Código fuente C#

Código asociado al formulario Splash

```
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;

namespace proyectoAlarma
{
   public partial class FormSplash : Form
   {
      public FormSplash()
      {
         //Se inicializa el timer y se establece su cuenta en cinco segundos.
         InitializeComponent();
         tiempo.Enabled = true;
         tiempo.Interval = 5000;
      }

      private void tiempo_Tick(object sender, EventArgs e)
      {
         //En el momento que pasan los cinco segundos el timer se para.
         //Posteriormente, se cierra esta ventana para dar paso al formulario
principal.
         tiempo.Stop();
         this.DialogResult = DialogResult.OK;
         this.Close();
      }
   }
}
```

Código asociado al formulario principal

```
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 System.IO;
using System.IO.Ports;
using proyectoAlarma.Properties;
using System.Net.Mail;

namespace proyectoAlarma
{
   public partial class frmSmartAlarm : Form
   {
      //En este apartado declaro las variables de caracter general.
      #region Variables
      int contador = 0;
      string fecha, registro, recibirDato, To;
      string From = "proyectoalarmasanjose2014@gmail.com";
      string Subject = "¡Alerta, intruso en casa!";
      string Contraseña = "fundacionloyola2014";
      string Message = "Acaba de entrar un intruso en su casa. Es necesario
que avise a la policía.";
      #endregion

      public frmSmartAlarm()
      {
         InitializeComponent();
         CheckForIllegalCrossThreadCalls = false;
      }


      private void frmSmartAlarm_Load(object sender, EventArgs e)
      {
         //En este apartado cargo en los ComboBox las características
seleccionables del puerto serie.
         this.Size = new Size(328, 281);

         #region BindingSource cbPort
         Dictionary<string, string> dPort = new Dictionary<string, string>();
         string[] PortNames = System.IO.Ports.SerialPort.GetPortNames();
         Array.Sort<string>(PortNames);
         if (PortNames != null && PortNames.Length > 0)
         {
```

```
    foreach (string portName in PortNames)
    {

        dPort.Add(portName.Substring(3), portName);

    }
    cbPort.DisplayMember = "Value";
    cbPort.DataSource = new BindingSource(dPort, null);
    cbPort.SelectedIndex = 0;
}
#endregion

#region BindingSource cbData
Dictionary<int, int> dData = new Dictionary<int, int>();
dData.Add(7, 7);
dData.Add(8, 8);
cbData.DisplayMember = "Value";
cbData.DataSource = new BindingSource(dData, null);
cbData.SelectedIndex = 1;
#endregion
#region BindingSource cbParity
Dictionary<System.IO.Ports.Parity, string> dParity = new
Dictionary<System.IO.Ports.Parity, string>();
dParity.Add(System.IO.Ports.Parity.None, "(Ninguno)");
dParity.Add(System.IO.Ports.Parity.Even, "Par");
dParity.Add(System.IO.Ports.Parity.Odd, "Impar");
cbParity.DisplayMember = "Value";
cbParity.DataSource = new BindingSource(dParity, null);
cbParity.SelectedIndex = 0;
#endregion
#region BindingSource cbStop
Dictionary<System.IO.Ports.StopBits, string> dStop = new
Dictionary<System.IO.Ports.StopBits, string>();
dStop.Add(System.IO.Ports.StopBits.One, "1");
dStop.Add(System.IO.Ports.StopBits.OnePointFive, "1.5");
dStop.Add(System.IO.Ports.StopBits.Two, "2");
cbStop.DisplayMember = "Value";
cbStop.DataSource = new BindingSource(dStop, null);
cbStop.SelectedIndex = 0;
#endregion
#region BindingSource cbBaud
Dictionary<int, int> items = new Dictionary<int, int>();
items.Add(1, 1200);
items.Add(2, 2400);
items.Add(3, 4800);
items.Add(4, 9600);
items.Add(5, 19200);
cbBaud.DisplayMember = "Value";
cbBaud.DataSource = new BindingSource(items, null);
cbBaud.SelectedIndex = 4;
```

```
            #endregion

            //Inicializo el puerto serie con los valores por defecto que he determinado e inhabilito los combobox.
