DIVERSA

Pasos para desarrollar una Interfaz Gráfica de Usuario simple usando Visual Studio 2022

Los pasos descritos a continuación te permitirán desarrollar una interfaz gráfica de usuario muy sencilla mediante la cual podrás controlar el encendido y apagado de un LED el cual está conectado al pin RC0 de un microcontrolador PIC18F46k42. La conexión entre la computadora, donde se encuentra la GUI, y el microcontrolador se realiza mediante un cable USB-UART. Aunque la GUI es bastante básica, te ayudará para conocer los pasos a seguir en el desarrollo una GUI utilizando Visual Studio 2022 y el lenguaje C#.

Pasos:

1. Hacer doble clic en el ícono de Visual Studio 2022.

2. Hacer clic en la pestaña **Create a new project**

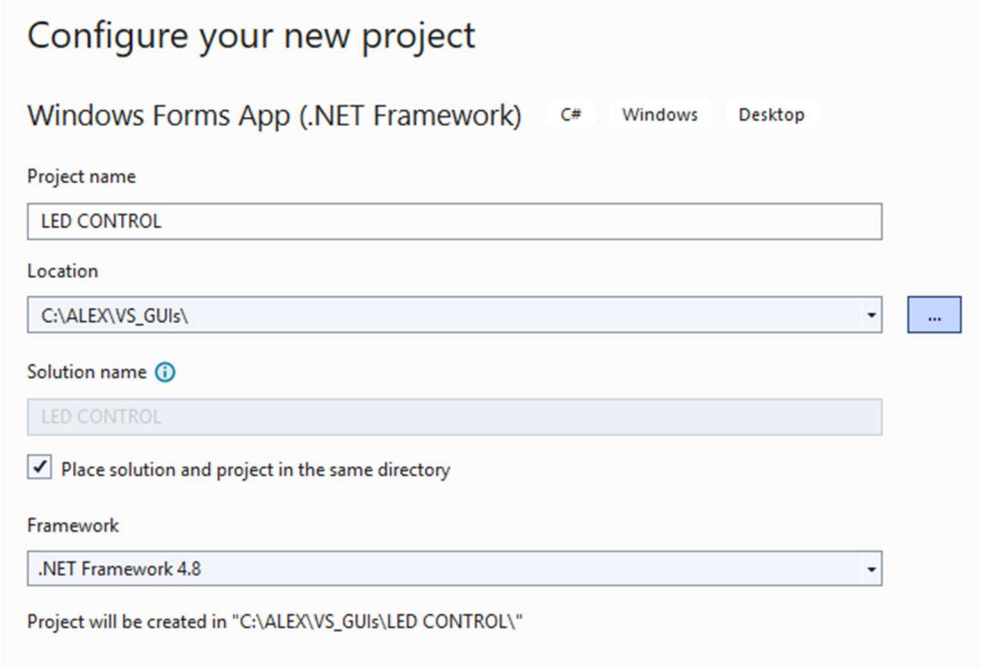
3. Seleccionar **Windows Form App** (**.**Net Framework)

4. Nombramos el Proyecto, definimos su ubicación. Respecto a la casilla **Place solution and Project in the same directory** tenemos dos alternativas:

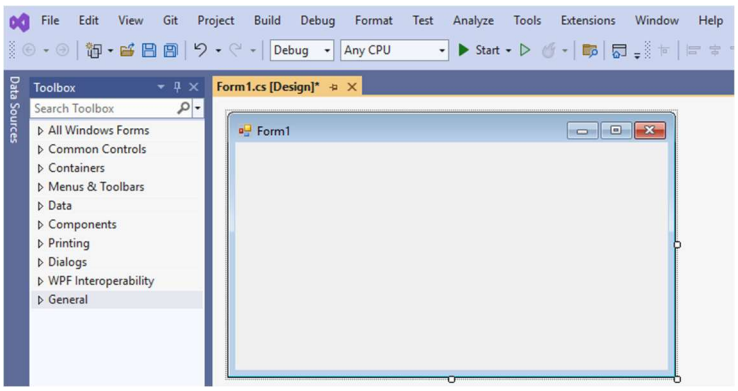
   - Si ponemos un check el proyecto y la solución estarán en el mismo directorio (Users\alxof\source\repos)

   - Sin el check podemos ubicar la solución en otro directorio.

