

Guide complet

Mise à jour du hardware et software de deux
spectrofluorimètres



Auteur :
Youssef BOUSSALEM

Rôle :
Stagiaire, étudiant en 4ème année à l'INSA Strasbourg

Encadrant :

Lorry Engel

Rôle :

Ingénieur en électronique

**IPCMS - Institut de Physique et Chimie des Matériaux de
Strasbourg**

1^{er} août 2025

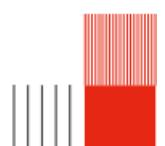
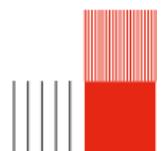


Table des matières

Introduction	1
1 Manuel de fabrication	3
1.1 Où trouver les ressources pour ce projet ?	3
1.2 Analyse des spectrofluorimètres récupérés	3
1.3 Mise à jour du spectrofluorimètre A	4
1.3.1 Éléments nouveaux	4
1.3.2 Démontage	6
1.3.3 Montage	6
1.3.4 Configuration et étalonnage	7
1.4 Mise à jour du spectrofluorimètre B	8
1.4.1 Éléments nouveaux	8
1.4.2 Démontage des monochromateurs	10
1.4.3 Montage des monochromateurs	12
1.4.4 Démontage du boîtier central	14
1.4.5 Montage du boîtier central	14
1.4.6 Configuration et étalonnage	14
2 Documentation du programme de contrôle des spectrofluorimètres	16
2.1 Programme Arduino	16
2.2 Programme Python	17
2.2.1 main.py	17
2.2.2 Classe Wavelength	18
2.2.3 Classe WLRange	19
2.2.4 Classe IntegrationTime	19
2.2.5 Classe Slit	19
2.2.6 Classe Monochromator	20
2.2.7 Classe Signal	20
2.2.8 Classe Measure	20
3 Guide utilisateur	22
3.1 Faire une mesure	22
3.1.1 Fichier de configuration des mesures	22
3.1.2 Lancer le programme	23

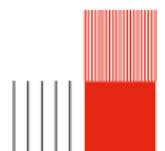


Introduction

Lors d'un stage à l'IPCMS, j'ai eu l'occasion de travailler sur la mise à jour du hardware et software de deux spectrofluorimètres différents.

L'objectif est de proposer des spectrofluorimètres fonctionnels avec des composants plus récents. Pour cela, il a fallu remplacer les éléments de contrôle des moteurs dans les monochromateurs ainsi que le système d'obtention des signaux de sortie et de référence.

Dans ce document, nous verrons un tutoriel précis afin de répliquer ce projet, puis de la documentation sur le code utilisé, et finalement un guide utilisateur.



Chapitre 1

Manuel de fabrication

1.1 Où trouver les ressources pour ce projet ?

L'ensemble des ressources de ce projet est disponible sur GitHub :
<https://github.com/YoussefBoussalem/Spectrofluorimetre>.

Vous y trouverez tout le nécessaire pour mener à bien ce projet : le code source, les schémas électriques ainsi que les pièces à imprimer.

1.2 Analyse des spectrofluorimètres récupérés

Deux spectrofluorimètres ont été récupérés :

- Le premier, un **Spex 1680B 0.22m Double Spectrometer**, sera désigné par la suite comme *spectrofluorimètre A*.
- Le second, un **Fluorolog 3-11**, sera désigné comme *spectrofluorimètre B*.



FIGURE 1.1 – Photo d'un monochromateur du spectrofluorimètre A

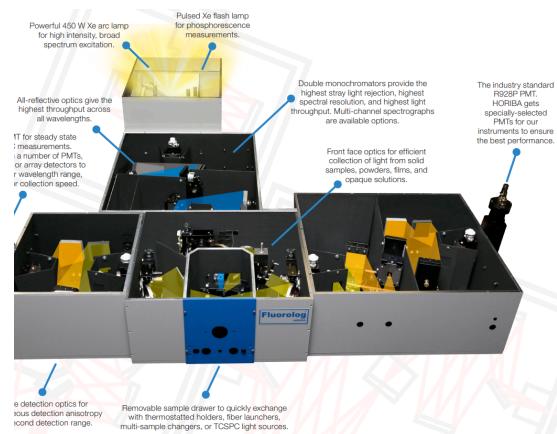


FIGURE 1.2 – Photo du spectrofluorimètre B

Ces deux modèles sont des spectrofluorimètres à double réseau de diffraction, permettant une meilleure résolution spectrale. Ils utilisent des moteurs pas à pas pour l'actionnement, ainsi que des solénoïdes pour contrôler les obturateurs (*shutters*).

Ils sont pilotés principalement à l'aide :

- de cartes Arduino Uno, accompagnées de shields et de drivers pour moteurs pas à pas ;
- d'une Raspberry Pi ;
- d'un microcontrôleur ATTiny414 pour le comptage du signal issu du tube photomultiplicateur ;
- de circuits imprimés (PCB) conçus pour les capteurs et le pilotage des solénoïdes.



Les différences majeures entre les deux spectrofluorimètres sont les suivantes :

- Les monochromateurs du spectrofluorimètre A intègrent un système mécanique permettant de linéariser la relation entre l'angle du réseau de diffraction et la longueur d'onde de sortie. Ce dispositif est absent du spectrofluorimètre B, où la relation est de la forme : $\lambda = \lambda_0 + A \sin(k\theta)$. Voir [1] pour plus de détails.
- Bien que les deux modèles soient équipés de fentes, celles du spectrofluorimètre A ne sont pas motorisées et doivent être réglées manuellement.
- Le spectrofluorimètre A utilisait initialement un tube photomultiplicateur pour la détection du signal de référence. Celui-ci a été remplacé par une photodiode **S1223-01**, nécessitant la fabrication d'un support ajustable à monter dans la chambre optique (où est placé l'échantillon).
- Le spectrofluorimètre A utilise des interrupteurs de fin de course comme capteurs de position, tandis que le spectrofluorimètre B utilise des capteurs optiques infrarouges. Un PCB spécifique est donc requis pour faire fonctionner les monochromateurs du modèle B.

La stratégie globale adoptée pour la remise en état et la modernisation des appareils a été la suivante :

- Montage de l'Arduino, du shield, des drivers et des PCB nécessaires au pilotage.
- Installation des programmes et bibliothèques nécessaires sur l'Arduino et la Raspberry Pi.
- Démontage des spectrofluorimètres pour accéder aux monochromateurs et à l'intérieur des chambres optiques.
- Intégration des composants nécessaires à l'intérieur des monochromateurs puis du boîtier central.
- Assemblage final des différents éléments, configuration des dispositifs puis étalonnage.

La suite de ce chapitre détaille, étape par étape pour chaque spectrofluorimètre, les procédures à suivre pour mener à bien la réalisation du projet.

1.3 Mise à jour du spectrofluorimètre A

1.3.1 Éléments nouveaux

Raspberry Pi – Installation

Si ce n'est pas déjà fait, procurez-vous une Raspberry Pi accompagnée d'un clavier, d'une souris et disposant d'un nombre suffisant de ports USB pour y connecter deux Arduino Nano ainsi qu'un ATtiny414.

