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

Institut d'Optique Graduate School

Projet d'Ingénierie Électronique pour le Traitement de l'Information

1ère année | 2020-2021

Acquisition de spectres optiques au moyen d'une barrette CCD

Romain Perron, Oscar Boucher, Eliott Béraud, Haoyu Tong, Nassim M'hammedi

Table des matières

Version du 1er juin 2021

1 Présentation générale

Nous présentons un système d'acquisition de spectres optiques de sources lumineuses, obtenus en sortie d'un spectromètre. Le matériel utilisé est constitué d'une barrette CCD 64 pixels TSL201, d'une carte Nucléo de type L476RG, d'un ordinateur avec le logiciel Matlab muni de l'App Designer, et d'un goniomètre. Ce projet, dont le découpage fonctionnel est détaillé dans la **Figure 1**, a ainsi permis la réalisation des objectifs suivants :

— utiliser une barrette CCD pour l'acquisition d'un flux lumineux *via* la génération de signaux adéquats avec la carte Nucléo,

— collecter et envoyer les données issues des 64 photosites à l'ordinateur,

— réaliser une interface Matlab qui communique avec la carte Nucléo et qui intègre le contrôle du temps d'intégration, la calibration de l'échelle en longueur d'onde avec une lampe spectrale connue, l'acquisition du spectre en choisissant le nombre de mesures effectuées (moyennage) ainsi que la sauvegarde du graphique et des données brutes.

Le projet a donc abouti à la création de 3 fichiers :

— un programme en langage C pour faire fonctionner la carte Nucléo,

— une interface Matlab,

— une aide Matlab pour l'utilisation de l'interface, sous forme de live script que l'on peut afficher avec le bouton « aide » de l'interface.

La **Figure 2** donne un aperçu du système.

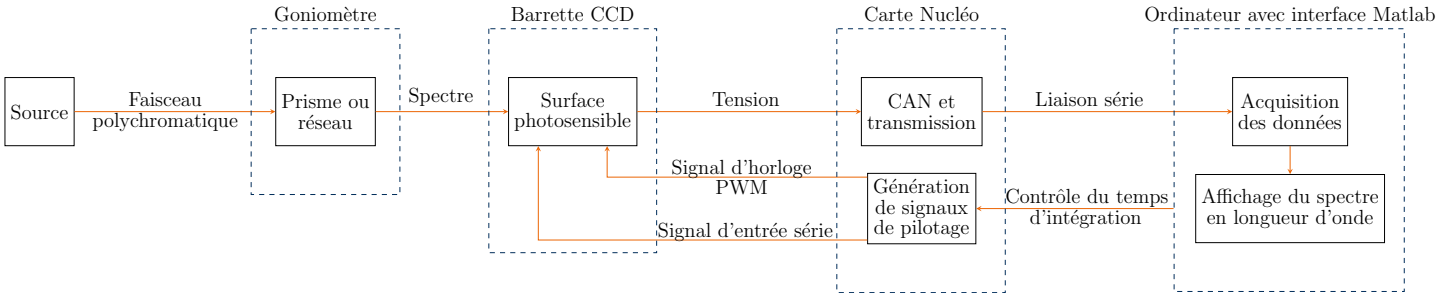


Figure 1 – *Schéma fonctionnel du projet.*

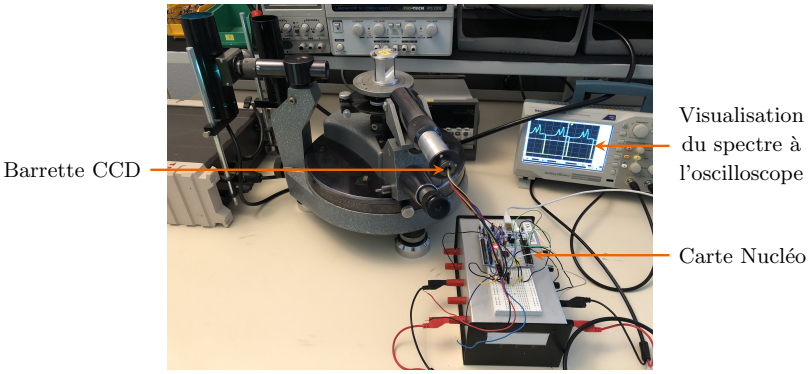


Figure 2 – *Montage de la barrette CCD sur un goniomètre.*

2 Utilisation du capteur CCD

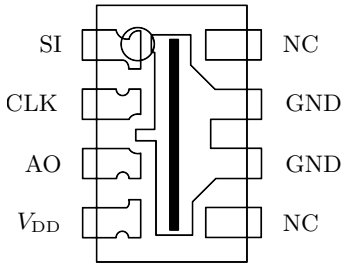
2.1 Fonctionnement de la barrette

Pour fonctionner, la barrette CCD nécessite deux signaux, en plus de l'alimentation 5V :

