**Practico Nº2 Comunicación Serial y Socket**

Ejercicio 1: “Trabajando con SerialPort en Scorbot”

Descripción: El objetivo de este práctico es enviar datos a través del puerto COM a un brazo robótico que espera datos para realizar las instrucciones de abrir y cerrar el gripper, o ejecutar algún programa previamente grabado.

Abrir el programa Visual Studio y seleccionar **Crear un proyecto**, luego se debe seleccionar la opción **Aplicación de Windows Forms (.Net Framework),** implementado en C#. Asigne el nombre que usted desee y luego diseñe el formulario agregando los siguientes componentes:

|  |  |
| --- | --- |
| **Nombre** | **Propiedades** |
| Form1 | Text: Comunicación Serial  Size 485; 354 |
| Button1 | Text: Conectar |
| Button2 | Text: Desconectar |
| Button3 | Text: Enviar |
| Button4 | Text: Guardar |
| Button5 | Text: Run pcplc |
| Button6 | Text: Abortar |
| Button7 | Text: Run ttsib |
| Button8 | Text: coff |
| Button9 | Text: move 0 |
| Button10 | Text: open |
| Button11 | Text: close |
| ComboBox1 | Ítems: COM1, COM2…,  DropDownStyle: DropDownList |
| textBox1 | Text: Desconectado  TextAlign: Center |
| RichTextBox1 | Text: |
| RichTextBox2 | Text: |
| SerialPort1 |  |
| Timer1 |  |
| SaveFileDialog1 |  |
| Label1 | Text: Datos Enviados |
| Label2 | Text: Datos Recibidos |

El diseño del formulario debería quedar de la siguiente manera:

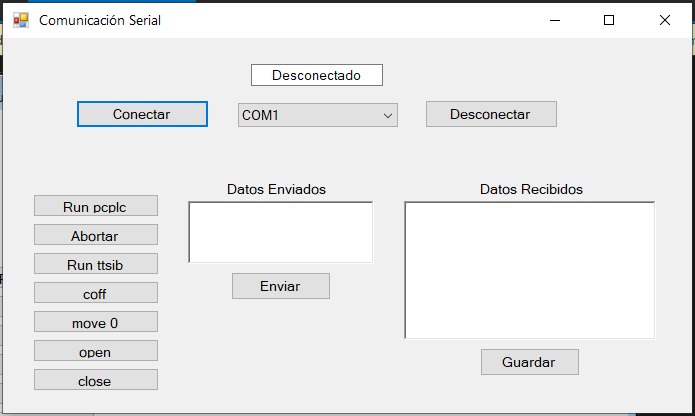


Figura 1. Diseño del Form

**Importante**: para llevar a cabo la actividad agregue bibliotecas adicionales de entrada y salida para comunicación, para ello haga doble clic en el Form1 y en el encabezado añada lo siguiente:



1. Llene el *ComboBox1* que contendrá los puertos COM a disposición por medio de la propiedad Ítems, además para bloquear la edición del *ComboBox1*, debe ir a la propiedad *DropDownStyle* y cambiarla a *DropDownList*.

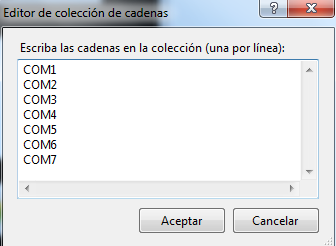
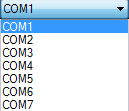
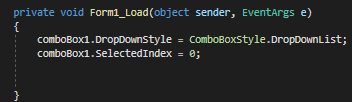
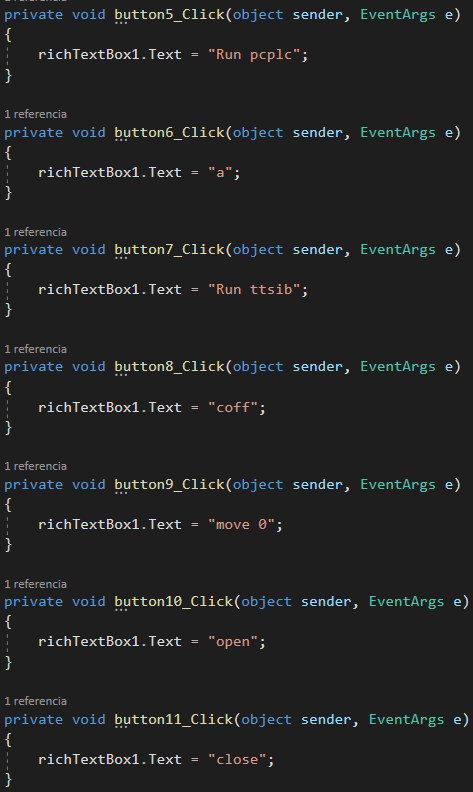
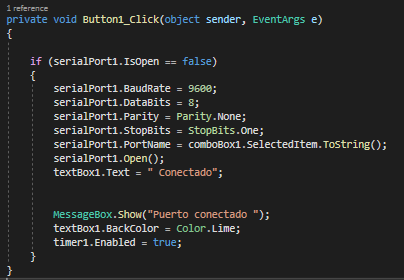


