordinateur à un niveau général ou abstrait plutôt qu'à un niveau détaillé (voir HUANG et al. [45]). D'autres propositions de HAL ont émergé pour le domaine de la robotique [28] ou même des réseaux informatiques (voir PARNIEWICZ et al. [46]).

Pour l'instrumentation à distance, l'objectif est d'augmenter le nombre de protocoles et d'instruments pris en charge, de simplifier le code logiciel utilisé pour la télémesure et le télécontrôle, et de standardiser les informations présentées à l'utilisateur final.

Nous avons adapté l'approche modulaire de PARNIEWICZ et al. [46] à PyScada sur la figure 3.7. Cette architecture permet de modifier ou d'ajouter des fonctionnalités sans remettre en cause l'ensemble du système. La communication propre à chaque

26. https://redis.io/
27. https://channels.readthedocs.io/en/stable/
28. https://www.omg.org/spec/HAL4RT/1.0/Beta1/About-HAL4RT/

protocole, sa complexité et ses spécificités sont séparées de la logique d'exécution du logiciel de supervision et d'instrumentation. Les protocoles sont cachés pour l'utilisateur qui ne voit que des variables représentées de manière identique. La couche HAL est décomposée en deux sous-couches, la couche de la plateforme multi-matérielle (CHPL de l'anglais *Cross-Hardware Platform Layer*) et la couche spécifique au matériel (HSL de l'anglais *Hardware Specific Layer*).

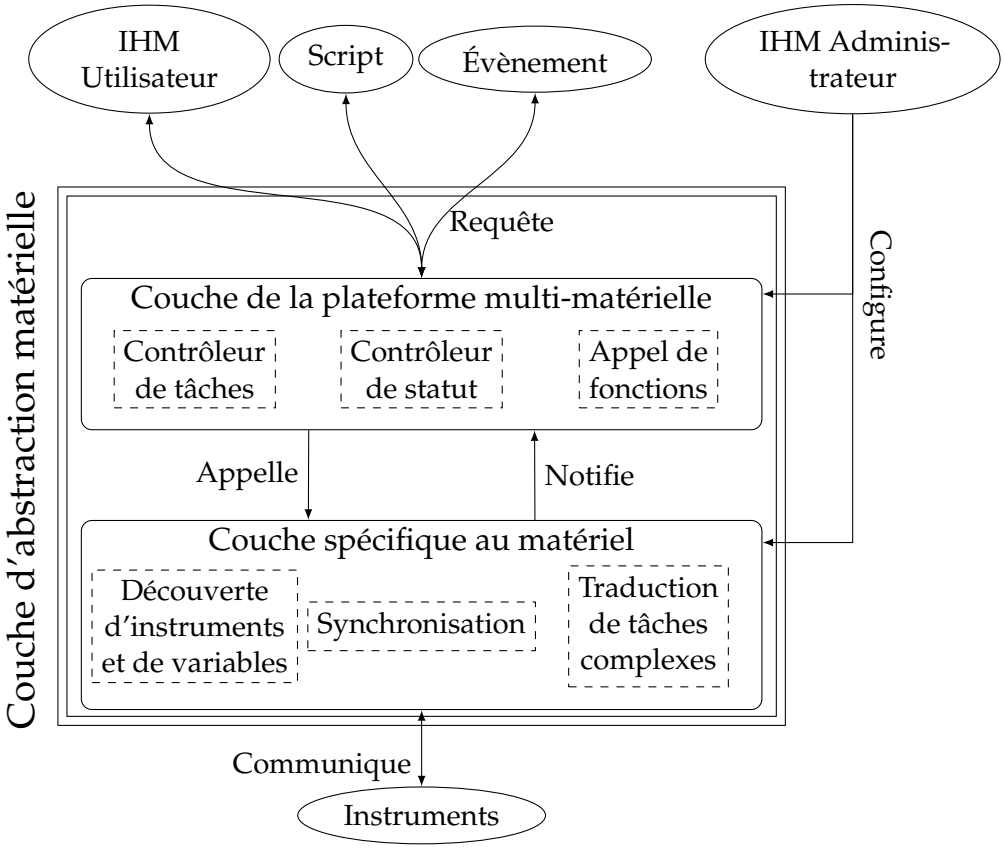


FIGURE 3.7 – Architecture HAL de PyScada.

3.2.3.1 Couche de la plateforme multi-matérielle

La couche CHPL est la partie du logiciel **agnostique** du point de vue de l'instrument et du protocole de communication utilisés. Elle traite les différentes tâches de **demande de lecture** et **d'écriture** de variables provenant de l'interface utilisateur, d'un script ou d'un événement. Elle appelle des fonctions génériques de la HSL pour se connecter et communiquer avec les instruments. Elle contrôle le statut des instruments et des variables et peut déclencher une alerte si un défaut survient.

3.2.3.2 Couche spécifique au matériel

La couche HSL est la partie qui cache la complexité et la diversité des instruments d'une installation. Elle doit fournir une interface standard permettant au CHPL de communiquer avec un instrument. Elle réagit aux appels du CHPL et lui notifie le résultat. Lorsqu'un nouveau protocole est ajouté, la HSL correspondante doit être développée pour traduire les exigences du CHPL aux instruments connectés utilisant ce protocole. Les fonctions génériques de la HSL peuvent être traitées à l'aide d'un *handler* qui se charge de réaliser des tâches complexes en décomposant la demande du

CHPL en sous-fonctions (voir l'exemple de la section 3.1.2.2). Selon le protocole et la librairie utilisés, la HSL peut découvrir automatiquement les instruments connectés au serveur exécutant PyScada ainsi que la liste de leurs variables et toute méta-donnée associée. La configuration nécessaire à une HSL pour un protocole ou un instrument donné est renseignée via l'IHM d'administration (voir l'exemple de la section 3.1.1). La HSL gère aussi l'abstraction de la synchronisation de la librairie spécifique au protocole utilisé.

3.2.4 Interfaces web

Pour offrir un accès à distance, l'architecture Django fournit deux interfaces. Premièrement, l'interface utilisateur permet à toute personne d'accéder à une expérimentation en visualisant des données liées à celle-ci, et en lui envoyant des consignes ou requêtes si besoin. Deuxièmement, l'interface d'administration est utilisée pour **construire** le système de supervision. Elle permet notamment de configurer les instruments, de définir les variables et de créer l'interface utilisateur via un système de gestion de contenu (SGC ou CMS de l'anglais *Content Management System*).

3.2.4.1 Interface d'administration

Django installe automatiquement une interface d'administration permettant d'accéder aux modèles de l'application pour créer, lire, mettre à jour et supprimer (C.R.U.D.) des données.

Le modèle, partiellement schématisé dans la figure 3.8, est composé de plusieurs blocs. Certains blocs sont inclus par défaut avec Django, comme le bloc *Authentication* (gestion des utilisateurs et des groupes), d'autres sont installés avec PyScada comme le bloc *Core* et le bloc IHM. Lorsqu'un *plugin* est installé, un nouveau bloc peut apparaître pour ajouter des fonctionnalités particulières. Chaque bloc contient un modèle définissant les champs disponibles dans l'interface d'administration.

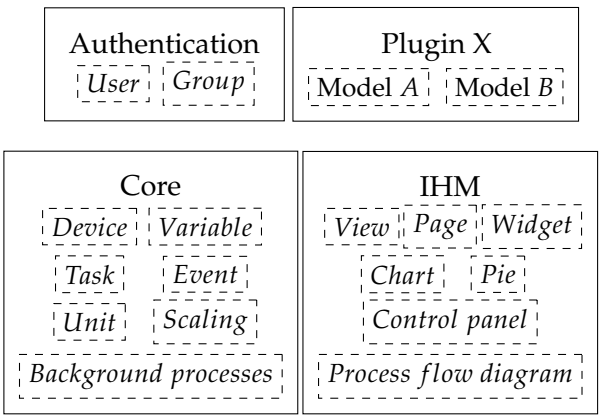


FIGURE 3.8 – Architecture de l'interface d'administration incluant certains des principaux blocs de PyScada et une liste non exhaustive de leur modèle.

La figure 3.9 présente la page d'administration de PyScada où apparaissent les blocs par défaut *Authentication*, *Core* et IHM et des blocs qui proviennent de *plugins* : *Export*, *Scripting* et *WebService*.

Selon l'architecture du modèle d'un *plugin*, de nouvelles options peuvent être ajoutées dans les blocs existants. Par exemple la figure 3.10 montre que le *plugin*



FIGURE 3.9 – Exemple de l'interface d'administration de PyScada incluant le bloc d'authentification de Django, les blocs *Core* et IHM de PyScada et les blocs des *plugins Export*, *Scripting* et *WebService* de PyScada.

PyScada-OPCUA offre la possibilité de connecter des instruments utilisant le protocole **OPC-UA** en ajoutant des options de configurations propres à ce protocole dans la page de configuration du modèle *Device* du bloc *Core*.

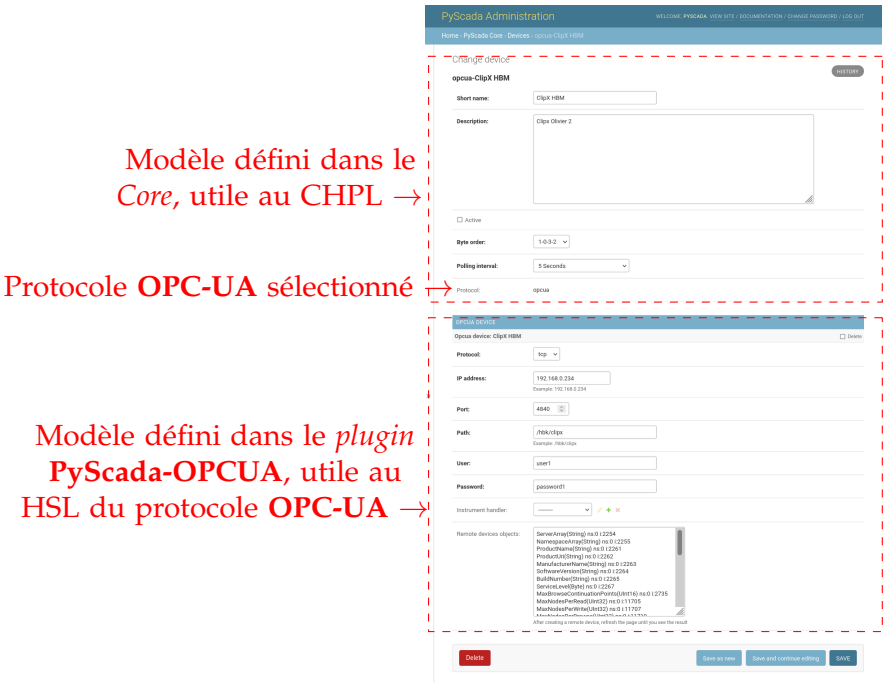


FIGURE 3.10 – Exemple de configuration (CHPL et HSL) d'un instrument utilisant le protocole OPC-UA dans PyScada.

L'application ***Authentication*** fournie par Django permet de créer des utilisateurs et de les ajouter à des groupes. Elle gère les utilisateurs et les groupes ayant accès à l'ensemble ou à une partie de l'interface d'administration à l'aide d'une gestion de droits afin d'autoriser pour chaque élément du modèle de voir, modifier, créer ou supprimer un élément. Par exemple, la figure 3.11 montre l'interface d'administration pour un utilisateur pouvant seulement **voir** les utilisateurs, les groupes ainsi que quelques modèles de PyScada, sans pouvoir les modifier.

Le bloc ***Core*** de PyScada est utilisé pour :

— créer des instruments en spécifiant le protocole de communication utilisé et l'intervalle de rafraîchissement,
— définir des variables en indiquant l'instrument utilisé,
— créer des unités à assigner aux variables,
— créer des mises à l'échelle, ce qui permettra par exemple de stocker en ampère une valeur lue en milliampères,
— surveiller l'état et redémarrer les processus lancés, chaque instrument ayant son propre processus d'exécution,
— définir des événements,
— accéder à l'historique des **demandes d'écriture** ou de **lecture** pour chaque instrument.

Le bloc **IHM** de PyScada permet de créer l'interface utilisateur présentée dans la section suivante.

Le *plugin* ***Export*** sert à programmer l'archivage des données d'un groupe de variables pour une période dans un fichier CSV ou HDF5. Le *plugin* ***Scripting*** autorise l'ajout de *scripts* en Python (voir section 3.2.6). Le *plugin* ***WebService*** ajoute le protocole



FIGURE 3.11 – Exemple d'interface d'administration de PyScada avec des droits limités.

webservice à la liste des protocoles disponibles dans la configuration d'un instrument et permet de dialoguer avec des interfaces web (lecture et écriture).

Le fonctionnement de l'interface d'administration de PyScada permet de répondre aux problématiques, définies en section 1.3.3.4, des utilisateurs voulant configurer, utiliser et visualiser le système de supervision.

3.2.4.2 IHM utilisateur

L'interface utilisateur permet d'interagir avec l'expérimentation à distance en utilisant des éléments dynamiques. La figure 3.12 présente un schéma de l'interface utilisateur qui est composée :

- de vues contenant des pages,
- de pages,
- de *widgets* qui sont des blocs de contenu à positionner dans des pages,
- différents composants graphiques statiques (texte, image) ou dynamiques (graphique, indicateur, bouton, entrée) à placer dans un *widget*.

Ce schéma présente des *widgets* en pointillés permettant de placer des composants graphiques dans une page selon :

- une grille définie par :
  - des lignes : de une à douze,
  - des colonnes : de une à quatre,
- une proportion de la largeur totale :
  - 25% équivaut à une colonne,
  - 50% équivaut à deux colonnes,
  - 75% équivaut à trois colonnes,
  - 100% équivaut à quatre colonnes.

Différentes vues peuvent par exemple être créées pour répondre aux besoins des différents utilisateurs ou pour représenter plusieurs sites géographiques. La décentralisation (voir section 1.2.6) peut être gérée au niveau des vues en intégrant des vues provenant de différentes instances sur une même page. Une amélioration à venir sera de pouvoir décentraliser tous les éléments visuels. Ainsi les instances

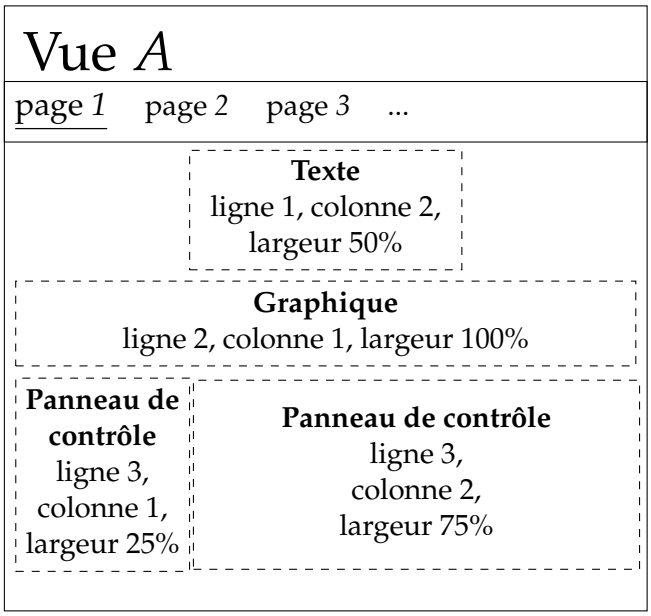


FIGURE 3.12 – Schéma d'une **vue A** contenant trois pages. La **page 1** est sélectionnée et affiche quatre *widgets*. Le *widget* de la première ligne contient du **texte** et est situé en colonnes deux et trois, il occupe la moitié de la largeur de la page. Le *widget* de la deuxième ligne contient un **graphique** et occupe 100% de la largeur de la page. Les *widgets* de la troisième ligne contiennent chacun un **panneau de contrôle** permettant d'afficher des valeurs ou d'envoyer des consignes, ils occupent respectivement une et trois colonnes de large.

pourront être connectées entre elles, on parle alors de fédération. Par exemple les pages, les *widgets* et les graphiques d'une instance fédérée pourront être intégrés dans une interface utilisateur de PyScada.

Les variables utilisées pour la télémesure peuvent être affichées via des valeurs instantanées, des séries temporelles, des graphiques XY, des camemberts ou des jauges. Des options rendent l'interface plus ergonomique, il est par exemple possible d'ajouter un code couleur en fonction de la valeur, de remplacer la valeur par un texte plus explicite, de transformer un *timestamp* (entier représentant une date en secondes et millisecondes depuis le 1er janvier 1970) en une date et une heure plus compréhensibles. Les variables utilisées pour le télécontrôle sont accessibles par des boutons pour les variables booléennes, des champs de saisie pour les entiers ou les flottants, et des listes déroulantes pour le texte.

La figure 3.13 montre la première page d'une vue avec du texte (ligne 1), un graphique temporel (ligne 2), les valeurs des variables (ligne 3, colonne 1), un bouton, une liste déroulante et un champ de saisie (ligne 3, colonne 2). Une barre de navigation en haut permet d'accéder, à gauche, à la page d'accueil pour sélectionner une autre vue et aux autres pages de cette vue ; à droite, à la période affichée dans les graphiques (date et heure de début et de fin), à la page de déconnexion, et au rafraîchissement automatique des valeurs (désactivation et taux, voir figure 3.6).

Dans de nombreuses installations techniques, plusieurs opérateurs auront accès à une installation avec des besoins et des droits différents pour afficher (télémesure) ou manipuler (télécontrôle) les données des appareils connectés. Une caractéristique importante de PyScada, que l'on ne trouve pas dans tous les autres logiciels de télé-instrumentation, est la possibilité de **créer simplement plusieurs interfaces** utilisateur similaires. Lorsque plusieurs interfaces diffèrent seulement de quelques



FIGURE 3.13 – Exemple d'interface utilisateur de PyScada.

éléments (page, *widget*, graphique, variable...), il est possible de créer une seule interface et d'adapter l'affichage de certains éléments en autorisant leur affichage ou en les cachant à un utilisateur ou un groupe d'utilisateurs.

Par exemple deux utilisateurs **A** et **B** accèdent à une interface presque identique et l'utilisateur **A** dispose d'un bouton permettant d'allumer un instrument auquel l'utilisateur **B** ne doit pas avoir accès. Dans PyScada il suffit de spécifier dans les droits d'accès que **B** ne doit pas voir le bouton et l'IHM s'adaptera automatiquement à l'utilisateur connecté. Dans ce cas, l'objectif est d'éviter de dupliquer les éléments lorsque c'est possible.

La figure 3.14 présente la vue de la figure 3.13 où certains éléments (page, valeur d'une variable et *widget*) ne sont pas visibles. La figure 3.15 montre la configuration de l'interface d'administration qui enlève de l'interface utilisateur certains éléments pour les utilisateurs appartenant au groupe sélectionné.

