ESCUELA SUPERIOR DE INFORMÁTICA

UNIVERSIDAD DE CASILLA-LA MANCHA

Automatización Industrial

Check Interface

Cuarta práctica

Jose Domingo López López
josed.lopez1@alu.uclm.es

Raúl Arias García
raul.arias2@alu.uclm.es

Grupo: ELF 08 (Tarde)
16 de Abril del 2009

Índice de contenidos

Índice de figuras

1. Introducción

El objetivo de esta práctica consiste en realizar un programa que tenga una funcionalidad similar a la herramienta *"Check Interface"* incluida en LLWin (ver Ilustración 1). El programa debe permitir visualizar en todo instante el estado de las entradas –tanto digitales como analógicas- de la interfaz y ajustar el estado deseado para cualquiera de las salidas.

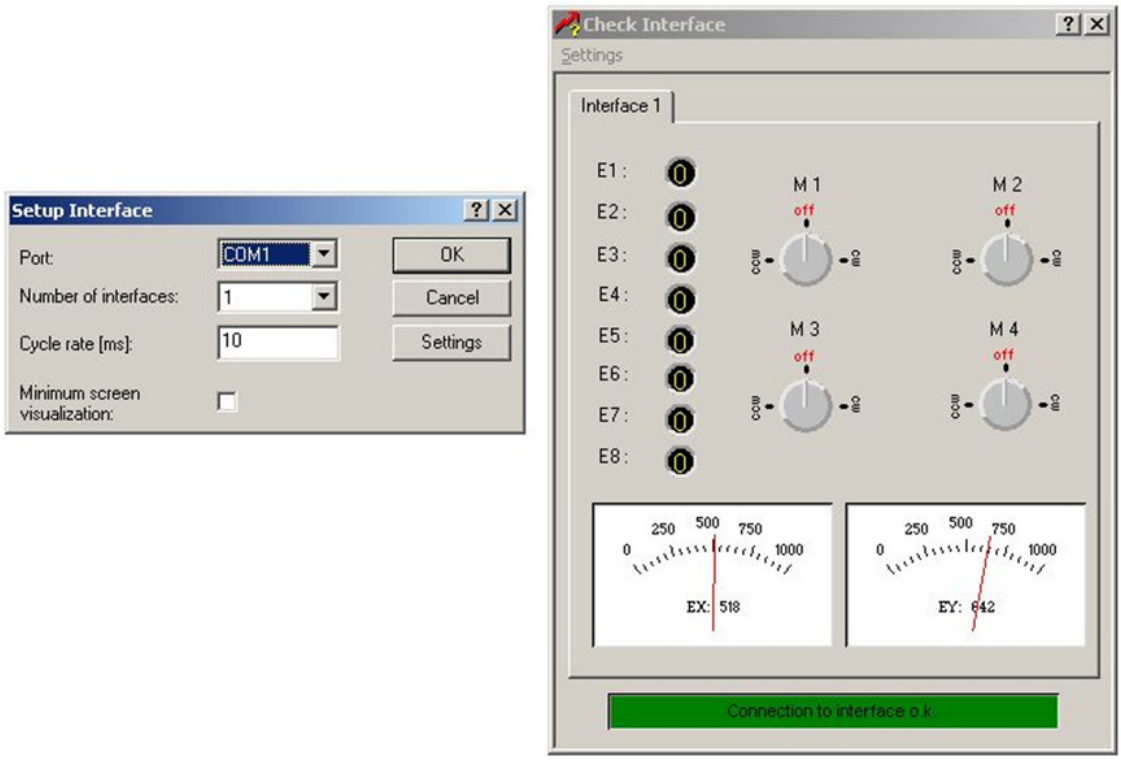

Ilustración 1: Check Interface incluido en LLWin

2. Decisiones de diseño

La primera cuestión que nos planteamos fue meramente estética: ¿Qué iconos utilizar para que la aplicación sea atractiva e intuitiva? Y dado que el diseño de iconos no es algo trivial y requiere mucha experiencia y dedicación, nos decantamos por el uso de los iconos del set Gartoon (ver Ilustración 2). A continuación se muestran los iconos empleados:

Ilustración 2: Iconos Gartoon

Después, estuvimos pensando cómo distribuir los controles y visores necesarios de la aplicación para que sea lo más amigable posible para nuevos usuarios y que los antiguos usuarios del Check Interface de LLWin no tengan que volver a aprender a utilizar un nuevo programa. Por todo esto, hicimos una interfaz gráfica lo más parecida posible al Check Interface de LLWin (ver Ilustración 3)

Una vez que las cuestiones de diseño en términos de interfaz de usuario estaban determinadas, debatimos como resolver el problema de que el estado de las entradas sea refrescado cada X milisegundos y que la aplicación auto-detecte cuándo se conecta y desconecta el interfaz. Estos dos problemas se resolvieron mediante el uso de dos *timers*[1].

[1] Un **timer** es un *widget* con dos propiedades y un evento:

- Propiedades:
  - Enabled: permite la generación de eventos.
  - Interval: frecuencia con la que se generan los eventos (en milisegundos).
- Evento:
  - Tick: ocurre cuando transcurre el tiempo especificado en la propiedad *Interval*.

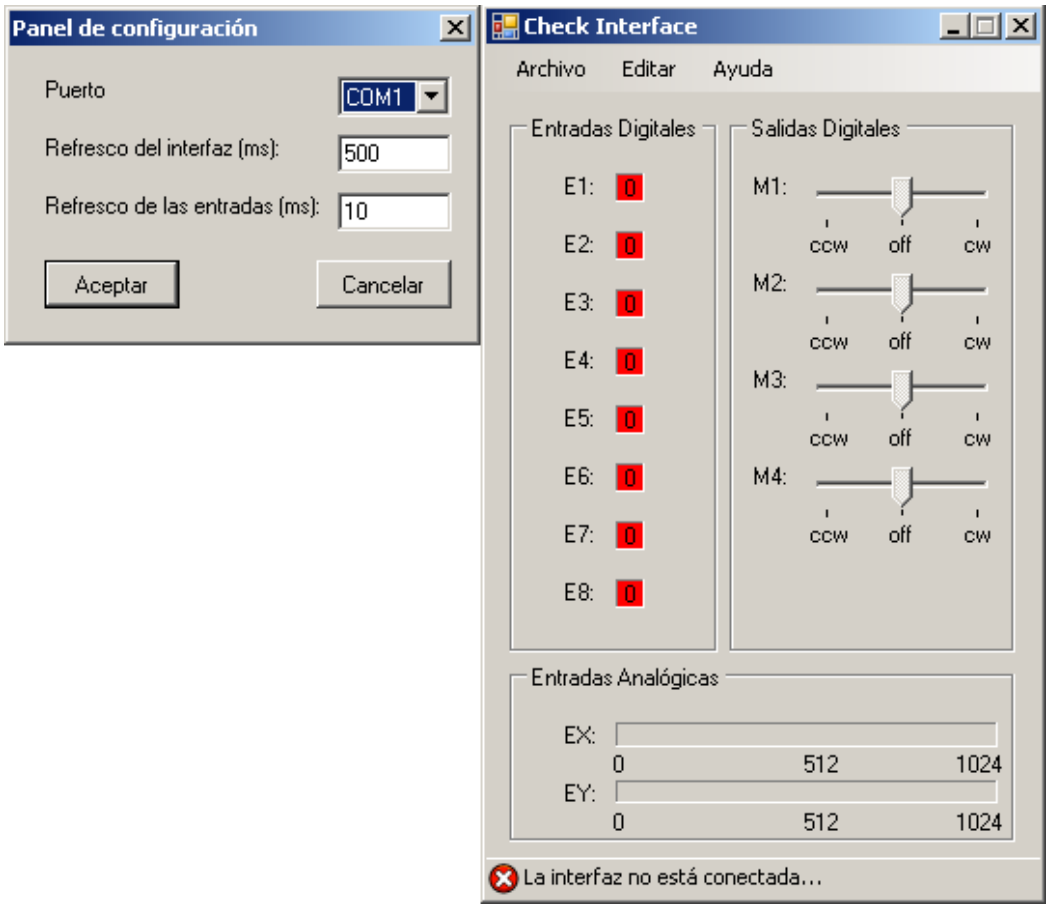

Ilustración 3: Check Interface resultante tras la realización de la práctica

3. Manual de usuario

La ventana principal de la aplicación se divide en 5 zonas bien diferenciadas (ver Ilustración 4):

- Una barra de menús situada en la parte superior.
- Una agrupación, situada en la parte izquierda, en la cual se puede observar el estado de las entradas digitales.
- Una agrupación, situada en la parte derecha, en la cual se puede modificar el estado de las salidas digitales.
- Una agrupación, situada en la parte inferior, en la cual se puede observar el estado y el valor de las entradas analógicas.
- Una barra de estado, situada debajo de esta última agrupación, en la cual se puede observar el estado de conexión del interfaz.

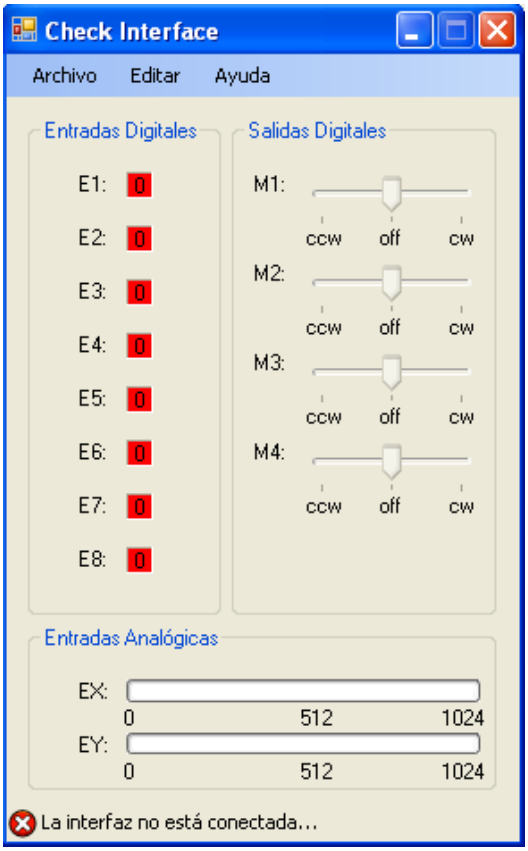

Ilustración 4: Check Interface con la interfaz de fishertechnik desconectada

En la barra de menús podemos encontrar los siguientes ítems:

- Archivo, que contiene el ítem *Salir* para salir del programa.
- Editar, que contiene el ítem *Configuración* para configurar ciertos parámetros del programa.
- Ayuda, que contiene el ítem *Acerca De* para mostrar información relativa al software.

Una vez iniciada la aplicación, cuando se conecta el interfaz de fishertechnik al ordenador, podemos observar como la aplicación lo detecta y muestra en su barra de estado un mensaje informativo con su correspondiente icono indicando que todo está funcionando correctamente (ver Ilustración 5). Si el interfaz se encuentra desconectado, todos los controles se mostrarán deshabilitados.

