INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

Institut d'Optique Graduate School
Interfaçage Numérique

INTERFAÇAGE NUMÉRIQUE

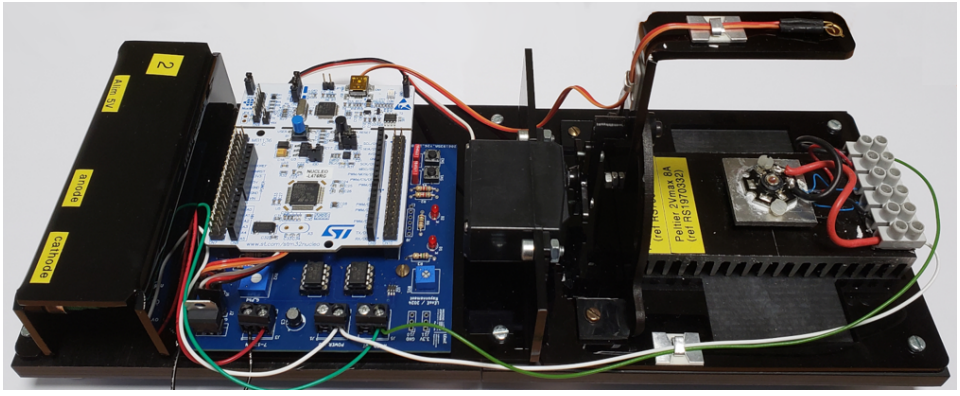
Travaux Pratiques

Semestre 6

Banc de mesure de rayonnement

---

4 séances



Ce sujet est disponible au format électronique sur le site du LEnsE - https ://lense.institutoptique.fr/ dans la rubrique Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Banc de mesure de rayonnement lumineux.

Banc de mesure de rayonnement

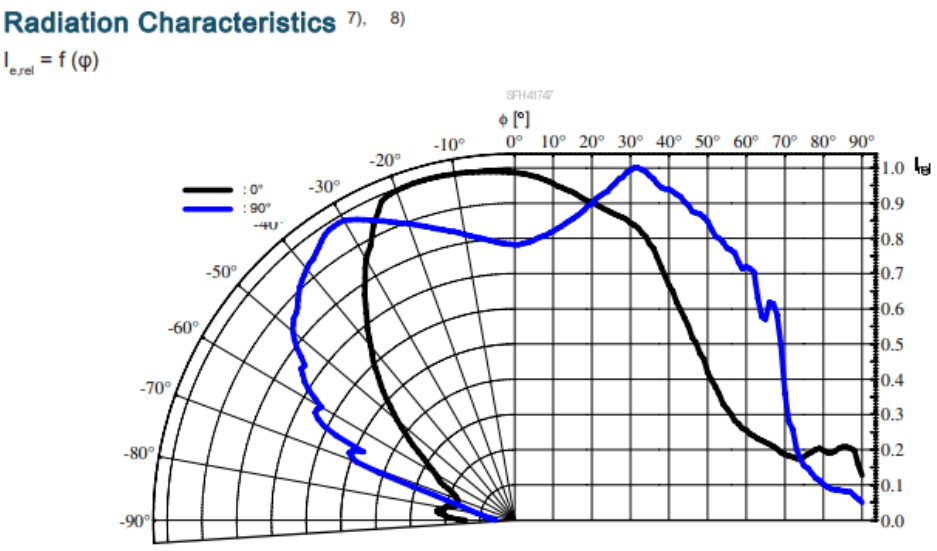
À l'issue des séances de TP concernant le **bloc d'acquisition d'un diagramme de rayonnement**, les étudiant·es seront capables de :

- Développer et mettre en œuvre une **solution d'électronique embarquée** pour **acquérir des données analogiques** et commander un élément mobile.
- Mettre en œuvre un **protocole simple de communication** entre un ordinateur et un microcontrôleur pour transmettre des commandes et lire des données
- Optimiser une **interface informatique** de pilotage et d'affichage de données

Objectifs du mini-projet

L'objectif principal de ce mini-projet est d'**automatiser un banc de mesure de rayonnement lumineux** à l'aide d'un ordinateur et d'une interface en Python. Le matériel est piloté par une carte à microcontroleur à laquelle il faudra envoyer des commandes selon un protocole série.

Un diagramme de rayonnement lumineux est la **représentation graphique** de la **distribution angulaire** d'une grandeur caractérisant le rayonnement d'une source lumineuse.



