Monitoriza

Acimut

MANUAL DE USUARIO

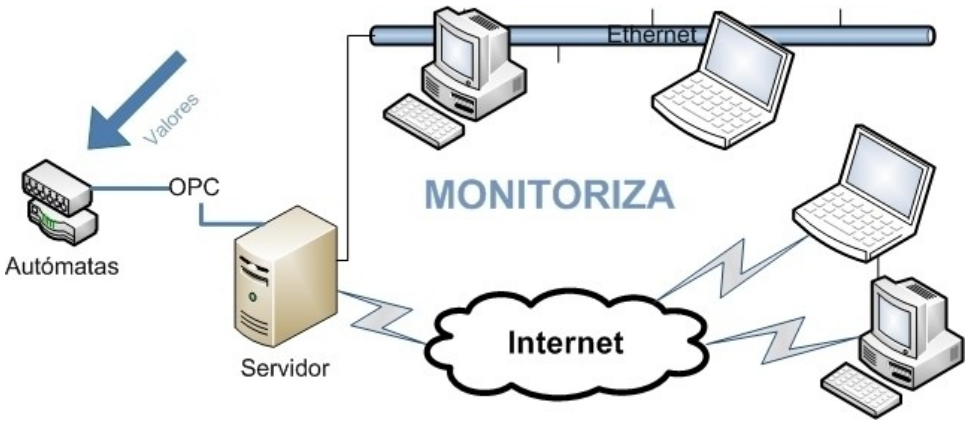
v. 3.2

http://monitoriza.acimut.com

2012

MONITORIZA

Supervisory Control and Data Acquisition (SCADA)



INDICE

MONITORIZA

¿QUÉ ES MONITORIZA?

Monitoriza es un sistema de monitorización y control (SCADA *Supervisory Control & Data Adquisition*) que cubre los requerimientos de cualquier proyecto, tanto básicos como avanzados.

Monitoriza nos permite crear soluciones para la captura de información en procesos industriales o de cualquier otro ámbito. Con esa información se retroalimenta el proceso y se emplea como ayuda en la toma de decisiones.

Consta de tres partes:

- Un editor de proyectos en el que se definen todos los elementos a tratar.
- Un servidor que ejecutará el proyecto y se ocupará de las comunicaciones con los procesos (adquisición de datos, establecimiento de parámetros del proceso, etc.)
- Un cliente que mostrará, de forma visual, la información de los procesos que se estén supervisando.

Monitoriza es un sistema flexible en cuanto a su configuración, ya que puede ejecutarse por varios usuarios simultáneamente. Puede funcionar sobre una infraestructura monopuesto o multipuesto, tanto en una red local como a través de puestos remotos conectados a través de internet.

Entre las características principales de Monitoriza se pueden destacar:

- Instalación sencilla e inmediata del producto.
- Fácil configuración, incluso cuando se trata de una instalación con puestos remotos (WAN) ya que las comunicaciones entre los equipos cliente y el servidor se basan en los estándares de internet (protocolo HTTP).
- Incluye comunicaciones nativas ModBUS, Ethernet S7 para S7-300 y conectividad OPC.
- No precisa programación para la creación de proyectos completamente funcionales, basta “pinchar y arrastrar” los objetos SCADA sobre la superficie de los formularios y establecer las propiedades correspondientes para obtener una solución operativa.
- Si se requiere una funcionalidad avanzada que no esté contemplada en los objetos SCADA definidos en Monitoriza no hay problema ya que Monitoriza es extensible mediante programación en C# o VB.Net. También es posible la utilización de librerías de terceros desarrolladas para el .NET Framework de Windows.
- La creación de la interfaz gráfica de usuarios está basada en la tecnología de Windows Forms Designer de Microsoft© lo que facilita enormemente el diseño.
- A nivel de proyecto podemos definir los usuarios y los permisos asignados a cada uno ellos. Por ejemplo si solo se tiene permiso de lectura en un determinado formulario o si se tiene acceso total a este.
- Definición inmediata de alarmas.
- Control efectivo de operaciones.
- Incremento instantáneo de información.
- Fácil seguimiento de variables.
- Datos en formatos accesibles. Monitoriza permite almacenar las variables que se monitorizan en bases de datos estándar del mercado (Microsoft© SQL Server™, Microsoft© Access™, Oracle®, etc.)
- Inversión mínima amortizable inmediatamente.
- Definición de recetas mediante plantillas, control de usuario para utilización de recetas.
- Funciones Batch para la carga de recetas por evento.

- El servidor de Monitoriza ofrece las variables definidas mediante servicios OPC, así aplicaciones externas pueden conectarse a Monitoriza y acceder a las variables para su uso. (Este servicio solo está disponible en la versión Profesional y superiores de Monitoriza).

INSTALACIÓN

Monitoriza tiene una instalación muy sencilla. Consta de dos ficheros, *setup.exe* e *Instalación Monitoriza.msi*. Haciendo un doble clic sobre setup.exe se inicia la instalación.

Monitoriza requiere para instalarse el .Net Framework™ 3.5 SP1 de Windows™ en el caso que no se encuentre instalado nos aparecerá el siguiente dialogo

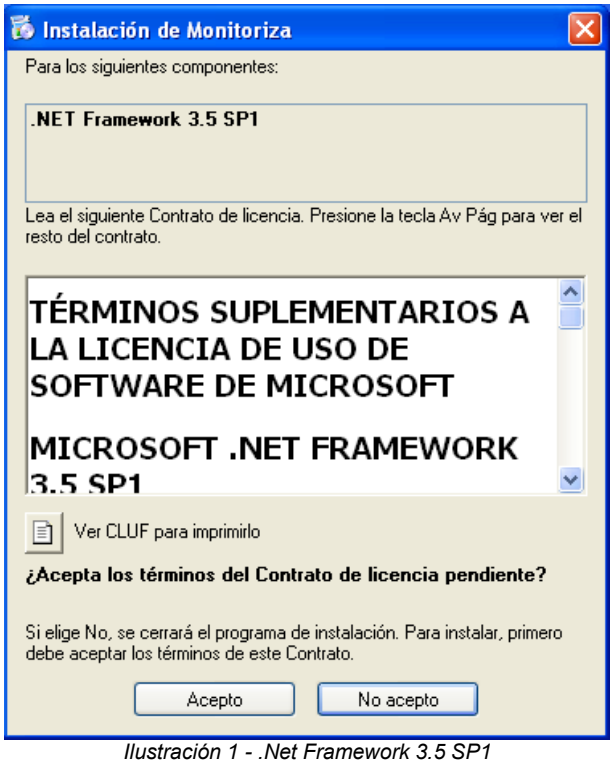


Ilustración 1 - .Net Framework 3.5 SP1

en el que se nos pedirá si aceptamos el Contrato de licencia del .Net Framework de Microsoft.

A continuación procede a descargar desde la web de Microsoft el paquete correspondiente e instalarlo

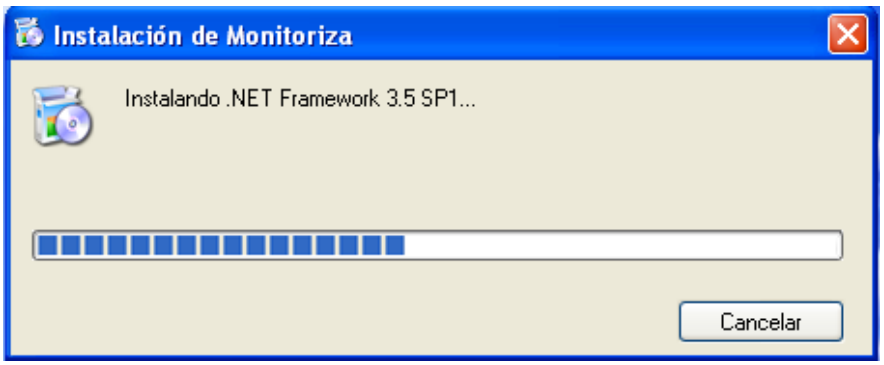


Ilustración 2 – Descarga del .Net Framework 3.5 SP1

Una vez instalado se inicia la instalación de Acimut Monitoriza propiamente dicha

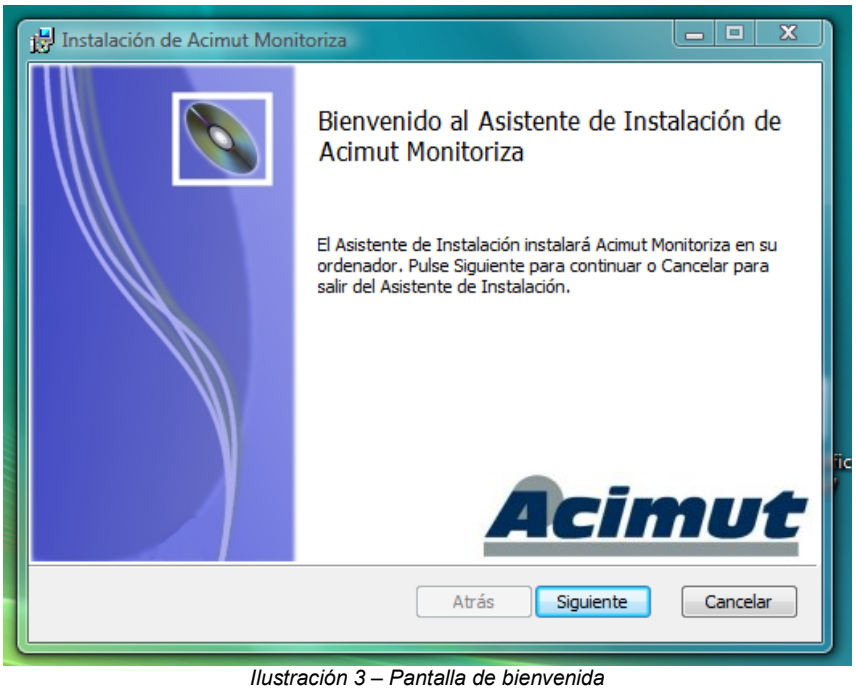


Ilustración 3 – Pantalla de bienvenida

Pulsando sobre el botón *Siguiente* nos aparece la pantalla de instalación personalizada, en la que seleccionamos los elementos del sistema Acimut Monitoriza que se desea instalar. Estos elementos son el **Editor** que nos permite crear y modificar nuestros proyectos Scada, el **Cliente** mediante el cual establecemos el entorno de ejecución de los proyectos Scada y el **Servidor de Comunicaciones** a través del cual establecemos las comunicaciones tanto con los servidores OPC, como con las bases de datos y los autómatas.

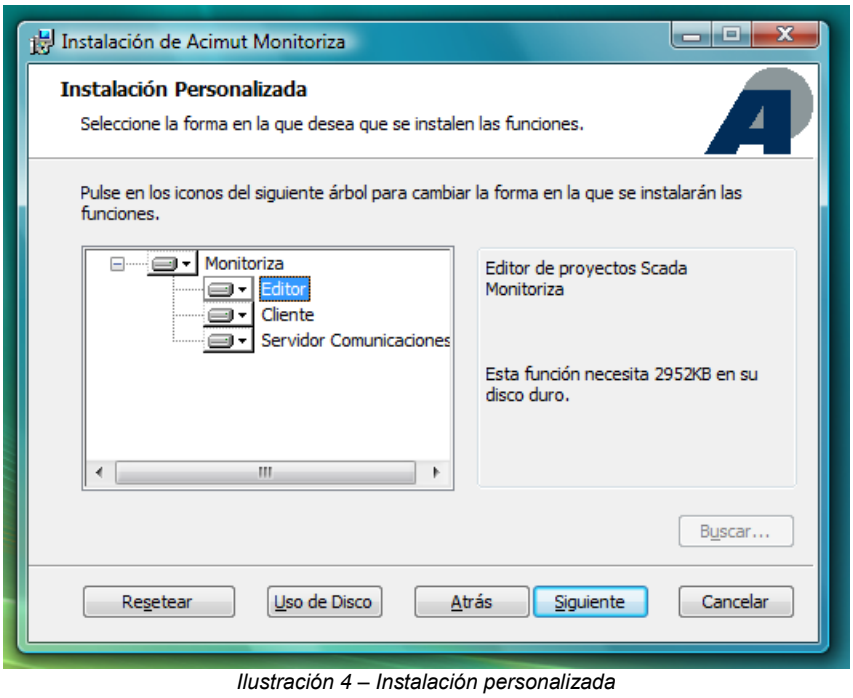


Ilustración 4 – Instalación personalizada

En la siguiente pantalla pulsaremos sobre el botón Instalar para iniciar el proceso de instalación en sí.

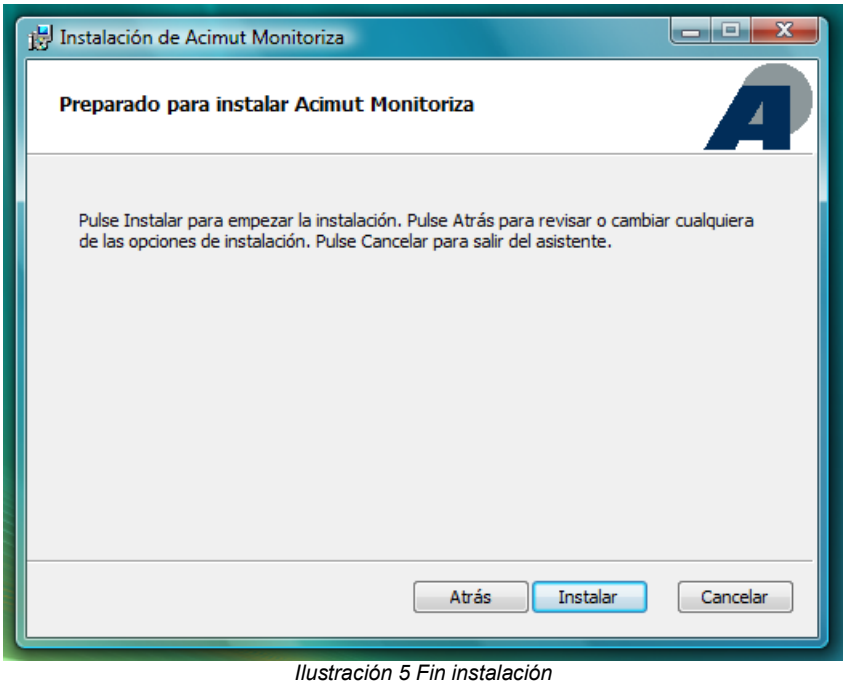


Ilustración 5 Fin instalación

Acimut Monitoriza es compatible con los sistemas operativos Windows™ 7, Vista, XP, 2003 y 2008, siendo el único requerimiento que esté instalado el .Net Framework 3.5 SP1 de Microsoft.

EDITOR

El Editor de Acimut Monitoriza es uno de los tres componentes principales del sistema, con el vamos a crear, diseñar y modificar nuestros proyectos de scada que luego se ejecutaran a través del Servidor de Comunicaciones y del Cliente Scada.

Al crear o modificar un proyecto scada mediante el Editor podremos definir variables y alarmas, crear formularios para mostrar de forma gráfica los valores de las variables, guardar en base de datos los valores de las variables, mantener un histórico de alarmas, mostrar gráficas de los valores de variables almacenados, escribir variables sobre un autómata (u otros dispositivos) y gestionar los usuarios que podrán acceder a los recursos del proyecto.

La interfaz de usuario del Editor es la que puede verse en las dos ilustraciones siguientes:

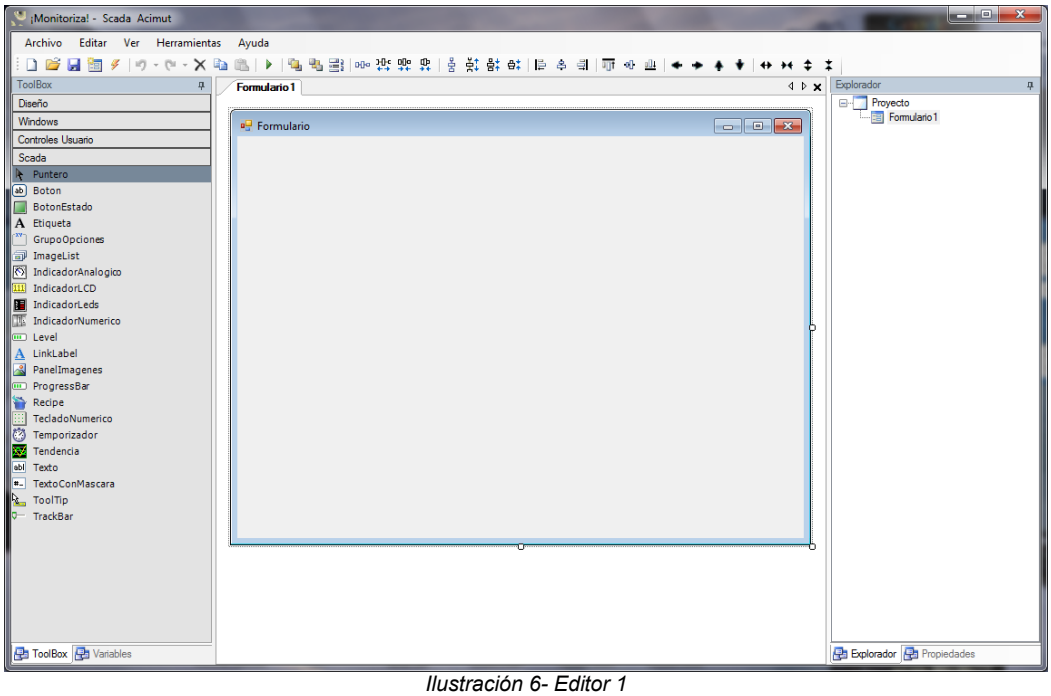


Ilustración 6- Editor 1

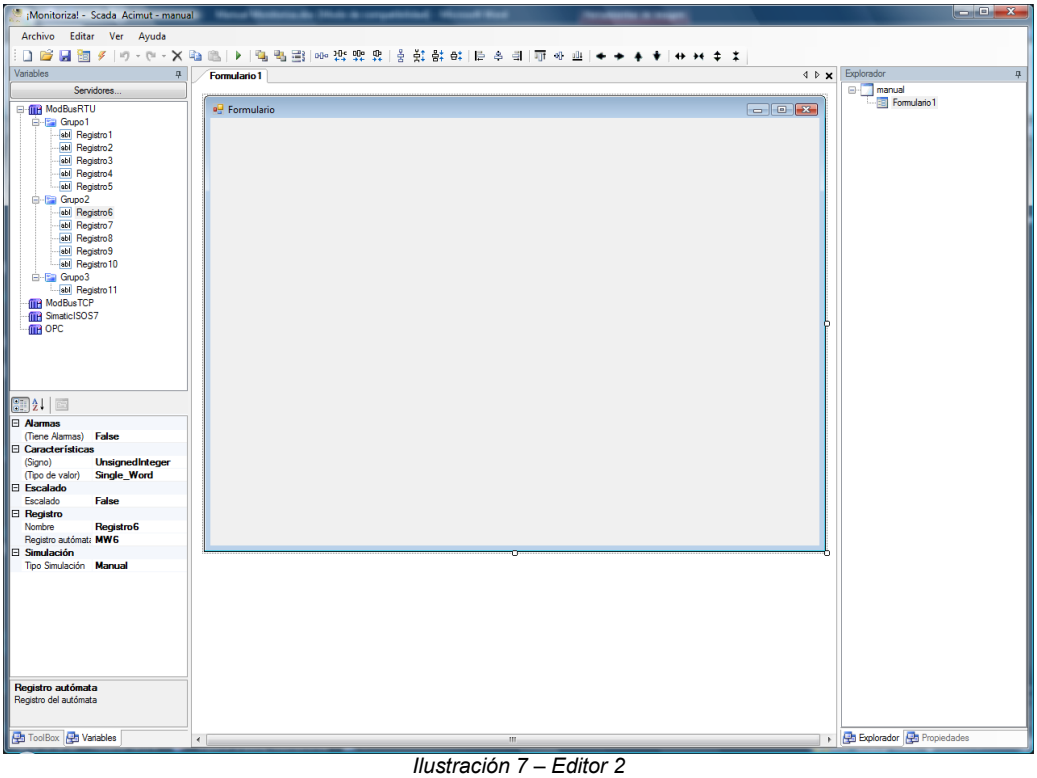


Ilustración 7 – Editor 2

Veamos con un poco de detalle cada una de las partes de la interfaz de usuario.
En la parte superior se encuentra la barra de menús y herramientas en la que se encuentran elementos habituales como las operaciones de abrir o guardar un proyecto, operaciones de edición (copiar, cortar, pegar…) y operaciones de formateo y alineación.

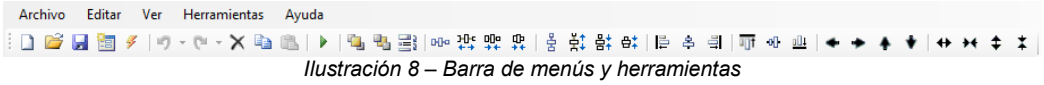


Ilustración 8 – Barra de menús y herramientas

En los laterales nos encontramos las ventanas del explorador del proyecto, el ToolBox, la de características de los servidores y variables y la ventana de propiedades.

La ventana del explorador del proyecto nos muestra la relación de formularios de que consta nuestro proyecto y mediante un doble clic podemos visualizar el formulario en cuestión, también podremos a través del menú contextual realizar ciertas operaciones sobre los formularios tal y como se muestra en la ilustración siguiente.

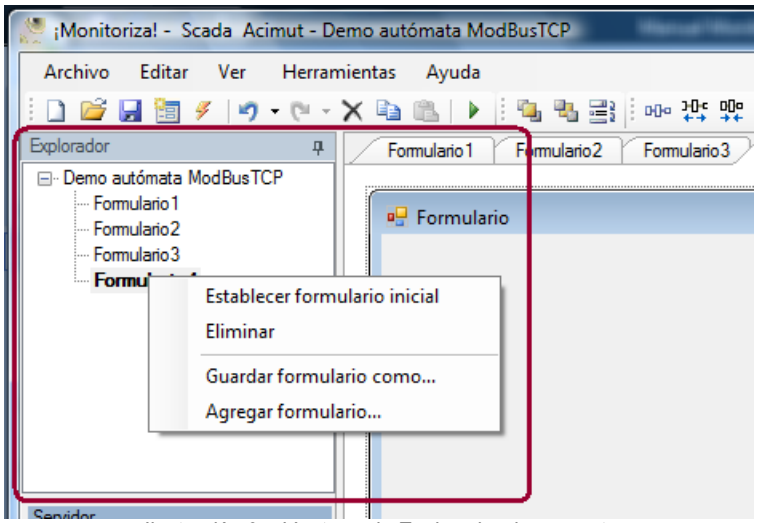


Ilustración 9 – Ventana de Explorador de proyecto

La ventana del ToolBox nos permite agregar componentes al formulario, mediante estos componentes iremos diseñando los formularios y dándoles la funcionalidad deseada, bien a través de componentes que nos permiten ver o modificar las variables o parámetros que deseamos monitorizar (solapa Scada), bien a través de componentes que sirven para controlar el flujo de la aplicación y definir su diseño o apariencia (solapas Diseño y Windows).

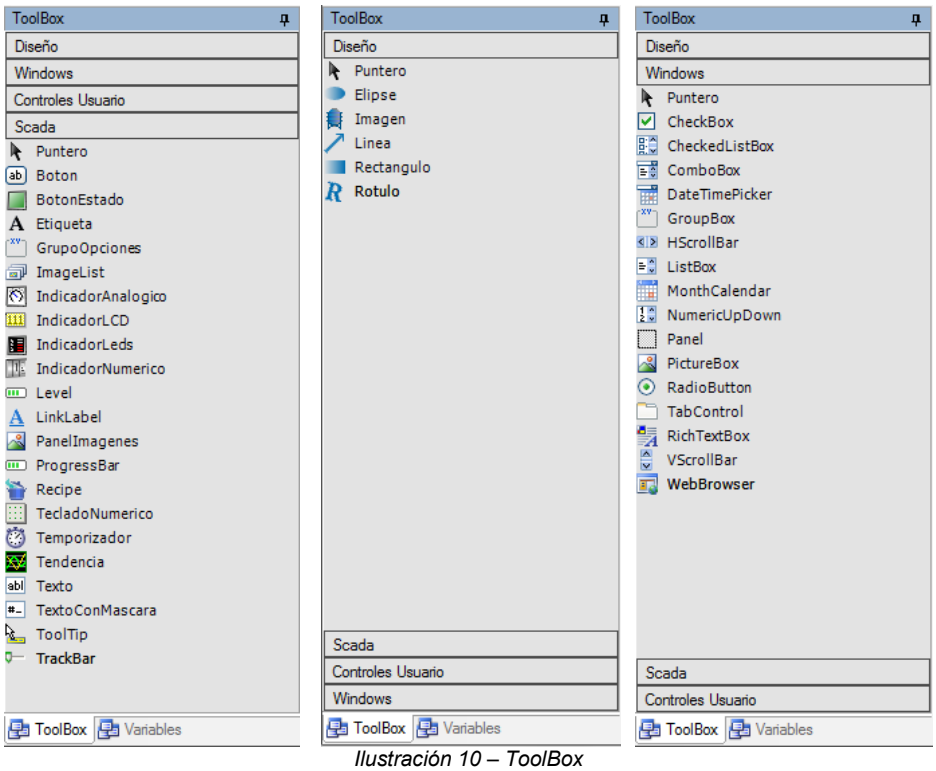


Ilustración 10 – ToolBox

La ventana de características de variables y servidores nos permite definir los parámetros correspondientes a cada uno de los servidores de comunicación que

definamos en el proyecto, como por ejemplo la dirección IP del autómata con el que queremos comunicar y las características de los grupos y variables que definamos en cada servidor.

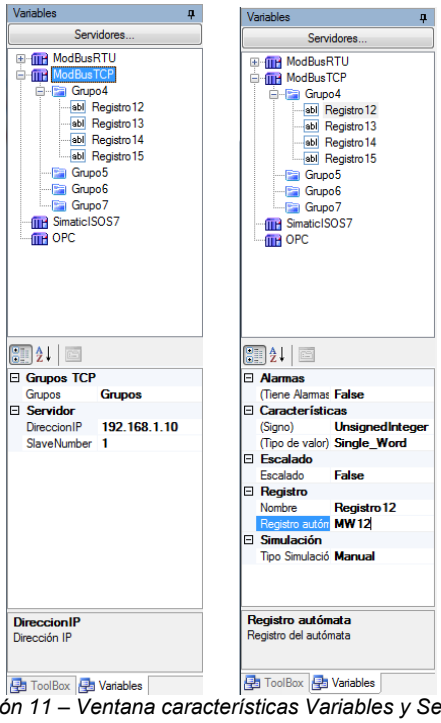


Ilustración 11 – Ventana características Variables y Servidores

En la ventana de propiedades definiremos las características de cada uno de los componentes que agreguemos a los formularios, así por ejemplo vemos en la ilustración siguiente que al seleccionar el componente IndicadorNumerico, en la ventana de propiedades se nos muestran las características o propiedades de ese componente en particular

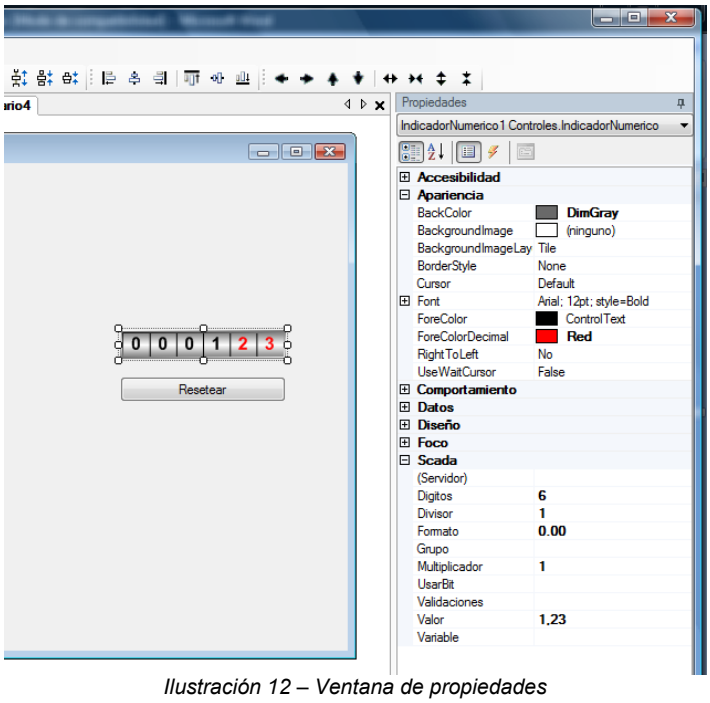


Ilustración 12 – Ventana de propiedades

Hay que tener en cuenta que si bien al abrir el Editor las distintas ventanas se encuentran dispuestas en los laterales, esta disposición es configurable y simplemente arrastrando sobre el título de la ventana podemos posicionarlas donde nos sea más cómodo para nuestro trabajo.

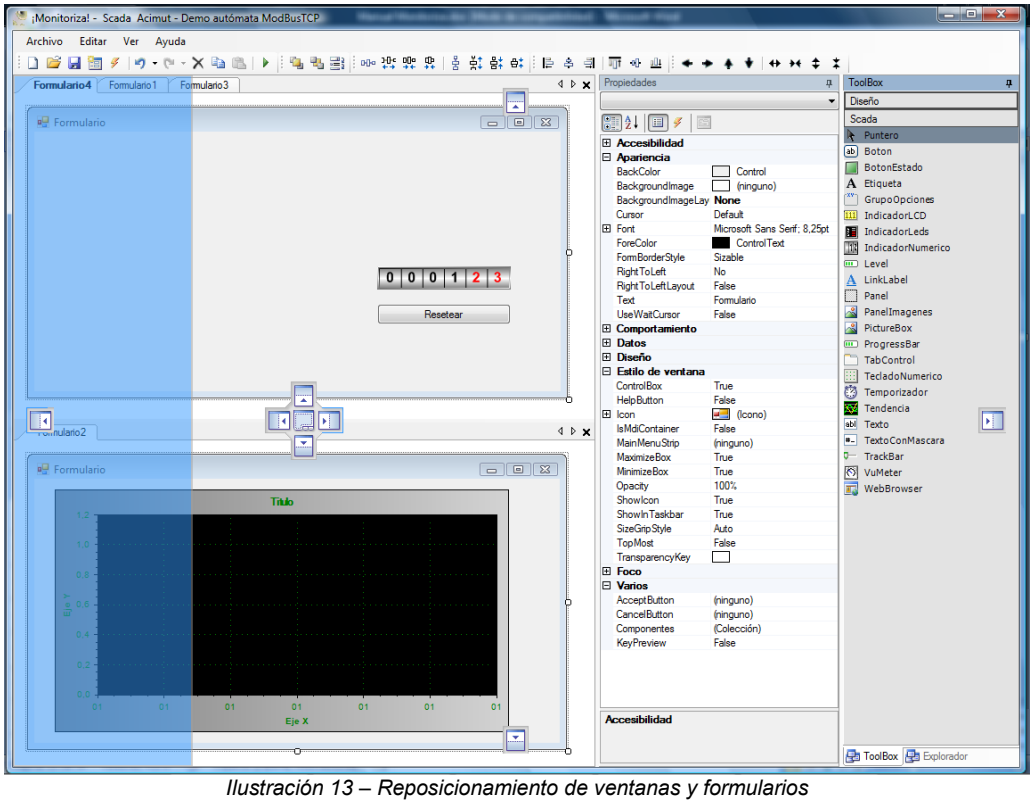


Ilustración 13 – Reposicionamiento de ventanas y formularios

Demostración: http://www.acimut.com/monitoriza/videos/editor1/default.html

Igualmente las ventanas de Variables, Propiedades, ToolBox y Explorador podemos configurarlas para que se oculten automáticamente cuando no tienen el foco con ello logramos que la superficie de trabajo sea más grande, simplemente deberemos pulsar sobre el correspondiente botón del título de la ventana (marcado en rojo en la ilustración siguiente) y la ventana se ocultará cuando no tenga el foco. Para volver a visualizarla deberemos mover el ratón sobre la solapa que queda visible representando la ventana y si queremos que deje de ocultarse volveremos a pulsar sobre el mismo botón para fijar su posición La configuración de posicionamiento de cada ventana se almacena con cada proyecto de forma que al abrir de nuevo un determinado proyecto las ventanas se posicionan tal y como estaban al guardar el proyecto.

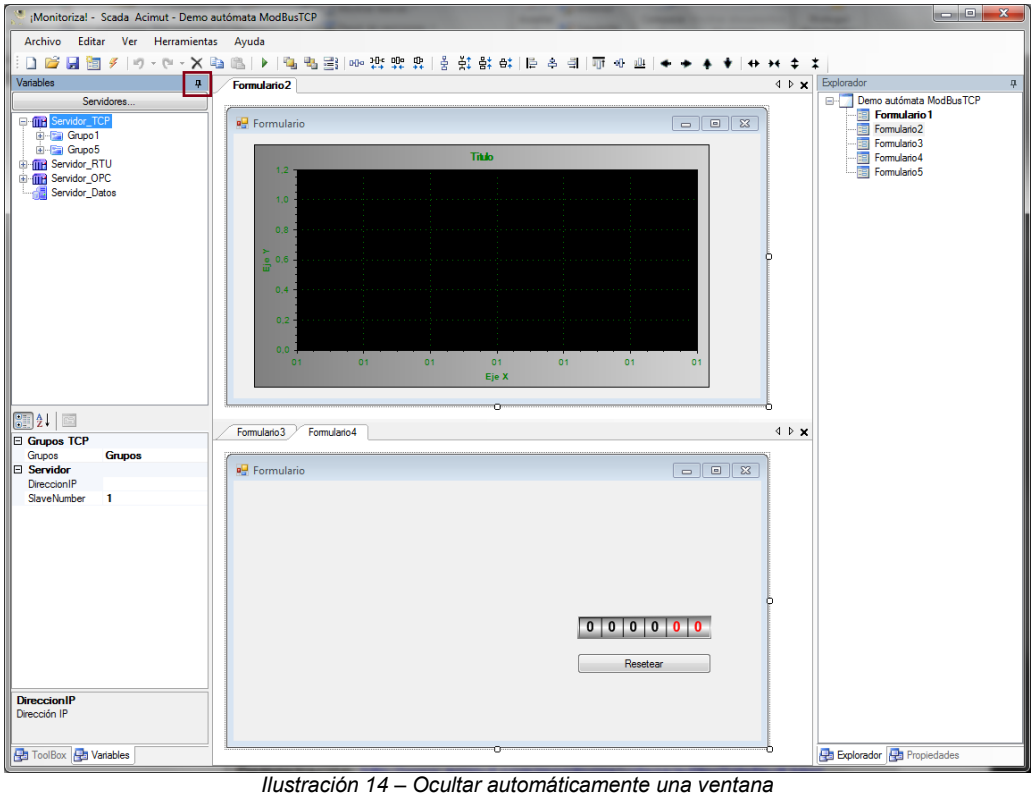
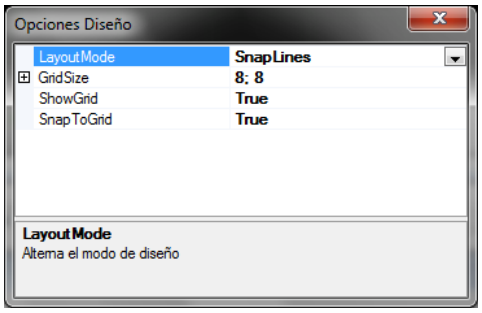


Ilustración 14 – Ocultar automáticamente una ventana

Demostración: http://www.acimut.com/monitoriza/videos/editor2/default.html

En el menú Herramientas disponemos de opciones que nos ayudarán en las tareas de diseño de los formularios.

