DOCUMENTO Nº 6

Anexos

Título del proyecto:

Diseño e Implementación de un Sistema de Control para un dispositivo de transporte equilibrado en dos ruedas teledirigido

Autor:

Francisco Javier Antón Díaz

Índice

6.1.- Código NXC completo

```
// Puertos de entrada y salida
#define Puerto_Giroscopio   IN_3
#define Motor_Izquierdo     OUT_C
#define Motor_Derecho       OUT_A
#define Ambos_Motores       OUT_AC

// Constantes del equilibrio

#define Ki_Giro 17.5
#define Kp_Giro 1.15
#define Ki_Motores 0.07
#define Kp_Motores 0.1
#define KDRIVE -0.02
#define K_Direccion 0.25

// El sensor giroscopio tiene efecto Deriva. Para corregirlo usar la constante Deriva
#define Deriva 0.0005



#define OFFSET_SAMPLES 300
// Esta constante limita el tiempo que el robot no arregla su posicion. Si la supera
// quiere decir que se ha caido
#define LimiteTiempoCaida     1000

// Esta constante es la velocidad maxima en grados por segundo. 300 grados/seg por
motor

#define Velocidad_Maxima  600.0
// Variables globales

// Estas variables son para el movimiento del robot. Se expresan en grados/segundo
   float motorControlDrive = 0.0;
   float motorControlSteer = 0.0;
   float motorDiffTarget = 0.0;
```

```
long Comienzo;
float Intervalo;

// Giroscopio
float Offset_Giroscopio;

// Motor
float Avance_Motores = 0;
long Cuenta_Vueltas_Motores_Suma = 0;
long Cuenta_Vueltas_Motores_Suma_Previa =0;
long Cuenta_Vueltas_Motores_Resta;
long Cuenta_Vueltas_Motores_DP3 = 0;
long Cuenta_Vueltas_Motores_DP2 = 0;
long Cuenta_Vueltas_Motores_DP1 = 0;
//Variables para guardar datos en fichero
byte handle;
short bytes_escritos;

//===============================================================================
=================

void Offset_giroscopio()
{
float Suma_Lecturas;
int  i, Lectura_Minima, Lectura_Maxima, Lectura;
ClearScreen();
TextOut(14, 56, "PROYECTO J&J");
TextOut(8, 16, "--Calibrando--");
Off(Ambos_Motores);
PlayTone(420,1000);
do
    {
    Suma_Lecturas = 0.0;
    Lectura_Minima = 1000;
    Lectura_Maxima = -1000;
    for (i=0; i<OFFSET_SAMPLES; i++)
        {
        Lectura = SensorHTGyro(Puerto_Giroscopio);
```

```
        if (Lectura > Lectura_Maxima)
           {
           Lectura_Maxima = Lectura;
           }
        if (Lectura < Lectura_Minima)
           {
           Lectura_Minima = Lectura;
           }
        Suma_Lecturas += Lectura;
        Wait(5);
        }
    }
    while ((Lectura_Maxima - Lectura_Minima) > 1);
// Hasta que la toma de muestras no es fina no sale del muestreo
     Offset_Giroscopio = Suma_Lecturas / OFFSET_SAMPLES +1;
     PlaySound(SOUND_UP);
     Wait(1000);
     NumOut(0,0,Offset_Giroscopio);
}

//===========================================================================
// Implementa 4 pitidos de 440 Hz y un quinto mas agudo

void Pitidos()
{
int i;
ClearScreen();
TextOut(0, LCD_LINE1, "Proyecto J&J");
TextOut(20, LCD_LINE3, "Ponme en pie");
NumOut(0,0,Offset_Giroscopio);
for (i=5; i>0;i--)
    {
    NumOut(47, LCD_LINE4, i);
    PlayTone(440,100);
    Wait(1000);
    }
NumOut(47, LCD_LINE4, 0);
PlayTone(470,1000);
```

```
Wait(1000);
ClearScreen();
}

//=============================================================================

task main()
{
DeleteFile ("Datos.txt");                    //Borrar el archivo anterior para
iniciar uno nuevo
CreateFile ("Datos.txt", 50000, handle);
SetSensorHTGyro(Puerto_Giroscopio);
Wait(50);
Offset_giroscopio();
Pitidos();

}

//=============================================================================

inline void Lee_giroscopio(float &Velocidad_Giroscopio, float &Angulo_Giroscopio)
{
float Lectura_Giroscopio;
Lectura_Giroscopio = SensorHTGyro(Puerto_Giroscopio);
Offset_Giroscopio = Deriva * Lectura_Giroscopio + (1-Deriva) * Offset_Giroscopio;
Velocidad_Giroscopio = Lectura_Giroscopio - Offset_Giroscopio;
Angulo_Giroscopio += Velocidad_Giroscopio*Intervalo;
}

//=============================================================================

/* El objetivo de Lectura_Motores es contar las vueltas que dan los motores.
   El resultado de la funcion ó el objetivo de la funcion es actualizar la variable
"Velocidad_Motores"
   que es la suma de la velocidad de ambos motores. Suma los datos de los encoders
   de los dos motores durante las 4 ultimas llamadas a la funcion y lo divide entre
cuatro. Osea,
```

```
   hace la media de las 4 ultimas tomas de datos. AL dividirlo por el tiempo del
intervalo
   se obtiene la velocidad media de los motores de los 4 ultimos ciclos.
*/
inline void Lectura_Motores(float &Velocidad_Motores, float &Avance_Motores)
{
long Cuenta_Vueltas_Motor_Izquierdo, Cuenta_Vueltas_Motor_Derecho,
Cuenta_Vueltas_Motores_D;

Cuenta_Vueltas_Motor_Izquierdo = MotorRotationCount(Motor_Izquierdo);
Cuenta_Vueltas_Motor_Derecho = MotorRotationCount(Motor_Derecho);
Cuenta_Vueltas_Motores_Suma_Previa = Cuenta_Vueltas_Motores_Suma;