3.2.5 Évènements et alertes

PyScada intègre une gestion d'évènements et d'alertes dans un même module. Les utilisateurs configurant le système peuvent créer différentes alertes, pour prévenir l'utilisateur, et évènements, pour réagir à des changements. Les données d'entrée sont les valeurs des variables qui sont scrutées toutes les cinq secondes. Les évènements sont définis par :

- une liste de variables d'entrée dont chaque variable possède :
  - une valeur limite basse fixée ou dépendant d'une autre variable,
  - un hystérésis bas,
  - une valeur limite haute fixée ou dépendant d'une autre variable,
  - un hystérésis haut,

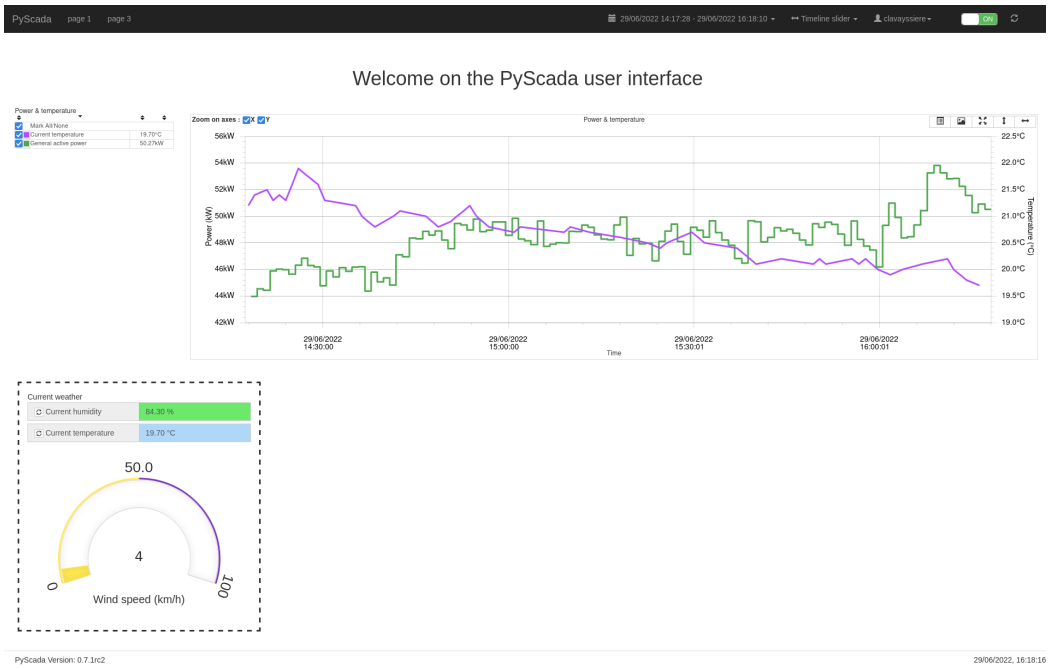


FIGURE 3.14 – Exemple d'interface utilisateur de PyScada pour un utilisateur ne disposant pas des droits pour voir tous les éléments.

— une condition de validation d'intersection (et) ou d'union (ou),
— un niveau d'alerte parmi quatre : informatif, ok, avertissement et danger.

Une variable d'un évènement s'active si elle est comprise entre sa valeur limite basse et sa valeur limite haute, chaque limite étant pondérée par l'hystérésis. Dans le cas d'un évènement avec une condition de validation d'intersection, il est nécessaire que toutes les variables soient actives pour que l'événement soit actif. Dans le cas d'une condition d'union, si l'une des variables est active alors l'évènement est actif. Lorsqu'un évènement s'active, il est possible de configurer des sorties :

— envoyer des mails à différents utilisateurs de PyScada,
— changer la valeur d'une variable pour avertir un utilisateur ou agir sur un instrument.

Le gestionnaire d'évènement permet de créer des alertes pour les utilisateurs et d'automatiser certaines tâches simples en changeant la valeur d'une variable lorsqu'un évènement s'active.

3.2.6 Scripts et algorithmes

Afin d'automatiser des systèmes instrumentés, il est nécessaire d'ajouter des algorithmes utilisant les données des variables de lecture et envoyant des consignes aux variables en écriture.

Cette interaction avec les instruments connectés à PyScada se réalise via un *plugin* **PyScada-Scripting**.

Un script Python dans PyScada est divisé en 3 parties. Une partie est lancée au démarrage, une autre à l'arrêt et la partie principale à intervalles réguliers comme expliqué dans la figure 3.16.

La figure 3.17 détaille les différentes étapes d'un script permettant la visualisation des diagrammes de Bode pour des étudiants du projet Laborem (voir section 4.1) en effectuant des mesures à l'aide d'un oscilloscope et d'un générateur de fonctions basses fréquences et en retournant les valeurs dans l'IHM utilisateur. Le code principal

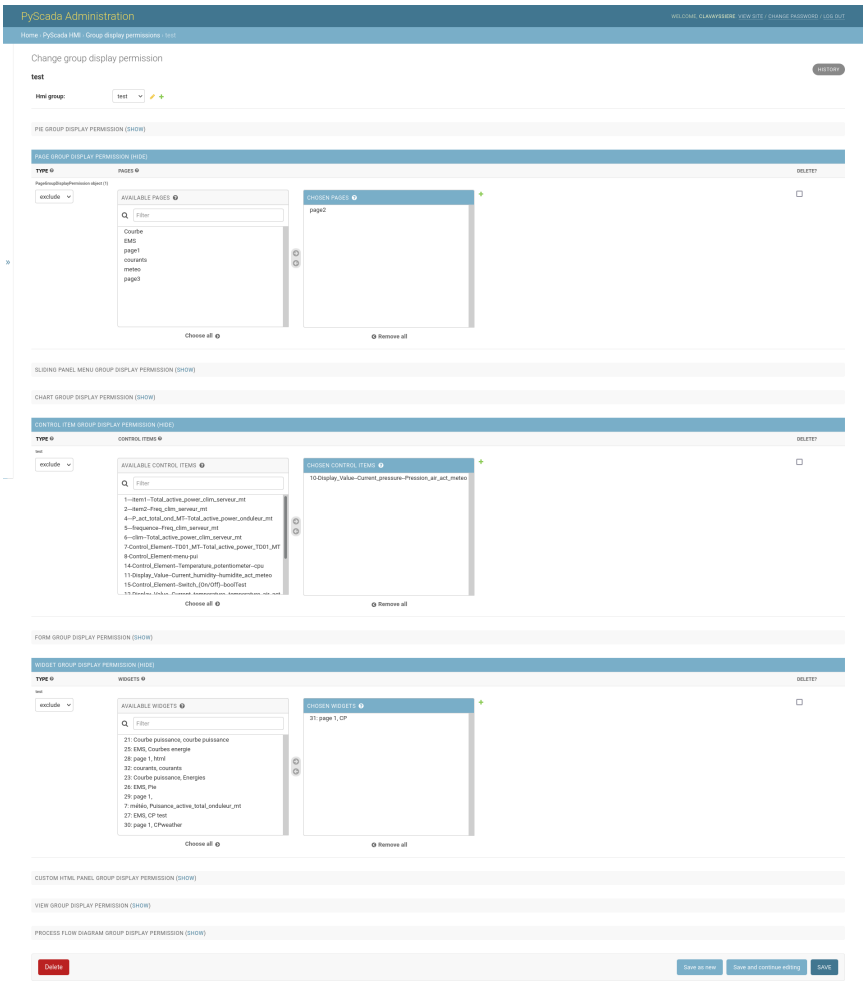


FIGURE 3.15 – Gestion des permissions d'affichage de certains éléments graphiques pour un groupe d'utilisateurs. Voir affichage en pleine page en annexe A.

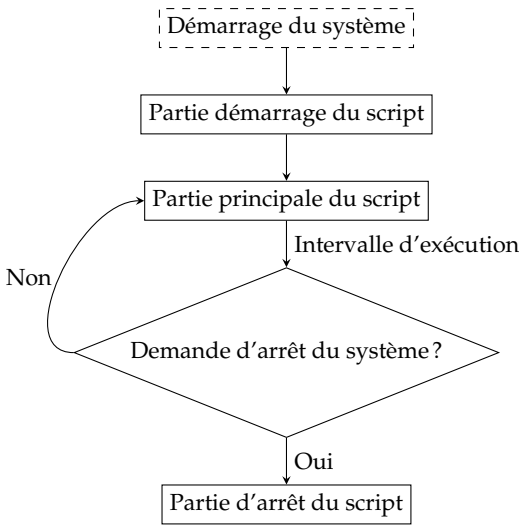


FIGURE 3.16 – Flux d'un *script* dans PyScada.

scrute toutes les secondes si un diagramme de Bode a été demandé par un étudiant. Ce script est disponible sur la plateforme GitHub [29].

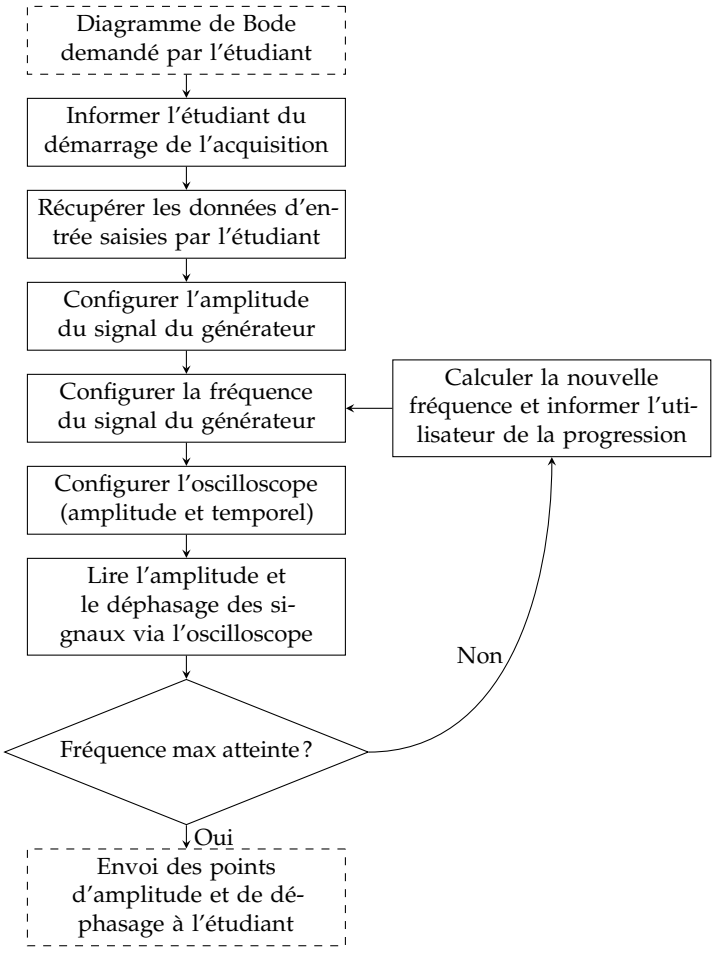


FIGURE 3.17 – Flux du *script* permettant de réaliser des diagrammes de Bode.

Un script peut servir à chercher et traiter les données de manière automatique en parcourant les méta-données disponibles sur le web (voir section 1.2.3).

3.2.7 Approche modulaire par *plugin*

L'architecture modulaire de PyScada présentée en figure 3.4 est implémentée grâce à l'architecture logicielle du *framework* Django. Elle permet de facilement ajouter des *plugins* qui modifient les possibilités de configurations actuelles ou en ajoutent de nouvelles, afin de faire évoluer PyScada.

Chaque module peut prendre part à différents aspects de PyScada, que ce soit la gestion des utilisateurs, l'IHM d'administration ou utilisateur, les protocoles de communication disponibles, l'ajout d'un nouveau type d'évènement...

Par exemple si l'on veut créer un nouveau thème dans PyScada il suffit de créer un *plugin* contenant un fichier CSS pour définir l'aspect visuel et des fichiers HTML remplaçant les gabarits par défaut (voir section 3.2.4.2) afin de proposer une nouvelle vue en utilisant les mêmes données et le même modèle.

29. https://github.com/clavay/PyScada-Laborem/blob/master/pyscada/laborem/scripts/bode_and_spectrum_with_handlers.py

Pour rajouter un protocole, il suffit de créer un *plugin* définissant les champs à configurer des instruments et des variables utilisant ce protocole (voir figure 3.10), ainsi que la manière d'utiliser la librairie correspondante (HSL).

La contamination de la GPL v3 impose à tout *plugin* d'être sous licence GPL v3 sauf si une autre librairie utilisée est en AGPL, dans ce cas le *plugin* sera sous licence AGPL (voir section 1.3.3.3).

3.2.8 Faible besoin matériel

Le logiciel PyScada nécessite de faibles besoins matériels, il peut être installé sur un ordinateur de bureau comme sur un serveur. Il est recommandé d'avoir au moins 1 Go de mémoire vive, 1 Go d'espace disque et un processeur équivalent à l'ARM Cortex-A53. Pour des applications se connectant à une dizaine d'instruments, et où moins d'une dizaine d'utilisateurs se connectent de manière simultanée, des nano-ordinateurs comme le Raspberry Pi [30] peuvent être utilisés.

3.2.9 Évaluation face aux principes du FAIR des logiciels

Le tableau 3.1 présente une illustration des principes proposés dans LAMPRECHT et al. [15] en les utilisant pour évaluer PyScada (voir section 1.2.2). Cette évaluation montre qu'il est nécessaire d'améliorer en priorité les deux points négatifs à savoir : inclure la version du logiciel dans les méta-données et définir un vocabulaire contrôlé pour décrire le logiciel et les méta-données.

TABLE 3.1 – Évaluation de PyScada pour les principes des logiciels FAIR énoncés par Lamprecht.

Qualificatif	Sigle	Définition	Satisfait	Commentaire
Trouvable	*F1	Le logiciel et ses méta-données associées ont un identifiant global, unique et persistant pour chaque version publiée.	Oui (**partiel**)	Les versions Github ont un nom de balise unique. Il n'y a pas de PID persistant.
	*F2	Les logiciels sont décrits avec des méta-données riches.	Oui	Le dépôt Github contient la liste des auteurs, le journal des modifications, la licence et la notice d'utilisation (*readme*), ainsi que des liens vers le forum des problèmes rencontrés et le code source.
				Suite à la prochaine page

30. https://www.raspberrypi.com

Table 3.1 – suite de la page précédente

Qualificatif	Sigle	Définition	Satisfait	Commentaire
	*F3	Les méta-données incluent clairement et explicitement les identifiants de toutes les versions du logiciel qu'elles décrivent.	**Non**	Les méta-données ne contiennent pas la version à laquelle elles s'appliquent.
	*F4	Les logiciels et leurs méta-données sont inclus dans un registre de logiciels consultable.	Oui	**PyPI** [31] et **Github** [32].
Accessible	*A1	Les logiciels et leurs méta-données associées sont accessibles par leur identifiant au moyen d'un protocole de communication normalisé.	Oui	Les méta-données et le logiciel sont accessibles via des pages web **HTTPS**.
	*A1.1	Le protocole est ouvert, gratuit et universellement implémentable.	Oui	Les logiciels sont accessibles par le biais de plateformes ouvertes et gratuites, accessibles à tous.
	*A1.2	Le protocole permet une procédure d'authentification et d'autorisation, si nécessaire.	Oui	Aucune authentification ou autorisation n'est nécessaire.
	*A2	Les méta-données des logiciels sont accessibles, même lorsque le logiciel n'est plus disponible.	Oui	Les méta-données des logiciels sont toujours accessibles sur Github et PyPI.
Interopérable	*I1	Le logiciel et ses méta-données associées utilisent un langage formel, accessible, partagé et largement applicable pour faciliter la lisibilité par la machine et l'échange de données.	Oui (**partiel**)	Les logiciels sont écrits en Python, HTML, Javascript et CSS qui sont lisibles par les machines. Les méta-données dans Github sont écrites en **texte brut**.
				Suite à la prochaine page

31. https://pypi.org/project/PyScada/
32. https://github.com/pyscada/pyscada/

Table 3.1 – suite de la page précédente

Qualificatif	Sigle	Définition	Satisfait	Commentaire
	*I2S.1	Les logiciels et leurs méta-données sont formellement décrits à l'aide de **vocabulaires** contrôlés qui respectent les principes FAIR.	**Non**	Les termes utilisés ne suivent pas un vocabulaire contrôlé.
	*I2S.2	Les logiciels utilisent et produisent des types et des formats de données qui sont formellement décrits à l'aide de vocabulaires contrôlés conformes aux principes du FAIR.	Oui	Le logiciel produit des exportations de données sous forme de fichiers **CSV** ou **HDF5**. Les données sont accessibles via une API web.
	*I4S	Les dépendances des logiciels sont documentées et il existe des mécanismes pour y accéder.	Oui	Les dépendances sont listées dans la notice d'utilisation (***readme***) et dans les fichiers d'installation.
Réutilisable	*R1	Les logiciels et leurs méta-données sont décrits de manière riche avec une pluralité d'attributs précis et pertinents.	Oui	Voir les commentaires de *R1.1 et *R1.2.
	*R1.1	Le logiciel et ses méta-données associées ont des licences d'utilisation indépendantes, claires et accessibles, compatibles avec les dépendances du logiciel.	Oui	Le logiciel utilise la Licence Publique Générale GNU (GNU GPL) version 3.0 telle que publiée par la Free Software Foundation.
	*R1.2	Les méta-données du logiciel incluent une provenance détaillée, le niveau de détail doit être convenu par la communauté.	Oui (**partiel**)	Publication du code sur Github depuis la version 0.1. **Aucune provenance** dans les méta-données restantes.

Suite à la prochaine page

Table 3.1 – suite de la page précédente

Qualificatif	Sigle	Définition	Satisfait	Commentaire
	*R1.3	Les méta-données et la documentation du logiciel sont conformes aux normes communautaires pertinentes pour le domaine.	Oui (**partiel**)	La documentation est disponible sur **ReadTheDocs** [33]. Les méta-données ne suivent **aucune norme** communautaire.

3.3 Bilan

Dans ce chapitre j'ai présenté les différents apports réalisés durant ma thèse pour répondre aux problématiques et besoins identifiés en section 2.4. Cette approche novatrice de la supervision et de l'instrumentation à distance se caractérise par :

- différentes abstractions nécessaires pour offrir une solution logicielle interopérable, séparant l'application en différents modules,
- un logiciel organisé autour d'une architecture permettant de :
  - présenter l'ensemble d'une expérimentation distante dans une interface web accessible depuis tout terminal informatique récent,
  - échanger des informations avec des capteurs et actionneurs afin d'étudier des systèmes physiques quel que soit le domaine d'application,
  - traiter des données en leur donnant du sens à l'aide d'algorithmes ou d'une gestion événementielle,
  - s'adapter aux différents types d'utilisateurs et proposer différents niveaux techniques d'accès à une instrumentation,
  - le faire évoluer face à de nouvelles problématiques via une approche agnostique,
- la liberté d'utiliser, modifier et distribuer le logiciel original ou modifié.

Le tableau 3.2 présente l'évolution de PyScada durant ma thèse face aux 8 critères définis en section 2.4.

TABLE 3.2 – Comparaison de PyScada avec les critères définis en section 2.4 (O = Oui ; N = Non ; ? = Inconnu).