Nous recommandons un modèle de Raspberry Pi disposant d'au moins 4 ports USB, ainsi que le **clavier officiel Raspberry Pi**, qui permet également d'étendre le nombre de ports.

Si votre Raspberry Pi est livrée avec un boîtier (recommandé), montez celui-ci, puis installez l'OS à l'aide de la carte micro SD fournie, ou en utilisant l'outil *Raspberry Pi Imager* sur une carte vierge.

Assurez-vous que Python 3 est bien installé et à jour. Vous pouvez le vérifier dans l'invite de commande avec la ligne suivante : **python -V**.

Enfin, installez les bibliothèques nécessaires :

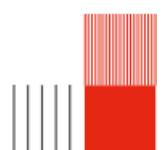
- Installation de PyYAML via la commande : **pip3 install PyYAML**

L'installation du logiciel Arduino sur la Raspberry Pi n'est pas nécessaire. Toutefois, le téléchargement du programme sur les deux Arduino Uno est indispensable.

Arduino Uno, Shield et Drivers

Le montage se fait avec un Arduino Uno, un shield compatible CNC et les drivers correspondants, comme ceux du **kit ARD-CNC-Kit1**.

Le montage doit suivre précisément les instructions de la documentation technique du kit. Soyez particulièrement vigilant au sens de montage des drivers, et veillez à bien installer les radiateurs sur ceux-ci.



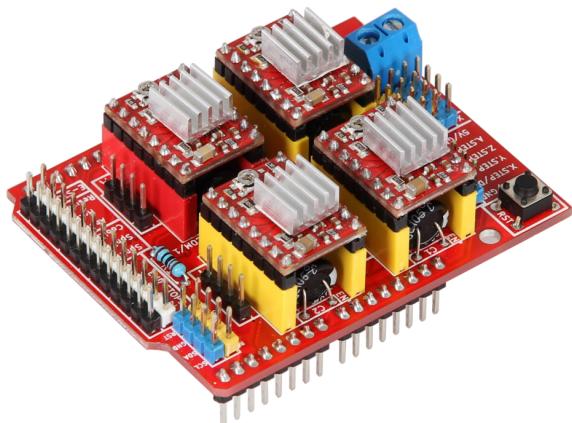


FIGURE 1.3 – Image du shield Arduino avec les 4 drivers du kit CNC

Pour le spectrofluorimètre A, un seul driver est nécessaire : il sert à contrôler le moteur responsable de l'orientation du réseau de diffraction.

Alimentation des moteurs

Les moteurs des deux spectrofluorimètres sont alimentés par une source 9V (12V est également compatible). Pour le spectrofluorimètre A, une alimentation de 1A suffit, tandis que pour le modèle B, il est recommandé de prévoir au moins 3A.

Chaque monochromateur nécessite sa propre alimentation : comptez donc deux alimentations par spectrofluorimètre.

Afin de pouvoir utiliser les modèles 3D fournis sans modification, nous conseillons de choisir :

- une alimentation avec une prise *barrel jack*,
- deux ports de connexion 5.5 mm × 2.1 mm,
- un interrupteur (optionnel) de dimensions 19 mm × 13 mm.



FIGURE 1.4 – Photo d'une ali-
mentation DC



FIGURE 1.5 – Photo du connec-
teur d'alimentation

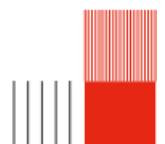
Impression 3D

Pour effectuer l'assemblage complet, les pièces suivantes doivent être imprimées :

- **SPECA - monochromator - Part 1** (2 exemplaires)
- **SPECA - monochromator - Part 2** (2 exemplaires)

Si vous optez pour le remplacement du tube photomultiplicateur par une photodiode, il sera également nécessaire d'imprimer les pièces suivantes :

- **SPECA - ref diode holder - CAP**
- **SPECA - ref diode holder - SLIDER**
- **SPECA - ref diode holder - TOWER**



1.3.2 Démontage

Séparation des boîtiers

Commencer par séparer les différents boîtiers afin d'accéder à l'intérieur des monochromateurs et du boîtier central. Pour cela, dévisser les vis reliant les différents modules.

Ouverture des monochromateurs

Placer les monochromateurs sur leur flanc pour accéder aux éléments électroniques situés en dessous, puis retirer la tôle de protection en dévissant toutes les vis.

Démonter ensuite la plaque frontale du monochromateur et déconnecter les câbles du PCB.



FIGURE 1.6 – Photo de l'intérieur d'un monochromateur

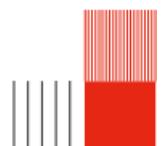
1.3.3 Montage

Couper les câbles d'origine puis les sertir afin de pouvoir les connecter directement à l'Arduino Uno et à son shield CNC.

Connecter les éléments comme suit :

- Brancher les 4 fils du moteur pas à pas (les deux aux extrémités et les deux centraux parmi les 6) sur les bornes du driver X du shield.
- Brancher les capteurs de fin de course au shield :
 - Le capteur correspondant à la longueur d'onde la plus faible sur la broche **RESET/ABORT**.
 - Le second capteur sur la broche **FEED/HOLD**.

Enfin, monter l'Arduino sur la pièce **SPECA - monochromator - Part 2**. Faire passer le câble Arduino, le bouton, et le connecteur d'alimentation DC à travers l'assemblage avec la pièce **SPECA - monochromator - Part 1**, puis visser l'ensemble à l'avant du monochromateur.



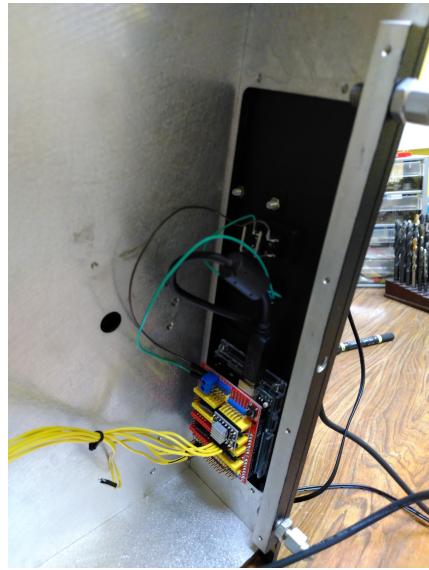


FIGURE 1.7 – Photo de l’intérieur d’un monochromateur avec l’Arduino Uno monté

1.3.4 Configuration et étalonnage

Pour que le spectrofluorimètre fonctionne correctement, deux étapes sont nécessaires :

- Configurer le sens de rotation des moteurs.
- Étalonner les monochromateurs.

Configuration des moteurs

La configuration s’effectue via une communication série avec l’Arduino Uno. Pour cela, utiliser l’IDE Arduino depuis un ordinateur ou depuis la Raspberry Pi (l’IDE n’est pas installé par défaut).

