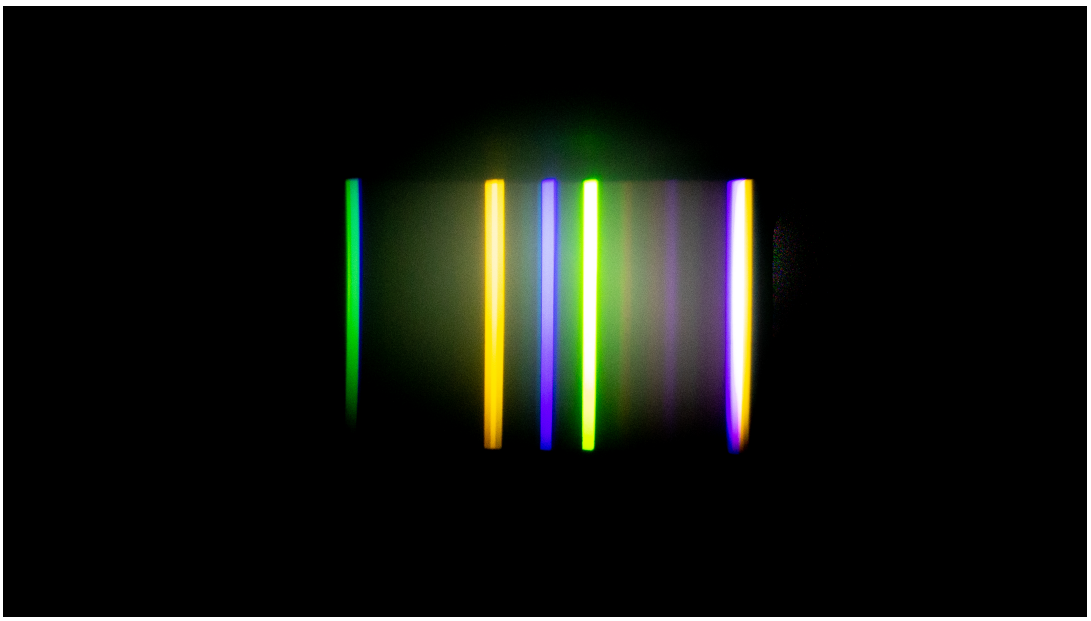
26 mai 2022

Fiche Technique Spectromètre

Inès VERGARA, Nathan MASSE, Pierrick ARPIN, Guillaume CHAPELANT



INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech

LEnsE Laboratoire d'Enseignement Expérimental

Table des matières

1 Introduction

L'objectif de ce projet est d'obtenir numériquement le spectre de raie d'une source de lumière quelconque. Pour cela, nous avons à disposition un goniomètre, plusieurs réseaux différents ainsi qu'une lampe à mercure et une barrette CCD.

Nous nous sommes fixés comme objectif que l'obtention du spectre devrait être interactif, que l'utilisateur puisse facilement calibrer l'appareil et qu'il puisse obtenir son spectre de raie en un clic.

2 Mise en place du projet

2.0.1 Cahier des charges

Afin de cadrer notre projet, nous avons effectué un cahier des charges ainsi qu'un schéma fonctionnel du fonctionnement de notre spectromètre à réseau. Voici ci-dessous le cahier des charges.

Cahier des Charges		
	Contraintes	**Valeurs**
Afficher le spectre de la lampe étudiée	Bande de fréquence	400 - 1000 nm
	Largeur à mi-hauteur	27nm
	Précision	< 10nm/pixel
	Filtrage	Filtre passe bas
Avoir un affichage directement	Affichage en temps-réel	retard de 1s max
	Temps de réponse	< 1s
Prise en main facile et pratique	Lampe d'étalonnage	Lampe à mercure de raie très visible à 578nm
	Transportable	Pas de source d'alimentation externe (autre que l'ordi)
	Points d'avancement possibles	
Asservir le système pour augmenter la précision	Automatisation	Moteur
	Pre-étalonnage par l'appareil	
	Augmenter la précision	<5nm/pixel

