File "/tri_banc/main.cpp" printed from os.mbed.com on 4/5/2022

```
/*
Projet PROTIS 2021-2022 - SOLEC - Vision Industrielle - Jeanne Jolivet ;
Sacha Mugnier ; Damien Rinnert ; Haoyu Tong

Ce script est celui à implémenter sur une carte Nucléo - L476RG. Son
fonctionnement est décrit ci-après.

Une de ses fonctionnalités consiste à alimenter les différents actionneurs du
système : le détecteur, le moteur pas à pas du convoyeur et les servomoteurs.

Un détecteur de présence permet d'ordonner la prise d'une photo à MatLab via la
laison série RS232. En retour, nous recevons la couleur de l'objet détecté.
Lorsque la carte Nucléo reçoit l'information de la couleur via la liaison série
RS232,la gestion en temps réel de la carte permet d'activer le piston
correspondant au bon moment (connaissant la vitesse du convoyeur).

Un sysème de deux tableaux complémentaires permet de gérer jusqu'à 6 objects
détectés en simultané. Le premier stocke l'information de la couleur, et le
second celui du temps au bout duquel le piston correspondant doit être activé.

Dans tous nos codes nous utilisons la correspondance MatLab - MBED suivante :
'Y'-0 ; 'R'-1 ; 'G'-2 ; 'B'-3
*/

// TELECHARGEMENT DES BIBLIOTHEQUES ADAPTEES
#include "mbed.h"
#include <math.h>

// DECLARATION DES ENTREES - SORTIES
// - Liaison série RS232 entre MatLab et MBED
Serial      rs232(USBTX, USBRX);
// - Signaux d'alimentation du moteur pas à pas du convoyeur
PwmOut      CLOCK(D8);
DigitalOut  CW(D12);
DigitalOut  ENA(D11);
DigitalOut  RES(D10);
DigitalOut  HALF(D9);
AnalogOut   VREF(A2);
// - Signal en sortie du détecteur de présence
InterruptIn mon_interrupt(D2);
// - Signaux d'alimentation des différents pistons
PwmOut      Y(D3);   // Piston Jaune
PwmOut      R(D4);   // Piston Rouge
PwmOut      G(D5);   // Piston Vert
PwmOut      B(D6);   // Piston Bleu

// DECLARATION DES CONSTANTES DE TEMPS
// Ces dernières sont déterminées expérimentalement en mesurant le temps
// nécessaire à une pièce pour passer devant le piston de tri associé après
// avoir été détectée. Le nombre indiqué correspond au nombre de demi-secondes
// nécessaires.
#define TimeRed 8
#define TimeGreen 11
#define TimeYellow 17
#define TimeBlue 20
// Remarque : En pratique, une des couleurs est traitée en bout de convoyeur
// et ne nécessite donc pas de piston, donc pas de constante de temps associée
// (ici le bleu).

// DECLARATION DU TICKER
// Ce dernier nous permet de gérer les informations en temps réel
Ticker mon_ticker;

// DECLARATION DES VARIABLES
double omega = 2;           // Pulsation du moteur pas à pas (rad/s)
char data_received = 0;     // Variable de réception des données(RS232)
```

```
67  int T[6];                   // Tableau des timers (en demi-seconde)
68  int C[6];                   // Tableau des couleurs associées aux timers
69  int k;                      // Variable d'itération
70  double var_detection;       // Signal détecté en sortie du détecteur de présence
71
72  // DECLARATION DES FONCTIONS
73  void ISR_get_data(void);
74  // Réception des données sur la liaisons RS232, et traite les données en
75  // fonction de la couleur détectée.
76  void select_timer(int color, int Time_color);
77  // Complète le premier timer disponible dans les tableaux T et C en fonction
78  // de la couleur de la pièce.
79  void decompte(void);
80  // Permet de décompter les timers (gestion en temps réel).
81  void ISR_set_data(void);
82  // Envoie l'ordre de prendre une photo à MatLab lorsqu'un objet est détecté.