Voici les commandes de base utilisées pour configurer les moteurs :

- MOVE,<Nom Du Moteur>, <Nombre de pas>, <Sens de rotation (0 ou 1)>
- Les noms de moteurs reconnus sont : WL, SLIT1, SLIT2, et SLIT3, qui correspondent respectivement au :
 - moteur d’orientation du réseau de diffraction,
 - moteur de la fente au-dessus du port de connexion,
 - moteur de la fente centrale,
 - moteur de la fente la plus éloignée du port.

Aucune convention de sens n’étant imposée au câblage des moteurs, il est nécessaire de configurer manuellement leur direction dans le fichier de configuration Arduino :

- Ouvrir le fichier **config.h** utilisé dans le code Arduino (ligne 2). Adapter ce fichier selon que vous utilisez le premier ou le second monochromateur (type A ou B).
- Dans la structure **motors[]**, ajuster la vitesse de chaque moteur. Celle-ci est définie par le temps d’attente entre deux pas : diminuer ce temps pour accélérer, l’augmenter pour ralentir. Le but est d’obtenir une rotation fluide, sans à-coups.
- Modifier la variable **zeroDirection** (avant-dernière variable) en LOW ou HIGH afin que la direction 0 de la commande MOVE corresponde à :
 - une diminution de la longueur d’onde pour le moteur WL ;
 - une fermeture de fente pour les moteurs SLIT.

Une fois cette configuration terminée, téléverser le programme sur l’Arduino.

Configuration série et étalonnage

Ouvrir le programme Python et localiser le fichier YAML correspondant au spectrofluorimètre de type A ou B dans le dossier **SYSTEM_CONFIG**.



Effectuer les réglages suivants :

- Modifier les ports de connexion pour chaque monochromateur :
`/dev/ttyUSB<numero du port>` (entre 1 et 4).
- Déterminer les coefficients de conversion entre nombre de pas moteur et longueur d'onde :
 - Pour les monochromateurs de type A :
relation linéaire : $\lambda = \lambda_0 + \alpha \cdot k_{\text{pas}}$.
 - Pour les monochromateurs de type B :
relation non linéaire : $\lambda = \lambda_0 + \alpha \cdot \sin(\beta \cdot k_{\text{pas}})$.
- Pour les monochromateurs de type B, calibrer également la résolution spectrale (longueur d'onde par pas) selon une relation linéaire.
- Enfin, ajuster expérimentalement les valeurs minimale et maximale de pas autorisées pour chaque moteur. Cela permet de garantir que les longueurs d'onde accessibles et la résolution restent dans les plages physiques de fonctionnement.

1.4 Mise à jour du spectrofluorimètre B

1.4.1 Éléments nouveaux

Raspberry Pi – Installation

Si ce n'est pas déjà fait, procurez-vous une Raspberry Pi accompagnée d'un clavier, d'une souris et disposant d'un nombre suffisant de ports pour y connecter deux Arduino Nano et un ATtiny414.

Il est recommandé d'utiliser un modèle de Raspberry Pi avec au moins 4 ports USB, ainsi que le [clavier officiel Raspberry Pi](#), qui permet d'étendre le nombre de ports disponibles.

Si votre Raspberry Pi est fournie avec un boîtier (recommandé), montez-le, puis installez le système d'exploitation via la carte micro SD incluse ou en utilisant le logiciel Raspberry Pi Imager.

Vérifiez que la version de Python installée est à jour (Python 3) en exécutant la commande suivante dans le terminal : **`python -V`**.

Installez ensuite les bibliothèques nécessaires :

- yaml via : **`pip3 install PyYAML`**

L'installation de l'IDE Arduino sur la Raspberry Pi n'est pas obligatoire, mais vous devrez impérativement téléverser le programme sur vos deux Arduino Uno.

Arduino Uno, Shield et Drivers

On utilise un Arduino Uno avec un shield CNC et ses drivers ([ARD-CNC-Kit1](#)).

Le montage suit les instructions de la documentation fournie avec le kit. Veillez à orienter correctement les drivers et à installer les radiateurs sur chacun d'eux.

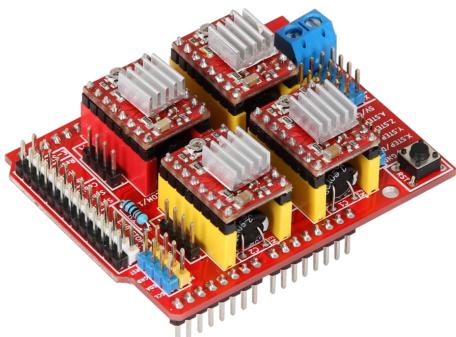


FIGURE 1.8 – Image du shield Arduino avec les 4 drivers du kit CNC



Pour le spectrofluorimètre B, quatre drivers sont nécessaires : un pour le moteur d'orientation du réseau de diffraction et trois pour les moteurs des fentes.

Alimentation des moteurs

Les moteurs sont alimentés via une source 9 V (ou 12 V, également compatible). Pour le spectrofluorimètre B, une alimentation d'au moins 3 A est requise, contre 1 A pour le modèle A.

Chaque monochromateur nécessite une alimentation indépendante : soit deux alimentations par spectrofluorimètre.

Pour réutiliser les modèles 3D fournis sans modification, privilégier une alimentation avec :

- une prise *barrel jack*,
- deux ports de connexion 5.5 mm × 2.1 mm.



FIGURE 1.9 – Photo d'une ali-
mentation DC



FIGURE 1.10 – Photo du
connecteur d'alimentation

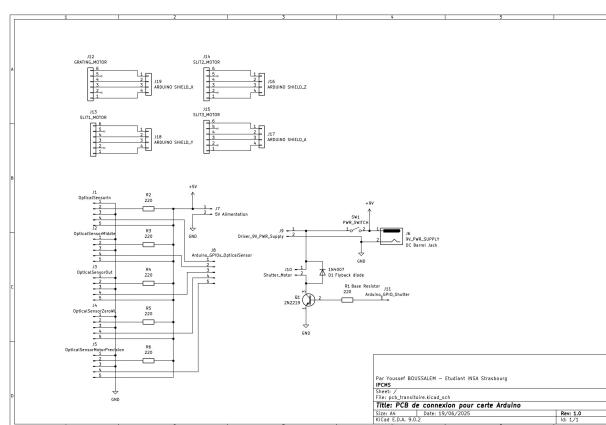
PCB pour les monochromateurs

Un nouveau PCB est nécessaire pour :

- gérer les capteurs optiques de position,
- contrôler le solénoïde de l'obturateur (shutter) dans le monochromateur d'excitation,
- simplifier le branchement des moteurs pas à pas.

Son [schéma électrique](#), réalisé avec KiCad, est relativement simple. Il comprend :

- l'alimentation des LED infrarouges pour les cinq capteurs optiques,
- un transistor pour le pilotage du solénoïde,
- quatre connecteurs permettant de convertir les connexions moteur de 6 fils (unipolaires) en 4 fils (bipolaires).



