Cliente OPC realizado en Visual Basic .NET

Por Marlon Martínez
Analista de Sistemas (CENTELSA)

Descarga el código fuente (80 KB).

Este articulo y el código fuente incluido permiten conocer como con Visual Basic .NET es posible realizar un Cliente OPC que obtenga datos de un servidor OPC conectado a un PLC o más.

Contenido

Sección 1

Acerca de OPC

Leame - SOBRE OPC (189 KB).

Sección 2

El Cliente OPC Realizado en Visual Basic .NET.

- El programa ejemplo y este articulo se dirigen principalmente a los interesados en desarrollar aplicaciones .NET Windows Forms para la industria, en particular aplicaciones que permitan la comunicación con controladores como PLCs u otros dispositivos electrónicos.
- El programa esta desarrollado en Visual Basic .NET se basa en la especificación OPC, si usted no sabe que es OPC, le recomiendo leer la Sección 1 "Acerca de OPC".
- El programa usa COM Interop pues OPC esta basado en COM y aún OPC Foundation no ha desarrollado una especificación para el Framework .NET.
- El programa ejemplo no es la última palabra en clientes OPC con .NET solo quiere mostrarle una idea de lo que se puede hacer.
- En este artículo documentaremos y detallaremos las porciones de código más relevantes. De todas formas usted puede descargar el código fuente completo.

Prueba del cliente OPC en Visual Basic .NET

1. Lo primero que usted debe hacer es tener una conexión física a un PLC, ya sea por medio de una tarjeta de Adquisición de datos o por el puerto serial.

   Como lo más seguro es que a usted no le quede tan fácil hacer esto, pues no se tiene un PLC a la mano y la cuestión no es tan simple como parece pues se requieren elementos y conocimientos especiales para lograr llevar información desde un PLC a un PC, no se preocupe, siga leyendo.
2. Debe instalar un Servidor OPC. Aunque el documento que recomendé anteriormente explica muy bien lo que es un Servidor OPC, le recuerdo que este es un software que encapsula todo un conjunto de controladores de PLCs y le puede entregar información a un cliente con el estándar OPC (Aunque pueden en ocasiones ofrecer DDE, no explicaré esto). La complejidad de los controladores la maneja el Servidor OPC y este le habla a un cliente solo OPC.
3. Si desea fácilmente cumplir los puntos 1 y 2, instale un Servidor OPC que también le sirva para generar datos simulados como si estuviese conectado a un PLC real. Le recomiendo usar el programa Omron pues con base en este seguiremos el ejemplo. Este programa lo puede descargar de Internet desde varios sitios, le cito estos dos:

   http://www.plcdriver.com/omronopc.htm
   http://www.ingearopc.com/html/omronopc.html

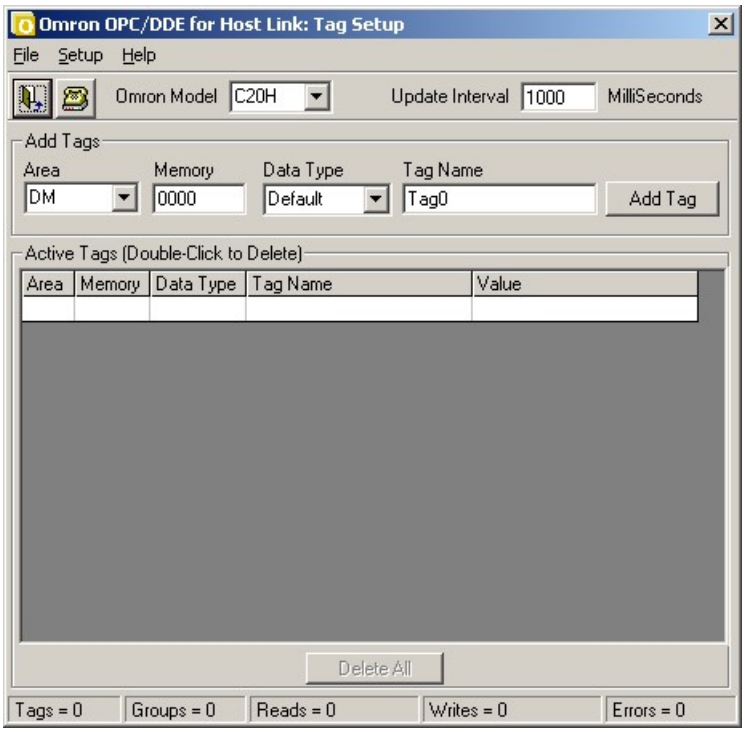
   Estos sitios le pedirán realizar un registro antes de descargar el programa

Instale el programa una vez descargado.

