08 avril 2022

RAPPORT TECHNIQUE

Banc optique pour asservissement de la position d'un laser

Développement de Solutions Electroniques

AUTEURS :

Mohamed Amine Adnani
Sarah Cherif
Hector Galle
Islem Ketata
Nesin Mehmed

ENCADRANTS :

Fabien Adam
Aliénor De la Gorce
Christophe Hecquet
Julien Villemejane

PERIODE DU PROJET :

Du 25 janvier au 29 mars

Table des matières

Préambule

La société SOLEC a fait appel à notre équipe afin de développer un banc optique pour pointer un faisceau laser de manière précise sur une cible. Pour cela, nous devrons asservir la position de notre cible le long d'un segment. Entre autres, nous devrons commander un servomoteur via une carte Nucléo.

La solution que nous proposons est d'asservir la position d'une photodiode le long d'un rail de 15 cm. Cette photodiode devra être capable de retrouver le faisceau lumineux du laser et de le positionner en un point précis du capteur.

I. Matériels à disposition et schéma bloc du système

Matériels utilisés

Pour mener à bien notre projet, nous avons à notre disposition un laser, une photodiode (capteur), une carte nucléo (pour commander notre système) ainsi que des outils électroniques tels que des fils et une boite électronique afin de relier tout le système.

Laser :

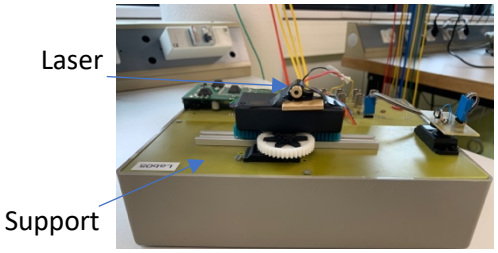


Figure 1 Photographie du laser du projet

Le laser qui sera utiliser est un laser de 6V, alimenté par un générateur. Lors de nos expérimentations, le laser restera fixe, seul le capteur se déplacera.

Capteur :

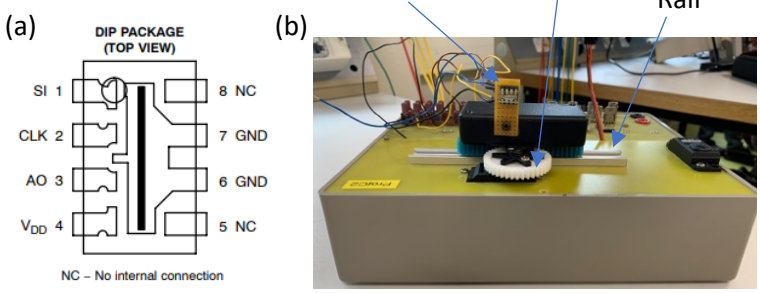
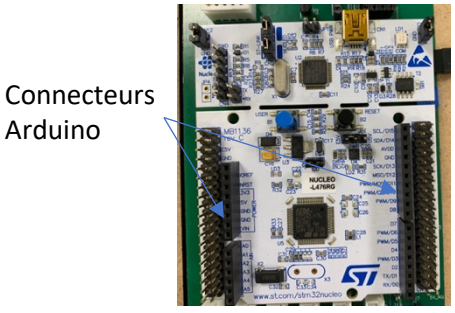


Figure 2 (a) sortie du capteur ; (b) photographie du capteur du projet

La photodiode est un capteur linéaire *TSL201R-LF* composé de 64 pixels, ayant les sorties représentées dans la figure 2a. Ce capteur se déplace sur un rail de 15cm à l'aide d'un servomoteur contrôlé par la carte Nucléo.

Carte Nucléo



La carte Nucléo *STM32* utilisée est une carte de développement permettant de simuler des systèmes embarqués de prototypes. Elle fonctionne en parallèles avec le logiciel *MBED Compiler* contenant les commandes à appliquer au système.

Cette carte est reliée à l'ordinateur de travail mais également au montage expérimental via des connecteurs Arduino.