En este menú disponemos del ítem Opciones:



Las distintas opciones de diseño son:

Layout Mode: Podemos elegir entre *SnapLines* (por defecto), que usa las características avanzadas de posicionamiento de las últimas aplicaciones de Microsoft; o podemos usar *Grid* para usar la rejilla clásica de posicionamiento, pudiendo elegir en este caso el tamaño de la misma desde la propiedad **GridSize**.

También podemos elegir las opciones de la rejilla mediante las propiedades:
ShowGrid: Para mostrar o no la rejilla en el formulario.
SnapToGrid: Si deseamos que los controles se ajusten a la rejilla o no.

CREAR UN PROYECTO

COMO EMPEZAR

Para crear un proyecto usaremos el Editor de Monitoriza, como mínimo deberemos definir un servidor de comunicaciones con el autómata, las variables del autómata a las que queramos acceder, que las uniremos en grupos a los que daremos nombre, y por lo menos un formulario en el que diseñaremos como será la interfaz de usuario del proyecto.

Adicionalmente podemos definir alarmas que se dispararan cuando se cumplan las condiciones que definamos y usuarios y permisos para poder establecer a que formularios puede acceder cada uno de los usuarios definidos.

En la definición de servidores también podemos establecer los servidores de base de datos en los que almacenaremos permanentemente los valores de las variables que estemos monitorizando.

Veamos cómo hacer cada una de las tareas. Para empezar en el Editor de Monitoriza tendremos que decir que queremos iniciar un nuevo proyecto, para ello pulsaremos sobre la opción ***Nuevo proyecto*** del menú ***Archivo***.

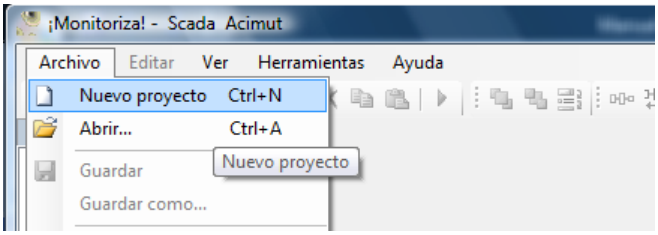


Ilustración 15 – Menú Nuevo proyecto

O bien en la barra de herramientas sobre el botón ***Nuevo Proyecto***.

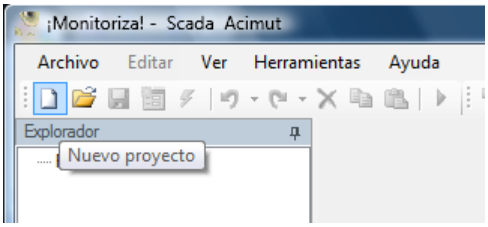


Ilustración 16 Barra de herramientas Nuevo proyecto

A continuación deberemos definir los servidores de comunicaciones con los autómatas y bases de datos, para ello pulsaremos sobre la opción ***Servidores*** del menú ***Ver***.

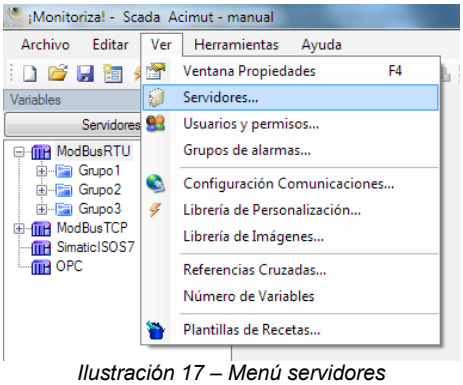


Ilustración 17 – Menú servidores

O bien en el botón ***Servidores*** del la ventana Variables

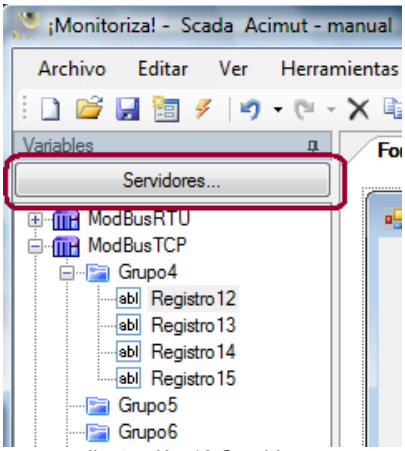


Ilustración 18 Servidores

Y nos aparecerá el dialogo para la definición de los servidores.

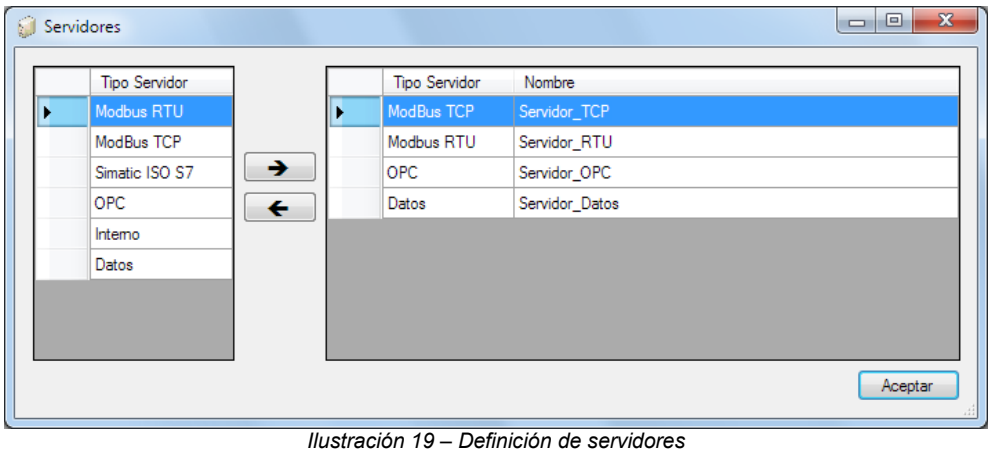


Ilustración 19 – Definición de servidores

Demostración: http://www.acimut.com/monitoriza/videos/definir_servidores/default.html

Un nuevo proyecto necesita al menos un *servidor*, aquí tenemos la lista de los diferentes tipos de servidor.

Monitoriza tiene servidores nativos de comunicaciones con autómatas (ModBUS RTU, ModBUS TCP y Simatic ISO S7) y de terceros como OPC, esto le permite conectar con multitud de dispositivos.

Si queremos almacenar un histórico de valores de las variables y/o un histórico de las alarmas que se produzcan en el sistema, deberemos definir uno o varios servidores de base de datos en los que estableceremos cual es la base de datos a utilizar. La definición de los servidores de datos se verá en detalle en la sección Guardar en Base de Datos

Otra posibilidad que tenemos es definir un servidor de datos internos, mediante este servidor podemos especificar un conjunto de variables que no dependen de las variables de los autómatas sino que nos puede servir para almacenar valores por ejemplo de parámetros de configuración o valores de cálculo.

Cada uno de los servidores de comunicaciones con el autómata tendrá diferentes propiedades como veremos a continuación.

ModBUS RTU

Las diferentes propiedades de un servidor de tipo ModBUS RTU son las que se muestran a continuación.

Estas propiedades establecen los parámetros de comunicación serie así como el número de esclavo al que se conectará en un entorno ModBUS. Hay que hacer notar que Monitoriza establece comunicaciones ModBUS como maestro.

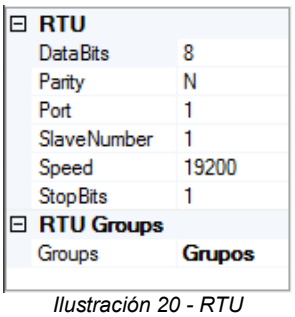


Ilustración 20 - RTU

ModBUS TCP

Las propiedades de un servidor de tipo ModBUS TCP son las siguientes:

Son la dirección IP del dispositivo ModBUS que en un entorno Ethernet es necesaria y el número de esclavo al que se conectará en un entorno ModBUS. Hay que hacer notar que Monitoriza establece comunicaciones ModBUS como maestro. El utilizar número de esclavo permite su conexión a pasarelas.

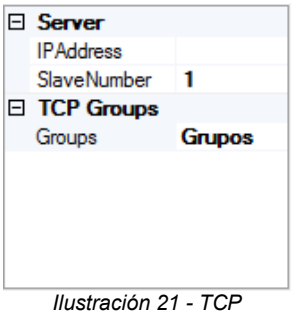


Ilustración 21 - TCP

OPC

OPC es el acrónimo de OLE for Process Control, se trata de un estándar definido por Microsoft que tiene como objetivo unificar los métodos de comunicación con todo tipo de dispositivos, dada su buena aceptación por diversos fabricantes, se puede decir que existe un servidor OPC apropiado para cualquier dispositivo que se encuentre en el mercado.

Las propiedades que nos permitirán conectarnos a un servidor OPC son las siguientes

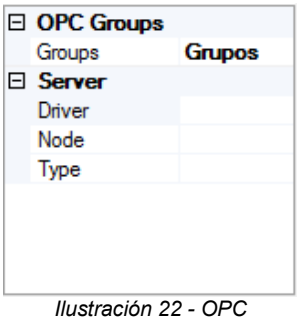


Ilustración 22 - OPC

Comenzaremos describiendo la última de estas propiedades, ***Type***, se trata del nombre que el fabricante ha dado a su servidor OPC, un desplegable nos permitirá elegir el servidor OPC entre los instalados en nuestra máquina o en el ordenador de nuestra red que especifiquemos en ***Node*** (siempre y cuando se tengan los permisos adecuados, los permisos para utilizar un servidor OPC en red dependen del sistema operativo utilizado, hay abundante documentación sobre este tema que queda fuera de las pretensiones de este manual). La propiedad ***Driver*** es específica del fabricante, habrá que consultar la documentación de cada servidor OPC, es un prefijo que suele definir el medio por el que comunicaremos con el autómata así como su dirección y características.

En la siguiente tabla podemos ver algunos ejemplos.

Fabricante	***Tipo***	***Driver***	Variable
Siemens S7-200	S7200.OPCServer	192.168.100.120:1200:1001,	VW1000,INT,RW
Telemecanique	Schneider-Aut.OFS.2	UNTLW01:0.254.0!	%MW2000
		MBS03:1/T!	%MW2000
		XIP01:192.168.1.2.10.2!	%MW2000

Simatic ISO S7

Las propiedades de un servidor de tipo Simatic ISO S7 son las siguientes:

Son la dirección IP del dispositivo Simatic, el número de Rack en que se encuentra la CPU (suele ser el 0), y el slot dentro del Rack en el que se encuentra la CPU (suele ser el 2).

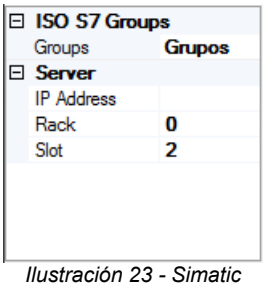
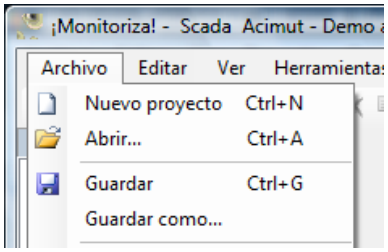


Ilustración 23 - Simatic

GUARDAR O ABRIR UN PROYECTO

En el menú ***Archivo*** aparecen las típicas opciones de ***Abrir…***, ***Guardar*** y ***Guardar como…*** que nos permiten abrir los proyectos previamente guardados y guardar los cambios que vayamos haciendo a nuestros proyectos. Estas operaciones también pueden hacerse desde la barra de herramientas.



CREAR VARIABLES

Las variables en Acimut Monitoriza son el elemento básico de información, mediante ellas establecemos el vínculo de unión entre los tags o registros del autómata y los

distintos elementos de nuestro proyecto, posibilitando por tanto la visualización y modificación de los diferentes elementos que estemos monitorizando.

Las variables son en definitiva lo que vamos a estar midiendo o controlando, son la temperatura, la presión, el voltaje, el estado abierto o cerrado de cualquiera de los dispositivos de nuestro proceso.

Cada variable de Acimut Monitoriza viene definida por tres elementos:

- **El servidor de comunicaciones**, que hace referencia al autómata que gestiona y controla el dispositivo o dispositivos sobre los que queremos actuar o controlar
- **El grupo**, que es una agrupación de variables en función de las características de estas, por ejemplo, agruparemos las variables que solo nos interesa leer en un grupo y las que queramos leer y escribir en otro, o si tenemos un grupo de variables que queremos leer con una cadencia distinta a otras variables las agruparemos en distintos grupos. También es importante tener en cuenta los protocolos de comunicaciones, así pues, cuando queremos leer un conjunto de registros del autómata si estos registros ocupan posiciones contiguas de memoria, la lectura, ya sea por un servidor OPC o por un servidor ModBUS, es más eficiente con lo cual si ponemos los registros contiguos en el mismo grupo mejoraremos las prestaciones de nuestro SCADA.
- **Nombre** es el nombre simbólico que le damos al registro del autómata.

Por tanto para crear variables una vez tenemos creados los servidores de comunicaciones tal y como vimos en el capitulo anterior, tenemos que crear los grupos a los que van a pertenecer las variables

Para ello en la ventana de servidores pulsaremos sobre los tres puntos de la propiedad ***Grupos***

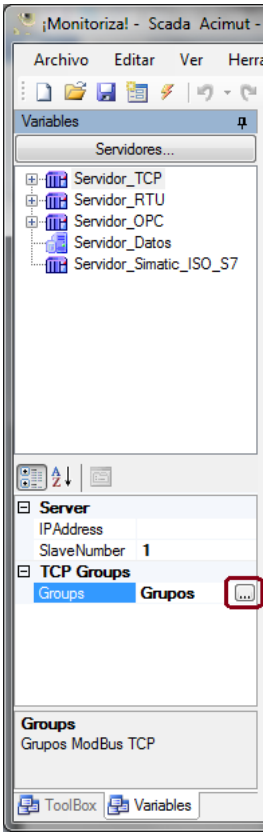


Ilustración 24 – Crear/Editar grupos

En función del tipo de servidor las características del grupo son similares pero con alguna diferencia, por tanto vamos a ver cada una con detalle.

Grupos ModBUS TCP y ModBUS RTU

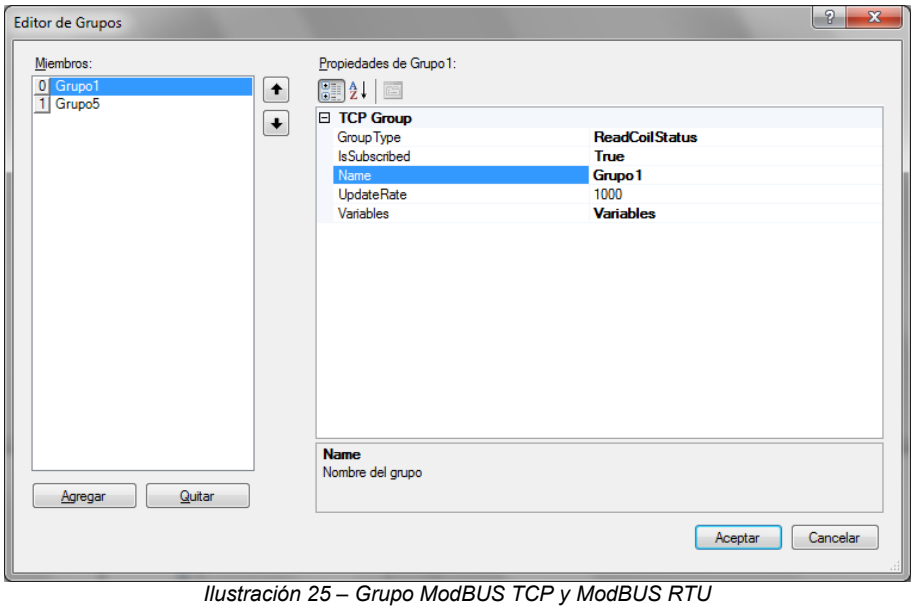


Ilustración 25 – Grupo ModBUS TCP y ModBUS RTU

Las propiedades que definen un grupo de un servidor ModBUS TCP son las siguientes:

- **GroupType:** Tipo de Grupo, nos indica si las variables son de tipo registro o de tipo marca. Seleccionaremos *ReadHoldingRegisters* cuando las variables que se lean/escriban son de tipo registro (una palabra, 2 bytes), y *ReadCoilStatus* cuando se lean variables de tipo marca (un bit).
- **IsSubscribed:** Indica que este grupo de variables se leerá continuamente. Si definimos un grupo con variables que tan solo se van a escribir y que no necesitaremos leer podremos dejar a False esta propiedad.
- **Name:** Denomina al grupo, es conveniente que el nombre sea lo más descriptivo posible por razones de organización.
- **UpdateRate:** En caso de que IsSuscribed esté a True esta propiedad indicará cada cuántos milisegundos se leerá el grupo. Es conveniente adecuar este valor al tipo de comunicación de que se disponga.
- **Variables:** Es la propiedad a través de la cual podremos dar de alta las variables a utilizar. Esta propiedad la veremos con más detalle más adelante.

Grupo OPC

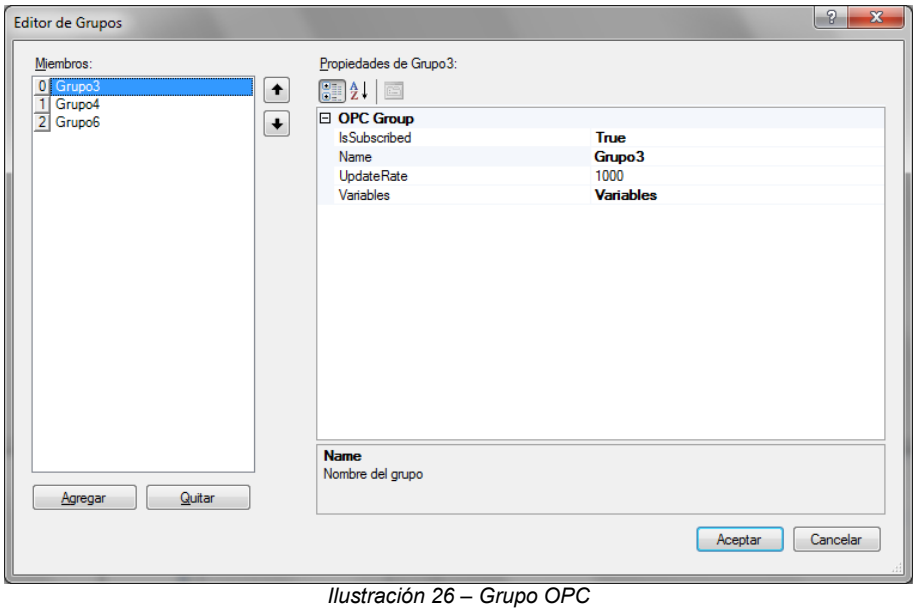


Ilustración 26 – Grupo OPC

Las propiedades que definen un grupo de un servidor OPC son las siguientes:

- **IsSubscribed:** Indica que este grupo de variables se leerá continuamente. Si definimos un grupo con variables que tan solo se van a escribir y que no necesitaremos leer podremos dejar a False esta propiedad.
- **Name:** Denomina al grupo, es conveniente que el nombre sea lo más descriptivo posible por razones de organización.
- **UpdateRate:** En caso de que IsSuscribed esté a True esta propiedad indicará cada cuántos milisegundos se leerá el grupo. Es conveniente adecuar este valor al tipo de comunicación de que se disponga.
- **Variables:** Es la propiedad a través de la cual podremos dar de alta las variables a utilizar.

Una vez tenemos definido el grupo ya podemos definir las variables o registros, para ello en la misma ventana como hemos visto, tenemos la propiedad ***Variables*** en la que si pulsamos sobre el botón de tres puntos de la propiedad ***Variables*** nos aparece la ventana de definición de variables o registros.

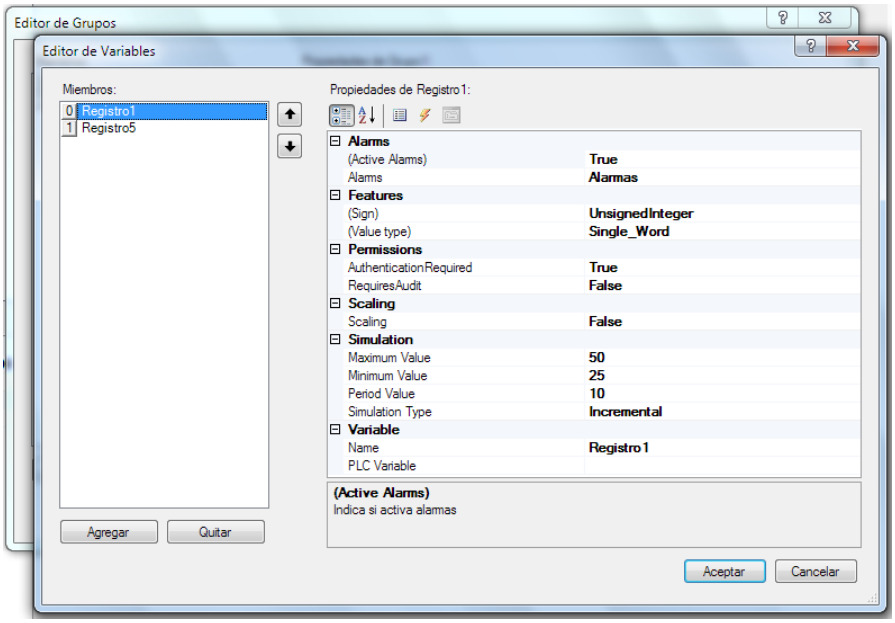
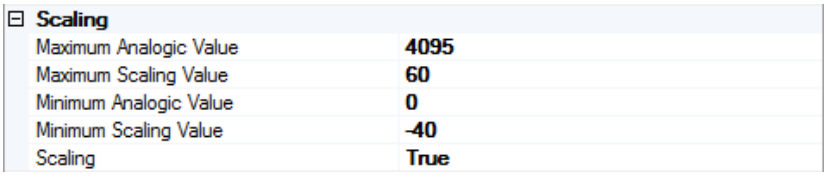


Ilustración 27 – Crear/Editar variables

Que tiene las siguientes propiedades:

- **Active Alarms:** Indica si deben activarse alarmas para esa variable. Se verá con más detalle en el capitulo siguiente.
- **Value type:** Define el tamaño de la variable. Las opciones posibles son: *Single_Word* (2 bytes) y *Double_Word* (4 bytes)
- **Sign:** Define si las variables se consideran con signo o no. Las opciones posibles son: *UnsignedInteger* – Entero sin signo y SignedInteger – Entero con signo
- **Name:** Nombre simbólico que le damos a la variable. Debe estar formado por caracteres alfanuméricos y no contener ni blancos ni símbolos.
- **PLC Variable:** Identifica el registro del autómata al que se quiere hacer referencia y debe seguir las normas y convenciones del autómata en cuestión.
- **Permissions:** Las propiedades **AuthenticationRequired** y **RequiresAudit** establecen si se deberá autenticar un usuario para poder cambiar un valor y si se guardará un registro de los cambios efectuados respectivamente. Estas propiedades se verán con más detalle en el capítulo de Usuarios y Permisos.
- **Scaling:** Indica si el valor de este registro se escalará, es decir podremos indicar los valores Máximo y Mínimo para los valores Analógico y Escalado, si por ejemplo tenemos un registro que nos va a dar el valor de entrada de una sonda, y tenemos por una parte una señal de 12 bits y por otra una sonda 4-20 mA. que mide temperaturas entre -40 ºC y 60 ºC configuraremos estas propiedades así

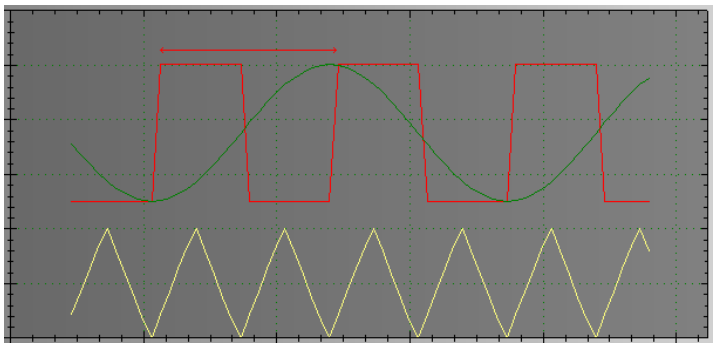


De esta manera tenemos ya escalado el valor, hay que tener en cuenta que el valor escalado es el que se utilizará en las alarmas, recetas, etc. Sin embargo los valores en simulación no se escalan. Se utiliza un escalado lineal.

- **Simulation Type:** Cuando en diseño se 'ejecuta' un proyecto, esta opción nos permite cambiar de forma automática el valor de las variables. Si se escoge cualquier tipo de simulación que no sea Manual nos aparecerán las siguientes tres propiedades.

- **Maximum Value:** Es el máximo valor que el sistema de simulación dará a la variable.
- **Minimum Value:** Es el mínimo valor que el sistema de simulación dará a la variable.
- **Period Value:** Es el tiempo, en segundos, de cada ciclo o estado, influirá en la velocidad a la que cambiará el valor de la variable.

Existen simulaciones para valores Aleatorios, Incrementales, Decrementales, de Onda Cuadrada, Onda Sinusoidal y Onda Triangular. Vemos en la siguiente imagen un ejemplo de ondas generadas con Monitoriza y el periodo señalado para la Onda Cuadrada.



Las propiedades **Value type** y **Sign** solo nos aparecerán cuando el grupo sea ModBUS TCP o ModBUS RTU y no hallamos definido el tipo del grupo como *ReadCoilStatus* ya que este tipo implica una marca de un bit.

Demostración: http://www.acimut.com/monitoriza/videos/crear_variables/default.html

Una vez tenemos creados los grupos y variables también podemos editarlos seleccionando el correspondiente grupo o variable en el árbol de variables de la parte superior de la ventana variables y editando su propiedades en el panel inferior de la ventana

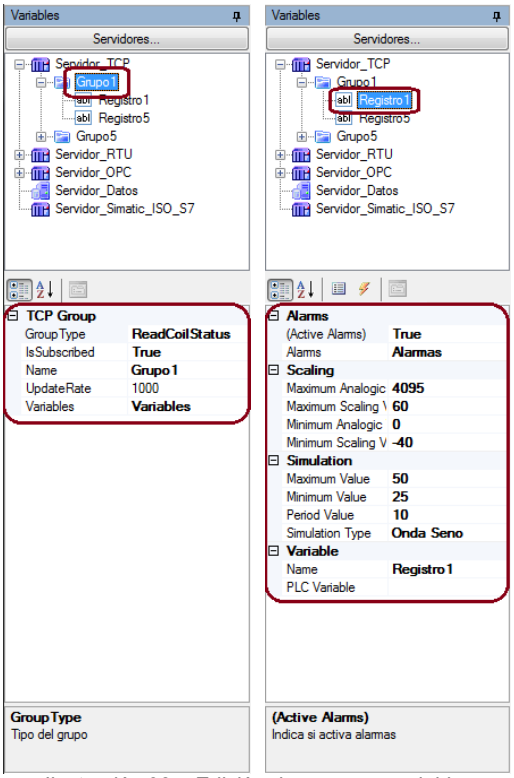


Ilustración 28 – Edición de grupos y variables

Evento Variables

Una vez tenemos definida una variable podemos definir alarmas asociadas a esa variable como se verá en el capítulo siguiente de Alarmas o bien controlar el evento *ValueChanged* de la variable.

Mediante ambos métodos tenemos la posibilidad de realizar acciones cuando cambia el valor de una variable.

Usaremos las alarmas cuando queremos que nos avise de que una determinada condición del valor de la variable y usaremos el evento *ValueChanged* cuando queremos realizar alguna acción simplemente porque el valor ha cambiado.

Para definir el evento ValueChanged debemos pulsar sobre el botón Eventos de la ventana de propiedades de la Variable (marcado en rojo en la figura).

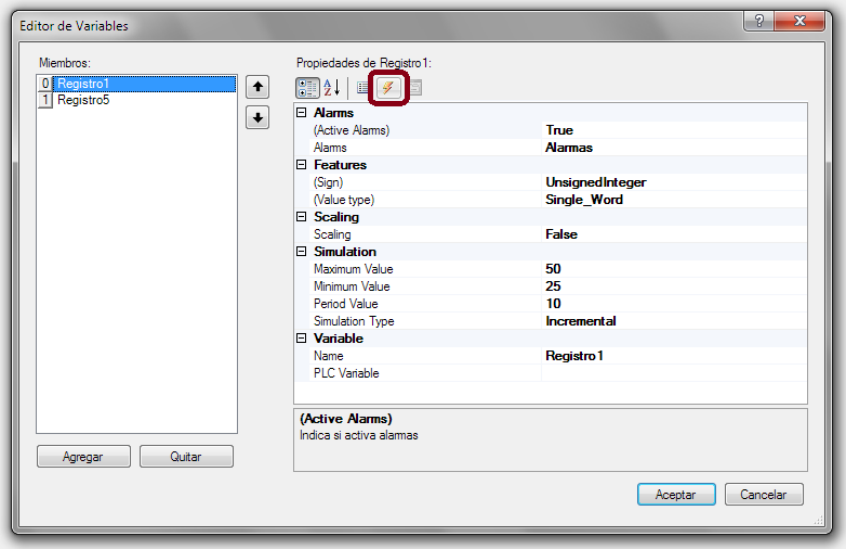


Ilustración 29 – Activar ventana eventos

Una vez en la ventana de eventos pulsaremos sobre el botón de tres puntos

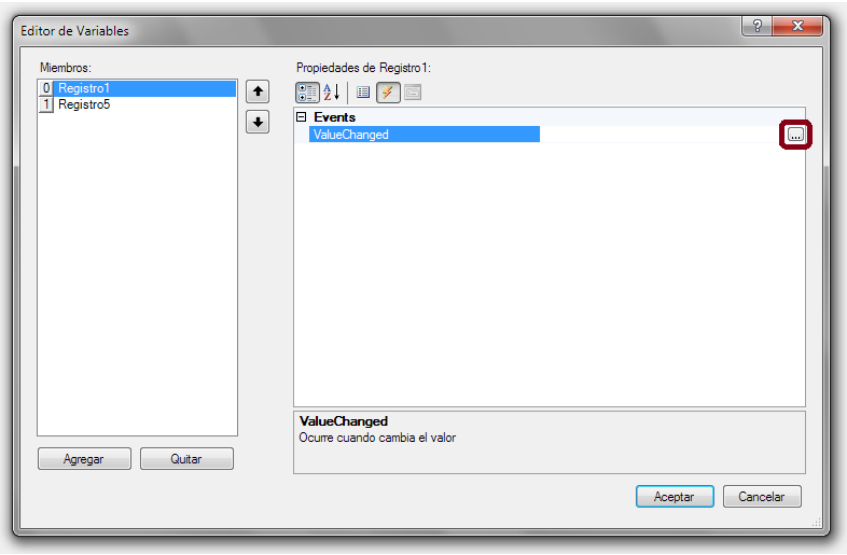


Ilustración 30 – Evento de variable

Y nos mostrará un editor de código para que podamos programar la acción que queramos realizar cuando el valor de la variable cambia.

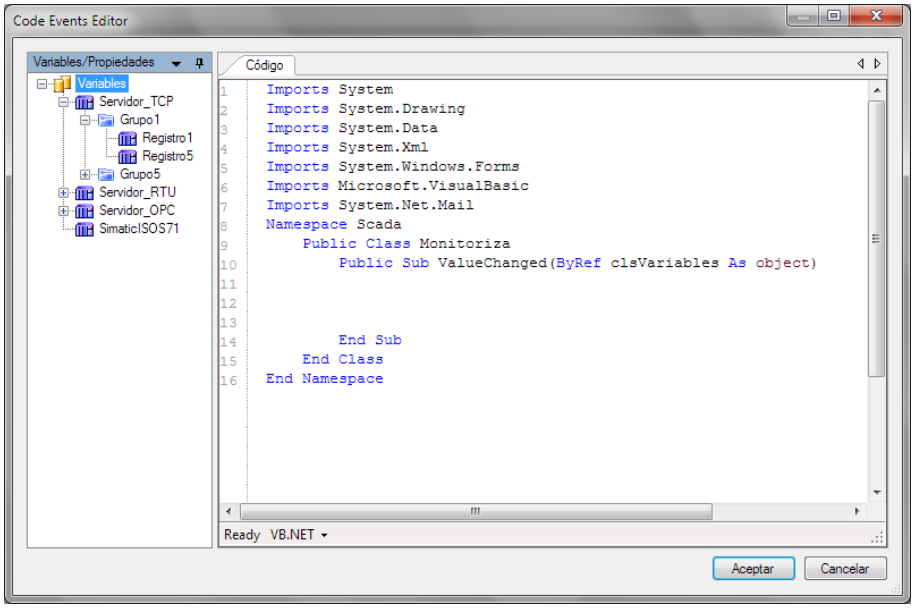


Ilustración 31 – Editor de código de eventos

El código que implementemos se ejecutará en el servidor de comunicaciones de Acimut Monitoriza por tanto tendrá acceso a todos los recursos del servidor sobre el que esté instalado.

ALARMAS

Crear Alarmas

Siempre que queramos que el sistema Scada nos avise de una determinada condición deberemos definir una Alarma. Estas alarmas pueden ser por ejemplo que una temperatura este por encima o por debajo de un determinado valor, que la presión haya alcanzado un valor o cualquier otro aspecto que queramos considerar.

Las alarmas están asociadas a las variables y por tanto es en la ventana de definición de variables donde establecemos si queremos vincular una variable a una alarma. Para ello cuando estamos creando o editando una variable pondremos a True la propiedad **Active Alarms** con lo cual nos aparecerá la propiedad **Alarms** en la que pulsando en el botón de tres puntos podremos definir las alarmas asociadas a la variable.

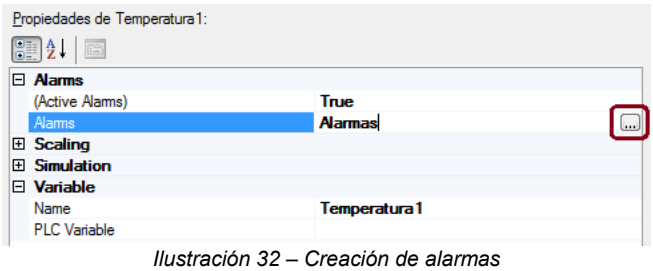


Ilustración 32 – Creación de alarmas

Asociada a una variable podemos definir una o varias alarmas en función de las condiciones que queramos controlar.

Al definir la alarma establecemos el mensaje que queremos mostrar, si queremos almacenar el evento para tener constancia de que ha ocurrido o si es necesario que un usuario valide la alarma, así como las condiciones en las que se debe disparar la alarma.

Podemos hacer que la evaluación de las alarmas sea por comparación de la variable en la que está definida la alarma con un valor constante (por ejemplo que la temperatura

sea menor que 40) o bien nos puede interesar comparar la variable en la que está definida la alarma con otra variable del sistema (por ejemplo si estas monitorizando la producción de distintos productos podemos tener almacenado en una variable a que temperatura queremos disparar la alarma en función del producto que estemos produciendo). Estableceremos la propiedad **TypeValue** a *Constant* o *Variable* respectivamente para uno u otro caso.

Adicionalmente podemos definir en grupo de alarmas, mediante la propiedad **AlarmGroup**, este grupo de alarmas nos sirve para especificar permisos a nivel de usuario, o sea, que usuarios podrán ver que alarmas o si pueden validarlas o no tal como se verá en el capítulo de Usuarios y Permisos.

