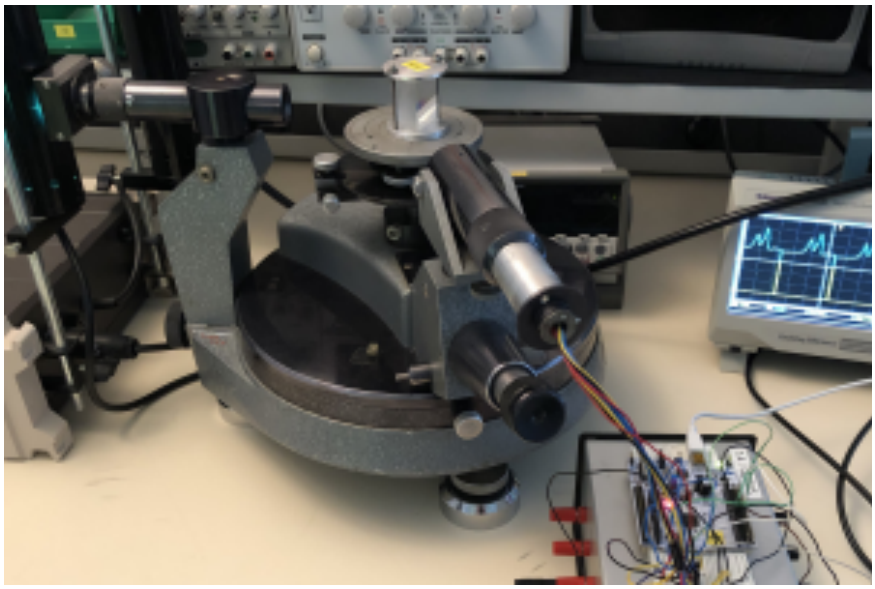
Projet IETI

4 juin 2022

Spectromètre à réseau

Fiche Technique



Simon MILCENT, Nada JAGHLAL, Aliochka DURAND, Pierre MARULLO

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

Table des matières

Nous attestons que ce travail est original, que nous citons en référence toutes les sources utilisées et qu'il ne comporte pas de plagiat.

Introduction

Lors de ce projet, nous avons essayé d'obtenir un système permettant d'acquérir le profil spectral d'une lumière. Nous nous sommes servis pour cela d'un goniomètre, d'un réseau, d'un capteur CCD et d'une carte Nucléo. Dans ce rapport technique, nous allons présenter les objectifs que nous nous étions fixés, les différentes parties du projet et nos résultats.

1 Présentation du cadre du projet

1.1 Cahier des charges

Pour cadrer un peu notre projet et pour ne pas partir dans tous les sens, nous avions établi dans les premières séances un cahier des charges à respecter. Le voilà :

Cahier des Charges

— Afficher en temps réel le spectre d'une source de lumière à l'aide d'un réseau.

Partie optique

— Régler correctement le goniomètre.
— Placer la barrette CCD au bon endroit après le système imageant, régler le système imageant pour avoir toute l'image nette sur le capteur.
— Régler la planéité du plateau.

Partie acquisition numérique

— Générer un signal d'horloge et une impulsion de départ avec la carte nucléo pour utiliser le capteur CCD.
— Récupérer le signal délivré par le capteur CCD et le convertir sous forme de tableau.

Partie affichage et transmission

— Établir une communication entre le PC et la carte nucléo à l'aide du protocole RS232.

