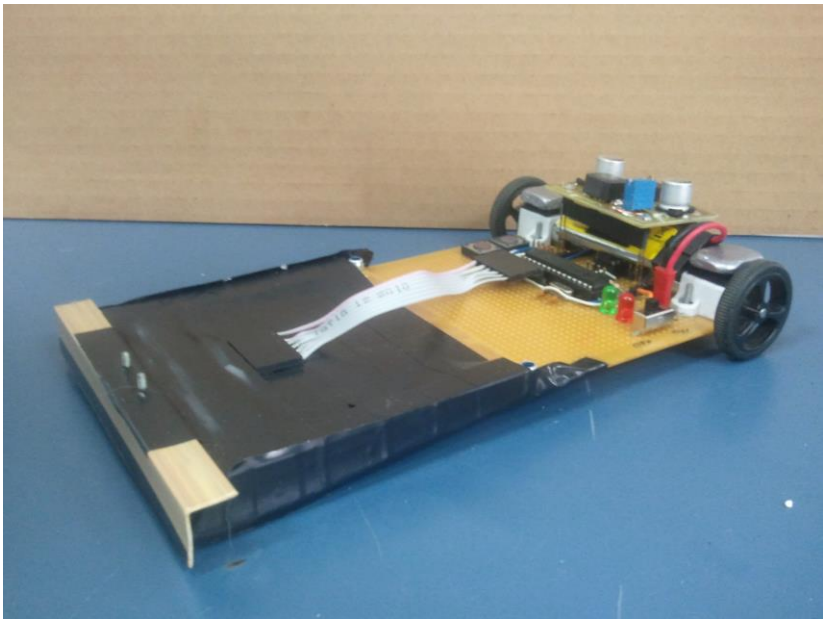
Competencia de robótica

JOHNNY 5

~2015~

Categoría: **Velocista**

Nombre robot: **El Duque**



Institución: **Escuela ORT, sede Almagro**

Mariano Fouiller (mfouiller@ort.edu.ar)

Mecánica

- Dimensiones: 120 x 200mm. Altura 50mm
- Ruedas de 35mm de diámetro
- Estructura: 2 plaquetas de circuito impreso universal
- Tracción diferencial (dos motores en la parte posterior)
- Punto de apoyo, tipo ball-caster
- Peso: 200 g

Electrónica

- Micro Controlador PIC18F2550
- Driver de motores dc L293D
- Elevador DC-DC
- Regulador lineal 5v (7805)
- 5 Sensores reflectivos CNY70
- Modulo UART inalámbrico (WIXEL).

Funcionamiento básico

Básicamente el funcionamiento tiene 4 etapas:

Calibración de los sensores:

Se utilizan 5 sensores, cada sensor responde de manera parecida pero no idéntica al color detectado (emiten luz infrarroja y miden la intensidad de la luz recibida). Al presionar el pulsador, el robot barre 180º con sus motores, y durante ese movimiento captura muchas muestras de cada sensor, obteniendo al final un valor máximo y otro mínimo para cada uno individuamente. Estos valores son almacenados en RAM y luego utilizados.

Luego, se presiona una vez más el pulsador y el programa entra en un loop continuo ejecutando 2 funciones principales:

Adquisición, ajuste y procesamiento de la posición:

Se leen los sensores, utilizando los máximos y mínimos obtenidos en el proceso de calibración, se ajusta la lectura de cada sensor, con una ecuación lineal a un valor entre 0 y 1000 (0 totalmente negro – 1000 totalmente blanco). Con esos 5 valores entre 0 y 1000, se realiza una cuenta, ponderando la posición de los sensores y su valor, para obtener finalmente un número entero entre 0 y 4000 que indica el lugar donde se encuentra el robot respecto de la línea, siendo 0, 1000, 2000, 3000 y 4000 los valores para los cuales el sensor 0, 1, 2, 3, 4 está justo arriba de la línea blanca respectivamente.