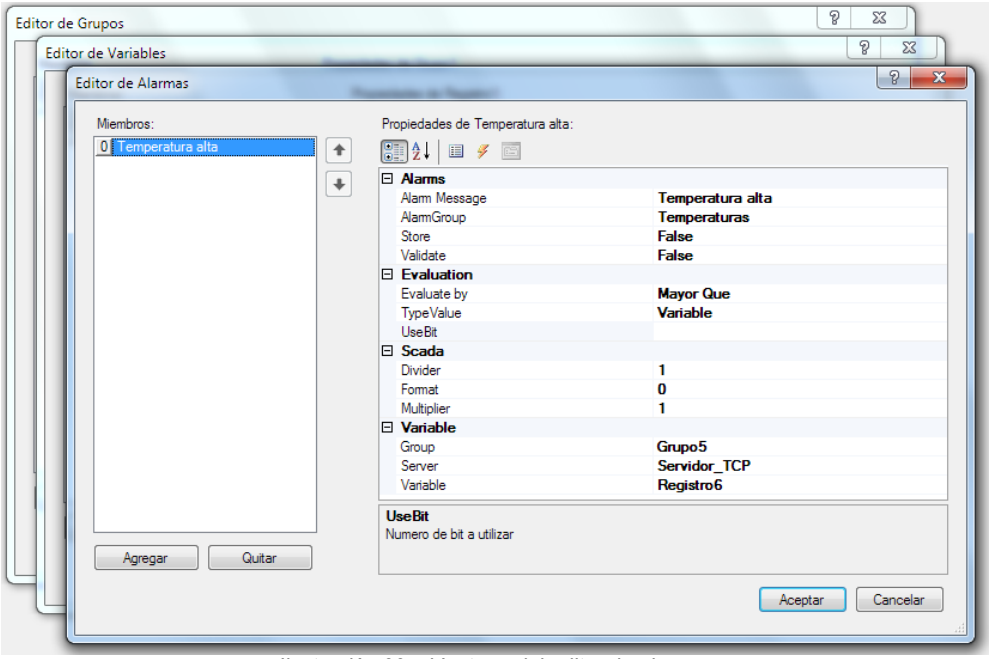


Ilustración 33 – Ventana del editor de alarmas

Veámoslo con detalle examinando cada una de las propiedades de la alarma.

- **Store:** Permite indicar si quedará constancia de la alarma. Para poder almacenar las alarmas se tendrá que definir adicionalmente un servidor de base de datos en el que, en su propiedad **Alarms,** se establezca que es el que se utilizará para guardar éstas.
- **Alarm Message:** Mensaje que mostrará la alarma al activarse.
- **AlarmGroup:** Grupo de permisos a la que está vinculada la alarma.
- **Validate:** Si la establecemos a True, exigirá que un usuario del proyecto scada la valide para que deje de mostrarse en la ventana de alarmas cuando ya no está activa. Para que la validación actúe correctamente es necesario almacenar la alarma. Veremos cómo definir la base de datos, tabla y campos en el punto Guardar en base de datos.
- **Evaluate by:** Establece la condición de comparación que determina cuando estará activa la alarma. Los posibles valores son: *Igual*, *MayorQue*, *MenorQue* y *Distinto*.
- **TypeValue:** Los posibles valores son *Constant* y *Variable*. Especificaremos el valor Constant si la comparación establecida por la propiedad **Evaluate by** es frente a un valor absoluto estando este valor definido por la propiedad Value. Si el disparo de la alarma queremos que sea en función del valor de otra variable definida en el sistema, el valor de **TypeValue** se debe de poner a *Variable* y se usaran las propiedades **Server**, **Group** y **Variable** para especificar la variable con quien comparar.
- **Value:** Establece el valor que al aplicarle la condición de **Evaluate by** hará que se active la alarma, si hemos establecido **TypeValue** a *Constant*. Por ejemplo si

estamos monitorizando una temperatura y definimos la propiedad **Value** a 40 y la propiedad **Evaluate by** a *MayorQue* se nos activará la alarma siempre que el valor de la temperatura supere el valor de 40.

- **UseBit:** Nos permite utilizar de forma aislada los bits de la variable en la evaluación teniendo en cuenta que el primer bit es el 0. Es evidente, que en este caso, solo podremos comparar con los valores 0 y 1.
- **Server, Group y Variable:** Establece que variable del sistema usamos para realizar la comparación establecida por **Evalute by** siempre que **TypeValue** esté establecido a Variable.

El grupo de propiedades Scada nos permite especificar el valor de la variable para su almacenamiento en la base de datos y por tanto para su visualización en el Visor de Alarmas.

- **Divider:** Mostrará el valor de la variable dividido por el número indicado en la propiedad.
- **Multiplier:** Mostrará el valor de la variable multiplicado por el por el número indicado en la propiedad.
- **Format:** Modifica el aspecto del valor de la variable, por ejemplo un formato igual a ##.00 hará que el la variable se muestre con dos decimales.

Los grupos de alarmas los podemos definir bien cuando se está definiendo la alarma mediante la opción New de la propiedad **AlarmGroup** o bien mediante el menú Grupos de Alarma del menú Ver, en el que se nos muestra la siguiente ventana

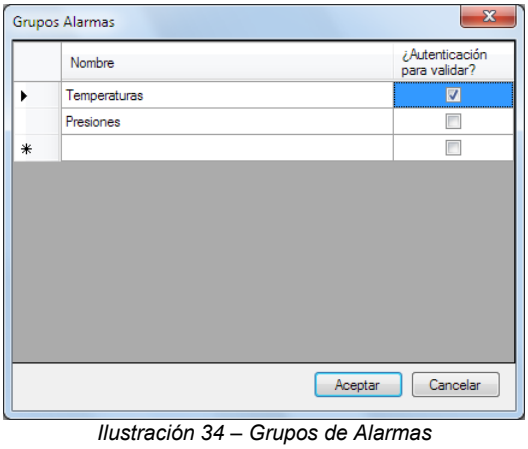


Ilustración 34 – Grupos de Alarmas

En el que además de poder especificar el nombre del grupo de alarma, o cambiarlo si al crearlo nuevo nos hemos equivocado, determinaremos si se quiere forzar una autenticación del usuario volviendo a introducir el nombre y la contraseña cada vez que se quiera validar una alarma que pertenezca al grupo en cuestión.

Tal y como se verá en el capitulo Usuarios y Permisos hay parámetros generales del proyecto que afectan a este comportamiento.

Demostración: http://www.acimut.com/monitoriza/videos/crear_alarmas/default.html

Visor de Alarmas

El visor de alarmas es el elemento central de notificación y validación de las alarmas que se producen en la ejecución del proyecto Scada.

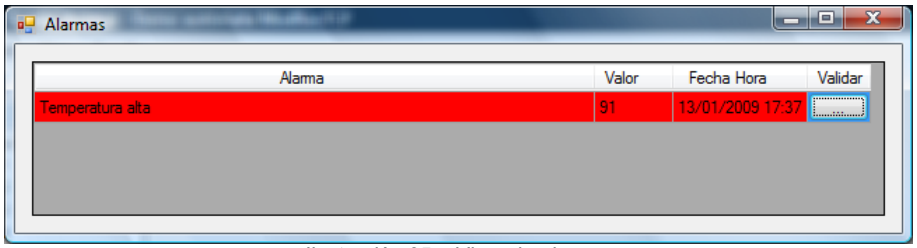
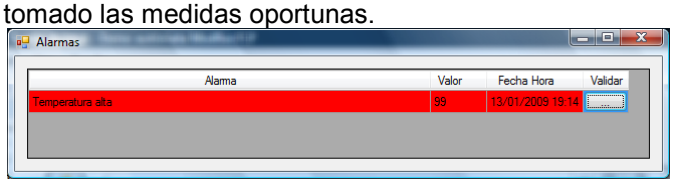


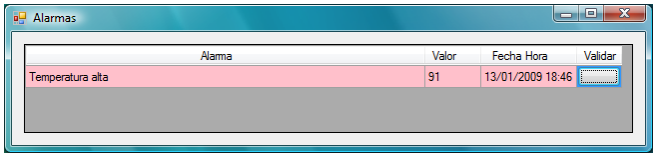
Ilustración 35 – Visor de alarmas

Cada vez que se active una alarma, porque se cumplen las condiciones que se han establecido al diseñar el Scada, se muestra el visor de alarmas, en el tenemos diferentes posibles estados de la alarma en función de si ha definido como que necesita validación o no y estos estados están representados por diferentes colores.

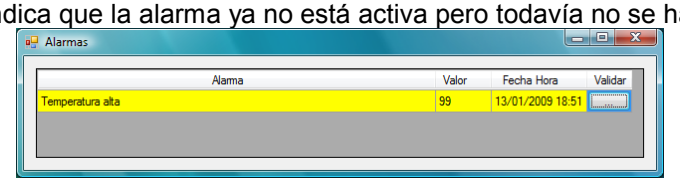
- **Rojo**: Indica que la alarma está activa y requiere que se valide, o sea, hay que pulsar sobre el botón ***Validar*** para hacer constar que un usuario ha tenido en cuenta la alarma y ha tomado las medidas oportunas.



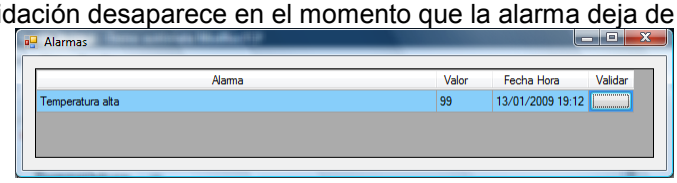
- **Rosa**: Indica que la alarma está activa y se ha validado, o sea, un usuario de Scada ha pulsado sobre el botón ***Validar***.



- **Amarillo**: Indica que la alarma ya no está activa pero todavía no se ha validado.



- **Azul**: Indica que la alarma está activa y no requiere validación. En este caso al no requerir validación desaparece en el momento que la alarma deja de estar activa.



Si una alarma requiere validación, en función del grupo de alarma al que pertenezca, puede que sea necesario volverse a autenticar, esto es, puede que se le vuelva a solicitar el usuario y la contraseña.

Esto se verá con más detalle en el capítulo de Usuarios y permisos. Su función es reforzar la seguridad y que solamente los usuarios autenticados puedan validar la alarma.

Demostración: http://www.acimut.com/monitoriza/videos/validar_alarma/default.html

Eventos de Alarmas

Los Eventos de Alarmas es uno de los elementos de extensibilidad de Acimut Monitoriza más importantes.

Mediante el Visor de Alarmas, descrito en el apartado anterior, hemos visto que tenemos la oportunidad de ser notificados cada vez que se produce una alarma y que podremos validarla, en el caso que así este definido, de forma que nos seguirá apareciendo la alarma hasta estar validada aunque la situación de alarma haya dejado de producirse.

El problema es que el Visor de Alarmas nos notifica mediante una ventana emergente, en el puesto de operación correspondiente, que la alarma se ha producido, y por tanto si no se está controlando esa pantalla no nos daremos cuenta de la alarma.

Para solucionar esto tenemos los Eventos de Alarmas, que los podemos definir como procedimientos que se disparan, en el servidor de Acimut Monitoriza, cada vez que una situación de alarma se **inicia**, **finaliza** o se **valida**.

El uso típico de estos eventos es enviar una notificación de que la alarma ha entrado en un determinado estado, por ejemplo enviando un correo electrónico o un mensaje de SMS. De esta forma, si estamos controlando, por ejemplo, una temperatura y esta supera un determinado valor podemos hacer que se nos envíe a un determinado teléfono o a una cuenta de correo un mensaje indicándonos que la temperatura ha superado el valor establecido.

Si bien esto es un uso típico, la verdad es que las posibles acciones que queramos hacer cuando se produce una alarma es muy variada (poner en marcha un proceso en el servidor, o pararlo, grabar en una base de datos, invocar a un servicio externo a nuestro sistema…) por tanto la forma en que Acimut Monitoriza permite definir los procedimientos que se dispararan cuando la alarma se inicie, finalice o se valide es mediante programación del procedimiento en C# o VB.NET.

Vamos a ver cómo hacer todo esto:

Desde la ventana del Editor de Alarmas nos situaremos en la alarma a la que queremos asignarle eventos

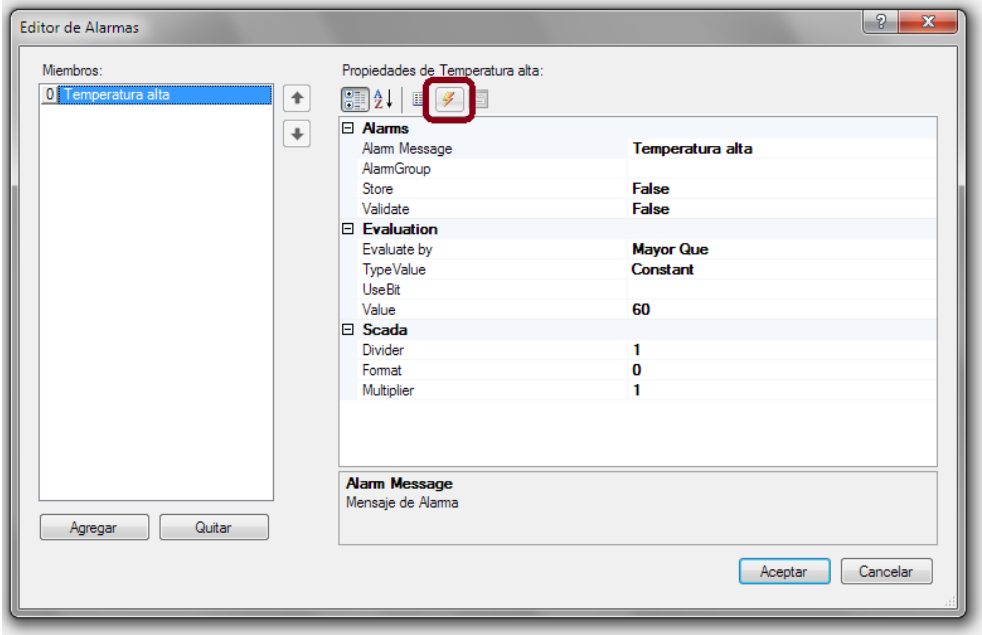


Ilustración 36 – Editor de alarmas

Pulsaremos sobre el botón Eventos (marcado en rojo en la figura) y se nos mostrará la siguiente ventana.

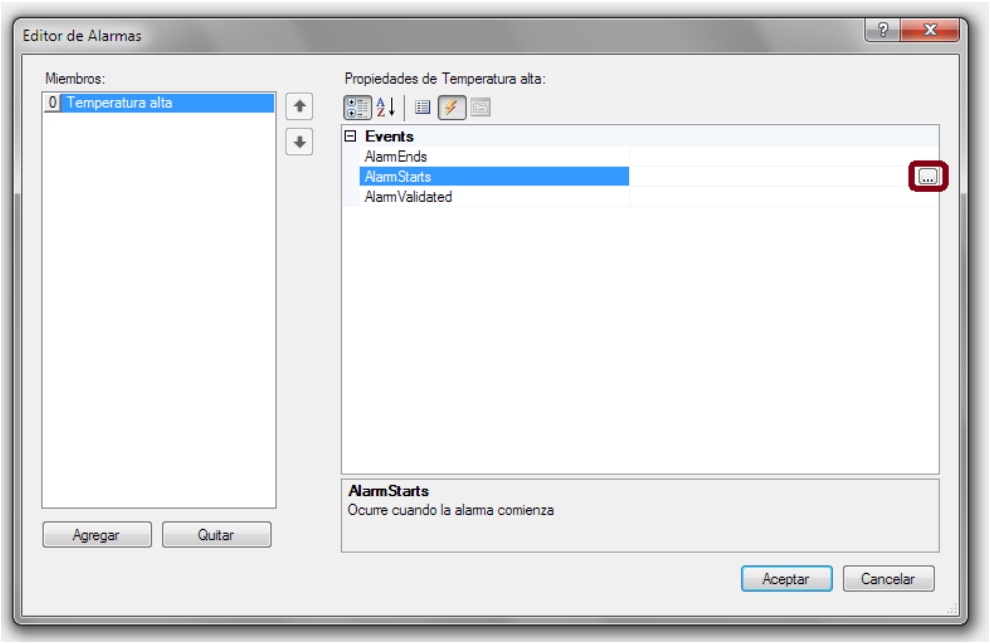


Ilustración 37 – Eventos de alarmas

En ella aparecen tres eventos que son AlarmStars, AlarmEnd y AlarmValidated que son los eventos que se dispararan cuando se inicie una situación de alarma, finalice o se valide respectivamente.

Podremos asociar código a uno o a todos los eventos indistintamente. Por ejemplo para asociar código al evento AlarmStarts pulsaremos sobre el botón de tres puntos (marcado en rojo en la figura) para que se abra el editor de código del evento.

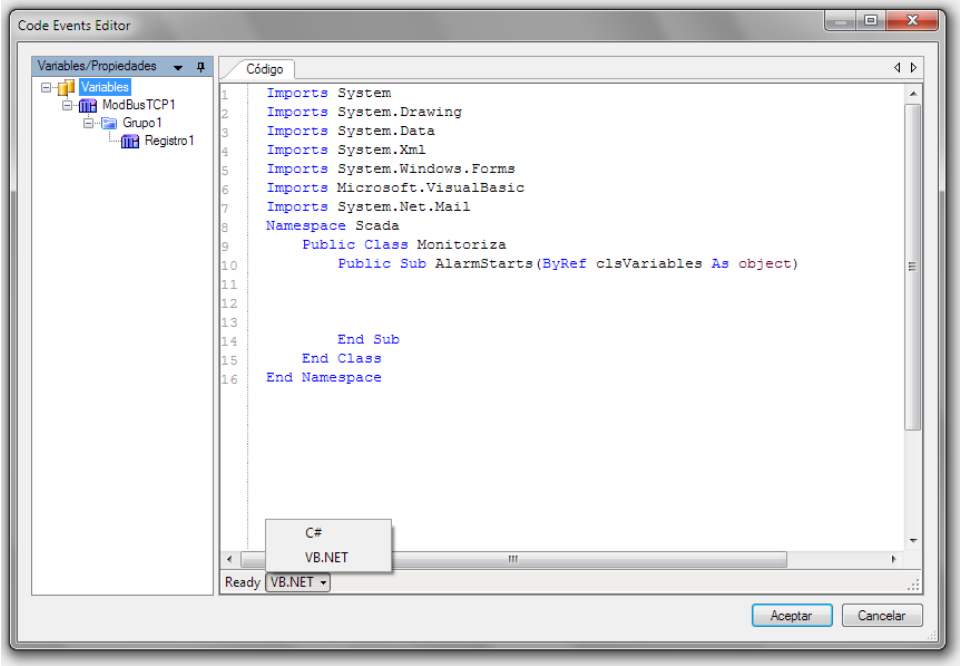


Ilustración 38 – Editor de código de eventos de alarmas

En el Editor de Código de Eventos vemos que podemos escribir el código tanto en C# como en VB.NET, simplemente seleccionando del menú correspondiente.

En el panel izquierdo se nos muestra la relación de Servidores, Grupos y Variables que tengamos definidos, teniendo la posibilidad de arrastrar cualquiera de las variables sobre el editor para así poder recoger su valor o bien modificarlo.

A continuación se muestra en ejemplo del código que sería necesario para enviar un mensaje de correo electrónico utilizando una cuenta de *gmail.com*. En el que se puede observar como simplemente consiste en la creación de un objeto `SmtpClient` que nos establece la conexión con el servidor de correo, un objeto `MailMessage` que nos permite definir el remitente, los destinatarios, el asunto y cuerpo del mensaje. En particular en el ejemplo se ve como para componer el cuerpo del mensaje `MyMailMessage.Body` se utiliza el valor de la variable `Registro1`.

```
Imports System
Imports System.Drawing
Imports System.Data
Imports System.Xml
Imports System.Windows.Forms
Imports Microsoft.VisualBasic
Imports System.Net.Mail
Namespace Scada
    Public Class Monitoriza
        Public Sub AlarmStarts(ByRef clsVariables As object)
           Try
                'Comience por crear un objeto de mensaje de correo
                Dim MyMailMessage As New MailMessage()

                'El campo From requiere una instancia del tipo MailAddress
                MyMailMessage.From = New MailAddress("remitente@gmail.com")

                'El campo To es una colección de tipos MailAddress
                MyMailMessage.To.Add("destinatario@dominio.com")

                MyMailMessage.Subject = "Inicio de Alarma Grupo1, Registro1"

                Dim strBody As String
```

```
                strBody = String.Format("{0}. La variable Grupo1, Registro1 ha
alcanzado el valor {1}", Now(), clsVariables.Variables( "ModBusTCP1", "Grupo1",
"Registro1", ""))

                MyMailMessage.Body = strBody

                'Crea el objeto SmtpClient y especifica las credenciales del usuario
                Dim SMTPServer As New SmtpClient("smtp.gmail.com")
                SMTPServer.Port = 587
                SMTPServer.Credentials = New System.Net.NetworkCredential(
"usuario@gmail.com", "contraseña")
                SMTPServer.EnableSsl = True

                'Envia el mensaje
                Try
                  SMTPServer.Send(MyMailMessage)
                Catch ex As SmtpException
                   MessageBox.Show("Ex1=" & ex.Message)
                End Try
            Catch ex2 As SmtpException
               MessageBox.Show("Ex2=" & ex2.Message)
            End Try
        End Sub
    End Class
End Namespace
```

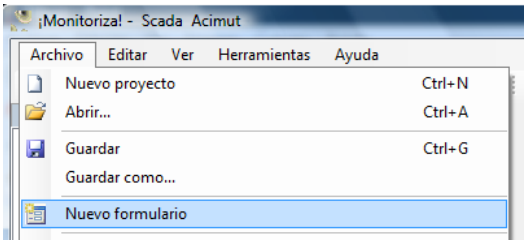
CREAR FORMULARIOS

En Acimut Monitoriza para crear un proyecto Scada lo mínimo necesario es definir los servidores de comunicaciones con los autómatas y la interfaz de usuario.

Para crear la interfaz de usuario diseñaremos tantos formularios como nos sea necesario para nuestro proyecto y para ello le agregaremos cualquiera de los controles que se muestran en la ventana ***ToolBox***, en cualquiera de sus la solapas: *Scada, Diseño, Windows y Controles de usuario.* También podemos agregar los controles arrastrando directamente desde la ventana de ***Variables*** la variable correspondiente como veremos un poco más adelante.

Para añadir un formulario a un proyecto tan solo hay que hacer clic en el botón de la barra de herramientas que mostramos a continuación.

O bien seleccionar la opción ***Nuevo Formulario*** del menú ***Archivo***.



El aspecto que tendrá ahora el editor de Monitoriza es el siguiente

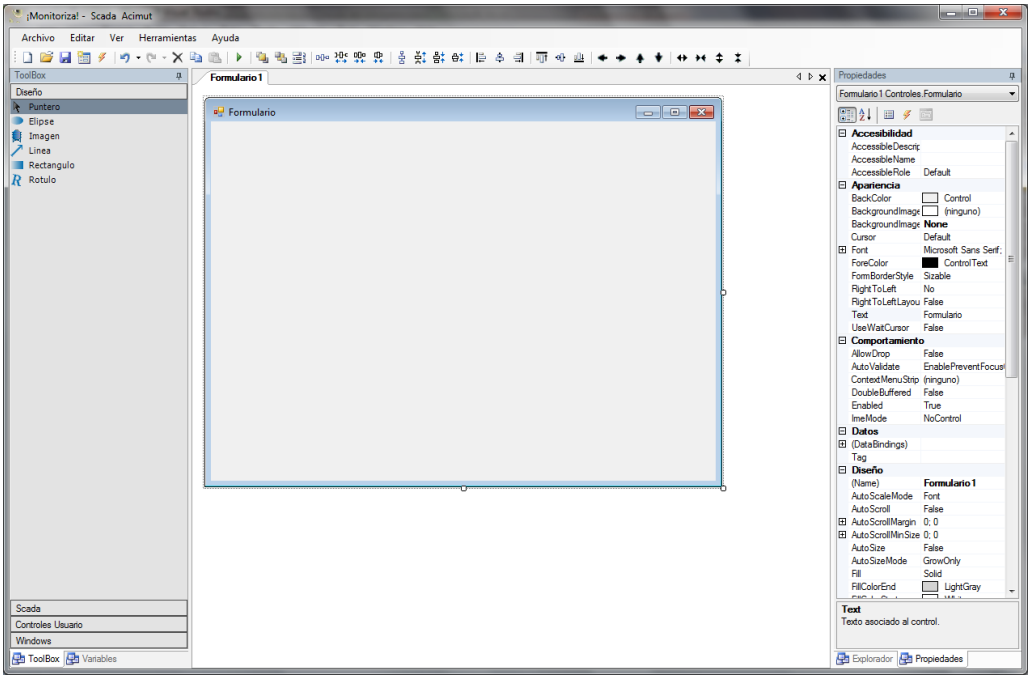


Ilustración 39 – Editor de Monitoriza

Tenemos ahora un nuevo formulario y sus propiedades a la derecha.

También observamos que ahora ya se han activado las herramientas de edición en la barra de herramientas.

Los formularios adicionales que se creen aparecerán en nuevas solapas, además de en el árbol del Explorador de la derecha. Cerrar una solapa no implica que se elimine un formulario.

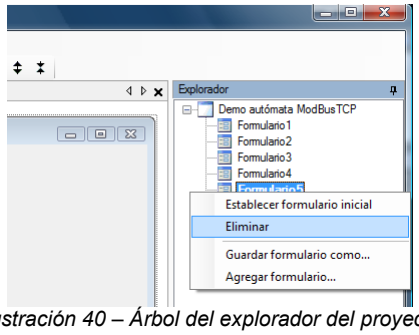


Ilustración 40 – Árbol del explorador del proyecto

El explorador de proyectos nos permite con un doble clic abrir un formulario y mediante un menú contextual *Eliminar*, *Exportar* e *Importar* formularios.

Demostración: http://www.acimut.com/monitoriza/videos/crear_formularios/default.html

Como decíamos anteriormente para añadir controles al formulario podemos arrastrarlos desde la ventana de ***ToolBox*** como se indica en la siguiente ilustración

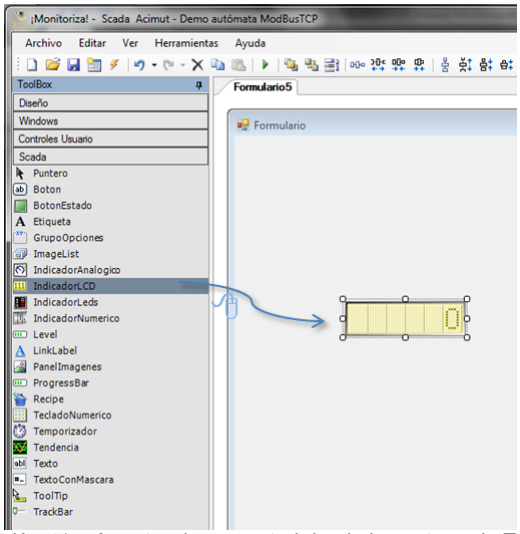


Ilustración 41 – Arrastre de un control desde la ventana de ToolBox

O bien arrastrar una variable desde la ventana de Variables tal y como se aprecia en la siguiente ilustración

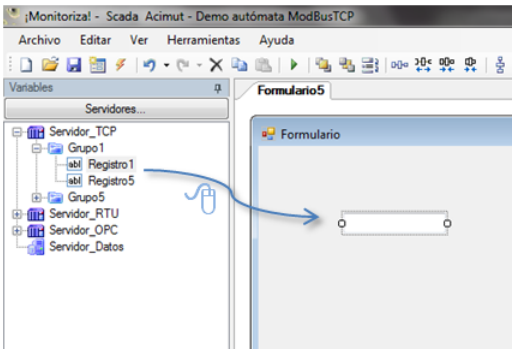
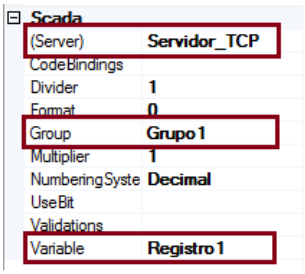


Ilustración 42 – Arrastre de un control desde la ventana de variables

El arrastrar una variable tiene la ventaja que en la misma operación estamos realizando dos acciones ya que por una parte diseñamos el formulario estableciendo los controles necesarios y por otra a esos controles les asignamos directamente las propiedades *Server*, *Group* y *Variable*



Por defecto al arrastrar una variable nos crea un control Texto, pero pulsando con el ratón sobre el nombre de la variable le podemos cambiar a esa variable el tipo de control asociado a ella, simplemente eligiendo el tipo correspondiente de la lista que se nos despliega.

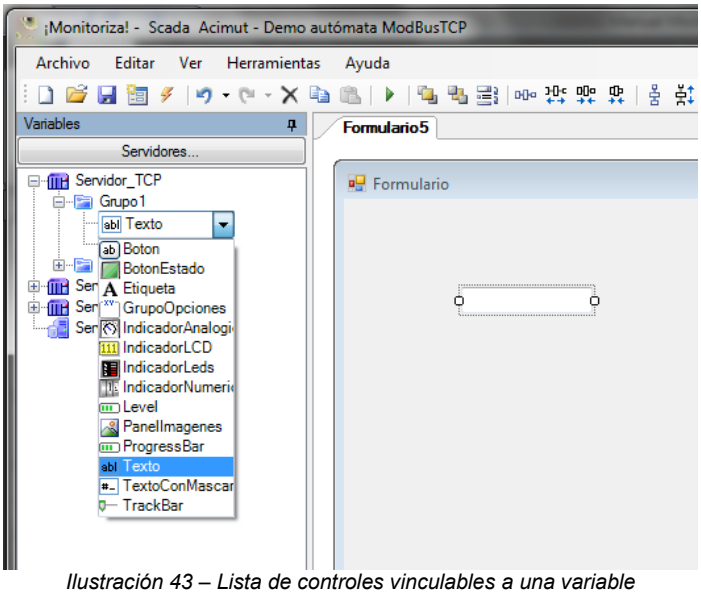
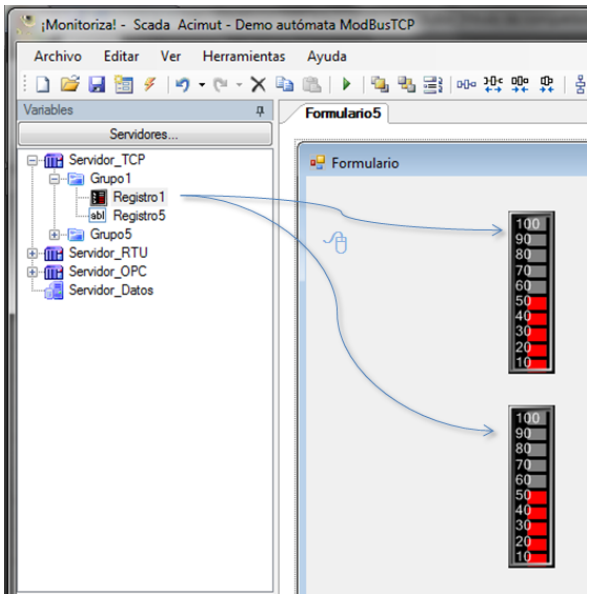


Ilustración 43 – Lista de controles vinculables a una variable

La lista de controles que se despliega está compuesta por todos los controles de Acimut Monitoriza que pueden mostrar o editar el valor de una variable.

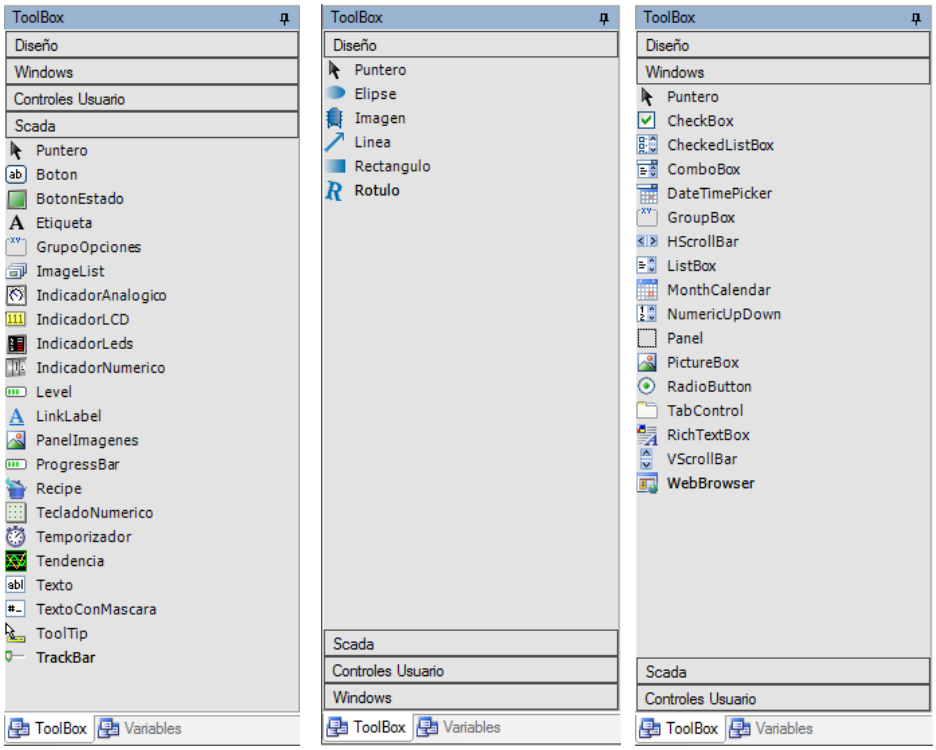
Una vez que tenemos asignado un tipo de control para la variable cada vez que arrastremos esa variable se nos creará un control del tipo correspondiente en cualquiera de los formularios de nuestro proyecto.



Otra operación que se puede realizar es arrastrar una variable pero dejándola caer sobre un control existente, mediante está operación lo que conseguimos es asignar al control existente las propiedades Server, Group y Variable de la variable que estamos arrastrando. En el caso que arrastremos la variable sobre un control que no tenga las propiedades Server, Group y Variable evidentemente no tendrá ningún efecto la operación.

CONTROLES DE MONITORIZA

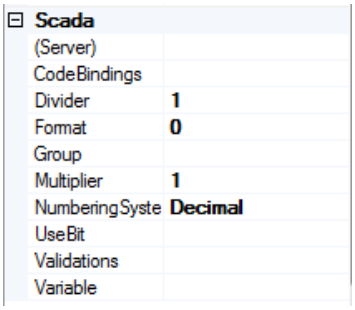
Los controles que pueden utilizarse son los que están en la ToolBox, en cualquiera de las solapas Scada, Diseño, Windows o Controles Usuario.



Todos estos controles pueden arrastrarse a la superficie de un formulario, una vez sobre el formulario podrán editarse sus propiedades mediante las cuales le daremos la funcionalidad deseada.

Existen tres grupos principales de controles, los que nos permiten agrupar funcionalidades, como son Panel y TabControl, y los que nos permitirán mostrar al usuario los valores de las Variables.

Son estos últimos controles los que tienen propiedades para poder definir cómo mostrarán el valor de una variable.



Vemos a continuación un ejemplo de estas propiedades, comunes en la mayor parte de los casos.

Las propiedades (Server), Group y Variable hacen referencia a la variable que se representará. La propiedad UseBit, si se utiliza, hace referencia al bit, entre 0 y 15, que se representará (valor 0 o 1).

Las propiedades Divider y Multiplier alteran el valor mostrado. La propiedad NumberingSystem permite mostrar valores en formato hexadecimal.

Format nos permite, por ejemplo, mostrar y limitar el número de decimales mostrados.

Por último, la propiedad Validations, nos permite establecer condiciones de escritura y de visualización sobre el control, en caso de cumplirse las validaciones aparecerá un aviso de incidencia junto al control.