Impression 3D

Les pièces suivantes doivent être imprimées pour compléter le montage :

- **SPECB - monochromator - Part 1** (2 exemplaires)
- **SPECB - monochromator - Part 2** (2 exemplaires)
- **SPECB - central unit IN - Part 1** (2 exemplaires)
- **SPECB - central unit IN - Part 2** (2 exemplaires)
- **SPECB - central unit OUT - Part 1** (1 exemplaire)
- **SPECB - central unit OUT - Part 2** (1 exemplaire)

1.4.2 Démontage des monochromateurs

Déconnexion des câbles

Débrancher tous les câbles coaxiaux et autres connectiques du spectrofluorimètre.

Séparation des boîtiers

Desserrer les vis situées sur la partie supérieure afin de séparer les différents compartiments (monochromateurs et chambre principale).

Ouverture des monochromateurs

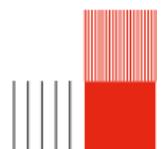
Retirer toutes les vis du couvercle supérieur pour ouvrir les monochromateurs.

Avant toute manipulation interne, il est recommandé :

- de retirer les réseaux de diffraction pour éviter tout dommage (à l'aide de la vis située à l'arrière du réseau),
- de protéger les miroirs à l'aide de petits sachets plastiques.



FIGURE 1.12 – Photo du monochromateur ouvert



Retrait de l'ancien PCB

Commencer par enlever la paroi verticale située devant le PCB à l'aide d'une clé Allen 7/64 pour retirer les deux vis situées au fond du boîtier.



FIGURE 1.13 – Photo de la paroi à enlever

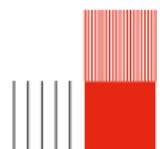
Ensuite, retirer le PCB en utilisant un tournevis cruciforme ainsi qu'un cliquet avec douilles. Débrancher ensuite tous les câbles en tirant délicatement dessus.



FIGURE 1.14 – Photo du PCB

Retrait du port de connexion

Le port de connexion situé sous l'une des fentes doit également être retiré.



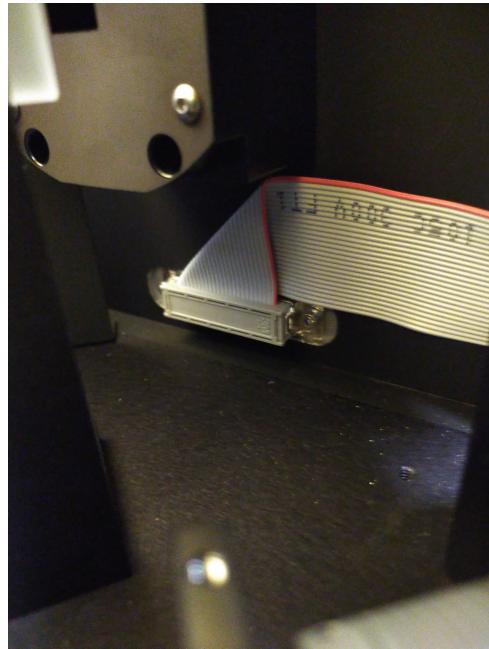


FIGURE 1.15 – Photo du port de connexion

Voici les étapes à suivre :

- Enlever le mur en forme de V situé devant le port à l'aide d'une clé Allen 7/64.
- Utiliser ensuite une clé Allen 3/32 pour retirer les deux vis maintenant le port de connexion en place.

Cette opération peut être délicate à cause de l'espace réduit. Protéger les miroirs à proximité si besoin.

Éléments à conserver

Parmi les éléments démontés, conservez :

- les vis du port de connexion,
- les deux parois verticales et leurs vis.

Le reste, notamment le PCB et le port de connexion, ne sera plus utilisé.

1.4.3 Montage des monochromateurs

Branchement de l'Arduino et du PCB

Commencer par identifier les différents câbles, puis les connecter comme suit :

- Trois moteurs pas à pas contrôlant les fentes (6 fils chacun), accompagnés de leurs capteurs optiques (3 ou 5 fils).
- Un moteur pas à pas pour l'orientation des réseaux de diffraction, avec deux capteurs optiques :
 - Un capteur de mise à zéro (situé sous la plateforme d'un des réseaux).
 - Un capteur de précision pour améliorer le positionnement du moteur.
- Un solénoïde pour le contrôle du shutter (si présent).

Branchement du moteur d'orientation :

- Connecter les deux capteurs optiques au PCB.
- Relier les sorties de signal du PCB au shield Arduino :
 - Le capteur de mise à zéro sur la broche **RESET/ABORT**.
 - Le capteur de précision sur la broche **FEED/HOLD**.
- Connecter le moteur pas à pas au driver **X** du shield Arduino, en utilisant le PCB (conversion 6 fils → 4 fils) ou en sertissant manuellement les câbles.



Branchement des moteurs de fentes :

- Connecter chaque capteur optique au PCB.
- Relier les signaux comme suit :
 - Fente 1 (au-dessus du port de connexion) → **END STOP Z**.
 - Fente 2 (au centre du boîtier) → **END STOP Y**.
 - Fente 3 (côté opposé au port de connexion) → **RESUME**.
- Connecter les moteurs aux drivers du shield Arduino :
 - Fente 1 → driver **A**,
 - Fente 2 → driver **Z**,
 - Fente 3 → driver **Y**.

Utiliser les connecteurs du PCB ou sertir les câbles si nécessaire.

Branchement du shutter (si présent) :

- Connecter le solénoïde aux bornes prévues sur le PCB.
- Relier le signal de contrôle du PCB à la broche **END STOP X** du shield Arduino.

Enfin, connecter les lignes **GND**, **9 V ou 12 V**, et **5 V** entre le shield Arduino et le PCB.

Une fois les connexions terminées, replacer la paroi verticale. L'Arduino et le PCB peuvent être insérés dans l'espace disponible. Si besoin, utilisez les trous filetés de l'ancienne PCB pour les fixer solidement.

Connecteur de puissance

Soudez un connecteur DC *barrel jack* à deux câbles longs, afin de le placer à l'emplacement de l'ancien port de connexion. Connectez l'autre extrémité au PCB (sertissage ou soudure selon le besoin).

Nouveau port de connexion

Faites passer le câble Arduino et celui de l'alimentation DC dans les pièces 3D **SPECB - mono-chromator - Part 1** et **Part 2**, puis fixez-les à l'aide des vis de l'ancien port de connexion.

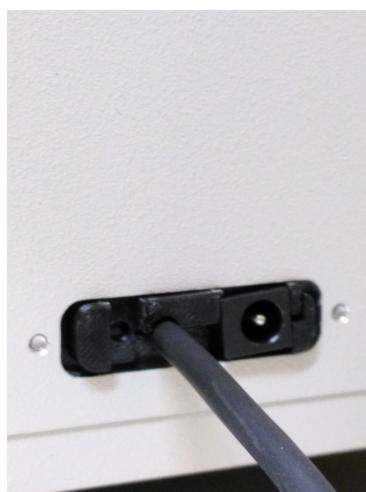


FIGURE 1.16 – Photo du nouveau port de connexion

Ensuite, repositionnez la paroi située devant le port de connexion.

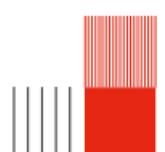




FIGURE 1.17 – Photo de l'intérieur du monochromateur monté

1.4.4 Démontage du boîtier central

Retirer les différentes vis sur la partie supérieure du boîtier, puis l'ouvrir délicatement. Des câbles étant collés sur la paroi supérieure, il est recommandé de les glisser légèrement sans les déconnecter ni les couper.