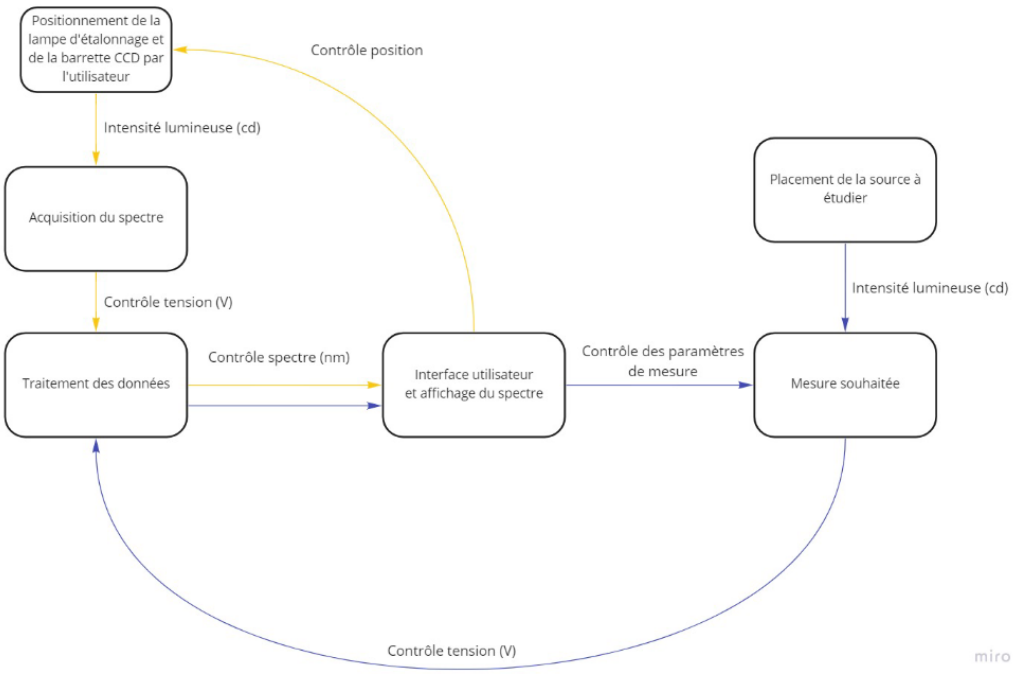
Il était important pour nous que l'utilisateur ait juste à brancher son ordinateur à la carte nucléo pour que le programme fonctionne. Nous ne voulions absolument pas que l'utilisateur ait l'obligation d'avoir des générateurs à portée de main.

De plus, l'affichage en temps réel est un deuxième point très important.

La précision est difficile à améliorer car elle dépend principalement du capteur et des réseaux dont nous disposons.

2.1 Schéma fonctionnel

Et voici ci-dessous le fonctionnement que l'on souhaite.



3 Étude optique

3.1 Fonctionnement du goniomètre à réseau

Un goniomètre à réseau est un appareil optique qui permet d'obtenir le spectre de raies d'une source. Pour cela, on utilise un réseau qui permet de diffracter la lumière. On obtient plusieurs ordres de diffraction. Il faut donc chercher l'ordre 0 où la lumière est blanche puis se décaler pour obtenir le spectre qui nous intéresse. Pour récupérer le spectre de raies, on l'envoie vers une lentille de focale 16 cm selon nos mesures.

Le choix du réseau permet de changer la taille du spectre de raies et donc de 'zoomer' sur les raies. Afin d'étalonner notre montage pour être sûr de récupérer le spectre de raies, on utilise la lampe à mercure. En effet, son spectre de raies est connu avec la raie verte à 578 nm. Il nous suffit donc d'utiliser un filtre vert pour placer correctement le spectre par rapport à la raie verte.