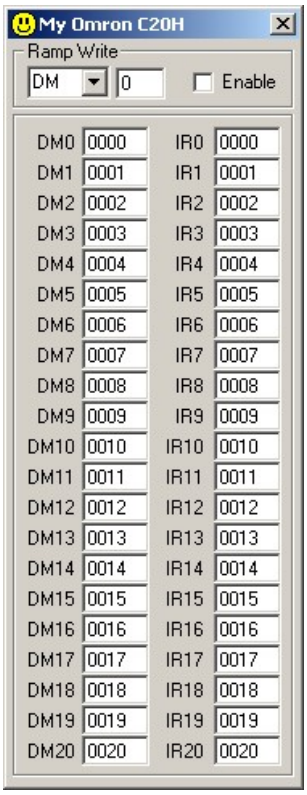
Le recuerdo que OPC es un estándar, el Cliente OPC realizado en .NET puede conectarse a cualquier servidor OPC, si usted tiene instalado o ha descargado un servidor OPC diferente al sugerido, consulte la ayuda de ese programa para saber como simula datos y como un cliente debe nombrar las variables. Regularmente la forma como operan los diversos Servidores OPC es muy similar.

4. Voy a asumir que usted va a usar el programa Omron. (Aunque, como indique anteriormente, el Cliente OPC le funciona con cualquier servidor OPC que este leyendo un PLC Físico, o que también emule datos como lo hace Omron)
5. Una vez instalado el Servidor OPC y Emulador Omron, ejecútelo (Inicio - Programas …)
6. Notará en la barra de tareas un icono amarillo.

7. Haga clic derecho sobre el icono Amarillo y seleccione "Configure". Se presentará la siguiente ventana (Servidor OPC)

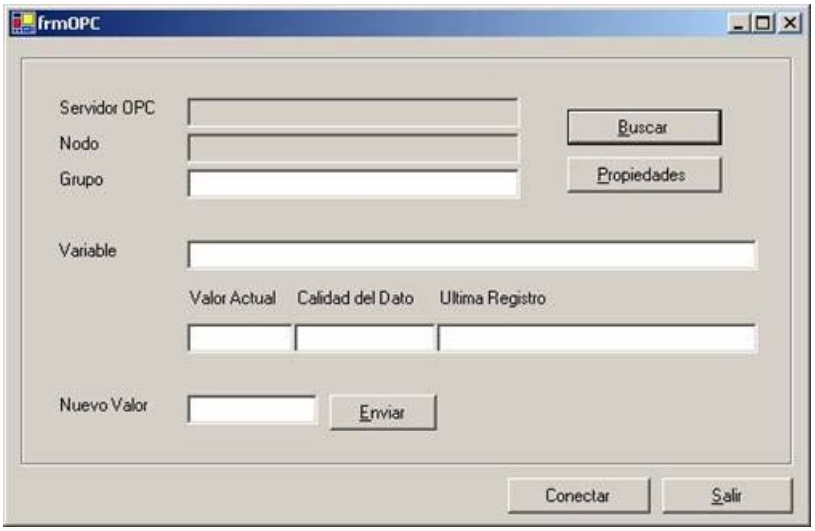


8. Bajo la Opción de menú "Setup", seleccione "Show PLC Simulator". Se deberá presentar la siguiente Ventana:



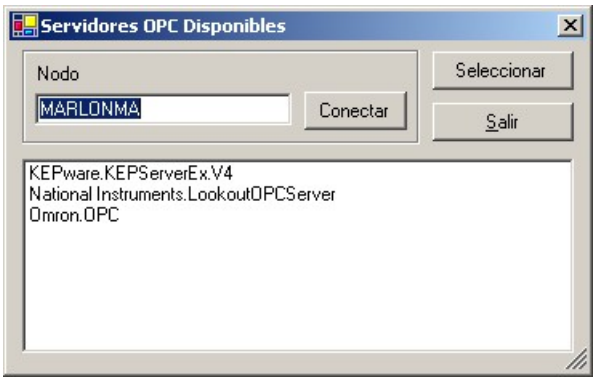
9. Lo que hará ahora es mapear la dirección DM1 desde la Ventana obtenida en el punto 7. En la caja "Memory" escriba 1, en lugar de los Ceros que aparecen de manera predeterminada. En "Tag Name", escriba Tag1, en lugar de Tag0 (Aunque en realidad usted puede colocar un nombre representativo de la variable, pero lo deberá tener presente pues luego esté nombre lo indicaremos en el Cliente OPC escrito en Visual Basic .NET). Presione Add Tag y notará que se crea un nuevo Tag en la Grilla y ya le está mostrando el Valor actual del Tag, ósea "0001". Vaya a la ventana obtenida en el punto anterior y haga un cambio en el valor DM1 y notará que este cambio se actualizará en el Tag que acaba de crear.
10. Si hasta aquí todo ha funcionado sin problemas, ejecute el Cliente OPC en VBNET. Ya sea el Assembly directamente o desde código fuente.

    Tendremos la ventana principal:



11. El programa cliente OPC debe saber cual será el servidor que le suministre datos, en este caso "Omron". Presione el botón buscar y aparecerá la siguiente ventana mostrándole todos los servidores OPC disponibles en su equipo (usted

puede comunicarse con otro equipo, en esta caso la Automatización usa DCOM)

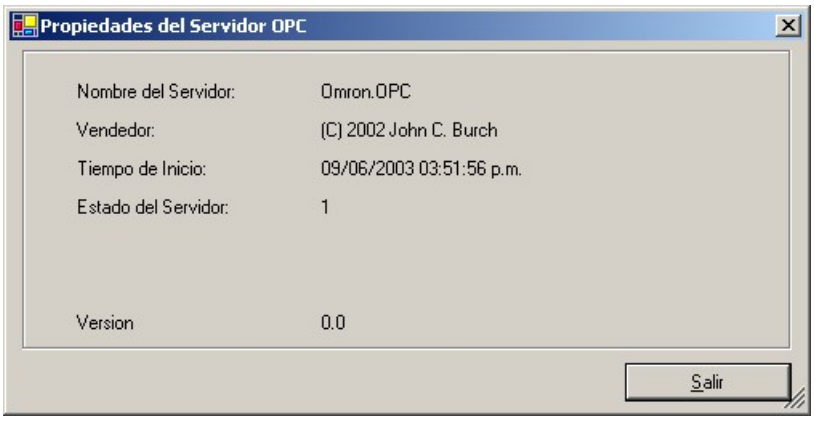


Veamos como este formulario logra obtener los Servidores OPC instalados en el equipo que se señale en el campo Nodo:

```
Private Sub frmServidoresOPC_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyB
        'Por defecto se asume que el cliente se conectará
        'a un servidor OPC local. Para lo cual podemos usar
        'usar el objeto Environment
        txtNodo.Text = Environment.MachineName
        'Llamamos al metodo que nos permite realizar la conexion OPC
        'para obtener los Servidores OPC locales
        Conectar()
    End Sub


Sub Conectar()
    'Arreglo para los Servidores OPC encontrados
    Dim arregloOPCServer As Object
    Dim i As Integer
    'Instancia de la Clase OPCServer en la automatizacion OPC
    'El proyecto debe tener una referencia a la libreria(COM)
    'OPCAutomation. Regularmente este libreria se instala
    'con la instalcion de un servidor OPC.
    Dim objOPCServer As New OPCAutomation.OPCServerClass()

    'Limpiar la lista de Servidores
    lstServidores.Items.Clear()

    Try
        'El metodo GetOPCServer de OPCServer nos devuelve el arreglo
        'de servidores OPC que se encuentren en el nodo, esto ultimo
        'es parametro del metodo
        arregloOPCServer = objOPCServer.GetOPCServers(txtNodo.Text)

        'Se recorre el arreglo de servidores y se agregan en la lista
        For i = LBound(arregloOPCServer) To UBound(arregloOPCServer)
            lstServidores.Items.Add(arregloOPCServer(i))
        Next i
    Catch ex As System.Exception
        MsgBox("No fué posible obtener Servidores OPC en el Nodo especificado", MsgBoxStyle.Critic
    End Try

End Sub
```

La librería OPCAutomation se instala con el Servidor OPC. La empresa proveedora del Servidor OPC, no se inventa por su propia cuenta la librería OPCAutomation, simplemente toma el código fuente y la especificación ofrecida por OPCFoundation para crear el compilado(dll) que se instala a la vez que se instala su Servidor OPC. Algunos proveedores de Servidores OPC instalan la especificación genérica de OPCAutomation y además una librería de automatización optimizada para conectarse a su propio Servidor OPC. Queda a criterio del desarrollador del Cliente OPC determinar cual librería usar.

12. De un clic para resaltar el item Omron en la lista y presione el botón "Seleccionar". Presione el Botón Salir para regresar a la ventana Principal.
13. Presione propiedades desde la ventana principal para obtener detalles sobre el Servidor OPC Seleccionado.

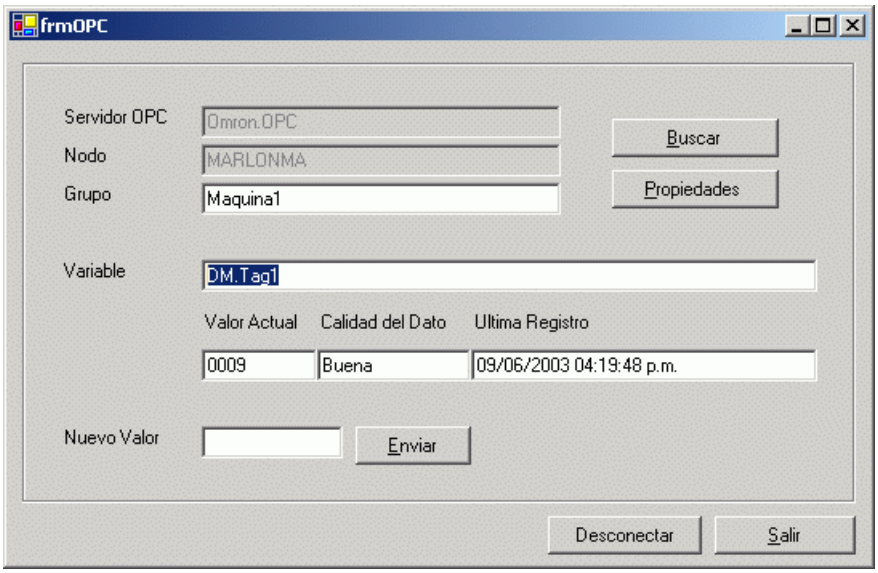


Estas propiedades se obtienen en el evento load del formulario.

```
Private Sub frmPropiedadesOPC_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles My
    'Crea una instancia OPCServer
    Dim objOPCServer As New OPCAutomation.OPCServerClass()

    Try
        'Se conecta al Servidor con el código del Servidor y Nodo
        objOPCServer.Connect(Me.Servidor, Me.Nodo)
        'Obtiene las propiedades del Servidor
        Me.lblNombreServidor.Text = objOPCServer.ServerName
        Me.lblVendedor.Text = objOPCServer.VendorInfo
        Me.lblTiempoInicio.Text = objOPCServer.StartTime
        Me.lblVersion.Text = objOPCServer.MajorVersion & "." & objOPCServer.MinorVersion
        Me.lblEstadoServidor.Text = objOPCServer.ServerState
    Catch ex As System.Exception
        MsgBox("No se encontró disponible el Servidor OPC", MsgBoxStyle.Critical, "ERROR")
    End Try
End Sub
```

Presione Salir para regresar a la ventana principal

14. En Grupo digite por ejemplo "Maquina1"
15. En Variable digite "DM.Tag1" para mapear la Variable que también se mapeo en el Servidor OPC. Presione Conectar y obtendrá el valor actual de la dirección de memoria "uno" en el Simulador del PLC. Haga cambios en el simulador y compruebe que se reflejen en el Cliente OPC. (Qué bonito ¿cierto?)



Veamos los detalles para saber todo lo que sucede cuando se presiona clic en el boton "Conectar"

```
    Private Sub btnConectar_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnConectar.Click
        'Se conecta o desconecta del Servidor OPC
        If btnConectar.Text = "Conectar" Then
            btnConectar.Text = "Desconectar"
            IniciarLectura()
        Else
            btnConectar.Text = "Conectar"
            TerminarLectura()
        End If
    End Sub
```

Nótese que el mismo bóton sirve para realizar la conexión y desconexión.

A continuación los métodos IniciarLectura y TerminarLectura

```
Private Sub IniciarLectura()
    'Creación y configuración del Arreglo del Items
    'Como podrán ver, el ejemplo maneja un solo item
    Dim Items(1, 2) As String

    Items(0, 0) = Trim(txtVariable.Text)
    Items(0, 1) = 1
    Items(0, 2) = True

    'Creación de la Conexión con el Servidor OPC
    'El objeto objOPC se declaró en sección general
    'y es una instancia de una clase que facilita la conexion
    'al servidor OPC, la clase clOPC
    objOPC.CrearServidorGrupo(txtServidorOPC.Text, txtNodoOPC.Text, txtGrupoOPC.Text, Items, 1)
End Sub

Private Sub TerminarLectura()
    'Terminar la Conexión con el Servidor OPC
    If Not IsNothing(objOPC) Then
        objOPC.TerminarLectura()
        objOPC = Nothing
    End If
End Sub
```

El Método CrearServidorGrupo en la clase clOPC:

```
    Public Sub CrearServidorGrupo(ByVal Servidor As String, ByVal Nodo As String, _
        ByVal Grupo As String, ByVal Items As Object, ByVal CantidadItems As Integer)
        Dim i As Integer
        Dim a As Integer
        Dim b As Integer

        Try
            'Se configuran las propiedades del objeto
            Me.Servidor = Servidor
            Me.Nodo = Nodo
            Me.Grupo = Grupo

            'Creacion del objeto OPCServer
            'Este objeto es el mas alto de la jerarquia. Con el se inicia
            'el trabajo de conexion al servidor OPC
            objServidor = New OPCAutomation.OPCServerClass()
            'Conexion al Servidor OPC
            objServidor.Connect(Servidor, Nodo)
            'Creacion del objeto OPCGroups. Este representa una coleccion de cada
            'uno de los grupos de variables que se leeran
            'Se requiere usar Marshal para el manejo de tipos. Recuerde que
            'OPCAutomation es COM
            objGrupos = CType(Marshal.CreateWrapperOfType(objServidor.OPCGroups, _
                        GetType(OPCGroupsClass)), OPCGroupsClass)

            'Estado activo por defecto para los Grupos que se agreguen a la coleccion
            'de Grupos OPC
            objGrupos.DefaultGroupIsActive = True
            'Valor de banda muerta.
```

```
            objGrupos.DefaultGroupDeadband = 0

            'Adicion de un Grupo OPC a la coleccion de Grupos.
            'Un grupo representa un conjunto de variables. Por ejemplo
            'si voy a leer todas las variables de una maquina determinada
            'creo un Grupo para esas variables.
            objGrupo = objGrupos.Add(Grupo)
            'Estado del Grupo
            objGrupo.IsActive = True
            objGrupo.IsSubscribed = True
            'Tiempo del lectura en milisegundos. Es decir, cada 1 segundo
            'el Cliente OPC(este programa) le preguntará al Servidor OPC
            'si existen cambios en alguna de las variables del Grupo
            objGrupo.UpdateRate = 1000

            'Creacion de la Objeto que representa la Coleccion de Items o Variables
            'a leer
            objItems = objGrupo.OPCItems
            'Estado por defecto para las variables que se agrequen a la coleccion
            'de Items
            objItems.DefaultIsActive = False

            'Se crean arreglo de variables
            For i = 1 To CantidadItems
                objItem(i) = objItems.AddItem(Items(i - 1, 0), Items(i - 1, 1))
                objItem(i).IsActive = Items(i - 1, 2)
            Next i

        Catch ex As System.Exception
            Exit Sub
        End Try
    End Sub
```

El método anterior representa claramente la forma como la especificación OPC determina la instanciación de objetos y llamado de métodos para lograr la conexión con un Servidor OPC.