Aplicación de ecuación de corrección:

Al ser el sensor 2 el “centro” del robot, nuestro objetivo es mantener el valor de la lectura en 2000. Para esto se resta el valor 2000 al valor calculado (Línea), quedando un valor entre -2000 y 2000, siendo los extremos derecho e izquierdo, y el centro “0”.
Este valor (entre -2000 y 2000) indica cuan lejos estamos del valor ideal (0), por lo que lo multiplicamos por un factor que indica que debemos hacer con los motores en función de cuan alejados estamos del centro. A esto se lo llama constante proporcional. Además, se calcula la diferencia entre los dos últimos valores de la línea, y se obtiene un término derivativo, que nos permite tener una idea de la variación entre una lectura y la siguiente. Este factor, multiplicado por la constante derivativa, se suma también a la corrección.

Modificación de velocidad de los motores:

El valor obtenido se considera la diferencia de velocidad entre el motor derecho e izquierdo. Un motor siempre irá al máximo y el otro más lento (según lo calculado por la ecuación)

NOTA: se cuenta con la posibilidad de enviar valores por la UART para debug de forma inalámbrica (modulo WIXEL)

Componentes y precios

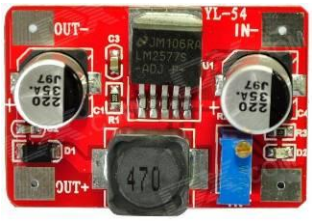
Costo total: $576

Microcontrolador PIC 18F2550 ($85)

Driver motores dc L293D ($70)

http://articulo.mercadolibre.com.ar/MLA-460868479-l293d-l293-puente-medio-h-motores-robotica-hobby-_JM

Elevador de tensión DC-DC ($25)
(LM2577-adj)



http://dx.com/p/lm2577-dc-3-5-18v-to-dc-4-0-24v-voltage-step-up-boost-module-red-154906

Micro Motores 30:1 ($170)

http://www.pololu.com/catalog/product/1098

Soporte para motores ($26)

http://www.pololu.com/catalog/product/1089

Ruedas ($42)

http://www.pololu.com/catalog/product/1420

Punto de apoyo – rueda loca ($16)

http://www.pololu.com/catalog/product/951

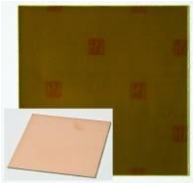
Modulo Wixel ($106)

http://www.pololu.com/catalog/product/1337

Bateria LiPo 3,7v 600mAh ($16)

http://dx.com/p/tiger-5001s-1s-3-7v-15c-600mah-li-ion-polymer-battery-pack-for-r-c-aircraft-more-3-pcs-180594