Vemos a continuación el Editor de Validaciones.

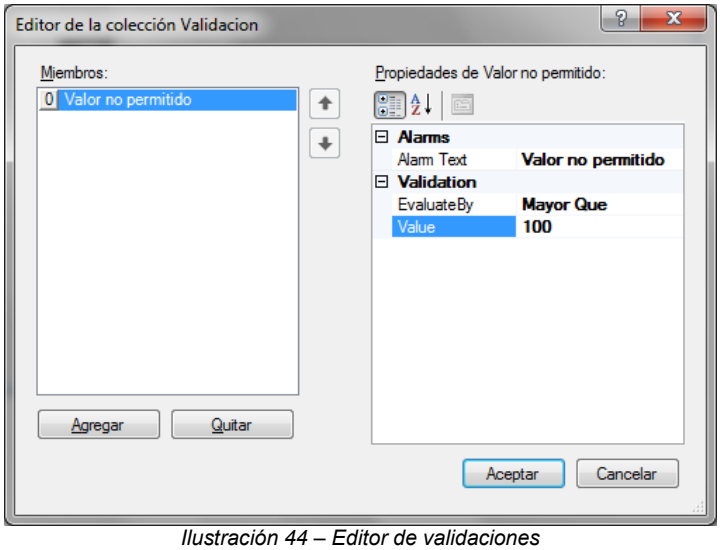


Ilustración 44 – Editor de validaciones

La relación completa de los controles disponibles es la siguiente:

Solapa Scada

Botón

Las propiedades principales de este control son: ***Action*** y ***Actions***, usaremos la primera cuando queremos que el botón realice una única acción y la propiedad ***Actions*** cuando se deben realizar un conjunto de acciones de forma consecutiva.

Las posibles opciones de la propiedad ***Action*** y del conjunto de ***Actions*** son las siguientes:

- **NoAccion:** Indica que el botón no realiza ninguna acción.
- **AbrirFormulario:** Abre formulario indicado en la propiedad ***Form***.
- **CerrarFormulario:** Cierra el formulario actual.
- **SalirScada:** Cierra la aplicación.
- **ForzarValor:** Fuerza a que la variable indicada mediante las propiedades ***Server***, ***Group*** y ***Variable*** tome el valor indicado en la propiedad ***Value***.
- **MostrarHistoricoAlarmas:** Muestra el formulario correspondiente al histórico de alarmas.
- **MostrarGráfica:** Muestra el formulario correspondiente al histórico de datos.

Para establecer un conjunto de acciones para el botón se hace mediante el editor de la colección de acciones como se ve en la siguiente imagen:

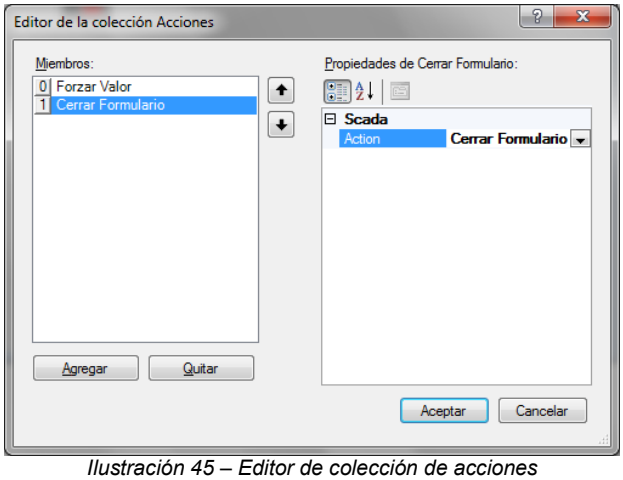


Ilustración 45 – Editor de colección de acciones

BotonEstado

Este control nos servirá normalmente para mostrar y modificar el estado de una variable.

Por defecto el funcionamiento del ***BotonEstado*** consiste en asignarle una variable de las definidas en los servidores de comunicaciones y el botón mostrará un color u otro en función de si la variable vale 0 ó 1 y además cada vez que pulsemos el botón intercambiará el valor de la variable a 0 ó 1

Pero también podemos hacer que muestre un conjunto de imágenes o una relación de colores en función del valor de la variable y que vaya recorriendo la colección de imágenes o los colores cada vez que pulsemos el botón. A cada una de las imágenes o colores se le asigna un valor o rango y por tanto mostrará la imagen o el color en función del valor de la variable.

Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Que nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad **UseBit** si queremos referirnos a un determinado bit en concreto.
- **Color v=0 y Color v=1:** Definen el color que mostrará el botón cuando la variable vale 0 y 1 respectivamente. Los colores por defecto son Rojo para el valor 0 y Verde para el valor 1
- **Shape:** Los posibles valores son *Circulo* y *Cuadrado* harán que el botón se represente respectivamente como se puede ver en las siguientes figuras:

- **BorderWidth:** Establece el ancho del borde.
- **ReadOnly:** Indica que el botón simplemente mostrará el valor de la variable, la pulsación del botón no tendrá ningún efecto.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Text:** La propiedad text nos permite establecer un texto que se mostrará en el control
- **Images:** Cuando queremos hacer que el botón muestre un conjunto de imágenes o una relación de colores en función del valor de la variable lo haremos mediante esta propiedad con el editor de la colección que se muestra en la figura siguiente:

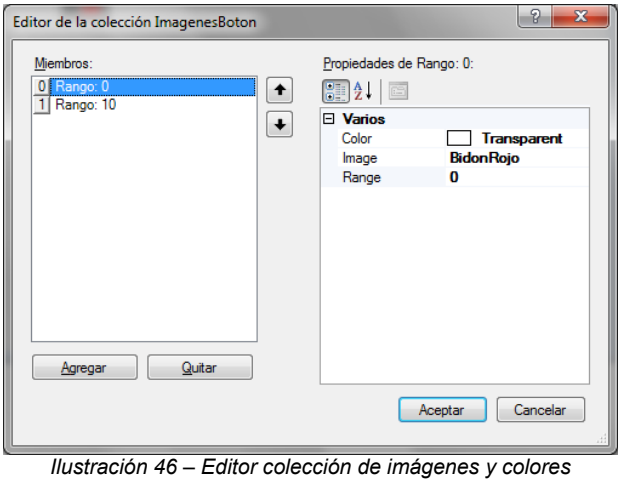


Ilustración 46 – Editor colección de imágenes y colores

En el agregaremos tantos rangos de valores como estados queramos tener, para cada estado podemos asignarle un color y una imagen, la forma en que se comporta el control es que si un determinado rango no tiene imagen asignada mostrará el color correspondiente a ese rango, mientras que si hay una imagen asignada prevalecerá la imagen sobre el color y mostrará por tanto la imagen correspondiente

De igual forma la propiedad ***Images*** prevalece sobre las propiedades ***Color v=0*** y ***Color v=1***, o sea, que si tenemos definidos uno o varios rangos de valores en la propiedad ***Images*** el comportamiento del control será el establecido por esta propiedad y no tendrá en cuenta los valores que tengan las propiedades ***Color v=0*** y ***Color v=1***

Adicionalmente, las imágenes se ven afectadas por la propiedad **Rotate**, que como se describe a continuación, modifica el aspecto de las imágenes, transformando si se desea la forma de representarlas.

Rotate integra en una sola propiedad la posibilidad de rotar la imagen (90,180 y 270) y la posibilidad de generar una imagen reflejada (en el eje X, el eje Y o ambos). Veamos algunos ejemplos:

Etiqueta1 Etiqueta

El control etiqueta nos permite mostrar textos fijos, valores de variables y textos variables en función del valor de la variable

Es un control de solo lectura, o sea, su funcionalidad es mostrar textos o valores pero nunca podremos modificar un valor con este control.

El comportamiento por defecto del control es el de mostrar un texto fijo, para mostrar el valor de una variable deberemos vincular una variable con el control a través de las propiedades ***Server***, ***Group***, ***Variable***

Si lo que queremos es que nos muestre distintos textos lo haremos a través de la propiedad ***Messages***.

Veamos por tanto las principales propiedades del control:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.

  El establecer este conjunto de variables hace que cambie el comportamiento del control de manera que se mostrará el valor de la variable referenciada por Server, Group y Variable en vez del texto que esté establecido en la propiedad Text.

- **Text:** Texto asociado al control. Esta propiedad no tiene efecto si se establecen las propiedades Server, Group y Variable.
- **AutoEllipsis:** Si se establece a True, los textos que se extienden más allá del ancho del control se sustituyen por puntos suspensivos.
- **AutoSize:** Si se establece a True se habilita el cambio automático del tamaño del control para ajustarse al tamaño del texto.
- **Divider:** Cuando el control está vinculado a una variable, muestra el valor de la variable dividido por el valor de la propiedad.
- **Format:** Cuando el control está vinculado a una variable, muestra el valor según el formato especificado por la propiedad, por ejemplo un formato igual a ##.00 hará que el la variable se muestre con dos decimales.
- **Multiplier:** Cuando el control está vinculado a una variable, muestra el valor de la variable multiplicado por el valor de la propiedad.
- **UseBit:** Número del bit a utilizar, el control mostrará un valor de 0 ó 1 en función del valor del bit especificado de la variable a la que esté vinculado.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Messages:** Mediante esta propiedad podemos hacer que el control muestre distintos mensajes en función del valor de la variable vinculada, para ello mediante el editor de la colección de mensajes estableceremos los rangos a los que queremos que se muestre cada uno de los mensajes.

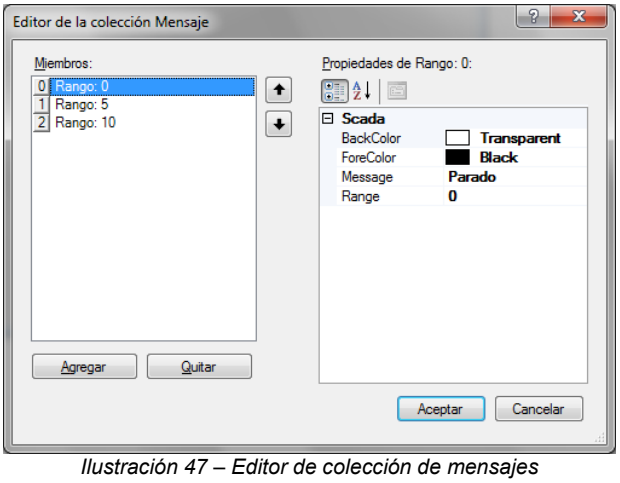
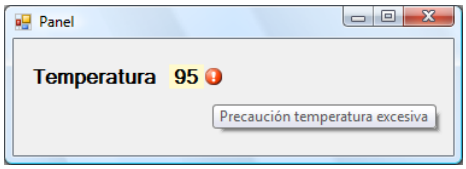


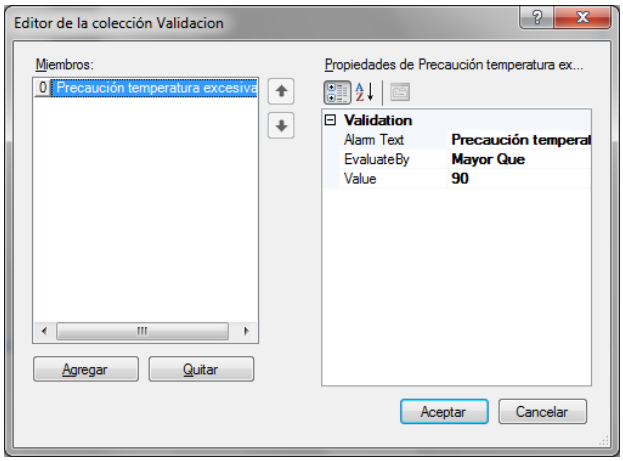
Ilustración 47 – Editor de colección de mensajes

Para cada uno de los posibles mensajes estableceremos las propiedades Range, Message, BackColor y ForeColor, siendo el *Range* el valor máximo por debajo del cual queremos que se muestre el mensaje, *Message* el texto que se mostrará y BackColor y ForeColor los colores del fondo de la etiqueta y del texto respectivamente.

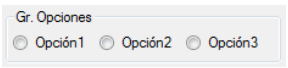
- **Validations:** La propiedad Validations nos permite definir un conjunto de condiciones, sobre el valor de la variable a la que esté vinculado el control, de forma que si en la ejecución del proyecto se cumple alguna de esas condiciones se mostrará un aviso junto el control, por ejemplo, podemos diseñar un formulario como el de la figura siguiente



en el que se han puesto dos etiquetas una con un texto fijo y otra vinculada a una variable, en esta se ha establecido la propiedad Validations mediante el editor de colección de validaciones tal y como se muestra en la figura siguiente



en el que se ha especificado que se muestre un mensaje de alarma asociado a este control cuando el valor de la variable es mayor que 90.



GrupoOpciones

El control GrupoOpciones nos sirve para establecer o mostrar el valor de una variable que tiene un conjunto de valores definido.

Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Text:** Texto que se muestra como cabecera del grupo de opciones.

- **ReadOnly:** Si se establece a True indica que el control mostrará el valor de la variable pero no nos dejará modificarlo.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Options:** Mediante esta propiedad establecemos el conjunto de opciones asociadas al control, para ello se utiliza el Editor de la colección de opciones que se muestra en la figura siguiente

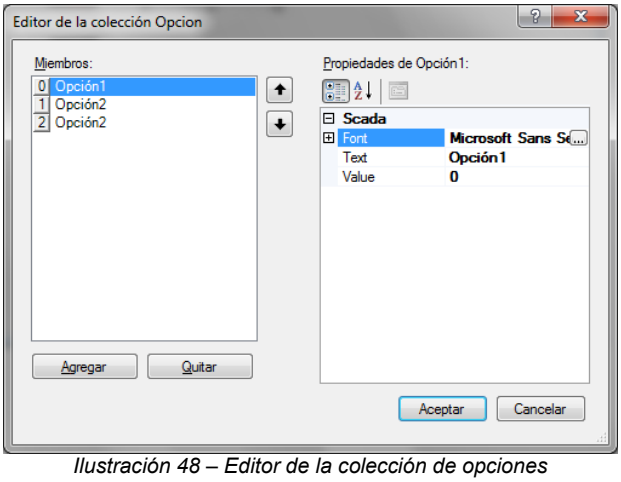


Ilustración 48 – Editor de la colección de opciones

en el que para cada una de las opciones estableceremos las propiedades *Value*, *Text* y *Font* donde *Value* es el valor de la variable vinculada para el cual se activará esta opción y por tanto el valor que se le asignará cuando se seleccione la opción y *Text* y *Font* son el texto que se mostrará con la fuente correspondiente.

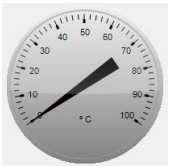
ImageList1 ImageList

El control ImageList se utiliza para almacenar imágenes que, a partir de ese momento, podrán mostrar otros controles.

Puede utilizar una lista de imágenes con cualquier control que tenga una propiedad ImageList

Las principales propiedades de este control son las siguientes:

- **ColorDepth:** Obtiene o establece el número de colores utilizado para procesar la imagen de la lista de imágenes. Esta propiedad es necesario establecerla antes que la propiedad Images.
- **ImageSize:** Tamaño de las imágenes en píxeles. Esta propiedad es necesario establecerla antes que la propiedad Images.
- **Images:** Colección de imágenes almacenadas en el ImageList.



IndicadorAnalogico

La función de este control es la de mostrarnos el valor de una variable de manera visual, teniendo una representación analógica de la variable, nos servirá normalmente cuando sabemos entre que intervalo de valores se encuentra la variable y queremos hacer una representación gráfica de ella.

El control tiene la posibilidad de almacenar el valor máximo que alcance la variable en la sesión correspondiente, o sea, mientras que se está ejecutando la aplicación, no lo almacena de forma permanente.

Si queremos durante una sesión de ejecución de la aplicación se restablezcan el valor máximo podremos hacerlo mediante un Click sobre el control.

Mediante programación se puede invocar al método Reset del control para restablecer el valor máximo.

Las principales propiedades de este control son las siguientes:

- **ArrowColor:** Obtiene o establece el color con que se mostrará la flecha que indica el valor de la variable.
- **Decimals:** Número de decimales con que se muestran en los números de la escala de valores.
- **StoreMaximum:** Se establecerá True si se desea que se visualice durante la sesión de ejecución el máximo alcanzado por la variable. Por ejemplo en la siguiente figura se muestra el IndicadorAnalogico representando un valor de 60, habiendo alcanzado un máximo de 80

- **StartingAngle:** Ángulo inicial en grados en el cual se empezara a dibujar la escala de valores.
- **BarsBetweenValues**: Número de divisiones o marcas secundarias que se dibujarán entre cada una de las marcas principales establecidas para la escala de valores.
- **EspacingBetweenNumbers**: Espaciado en grados entre las marcas principales de la escala de valores.
- **Interval**: Número de unidades de la escala de valores entre marcas principales.
- **Maximum**: Valor máximo de la escala de valores que puede representar el control. Cualquier valor de la variable superior al valor de la propiedad se representará con el valor de la propiedad.
- **MaximumStored**: Valor máximo alcanzado por la variable durante la sesión de ejecución. No almacena el máximo histórico de la variable.

- **Minimum**: Valor mínimo de la escala de valores que puede representar el control. Cualquier valor de la variable inferior al valor de la propiedad se representará con el valor de la propiedad.
- **ClockWiseDirection**: Sentido de giro en que se representan los valores de la escala. Si se establece a True el sentido de giro será el de las agujas del reloj.
- **Text**: Texto asociado al control. Normalmente usaremos este texto para indicar lo que estamos midiendo o las unidades de la variable.
- **Server, Group y Variable:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.

IndicadorLCD

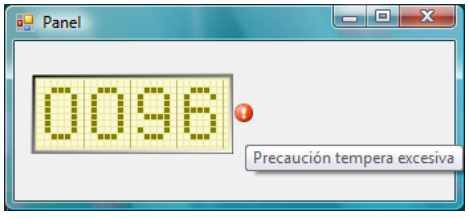
La función del control IndicadorLCD es fundamentalmente de diseño, en el sentido que nos servirá para mostrar el valor de una variable pero de una forma en la que simula la representación mediante un panel LCD de matriz de puntos.

Su función es de solo lectura, ya que mediante este control solo podremos visualizar datos pero no podremos modificarlos.

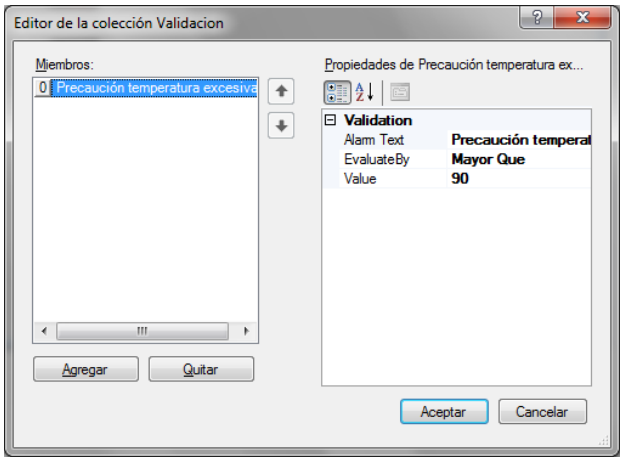
Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Format:** Muestra el valor según el formato especificado por la propiedad, por ejemplo un formato igual a 0000 hará que el la variable se muestre rellenando con ceros las casillas por la izquierda del valor tal y como se muestra en la figura.

- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Validations:** La propiedad Validations nos permite definir un conjunto de condiciones, sobre el valor de la variable a la que esté vinculado el control, de forma que si en la ejecución del proyecto se cumple alguna de esas condiciones se mostrará un aviso junto el control, por ejemplo, podemos diseñar un formulario como el de la figura siguiente

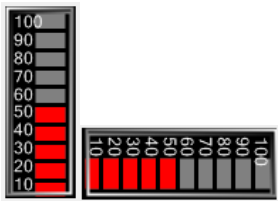


en el que se han puesto un IndicadorLCD vinculado a una variable, en este se ha establecido la propiedad Validations mediante el editor de colección de validaciones tal y como se muestra en la figura siguiente



en el que se ha especificado que se muestre un mensaje de alarma asociado a este control cuando el valor de la variable es mayor que 90.

- **BorderWidth:** Indica el grosor en pixeles del borde que se dibuja alrededor del control.



IndicadorLeds

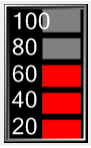
El IndicadorLeds pertenece al conjunto de controles que nos sirve para mostrar de una forma gráfica el valor de una variable.

Su representación se basa en la simulación de que según el valor de la variable se enciende o se apagan los “leds” que se muestran en el control.

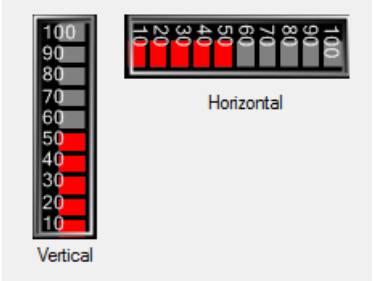
Su función es de solo lectura, ya que mediante este control solo podremos visualizar datos pero no podremos modificarlos.

Las principales propiedades de este control son las siguientes:

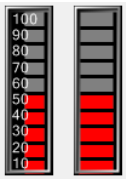
- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Maximum:** Valor máximo de la escala de valores. Cualquier valor de la variable superior a este valor se representará con el valor máximo.

- **LedsNumber**: Número de leds que se mostrarán en el control. Por ejemplo si establecemos su valor a 5 con la propiedad máximo a 100 se representará según se muestra en la figura. 

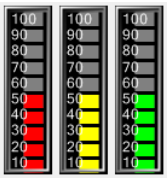
- **Orientation**: Los posibles valores de esta propiedad son: Vertical y Horizontal.



- **ShowScale**: Determina si se muestra la escala de valores del control.



- **ForeColorLeds**: Color de los leds al iluminarse.



IndicadorNumerico

Con el control IndicadorNumerico podremos mostrar el valor de una variable como si fuera un odómetro o cuentarrevoluciones.

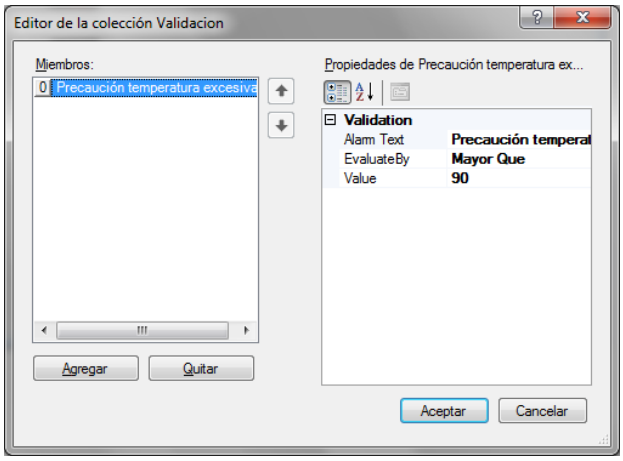
Su función es de solo lectura, ya que mediante este control solo podremos visualizar datos pero no podremos modificarlos.

Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Format:** Muestra el valor según el formato especificado por la propiedad, por ejemplo un formato igual a ##.00 hará que el la variable se muestre con dos decimales usando para ello el color definido en *ForeColorDecimal*. 

- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Validations:** La propiedad Validations nos permite definir un conjunto de condiciones, sobre el valor de la variable a la que esté vinculado el control, de forma

que si en la ejecución del proyecto se cumple alguna de esas condiciones se mostrará un aviso junto el control. Para definir las reglas de validación se usa el Editor de la colección de validaciones que se muestra en la siguiente figura:

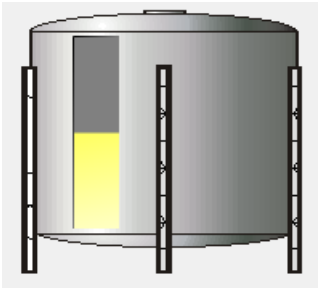


- **Digits:** Número de dígitos o “ruedas del cuentarrevoluciones” que se mostraran.
- **ForeColorDecimal:** Color con que se mostrará la parte decimal de la variable. Por ejemplo el valor 30,56 se mostrará como 

si se establece ForeColorDecimal a Rojo y Formato a la expresión ##.00

Level

El control Level nos sirve para representar de una forma gráfica el valor de una variable. Por ejemplo si tenemos un depósito en que estemos controlando su volumen podemos poner un control Level asociado al volumen sobre una imagen del depósito y tenemos una representación visual del volumen tal y como podemos ver un la figura:



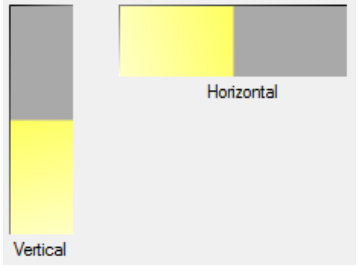
Su función es de solo lectura, ya que mediante este control solo podremos visualizar datos pero no podremos modificarlos.

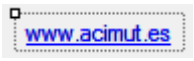
Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente

establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.

- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Maximum:** Valor máximo de la escala. Cuando el valor de la variable supere el máximo se mostrará con el valor correspondiente al máximo.
- **Minimum:** Valor mínimo de la escala. Cuando el valor de la variable sea inferior al mínimo se mostrará con el valor correspondiente al mínimo.
- **ForeColorLevel:** Color con que se representará la parte correspondiente al valor de la variable.
- **Orientation**: Los posibles valores de esta propiedad son: Vertical y Horizontal.

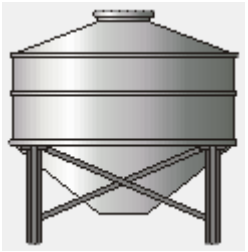




LinkLabel

La función del control LinkLabel es la de poder poner enlaces a sitios web dentro del Scada. De esta forma cuando pulsemos sobre un control LinkLabel se abrirá el explorador de Internet y navegará a la dirección especificada por la propiedad URL.

El control LinkLabel no está asociado a las variables del Scada y por tanto no puede representar ningún valor.



PanelImagenes

El control PanelImagenes nos permite visualizar distintas imágenes en función del valor de una variable.

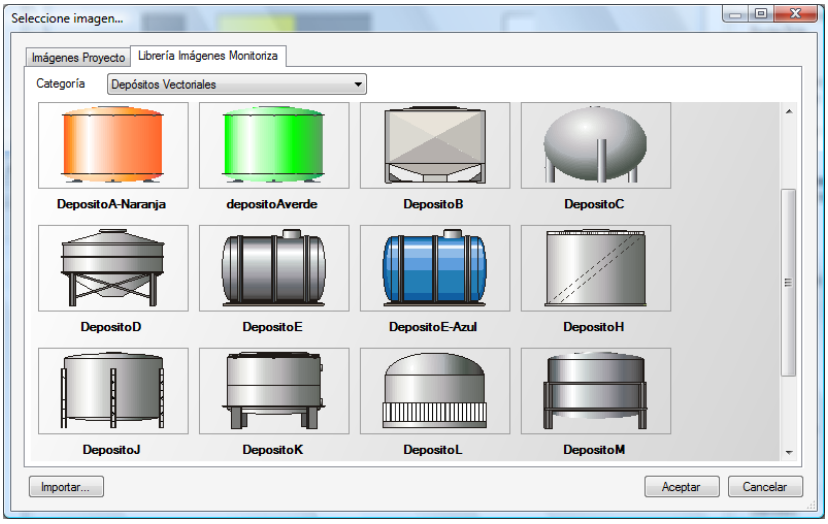
Así, por ejemplo, podemos hacer que el control PanelImagenes nos muestre la imagen siguiente si el valor de la variable es menor que 150

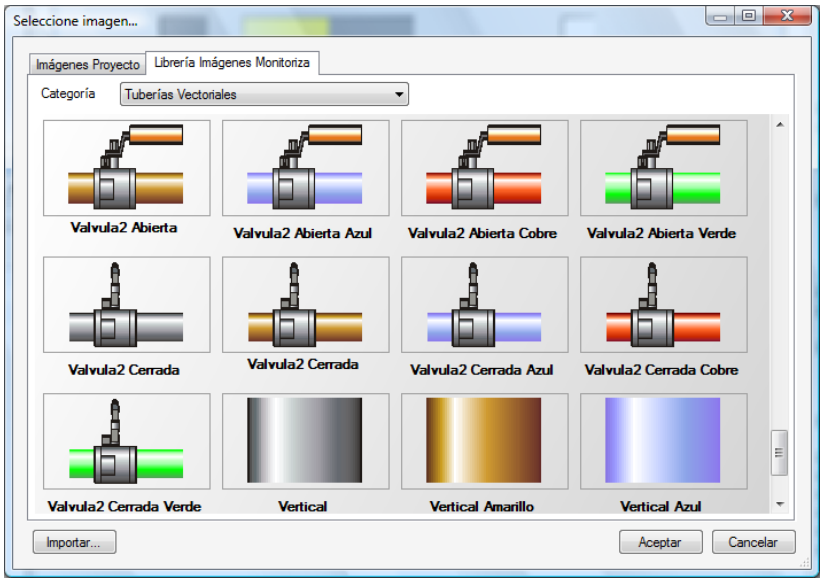
y la imagen siguiente si es mayor o igual que 150

Para asignar las imágenes usaremos la propiedad ***images*** y mediante el editor de la colección de imágenes iremos definiendo las imágenes y los valores a los que queremos que se muestren.

Acimut Monitoriza dispone de una gran variedad de imágenes predefinidas gracias a su Librería de imágenes que nos posibilitará realizar unos diseños atractivos.

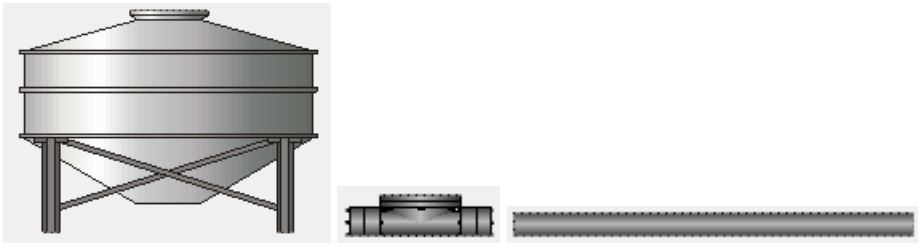
Algunas de estas imágenes son las que se muestran en las siguientes figuras:



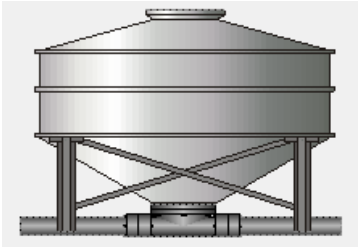


Un tema importante es que las imágenes que se muestran en el PanelImagenes pueden tener zonas transparentes, lo que nos permitirá poner varios controles PanelImagenes que se solapen entre sí y que se vea la parte de las imágenes que quedan por debajo siempre y cuando la de arriba sea transparente.

Por ejemplo supongamos que tenemos estas tres imágenes de la librería de imágenes de Acimut Monitoriza:



Podemos combinarlas poniendo unas delante de las otras para obtener esta otra figura, gracias a que las partes que no forman parte del depósito son transparentes.



En este caso la transparencia está conseguida debido a que las imágenes usadas son vectoriales creadas en formato wmf y solo se ha definido como sólidas las zonas en cuestión.

Si lo que queremos es que una imagen de mapa de bits pueda tener transparencia, deberemos definir la imagen con el canal Alfa con valor 0.

Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Images:** La propiedad ***images*** nos permite definir la colección de imágenes que queremos asociar a la variable para ello usaremos el editor de la colección de Imágenes, en el que iremos definiendo los rangos de variable y las imágenes que asociamos a cada valor

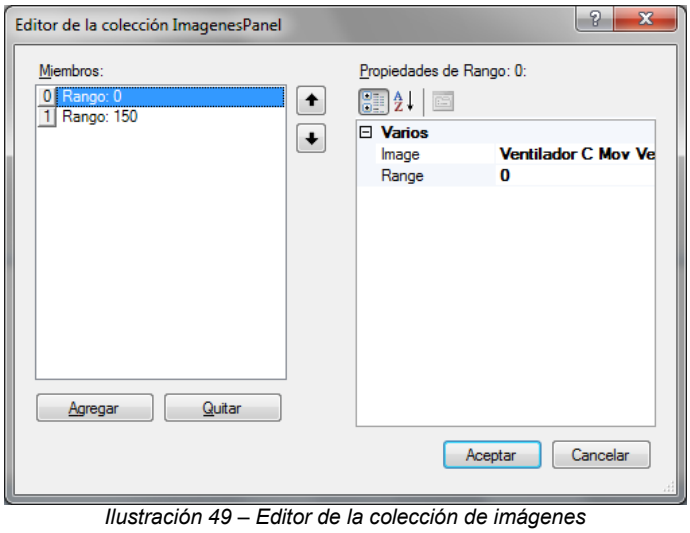


Ilustración 49 – Editor de la colección de imágenes

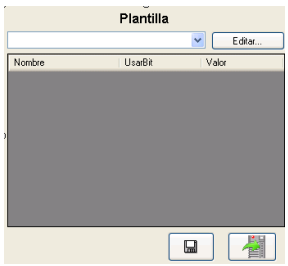
Adicionalmente, las imágenes se ven afectadas por la propiedad **Rotate**, que modifica el aspecto de las imágenes, transformando, si se desea, la forma de representarlas, tal y como se describe en el control BotonEstado.

ProgressBar

La funcionalidad del control ProgressBar es la de representar el valor de una variable, de forma que el tamaño de la barra de progreso es proporcional al valor de la variable.

Las principales propiedades de este control son las siguientes:

- **Server, Group y Variable:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **Maximun:** Límite superior de la escala con la que trabaja el ProgressBar.
- **Minimum:** Límite inferior de la escala con la que trabaja el ProgressBar.



Recipe

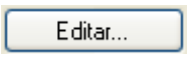
El control Recipe, es el control que permite a los usuarios ver, editar y cargar Recetas en el autómata siempre y cuando tengan permiso para ello.

Para entender el funcionamiento de este control es adecuado completar esta información con la del capítulo dedicado a Recetas y control Batch.

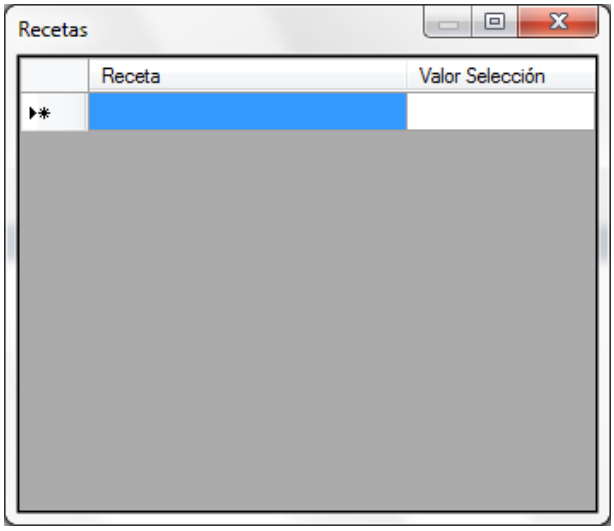
Las propiedades que afectan al funcionamiento de este control son las siguientes:

- **Template:** Aquí se indicará a qué plantilla de recetas hará referencia este control.
- **SendToPLC:** Indica si desde este control se podrá cargar una receta en el PLC. Cuando en ejecución se intenta cargar una receta, si ocurre cualquier problema, o bien, no se cumple con alguno de los requerimientos de la receta algunos elementos del grid mostrarán color rojo, en caso contrario el color será verde.
- **ReadOnly:** Indica si este control permitirá la edición de recetas.

Los botones del control tienen las siguientes funcionalidades:



Editar o añadir recetas. Se mostrará la siguiente pantalla.