Dévisser le PCB principal ainsi que le port de connexion qui y est soudé.

Retirer ensuite :

- le PCB,
- les ports de connexion situés sous les deux fentes.

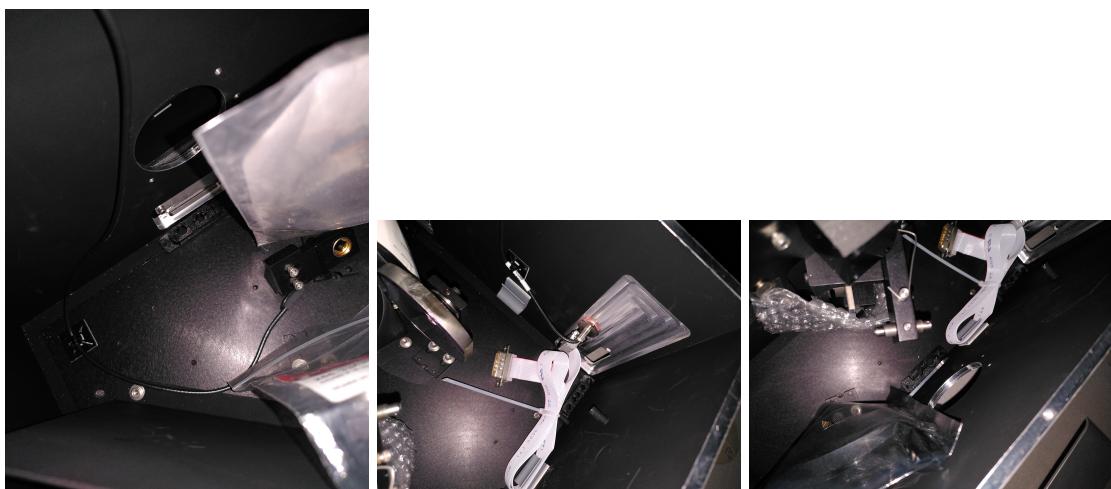


FIGURE 1.18 – Port sous la première fente

FIGURE 1.19 – Port vers ordinateur de contrôle

FIGURE 1.20 – Port sous la deuxième fente

1.4.5 Montage du boîtier central

Une fois le montage des deux monochromateurs terminé, faire passer les câbles Arduino ainsi que les câbles d'alimentation à travers les pièces 3D nommées **SPECB - central unit IN/OUT - Part 1 et Part 2**.

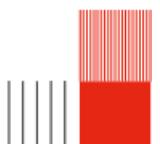
Fixer ces pièces à l'emplacement des anciens ports de connexion. Cela permet :

- de relier les câbles Arduino à la Raspberry Pi,
- d'alimenter chaque monochromateur via les prises électriques.

1.4.6 Configuration et étalonnage

Afin de faire fonctionner correctement le spectrofluorimètre, deux étapes de configuration sont nécessaires :

- Définir le sens de rotation des moteurs.
- Étalonner les monochromateurs.



Configuration des moteurs

La configuration des moteurs s'effectue par communication série avec l'Arduino Uno. Pour cela, utiliser l'IDE Arduino depuis un ordinateur ou depuis la Raspberry Pi (l'IDE n'est pas installé par défaut).

Les commandes suivantes permettent de tester les moteurs :

- `MOVE,<Nom Du Moteur>, <Nombre de pas>, <Sens de rotation (0 ou 1)>`
- Les noms de moteurs disponibles sont :
 - WL : moteur d'orientation des réseaux de diffraction,
 - SLIT1 : moteur de la fente au-dessus du port de connexion,
 - SLIT2 : moteur de la fente centrale,
 - SLIT3 : moteur de la fente opposée au port de connexion.

Comme les moteurs ont été branchés sans contrainte d'orientation, il faut configurer leur sens de rotation manuellement dans le code Arduino :

- Ouvrir le fichier `config.h` (ligne 2 du fichier principal). Sélectionner le fichier correspondant au type de monochromateur (A ou B).
- Dans la structure `motors[]`, ajuster la vitesse de chaque moteur. Cette vitesse est définie par le temps d'attente entre deux pas :
 - Pour augmenter la vitesse : diminuer le temps d'attente.
 - Pour diminuer la vitesse : augmenter ce temps.

Choisir des vitesses assurant une rotation fluide, sans à-coups.

Cette étape est optionnelle si vos moteurs se comportent exactement comme ceux utilisés dans le projet.

- Modifier l'avant-dernière variable de chaque moteur : `zeroDirection`, à `LOW` ou `HIGH`, de manière à ce que :
 - la longueur d'onde diminue (pour le moteur WL),
 - la fente se ferme (pour les moteurs SLIT) lorsque la commande `MOVE` est utilisée avec le sens de rotation 0.

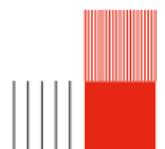
Une fois ces réglages effectués, téléverser le code sur l'Arduino Uno.

Configuration série et étalonnage

Ouvrir le programme Python, puis localiser le fichier YAML correspondant au spectrofluorimètre B dans le dossier `SYSTEM_CONFIG`.

Effectuer les réglages suivants :

- Modifier le port de connexion de chaque monochromateur selon l'identifiant de votre système : `/dev/ttyUSB<n° du port>` (généralement entre 1 et 4).
- Déterminer les coefficients de la relation entre le nombre de pas et la longueur d'onde :
 - Pour les monochromateurs de type A :
relation linéaire : $\lambda = \lambda_0 + \alpha \cdot k_{\text{pas}}$.
 - Pour les monochromateurs de type B :
relation non linéaire : $\lambda = \lambda_0 + \alpha \cdot \sin(\beta \cdot k_{\text{pas}})$.
- Pour les monochromateurs de type B, déterminer également la relation entre la **résolution spectrale** (en longueur d'onde) et le pas moteur. Cette relation est linéaire, donc deux coefficients suffisent.
- Enfin, ajuster expérimentalement les valeurs de pas `minimum` et `maximum` pour chaque moteur. Cela permet de :
 - garantir la sécurité mécanique,
 - limiter les longueurs d'onde aux plages utiles,
 - obtenir une bonne précision de balayage.



Chapitre 2

Documentation du programme de contrôle des spectrofluorimètres

Le fonctionnement du spectrofluorimètre repose sur une interaction continue entre un code Arduino et un programme Python. Le code Arduino tourne sur une carte Uno et assure le contrôle bas niveau des moteurs pas à pas via un Shield CNC. De son côté, le script Python s'exécute sur une Raspberry Pi et joue un rôle central de coordination : c'est lui qui envoie les ordres, récupère les données, et orchestre toute la séquence de mesure.

La communication entre les deux se fait par port série, avec un protocole simple. Le Python envoie des commandes sous forme de texte (par exemple pour déplacer un moteur ou initialiser une fente), et l'Arduino répond une fois l'opération terminée.