83
84  // MAIN
85  int main(){
86      // - Gestion de la liaison MATLAB - MBED
87      rs232.baud(9600);
88      rs232.attach(&ISR_get_data);
89
90      // - Initialisation des pistons en position
91      Y.period_ms(20);
92      Y.pulsewidth_us(800);
93      R.period_ms(20);
94      R.pulsewidth_us(800);
95      G.period_ms(20);
96      G.pulsewidth_us(800);
97      B.period_ms(20);
98      B.pulsewidth_us(800);
99
100     // - Rotation du convoyeur
101     CLOCK.period(2*3.14/omega/1000);
102     // -- Permet de fixer la vitesse voulue
103     CLOCK.write(0.5);
104     // -- Génération d'un signal créneau
105     CW = 1;
106     // -- Définit le sens de rotation du convoyeur
107     ENA = 1;
108     // -- Autorise la circulation du courant
109     RES = 1;
110     // -- Permet de ne pas rénitialiser en boucle
111     HALF = 0;
112     // -- Mode pas à pas
113     VREF.write(1);
114     // -- Permet de borner la tension dans le composant
115
116     // - Attribution de la fonction de détection
117     mon_interrupt.fall(&ISR_set_data);
118
119     // - Initialisation des timers à 0s
120     for(k=0; k<6; k++){
121         T[k] = 0;
122     }
123
124     // - Décompte des timers toutes les demi-secondes
125     mon_ticker.attach(&decompte,0.5);
126
127     // - Activation des pistons en temps voulu
128     // Lorsqu'un timer est sur le point de tomber à 0s, nous récupérons
129     // l'information de la couleur et nous activons le piston correspondant
130     // pour trier la pièce.
131     while (1) {
132         for (k=0;k<6;k++)
133         {
134         if(T[k] == 1){
```

```
            if (C[k]==0){
                Y.pulsewidth_us(2200);
                wait(1);
                Y.pulsewidth_us(800);
                wait(1);}
            if (C[k]==1){
                R.pulsewidth_us(2200);
                wait(1);
                R.pulsewidth_us(800);
                wait(1);}
            if (C[k]==2){
                G.pulsewidth_us(2200);
                wait(1);
                G.pulsewidth_us(800);
                wait(1);}
            if (C[k]==3){
                B.pulsewidth_us(2200);
                wait(1);
                B.pulsewidth_us(800);
                wait(1);}
             }
        }
}
}

void ISR_get_data(){
    data_received = rs232.getc();
    switch(data_received){
        case 'Y':
            rs232.putc('Y');
            select_timer(0, TimeYellow);
            break;
        case 'R':
            rs232.putc('R');
            select_timer(1, TimeRed);
            break;
        case 'G':
            rs232.putc('G');
            select_timer(2, TimeGreen);
            break;
        case 'B':
            rs232.putc('B');
            select_timer(3, TimeBlue);
            break;
        default:
            rs232.putc('o');
    }
}

void select_timer(int color, int Time_color){
    if (T[1] == 0){
        T[1] = Time_color;
        C[1] = color;
        return;
    }
    if (T[2] == 0){
        T[2] = Time_color;
        C[2] = color;
        return;
    }
    if (T[3] == 0){
        T[3] = Time_color;
        C[3] = color;
        return;
    }
    if (T[4] == 0){
        T[4] = Time_color;
        C[4] = color;
```

```
            return;
        }
        if (T[5] == 0){
            T[5] = Time_color;
            C[5] = color;
            return;
        }
        if (T[6] == 0){
            T[6] = Time_color;
            C[6] = color;
            return;
        }
    }

    void decompte(){
        for (k=0;k<6;k++){
            if(T[k] != 0) {T[k]--;}
    }}

    void ISR_set_data(){
        var_detection=mon_interrupt.read();
        rs232.printf("%lf", var_detection);}
```

File "/tri_banc/main.cpp" printed from os.mbed.com on 4/5/2022