iut NICE CÔTE D'AZUR – UNIVERSITÉ CÔTE D'AZUR	GEII BUT2 ALT **SAE ROUE** 2023 - 2024	**Partie 2 (commande microcontroleur)**

Objectifs :

- **Commander la roue et effectuer tous les tests unitaires**

1 Rappel du cahier des charges

le microcontrôleur doit au travers une interface série être capable de :

```
veuillez saisir un choix parmi la liste proposee:
a: Consigne pwm
f: Calibration
c: Courant
v: Tension
t: Temperature
f: Frein
d: Debug
q: Quitter
```

Menu pour afficher les différentes questions

1)commander avec la poignée la vitesse de la roue

```
void consignePoigne() {
  consigne_pwm = 255 * (Pgaz.read() - gaz_min) /
                 (gaz_max - gaz_min); // Conversion de la poignee de gaz en pwm

  MyPLD.write((int)consigne_pwm);
}
```

2)commander avec un scanf la vitesse de la roue

Detache la poignée car pas possible de faire les 2 en même temps

```
    case 'a':
      ticPoigne.detach(&consignePoigne, TProgressive);
      pc.printf("Valeur pwm (0-255) :");
      VALID_PWM.write(1); // PWM de la roue activée
      pc.scanf(" %d", &consigne_pwm);
      MyPLD.write(consigne_pwm);
      break;
    case 'p': // mode poignee
      ticPoigne.attach(&consignePoigne, TProgressive);
      break;
```

3)lire le courant

Calcul de courant

```
1382.702515 mA
```

```
    case 'c':
      current =
          ((((VIDC.read() * 3.3) * (R1 + R2) / (R2)) - 2.5) / 0.185) * 1000;
      printf("%f mA\n\r", current);
      break;
```

4)lire la tension

Lecture de la tension

```
case 'v':
    Vbat_reel = Vbat.read() * 77.5;
    printf("%f V\n\r", Vbat_reel);
    break;
```

```
v23.221613 V
```

5)lire la température

Calcul de la température

```
case 't':
    temperature = (VTemp.read() * 3.3 / 0.01) - 273.15; // En degrés
    printf("%f degres C\n\r", temperature);
    break;
```

```
32.271267 degres C
```

6)lire la tension fournie par la poignée

```
pc.printf("%.2f\n\r",Pgaz.read());
```

7)faire un calibrage de la poignée (min = vitesse nulle et max = vitesse max) dans un fichier texte sur le LPC1768

```
case 'f':
    ticPoigne.detach(); // Arret de l'appel de la fonction de la poignee
    consigne_pwm = 0;
    MyPLD.write((int)consigne_pwm);
    printf("Mettez la poignee au minimum : \n\r");
    wait(3);
    gaz_min = Pgaz.read();
    printf("Pgaz_min=%f \n\r", gaz_min);
    printf("Mettez la poignee au maximum : \n\r");
    wait(3);
    gaz_max = Pgaz.read();
    printf("Pgaz_max=%f \n\r", gaz_max);
    FILE *fp = fopen("/local/GRA_INI.txt",
                     "w"); // Ouverture du fichier texte en lecture
    if (fp != NULL) {
        fprintf(fp, "gaz_min=%f gaz_max=%f", gaz_min, gaz_max);
        fclose(fp);
        printf("Fichier ecrit \n\r");
    } else {
        printf("Impossible d'ecrire le fichier \n\r");
    }
    ticPoigne.attach(&consignePoigne, 0.05);
    break;
```