            #region Inicio Puerto Serie
            serialPort1.PortName = cbPort.Text;
            serialPort1.BaudRate = int.Parse(cbBaud.Text);
            cbData.Enabled = false;
            cbParity.Enabled = false;
            cbStop.Enabled = false;
            lbData.Enabled = false;
            lblParity.Enabled = false;
            lblStopBits.Enabled = false;
            lblPort.Enabled = false;
            lbBauds.Enabled = false;
            cbPort.Enabled = false;
            cbBaud.Enabled = false;
            btnAceptar.Enabled = false;
            chkConfAvanz.Enabled = false;
            #endregion

            //Posteriormente abro el puerto serie.
            #region Abrir Puerto Serie
            if (!serialPort1.IsOpen)
            {
                try
                {
                    serialPort1.Open();
                }
                catch
                {
                    MessageBox.Show("Puerto no válido, configure el puerto serial");
                    return;
                }
            }
            #endregion

        }
        private void frmSmartAlarm_FormClosing(object sender, FormClosingEventArgs e)
        {
            //Al cerrar la aplicación cierro el puerto serie.
            if (serialPort1.IsOpen)
            {
                serialPort1.Close();
            }
        }

         private void salirToolStripMenuItem_Click(object sender, EventArgs e)
         {
```

```
            //Si se pulsa el botón salir del menú la aplicación se cierra.
            this.Close();
        }

        private void abrirToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            //Se abre un nuevo formulario para poder cargar un archivo ya existente del sistema.
            frmRegistroBaseDatos miRegistroBaseDatos = new frmRegistroBaseDatos();
            miRegistroBaseDatos.ShowDialog();
        }


        private void acercaDeToolStripMenuItem_Click_1(object sender, EventArgs e)
        {
            //Se abre un nuevo formulario con una ventana acerca de.
            frmAbout miAbout = new frmAbout();
            miAbout.ShowDialog();
        }

        private void btnAceptar_Click(object sender, EventArgs e)
        {
            //Establezco el puerto serie con los valores que he modificado, abro el puerto e inhabilito los combobox.
            #region Establecimineto de parámetros en el Puerto Serie
            serialPort1.PortName = cbPort.Text;
            serialPort1.BaudRate = int.Parse(cbBaud.Text);
            serialPort1.DataBits = int.Parse(cbData.Text);
            if (cbParity.Text == "(Ninguno)")
            {
                serialPort1.Parity = Parity.None;
            }
            else if (cbParity.Text == "Par")
            {
                serialPort1.Parity = Parity.Even;
            }
            else
            {
                serialPort1.Parity = Parity.Odd;
            }
            if (cbStop.Text == "1")
            {
                serialPort1.StopBits = StopBits.One;
            }
            else if (cbStop.Text == "1.5")
            {
                serialPort1.StopBits = StopBits.OnePointFive;
            }
```

```
            else
            {
                serialPort1.StopBits = StopBits.Two;
            }
            if (!serialPort1.IsOpen)
            {
                try
                {
                    serialPort1.Open();
                }
                catch
                {
                    MessageBox.Show("Puerto no válido");
                    return;
                }
            }
            #endregion

            #region Inhabilitar botones del Puerto Serie
            lblPort.Enabled = false;
            lbBauds.Enabled = false;
            lbData.Enabled = false;
            lblParity.Enabled = false;
            lblStopBits.Enabled = false;
            cbPort.Enabled = false;
            cbData.Enabled = false;
            cbParity.Enabled = false;
            cbStop.Enabled = false;
            cbBaud.Enabled = false;
            btnCancelar.Enabled = true;
            btnAceptar.Enabled = false;
            chkConfAvanz.Enabled = false;
            #endregion

            this.Size = new Size(328, 281);
        }

        private void btnCancelar_Click(object sender, EventArgs e)
        {
            //Cierro el puerto serie e habilito los combobox para que estos puedan
ser modificados.
