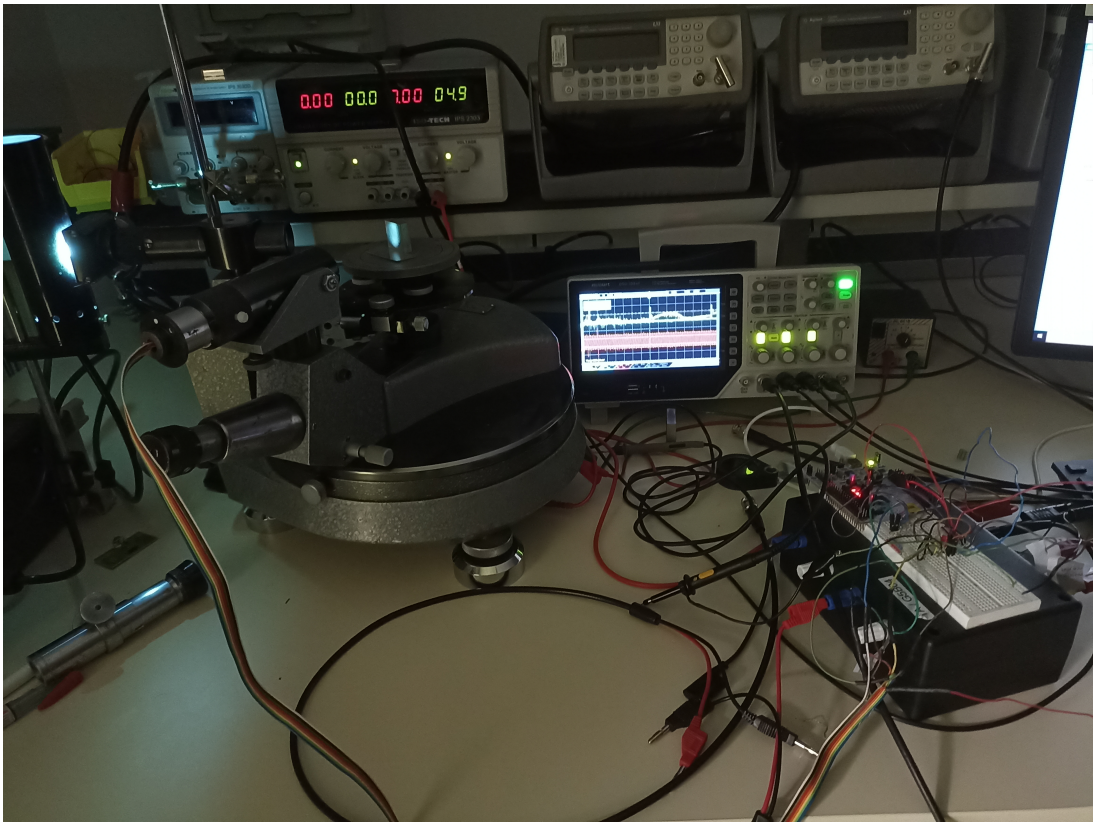
Projet IETI : Spectromètre à réseau

Noé Espanet, Nada Mhiriga Rosalie Tabarie, Lisa Lallemand

1 Introduction

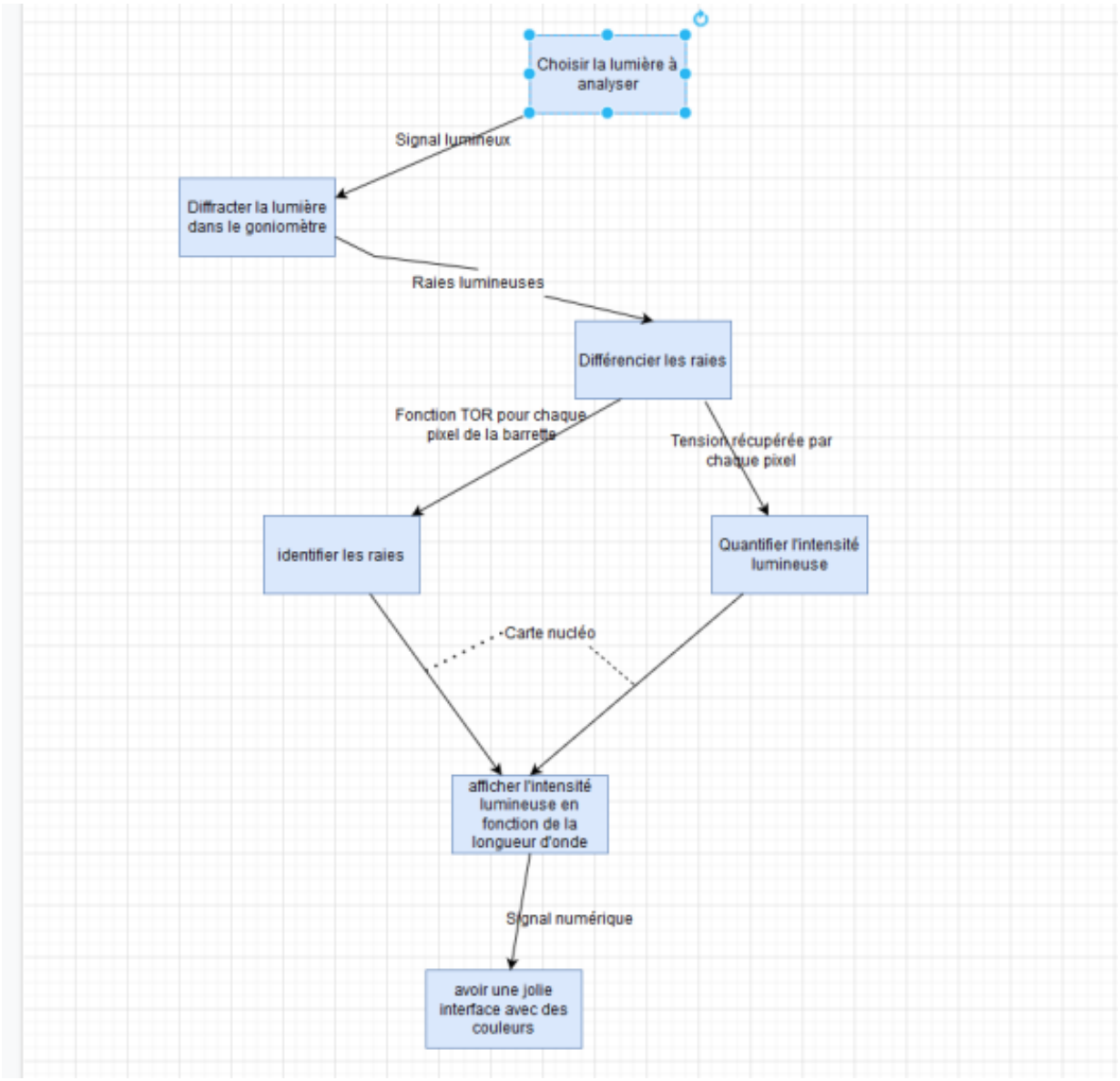
Le projet a pour but de réaliser un spectromètre, c'est à dire un système qui permet de récupérer le spectre de sources lumineuses.
On utilisera un goniomètre avec un réseau pour diffracter la lumière émise par la source et envoyer les rayons lumineux correspondants aux différentes longueurs d'ondes sur la barette ccd. Celle-ci va permettre de mesurer l'intensité lumineuse pour chaque longueur d'onde, et ces données seront ensuite transmises à l'ordinateur par l'intermédiaire de la carte nucléo. Dans un dernier temps, on pourra afficher ces données grâce à une interface réalisée avec matlab.
Dans la suite de ce dossier, on décrira plus en détail chaque étape de la réalisation de ce projet, les fonctionnalités de chaque partie et les résulats que nous avons obtenus.