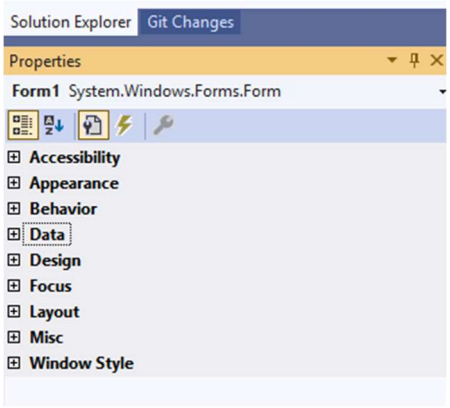
   En Framework seleccionamos .NET Framework 4.8



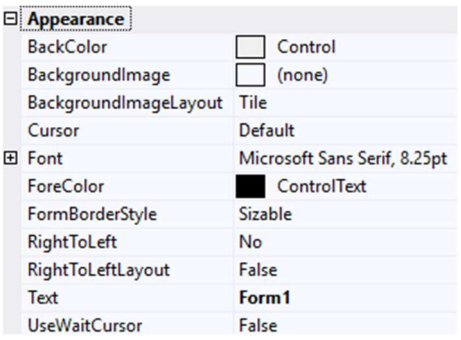
5. Al presionar el botón **Create** deberá aparecer la ventana (parcial) mostrada a continuación.

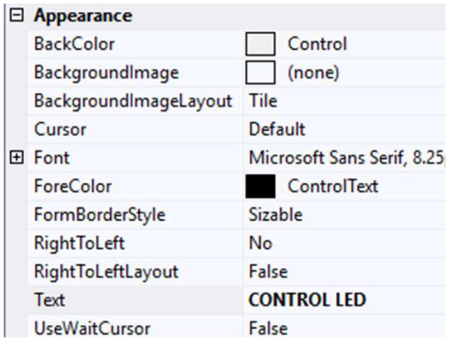


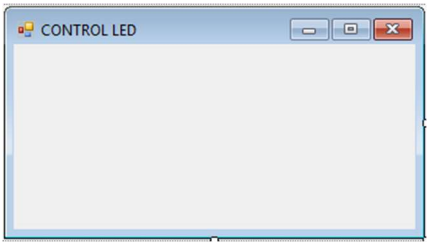
Si hacemos clic con el botón derecho del ratón sobre la forma (Form1) y hacemos clic en Properties, en la parte inferior derecha de la pantalla serán mostradas las propiedades relativas a la forma.



Si hacemos clic en el símbolo **+** en Appearance, en el campo Text es mostrado el nombre asignado a la forma (Form1). Podemos cambiarlo escribiendo un nombre nuevo en el campo Text. Escribiremos CONTROL LED. Se recomienda revisar los tutoriales donde se describe la función de las diferentes propiedades.



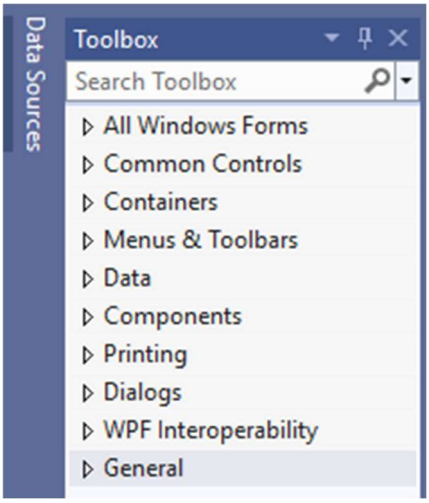


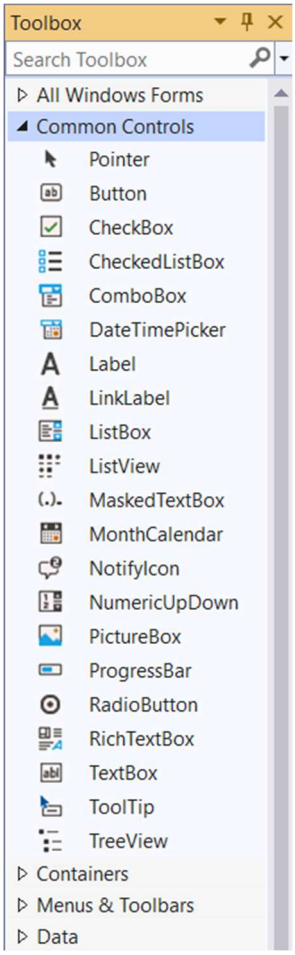


6. Al presionar **Start** (iniciar depuración) veremos el resultado, una forma llamada CONTROL LED en la cual agregaremos los controles requeridos para lograr el funcionamiento deseado de la GUI.