Le fichier `main.py` lit un fichier de configuration YAML qui décrit les mesures à réaliser avec leurs longueurs d'onde à scanner, les temps d'intégration et autre. Ensuite, il initialise les composants nécessaires (monochromateurs, détecteurs, fentes) et lance une boucle dans laquelle il déplace les moteurs, effectue les mesures, puis enregistre les résultats.

Le code Arduino (`main.ino`) lit les commandes série, les découpe, puis agit selon la demande. Il gère les mouvements des moteurs, la lecture des capteurs de position, et renvoie un accusé de réception à Python une fois l'action réalisée. C'est un système synchrone.

L'ensemble fonctionne comme un système maître-esclave : Python donne les ordres, Arduino exécute. C'est cette séparation des responsabilités qui rend l'ensemble modulaire et adaptable à différents types de spectrofluorimètres.

2.1 Programme Arduino

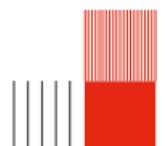
Setup

La fonction `setup()` initialise le système. Elle configure les broches des moteurs et des capteurs de fin de course, et établit la communication série à 115200 bauds avec le Raspberry Pi.

Elle appelle d'abord la fonction `initMotorPins()` pour préparer les broches `direction`, `step` et `enable` des moteurs, puis désactive les moteurs (`enable = HIGH`). Ensuite, elle configure les pins des capteurs comme entrées (`INPUT_PULLUP`), de plus configure aussi la pin pour le shutter.

Loop

La fonction `loop()` est une boucle infinie qui surveille en permanence s'il y a des données arrivant par le port série. Si une ligne de commande est reçue (terminée par un retour à la ligne), elle est stockée dans `line` et analysée.



Elle est alors transmise à la fonction `handleCommand()` qui va interpréter l'ordre, exécuter la commande et renvoyer un accusé de réception (`Serial.println()`).

findZero

Cette fonction est responsable de la mise à zéro d'un moteur. Elle sert à aligner mécaniquement le moteur avec une position de référence fixée par un capteur.

Lorsqu'elle est appelée, la fonction active la direction définie dans `motor.zeroDirection`, puis lance le moteur pas à pas tant que le capteur n'est pas encore déclenché. L'état du capteur est vérifié avec un `digitalRead()` sur la pin `motor.limitSwitchPin`. Le moteur continue de tourner tant que le capteur ne retourne pas la valeur attendue, définie dans `motor.zeroPinState`.

Un timeout de sécurité est mis en place : si la position zéro n'est pas atteinte après un certain temps (défini dans `ZERO_TIMEOUT_MS`), la fonction s'interrompt et écrit `ZERO_TIMEOUT` sur le port série. Cela évite de bloquer le moteur indéfiniment en cas de capteur défectueux ou mal câblé. Sinon, si tout se passe bien, elle affiche `ZERO,DONE`.

moveStepper

Cette fonction permet de faire tourner un moteur d'un nombre de pas donné, dans une direction précise, à une vitesse choisie (rapide ou lente).

La direction est d'abord envoyée à la pin `motor.dirPin` via `digitalWrite()`. Puis, une boucle `for` est lancée sur le nombre de pas souhaité. À chaque itération, un front montant puis descendant est appliqué sur la pin `motor.stepPin`, avec un `delayMicroseconds()` pour espacer les impulsions. Le temps d'attente est déterminé par la vitesse sélectionnée : `motor.slow_stepSpeed` ou `motor.fast_stepSpeed`.

La fonction ne vérifie pas l'état du capteur de fin de course durant le mouvement — elle s'exécute de façon purement aveugle. Ce comportement est volontaire : la sécurité repose sur le bon étalonnage initial et l'usage de limites logicielles côté Python. Une fois le déplacement terminé, elle écrit `MOVE,DONE` sur le port série pour informer Python que l'action est complétée.

handleCommand

Cette fonction est au centre de la logique de la boucle `loop()`. Cette fonction analyse les commandes reçues via la liaison série, identifie l'action à exécuter, et appelle la fonction correspondante (`moveStepper()` ou `findZero()`).

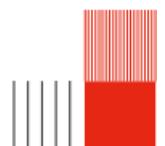
La commande est découpée avec la méthode `command.substring()` pour extraire les champs séparés par des virgules. Le premier champ indique l'action à effectuer (`MOVE`, `ZERO`, `SHUTTER` ou autre). Ensuite, la fonction recherche dans le tableau `motors[]` le moteur dont le nom correspond à celui fourni dans la commande.

Pour un `MOVE`, la fonction lit le nombre de pas et la direction (0 ou 1), puis appelle `moveStepper()` avec ces paramètres. Pour un `ZERO`, elle appelle simplement `findZero()` sur le moteur sélectionné. Si le nom du moteur est inconnu ou la commande mal formée, aucune action n'est entreprise et un message d'erreur est envoyé au programme python.

2.2 Programme Python

2.2.1 main.py

Ce script contient la logique principale du code python et est chargé de la communication entre le système et l'utilisateur.



createConfiguredMonochromators

C'est ici que l'on crée les objets représentant les monochromateurs d'excitation et d'émission. On lit le fichier de configuration YAML, on détecte pour chaque monochromateur s'il s'agit d'un modèle de type A ou B, puis on crée un objet MonochromatorA ou MonochromatorB en conséquence.

Pour un type A, l'objet est instancié avec uniquement la relation longueur d'onde - pas, le port série et les limites de pas. Pour un type B, en plus des paramètres précédents, on doit créer un objet Slit (fente) avec ses propres paramètres. Chaque objet ainsi créé est ensuite stocké dans un dictionnaire avec pour clé "excitation monochromator" ou "émission monochromator".

createMeasurements

Une fois les monochromateurs instanciés, cette fonction génère les objets de mesure. On lit ici un second fichier YAML (dans le dossier Measure Config) qui décrit les mesures à effectuer : type (émission, excitation, synchrone ou unique), temps d'intégration, résolution et plages de longueurs d'onde.

Chaque type de mesure est associé à une classe différente (EmScanMeasure, ExScanMeasure, etc.), et les objets sont instanciés avec les bons monochromateurs, les longueurs d'onde, et la configuration de balayage. Le dictionnaire retourné contient tous les scans à exécuter.

interactiveRun()

C'est la fonction principale du script, exécutée automatiquement à la fin. Voici son déroulement logique :

1. Lecture de la configuration des monochromateurs depuis le YAML config système.
2. Crédit des objets Monochromator via createConfiguredMonochromators.
3. Proposition à l'utilisateur d'initialiser les moteurs (via .initMotors()).
4. Affichage des fichiers YAML de configuration de mesure disponibles dans le dossier.
5. Sélection du fichier de mesure.
6. Crédit des objets Measure via createMeasurements.
7. Affichage d'informations utiles : longueurs d'onde min/max, plage de résolution.
8. Confirmation utilisateur pour lancer ou annuler la mesure.
9. Exécution des mesures : pour chaque scan, appel à measure.measure() puis sauvegarde des résultats avec measure.saveResults().