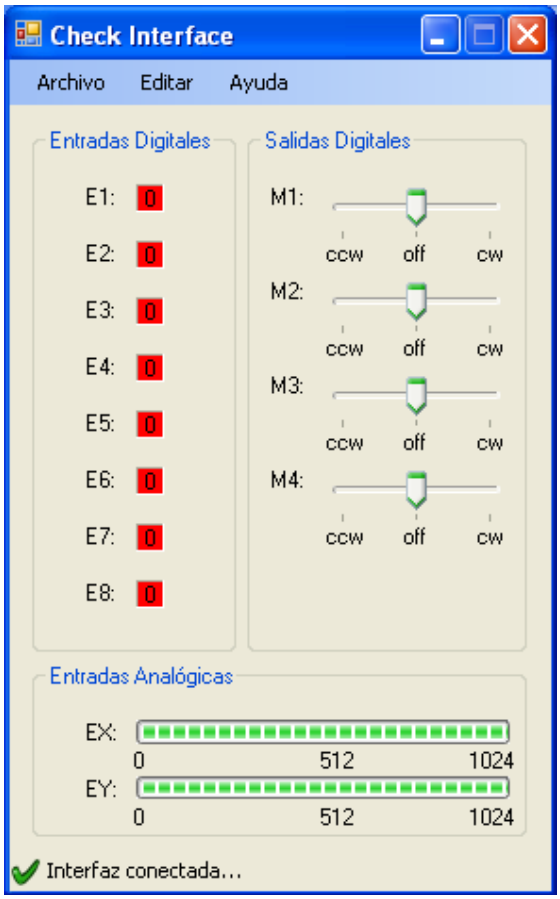

Ilustración 5: Check Interface con la interfaz de fishertechnik conectada

A continuación, es posible configurar ciertos parámetros como el puerto COM al que conectaremos el interfaz, la tasa de refresco del interfaz (para auto-detectar la conexión y desconexión de ésta) y la tasa de refresco de las entradas (ver Ilustración 6 e Ilustración 7).

Ilustración 6: Panel de configuración del Check Interface

Ilustración 7: Panel de configuración del Check Interface (Lista de puertos COM)

Una vez que el interfaz está conectado y la aplicación configurada, podemos comenzar a variar el estado de las entradas y las salidas (ver Ilustración 8).

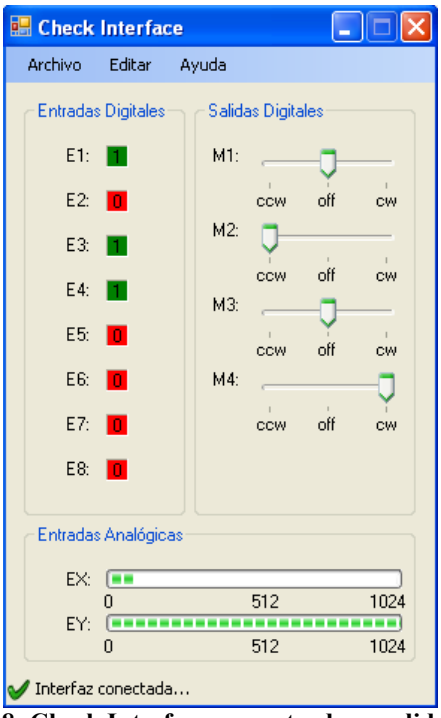

Ilustración 8: Check Interface con entradas y salidas alteradas

En la agrupación de **entradas digitales** parte se puede comprobar si las entradas están activadas (se muestra un 1 con el fondo en color verde) o desactivadas (se muestra un 0 con el fondo en color rojo).

En la agrupación de **salidas digitales** podemos modificar el sentido de giro de un motor o encender una lámpara, en función del actuador que se encuentre conectado a la salida correspondiente, mediante una barra de desplazamiento. Si el actuador conectado es un motor, la posición *cw* (clock-wise) hará que éste gire en sentido de las agujas del reloj; si por el contrario la barra de desplazamiento se encuentra en la posición *ccw* (counter clock-wise) hará que éste gire en sentido contrario a las agujas del reloj.

En la agrupación de **salidas analógicas** se observa el valor que toma cada una de estas mediante una barra de progreso. Para probar estas entradas, se podría conectar una célula fotoeléctrica y observar su valor al exponerla a la luz.

4. Conclusiones

Como se ha comentado en secciones anteriores, la interfaz de usuario de la aplicación se ha hecho lo más parecida posible a la del Check Interface incluido en LLWin para que los usuarios expertos no tengan que aprender a utilizar una aplicación nueva. Además, esta interfaz es muy sencilla e intuitiva para los nuevos usuarios.