            #region Habilitar botones Puerto Serie
            serialPort1.Close();
            lblPort.Enabled = true;
            lbBauds.Enabled = true;
            cbPort.Enabled = true;
            cbBaud.Enabled = true;
            btnCancelar.Enabled = false;
            btnAceptar.Enabled = true;
            chkConfAvanz.Enabled = true;
```

```
            #endregion

            //Se mostrarán habilitados los combobox de datos, paridad y bits de
stop en el caso de que se hubieran habilitado la vez anterior.
            #region Configuración avanzada
            if (chkConfAvanz.Checked == true)
            {
                cbData.Enabled = true;
                cbParity.Enabled = true;
                cbStop.Enabled = true;
                lbData.Enabled = true;
                lblParity.Enabled = true;
                lblStopBits.Enabled = true;
            }
            else
            {
                cbData.Enabled = false;
                cbParity.Enabled = false;
                cbStop.Enabled = false;
                lbData.Enabled = false;
                lblParity.Enabled = false;
                lblStopBits.Enabled = false;
            }
            #endregion

        }

        private void chkConfAvanz_CheckedChanged(object sender, EventArgs
e)
        {
            //Si marco configuración avanzada, puedo modificar los combobox de
datos, paridad y bits de stop del puerto serie.
            #region Configuración avanzada
            if (chkConfAvanz.Checked == true)
            {
                cbData.Enabled = true;
                cbParity.Enabled = true;
                cbStop.Enabled = true;
                lbData.Enabled = true;
                lblParity.Enabled = true;
                lblStopBits.Enabled = true;
            }

            else
            {
                cbData.Enabled = false;
                cbParity.Enabled = false;
                cbStop.Enabled = false;
                lbData.Enabled = false;
                lblParity.Enabled = false;
```

```
            lblStopBits.Enabled = false;
        }
        #endregion
    }

    private void guardarToolStripButton_Click(object sender, EventArgs e)
    {
        //El siguiente código permite guardar un documento en la carpeta que
el usuario desee.
        if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)
        {
            richTextBox1.SaveFile(saveFileDialog1.FileName,
RichTextBoxStreamType.PlainText);
        }
        else
        {
            MessageBox.Show("Operación Abortada");
        }
    }

    private void imprimirToolStripButton_Click(object sender, EventArgs e)
    {
        //El siguiente código permite imprimir un documento.
        PrintDialog PrintDialog1 = new PrintDialog();
        if (PrintDialog1.ShowDialog() == DialogResult.OK)
        {
            printDocument1.Print();
        }
    }

    private void cortarToolStripButton_Click(object sender, EventArgs e)
    {
        //Este código se usa para cortar un texto seleccionado.
        richTextBox1.Cut();
    }

    private void copiarToolStripButton_Click(object sender, EventArgs e)
    {
        //Este, permite copiarlo al portapapeles.
        richTextBox1.Copy();
    }

    private void pegarToolStripButton_Click(object sender, EventArgs e)
    {
        //Y este pegarlo en el lugar qué el usuario desee.
        richTextBox1.Paste();
    }

    private void registroToolStripMenuItem_Click(object sender, EventArgs e)
    {
```

```
            this.Size = new Size(633, 282);

            #region Barra automática para la caja de texto
            this.richTextBox1.Size = new Size(278, 185);
            richTextBox1.SelectionStart = richTextBox1.Text.Length;
            richTextBox1.ScrollToCaret();
            richTextBox1.Refresh();
            #endregion
        }

        private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
        {
            //Aquí, se determinan las características de impresión.
            e.Graphics.DrawString(richTextBox1.Text, new Font("Calibri", 11,
FontStyle.Bold), Brushes.Black, 60, 80);
        }

        private void verLaInterfazCompletaToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            this.Size = new Size(633, 557);

            #region Barra automática para la caja de texto
            this.richTextBox1.Size = new Size(278, 466);
            richTextBox1.SelectionStart = richTextBox1.Text.Length;
            richTextBox1.ScrollToCaret();
            richTextBox1.Refresh();
            #endregion
        }

        private void ocultarToolStripButton_Click(object sender, EventArgs e)
        {
            this.Size = new Size(328, 281);
        }

       //Las siguientes funciones muestran el código asociado a los botones de la
interfaz.
