Définitions, fonctions et librairies pour le xBot

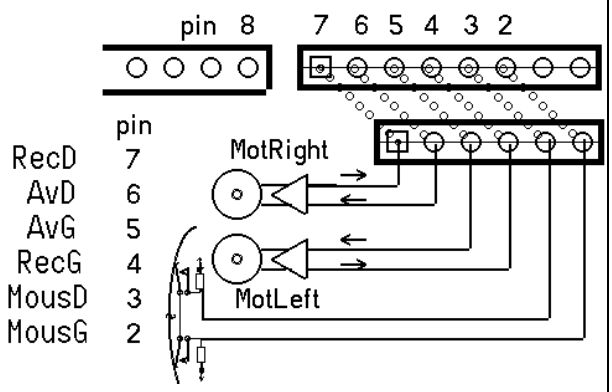
Compatible LearnCbot www.didel.com/coursera/LC.pdf

Aide-mémoire, à consulter à chaque nouveau projet

Aide-mémoire C: www.didel.com/C/Resume.pdf

Câblage standard

Rappelons que les pins 2 à 7 sont sur les bits 2 à 7 port D du processeur Avr368.
Les bits 2 et 3 sont en entrée pour lire les moustaches. Les bits 4 à 7 commandent les moteurs, avec les bits 5 et 6 qui permettent le PWM Arduino, ils sont donc utilisés comme signaux pour faire avancer.
Si le câblage est différent, il faut revoir le fichier de définition et le fichier XBotPwm.h
Le bit 1 est réservé pour un HP avec une résistance de plus de 30 Ohm, comme sur le LearnCbot.



Arduino versus C

Les premier programmes dans Start utilisent les notations Arduino pour faciliter la transition vers le C et notre approche temps réel basées sur le Timer1. Il faut bien comprendre notre ligne de conduite avant de vouloir intégrer des librairies Arduino (section x)
Les notations Arduino ne sont utilisées que dans les fichiers de définition, en liaisons avec des spécificités Arduino, comme la pin13 et sa Led, ou les pins pwm qui ne sont initialisées correctement que via des instructions Arduino,

Règles et mise en garde

Les noms de dossier commencent par une minuscules, sauf pour les noms de croquis.
Dossiers xBot… Croquis Xbot Documentation, figure XBot.. (exeptions historiques)
Attention à la différence d'appel d'une macro et d'une fonction. Le compilateur réagit très mal en cas d'erreur: message incompréhensible ou exécution incorrecte!
Il faut être maniaque pour les minuscules et majuscules. Un b en première lettre distingue le numéro de bit sur un port et le nom de la pin Arduino. Un m en première lettre caractérise un masque.

La commande des moteurs se fait de trois façons
1) en tout ou rien avec les macros Avance; Recule; TourneD; TourneG; Stop;
   (déclarés dans XBotDef.h)
2) en PWM avec les fonctions SetPwmD (int); SetPwmG (int);
   (déclarés dans XBotPwm.h) valeur pwm entre –255 et +255
3) en PFM avec les fonctions VitD (byte); VitG (byte);
   (déclarés dans XBotPfm.h) valeur pfm entre –20 et +20 (ou 32)
   - appels dans une boucle de 2ms
     (exemple dans XBotSoft.pfm)
   - appel par l'interruption du timer1
     (exemple dans XBotInt.pdf)

La lecture des moustaches se fait de 2 façons
1) Avec les macros ObsG ObsD – vrais si la moustache est pressée, donc led allumée
   (déclarés dans XBotDef.h)
2) En testant dans la variable byte EtatMous les flags bObsG bObsD quittancés automatiquement.

1. Fichier de définitions `XbotCDef.h`

(le fichier `XbotADef.h` est style Arduino, n'utilise pas les PORTs) XbotDef id XbotCDef

```
// XbotCDef.h definitions et fonctions
#include <Arduino.h>
#define bLed 5 // pin 13  debug
#define LedOn  digitalWrite (13,HIGH)
#define LedOff digitalWrite (13,LOW)
#define LedToggle PORTB ^= 1<<bLed

#define bRecG 4 // PORTD  actifs à 1
#define bAvG 5
#define bAvD 6
#define bRecD 7
// pour les exercice simples
#define Avance  PORTD |= 0b01100000; PORTD &= 0b01101111
#define Recule  PORTD |= 0b10010000; PORTD &= 0b10011111
#define TourneD PORTD |= 0b10100000; PORTD &= 0b10101111
#define TourneG PORTD |= 0b01010000; PORTD &= 0b01011111
#define Stop PORTD &= ~0b11110000
// pour le Pfm
#define AvD   bitClear (PORTD,bRecD); bitSet   (PORTD,bAvD)
#define RecD  bitSet   (PORTD,bRecD); bitClear (PORTD,bAvD)
#define StopD bitClear (PORTD,bRecD); bitClear (PORTD,bAvD)
#define AvG   bitClear (PORTD,bRecG); bitSet   (PORTD,bAvG)
#define RecG  bitSet   (PORTD,bRecG); bitClear (PORTD,bAvG)
#define StopG bitClear (PORTD,bRecG); bitClear (PORTD,bAvG)

#define bMousD 3 // PORTD,3)
#define bMousG 2
#define ObsG  !(PIND&1<<bMousD)
#define ObsD  !(PIND&1<<bMousG)

#define LedOn  digitalWrite (13,HIGH)
#define LedOff digitalWrite (13,LOW)

void SetupXBot ()  {
   DDRD |=  0b11110000 ; // force les 1 out
   DDRD &=  0b11110011 ; // force les 0 in
   PORTD = 0x0F ; // --> 0 Mot stop
   DDRB |= 1<<bLed ; // Led13
}
```

