J3eA, Journal sur l'enseignement des sciences et technologies de l'information et des systèmes
J3eA 24 (2025) 37
https://doi.org/10.1051/j3ea/20250037
https://www.j3ea.org

Objet Connecté Personnel : du Bluetooth au webserveur

Fabrice Sthal[1*], Joël Imbaud[1]

[1] Université Marie et Louis Pasteur, SUPMICROTECH, CNRS, institut FEMTO-ST, F-25000 Besançon, France

*Auteur de correspondance : fsthal@supmicrotech.fr

Résumé : *Cet article présente une série de TD/TP permettant la mise en œuvre du protocole Bluetooth dans le cadre d'une formation ingénieur consacrée à la conception et réalisation d'objets connectés. Les étudiants utilisent leur Objet Connecté Personnel (ObCP) réalisé en début de formation pour communiquer avec une Raspberry. L'aspect programmation microcontrôleur bas niveau et l'aspect système d'exploitation environné sont abordés pour finalement réaliser un ObCP communicant en Bluetooth avec sa passerelle web serveur.*

Mots clés : STM32, MyRio, Raspberry, programmation C embarquée, Bluetooth, objet connecté.

Abstract: **Personal Connected Object: from Bluetooth to webserver**
This article presents a series of tutorials and practical exercises for implementing the Bluetooth protocol as part of an engineering training course dedicated to designing and producing connected objects. Students use the Personal Connected Object (PCO) they created at the start of the course to communicate with a Raspberry Pi. The tutorials cover low-level microcontroller programming and the surrounding operating system to create an IoT device that communicates with its web server gateway via Bluetooth.

Keywords: STM32, MyRio, Raspberry, embedded C sofware, Bluetooth, IOT.

1. Introduction

SUPMICROTECH-ENSMM est une école d'ingénieurs qui recrute principalement à Bac +3 via les concours des grandes écoles. Elle forme des ingénieurs généralistes avec une spécialisation en mécanique [1]. En 2018/2019, une réforme pédagogique a permis la création d'une option de 3e année dédiée à **la Conception et Réalisation d'Objets Connectés (CROC)**, proposée au premier semestre. Cette option repose sur trois unités d'enseignement (UE) :

- **Une UE scientifique (UES)**, composée de quatre modules thématiques de 60 heures : conception, design et ergonomie (CROC1) ; composants des objets connectés (CROC2) ; contrôle et réseaux (CROC3) ; et traitement des données (CROC4) [2].
- **Une UE en sciences humaines**, de 120 heures.
- **Une UE projet**, de 90 heures.

Un projet fil rouge structure l'apprentissage : **chaque étudiant conçoit et met en situation un Objet Connecté Personnel (ObCP)**.

- Dans **CROC1**, les étudiants abordent l'architecture des objets connectés ainsi que la conception mécanique (sous Catia) et électronique (sous ORCAD).
- Dans **CROC2**, ils étudient les capteurs, les actionneurs et la gestion de l'énergie, avec des travaux pratiques dédiés à la réalisation et au test de l'ObCP.
- Dans **CROC3**, la programmation des microcontrôleurs en C/C++, les réseaux et les systèmes embarqués sont enseignés sous forme de cours, TD et TP.
- Enfin, dans **CROC4**, une initiation à la programmation d'applications mobiles sous Android Studio permet aux étudiants de connecter l'ObCP à une Raspberry Pi, une tablette ou un smartphone.

Cet article détaille les travaux dirigés et pratiques liés à la mise en œuvre du protocole Bluetooth sur l'ObCP et son interaction avec une Raspberry Pi servant de passerelle web. Les étudiants suivent **10 h de TD et 8 h de TP** sur la programmation des microcontrôleurs, ainsi que **20 h de TD et 4 h de TP** sur les réseaux et systèmes embarqués. Une enquête menée auprès des étudiants et enseignants permettra également d'évaluer cette approche pédagogique.

2. Matériels utilisés

Les dispositifs présentés dans cette section illustrent la diversité des matériels employés pour la conception d'objets connectés. Trois principales plateformes sont utilisées :

- **Un microcontrôleur STMicroelectronics**, intégré à une carte de développement et à l'ObCP, programmé sous STM32CubeIDE (basé sur Eclipse).
- **Un MyRio**, exploité à la fois avec LabVIEW et en ligne de commande sous Linux.
- **Une Raspberry Pi**, servant principalement de passerelle web, utilisée sous un environnement Linux avec interface graphique.

Chaque dispositif est détaillé dans les sections suivantes.

2.1. Carte microcontrôleur : L'ObCP

La conception et la réalisation de l'OBCP ont été largement présentées dans l'article [3]. Le cahier des charges de l'ObCP est discuté sous la forme de TD interactifs, guidés par l'enseignant. Le schéma complet de l'ObCP et l'ensemble des documentations des composants sont fournis aux étudiants. La réalisation est effectuée avec 8h de TP spécifique. La conception de la partie microcontrôleur STM32L476 et de la partie Bluetooth low energy (BLE) se base

sur les schémas des cartes de développement ST de la Nucleo F411RE [4] et de la carte shield IDB05A1 [5]. L'architecture de l'ObCP est présentée figure 1.

