ProTIS - Pierre MARULLO - Louis GONORD - Maxence VANDEWYNCKELE

Rapport technique

Asservissement en position d'un laser

Sommaire

Introduction

Objectif - Schéma de principe

Notre projet consiste à réaliser un prototype de système d'asservissement en position linéaire d'une barrette CCD, afin qu'un laser HeNe soit pointé en son centre quel que soit son déplacement. Pour réaliser ce prototype, nous disposons d'une carte Nucléo pour réaliser l'asservissement en traitant les signaux issus de la barrette CCD, et en commandant le servomoteur qui contrôle la position de la barrette CCD via une crémaillère. Le schéma du prototype est présenté ci-dessous.

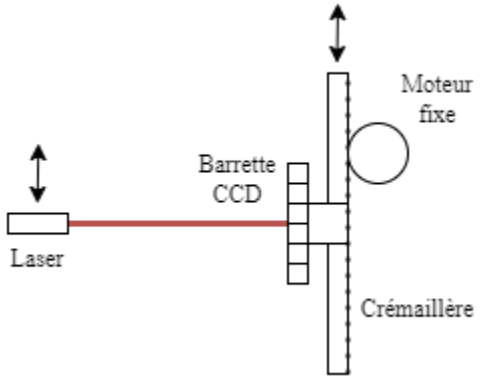

Figure 1 - Schéma du prototype

Notice d'utilisation

Pour utiliser notre prototype, il faut effectuer les branchements, placer les densités optiques adéquates devant le capteur, téléverser le code compilé sur la carte Nucléo, et s'assurer que la tache laser soit bien placée sur la barrette CCD.

Cahier des charges / Contraintes et performances attendues

Cahier des charges : Assurer le positionnement d'un faisceau laser au centre du laser quel que soit le déplacement horizontal de celui-ci.

Performances attendues :

- Vitesse linéaire maximale de 10 cm/s.
- Erreur de pointage nulle
- Interface Humain-Machine utilisable par des étudiants en sciences.

Contraintes :

- Servomoteur
- Carte Nucléo
- Asservissement numérique par carte Nucléo

Schéma fonctionnel

Notre schéma fonctionnel se présente sous la forme d'un schéma-bloc d'asservissement. Le développement de notre projet s'est axé sur trois fonctions : l'acquisition, l'actionneur, et le correcteur.

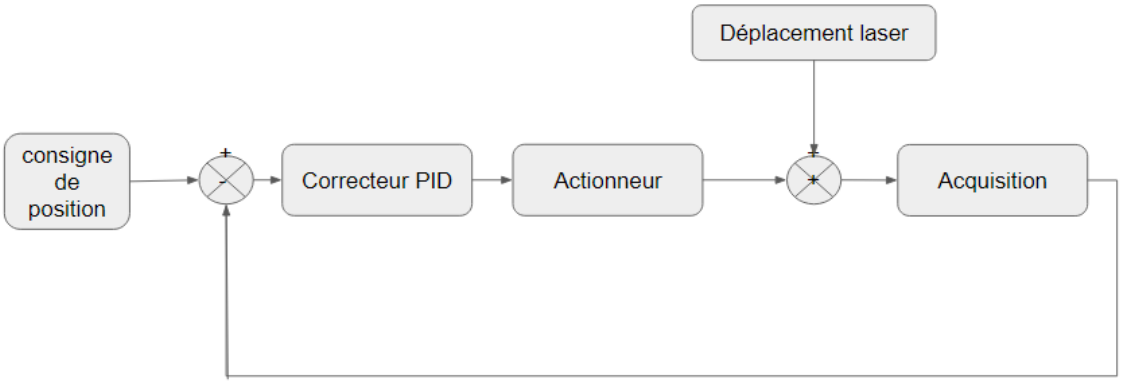

Figure 2 - Schéma-bloc fonctionnel

Description du système

Acquisition

Le développement première fonction consiste à créer un code sur Keil Studio permettant de déduire du signal brut temporel issu de la barrette CCD la position du centre de notre tache laser.

