```
//*****************************************************************************************************
//*****************************************************************************************************
//            PROGRAMA CONTROL MOVIMENT ROBOT MÓBIL. PROJECTE FI DE CARRERA 07/06/2011.
//                                          ARDUINO UNO
//*****************************************************************************************************
//*****************************************************************************************************


#include <MsTimer2.h>// Habilita llibreria funció habilitació servei
                     //interrupció timer 2

int  MOTOR_1_AVANZ = 4;// Pin 4. Sortida avnaÃ§ motor 1
int  MOTOR_1_RETROCES = 5;// Pin 5. Sortida retroces motor 1
int  MOTOR_1_SPEED= 9;// Pin 6. Sortida PWM motor 1
int  MOTOR_2_AVANZ = 6;// Pin 7. Sortida avanÃ§ motor 2
int  MOTOR_2_RETROCES = 7;// Pin 8. Sortida retroces motor 2
int  MOTOR_2_SPEED = 10;// Pin 9. Sortida PWM motor 2
int  SENSOR_DAVANT = A0;// Pin 0. Entrada analÃ²ica sensor davanter
int  SENSOR_DRETA = A2;// Pin 1. Entada analÃ²gica sensor part dreta
int  SENSOR_ESQUERRA = A1;// Pin 2. Entrada analÃ²gica sensor part esquerra
int  VAL_ENCODER_M1 = A3;
int  VAL_ENCODER_M2 = A4;
int VALOR_SENSOR_DAVANT;// Valor transferit entrada analÃ²gica sensor davanter
int VALOR_SENSOR_DRETA;// Valor transferit entrada analÃ²gica sensor part dreta
int VALOR_SENSOR_ESQUERRA;// Valor transferit entrada analÃ²gica sensor part esquerra
int  VELOCITAT_REAL_M1 = 0;
int  VELOCITAT_REAL_M2 = 0;
int  CONTROL;// Guarda el valor del Byte de control de la trama rebuda pel port sèrie
int  LEDPIN = 13;
int  AUTO = 0;
byte BYTE_PULSADORS =B00000000;// Byte on es guarad el contingut de tots els
```

```
                                    // pulsadors rebuts pel port sèrie
byte PULS_AVANZ  =B00000000 ;// Pulsador moviment endevant
byte PULS_RETROCES  =B00000000;// Pulsador moviment enrrera
byte PULS_DRETA  =B00000000;// Pulsador moviment dreta
byte PULS_ESQUERRA  =B00000000;// Pulsador moviment esquerra


boolean MEM_FI_ACELERACIO  =false;// Bit indicador funció acceleració realitzada
boolean BIT_ACT_ACELERACIO  =false;// Bit indicador execució funció acceleració
boolean BIT_ACT_DECELERACIO  =false;// Bit indicador execució funció acceleració



float T_INT_TIMER2 = 10.0;// Temps execucució rutina servei interrupció TIMER 2
float T_ACELERACIO = 5.0;// Temps acceleració motors enviada des de el Iphone
float T_DECELERACIO = 3.0;// Temps deceleració motors enviada des de el Iphone
float CONSG_VELOCITAT = 150.0;// Consigna velocitat enviada des de el Iphone
float CONSG_VELOCITAT_M1;// Consigna velocitat control PI motor 1
float CONSG_VELOCITAT_M2;// Consigna velocitat control PI motor 2
float  INFO;// Guarda el valor dels Bytes d'informació de la trama rebuda pel port sèrie
float DECREMENT_VEL = 10.0;

long CLK_MILIS = 0;
long TEMPS_CAPTURA_F1 = 0;
long TEMPS_CAPTURA_F2 = 0;
long TEMPS_CAPTURA_D1 = 0;
long TEMPS_CAPTURA_D2 = 0;
long TEMPS_CAPTURA_E1 = 0;
long TEMPS_CAPTURA_E2 = 0;

long TIMER = 0;

int F = 0;
int D = 0;
```

```
int E = 0;
boolean GIR_DRETA =false;
boolean GIR_ESQUERRA =false;


//*****************************************************************************************************
//************************** RUTINA SERVEI INTERRUPCIÓ TIMER 2. ***************************************