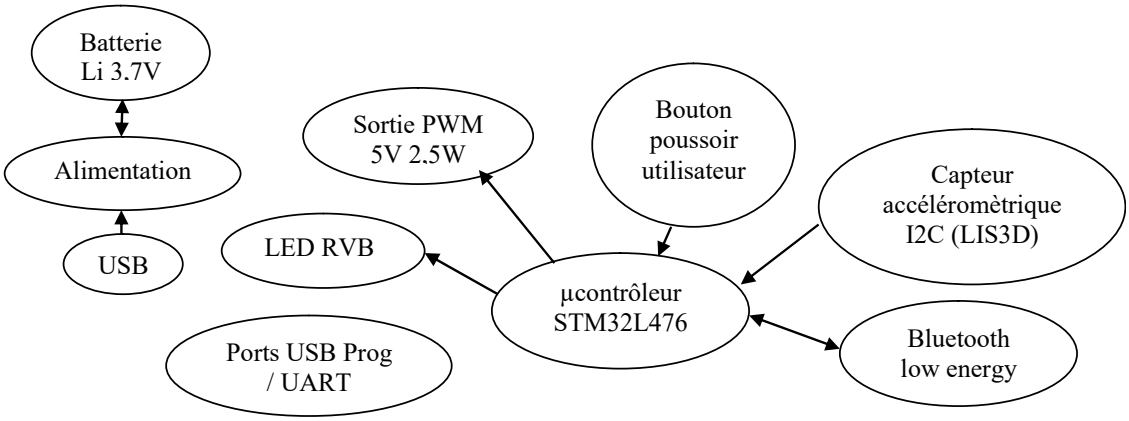

Figure 1. Architecture de l'ObCP

Le microcontrôleur est un STM32L476RG. La puce Bluetooth est un **BlueNRG-M0** [6] remplaçant compatible du module **SPBTLE-RF0** utilisé dans la première version. La réalisation complète de l'ObCP est visible figure 2. Il est à noter que cette carte est conservée par l'étudiant en fin de formation.

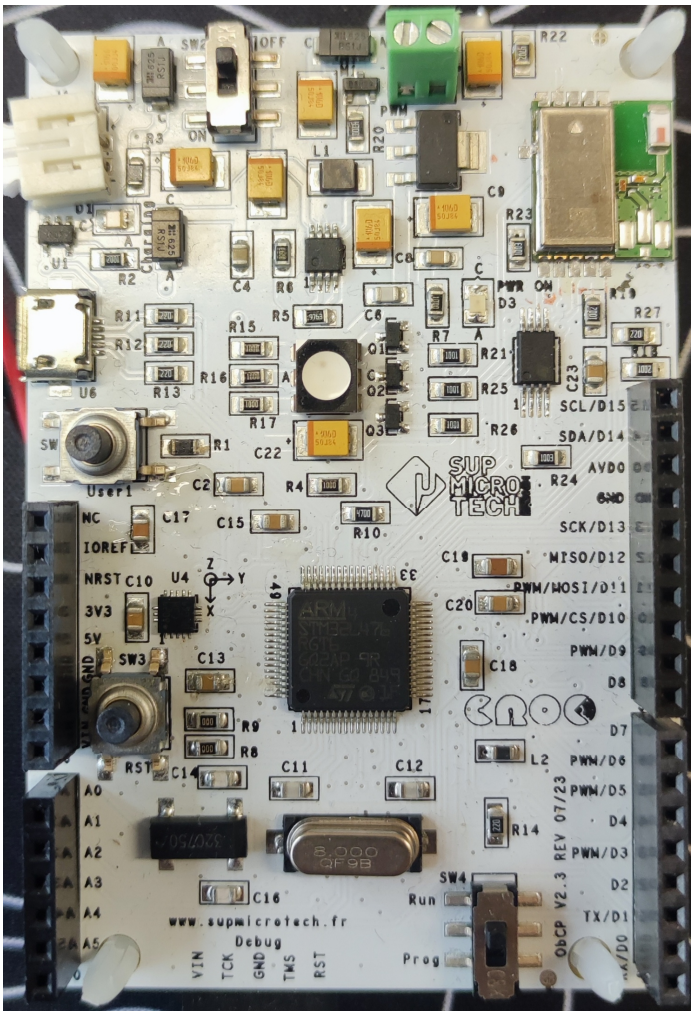

Figure 2. Vues de la maquette réelle de l'ObCP

2.2. MyRio

Les étudiants ont fait une formation labVIEW en deuxième année avec une initiation au circuit logique programmable (FPGA) en utilisant un MyRio [7]. Il est composé d'une puce ZynQ Z-7010 de chez Xilinx. Le MyRIO est essentiellement un outil pour l'enseignement (figure 3).

Figure 3. MyRio

Il est doté d'E/S accessibles sur deux côtés par le biais de connecteurs MXP et MSP. Il inclut 10 entrées analogiques, 6 sorties analogiques, 40 lignes d'E/S numériques, quatre LED, un bouton-poussoir, un accéléromètre embarqué. Le Z-7010 possède un processeur double cœur ARM Cortex_A9 cadencé à 667 MHz. Ce dispositif éducatif possède une puce wifi permettant d'utiliser ce type de réseau. Il est possible de générer un réseau wifi à partir du MyRio ou de se connecter à un réseau existant. Une connectique filaire de type USB est utilisée pour le connecter à un PC hôte. Dans ce cas, il répond à une configuration réseau Ethernet local entre lui-même et le PC. Le MyRIO_1900 est programmable en LabVIEW ou en C. Il sera utilisé dans notre formation en 3ème année pour l'initiation à la structuration d'un serveur détaillée dans la partie programmation.

Il est à noter que ce dispositif est géré par un Linux temps réel préempté (PREMPT-RT) accompagné d'une librairie de commande optimisée de type busybox. Le Hardware étant limité, il n'est pas possible d'installer un environnement fenêtré. Toute la partie Linux doit être gérée en ligne de commande dans une console.

2.3. Raspberry

Les Raspberry utilisées dans les enseignements sont des Raspberry Pi3 B+ [8]. Ce modèle est basé sur un processeur ARM Cortex-A53 64 bits quatre cœurs à 1,4 GHz, possède 1 GB de mémoire RAM (figure 4).

Figure 4. Raspberry Pi3 B+

La carte SD utilisée pour le système est de 32 Go. Du point de vue connectivité sans fil, elle dispose d'une interface Wi-Fi Dual-band 2,4 et 5 GHz, 802.11b/g/n/ac, et d'une interface Bluetooth 4.2 intégrant le BLE. La distribution Linux installée est la bookworm version 12. Elle est installée par les étudiants en début d'enseignement. Elle est utilisée pour compléter l'approche Linux dans un environnement graphique plus approprié et agréable pour les étudiants par rapport au MyRio. La programmation principale se fait en python et PHP pour le serveur web.

3. TD et TP de programmation

Au niveau de la démarche pédagogique, une des difficultés est de coordonner le séquençage des apprentissages. La partie microcontrôleur et ObCP démarre dès le début du semestre. La suite présente les parties programmation techniques abordées.