Sommaire

2 Découpage fonctionnel

Ci-dessous, voilà le schéma fonctionnel que nous avions réalisé au début du projet:



En pratique, à cause de différents problèmes apparus au cours du projet(qui seront détaillés plus bas dans le dossier) nous n'avons pas pu réaliser l'interface qui devait permettre d'afficher l'évolution de l'intensité lumineuse en fonction de la longueur d'onde pour la source. En revanche, le goniomètre et son réseau envoient bien les rayons lumineux correspondant aux différentes longueurs d'onde de la source sur le capteur, et on peut afficher la sortie de ce dernier sur l'oscilloscope afin d'avoir une idée approximative des intensités lumineuses.

3 Fonctions et sections détaillées

3.1 Partie optique: réglage du goniomètre

Le spectromètre à réseau se base sur un système optique composé d'un goniomètre et d'un réseau pour disperser la lumière. Le choix du réseau est très important car il est au centre de ce montage. Étant donné que l'on veut que l'image se forme sur le capteur CCD qui fait seulement 1cm de long, il faut un réseau peu dispersif. On choisit un réseau de pas $75\mu m$.

Ensuite, il faut régler le goniomètre pour que le spectre de la lumière s'étale précisément sur toute la barette et que l'image soit nette évidemment. On choisit l'image d'ordre 1 car c'est le plus simple et le moins dispersé. On effectue d'abord un premier réglage afin d'avoir une image nette à une position connue sur le goniomètre. Après avoir obtenue cette image, on marque sa position à l'aide de typex pour avoir un moyen rapide de la retrouver. On place le capteur CCD à cet endroit en prenant soin de le mettre bien à l'horizontal.

Enfin après avoir mis en place ce montage on peut affiner le réglage en utilisant le retour donné par le capteur CCD. On a alors une résolution d'environ 20nm car le capteur ne fait que 64 pixel de long.

Figure 3: Spectre observé en sortie du goniomètre

3.2 Partie électronique: Barette CCD

La barette CCD (utilisée dans ce projet) est un dispositif à transfert de charges. Elle est composée de 64 pixels qui interagissent avec la lumière arrivant dessus et qui font que la charge éléctrique a une réponse proportionnelle à l'éclairement reçu par le capteur. Cette propriété linéaire de la barette assure l'adéquation de son utilisation pour l'étude d'un spectre.

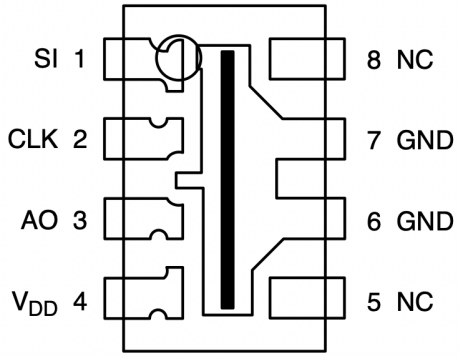


Figure 4: Broches des entrées et sorties de la barette CCD

3.2.1 Principe de fonctionnement du capteur CCD

Pour fonctionner, la barette CCD nécessite une alimentation de 5V sur la broche V_{DD}. De plus deux autres signaux sont essentiels pour le bon fonctionnement du capteur, notamment pour assurer le déplacement des charches accumulées par les pixels :

Signal SI : (Serial Input) C'est le signal d'horloge qui enclence le cycle des mesures. En effet, à chaque impulsion de SI débute un nouveau cycle comportant 64 mesures. Cette tension est liée avec la carte Nucléo sur la sortie Digitalout.

Signal Clock CLK : C'est le deuxième signal d'horloge qui permet de calibrer la durée d'une séquence pour chacun des 64 pixels du capteur et assure le transfert des charges électriques provenant de ces pixels à travers la broche AO, ainsi que le passage d'un pixel à l'autre. Cette tension peut être génerée avec la carte Nucléo grâce à une sortie configurée à l'aide du module PWM.

Figure 5: Signaux Clock et SI générés par le GBF et observés sur l'oscilloscope (jaune pour SI et vert pour Clock)

Finalement on récupère la tension de sortie sur la broche AO du capteur. Cette broche sera reliée par la suite à l'entrée AnalogIn de la carte Nucléo.