//*****************************************************************************************************

void rsi_timer_2 ()
{

  CLK_MILIS = CLK_MILIS + 10;

  if (BIT_ACT_ACELERACIO ==true)
  {
   rutina_Aceleracio (CONSG_VELOCITAT,T_INT_TIMER2,T_ACELERACIO);
  }
  if (BIT_ACT_DECELERACIO ==true)
  {
   rutina_Deceleracio (CONSG_VELOCITAT,T_INT_TIMER2,T_DECELERACIO);
  }

}


//*****************************************************************************************************
//*****************************************************************************************************
//*****************************************************************************************************

void setup()
{

  Serial.begin (9600);// Velocitat de comunicació del port serie
```

```
  MsTimer2::set(10, rsi_timer_2);// Habilita interrupcions Timer 2. Periode 10 ms
  MsTimer2::start();// Activa interrupcions Timer 2

   pinMode (MOTOR_1_AVANZ,OUTPUT);// Configura com entrades digitals
   pinMode (MOTOR_1_RETROCES,OUTPUT);
   pinMode (MOTOR_1_SPEED,OUTPUT);
   pinMode (MOTOR_2_AVANZ,OUTPUT);
   pinMode (MOTOR_2_RETROCES,OUTPUT);
   pinMode (MOTOR_2_SPEED,OUTPUT);
   pinMode (SENSOR_DAVANT,INPUT);
   pinMode (SENSOR_DRETA,INPUT);
   pinMode (SENSOR_ESQUERRA,INPUT);
   pinMode (LEDPIN,OUTPUT);
   digitalWrite(MOTOR_1_AVANZ,LOW);
   digitalWrite(MOTOR_1_RETROCES,LOW);
   digitalWrite(MOTOR_2_AVANZ,LOW);
   digitalWrite(MOTOR_2_RETROCES,LOW);


}



//*************************************************************************************************
//*********************************************PROGRAMAPRINCIPAL*********************************
//*************************************************************************************************

void loop ()
{

  rutina_Recepcio_Port_Serie ();// Crida a la rutina de recepció del port sèrie

     switch  (CONTROL)// Selecciona el valor de la paraula de control
   {
```

```
case 65:// Temps d'acceleració
  T_ACELERACIO = INFO / 10;
  break;

case 67:// Paraula control pulsadors moviment

   BYTE_PULSADORS =byte(INFO);
   PULS_AVANZ = BYTE_PULSADORS &B00000001;    // Mascara AND per obtenir el estat
                                              // de cada pulsador contingut en el Byte
   PULS_RETROCES = BYTE_PULSADORS &B00000010;
   PULS_DRETA = BYTE_PULSADORS &B00000100;
   PULS_ESQUERRA = BYTE_PULSADORS &B00001000;
  break;

case 68://Temps de deceleració
  T_DECELERACIO = INFO / 10.0;
  break;

case 75://Constant proporcional
  Kp = INFO / 10.0;
  break;

case 84://Temps d'integració
  Ti = INFO / 10.0;
  break;

case 86:// Consigna velocitat
  CONSG_VELOCITAT = ((INFO / 10) * 250.0) / 10.0;
  break;

case 87:
   AUTO = INFO;
  break;
}
```

```
  rutina_Control_Moviment ();// Crida a la rutina de control de moviment
}

//*************************************************************************************************
//*****************************************RUTINA CONTROL MOVIMENT*********************************
//*************************************************************************************************


void rutina_Control_Moviment ()
{
  if (PULS_AVANZ != 0)// Si pulsador d'avanç esta activat
  {
    if (digitalRead(MOTOR_1_RETROCES) == LOW  &&digitalRead(MOTOR_2_RETROCES) == LOW)
    // I si els bits de retroces no estan actius
    {
      rutina_Avanz ();// Crida a la rutina control avanç
    }
  }
  else // Si pulsador d'avanç no està activat
  {
    if (digitalRead(MOTOR_1_AVANZ) == HIGH  &&digitalRead(MOTOR_2_AVANZ) == HIGH)