3.1. Programmation microcontrôleur

Après un cours d'introduction de 2h comprenant une présentation de généralités sur les microcontrôleurs 32 bits de la famille des STM32 puis d'une introduction à la programmation en C embarquée, les étudiants poursuivent leur apprentissage par la pratique sous l'environnement de programmation STM32CubeIDE visible sur la figure 5 [9].

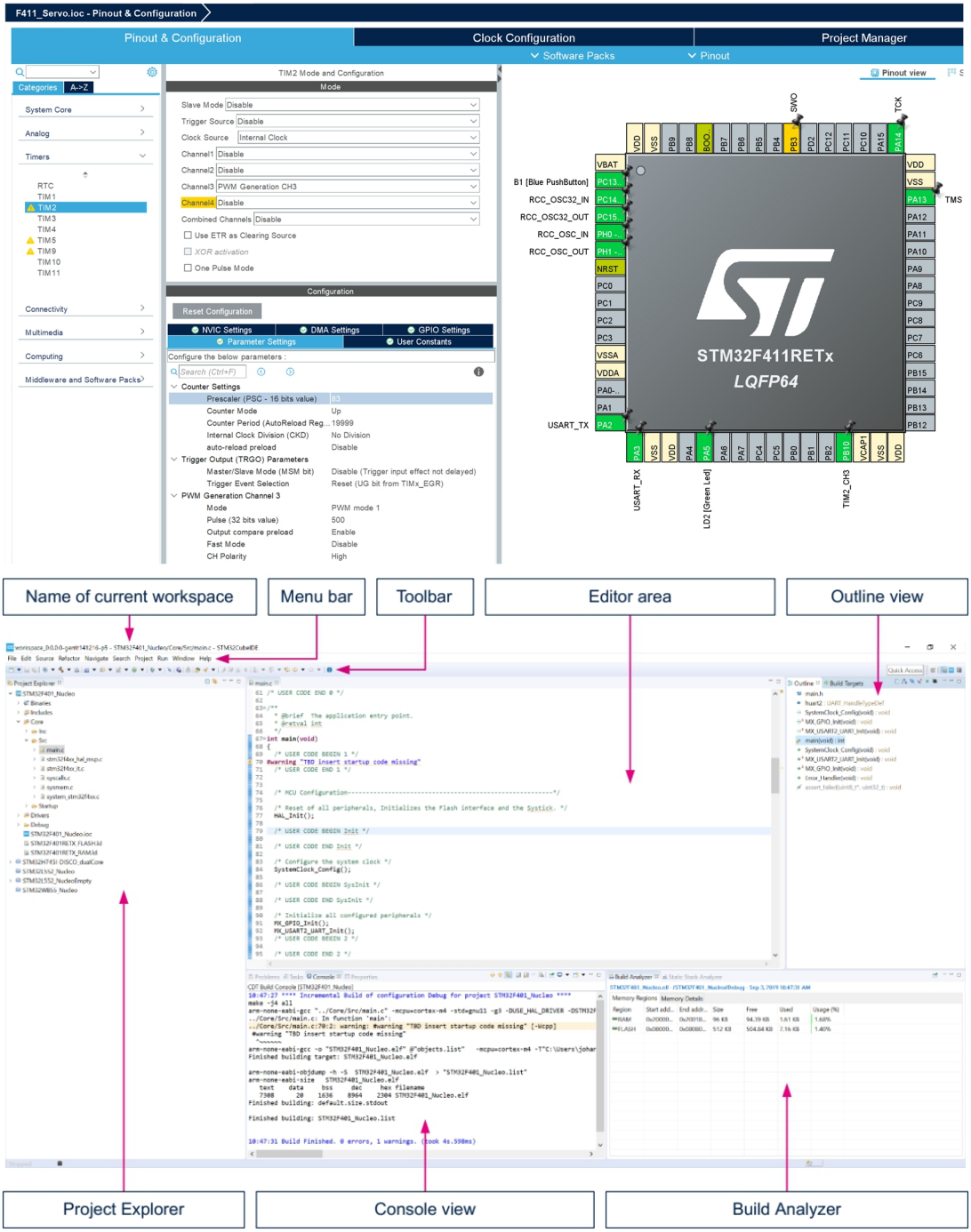

Figure 5. Environnement de programmation STM32CubeIDE

Ils débutent avec la carte de développement Nucleo F411RE de chez ST (figure 6), similaire à l'ObCP en termes de connectivités-fonctions pour être en avance de phase de la réalisation.

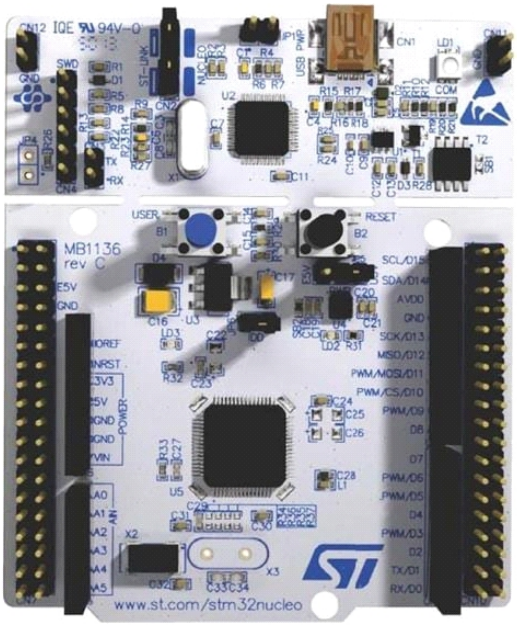

Figure 6. Carte Nucleo F411RE

Après un cours d'introduction, l'enchaînement TP/TD se fait en autonomie avec l'aide ponctuelle de l'enseignant en suivant un texte d'expérimentations traitant le contrôle d'entrées/sorties classiques, la communication UART, les timers, les interruptions, l'horloge temps réel RTC, les sorties PWM, la communication avec un capteur I2C, la réalisation d'interfaces utilisateur à l'aide d'afficheurs LCD ou graphiques et d'encodeurs rotatifs (figure 7), la commande de petits actionneurs (type moteurs pas à pas, continu et servo) et enfin la mise en œuvre du BLE avec le module IDB05A1 (figure 8) compatible avec celui de l'ObCP correspondant au IDB05A2.