8)lire le frein, l'erreur et la position des capteurs à effet hall

9)calculer la vitesse réelle

InterruptIn les capteurs a effet hall et avec un ticker regarder les fronts et ensuite on calcule.

```
void speed_mesureament() {
  // frequency = ((float)cpt_hall) / DELTA_T; // Fréquence 1/T

  tr_min = (float)cpt_hall * 60 / 6 / NBPOLES / NBPOLES;
  // tr_min = (frequency * 60) / (6 * NBPOLES);
  vitesse = (2.0f * RADIUS * tr_min * 3.14) * 60; // conversion de tr/min en m/h
  cpt_hall = 0;
}

void hall_cpt(void) {
  cpt_hall = cpt_hall + 1;
} // Incrémentation des tops de VALID_HALL
```

```
pc.printf("La vitesse de la roue est de %.2f m/h\n\r", vitesse);
pc.printf("La vitesse de la roue est de %.2f tr/s\n\r", tr_min);
pc.printf("cpt_hall %d \n\r", cpt_hall);
```

```
La vitesse de la roue est de 259.05 m/h
La vitesse de la roue est de 3.44 tr/s
cpt_hall 12
```

CODE ENTI

sae_roue_2024 > main.cpp > main

```
//#include "EthernetInterface.h"
#include "bloc_io.h"
#include "html.h" // need for html patch working with web server
#include "mbed.h"
#include "rtos.h" // need for main thread sleep
#include <stdlib.h>
#include <string.h>

#define RADIUS 0.2F  // wheel size
#define NBPOLES 8    // magnetic pole number
#define DELTA_T 0.1F // speed measurement counting period
#define Pgaz_min 0.158974F
#define Pgaz_max 0.709F
#define TProgressive 0.005F
#define R1 28000.0F
#define R2 33000.0F

DigitalOut VALID_PWM(p21);    // Générateur PWM
InterruptIn VALID_HALL(p22); // Capteurs HALL
AnalogIn Pgaz(p17);          // Poignée de gaz
AnalogIn Vbat(p18);          // Tension de la batterie
AnalogIn VTemp(p19);         // Temperature du convertisseur
AnalogIn VIDC(p20);          // Courant mesuré de la batterie

int consigne_pwm;
Ticker ticPoigne;
Ticker ticVitesse;
Bloc_IO MyPLD(p25, p26, p5, p6, p7, p8, p9, p10, p23,
              p24);      // instantiate object needed to communicate with PLD
                         // analog input connected to mbed
                         //valid pmw mbed pin
Serial pc(USBTX, USBRX); //tx, rx
/************persistent file parameters section***************/
LocalFileSystem
    local("local"); // Create the local filesystem under the name "local"
```

```
/********************* web server section *********************************/

var_field_t tab_balise[10]; // une balise est présente dans le squelette
int giCounter = 0;          // acces counting

/********************* can bus section ***********/
// determine message ID used to send Gaz ref over can bus
#define _CAN_DEBUG // used to debug can bus activity
//#define USE_CAN_REF // uncomment to receive gaz ref over can_bus

CAN can_port(p30, p29); // initialisation du Bus CAN sur les broches 30 (rd) et
                        // 29(td) for lpc1768 + mbed shield
bool bCan_Active = false;

float frequency = 0;
DigitalOut led1(
    LED1); // initialisation des Leds présentes sur le micro-controleur Mbed*/
DigitalOut led2(LED2);
DigitalOut led3(LED3); // blink when can message is sent
DigitalOut led4(LED4); // blink when can message is receive
int n, DataIn;
float gaz_min, gaz_max;
float Vbat_reel = 0.0;
float temperature = 0.0;
float current = 0.0;
float vitesse = 0, tr_min = 0;
int cpt_hall = 0; // Compteur secteur d'alimentation du moteur brushless
int cpt_total = 0, cpt_partiel = 0, cpt_total_old = 0;
float gaz_pourcentage = 0.0;
char cChoix = 0;

// LocalFileSystem local("local"); // Create the local filesystem under the name
// "local"
```

```
void consignePoigne(void);

void consignePoigne() {
    consigne_pwm = 255 * (Pgaz.read() - gaz_min) /
                   (gaz_max - gaz_min); // Conversion de la poignee de gaz en pwm

    MyPLD.write((int)consigne_pwm);
}

void Init(void);
void CGI_Function(void) // cgi function that patch web data to empty web page
{
    current = ((((VIDC.read() * 3.3) * (R1 + R2) / (R2)) - 2.5) / 0.185) * 1000;
    temperature = (VTemp.read() * 3.3 / 0.01) - 273.15; // En degrés
    gaz_pourcentage = 100 * (Pgaz.read() - gaz_min) / (gaz_max - gaz_min);
    Vbat_reel = Vbat.read() * 77.5;

    char ma_chaine4[20] = {}; // needed to form html response

    sprintf(ma_chaine4, "%.2f",
            vitesse); // ticker    // convert speed as ascii string
    Html_Patch(tab_balise, 0, ma_chaine4); // patch first label with dyn.string
    sprintf(ma_chaine4, "%.2f", gaz_pourcentage); // convert speed as ascii string
    Html_Patch(tab_balise, 1, ma_chaine4);  // patch first label with dyn.string
    sprintf(ma_chaine4, "%.2f", current);   // convert speed as ascii string
    Html_Patch(tab_balise, 2, ma_chaine4);  // patch first label with dyn.string
    sprintf(ma_chaine4, "%.2f", Vbat_reel); // convert speed as ascii string
    Html_Patch(tab_balise, 3, ma_chaine4);  // patch first label with dyn.string

    temperature = (VTemp.read() * 3.3 / 0.01) - 273.15; // En degrés
    current = ((((VIDC.read() * 3.3) * (R1 + R2) / (R2)) - 2.5) / 0.185) * 1000;
}
```