1.2 Schéma fonctionnel

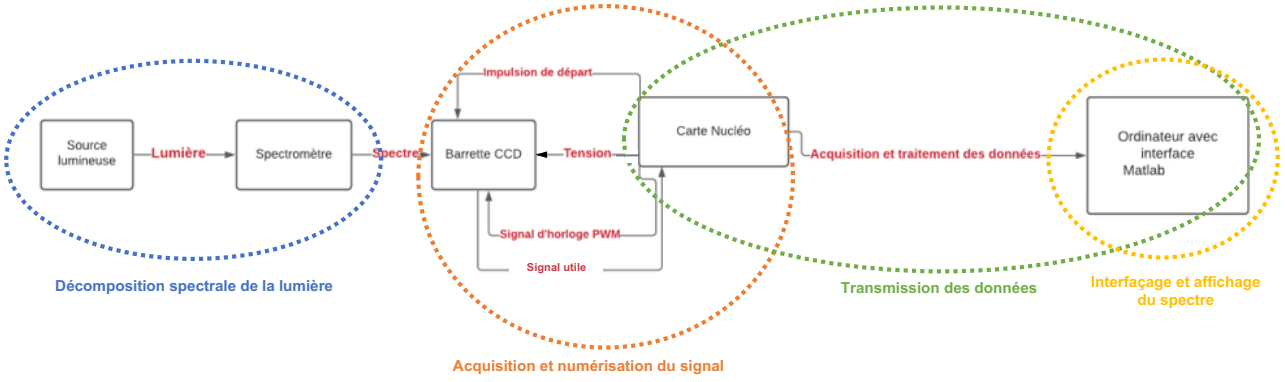


FIGURE 1 – Schéma fonctionnel du spectromètre à réseau.

Nous avons aussi réalisé un schéma fonctionnel pour lister les différentes fonctions de notre système, les éléments qui assurent ces fonctions et les liens entre ces différentes fonctions.

Nous avons découpé le projet en plusieurs blocs pour nous faciliter le travail et avancer de front.

Simon et Nada se sont occupés des blocs « décomposition spectrale de la lumière »et « Interfaçage et affichage du spectre », alors que Aliochka et Pierre se sont occupés des blocs « Acquisition et numérisation du signal »et « Transmission des données ».

2 Présentation des différents éléments du projet

2.1 Partie optique : fonctionnement du spectromètre à réseau

Le spectromètre est monté à l'aide d'un goniomètre possédant un réseau sur son plateau, le réseau permet de dévier la lumière différemment selon sa longueur d'onde. $p.n.\lambda = sin(\theta) + sin(i)$

Il faut choisir un réseau avec un pas permettant d'avoir tout le spectre focalisé dans le plan image de la lunette afocale (L).

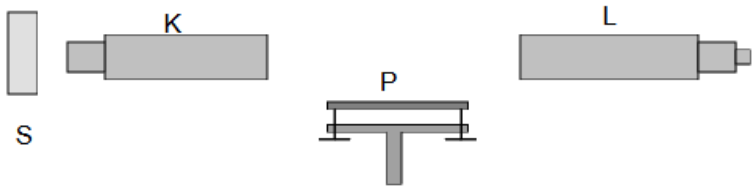


FIGURE 2 – Schéma d'un goniomètre

2.2 Partie électronique : fonctionnement du capteur CCD et montage électronique

La barrette CCD 64 pixels TSL201 que nous avons à disposition est composée de 64 photodiodes. Pour chacune de ces photodiodes, la lumière incidente crée un photocourant qui est transformé par l'électronique dans la puce en une tension. La barrette envoie ces différentes tensions une par une. Elle se calibre pour cela sur un signal d'horloge qu'il faut lui fournir. À chaque nouvelle période du signal d'horloge, le capteur CCD émet la tension correspondant à la photodiode suivante. La barrette émet ces 64 tensions à la suite après avoir reçu une impulsion appelée « Start Impulse »(SI). Ceci est résumé dans le schéma ci-dessous :