Cuenta_Vueltas_Motores_Suma = Cuenta_Vueltas_Motor_Izquierdo +
Cuenta_Vueltas_Motor_Derecho;
Cuenta_Vueltas_Motores_Resta = Cuenta_Vueltas_Motor_Izquierdo -
Cuenta_Vueltas_Motor_Derecho;
Cuenta_Vueltas_Motores_D = Cuenta_Vueltas_Motores_Suma -
Cuenta_Vueltas_Motores_Suma_Previa;
Avance_Motores += Cuenta_Vueltas_Motores_D;              //CALCULO DEL INTEGRAL
Velocidad_Motores =
(Cuenta_Vueltas_Motores_D+Cuenta_Vueltas_Motores_DP1+Cuenta_Vueltas_Motores_DP2+Cuent
a_Vueltas_Motores_DP3)/(4*Intervalo); //CALCULO DEL ERROR
Cuenta_Vueltas_Motores_DP3 = Cuenta_Vueltas_Motores_DP2;
Cuenta_Vueltas_Motores_DP2 = Cuenta_Vueltas_Motores_DP1;
Cuenta_Vueltas_Motores_DP1 = Cuenta_Vueltas_Motores_D;
}

//=============================================================================

/* Esta funcion determina el valor que se le pasara a los motores basado en el
equilibrio y el control de direccion
   Asimismo limita esos valores ya que a los motores el valor absoluto maximo que se
les puede pasar es 100
*/

inline void ControlDireccion(int Fuerza, int &Fuerza_Izq, int &Fuerza_Der)
{
int Fuerza_Girar;
```

```
motorDiffTarget += motorControlSteer * Intervalo;
Fuerza_Girar = K_Direccion * (motorDiffTarget-Cuenta_Vueltas_Motores_Resta);
Fuerza_Izq = Fuerza + Fuerza_Girar;
Fuerza_Der = Fuerza - Fuerza_Girar;
if (Fuerza_Izq > 100)    Fuerza_Izq = 100;
if (Fuerza_Izq < -100)   Fuerza_Izq = -100;
if (Fuerza_Der > 100)   Fuerza_Der = 100;
if (Fuerza_Der < -100) Fuerza_Der = -100;
}

//===========================================================================
// Calcula el tiempo de cada iteración del loop

inline void Calculo_Intervalo(long Numero_Loops)
{
if (Numero_Loops == 0)
    {
    Intervalo = 0.0055;
    Comienzo = CurrentTick();
    }
else
    {
    Intervalo = (CurrentTick() - Comienzo)/(Numero_Loops*1000.0);
    }
}

//===========================================================================

task equilibrio()
{
Follows(main);
float Velocidad_Giroscopio, Angulo_Giroscopio;
float Velocidad_Motores;
int Fuerza, Fuerza_Izq, Fuerza_Der;
long PosicionCorrecta;
long Numero_Loops = 0;

PosicionCorrecta = CurrentTick();
```

```
 /* Las variables principales son:
 -Angulo_Giroscopio: Es el angulo de inclinacion del robot. Grados.
 -Velocidad_Giroscopio Valor del giroscopio despues de restarle el offset.
Grados/Segundo
 -Avance_Motores  Es la posicion de los motores. Grados. Es la suma de los dos
motores
 -Velocidad_Motores Velocidad de las ruedas segun los encoders de los motores.
Grados/Segundo

 */
ResetRotationCount(Motor_Izquierdo);                    // Resetea la cuenta de los
encoders de los motores para asegurarnos que empiezan en la posicion cero
ResetRotationCount(Motor_Derecho);
while(1)
        {
        Calculo_Intervalo(Numero_Loops++);
        Lee_giroscopio(Velocidad_Giroscopio, Angulo_Giroscopio);
        Lectura_Motores(Velocidad_Motores, Avance_Motores);
        Avance_Motores -= motorControlDrive * Intervalo;

// Ecuacion principal del equilibrio:

        Fuerza=
Ki_Motores*Avance_Motores+Kp_Motores*Velocidad_Motores+Kp_Giro*Velocidad_Giroscopio+K
i_Giro*Angulo_Giroscopio+KDRIVE*motorControlDrive;          //

        if (abs(Fuerza) < 100)
             {
             PosicionCorrecta = CurrentTick();
             }
        ControlDireccion(Fuerza, Fuerza_Izq, Fuerza_Der);
        OnFwd(Motor_Izquierdo, Fuerza_Izq);
        OnFwd(Motor_Derecho, Fuerza_Der);
        if((CurrentTick()-PosicionCorrecta)>LimiteTiempoCaida)
             {
             break;
             }
        Wait(8);
```

```
string angulo = NumToStr(Angulo_Giroscopio);
//Cogemos los datos del angulo y lo guardamos en una cadena de caracteres
string motor1 = NumToStr(Fuerza_Izq);
string motor2 = NumToStr(Fuerza_Der);
string frase = StrCat(angulo,",",motor1,",",motor2);
WriteLnString(handle,frase,bytes_escritos);
        }
PlaySound(SOUND_DOWN);
Off(Ambos_Motores);
Wait(1000);
}



//===========================================================
// taskControl
// This task runs in parallel to taskBalance.  This task monitors
// the IR Receiver and sets the global variables motorControlDrive
// and motorControlSteer.  Both of these values are in degrees/second.
//
task taskControl()
{
Follows(main);
string msg0;
string msg1;
float valor0;
float valor1;
while(true)
   {
    ClearScreen();
         ReceiveRemoteString(0,false,msg0);
         ReceiveRemoteString(1,false,msg1);
    valor0=atof(msg0)-100;
    valor1=atof(msg1)-100;
    NumOut(10,LCD_LINE7,valor0);                //Ver los valores que le vienen del
bluetooth
    NumOut(10,LCD_LINE8,valor1);                //Restarle 100 para que vaya desde -
100 a 100
    if (valor0 == -100) valor0 = 0;
```

```
    if (valor1 == -100) valor1 = 0;
    motorControlDrive = (valor0 + valor1) * Velocidad_Maxima / 200.0;
    motorControlSteer = (valor0 - valor1) * Velocidad_Maxima / 200.0;
    Wait(25);
   }
//Wait(10000);
}
```