	Critères								
Solution	1	2	3	4	5	6	7	8	Score
PyScada pre-thèse	O	N	O	O	O	O	N	N	5
PyScada post-thèse	O	O	O	O	O	O	O	O	8

Il reste encore plusieurs points à développer comme la mise en place de **PID** pour toute donnée, processus ou algorithme de la solution proposée (voir section 1.3.2), l'ajout de la version du logiciel dans les méta-données, la création d'un **vocabulaire** défini pour utiliser, configurer et développer le logiciel et la possibilité de décentraliser toute donnée en créant une **fédération** de logiciels de supervision et d'instrumentation.

33. https://pyscada.readthedocs.io/

Chapitre 4

Applications

Sommaire

4.1 Enseignement hybride 69
4.1.1 Laborem Box 73
4.1.2 Comportement étudiant 76
4.1.3 Futures études autour de la plateforme Laborem 77
4.2 Supervision d'ouvrages du littoral 78
4.2.1 Première station : digue de l'Artha 78
4.2.2 Seconde station : fort de Socoa 82
4.2.3 Troisième station : structure mobile 84
4.2.4 Vers un maillage de la côte basque 85
4.2.5 Études en laboratoire 85
4.3 Suivi énergétique du bâtiment 86
4.3.1 Première expérimentation : IUT de Bayonne 87
4.3.2 Seconde expérimentation : Domolandes 87
4.3.3 Troisième expérimentation : Elis 88
4.4 Bilan, validation et évaluation 88

Ce chapitre a comme objectif de présenter les différentes applications où PyScada est utilisé, ses limites et ses futurs développements. Chaque installation a des besoins différents et fait appel aux notions de télémesure, télécontrôle, interopérabilité, décentralisation, stockage de données, automatisation, accès distant, accessibilité et réutilisation décrites au chapitre 1.

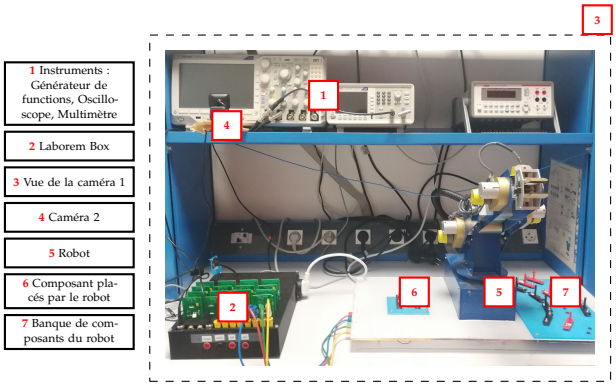
Le projet éducatif Laborem sera présenté dans un premier temps. Ensuite les travaux de recherche utilisant la supervision d'ouvrages du littoral basque seront détaillés. Enfin l'utilisation de PyScada pour l'étude de données énergétiques de différents bâtiments sera exposée.

Ces trois exemples applicatifs ont vocation à illustrer la généricité et la versatilité du logiciel PyScada.

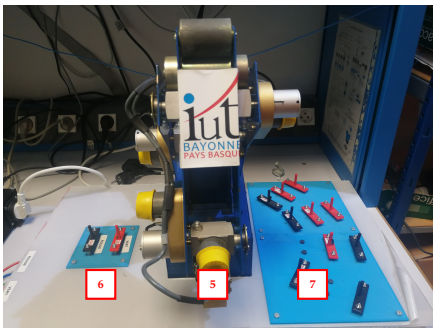
4.1 Enseignement hybride

Le projet Laborem développé à l'institut universitaire de technologie (IUT) de Bayonne depuis 2009, permet à des étudiants de premier cycle de réaliser à distance une partie de leurs travaux pratiques en électronique. Il vise à proposer des travaux pratiques d'électronique à distance via une plateforme logicielle et matérielle peu coûteuse, simple à déployer, *open source* et évolutive. La première version de Laborem était basée sur du matériel et des logiciels propriétaires, utilisant des solutions de National Instruments, dont le logiciel **LabVIEW** et la carte de prototypage **Elvis Protoboard**. Cette solution initiale était coûteuse et complexe à reproduire. Cependant,

elle a permis de tester le concept d'un laboratoire à distance dans notre université (voir LUTHON et al. [2] et LETOWSKI et al. [47]). Un des objectifs de Laborem est de maximiser la motivation et l'immersion de l'étudiant en proposant une liste de circuits, dont certains sont à réaliser à l'aide d'un robot et à l'aide de caméras permettant de visualiser la plateforme (instruments, robot et Laborem Box, voir figure 4.1).



(A) Banc d'essai de Laborem.



(B) Robot.

FIGURE 4.1 – Vue d'ensemble du banc d'essai Laborem.

Depuis 2018 la plateforme a migré vers le logiciel libre et ouvert PyScada. Depuis 2019 nous avons développé un boîtier d'interface *open source* (**Laborem Box**) afin de permettre la connexion de plusieurs circuits imprimés à étudier en remplacement de la carte de prototypage **Elvis Protoboard**. La Laborem Box se compose d'une boîte imprimable en 3D, d'une carte d'alimentation, d'un ensemble de cartes électroniques amovibles, appelées *plugs*, et d'une carte mère qui peut être connectée à plusieurs instruments, permettant aux étudiants d'étudier le circuit sélectionné. Ces *plugs* sont facilement interchangeables et permettent aux enseignants d'adapter rapidement à leur cours les circuits proposés. En outre, un nano-ordinateur mono-carte est intégré et un disque dur peut être utilisé si nécessaire pour augmenter les capacités de stockage et prolonger la durée de vie de la carte mémoire du nano-ordinateur. En effet une base de données réalise beaucoup d'écritures sur l'espace de stockage, endommageant rapidement les cartes mémoires. Le logiciel offre également une interface simple adaptable aux besoins des étudiants et enseignants.

A ce jour, il n'existe pas de laboratoire à distance proposant des cartes électroniques *open source* développées pour réaliser des travaux pratiques (voir tableau 2.1, p. 37). L'*Open Source Hardware Association* [1] (OSHWA) définit le matériel *open source* (OSHW de l'anglais *Open Source HardWare*) comme des machines, des dispositifs ou d'autres objets physiques dont la conception a été rendue publique afin que chacun puisse fabriquer, modifier, distribuer et utiliser ces objets. Cependant, il a été observé qu'il existe une tendance à utiliser du matériel existant à faible coût ou des logiciels libres pour améliorer la standardisation et la simplicité d'utilisation. De nombreux laboratoires d'enseignement à distance utilisent des ordinateurs mono-carte, tels que le **Raspberry Pi**, pour contrôler leur équipement (voir BERMÚDEZ-ORTEGA et al. [41], KALUZ et al. [48] et RECK et al. [49]). Cela permet de concevoir des plateformes à faible coût par rapport à une station de travail ou à un serveur (voir GUERRERO-RODRÍGUEZ et al. [50]). Pour réaliser des circuits électriques et connecter plusieurs instruments, certains utilisent encore des cartes de prototypage (voir RECK et al. [49]) comme notre projet avant 2017, d'autres conçoivent leurs propres circuits imprimés

1. https://www.oshwa.org/

(PCB de l'anglais *Printed Circuit Board*) (voir GUERRERO-RODRÍGUEZ et al. [50] et VILLAR-MARTÍNEZ et al. [40]). D'autres laboratoires utilisent des micro-contrôleurs comme **Arduino**, qui peuvent être programmés pour acquérir et générer des signaux électriques afin d'effectuer une grande variété de tâches telles que l'éclairage, le chauffage, le contrôle de robots, l'informatique embarquée, etc (voir RECK et al. [49], GUERRERO-RODRÍGUEZ et al. [50], VILLAR-MARTÍNEZ et al. [40], SUKOP et al. [51], WANG et al. [52] et TRAN et al. [53]).

Dans l'interface de Laborem, les étudiants sont invités à répondre à des questions lorsqu'ils réalisent une expérience, et ils peuvent également se comparer aux autres via un classement (*TOP10*, qui a été conçu pour ressembler à la partie *hall of fame*, pour temple de la renommée, d'un jeu, voir LUTHON et al. [54]).

Afin de gérer plusieurs étudiants connectés en même temps sur la plateforme, une gestion de file d'attente a été mise en place. La plateforme Laborem gère différents groupes d'utilisateurs n'ayant pas les mêmes droits d'accès et d'action sur les contenus de l'interface utilisateur. Certains groupes servent à organiser les étudiants dans la file d'attente. Les quatre groupes sont :

- l'étudiant qui manipule (expérimentateur) : ce groupe ne peut contenir qu'un seul utilisateur ayant le droit de réaliser des expériences. C'est le premier étudiant dans la file d'attente. Dès qu'un autre élève apparaît dans la file d'attente, cet étudiant dispose d'un temps limité avant de passer le relais au prochain élève de la file,
- les étudiants regardant celui qui manipule (observateurs) : ce groupe permet le travail en binôme ou plus. Les utilisateurs de ce groupe ne peuvent pas manipuler mais uniquement voir les paramètres entrés par l'étudiant qui manipule ainsi que les résultats obtenus (par exemple : configuration des instruments, visualisation des courbes),
- les étudiants qui attendent leur tour : ce groupe dispose seulement de son temps d'attente, de son rang dans la file d'attente et du classement des étudiants au *TOP10*,
- enseignant : peut accéder à une interface supplémentaire permettant de :
  - visualiser le *plug* sélectionné par l'étudiant qui manipule,
  - éteindre et allumer les instruments connectés à la Laborem Box,
  - visualiser les réponses et scores des étudiants aux questions du *TOP10*,
  - changer le temps disponible pour l'expérimentateur quand un étudiant est présent dans la file d'attente,
  - changer le nombre d'étudiants pouvant voir ce que fait celui qui manipule pour permettre du travail collaboratif.

La figure 4.2 présente la page d'accueil de l'interface utilisateur. L'interface enseignant, à gauche sur la figure 4.2, n'est visible que pour les utilisateurs du groupe enseignant. Les étudiants sélectionnent "Laborem RemoteLab" pour entrer dans le laboratoire distant.

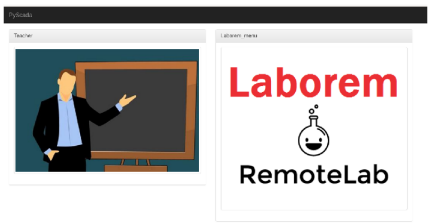
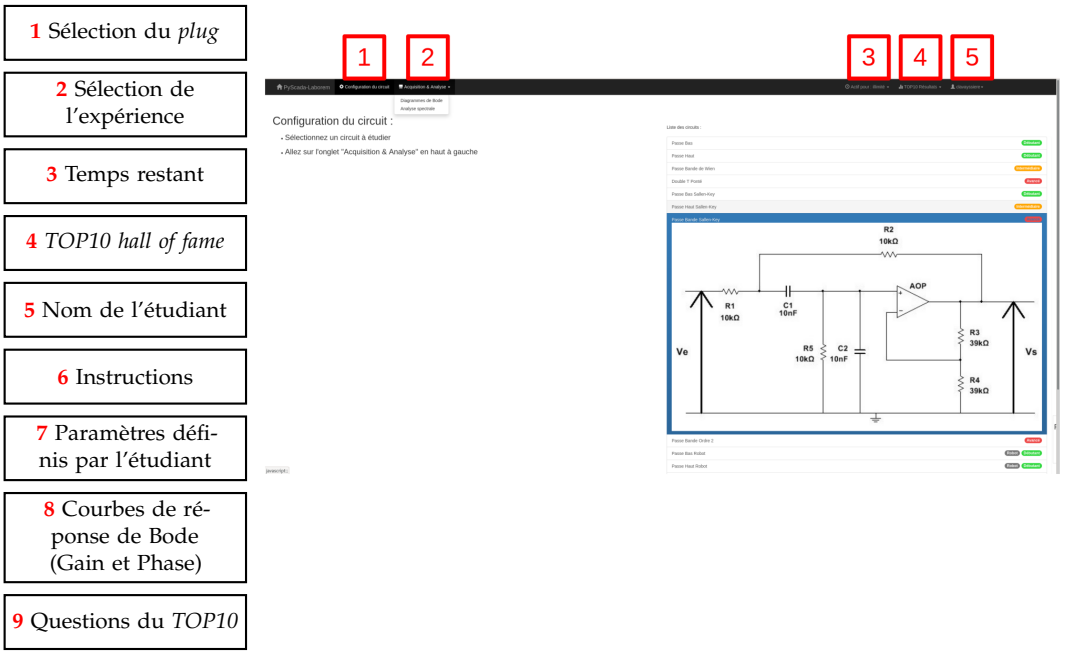


FIGURE 4.2 – Accès aux interfaces enseignant (à gauche) et étudiant (à droite) de Laborem.

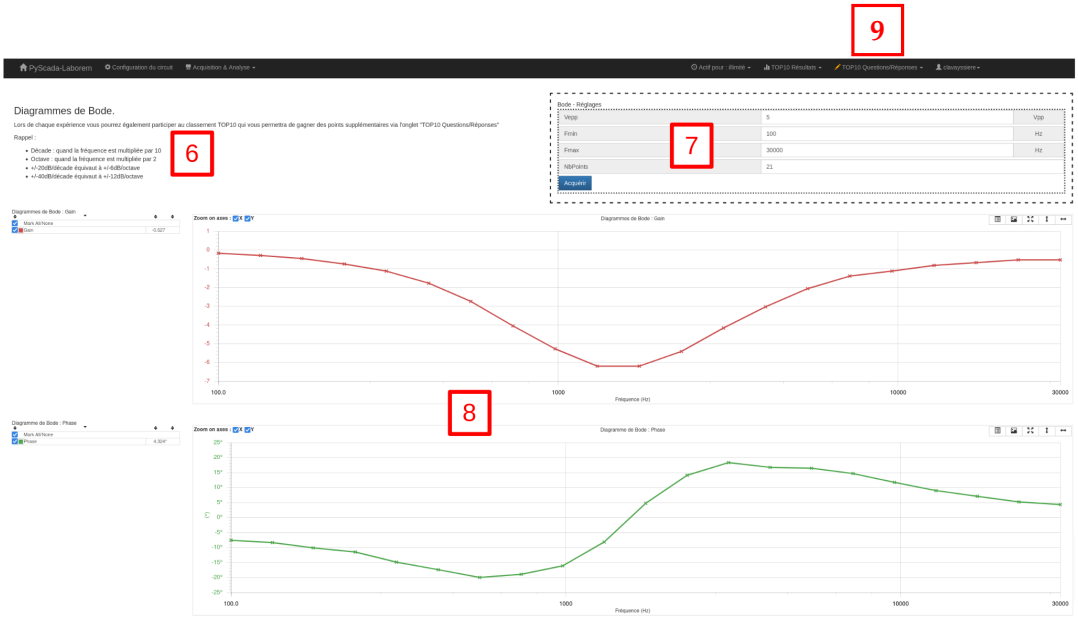
L'interface de l'étudiant est composée de trois parties, accessibles via la barre de navigation :

— l'**accueil** explique l'objectif des travaux pratiques à distance, la gestion de la file d'attente et du *TOP10*.
— **Sélection du *plug*** permet de choisir le circuit à étudier.
— **Expériences** liste les expériences disponibles, comme l'étude des diagrammes de Bode ou l'analyse fréquentielle d'un filtre.

La page de sélection du *plug* et la page de l'expérience permettant de réaliser des études de diagrammes de Bode sont présentées en figure 4.3.



(A) Configuration du circuit.



(B) Résultats d'un diagramme de Bode.

FIGURE 4.3 – L'interface étudiant simple à utiliser.

Les différentes fonctionnalités spécifiques à Laborem que l'on retrouve dans

PyScada s'ajoutent à l'aide d'un *plugin* nommé **PyScada-Laborem** disponible sur GitHub [2]. Ainsi il est très facile de dupliquer et réutiliser l'ensemble de l'interface et de la configuration de Laborem.

L'interface administrateur de PyScada permet d'accéder à l'ensemble de la configuration de Laborem :

— éditer l'IHM de l'étudiant et de l'enseignant (section **IHM** de PyScada)
— modifier les scripts gérant les expériences (section PyScada ***Scripting***)
— modifier les instruments connectés à la carte mère (section PyScada ***Core***)
— modifier les circuirs étudiés et la liste subséquente des *plugs* apparaissant sur l'interface de l'étudiant (section PyScada Laborem)
— modifier l'ensemble des questions posées et visualiser les réponses de l'étudiant (section PyScada Laborem).

4.1.1 Laborem Box

Dans cette thèse, nous avons fait le choix des logiciels libres et, autant que possible, d'utiliser du matériel *open source* pour l'architecture de Laborem, centrée sur la Laborem Box (voir figure 4.4).

Cette cohérence, à proposer du matériel et du logiciel libre, permet le déploiement, l'utilisation et la modification par tous du système Laborem.

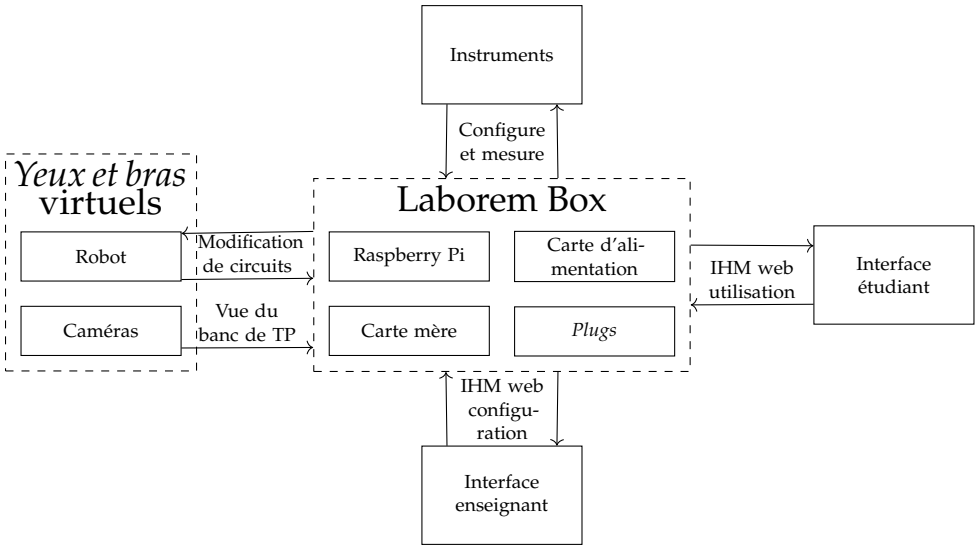
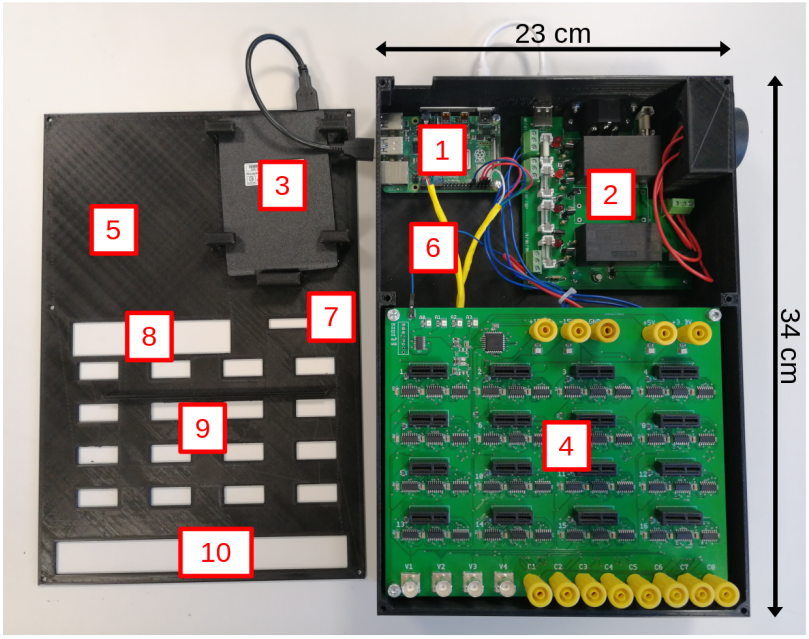


FIGURE 4.4 – Vue d'ensemble de l'architecture du système Laborem.