    // I si els bits d'avaç estan actius
    {
       BIT_ACT_ACELERACIO =false;
       BIT_ACT_DECELERACIO =true;
      rutina_Control_Velocitat_PID ();

    }
  }
  if (PULS_RETROCES != 0)// Si pulsador de retroces esta activat
   {
    if (digitalRead(MOTOR_1_AVANZ) == LOW  &&digitalRead(MOTOR_2_AVANZ) == LOW)
    // I si els bits d'avaç no estan actius
```

```
    {
      rutina_Retroces();// Crida a la rutina control retroces
    }
  }
  else // Si pulsador de retroces no està activat
  {
    if (digitalRead(MOTOR_1_RETROCES) == HIGH  &&digitalRead(MOTOR_2_RETROCES) == HIGH)
    // I si els bits de retroces estan actius
    {
      BIT_ACT_ACELERACIO =false;
      BIT_ACT_DECELERACIO =true;
     rutina_Control_Velocitat_PID ();
    }
  }
}

//*****************************************************************************************************
//*****************************************RUTINA AVANÇ MOTORS ****************************************
//*****************************************************************************************************

void rutina_Avanz ()
{
  digitalWrite(MOTOR_1_AVANZ,HIGH); // Convinació bits avanç
  digitalWrite(MOTOR_1_RETROCES,LOW);
  digitalWrite(MOTOR_2_AVANZ,HIGH);
  digitalWrite(MOTOR_2_RETROCES,LOW);

  if (MEM_FI_ACELERACIO ==false)
  // Si la rampa d'acceleració no ha arribat al valor màxim
   {
    BIT_ACT_ACELERACIO =true; // Activa el bit d'execució rutina aceleracio
     rutina_Control_PWM ();// Crida a la rutina de control de velocitat pwm
  }
  else // Si la rampa d'acceleració no a arrivat al màxim
```

```
  {
     rutina_Control_PWM ();// Crida a la rutina de control pwm
  }
}

//*************************************************************************************************
//*************************************RUTINA RETROCES MOTORS ***********************************
//*************************************************************************************************

void rutina_Retroces ()
{
  digitalWrite(MOTOR_1_RETROCES,HIGH);// Convinació bits retrocès
  digitalWrite(MOTOR_1_AVANZ,LOW);
  digitalWrite(MOTOR_2_RETROCES,HIGH);
  digitalWrite(MOTOR_2_AVANZ,LOW);

  if (MEM_FI_ACELERACIO ==false)
  // Si la rampa d'acceleració no ha arribat al valor màxim
  {
    BIT_ACT_ACELERACIO =true;// Activa el bit d'execució rutina aceleracio
     rutina_Control_PWM ();// Crida a la rutina de control de velocitat pwm
  }
  else // Si la rampa d'acceleració no a arrivat al màxim
  {
     rutina_Control_PWM ();// Crida a la rutina de control de velocitat pwm
    }
}

//*************************************************************************************************
//*************************************RUTINAACELERACIO*******************************************
//*************************************************************************************************

// VEL_MAX = Velocitat maxima per calcul rampa acceleració
// Tint = Temps del periode de la rutina servei interrupció Timer 2
```

```
// Ta = Temps d'acceleració
// X = Iteracions  per fer els increments de velocitat
// I = Valor del increment de velocitat
// Tint * X = Ta
// X = Ta / Tint
// I = VEL_MAX / X

int  X;// Iteracions  per fer els increments de velocitat
float  I;// Valor del increment de velocitat

void rutina_Aceleracio (float VEL_MAX,float  Tint,float Ta)
{
  if (CONSG_VELOCITAT_M1 && CONSG_VELOCITAT_M2 >= VEL_MAX)
  // Si la velocitat dels motors es >= a la veloctat màxima