Montage du système

Ainsi, à l'aide du matériel cité, nous avons pu établir le système suivant :

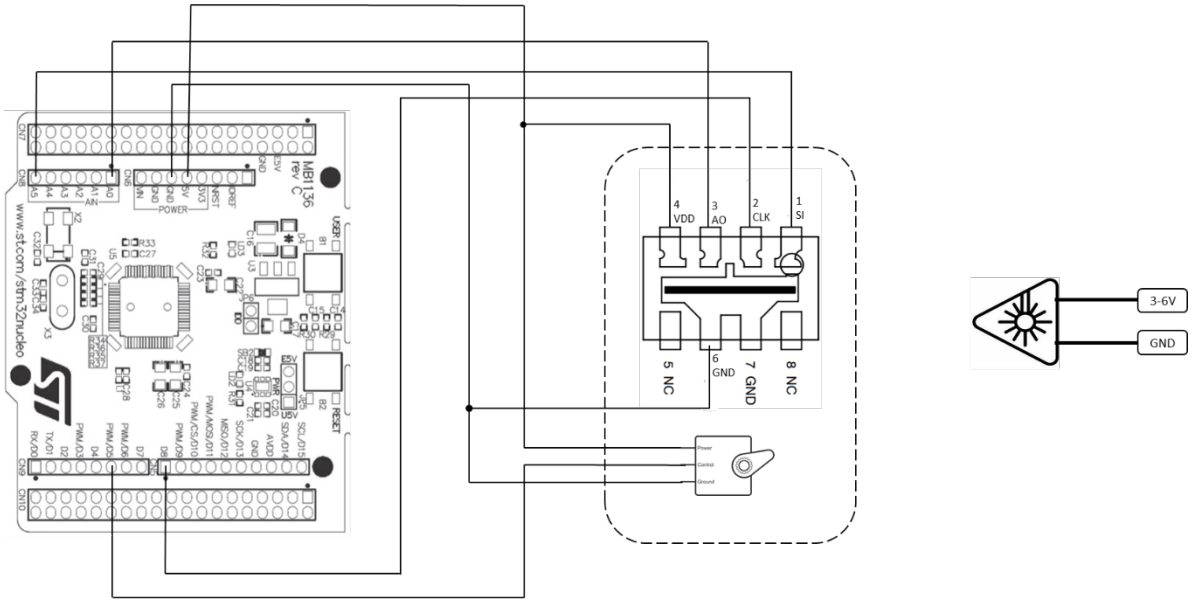


Figure 3 Schéma fonctionnel du banc de test réalisé pour l'asservissement de la position du laser

Le schéma bloc ci-dessus, nous montre tous les branchements de notre montage. Un laser alimenté en externe, un bloc contenant le capteur ainsi que le servomoteur connecté à la carte NUCLEO pour la génération des signaux et leurs réceptions et enfin la commande sur le servomoteur pour effectuer l'asservissement.

II. Cahier des charges

Le système que nous proposons devra être performant et devra satisfaire quelques contraintes. Un récapitulatif du cahier des charges est représenté dans le tableau 1.

Fonctions	Contraintes	Performances
Positionner le capteur de manière à récupérer l'information issue du faisceau laser		Maintenir la position d'un faisceau laser de manière précise en un point précis d'une photodiode
Déplacer la photodiode le long de l'axe	Servomoteurs classiques	Exploiter la vitesse maximale du servomoteur pour satisfaire le critère de vitesse de 10 cm/s
Concevoir un système asservi	Asservissement numérique Modifier le coefficient du correcteur en temps réel	Erreur de pointage très faible (voire nulle)
Ergonomie du système		Facilité l'interaction Homme-Machine afin de permettre aux utilisateurs du système de pouvoir facilement le contrôler