Exemple de diagramme de rayonnement d'une LED OSRAM SFH 41747 (documentation technique)

Vous aurez à votre disposition une **maquette** permettant la mise en mouvement d'un système de photodétection autour d'une source de puissance à LED. Cette maquette est pilotée par une carte Nucléo (contenant un microcontroleur).

Déroulement du bloc

La liste des étapes à suivre pour la réalisation du programme embarqué de la plateforme de rayonnement lumineux est donnée à titre indicatif. L'ordre et le choix des différentes étapes sont laissés à l'appréciation des différents binômes.

Afin de faciliter la réutilisation des codes, il pourra être intéressant de définir des fonctions pour le pilotage des différents éléments.

Séance 1 / Arduino et Nucléo-STM32 (sans maquette ! !)

Le sujet de cette séance est fourni dans un document annexe, disponible aussi sur le site du LEnsE - https ://-lense.institutoptique.fr/ dans la rubrique Année / Première Année / Interfaçage Numérique S6 / Bloc 1 Systèmes embarqués / Intro Arduino et STM32.

Etape 0 - 30 min Installer les drivers STM32 et tester un premier programme

Etape 1 - 45 min Piloter des sorties numériques - LED

Etape 2 - 45 min Acquérir des données numériques - Bouton-poussoirs

Etape 3 - 45 min Mettre en œuvre des interruptions sur des événements externes

Etape 4 - 45 min Utiliser des sorties modulées en largeur d'impulsion (PWM) - LEDs

Etape 5 - 60 min Acquérir des données analogiques - Potentiomètre

Séance 2 / Prise en main de la maquette et automatisation de la mesure

Etape 6 - 60 min Piloter l'intensité des LEDs de la maquette

Etape 7 - 60 min Piloter le servomoteur de la maquette

Etape 8 - 90 min Définir et tester une première structure de code permettant de piloter le servomoteur en faisant une acquisition de la luminosité à pas régulier

Etape 9 - 60 min Récupérer les données à l'aide du moniteur Série du logiciel Arduino

Séance 3 / Mise en place d'un protocole de communication

Etape 10 - 90 min Mettre en place un protocole de communication basé sur une liste de commandes et intégrer à la structure du code embarqué

Etape 11 - 90 min Utiliser la bibliothèque pySerial en Python pour envoyer les commandes à la carte Arduino

Etape 12 - 90 min Tester la communication entre la carte Arduino et le script Python pour afficher les données

Séance 4 / Application complète

Cette dernière séance sera consacrée à la finalisation des différents programmes : embarqué sur la carte Arduino pour la mesure automatique et sur l'ordinateur pour le pilotage de la maquette et l'affichage des données.

Selon le temps restant, il sera possible d'intégrer les fonctions écrites en Python dans une interface graphique, dont la structure de base est fournie.

Séance 2 / Prise en main de la maquette et automatisation de la mesure

Objectifs de la séance

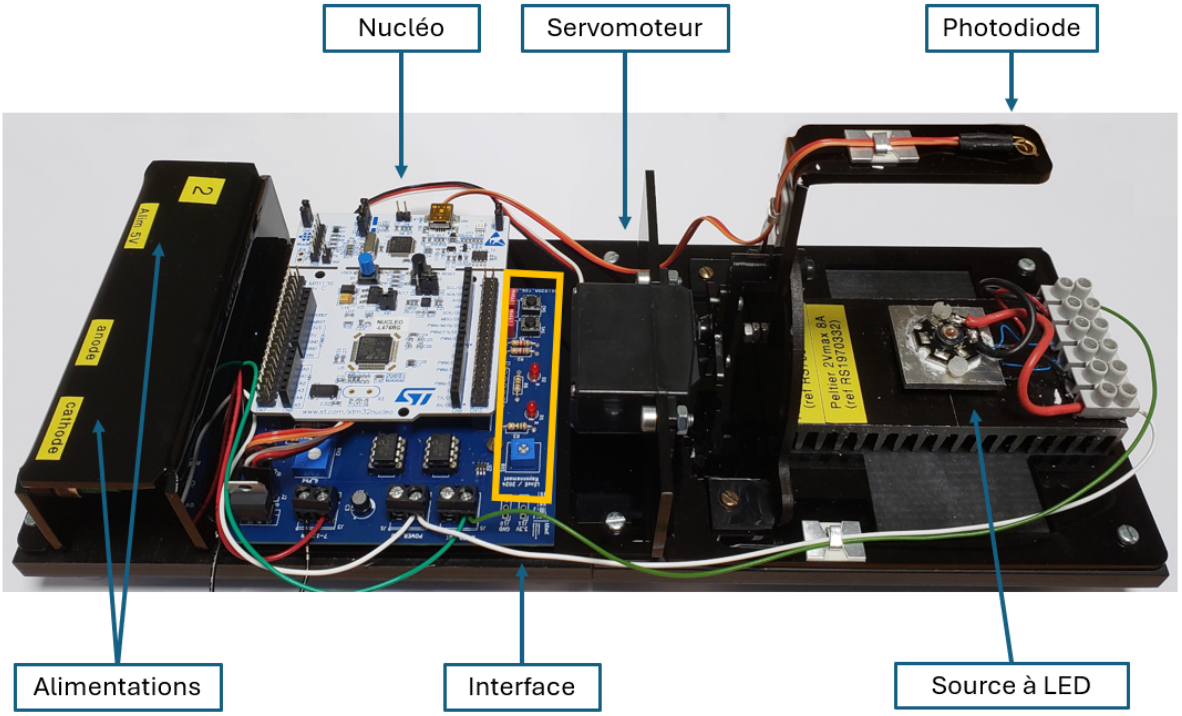
Cette seconde séance est consacrée à la **prise en main de la maquette** et à l'automatisation de la mesure de l'intensité lumineuse d'une source à LED (par exemple).