La Laborem Box repose sur un serveur, qui est ici un Raspberry Pi, hébergeant l'IHM proposée par PyScada. Le boîtier réalisé avec une imprimante 3D, composé de deux parties (face avant et face arrière), est conçu pour contenir la carte mère, la carte d'alimentation, le Raspberry Pi et un disque dur optionnel (voir figure 4.5). Il sécurise l'installation et ne doit être ouvert que lorsque le système est hors tension et que tous les câbles sont débranchés.

La carte d'alimentation (voir figure 4.5), qui est alimentée par une prise principale (220V AC), fournit une tension continue au Raspberry Pi (5V, 3A) et des tensions continues (+/-15V, 5V, 3.3V) à la carte mère, aux *plugs* connectés et aux circuits intégrés (commutateurs, multiplexeur, amplificateurs opérationnels). De plus, la carte d'alimentation est équipée d'un relais qui permet de déconnecter une multiprise

2. https://github.com/clavay/PyScada-Laborem



1 Raspberry Pi

2 Carte d'alimentation

3 Disque dur

4 Carte mère

5 Dessus de la boite 3D

6 Dessous de la boite 3D

7 Fenêtre pour l'affichage du *plug* sélectionné

8 Fenêtre des connecteurs d'alimentation

9 Fenêtres des *plugs*

10 Fenêtres des connecteurs BNC et banane

FIGURE 4.5 – Les composants de la Laborem Box.

où sont connectés les instruments externes (oscilloscope, multimètre, générateur de fonctions, robot, lampe éclairant le laboratoire et si besoin une alimentation externe de puissance) afin d'économiser de l'énergie lorsque la plateforme n'est pas utilisée par un étudiant. Toutes les tensions de sortie sont protégées par des fusibles.

Un circuit imprimé appelé carte mère (voir figure 4.6) est conçu afin de :

— connecter tous les éléments électroniques de Laborem,
— fournir les alimentations et tous les signaux utiles (typiquement les entrées/sorties) sur chaque *plug*,
— concevoir un système robuste et facile pour changer rapidement les *plugs* (emplacements PCI Express).

La carte mère dispose de 4 connecteurs d'entrée/sortie polyvalents (GPIO de l'anglais *General Purpose Input/Output*) qui permettent de sélectionner un *plug* spécifique à étudier parmi les 16 *plugs* connectés. Le *plug* est sélectionné via un multiplexeur contrôlé par 4 bits provenant des GPIO du Raspberry Pi. 12 entrées/sorties (4 BNC de V_1 à V_4, et 8 connecteurs banane de C_1 à C_8) sont également disponibles pour connecter facilement d'autres signaux provenant d'instruments périphériques tels qu'un oscilloscope, un multimètre, un générateur de fonctions, un bras robotique ou des relais (voir la figure 4.7). Les tensions de signal sont limitées entre -15V et +15V avec une intensité maximale de 1A.

Les *plugs* sont conçus pour offrir un système robuste et facile à changer qui permet de construire et d'étudier n'importe quel circuit électrique.

Un bras robotique optionnel (5 degrés de liberté) est utilisé pour la modification des circuits (voir LUTHON et al. [54] et LUTHON et al. [55] et la vidéo de démonstration [3]). Deux caméras remplacent les yeux de l'élève pour voir tous les équipements du banc de travail (voir la figure 4.1) permettant de suivre les mesures en cours. Le robot donne la possibilité à l'enseignant de proposer des circuits incomplets (avec 2

3. https://github.com/clavay/PyScada-Laborem/raw/master/extras/Laborem_robot.mp4

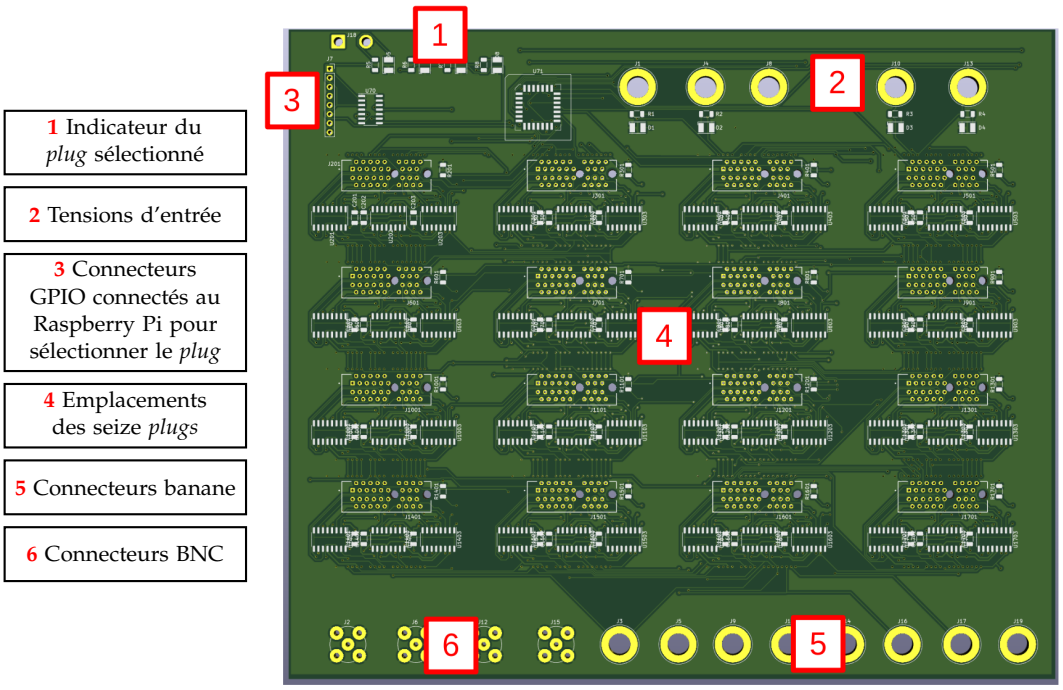


FIGURE 4.6 – Vue de face de la conception 3D de la carte mère sans les composants soudés.

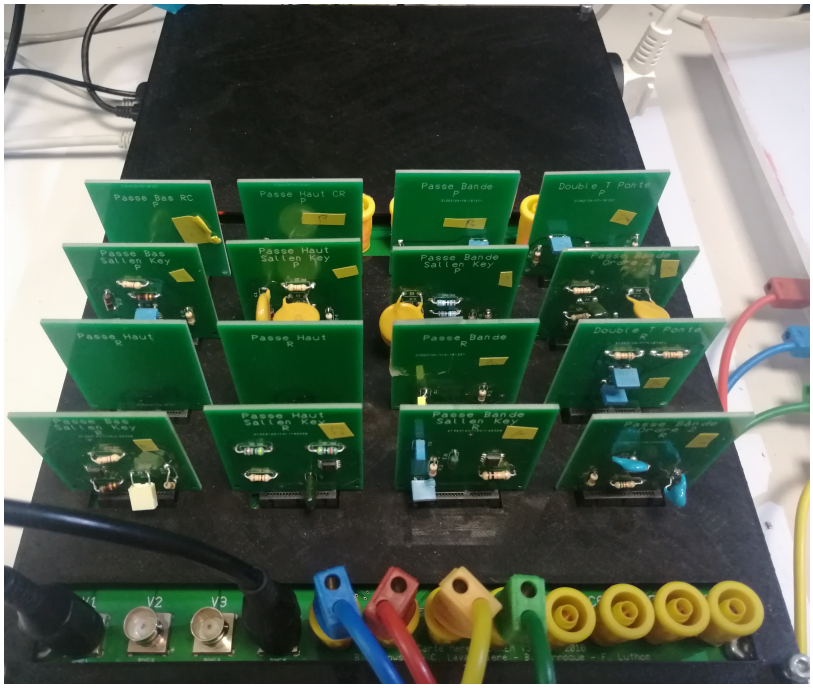


FIGURE 4.7 – *Plugs* et câbles BNC et bananes connectés à la Laborem Box.

composants manquants) que l'élève doit compléter en choisissant des composants disponibles dans la banque de composants, qui sont ensuite placés grâce à la pince du bras robotique (voir figure 4.1, p.70).

La particularité de la Laborem Box est de permettre simplement aux enseignants de :

- proposer leurs travaux pratiques électroniques aux élèves via un accès à distance et à l'aide d'une interface d'administration simple,
- choisir les instruments nécessaires aux travaux pratiques,
- connecter les instruments spécifiques à l'aide de connecteurs BNC et Banane,
- créer les *plugs* nécessaires à l'étude d'un ensemble de circuits électriques,
- combiner différents circuits en un seul *plug*, en utilisant des interrupteurs pour sélectionner le sous-circuit souhaité,
- changer rapidement les *plugs* pour proposer facilement d'autres travaux pratiques.

Les instruments actuellement connectés à la Laborem Box utilisent les standards SCPI et VISA. Ils sont connectés au Raspberry Pi via des ports GPIB ou USB. Tout instrument utilisant un protocole géré par PyScada peut être utilisé comme par exemple : VISA, Modbus, I2C, OneWire, BACnet, OPC-UA, WebService, Serial.

L'article LAVAYSSIÈRE et al. [56] que nous avons publié dans le journal HardwareX détaille l'ensemble des instructions nécessaires pour construire et assembler une Laborem Box. Les plans sont disponibles sur Zenodo [4].

4.1.2 Comportement étudiant

La plateforme Laborem est intégrée à un système de gestion de l'apprentissage (LMS de l'anglais *Learning Managment System*) *open source* appelé Moodle. Les accès et les résultats de chaque utilisateur à Laborem et à Moodle peuvent être analysés pour comparer différents usages du laboratoire distant. La session d'utilisation de Laborem présentée dans LAVAYSSIÈRE et al. [57] s'inscrit dans un scénario hybride où les étudiants ont suivi une première séquence du cours par une activité pratique en présentiel, suivie d'une session d'apprentissage à distance. Pour la deuxième séquence, le scénario type (illustré à la figure 4.8) consiste en une présentation vidéo et un cours en ligne (étape 1), un test de pré-requis et un test formatif (étape 2), deux travaux pratiques à distance avec trois niveaux de difficulté (débutant, intermédiaire ou avancé) utilisant la plateforme Laborem (étapes 3 et 4), et un test final sommatif accompagné du téléchargement du rapport de l'étudiant suivi d'une enquête de satisfaction (étape 5). Les étudiants peuvent échanger entre eux ou avec les enseignants en utilisant un forum en ligne sur la plateforme Moodle.



FIGURE 4.8 – Scénario en cinq étapes en ligne dans Moodle.

L'ordonnancement et la gestion des files d'attente pour les laboratoires à distance ont un impact significatif sur les temps d'attente et les niveaux d'utilisation des ressources (voir LOWE [58]). L'apprentissage en binôme en programmation a montré ses

4. https://doi.org/10.5281/zenodo.5564369

avantages en termes de qualité du travail produit et de productivité (voir WILLIAMS et al. [59] et SALLEH et al. [60]). Il tend à améliorer le plaisir, à accroître la confiance des étudiants, à réduire la charge de travail, à améliorer le taux de réussite des cours, à augmenter le taux de soumission des devoirs et à améliorer les résultats des examens. Afin d'analyser l'influence de ces critères dans le contexte des laboratoires à distance, l'influence de la gestion du temps et de l'apprentissage en binôme a été étudiée.

La classe suivie en 2021 sur la plateforme Laborem était composée de 42 étudiants de première année de DUT de l'IUT de Bayonne. Pour déterminer l'efficacité et la satisfaction de Laborem, la classe d'étudiants a été divisée en trois groupes afin de comparer différents scénarios d'utilisation. En modifiant deux critères, à savoir le passage individuel ou en binôme, et l'accès temporel libre ou contraint, l'utilisation (fréquence, durée, régularité), les résultats (réalisation, notes) et la satisfaction (sentiments, attentes) de chaque groupe sont comparés. Le premier groupe a disposé de 14 jours entièrement libres pour réaliser le sujet de travaux pratiques en utilisant Moodle et Laborem. Il avait accès à la plateforme à tout moment. Le deuxième groupe avait une journée réservée à chaque étudiant, plus 7 jours d'accès libre. Le dernier groupe était formé de binômes qui avaient un jour réservé pour chaque binôme, plus 6 jours d'accès libre.

Les étudiants travaillant individuellement (groupes 1 et 2) utilisèrent principalement le dernier jour disponible. Afin de répondre à cette gestion du temps, un laboratoire à distance doit être robuste à un grand nombre de connexions simultanées ou imposer un horaire de passage pour un groupe restreint afin de prévoir le nombre maximum de connexions simultanées.

Afin de maintenir la motivation étudiante pendant un travail à distance de plusieurs heures, ce qui est la mission normale d'un enseignant lors d'un cours en présentiel, il faut disposer d'outils spécifiques. Ces outils doivent servir à contrôler la progression des étudiants et les avertir (détection automatique par le laboratoire distant, remplaçant le suivi manuel de l'enseignant), mais aussi à les renforcer ou les récompenser (éviter l'abandon, la tricherie, solliciter l'étudiant, l'inciter par le jeu...). Pour approfondir l'intérêt de la collaboration pour les travaux pratiques à distance, il semble nécessaire de disposer d'outils fiables permettant le travail en binôme (vidéo, partage d'écran, contrôle à distance, conversation) et de mesurer et suivre l'utilisation de ces outils. L'enseignant doit pouvoir assister le travail individuel en autonomie afin d'éviter qu'un élève soit bloqué ou laisse des réponses vides. Ce besoin pourrait être résolu à l'aide de balises ou de garde-fous (avertissements contextuels).

Que ce soit du point de vue de l'étudiant (motivation, ressenti), de l'utilisation (régularité, fréquence, durée) ou des résultats pédagogiques (score, complétion), notre étude a démontré l'intérêt significatif du travail en binôme dans l'utilisation d'un laboratoire à distance, alors que le travail autonome en accès semi-restreint par rapport à l'accès libre n'apportait pas beaucoup de valeur ajoutée.

4.1.3 Futures études autour de la plateforme Laborem

Puisque les laboratoires distants doivent être faciles à adapter et à dupliquer, le choix de Laborem d'utiliser des logiciels et du matériel libres et ouverts, des protocoles standard et du matériel à faible coût offre une solution intéressante. Cet avantage nous a permis d'adapter rapidement la plateforme à la fermeture totale ou partielle des universités françaises en mars 2020 pour cause de pandémie. Il peut aussi être utile pour facilement s'adapter aux changement de programmes par l'ajout de nouveaux circuits, comme lors du passage du DUT vers le BUT. La mise en œuvre massive de la formation à distance dans le domaine des sciences

de l'ingénieur en réponse aux enjeux économiques est à mettre en perspective avec l'intérêt de l'étudiant et de l'enseignant. Afin de maintenir un niveau d'attention élevé, il est nécessaire que l'étudiant assiste régulièrement aux cours, ce qui ne peut être compensé par l'auto-apprentissage ou par l'apprentissage asynchrone en ligne (voir SOKELE et al. [61]).

Notre travail vise maintenant à améliorer et à identifier les limites de l'utilisation massive des travaux pratiques à distance afin de répondre aux problèmes d'isolement des étudiants. Par ailleurs, notre étude se concentre sur les domaines technologique et pédagogique et laisse de côté le domaine psychologique. Or, les recherches psychologiques en sciences cognitives suggèrent que les expériences et les croyances des étudiants peuvent être déterminées davantage par les caractéristiques et la nature des interfaces que par la réalité objective de la technologie du laboratoire (voir MA et al. [62]).

Les problématiques de la supervision et de l'instrumentation à distance abordées dans le projet Laborem sont celles de la réutilisation et la duplication simple de la plateforme, de l'automatisation, de l'interopérabilité humaine (étudiants et enseignants) et des protocoles de communication utilisés. La plateforme utilise des scripts et des *plugins* pour personnaliser l'IHM et répondre aux besoins des étudiants et des enseignants.

La solution technique proposée par le projet Laborem permet d'intégrer facilement le matériel spécifique développé (Laborem Box) et d'étudier les impacts pédagogiques sur les étudiants ainsi que les méthodes de travail des enseignants dans le cadre de cours hybrides. La richesse de la contribution exposée dans ce projet est possible grâce à l'interopérabilité humaine et matérielle mise en œuvre, à l'interface modulaire et évolutive ainsi qu'à la valeur extraite des données provenant du système instrumenté.

4.2 Supervision d'ouvrages du littoral

Le déferlement de vagues est étudié dans le but de mieux comprendre ce phénomène qui endommage les côtes et les ouvrages qui les protègent. Ceci permet d'améliorer la conception des structures côtières en supervisant les installations. Beaucoup d'études sont réalisées en laboratoire du fait de la difficulté d'étude dans un environnement aussi hostile. Cependant les études en laboratoire présentent des limites du fait des fluides utilisés, des différences d'échelle, de paramètres manquants (vent, courant) ou d'analyse seulement en deux dimensions, et ne représentent pas la réalité.

Les travaux de recherche menés au sein du laboratoire SIAME et son équipe IVS cherchent à comprendre les phénomènes physiques endommageant le littoral basque pour mieux renforcer les digues et prévoir les évènements extrêmes (forte pression d'impact, vagues de submersion, érosion).

4.2.1 Première station : digue de l'Artha

La digue de l'Artha est située dans la baie de Saint-Jean-de-Luz en France (voir figure 4.9).

L'objectif de la station de supervision de la digue de l'Artha installée depuis 2017 est d'avoir une station fonctionnant en permanence sur un site isolé (sans raccordement à un réseau électrique) pouvant récolter des données de pression en deux dimensions sur un flanc de la digue, quelles que soient les conditions météorologiques. Ces mesures recueillies en continu génèrent un grand nombre de données

1 Digue de l'Artha

2 Fort de Socoa



FIGURE 4.9 – Baie de Saint-Jean-de-Luz, France.

qui sont ensuite rapatriées et analysées par des océanographes pour comprendre la dynamique des vagues et suivre l'évolution des déformations subies par la structure (voir PONCET et al. [63]).

La digue de l'Artha, située à 200 mètres de la digue de Socoa mesure 250 mètres. Des blocs de béton de plusieurs tonnes (typiquement cinquante tonnes) sont immergés en amont de la digue pour la protéger. Les vagues peuvent être suffisamment puissantes pour déplacer ces blocs comme le montre la figure 4.10. Un bloc a été déposé sur la digue par une vague lors de la tempête de 1951, évènement qui s'est répété en 2017 et dont un photographe a saisi l'instant.

Seize capteurs de pression sont installés sur le musoir nord ouest de la digue (voir figure 4.11c). La station de mesure est située sur le flanc sud de la digue (voir figure 4.11a). La figure 4.11b détaille les composants de l'installation accessible à l'aide d'un routeur connecté au réseau de téléphonie mobile de quatrième génération (4G) et autonome en énergie via un panneau solaire rechargeant une batterie. Une centrale de mesure (CompactRIO de National Instruments) se charge de l'acquisition haute fréquence (10 kHz) des seize capteurs et de l'enregistrement des données dans des fichiers binaires. Une fois un enregistrement terminé, il est conservé s'il respecte des paramètres prédéfinis : au moins un impact a été identifié, la pression dépassant un seuil fixé.

PyScada est installé sur un Raspberry Pi. Son interface (voir figure 4.12) permet de :

- connaître l'état de la centrale de mesure (acquisition en cours, archivage des données, attente de la prochaine acquisition),
- afficher et modifier les paramètres de mesure (fréquence d'acquisition, mode manuel ou automatique d'acquisition, durée de la mesure en secondes, temps restant de l'acquisition en cours),
- lancer une acquisition manuelle,
- connaître la valeur des capteurs de pression,
- afficher l'état du routeur (signal, quantité de données transmises),
- afficher l'état du stockage sur le Raspberry Pi et les différentes connexions Ethernet avec le routeur, la centrale de mesure et le réseau virtuel privé (VPN



(A) Bloc de béton de 40 tonnes déposé sur la digue de l'Artha lors de la tempête de 1951.



(B) Bloc de béton de 50 tonnes déposé sur la digue de l'Artha en 2017.



(C) Séquence photo de la vague déposant le bloc de 50 tonnes.

FIGURE 4.10 – Déplacement des blocs de béton protégeant la digue de l'Artha.

de l'anglais *Virtual Private Network*) de l'université,

— lister les enregistrements en cours de transfert vers les serveurs du laboratoire.

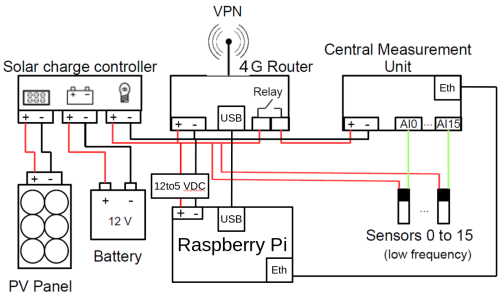
PyScada communique avec la centrale de mesure à l'aide de *Web Services*, avec le routeur via des commandes AT (de l'anglais *Attention*, définies dans la norme ETSI GSM 07.07[5]) et avec un relais qui éteint la centrale de mesure afin d'alléger la consommation du système quand des mesures ne sont pas prévues. Différentes alertes sont configurées et envoient un mail si le niveau de la batterie ou si l'espace disque du Raspberry Pi deviennent trop faibles.

PyScada permet une interopérabilité des librairies graphiques c'est-à-dire de changer de librairie pour un besoin particulier. Avec les progiciels classiques basés sur un navigateur, tous les points de données sont transmis directement au navigateur pour être affichés, ce qui permet une interaction spécifique avec chaque courbe ou point, y compris l'affichage des méta-données, les liens vers les sources, etc. Cette approche offre la plus grande flexibilité par point ou par courbe, mais se heurte rapidement aux limites de la quantité de données pouvant être traitées par le navigateur et de la quantité pouvant être affichée à l'écran et résolue par le système visuel humain.

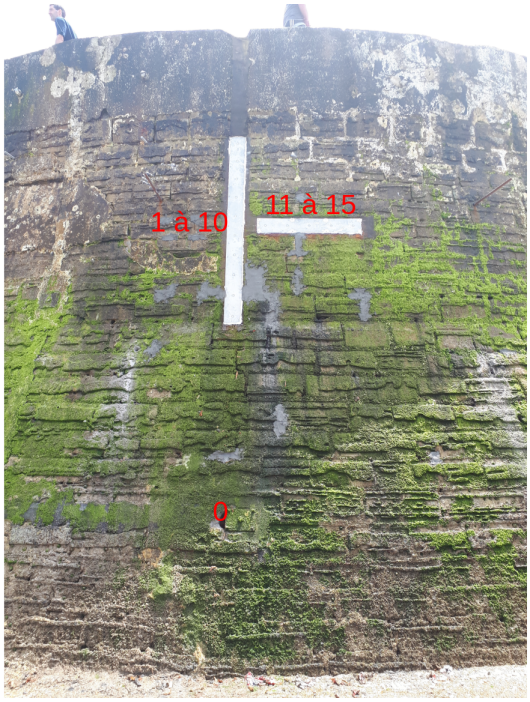
5. https://www.etsi.org/deliver/etsi_gts/07/0707/05.00.00_60/gsmts_0707v050000p.pdf



(A) Station de supervision de la digue de l'Artha. Les capteurs sont de l'autre côté du musoir peint en blanc.



(B) Schéma de l'installation de la digue de l'Artha.



(C) Installation des seize capteurs sur le musoir nord-ouest. Le capteur zéro est en bas.

FIGURE 4.11 – Installation technique de la digue de l'Artha.

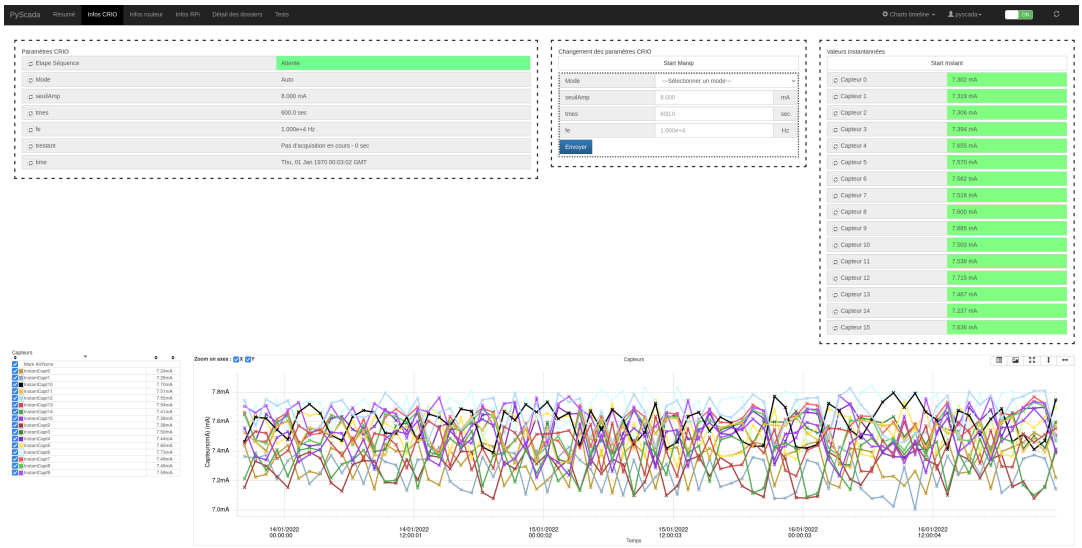


FIGURE 4.12 – Interface utilisateur de PyScada sur la station de la digue de l'Artha. Voir agrandissement pleine page en annexe A.

L'intégration de la librairie graphique Plotly [6] utilisant Datashader [7] permet de résoudre ce problème et de visualiser une courbe contenant des millions de points sans perte d'information. Datashader est un système de *pipeline* (canalisation) graphique permettant de créer des représentations pertinentes de grands ensembles de données de manière rapide et flexible. Datashader décompose la création d'images en une

6. https://plotly.com/
7. https://datashader.org

série d'étapes explicites qui permettent d'effectuer des calculs sur des représentations intermédiaires.

Le résultat d'un enregistrement de dix minutes pour trois capteurs totalisant dix huit millions de points est présenté à la figure 4.13.

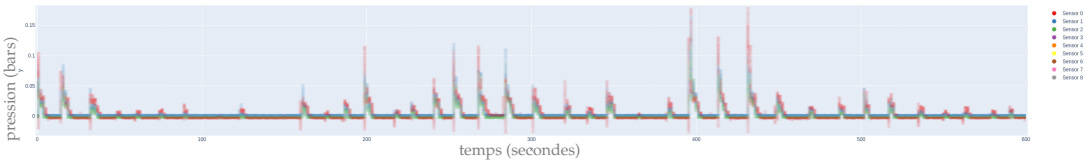


FIGURE 4.13 – Image produite par Plotly et Datashader dans l'IHM web permettant de visualiser 18 millions de points. Il est possible de zoomer sur l'image, Plotly générant une nouvelle image. Voir agrandissement pleine page en annexe A.

4.2.2 Seconde station : fort de Socoa

La station vidéo du fort de Socoa a été installée en 2021 pour filmer différents éléments du littoral. Elle sert à filmer des évènements particuliers ou à suivre l'érosion du littoral. Les stations de Socoa et de l'Artha sont synchronisées avec un écart de temps inférieur à 10 millisecondes. Une des caméras filme la digue de l'Artha (voir figure 4.14) et permet d'analyser la séquence vidéo correspondant à un enregistrement des capteurs de pression afin de mieux identifier les types d'impacts de vagues (voir D'AMICO [3]).



FIGURE 4.14 – Vue de la digue de l'Artha depuis la caméra installée sur le fort de Socoa.

Cette opération est rendue possible par l'**interconnexion des deux plateformes** utilisant PyScada. La station de la digue de l'Artha envoie un ordre automatique, via un *web service*, à la station du fort de Socoa pour démarrer un enregistrement vidéo dès qu'un enregistrement par la centrale de mesure commence. Lorsque l'enregistrement des capteurs de pression se termine, la station de l'Artha demande d'arrêter l'enregistrement vidéo. La vidéo est supprimée si la centrale de mesure n'a

pas conservé son enregistrement (voir le quatrième paragraphe de la section 4.2.1). L'ensemble des vidéos est envoyé sur les serveurs du laboratoire pour être analysé.

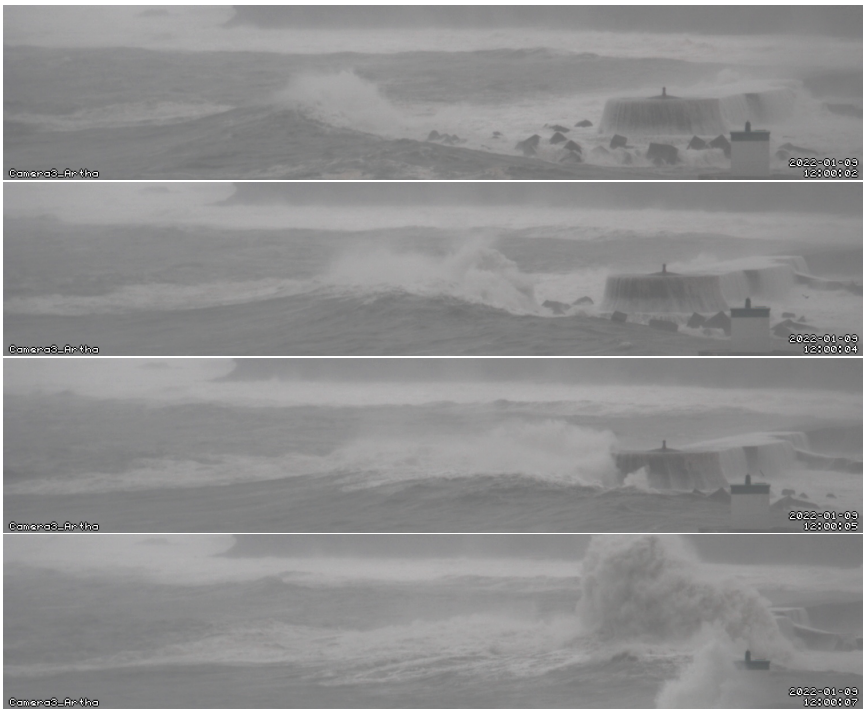


FIGURE 4.15 – Impact d'une vague filmée pendant une tempête depuis la station du fort de Socoa.

La figure 4.16 est le résultat pour deux capteurs de la partie de la figure 4.13 correspondant à l'impact de la figure 4.15.

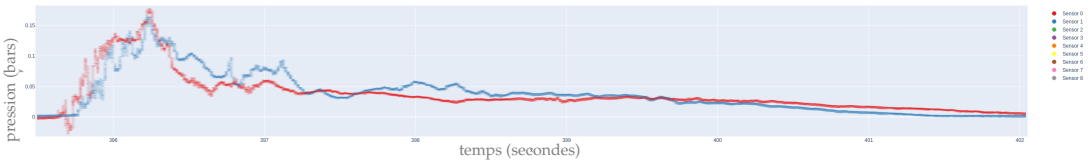


FIGURE 4.16 – Représentation temporelle des données de deux capteurs de l'impact de la figure 4.15. Voir affichage pleine page en annexe A.

De futurs travaux de recherche viseront à identifier les blocs par la vidéo et détecter leurs déplacements à l'aide d'algorithmes de traitement vidéo.

L'instance de PyScada installée au fort de Socoa est aussi utilisée avec deux autres caméras (voir figure 4.17) pour un autre projet utilisant un logiciel (appelé Sirena) développé par une entreprise espagnole pour prendre des photos du littoral. Cette installation a pour but d'étudier :

— la réponse des ouvrages de protection à un événement de tempête,
— l'influence des ouvrages de protection sur l'hydrodynamique,
— le développement des modèles bathymétriques et topographiques,
— le test et la validation des modèles numériques.

PyScada permet de vérifier l'état du logiciel, de le relancer, d'accéder aux caméras en temps réel par l'interface web et d'avertir les utilisateurs de cette installation en cas de défaillance du système. Le logiciel MotionEye facilitant la configuration et



FIGURE 4.17 – Vues de caméras situées au fort de Socoa et utilisées par le logiciel Sirena.

la supervision de divers types de caméras a été utilisé et intégré dans l'interface de PyScada.

4.2.3 Troisième station : structure mobile

La figure 4.18 présente la structure mobile pour l'étude de vagues de submersion. Elle a été placée sur la plage de Biarritz lors d'évènements à fort coefficient (niveau élevé de la marée haute). Elle est immobilisée à l'aide de sac de sables. Le système de supervision est identique à celui de la digue de l'Artha hormis qu'il ne possède pas de panneau solaire, la batterie étant rechargée manuellement à chaque déploiement de la structure.



FIGURE 4.18 – Structure mobile pour l'étude des phénomènes de submersion côtiers. A gauche : le câblage du système de supervision. Au centre : la face avant de la structure avec les 9 capteurs de pression installés. A droite : la face arrière de la structure avec les capteurs à connecter.

4.2.4 Vers un maillage de la côte basque

Les diverses stations installées sur le littoral basque vont permettre de réaliser un maillage de la prise de mesures et de vidéo pour l'étude des évènements marins. Cette opération demande :

— une synchronisation des horloges des différents systèmes isolés,
— un rapatriement des données vers un serveur centralisé pour l'analyse des données, éventuellement après un pré-traitement sur site à la périphérie du réseau (voir *Edge computing* en section 1.2.6),
— une gestion décentralisée afin et faire communiquer les stations entre elles et d'accéder à toute station quel que soit le point d'entrée (la station à laquelle on se connecte ou via le réseau de l'université).

Ce maillage permet d'analyser les différentes réponses des structures et ouvrages du littoral à une même tempête pour comprendre le rôle des différents paramètres (bathymétriques et topographiques).

4.2.5 Études en laboratoire

Le projet visant à recréer en laboratoire les vagues étudiées en milieu marin a commencé en 2021. La figure 4.19 présente le banc permettant de lâcher une masse d'eau via une porte coulissant vers le haut (à gauche) sur une paroi équipée de capteur représentant un ouvrage (à droite). Entre ces deux éléments peuvent être placés des objets recréant des situations réelles (bloc de béton, digue).

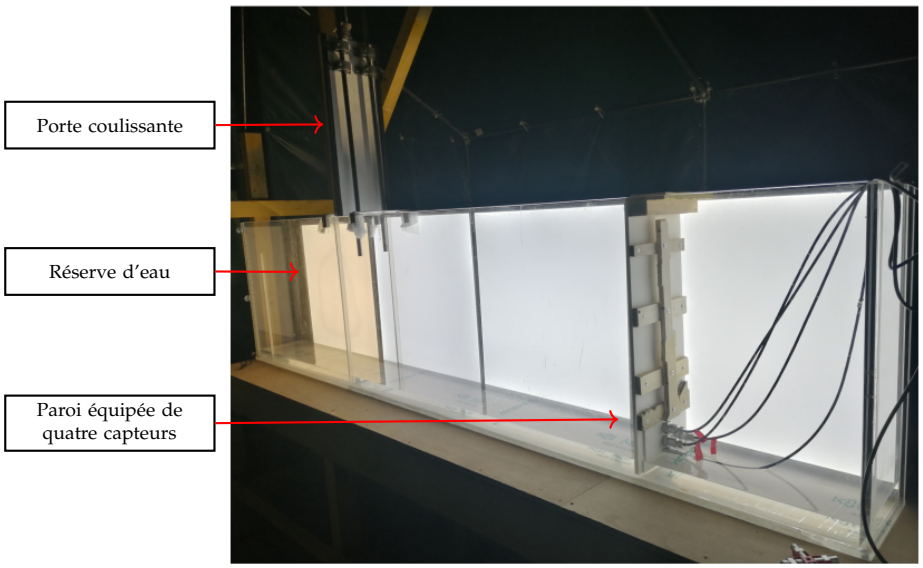


FIGURE 4.19 – Étude en laboratoire des interactions entre vagues et structures.

PyScada permet de relever l'ensemble des données des capteurs (voir figure 4.20).

D'autres ingénieurs, chercheurs ou étudiants en stage commencent à utiliser PyScada pour l'instrumentation dans différents domaines comme la résistance des matériaux (utilisation de presses communiquant via le protocole OPC-UA) ou la rigidité diélectrique d'un liquide (utilisation d'un oscilloscope via le protocole VISA pour des mesures de tensions).