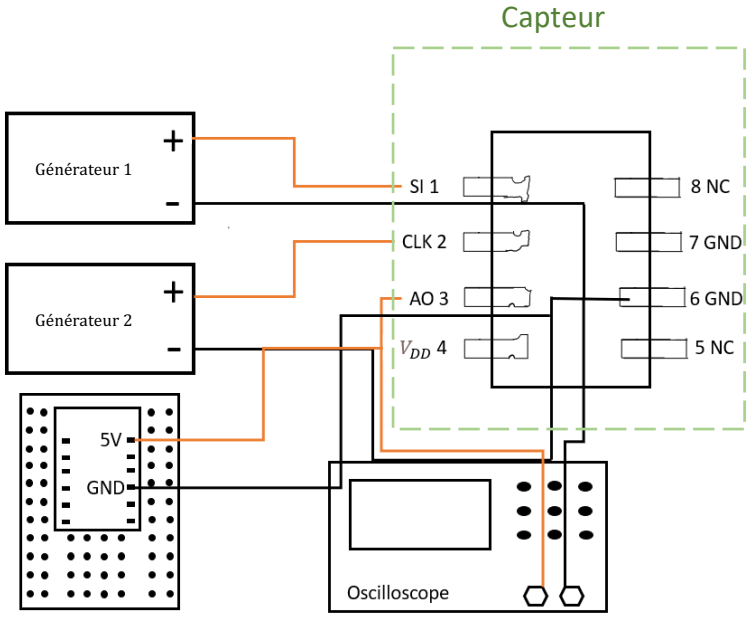
Tableau 1 Tableau récapitulatifs du cahier des charges

III. Démarche technique

Notre système a été conçu en deux parties : câblage et code informatique.

Câblage

Dans un premier temps, nous avons commencés par envoyer via les générateurs deux signaux bien distincts. L'un d'entre eux était un signal créneaux de fréquence $30\,kHz$ qui modélisait le signal « Clock » CLK, l'autre signal était un signal impulsionnel dont la durée d'impulsion était égale à $20\,\mu s$ et dont la fréquence était plus de soixante-quatre fois inférieure à la fréquence du signal Clock. Le schéma du montage est représenté en figure 4.



Ainsi, nous alimentons les bornes SI 1 et CLK 2 par les deux signaux issus des générateurs. De plus, on alimente le capteur avec le 5V de la carte nucléo afin d'assurer le fonctionnement de celui-ci. Enfin, on visualise à l'oscilloscope la sortie du capteur, qui peut s'obtenir en reliant l'oscilloscope à la sortie A0 3 du capteur. On a choisi de brancher la masse sur l'une des deux sorties GND du capteur, de manière complètement aléatoire.

Le choix des signaux issus des générateurs doit satisfaire la logique suivante : lorsque la sortie SI est à 1 (représentant le signal impulsionnel), un cycle de sortie de pixel est alors initié avec le signal CLK qui se répète 64 fois avant une nouvelle impulsion, représentant les 64 photodiodes du capteur (cf figure 8 en annexe).
Voici les signaux que nous obtenons à l'oscilloscope sur la figure 5 :

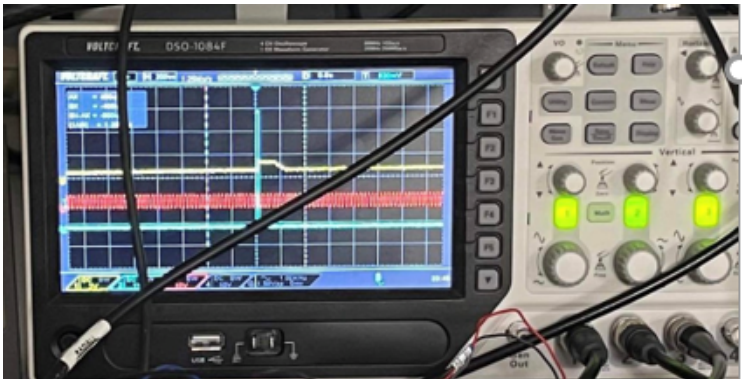


Figure 5 : Signaux issus des générateurs et obtenu sur l'oscilloscope. En rouge le signal "Clock", en bleu le signal "SI" et en jaune ce qui est reçu sur le capteur

La figure 5 ne reflète pas très bien les résultats que nous avions eu, mais nous pouvions distinguer de manière assez claire un pic de la courbe jaune, c'est-à-dire le signal obtenu en sortie du capteur. Ce signal était situé entre deux pics bleus (impulsions) qui représentait les délimitations du capteur. Ainsi, on a pu s'apercevoir qu'en déplaçant le signal laser le long du capteur la position du maximum de la courbe jaune se déplaçait le long des deux impulsions.