Figura 2. ComboBox

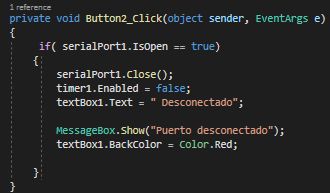
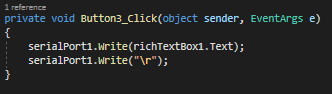
Lo anterior también se puede modificar agregando las líneas de código que se muestran a continuación:

1. El código de los botones 5 al 11 puede modificarse para escribir automáticamente en el RichTextBox1 comandos usados con frecuencia, de la siguiente manera:

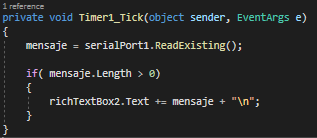


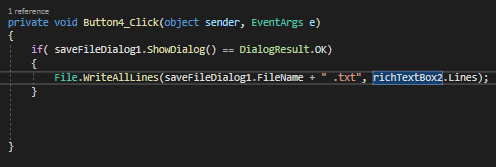
1. Agregue el código del botón Conectar, haciendo doble clic sobre ese botón:

En la condición *if* se configuran los parámetros del puerto COM. Una vez que está configurado se abre el puerto, se muestra un mensaje de apertura y habilita un *timer*.

1. Luego agregue el código del botón desconectar:
2. Agregue el siguiente código al botón Enviar:

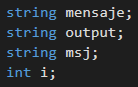
Las operaciones que se van realizando son almacenadas en el richtextBox1 para tener un seguimiento de lo realizado.

1. Para finalizar se emplea la función *Timer1\_Tick* y un botón para guardar el contenido de la comunicación:

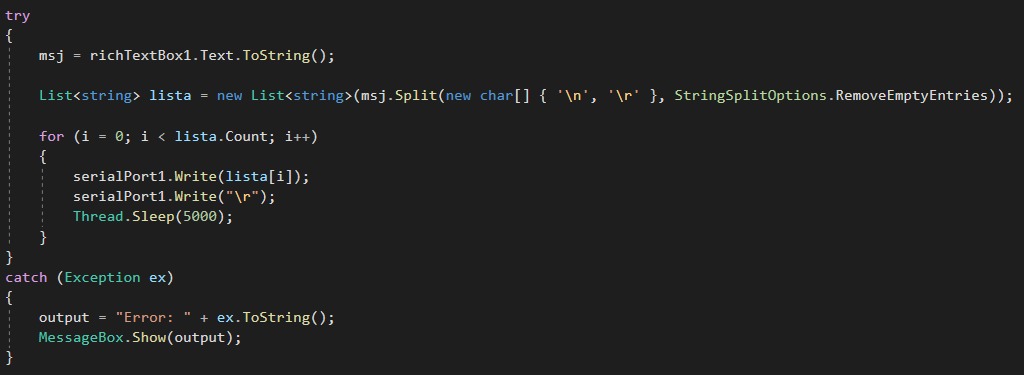


Mediante el *timer1* se almacena en el *richtextBox2* el contenido recibido de la comunicación serial, así se tiene un registro de lo que se recibe en la comunicación.

1. Declare las siguientes variables globales y compile.



1. Configure como enviar la secuencia de instrucciones en el botón Enviar:



La variable de tipo string “*msj*” recibe todo lo captado desde el *Richtexbox1*. Se crea una lista *List<String> lista* cuyo método *Split* divide el *string* en *substrings* cuando cumple la condición *{‘\n’, ‘\r’},* es decir, almacena y “limpia” los string recibidos, separándolos en función de los saltos o retorno de línea.

Se crea un ciclo for para:

• Recorrer las listas mediante la variable “i”.

• En cada ciclo se envía la lista al serialport.

• Con *Thread.Sleep(5000),* cada 5 segundos pasa al siguiente ciclo (instrucción).

Se debe incluir la siguiente biblioteca:



En la función *Timer1\_Tick* añadimos una sentencia *if(mensaje.Contains("Done.").* Existe un problema con la recepción del símbolo “?”. El mensaje recibido es guardado en la variable *mensaje*, pero la frase “?Done” no es reconocida por el programa. Por ello se emplea la función *Contains* que ayuda a identificar el mensaje recibido para mostrarlo correctamente.

El comando enviado al puerto serial se recibe de vuelta, como no se desea recibir los mensajes enviados deben ser removidos. El comando enviado está almacenado en la variable “msj” y la función *Remove* la elimina del listado de mensajes recibidos. Posteriormente estos mensajes son enviados al *richTextBox* para ser mostrados por la pantalla del Form.

Finalmente, los mensajes recibidos son mostrados tal y como serían mostrados en una consola (línea por línea) en el *richTextBox2*:

