*CREANDO INGENIOS*
*ISSN: 3028-8924*
*Correo: editor.revista@tecnologicoismac.edu.ec*
*URL: https://ismaconline.net/investigacion/index.php/CreaIngenio_2021/index*
*Volumen 2, Número 1 / Enero – Junio 2022 pp. 60-65*

# Interfaz entre un PLC y Visual Studio con C#

## *Interface between a PLC and Visual Studio with C#*

**Ricardo Tipán** [1]

[1] Universidad Tecnológica Israel–Carrera de Sistemas de Información, 170516, Quito, Ecuador

**Fecha de recepción:** abril 2022 **Fecha de aprobación:** mayo 2022

**RESUMEN**

Con el avance del internet y las tecnologías electrónicas se dio búsqueda a la automatización industrial, donde el uso de dispositivos electrónicos aliviara la carga de tareas repetitivas en las fábricas. Es así que dio origen a una nueva evolución, en donde el profesional dejaba de realizar el trabajo y comenzaba a vigilar que la máquina funcionara de forma correcta. Por ende, a partir de los años 90, y hasta la actualidad se modernizaron las características físicas de los dispositivos y los módulos de programación, abriendo las posibilidades de programación, control y los canales de comunicación. En este sentido, un PLC es denominado como el cerebro electrónico, que se encarga de accionar a otros componentes de maquinaria, y estas realicen acciones de tipo industrial.

***Palabras Clave:*** PLC (Programable Logic Controller), CLR (Common Language Runtime), LADER.

**ABSTRACT**

With the advancement of the internet and electronic technologies, industrial automation was sought, where the use of electronic devices would alleviate the burden of repetitive tasks in factories. This is how it gave rise to a new evolution, where the professional stopped doing the work and began to monitor that the machine worked correctly. Therefore, from the 90s, and up to the present, the physical characteristics of the devices and the programming modules were modernized, opening the possibilities of programming, control and communication channels. In this sense, a PLC is called the electronic brain, which is responsible for driving other machinery components, and these perform industrial-type actions.

***Key Words:*** PLC (Programable Logic Controller), CLR (Common Language Runtime), LADER.

[1] Estudiante de Ingeniería en Sistemas, javi_asm@hotmail.com

*CREANDO INGENIOS*
*ISSN: 3028-8924*
*Correo: editor.revista@tecnologicoismac.edu.ec*
*URL: https://ismaconline.net/investigacion/index.php/CreaIngenio_2021/index*
*Volumen 2, Número 1 / Enero – Junio 2022 pp. 60-65*

## 1. INTRODUCCIÓN

En esta investigación se tratará sobre los diferentes elementos que permiten que sea posible la comunicación entre un dispositivo de control Industrial, con una interfaz desarrolla en el IDE Visual Studio.

Programable Logic Controller por sus siglas en inglés, es un dispositivo electrónico que se programa para realizar acciones de control automáticamente. El PLC es un cerebro que activa componentes de maquinarias que ejecutan tareas que pudieran ser peligrosas para el ser humano.

Los PLC se usan en la actualidad en todo tipo de aplicaciones industriales, resolviendo requerimientos en control de procesos y secuencias de la maquinaria, dentro del sector industrial y ha penetrado las aplicaciones domésticas y comerciales con mayor auge en la última década.

**Figura 1.** Control Industrial

## 2. DESARROLLO

El PLC tiene interfaces de entrada y salida que sirven de comunicación entre elementos como son:

Entrada: sensores

Salida: sirve de comunicación entre la CPU y actuadores como por ejemplo motores.