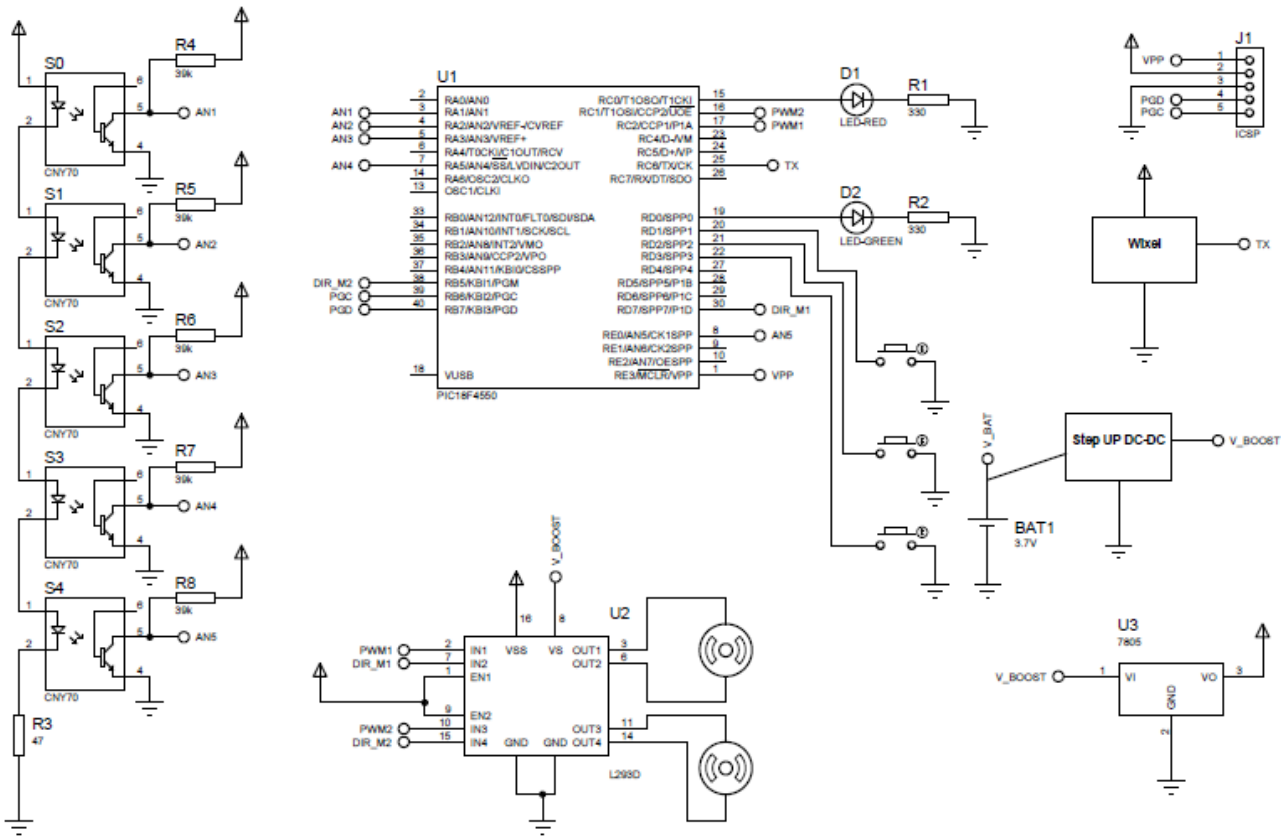
Placa circuito impreso FR4 ($20)



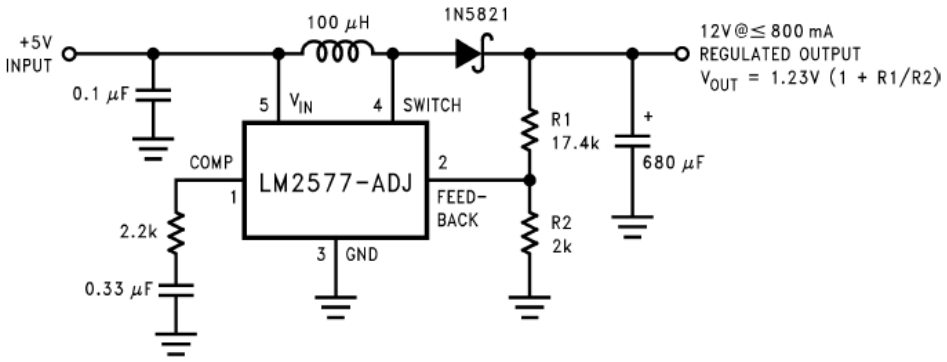
http://articulo.mercadolibre.com.ar/MLA-463315256-plaquetas-virgenes-epoxi-10x10-cm-de-fibra-de-vidrio-oferta-_JM