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 :

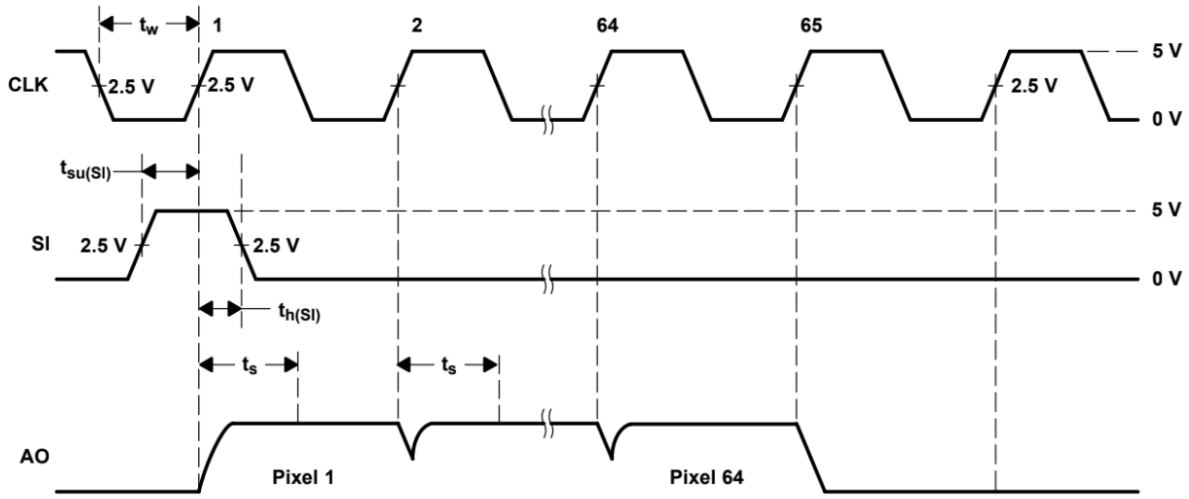

Figure 2. Operational Waveforms

Figure 3 – Synchronisation du capteur CCD avec le CLK et le SI. AO est le signal utile renvoyé par le 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 :

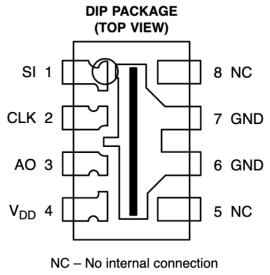

Figure 4 – Schéma électrique du capteur CCD

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.

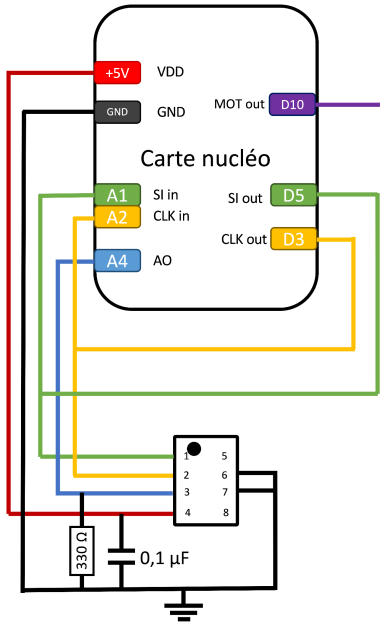

Figure 5 – Montage électronique du projet. En vert le SI, en jaune le CLK, en bleu le signal utile, en violet, le signal de commande du servomoteur, en rouge le +5V et en noir la masse.

Après que nous soyons parvenus à obtenir le signal temporel brut issu du capteur CCD, il a fallu travailler à en déduire la position du centre de la tache laser sur le capteur. Pour cela, nous avons d'abord réduit la saturation des photosites de la barrette, qui était très importante au flux nominal de notre laser, nous empêchant d'aller plus loin. Pour la réduire, nous avons eu recours à des densités optiques, pour réduire la puissance optique du laser. Nous avions le choix entre le placement en amont, juste devant le laser, et en aval, juste devant la barrette CCD, et nous avons opté pour le placement devant la barrette. En effet, cela permettait par la même occasion de réduire les flux parasites qui provenaient de l'éclairage ambiant, et donc d'augmenter le rapport signal sur bruit.

Nous obtenions alors un profil de tension à l'oscilloscope consistant en une belle bosse non saturée. Ensuite, par le calcul de barycentre de cette courbe, nous en déduisions la coordonnée du maximum d'intensité laser à chaque acquisition, permettant d'en déduire la position du spot laser, utile pour l'asservissement présenté ultérieurement.

Actionneur

Le servomoteur est l'élément du système qui agit sur la position du capteur, afin que celui-ci aie toujours en son centre la tâche laser.

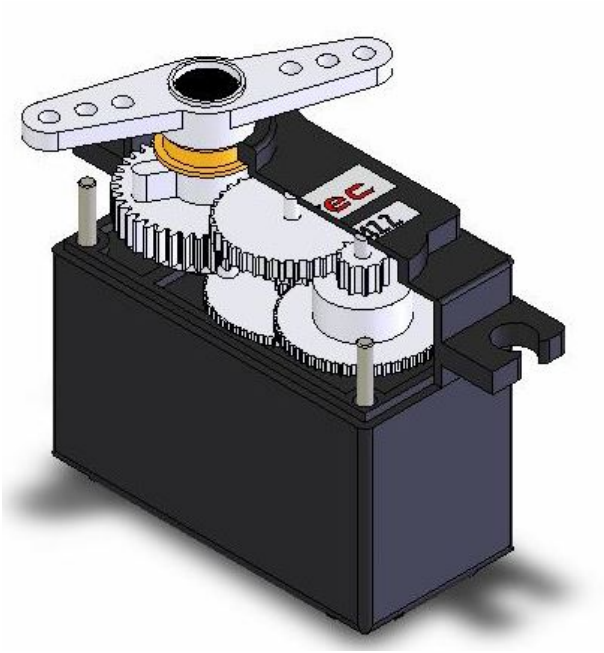

Figure 6 – Servomoteur

Il consiste en un moteur à courant continu accompagné d'un capteur de position et d'un contrôleur numérique. Il permet de réaliser des rotations d'angle calibrées, à la différence d'un classique moteur à courant continu qui renvoie un mouvement rotatif continu, modifiable en terme de vitesse de rotation. Le servomoteur permet ainsi de réaliser de courts mouvements, pouvant changer de sens régulièrement. Dans le cadre d'un asservissement laser ce composant est donc tout à fait adapté : nous avons besoin de mouvements courts, dans un sens ou dans l'autre, afin de suivre au mieux les fluctuations spatiales du faisceau.

L'axe de rotation du servomoteur est directement lié à une roue dentée, liée elle-même à une crémaillère transformant la rotation en une translation. Le capteur est monté sur la crémaillère, comme illustré sur la figure 7.