2.2.2 Classe Wavelength

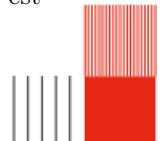
L'objet WaveLength représente une valeur physique de longueur d'onde exprimée en nanomètres (nm). Ce choix d'encapsulation permet une abstraction propre : on ne manipule jamais directement des float, mais des objets dédiés, ce qui ouvre la voie à des conversions et opérations intégrées sans alourdir le code principal.

Lors de son initialisation, on fournit une valeur numérique unique, stockée dans l'attribut self.value. Toutes les opérations numériques ou conversions sont alors basées sur cette valeur.

Conversions

Deux méthodes sont déjà disponibles pour convertir la longueur d'onde :

- to_waveNumber() transforme la longueur d'onde en nombre d'ondes via la formule : $\nu = \frac{10^4}{\lambda}$ avec λ en nm et le résultat en cm^{-1} . Elle intègre un contrôle d'erreur si la longueur d'onde est nulle.



- `to_frequency()` convertit en fréquence (Hz) via la relation : $f = \frac{c}{\lambda}$
où la vitesse de la lumière c est prise comme constante en m/s. Ici aussi, un `ValueError` est levé si $\lambda = 0$ pour éviter toute division par zéro.

Même si ces fonctionnalités ne sont pas encore exploitées dans le code principal, elles permettent déjà de visualiser ou stocker les données dans d'autres unités sans avoir à dupliquer de logique ailleurs.

2.2.3 Classe WLRange

La classe `WLRange` regroupe un intervalle de longueurs d'onde : minimum, maximum, et pas (`wLStep`). Tous trois sont des objets `WaveLength`.

Cette classe permet de décrire proprement les gammes de balayage utilisées dans les mesures (émission, excitation, etc.), en évitant les erreurs de type ou d'unité. Elle vérifie aussi que `wLMin` est bien inférieur à `wLMax`, ce qui protège des erreurs de configuration en amont.

2.2.4 Classe IntegrationTime

La classe `IntegrationTime` encapsule simplement un temps d'intégration, en millisecondes. Elle sert à garantir que les valeurs utilisées soient toujours positives et raisonnables (entre 0 et 1 000 000 ms), en centralisant les contrôles dans un objet dédié.

Elle offre une méthode `to_seconds()` pour convertir en secondes pour l'affichage et pour les fonctions de temporisation.

2.2.5 Classe Slit

La classe `Slit`, utilisée exclusivement dans les monochromateurs de type B, représente la fente réglable électroniquement en termes de résolution spectrale. Elle encapsule à la fois la conversion entre pas moteur et largeur de fente (exprimée ici en longueur d'onde effective), le dialogue série avec l'Arduino, et les contrôles d'intégrité liés aux déplacements. Voici une explication détaillée de ses attributs et méthodes.

Instanciation

Le constructeur `__init__()` initialise les caractéristiques d'une ou plusieurs fentes. Il prend notamment un `numberOfSlits` (3), le nombre de fentes dans le monochromateur, une valeur de longueur d'onde courante (objet `WaveLength`), un décalage (`offsetWL`) et un coefficient pour la conversion pas - valeur physique (`coefWL`). On peut également fournir une position actuelle (`step`) et les bornes (`minStep`, `maxStep`). L'objet série `serialConnection` représente le lien de communication avec le microcontrôleur. Une fois l'objet créé, la méthode `updateWaveLength()` permet de calculer la valeur actuelle de la fente à partir du pas courant.

Conversion pas-longueur d'onde

Les méthodes `getValueFromStep()` et `getStepFromValue()` sont des fonctions de conversion. La première convertit un nombre de pas en une longueur d'onde équivalente (selon la calibration spécifique à chaque fente), tandis que la seconde fait l'opération inverse. Ces méthodes permettent une grande souplesse de manipulation : on peut raisonner en longueur d'onde plutôt qu'en position moteur, tout en gardant un pilotage précis en pas moteur en arrière-plan.

Trouver la position zéro

La méthode `findZero()` permet de retrouver la position zéro des fentes, ce qui correspond à une calibration physique du système. Pour chaque fente (généralement trois), la méthode envoie une commande série de type `ZERO,SLITx` au microcontrôleur, puis attend une réponse. Si la réponse est "ZERO,DONE", le pas est mis à 0 et la valeur réinitialisée à `offsetWL`. En cas d'échec ou de dépassement de temps, une



erreur est levée. Ce processus est indispensable pour garantir une référence stable avant tout déplacement.

Déplacement

Les méthodes `moveToValue()` et `moveToPercentage()` permettent de déplacer les fentes à une position donnée, soit en longueur d'onde, soit en pourcentage de l'ouverture maximale. Dans les deux cas, on calcule d'abord le nombre de pas à effectuer, la direction du déplacement, puis on envoie la commande `MOVE,SLITx,nombre_de_pas,direction` à chaque fente. On attend ensuite une réponse "`MOVE,DONE`" pour valider le mouvement. Une fois le déplacement terminé, la position (step) et la valeur associée (WaveLength) sont mises à jour.

2.2.6 Classe Monochromator

Les classes `Monochromator`, `MonochromatorA` et `MonochromatorB` modélisent le comportement de deux types distincts de monochromateurs contrôlés via port série. Ces objets pilotent les déplacements moteurs pour sélectionner des longueurs d'onde précises, et dans le cas du type B, gèrent également les fentes motorisées. Toute la structure repose sur une classe mère abstraite (`Monochromator`) qui formalise l'interface commune, ensuite spécialisée dans `MonochromatorA` et `MonochromatorB`.

La classe `MonochromatorA` hérite de la classe mère et correspond à un modèle de monochromateur où la relation entre le nombre de pas et la longueur d'onde est linéaire. Elle implémente donc les deux méthodes de conversion en utilisant une simple équation affine : $\lambda = \lambda_0 + \alpha \cdot k_{pas}$ pour aller du pas à la longueur d'onde, et l'inverse pour la commande. Elle redéfinit aussi la fonction `initMotors()`, qui attend le signal d'initialisation du microcontrôleur avant de lancer le `findZero()`.

La classe `MonochromatorB` représente quant à elle les modèles plus complexes, notamment les Fluorolog, où la conversion entre pas et longueur d'onde est non-linéaire. Ici, la relation est de la forme $\lambda = \lambda_0 + \alpha \cdot \sin(\beta \cdot k_{pas})$ et l'inverse utilise `asin()`, ce qui reflète la géométrie du mécanisme optique. En plus des méthodes classiques héritées de la classe mère, elle prend en charge un objet `Slit` représentant les trois fentes motorisées. C'est lors de la création de l'objet que la connexion série est transmise à l'objet `Slit`, assurant un canal de communication unique pour l'ensemble du module. Sa méthode `initMotors()` appelle donc à la fois `findZero()` pour le moteur principal et `findZero()` pour les fentes. Elle peut aussi modifier dynamiquement la résolution via la méthode `setResolution()`, qui agit directement sur les fentes.