En términos de nuestra experiencia personal, hemos entendido y demostrado cómo funciona la aplicación y, tras la programación de ésta, nos vemos capaces de desarrollar el resto de prácticas sin problema alguno ya que con esta práctica hemos cubierto conceptos muy básicos e interesantes, tanto a nivel de uso de la librería FishFa30, como a nivel de programación en el entorno Visual Studio .NET C#.

5. Referencias

1: Gartoon Icons (GNU General Public License):
http://www.zeusboxstudio.com

6. Apéndice I: Código fuente

Checkinterface.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using FishFa30;

namespace CheckInterface
{
    public partial class CheckInterface : Form
    {
        private Port Puerto;
        private int rateIface;
        private int rateInputs;
        private Boolean encendido;
        FishFace fish = new FishFace(true, false, 0);

        public CheckInterface()
        {
            // Inicializamos los atributos con los valores por defecto
            this.encendido = false;
            this.Puerto = Port.COM1;
            this.rateIface = 500;
            this.rateInputs = 10;
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            /* Configuramos ciertos parámetros de la interfaz de forma
             * explícita para evitar errores */
            int min = 0;
            int max = 1024;
            // Barra de progreso de la entrada analógica EX
            EXProgressBar.Minimum = min;
            EXProgressBar.Maximum = max;
            EXProgressBar.Value = min;
            // Barra de progreso de la entrada analógica EY
            EYProgressBar.Minimum = min;
            EYProgressBar.Maximum = max;
            EYProgressBar.Value = min;
            // Auxiliares
            EDigAuxiliar.Visible = false;
            // Inicializamos los timers con los valores por defecto
```

```
            timerCheckInterface.Interval = this.rateIface;
            timerCheckInterface.Enabled = true;
            timerEntradas.Interval = this.rateInputs;
        }

        private void cambiarEstado(int estado)
        {
            EDigAuxiliar.BackColor = Color.Red;
            EDigAuxiliar.Text = "0";
            // Valores de estado: 0(ok) - 1(error)
            if (estado == 0)
            { // La interfaz está conectada
                // Indicamos en la variable de control que la interfaz
está encendida
                this.encendido = true;
                // Activamos el timer que dispara la lectura de las
entradas
                timerEntradas.Enabled = true;
                // Configuramos el interfaz a su estado de START
                this.tsIcono.Image =
global::CheckInterface.Properties.Resources.gtk_yes;
                this.tsIcono.Size = new System.Drawing.Size(16, 16);
                tsBarraEstado.Text = "Interfaz conectada...";
                M1TrackBar.Enabled = true;
                M2TrackBar.Enabled = true;
                M3TrackBar.Enabled = true;
                M4TrackBar.Enabled = true;
            }
            if (estado == 1)
            { // La interfaz NO está conectada
                // Indicamos en la variable de control que la interfaz no
esta encendida
                this.encendido = false;
                // Paramos el timer que dispara la lectura de las
entradas
                timerEntradas.Enabled = false;
                // Cerramos la conexión con la interfaz
                this.fish.CloseInterface();
                // Configuramos el interfaz a su estado de STOP
                this.tsIcono.Image =
global::CheckInterface.Properties.Resources.gtk_no;
                this.tsIcono.Size = new System.Drawing.Size(16, 16);
                tsBarraEstado.Text = "La interfaz no está conectada...";
                M1TrackBar.Value = 1;
                M1TrackBar.Enabled = false;
                M2TrackBar.Value = 1;
                M2TrackBar.Enabled = false;
                M3TrackBar.Value = 1;
                M3TrackBar.Enabled = false;
                M4TrackBar.Value = 1;
                M4TrackBar.Enabled = false;
                EXProgressBar.Value = EXProgressBar.Minimum;
                EYProgressBar.Value = EYProgressBar.Minimum;
            }
            E1Label.BackColor = EDigAuxiliar.BackColor;
            E1Label.Text = EDigAuxiliar.Text;
            E2Label.BackColor = EDigAuxiliar.BackColor;
```