3.2 Étude de la barrette CCD

Fonctionnement de la barrette CCD

Afin de récupérer le spectre de raies, on utilise une barrette CCD comme ci-dessous.

Ce capteur est composé de 64 photodiodes en série. Chaque photodiode renvoie comme information une tension qui est proportionnelle à l'intensité reçue par les photodiodes. Ainsi, en récupérant la tension des photodiodes, on peut connaître le spectre de raies de la source.

Cependant, les photodiodes sont plus ou moins sensibles à la longueur d'onde incidente comme le montre cette figure ci-dessous provenant de la documentation technique de la barrette CCD.

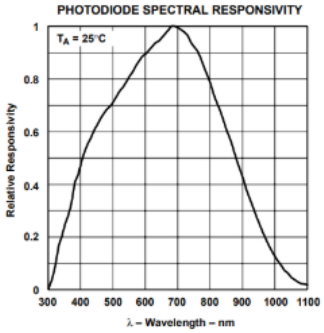


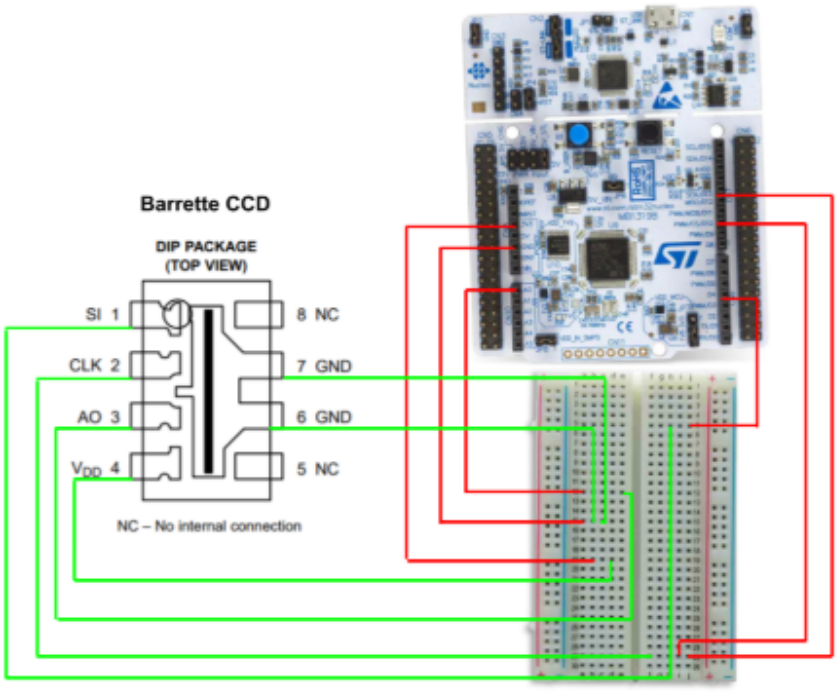
Figure 3

Il faut ainsi prendre cela en compte dans notre programme pour pouvoir avoir un résultat qui se rapproche au mieux de la réalité comme nous verrons plus tard.

3.3 Branchement de la barrette CCD

La barrette CCD fonctionne avec un principe de 'CLOCK' pour récupérer les données. En effet, pour pouvoir avoir les données, il faut envoyer un ping puis enclencher la CLOCK.

Notre capteur est branché avec une carte nucléo qui aura pour but de récupérer les données du capteur et de les envoyer vers Matlab. Ci-dessous se trouve le schéma de principe du branchement que l'on a effectué.



On verra dans la partie suivante les entrées utilisées pour récupérer les informations de la barrette.

4 Programmation pour récupérer les données

4.1 Programmation sur MBED

Afin de mettre en route et de récupérer les données renvoyées par notre capteur, on utilise une carte nucléo.