3.2.2 Test et vérification du fonctionnement de la barette

Pour vérifier que le capteur CCD marche bien, on a généré les signaux d'horloges Clock et SI à l'aide de deux GBF. On a également utilisé un condensateur de capacité 100nF et deux résistances $R_1 = 3,3k\Omega$ et $R_2 = 1,7k\Omega$. On a, par ailleurs, alimenté la barette par une tension de 5V. Ensuite, on a récupéré la tension de sortie sur l'oscilloscope (Cf Figure 6)
Afin de vérifier que les tensions des 64 pixels étaient bien envoyés successivement pour les signaux d'horloges Clock et SI choisis, il suffit de laisser la barette exposée à la lumière de la salle et de ne cacher qu'une partie des pixels du capteur et d'observer que cela conduit bien à une dimunition de l'amplitude du signal AO.

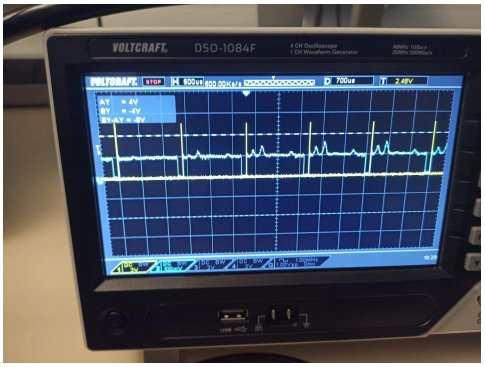
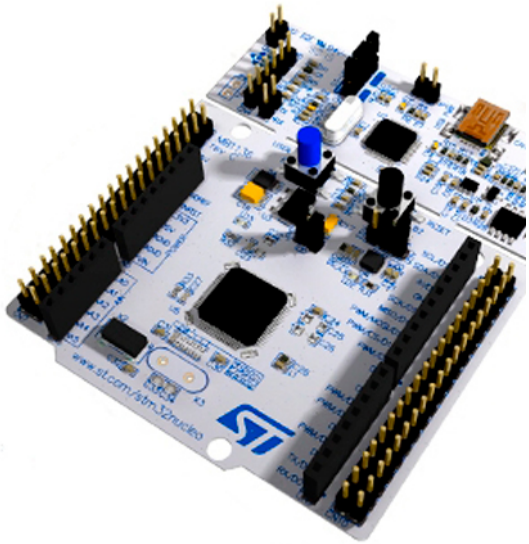


Figure 6: Graphiques des signaux d'entrée SI(jaune) et de sortie AO(bleu ciel)

3.3 Partie informatique: interface, programme nucléo et matlab

Il faut maintenant récuperer les données du capteur et les transmettre à l'ordinateur grâce à la carte nucléo. A partir de ces données, nous aurions souhaité mettre en place une interface pilotée par matlab qui permet d'afficher le graphique de l'évolution de l'intensité lumineuse de la source en fonction de la longueur d'onde. Cependant, nous n'avons pas pu le faire donc nous nous contenterons ici de nous intéresser au programme commandant la carte nucléo, trouvable en annexe.