Circuitos Esquemáticos

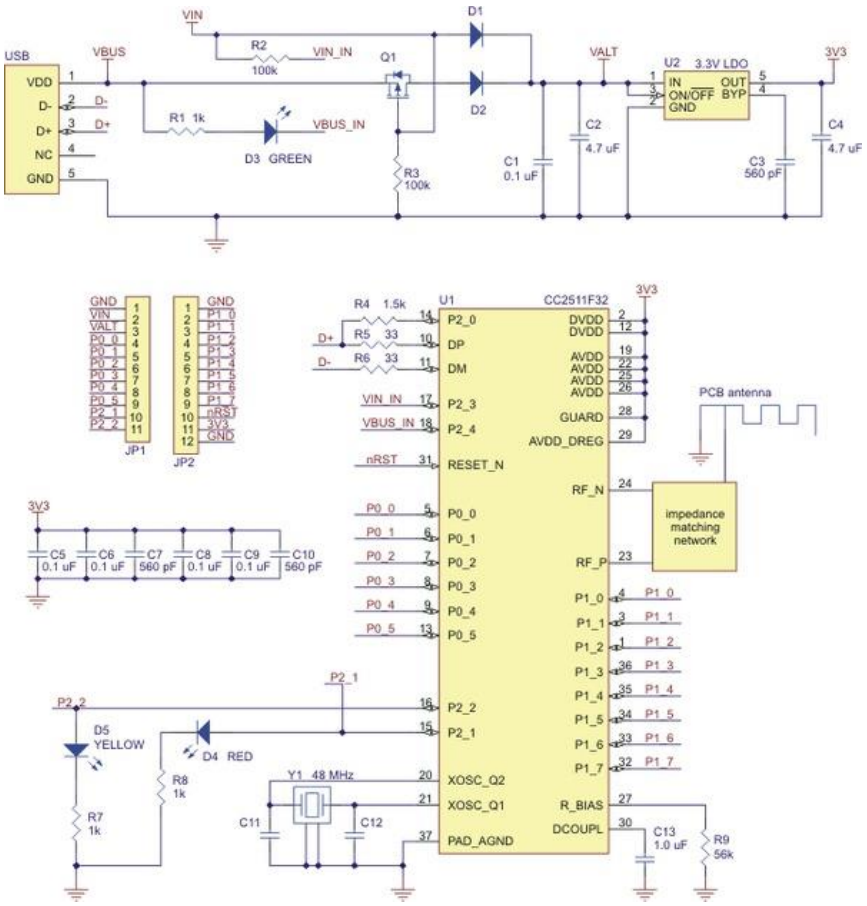
Placa Principal



Elevador de tensión DC-DC



WIXEL



Código fuente

```
#define _XTAL_FREQ 48000000
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>

#pragma config FOSC=HSPLL_HS, PLLDIV=3, CPUDIV=OSC1_PLL2 //para crystal de 12m y salida de 48m
#pragma config IESO=OFF, FCMEN=OFF, USBDIV = 2
#pragma config PWRT=ON, BOR=OFF, BORV=46, VREGEN=OFF, WDT=OFF
#pragma config MCLRE=OFF, XINST=OFF, LVP=OFF, PBADEN=OFF
#pragma config WDTPS =32768, LPT1OSC=OFF, DEBUG =OFF, STVREN = OFF, ICPRT =OFF
#pragma config LVP = OFF, CP0 = OFF,CP1 = OFF,CPB = OFF,WRT0 = OFF,WRT1 = OFF,WRTB = OFF
#pragma config WRTC = OFF,EBTR0 = OFF,EBTR1 = OFF,EBTRB = OFF



//=== PINES ==
#define BOTON_DER RD1
```

```
#define BOTON_IZQ RD3
#define BOTON_AUX RD2
#define LED_DER LATD0
#define LED_IZQ LATC0
#define LED_ROJO LATB3
#define LED_AMA LATB4


#define PWM1 LATC2
#define PWM2 LATC1
#define DIR_M1 LATD7
#define DIR_M2 LATB5
#define ADELANTE 0
#define ATRAS 1

#define CANAL_POTE 0


// == TRIS ==
#define TRIS_BOTON_DER TRISD1
#define TRIS_BOTON_IZQ TRISD3
#define TRIS_BOTON_AUX TRISD2

#define TRIS_LED_DER TRISD0
#define TRIS_LED_IZQ TRISC0
#define TRIS_LED_ROJO TRISB3
#define TRIS_LED_AMA TRISB4

#define TRIS_POTE TRISA0

#define TRIS_AN0 TRISA0
#define TRIS_AN1 TRISA1
#define TRIS_AN2 TRISA2
#define TRIS_AN3 TRISA3
#define TRIS_AN4 TRISA5
#define TRIS_AN5 TRISE0
#define TRIS_AN6 TRISE1
#define TRIS_AN7 TRISE2
#define TRIS_AN8 TRISB2


#define TRIS_RX TRISC7
#define TRIS_TX TRISC6

#define TRIS_PWM1 TRISC2
#define TRIS_PWM2 TRISC1

#define TRIS_DIR_M1 TRISD7
#define TRIS_DIR_M2 TRISB5

//===== ESTADOS ======


#define INICIAL 1
#define CALIBRACION 2
```

```
#define ARRANQUE 3
#define ANDANDO 4
#define DETENIDO 5
#define MOSTRAR 6



//=== PROTOTIPOS DE FUNCIONES

void ConfigurarIO(void);
void Configurar_UART(void);
void Configurar_Timer2(void);
void Configurar_PWM(void);
void Configurar_Duty(unsigned char canal,unsigned int duty);
void Configurar_Interrupciones(void);
void Configurar_AD(void);
unsigned int Leer_Pote(void);
void Leer_Sensores(void);
void Setear_Motor(unsigned char mot,int vel);
void Maquina(void);
void Hola(void);
void Motores(int m1,int m2);
void Calibrar(void);
void Leer_Linea(void);
void Seguir_Linea(void);




//==== VARIABLES GLOBALES =====
unsigned int Contador=0,cont2=0,Tiempo=0;
unsigned char Estado,i;
unsigned int Sensores[5],Maximos[5],Minimos[5],Linea;
bit Flag_Nuevo_Estado=0;
int Vel_M1,Vel_M2,Vel_max;

void main(void)
{

/*ConfigurarIO();
while(TRUE)
        {
                LED_DER=0; // Apagamos led
      __delay_ms(10);
      LED_DER=1; // Encendemos led