6.2.- Código Visual Basic completo

```
Public Class Form1
    Dim orden As Byte
    Private Sub Button1_Click() Handles Button1.Click
        If ComboBox1.SelectedIndex = -1 Then Exit Sub
        'MsgBox("Conectando a : " & ComboBox1.Items(ComboBox1.SelectedIndex))
        Try
            With SerialPort1
                .PortName = ComboBox1.Items(ComboBox1.SelectedIndex)
                .BaudRate = 9600
                .Parity = IO.Ports.Parity.None
                .DataBits = 8
                .StopBits = IO.Ports.StopBits.One
                .ReadTimeout = 300 '300ms
                .WriteTimeout = 300 '300ms
            End With
            SerialPort1.Open()
            Label1.Text = "Conectado"
            Button1.Enabled = False
            Timer1.Enabled = True
        Catch ex As Exception
            Label1.Text = "Error. No Conectado "
            SerialPort1.Close()
            Button1.Enabled = True
            Timer1.Enabled = False
        End Try
    End Sub
```

```
    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load
        ComboBox1.Items.Clear()
        For Each sp As String In My.Computer.Ports.SerialPortNames
            ComboBox1.Items.Add(sp)
        Next
    End Sub

    Private Sub Button7_Click(sender As System.Object, e As System.EventArgs) Handles
Button7.Click
        SerialPort1.Close()
        Label1.Text = "Desconectado"
        Button1.Enabled = True
        Timer1.Enabled = False
    End Sub

    Private Sub Button8_Click(sender As System.Object, e As System.EventArgs) Handles
Button8.Click
        Dim byteOut(64) As Byte
        TrackBar1.Value = 100
        TrackBar2.Value = 100
    End Sub

    Private Sub CheckBox1_CheckedChanged(sender As System.Object, e As
System.EventArgs) Handles CheckBox1.CheckedChanged
        If CheckBox1.Checked = True Then
            TrackBar2.Enabled = True
            TrackBar2.Value = TrackBar1.Value
            TrackBar2.Enabled = False
        Else
            TrackBar2.Enabled = True
        End If
    End Sub

    Private Sub Timer1_Tick(sender As System.Object, e As System.EventArgs) Handles
Timer1.Tick
        'track bar 1
```