L'objectif du programme MBED suivant est de générer les deux signaux CLOCK et SI. Nous avons choisi de générer le signal CLOCK grâce à une sortie PWM. Ensuite pour synchroniser correctement les deux signaux on génère le signal SI sur un front descendant du signal CLOCK à l'aide d'une fonction d'interruption.

```
///     DECALARATION VARIABLES GLOBALES      ////

int compteur=0;                   /* Déclaration du compteur (qui va de 0 à 65)*/
int T=30;                         /* période du signal clock en µs */
int T_prim=T*2/3;                 /* Durée du créneau de mesure en µs */
int Spectre[65];                  /* Tableau qui contient le spectre */

char data_received = 0;           /* Permet de lancer le transfert des données */
char Spectre_c[64];               /* Tableau qui contient le spectre en 8bits */

///      DECLARATION ENTREES/SORTIES      ///

DigitalOut sync_debut(D10,0);     /* Créneaux des débuts de mesure */
PwmOut  cl(D3);                   /* Signal clock (permanent) */
InterruptIn sync(D13);            /* Déclaration de l'interruption (On réinjecte le signal clock dans PA_12)*/
AnalogIn Lecture(A0);             /* Pin de lecture du signal de sortie du capteur */
Serial      rs232(USBTX, USBRX);  /* Liaison RS232 */




///     FONCTIONS      ///

void mesure();
void Envoie_donnees();

int main()
{
    rs232.baud(115200);

    sync.fall(&mesure);           /* Déclaration de l'interuption sur un front déscendant de Clock */
    cl.period_us(T);              /* Initialisation du signal Clock de période T */
    cl.write(0);
    rs232.attach(&Envoie_donnees); /* Déclaration de l'interuption sur le caractère 'a' */

    while(1){}
}
```

```
void mesure()
{
    if (compteur==0)              /* Arrivé au 66 front descendant = 64e pixel...*/
    {
        sync_debut=1;             /* Créneau de début */
        wait_us(T_prim);
        sync_debut=0;
        compteur=65;              /* Début d'un nouveau cycle de mesure */
    }
    else
    {
        Spectre[65-compteur] = Lecture.read_u16();    /* Enregistrement du spectre dans un tableau */
                                                      /* compteur va de 65 à 1(dans le else) donc 65-compteur va de 0 à 64*/
        compteur=compteur-1;                          /* Lecture d'un pixel suplémentaire */
    }
}


void Envoie_donnees(){
    data_received = rs232.getc();
    switch(data_received){
        case 'a':
            int i;
            for (i=0;i<64;i++)
            {
                Spectre_c[i]=(Spectre[i]>>8);
                if (Spectre_c[i]==255)
                {
                    Spectre_c[i]=254;
                }
                rs232.putc(i);
                rs232.putc(Spectre_c[i]);
                rs232.putc(255);
            }
            break;
        case 'z':
            {
                cl.write(0.5);
            }
            break;
        default:
            break;
    }
}
```

Puis on envoie tout vers Matlab.

4.2 Programmation Matlab

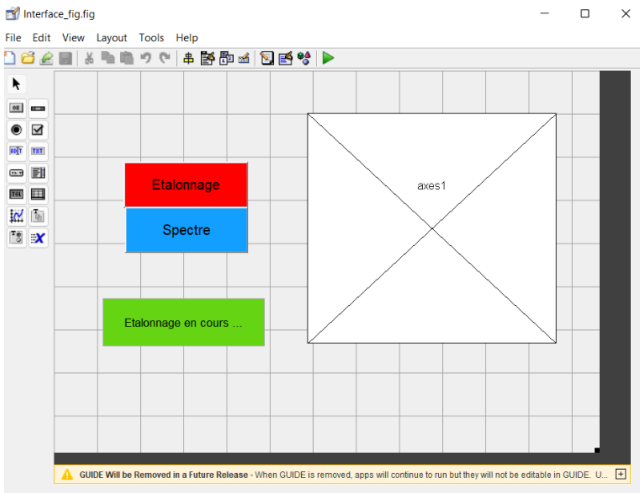
La fonction TGV a pour but de lancer un train de mesures et de récupérer les données de la photodiode. Pour cela nous avons établi un protocole de transmission pour être sûr de ne pas mélanger les données. Dans le code MBED on envoie d'abord à MATLAB l'indice de la valeur, puis la valeur et enfin un séparateur (ici 255). Il s'agit de la valeur max. On effectue donc un seuillage des valeurs à 254.