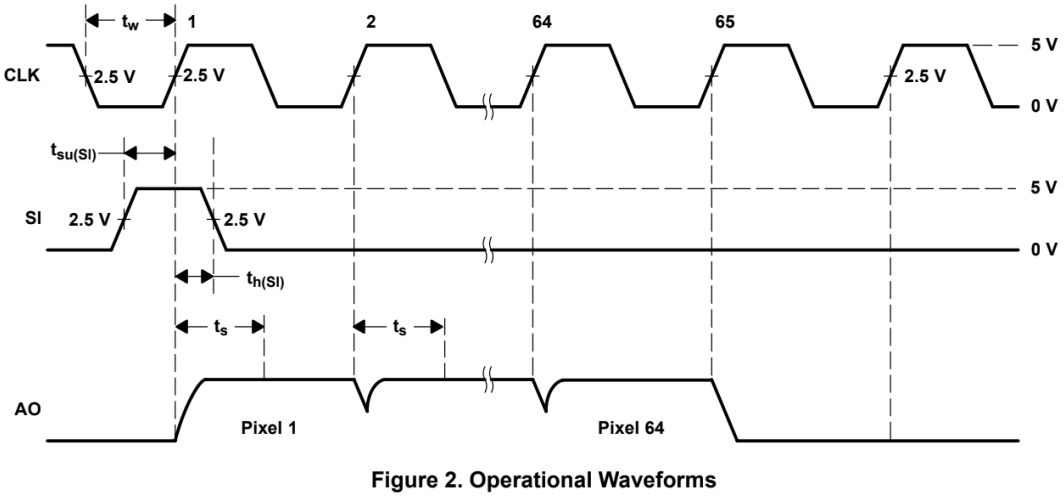


FIGURE 3 – Synchronisation du capteur CCD avec le CLK et le SI. AO est le signal utile renvoyé par le capteur.
Source : Documentation technique du capteur.

Pour assurer le fonctionnement des composants dans la puce (comme des AOP par exemple) il faut l'alimenter avec un signal continu de +5 V ainsi que lui fournir une masse. Voilà dans le schéma qui suit les différentes broches du capteur :

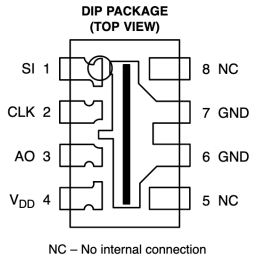


FIGURE 4 – Schéma électrique du capteur CCD. Source : Documentation technique du capteur.

La documentation technique du capteur CCD indique qu'il faut placer une résistance de 330 Ω au niveau de AO. De plus il faut placer un condensateur de $0,1$ µF entre la tension d'alimentation et la masse.

Voici le schéma du montage électronique comprenant la carte nucléo et le capteur CCD.

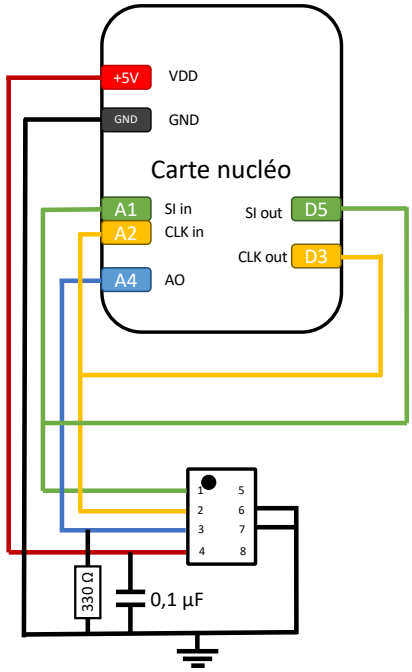


FIGURE 5 – Montage électronique du projet. En vert le SI, en jaune le CLK, en bleu le signal utile, en rouge le +5V et en noir la masse.

3 Mise en place de la partie optique

3.1 Réglage de l'instrument

Source (S)
Régler la lunette afocale (L)
Régler le collimateur ainsi que la taille de la fente (K)
Viser un ordre à l'aide de la lunette
Mettre la barrette CCD dans le plan image de la lunette afocale (*un papier blanc ainsi qu'une règle permette de savoir où placer la barrette.*)