      __delay_ms(10);
        }*/

        ConfigurarIO();
        Configurar_UART();
        Configurar_Timer2();
        Configurar_PWM();
        Configurar_Duty(1,0);
        Configurar_Duty(2,0);
        Configurar_Interrupciones();
```

```
    Configurar_AD();
        __delay_ms(10);


        Estado=INICIAL;
        Flag_Nuevo_Estado=1;
        Contador=0;
        Hola();
        T2CONbits.TMR2ON=1;
    while(TRUE)
        {
            Maquina();
    }
}


void Maquina(void)
{
switch (Estado)
{
        case INICIAL:
            if (Flag_Nuevo_Estado==1) {Flag_Nuevo_Estado=0;  /*printf("\rHOLA MUNDO\r");*/}


            //if (BOTON_AUX==0) Estado=MOSTRAR;


            if (BOTON_DER==0)
                {
                    Estado=CALIBRACION;
                    Flag_Nuevo_Estado=1;
                }


        /*      if (BOTON_IZQ==0)
                {
                    Contador=0;
                    Leer_Sensores();
                    for (i=0;i<5;i++) printf("%d: %u\r",i,Sensores[i]);
                    while(Contador<500) {}

                }*/

        break;

        case CALIBRACION:

            if (Flag_Nuevo_Estado==1)
                {
                    Flag_Nuevo_Estado=0;
                    //printf("\rCALIBRAR\r");
                    Hola();
                    Calibrar();
                    //printf("\r");
                    //for (i=0;i<5;i++) printf("s%d  MAX: %04u  MIN: %04u\r",i,Maximos[i],Minimos[i]);
```

```
                        //printf("\r");
                }

        if (BOTON_DER==0)
                {
                        Estado=CALIBRACION;
                        Flag_Nuevo_Estado=1;
                }

        if (BOTON_AUX==0) {Estado=MOSTRAR; Flag_Nuevo_Estado=1;}

break;


case MOSTRAR:
        if (Flag_Nuevo_Estado==1)
                                        {
                                                Flag_Nuevo_Estado=0;
                                                //printf("\rMOSTRAR\r");
                                                Contador=0;
                                                Tiempo=0;
                                                LED_ROJO=0;
                                                LED_AMA=1;
                                        }

        if (Contador==100)
                {
                        Contador=0;
                        Leer_Linea();
                        if (Linea>1900 && Linea<2100) LED_ROJO=1;
                        else LED_ROJO=0;

                        //printf("%u\r",Linea);
                        //for (i=0;i<5;i++) printf("%04u  ",Sensores[i]);
                        //===printf("%04u\r",Linea);