L'ensemble des instances de PyScada utilisées par le laboratoire SIAME utilise divers protocoles de communication (Web service, série, Modbus, GPIO), des alertes,

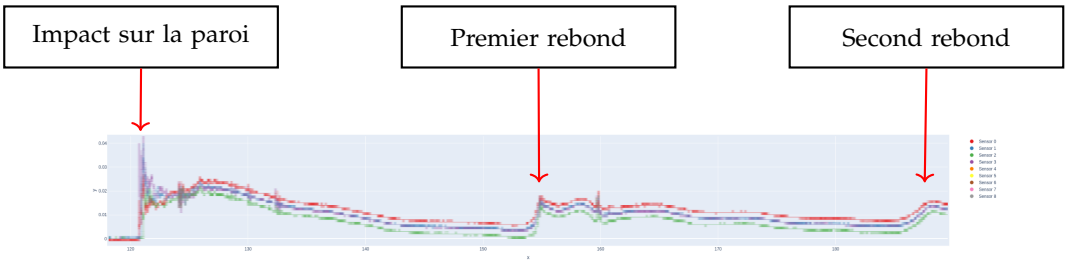


FIGURE 4.20 – Évolution temporelle de la mesure de l'impact d'un lâché d'eau par les quatre capteurs du banc de laboratoire (voir figure 4.19). Les rebonds sont dus aux aller-retours de l'eau entre la paroi de gauche (réserve d'eau) et la paroi des capteurs. Voir agrandissement pleine page en annexe A.

des évènements et des scripts. Les problématiques abordées sont celles de la décentralisation, de la gestion de grandes quantités de données, de l'automatisation de processus et de l'interopérabilité matérielle, logicielle et humaine. Les personnes utilisant et configurant les installations sont des étudiants de master en stage, des doctorants, des chercheurs et des enseignants. Le SGC inclus dans PyScada permet à chaque utilisateur de développer l'interface dont il a besoin pour ses travaux de recherche. Une interface est en développement pour donner un accès public aux informations, images et vidéos recueillies via le maillage de la côte basque.

4.3 Suivi énergétique du bâtiment

Le projet de mesure et de contrôle de l'énergie dans les bâtiments répond aux besoins d'agir pour la transition écologique et numérique dans la construction sur le territoire. Il permet l'optimisation de l'exploitation, le suivi en temps réel de la performance énergétique des bâtiments, les économies d'énergie et la conformité aux nouvelles réglementations en vigueur (COP21 [8], RE2020 [9], décret tertiaire [10]).

Une entreprise nommée InviseO est en cours de création pour proposer une supervision ergonomique personnalisée, connectée et peu intrusive en termes d'instrumentation matérielle, utilisant PyScada comme logiciel.

Cette solution peut apporter une réponse aux problématiques de complexité d'exploitation, par exemple les systèmes de GTC/GTB (Gestion Technique Centralisée/Gestion Technique du Bâtiment) sont destinés à un public expert, nécessitant une connaissance technique approfondie du bâtiment. Elle permet une meilleure connaissance et compréhension du bâti, de sa systémique et de son fonctionnement. Après échange avec les différents acteurs autour de l'exploitation des bâtiments, le constat est que la connaissance se perd notamment à cause du taux de rotation des intervenants (maintenance, usagers...) et à cause de la **complexité de l'interface**.

L'information centralisée autour d'un support unique via l'intégration de données dans la maquette numérique (jumeau numérique et BIM de l'anglais *Building Information Modeling*, permettant la modélisation des données du bâtiment) est rendue accessible en temps réel via la plate-forme web de PyScada. Elle est intuitive et doit pouvoir être utilisée et compréhensible par une personne qui n'est pas experte technique. Elle met ainsi en avant les indicateurs clefs de performance énergétique

8. https://www.gouvernement.fr/action/la-conference-de-paris-sur-le-climat
9. https://www.ecologie.gouv.fr/reglementation-environnementale-re2020
10. https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000038812251

(tableau de bord, résultats d'une Simulation Thermique Dynamique (STD), prévision/anticipation de la consommation via des techniques d'intelligence artificielle (IA)) et devient le support central du suivi et de l'optimisation énergétique des bâtiments.

Sur le caractère durable et environnemental l'outil permet de faire des économies d'énergie grâce à une meilleure connaissance et gestion du patrimoine bâti en informant l'utilisateur en temps réel sur son impact énergétique lors de l'usage du bâtiment et en procurant des ajustements automatiques sur les systèmes de Climatisation-Ventilation-Chauffage (CVC).

4.3.1 Première expérimentation : IUT de Bayonne

PyScada est utilisé pour proposer aux étudiants et enseignants du site de Montaury à Anglet de construire une interface de supervision de la consommation énergétique du bâtiment instrumenté de l'IUT de Bayonne.

Ce projet permet aux étudiants d'apprendre à configurer et se connecter à des instruments distants via le protocole Modbus ou des web services et de construire une interface à l'aide d'un SGC.

Les capteurs fournissent des relevés de consommation électrique, via des mesures de puissance, de tension et de courant instantanées ou d'énergie, séparées en différents usages (éclairages, par étage, CVC, serveurs, et départ général). Des capteurs de consommation d'eau et de gaz sont aussi disponibles. Une station météorologique est installée sur le toit du bâtiment et donne accès à la température, la pression, l'humidité relative, le sens du vent et la radiation.

Ce projet démontre la capacité de PyScada d'offrir une interface de configuration accessible à divers profils utilisateurs (voir interopérabilité humaine pour la configuration en section 1.3.3.4.1).

L'instrumentation du bâtiment intéresse le patrimoine de l'université UPPA comme site pilote pour identifier et comprendre les consommations énergétiques en vue de les réduire.

4.3.2 Seconde expérimentation : Domolandes

Domolandes est un technopôle tourné vers la construction durable et numérique pour les acteurs publics et privés qui s'investissent dans l'écoconstruction, l'habitat et le cadre de vie. L'intérêt d'utiliser PyScada pour le suivi énergétique des bâtiments est venu d'une collaboration entre les sociétés Hubics et Gallium hébergées sur le technopôle et le laboratoire SIAME.

Cette collaboration a permis de faire travailler en 2022 trois stagiaires (deux du DUT Informatique et un de la licence professionnelle Écologie Industrielle de l'IUT de Bayonne) sur l'intégration de nouveaux éléments graphiques (librairie et maquette numérique, voir figure 4.21) et sur une solution non intrusive de comptage de consommation électrique.

Les deux étudiants de deuxième année du DUT Informatique ont développé chacun un *plugin* de PyScada. Ceci montre que l'intégration de nouveaux éléments par le développement et l'ajout de *plugin* est accessible à des informaticiens débutants (voir interopérabilité humaine pour le développement en section 1.3.3.4.4).

La société InviseO en cours de création et hébergée à Domolandes est le fruit de ce travail collaboratif.

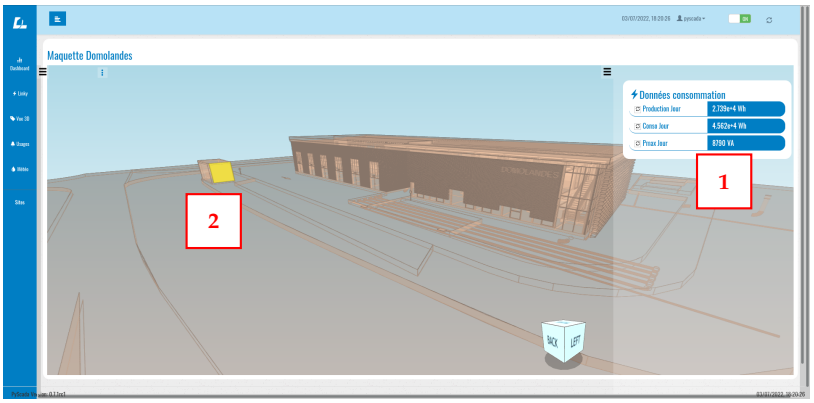


FIGURE 4.21 – Interface de PyScada montrant l'intégration d'éléments graphiques (panneau de contrôle, graphique, bouton, camembert...) dans une maquette numérique 3D. Chaque élément de la maquette peut être lié à un ou plusieurs éléments graphiques.

4.3.3 Troisième expérimentation : Elis

Un étudiant de DUT Génie Industriel et Maintenance (GIM) de l'IUT de Bayonne en alternance dans la société Elis (location et entretien de textiles) ayant utilisé PyScada lors de sa formation a pu mettre en place cette solution pour remplacer les relevés manuels, faits par un opérateur humain, des compteurs électriques communiquant en Modbus. L'utilisation principale est l'extraction des données en CSV afin de les intégrer dans le tableur du groupe Elis pour le suivi énergétique. Cette installation libre par un utilisateur est possible grâce à l'utilisation simple et intuitive de PyScada sous licence GPL.

Les protocoles de communications utilisés par ces différents projets autour du suivi énergétique du bâtiment sont BACnet, Modbus, I2C et Web service.

L'ensemble des personnes travaillant autour de PyScada dans ces projets de mesure et de contrôle de l'énergie dans les bâtiments montre que toutes les interopérabilités humaines présentées en section 1.3.3 sont mises en application, que ce soit pour la configuration, l'utilisation, la visualisation ou le développement.

4.4 Bilan, validation et évaluation

Les différents cas d'applications sont à l'origine de l'évolution de PyScada et démontrent l'intérêt à utiliser le logiciel dans divers domaines. Chaque cas apporte ses besoins, faisant évoluer la solution de supervision et d'instrumentation. A chaque nouveau besoin, notre approche cherche à abstraire le problème pour développer une solution agnostique et réutilisable pour des besoins similaires.

Que le secteur soit l'enseignement, le laboratoire de recherche ou l'industrie et qu'il s'intéresse aux domaines de l'énergie, de l'éducation ou de l'environnement, nous retrouvons les problématiques et besoins communs identifiés aux chapitres 1 et 2.

Un logiciel de supervision et d'instrumentation qui se veut interopérable doit décomposer l'ensemble de ses modules en une couche abstraite et une couche spécifique comme présenté en section 3.2.3. Ceci a pour but de faciliter l'intégration de nouvelles fonctionnalité en se focalisant sur le développement de la couche spécifique.

L'architecture libre et interopérable de PyScada nous permet de rapidement répondre aux besoins d'instrumentation et de supervision à distance de nombreux collègues, partenaires ou nouveaux utilisateurs.

Le nombre de cas d'utilisation est actuellement d'une dizaine d'installations et tend à s'accroître avec les nouvelles sollicitations que nous recevons (duplication de plateformes de travaux pratiques à distance, maillage de la côte basque pour le suivi côtier et création d'entreprise autour du suivi énergétique de bâtiments). Ces nouveaux cas d'étude permettent de valider et d'enrichir l'approche que nous avons choisie et qui se définit par les critères d'interopérabilité, de traitements automatisés, de décentralisation, de facilité de réutilisation et d'accessibilité, et de liberté d'utilisation, de modification et de distribution.

L'évaluation du logiciel est réalisée manuellement de manière régulière auprès des utilisateurs que nous côtoyons, et qui sont près d'une centaine actuellement, mais aussi via les retours que nous font les utilisateurs téléchargeant et installant le logiciel à travers le monde via le dépôt en ligne. Cette évaluation mériterait d'être automatisée afin de mieux la formaliser et de recueillir plus de commentaires.

Conclusion et perspectives

Aujourd'hui les outils et méthodes destinés à l'instrumentation et la supervision informatique de systèmes physiques permettent d'étudier des procédés automatisés, de connaître l'état d'un système physique et d'agir sur celui-ci. La présence de l'informatique offre des outils d'acquisition de données (télémesure, alarme, retour d'état de fonctionnement) et de paramétrage des processus (télécontrôle, configuration) permettant de simplifier le traitement réalisé par l'homme et d'accéder à des expérimentations éloignées ou dangereuses. Les réponses logicielles actuelles sont souvent conçues pour un type d'application et dédiés à un domaine particulier. Or en prenant le temps d'abstraire les besoins des utilisateurs, il apparaît une partie générique du problème posé et indépendante du domaine d'application, comme par exemple d'être capable de traiter de grandes quantités de données, d'avoir un faible temps de réponse ou la nécessité d'avoir un système autonome en énergie ou en prise de décision et frugal. Cette thèse présente de manière **empirique** une méthode pour la supervision et l'instrumentation à distance à partir d'applications dans les domaines de l'éducation, l'énergie et l'environnement et les secteurs de l'enseignement, de la recherche ou de l'industrie.

Les lignes directrices du FAIR (*Findable, Accessible, Interoperable, Reusable*) des données, des logiciels et des algorithmes ont été appliquées dans cette thèse afin de répondre à la problématique de solutions logicielles génériques et agnostiques du domaine d'étude pour la supervision, l'instrumentation et l'expérimentation. La solution transdisciplinaire proposée par PyScada offre une réponse simple et accessible, et a comme objectif de confronter sans cesse la problématique posée à de nouveaux champs d'applications.

Communiquer avec des instruments, lire, enregistrer, traiter et visualiser un grand nombre de données provenant de capteurs, envoyer des consignes à des actionneurs, définir des évènements et automatiser des tâches, construire des interfaces correspondant aux besoins de différents utilisateurs avec une gestion de droits d'accès sur de petites infrastructures, tout en donnant la liberté à chacun de modifier et de distribuer la solution, tels étaient les objectifs de cette thèse.

Pour réussir à proposer un logiciel répondant à ces objectifs il a été nécessaire de définir les composants d'un processus de supervision (télémesure, télécontrôle, stockage de données, automatisation de tâches, accès distant et sécurité). Ensuite la problématique de l'interopérabilité a été séparée en différents niveaux (juridique, organisationnel, technique, syntaxique et sémantique) permettant d'apporter une réponse globale. L'interopérabilité permet notamment de traiter automatiquement les données par les machines pour **dépasser l'internet** fait par et pour les humains et aller vers le **web sémantique**. Mais elle doit aussi autoriser de faire fonctionner le logiciel sur un maximum de serveurs, d'ajouter de nouveaux protocoles de communication et librairies, et de s'adapter à tous les profils d'utilisateurs, qu'ils développent, utilisent, visualisent ou configurent le système, s'adaptant à la sémantique de chacun.

Cette solution n'est possible que si l'utilisateur dispose de la liberté de faire fonctionner, d'utiliser, d'étudier, de modifier et de distribuer les logiciels, algorithmes et données nécessaires pour travailler. Les notions politiques et philosophiques de

liberté logicielle et de *copyleft* sont trop souvent oubliées au profit des seuls aspects d'accessibilité et d'usage de l'*open source*. Afin de revenir à une science libre et ouverte, nous recommandons l'utilisation des licences GPL ou AGPL.

PyScada permet différentes abstractions (connexion, expérimentation, transport, charge utile, synchronisation, erreur) en utilisant l'architecture logicielle offerte par le *framework* Django. Ceci facilite le développement et l'intégration de fonctionnalités, la communication à la base de données et l'uniformisation des informations présentées à l'utilisateur final.

Une amélioration à venir sera de pouvoir décentraliser tous les éléments (instruments, variables, objets graphiques) parmi des instances fédérées, c'est-à-dire inter-connectées. Par exemple les pages, les *widgets* et les graphiques pourront être intégrés dans l'interface utilisateur d'une autre instance de PyScada. La décentralisation permet aussi de lutter contre le monopole, contre les restrictions d'accès, de répartir la charge (bande passante et calculs) et de diminuer les risques de perte de connexion et de données.

Concernant le processus de supervision, défini par des variables d'entrée, de sortie et des algorithmes, il est nécessaire de pouvoir lui attribuer, ainsi qu'à tout objet qu'il contient, un PID et des méta-données. La poursuite de ces travaux nécessitera de **proposer une autorité de gestion des PID** (en utilisant par exemple l'ARK) pour les logiciels de supervision. Ceci permet de répondre à trois principes du FAIR, à savoir trouver, accéder et réutiliser les données. Il sera aussi nécessaire d'améliorer en priorité les deux points négatifs à savoir : inclure la **version** du logiciel dans les méta-données et définir un **vocabulaire** contrôlé pour décrire le logiciel et les méta-données.

A travers les différentes applications présentées (Laborem, surveillance de structures côtières et suivi énergétique du bâtiment), nous avons pu donner du sens et de la valeur à des données en utilisant des algorithmes et en réagissant à leurs variations (événement et alerte). Nous avons pu abstraire la distance du système physique étudié pour l'utilisateur final mais aussi adapter simplement une même interface à différents groupes d'utilisateurs en cachant certains éléments.

Ceci nous a permis notamment de démontrer l'intérêt du travail en binôme lors de la réalisation de travaux pratiques à distance grâce au développement de la gestion d'une file d'attente et de groupes d'utilisateurs pour le projet pédagogique Laborem.

Pour l'étude du littoral, l'interopérabilité offerte par les divers protocoles de communication disponibles et par l'IHM permet de visualiser des données importantes de pression d'impact de vagues provenant de stations isolées, autonomes et interconnectées, de pouvoir utiliser la vidéo pour améliorer la classification des vagues et d'être en capacité de proposer un maillage d'instrumentations de la côte basque pour l'étude du littoral.

Enfin l'architecture modulaire de PyScada nous permet de travailler en collaboration avec des entreprises du bâtiment et des étudiants en stage afin de proposer une solution pour l'optimisation de l'exploitation d'un bâtiment. Cet outil rend possible le suivi en temps réel de la performance énergétique, les économies d'énergie et la conformité aux nouvelles réglementations en vigueur pour des systèmes complexes via par exemple la jonction de la supervision et de la maquette numérique en 3D.

Finalement cette thèse a aussi demandé le développement de matériel *open source* pour la réalisation de travaux pratiques à distance : la Laborem Box.

Publications et conférences

Publications en revues internationales

— **C. Lavayssière**, B. Larroque, F. Luthon, Laborem Box : A scalable and open source platform to design remote lab experiments in electronics, HardwareX, Volume 11, 2022, e00301, ISSN 2468-0672, https://doi.org/10.1016/j.ohx.2022.e00301.
— Letowski, B. ; **Lavayssière, C.** ; Larroque, B. ; Schröder, M. ; Luthon, F. A Fully Open Source Remote Laboratory for Practical Learning. Electronics 2020, 9, 1832, https://doi.org/10.3390/electronics9111832.
— **C. Lavayssière**, B. Letowski, B. Larroque, F. Luthon, LaboREM - A network of open source remote laboratories for learning, Int. Journal of Advances in Electronics and Computer Science (IJAECS), Vol. 5, No. 10, pp. 41-45, Oct. 2018, IJAECS-IRAJ-DOI-13876.

Conférences internationales

— (En cours de publication) E. Imbertie, D. Morichon, B. Larroque, M. Delpey, B. Hernanz, **C. Lavayssière** (2022) Autonomous measuring system of overtopping wave impact pressure in real field conditions, JNGCGC 2022 Proceedings.
— **C. Lavayssière**, B. Larroque, F. Luthon (2021) Analysis of students' behavior regarding the use of open source remote laboratories, EDULEARN21 Proceedings, pp. 6791-6799.
— B. Letowski, **C. Lavayssière**, B. Larroque, F. Luthon (2019) An open source remote laboratory network based on a ready to use solution : Laborem, ICERI2019 Proceedings, pp. 5726-5731.
— **C. Lavayssière**, B. Letowski, B. Larroque, F. Luthon (Nov. 2018) Easy applied sciences learning through open source remote laboratory, 11th annual Int. Conf. Education, Research and Innovation (ICERI2018), Seville, Spain, pp. 2576-2583, DOI : 10.21125/iceri.2018.1573.