Une fois cette première manipulation réussie, il est alors nécessaire de réaliser la même chose en modifiant les sources des signaux SI 1 et CLK 2. En effet, comme expliqué dans la partie « *Code* » nous avons envoyé des signaux numériques de type PWM via la carte nucléo, mais le montage était identique à celui de la figure 4 en envoyant les signaux SI 1 et CLK 2 via les sorties D3 et D8 de la carte nucléo.

Enfin, on effectue le câblage du servomoteur en respectant le schéma de la figure 6 :

Dans un premier temps, il est nécessaire de câbler le servomoteur seul pour voir si le montage fonctionne correctement et si tout est en ordre. Pour cela, on commence par alimenter le servomoteur par une source de tension indépendante en reliant les sorties VCC et GND aux bornes + et – de la source. Ensuite, si on veut commander le servomoteur avec la carte, il faut relier le dernier câble à la sortie ou on envoie les commandes, D10 dans le cas du schéma.

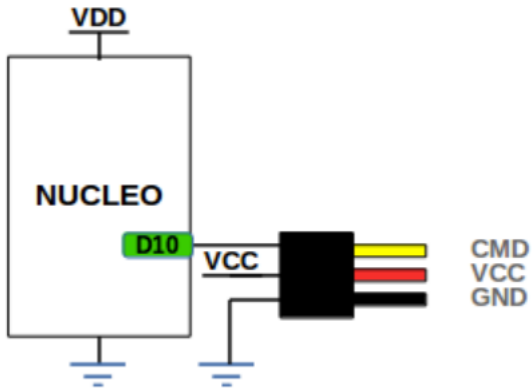


Figure 6 : Schéma représentant le montage du servomoteur seul

Lorsque toutes nos montages fonctionnaient correctement et indépendamment les unes des autres, on a câblé les montages entre eux pour obtenir le montage final. Dans notre cas, nous avons fait cela en trois étapes comme décrit dans les paragraphes ci-dessus. La dernière étape était de faire fonctionner le servomoteur au même moment que le montage précédant. La partie câblage étant effectuée il faut à présent présenter le code qui nous a permis d'envoyer les bonnes commandes aux bonnes sorties, afin d'obtenir les performances attendues par le système.

Code

La première étape est de récupérer les deux signaux délivrés par la photodiode, un premier signal (qui une période environ 64 fois (nombre de pixel) plus grande que le deuxième signal.

On commence la récupération à front montant du premier signal : on récupère le signal délivré par chaque pixel des 64 constituant la diode, ce signal correspond à l'intensité lumineuse récupérée par chaque pixel. Puis on stocke ses valeurs dans un tableau de taille 64 et on détermine le maximum de ce tableau. Si cette valeur max est supérieure à 300 (une valeur seuil qu'on a fixé), cela veut dire que le faisceau laser tape sur les 64 pixels, ainsi on note l'indice de cette valeur maximale dans le tableau et en fonction de l'écart entre cet indice et 32 (qui correspond à l'indice du pixel central), on envoie une consigne au servomoteur pour qu'il agisse sur la photodiode et la translate de façon à obtenir l'intensité maximal au pixel central.

Dans le cas où le faisceau laser est hors du champ de la photodiode, on envoie une consigne au servomoteur pour qu'il se mette en marche et que la photodiode commence à chercher le faisceau laser. On a également fixé pour le servomoteur deux valeurs limites qui correspondent aux valeurs du bout de course pour que lorsque le capteur arrive au bord, il se mette à aller dans l'autre sens afin de continuer la recherche du faisceau.

Dans les deux cas cités, on a intégré dans notre code un correcteur à action proportionnelle - intégrale qui permet dans le cas idéal (qui n'est malheureusement pas le cas de notre prototype) de respecter les critères de stabilité et de précision (on a tout de même obtenu de la stabilité lors d'une séance mais perdue à la séance suivante).