Description de la maquette

Eléments constitutifs

La maquette est constituée :

— d'une source de lumière, une LED de puissance
— d'une photodiode BPX65, associée à un circuit de photodétection simple
— d'un servomoteur, permettant de positionner le bras supportant la photodiode au dessus de la source lumineuse selon un angle donné
— d'une carte de contrôle
— d'une carte Nucléo L476RG



Alimentation électrique

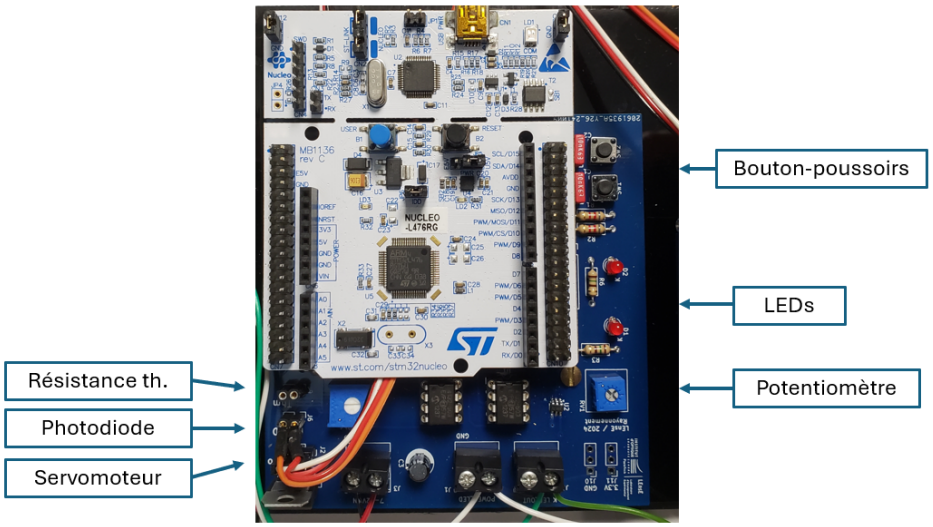
Les bornes d'alimentation notées **Alim** $5\,\mathrm{V}$ permet d'alimenter le servomoteur.
Les bornes notées **anode** et **cathode** permettent d'alimenter la LED de puissance.

La tension maximale admissible par le servomoteur est de $6\,\mathrm{V}$!
Le courant maximal admissible par la LED de puissance est de $200\,\mathrm{mA}$!

Brochage

La **photodiode BPX65** est polarisée. L'**anode** est représentée par un ergot. Sur cette maquette, l'**anode** est reliée par un fil **ORANGE**, la **cathode** par un file **ROUGE**.

Le servomoteur est standard. Sa connectique est constituée de 3 broches : BLANC = signal de pilotage, ROUGE = Alimentation de 5 à 6V, NOIR = masse (GND).