        //    Características:
        //          - Si se pulsa el botón '*' o '#' se limpiará la pantalla y se pondrá a
cero el contador de números marcados.
        //          - En el caso de marcar alguno de los otros botones, en pantalla
saldrá '*' y el número será guardado en una caja de texto no visible.
        //        - Si el contador es mayor que tres, no se admitirán nuevos números
marcados.

        private void btnAsterisco_Click(object sender, EventArgs e)
        {
            txtPinPuk.Text = "";
            txtPinPukOculto.Text = "";
```

```
    contador = 0;
}

private void btn0_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
    }

    else
    {
        txtPinPukOculto.Text = txtPinPukOculto.Text + "0";
        txtPinPuk.Text = txtPinPuk.Text + "*";
        contador++;
    }
}

private void btnAlmohadilla_Click(object sender, EventArgs e)
{
    txtPinPuk.Text = "";
    txtPinPukOculto.Text = "";
    contador = 0;
}

private void btn1_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
    }

    else
    {
        txtPinPukOculto.Text = txtPinPukOculto.Text + "1";
        txtPinPuk.Text = txtPinPuk.Text + "*";
        contador++;
    }
}

private void btn2_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
    }

    else
    {
        txtPinPukOculto.Text = txtPinPukOculto.Text + "2";
```

```
            txtPinPuk.Text = txtPinPuk.Text + "*";
            contador++;
        }
    }

    private void btn3_Click(object sender, EventArgs e)
    {
        if (contador > 3)
        {
            return;
        }

        else
        {
            txtPinPukOculto.Text = txtPinPukOculto.Text + "3";
            txtPinPuk.Text = txtPinPuk.Text + "*";
            contador++;
        }
    }

    private void btn4_Click(object sender, EventArgs e)
    {
        if (contador > 3)
        {
            return;
        }

        else
        {
            txtPinPukOculto.Text = txtPinPukOculto.Text + "4";
            txtPinPuk.Text = txtPinPuk.Text + "*";
            contador++;
        }
    }

    private void btn5_Click(object sender, EventArgs e)
    {
        if (contador > 3)
        {
            return;
        }

        else
        {
            txtPinPukOculto.Text = txtPinPukOculto.Text + "5";
            txtPinPuk.Text = txtPinPuk.Text + "*";
            contador++;
        }
    }
```

```
private void btn6_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
    }

    else
    {
        txtPinPukOculto.Text = txtPinPukOculto.Text + "6";
        txtPinPuk.Text = txtPinPuk.Text + "*";
        contador++;
    }
}

private void btn7_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
    }

    else
    {
        txtPinPukOculto.Text = txtPinPukOculto.Text + "7";
        txtPinPuk.Text = txtPinPuk.Text + "*";
        contador++;
    }
}

private void btn8_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
    }

    else
    {
        txtPinPukOculto.Text = txtPinPukOculto.Text + "8";
        txtPinPuk.Text = txtPinPuk.Text + "*";
        contador++;
    }
}

private void btn9_Click(object sender, EventArgs e)
{
    if (contador > 3)
    {
        return;
```

```
            }

            else
            {
                txtPinPukOculto.Text = txtPinPukOculto.Text + "9";
                txtPinPuk.Text = txtPinPuk.Text + "*";
                contador++;
            }
        }

        private void btnOk_Click(object sender, EventArgs e)
        {
            //Características del botón OK de validación de código:
            //      - En el caso de que el código guardado en la caja de texto no visible coincida con la clave, se enviará
            //        un '1' o un '0' al puerto serie. El valor del dato que se envía al puerto serie viene determinado por
            //      el estado en el que se encuentra la alarma.
            //      - Si por el contrario no coincide el número marcado con la clave, aparecerá un mensaje de aviso diciendo
            //      que el código introducido es incorrecto.