Además el campo ValorSeleccion nos permitirá especificará el valor de carga de cada receta en Batch si es necesario.

 Guardar las modificaciones en las recetas. **Importante**: Las recetas las almacena el servidor en un fichero en la misma localización y nombre que el proyecto pero con la extensión *recipes*.

 Cargar la receta seleccionada en el autómata.

En el grid tendremos los registros a los que se podrá, si se dispone de permiso, modificar el valor a cargar en cada uno de los registros. Por otra parte, cuando se intente cargar una receta en el autómata el grid se coloreará en verde si se lleva a cabo con éxito y en rojo si hay algún problema o no se cumplen las condiciones fijadas en la receta.

TecladoNumerico

La función del control TecladoNumerico es la de poder tener un teclado en pantalla en los casos en los que en el sistema en ejecución no se dispone de un teclado físico, bien por razones de seguridad o por estar diseñado para un ambiente industrial y solo se dispone de una pantalla táctil.

El control funciona enviado la tecla que se pulse al control del formulario que tenga en foco.

Mediante el control TecladoNumerico podemos ir cambiando de control a control pulsando la tecla 

.

Las teclas  y  mueven el punto de inserción un carácter a la izquierda y a la derecha respetivamente.

Al pulsar la tecla  se borrará la tecla inmediatamente a la izquierda del punto de inserción.

La tecla  envíala pulsación correspondiente a la tecla Intro/Enter con lo cual en función del control que tenga el foco su funcionalidad será diferente.

Las principales propiedades de este control son las correspondientes a la apariencia visual del control y son las siguientes:

- **BackColor:** Color de fondo del control.
- **ForeColor:** Color del primer plano del control. Utilizado para mostrar el texto de las teclas y el borde de ellas.
- **Font:** Fuente del texto de las teclas.
- **Fill:** Determina como se pinta el fondo de cada tecla. Los posibles valores son: None -  , Horizontal –  , Vertical -  , ForwardDiagonal -  y BackwardDiagonal -  .
- **FillColorStart:** Color de inicio del degradado del fondo de la tecla
- **FillColorEnd:** Color final del degradado del fondo de la tecla.

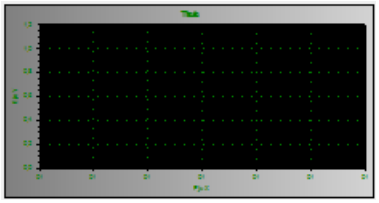
Temporizador

Se utiliza el control ***Temporizador*** para provocar un evento en los intervalos definidos por el usuario.

Acimut Monitoriza soporta extensibilidad del marco de trabajo a través de programación de funciones y librerías en los lenguajes de .Net (Visual Basic y C#) y es dentro de este marco de trabajo donde tiene sentido el temporizador, ya que en él lo que podemos hacer es personalizar el evento Tick del temporizador tal y como se verá en el capitulo Extensibilidad y Programación y en este evento realizar las funciones correspondientes.

Las principales propiedades de este control son las siguientes:

- **Interval:** Frecuencia con que se dispara el evento Tick en milisegundos.
- **Enabled:** Habilita o deshabilita la generación de eventos Tick.



Tendencia

El control ***Tendencia*** sirve para visualizar el valor de una o más variables a lo largo del tiempo.

Pondremos este control en un formulario si queremos que el usuario del scada tenga una pantalla en la que pueda ver cómo evoluciona el conjunto de variables que se haya definido en el diseño.

Si lo que queremos es que en un determinado momento el usuario pueda ver un histórico de datos pudiendo seleccionar la variable y el periodo lo que pondremos es un botón con la propiedad ***Action*** establecida a MostrarGrafica tal y como se ve en el apartado Mostrar Histórico de Datos

Una cosa importante a tener en cuenta cuando usamos un control ***Tendencia*** es que si queremos tener un histórico de valores de las variables deberemos definir acciones de guardado para estas variables tal y como se establece en el apartado Guardar en Base de Datos.

Si no guardamos los valores de las variables en base de datos el control ***Tendencia*** también nos podrá visualizar la evolución de los valores de las variables pero tan solo mostrará los valores recogidos durante la sesión actual, o sea, que empezará con la gráfica vacía sin ningún valor representado y a medida que va pasando el tiempo irá generando la gráfica correspondiente pero nada más cerremos la aplicación estos valores se perderán y volverá a estar vacía la siguiente vez que ejecutemos el scada.

En un mismo gráfico podemos mezclar ambos tipos de variables y hablaremos de variables tipo *AccionGuardado* cuando tenemos un histórico de valores en la base de datos y variables tipo *Dinamica* cuando solo se representan los valores recogidos en la sesión actual.

Las principales propiedades de este control son las siguientes:

- **Frequency:** Frecuencia con que se actualizan los datos en la gráfica, expresada en segundos.
- **Period:** Se establece en minutos y corresponde con el intervalo de tiempo que se muestra en la gráfica. Por ejemplo si se establece un periodo de 100 minutos en la gráfica se visualizaran los valores de las variables correspondientes a los últimos 100 minutos.

- **Series:** Conjunto de variables que queremos representar. Para definir las series se utiliza el Editor de Series que se muestra en la figura siguiente:

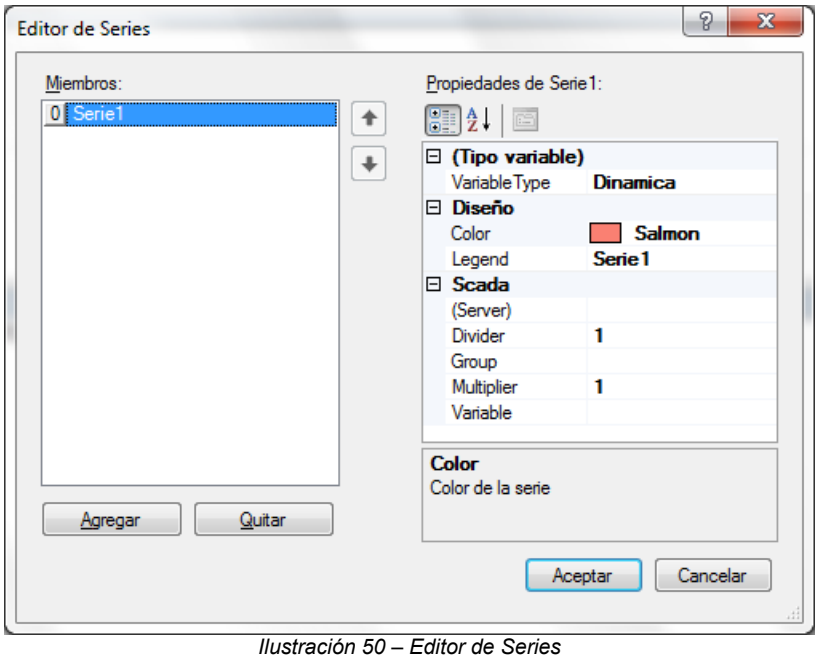


Ilustración 50 – Editor de Series

Para cada serie tenemos que establecer las siguientes propiedades: *VariableType* determina si la variable está guardada en base de datos o no y acepta como posibles valores AccionGuardado y Dinamica, siendo respectivamente cuando disponemos de valores guardados en base de datos y cuando no.

En el caso de que el tipo de variable sea *AccionGuardado* tendremos que establecer las propiedades DataServer, SaveAction y SaveVariables siendo respectivamente el servidor de base de datos, la acción de guardado donde está definida la variable a representar y la variable en concreto que queremos representar.

En el caso que el tipo de variable sea *Dinamica* tendremos que establecer las propiedades Server, Group y Variable correspondiendo al Servidor de comunicaciones con el autómata, el Grupo de variables donde está definida y la variable en concreto que queremos representar.

Para ambos tipos de variables deberemos especificar el color con que queremos graficar la variable mediante la propiedad Color y el texto que aparecerá asociado a la variable en la propiedad Legend.

- **XAxisColor:** Color con que se dibujará el eje abscisas.
- **XAxisText:** Texto que aparecerá junto al eje de abscisas.
- **YAxisColor:** Color con que se dibujará el eje de ordenadas.
- **YAxisText:** Texto que aparecerá junto al eje de ordenadas.
- **GraphicStartColor:** Color inicial del gradiente de colores del fondo del gráfico.
- **GraphicEndColor:** Color final del gradiente de colores del fondo del gráfico.
- **PanelStartColor:** Color inicial del gradiente de colores del fondo del área alrededor del gráfico.
- **PanelEndColor:** Color final del gradiente de colores del fondo del área alrededor del gráfico.
- **Legend:** Se establecerá a True si queremos que se muestren las leyendas asociadas a cada variable, sino se establecerá a False.
- **TitleColor:** Color del texto correspondiente al título.

- **FontTitle:** Fuente que se usará para mostrar el título.
- **TitleText:** Texto del título del gráfico.

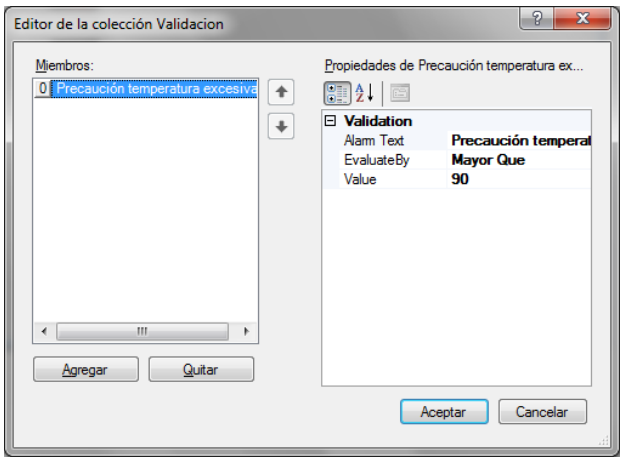
Texto

El control texto posiblemente es el control más usado en cualquier proyecto Scada, su objetivo es tanto poder visualizar valores de variables como poder editar y modificar esos valores.

La edición de un valor se finaliza por la pulsación de la tecla *Intro*  , hasta que no se pulsa Intro no se trasmite el nuevo valor al autómata.

Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Format:** Muestra el valor según el formato especificado por la propiedad, por ejemplo un formato igual a ##.00 hará que el la variable se muestre con dos decimales.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Text:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **ReadOnly:** Controla si se puede cambiar el valor en el control de edición. Si se establece a True solo se podrá visualizar valores pero no se podrán modificar.
- **NumberingSystem:** Muestra el valor de la variable en el sistema de numeración especificado. Los posibles valores son Decimal y Hexadecimal.
- **Validations:** La propiedad Validations nos permite definir un conjunto de condiciones, sobre el valor de la variable a la que esté vinculado el control, de forma que si en la ejecución del proyecto se cumple alguna de esas condiciones se mostrará un aviso junto el control. Para definir las validaciones se usa el Editor de la colección de validaciones en el que se introduce para cada condición que queremos definir las siguientes propiedades: A*larma Text*, *EvaluateBy* y *Value*, donde *Alarm Text* es el texto que aparecerá si se cumple la condición, *EvaluateBy* es el tipo de condición que queremos establecer, siendo los posibles valores *Igual, MayorQue, MenorQue y Distinto* y *Value* es el valor con que queremos comparar.



TextoConMascara

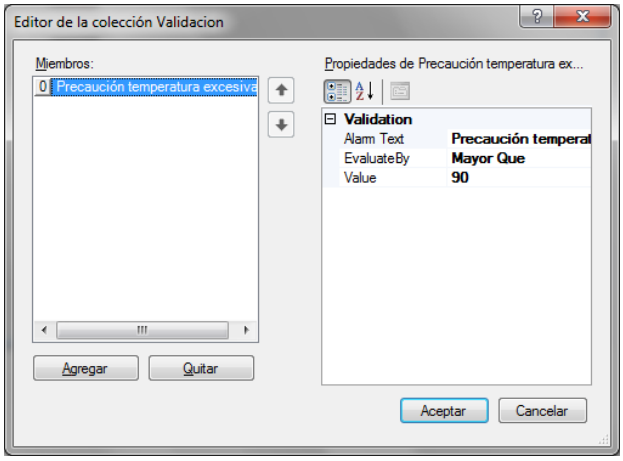
El control TextoConMascara es muy parecido al control Texto, la única diferencia es que nos permite definir una máscara de edición del texto, por ejemplo, si estamos mostrando una variable que representa información horaria podemos usar una máscara como __:__ en la que separamos las horas de los minutos.

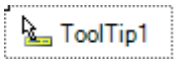
Al igual que con el control Texto con el control TextoConMascara podemos tanto visualizar valores de variables como editarlos y modificarlos.

La edición de un valor se finaliza por la pulsación de la tecla *Intro*  , hasta que no se pulsa Intro no se trasmite el nuevo valor al autómata.

Las principales propiedades de este control son las siguientes:

- **Server, Group, Variable y UseBit:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado y adicionalmente establecer la propiedad UseBit si queremos referirnos a un determinado bit en concreto.
- **Mask:** Establece la cadena que controla la entrada permitida.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Format:** Muestra el valor según el formato especificado por la propiedad, por ejemplo un formato igual a ##.00 hará que el la variable se muestre con dos decimales.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Text:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **ReadOnly:** Controla si se puede cambiar el valor en el control de edición. Si se establece a True solo se podrá visualizar valores pero no se podrán modificar.
- **NumberingSystem:** Muestra el valor de la variable en el sistema de numeración especificado. Los posibles valores son Decimal y Hexadecimal.
- **Validations:** La propiedad Validations nos permite definir un conjunto de condiciones, sobre el valor de la variable a la que esté vinculado el control, de forma que si en la ejecución del proyecto se cumple alguna de esas condiciones se mostrará un aviso junto el control. Para definir las validaciones se usa el Editor de la colección de validaciones en el que se introduce para cada condición que queremos definir las siguientes propiedades: A*larm Text*, *EvaluateBy* y *Value*, donde *Alarm Text* es el texto que aparecerá si se cumple la condición, *EvaluateBy* es el tipo de condición que queremos establecer, siendo los posibles valores *Igual, MayorQue, MenorQue y Distinto* y *Value* es el valor con que queremos comparar.





ToolTip

Se utiliza el control ***ToolTip*** para mostrar una ventana de información cuando tenemos el ratón sobre un determinado control.

La principal propiedad de este control realmente no se asigna en las propiedades de este control, sino que simplemente por la presencia de un control ToolTip sobre el formulario todos los controles que contenga ese formulario adquieren una propiedad más que es la propiedad ToolTip.

Es en esta propiedad ToolTip de cada uno de los controles donde introduciremos el texto que queremos mostrar como ayuda o información del control correspondiente.

TrackBar

El control ***TrackBar*** representa una barra de seguimiento estándar de Windows.

Para configurar los intervalos entre los que se desplaza el valor de la propiedad *Value* de una barra de seguimiento, deberá establecer la propiedad *Minimum* para especificar el extremo inferior del intervalo y la propiedad *Maximum* para especificar el extremo superior del intervalo.

Las principales propiedades de este control son las siguientes:

- **Server, Group y Variable:** Nos servirán para poder hacer referencia a la variable del autómata en función del servidor de comunicaciones, el grupo en el que está definida y el nombre simbólico que le hayamos dado.
- **Divider:** Muestra el valor de la variable dividido por el valor de la propiedad.
- **Multiplier:** Muestra el valor de la variable multiplicado por el valor de la propiedad.
- **Value:** Valor de la variable asociada con las propiedades Server, Group y Variable.
- **ReadOnly:** Controla si se puede cambiar el valor de la variable. Si se establece a True solo se podrá visualizar valores pero no se podrán modificar.
- **Minimun:** Extremo inferior del intervalo de la escala de valores.
- **Maximun:** Extremo superior del intervalo de la escala de valores.

Solapa Diseño

A partir de aquí se van a describir los controles pertenecientes a la solapa Diseño de la ToolBox de Acimut Monitoriza.

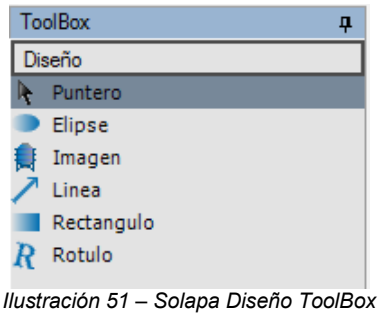


Ilustración 51 – Solapa Diseño ToolBox

Estos controles tienen fundamentalmente una función de diseño de muestro formulario y no están asociados en ningún caso a la visualización de valores del Scada.

Todos los controles de esta solapan están diseñados para que dispongan de transparencia tanto visual como a los clics del ratón. O sea, que los controles no ocupan una región rectangular del formulario, tal y como es normal en Windows, sino que en aquellas regiones en las que el control no dibuja nada, se muestra totalmente cualquier control que esté por debajo de éste, de forma que se encuentren solapados y no solo se muestra sino si pulsamos con el ratón la pulsación le llegará al control de abajo. Por ejemplo, podemos tener un control ***Boton*** que tenga encima un control ***Rotulo***, tal y como se muestra en la siguiente imagen.

En este caso el control ***Rotulo***, a pesar de ocupar la misma área rectangular del formulario, nos dejar ver el control ***Boton*** que hay detrás y si pulsamos con el ratón por ejemplo en el centro de la O de la palabra Rotulo el clic se realiza sobre el control Boton.

Elipse

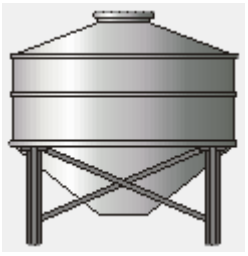
El control ***Elipse*** nos permite dibujar Elipses y Círculos

Las principales propiedades de este control son las siguientes:

- **BackColor:** Color del relleno de la elipse si la propiedad *Fill* está establecida a *None*.
- **ForeColor:** Color de la línea que forma la elipse. Si la propiedad Fill está establecida a Solid se utilizará este color como color del relleno de la elipse.
- **Fill:** Tipo de relleno de la elipse. Los posibles valores son: *None, Solid, Horizontal, Vertical, ForwardDiagonal y BackwardDiagona*l. Las opciones *None* y *Solid* producen un relleno con un color sólido utilizando respectivamente el color BackColor y

ForeColor para rellenar. Las opciones *Horizontal, Vertical, ForwardDiagonal y BackwardDiagonal* rellenan mediante un degradado de color en la dirección especificada por la opción.

- **FillColorStart:** Color de inicio del degradado del relleno.
- **FillColorEnd:** Color de fin del degradado del relleno.
- **Thickness:** Grosor de la línea que dibuja la elipse.



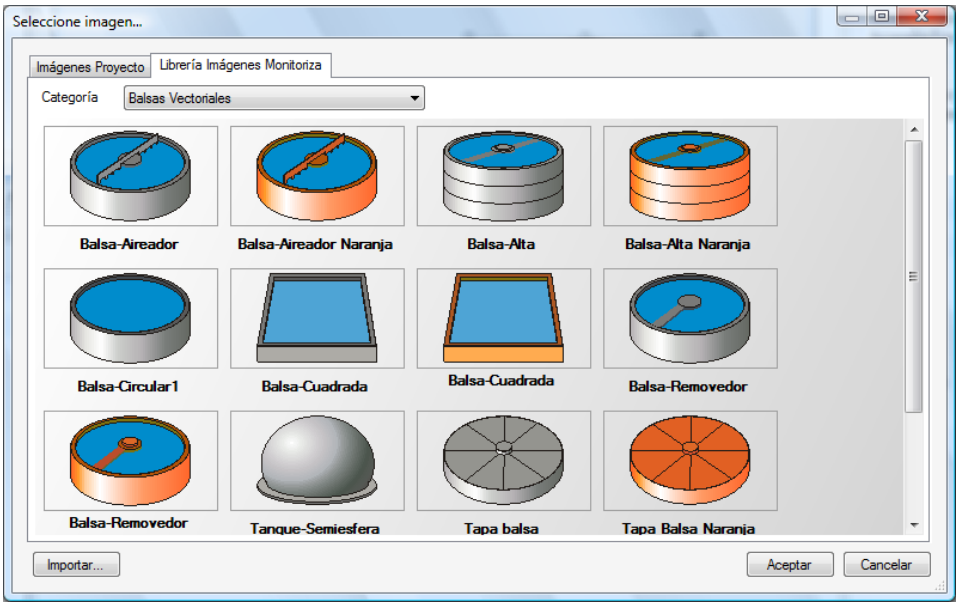
Imagen

El control ***Imagen*** nos sirve para visualizar una imagen en nuestro formulario.

La principal propiedad en el control ***Imagen*** es la propiedad *Imagen*. Mediante esta propiedad establecemos que imagen queremos visualizar.

Acimut Monitoriza dispone de una gran variedad de imágenes predefinidas gracias a su Librería de imágenes que nos permite realizar unos diseños atractivos.

Algunas de estas imágenes son las que se muestran en la figura siguiente:



En el control **Imagen** cuando cambiamos el tamaño del control, la imagen se reajusta adecuadamente al nuevo tamaño. Este reajuste es más perfecto si la imagen que hemos utilizado es vectorial en vez de mapa de bits. En la Librería de Imágenes de Acimut Monitoriza se proporcionan imágenes de los dos tipos.

Igualmente se proporcionan imágenes con movimiento, por ejemplo en la siguiente figura

Se dispone de tres versiones del mismo ventilador la primera corresponde con una imagen que tiene movimiento mientras que la última corresponde con una imagen parada. Cuando seleccionamos una imagen de la Librería de imágenes las imágenes que están animadas se muestran en movimiento para que quede clara su funcionalidad.

Adicionalmente, las imágenes se ven afectadas por la propiedad **Rotate**, que modifica el aspecto de las imágenes, transformando si se desea la forma de representarlas, tal y como se describe en el control BotonEstado.

Linea

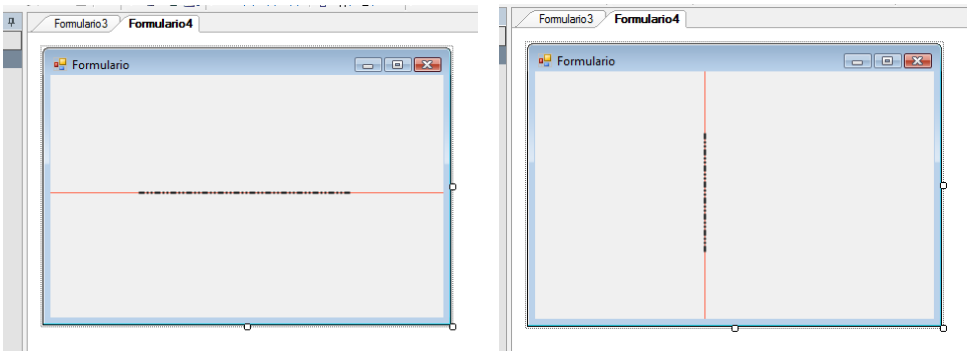
El control **Linea** nos sirve para trazar una línea sobre la superficie del formulario o dentro de cualquier otro control que sea contenedor de controles.

Cuando en el Editor seleccionamos un control Linea tal y como se muestra en la figura siguiente:

Aparte del marco identificativo de que el control está seleccionado nos aparecen dos puntos de color rojo. Estos dos puntos que llamaremos manejadores nos sirven para arrastrar el inicio o el fin de la línea a otra posición. Para ello simplemente deberemos seleccionar el manejador con el ratón y arrastrarlo a otra posición, en el momento que soltemos se nos dibujará la línea con el inicio o el fin en la posición final del ratón en función del manejador que hayamos seleccionado.

Lo mismo podríamos haberlo conseguido modificando en la ventana de propiedades los valores de las propiedades *PuntoInicial* o *PuntoFinal* pero el poderlo hacer mediante los manejadores facilita mucho el trabajo de diseño.

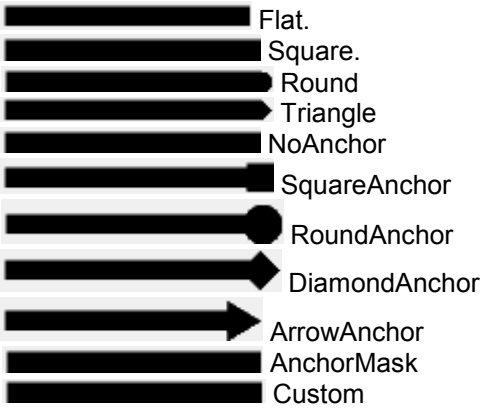
Otra herramienta útil a la hora de trazar una línea son los indicadores de verticalidad y horizontalidad, estos indicadores se muestran cuando estamos moviendo el punto inicial o final de una línea mediante sus manejadores y la línea se encuentra paralela a alguno de los ejes de coordenadas tal y como se puede ver en las siguientes imágenes.



Demostración: http://www.acimut.com/monitoriza/videos/mover_linea/default.html

Las principales propiedades de este control son las siguientes:

- **ForeColor:** Color de la línea si la propiedad Fill está establecida a Solid.
- **Fill:** Tipo de relleno de la línea. Los posibles valores son: *Solid, Horizontal, Vertical, ForwardDiagonal y BackwardDiagonal*. La opción *Solid* produce un relleno con un color sólido utilizando el color establecido en ForeColor. Las opciones *Horizontal, Vertical, ForwardDiagonal y BackwardDiagonal* rellenan mediante un degradado de color en la dirección especificada por la opción.
- **FillColorStart:** Color de inicio del degradado del relleno.
- **FillColorEnd:** Color de fin del degradado del relleno.
- **Thickness:** Grosor de la línea.
- **StartPoint:** Coordenadas del punto inicial de la línea.
- **EndPoint:** Coordenadas del punto final de la línea.
- **StartCap y EndCap:** Establecen las formas del inicio y fin de la línea respectivamente. Los posibles valores son:



Rectangulo

El control ***Rectangulo*** nos permite dibujar rectángulos que podemos rellenar con degradados.

Las principales propiedades de este control son las siguientes:

- **BackColor:** Color del relleno del rectángulo si la propiedad *Fill* está establecida a *None*.
- **ForeColor:** Color de la línea que forma el rectángulo. Si la propiedad Fill está establecida a Solid se utilizará este color como color del relleno del rectangulo.
- **Fill:** Tipo de relleno del rectángulo. Los posibles valores son: *None, Solid, Horizontal, Vertical, ForwardDiagonal y BackwardDiagonal*. Las opciones *None* y *Solid* producen un relleno con un color sólido utilizando respectivamente el color BackColor y ForeColor para rellenar. Las opciones *Horizontal, Vertical, ForwardDiagonal y BackwardDiagonal* rellenan mediante un degradado de color en la dirección especificada por la opción.
- **FillColorStart:** Color de inicio del degradado del relleno.
- **FillColorEnd:** Color de fin del degradado del relleno.
- **Thickness:** Grosor de la línea que dibuja la elipse.
- **CornerRadius:** Radio de las esquinas redondeadas.

Rotulo

El control ***Rotulo*** proporciona la posibilidad de mostrar textos orientados en cualquier dirección y con un relleno de las letras que puede ser tanto de color sólido como con un degradado.

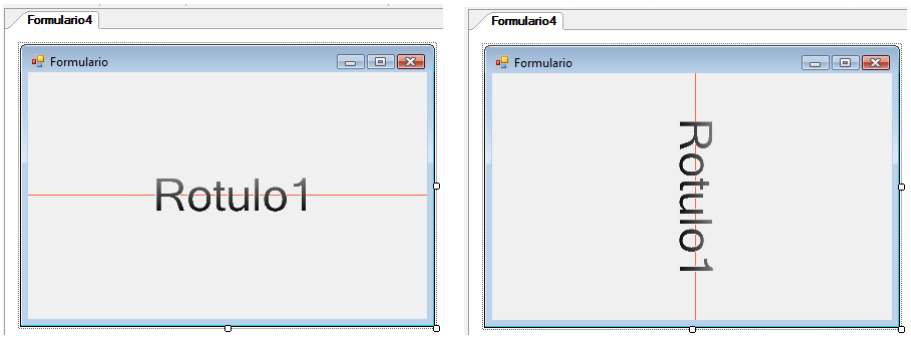
Para girar el Rótulo en cualquier dirección tenemos dos posibilidades establecer la propiedad *Angulo* al valor que deseemos o bien utilizar el Editor de Formularios, es decir, al igual que con el control ***Linea*** cuando seleccionamos un control ***Rotulo***, aparte del marco identificativo de que el control está seleccionado nos aparece un punto de color rojo que denominaremos manejador.

Si arrastramos este manejador en el momento que soltemos se nos dibujará el rótulo en la dirección especificada por el manejador.

El poder girar el rótulo mediante el manejador nos permite frente a establecer la propiedad *Angulo* facilita mucho la labor de diseño.

Otra herramienta útil a la hora de trazar un rótulo son los indicadores de verticalidad y horizontalidad, estos indicadores se muestran cuando estamos moviendo el rótulo

mediante su manejador y el rótulo se encuentra paralelo a alguno de los ejes de coordenadas tal y como se puede ver en las siguientes imágenes.



Demostración: http://www.acimut.com/monitoriza/videos/mover_rotulo/default.html

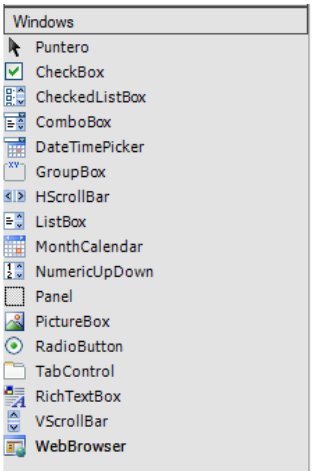
Con el control ***Rotulo*** hay que tener en cuenta que cuando intentamos seleccionarlo con el ratón el control solo existe sobre el perfil que forman las letras, o sea, que si intentamos seleccionarlo pulsando sobre el centro de por ejemplo una O no se nos seleccionará ya que debemos pulsar dentro del contorno de la letra.

Las principales propiedades de este control son las siguientes:

- **BackColor:** Color del relleno de las letras del rótulo si la propiedad *Fill* está establecida a *None*.
- **ForeColor:** Color de la línea que forma el contorno de las letras del rótulo.
- **Fill:** Tipo de relleno de las letras del rótulo. Los posibles valores son: *None, Solid, Horizontal, Vertical, ForwardDiagonal y BackwardDiagona*l. La opción None produce que el relleno de las letras sea con el color establecido en la propiedad BackColor. La opción *Solid* produce que el relleno de las letras sea con el color establecido en la propiedad ForeColor. Las opciones *Horizontal, Vertical, ForwardDiagonal y BackwardDiagona*l rellenan mediante un degradado de color en la dirección especificada por la opción.
- **FillColorStart:** Color de inicio del degradado del relleno.
- **FillColorEnd:** Color de fin del degradado del relleno.
- **Thickness:** Grosor de la línea.
- **Angle:** Angulo de giro del rótulo.

Solapa Windows

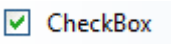
A partir de aquí se van a describir los controles pertenecientes a la solapa Windows de la ToolBox de Acimut Monitoriza.



En esta solapa se muestran controles estándar de Windows y que por tanto no se pueden asociar a las variables de Acimut Monitoriza ya que no poseen las propiedades Server, Group y Variable y por tanto no pueden representar los valores asociados a estas.

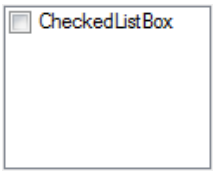
Su inclusión en la ToolBox se debe a que a través de la Programación y Extensibilidad de Acimut Monitoriza y en particular a través de la tecnología de “CodeBinding”, que si que se ha incorporado a estos controles, sí que se puede dar funcionalidad adicional a la proporcionada por los controles de la solapa Scada.

A continuación se describe brevemente cada uno de ellos



CheckBox

El control CheckBox se usa para facilitar a los usuarios la elección de valores Verdadero/Falso.

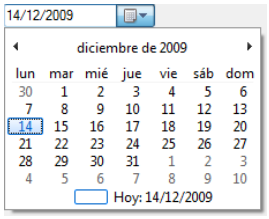


CheckedListBox

El control CheckedListBox permite mostrar una lista de ítems con un CheckBox asociado a cada uno de ellos para permitir definir los ítems seleccionados.

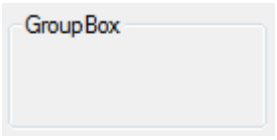
ComboBox

El control ComboBox permite mostrar una lista de ítems desplegable. El ComboBox consta de dos partes una parte siempre visible que permite al usuario escribir como en un TextBox y una segunda parte que es un ListBox que normalmente está oculta pero que podemos desplegar para seleccionar un ítem de la lista.



DateTimerPicker

El control DateTimerPicker le permite al usuario seleccionar una determinada fecha u hora.



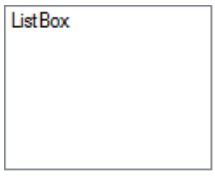
GroupBox

El control GroupBox se usa para agrupar otros controles. Hay dos razones principales por las que podemos querer agrupar los controles:

- Crear un agrupamiento visual de los controles para crear una interfaz de usuario más clara.
- Crear un agrupamiento de funcionalidad como ocurre cuando incluimos varios RadioButton que al estar dentro de un GroupBox solo se va a poder seleccionar uno de ellos.

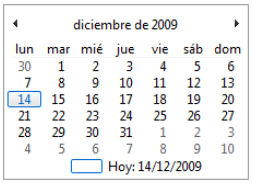
HScrollBar

El control HScrollBar permite diseñar una navegación sencilla cuando tenemos un gran conjunto de ítems de información que queramos mostrar.



ListBox

El control ListBox nos sirve para mostrar una lista de ítems de los cuales el usuario puede seleccionar uno o varios.



MonthCalendar

El control MonthCalendar permite al usuario seleccionar una fecha mostrándole un calendario completo por el que puede ir navegando por los distintos meses o años.

NumericUpDown

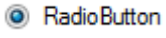
El control NumericUpDown es la combinación de un control TextBox y un par de botones que permiten incrementar o disminuir el valor mostrado.

Panel

El control Panel nos sirve para agrupar visualmente otros controles y así poder hacer un diseño más claro y visual de nuestro Scada. Por ejemplo estableciendo el color del fondo del Panel a colores distintos según el significado de lo que estemos representando.

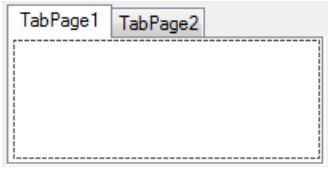
PictureBox

El control PictureBox nos permite mostrar una imagen. Mediante los controles Imagen y PanelImagenes de Acimut Monitoriza se puede conseguir una funcionalidad más avanzada ya que desde ellos se puede utilizar la librería de imágenes de Monitoriza.



RadioButton

El control RadioButton permite al usuario seleccionar una opción dentro de un grupo de opciones cuando está emparejado con otro u otros RadioButtons.



TabControl

El control TabControl muestra múltiples fichas, similares a los divisores de un cuaderno o a las etiquetas de las carpetas de un archivador. Las fichas pueden contener imágenes y otros controles.

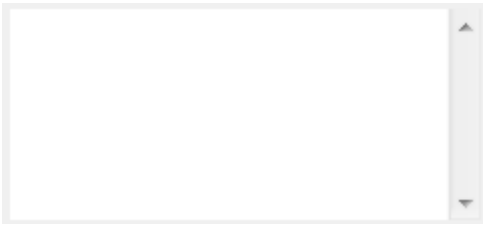
La propiedad más importante de TabControl es TabPages, que contiene las fichas individuales. Cada ficha individual es un objeto TabPage.

RichTextBox

El control RichTextBox permite mostrar texto con formato, o sea, con distintas fuentes y estilos.

VScrollBar

El control VScrollBar diseñar una navegación sencilla cuando tenemos un gran conjunto de ítems de información que queramos mostrar.