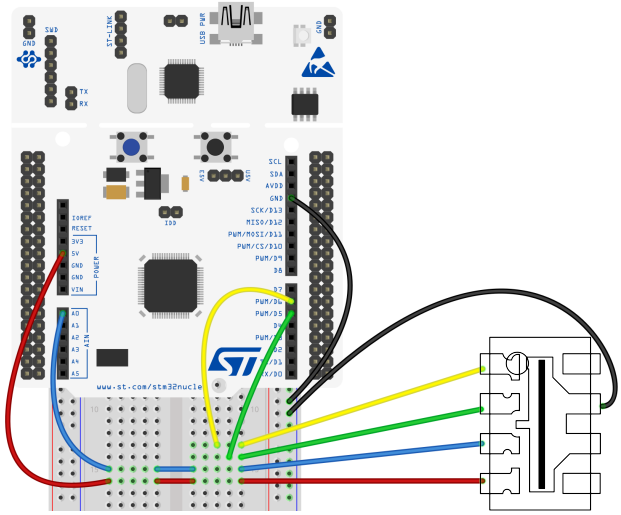
— Un signal d'horloge CLK, constitué de créneaux de période égale au temps d'intégration T_{int} souhaité : ce signal permet à chacun des 64 photosites de délivrer un signal de sortie AO. Avec la carte Nucléo, il est généré grâce à une sortie configurée en mode PWM.

— Un signal d'entrée SI, qui passe de l'état logique 0 à l'état 1 pour définir le début d'une séquence de sortie des données. Dans notre cas, l'acquisition se faisant en continu (génération périodique d'un nouveau signal SI après chaque séquence de sortie des 64 données), les impulsions sur SI sont générées après la 65ème impulsion d'horloge pour que la 64ème valeur puisse être délivrée.

La **Figure 3** précise la position des entrées et sorties sur la barrette et illustre un exemple de câblage sur la carte Nucléo. Les broches choisies correspondent à celle utilisées dans le code décrit à la section **3**.



(a) *Entrées et sortie de la barrette CCD (NC : no internal connection ; AO : sortie ; V_{DD} : tension d'alimentation).*



(b) *Exemple de câblage de la barrette CCD sur la carte Nucléo.*

Figure 3 – *Utilisation de la barrette CCD.*

2.2 Tests de fonctionnement

La vérification du bon fonctionnement de la barrette se fait grâce à la visualisation en directe des différents signaux générés et reçus sur un oscilloscope. Dans un premier temps, il suffit de laisser le capteur dans la lumière ambiante et de vérifier que l'obstruction de la surface photosensible conduit bien à diminution de l'amplitude de AO. La vérification du bon fonctionnement du signal d'horloge se fait par exemple en observant des effets de saturation lorsque sa période augmente. La **Figure 4** illustre la visualisation des signaux SI et AO, cette fois dans le cas de l'acquisition d'une partie d'un spectre constituée de 2 raies.