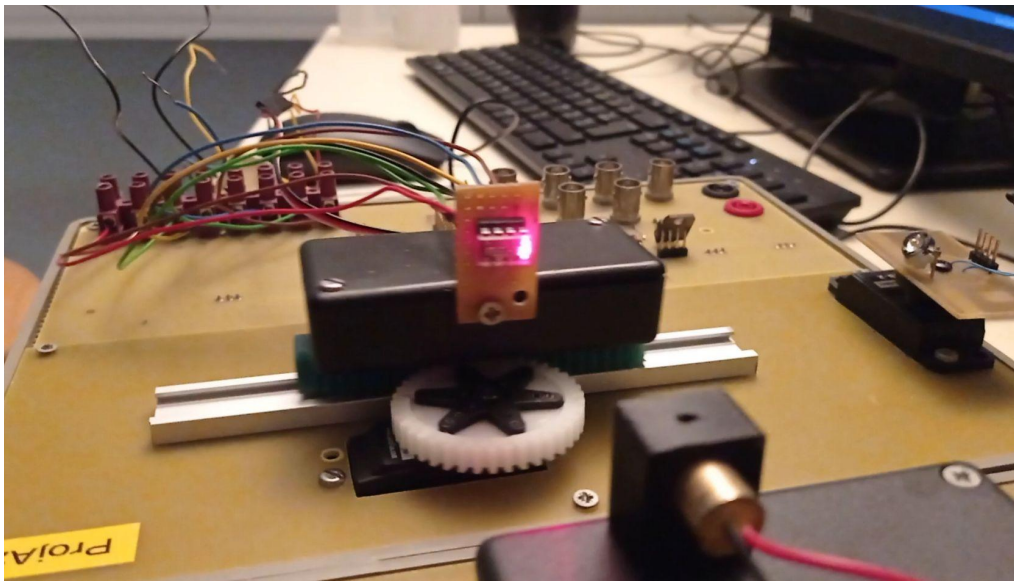

Figure 7 – Capteur monté sur un boîtier, lui-même monté sur la crémaillère

Le servomoteur fonctionne à une période de $20\,ms$, donc peut recevoir une nouvelle commande toutes les $20\,ms$, cette période pouvant être allongée. La durée du temps haut au début de chaque période détermine la consigne d'angle du servomoteur.

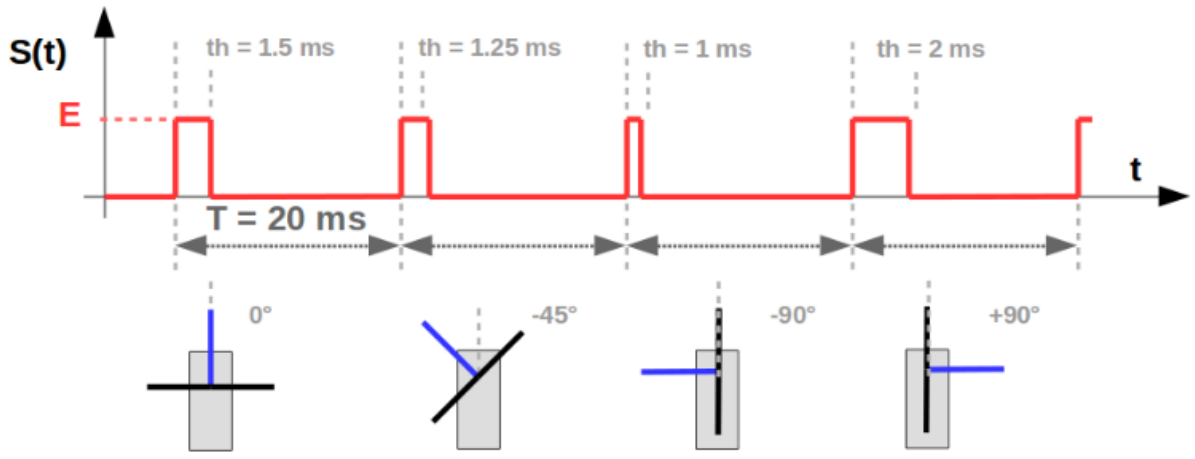

Figure 8 – Correspondances angle/temps haut

Cependant, ce qui nous intéresse ici c'est plutôt la position atteinte sur la crémaillère, ainsi nous avons observé à quelle durée de temps haut correspondait aux positions extrêmes de la crémaillère afin de définir la plage de fonctionnement de l'actionneur.

Enfin, nous nous sommes penché sur le nœud du problème : la réponse en position de l'actionneur à l'indication de position du faisceau sur la barrette donnée par toute la partie acquisition.

L'information donnée est un entier relatif $i \in [-31, 32]$, indiquant la position du laser sur la barrette : s'il vaut 0, alors le laser est bien au centre, sinon il faut déplacer la barrette à l'aide du servomoteur pour le ramener à 0.

Une étude expérimentale nous a permi de déterminer que la distance entre les deux extrémités de la barrette correspond à une différence de $250\,ms$ de temps haut dans les commandes du servomoteur. Donc pour une variation élémentaire de la position du laser sur la barrette $\Delta i = 1$, il faut faire varier la durée de temps haut de $3,9\,ms$. Donc connaissant le nombre i, on fait tourner le servomoteur avec une commande de $-\ 3,9\ i\ ms$.

Le minimum de temps pour que le servomoteur parcourt l'entièreté de la distance correspondant à la largeur de la barrette est de $200\,ms$. Cette durée incompressible sera donc le facteur temporel limitant de notre système.