                        LED_AMA=!LED_AMA;
                }

        if (BOTON_IZQ==0)
                {
                        Estado=ARRANQUE;
                        Flag_Nuevo_Estado=1;
                        Vel_max=800;
                }

        if (BOTON_DER==0)
                {
                        Estado=ARRANQUE;
                        Flag_Nuevo_Estado=1;
                        Vel_max=800;
                }


break;
```

```
	case ARRANQUE:
		if (Flag_Nuevo_Estado==1)
				{
					Hola();
					Flag_Nuevo_Estado=0;
					//printf("\rARRANQUE\r");
					Contador=0;
					Tiempo=0;
					LED_ROJO=0;
					LED_AMA=0;
				}

		Seguir_Linea();

		if (Tiempo>=10000)
		{
			Estado=MOSTRAR;
			Flag_Nuevo_Estado=1;
			LED_DER=0;
			LED_IZQ=0;
			Motores(0,0);
			return;
		}

	break;



}//fin switch


}




void Seguir_Linea(void)
{
float Kp,Kd,Ki;
static int prop=0,deriv,old_prop=0;
static long integral=0;
int PID;



if (Contador==5)
	{
		Contador=0;
		Leer_Linea();
		prop=Linea-2000;

		if (abs(prop)<100) {LED_DER=1; LED_IZQ=1;}
		else if (prop>0) {LED_IZQ=0; LED_DER=1; }
			else {LED_IZQ=1; LED_DER=0;}
```

```
                deriv=prop-old_prop;
                old_prop=prop;
                integral+=prop;
                //printf("%05d\r",prop);

                Kp=7;
                Ki=0.5;
                Kd=0.25;
                Vel_max=950;

                PID=prop*Kp + deriv*Kd + integral*Ki;

                printf("%d\r",Linea);

                if (PID>Vel_max) PID=Vel_max;
                if (PID<-Vel_max) PID=-Vel_max;

                if (PID<0) Motores(Vel_max,Vel_max+PID);
                else Motores(Vel_max-PID,Vel_max);
        }
}




void Leer_Linea(void)
{

char enlinea=0;
unsigned char i,k,j;
unsigned long aux[5];
unsigned long divisor;
unsigned int suma;

Leer_Sensores();

for (j=0;j<5;j++)
                                        {
                                                Ajustar_Valores(Sensores[j];
                                        }

for (i=0;i<5;i++) if (Sensores[i]<50) Sensores[i]=0;

suma=0;
divisor=0;

for (j=0;j<5;j++)
                                        {
                                                suma+=Sensores[j];
                                                divisor+=(unsigned long)Sensores[j]*j*1000;
                                        }
```

```
for (i=0;i<5;i++){if (Sensores[i]>100) enlinea=1;}

if (enlinea==1) Linea=divisor/suma;
else if (Linea>2000) Linea=4000;
else Linea=0;

}




void Calibrar(void)
{
unsigned char i,j,k;
unsigned int prom[5];


Contador=0;
LED_ROJO=0;
LED_AMA=1;
for (i=0;i<100;i++)
        {

        for (j=0;j<5;j++) prom[j]=0;

