Código utilizado para el Torneo Mexicano de Robótica

Código utilizado en robot que participó en el Torneo Mexicano de Robótica, que se realizó del 4 al 6 de mayo de 2011 en el Instituto Tecnológico Autónomo de México, que le proporciona al robot el comportamiento de solución de laberinto mediante la técnica de la lógica de seguimiento de la mano izquierda o derecha. El reglamento para esta competencia cambió para el efecto de facilitar la solución del laberinto.

main.c

```
#include <header.h>
void main(void)
{
   configura();
   set_pwm1_duty(626);
   set_pwm2_duty(625);
   FORWARD;
   delay_ms(20);

   //if(input(PIN_E0))
   //   while(TRUE){ManoDerecha();}
   //else
   //   while(TRUE){ManoIzquierda();}

   while(TRUE)
   {
      ManoDerecha();
   }
}
```

header.c

```
#include <16F877A.h>
#fuses HS, NOWDT, NOPROTECT, NOLVP
#device ADC=8
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#use fast_io(D)
#include <config.h>
#include <sensors.h>
#include <motors.h>
#include <behavior.h>
```

config.h

```
void configura()
{
   setup_adc(ADC_CLOCK_INTERNAL);
   setup_adc_ports(ALL_ANALOG);
   setup_ccp1(CCP_PWM);
   setup_ccp2(CCP_PWM);
   setup_timer_2(T2_DIV_BY_16, 156,1);

   set_tris_d(0x00);
   output_d(0x00);
   set_tris_b(0x00);
   output_b(0x00);

   //botón de inicio
```

```
   output_e(PIN_E0);
   set_tris_e(0x01);

   delay_ms(1000);
}
```

sensors.h

```
#define TIMEsensor   10//
#define WallExist    100

unsigned int8 ReadSLI(void)
{
   unsigned int8 ADCvalue;
   set_adc_channel(0);
   delay_us(20);
   output_high(PIN_D3);
   delay_us(TIMEsensor);
   ADCvalue=read_adc();
   output_low(PIN_D3);
   return ADCvalue;
}

unsigned int8 ReadSFI(void)
{
   unsigned int8 ADCvalue;
   set_adc_channel(1);
   delay_us(20);
   output_high(PIN_D2);
   delay_us(TIMEsensor);
   ADCvalue=read_adc();
   output_low(PIN_D2);
   return ADCvalue;
}

unsigned int8 ReadSFD(void)
{
   unsigned int8 ADCvalue;
   set_adc_channel(2);
   delay_us(20);
   output_high(PIN_D1);
   delay_us(TIMEsensor);
   ADCvalue=read_adc();
   output_low(PIN_D1);
   return ADCvalue;
}

unsigned int8 ReadSLD(void)
{
   unsigned int8 ADCvalue;
   set_adc_channel(3);
   delay_us(20);
   output_high(PIN_D0);
   delay_us(TIMEsensor);
   ADCvalue=read_adc();
   output_low(PIN_D0);
   return ADCvalue;
}

unsigned int WallCheck(){
```

```
   //walls en un entero no signado que representa la presencia de las
   //paredes derecha, izquierda y hacia el frente, el bit se coloca a 1
   //si la pared correspondiente está presente
   unsigned int SLI, SFI, SFD, SLD, walls=0;
   SLI=ReadSLI();
   SFI=ReadSFI();
   SFD=ReadSFD();
   SLD=ReadSLD();
   if(SLI<WallExist) walls = walls | 8;
   if(SFI<WallExist || SFD<WallExist) walls = walls | 6;
   if(SLD<WallExist) walls = walls | 1;
   return walls;}
```

motors.h

```
#define STOP      output_b(0x00)
#define FORWARD   output_b(0x05)
#define BACKWARD  output_b(0x22)
#define LEFTTURN  output_b(0x21)
#define RIGHTTURN output_b(0x06)

#define BIAS 100
#define TIMEms 10
#define TIME_90 290

//La velocidad de la rueda derecha está controlada por el Puerto pwm1
//La velocidad de la rueda izquierda está controlada por pwm2

void TurnRight90()
{
   set_pwm1_duty(55*625/100);
   set_pwm2_duty(55*625/100);
   RIGHTTURN;
   delay_ms(TIME_90);//300 FOR 9 VOLTS, 400 for 7.4 v
   STOP;
}

void TurnLeft90()
{
   set_pwm1_duty(55*625/100);
   set_pwm2_duty(55*625/100);
   LEFTTURN;
   delay_ms(TIME_90);//300 FOR 9 VOLTS, 400 for 7.4 v
   STOP;
}

void Forward_()
{
   set_pwm1_duty(45*625/100);
   set_pwm2_duty(45*625/100);
   FORWARD;
   delay_ms(TIMEms); //delay_ms(840);   //Tiempo para moverse de celda a
                                        //celda
}

void Avoid()
{
   unsigned int SLI, SLD;
   SLI=ReadSLI();
   SLD=ReadSLD();
   if(SLI<BIAS && SLD<BIAS){
```

```
         set_pwm1_duty((BIAS-SLD)*(625/100));
         set_pwm2_duty((BIAS-SLI)*(625/100));
}
   else
   {
      set_pwm1_duty(50*625/100);
      set_pwm2_duty(50*625/100);
   }
   FORWARD;
   delay_ms(TIMEms);
}
```

behavior.c

```
#define TimeToCellToCell   590//basis 600
#define TimeToReachCenter  430//basis 450

void GoReturn()
{
      Forward_();
      if(ReadSFI()<40 || ReadSFD()<40)
      {
         TurnRight90();
         TurnRight90();
      }
      else Avoid();
}

void ToLeftCell()
{
   TurnLeft90();
   Forward_();
   delay_ms(TimeToCellToCell-30);
}

void ToRightCell()
{
   TurnRight90();
   Forward_();
   delay_ms(TimeToCellToCell);
}

void Turn180()
{
   TurnRight90();
   TurnRight90();
   Forward_();
   delay_ms(TimeToCellToCell);
}

void RightHand()
{
   Forward_();
   if(ReadSFI()<35 || ReadSFD()<35)
   {
      TurnRight90();
      TurnRight90();
   }
   else
   {
      Forward_();
      if(ReadSLD()>100 || ReadSLI()>100)
```

```
      {
         unsigned int wall;
         Forward_();
         delay_ms(TimeToReachCenter);
         wall = WallCheck();
         switch(wall){
         case  6 : ToRightCell(); break;
         case  14: ToRightCell(); break;
         case  8 : ToRightCell(); break;
         case  0 : ToRightCell(); break;
         case  7 : ToLeftCell() ; break;
         case  1 : Forward_()   ; break;
         case  15: Turn180()    ; break;}
      }
      else Avoid();
   }
}

void LeftHand()
{
   Forward_();
   if(ReadSFI()<35 || ReadSFD()<35)
   {
      TurnRight90();
      TurnRight90();
   }
   else
   {
      Forward_();
      if(ReadSLD()>100 || ReadSLI()>100)
      {
         unsigned int wall;
         Forward_();
         delay_ms(TimeToReachCenter);
         wall = WallCheck();
         switch(wall){
         case  6 : ToLeftCell();    break;
         case  14: ToRightCell();   break;
         case  8 : Forward_();      break;
         case  0 : ToLeftCell();    break;
         case  7 : ToLeftCell();    break;
         case  1 : ToLeftCell();    break;
         case  15: Turn180();       break;}
      }
      else Avoid();
   }
}
```