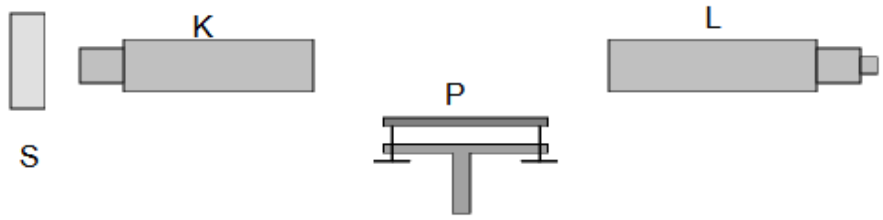


FIGURE 6 – Schéma d'un goniomètre

3.2 Étalonnage

Faire une première acquisition à l'aide d'un filtre vert
Lancer une seconde acquisition (*On peut ainsi reconnaître les pics et donc décider d'où sont les longueurs d'ondes sur l'axe des abscisses*)

4 Explication des programmes informatiques

4.1 Programme Mbed

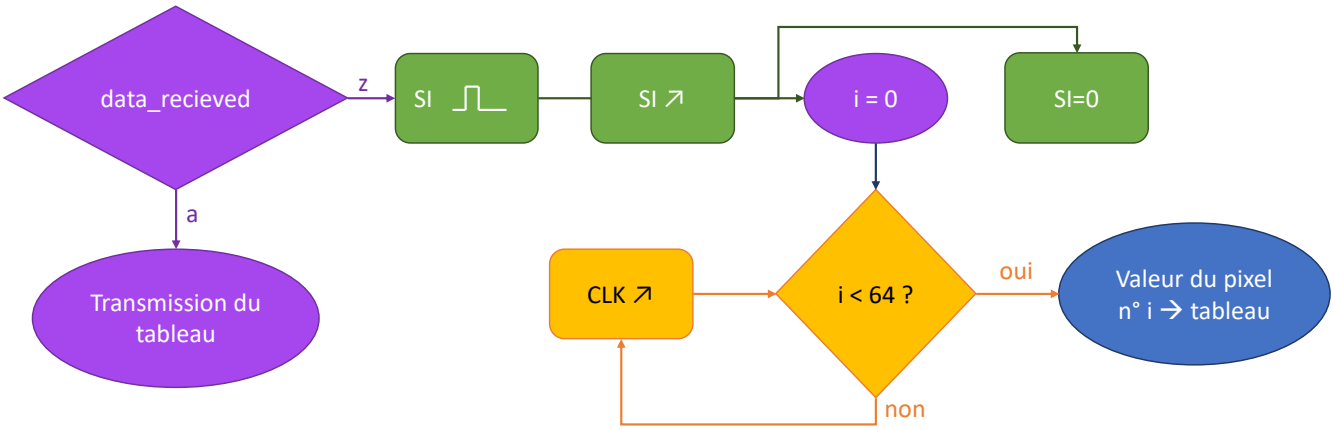


FIGURE 7 – Schéma expliquant le fonctionnement de l'algorithme Mbed qui contrôle la carte nucléo.

4.1.1 Génération de signaux périodiques

On utilise la carte nucléo pour générer les signaux utiles au capteur. Le +5 V et la masse sont directement pris sur la carte. On génère le signal `SI` et le signal `CLK` en tant que signaux PWM.

```
13 // DÉCLARATION DES SORTIES
14 PwmOut  CLK_out(D3);
15 PwmOut  SI_out(D5);
16
17 // DÉCLARATION DES PARAMÈTRES DES SIGNAUX DE SORTIE
18 int CLK_period=30; //us
19 int SI_period=2100; //us
20 double CLK_rc=0.5;
21 int SI_pw=17; //us
```

Les signaux sont générés dans le main et continuent de tourner :

```
39 int main(){
40     matlab.baud(115200);
```

```
41
42      // Génération des signaux réguliers
43      CLK_out.period_us(CLK_period);
44      CLK_out.write(CLK_rc);
45      SI_out.period_us(SI_period);
46      SI_out.pulsewidth_us(0);  //Prend la valeur SI_pw quand on lance l'acquisition
```