            if (txtPinPukOculto.Text == "1234" && txtEstado.Text == "Alarma desactivada, necesita clave para activarse")
            {
                serialPort1.Write("1");
                contador = 0;
                txtPinPuk.Text = "";
                txtPinPukOculto.Text = "";
            }
            else if (txtPinPukOculto.Text == "1234" && txtEstado.Text == "la sirena está en funcionamiento, se requiere código para desactivar la alarma")
            {
                    serialPort1.Write("0");
                    contador = 0;
                    txtPinPuk.Text = "";
                    txtPinPukOculto.Text = "";
             }
            else if (txtPinPukOculto.Text == "1234" && txtEstado.Text == "Alarma activa, necesita clave para desactivarse")
            {
                serialPort1.Write("0");
                contador = 0;
                txtPinPuk.Text = "";
                txtPinPukOculto.Text = "";
            }
            else if (txtPinPukOculto.Text == "1234" && txtEstado.Text == "Intruso en casa, se requiere desactivar la alarma en 20 segundos")
            {
                serialPort1.Write("0");
```

```
            contador = 0;
            txtPinPuk.Text = "";
            txtPinPukOculto.Text = "";
        }

        else if (txtEstado.Text == "")
        {
            txtPinPukOculto.Text = "";
            contador = 0;
            MessageBox.Show("No existe comunicación con la alarma, pruebe a
conectarla de nuevo");
        }

        else
        {
            txtPinPuk.Text = "";
            txtPinPukOculto.Text = "";
            contador = 0;
            MessageBox.Show("El código introducido es incorrecto");
        }
    }

    private void serialPort1_DataReceived(object sender,
SerialDataReceivedEventArgs e)
    {
        // Características de la siguiente función:
        //     - Los datos de que recibe el puerto serie se guardan en la variable
recibirDato.
        //     - El dato que se recibe indica el estado en el que se encuentra la
alarma.
        //       - Todos los estados en los que se encuentre la alarma serán
mostrados en una caja
        //       de texto y escritos en un archivo .txt del disco duro.
        //     - La primera vez que se utilice esta aplicación y se reciba un dato
por el puerto
        //       serie se generará de forma automática un archivo .txt en el disco
duro del pc.
        //     - Si se recibe un '4' o un '6' en el puerto serie, indica que la alarma
ha detectado
        //       a un intruso y se enviará un correo electrónico a la dirección que
el usuario haya
        //       establecido.

        recibirDato = this.serialPort1.ReadExisting();
        System.Net.Mail.MailMessage Email;
        switch(recibirDato)
        {
            case"0":

                fecha = DateTime.Now.ToString();
```

```
                registro = fecha + " - Alarma desactivada, necesita clave para
activarse\r\n";
                this.richTextBox1.Text += registro;
                this.txtEstado.Text = "Alarma desactivada, necesita clave para
activarse";
                try
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic,
true);
                    sw.WriteLine(registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                catch
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                    sw.WriteLine("(Catch) " + registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                break;

            case"1":

                fecha = DateTime.Now.ToString();
                registro = fecha + " - Clave correcta, la alarma se activará en 20
segundos\r\n";
                this.richTextBox1.Text += registro;
                this.txtEstado.Text = "Clave correcta, la alarma se activará en 20
segundos";
                try
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic,
true);
                    sw.WriteLine(registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                catch
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                    sw.WriteLine("(Catch) " + registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                break;
```

```
            case "2":

                fecha = DateTime.Now.ToString();
                registro = fecha + " - Alarma activa, necesita clave para
desactivarse\r\n";
                this.richTextBox1.Text += registro;
                this.txtEstado.Text = "Alarma activa, necesita clave para
desactivarse";
                try
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic,
true);
                    sw.WriteLine(registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                catch
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                    sw.WriteLine("(Catch) " + registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                break;

            case "3":

                fecha = DateTime.Now.ToString();
                registro = fecha + " - Intruso en casa, se requiere desactivar la
alarma en 20 segundos\r\n";
                this.richTextBox1.Text += registro;
                this.txtEstado.Text = "Intruso en casa, se requiere desactivar la
alarma en 20 segundos";
                try
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic,
true);
                    sw.WriteLine(registro);
                    this.txtOculto.Clear();
                    sw.Close();
                }
                catch
                {