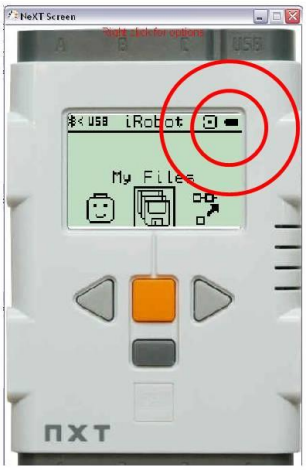
```
        Dim byteOut(64) As Byte, Cadena As String
        Cadena = Str(TrackBar1.Value)
        lbl_tb1.Text = TrackBar1.Value - 100
        lbl_tb2.Text = TrackBar2.Value - 100
        Dim i As Integer
        Try
            byteOut(0) = Len(Cadena) + 5 'number bytes in output message
            byteOut(1) = &H0 'should be 0 for NXT
            byteOut(2) = &H80 '&H0 = reply expected &H80 = no reply expected
            byteOut(3) = &H9 'Send Bluetooth
            byteOut(4) = &H1 'Box Number - 1
            byteOut(5) = Len(Cadena) + 1 'message size with null terminator
            For i = 1 To Len(Cadena) 'copy bytes into output array
                byteOut(i + 5) = Asc(Mid(Cadena, i, 1))
            Next
            byteOut(Len(Cadena) + 6) = &H0 'add null terminator
            SerialPort1.Write(byteOut, 0, Len(Cadena) + 7) 'send message
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try
        If CheckBox1.Checked = True Then
            TrackBar2.Enabled = True
            TrackBar2.Value = TrackBar1.Value
            TrackBar2.Enabled = False
        End If

        'track bar 2
        Dim byteOut1(64) As Byte
        Cadena = Str(TrackBar2.Value)
        Try
            byteOut1(0) = Len(Cadena) + 5 'number bytes in output message
            byteOut1(1) = &H0 'should be 0 for NXT
            byteOut1(2) = &H80 '&H0 = reply expected &H80 = no reply expected
            byteOut1(3) = &H9 'Send Bluetooth
            byteOut1(4) = &H0 'Box Number - 1
            byteOut1(5) = Len(Cadena) + 1 'message size with null terminator
            For i = 1 To Len(Cadena) 'copy bytes into output array
                byteOut1(i + 5) = Asc(Mid(Cadena, i, 1))
```

```
            Next
            byteOut1(Len(Cadena) + 6) = &H0 'add null terminator
            SerialPort1.Write(byteOut1, 0, Len(Cadena) + 7) 'send message
        Catch ex As Exception
            MsgBox(ex.ToString)
        End Try

    End Sub


End Class
```

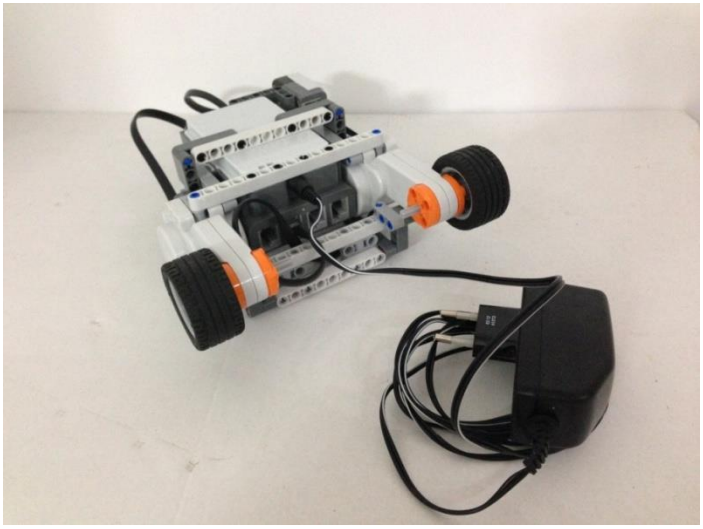
6.3.- Manual de usuario del dispositivo

1.- Encender el Brick o microcontrolador:

2.- Comprobar el estado de las baterías del robot

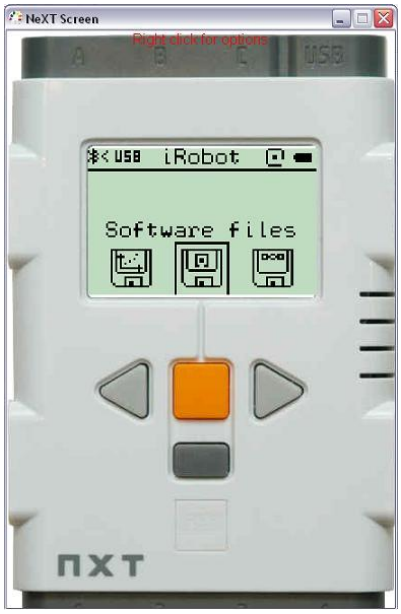


3.- Si la batería se encuentra baja habrá que conectar el cargador de Lego en el puerto de carga del Brick



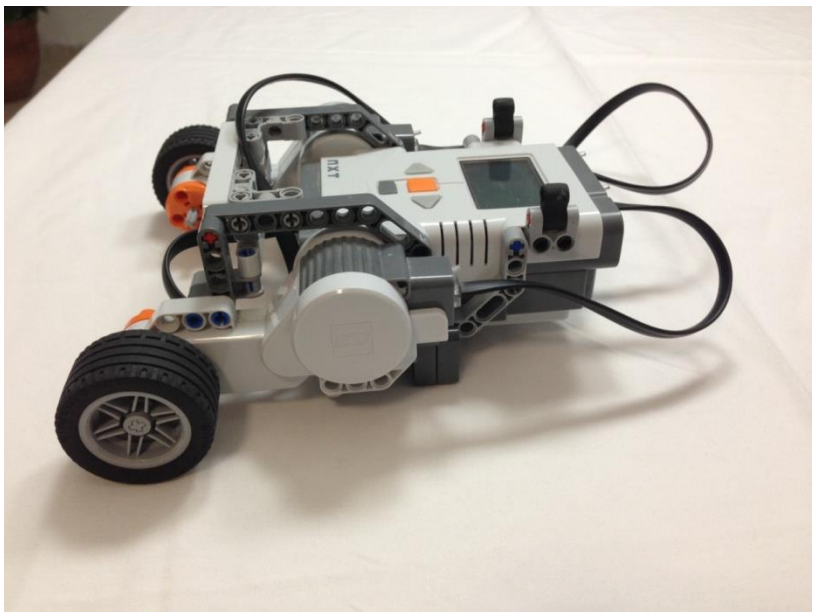
4.- Con la batería cargada y el cargador desenchufado habrá que navegar por el menú del Brick para ejecutar nuestro programa. Primero localizar el icono “My Files” con los controles izquierda y derecha del Brick. A continuación pulsar el botón naranja para acceder a los ficheros de programación existentes en la memoria del Brick