Correcteur et asservissement numérique

Pour asservir le moteur, nous avons choisi un contrôleur proportionnel intégrateur. A chaque nouvelle boucle d'asservissement, nous avons calculé l'erreur, entre la consigne de position (0 dans notre cas) et la position mesurée comme expliqué précédemment. Nous calculons ensuite l'intégrale de l'erreur, qui est la somme des erreurs depuis le début de l'asservissement. Ensuite, à l'aide de la fonction `setmot2posPI(int err, int Ierr)`, nous calculons la consigne à donner au moteur, et nous l'envoyons. Pour le correcteur proportionnel intégral, cette consigne s'écrivait : `MOT_pw = MOT_pw - K_mot * err - K_int * Ierr` . Le gain proportionnel K_mot a été déterminé ci dessus, c'est 3,9 ms. Le gain K_int a été déterminé de manière empirique, nous l'avons fixé à 0,04.

De plus, nous avons tenté de mettre en place une fonctionnalité qui permet de chercher le spot laser lorsqu'il est en dehors du capteur. En effet, parfois le laser se déplaçait de manière trop importante, et il n'était plus dans le domaine de la barrette CCD. Nous avons alors réalisé une fonction de test, qui comparait le rapport entre l'intensité maximale et l'intensité minimale à une valeur seuil déterminée expérimentalement à 2,0.

Ensuite, lorsque l'on estimait que le signal n'était pas détecté, le programme ordonnait au servomoteur de déplacer la barrette CCD pour chercher à récupérer le signal. Le déplacement commandé était choisi de manière à ce que la nouvelle position de la barrette ait quelques pixels en commun avec la précédente, pour ne pas laisser de trou où on aurait pu oublier de détecter le laser. Le déplacement commandé alternait de sens pour parcourir l'ensemble des positions possibles, et il augmentait toujours en amplitude, de manière à parcourir d'abord la position -1, puis 1 puis -2, puis 2, puis -3 etc.

Le problème de cette méthode est que l'on force le système à diverger pour retrouver le signal, ainsi, nous avons borné ce déplacement de manière à ne pas endommager la crémaillère.

Bilan du projet

Partie technique / Avancement final

La partie acquisition a été un franc succès, car nous sommes parvenus à régler la chaîne d'acquisition de manière à obtenir une position précise du centre de la tache laser sur la barrette CCD. Nous avons ensuite pu expérimenter le fonctionnement du servomoteur contrôlant la position de la barrette CCD, et nous en avons déduit les bornes des commandes numériques à leur envoyer, ainsi que la fréquence maximale de commande numérique imposée par le servomoteur.

Grâce à ces deux étapes, nous avons réussi à fermer la boucle d'asservissement. Nous avons ensuite ajouté la fonctionnalité permettant de régler un correcteur proportionnel intégral, sous forme coefficients pouvant être réglés dans le code source de notre programme embarqué.

La fréquence de traitement numérique de notre programme étant limitée, notre asservissement n'atteignait pas la réponse à une vitesse linéaire du point laser de 10 cm/s. Une étape supplémentaire de développement aurait donc été de comprendre l'origine de cette limitation pour la pallier.

Une autre amélioration qui nous a été proposée, intéressante du point de vue fonctionnel, aurait été l'ajout d'une routine lançant un déplacement de la barrette le long de sa course complète lorsque la tache n'y est pas détectée, afin de retrouver sa position lorsqu'elle a été plus rapide que ce que l'asservissement pouvait suivre.

Enfin, dans notre démarche de prototypie centrée prioritairement sur la succès technique, nous n'avons pas participé à l'ergonomie de l'utilisateur en développant une interface graphique permettant le réglage en temps réel des coefficients du correcteur PI.

Retour d'expérience de l'équipe

Ce projet a été l'occasion de mettre en œuvre des notions d'Électronique, de Détection et d'Automatique, et ce dans le cadre d'un projet en autonomie. Ce cadre impose d'adopter des méthodes d'organisation plus formelles qu'un simple travail individuel. Nous avons ainsi suivi un formalisme d'ingénierie en axant notre développement suivant un schéma fonctionnel permettant la répartition du travail. D'un point de vue temporel, nous disposions de quatre heures hebdomadaires. Il nous a alors été conseillé de réfléchir en amont à l'organisation des différentes étapes du développement, par le biais d'un diagramme de Gantt. Notre diagramme figure en Annexe.