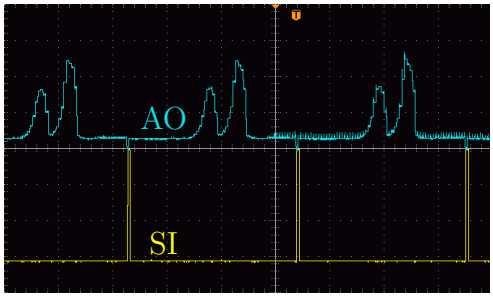


Figure 4 – *Visualisation du signal SI et du spectre obtenu à l'oscilloscope.*

3 Programme Nucléo

3.1 Principe général

Le programme est constitué d'une boucle infinie et d'une routine d'interruption. Au lancement, le signal d'horloge est généré avec une période donnée. La boucle principale permet ensuite de tester en permanence la réception de lettres depuis la liaison avec l'ordinateur et d'effectuer les actions associées : envoi des données et modification de la période du signal d'horloge (= temps d'intégration). En parallèle, une entrée d'interruption permet de détecter les fronts descendants du signal d'horloge et d'interrompre la boucle principale à chaque détection : la routine d'interruption associée permet alors de stocker les valeurs issues des photosites, puis de générer une entrée logique sur le signal SI une fois les 64 pixels stockées pour démarrer une nouvelle acquisition. Le choix de la détection des fronts descendants (et non montants) provient de la documentation technique qui indique que la sortie des données commence à chaque front montant ; il est donc plus sûr de les récupérer une fois que le signal de sortie d'un pixel est complètement généré (une valeur de pixel reste disponible pendant une période d'horloge). La **Figure 5** illustre le fonctionnement du programme.

3.2 Code en langage C pour la carte Nucléo

Le code complet est donné dans l'**Annexe 1**. Plusieurs variables globales sont d'abord initialisées, dont `T_INT`, le temps d'intégration qui définit la période du signal d'horloge CLK. La variable `a` est un compteur décrivant le nombre de valeurs stockées à tout instant du programme, sauf au premier lancement, lorsqu'il est initialisé à 64 pour qu'un signal d'entrée SI soit généré lors du premier front descendant de CLK détecté (début de la toute première acquisition). Les valeurs récoltées sont stockées au fur et à mesure de leur réception dans le tableau `tab`, `tab2` stockant les 64 valeurs récoltées après une lecture complète de tous les pixels ; il permet d'éviter que les données envoyées vers l'ordinateur ne changent avec l'actualisation de `tab` dès l'acquisition suivante.

Le préambule continue par la déclaration des broches sur la carte Nucléo. En particulier, la broche D4, configurée en entrée d'interruption, reçoit le signal d'horloge rebouclé pour la détection des fronts (cette connexion n'est pas présente dans la **Figure 3** car elle n'est nécessaire que pour la récupération des données, pas pour l'affichage sur un oscilloscope par exemple).

La routine d'interruption, nommée `acquisition` traite deux cas. Si le nombre de valeur récoltées est inférieur (strictement) à 64, une nouvelle valeur est récoltée avec la fonction `read()`, puis stockée dans `tab`. La multiplication par 1000 permet simplement d'obtenir quelques décimales sous forme d'un nombre entier, qui sera ensuite divisé par 1000 dans le code Matlab... Si l'acquisition est finie (`a` vaut 64), l'ensemble des valeurs est copié dans `tab2` puis un nouveau signal SI est généré.

Quant au main, il commence par définir la routine d'interruption de `CLK_IN` puis génère le signal d'horloge `CLK` . Il lance ensuite une boucle infinie pour tester en permanence la réception des instructions depuis l'ordinateur *via* la détection des lettres a, b et c grâce à la fonction `getc()`.

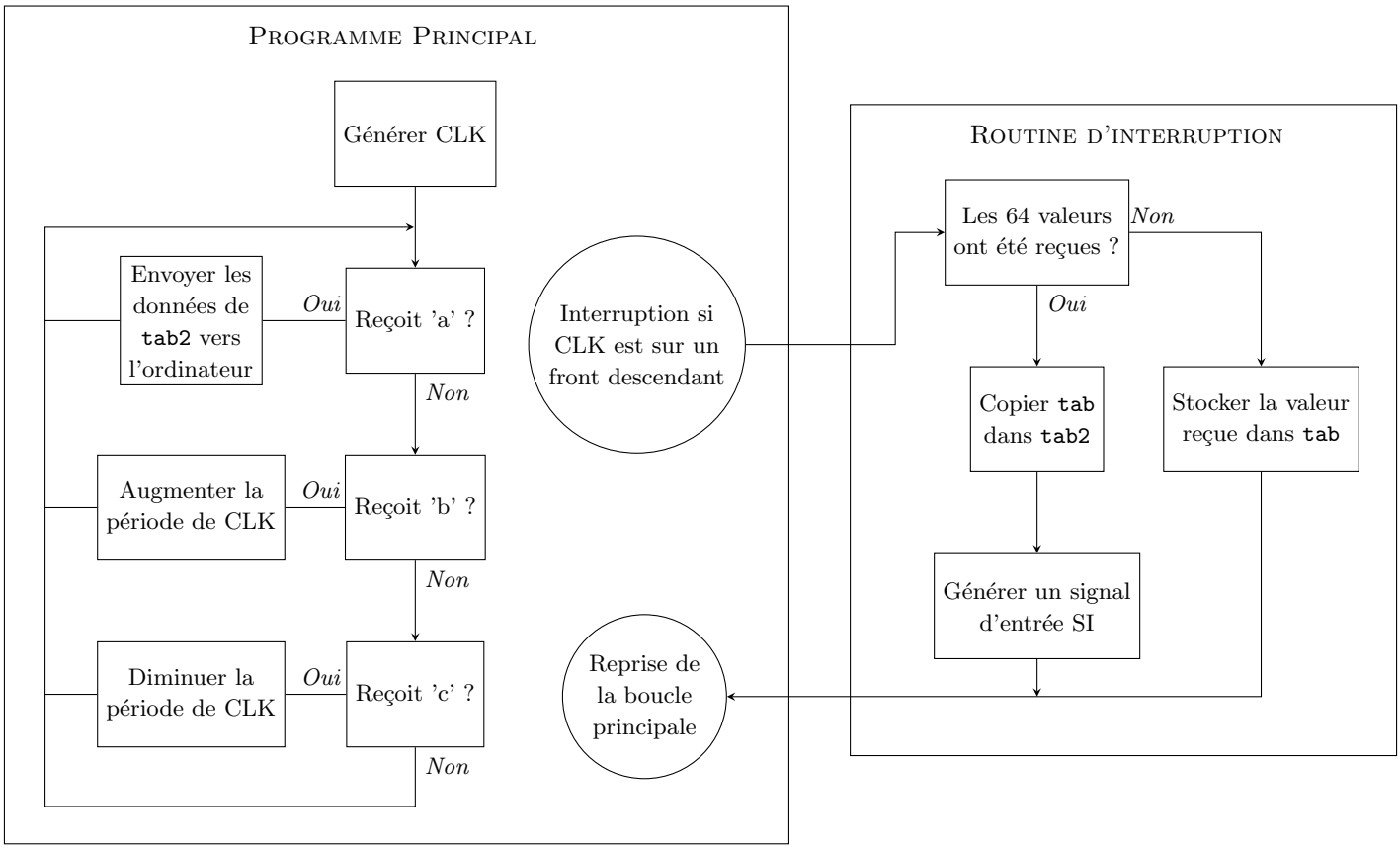


Figure 5 – *Représentation schématique du code Nucléo.*

4 Interface Matlab

L'interface a été réalisée avec l'App Designer de Matlab. La plupart du code a donc été généré automatiquement en glissant et déposant les différents blocs de notre interface. Nous avons ensuite codé les fonctions d'appel des différents boutons permettant la communication avec la carte Nucléo et le traitement des données. L'interface est présentée dans la **Figure 6**.

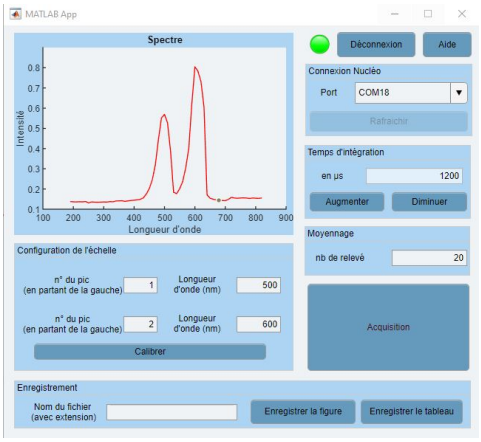


Figure 6 – *Interface Matlab.*

4.1 Récupération des données

Avant toute utilisation, il est nécessaire de bien définir le port par lequel les données vont transiter ainsi que le terminateur de fin de chaîne. Le bouton *Rafraichir* permet donc d'actualiser la liste des ports disponibles grâce à la fonction `serialportlist`. En pressant le bouton Connexion/Déconnexion, la connexion est alors établie sur le port sélectionné, avec une rapidité définie à 115200 bauds dans la fonction `serialport`. Le terminateur est configuré en mode CR/LF (retour chariot puis saut de ligne) grâce à la fonction `configureTerminator`.

On peut alors lancer l'acquisition des données avec le bouton *Acquisition* : Matlab envoie la lettre `a` avec la fonction `writeline` pour déclencher l'envoi des valeurs des pixels depuis la carte Nucléo. Pour l'enregistrement des données, un vecteur ligne de 64 colonnes est créé puis on y enregistre les valeurs successives des pixels récupées par la fonction `readline`. Il est cependant nécessaire d'effectuer une conversion « string to number » avec la fonction `str2num` puisque les données reçues sont chaînes de caractères. On trace ensuite le spectre directement dans le graphique de l'interface avec la fonction `plot`.