2.2.7 Classe Signal

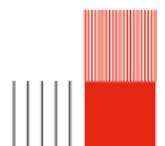
La classe `Signal`, définie dans le fichier `spectro_signal.py`, représente une courbe de mesure. Elle encapsule à la fois un nom pour identifier la nature du signal (par exemple "émission", "référence", etc.) et une liste de valeurs numériques représentant l'intensité mesurée à différentes longueurs d'onde.

Si aucune donnée n'est fournie à la création, la liste est simplement vide. Cela permet de construire progressivement un signal, soit en l'alimentant au fur et à mesure des acquisitions, soit en important directement une liste existante.

La classe offre plusieurs méthodes pour enrichir ou modifier le signal. `add_signal()` permet d'ajouter une liste entière de valeurs à la suite des données existantes. `append_signal()` ajoute une valeur unique, ce qui est utile pour une construction point par point lors d'un balayage. `clear_signal()` réinitialise complètement le signal.

2.2.8 Classe Measure

Les classes définies dans le fichier `measure.py` forment le cœur de la logique de mesure du spectrofluorimètre. L'architecture est pensée de manière modulaire autour d'une classe de base abstraite `Measure`, qui définit les fondations communes à toutes les mesures possibles, puis étendue par plusieurs sous-classes qui implémentent chacune un mode de mesure spécifique.

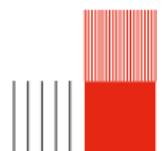


La classe Measure n'est pas destinée à être utilisée directement. Elle encapsule les éléments communs à toute mesure : les plages de longueurs d'onde (excitatrice et émissive), le temps d'intégration, les deux monochromateurs utilisés (un pour l'excitation, l'autre pour l'émission), et enfin les signaux mesurés et de référence. Le constructeur valide notamment que les deux monochromateurs ont bien les rôles attendus (ex et em) pour éviter des erreurs de configuration. Quelques méthodes abstraites (measure, plot, saveResults) obligent les sous-classes à fournir leur propre implémentation de la logique.

La classe uniqueWLMeasure représente une mesure à deux longueurs d'onde fixes (excitatrice et émissive). Lors de l'appel à la méthode measure, chaque monochromateur se déplace vers sa longueur d'onde cible, puis les shutters s'ouvrent pour permettre la mesure du signal et de la référence. Une fois cela effectué, les shutters se ferment. La méthode saveResults enregistre les valeurs obtenues dans un fichier CSV, incluant date, heure, et noms des longueurs d'onde.

Les autres sous-classes étendent la logique à des balayages spectroscopiques. synchroScanMeasure permet de déplacer simultanément les deux longueurs d'onde avec un décalage constant entre excitation et émission, ce qui est typiquement utilisé pour suivre des comportements dépendants du déplacement spectral. ExScanMeasure fixe l'émission et balaye l'excitation, tandis que EmScanMeasure fait l'inverse. Ces classes déclenchent les mesures, et enregistrent les résultats de manière similaire à uniqueWLMeasure.

Cette structure permet d'étendre facilement le système à de nouveaux types de mesures en dérivant Measure sans devoir réécrire la logique de bas niveau comme l'ouverture des shutters ou la gestion des signaux.



Chapitre 3

Guide utilisateur

Ce guide vise à fournir toutes les informations nécessaires pour effectuer des mesures. Tous les éléments requis sont regroupés dans le dossier nommé **Spectrofluorimetre**.

Pour démarrer, exécutez le fichier **main.py** après avoir créé et placé le fichier de configuration YAML dans le dossier **Measure Config**. Un fichier d'exemple est fourni afin de facilement remplir les paramètres de la mesure qui doit être réalisée.

Une fois la mesure effectuée, les résultats sont enregistrés dans un fichier CSV situé dans le dossier **Measure CSV**. Enfin, le dossier **SYSTEM_CONFIG** contient les paramètres de configuration du spectrofluorimètre.

3.1 Faire une mesure

3.1.1 Fichier de configuration des mesures

Créer un document .YAML, pour cela copier directement le fichier exemple **scan_exampleFile.yml** ou créer un document .txt et modifier l'extension en .yml.

Le fichier YAML permet de décrire de manière simple et lisible tous les paramètres nécessaires à la mesure sans avoir à modifier directement le code Python. Il suffit de remplir correctement ce fichier pour effectuer une acquisition.

La première clé à renseigner est "measureType". Elle détermine le type de balayage à effectuer, et elle accepte quatre valeurs possibles :

- "singular" : pour mesurer à une seule longueur d'onde d'excitation et d'émission.
- "emission" : pour effectuer un balayage de la longueur d'onde d'émission en maintenant l'excitation fixe.
- "excitation" : pour balayer l'excitation tout en gardant l'émission constante.
- "synchronous" : pour balayer simultanément l'émission et l'excitation avec un décalage constant entre les deux.

Les éléments suivants sont partagés pour toutes les mesures : la section "parameters" contient les réglages essentiels pour la mesure. Le champ "integrationTime" spécifie la durée, en millisecondes, pendant laquelle le signal sera accumulé par le capteur (ex. integrationTime : 1000 pour 1 seconde). Le champ "name" permet d'indiquer le nom du fichier CSV dans lequel le résultat sera enregistré. Le champ "resolution" permet d'indiquer la résolution souhaitée.

Singular

La mesure de type "singular" permet d'effectuer une acquisition ponctuelle, à une longueur d'onde d'excitation et une autre d'émission bien définies. C'est le mode de mesure le plus simple, utilisé lorsque l'on veut obtenir rapidement une seule valeur d'intensité, sans balayage spectral. Ce mode est idéal pour



calibrer ou tester le système.

Dans le fichier YAML, le champ "type" doit être défini à "singular" pour activer ce mode. Ensuite utiliser les champs "excitation wavelength" et "excitation wavelength" pour définir les longueur d'onde de la mesure.

Emission scan

Dans le fichier YAML, le champ "type" doit être défini à "emission" pour activer ce mode. Ensuite, utiliser les champs "excitation wavelength" et "scan range" avec "start", "end" et "step" pour définir les longueur d'onde de la mesure.

Excitation scan

Dans le fichier YAML, le champ "type" doit être défini à "excitation" pour activer ce mode. Ensuite, utiliser les champs "emission wavelength" et "scan range" avec "start", "end" et "step" pour définir les longueur d'onde de la mesure.

Synchronous scan

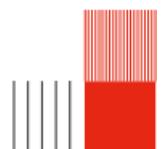
Dans le fichier YAML, le champ "type" doit être défini à "synchronous" pour activer ce mode. Ensuite, utiliser les champs "offset wavelength" et "scan range" avec "start", "end" et "step" pour définir les longueur d'onde de la mesure.

Ainsi, la plage parcouru d'émission sera la plage de l'excitation au quel on ajoute la valeur d'offset.

3.1.2 Lancer le programme

Pour lancer le programme, il suffit d'exécuter le programme main.py. Pour cela, ouvrir le fichier et appuyer sur le bouton run dans l'éditeur python.

Par la suite, il suffit de suivre les instructions données dans l'invite de commande.



Bibliographie

- [1] Murty Mantravadi. Theory and principles of monochromators, spectrometers and spectrographs. *Optical Engineering*, 13(1) :23–29, 1974.