5.- Acceder a la carpeta “Software files”



6.- Seleccionar el programa que deseamos ejecutar, en nuestro caso “Robot 3.2”

7.- El robot está listo para iniciar el programa. Lo primero que necesita es estabilizar el giróscopo para poder tomar una buena lectura. Lo mas aconsejable es dejar el robot en una superficie firme y estable como el suelo



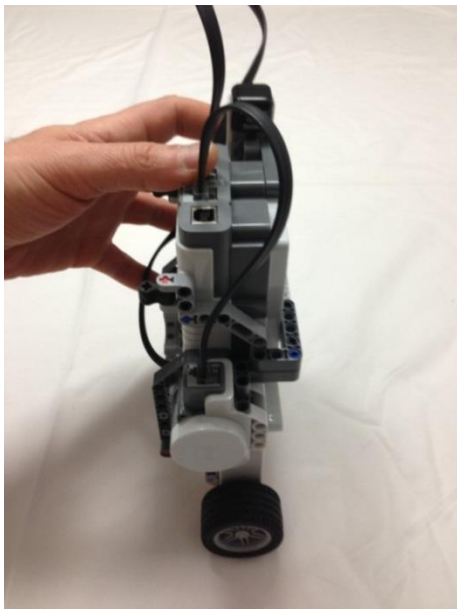
8.- Todo está preparado para comenzar el programa. Pulsar el botón central naranja para ejecutar el comando “Run” para procesar el código:

9.- El robot emitirá un pitido y un mensaje en pantalla indicando que deje el dispositivo en reposo, lo mas quieto posible. En la pantalla aparecerá un mensaje que nos indica que el giróscopo se está calibrando

10.- Cuando el robot se encuentre estable durante poco mas de un segundo, el giróscopo estará listo para hacer mediciones. Hasta que no haga una buena medición del cero no

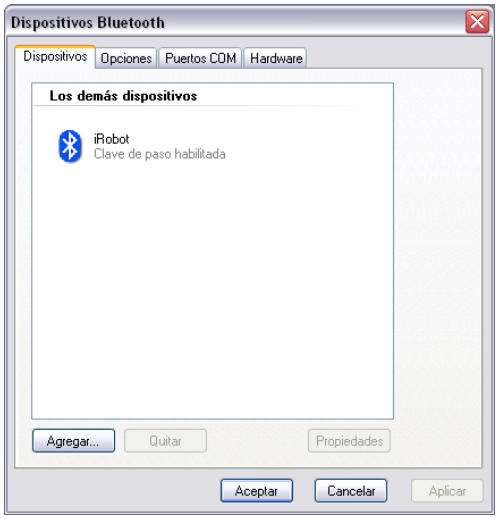
saldrá de este estado. Por eso lo aconsejable es dejar el robot en el suelo aún. Al tomar el cero correcto el robot emitirá un pitido que dará paso a una cuenta atrás de 5 segundos

11.- Rapidamente lo siguiente es poner el robot en pie e intentar mantenerlo lo mas cerca posible de su punto de equilibrio. Esta fase durará 5 segundos. El robot emitirá un pitido por segundo. El último pitido tendrá una frecuencia diferente a las demás, mas aguda. Esta es la señal que indica que el robot está listo para autoequilibrarse por lo tanto ya podremos soltarlo.