4.1.2 Acquisition des données

On récupère aussi ces signaux dans la carte, puisqu'ils servent à déclencher l'acquisition des données qui seront stockées dans une variable globale.

```
7   // DÉCLARATION DES ENTRÉES
8   AnalogIn    Entree(PA_4); //car l'entrée A_0 ne marche plus sur la carte nucléo
9   AnalogIn    Entree_value(A4);
10  InterruptIn StartSI(A1);
11  InterruptIn StartCLK(A2);

23  // DÉCLARATION DES VARIABLES GLOBALES
24  int TAB[64];   // Le tableau qui récupère les valeurs des 64 PHD
25  int i=65;      // Compteur qui permet de switcher d'une PHD à une autre
26                 // La valeur initiale est de 65 pour ne pas lancer l'acquisition
```

Nous avons créé des fonctions d'interruptions qui fonctionnent sur les fronts montants de `SI` et de `CLK` pour analyser correctement le signal `AO`.

```
60  void fonctionSI(){
61      int i=0;
62      SI_out.pulsewidth_us(0);// On n'observe qu'une seule impulsion
63  }
64
65  void fonctionCLK(){    // Effectue la lecture de manière synchronisée sur le CLK
66      if(i < 64){
67          TAB[i]=Entree_value.read_u16()  » 8;  // LA LIGNE DURE 24~25 US
68          i++;
69      }
70  }
```

4.1.3 Communication avec le PC et transmission des données

L'acquisition et la transmission sont commandées par le PC. On stocke dans la variable globale `data_received`.

```
28  char data_received = 0; // variable rentrée au clavier qui vaut a, ou z
29                          // => représente les ordres de matlab
```

Une fonction d'interruption déclenchée par des messages envoyés par Matlab permet de lancer la transmission (cas `a`) ou l'acquisition (cas `z`) à travers le lancement du signal `SI`.

```
73  void data_transfert(){
74      data_received = matlab.getc();//en fait on rentre le data received et selon
75                                    //sa valeur il va sur un cas ou l'autre puis
76      switch(data_received){        //il effectue la commande de la condition
77          case 'a':     // Lance le transfert des données
78              recuptab(TAB);
79              break;
80          case 'z':     // Lance une nouvelle série d'acquisition
81              SI_out.pulsewidth_us(SI_pw);
82              break;
83          default:
84              break;
85      }
86  }
```

La fonction `recuptab` permet d'envoyer un tableau d'entiers non signés sur 8 bits sous la formes de caractères à Matlab.

```
88  void recuptab(int tab[64]){
89      int l=0;
90      for(l=0;l<64;l++){
91                  matlab.putc(TAB[l]);
92                  tab[l]=0;
93      }
94  }
```

4.2 Programme Matlab

4.2.1 Acquisition et traitement des données

Du côté de l'ordinateur, sur Matlab, on établit d'abord la connexion en déclarant le port utilisé, et le type de connexion. On vide ensuite le port par sécurité.

```
nucleo = serialport('COM5',115200);
flush(nucleo)
```

Ensuite, on envoie les ordres à la carte sous la forme d'un caractère. On attend entre chaque ordre pour que la carte ait bien le temps d'acquérir les données. Ces temps d'attente sont beaucoup trop longs, mais nous les avions volontairement augmentés pour être sûr que cela ne perturbait pas l'acquisition.

```
write(nucleo,'z',"char");
pause(1)
write(nucleo,'a',"char");
pause(0.7)
```

Puis on récupère les données envoyées par la carte Nucléo, et on les stocke dans un tableau.

```
y = read(nucleo,64,"uint8"); % On lit les 64 valeurs d'un coup
```

Ce tableau est adapté pour prendre en compte la réponse spectrale du système.

```
ycorr = adaptationspectre(y,400,800);
```

Les fonctions utiles pour l'adaptation spectrale sont détaillées en annexe.

4.2.2 Partie interface homme-machine

Cette étape consiste à récupérer, sur Matlab, les données envoyées par la carte Nucléo en utilisant une interface Matlab. Celle-ci a été réalisée grâce à l'App Designer, qui est un environnement permettant de créer des applications contenant des interfaces graphiques utilisateur (GUI). L'avantage de App Designer c'est qu'il permet de générer automatiquement les codes en glissant simplement les différents blocs qu'on souhaite intégrer dans notre projet. Ensuite, nous avons codé des fonctions permettant d'activer les boutons et de les relier avec les données envoyées par la carte Nucléo.