Annexe 1 : Diagramme de Gantt

Missions	Séance 1	Séance 2	Séance 3	Séance 4	Séance 5	Séance 6	Séance 7
	Phase 2 Réalisation prototype, rédaction documentation				**Phase 3 Audit circulaire**	**Phase 4 Finition du prototype**	
Recherche des documentations laser, capteurs et moteurs							
Création des outils de gestion de projet (journal de bord, schéma fonctionnel, diagramme de Gantt)							
Acquisition							
Optimiser la tache laser avec densités							
Capter : Nucléo,							
Convertir							
Traiter du signal : Première approche, moyennage							
Corriger le bruit lumineux avec des systèmes optiques (densité, lentilles, etc)							
Actionneur							
Découvrir le fonctionnement d'un moteur							
Commander un moteur en position : calibration de l'échelle des commandes							
Asservissement							
Etablir la fonction de transfert et le schéma-bloc de l'asservissement							
Mettre en place la boucle ouverte							
Mettre en place la boucle fermée avec correcteur PID							
Créer une interface graphique de contrôle du correcteur							

Annexe 2 : Code C++ du programme embarqué

```
// PROJET :  Asservissement en position d'un laser
// DATE : mars 2023 NOMS : Louis GONORD  Pierre MARULLO  Maxence VANDEWYNCKLE
// MATERIEL : CARTE NUCLEO-L476RG
//            CAPTEUR : barette CCD TSL201
// BUT DU PROJET : - Acquérir la position du laser sur la barrette CCD
//                 - Contrôler un servo-moteur
//                 - Asservir la position du laser à l'aide du servo-moteur

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

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

// DÉCLARATION DES ENTRÉES
AnalogIn    Entree(PA_4);       // port du signal d'entrée issu de la barrette CCD
AnalogIn    Entree_value(A4);   // valeur du signal d'entrée issu de la barrette CCD
InterruptIn StartSI(A1);       // start Impulse : pic indiquant le début de la lecture
des pixels CCD
InterruptIn StartCLK(A2);    // clock : horloge, suite de pics cadençant l'acquisition
des pixels

// DÉCLARATION DES SORTIES
PwmOut  CLK_out(D3);            // sortie de visualisation du signal d'horloge
"StartCLK"
PwmOut  SI_out(D5);             // sortie de visualisation du signal de Start Impulse
"StartSI"
PwmOut  MOT_out(D10);           // sortie de commande du moteur en position angulaire

// Débogage :
DigitalOut  debug_led(LED1); // la led sur la carte que l'on peut faire clignoter
Timer t; // pour paramétrer

// DÉCLARATION DES PARAMÈTRES DES SIGNAUX DE SORTIE
// Pour la lecture de la barette CCD
int CLK_period=30; //us
int SI_period=2100; //us
double CLK_rc=0.5;  // temps haut = temps bas
int SI_pw=15; //us

//Pour la commande du moteur
int MOT_period=20; //ms      | essayer d'ajuster les périodes avec celles de lecture
int MOT_pw=1500; //us        | le pulsewidth du moteur code la position angulaire,
donc le déplacement transverse du capteur
//                          | On initialise à une position qui permet de se déplacer
des deux côtés
```

```
// DÉCLARATION DES VARIABLES GLOBALES
int TAB[64];   //Le tableau qui récupère les valeurs des 64 PHD
int i=0;    //Compteur qui permet de switcher d'une PHD à une autre

int pos_laser_cons = 0; // Consigne de la position du laser sur la barette CCD (entre
-31 et 32)
int pos_laser; // Position du laser sur la barette CCD (entre -31 et 32)

int err_pos; // l'erreur sur la position
int Ierr = 0; // l'intégrale de l'erreur

double T_boucle=0.06; // le temps que l'on passe dans chaque boucle d'asservissement

int compt_boucle = 0 ; // compteur du nombre de boucles d'asservissement
int compt_perdu = 0 ;  // sous compteur qui compte le nombre de boucles
d'asservissement où on a perdu le signal