Para salir de la depuración presionar el cuadrado color rojo (detener depuración)

7. En la parte izquierda de la pantalla aparece la **Toolbox (**caja de herramientas**)** en la cual encontraremos los diferentes controles y recursos para la construcción de la GUI. Al hacer clic en **Common Controls** aparecerá una lista con los recursos disponibles para construir la GUI.



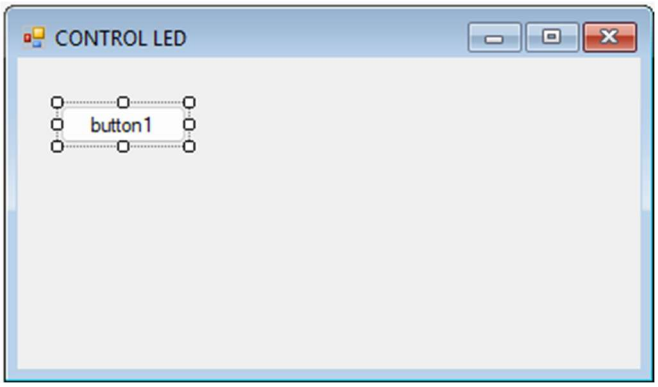


Descripción breve de algunos controles

- **Button**: Usado para iniciar acciones o comandos cuando se hace clic sobre este.
- **CheckBox**: Permite al usuario seleccionar o deseleccionar opciones.
- **TextBox**: Habilita al usuario agregar y mostrar texto.
- **ListBox**: Permite mostrar una lista de alternativas que los usuarios pueden seleccionar.
- **Label**: Son utilizados para mostrar texto que no puede ser editado por el usuario.

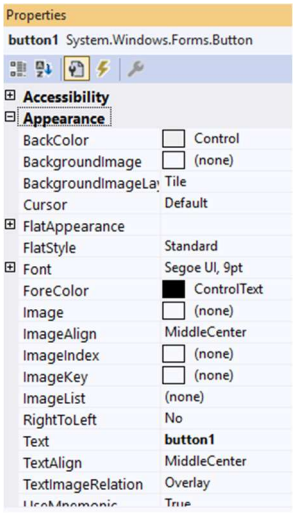
8. Para agregar una instancia del control **Button**, hacer clic en **Button** y luego hacer clic sobre la forma, la instancia del control Button es incorporada a esta.

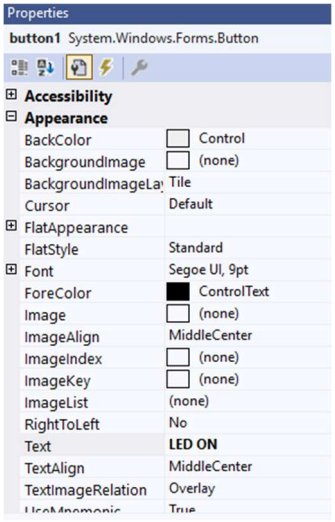
9. Podemos ver las propiedades del control **Button** haciendo clic encima de button1. En la parte inferior derecha de la pantalla aparecerán las **Propiedades** relacionadas con el control Button.

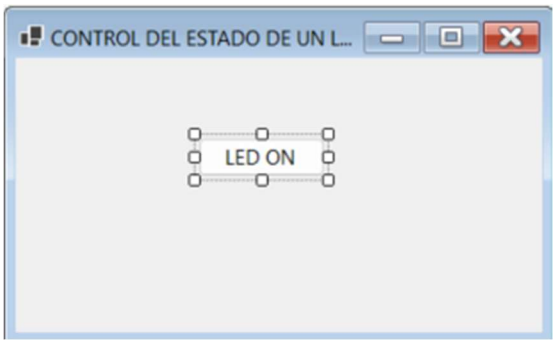


- Accessibility
- Appearance
- Behavior
- Data
- Design
- Focus
- Layout

10. En la categoría **Appearance** aparece el campo **Text** y a la derecha **button1.** En Text indicamos el nombre que deseamos aparezca encima del control **Button**. Cambiaremos button1 por **LED ON**.







11. En la categoría **Design** aparece el campo **(Name)** y a la derecha button1. En Name indicamos el nombre de la variable, es decir el nombre que será utilizado en la programación. Cambiaremos a **BtnEncender**.