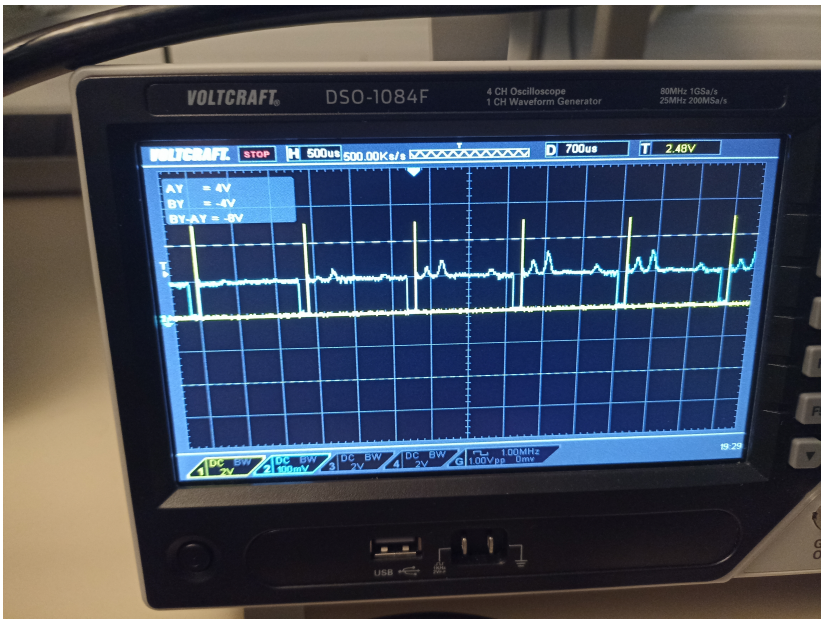


Dans un premier temps, ce programme permet d'envoyer les tensions Clock et SI au capteur, ces dernières étant primordiales à son bon fonctionnement comme expliqué dans la partie précédente. Pour cela, on utilise des sorties PWM pour lesquelles on définit des périodes et des durées d'impulsion bien choisies en cohérence avec la documentation du capteur TSL201R. Ici, on a fait le choix de déclencher une impulsion SI par le bouton de la carte Nucléo pour essayer de prélever les données récupérées par le capteur de manière plus commode.

Dans un deuxième temps, nous allons prélever la tension en sortie du capteur à intervalles réguliers, afin de pouvoir retrouver l'intensité lumineuse reçue par chaque pixel. On déclenche l'acquisition de la tension à chaque front montant de clock pour être sur de récupérer la tension qui correspond à chaque pixel. On stocke ces données dans un tableau.

4 Résultats

Au bout quelques séances, nous avons réussi à régler le goniomètre et à réaliser correctement les branchements du capteur(en l'alimentant d'abord avec des gbf et un générateur de tension continue). Nous avons pu observer à l'oscilloscope le signal de sortie du capteur, et ainsi avoir une première idée de l'intensité lumineuse pour les longueurs d'onde de la source. Ainsi, on peut voir sur l'image ci-dessous trois pics qui correspondent à trois raies de la lampe à mercure utilisée:



Plus tard, nous avons réussi à faire fonctionner le capteur directement grâce aux signaux clock et si envoyés par la carte nucléo et également à prelever régulièrement les données du capteur. Nous les avons d'abord affichées à l'aide de TeraTerm, pour vérifier que cette partie de code fonctionnait correctement. Nous n'avons cependant pas réussi à envoyer les données de la carte nucléo vers l'ordinateur, ni à mettre en place l'interface dans le délai imparti.

5 Etapes de réalisation et organisation du groupe

Dès les premières séances, nous nous sommes répartis les taches: deux membres du groupe se concentraient sur la compréhension du fonctionnement du capteur et le montage éléctrique et les deux autres s'occupaient de la partie optique avec le réglage du goniomètre. Plus tard, le premier binôme a conçu le code nucléo tandis que le deuxième s'attaquait à la réalisation de l'interface. Toutefois, il s'est retrouvé assez vite bloqué, ne pouvant pas travailler sur le code matlab avant que le code nucléo soit terminé. De manière plus générale, la réalisation du projet a beaucoup été perturbée pendant les dernières séances par différents problèmes et blocages.

Tout au long du projet, nous avons pris des notes sur la réalisation du projet dans un cahier commun et pour l'organisation du groupe, nous nous sommes également appuyés sur Basecamp.

6 Problémes rencontrés

Tout au long de notre projet, nous avons rencontré un certain nombre de difficultés.

La maitrise du capteur et du code nucléo a été compliquée: nous avons mis du temps à comprendre qu'il fallait mieux utiliser une fonction d'interruption qu'un ticker pour prélever les tensions qui correspondent aux pixels. En effet, c'est le meilleur moyen d'être sur que les valeurs prélevées sont bien celles de chaque pixel, synchonisées avec les impulsions du signal clock.