    {
    MEM_FI_ACELERACIO =true;  // Activa el bit indicador d'acceleració finalitzada
    BIT_ACT_ACELERACIO =false; // Desactiva  el bit d'execució rutina acceleració
  }
  else
  {
    BIT_ACT_DECELERACIO =false;// Desactiva el bit d'execució rutina deceleració
   X = Ta / (Tint/1000.0);// Calcul número de execucions per arribar al temps d'acceleració
    I = VEL_MAX / X;//     Calcul     d'increments     de     velocitat     per     accelerar
    CONSG_VELOCITAT_M1 = CONSG_VELOCITAT_M1 + I;// Increment de velocitat motor 1
    CONSG_VELOCITAT_M2 = CONSG_VELOCITAT_M2 + I;// Increment de velocitat motor 2
  }
}

//*************************************************************************************************
//*********************************************RUTINADECELERACIO*********************************
//*************************************************************************************************

// VEL_MAX = Velocitat maxima per calcul rampa aceleració
// Tint = Temps del periode de la rutina servei interrupció Timer 2
```

```
// Ta = Temps d'acceleració
// X = Iteracions  per fer els increments de velocitat
// I = Valor del increment de velocitat
// Tint * X = Ta
// X = Ta / Tint
// I = VEL_MAX / X

void rutina_Deceleracio (float VEL_MAX,float  Tint,float Ta)
{
  if (CONSG_VELOCITAT_M1 && CONSG_VELOCITAT_M2 < 5)
  // Si la velocitat dels motors es igual a zero
  {
    CONSG_VELOCITAT_M1 = 0;
    CONSG_VELOCITAT_M2 = 0;
     BIT_ACT_DECELERACIO =false;// Desactiva  el bit d'execució rutina deceleració
    if (digitalRead(MOTOR_1_AVANZ) == HIGH  &&digitalRead(MOTOR_2_AVANZ) == HIGH)
    // Si els bits d'avanç dels motors estan actius
    {
      digitalWrite(MOTOR_1_AVANZ,LOW);// Desactivo bits avanç motors
      digitalWrite(MOTOR_2_AVANZ,LOW);
    }
    if (digitalRead(MOTOR_1_RETROCES) == HIGH  &&digitalRead(MOTOR_2_RETROCES) == HIGH)
    // Si els bits de retrocès dels motors estan actius
    {
      digitalWrite(MOTOR_1_RETROCES,LOW);// Desactivo bits de retrocès motors
      digitalWrite(MOTOR_2_RETROCES,LOW);
    }
  }
  else
  {
    X = Ta / (Tint/1000.0);// Calcul número de execucions per arribar al temps d'aceleració
     I = VEL_MAX / X;// Calcul de decrements de velocitat per decelerar
     MEM_FI_ACELERACIO =false;// Desactiva el bit indicador d'acceleració finalitzada
    CONSG_VELOCITAT_M1 = CONSG_VELOCITAT_M1 - I;// Decrement de velocitat motor 1
```

```
    CONSG_VELOCITAT_M2 = CONSG_VELOCITAT_M2 - I;// Decrement de velocitat motor 2
  }
}

//*************************************************************************************************
//************************************ RUTINA CONTROL VELOCITAT PWM ********************************
//*************************************************************************************************

void rutina_Control_PWM ()
{

  float VAL_1;
  float VAL_2;

 rutina_Adquisicio_Sensors();

  if(PULS_ESQUERRA != B00000000 || GIR_ESQUERRA ==true)
  {
    VAL_1 =  (DECREMENT_VEL * CONSG_VELOCITAT_M1) / 100.0;
  }
  else
  {
    VAL_1 = CONSG_VELOCITAT_M1;
  }

  if(PULS_DRETA != B00000000 || GIR_DRETA ==true)
  {
    VAL_2 = (DECREMENT_VEL * CONSG_VELOCITAT_M2) / 100.0;
  }
  else
  {
    VAL_2 = CONSG_VELOCITAT_M2;
  }
```

```
  analogWrite(MOTOR_1_SPEED,int(VAL_1));
  analogWrite(MOTOR_2_SPEED,int(VAL_2));
}

//*************************************************************************************************