double K_mot = 3.9; // gain que l'on applique au moteur
//      calcul du gain théorique:  | i_max = 64     i=1
//                                 | N_max = 250    N=250/64 = 3.9
double K_int = 0.04 ; // Gain du facteur intégratif
double seuil_laser = 2.0 ; // Facteur entre le max et le min d'intensité

// Déclaration du ticker qui crée l'interruption
Ticker ticker_regul;

// DÉCLARATION DES FONCTIONS UTILISÉES
void fonctionSI(void); // Se déclenche à chaque front montant du signal SI
void fonctionCLK(void);// Se déclenche à chaque front montant du signal CLK
void asservissement(void); // Se déclenche régulièrement, grace au ticker ticker_regul
int recherchemax(int tab[64]); // Permet de trouver l'indice de la phd où l'intensité
est maximale
void recuptab(int tab[]);//Non utilisé, sert à envoyer les valeurs du tableau à matlab
void setmot2posP(int err); // Calcule et envoie la commande au servomoteur
(Correcteur Proportionnel)
void setmot2posPI(int err, int Ierr);  // Calcule et envoie la commande au servomoteur
(Correcteur Proportionnel Intégrateur)
bool isinrange(int tab[64], double R_seuil); // Teste si le signal est détecté ou non.
double mean(int tab[64]);// Calcule la moyenne d'un tableau de 64 valeurs NON UTILISÉE
double minimum(int tab[64]); // Calcule le minimum d'un tableau de 64 valeurs


//#####################         DEBUT DU MAIN          ########################//
int main(){
   pc.baud(115200); // Initialisation du port USB.
   // Uniquement pour afficher sur Teraterm, servirait pour l'interface graphique

   // Initialisation des sorties pour alimenter le capteur et pour commander le moteur
   CLK_out.period_us(CLK_period);
   CLK_out.write(CLK_rc);
```

```
   SI_out.period_us(SI_period);
   SI_out.pulsewidth_us(SI_pw);

   MOT_out.period_ms(MOT_period);
   MOT_out.pulsewidth_us(MOT_pw);


   // Déclaration des interruptions et des fonctions associées pour la mesure
   StartSI.rise(&fonctionSI);   //Détection de front montant pour le signal SI
   StartCLK.rise(&fonctionCLK); //Détection de front descendant pour le signal CLK

   // Déclaration du ticker
   ticker_regul.attach(&asservissement,T_boucle);

   while(1){ }
}
//######################         FIN DU MAIN          ########################//

/////////////     FONCTIONS D'INTERRUPTIONS POUR LA LECTURE     ///////////////

void fonctionSI(){
   i=0; // dès que l'on reçoit le signal SI, on peut rentrer dans la boucle de la
fonctionCLK, pour lire les valeurs des photodiodes.
}

void fonctionCLK(){ // Effectue la lecture de manière synchronisée sur le CLK
   // C'est ici que se crée le tableau des valeurs des Photodiodes
   if(i < 64){
       TAB[i]=Entree_value.read_u16()  >> 4;  // LA LIGNE DURE 24~25 US
       i++;
   }
}

/////////////      FONCTION D'INTERRUPTION D'ASSERVISSEMENT     ////////////////
void asservissement(){
 //Fonction d'interruption qui, à chaque boucle, recherche la postition du laser,
calcule la commande moteur et pilote le moteur
   compt_boucle ++ ; //Un compteur qui compte le nombre de boucles d'asservissement
NON UTILISÉ
   if (isinrange(TAB,seuil_laser)){
       // On est sûr de faire l'asservissement sur le laser et pas sur un bruit de
mesure
       pos_laser = recherchemax(TAB);
       debug_led = 1; // Pour le debugage, pour savoir le résultat du test
       Ierr = Ierr + err_pos ; // Intégrale de l'erreur
       err_pos = pos_laser_cons - pos_laser;
       setmot2posPI(err_pos,Ierr); // Si on utilise le correcteur Proportionnel
Intégrateur
       //setmot2posP(err_pos) ;    // Si on utilise le correcteur Proportionnel
```

```
        compt_perdu = 0; // Le capteur n'est plus perdu : il a trouvé la tache