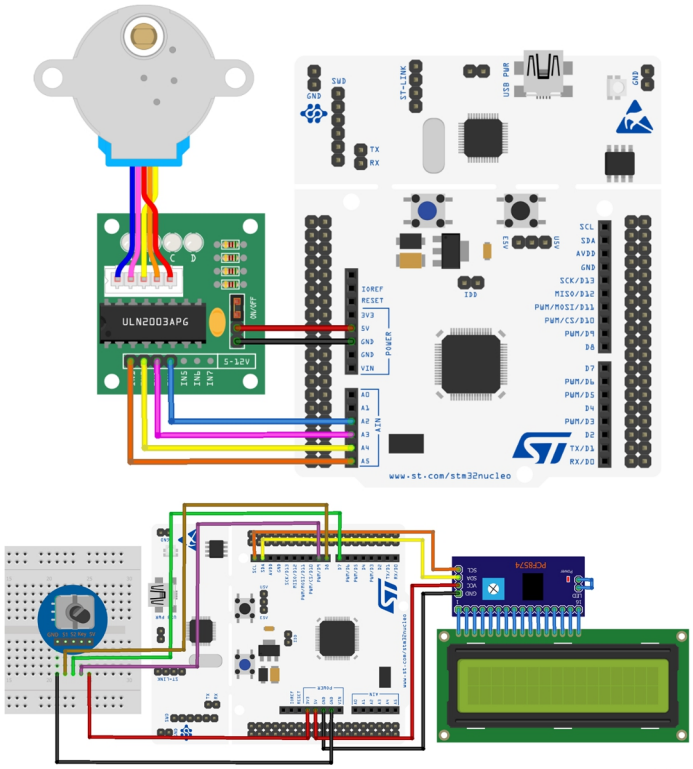

Figure 7. Exemples de mise en œuvre d'un moteur pas à pas et d'un écran LCD avec un encodeur rotatif

L'enseignement, de 10H TD et 8H TP, se base sur un document tutoriel de 270 pages complémenté d'une collection d'exemples sur un dépôt collaboratif GitHub [10].

Figure 8. Module Bluetooth Low Energy IDB05A1

La fin de cet enseignement consiste en la mise en œuvre de l'ObCP qui a été réalisé en parallèle. Une procédure de programmation est mise en place pour son utilisation. En effet, la carte de développement comporte un programmateur/debugger ST, comme le ST-Link V3 mini [11], qui n'est pas présent sur l'ObCP. La conception de la carte permet d'utiliser au choix un ST-Link externe à connecter sur l'ObCP ou le DFU (Device Firmware Update) via son port USB et le bootloader intégré au microcontrôleur. Pour cela, il suffit de générer le fichier de programmation binaire au format BIN HEX ou ELF dans l'environnement STM32CubeIDE puis de le téléverser par l'utilisation du programmateur STM32CubeProgrammer [12] fourni par ST. La procédure de programmation est intégralement décrite et documentée sur le repository LAB_ObCP_L476_STM32CubeIDE de l'espace GitHub.

L'ObCP utilise le **Nordic UART Service** (NUS), un service propriétaire développé par Nordic Semiconductors, qui permet d'émuler une interface UART (Universal Asynchronous Receiver-Transmitter) sur une connexion BLE (tableau 1).

Tableau 1. Caractéristiques du service NUS

	UUID
Service	6E400001-B5A3-F393-E0A9-E50E24DCCA9E
Carac. TX	6E400003-B5A3-F393-E0A9-E50E24DCCA9E
Carac. RX	6E400002-B5A3-F393-E0A9-E50E24DCCA9E

- **TX (Transmission)** : Cette caractéristique est utilisée pour envoyer des données du périphérique central (par exemple, un smartphone ou une tablette) vers l'ObCP. Le périphérique central écrit des données dans cette caractéristique, qui sont ensuite transmises à l'ObCP.

- **RX (Réception)** : Cette caractéristique est utilisée pour envoyer des données de l'ObCP vers le périphérique central. L'ObCP écrit des données dans cette caractéristique, et le périphérique central est notifié de la disponibilité de nouvelles données.

L'ObCP interprète les données reçues via la caractéristique TX ou un terminal sur le port USB comme des commandes spécifiques (tableau 2).

Tableau 2. Tableau des commandes disponibles de l'ObCP

Commandes	Description
R ou r suivi d'un entier de 0 à 65535	Contrôle de la LED rouge via PWM, exemple : r500 pour une intensité de 500.
V ou v suivi d'un entier de 0 à 65535	Contrôle de la LED verte via PWM, exemple : v500.
B ou b suivi d'un entier de 0 à 65535	Contrôle de la LED bleue via PWM, exemple : b500.
M ou m suivi d'un entier de 0 à 65535	Contrôle du PWM général, exemple : m500.
A ou a	Lecture de l'accéléromètre, retourne les valeurs X, Y et Z avec la chaîne : « Acceleration : X=x.xx Y=y.yy Z=z.zz »
T ou t	Lecture de la température. Retour de la chaîne : « T = tt »
cls	Effacement de l'écran du terminal USB.
start timer	Démarrage du timer 1s (BLE). Envoi périodique de 2 chaînes : « Xx.xxYy.yyZz.zz T = tt »
stop timer	Arrêt du timer 1s (BLE).
#rst	Réinitialisation de toutes les fonctions.
?	Affiche l'aide avec la liste des commandes.

Exemple de communication :

1. Le périphérique central envoie la commande « A » en écrivant cette chaîne de caractères dans la caractéristique TX.

2. L'ObCP reçoit cette commande, mesure l'accélération et envoie les valeurs mesurées dans la caractéristique RX.

3. Le périphérique central est notifié de la disponibilité de nouvelles données dans la caractéristique RX et peut lire les valeurs d'accélérations sous le format : « Acceleration : X=x.xx Y=y.yy Z=z.zz ».

A noter que les données envoyées via BLE sont découpées en **paquets de 20 caractères dans le programme** en raison de la **limitation de la taille des trames BLE** imposée par la spécification Bluetooth Low Energy. Cette contrainte vient du **Maximum Transmission Unit**

(MTU) par défaut, qui est souvent fixé à 23 octets, dont **3 octets sont réservés au protocole ATT** (Attribute Protocol), laissant ainsi **20 octets pour les données utiles**.