//************************************RUTINA ADQUISICIÓ SENSORS ***********************************
//*************************************************************************************************


void rutina_Adquisicio_Sensors ()
{

  VALOR_SENSOR_DAVANT =analogRead  (SENSOR_DAVANT);// Lectura sensors analÃ²gics
  VALOR_SENSOR_DRETA =analogRead (SENSOR_DRETA);
  VALOR_SENSOR_ESQUERRA =analogRead (SENSOR_ESQUERRA);

  if ( VALOR_SENSOR_DAVANT >= 200)
  {
    F = 1;
   TEMPS_CAPTURA_F1 = CLK_MILIS;
  }
  else
  {
   TEMPS_CAPTURA_F2 = CLK_MILIS;
    if (TEMPS_CAPTURA_F2 - TEMPS_CAPTURA_F1 > 100)
    {
      F = 0;
    }
  }


  if ( VALOR_SENSOR_ESQUERRA >= 400)
  {
```

```
  E= 1;
  TEMPS_CAPTURA_E1 = CLK_MILIS;
}
else
{
  TEMPS_CAPTURA_E2 = CLK_MILIS;
   if (TEMPS_CAPTURA_E2 - TEMPS_CAPTURA_E1 > 100)
   {
     E = 0;
   }
}


if ( VALOR_SENSOR_DRETA >= 400)
{
  D = 1;
  TEMPS_CAPTURA_D1 = CLK_MILIS;
}
else
{
  TEMPS_CAPTURA_D2 = CLK_MILIS;
   if (TEMPS_CAPTURA_D2 - TEMPS_CAPTURA_D1 > 100)
   {
     D = 0;
   }
}


if((D == true && E ==false ) || (F ==true && E ==false) || (F ==true && D ==true))
{
   GIR_ESQUERRA =true;
}
else
{
```

```
      GIR_ESQUERRA =false;
    }
    if((D == false &&  E  ==true)  ||  (F  ==true  &&  E  ==true))
    {
      GIR_DRETA =true;
    }
    else
    {
      GIR_DRETA =false;
    }

}

//*************************************************************************************************
//************************************ RUTINA RECEPCIÖ PORT SÈRIE ***********************************
//*************************************************************************************************

//*****************************************TRAMA COMUNICACIÖ*****************************************
//   8 bits    *  8 bits  *    8 bits   *   8 bits   *   8 bits   *   8 bits   *       8 bits       *  8 bits  *
// INICI TRAMA *  CONTROL *     INFO    *    INFO    *    INFO    *    INFO    *   CONTROL ERRORS   * FI TRAMA *
//*************************************************************************************************

// ********* CODIS DEL BYTE DE CONTROL DE LA TRAMA *****************
// HEX  ASCII  FUNCIÖ
//*****************************************************************
// 02h = STX = Inici de trama
// 03h = ETX = Final de trama
// O6h = ACK = Trama rebuda correctament
// 15h = NACK = Trama rebuda incorrectament
// 65h = A = Temps d'acceleració
// 68h = D = Temps de deceleració
// 67h = C = Paraula de control estat pulsadors moviment
// 86h = V = Consigna velocitat
// 75h = K = Constant proporcional
```

```
// 84h = T = Temps Integració


void rutina_Recepcio_Port_Serie ()
{

  int TRAMA[10];// Trama rebuda a través del port sèrie
  int i  =  0;
  int checksum = 0;
  int  var;


  while  (Serial.available() >0)
  {
     var =Serial.read();
    delay(15);
    TRAMA[i]=var;
    i++;

    if(i==8)
       {
      if (TRAMA[7] == 3 && TRAMA[0] == 2)
      {
        for(i=0;i<=5;i++)
        {
          checksum = checksum + TRAMA[i];
        }
        if(checksum == TRAMA[6])
        {
          CONTROL = TRAMA[1];
          TRAMA[2] = TRAMA[2]*1000;
          TRAMA[3]= TRAMA[3]*100;
          TRAMA[4]= TRAMA[4]*10;
          TRAMA[5]= TRAMA[5]*1;
```

```
          INFO = (TRAMA[2] + TRAMA[3] + TRAMA[4] + TRAMA[5]);
           Serial.print(6);
         }
         else
         {
           Serial.print(15);
         }
       }
       Serial.flush();
        i=0;
     }
   }
}
```