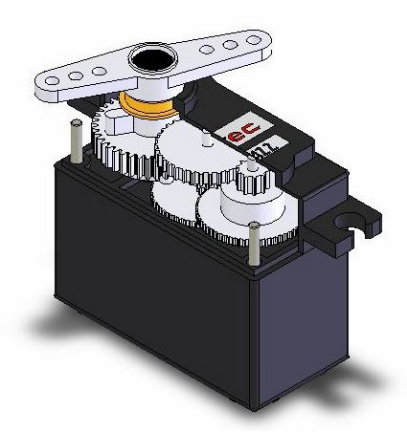
Maquette	**Broche Nucléo**	Type	Description
LED1	PC7	Sortie / PWM	Led active à l'**état haut**
LED2	PB13	Sortie / PWM	Led active à l'**état bas**
SW1	PC6	Entrée	Bouton-poussoir, par défaut état bas
SW2	PC8	Entrée	Bouton-poussoir, par défaut état bas
UserButton	PC13	Entrée	Bouton-poussoir, par défaut état haut
Servo	PB7	Sortie / PWM	Servomoteur - T = 20 ms
Photodiode	PC3	Entrée analogique	Photodiode (montage simple avec R variable)
Résistance Thermique	PC1	Entrée analogique	Résistance Thermique CT10k

Etape 6 / Piloter l'intensité des LEDs de la maquette

➜ **M** Réaliser un programme qui permet de modifier la luminosité des diodes LED1 et LED2 de la maquette à l'aide des deux bouton-poussoirs SW1 et SW2 (un pour augmenter, l'autre pour diminuer la luminosité). **Votre programme devra utiliser uniquement le principe d'interruption.**

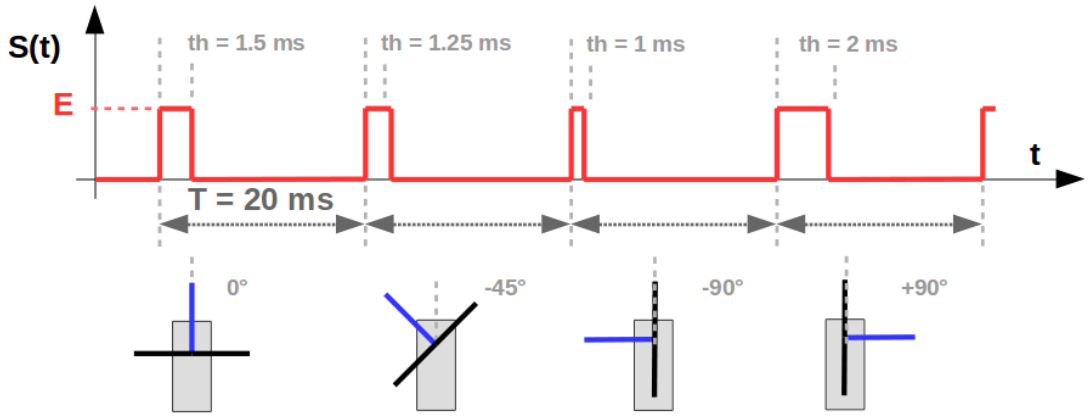
Etape 7 / Piloter le servomoteur de la maquette

Un servomoteur est un actionneur qui réalise une rotation d'un angle calibré en fonction d'une commande externe.



Servomoteur - coupe interne (Crédit : redohm.fr)

Le pilotage d'un servomoteur se fait à l'aide d'un **signal modulé en largeur d'impulsions**. Le signal de commande est un **signal rectangulaire** (numérique) de période fixée à 20 ms. C'est ensuite la durée du temps haut qui permet de modifier l'angle de sortie du servomoteur :



Bibliothèque Servo.h

Pour piloter un servomoteur, il existe une bibliothèque nommée *Servo.h*. On pourra s'inspirer du programme d'exemple pour Nucleo-64 SERVO / SWEEP.

Travail à réaliser

➜ **M** Tester le code *Sweep* fourni par Arduino, en adaptant la sortie utilisée par le servomoteur sur la maquette.

➜ **M** Créer une fonction permettant de positionner le servomoteur à un angle particulier. Tester votre fonction.

Etape 8 / Définir et tester une première structure de code

On cherche à présent à **automatiser l'acquisition des données** de luminosité en fonction de l'angle d'observation.

➜ **M** Réaliser une fonction qui, à partir d'un **angle de départ**, d'un **angle d'arrivée** et d'un **pas angulaire**, permet, pour tous les angles concernés, de :

— déplacer le servomoteur au prochain angle ;

— faire l'acquisition de la luminosité dans un tableau de données.

Le servomoteur ne réagit pas instantanément. Il est donc nécessaire d'ajouter un temps de pause entre le déplacement et l'acquisition.

➜ **M** Tester votre fonction.

Etape 9 / Récupérer les données à l'aide du moniteur Série du logiciel Arduino

Transmission des données par liaison série

Il est possible de transmettre des données par l'intermédiaire d'une **liaison dite Série** entre le microcontroleur et l'ordinateur.