```
            E2Label.Text = EDigAuxiliar.Text;
            E3Label.BackColor = EDigAuxiliar.BackColor;
            E3Label.Text = EDigAuxiliar.Text;
            E4Label.BackColor = EDigAuxiliar.BackColor;
            E4Label.Text = EDigAuxiliar.Text;
            E5Label.BackColor = EDigAuxiliar.BackColor;
            E5Label.Text = EDigAuxiliar.Text;
            E6Label.BackColor = EDigAuxiliar.BackColor;
            E6Label.Text = EDigAuxiliar.Text;
            E7Label.BackColor = EDigAuxiliar.BackColor;
            E7Label.Text = EDigAuxiliar.Text;
            E8Label.BackColor = EDigAuxiliar.BackColor;
            E8Label.Text = EDigAuxiliar.Text;
        }

        private void activarMotor(Nr motor, int pos)
        {
            // Valores de pos: 0(ccw) - 1(off) - 2(cw)
            if (pos == 0) fish.SetMotor(motor, Dir.Left);
            else if (pos == 2) fish.SetMotor(motor, Dir.Right);
            else fish.SetMotor(motor, Dir.Off);
        }

        private void M1TrackBar_Scroll(object sender, EventArgs e)
        {
            activarMotor(Nr.M1, M1TrackBar.Value);
        }

        private void M2TrackBar_Scroll(object sender, EventArgs e)
        {
            activarMotor(Nr.M2, M2TrackBar.Value);
        }

        private void M3TrackBar_Scroll(object sender, EventArgs e)
        {
            activarMotor(Nr.M3, M3TrackBar.Value);
        }

        private void M4TrackBar_Scroll(object sender, EventArgs e)
        {
            activarMotor(Nr.M4, M4TrackBar.Value);
        }

        private void configuraciónToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Configuracion conf = new Configuracion(this.Puerto,
timerCheckInterface.Interval, timerEntradas.Interval);
            conf.ShowDialog();
            if (conf.estaConfigurado())
            {
                if (this.Puerto != conf.getPuerto())
                {
                    cambiarEstado(1);
                    this.Puerto = conf.getPuerto();
                    try
                    {
```

```
                        if (!this.encendido)
                        {
                            fish.OpenInterface(this.Puerto);
                            cambiarEstado(0);
                        }
                    }
                    catch (FishFaceException)
                    {
                        cambiarEstado(1);
                    }
                }
                this.rateIface = conf.getRefrescoInterfaz();
                this.rateInputs = conf.getRefrescoEntradas();
                timerCheckInterface.Interval = this.rateIface;
                timerEntradas.Interval = this.rateInputs;
            }
        }

        private void cambiarEstadoEntradaDigital(Nr input, Label label)
        {
            if (fish.GetInput(input))
            {
                label.Text = "1";
                label.BackColor = Color.Green;
            }
            else
            {
                label.Text = "0";
                label.BackColor = Color.Red;
            }
        }

        private void cambiarEstadoEntradaAnalogica(Nr input, ProgressBar
bar)
        {
            int valor = 0;
            valor = fish.GetAnalog(input);
            bar.Value = valor;
        }

        private void timerEntradas_Tick(object sender, EventArgs e)
        {
            try
            {
                if (this.encendido)
                {
                    cambiarEstadoEntradaDigital(Nr.E1, E1Label);
                    cambiarEstadoEntradaDigital(Nr.E2, E2Label);
                    cambiarEstadoEntradaDigital(Nr.E3, E3Label);
                    cambiarEstadoEntradaDigital(Nr.E4, E4Label);
                    cambiarEstadoEntradaDigital(Nr.E5, E5Label);
                    cambiarEstadoEntradaDigital(Nr.E6, E6Label);
                    cambiarEstadoEntradaDigital(Nr.E7, E7Label);
                    cambiarEstadoEntradaDigital(Nr.E8, E8Label);
                    cambiarEstadoEntradaAnalogica(Nr.EX, EXProgressBar);
                    cambiarEstadoEntradaAnalogica(Nr.EY, EYProgressBar);
                }
```