sae_roue_2024 > c++ main.cpp > main

```
/********************* CAN BUS SECTION *********************/
void CAN_REC_THREAD(void const *args) {
    int iCount, iError;

    while (bCan_Active) {
        Thread::wait(100); // wait 100ms
        // code todo
    }
}

void speed_mesureament() {
    // frequency = ((float)cpt_hall) / DELTA_T; // Fréquence 1/T

    tr_min = (float)cpt_hall * 60 / 6 / NBPOLES / NBPOLES;
    // tr_min = (frequency * 60) / (6 * NBPOLES);
    vitesse = (2.0f * RADIUS * tr_min * 3.14) * 60; // conversion de tr/min en m/h
    cpt_hall = 0;
}

void hall_cpt(void) {
    cpt_hall = cpt_hall + 1;
} // Incrémentation des tops de VALID_HALL

void Init() {

    consigne_pwm = 0;               // rapport cyclique fixé à 0%
    VALID_PWM.write(0);             // PWM de la roue desactivée
    MyPLD.write((unsigned char)0); // écriture sur le bus de donnée
    VALID_PWM.write(1);             // PWM de la roue activée
    VALID_HALL.mode(PullUp);
    VALID_HALL.rise(&hall_cpt);
    ticPoigne.attach(&consignePoigne, TProgressive);
    ticVitesse.attach(&speed_mesureament,
                      DELTA_T);     // Appel de la fct à période DELTA_T
    Init_Web_Server(&CGI_Function); // create and initialize tcp server socket and
```

```
Init_Web_Server(&CGI_Function); // create and initialize tcp server socket and
Thread WebThread(Web_Server_Thread); // create and launch web server thread
Gen_HtmlCode_From_File(
    "/local/pagecgi2.htm", tab_balise,
    4);             // read and localise ^VARDEF[X] tag in empty html file
bCan_Active = true; // needed to lauchn CAN thread
Thread CanThread(CAN_REC_THREAD); // create and launch can receiver thread
FILE *file = fopen("/local/.GRA_INI.txt",
                   "r"); // Ouverture du fichier texte en lecture
if (file == NULL) {      // Si impossible d'ouvrir le fichier
  printf("Impossible d'ouvrir le fichier \n\r");
  gaz_min = Pgaz_min;
  gaz_max = Pgaz_max;
} else {
  fscanf(file, "gaz_min=%f gaz_max=%f", &gaz_min,
         &gaz_max); // Scan du fichier
  fclose(file);
  printf("Fichier ouvert \n\r");
}
pc.printf(" programme scooter mbed \n");
}

int main() {
 Init();
 while (cChoix != 'q' and cChoix != 'Q') {
  printf(" veuillez saisir un choix parmi la liste proposee: \n");
  printf(" p: mode poigne \n\r");
  printf(" a: Consigne pwm \n\r");
  printf(" f: Calibration\n\r");
  printf(" c: Courant \n\r");
  printf(" v: Tension \n\r");
  printf(" t: Temperature\n\r");
  printf(" f: Frein\n\r");