C'est un protocole asynchrone de transfert de données point à point. Les échanges se font en **Full-Duplex** grâce à deux signaux distincts **RX** (réception) et **TX** (transmission).

Pour tester la liaison Série ainsi que l'affichage, on pourra utiliser le programme 01. BASICS / ANALOGREADSERIAL et l'outil **Traceur Série** (dans OUTILS / TRACEUR SÉRIE du logiciel Arduino).

➜ **M** Réaliser une fonction qui permet de transmettre par la liaison série les valeurs sauvegardées dans le tableau de données précédemment acquis.

➜ **M** Tester votre fonction.

Premier diagramme de rayonnement

➜ **M** Alimenter la LED de puissance avec un courant d'environ 50mA.

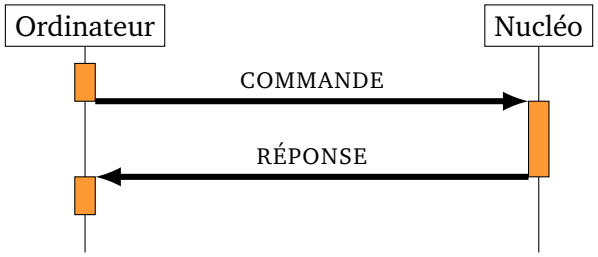
➜ **M** Faire l'acquisition des valeurs d'intensité lumineuse pour des angles allant de 0 à $180°$ par pas de $10°$ et transmettre les données par liaison Série afin d'afficher le diagramme résultant sur le traceur Série.

Séance 3 / Mise en place d'un protocole de communication

Objectifs de la séance

Cette séance a pour but de mettre en place un **protocole de communication** entre la carte embarquée et l'ordinateur, d'abord à l'aide d'un moniteur série puis à l'aide d'un script en Python.

L'ordinateur, maitre de la communication, devra envoyer des commandes à la carte embarquée, qui devra les exécuter puis acquitter du bon déroulement de l'opération.



Un exemple de code pour la transmission des commandes A et D, à analyser et à tester, est fourni.

Protocole de communication

Un **protocole de communication** est un ensemble de règles, de conventions et de formats définis pour permettre l'échange d'informations entre différents systèmes. Il détermine comment les données sont structurées, envoyées, reçues et interprétées par les parties communicantes.

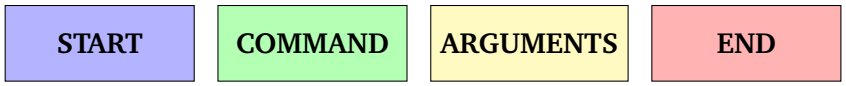
Il est important lorsqu'on souhaite **construire un protocole de communication** de :
- **choisir un protocole de bas niveau** pour la transmission des commandes et des réponses
- définir la **structure des commandes**
- planifier la **liste des commandes**
- coder la récupération des données sur la carte Nucléo (et tester avec un moniteur Série)
- coder l'envoi des commandes sur le PC (en Python par exemple)

Choix du protocole de bas niveau

Nous utiliserons dans le cas de notre application la **liaison Série** (ou RS232) qui nous a déjà servi à afficher des données depuis la carte Nucléo.

Structure des commandes

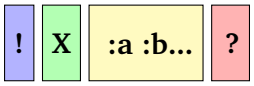
Chaque commande doit avoir un **format bien défini** pour que le microcontroleur puisse la comprendre et la traiter.



La liaison Série étant asynchrone, il est indispensable de prévoir dans le protocole des **caractères de début et de fin** de chaines de commande et de réponse.

Commande

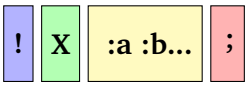
Par exemple, on pourra se baser sur la structure suivante pour le cas d'une commande (entre le PC et le microcontroleur) :



Les arguments sont facultatifs, selon les commandes.

Réponse

Par exemple, on pourra se baser sur la structure suivante pour le cas d'une réponse (entre le PC et le microcontroleur) :



Les arguments sont facultatifs, selon les commandes.