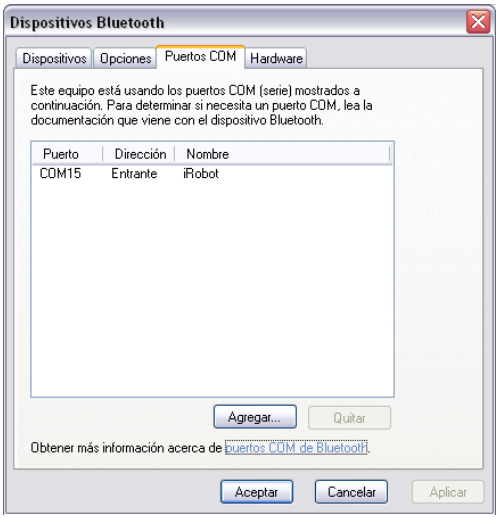


6.4.- Manual de usuario para el HMI (Human Machine Interface)

1.- Vincular dispositivos BLUETOOTH®. El dispositivo debe estar vinculado con el PC. Eso se puede hacer con la aplicación creada para tal efecto propia de cada sistema operativo.

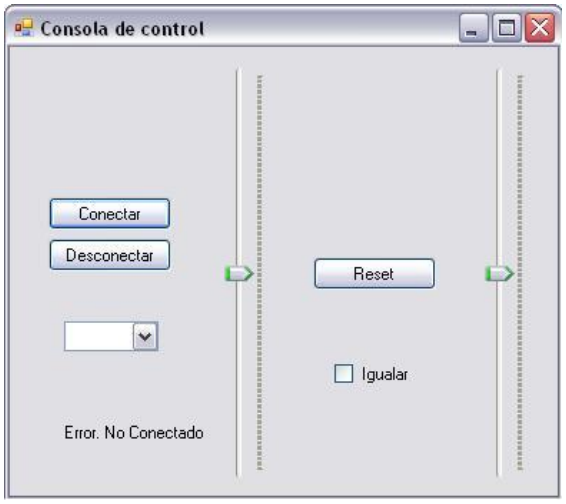


La aplicación conectará el dispositivo BLUETOOTH® con un puerto COM del PC.



2.- Abrir aplicación HMI "Robot"

La consola de control tiene éste aspecto:



La botonera queda escueta, accesible e intuitiva. No obstante, se detalla a continuación:

- Desplegable con opción de elegir el puerto de conexión con el robot. Solo tenemos que seleccionar el puerto COM donde hayamos conectado previamente el robot por BLUETOOTH®.
- Botón Conectar. Conecta el dispositivo con el PC.
- Botón Desconectar. Desconecta el dispositivo del PC.
- 2 Barras deslizantes. La barra izquierda controla el motor izquierdo y la derecha controla el motor derecho. Las barras dan valores entre -100 y 100, valor que nos servirán para avanzar o retroceder. Evidentemente, si solo aumentamos una de las barras, el robot girará. La velocidad y el sentido de giro lo elegimos con el control deslizante de las barras.
- Botón Reset. Deja las dos barras deslizantes justo en medio, o sea, en el cero. Así conseguiremos dejar el robot quieto.
- Un Botón on/off Igualar. Al activar esta casilla emparejamos las dos barras deslizantes lo que equivale a tomar el control de los dos motores con una sola barra. Esto se utiliza para avanzar o retroceder en línea recta, sin giro alguno.

Al comenzar a usar la consola lo primero que debemos hacer es elegir el puerto COM donde previamente ha sido conectado el dispositivo vía BLUETOOTH®.

Una vez seleccionado, habrá que pulsar el botón “conectar” para vincular el dispositivo con el PC. Si la conexión ha sido exitosa el mensaje de la consola será “conectado”, si por el contrario ha habido algún problema aparecerá un error en pantalla.

Ya con todo lo anterior hecho podemos comenzar a dirigir el robot.

Agradecimientos:

A mis padres Francisco y Encarna por darme tanta enseñanza natural

A mi mujer Paqui por regalarme tanto tiempo para realizar este proyecto

A mi amigo Jose Mª por tantos años de ayuda codo a codo

A mi tutor Julio por implicarse tanto en este proyecto