Annexe A

Figures pleines pages

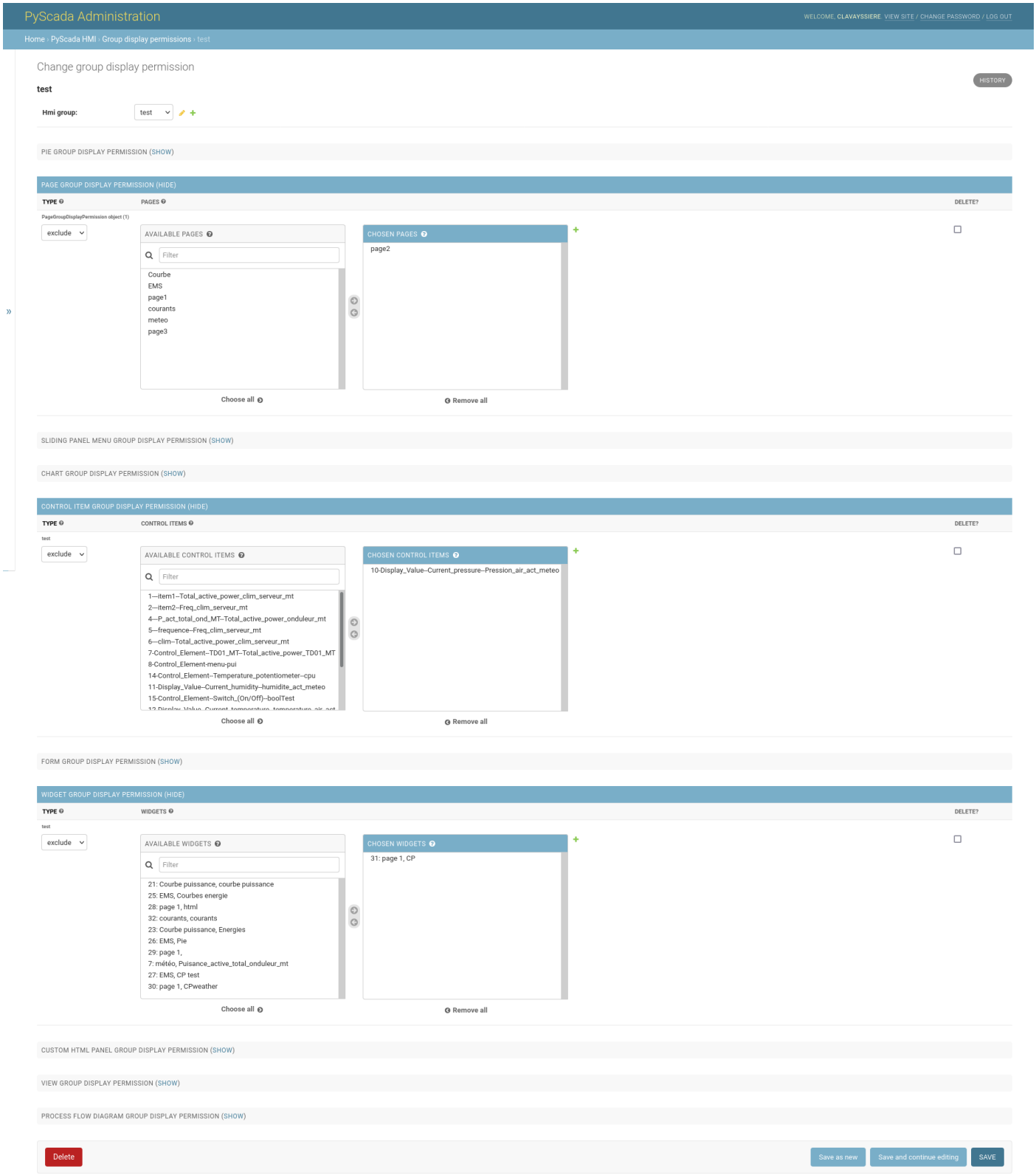


FIGURE A.1 – Gestion des permissions d'affichage de certains éléments graphiques pour un groupe d'utilisateurs.

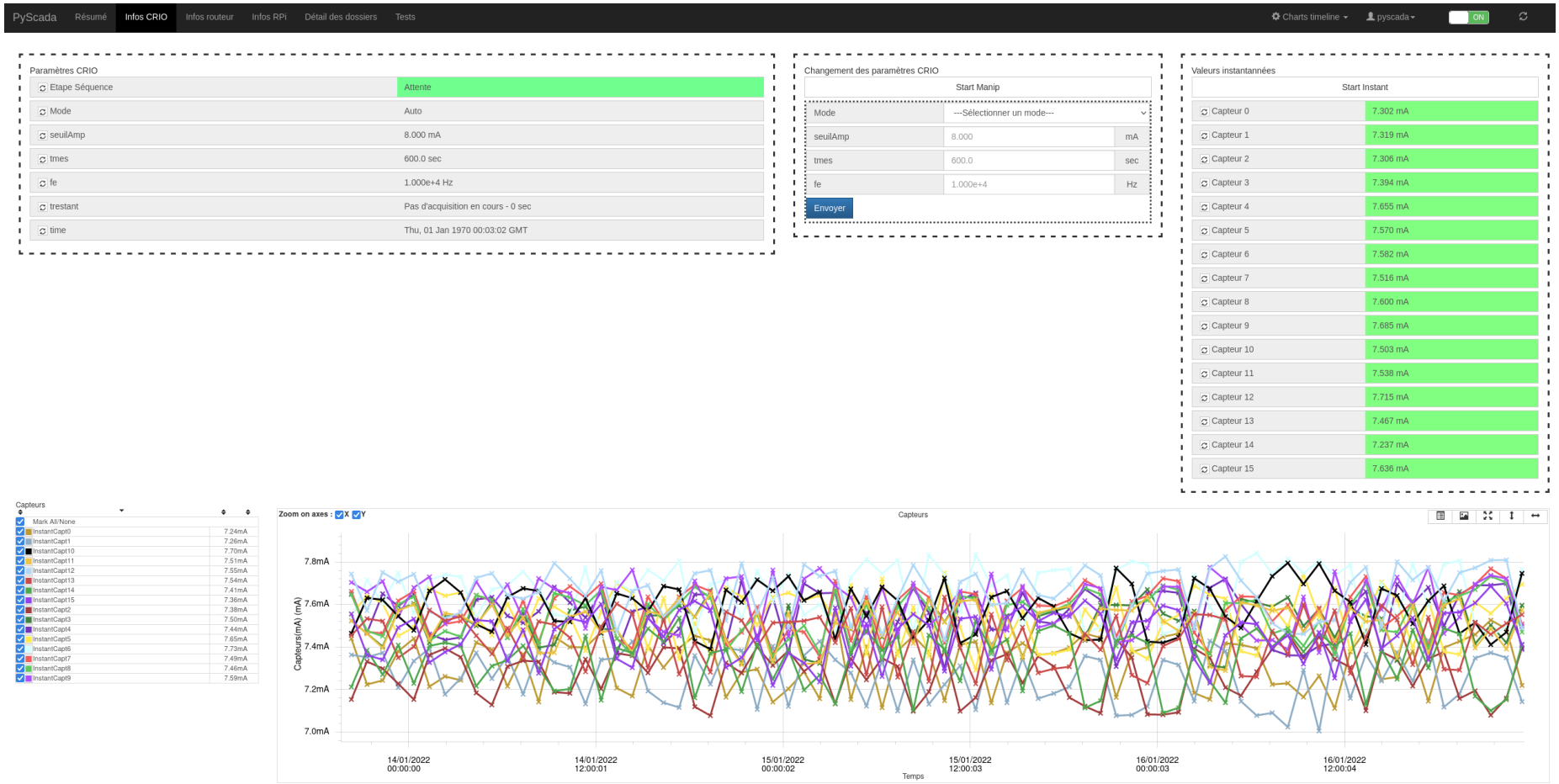


FIGURE A.2 – Interface utilisateur de PyScada sur la station de la digue de l'Artha.

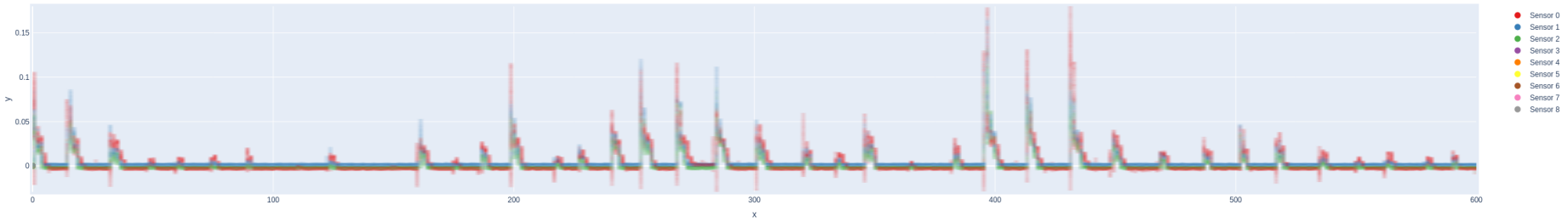


FIGURE A.3 – Image produite par Plotly et Datashader permettant de visualiser 18 millions de points sans perte d'information. Il est possible de zoomer sur l'image, Plotly générant une nouvelle image.

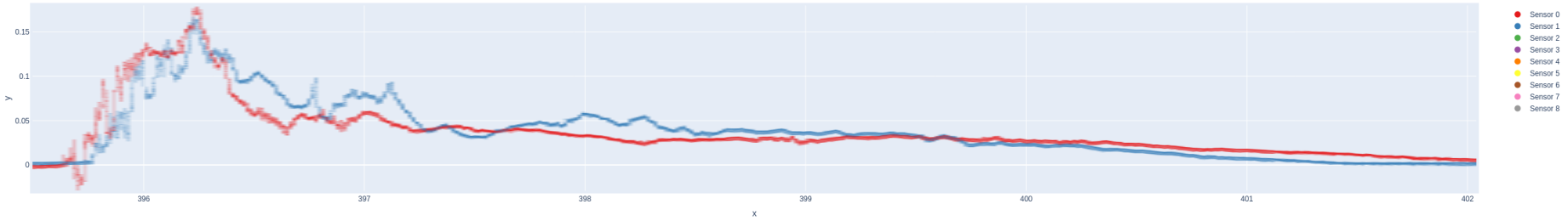


FIGURE A.4 – Représentation temporelle des données de deux capteurs de l'impact de la figure 4.15.

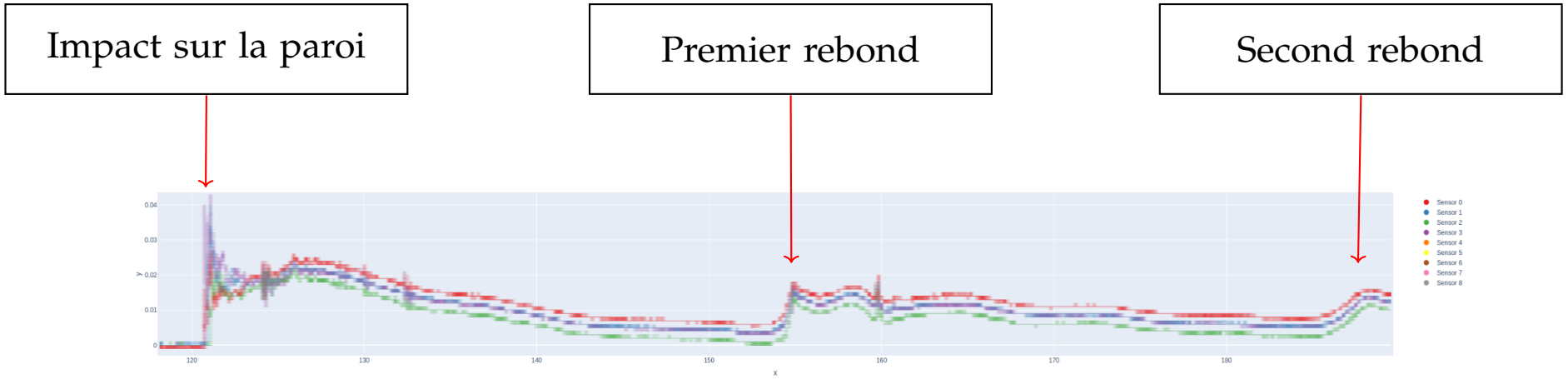


FIGURE A.5 – Évolution temporelle de la mesure de l'impact d'un lâché d'eau par les quatre capteurs du banc de laboratoire (voir figure 4.19). Les rebonds sont dus aux allers-retours de l'eau entre la paroi de gauche (réserve d'eau) et la paroi des capteurs.

Bibliographie

[1] LÉGIFRANCE. *Article L122-6-1 - Code de La Propriété Intellectuelle*. URL : https://www.legifrance.gouv.fr/codes/article_lc/LEGIARTI000044365559/2022-05-30.

[2] Franck LUTHON et Benoit LARROQUE. « LaboREM—A Remote Laboratory for Game-Like Training in Electronics ». In : *IEEE Transactions on Learning Technologies* 8.3 (1er juill. 2015), p. 311-321. ISSN : 1939-1382. DOI : 10.1109/TLT.2014.2386337.

[3] Dorian D'AMICO. « Système de traitement de données massives appliqué à la supervision de milieux côtiers ». Mémoire de thèse. 15 avr. 2021.

[4] Fabrice DEMARTHON, Denis DELBECQ et Grégory FLÉCHET. *CNRS International Magazine N°28*. URL : https://www.cnrs.fr/fr/pdf/cim/CIM28.pdf.

[5] Andrea DE MAURO, Marco GRECO et Michele GRIMALDI. « A Formal Definition of Big Data Based on Its Essential Features ». In : *Library Review* 65.3 (1er jan. 2016), p. 122-135. ISSN : 0024-2535. DOI : 10.1108/LR-06-2015-0061.

[6] Chang CHAOWEN et al. « A Review of Encryption Storage ». In : *Information Technology Journal* (2010). DOI : 10.3923/itj.2010.1517.1520.

[7] J. HUGHES. « IEEE Standards for Encrypted Storage ». In : *Computer* 37.11 (nov. 2004), p. 110-112. ISSN : 1558-0814. DOI : 10.1109/MC.2004.210.

[8] Geeta YADAV et Kolin PAUL. « Architecture and Security of SCADA Systems : A Review ». In : *International Journal of Critical Infrastructure Protection* 34 (1er sept. 2021), p. 100433. ISSN : 1874-5482. DOI : 10.1016/j.ijcip.2021.100433.

[9] Dimitrios PLIATSIOS et al. « A Survey on SCADA Systems : Secure Protocols, Incidents, Threats and Tactics ». In : *IEEE Communications Surveys Tutorials* 22.3 (2020), p. 1942-1976. ISSN : 1553-877X. DOI : 10.1109/COMST.2020.2987688.

[10] Anam SAJID, Haider ABBAS et Kashif SALEEM. « Cloud-Assisted IoT-Based SCADA Systems Security : A Review of the State of the Art and Future Challenges ». In : *IEEE Access* 4 (2016), p. 1375-1384. ISSN : 2169-3536. DOI : 10.1109/ACCESS.2016.2549047.

[11] Dave EVANS. « How the Next Evolution of the Internet Is Changing Everything ». In : *Cisco Internet Business Solutions Group (IBSG)* (2011), p. 11.

[12] O. J. REICHMAN, Matthew B. JONES et Mark P. SCHILDHAUER. « Challenges and Opportunities of Open Data in Ecology ». In : *Science* 331.6018 (11 fév. 2011), p. 703-705. DOI : 10.1126/science.1197962.

[13] Mark D. WILKINSON et al. « The FAIR Guiding Principles for Scientific Data Management and Stewardship ». In : *Scientific Data* 3.1 (1 15 mars 2016), p. 160018. ISSN : 2052-4463. DOI : 10.1038/sdata.2016.18.

[14] Guillaume PITRON. *L'enfer numérique*. Les liens qui libèrent, 2021. 350 p.

[15] Anna-Lena LAMPRECHT et al. « Towards FAIR Principles for Research Software ». In : *Data Science* 3.1 (1er jan. 2020), p. 37-59. ISSN : 2451-8484. DOI : 10.3233/DS-190026.

[16] Tim BERNERS-LEE, James HENDLER et Ora LASSILA. « The Semantic Web : A New Form of Web Content That Is Meaningful to Computers Will Unleash a Revolution of New Possibilities ». In : *ScientificAmerican.com* (1er mai 2001).

[17] Pierre-Joseph PROUDHON. *Qu'est-ce que la propriété ? - Librairie Generale Francaise - Poche - Librairie des Sciences-Politiques PARIS*. 1840.

[18] LÉGIFRANCE. *Loi N° 2004-575 Du 21 Juin 2004 Pour La Confiance Dans l'économie Numérique.* 21 juin 2004. URL : https://www.legifrance.gouv.fr/loda/id/LEGITEXT000005789847/.

[19] Rafael C. JIMÉNEZ et al. « Four Simple Recommendations to Encourage Best Practices in Research Software ». In : *F1000Research* (13 juin 2017). DOI : 10.12688/f1000research.11407.1.

[20] DINUM. *Référentiel général d'interopérabilité (RGI)*. Direction Interministérielle du Numérique et du Système d Information et de Communication de l'Etat. Déc. 2015. URL : https://www.numerique.gouv.fr/publications/interoperabilite/.

[21] European COMMISSION. *Antitrust : la Commission se félicite de la confirmation par le TPI de sa décision sanctionnant deux abus de position dominante dans l'affaire Microsoft*. URL : https://ec.europa.eu/commission/presscorner/detail/fr/MEMO_07_359.

[22] Edward BALKOVICH et Colin WHITBY-STREVENS. « On the Performance of Decentralized Software ». In : *ACM SIGMETRICS Performance Evaluation Review* 9 (1er août 1980), p. 173-180. ISSN : 0-89791-019-2. DOI : 10.1145/1009375.806161.

[23] Violeta ILIK et Lukas KOSTER. « Information Sharing Pipeline ». In : *The Serials Librarian* 76.1-4 (14 juin 2019), p. 55-65. ISSN : 0361-526X. DOI : 10.1080/0361526X.2019.1583045.

[24] Prabhjot KAUR, Vipin JOSHI et Tushar SABHANI. « Decentralized Internet : An Extension of BitTorrent Network ». In : *GIS-Zeitschrift für Geoinformatik* 9 (12 jan. 2022), p. 335-343.

[25] Arif AHMED et Ejaz AHMED. « A Survey on Mobile Edge Computing ». In : 2016 10th International Conference on Intelligent Systems and Control (ISCO). Jan. 2016, p. 1-8. DOI : 10.1109/ISCO.2016.7727082.

[26] Avi GOPSTEIN et al. « NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 4.0 ». In : *NIST Special Publication* (17 fév. 2021).

[27] Sandra HEILER. « Semantic Interoperability | ACM Computing Surveys ». In : *ACM Computmg Surveys, Vol. 27, No 2* (1995).

[28] Karzan WAKIL et Dayang N.A.JAWAWI. « Intelligent Web Applications as Future Generation of Web Applications ». In : *Scientific Journal of Informatics* 6.2 (30 déc. 2019), p. 213-221. ISSN : 2460-0040. DOI : 10.15294/sji.v6i2.19297.

[29] A. DANEELS et W. SALTER. « What is SCADA ? » In : *CERN Document Server*. International Conference on Accelerator and Large Experimental Physics Control Systems. Trieste, Italy, 1999.