Cette segmentation garantit une compatibilité maximale avec les périphériques BLE (smartphones, tablettes, Raspberry Pi) et assure une transmission fiable sans dépassement de la taille maximale autorisée pour un seul paquet. Pour envoyer des données plus longues, plusieurs paquets successifs sont nécessaires, et le récepteur doit les reconstituer correctement.

3.2. Programmation MyRio

Les étudiants ayant une formation LabVIEW en deuxième année, la partie webserveur est initiée sur ce composant pour sa simplicité de mise en œuvre à l'aide d'un tutoriel. Un clic droit sur la cible MyRio permet la création du serveur web (figure 9).

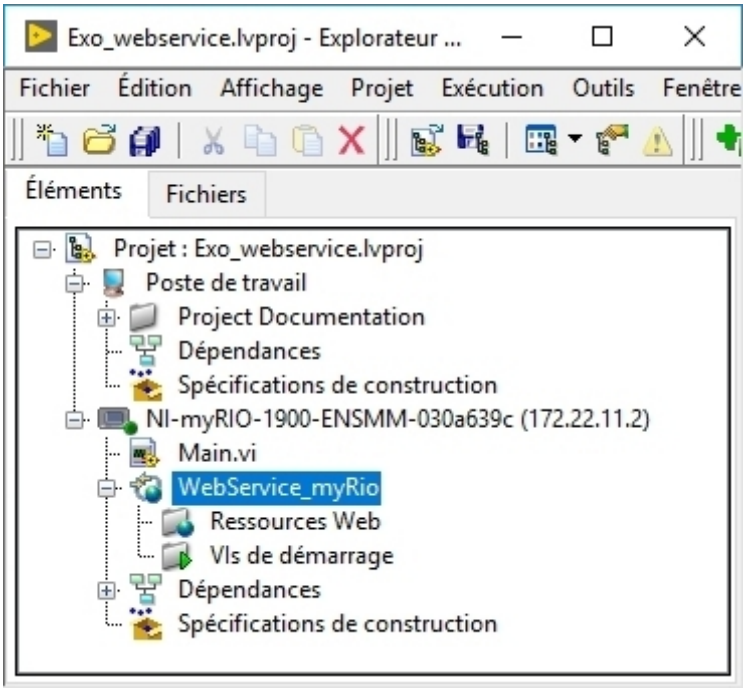

Figure 9. Web Serveur LabVIEW

La réalisation d'un programme LabVIEW en insérant un VI méthode HTTP permet de partager les variables éditées dans le connecteur comme les valeurs de l'accélération en lecture et les leds intégrées au MyRIO en écriture (figure 10).

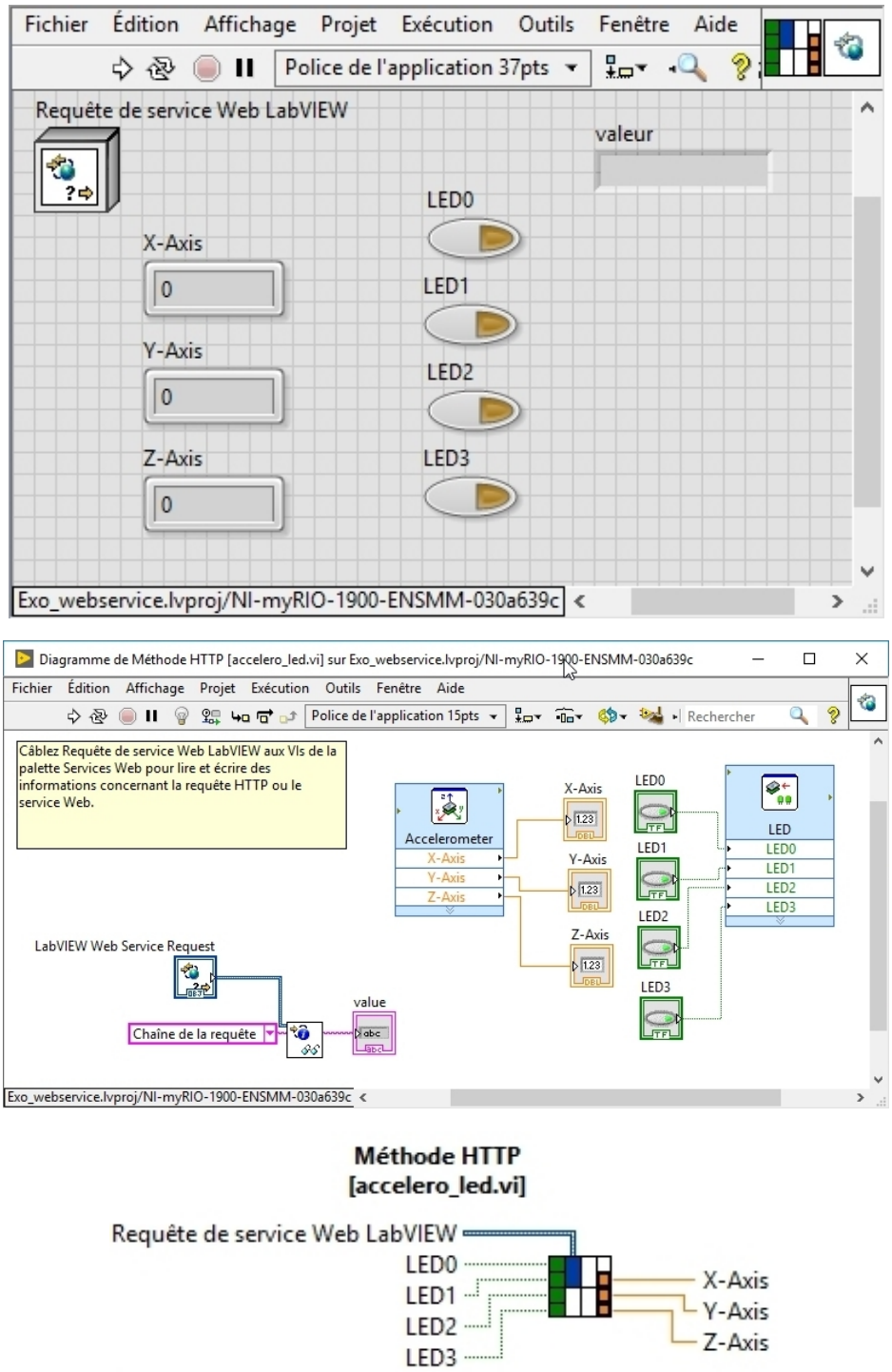

Figure 10. Programme LabVIEW intégrant la méthode HTTP

La sortie du webserveur reste basique et se présente sous la forme d'une page XML simple (figure 11).

```
<?xml version="1.0"?>
- <Response>
  - <Terminal>
      <Name>Z-Axis</Name>
      <Value>0.98046875</Value>
    </Terminal>
  - <Terminal>
      <Name>Y-Axis</Name>
      <Value>0.03125</Value>
    </Terminal>
  - <Terminal>
      <Name>X-Axis</Name>
      <Value>-0.03125</Value>
    </Terminal>
</Response>
```