        for (k=0;k<3;k++)
         {
                Leer_Sensores();
                for (j=0;j<5;j++) prom[j]+=Sensores[j];
         }

        for (j=0;j<5;j++) Sensores[j]=prom[j]/3;

        for (j=0;j<5;j++)
                                        {
                                                if (Sensores[j]<Minimos[j]) Minimos[j]=Sensores[j];
                                                if (Sensores[j]>Maximos[j]) Maximos[j]=Sensores[j];
                                        }

                if (i<50) Motores(300,-300);
                else Motores(-300,300);

                while (Contador<20) {}
                Contador=0;
        }
Motores(0,0);
LED_ROJO=1;
LED_AMA=0;
}



void Hola(void)
{
char i,j;
```

```
LED_DER=0;
LED_IZQ=1;
Contador=0;

for (i=0;i<5;i++)
          {
           for (j=0;j<20;j++) __delay_ms(10);
    LED_DER=!LED_DER;
           LED_IZQ=!LED_IZQ;
   }

LED_DER=0;
LED_IZQ=0;
}




void ConfigurarIO(void)
{
nRBPU=0;
RDPU=1;

TRIS_BOTON_DER=1;
TRIS_BOTON_IZQ=1;
TRIS_BOTON_AUX=1;

TRIS_AN0=1;
TRIS_AN1=1;
TRIS_AN2=1;
TRIS_AN3=1;
TRIS_AN4=1;
TRIS_AN5=1;
TRIS_AN6=1;
TRIS_AN7=1;
TRIS_AN8=1;

TRIS_RX=1;
TRIS_TX=1;

TRIS_PWM1=0;
TRIS_PWM2=0;
PWM1=0;
PWM2=0;

TRIS_DIR_M1=0;
TRIS_DIR_M2=0;
DIR_M1=0;
DIR_M2=0;

TRIS_LED_DER=0;
TRIS_LED_IZQ=0;
```

```
TRIS_LED_AMA=0;
TRIS_LED_ROJO=0;
LED_DER=0;
LED_IZQ=0;
LED_AMA=0;
LED_ROJO=0;
}


void Configurar_UART(void)
{
BAUDCONbits.RXDTP=0; //Asynchronous mode: 1 = RX data is inverted  0 = RX data received is not inverted
BAUDCONbits.TXCKP=0; //XCKP: Clock and Data Polarity Select bit Asynchronous mode: 1 = TX data is inverted  0 = TX data is not inverted
BAUDCONbits.BRG16=1; //BRG16: 16-Bit Baud Rate Register Enable bit 1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG 0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored
BAUDCONbits.WUE=0; //WUE: Wake-up Enable bit
BAUDCONbits.ABDEN=0; //Auto-Baud Detect Enable bit
SPBRG=51;  // a 48MHZ da 57600 bps
SPBRGH=0;

TXSTAbits.CSRC=0;
TXSTAbits.TX9=0; //1 = Selects 9-bit transmission 0 = Selects 8-bit transmission
TXSTAbits.TXEN=1; //1 = Transmit enabled  0 = Transmit disabled
TXSTAbits.SYNC=0; //1 = Synchronous mode  0 = Asynchronous mode
TXSTAbits.SENDB=0; //1 = Send Sync Break on next transmission (cleared by hardware upon completion)  0 = Sync Break transmission completed
TXSTAbits.BRGH=0;  //1 = High speed 0 = Low speed

RCSTAbits.RX9=0;
RCSTAbits.CREN=1; // CREN: Continuous Receive Enable bit Asynchronous mode: 1 = Enables receiver 0 = Disables receiver
RCSTAbits.SPEN=1; //1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)  0 = Serial port disabled (held in Reset)
}