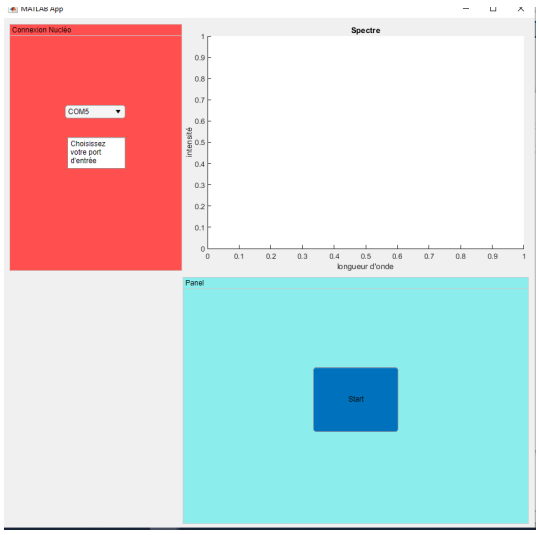
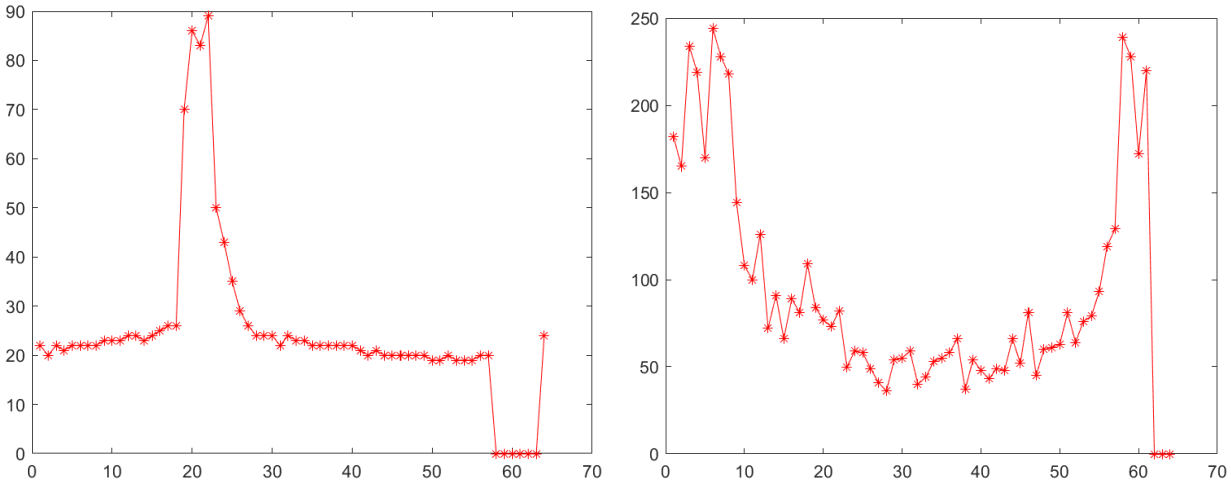


FIGURE 8 – Vue de l'interface dessinée sur App Designer.

5 Résultats et analyse

5.1 Résultats obtenus

Lors de la dernière séance, nous n'avons pas réussi à obtenir le spectre sur Matlab. Cependant, lors d'une séance supplémentaire juste avant, nous avions pu obtenir le spectre de la raie verte du mercure situé ci-dessous.



(a) Pic vert du mercure en visant le milieu du capteur (1 seul ordre visualisé)

(b) Pics verts du mercure en visant les extrémités du capteur (plusieurs ordres présents)

FIGURE 9 – Pic vert de la lampe spectrale à mercure. (Unités arbitraires)

On peut probablement expliquer le fait que le programme marche des fois et pas d'autres par le fait que le `SI` et le `CLK` ne sont pas synchronisés, or pour que la carte fonctionne, il faut que le `SI` soit haut lors du premier front montant de `CLK`.