Liste des commandes

Commande PC	Description	**Réponse Nucléo**
!T ?	Test de la transmission	!T ;
!A :VALUE ?	Transmission de l'angle de départ souhaité pour le diagramme valeur entière en degré *Si angle non compris entre -90 et 90*	!A :VALUE ; VALUE= -100
!B :VALUE ?	Transmission de l'angle final souhaité pour le diagramme valeur entière en degré *Si angle non compris entre -90 et 90*	!B :VALUE ; VALUE= -100
!C :VALUE ?	Transmission du pas angulaire souhaité pour le diagramme valeur entière en degré *Si angle non compris entre -90 et 90*	!C :VALUE ; VALUE= -100
!S ?	Lancement de l'acquisition nb est le nombre d'échantillons qui seront acquis par la carte *Si les angles fournis sont non compatibles*	!S :NB ; NB= 0
!E ?	Test de fin de l'acquisition (Y)es or (N)o	!E :Y/N ;
!D :INDEX ?	Demande de récupération d'une donnée index correspond au numéro de l'échantillon souhaité value correspond à la valeur de l'échantillon souhaité	!D :INDEX :VALUE ;

Etape 10 / Mettre en place un protocole de communication basé sur une liste de commandes

Structure de base

On se propose de tester le code *test_version_a_d.ino*. Ce fichier est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Banc de mesure de rayonnement lumineux / Exemples de Codes*.

➜ **M** Compiler ce code et le téléverser sur la carte Nucléo.

➜ **Q** A quoi sert la fonction *is_angle_ok()* ?

➜ **Q** A quel moment est appelée la fonction *serialEvent* ?

➜ **M** Ouvrir le moniteur Série et envoyer la commande suivante : !A :10 ?.

➜ **Q** Que renvoie la carte Nucléo ? Expliquer les fonctions *serialEvent* (vis-à-vis du choix de la structure d'une commande) et *loop()*. On s'intéressera notamment aux changements de valeur des variables *string_complete* et *input_cnt*.

➜ **Q** Que se passe-t-il lorsqu'on transmet une mauvaise valeur (soit avec la commande A, soit avec la commande D) ?

➜ **Q** Faire un algorigramme du code.

Ajout des autres commandes

On souhaite à présent compléter ce code pour pouvoir traiter l'ensemble des commandes mentionnées dans la section précédente (T, A, B, C, S, E, D).

➜ **M** Compléter le code précédent et le tester.

➜ **Q** A quel moment doit-on utiliser la **fonction d'acquisition** réalisée à l'étape 8 ? D'autres commandes sont-elles acceptables lors de l'acquisition ? Comment bloquer l'exécution des autres commandes ?

➜ **M** Mettre en place ce blocage.

➜ **Q** A quel moment doit-on utiliser la **fonction de transmission** des données réalisée à l'étape 9 ?

Test complet

➜ **Q** Quelle est la suite de commandes à transmettre à la carte Nucléo pour permettre l'acquisition de données pour des angles allant de $-40\,\mathrm{deg}$ à $+60\,\mathrm{deg}$ par pas de $10\,\mathrm{deg}$ et la récupération des données sur le moniteur Série ?

➜ **Q** Tracer le diagramme de séquence associé à ces échanges de commandes et de réponses entre le PC et la carte Nucléo.

➜ **M** Tester l'acquisition de cette série de données.

Etape 11 / Utiliser la bibliothèque pySerial en Python pour envoyer les commandes à la carte Arduino

ATTENTION / Lors de l'utilisation d'un script Python nécessitant l'accès à la liaison Série, les outils **Moniteur Série** et **Traceur Série** d'Arduino doivent être **fermés**.

On se propose à présent d'utiliser un script en langage Python pour pouvoir transmettre les commandes et recevoir les données sur l'ordinateur. Nous utiliserons la bibliothèques **pySerial** (à installer par la commande *pip install pyserial* dans une console ou un prompt d'Anaconda).

Un code de base, nommé *python_to_Nucleo_via_Serial.py*, est disponible sur le site du LEnsE dans la rubrique *Année / Première Année / Interfaçage Numérique S6 / Banc de mesure de rayonnement lumineux / Exemples de Codes*.

➜ **M** Exécuter ce code et tester les deux commandes pré-configurées ('a' et 'd').

➜ **M** A partir de cet exemple, réaliser un script qui permette de lancer l'acquisition de données pour des angles allant de $-40°$ à $+60°$ par pas de $10°$ et la récupération des données acquises dans une liste en Python.

➜ **M** Afficher les données obtenues sur un graphique à l'aide de Matplotlib. Générer l'axe des angles.

Etape 12 / Tester la communication entre la carte Arduino et le script Python pour afficher les données

On souhaite enfin créer une application en console complète permettant à l'utilisateur de saisir les angles de départ et de fin, le pas angulaire et de lancer l'acquisition. Votre application devra ensuite afficher les données acquises.