```
            }
            catch (FishFaceException)
            {
                cambiarEstado(1);
            }
        }

        private void timerCheckInterface_Tick(object sender, EventArgs e)
        {
            try
            {
                if (!this.encendido)
                {
                    fish.OpenInterface(this.Puerto);
                    cambiarEstado(0);
                }
            }
            catch (FishFaceException)
            {
                cambiarEstado(1);
            }
        }

        private void salirMenuItem_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void acercaDeToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            AboutBox about = new AboutBox();
            about.ShowDialog();
        }
    }
}
```

Configuracion.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using FishFa30;

namespace CheckInterface
{
    public partial class Configuracion : Form
    {
        private Boolean configurado = false;
        private Port puerto;
        private int refrescoInterfaz;
        private int refrescoEntradas;

        public Configuracion(Port port, int rateIface, int rateInputs)
        {
            InitializeComponent();
            puertoComboBox.Items.Add("COM1");
            puertoComboBox.Items.Add("COM2");
            puertoComboBox.Items.Add("COM3");
            puertoComboBox.Items.Add("COM4");
            switch (port)
            {
                case Port.COM1: puertoComboBox.SelectedIndex = 0; break;
                case Port.COM2: puertoComboBox.SelectedIndex = 1; break;
                case Port.COM3: puertoComboBox.SelectedIndex = 2; break;
                case Port.COM4: puertoComboBox.SelectedIndex = 3; break;
            }
            this.refrescoInterfaz = rateIface;
            refrescoInterfazTextBox.Text =
this.refrescoInterfaz.ToString();
            this.refrescoEntradas = rateInputs;
            refrescoEntradasTextBox.Text =
this.refrescoEntradas.ToString();
        }

        public Boolean estaConfigurado()
        {
            return this.configurado;
        }

        public Port getPuerto()
        {
            return this.puerto;
        }

        public int getRefrescoInterfaz()
        {
            return this.refrescoInterfaz;
```

```
        }

        public int getRefrescoEntradas()
        {
            return this.refrescoEntradas;
        }

        private void botonAceptar_Click(object sender, EventArgs e)
        {
            this.configurado = true;
            switch (puertoComboBox.SelectedIndex)
            {
                case 0: this.puerto = Port.COM1; break;
                case 1: this.puerto = Port.COM2; break;
                case 2: this.puerto = Port.COM3; break;
                case 3: this.puerto = Port.COM4; break;
            }
            if (refrescoInterfazTextBox.Text != "")
                this.refrescoInterfaz =
Convert.ToInt32(refrescoInterfazTextBox.Text);
            if (refrescoEntradasTextBox.Text != "")
                this.refrescoEntradas =
Convert.ToInt32(refrescoEntradasTextBox.Text);
            this.Hide();
        }

        private void botonCancelar_Click(object sender, EventArgs e)
        {
            this.configurado = false;
            this.Hide();
        }
    }
}
```

AboutBox.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Windows.Forms;

namespace CheckInterface
{
    partial class AboutBox : Form
    {
        public AboutBox()
        {
            InitializeComponent();
            this.Text = "Acerca de CheckInterface";
            this.labelProductName.Text = "CheckInterface";
            this.labelVersion.Text = "Versión 1.0";
            this.labelCopyright.Text = "Copyleft © 2009 Jose Domingo
López y Raul Arias";
            this.labelCompanyName.Text = "UCLM";
            this.textBoxDescription.Text = "CheckInterface es un programa
con una funcionalidad similar a la herramienta \"Check Interface\"
incluida en LLWin.\r\nCheckInterface permite visualizar en todo instante
el estado de las entradas de la interfaz y ajustar el estado deseado para
cualquiera de las salidas.";
        }
    }
}
```