Via les boutons *Diminuer* et *Augmenter* du cadre *Temps d'intégration*, on peut aussi modifier le temps d'exposition : Matlab envoie à la carte Nucléo les caractères `b` pour augmenter et `c` pour diminuer.

4.2 Traitement

Notre code permet de faire des moyennes sur un nombre n de captures défini par l'utilisateur dans le cadre *Moyennage*. Le code réitère donc les acquisitions n fois puis les enregistre dans une matrice de taille $n \times 64$. La moyenne s'effectue alors avec la fonction `mean` qui peut effectuer la moyenne par ligne (donc pixel).

La fonction de calibration des abscisses (bouton *Calibrer*) utilise la fonction `findpeaks` avec laquelle il a été nécessaire de définir des seuils de hauteur de pics et de distance entre deux pics consécutifs. En effet, cette fonction cherche des extrema locaux à une courbe ; les arguments de cette fonction sont donc primordiaux afin d'éviter la détection de petites variations. Nous avons défini une hauteur minimale de pic à 20% du maximum d'intensité des 64 pixels de la barrette ainsi qu'un écart entre deux pics de 5 points. Une fois les coordonnées des deux raies enregistrées, nous pouvons établir une échelle en longueur d'onde linéaire (cadre *Configuration de l'échelle*).

Finalement, le cadre *Enregistrement* permet d'enregistrer le graphique sous un format image à préciser sous forme d'extension (*Enregistrer la figure*) ainsi que les mesures (abscisses dans une variables `l` ordonnées dans `r`) au sein d'un fichier .mat (*Enregistrer les données*) dans le répertoire courant de Matlab. Le bouton *Aide* ouvre le live script d'aide si celui-ci se trouve dans le répertoire courant.

5 Conclusion

Ce projet nous a permis de réinvestir nos connaissances en programmation de systèmes embarqués pour l'utilisation d'un capteur CCD. Nous avons aussi découvert le fonctionnement des communications entre une carte Nucléo et un logiciel tel que Matlab, ainsi que la création d'une interface graphique. Nous sommes globalement très contents du déroulement de ce projet, qui a pu être mené jusqu'au bout des objectifs fixés. Le point le plus délicat auquel nous avons dû faire face est la communication entre Matlab et la carte Nucléo. Il aurait peut être été plus judicieux d'effectuer quelques recherches préalables avant de tenter d'envoyer des données dans un format que nous comprenions mal. Le résultat est une petite perte de précision sur le réglage du temps d'intégration, qui est contrôlé avec un pas de 100 μs par l'envoi de caractères plutôt que par l'envoi d'entiers précis ; l'envoi d'entiers a été un point mal compris que nous aurions aimé éclaircir. On pourrait maintenant imaginer la mise en place d'un système de déplacement motorisé du spectromètre, pour obtenir un ensemble entièrement pilotable par l'ordinateur, de la visée jusqu'à l'acquisition...

A Annexe 1

```
#include "mbed.h"

int T_INT=500;
int a=64;
int tab[64];
int tab2[64];
char l;

PwmOut  CLK(D5);
InterruptIn CLK_IN(D4);
DigitalOut  SI(D6);
Serial pc(USBTX, USBRX);
AnalogIn capteur(A0);

void acquisition(void);

int main(){
    pc.baud(115200);
    CLK_IN.fall(&acquisition);
    CLK.period_us(T_INT);
    CLK.write(0.5);
    while(1){
        l=pc.getc();
        if (l == 'a'){
            for (int i=0;i<64;i++) pc.printf("%d\r\n",tab2[i]);
        }
        else if (l=='b') {
            T_INT = T_INT+100;
            CLK.period_us(T_INT);
        }
        else if (l == 'c') {
            T_INT = T_INT-100;
            CLK.period_us(T_INT);
        }
    }
}

void acquisition(void){
    if (a<64){
        double x = capteur.read()*1000.0;
        tab[a]=(int)x;
        a++;
    }
    if(a==64){
        for (int j=0;j<64;j++) tab2[j]=tab[j];
        wait_us(T_INT);
        SI=1;
        wait_us(T_INT);
        SI=0;
        a=0;
    }
}
```

Annexe 1 – Code en langage C pour la carte Nucléo.