Cependant, d'autres parties du projet, comme la transmission entre l'ordinateur et Matlab ou la partie réglage optique fonctionnent très bien.

5.2 Validation du cahier des charges

On rappelle le cahier des charges situé ci-dessous :

L'objectif principal du cahier des charges, (*Afficher en temps réel le spectre d'une source de lumière à l'aide d'un réseau*) n'est pas pleinement atteint. En effet, la contrainte du temps réel n'a pas été possible puisque dans notre dernière version, le logiciel attend presque 2 s pour afficher le spectre à l'écran. De plus, l'acquisition des données dans un tableau ne fonctionne toujours pas, sûrement faute de synchronisation entre le `SI` et le `CLK` comme nous l'avons précisé auparavant.

Cependant, la partie optique ainsi que la partie transmission et affichage respectent bien le cahier des charges.

5.3 Difficultés rencontrées

Lors de ce projet, les principales difficultés auquelles nous avons dû faire face sont liées au fonctionnement du capteur CCD. Nous aurions économisé beaucoup plus de temps en lisant attentivement la documentation technique dès le départ. (Cela nous aurait évité d'attendre la séance 6 pour comprendre qu'il nous fallait mettre une résistance en sortie du capteur !) Elles ont aussi pour origine le problème de désynchronisation entre `SI` et `CLK`.

5.4 Axes d'amélioration

Pour résoudre nos problèmes, nous avons envisagés quelques solutions que nous n'avons malheureusement pas eu le temps de tester. Parmi elles, pour résoudre le problème de synchronisation entre le `SI` et `CLK`, on pourrait augmenter le temps haut du signal `SI`.

De plus, pour augmenter la justesse de nos mesures qui sont bruitées par des signaux électroniques parasites, nous avons essayé de moyenner plusieurs acquisitions, et on aurait même pu envisager un traitement du signal *via* Matlab.

Conclusion

Lors de ce projet, nous avons pu nous rendre compte que le travail en projet n'est pas évident, et qu'un planning est difficile à suivre rigoureusement, car on ne s'attend évidemment pas à tous les problèmes rencontrés lors du projet. Nous avons dû faire des séances supplémentaires pour essayer de finir notre système dans les temps.

Nous avons aussi appris à résoudre des problèmes de manière plus ou moins autonome, car tout au long de notre projet, nous avons trouvé du fil à retordre !