WebBrowser

El control WebBrowser permite que el usuario explore páginas Web.

Para su utilización deberemos especificar en la propiedad ***Url*** una dirección web válida, por ejemplo podemos poner en la propiedad ***Url*** el valor http://www.acimut.es

PROBAR EL PROYECTO

Mientras que estamos desarrollando un proyecto scada con Acimut Monitoriza tenemos dos formas de probar nuestro proyecto.

Si en nuestro entorno de desarrollo no tenemos posibilidad de tener acceso a los autómatas o dispositivos que queremos monitorizar, podemos probar el proyecto en modo *Simulación*.

Si por el contrario podemos acceder a los dispositivos probaremos el proyecto en modo *Servidor*.

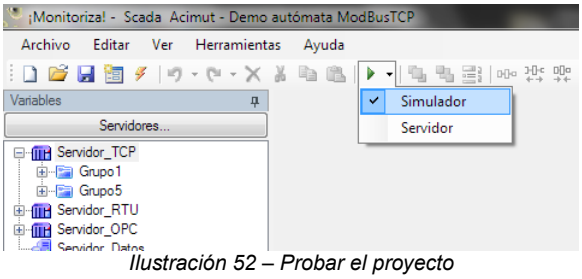


Ilustración 52 – Probar el proyecto

Al elegir la opción *Servidor* lo que hace es ejecutar el Servidor de Acimut Monitoriza que se encarga de las comunicaciones con los dispositivos y los autómatas y a continuación ejecutar el cliente de scada que nos muestra la interfaz de formularios que hayamos diseñado.

Mediante esta forma de probar nos acercamos lo más posible a lo que será el entorno de producción ya que el servidor se conecta de forma real a los dispositivos, la diferencia la tenemos en que en un entorno de producción, lo normal es que el servidor esté alojado en un ordenador y el cliente en otro, normalmente en un puesto de control o monitorización. Mientras que cuando lo ejecutamos desde el entorno de desarrollo va a estar todo alojado en una misma máquina, el editor, el servidor y el cliente.

Una vez hemos elegido una opción (Servidor o Simulador) el editor de proyectos recuerda la opción elegida y podremos simplemente pulsar sobre el botón Play para volver a ejecutarlo sin necesidad de desplegar el menú de opciones.

Si elegimos la opción de Simulación es porque no tenemos acceso a los dispositivos, o bien simplemente queremos hacer una prueba de navegabilidad entre los formularios que componen la interfaz de usuario.

Por tanto como en simulación no hay conexión con los autómatas no podemos tener valores reales de las variables.

Como esta es una situación bastante normal Acimut Monitoriza lo que nos proporciona es un formulario de simulación de las variables, en este formulario podemos ir introduciendo valores a las variables para probar por ejemplo si una alarma se dispara como teníamos previsto o no.

Además a la hora de definir las variables vimos que se dispone de un apartado dentro de cada variable que es simulación.

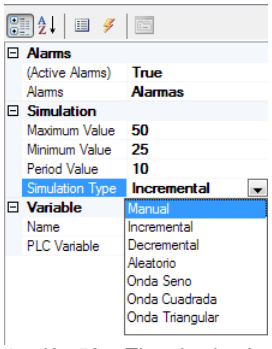


Ilustración 53 – Tipo de simulación

Si elegimos un tipo diferente a Manual tendremos que la variable irá variando su valor automáticamente.

También podemos en la simulación, en los controles de entrada, como por ejemplo el control Texto introducir valores a las variables y por tanto probar el Scada como si estuviéramos recibiendo datos.

Mediante la simulación también podemos probar los permisos que se hayan establecido para cada uno de los usuarios definidos.

Al iniciar la simulación se nos pedirá que guardemos el proyecto si no lo hemos hecho anteriormente y pondrá en marcha la simulación.

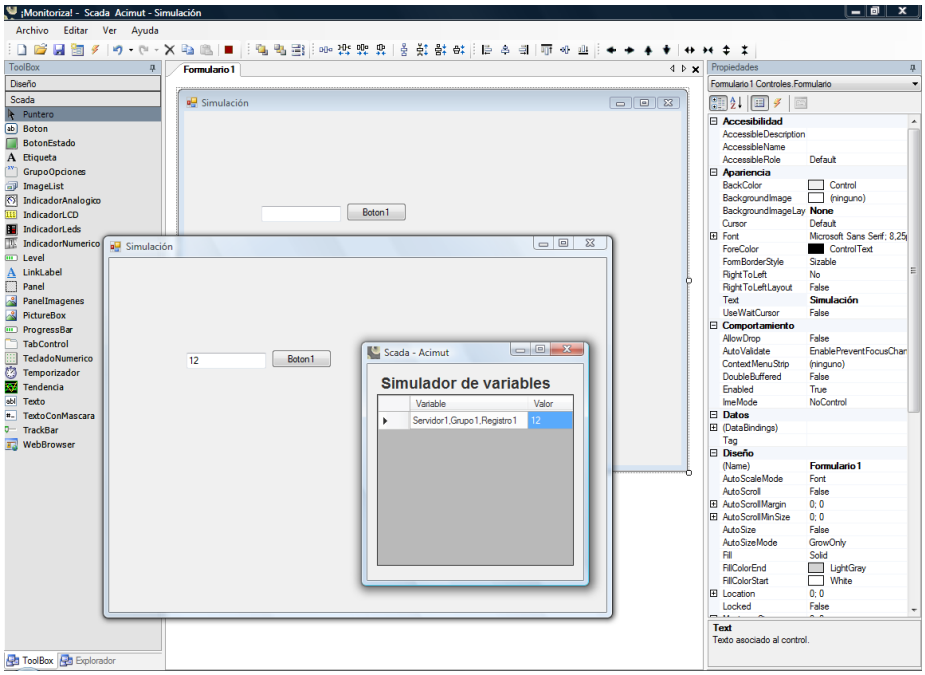


Ilustración 54 – Simulación de ejecución

En la imagen anterior se puede ver el Editor de Monitoriza y el formulario que estamos diseñando en ejecución.

En el formulario se ha añadido un control Texto que se ha relacionado con la variable Registro1 del Grupo1 del Servidor1, por lo tanto cuando en la ventana del simulador de variables se introduce un valor en esa variable este se muestra inmediatamente en el control texto asociado a la variable.

Para finalizar la simulación podemos bien cerrar la ventana del simulador de variables o cerrar todos los formularios del proyecto que hayamos abierto o bien volver a pulsar sobre el botón Play que en este caso será en cuadrado rojo en vez de un triangulo verde para indicarnos que podemos parar la ejecución.

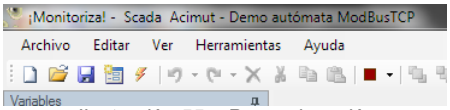


Ilustración 55 – Parar ejecución

Demostración: http://www.acimut.com/monitoriza/videos/simulacion/default.html

LIBRERÍA DE IMÁGENES

Una de las características de Acimut Monitoriza que más ayuda en el diseño de formularos es la Librería de Imágenes.

La Librería de Imágenes consiste por una parte en un conjunto de imágenes prediseñadas por Acimut que se pueden utilizar libremente en cualquier proyecto que desarrollemos y por otra en la posibilidad de crear nuestra propia librería de imágenes del proyecto de forma que podamos tener categorizadas y fácilmente disponibles las imágenes que vayamos a usar.

Otra ventaja de la Librería de Imágenes es que si utilizamos las imágenes de la Librería, solo existe en el proyecto una única instancia de la imagen, independientemente de las veces que la utilicemos, lo cual hace que el tamaño de nuestros proyectos no se incremente innecesariamente.

Las imágenes de la Librería prediseñadas por Acimut forman parte de la instalación y por tanto no incrementan el tamaño de nuestro proyecto, simplemente se hace una referencia a la imagen.

Los controles de Acimut Monitoriza que pueden utilizar la Librería de Imágenes son:

- ***BotonEstado***
- ***PanelImagenes***
- ***Imagen***

Al establecer la propiedad *Imagen* de la colección de imágenes de ***BotonEstado*** y ***PanelImagenes*** o la propiedad *Imagen* en el control ***Imagen*** nos aparece una ventana como la siguiente:

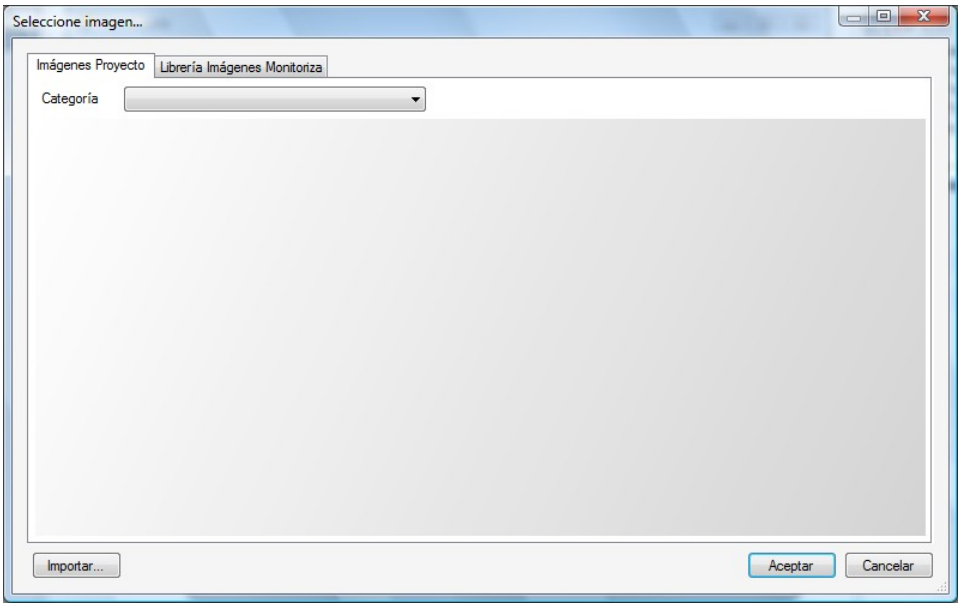


Ilustración 56 – Seleccionar imagen de la Librería

Que dispone de dos solapas para determinar el origen de la imagen. La solapa *Imágenes Proyecto* en la que figuraran todas las imágenes que hayamos agregado al proyecto y la solapa *Librería Imágenes Monitoriza* en la que aparecen las imágenes prediseñadas por Acimut.

En la misma ventana también hay un botón rotulado **Importar** que sirve para asignar al control una imagen que disponemos pero que no tenemos incorporada a la Librería. Las imágenes que incorporamos al control mediante el botón Importar si que incrementan el tamaño del proyecto en función del número de veces que las utilicemos.

Para crear la librería de imágenes del proyecto seleccionaremos la opción del menú *Librería de Imágenes* del menú *Ver* tal y como se muestra a continuación.

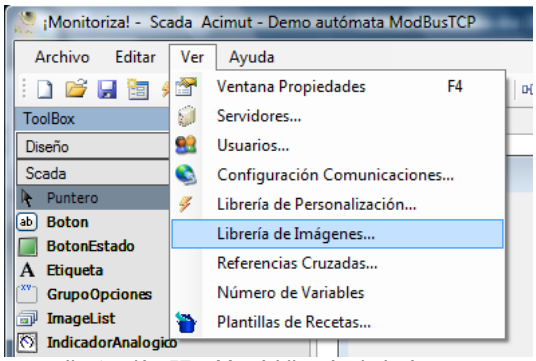


Ilustración 57 – Menú Librería de Imágenes

Con lo cual aparecerá la siguiente pantalla

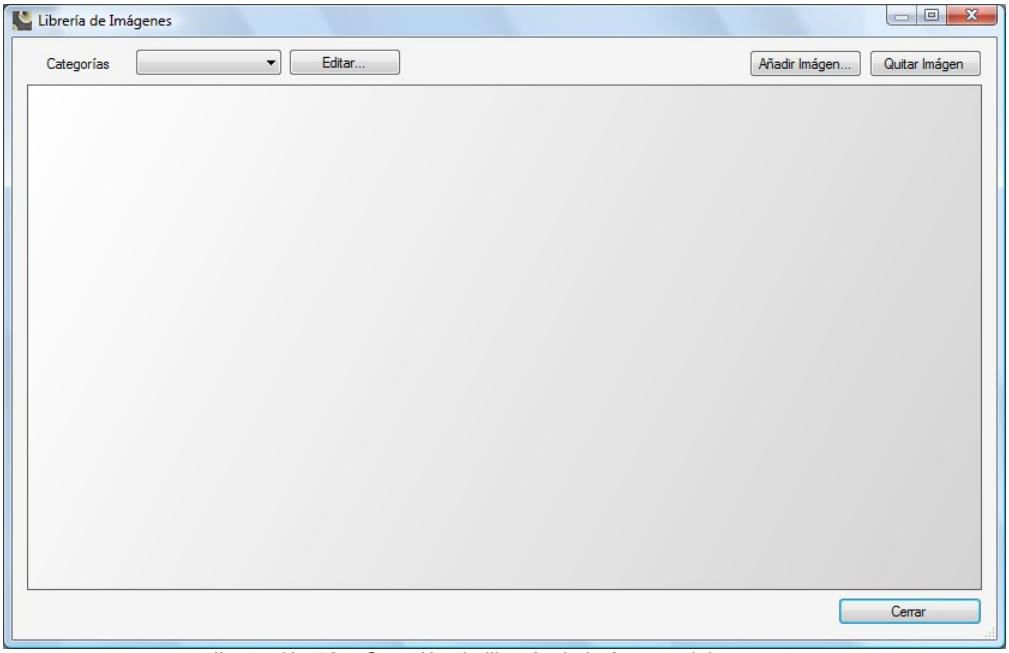


Ilustración 58 – Creación de librería de imágenes del proyecto

En la que lo primero que tenemos que hacer es definir las categorías en las que queremos clasificar nuestras imágenes. Al menos deberemos definir una categoría.

Para definir las categorías pulsamos el botón **Editar** y nos aparece la ventana de categorías

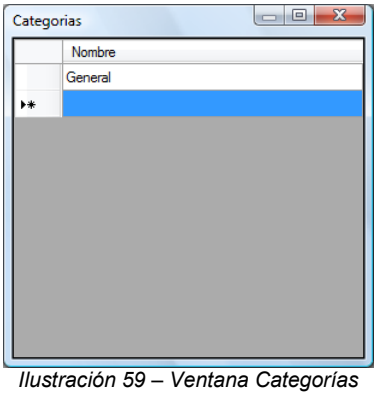
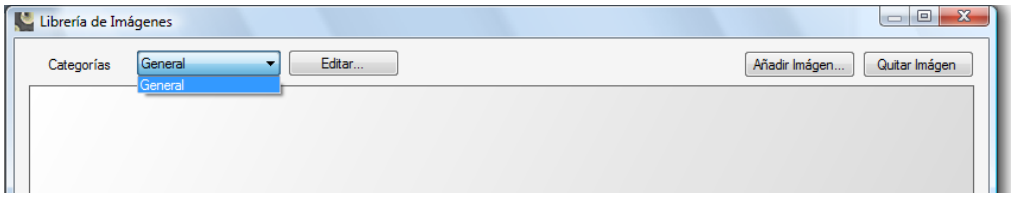


Ilustración 59 – Ventana Categorías

En la que introducimos cada una de las categorías y pulsamos el botón de cerrar la ventana. A partir de ahí en el desplegable de categorías deberemos seleccionar la categoría a la que queremos asignar la imagen.

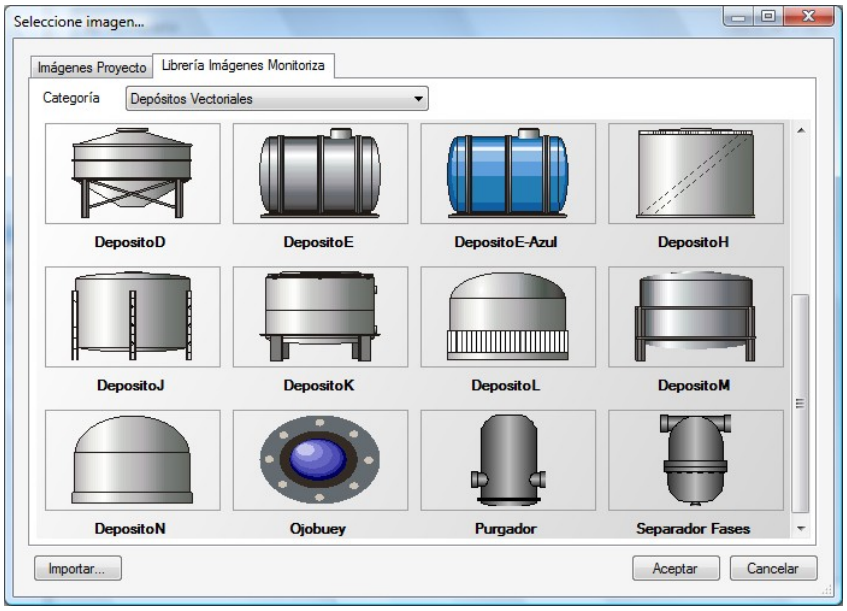


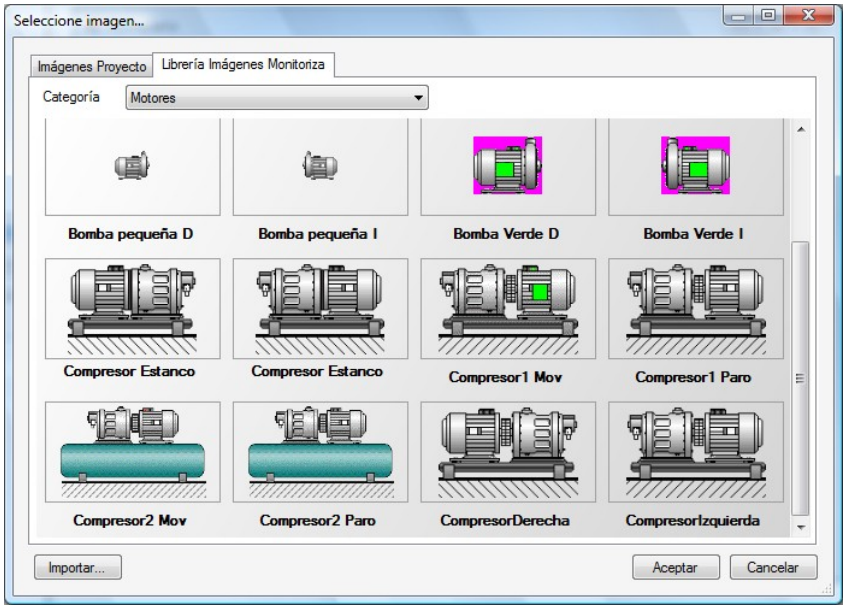
Una vez seleccionada disponemos del botón **Añadir imagen** para seleccionar la imagen que queremos añadir y el botón **Quitar imagen** para suprimir una imagen que teníamos añadida a la Librería del proyecto.

El tipo de imágenes que acepta son JPG y BMP como imágenes raster sin animación, GIF como imagen raster con animación y WMF como imagen vectorial.

La Librería de Imágenes de Acimut no es modificable, o sea, no se pueden añadir ni quitar imágenes de la colección de imágenes que la componen.

Algunos ejemplos de la librería de imágenes de Acimut son los siguientes:





USUARIOS Y PERMISOS

Para editar usuarios de la aplicación se dispone de la opción de menú *Usuarios y permisos* del menú *Ver*.

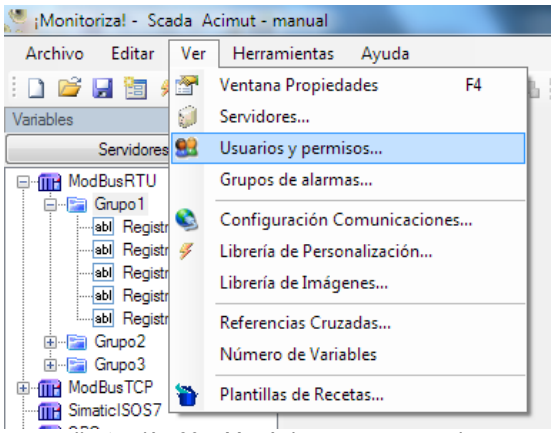


Ilustración 60 – Menú de acceso a usuarios

Esta opción de menú nos lleva a la siguiente pantalla

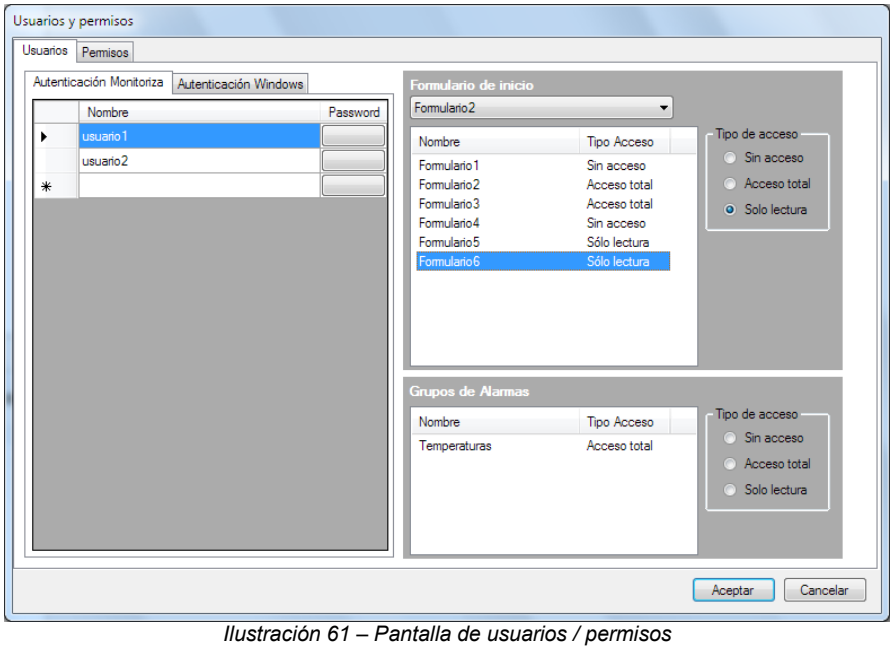


Ilustración 61 – Pantalla de usuarios / permisos

La pantalla se divide en dos solapas correspondientes la primera a la definición de *Usuarios* del Scada y la segunda a los *Permisos* desde un punto de vista global del proyecto.

En la solapa de *Usuarios* daremos de alta a los usuarios necesarios, especificaremos su contraseña y como podemos ver asignaremos el formulario de inicio y los permisos para cada formulario de la aplicación SCADA.

Como podemos ver se definen dos tipos de usuarios o dicho de otra forma dos tipos de autenticación de los usuarios. Por una parte tenemos la *Autenticación Monitoriza*, que es propia del proyecto Scada en cuestión y por otra, la *Autenticación de Windows* en la que especificaremos los usuarios del directorio activo (Dominio) de Windows que queramos que puedan usar el proyecto Scada.

Dentro de un mismo proyecto se pueden usar y crear indistintamente usuarios con autenticación Monitoriza o Windows; las políticas de seguridad establecidas en el entorno determinarán que tipo de autenticación es más adecuado.

Al definir un usuario con Autenticación Monitoriza deberemos especificar el nombre y la contraseña, mientras que al definir un usuario con Autenticación Windows simplemente tendremos que introducir el login de Windows, en el formato dominio\usuario tal y como figure en el directorio activo y la contraseña será la que se haya especificado en él para ese usuario, de forma que cuando se le requiera autenticarse en la aplicación deberá introducir el usuario y contraseña de Windows tal y como lo hace al iniciar sesión de Windows.

Sea un tipo de autenticación u otra deberemos para cada usuario especificar el formulario de inicio y los permisos para cada uno de los formularios.

Hay que tener en cuenta que si a un usuario le asignamos permiso de *Acceso total* en un formulario, el usuario podrá modificar valores de variables, a no ser que el control en cuestión lo impida específicamente; si le asignamos un permiso de *Sólo lectura,* independientemente de lo que marque la propiedad *Read only* de los controles, el usuario no podrá modificar valores de variables.

También especificaremos los permisos para los grupos de alarmas.

Los grupos de alarmas son una forma de agrupar conjuntos de alarmas en función de a qué usuarios van dirigidas esas alarmas. Por ejemplo, si tenemos dos puntos de control en una instalación que monitorizan procesos diferentes, para cada una de las alarmas que definamos, la asignaremos a un grupo mediante la propiedad AlarmGroup de la alarma. De esta forma en la ventana de permisos estableceremos para cada usuario si se le mostrará la alarma o no y si la puede validar o no.

Por defecto, todos los usuarios tienen acceso total a todos los grupos de alarmas, o sea, que si queremos restringir el acceso deberemos especificarlo.

Los tipos posibles de acceso son:

Sin Acceso: Al usuario no se le notifican las alarmas.
Solo Lectura: Al usuario se le notifica la alarma pero no puede validarla.
Acceso total: El usuario puede ver y validar la alarma.

La solapa Permisos de la ventana de usuarios es tal y como se muestra a continuación:

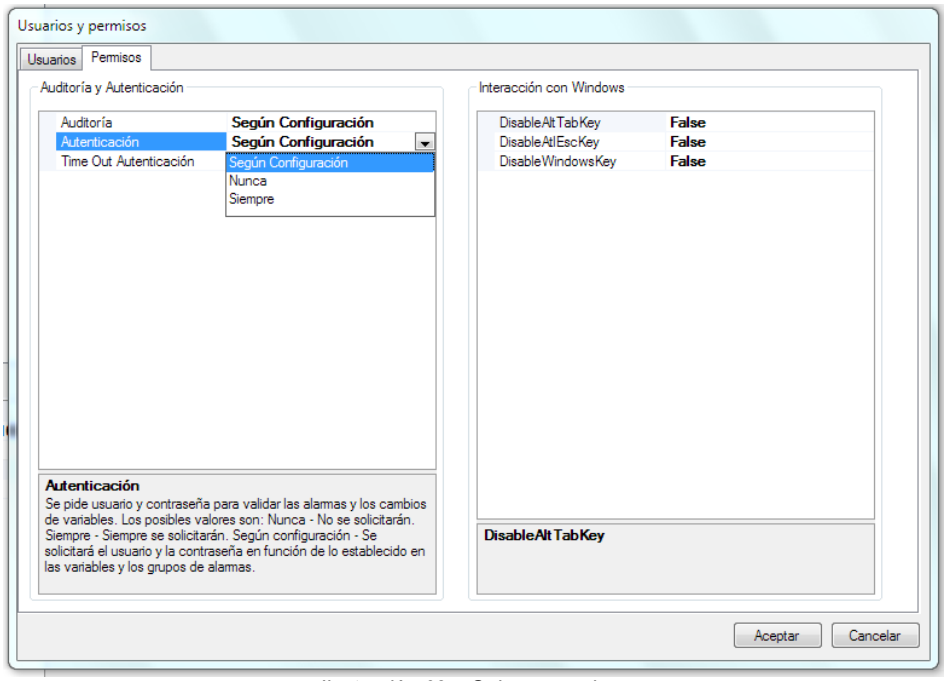


Ilustración 62 – Solapa permisos

En esta ventana se establecen criterios generales para la auditoría de sucesos, la autenticación en los cambios y la interacción de la aplicación con Windows.

Parámetro Auditoría

Mediante el parámetro Auditoría se establece el modo en que se auditará el uso de la aplicación, o sea, si se quiere establecer un registro de los sucesos que ocurren en el uso de la aplicación (cuando se ha iniciado, cuando se ha parado, cuando se ha cambiado el valor de una variable y que valor se le ha establecido, cuando se ha abierto un determinado formulario o se ha cerrado…).

Los posibles valores que se le pueden dar a este parámetro son:

Nunca: No se auditará nada.
Siempre: Se auditará todo.
Según configuración: Se auditaran los cambios de las variables en función de lo que determine la propiedad **RequiresAudit** de la variable en cuestión. El resto de sucesos no correspondientes a variables se auditaran todos.

Parámetro Autenticación
Mediante el parámetro Autenticación se establece si para cambiar el valor de una variable o para validar una alarma se requiere una nueva autenticación del usuario o no.

Los posibles valores que se le pueden dar a este parámetro son:
Nunca: Nunca se solicitará la autenticación del usuario antes de cambiar una variable o validar una alarma.
Siempre: Siempre se solicitará de nuevo el usuario y contraseña antes de cambiar un valor a una variable o validar una alarma.
Según configuración: Se solicitará usuario y contraseña en función de lo que determine la propiedad **AuthenticationRequired** de la variable en cuestión o del grupo de alarma al que pertenezca la alarma.

Parámetro Time Out Autenticación
Este parámetro establece el tiempo en segundos durante el cual será válida una autenticación activa, por ejemplo, si se establece un tiempo de Time Out de 30 segundos y se realiza un segundo cambio de valor de variable menos de 30 segundos después de una autenticación, la aplicación no volverá a pedir el usuario y la contraseña.

Un tiempo de 0 segundos establece que el Time Out es infinito.

Interacción con Windows
Los parámetros de interacción con Windows nos sirven para especificar cómo queremos que se comporte Windows con respecto a Monitoriza. Estos parámetros nos permiten definir un nivel de seguridad mayor para el proyecto Scada ya que con ellos se puede determinar que no se puedan usar simultáneamente al Scada otras aplicaciones que podrían afectar a su funcionamiento.

La propiedad DisableAltTabKey nos establece que si la ponemos a True el usuario no podrá cambiar de aplicación pulsando las teclas Alt + Tab.

La propiedad DisableAltEscKey nos establece que si la ponemos a True el usuario no podrá cambiar de aplicación pulsando las teclas Atl + Esc.

La propiedad DisableWindowsKey nos establece que si la ponemos a True el usuario no podrá acceder al menú de Windows ni por tanto iniciar otras aplicaciones o componentes de Windows.

RECETAS E INTERFASE BATCH

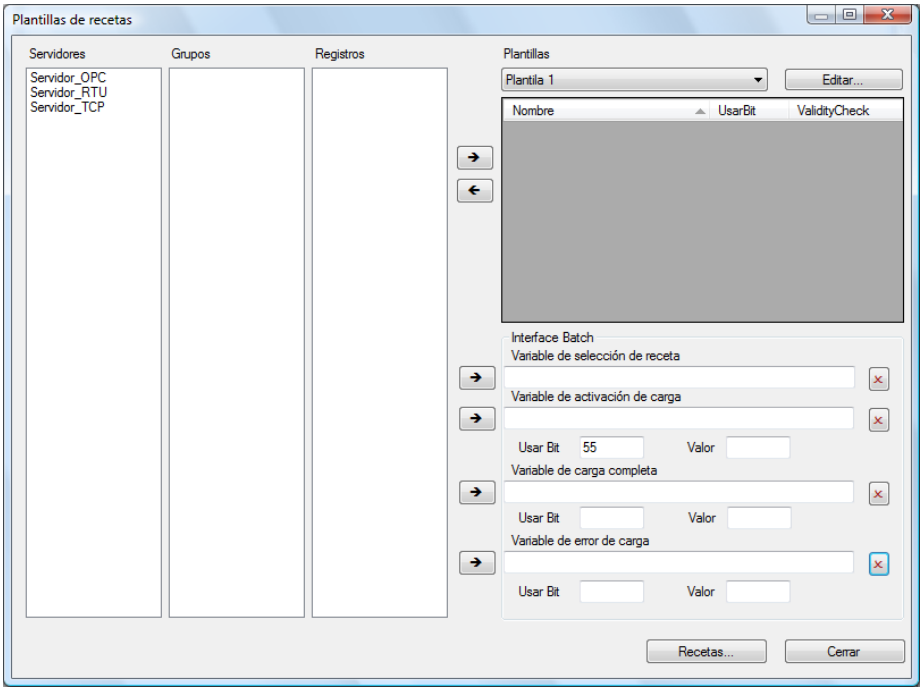
Definamos primero qué es una receta. Una receta es un conjunto de registros de autómata con un valor predeterminado. Podríamos poner como ejemplo una mezcla de varios componentes en una fabricación, dependiendo del porcentaje de estos elementos tendríamos un producto final diferente. La receta serían, por ejemplo, los porcentajes de cada elemento. Es decir tendríamos diferente receta en función del producto final que deseáramos obtener.

Monitoriza permite la carga de recetas tanto manualmente como de forma automática (Batch).

Así pues cada conjunto de recetas parte de una plantilla, que nos permitirá definir qué registros utilizaremos y si estos deberán cumplir con alguna condición.

Para editar plantillas y recetas se dispone de la opción de menú Plantillas de recetas del menú Ver.

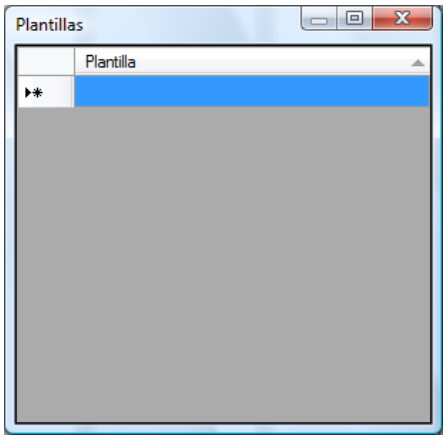
Esta opción nos llevará a la siguiente pantalla:



Cada plantilla tendrá asociados unos registros que serán genéricos para todas las recetas y, si así se decide, una interfase batch para carga en automático.

Para empezar a trabajar debemos dar de alta al menos una plantilla, el botón Editar… nos permitirá dar de alta, modificar o eliminar las plantillas de recetas.

Cuando hagamos clic sobre éste botón tendremos la pantalla siguiente:



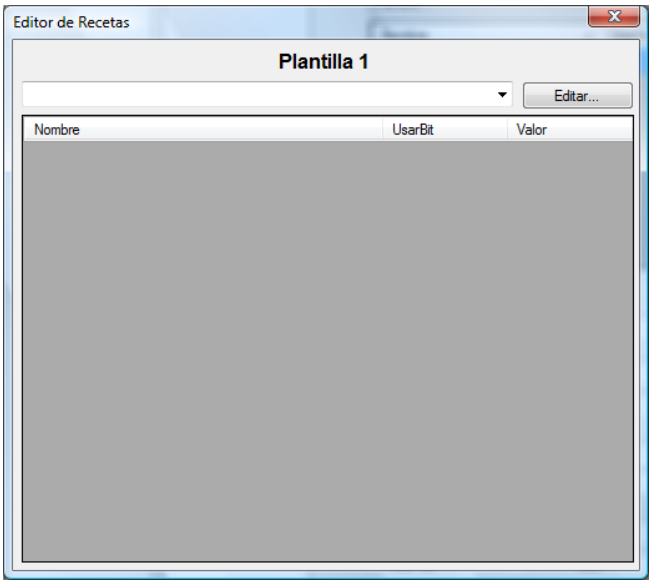
Una vez tengamos al menos una plantilla ya podremos añadir registros a la misma, para ello iremos seleccionando el servidor, grupo y variable que vayamos a utilizar en las listas de la pantalla.

Cada registro añadido a la plantilla tiene dos campos adicionales:

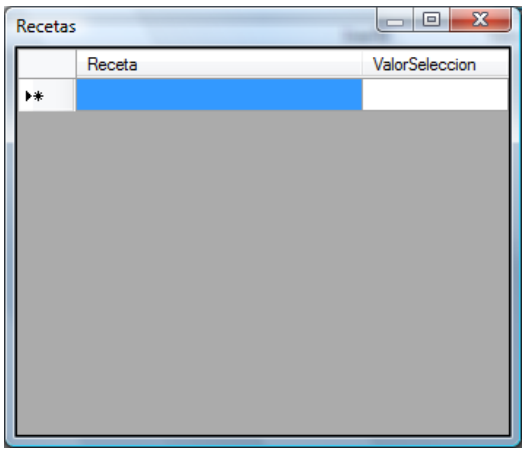
- **UseBit** nos permite utilizar bits individuales de la variable.
- **ValidityCheck** es un campo que, si se utiliza, supondrá un requerimiento previo a la carga de la receta. Es decir si el registro no tiene ese valor no se cargará la receta, ni de forma manual, ni por un procedimiento Batch.