Figure 11. Page web en XML du webserveur LabVIEW

Dans le cadre de l'amélioration du site web, il leur est proposé de structurer le site avec des pages de présentation intégrant une image et une mise en forme de la page d'accueil. Les étudiants prennent en main le fichier HTML de la page d'accès pour mettre en œuvre la visualisation vue sur figure 12.

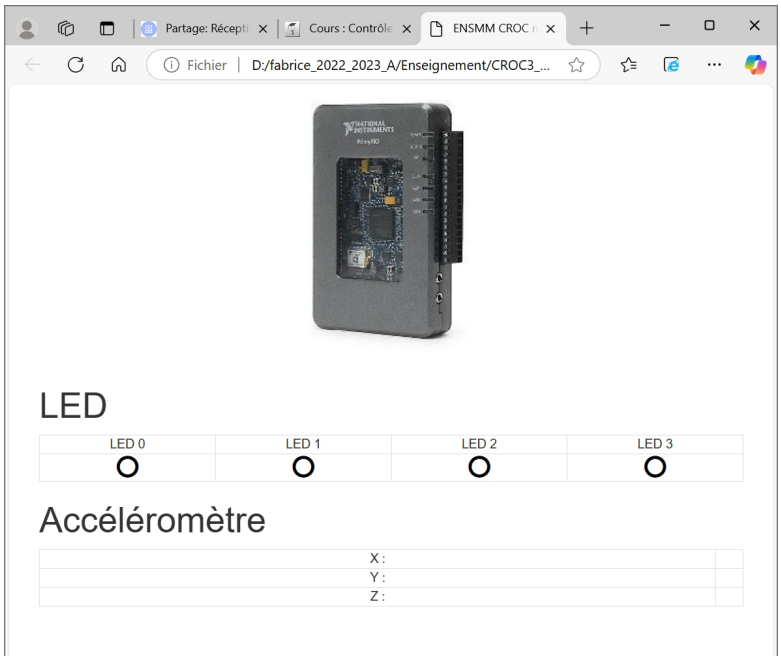

Figure 12. Page web finale

L'interaction avec le programme LabVIEW est effectuée en JavaScript. Le site web est structuré en termes de style avec un fichier CSS. L'exercice est proposé sous forme de tutoriel implémenté dans Moodle. Les concepts de bases sont ainsi diffusés en 8h de TD guidé sous Moodle.

3.3. Programmation sur Raspberry

La dernière partie est réalisée sur Raspberry avec un langage laissé au choix de l'étudiant. Le codage est quand même principalement effectué en Python. L'apprentissage de la Raspberry se fait par un tutoriel progressif assez classique : présentation de la Raspberry, installation de la distribution, prise en main à distance avec SSH, compilation en C, utilisation de Python, utilisation des GPIO, utilisation de la caméra, installation du webserveur apache et du langage PHP, installation d'une base de données.

La communication BLE a été vue durant les séances de TD sur microcontrôleur à travers une implémentation de commande du type AT sur l'ObCP, elle est reprise sur la Raspberry avec un programme en Python (figure 13).

```python
import bluepy.btle as btle
import time

class MyDelegate(btle.DefaultDelegate):
    def __init__(self):
        btle.DefaultDelegate.__init__(self)
        self.data = b''
    def handleNotification(self, cHandle, data):
        self.data += data

p = btle.Peripheral("F7:B2:93:F1:17:6C")
p.setDelegate(MyDelegate())
s = p.getServiceByUUID("6e400001-b5a3-f393-e0a9-e50e24dcca9e")
d = s.getCharacteristics("6e400002-b5a3-f393-e0a9-e50e24dcca9e")[0]
cccd = d.getDescriptors(forUUID=btle.AssignedNumbers.client_characteristic_configuration)[0]
notif = cccd.read()

c = s.getCharacteristics("6e400003-b5a3-f393-e0a9-e50e24dcca9e")[0]
c.write(bytes("A\r", "utf-8"))
time.sleep(0.1)
segment = d.read()
data3 = p.delegate.data
print(data3.decode('utf-8'))
p.disconnect()
```

Figure 13. Code Python pour le contrôle Bluetooth

Le code Python fait appel à la librairie bluepy. Dans le programme proposé, l'adresse mac de l'ObCP doit être connue. Elle est recherchée par les étudiants à travers l'utilisation de leur smartphone personnel et d'une application Android de détection type BLE Terminal. L'identification des services et caractéristiques BLE utilisés par l'ObCP est reprise de la partie microcontrôleur. Dans le programme python, la mise en œuvre des notifications de l'ObCP permet de simplifier la lecture des données quand le buffer BLE dépasse les 20 octets.

La réalisation du site web est obtenue à partir du modèle présenté sur MyRio, mais en privilégiant PHP comme langage d'interaction pour la page web. La figure 14 présente le visuel obtenu par les étudiants. L'objectif est de récupérer les données de l'accéléromètre de l'ObCP.