Toute ces étapes sont incluses dans une boucle *while (1)* utilisant un tableau à 64 valeurs qui se remplit à nouveau à chaque début de boucle pour mettre à jour les intensités sur les pixels. Ceci permet au système de fonctionner en continu.

Pour plus de détails sur ce code, veuillez se référer au code complet minutieusement commenté donné en annexe.

IV. Prototype final

Performances obtenues

Nous avons bien réussi à atteindre la mission principale en asservissant la photodiode de manière efficace.

En effet, si le laser n'arrivait pas initialement sur la photodiode, la photodiode recherchait bien le faisceau laser jusqu'à le trouver.

Après avoir trouvé le signal laser, la photodiode se positionnait de tels sorte que le faisceau se stabilise sur son bord droit comme on peut l'observer sur la photo ci-dessous après alignement :

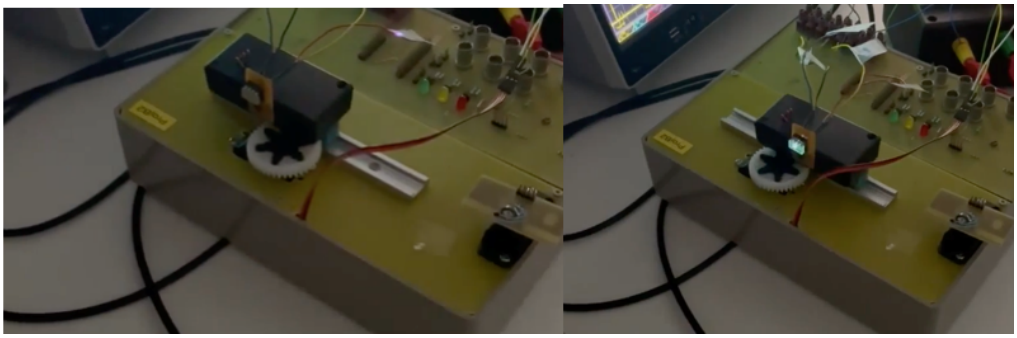


Figure 7 : Photodiode en position initiale (gauche) puis après alignement (droite)

Nous avons pu constater que notre asservissement permettait bien à la photodiode de parcourir toutes les positions offertes par le rail de déplacement. De plus, la stabilisation sur l'endroit souhaité de la photodiode se faisait toujours très rapidement ($\sim 1s$).

Cependant, nous avons rencontré des problèmes de stabilisation au cours du temps de la position du capteur, une fois le faisceau retrouvé (typiquement après une minute ou deux) qui seront détaillés dans la partie ci-dessous.

Retour au cahier des charges

Comme nous l'avons évoqué un peu plus haut, la fonction principale du cahier des charges, qui était de « Positionner le capteur de manière à récupérer l'information issue du faisceau laser » à bien était atteinte. En effet, comme nous pouvons le voir sur la vidéo de démonstration jointe au dossier, lorsque notre capteur ne recevait pas le signal laser, il essayait de le retrouver avec un mouvement de translation le long du rail, en exploitant toute la longueur du rail. De plus, lorsque le capteur retrouve le flux lumineux il se place immédiatement à la position qui lui permet d'avoir le faisceau laser au bord en haut à sa droite.

Avec la validation de cette fonction principale nous avons également validés les deux fonctions secondaires qui sont « Déplacer la photodiode le long de l'axe » et « Concevoir un système asservi ».

Cependant, des critiques peuvent être faits sur les performances que nous avons réussies à obtenir.

Tout d'abord, le critère qui impose de suivre des mouvements de l'ordre de $10\ cm/s$ ne peut pas être satisfait avec les servomoteurs que nous avions. Ces derniers nous permettent d'avoir au maximum une vitesse d'environ $4\ cm/s$.

Mais encore, la stabilité de notre système n'est pas optimale, au bout de plusieurs minutes de fonctionnement le système commence à osciller autour de la position souhaitée. Cette contrainte n'a malheureusement pas pu être réglée au cours des séances par manque de temps. Nous pensons que cela doit être dû à une erreur cumulée au cours du temps, qui rend donc le système moins efficace au bout de quelques minutes de fonctionnement.