Cependant, le problème le plus handicappant que avons rencontré concerne la récupération des données par la carte nucléo. Lorsque nous avons affiché les valeurs prelevées à l'aide de Teraterm, celles associées à certains pixels étaient nulles. Les pixels n'étaient pas les mêmes à chaque fois. Malgré de nombreuses tentatives de modification du code et même de changement de capteur et de carte nucléo, nous n'avons jamais réussi à comprendre d'où venait le problème ni comment le régler. Nous nous nous sommes retrouvés bloqués et avons perdu beaucoup de temps pendant les dernières séances à cause de cela.

A la toute fin du projet, le covid d'un des membre du groupe, même si ce problème n'est pas directement en lien avec le projet, nous a également handicappé.

7 Bilan

Ce projet nous a permis d'apprendre de nombreuses choses, non seulement en éléctronique et informatique, sur le fonctionnement de la barette CCD et de la carte nucléo, mais aussi de manière plus générale. Par exemple, il vaut mieux ne pas bloquer sur un difficulté comme nous avons pu le faire, mais plutôt demander rapidement de l'aide ou essayer de passser à une étape suivante. Il est également essentiel de bien prendre en note tout ce qu'il se passe pendant chaque séance pour pouvoir s'appuyer plus facilement sur ce qui a déjà été fait, et éviter de refaire les mêmes erreurs de séance en séance.
Globalement, le projet s'est bien déroulé, même si le problème des zéros dans les données nous a beaucoup retardé et a nuit à notre motivation. Notre répartition des taches sur les dernières séances n'était pas très judicieuse, puisqu'il y avait besoin du code nucléo et de la carte pour pouvoir travailler efficacement sur le code matlab et le tester.
Si nous avions plus de temps, nous pourrions tenter de developper quand même l'interface, en utilisant des fausses données(puisque nous n'avons toujours pas réussi à résoudre le problème des zéros). Nous pourions essayer de l'optimiser au maximum, afin que la lecture des informations soient la plus claire et agréable possible.

8 Annexe: code nucléo

```

/* Déclaration des ressources externes */
#include "mbed.h"

/* Déclaration des variables globales */
double meas;
double tension;
double V[64];
int i=1;/*compteur de pixels*/
int acq_on=0;
int k ;
int j=0;//j=0 avant le prelevement et j=1 apres le prelevement
void prelevtens(void);
clock_t debut, fin;
double duree;
int ts=185; /*temps à attendre aprés le front montant de clock en ns*/
/* Déclaration des entrées/sorties */
AnalogIn analog_in(A0); /*entrée des données*/
Serial pc(USBTX, USBRX);
//PwmOut SI(D3);
DigitalOut trig(D4);
PwmOut cl(D9);
InterruptIn clentree(A2);
//InterruptIn SIentree(A5);
InterruptIn SIentree(USER_BUTTON);


/*def fonction d'interruption debutprelev pour initialiser les prelevements*/
void debutprelev()
{


    i=0;
    acq_on=1;
    trig=1;
}

/*def fonction d'interruption prelevtens*/
void prelevtens()
{



    if(acq_on==1) {




        if (i<64) {
            meas = analog_in.read(); // Convertit et lit la tension d'entée analogique (valeur entre 0.0 et 1.0)
            tension = meas * 3.3;
            V[i]= tension;
            //pc.printf("Tension du pixel %d = %lf V \r\n", i,V[i]);
        }

        i=i+1;// ajoute 1 au compteur pour passer au pixel suivant

        if (i==64) {
            acq_on=0;
            trig=0;
            for (k=0; k<64; k++) {
                pc.printf("Tension du pixel %d = %lf V \r\n", k+1,V[k]);
            }
        }
    }
}


```

```
/* Fonction principale */
int main()
{
    /* Zone d'initialisation-définition de clock et SI */

    cl.period_us(40);
    cl.pulsewidth_us(20);
    debut=clock();

    /* Prelever les données du capteur*/
    SIentree.rise(&debutprelev);
    clentree.rise(&prelevtens);



    while(1);


}
```