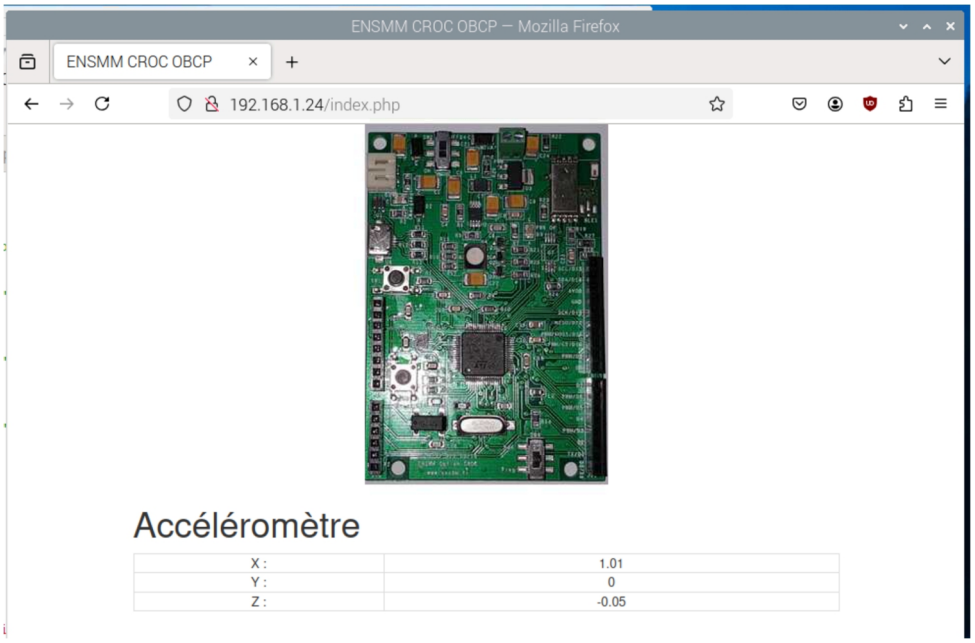

Figure 14. Page web finale

La page HTML est agrémentée du code PHP (figure 15). La commande principale consiste à lancer le programme python réalisé précédemment pour exécuter les commandes sur l'ObCP connecté en BLE à la Raspberry.

```php
<?php
$output3 = shell_exec('sudo python3 /home/pi/Documents/prog_python/ble_lecture_accel_obcp_v2.py');
$values = explode(' ', trim($output3));
// Extraire les valeurs des chaînes de caractères
$x = isset($values[2]) ? floatval(explode('=', $values[2])[1]) : 'N/A';
$y = isset($values[3]) ? floatval(explode('=', $values[3])[1]) : 'N/A';
$z = isset($values[4]) ? floatval(explode('=', $values[4])[1]) : 'N/A';
?>
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ENSMM CROC OBCP</title>
    <link rel="stylesheet" href="include/bootstrap/3.3.7/css/bootstrap.min.css">
    <link rel="stylesheet" href="include/bootstrap/3.3.7/css/bootstrap-theme.min.css">
    <link rel="stylesheet" href="include/font-awesome/4.7.0/css/font-awesome.min.css">
    <link rel="stylesheet" href="include/css/style.css">
</head>
<body>
    <header>
        <img src="include/img/obcp.png" alt="obcp" class="img-responsive center-block">
    </header>
    <section>
        <article>
            <div class="container">
                <h1>Accéléromètre</h1>
                <div class="row">
                    <div class="col-xs-12">
                        <table class="table-bordered text-center col-xs-12">
                            <tr>
                                <td>X :</td>
                                <td><span id="ac_x"><?php echo $x; ?></span></td>
                            </tr>
                            <tr>
                                <td>Y :</td>
                                <td><span id="ac_y"><?php echo $y; ?></span></td>
                            </tr>
                            <tr>
                                <td>Z :</td>
                                <td><span id="ac_z"><?php echo $z; ?></span></td>
                            </tr>
                        </table>
                    </div>
                </div>
            </div>
        </article>
    </section>
</body>
</html>
```

Figure 15. Code HTML et PHP

L'apprentissage de la programmation Raspberry se déroule sur 12h TD. La partie finale est implémentée durant la séance de TP, qui clôture l'enseignement réseau système embarqué avec le montage d'un réseau avec PC filaire, pc portable, tablette, smartphone, routeur MyRIO et Raspberry plus ObCP (figure 16).

L'évaluation des étudiants est faite sur l'avancement lors des TD dans les trois domaines de programmation, réseaux, microcontrôleurs et systèmes embarqués. La partie TP réseaux finale est challengée entre les groupes avec un bonus de note pour le réseau le plus complet Compte tenu des durées d'enseignement la moyenne est faite avec la même pondération pour les trois parties. Il n'y a pas d'examen classique sur le module CROC2 qui représente un quart de la valeur de l'option.

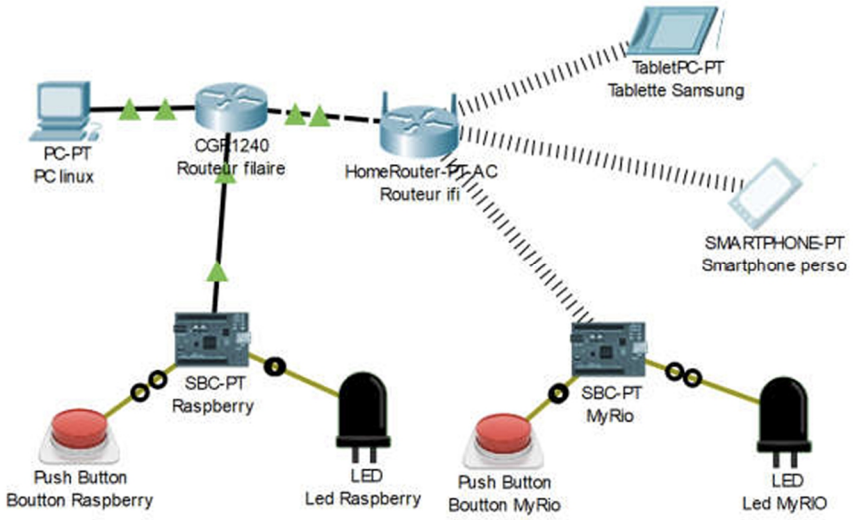

Figure 16. Architecture de principe du TP réseau

4. Retour étudiants – enseignants

4.1. Etudiants

Le retour étudiant est très positif, ils ont vraiment apprécié le lien fort entre enseignements, réalisations et projet. Les étudiants ont totalement adhéré au principe de l'ObCP. La partie programmation sur Raspberry est la plus appréciée par les étudiants, car plus accessible et plus simple à prendre en main par rapport à leur formation initiale.