Una vez tenemos definida la plantilla, podemos añadirle recetas mediante el botón Recetas (también es posible hacerlo en ejecución por parte del usuario con el control Recipe),.

Esta es la pantalla que veremos, que tiene un comportamiento idéntico al que se tiene en ejecución (excepto por el hecho de que no se puede cargar en el autómata la receta).



Con el botón Editar damos de alta, modificamos o eliminamos Recetas, además el campo ValorSeleccion nos permitirá especificará el valor de carga de cada receta en Batch si es necesario. **Importante**: Las recetas se almacenan en un fichero en la misma localización y nombre que el proyecto pero con la extensión *recipes*.



Cada vez que demos de alta una receta, el sistema nos rellenará el grid de la pantalla con la plantilla de registros, el usuario tan solo deberá rellenar los valores a cargar en el autómata.

Cada plantilla dispone de unos registros para configurar su comportamiento de carga en **Batch**, es decir independientemente de la acción de un usuario, el servidor de Monitoriza

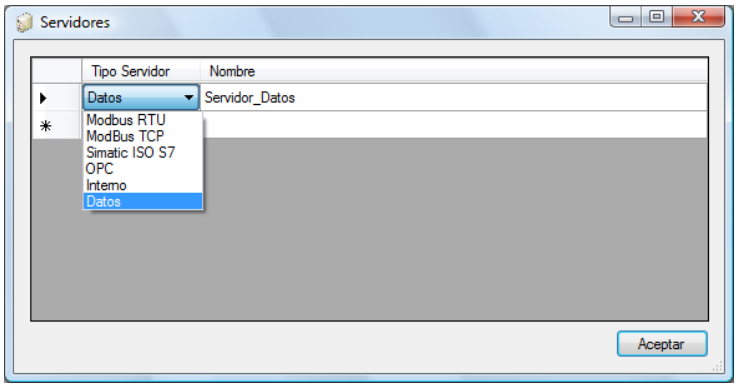
comprobará una serie de registros para determinar cuándo cargará una receta en el autómata.

La **configuración de Batch** para cada plantilla está en la sección Interface Batch, en esta sección tenemos cuatro Variables:

- **Variable de selección de receta**, este registro es el que junto al valor especificado al dar de alta la receta en el campo ValorSeleccion determinará qué receta se seleccionará para carga en el autómata. Si se desea configurar una carga Batch este campo es obligatorio.
- **Variable de activación de carga**, este registro es el que al tener el valor especificado en Valor desencadenará la carga. Si se desea configurar una carga Batch este campo es obligatorio.
- **Variable de carga completa**, si el procedimiento de carga batch se completa con éxito, se escribirá en este registro del autómata el valor especificado. No es un campo obligatorio.
- **Variable de error de carga**, si el procedimiento de carga batch no se completa con éxito, se escribirá en este registro del autómata el valor especificado. No es un campo obligatorio.

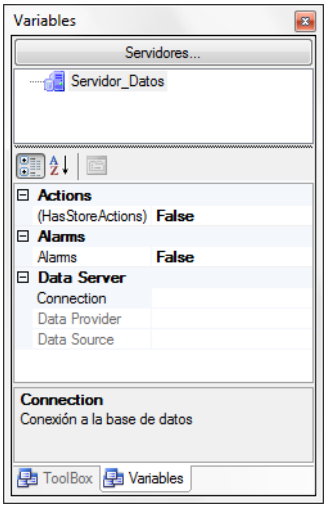
GUARDAR EN BASE DE DATOS

En primer lugar hay que crear un servidor de tipo Datos

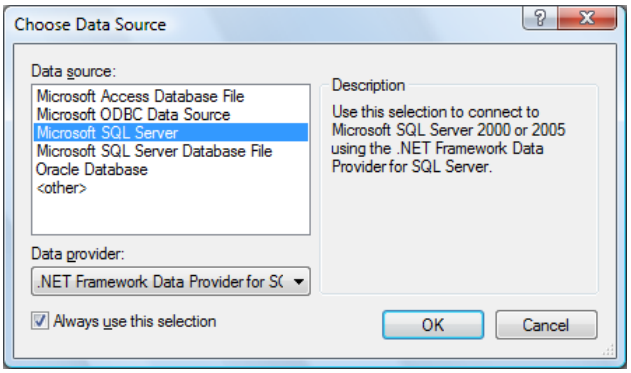


Pueden definirse tantos servidores de datos como se necesiten, cada uno de ellos nos dará acceso a una base de datos concreta.

Al editar la propiedad **Connection**



nos aparece un asistente que nos guiará en el proceso de conexión.



Las alarmas pueden guardarse en base de datos, para ello debe existir una tabla que se describe en los apéndices (Tabla alarmas). El servidor de datos en el que tengamos esa tabla deberá tener la propiedad **Alarms** a *True*. También será este servidor de datos en el que se almacenaran los registros de auditoría y también se deberá poner la propiedad **Alarms** a *True*. La tabla de Auditoría deberá tener los campos descritos en el apéndice Tabla Auditoría.

Por otra parte sobre los servidores de datos podemos definir Acciones de Guardado, para ello la propiedad **HasStoreActions** debe estar a *True*, y de esta manera definiremos éstas en la propiedad (colección) **StoreActions,** desde la cual podremos acceder al *Editor de Acciones de Guardado* que vemos a continuación.

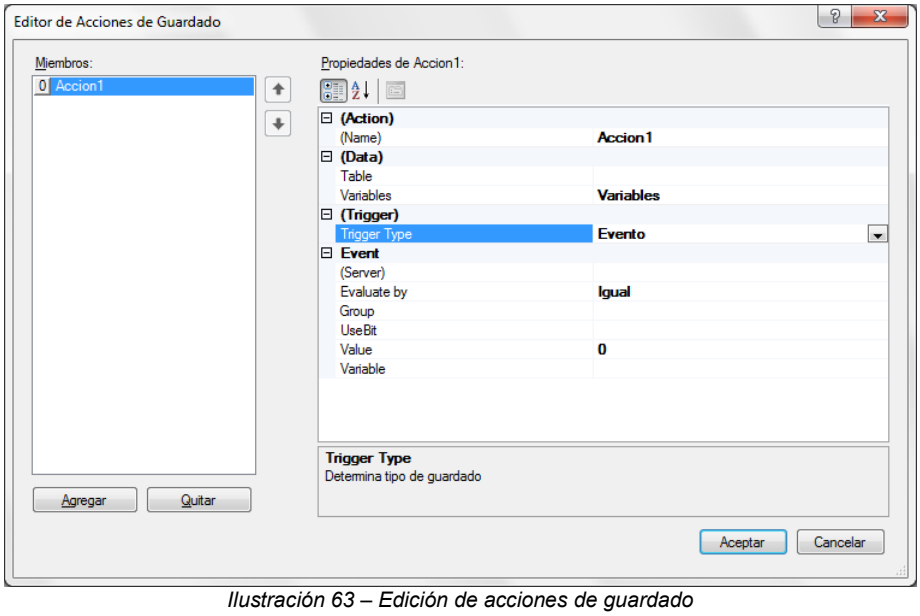


Ilustración 63 – Edición de acciones de guardado

En este editor definimos el *Nombre* que tendrá la acción de guardado, el **Trigger Type** *(Tipo de disparo)*: Evento o Cadencia.

Si se trata de Evento podemos ver que se escogerá una Variable (con **Server, Group y Variable)** y al igual que en el caso de las Alarmas escogeremos en **Evaluate by**, **UseBit** y **Value** la circunstancia que hará que se desencadene la Acción de Guardado.

Si escogemos Cadencia, simplemente haremos constar cada cuanto tiempo queremos que se ejecute la Acción de Guardado en segundos.

Por otra parte nos falta definir qué valores se guardarán, en qué tabla y en qué campos, la tabla la escribiremos en la propiedad **Table**, mientras que el resto lo haremos en el *Editor de Variables de Guardado* al que accederemos desde la propiedad (colección) **Variables**, vemos a continuación este editor.

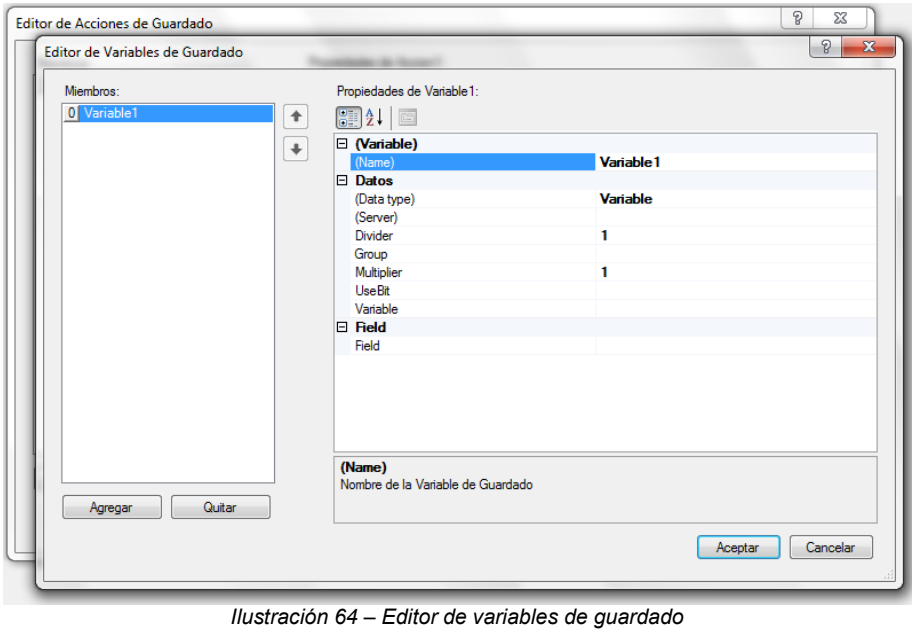


Ilustración 64 – Editor de variables de guardado

Aquí definiremos los campos que deseamos escribir en la Tabla y el tipo de datos a registrar, en la propiedad **Field** tendremos el nombre del campo, la propiedad **Data type** nos permitirá guardar valores de Variables Scada, Campos Fecha/Hora o constantes, el resto de propiedades cambian en función de la clase de dato.

Hay que hacer notar que si más tarde deseamos mostrar o graficar desde la aplicación Monitoriza estos datos, es necesario tener un campo de tipo Fecha-Hora.

MOSTRAR HISTÓRICO DE DATOS

Acimut Monitoriza, además de permitir guardar valores en base de datos permite mostrarlos con facilidad, para ello tan solo es necesario añadir un Botón en un Formulario en el que su propiedad **Actionn** se ponga a *MostrarGrafica*. Esto hará que en ejecución al hacer clic sobre el botón se nos muestre el siguiente formulario.

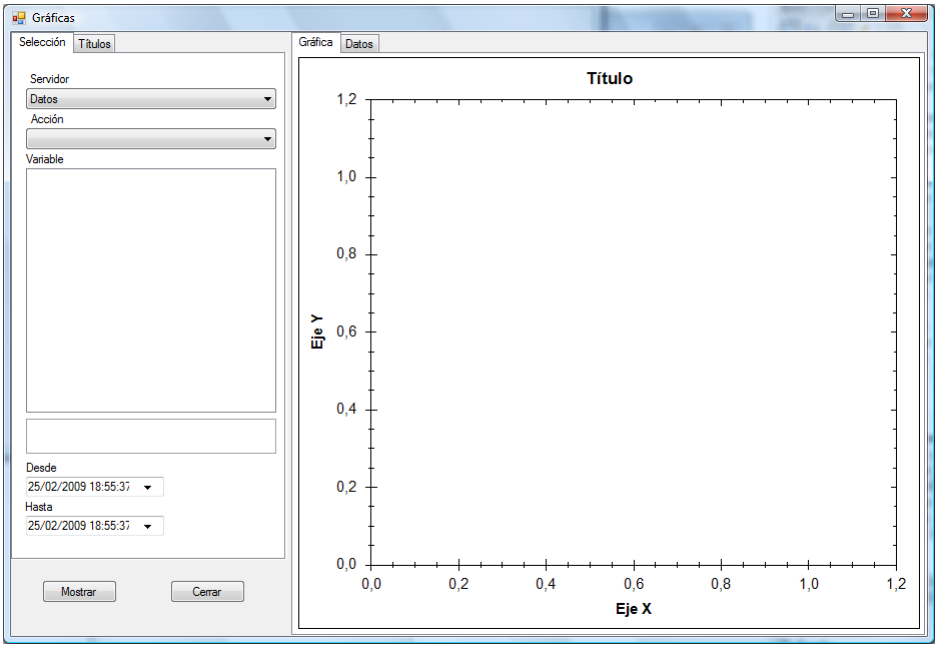


Ilustración 65 – Pantalla de representación de gráficos

Este formulario nos permitirá ver los valores y representar gráficamente los mismos de la Acción de Guardado que escojamos.

MOSTRAR HISTÓRICO ALARMAS

Las alarmas que se almacenan en base de datos pueden consultarse desde una aplicación Monitoriza con facilidad, para ello tan solo es necesario añadir un Botón en un Formulario en el que su propiedad **Action** se ponga a *MostrarHistoricoAlarmas*. Esto hará que en ejecución al hacer clic sobre el botón se nos muestre el siguiente formulario.

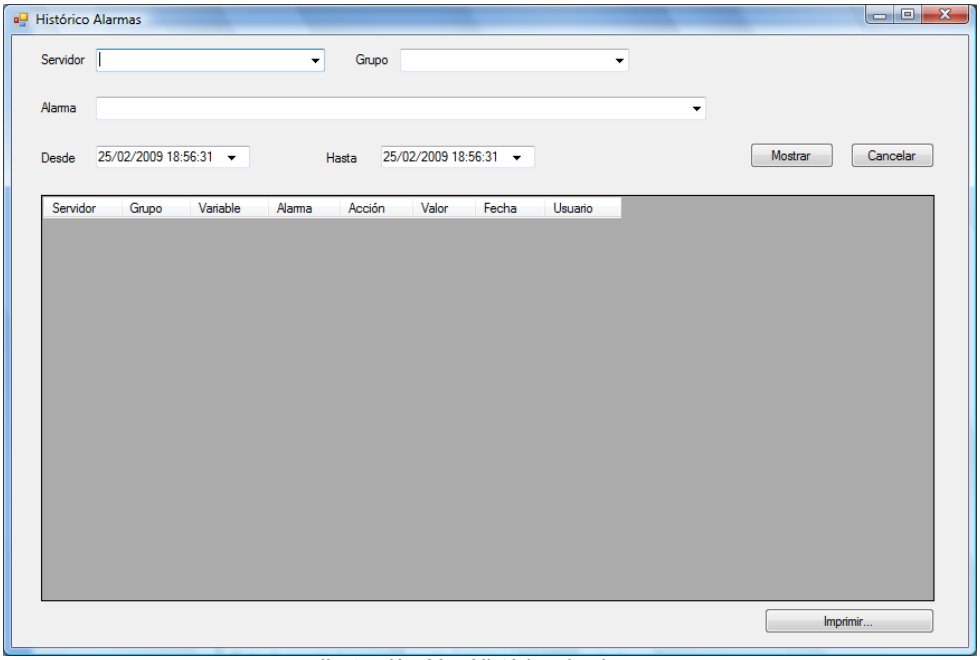


Ilustración 66 – Histórico de alarmas

Este formulario nos permitirá ver un histórico de las alarmas que escojamos.

CONFIGURAR COMUNICACIONES DEL SERVIDOR

Monitoriza es una aplicación que utiliza tecnología de comunicaciones cliente/servidor, así pues, es necesario configurar adecuadamente el servidor, para ello tenemos la siguiente opción de menú.

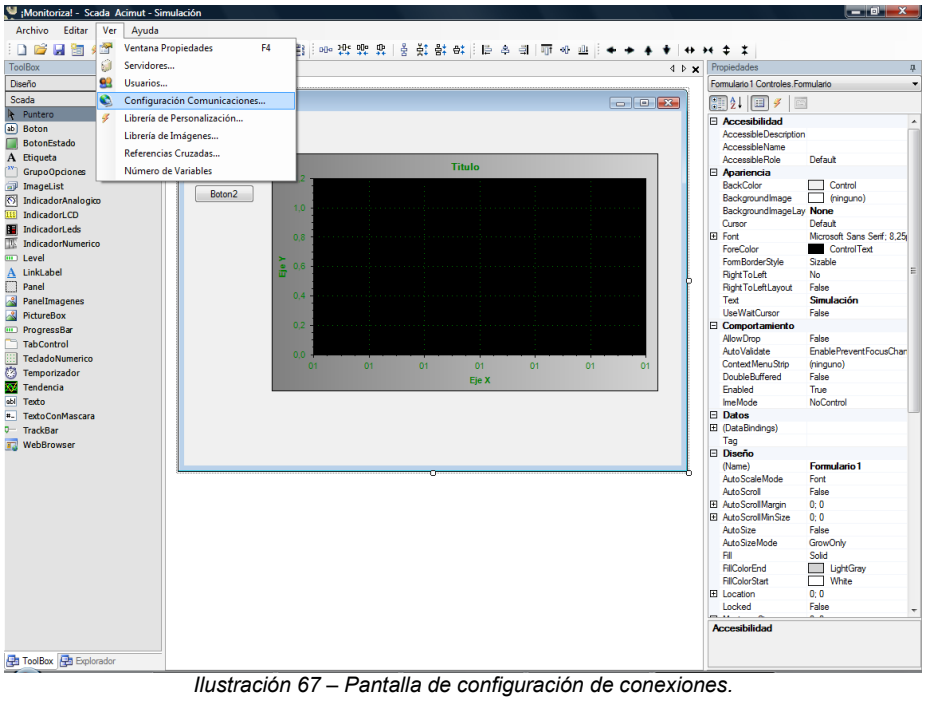
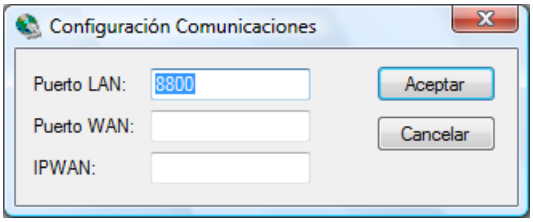


Ilustración 67 – Pantalla de configuración de conexiones.

Esta opción nos lleva al siguiente formulario.



Por una parte el servidor de Monitoriza puede ser accedido desde una LAN, el puerto por defecto es el 8800 como se ve en el campo Puerto LAN, por otra parte, cada vez es más necesario un acceso desde una red como es Internet, para ello hay que escoger un puerto en Puerto WAN y hay que informar a Monitoriza de cuál es la dirección IP pública que tendrá en Internet mediante el campo IPWAN.

La configuración necesaria en firewalls y routers está fuera del alcance de este manual. Hay que hacer constar que en el caso de comunicaciones a través de WAN se utiliza protocolo HTTP para mayor comodidad al configurar firewalls.

El servidor soporta simultáneamente conexiones LAN y WAN.

SERVIDOR

El servidor de Acimut Monitoriza es el encargado de establecer y coordinar las comunicaciones tanto con los dispositivos y autómatas como con las bases de datos.

El servidor monitoriza los cambios en las variables definidas en tiempo real e informa a los clientes, o sea, a los puestos de control, para que estos presenten la información a los usuarios del sistema.

También se encarga de evaluar los valores de las variables para comprobar si se cumple alguna de las condiciones establecidas para disparar una alarma.

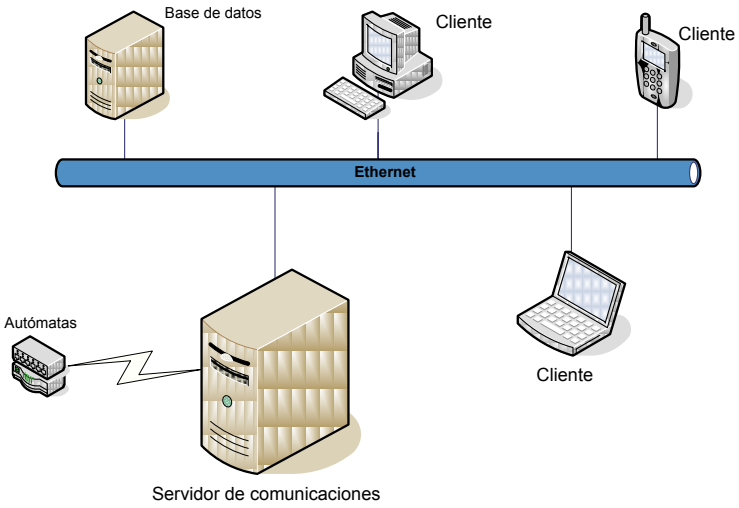


Ilustración 68 – Esquema comunicaciones

Cuando iniciamos el servidor lo primero que nos pregunta es si queremos configurar un acceso directo.

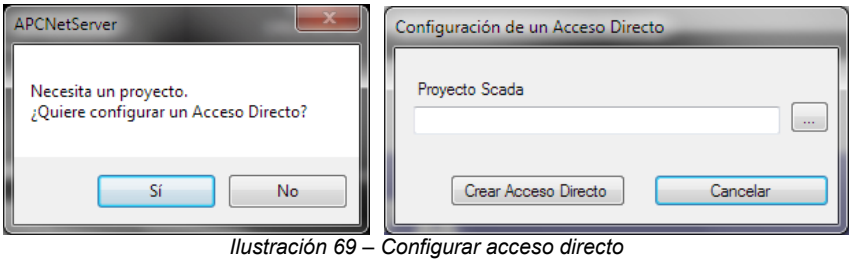


Ilustración 69 – Configurar acceso directo

Esto es así, ya que el servidor necesita saber cuál es el proyecto que queremos ejecutar.

Le tendremos que especificar un proyecto .scada creado con el Editor de Monitoriza y pulsando el botón *Crear Acceso Directo* nos creará en el escritorio un enlace al servidor que por ejemplo quedará como:

"C:\Archivos de programa\Acimut\APCNetServer.exe" C:\Documents and Settings\Usuario\Mis documentos\ServidorModBUSTCP.scada

Para finalizar el servidor tenemos que pulsar, con el botón derecho del ratón, sobre el icono del servidor que se encuentra en la barra de notificaciones del escritorio (marcado en rojo en la siguiente figura)

Ilustración 70 – Servidor comunicaciones

Y elegir la opción Salir.

Mediante el icono marcado en rojo en la siguiente figura

Ilustración 71 – Control de autómatas

Podemos ver el estado de comunicación del Servidor con los autómatas para poder comprobar si se está accediendo correctamente a cada uno de los dispositivos definidos.

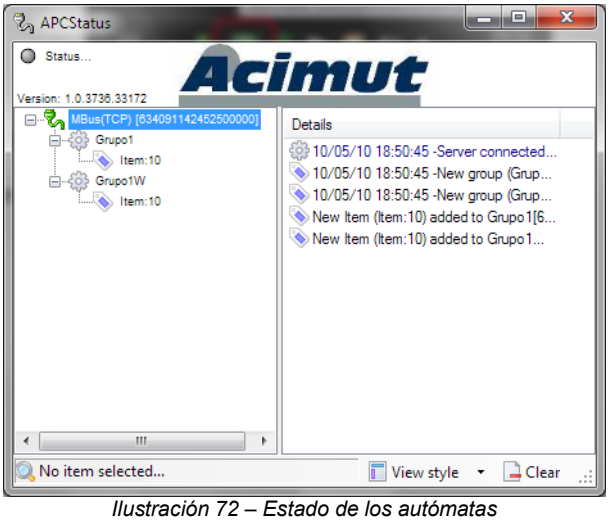


Ilustración 72 – Estado de los autómatas

CLIENTE

El cliente de Monitoriza debe hacer referencia a un servidor para mostrar la ejecución de un proyecto. El caso más sencillo, es en el que el servidor se encuentra en el mismo ordenador, no precisa de parámetros, pues por defecto se busca un servidor en la máquina local.

Sin embargo mostramos aquí los parámetros soportados por el cliente de Monitoriza.

NombreServidor Puerto Red

NombreServidor es el nombre o dirección IP del ordenador que ejecuta el servidor.
Puerto es el puerto TCP que ejecuta el servidor de Monitoriza, 8800 es el valor por defecto.

Red puede tener dos valores:

WAN indica que se utilizarán comunicaciones a través de Internet (por ejemplo), se utilizará un protocolo HTTP.

LAN indica que se utilizarán comunicaciones en una LAN, se utilizará un protocolo TCP.

EXTENSIBILIDAD Y PROGRAMACIÓN

Aunque, en gran parte de los proyectos de Monitoriza no es necesario programar, en ocasiones, o bien por la complejidad de lo que se pretende, o bien porque el programador se siente más cómodo, surge la necesidad de tener una herramienta que soporte la programación más avanzada, es entonces cuando podemos decir que Monitoriza es programable en .Net, o sea, con toda la potencia que nos brindan los lenguajes C# y Visual Basic.Net de Microsoft, permitiendo tanto sencillas como elaboradas funciones de usuario.

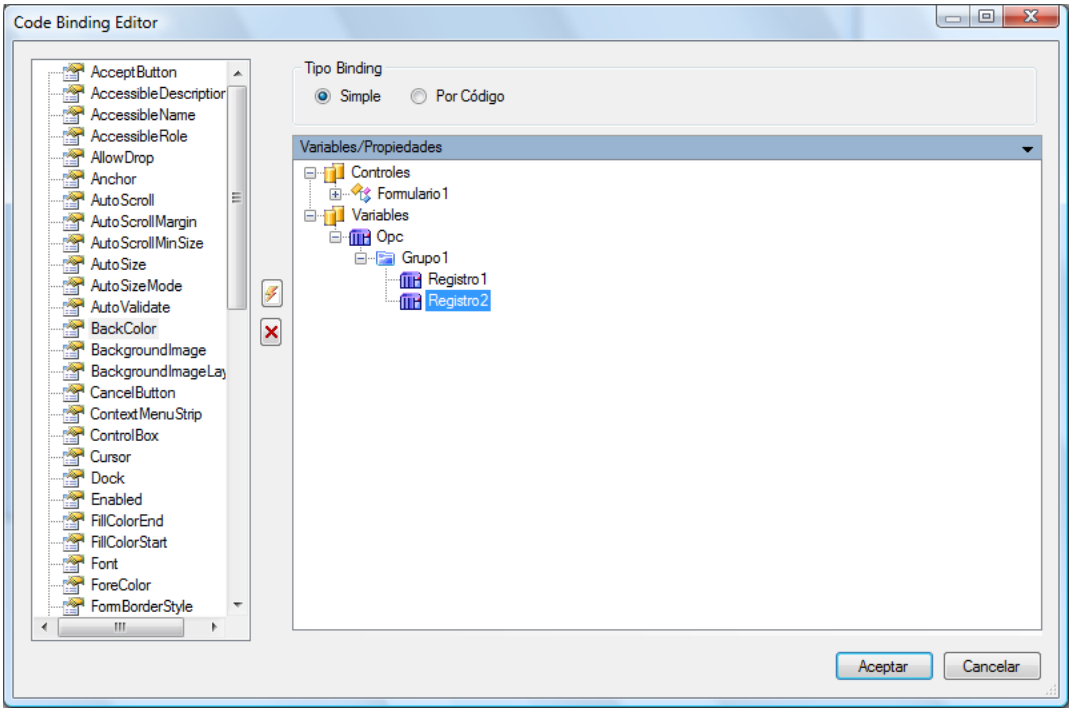
Por una parte podemos programar la asignación de propiedades y los eventos de los controles desde el propio Editor de Monitoriza y por otra parte podemos asignar librerías externas desarrolladas en .NET.

Hay que tener en cuenta que estas funciones se ejecutan en la parte cliente de Monitoriza.

CODIGO INTERNO “CodeBindings”

Los controles de Monitoriza, así como los formularios, tienen una serie de propiedades, que pueden ser asignadas en el momento del diseño. Sin embargo en ocasiones deseamos que esta asignación sea dinámica en función de lo que esté ocurriendo.

Es por esto que los controles de Monitoriza tienen la propiedad **CodeBindings,** mediante esta propiedad llegamos a la siguiente pantalla

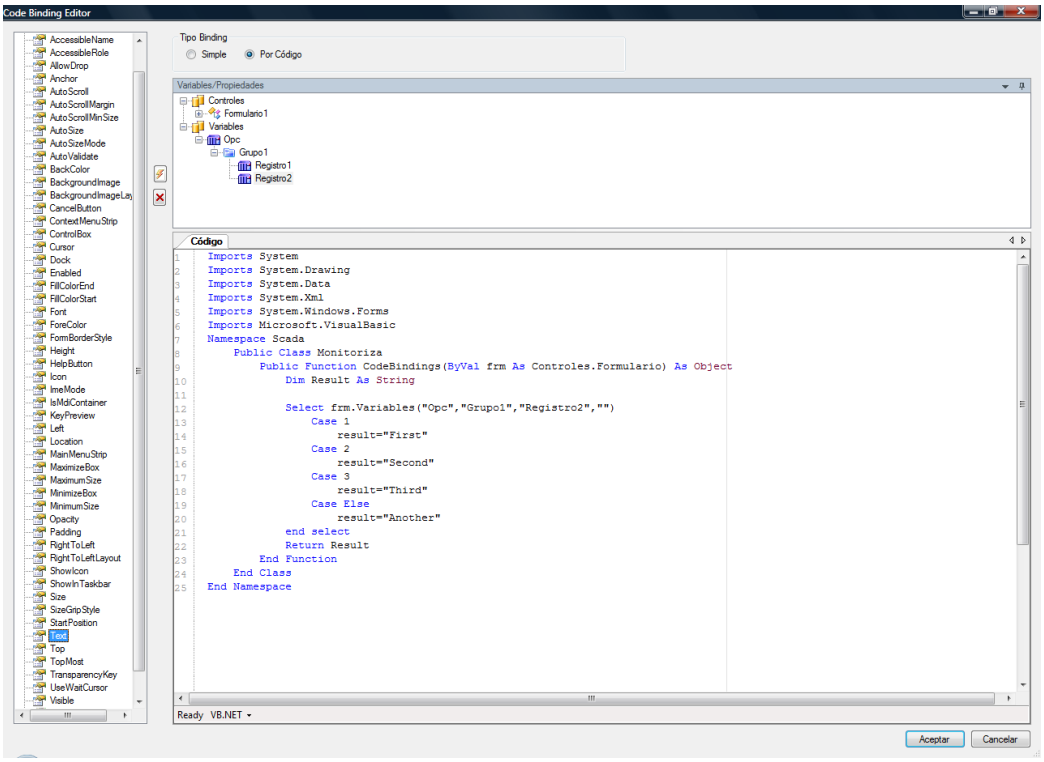


Podemos ver que disponemos de una lista en la parte izquierda con las propiedades del control seleccionado, estas propiedades son las que se podrán asignar dinámicamente. En la imagen podemos ver que en este caso a la propiedad BackColor se le asignará, en

ejecución, el valor de la variable Registro2, esto es un binding de tipo **Simple** y se asocia con el botón, para quitar una asociación se dispone del botón.

Además de valores de Variables también se pueden asociar valores de propiedades de otros controles.

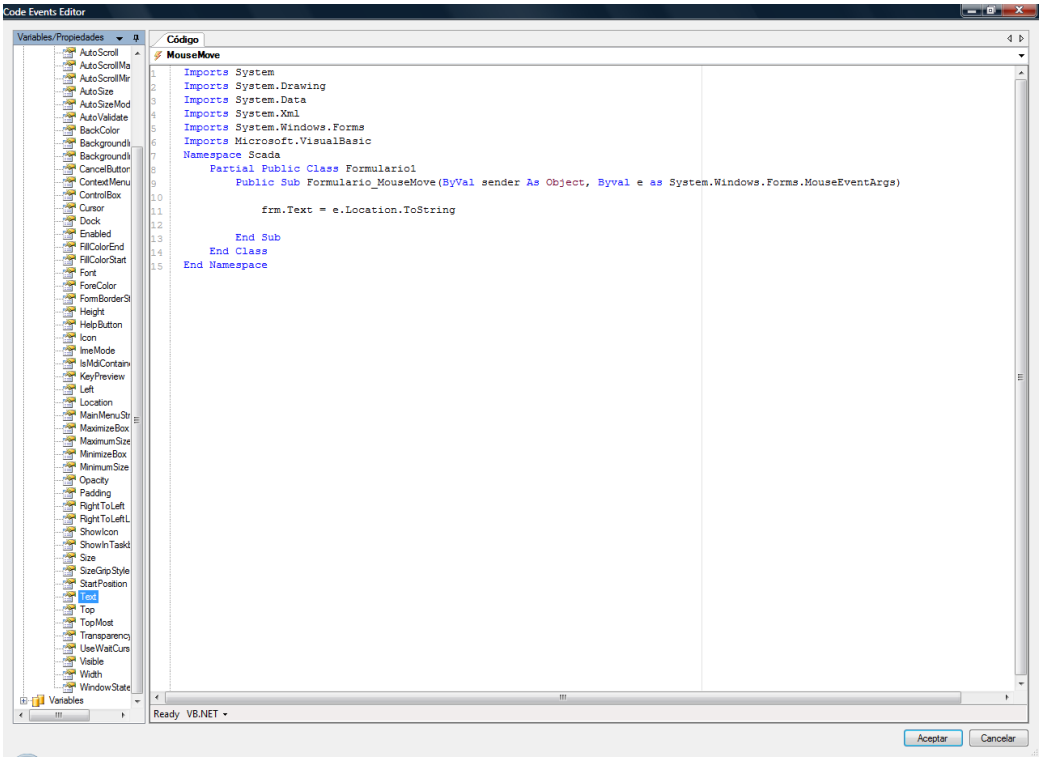
Como la asignación directa de valores de Variables o Propiedades no siempre cumple con nuestras expectativas y necesitamos algo más elaborado, aquí es cuando podemos elegir un binding de tipo **Por Código,** cuando escogemos este tipo la pantalla pasa a ser así:



Seguiremos teniendo la misma funcionalidad en la lista de la parte izquierda, pero ahora tenemos una nueva ventana de edición de código en la que podremos desarrollar una función cuyo valor devuelto por la función se asignará dinámicamente a la propiedad escogida. La lista de la parte superior nos sirve para *arrastrar y soltar* sobre la nueva ventana de código y no tener que escribir o bien las propiedades o bien las variables.

Cuando se haga la asociación con el botón, se compilará el código y se informará si existen errores de sintaxis. Como vemos en la imagen anterior se ha creado un procedimiento que en función del valor de una variable nos devuelve diferentes valores que se le asignaran a la propiedad **Text**.

Como hemos dicho, además de poder asignar de forma dinámica valores a las propiedades de los controles, también podemos ejecutar código sobre los diferentes eventos de los mismos. Para poder definir el código de estos eventos podemos hacer tanto un doble clic sobre un control como un doble clic sobre el evento elegido en la ventana de propiedades/eventos y se nos mostrará la siguiente ventana.



En este caso la lista de la parte izquierda nos sirve para *arrastrar y soltar* propiedades y variables sobre el código, y en la parte del código podemos navegar entre los diferentes eventos mediante el desplegable de la parte superior. Cuando en ejecución se de el evento se ejecutará este código.