Programme de test – EviteObstaclesC.ino – On recule et tourne s'il y a obstacle

```
// EvitObstacleC.ino 778 bytes
#include "XbotCDef.h"
void setup() {
  SetupXBot();
}
```

```
void loop()  {
  Avance ;
  if (ObsD) { //on recule et tourne
    Recule ;   delay (300) ;
    TourneG ;  delay (200) ;
  }
  if (ObsG) {  //on recule et tourne
    Recule ;   delay (300) ;
    TourneD ;  delay (100) ;
  }
}
```

saut de page

2. PWM `XbotPwm.h`

Les pins 5 et 6 `(AvG et AvD)` acceptent l'ordre `analogWrite (byte);` qui est géré par interruptino et interromp ée programmes pour 10 microsecondes toutes les millisecondes environ.
C'est donc facile de programmer pour avancer à vitesse variable (supérieure à 10% de Pwm). Pour reculer, il faut des inversions expliquées dans www.didel.com/robots/MotorControl.pdf

La librairie **`XbotPwm.h`** définit la fonctions `SetPwmGD (vitG,vitD);`
Les paramètres sont des nombres signés, entre –255 et +255.

```
// XbotPwm.h     –255 et +255
void SetPwmGD (int); (int pp,qq)  {
  if (pp > 0)
  {
    analogWrite(AvD, pp);
    digitalWrite(RecD, LOW);
  }  else   {
    analogWrite(AvD, 256+pp); // pp is neg
    digitalWrite(RecD, HIGH);
  }
  if (qq > 0)
  {
    analogWrite(AvG, qq);
    digitalWrite(RecG, LOW);
  }  else   {
    analogWrite(AvG, 256+qq); // pp is neg
    digitalWrite(RecG, HIGH);
  }
}
```

Programme de test

```
// AllerRetPwm.ino aller-retour

#include "XbotCDef.h"
#include "XbotPwm.h"
void setup() {
  SetupXBot();
}
```

```
// accélère en 2.5s
int vit;
void loop() {
   for (vit=0; vit<255; vit++) {
      SetPwmG (vit); SetPwmD (vit);
      delay (10);
   }
   for (vit=255; vit>-255; vit--) {
      SetPwmG (vit); SetPwmD (vit);
      delay (10);

   }
   for (vit=-255; vit<=0; vit++){
      SetPwmG (vit); SetPwmD (vit);
      delay (10);
   }
}
```

saut de page

2. PFM `XbotPfm.h`

Le Pfm est programmable sur n'importe quelle pin, et l'inconvénient du PWM sur 2 pins au lieu de 4 n'apparaît pas. Mais la contrainte est d'appeler la fonction DoPfmGD tous les DelPfm = 2ms avec les moteurs du xBot

```
// XbotPfm.h     –20 et +20 saturé
#define DelPfm 2  // durée min pour débloquer moteur
#define MaxPfm 20
char  pfmG, pfmD;  // -MaxPfm .. 0 .. MaxPfm
volatile byte pfmCntg;  volatile byte pfmCntd;
void DoPfmGD (char pfg, char pfd) {
  if (pfd > MaxPfm) pfd=MaxPfm; // saturer
  if (pfd < -MaxPfm) pfd= -MaxPfm;
  if (pfd >=0) {
     if ((pfmCntd += pfd) > MaxPfm) {
        pfmCntd &= MaxPfm;
        MotdAv;
     }
     else { MotdStop; }
  }
  if (pfd <0) {
     pfd= -pfd;  // ou pfd= ABS(pfd)
     if ((pfmCntd += pfd) > MaxPfm) {
        pfmCntd &= MaxPfm;
        MotdRec;
     }
     else { MotdStop; }
   }
if (pfg > MaxPfm) pfg=MaxPfm; // saturer
  if (pfg < -MaxPfm) pfg= -MaxPfm;
  if (pfg >=0) {
     if ((pfmCntg += pfg) > MaxPfm) {
        pfmCntg &= MaxPfm;
        MotgAv;
     }
     else { MotgStop; }
  }
  if (pfg <0) {
     pfg= -pfg;  // ou pfg= ABS(pfg)
     if ((pfmCntg += pfg) > MaxPfm) {
        pfmCntg &= MaxPfm;
        MotgRec;
     }
     else { MotgStop; }
   }
}
```

Programme de test sans interruptions

```
// TestPfm.ino Tourne sur place à vit variable

#include "XbotCDef.h"
#include "XbotPfm.h"  // var glo pfmG, pfmD
void setup() {
  SetupXBot();
}
```

```
void loop () {
  delay (2);
  DoPfmGD (pfm,-pfm);
  if (cnt++ > 100) {
     cnt=0;
     pfm++; if (pfm>MaxPfm) pfm=0;
  }
}
```

3. Interruptions timer 2

Voir www.didel.com/diduino/PfmPratique.pdf

```
// ISRT2Pfm.h
volatile byte ct1;
ISR (TIMER2_OVF_vect) {
    TCNT2 = 68;  //  100 us
    if (ct1++ > 20) {  // 2ms
      ct1 = 0;
      DoPfm (pfm);
    }
}
void SetupTimer2() {
  cli();
  TCCR2A = 0; //default
  TCCR2B = 0b00000010;
  TIMSK2 = 0b00000001;
  sei();
}
```

```
// TestPfmInter.ino
#include "XbotCDef.h"
#include "XbotPfm.h"
#include "ISRT2Pfm.h"

void setup()  {
  SetupXBot();
  SetupTimer2();
}
void loop () {
  pfmD++; if (pfmD>MaxPfm) pfmD=0;
  delay (200);
}
```

jdn 140622/170808