void putch(char data)
{
        while( ! TXIF)
        continue;
        TXREG = data;
}

void Configurar_Timer2(void)
{
        T2CONbits.TOUTPS=11; //postscale de 1:1 (0) a 1:16 (15) NO SE USA PARA PWM
        T2CONbits.T2CKPS=1; //prescaler 0: 1 ; 1:4 ; 2o3: 16
        T2CONbits.TMR2ON=0;
        PR2=249;
}


void Configurar_PWM(void)
{
        CCPR1L=0;
```

```
	CCP1CONbits.DC1B=0; //These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty
	CCP1CONbits.CCP1M=12; //11xx = PWM mode

	CCPR2L=0;
	CCP2CONbits.DC2B=0; //These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty
	CCP2CONbits.CCP2M=12; //11xx = PWM mode
}


void Configurar_Interrupciones(void)
{
	INTCON=0;
	INTCON2=0;
	INTCON3=0;
	PIE1=0;
	PIE2=0;
	RCONbits.IPEN=0;
	PIE1bits.TMR2IE=1;
	INTCONbits.PEIE_GIEL=1;
	INTCONbits.GIE_GIEH=1;
}



void Configurar_AD(void)
{
ADCON1bits.VCFG0=0;
ADCON1bits.VCFG1=0;

ADCON1bits.PCFG=6;

ADCON2bits.ADFM=0;
ADCON2bits.ACQT=7; //20tads
ADCON2bits.ADCS=6;  //fosc/64

ADCON0bits.CHS=0;
ADCON0bits.GODONE=0;
ADCON0bits.ADON=1;
}




void Configurar_Duty(unsigned char canal,unsigned int duty)
{
	if (duty>1000) duty=1000;

	if (canal==1)
		{
		 CCPR1L=duty/4;
		 CCP1CONbits.DC1B=duty%4;
		}
```

```
        if (canal==2)
                {
                 CCPR2L=duty/4;
                 CCP2CONbits.DC2B=duty%4;
                }
}


void interrupt timer2(void)
{
        TMR2IF=0;
        Contador++;
        Tiempo++;
        return;
}


unsigned int Leer_Pote(void)
{
unsigned int aux;
ADCON0bits.GODONE=1;
while (ADCON0bits.GODONE==1) {}
aux=ADRESH*4;
aux=aux+ADRESL/64;
return aux;
}




void Leer_Sensores(void)
{
unsigned char i;
unsigned int aux;

for (i=1;i<6;i++)
        {
                ADCON0bits.CHS=i;
                ADCON0bits.GODONE=1;
                while (ADCON0bits.GODONE==1) {}
                aux=ADRESH*4;
                aux=aux+ADRESL/64;
                Sensores[i-1]=aux;
        }
}


void Setear_Motor(unsigned char mot,int vel)
{

if (vel>1000) vel=1000;
if (vel<-1000) vel=-1000;
```

```
if (mot==1)
	{
		if (vel>=0)
			{
				Configurar_Duty(1,vel);
				DIR_M1=ADELANTE;
			}
		else
			{
				DIR_M1=ATRAS;
				Configurar_Duty(1,vel+1000);
			}
		if (vel==0) {DIR_M1=ATRAS; Configurar_Duty(1,1000); }

	}

if (mot==2)
	{
		if (vel>=0)
			{
				Configurar_Duty(2,vel);
				DIR_M2=ADELANTE;
			}
		else
			{
				DIR_M2=ATRAS;
				Configurar_Duty(2,vel+1000);
			}
		if (vel==0) {DIR_M2=ATRAS; Configurar_Duty(2,1000); }
	}
}

void Motores(int m1,int m2)
{
	if (m1>1000) m1=1000;
	if (m1<-1000) m1=-1000;

	if (m2>1000) m2=1000;
	if (m2<-1000) m2=-1000;

	Setear_Motor(1,m1);
	Setear_Motor(2,m2);
	Vel_M1=m1;
	Vel_M2=m2;
}
```