Annexes

Code Mbed

```
// APPEL DES BIBLIOTHÈQUES UTILISÉES
#include "mbed.h"

// DÉCLARATION DU PORT DE COMMUNICATION
Serial  matlab(USBTX,USBRX);

// DÉCLARATION DES ENTRÉES
AnalogIn    Entree(PA_4); //car l'entrée A_0 ne marche plus sur la carte nucléo
AnalogIn    Entree_value(A4);
InterruptIn StartSI(A1);
InterruptIn StartCLK(A2);

// DÉCLARATION DES SORTIES
PwmOut  CLK_out(D3);
PwmOut  SI_out(D5);

// DÉCLARATION DES PARAMÈTRES DES SIGNAUX DE SORTIE
int CLK_period=30; //us
int SI_period=2100; //us
double CLK_rc=0.5;
int SI_pw=17; //us

// DÉCLARATION DES VARIABLES GLOBALES
int TAB[64];   // Le tableau qui récupère les valeurs des 64 PHD
int i=65;      // Compteur qui permet de switcher d'une PHD à une autre
               // La valeur initiale est de 65 pour ne pas lancer l'acquisition

char data_received = 0; // variable rentrée au clavier qui vaut a, ou z
                        // => représente les ordres de matlab

// DÉCLARATION DES FONCTIONS UTILISÉES
void fonctionSI(void);
```

```
33  void fonctionCLK(void);
34  void data_transfert(void);
35  void recuptab(int tab[]);
36
37
38  //####################         DEBUT DU MAIN          ########################//
39  int main(){
40      matlab.baud(115200);
41
42      // Génération des signaux réguliers
43      CLK_out.period_us(CLK_period);
44      CLK_out.write(CLK_rc);
45      SI_out.period_us(SI_period);
46      SI_out.pulsewidth_us(0);  //Prend la valeur SI_pw quand on lance l'acquisition
47
48      // Déclaration des interruptions et des fonctions associées
49      StartSI.rise(&fonctionSI);   //Détection de front montant pour le signal SI
50      StartCLK.rise(&fonctionCLK); //Détection de front descendant pour le signal CLK
51
52      matlab.attach(&data_transfert);//
53
54      while(1){ }
55  }
56  //#######################         FIN DU MAIN          ########################//
57
58  ////////////     FONCTIONS D'INTERRUPTIONS POUR LA LECTURE     ///////////////
59
60  void fonctionSI(){
61      int i=0;
62      SI_out.pulsewidth_us(0);// On n'observe qu'une seule impulsion
63  }
64
65  void fonctionCLK(){    // Effectue la lecture de manière synchronisée sur le CLK
66      if(i < 64){
67          TAB[i]=Entree_value.read_u16()  » 8;  // LA LIGNE DURE 24-25 US
68          i++;
69      }
70  }
71
72  ////////////      FONCTIONS D'INTERRUPTIONS DE TRANSMISSION     ///////////////
73  void data_transfert(){
74      data_received = matlab.getc();//en fait on rentre le data received et selon
75                                    //sa valeur il va sur un cas ou l'autre puis
76      switch(data_received){        //il effectue la commande de la condition
77          case 'a':     // Lance le transfert des données
78              recuptab(TAB);
79              break;
80          case 'z':     // Lance une nouvelle série d'acquisition
81              SI_out.pulsewidth_us(SI_pw);
82              break;
83          default:
84              break;
85      }
86  }
87
88  void recuptab(int tab[64]){
89      int l=0;
90      for(l=0;l<64;l++){
91                  matlab.putc(TAB[l]);
92                  tab[l]=0;
93      }
94  }
95  //                          ---- o O o ----                          //
```

Code Matlab

Fonction principale

```
nucleo = serialport('COM5',115200);
flush(nucleo)
write(nucleo,'z',"char");
pause(1)
write(nucleo,'a',"char");
pause(0.7)

y = read(nucleo,64,"uint8"); % On lit les 64 valeurs d'un coup

ycorr = adaptationspectre(y,400,800);

clear nucleo;
```

Fonction `adaptationspectre`

```
function Ycorr = adaptationspectre (Ybrut,lambdamax,lambdamin)
%
% Corrige la réponse spectrale obtenue selon la sensibilité des photodiodes
%
% ENTREES : Ybrut      vecteur des valeurs spectrales récupérées par la
%                       carte MBED
%           lambdamax   valeur de la longueur d'onde maximale observée dans
%                       l'appareil
%           lambdamin   valeur de la longueur d'onde minimale observée dans
%                       l'appareil
% SORTIES : Ycorr       vecteur des valeurs spectrales corrigées de la
%                       sensibilité des photodiodes
%

%%% FONCTION A APPLIQUER

Ycorr = Ybrut;
i=1;
for x = reglin(lambdamin,lambdamax,64)
    Ycorr(i)=Ybrut(i)/phD_spectr_resp(x);
    i++;
end
```

Fonction `ph_spectr_resp`

```
function sr = phD_spectr_resp(lambda)
%
% Renvoie la valeur de la réponse spectrale de la photodiode
%
% ENTREE : lambda   valeur de la longueur d'onde (en nm) (in [300,1100] nm)
% SORTIE : sr       valeur de la réponse spectrale

if (lambda>=300)&&(lambda<400)
    sr = 0.5/100 *(lambda-300);

elseif (lambda>=400)&&(lambda<700)
    sr=0.5/300 * (lambda-400) + 0.5;

elseif (lambda>=700)&&(lambda<=1100)
    sr=-1/400 * (lambda-700) + 1;

elseif (lambda>1100)||(lambda<300)
    error('lambda out of range [300,1100] nm');

end
```