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

#include "mbed.h"

//Constantes du programme

#define T_clock 0.00005 //durée d'un cycle d'horloge = 50 µs
#define Nb_cycles 64

//Entrées et sorties

Serial pc(USBTX, USBRX);

//Entrées et sorties pour le signal d'horloge
PwmOut clock_out(D10);
InterruptIn clock_in(D12);

DigitalOut top_out(D11);

DigitalOut fin_lecture(D9);
InterruptIn debuter_traitement(PA_9);

//Entrées et sorties pour la mesure de la position laser
AnalogIn barrette(A0);
AnalogOut test_tab(A2); //test du bon remplissage du tableau

//Controle du servomoteur
PwmOut servomoteur(D5);

//Variables

//Variables pour la mesure de la position laser
double tab_barrette[Nb_cycles];
int N_debut;            //Numéro du front montant pour lequel v_barrette>3V i.e. la barrette rencontre le spot la période d'horloge N_debut
int N_fin;              //Numéro du front montant pour lequel v_barrette<3V i.e. la barrette cesse de lire le spot à la période d'horloge N_fin
int N_pos;              //Pixel au centre du spot laser
float t_max;            //Instant où est lu le pixel central du spot laser


//Variables pour le signal d'horloge
int N_clock;          //Entier codant les périodes d'horloges. Compris entre 0 et 66

//Variables liées à l'asservisement
double Kp,Ki;
double T_boucle=64*T_clock;     //Durée d'une lecture complète de la barrette
double KiT=Ki*T_boucle;
int erreur,Ierreur;     //Erreur et erreur intégrale par rapport au centre (32)
int t_commande;      //Durée du temps du haut controlant la position du servomoteur [µs]
double pas,rayon;


//Prototypes

//Prototypes pour le signal d'horloge
void cpt_clock(void);

//Prototype pour l'asservissement
void traitement(void);
void commande_PI(void);

//Programme principal
int main(){

    int N_clock=1;

    double Kp=1;
    double Ki=0;
    int t_commande=1500;        //temps haut correspondant à la position 0° du servomoteur
    double pas=0.127;       //pas du réseau de photodiodes [mm]
    double rayon=19;       //rayon de la roue dentée [mm]



    //Génération du signal d'horloge
    clock_out.period(T_clock);
    clock_out.write(0.5);

    //Compteur de l'horloge
    clock_in.fall(&cpt_clock);


    //Analyse des mesures
    debuter_traitement.rise(&traitement);


    //Initialisation du servomoteur
    servomoteur.period_ms(20); //Période d'horloge fixée à 20 ms = 20 000 µs
    servomoteur.pulsewidth_us(1500);


```

```
    while (1) {
    };
}



//Definitions des fonctions

//fonctions pour le signal d'horloge
void cpt_clock(void){

     //Processus de traitement
     if (N_clock==Nb_cycles+1){
        fin_lecture=1;
        N_clock+=1;
        }
    //Génération de Top
    else if (N_clock==Nb_cycles+2){
        top_out=1;
        N_clock+=1;
        }
    else if (N_clock==Nb_cycles+3){
        top_out=0;
        N_clock=1;
        }

    else {

        if (N_clock<Nb_cycles){
            tab_barrette[N_clock-1]=barrette.read();
        }

        N_clock+=1;
        }

}

void traitement(void){

    int i=0;        //indice de la boucle for
    int spot_on=0;      //variable numérique : 1 si le spot est en train d'être mesuré, 0 sinon
    int spot_mesure=0;      //variable numérique : 1 si une mesure du spot est en cours, 0 si le spot a déja été mesuré (sur un cycle d'horloge
    int sature;     //variable numérique : 1 si le pixel sature i.e. si la tension est>3V.

    int N_debut=0;
    int N_fin=0;


    for(i=0;i<Nb_cycles;i++){

        test_tab.write(tab_barrette[i]); //test

        if (tab_barrette[i]>0.9){
            sature=1;
            }

        else {
            sature=0;
            }

        if((spot_mesure==0)&&(sature==1)){
            N_debut=i+1;
            spot_mesure=1;
            spot_on=1;
        }
        else if((spot_on==1)&&(sature==0)){
            N_fin=i;
            spot_on=0;
        }
        else if((i==Nb_cycles-1)&&(sature==1)){
            N_fin=Nb_cycles;
            spot_on=0;
        }

    }


    N_pos=(N_debut+N_fin)/2;


    // pc.printf("%d\n\r",N_pos);


    commande_PI();

    fin_lecture=0;

}

void commande_PI(void){

    double commande,angle_commande;


    erreur=32-N_pos;
```

```
189
190     pc.printf("%d\n\r",erreur);
191
192     Ierreur=Ierreur+erreur;
193
194     commande=(double)(Kp*erreur+KiT*Ierreur); //Commande en pixels
195
196     angle_commande=pas/rayon*commande; //Commande en angle
197
198     t_commande+=1500+(int)(angle_commande/180*1000);     //Commande en temps [µs] : modification du temps haut lu par le servomoteur
199
200
201     /*
202     if((t_commande>2000)||(t_commande<1000)){ //Si la commande en angle dépasse +90° et -90°, on revient à l'origine
203         servomoteur.pulsewidth_us(1500);
204         t_commande=1500;
205         wait(0.05);
206         }
207     else{
208
209         servomoteur.pulsewidth_us(t_commande);
210         wait(0.05);
211     }*/
212
213 }
214
```