    }

    else{
        debug_led = 0 ; // Pour le debugage, pour savoir le résultat du test
        // On se déplace alternativement de droite à gauche toujours un peu plus,
Attention la méthode diverge !!
        if (compt_perdu%2==1){
            if ((MOT_pw <2000) & (MOT_pw>1000)) { // Pour empêcher au moteur de
diverger
            MOT_pw = MOT_pw + 10 * compt_perdu;
            }
        }
        else{
            if ((MOT_pw <2000) & (MOT_pw>1000)){ // Pour empêcher au moteur de diverger
            MOT_pw = MOT_pw - 10 * compt_perdu;
            // La valeur de 10 a été choisie pour que au passage suivant, le capteur
            //se soit décalé de moins d'une largeur de capteur
            } // On borne le décalage à 1000 - 2000
        }
        MOT_out.pulsewidth_us(MOT_pw) ;
        compt_perdu ++;
    }

    //pc.printf("compt boucle = %d , compt perdu = %d , PW = %d , position : %d
\n",compt_boucle,compt_perdu,MOT_pw,pos_laser); // Pour le débogage

}

void recuptab(int tab[64]){ // Juste pour l'affichage en fait, mais réinitialise le
tableau aussi
    int l=0 ;  //Compteur interne à la fonction
    int maximum=recherchemax(tab) ;
    for(l=0;l<64;l++){
            // pc.printf("%d ", tab[l]);
           //     pc.putc(tab[l]);
           //     tab[l]=0 ;
    }
    //pc.printf("le max est %d ", maximum);
}

//// Fonction recherche de maximum ////
//// Recoit un tableau de 64 valeurs, renvoie l'indice du centre de la bosse ////
int recherchemax(int tab[64]){
    /*
    Fonction recherche du maximum, reçoit un tableau (tab) de 64 valeurs,
    renvoie l'indice du centre de la bosse (ans) entre -31 et 32
    Usage :
    int indice_max= recherchemax(tab);
    */
```

```
   int l=0; // Compteur qui parcourt le tableau lors de la recherche du max
   int ans=l ; // On initialise ans au premier pixel d'indice 0.
   int sum=0 ; // Somme des valeurs des pixels pour le calcul du barycentre
   for(l=0;l<64;l++){
       /*
       // Version primitive, on cherche juste la valeur max du tableau
       if (tab[l]>tab[ans]){ans=l;}
       */
       // Autre solution : recherche du barycentre :
       ans=ans+tab[l]*l ;
       sum=sum+tab[l] ;
       }
   ans=ans/sum ;
   ans=ans-31 ;

   return ans ;
}

void setmot2posP(int err){

   // CORRECTEUR PROPORTIONNEL
   MOT_pw = MOT_pw - K_mot * err ; // Le signe - dépend de l'orientation du
servomoteur

   MOT_out.pulsewidth_us(MOT_pw) ;

}


void setmot2posPI(int err, int Ierr){

   // CORRECTEUR PROPORTIONNEL + INTEGRATEUR

   MOT_pw = MOT_pw - K_mot * err - K_int * Ierr ;

   MOT_out.pulsewidth_us(MOT_pw) ;

}


bool isinrange( int tab[64], double R_seuil){

   /* Teste si la tache laser est sortie de la barrette CCD.
   Renvoie TRUE si le laser est encore sur la barette, et FALSE sinon.

   Calcule le rapport entre l'intensité max et l'intensité minimale.

   Usage:  bool test = isinrange(TAB,seuil_laser) ;
   */
```

```
   int i_max = recherchemax(tab) ;
   bool test = ( tab[i_max] / minimum(tab) ) >= R_seuil ;

   return test;
}


double mean(int tab[64]){  //Pas utilisée

   /* Renvoie la moyenne d'un tableau de 64 entiers.

   Usage:  double moy = mean(TAB) ;
   */

   int compt = 0 ;
   double mean = 0 ;

   for (compt=0;compt<64;compt++){mean += tab[compt];}

   mean = mean / 64.0 ;
   return mean;
}

double minimum(int tab[64]){

   /* Renvoie la moyenne d'un tableau de 64 entiers.

   Usage:  double min = moyenne(TAB) ;
   */

   int compt = 0 ;
   double min = tab[0] ;

   for (compt=0;compt<64;compt++){if (tab[compt]<min){min = tab[compt];};}

   return min;
}


//############################  FIN DU CODE  ############################//
```