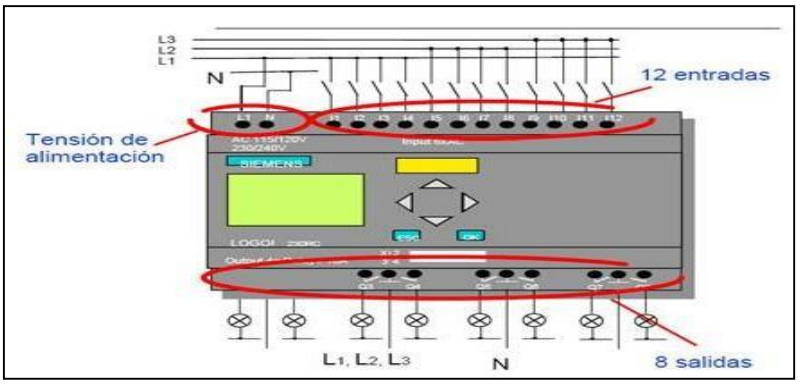


**Figura 2.** PLC Entradas/Salidas

El PLC maneja su propio lenguaje de configuración y programación como por ejemplo el lenguaje (LADER) o como son los diagramas de Bloques lo que deseamos es crear comunicación y verificar si es posible desarrollar una interfaz de usuario para controlar estos dispositivos.

C # requiere una implementación del Common Language Runtime (CLR) para ejecutar aplicaciones desarrolladas en ese idioma o en cualquier lenguaje .NET. Lo que se puede hacer es que la aplicación en C # lea y escriba datos desde el PLC usando el protocolo OPC. Hay algunas bibliotecas diferentes para usar OPC que están disponibles para .NET Framework.

Librería Libnodave es una biblioteca libre de funciones para comunicar con PLC, utilizando adaptadores MPI/PPI o Ethernet. Tiene bibliotecas para Linux, Windows (32 bits), .NET y Para programar podemos elegir una gran variedad de lenguajes y entornos: C, C++, C#, Delphi, Pascal, Perl, Visual Basic y Visual Basic for Aplications.

Otra librería es LibNodave.dll es una librería gratuita para VB.NET, Java, C, C++, etc para comunicarse con los autómatas de Siemens S7 300 y S7 400. Se pueden realizar desde aplicaciones de escritorio hasta apps para moviles y tabletas, la comunicación puede realizarse desde ethernet, MPI o profibus.

Como Ejemplo usaremos la librería Libnodave.dll

LibNodave.dll es una librería gratuita para VB.NET, Java, C, C++, etc para comunicarse con los autómatas de Siemens S7 300 y S7 400. Se pueden realizar desde aplicaciones de escritorio hasta apps para móviles y tabletas, la comunicación puede realizarse desde ethernet, MPI o profibus.

Desde este link se puede descargar la librería libnodave de forma gratuita:

libnodave.dll

Al descomprimir el fichero que nos hemos descargado podemos ver diferentes directorios con algunos ejemplos de VB, de Pascal, de PERL, etc.

Para trabajar en .net tenemos la librería libnodave.net.dll, debemos de crear una referencia a la dll libnodave.net:

```
'Variable daveOSerialType
Private fds As libnodave.daveOSserialType
'Variable daveInterface
Private di As libnodave.daveInterface
'Variable daveConnection
Private dc As libnodave.daveConnection
```

**Figura 3.** Variables

Para crear una conexión, podemos crear un método que realiza la conexión al PLC, donde p_port y p_ip son el puerto y la IP del PLC respectivamente:

***CREANDO INGENIOS***
*ISSN: 3028-8924*
*Correo: editor.revista@tecnologicoismac.edu.ec*
*URL: https://ismaconline.net/investigacion/index.php/CreaIngenio_2021/index*
*Volumen 2, Número 1 / Enero – Junio 2022 pp. 60-65*

```
Public Function conectar() As Boolean
    Try
      If fds.rfd = 0 Then
         fds.rfd = libnodave.openSocket(p_port, p_ip)
         fds.wfd = fds.rfd
         di = New libnodave.daveInterface(fds, "IF1", 0,
libnodave.daveProtoISOTCP, libnodave.daveSpeed187k)
         di.setTimeout(1000000)
         dc = New libnodave.daveConnection(di, 0, p_rack, p_slot)
         conexion = dc.connectPLC
      End If
      If conexion <> -1 Then
         Return True
      Else
         Return False
      End If
    Catch ex As Exception
        regErrores = ex.Message
        Return False
    End Try
End Function
```