Enfin, nous avons pris soin de réaliser un montage et un système en essayant d'optimiser l'ergonomie de celui-ci. En effet, nous avons mis en place un système facile à réaliser et à comprendre juste en lisant attentivement le rapport technique fourni.

Axes d'améliorations :

Pour avoir un système plus stable, tant sur le plan technique que sur le plan fonctionnel, nous avons réfléchit à une autre solution. Cette dernière consiste à ajouter un miroir à notre système de tel sorte que le capteur ainsi que le laser soient fixes et l'asservissement sera pour le miroir (soit en termes de rotation ou translation). Cette solution nous permet d'avoir un agissement sur deux directions et nous permet d'avoir plus de paramètres à contrôler pour réduire les instabilités mais aussi dans le but de réduire les limitations liées à la distance d'asservissement.

V. Annexe

Code informatique

```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */


 /* Team: Hector Galle, Mohamed Amine Adnani, Nesin Mehmed, Sarah Cherif, Islem Ketata */

#include "mbed.h"
#include "platform/mbed_thread.h"



int T_integration = 100;                //période du créneau en ms qui sert a lire l'information sur chaque photodiode
int a = 64;                             //taille du tableau ou l'on stocke chaque valeur de cellule de la photodiode
int val_tableau[64];                    //tableau ou l'on stocke les valeurs de chaque cellule de la photodiode
char l;                                 //déclaration de la chaine de caractere l
int indiceMax;                          //indice du maximum
int Ecart_pixel ;                       //ecart au centre en nombre de pixels
double Ecart_T;                         //écart en temps pour gouverner le servomteur
int moteur = 1500;                      //valeur d'initialisation pour le servomoteur
int sens_servo=1;                       //permet de definir dans quel sens va le moteur (sens_servo=1 on ajoute sens_servo=-1 on enleve)
int debeug=0;                           //entier définit pour le débeugage
double gain=1/12;                       //initialisation du gain pour faire le lien entre ecart en pixel et ecart en temps
double gain_integrateur = 1;  //initialisation du gain du correcteur intégrateur


PwmOut CLK(D3);
InterruptIn CLK_IN(D8);           //definition de l'entree D8 comme entree d'interruption
DigitalOut SI(A5);                //définition de la sortie A5
PwmOut servo_mot(D5);             //déclaration des sorties PWM

Serial pc(USBTX,USBRX);
AnalogIn capteur(A0);

void acquisition (void);                        //declaration de la fonction d'interruption
int main (){
    pc.baud(115200);                            //débit des échanges d'information entre l'ordinateur et la carte nucléo
    CLK_IN.fall(&acquisition);                  //commence l'acquisition sur un front descendant
    CLK.period_us(T_integration);               //inverse de la frequence de CLK en microseconde
    CLK.write (0.5);                            //on impose la valeur 0.5 a CLK
    servo_mot.period_ms(20);                    //inverse de la fréquence du servomoteur
    servo_mot.pulsewidth_us(moteur);            //"temps haut" du servomoteur
    a=0;
            SI = 1;
        wait_us(1000);
        SI = 0;

    while (1){                                  //boucle infinie
        l = pc.getc();                          //si pc a bien recu une première information



        if (l == '+') {
            T_integration = T_integration +10;                      //choix d'augmenter le temps d'intégration via '+'
            CLK.period_us(T_integration);
        }
        else if (l == '-') {
            T_integration = T_integration - 10;                     //choix de diminuer le temps d'intégration via '-'
            CLK.period_us(T_integration);
        }
        if (l == 'a'){
            gain = gain + 0.01 ;                                    // Augmentation du gain en temps réel via 'a'
        }
        else if (l == 'b'){
            gain = gain - 0.01 ;                                    // Diminution du gain en temps réel via 'b'
        }
    }
}

void acquisition (void){

    if (a <64){                                             //tant que toutes les cellules du capteur ne sont pas lues
        double x = capteur.read()*1000.0;                   //on multiplie la valeur de chaque cellule du capteur par 1000
        val_tableau[a]=(int)x;                              //remplissage de val_tableau avec les valeurs du capteur
        a++;                                                 //on passe à la lecture de la cellule d'après
    }
    if(a ==64){                                              //on passe au remplissage d'un nouveau tableau

        int max=val_tableau[0] ;                             //on initialise de nouveau la nouvelle valeur du max
        indiceMax=0;
            for (int i =0;i<63;i++) {
               if (val_tableau[i]>=max) {
                    max=val_tableau[i];                      //valeur du max du tableau
                    indiceMax=i;                             //valeur de l'indice du max
                }
            }

        if((max>300)){                                       //si le faisceau laser tape sur la photodiode
            Ecart_pixel=indiceMax-31 ;                       //ecart par rapport au pixel central
            sens_servo=1 ;
            if(abs(Ecart_pixel)>2){                                                 //si l'ecart au centre est plus grand que deux pixels
                Ecart_T=int(6*gain)*Ecart_pixel ;                                   //lien entre ecart en temps et ecart en pixels
                moteur=gain_integrateur*moteur+int(Ecart_T);                        //nouvelle valeur du rapport cyclique
                servo_mot.pulsewidth_us(moteur);                                    //diminution de l'écart au centre

            }
        }
        else {                      /*Si le laser ne tape pas sur la photodiode,on distingue les cas selon si on doit faire bouger
                                     le moteur vers la gauche ou la droite ( 1200 et 1800 sont deux valeurs seuils )*/
            if ((moteur<1800)&(sens_servo==1)){
                moteur=gain_integrateur*moteur+int(gain*64);
                servo_mot.pulsewidth_us(moteur);
            }
            if((moteur>=1800)){
                moteur=gain_integrateur*moteur-int(gain*64);
                servo_mot.pulsewidth_us(moteur);
                sens_servo=-1;
            }
            if ((moteur>1200)&(sens_servo==-1)){
                moteur=gain_integrateur*moteur-int(gain*64);
                servo_mot.pulsewidth_us(moteur);
            }
            if((moteur<=1200)){
                moteur=gain_integrateur*moteur+int(gain*64);
                servo_mot.pulsewidth_us(moteur);
                sens_servo=1;
            }

        }


            wait_us(T_integration);
            SI = 1;                                          //définition du créneau via SI

             wait_us(T_integration);
            SI = 0;
            a = 0;                                          //remplissage d'un nouveau tableau

    }
}
```