  printf(" d: Debug\n\r");
  printf(" q: Quitter \n");
```

```
  printf(" q: Quitter \n");

  /************ multithreading : main thread need to sleep in order to allow
   * web response */
  while (pc.readable() == 0) // determine if char availabler
  {
   Thread::wait(10);
  } // wait 10 until char available on serial input
  /************ end of main thread sleep ***************/
  pc.scanf(" %c", &cChoix);
  switch (cChoix) {
  case 'a':
   ticPoigne.detach();
   pc.printf("Valeur pwm (0-255) :");
   VALID_PWM.write(1); // PWM de la roue activée
   pc.scanf(" %d", &consigne_pwm);
   MyPLD.write(consigne_pwm);
   break;
  case 'p': // mode poignee
   ticPoigne.attach(&consignePoigne, TProgressive);
   break;
  case 'd':
   // Affichage vitesse
   pc.printf("%.2f\n\r", Pgaz.read());
   pc.printf("La vitesse de la roue est de %.2f m/h\n\r", vitesse);
   pc.printf("La vitesse de la roue est de %.2f tr/s\n\r", tr_min);
   pc.printf("cpt_hall %d \n\r", cpt_hall);
   // pc.printf("frequence %f \n\r", frequency);
   break;

  case 'f':

   ticPoigne.detach(); // Arret de l'appel de la fonction de la poignee
   consigne_pwm = 0;
   MyPLD.write((int)consigne_pwm);
```

```
   printf("Mettez la poignee au minimum :  \n\r");
   wait(3);
   gaz_min = Pgaz.read();
   printf("Pgaz_min=%f \n\r", gaz_min);
   printf("Mettez la poignee au maximum :  \n\r");
   wait(3);
   gaz_max = Pgaz.read();
   printf("Pgaz_max=%f \n\r", gaz_max);
   FILE *fp = fopen("/local/GRA_INI.txt",
                    "w"); // Ouverture du fichier texte en lecture
   if (fp != NULL) {
    fprintf(fp, "gaz_min=%f gaz_max=%f", gaz_min, gaz_max);
    fclose(fp);
    printf("Fichier ecrit \n\r");
   } else {
    printf("Impossible d'ecrire le fichier \n\r");
   }
   ticPoigne.attach(&consignePoigne, 0.05);
   break;
  case 'c':
   current =
       (((VIDC.read() * 3.3) * (R1 + R2) / (R2)) - 2.5) / 0.185) * 1000;
   printf("%f mA\n\r", current);
   break;
  case 'v':
   Vbat_reel = Vbat.read() * 77.5;
   printf("%f V\n\r", Vbat_reel);
   break;
  // frein_reel =
  // printf("%f V\n\r", frein_reel);
  case 't':
   temperature = (VTemp.read() * 3.3 / 0.01) - 273.15; // En degrés
   printf("%f degres C\n\r", temperature);
   break;
```

```
  }
 } // end while

 //************ thread deinit ********************
 DeInit_Web_Server();

 // bCan_Active=false;
 // CanThread=false;// close can received thread
 pc.printf(" fin programme scooter mbed \n");
} // end main
```