**Figura 4.** Funciones

La generación de las tramas Host-Link consistirá en una serie de funciones que generen una variable tipo String que incluya la trama con su respectivo FCS y finalización de bloque. Lo más conveniente es definir una serie de constantes que faciliten la creación de las tramas:

```
/***BLOQUES INIT/FIN***/
const String Init_Trama = "@00";
const String End_Trama = "*\r";

/***COMANDOS HOST-LINK***/
const String Read_IR = "RR"; //Lectura canal IR (0000-0255)
const String Write_IR = "WR"; //Escritura canal IR (0000-0252)
const String Abort = "XZ"; //Abortar la comunicación
. . .
```

**Figura 5.**

Una estructura posible para el algoritmo que genere las tramas es el siguiente:

***CREANDO INGENIOS***
*ISSN: 3028-8924*
*Correo: editor.revista@tecnologicoismac.edu.ec*
*URL: https://ismaconline.net/investigacion/index.php/CreaIngenio_2021/index*
*Volumen 2, Número 1 / Enero – Junio 2022 pp. 60-65*

```
private void Send_Trama(String orden, String datos)
{
        String trama = Init_Trama + orden + datos;
        String FCS = ComputeFCS(trama); //Calcula el FCS
        trama = trama + FCS.ToUpper() + End_Trama;
        serialPort.Write(trama); //Se envía a través del puerto serie
        String lectura = serialPort.ReadExisting();
}

private String ComputeFCS(String trama)
{
        //Generación del FCS
}
```

**Figura 6.**

Una vez realizada la conexión con el PLC con éxito, podemos leer y escribir del PLC, para leer y escribir de un DB:

```
'Lectura de un db del PLC
 res = dc.readBytes(libnodave.daveDB, DB, inicio, longitudLeer, Nothing)
 dc.getS32At(puntero)

'Escritura de un db del PLC
 res = dc.writeBytes(libnodave.daveDB, DB, inicio, longitud, escriure)
```

**Figura 7.**

Registrar Datos es un procedimiento que lo que haces es crear la conexión a la base de datos y ejecutar el comando pasado y la desconexión de la Base de Datos.

```
  Public Sub RegistrarDatos(ByVal Querystring As String)

      Dim Conexion As New OleDbConnection
      Dim Command As New OleDbCommand

      On Error GoTo fallo
      Conexion.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\OPC Client jm76\BD.accdb;Persist Security Info=False;"
      Conexion.Open()
      'MsgBox("connectados") 'Hasta aqui es el codigo para conectarse a la base de datos
      '"INSERT INTO camaras (Nombre,Referencia) VALUES ('Sala de Tetrabricks ', 12 )"
      Command.CommandText = Querystring
      Command.Connection = Conexion
      Command.ExecuteNonQuery()
      Conexion.Close()
      Exit Sub
fallo:
      MsgBox(Err.Description, MsgBoxStyle.Critical, "ERROR al Escribir en BD")
  End Sub
```

**Figura 8.**

## 3. CONCLUSIONES

Una aplicación de escritorio con VB.NET conectada a un autómata VIPA Speed 7 y funciona a la perfección sin ningún tipo de problemas.

Con esta librería podemos crear fácilmente un enlace entre una aplicación de esritorio, una app para móvil.

## REFERENCIAS

1. PLC-HDMI. (01 de 01 de 2020). Control Industrial. Obtenido de PLAC-HDMI SCADAS: https://plc-hmi-scadas.com/004.php
2. Pomares, J. (01 de 01 de 2020). Control de un PLC. Obtenido de Host-Link: https://rua.ua.es/dspace/bitstream/10045/18968/1/Practica3.pdf
3. Technologies, I. (05 de 02 de 2020). Interfaz entre u PLC y Visual Studio. Obtenido de Interfaz: https://iztechnologies.wordpress.com/2018/02/11/interfaz-entre-un-plc-siemens-y-visual-studio-con-c-intermedio/