12. Al hacer doble clic en el control (Button en este caso) abrirá **Form1.cs** donde veremos el código generado asociado a dicho control. Ahí escribiremos el código requerido para que se realice la función deseada cuando el botón sea presionado.

```
namespace LED_CONTROL
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void BtnEncender_Click(object sender, EventArgs e)
        {

        }
    }
}
```

13. Si ubicamos el cursor encima de InitializeComponent(), presionamos clic derecho y hacemos clic en **Go To Implementation** la inicialización para varias propiedades del control Button será mostrada.

```
private void InitializeComponent()
{
    this.BtnEncender = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // BtnEncender
    //
    this.BtnEncender.Location = new System.Drawing.Point(25, 28);
    this.BtnEncender.Name = "BtnEncender";
    this.BtnEncender.Size = new System.Drawing.Size(75, 23);
    this.BtnEncender.TabIndex = 0;
    this.BtnEncender.Text = "LED ON";
    this.BtnEncender.UseVisualStyleBackColor = true;
    this.BtnEncender.Click += new System.EventHandler(this.BtnEncender_Click);
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(368, 184);
    this.Controls.Add(this.BtnEncender);
    this.Name = "Form1";
    this.Text = "CONTROL LED";
    this.Load += new System.EventHandler(this.Form1_Load);
    this.ResumeLayout(false);

}
```

14. Agregaremos un segundo Button y en Text escribiremos **LED OFF** y en Name **BtnApagar**. Hacemos doble clic en el control y chequeamos el código generado.

```
namespace LED_CONTROL
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void BtnEncender_Click(object sender, EventArgs e)
        {

        }

        private void BtnApagar_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Como se puede apreciar en la imagen anterior, han sido generadas las funciones **BtnEncender_Click**() y **BtnApagar_Click**() donde escribiremos el código que será ejecutado cuando se presionen los botones respectivos.

15. Agregamos el código necesario para enviar la letra E cuando sea presionado el botón LED ON y para enviar la letra A cuando se presione el botón LED OFF.

```
private void BtnEncender_Click(object sender, EventArgs e)
{
    serialPort1.Write("E");
}

private void BtnApagar_Click(object sender, EventArgs e)
{
    serialPort1.Write("A");
}
```

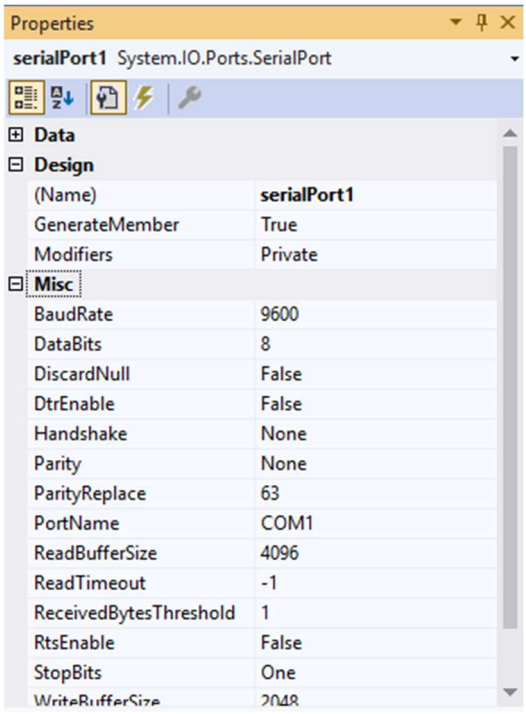
Aparece el mensaje mostrado a continuación: (cuándo?)

```
private void BtnEncender_Click(object sender, EventArgs e
{
    serialPort1.Write("E");
}

CS0103: The name 'serialPort1' does not exist in the current context
Show potential fixes (Alt+Enter or Ctrl+.)

private                                              s e)
```

16. Agregaremos y configuraremos el serialPort1 para eliminar el error. En el ToolBox en componentes hacemos doble clic en **SerialPort**. Hacemos clic con el botón derecho del ratón sobre la imagen de SerialPort y hacemos clic en Propiedades.

En **PortName** debemos seleccionar el puerto serial que será utilizado para la comunicación con el sistema que contiene el microcontrolador con el cual será controlado el estado del LED. En el **Device Manager** podemos ver los puertos disponibles.

Es importante configurar correctamente parámetros tales como el BaudRate, los DataBits, si se utilizará bit de paridad y cuantos bits de stop serán utilizados.



Luego escribimos el código para que al inicializar el sistema se abra el puerto serial. El código es mostrado a continuación:

```
namespace LED_CONTROL
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            try { serialPort1.Open(); }
            catch (Exception msg) { MessageBox.Show(msg.ToString());}

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void BtnEncender_Click(object sender, EventArgs e)
        {
            serialPort1.Write("E");
        }

        private void BtnApagar_Click(object sender, EventArgs e)
        {
            serialPort1.Write("A");
        }
    }
}
```

Cada vez que se presione el botón LED ON la letra **E** será enviada por el puerto serial y cuando sea presionado el botón LED OFF será enviada la letra **A**. El programa en el microcontrolador debe poder recibir y leer las letras (A y E en el ejemplo) enviadas desde la interfaz y ejecutar las acciones correspondientes.
La efectividad de la GUI puede ser verificada, sin tener el sistema con el microcontrolador, utilizando una terminal en la PC tal como la Terminal v1.93b. Se pueden habilitar para tal fin un par de puertos virtuales en la PC.

Asumiendo que la interfaz funciona correctamente, lo probaremos pronto con el sistema, procederemos a escribir el programa para el microcontrolador. Utilizaremos, por razones que serán explicadas posteriormente, el PIC18F46k42. Serán utilizados el MPLAB X IDE v6.25, el compilador XC8 y el MCC Melody.

A continuación, el programa parcial del microcontrolador:

```
int main(void)
{
    uint8_t letra = 0;
    SYSTEM_Initialize();

    // Enable the Global Interrupts
    //INTERRUPT_GlobalInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    while(1)
    {
        if(UART1_IsRxReady())
        {
          letra =  UART1_Read();
          if(letra == 'E')
          {
            LED0_SetHigh();
          }
          if(letra == 'A')
          {
            LED0_SetLow();
          }
        }
    }
}
```

Usualmente verificamos la efectividad del programa realizando una simulación con la Suite de PROTEUS. El PIC18F46k42 no cuenta con un modelo en el simulador y para verificar la efectividad del programa anterior, **si no tenemos lista la GUI**, podemos utilizar la Terminal v1.93b. El PIC18F46k42 se encuentra en un EASYPIC v7 y debemos conectar la PC con este tal como se muestra en la imagen.

En la terminal debemos asignar un puerto que se encuentre habilitado. El EASYPIC v7 cuenta con un módulo que permite la conversión de UART a USB.

Una vez realizada la conexión entre la PC y el EASYPIC v7, conectado el cable, procederemos de la siguiente forma:

1. Abrir la Terminal v1.93b y asignar el puerto serie.
2. Activar el EASYPIC v7, los pines RC6 y RC7 deberán estar encendidos.
3. En la terminal hacer clic en el botón conectar.

Si el código es correcto, cuando se escriba la letra E el LED en RC0 se encenderá y cuando se escriba la letra A dicho LED deberá apagarse.

Interfaz gráfica de usuario (GUI)

Para hacer la prueba con la interfaz gráfica de usuario y el sistema, en vez de abrir la terminal de usuario se activa la GUI (garantizar la asignación correcta del puerto serial).

Activar el EASYPIC v7, los pines RC6 y RC7 deberán estar encendidos.

Si el código es correcto, cuando se presione el botón LED ON el LED en RC0 se encenderá y cuando se presione el botón LED OFF dicho LED deberá apagarse.

Elaborado por:
Alejandro A Méndez T
Abril 08, 2025

In Visual Studio 2022, a **serial port** acts as a communication bridge between your graphical user interface (GUI) and a microcontroller-based system – like a PIC board. The serial port facilitates communication between your application and external devices like microcontrollers. It's the digital equivalent of passing notes between two brains: one running your Windows App, and the other embedded in hardware.

What is a Serial Port

- A serial port transmits data one bit at a time over a communication line (typically UART).
- In VS, you can access it using the System.IO.Ports.SerialPort class in C#
- It connects hardware via USB to serial converters or native COM ports.

Role Between GUI and Microcontroller

With the serial port your GUI can:

Its role in a GUI is to provide a use-friendly interface for controlling and monitoring the microcontroller-based system.

- Send commands to the microcontroller (e.g., "turn on LED" or "read sensor"
- Receive data from the microcontroller (e.g., temperature readings, status updates)

Configuration

You configure the SerialPort object with various parameters to match the microcontroller's serial communication settings. These include:

- PortName
- BaudRate
- Parity
- DataBits
- StopBits
- Handshake