TGV.m

```
function Tableau_valeurs = TGV()

sp= serialport("COM6",115200);
nb = 0;
p='e';
tab=zeros(1,192);
Tableau_valeurs=zeros(1,64);

write(sp, 'z', 'char');

write(sp, 'a', 'char');


while nb < 192
    if sp.NumBytesAvailable>0
        tab(nb+1)=read(sp,1,'uint8');
        nb = nb + 1;
    end
end


for i=1:64
    if tab(i)==1 && tab(i-1)==255
        i_0=i-3;
    end
end

for k=i_0:i_0+63
    Tableau_valeurs(k)=tab(3*k-1);
end

clear sp;
```

Puis on utilise cette fonction dans un autre programme qui permet à l'utilisateur d'interagir directement avec le programme. En effet, Matlab permet de créer une interface après lancement du programme. Cette fonctionnalité correspond exactement à notre envie d'interaction directe entre l'utilisateur et la programme.

Cela se présente de la façon suivante :



Nous avons décidé de mettre 2 boutons pour l'utilisateur et un bouton servant juste pour indiquer à l'utilisateur l'état du système. En effet, le bouton " Étalonnage " s'utilise lorsque l'on doit étalonner notre appareil avec la lampe à mercure. Ce programme s'arrête lorsque la raie verte du mercure est placée à la bonne place sur le graphe en temps réel (soit pour 578 nm). Tant que ce n'est pas le cas, le programme affiche en continu sur l'axe la courbe du spectre que le capteur reçoit.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

truc_nul=0;

while truc_nul==0


    Tableau_valeurs = TGV();

    a1 = 0.001678571; %Coefficient directeur de la droite de l'atténuation en fonction de la longueur d'onde (avant 680nm)

    a2 = -0.00191667; %Coefficient directeur de la droite de l'atténuation en fonction de la longueur d'onde (après 680nm)

    b1 = -0.14128572; %Coefficient pour avoir la bonne droite pour la partie avant 680nm

    b2 = 2.303333336; %Coefficient pour avoir la bonne droite pour la partie après 680nm



    for i = 1:45

        Tableau_valeurs(i)= Tableau_valeurs(i)/(a1*(400+6.25*(i-1))+b1);

    end



     for i = 46:64
         Tableau_valeurs(i)= Tableau_valeurs(i)/(a2*(400+6.25*(i-1))+b2);
     end

     max=0;
     indice_max=0;
     for indice=1:64
         if Tableau_valeurs(indice)>max
             max=Tableau_valeurs(indice);
             indice_max=indice;
         end
     end

     if indice_max==24
         break
     end
     x = linspace(400,800,64);
     plot(x,Tableau_valeurs,'r')
     newmap = turbo;                     %starting map
     ncol = size(newmap,1);             %how big is it?
     zpos = 1 + floor(1/4 * ncol);      %2/3 of way through
     newmap(zpos,:) = [1 1 1];          %set that position to white
     colormap(newmap);                  %activate it
     colorbar('southoutside')
     caxis([400 800])
     grid on
     drawnow;
end
set(handles.edit3,'String',num2str('Etalonnage Terminé !'))
```

Le bouton ‘Spectre’ permet quant à lui d’afficher, après l’étalonnage, le spectre de la source qui nous intéresse. Il y a donc un unique affichage de la courbe.