Vous pourrez faire plusieurs essais avec des courants dans la LED de puissance de $50\,\mathrm{mA}$, $100\,\mathrm{mA}$ et $200\,\mathrm{mA}$ par exemple.

Interfaçage Numérique

Travaux Pratiques

Semestre 6

Ressources

Bloc Rayonnement

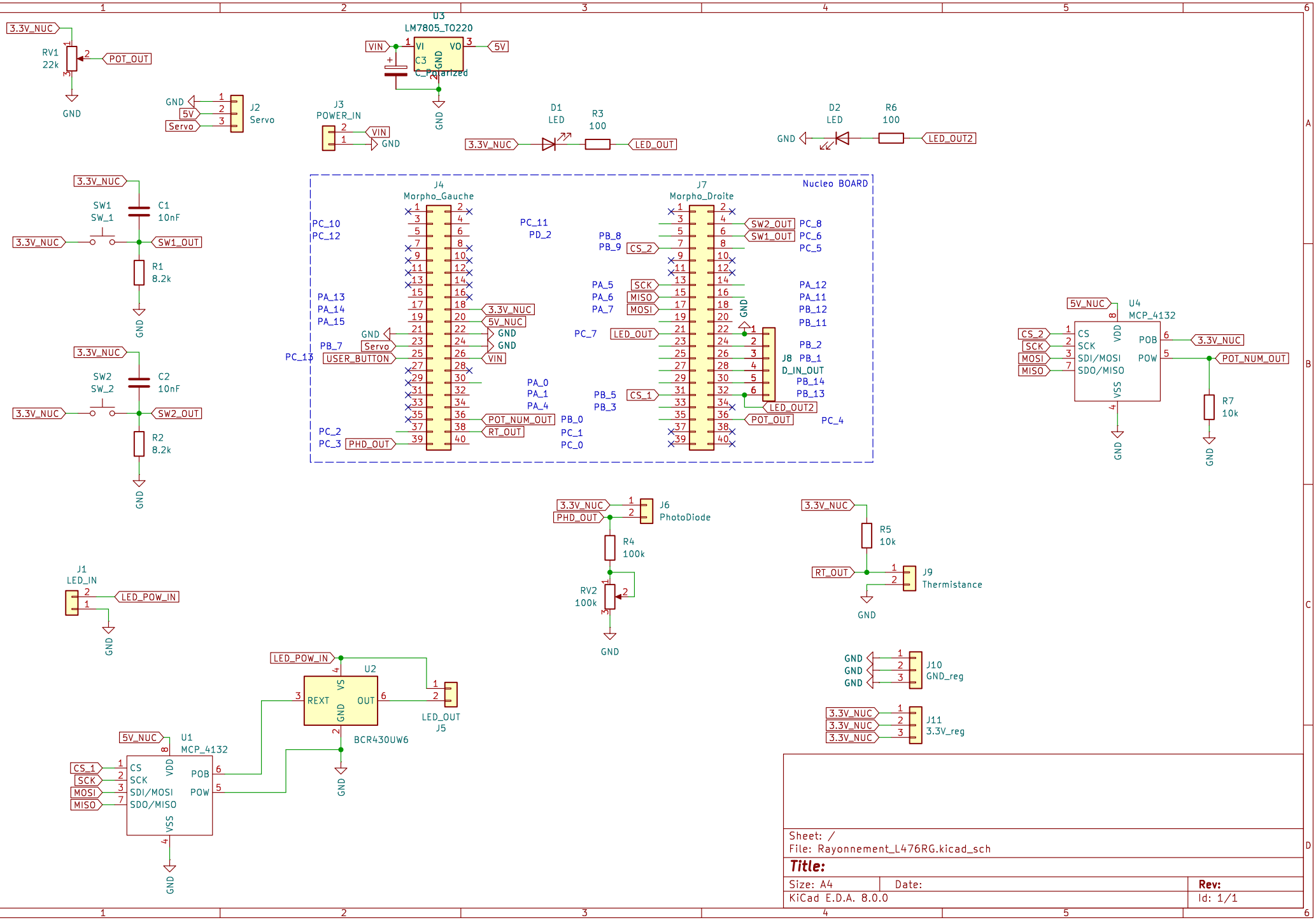
Liste des ressources

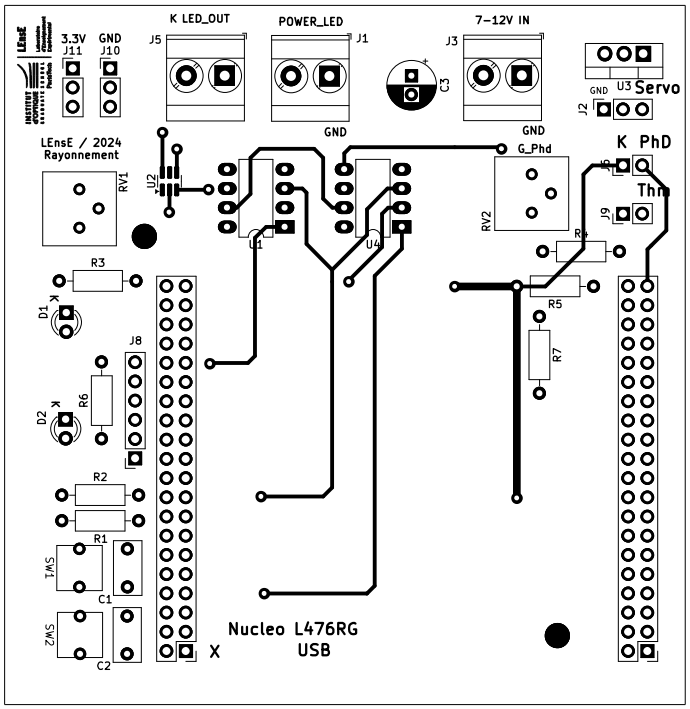
— Schéma de la carte de la maquette de rayonnement
— PCB de la carte de la maquette de rayonnement

Autres fonctionnalités

Maquette	**Broche Nucléo**	Type	Description
SPI			
SCK	PA5	Sortie	Signal d'horloge
MISO	PA6	Entrée	Données entrantes
MOSI	PA7	Sortie	Données sortantes
POT NUM	SPI	MCP4132	Potentiomètre Numérique Monté dans un pont diviseur
CS	PB9	Sortie	
Pot Num In	PB0	Entrée analogique	Sortie du pont diviseur
LED PUISSANCE	SPI	MCP4132	Potentiomètre numérique / Contrôle courant dans Driver BCR430-UW6
CS	PB5	Sortie	

```
1  LL_GPIO_SetAFPin_0_7(GPIOB,  GPIO_PIN_7,  GPIO_AF1_TIM2);
```


3.3V_NUC
RV1
22k
POT_OUT
GND
U3
LM7805_TO220
VIN
VI
VO
5V
C3
C_Polarized