                    const string fic = @"C:\tmp\SmartAlarm.txt";
                    System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                    sw.WriteLine("(Catch) " + registro);
```

```
                this.txtOculto.Clear();
                sw.Close();
            }
            break;

        case "4":

            fecha = DateTime.Now.ToString();
            registro = fecha + " - la sirena está en funcionamiento, se requiere código PIN para desactivar la alarma\r\n";
            this.richTextBox1.Text += registro;
            this.txtEstado.Text = "la sirena está en funcionamiento, se requiere código para desactivar la alarma";
            To = this.txtCorreo.Text;
            if (To == "")
            {
                To = "proyectoalarmasanjose2014@gmail.com";
            }
            try
            {
                #region Enviar Email
                Email = new System.Net.Mail.MailMessage(From, To, Subject, Message);
                System.Net.Mail.SmtpClient smtpMail = new System.Net.Mail.SmtpClient("smtp.gmail.com");
                Email.IsBodyHtml = false;
                smtpMail.EnableSsl = true;
                smtpMail.UseDefaultCredentials = false;
                smtpMail.Port = 25;
                smtpMail.Credentials = new System.Net.NetworkCredential(From, Contraseña);
                lblEnviar.Text = "";
                lblEnviar.Text = "Enviando E-mail....";
                smtpMail.Send(Email);
                lblEnviar.Text = "";
                #endregion
            }

            catch (System.Net.Mail.SmtpException)
            {
                lblEnviar.Text = "";
                lblEnviar.Text = "E-mail no enviado";
            }
            try
            {
                const string fic = @"C:\tmp\SmartAlarm.txt";
                System.IO.StreamWriter sw = new System.IO.StreamWriter(fic, true);
                sw.WriteLine(registro);
                this.txtOculto.Clear();
```

```
                sw.Close();
            }
            catch
            {
                const string fic = @"C:\tmp\SmartAlarm.txt";
                System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                sw.WriteLine("(Catch) " + registro);
                this.txtOculto.Clear();
                sw.Close();
            }
            break;

        case "5":

            fecha = DateTime.Now.ToString();
            registro = fecha + " - PIN bloqueado, se requiere PUK para activar la alarma\r\n";
            this.richTextBox1.Text += registro;
            this.txtEstado.Text = "Alarma desactivada, necesita clave para activarse";
            try
            {
                const string fic = @"C:\tmp\SmartAlarm.txt";
                System.IO.StreamWriter sw = new System.IO.StreamWriter(fic, true);
                sw.WriteLine(registro);
                this.txtOculto.Clear();
                sw.Close();
            }
            catch
            {
                const string fic = @"C:\tmp\SmartAlarm.txt";
                System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                sw.WriteLine("(Catch) " + registro);
                this.txtOculto.Clear();
                sw.Close();
            }
            break;

        case "6":

            fecha = DateTime.Now.ToString();
            registro = fecha + " - la sirena está en funcionamiento, se requiere código PUK para desactivar la alarma\r\n";
            this.richTextBox1.Text += registro;
            this.txtEstado.Text = "la sirena está en funcionamiento, se requiere código para desactivar la alarma";
            To = this.txtCorreo.Text;
            if (To == "")
            {
```

```
                To = "proyectoalarmasanjose2014@gmail.com";
            }
            try
            {
                #region Enviar Email
                Email = new System.Net.Mail.MailMessage(From, To, Subject,
Message);
                System.Net.Mail.SmtpClient smtpMail = new
System.Net.Mail.SmtpClient("smtp.gmail.com");
                Email.IsBodyHtml = false;
                smtpMail.EnableSsl = true;
                smtpMail.UseDefaultCredentials = false;
                smtpMail.Port = 25;
                smtpMail.Credentials = new
System.Net.NetworkCredential(From, Contraseña);
                lblEnviar.Text = "";
                lblEnviar.Text = "Enviando E-mail....";
                smtpMail.Send(Email);
                lblEnviar.Text = "";
                #endregion
            }
            catch (System.Net.Mail.SmtpException)
            {
                lblEnviar.Text = "";
                lblEnviar.Text = "E-mail no enviado";
            }
            try
            {
                const string fic = @"C:\tmp\SmartAlarm.txt";
                System.IO.StreamWriter sw = new System.IO.StreamWriter(fic,
true);
                sw.WriteLine(registro);
                this.txtOculto.Clear();
                sw.Close();
            }
            catch
            {
                const string fic = @"C:\tmp\SmartAlarm.txt";
                System.IO.StreamWriter sw = new System.IO.StreamWriter(fic);
                sw.WriteLine("(Catch) " + registro);
                this.txtOculto.Clear();
                sw.Close();
            }
            break;
        }

    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
```

```
            //El botón cancelar limpiará la pantalla y pondrá a cero el contador de números marcados.