```
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Tableau_valeurs_m=zeros(1,64);

for indice_m=0:29

    Tableau_valeurs = TGV();

    for j=1:64
        Tableau_valeurs_m(j)=Tableau_valeurs_m(j)+Tableau_valeurs(j);
    end

end

Tableau_valeurs=(1.0/30)*Tableau_valeurs_m;

    a1 = 0.001678571; %Coefficient directeur de la droite de l'atténuation en fonction de la longueur d'onde (avant 680nm)

    a2 = -0.00191667; %Coefficient directeur de la droite de l'atténuation en fonction de la longueur d'onde (après 680nm)

    b1 = -0.14128572; %Coefficient pour avoir la bonne droite pour la partie avant 680nm

    b2 = 2.303333336; %Coefficient pour avoir la bonne droite pour la partie après 680nm



for i = 1:45
    Tableau_valeurs(i)= Tableau_valeurs(i)/(a1*(400+6.25*(i-1))+b1);
end

for i = 46:64
    Tableau_valeurs(i)= Tableau_valeurs(i)/(a2*(400+6.25*(i-1))+b2);
end
x = linspace(400,800,64);
plot(x,Tableau_valeurs,'r')
newmap = turbo;                    %starting map
ncol = size(newmap,1);            %how big is it?
zpos = 1 + floor(1/4 * ncol);     %2/3 of way through
newmap(zpos,:) = [1 1 1];         %set that position to white
colormap(newmap);                 %activate it
colorbar('southoutside')
caxis([400 800])
grid on
```

On peut voir dans ces programmes les réajustements de courbes en fonction de la longueur d’onde reçue par les photodiodes.

5 Validation

Nous avons validé le fonctionnement de notre système lors de l’avant dernière séance. Nous avons réussi à faire en sorte que l’étalonnage soit très rapide et que le spectre obtenu soit bien propre. Lors de la dernière séance cependant le signal que nous obtenions était très mauvais. Nous pensons que cela peut-être dû soit à la lampe qui n’avais pas suffisamment chauffée, soit au capteur qui serait détérioré.

Finalement par rapport à notre cahier des charges, seule les clauses de la précision et de la largeur à mi-hauteur ne sont pas validées. Cependant nous nous sommes aperçus que ce n'était pas la largeur à mi-hauteur qui avait un intérêt (puisqu'elle dépend de la taille de la fente), mais plutôt la capacité à séparer les longueurs d'onde. C'est donc le choix de notre réseaux qui va nous permettre de valider ce critère. Nous avons fait tout les calculs pour adapter notre échelle de longueur d'onde en fonction du pas du réseau, mais nous n'avons malheureusement pas eu le temps de les vérifier expérimentalement avant de les implémenter.

6 Difficultés rencontrées

Notre principale difficulté a été de faire fonctionner le capteur. C'était une étape charnière qui conditionnait l'avancée de tout le projet et nous avons bloqué dessus pendant plusieurs séances. Finalement nous avons appris à mieux gérer les obstacles et à être plus méthodique sur la recherche des erreurs. Ainsi une fois que cette difficulté à été surmontée nous avons avancé beaucoup plus rapidement.

Nous n'avons pas réussi à respecter notre planning initial compte tenu des multiples difficultés que nous avons rencontrées sur les premières séances. Cependant nous avons fait preuve de suffisamment d'agilité pour nous adapter et terminer notre projet dans les temps.

7 Conclusion

Finalement, nous avons appris beaucoup de choses sur la gestion technique d'un projet : comment résoudre les problèmes, comment répartir les taches, comment adapter le plan initial pour répondre au mieux aux contraintes, comment penser un code pour que la mise en commun avec le reste du projet soit la plus naturelle et facile possible...

Ce projet nous a également permis d'apprendre à gérer un travail de groupe à moyen terme.

Références

[1] *Laboratoire d'enseignement Expérimental*. URL : `http://lense.institutoptique.fr/`.