GND
J2
Servo
GND
5V
Servo
J3
POWER_IN
VIN
GND
D1
LED
R3
100
3.3V_NUC
LED_OUT
D2
LED
R6
100
GND
LED_OUT2
SW1
SW_1
C1
10nF
SW1_OUT
R1
8.2k
SW2
SW_2
C2
10nF
SW2_OUT
R2
8.2k
J4
Morpho_Gauche
J7
Morpho_Droite
Nucleo BOARD
PC_10
PC_12
PC_11
PD_2
PA_13
PA_14
PA_15
PB_7
PC_13
USER_BUTTON
PA_0
PA_1
PA_4
PB_0
PC_1
PC_0
PC_2
PC_3
PHD_OUT
POT_NUM_OUT
RT_OUT
PB_8
PB_9
CS_2
PA_5
PA_6
PA_7
SCK
MISO
MOSI
PC_7
LED_OUT
PB_5
PB_3
CS_1
PA_12
PA_11
PB_12
PB_11
PB_2
PB_1
PB_14
PB_13
PC_4
J8
D_IN_OUT
U4
MCP_4132
5V_NUC
CS_2
SCK
MOSI
MISO
POB
POW
R7
10k
J1
LED_IN
LED_POW_IN
J6
PhotoDiode
R4
100k
RV2
100k
R5
10k
J9
Thermistance
J10
GND_reg
J11
3.3V_reg
U2
REXT
OUT
BCR430UW6
LED_OUT
J5
U1
MCP_4132
CS_1
SCK
MOSI
MISO
Sheet: /
File: Rayonnement_L476RG.kicad_sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.0
Id: 1/1


K LED_OUT
POWER_LED
7-12V IN
3.3V
GND
J11
J10
J5
J1
J3
C3
U3
Servo
GND
J2
GND
GND
LEnsE / 2024
Rayonnement
RV1
U2
U1
U4
G_Phd
RV2
K PhD
Thr
J9
R4
R3
R5
R7
D1
J8
D2
R6
R2
R1
SW1
C1
SW2
C2
Nucleo L476RG
X
USB