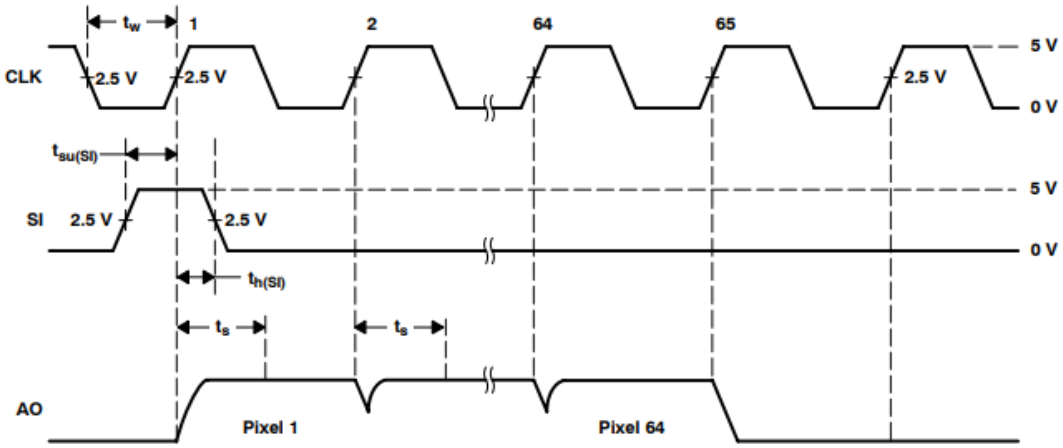


Figure 2. Operational Waveforms

Figure 8 Schéma représentant les signaux impulsionnels et créneaux modélisant notre système, pour un capteur à 64 pixels