Una vez se llame al método anterior, el objeto que es instancia de clOPC estará preguntándole al Servidor OPC cada un segundo (o lo establecido en UpdateRate del objeto OPCGroup), si existen cambios en las variables mapeadas para un Grupo, en tal caso se ejecutará el evento DataChange de ese Grupo.

```
Sub objGrupo_DataChange(ByVal TransactionID As Integer, ByVal NumItems As Integer, _
ByRef ClientHandles As System.Array, ByRef ItemValues As System.Array, _
ByRef Qualities As System.Array, ByRef TimeStamps As System.Array) Handles objGrupo.DataChange
    Dim Valores(99, 3) As Object
    Dim i As Integer

    'Creación de un vector que condensa toda la información enviada por
    'el Servidor OPC
    For i = 1 To NumItems
        Valores(i - 1, 0) = ItemValues.GetValue(i)
        Valores(i - 1, 1) = ClientHandles.GetValue(i)
        Valores(i - 1, 2) = TimeStamps.GetValue(i)

        'Se obtiene un valor significativo de la calidad de la informacion
        If Qualities.GetValue(i) And &HC0 Then
            Valores(i - 1, 3) = "Buena"
        Else
            Valores(i - 1, 3) = "Deficiente"
        End If
    Next i

    'Se llama al metodo ActualizarLista del formulario que instancio a clOPC
    'para que este obtenga el vector condensado con la informacion de los
    'cambios de variables enviados por el Servidor OPC
    FormularioAplicar.ActualizarLista(Me.Servidor, Me.Nodo, Me.Grupo, NumItems, Valores)
End Sub
```

16. OPC le permite no solo leer datos de un dispositivo, también puede escribir en el. Digite un valor en "Nuevo Valor y presione enviar; compruebe que el valor en el simulador en la Dirección DM1, cambió.

La clase clOPC ofrece este método:

```
Public Sub EnviarValor(ByVal Servidor As String, ByVal Nodo As String, _
ByVal Grupo As String, ByVal Item As String, ByVal ValorNuevo As Double, ByRef ValorActual As Doub
    'Creacion del objeto Servidor
    objServidor = New OPCAutomation.OPCServerClass()
    'Conexion al Servidor OPC
    objServidor.Connect(Servidor.Trim, Nodo.Trim)

    'Marshal para manejo de tipos de COM
    objGrupos = CType(Marshal.CreateWrapperOfType(objServidor.OPCGroups, _
                GetType(OPCGroupsClass)), OPCGroupsClass)
    objGrupo = objGrupos.Add(Grupo.Trim)
    objItems = objGrupo.OPCItems

    objItem(0) = objItems.AddItem(Item.Trim, 0)
    'Leer el valor actual de la variable
    objItem(0).Read(1, ValorActual)
    'Escribir un nuevo valor para la variable
    objItem(0).Write(ValorNuevo)
    objServidor = Nothing
    objGrupos = Nothing
    objItems = Nothing
    objItem(0) = Nothing
End Sub
```

Conclusión

Las aplicaciones de automatización o captura de datos que requieren las empresas industriales para tener un mayor control de los procesos productivos en muchos casos se realiza con Sistemas de Desarrollo llamados SCADA que permiten crear programas particulares para este tipo de necesidades. Pero como pudieron ver es posible realizar cosas útiles e interesantes en herramientas de desarrollo de propósito general como es el caso de VBNet. Esto se consigue gracias a OPC, una adaptación de OLE (COM) para el campo industrial. Sólo quedamos a la espera que el estándar OPC se especifique en el .NET Framework para evitar la interoperabilidad y tener un aplicativo totalmente administrado.

Marlon Martínez (marlonma@centelsa.com.co) estudió Ingeniería de Sistemas en la Universidad Santiago de Cali y trabaja como Analista de Sistemas en una importante Empresa Industrial de su Región. Ha sido programador de Visual Basic desde la versión 3.0 pasando por todas las versiones hasta Visual Basic .NET. Las tareas de programación que más le satisfacen son aquellas que tienen que ver con la comunicación a dispositivos industriales. Practica varios deportes sin ser el mejor en alguno, el Voleibol es el que más frecuenta a pesar de que el Fisiatra se lo prohibió por problemas en la espalda (¡que terco!).