[30] *Technical Information Bulletin*. National Communications System, 1er oct. 2004.

[31] Jesús CHACÓN et al. « EJS, JIL Server, and LabVIEW : An Architecture for Rapid Development of Remote Labs ». In : *IEEE Transactions on Learning Technologies* 8.4 (oct. 2015), p. 393-401. ISSN : 1939-1382. DOI : 10.1109/TLT.2015.2389245.

[32] Sebastian GRÖBER et al. « Experimenting from a Distance—Remotely Controlled Laboratory (RCL) ». In : *European Journal of Physics* 28.3 (avr. 2007), S127-S141. ISSN : 0143-0807. DOI : 10.1088/0143-0807/28/3/S12.

[33] Mohamed TAWFIK et al. « Virtual Instrument Systems in Reality (VISIR) for Remote Wiring and Measurement of Electronic Circuits on Breadboard ». In : *IEEE Transactions on Learning Technologies* 6.1 (jan. 2013), p. 60-72. ISSN : 1939-1382. DOI : 10.1109/TLT.2012.20.

[34] S. BHUTADA et al. « Implementation of a Fully Automated Greenhouse Using SCADA Tool like LabVIEW ». In : Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Juill. 2005, p. 741-746. DOI : 10.1109/AIM.2005.1511071.

[35] Yakub GANGREKAR et Prathamesh CHAUDHARI. « Automated Assembly Line Sorting Using LabVIEW as SCADA ». In : 2019 Global Conference for Advancement in Technology (GCAT). Oct. 2019, p. 1-5. DOI : 10.1109/GCAT47503.2019.8978382.

[36] Jacobo SÁENZ et al. « Open and Low-Cost Virtual and Remote Labs on Control Engineering ». In : *IEEE Access* 3 (8 juin 2015). DOI : 10.1109/ACCESS.2015.2442613.

[37] Javier GARCIA-ZUBIA. *Empowering STEM Education with Technology : Remote Laboratories*. World Scientific, 1er fév. 2021. ISBN : 978-1-78634-942-2. DOI : 10.1142/q0277.

[38] Elena TIKHONOVA et Lilia RAITSKAYA. « An Overview of Trends and Challenges in Higher Education on the Worldwide Research Agenda ». In : *Journal of Language and Education* 4.4 (31 déc. 2018), p. 4-7. ISSN : 2411-7390. DOI : 10.17323/2411-7390-2018-4-4-4-7.

[39] B. AKTAN et al. « Distance Learning Applied to Control Engineering Laboratories ». In : *IEEE Transactions on Education* 39.3 (1er août 1996), p. 320-326. ISSN : 0018-9359. DOI : 10.1109/13.538754.

[40] Aitor VILLAR-MARTÍNEZ et al. « Improving the Scalability and Replicability of Embedded Systems Remote Laboratories Through a Cost-Effective Architecture ». In : *IEEE Access* 7 (2019), p. 164164-164185. ISSN : 2169-3536. DOI : 10.1109/ACCESS.2019.2952321.

[41] Julián BERMÚDEZ-ORTEGA et al. « Remote Web-based Control Laboratory for Mobile Devices Based on EJsS, Raspberry Pi and Node.Js ». In : *IFAC-Papers On Line* 48 (31 déc. 2015), p. 158-163. DOI : 10.1016/j.ifacol.2015.11.230.

[42] Pablo ORDUÑA et al. « Increasing the Value of Remote Laboratory Federations Through an Open Sharing Platform : LabsLand ». In : *Online Engineering & Internet of Things*. 2018, p. 859-873. ISBN : 978-3-319-64352-6. DOI : 10.1007/978-3-319-64352-6_80.

[43] Jesús CHACÓN et al. « Remote Interoperability Protocol : A Bridge between Interactive Interfaces and Engineering Systems ». In : *IFAC-Papers On Line*. IFAC Workshop on Internet Based Control Education IBCE15 48.29 (1er jan. 2015), p. 247-252. ISSN : 2405-8963. DOI : 10.1016/j.ifacol.2015.11.244.

[44] JOHN JOSEPH ROETS. *Emelope - Service Oriented Architecture - Enterprise Application Integration - General Messaging Framework*. Notice logicielle. 28 sept. 2007.

[45] Dijiang HUANG et Huijun WU. « Chapter 2 - Virtualization ». In : *Mobile Cloud Computing*. Sous la dir. de Dijiang HUANG et Huijun WU. Morgan Kaufmann, 1er jan. 2018, p. 31-64. ISBN : 978-0-12-809641-3. DOI : 10.1016/B978-0-12-809641-3.00003-X.

[46] Damian PARNIEWICZ et al. « Design and Implementation of an OpenFlow Hardware Abstraction Layer ». In : *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*. DCC '14. New York, NY, USA : Association for Computing Machinery, 18 août 2014, p. 71-76. ISBN : 978-1-4503-2992-7. DOI : 10.1145/2627566.2627577.

[47] Bastien LETOWSKI et al. « A Fully Open Source Remote Laboratory for Practical Learning ». In : *MDPI Electronics* 9.11 (nov. 2020), p. 1832. DOI : 10.3390/electronics9111832.

[48] Martin KALUZ et al. « A Flexible and Configurable Architecture for Automatic Control Remote Laboratories ». In : *IEEE Transactions on Learning Technologies* 8 (1er juill. 2015), p. 299-310. ISSN : 1939-1382. DOI : 10.1109/TLT.2015.2389251.

[49] Rebecca M. RECK et R. S. SREENIVAS. « Developing an Affordable and Portable Control Systems Laboratory Kit with a Raspberry Pi ». In : *Electronics* 5.3 (sept. 2016), p. 36. DOI : 10.3390/electronics5030036.

[50] José-María GUERRERO-RODRÍGUEZ et al. « Emulation of Circuits under Test Using Low-Cost Embedded Platforms ». In : *Electronics* 10.16 (jan. 2021), p. 1990. DOI : 10.3390/electronics10161990.

[51] Marek SUKOP et al. « Remote Controlling of Automated Systems Based on Client – Server Architecture ». In : *Journal of Automation and Control* 5.2 (11 déc. 2017), p. 69-72. ISSN : 2372-3033. DOI : 10.12691/automation-5-2-8.

[52] Ning WANG et al. « A Novel Wiki-Based Remote Laboratory Platform for Engineering Education ». In : *IEEE Transactions on Learning Technologies* 10.3 (juill. 2017), p. 331-341. ISSN : 1939-1382. DOI : 10.1109/TLT.2016.2593461.

[53] Long Q TRAN, PJ RADCLIFFE et Liuping WANG. « A Low Budget Take-Home Control Engineering Laboratory for Undergraduate ». In : *The International Journal of Electrical Engineering & Education* (1er juin 2019). ISSN : 0020-7209. DOI : 10.1177/0020720919852784.

[54] Franck LUTHON et Benoit LARROQUE. « Remote Laboratory for Game-Based Distance Learning in Electronics ». In : Proceedings of the 4th International Conference on Electronics, Communications and Networks, 15 déc. 2014. ISBN : 978-1-138-02830-2. DOI : 10.1201/b18592-279.

[55] F. LUTHON et B. LARROQUE. « Real Labworks in Electronics : Yes ! ... But Remotely Controlled ». In : *ICERI2015 Proceedings* (2015), p. 8490-8500. ISSN : 2340-1095.

[56] Camille LAVAYSSIÈRE, Benoît LARROQUE et Franck LUTHON. « Laborem Box : A Scalable and Open Source Platform to Design Remote Lab Experiments in Electronics ». In : *HardwareX* 11 (1er avr. 2022), e00301. ISSN : 2468-0672. DOI : 10.1016/j.ohx.2022.e00301.

[57] C. LAVAYSSIÈRE, B. LARROQUE et F. LUTHON. « Analysis of Students' Behavior Regarding the Use of Open Source Remote Laboratories ». In : *EDULEARN21 Proceedings* (2021), p. 6791-6799. ISSN : 2340-1117.

[58] David LOWE. « Integrating Reservations and Queuing in Remote Laboratory Scheduling ». In : *IEEE Transactions on Learning Technologies* 6.1 (jan. 2013), p. 73-84. ISSN : 1939-1382. DOI : 10.1109/TLT.2013.5.

[59] L.A. WILLIAMS et R.R. KESSLER. « The Effects of "Pair-Pressure" and "Pair-Learning" on Software Engineering Education ». In : Thirteenth Conference on Software Engineering Education and Training. Austin, TX, USA : IEEE Comput. Soc, 2000, p. 59-65. ISBN : 978-0-7695-0421-6. DOI : 10.1109/CSEE.2000.827023.

[60] Norsaremah SALLEH, Emilia MENDES et John GRUNDY. « Empirical Studies of Pair Programming for CS/SE Teaching in Higher Education : A Systematic Literature Review ». In : *IEEE Transactions on Software Engineering* 37.4 (juill. 2011), p. 509-525. ISSN : 0098-5589. DOI : 10.1109/TSE.2010.59.

[61] Mladen SOKELE, Trpimir ALAJBEG et Frane BRKIC. « The Impact of Distance Learning on Student Success for Electrical Engineering Professional Courses ». In : 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO). Opatija, Croatia : IEEE, 28 sept. 2020, p. 617-621. ISBN : 978-953-233-099-1. DOI : 10.23919/MIPRO48935.2020.9245315.

[62] Jing MA et Jeffrey NICKERSON. « Hands-on, Simulated, and Remote Laboratories : A Comparative Literature Review ». In : *ACM Comput. Surv.* 38 (30 sept. 2006). DOI : 10.1145/1132960.1132961.

[63] P. A. PONCET et al. « In-Situ Measurements of Energetic Depth-Limited Wave Loading ». In : *Applied Ocean Research* 125 (1er août 2022), p. 103216. ISSN : 0141-1187. DOI : 10.1016/j.apor.2022.103216.

Glossaire

2D	Deux dimensions.
3D	Trois dimensions.
AGPL	*Affero General Public License* est variante de la GPL pour les applications web.
API	Deux significations : en français automate programmable industriel ; en anglais *Application Programming Interface* pour interface de programmation d'applications.
ARK	*Archival Resource Key*.
BIM	*Building Information Modeling* qui permet la modélisation des données du bâtiment.
BNF	Bibliothèque Nationale de France.
BUT	Bachelor Universitaire de Technologie : diplôme français en 3 ans remplaçant le DUT.
Bibliothèque logicielle	Collection d'un ensemble d'instructions qui prend en charge une certaine opération et produit un résultat prêtes à être utilisées par des programmes.
Bit	Unité pouvant prendre la valeur de 0 ou 1.
C.R.U.D.	*Create, Read, Update, Delete* signifie créer ; lire ; mettre à jour et supprimer.
CEMES	Centre d'élaboration de matériaux et d'études structurales.
CHPL	*Cross-Hardware Platform Layer* est la couche de la plateforme multi-matérielle dans une configuration HAL. Elle crée une couche interopérable et abstraite du matériel utilisé.
CMS	*Content Managment System*. Voir SGC.
CSS	*Cascading Style Sheets*.
CSV	*Comma-Separated values*.
CVC	Climatisation Ventilation Chauffage.
DAQ	*Data Acquisition*.
DOI	*Digital Object Identifier*.
DUT	Diplôme Universitaire de Technologie ancêtre du BUT.
DUT GIM	DUT Génie Industriel et Maintenance.

EPICS *Experimental Physics and Industrial Control System* est un ensemble de logiciels et bibliothèques *open source* pour contrôler des instruments scientifiques en temps réel.

Edge computing Informatique en périphérie de réseau. C'est une méthode d'optimisation où une partie du traitement des données est effectuée près des sources de données permettant de réduire la bande passante utilisée.

FAIR *Findable, Accessible, Interoperable, Reusable*.

FPGA *Field-Programmable Gate Array* est un réseau de portes logiques programmables *in situ* sur un circuit intégré.

FSF *Free Software Foundation*.

Fonction asynchrone Fonction lancée en parallèle de l'exécution du programme et ne le bloquant pas.

Fonction synchrone Fonction qui bloque l'exécution du programme tant qu'il n'a pas retourné son résultat.

GNU Système d'exploitation libre.

GPIB *General Purpose Interface Bus* est une norme définissant un bus de communications numériques.

GPIO *General Purpose Input/Output*.

GPL *General Public License* pour licence publique générale.

GTB Gestion Technique du Bâtiment.

GTC Gestion Technique Centralisée.

HAL *Hardware Abstraction Layer* est une organisation d'un logiciel permettant la communication entre le matériel et le logiciel d'une manière abstraite et uniforme.

HDF5 *Hierarchical Data Format* est un modèle de données : une librairie et un format de fichier pour stocker et gérer des données.

HSL *Hardware Specific Layer* est la couche spécifique au matériel dans une configuration de HAL. Elle permet à la CHPL de dialoguer avec du matériel spécifique.

HTML *HyperText Markup Language* désigne le langage conçu pour représenter les pages web.

Handler Fichier gérant les particularités d'un protocole ou d'un instrument.

IA Intelligence Artificielle.

IHM Interface Humain Machine : En anglais HMI pour *Human Machine Interface*.

IP *Internet Protocol* désigne le protocole internet.

IUT Institut Universitaire de Technologie.

IVI *Interchangeable Virtual Instrumentation*.

IVS Interaction Vagues / Structures.

IoT *Internet of Things* désigne l'internet des objets et le nombre croissant d'objets connectés à internet.

JSON *JavaScript Object Notation* est un format de fichier ouvert utilisé sur le web.

LAN *Local Area Network* désigne les réseaux locaux.

LIUPPA Laboratoire d'Informatique de l'Université de Pau et des Pays de l'Adour.

MOOC *Massive Open Online Course*.

NAAN *Name Assigning Authority Number* : distribue des PID pour l'ARK.

NI *National Instruments*.

NMA *Name Mapping Authority* permet de localiser les NAAN pour le PID ARK.

NSA *National Security Agency* désigne le service de renseignement intérieur des États Unis d'Amérique.

OPC *Open Platform Communications*.

ORM *Object-Relational Mapping* est un programme qui se place entre une application et une base de données pour simuler une base de données orientée objet.

OS *Operating System* pour système d'exploitation.

OSHW *Open Source HardWare*.

OSHWA *Open Source HardWare Association*.

PCB *Printed Circuit Board*.

PID *Persistent IDentifier* pour identifiant pérenne. Chaîne de caractères représentant une ressource indépendamment de son emplacement.

RPC *Remote Procedure Call*.

RTU *Remote Terminal Unit* pour terminal à distance.

Réseau 4G Réseau de téléphonie mobile de quatrième génération.

SCADA *Supervisory Control And Data Acquisition* est un système de contrôle et d'acquisition de données.

SCPI *Standard Commands for Programmable Instruments* est une norme permettant de contrôler des instruments de mesures.

SGC Système de Gestion de Contenu est un logiciel permettant de créer simplement des site web.

SIAME	Sciences pour l'Ingénieur Appliquées à la Mécanique et au génie Électrique.
SML	*Smart Message Language* est un protocole de communication pour les compteurs électriques.
SMS	*Short Message Service*.
STD	Simulation Thermique Dynamique.
STEM	*Science, Technology, Engineering, and Mathematics* signifie science ; technologie ; ingénierie et mathématiques.
Script	Fonction exécutant une série de tâches et retournant un résultat.
T2I	Traitements des Informations pour l'adaptation de l'Interaction au contexte et à l'utilisateur.
UPPA	Université de Pau et des Pays de l'Adour.
URI	*Uniform Resource Identifier* est une chaîne de caractère permettant d'identifier une ressource sur internet.
URL	*Uniform Resource Locator* est un sous-groupe des URI et localise une ressource.
USB	*Universal Serial Bus* est une norme de bus informatique.
VHDL	*VHSIC Hardware Description Language* est un langage de description de matériel pour les systèmes électroniques numériques.
VHSIC	*Very High Speed Integrated Circuits*.
VISA	*Virtual Instrument Software Architecture* est une API utilisée dans l'instrumentation et dans l'industrie pour le test et la mesure aujourd'hui gérée par la fondation IVI [1].
VPN	*Virtual Private Network* pour réseau virtuel privé créant un lien direct entre des ordinateurs distants.
W3C	*World Wide Web Consortium* est un organisme de standardisation du web.
WSGI	*Web Server Gateway Interface* définit en langage Python une interface entre le serveur web et l'application web.

1. https://www.ivifoundation.org

Résumé

Doctorat en informatique

Supervision et instrumentation à distance, libre et interopérable, du capteur à l'utilisateur.

par Camille LAVAYSSIÈRE

Cette thèse propose une étude des pratiques actuelles pour l'acquisition de données, la supervision et le contrôle distant de systèmes embarqués connectés (internet des objets IoT). Le but est de mettre en avant une approche pluridisciplinaire, interopérable et généraliste pour l'instrumentation à distance et d'accorder à chaque utilisateur la liberté de modifier, utiliser, distribuer la solution proposée.

Dans un premier chapitre, l'instrumentation et la supervision à distance de systèmes physiques sont présentés en s'articulant autour des concepts d'échange d'information, de télémesure et de télécontrôle, de stockage de données, d'automatisation de tâches et de sécurité. Les problématiques du FAIR (*Findable, Accessible, Interoperable and Reusable*) abordées dans ce domaine demandent de concevoir des algorithmes ou des logiciels et de traiter des données qui seront accessibles à l'aide d'un identifiant unique. Les utilisateurs ou les machines pourront trouver ces objets via des moteurs de recherche et des méta-données afin de les réutiliser. Cette approche novatrice, au regard des logiciels existants, demande aux objets développés d'être interopérables, libres, ouverts et décentralisés afin d'être utilisés par un maximum de personnes, de systèmes informatiques existants ou de domaines d'application.

Dans un second chapitre, des logiciels de supervision dans les secteurs de l'industrie, de la recherche et de l'éducation sont exposés et comparés avec les différents critères retenus. Les besoins identifiés à l'aide des cas d'application observés demandent d'échanger des informations permettant de connaître et de modifier l'état d'un système physique, d'organiser les données récoltées pour mieux les comprendre et les utiliser, et de proposer une solution agnostique du point de vue du domaine scientifique étudié et du type d'utilisateur.

Les travaux de cette thèse proposent de définir différentes abstractions dans l'architecture logicielle d'un système de supervision et d'instrumentation afin de rendre le système interopérable. Il s'agit de séparer la configuration et la communication des instruments de la configuration de l'interface humain-machine, de ne pas avoir à toucher au code de l'application pour la configurer et l'utiliser, de gérer des applications synchrones et asynchrones ainsi que les erreurs de communication entre les systèmes inter-connectés. La solution proposée dans la troisième chapitre repose sur une infrastructure logicielle dont l'architecture permet de faire évoluer, de modifier et d'adapter l'interface aux besoins de chaque utilisateur. L'interface est accessible via une page web et donne accès à divers outils comme la gestion d'événements, d'alertes et l'ajout de scripts.

Enfin les cas d'application étudiés durant cette thèse sont présentés au chapitre 4 et s'articulent autour des systèmes embarqués dans les domaines de l'ingénierie côtière, de la gestion énergétique du bâtiment ainsi que le déploiement de plateforme permettant de réaliser des travaux pratiques à distance dans l'éducation.