            txtPinPuk.Text = "";
            txtPinPukOculto.Text = "";
            contador = 0;
        }

       // Las siguientes funciones determinan el tamaño de la interfaz al seleccionar una opción u otra.
        private void puertoSerieToolStripMenuItem1_Click_1(object sender, EventArgs e)
        {
            this.Size = new Size(328, 557);
        }

        private void correoElectrónicoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Size = new Size(328, 326);
        }

        private void pantallaInicialToolStripMenuItem_Click(object sender, EventArgs e)
        {
           this.Size = new Size( 328, 281);
        }

        private void verlaayudaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Al pulsar ver la ayuda del menú, se abre un nuevo formulario con una ventana de ayuda.
            FormAyuda miAyuda = new FormAyuda();
            miAyuda.ShowDialog();
        }
    }
}
```

Código asociado al formulario con el que se abren los archivos .txt existentes (frmRegistroBaseDatos).

```
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;

namespace proyectoAlarma
{
    public partial class frmRegistroBaseDatos : Form
    {
        public frmRegistroBaseDatos()
        {
            InitializeComponent();
        }

        private void frmRegistroBaseDatos_Load(object sender, EventArgs e)
        {
            //El siguiente código se utiliza para abrir el directorio y posteriormente
poder seleccionar el
            // fichero que se desea abrir.
            richTextBox1.Clear();
            openFileDialog1.InitialDirectory = @"C:\";
            openFileDialog1.Title = "Selecciona un Fichero";
            if (openFileDialog1.ShowDialog() != DialogResult.Cancel)
            {
                richTextBox1.LoadFile(openFileDialog1.FileName,
RichTextBoxStreamType.PlainText);
            }
            else
            {
                MessageBox.Show("Selección Abortada");
            }
        }

        private void guardarToolStripButton_Click(object sender, EventArgs e)
        {
            //El siguiente código permite guardar un documento en la carpeta que el
usuario desee.
            if (saveFileDialog1.ShowDialog() != DialogResult.Cancel)
            {
```

```
            richTextBox1.SaveFile(saveFileDialog1.FileName,
RichTextBoxStreamType.PlainText);
        }
        else
        {
            MessageBox.Show("Operación Abortada");
        }
    }

    private void imprimirToolStripButton_Click(object sender, EventArgs e)
    {
        //El siguiente código permite imprimir un documento.
        PrintDialog PrintDialog1 = new PrintDialog();
        if (PrintDialog1.ShowDialog() == DialogResult.OK)
        {
            printDocument1.Print();
        }
    }

    private void cortarToolStripButton_Click(object sender, EventArgs e)
    {
        //Este código se usa para cortar un texto seleccionado.
        richTextBox1.Cut();
    }

    private void copiarToolStripButton_Click(object sender, EventArgs e)
    {
        //Este, permite copiarlo al portapapeles.
        richTextBox1.Copy();
    }

    private void pegarToolStripButton_Click(object sender, EventArgs e)
    {
        //Y este pegarlo en el lugar qué el usuario desee.
        richTextBox1.Paste();
    }

    private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
    {
        //Aquí se determinan las características de impresión.
        e.Graphics.DrawString(richTextBox1.Text, new Font("Calibri", 11,
FontStyle.Bold), Brushes.Black, 60, 80);
    }
  }
}
```