Ils comprennent la multiplicité des plateformes qui leur sont présentées, mais aimeraient passer plus de temps sur chacune d'elles. L'autorisation d'utiliser les outils comme ChatGpT ou Copilot est appréciée et leur donne un sentiment de réussite et de facilité.

Globalement, les étudiants apprécieraient une augmentation du volume horaire des enseignements concrets au détriment d'autres enseignements moins appliqués.

Les remarques proposées par les étudiants lors des enquêtes sur les enseignements sont quelques fois surprenantes. « Remarques/Suggestions générales sur l'ensemble des semestres académiques : J'aurais préféré des créneaux de 1h30 de cours plutôt que 2h00. Ça respecterait davantage le rythme ultradien d'apprentissage et de productivité humaine : de très nombreuses fois, j'ai ressenti le besoin de faire une pause au bout d'environ 1h30 de cours, et ai passé les 30 minutes restantes plutôt déconcentré et avec un sentiment d'inefficacité. Les journées de 8h de cours sont particulièrement longues, et il était difficile pour moi de devoir commencer quasi systématiquement les cours à 8h00. Les chronotypes de chacun étant différents, cela convient à certains et pas à d'autres, dont l'heure naturelle de lever est plus tardive. Commencer la journée à 8h50/9h00 m'aurait donc permis d'être plus en forme, plus concentré et plus efficace en cours et donc aussi d'être plus motivé à participer en TD ».

4.2. Enseignants

L'utilisation de LLM type Chat-Gpt est autorisée aux étudiants pour leur permettre de gagner du temps sur les phases de programmation, néanmoins leur analyse critique du code proposé reste limitée. La compréhension de la présence de chaque ligne de code dans le programme n'est pas forcément une préoccupation pour eux dès lors que le code est fonctionnel.

Les aspects des objets connectés étant très variés et nombreux, la difficulté principale consiste à faire passer les notions essentielles dans la programmation comme la rigueur du code et l'importance des commentaires dans le programme, quel que soit le langage utilisé.

Le travail est essentiellement individualisé, puisque chaque étudiant dispose de ses plateformes de programmation en travaux dirigés. Les étudiants peuvent évoluer à leur rythme. La mise à disposition en dehors des heures de cours est aussi proposée, mais rencontre un succès mitigé. Cependant, la progression des étudiants et leur montée en compétence sont continues sur la durée du semestre. Le challenge d'intégrer toutes les fonctionnalités dans un TP final est apprécié et motivant pour les groupes formés pour l'occasion.

5. Conclusion

Sans compter la réalisation de l'ObCP, l'ensemble des enseignements développés dans cet article représente quatorze heures de cours, trente heures de travaux dirigés sur dispositifs programmables et douze heures de travaux pratiques. Cela constitue environ un quart des enseignements de l'option. Le retour sur les deux dernières années de ces enseignements est très concluant. L'intégration des différentes plateformes sur la durée du semestre permet un lien dans les différentes activités.

Les exercices proposés pourront être fournis par les auteurs.

Références

[1] ENSMM : https://www.supmicrotech.fr/, consulté le 7 février 2025.
[2] Programme détaillé de la formation initiale, https://www.supmicrotech.fr/fr/contenu-et-organisation/ consulté le 7 février 2025.
[3] J. Imbaud, D. Vernier, P. Abbe, F. Sthal, "Objet Connecté Personnel : ObCP", J3eA, 21 (2022) 2019. DOI: https://doi.org/10.1051/j3ea/20222019.
[4] Nucleo F411RE : www.st.com/en/evaluation-tools/nucleo-f411re.html , consulté le 13 février 2025.
[5] Shield IDB05A1 : www.st.com/en/ecosystems/x-nucleo-idb05a1.html , consulté le 13 février 2025.
[6] GitHub JImbaud : https://github.com/jimbaud, consulté le 13 février 2025.
[7] https://www.ni.com/fr-fr/shop/model/myrio-1900.html, consulté le 7 février 2025.
[8] https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/, consulté le 7 février 2025.
[9] STM32CubeIDE : www.st.com/en/development-tools/stm32cubeide.html, consulté le 13 février 2025.
[10] Module Bluetooth BlueNRG-M0 : www.st.com/en/wireless-connectivity/bluenrg-m0.html, consulté le 13 février 2025.
[11] ST-LINK V3 mini: www.st.com/en/development-tools/stlink-v3minie.html, consulté le 13 février 2025.
[12] STM32CubeProgrammer : www.st.com/en/development-tools/stm32cubeprog.html , consulté le 13 février 2025.

Biographie des auteurs

Fabrice Sthal est professeur des universités en section CNU 63 à SUPMICROTECH-ENSMM depuis 2009. Il est responsable de l'option CROC de 3ème année et impliqué dans de nombreux développement de projets pédagogiques innovants (16 publications à caractère pédagogique). Spécialisé en instrumentation, piézoélectricité et métrologie temps-fréquence, il est membre du département Temps-Fréquence de l'institut FEMTO-ST et actuellement directeur de l'école doctorale Sciences Physique pour l'Ingénieur et Microtechniques.

Joël Imbaud : est Maître de conférences à SUPMICROTECH-ENSMM depuis 2009, rattaché à la section CNU 63. Spécialiste en conception électronique, industrialisation électronique et électronique embarquée, il développe ses activités d'enseignement et de recherche en lien étroit avec les besoins industriels et les enjeux technologiques actuels. Il est responsable de la plateforme partenariale SUPMICROTECH-ENSMM, un espace d'innovation pédagogique favorisant les collaborations entre l'enseignement supérieur, la recherche et le tissu industriel. À ce titre, il coordonne de nombreux projets multidisciplinaires associant étudiants, enseignants et entreprises, visant à renforcer l'apprentissage par projet et la professionnalisation des formations d'ingénieurs. Ses activités de recherche sont menées au sein du département Temps-Fréquence de l'Institut FEMTO-ST, où il s'intéresse notamment aux résonateurs acoustiques et oscillateurs de haute stabilité, tout en valorisant son expertise par la mise en place d'enseignements appliqués en électronique, microtechniques et systèmes embarqués.