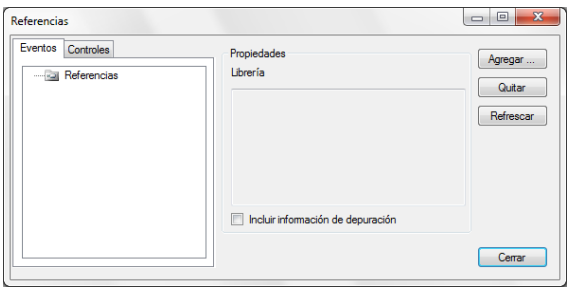
CODIGO EXTERNO

LIBRERÍA DE CONTROLES Y LIBRERÍA DE EVENTOS

Las librerías de eventos permiten asociar código de una librería externa a un evento de un control de Monitoriza.

Las librerías de controles, a diferencia de las librerías de eventos, permiten la inclusión de controles de usuario (*usercontrols*) directamente en el proyecto, pudiendo usarlos como cualquier otro control nativo de la aplicación Monitoriza.

Se pueden añadir referencias a librerías desde la pantalla de **Librería de Personalización**, en el menú **Ver**.



LA HERRAMIENTA DE PROGRAMACIÓN

La herramienta de programación es Visual Studio® ya sea en su versión 2005 o superior, el lenguaje de programación es cualquiera que esté soportado por Visual Studio® y sea del agrado del programador, Visual Basic, C#, J#, etc.

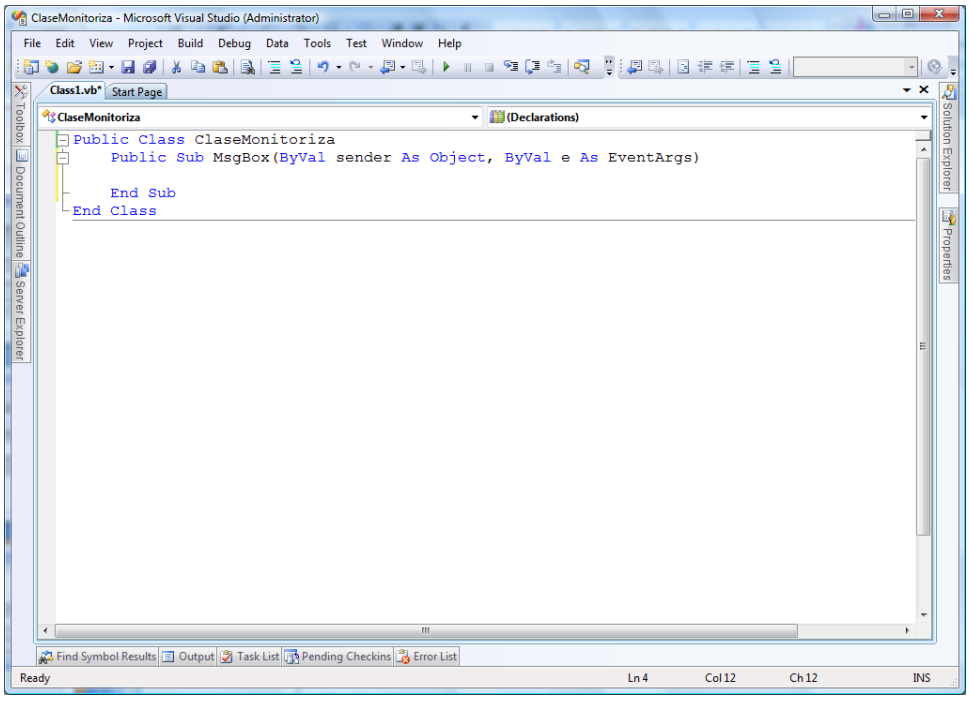
Hay que destacar aquí que Microsoft proporciona de forma gratuita la versión Express de Visual Studio® que es perfectamente válida para utilizar con Monitoriza.

En este documento veremos ejemplos desarrollados en Visual Basic.

LAS FUNCIONES. LIBRERÍA DE EVENTOS

Monitoriza utiliza proyectos DLL de .NET para ejecutar funciones, para crear un proyecto de este tipo, desde Visual Studio® elegiremos la opción de menú Archivo-Nuevo Proyecto y escogeremos un proyecto del tipo **Biblioteca de Clases** y por ejemplo le daremos el nombre ClaseMonitoriza**.**

Existen una serie de condiciones que una librería debe cumplir para poder ser utilizada desde Monitoriza, en primer lugar, aunque la librería puede tener cualquier número de clases, las que podrá ver desde Monitoriza son las de tipo **Public**, así mismo las funciones deberán ser también públicas. Vemos a continuación un código muy sencillo y utilizable desde Monitoriza.



Una importante observación, es que las funciones, que escribamos, se utilizarán desde eventos de los diferentes objetos de Monitoriza, por lo que deberán tener dos argumentos de la forma:

```
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)

    End Sub
```

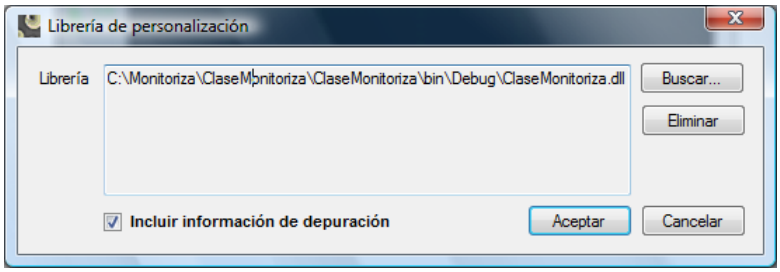
sender será el control, formulario, temporizador, etc que dispara el evento.
e será el parámetro asociado con el evento correspondiente.

Vamos a añadir un poco de código a la función:

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Microsoft.VisualBasic.MsgBox("Hola soy " & sender.name.ToString)
    End Sub
End Class
```

Para utilizar esta función que hemos escrito debemos Generar la librería y añadirla al proyecto de Monitoriza, para ello en Visual Studio® escogemos la opción de menú Generar-Generar ClaseMonitoriza.

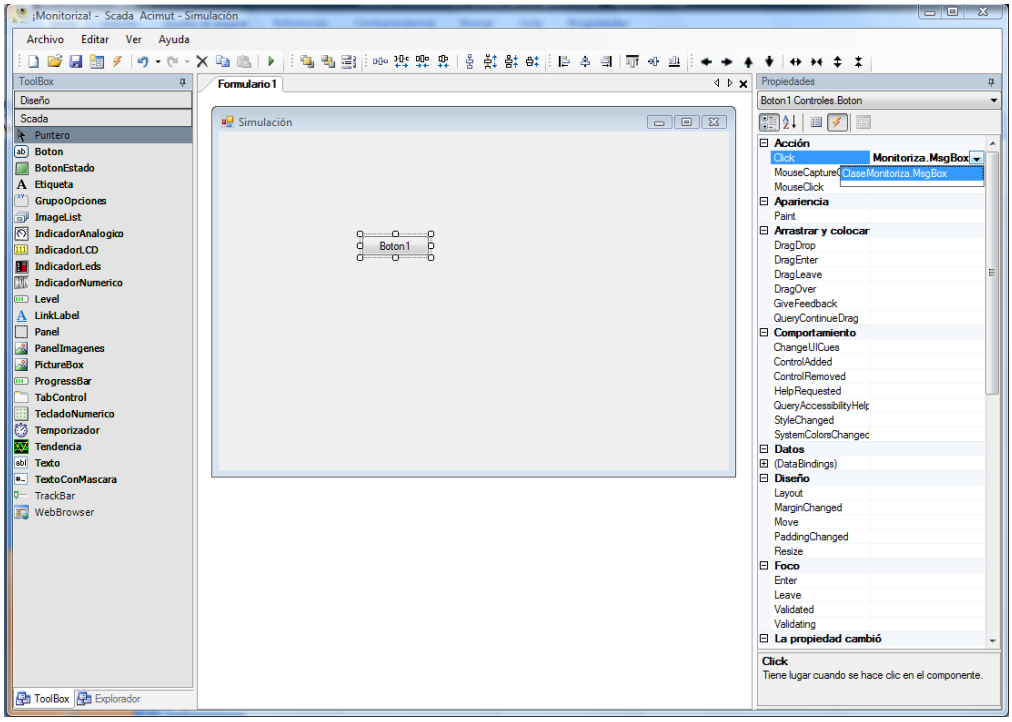
En el Editor de proyecto de Monitoriza generamos un nuevo proyecto y escogemos la opción de menú Ver-Librería de personalización, en la siguiente ventana buscaremos y asignaremos la librería.



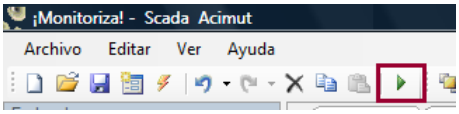
Vemos que hemos escogido **Incluir información de depuración,** veremos en un siguiente paso cómo poder depurar una función.

Hay que tener en cuenta que no necesitaremos distribuir la librería como un archivo adicional, Monitoriza embebe en el proyecto esta librería, es por esto que cada vez que cambiemos la librería es necesario refrescar la misma volviendo a acceder a la pantalla anterior y aceptando.

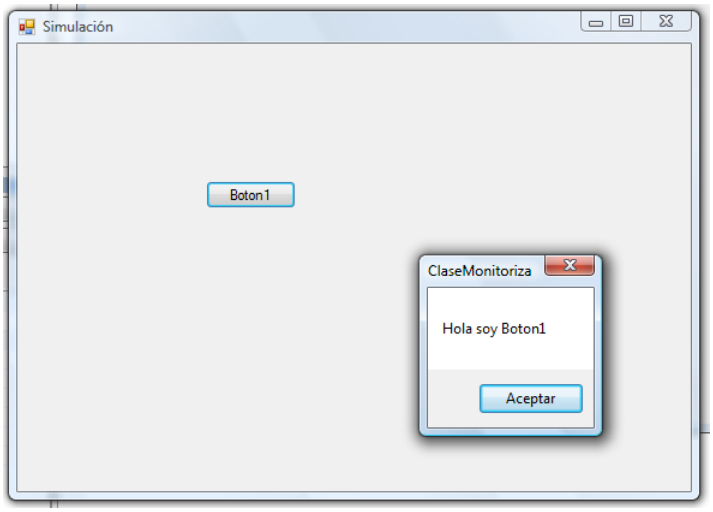
Una vez aceptada la opción anterior y sobre un nuevo formulario (declarado de inicio por defecto) pondremos un Botón y en el evento Click asignaremos ClaseMonitoriza.MsgBox, vemos a continuación cómo queda.



Ahora pondremos en marcha el proyecto con el botón Play



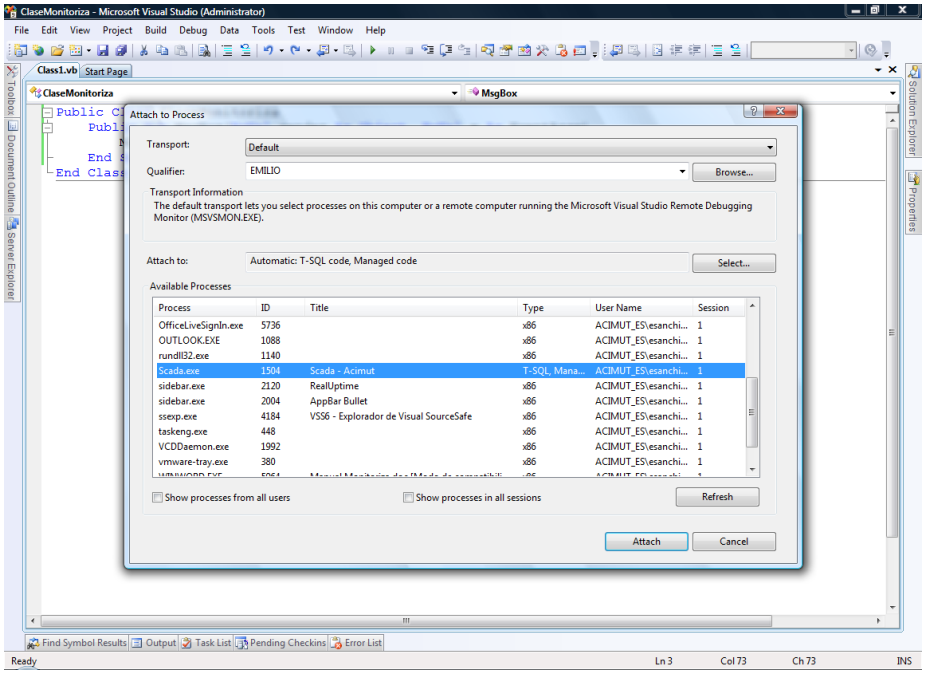
Nos aparecerá el proyecto en modo simulación y al pulsar sobre el botón obtendremos lo siguiente



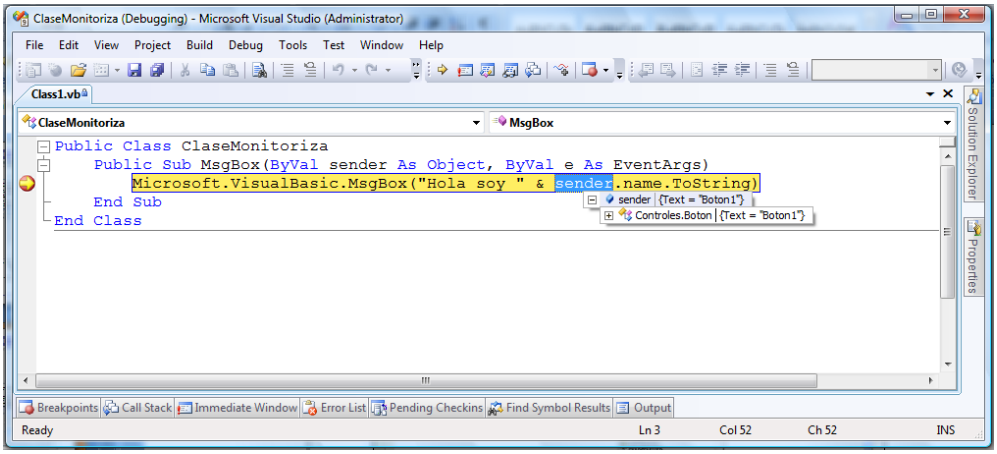
Como hemos dicho anteriormente vamos a aprender a depurar, para ello la librería hay que generarla con información de *debug*, y debemos incluir en el proyecto la información de depuración.

El primer paso consiste en abrir con Visual Studio® el proyecto con la librería, a continuación ponemos en marcha el proyecto y de nuevo sobre Visual Studio® al escoger

la opción de menú Herramientas- Asociar al proceso obtendremos una pantalla similar a la siguiente en donde elegiremos el proceso Scada.exe.

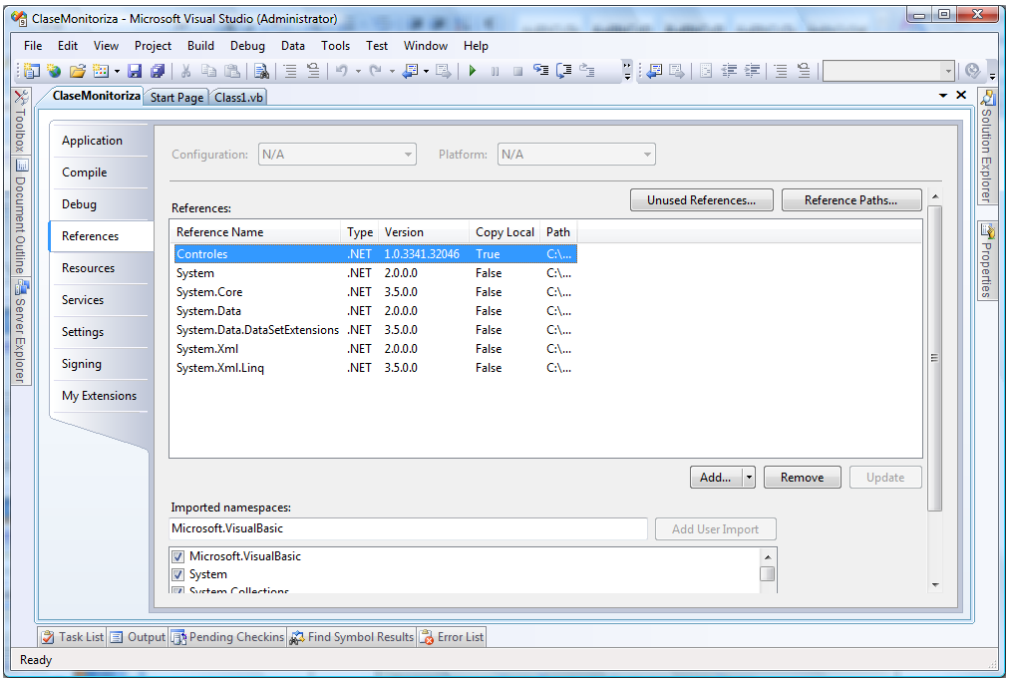


Vemos además que hemos establecido un punto de ruptura en nuestra función, cuando hagamos clic en el botón el código se detendrá y se nos mostrará en Visual Studio® en donde podremos examinar el contenido de las variables, ejecutar paso a paso y cualquier otra funcionalidad que nos proporcione Visual Studio®.



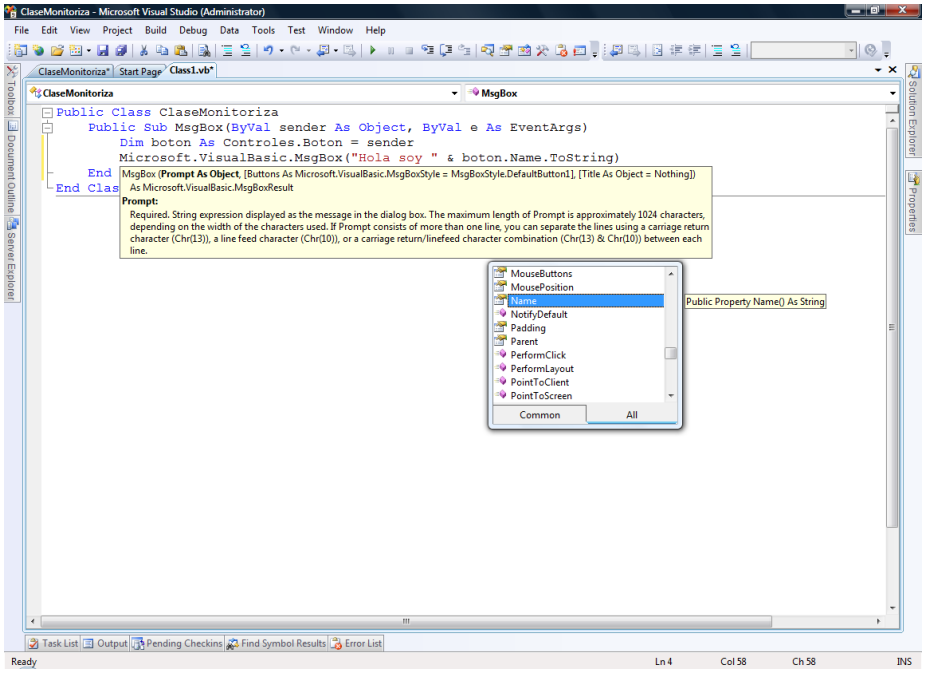
ACCESO A LAS PROPIEDADES DE LOS CONTROLES DE MONITORIZA

Como hemos visto en el ejemplo anterior, cuando tenemos que obtener o cambiar propiedades de un control, no disponemos de Intellisense, sin embargo esto es fácil de solucionar y además dispondremos de toda la ayuda que nos proporciona Visual Studio®. Para ello al proyecto le añadiremos una referencia a la librería Controles.dll que encontraremos en el directorio en que esté instalado el cliente de Monitoriza (Scada.exe).



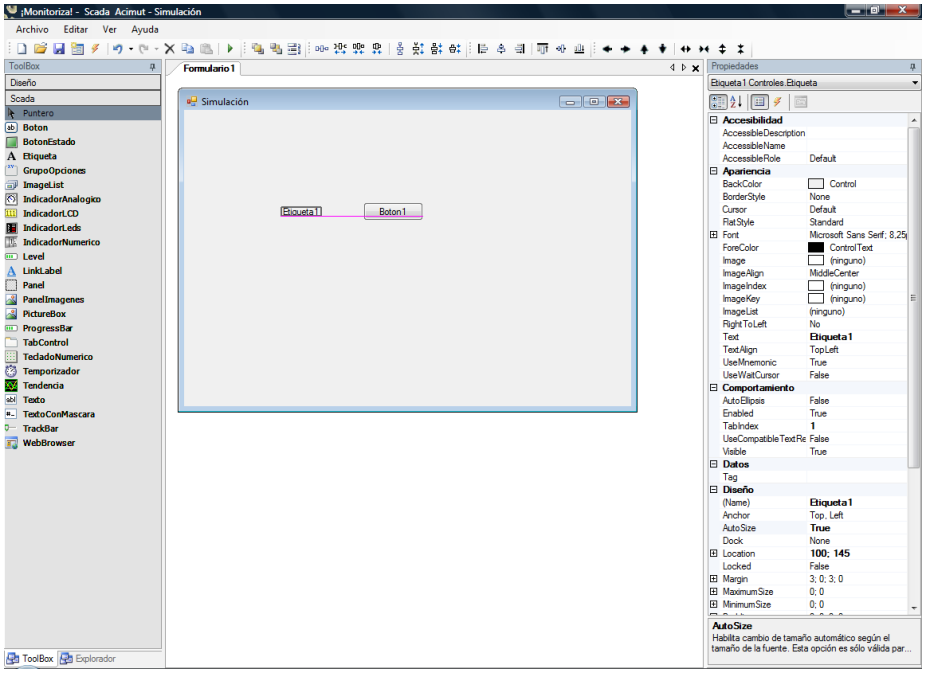
Ahora un pequeño cambio en el código nos proporcionará el Intellisense. Vemos a continuación el nuevo código y su efecto en el editor.

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Dim boton As Controles.Boton = sender
        Microsoft.VisualBasic.MsgBox("Hola soy " & boton.Name.ToString)
    End Sub
End Class
```



Otra particularidad en la posibilidad de, desde un control, alcanzar el formulario u otro control contenido en el mismo.

Para mostrar esto, añadiremos un Label al formulario del proyecto y modificaremos el código. El formulario quedaría



El código lo modificaremos así:

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Dim boton As Controles.Boton = sender

        'le cambiamos el título al formulario
        boton.TopLevelControl.Text = "Hemos hecho clic"

        'accedemos a otro control en el formulario
        Dim etiqueta As Windows.Forms.Label
        etiqueta = boton.TopLevelControl.Controls("Label1")
        etiqueta.Text = "Aqui también cambiamos el texto"

    End Sub
End Class
```

Para que estos cambios tengan efecto debemos por una parte Generar la nueva librería y por otra volver a cargarla en el proyecto antes de ejecutar.

ACCESO A VARIABLES Y COMPONENTES

Un elemento importante a destacar es el poder acceder desde el código a los valores de las variables, estos se encuentran en una propiedad de tipo Dictionary del Formulario, también podremos acceder a los componentes (como por ejemplo el Temporizador) en una propiedad similar a Controls llamada Componentes, vemos en un ejemplo cómo se accede fácilmente a estas dos propiedades del formulario.

Cuando accedemos a la propiedad variables, deberemos indicar de qué variable queremos obtener el valor, pasando como argumento el nombre del servidor, el nombre del grupo y el nombre de la variable separados por comas.

```
Public Class Monitoriza
    Public Sub MsgBox(ByVal sender As Object, ByVal e As EventArgs)
        Dim boton As Controles.Boton = sender

 'le cambiamos el título al formulario
        boton.TopLevelControl.Text = "Hemos hecho clic"

        'accedemos a otro control en el formulario
        Dim etiqueta As Windows.Forms.Label
        etiqueta = boton.TopLevelControl.Controls("Label1")
        etiqueta.Text = "Aqui también cambiamos el texto"

        'accedemos a las variables y componentes

        'definimos el formulario
        Dim formul As Controles.Formulario
        formul = CType(boton.TopLevelControl, Controles.Formulario)

        'accedemos a las variables
        Microsoft.VisualBasic.MsgBox("Variables:            "            &
formul.Variables.Count)

Microsoft.VisualBasic.MsgBox("Variable(""Servidor,Grupo1,Registro1"")=   "
& _
        formul.Variables("Servidor,Grupo1,Registro1"))

        'accedemos a los componentes
        Microsoft.VisualBasic.MsgBox("Componentes:            "            &
formul.Componentes.Count)
        Dim temp As Controles.Temporizador
        temp = formul.Componentes("Temporizador1")
```

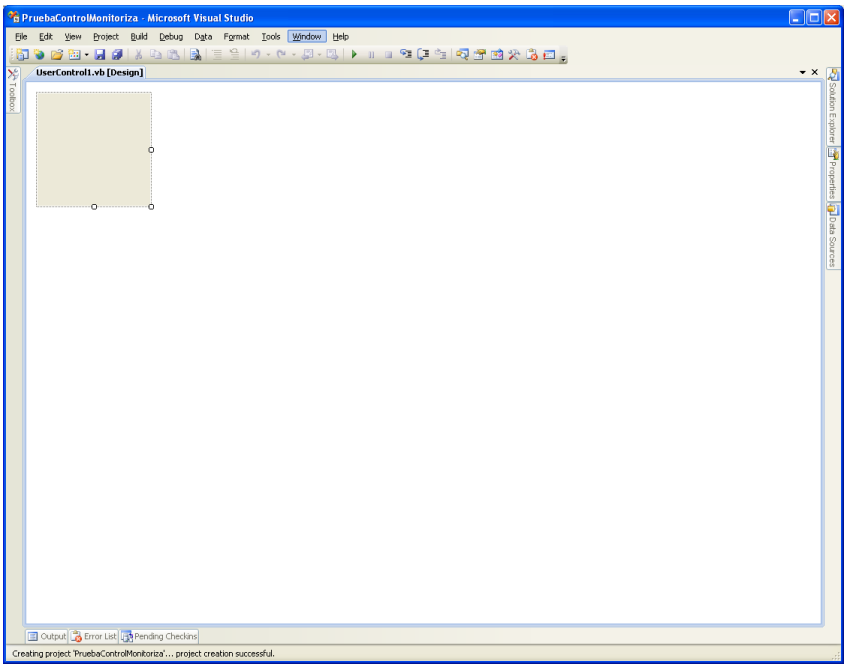
```
        Microsoft.VisualBasic.MsgBox("Interval de Temporizador1: " &
temp.Interval)

    End Sub
End Class
```

LIBRERIA DE CONTROLES

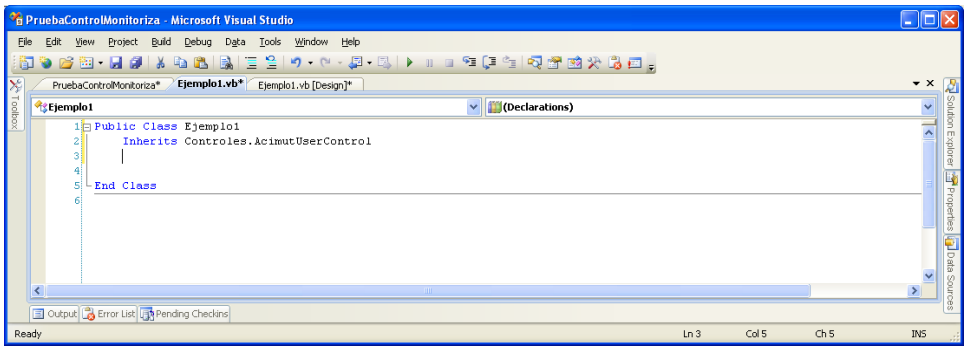
De la misma forma que se puede asociar código externo a los eventos de los controles de Monitoriza, también es posible incluir directamente controles diseñados en una aplicación externa dentro de Monitoriza. Estos son los ***UserControl***.

Para usar estos controles, debemos crear un proyecto de tipo Librería de controles de Windows Forms en Visual Studio.



A esta librería se le añaden los controles de usuario que posteriormente usaremos en Monitoriza.

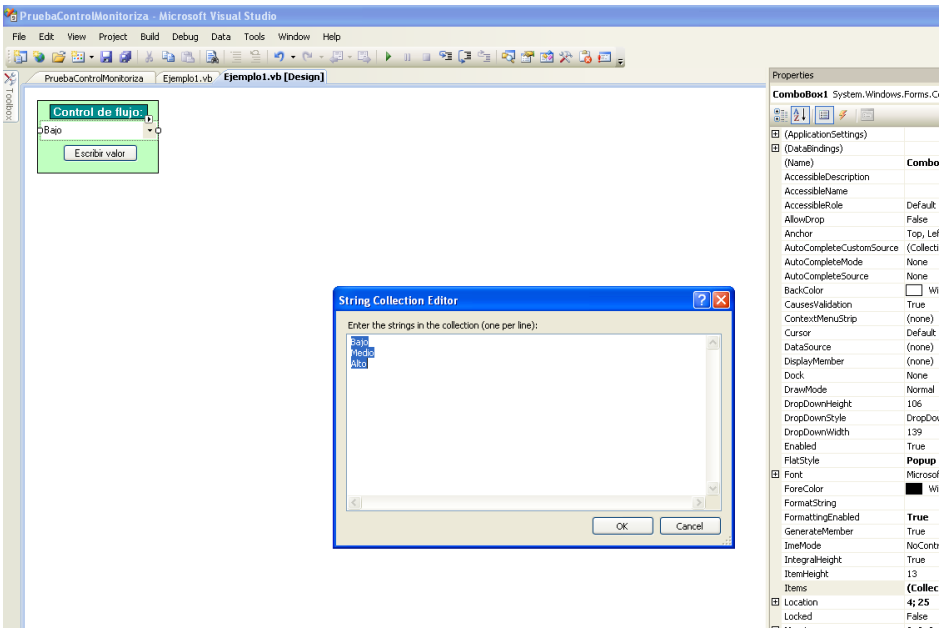
Para poder usar estos controles en Monitoriza correctamente, es necesario incluir la referencia a Controles.dll de Monitoriza (Proyecto – Añadir referencia) e indicar a cada uno de los controles que debe heredar la clase principal de Controles.AcimutUsercontrol.



Visual Studio nos informará que no es posible heredar de AcimutUsercontrol y de UserControl (nativo) al mismo tiempo, por lo que deberemos indicarle que debe heredar únicamente de AcimutUserControl.

Veamos un sencillo ejemplo de creación de un UserControl y su posterior uso en Monitoriza.

Una vez creado el proyecto vacío, añadimos un control de usuario llamado Ejemplo1. En este control añadimos una etiqueta, un botón y un combo con 3 ítems:



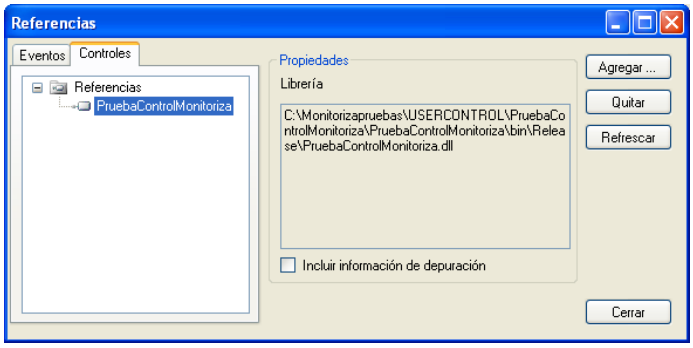
Añadimos algo de código al control:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim ValorAEscribir As Integer = 0
        'Seleccionamos el valor a escribir en la variable en funcion de
lo que se escoja en el combo
        'los 3 valores permitidos son BAJO,MEDIO y ALTO.
        Select Case Me.ComboBox1.Text
            Case "Bajo"
                ValorAEscribir = 10
                 Me.BackColor = Color.Coral
            Case "Medio"
                ValorAEscribir = 50
                 Me.BackColor = Color.Yellow
            Case "Alto"
                ValorAEscribir = 100
                 Me.BackColor = Color.Green
            Case Else
                ValorAEscribir = 0
        End Select
        'Asignamos el objeto frm al formulario padre, que en este caso es
el formulario de Monitoriza
        'que esté usando este usercontrol.
        Dim frm As Controles.Formulario = Me.TopLevelControl
        frm.Variables(Me.Servidor, Me.Grupo, Me.Variable)= ValorAEscribir
End Sub
```

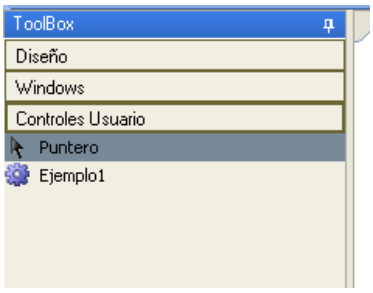
De esta forma provocamos que cuando pulsamos el botón se asigne un valor a la variable que se le ha indicado al UserControl en Monitoriza, acción que se realiza en la línea:

```
frm.Variables(Me.Servidor, Me.Grupo, Me.Variable) = ValorAEscribir
```

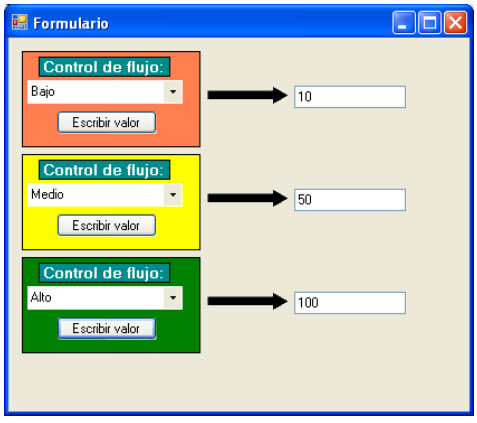
Una vez tengamos a nuestro gusto el control y hayamos generado la DLL correspondiente (Proyecto – Generar), debemos incluir el control en Monitoriza, para ello usaremos la pantalla de Referencias:



Al validar esta pantalla, en la barra de controles nos aparecerá un nuevo control en el grupo Controles Usuario:



Este control se puede usar como cualquier otro control nativo de Monitoriza. Una vez incluido en el formulario deseado, rellenamos las propiedades del grupo Scada (Servidor, Grupo, Variable, etc…) y ya está listo para usarse.

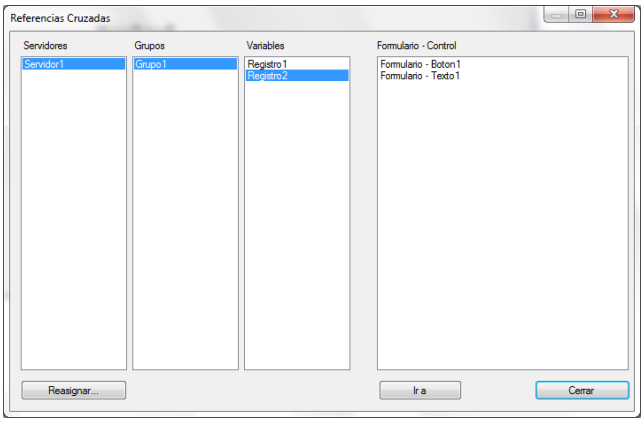


De la misma forma que las librerías de eventos, los UserControls pueden acceder a las ***variables y los componentes*** de Monitoriza usando el mismo método que el señalado en anteriormente en el apartado: **Acceso a variables y componentes.**

REFERENCIAS CRUZADAS

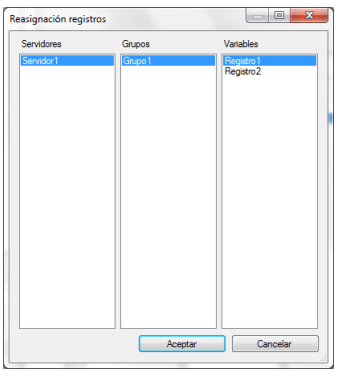
Acimut Monitoriza incluye una aplicación muy útil llamada Referencias Cruzadas en la que es posible mostrar donde se están usando las variables definidas en la aplicación.

Si queremos conocer donde, en que formulario y control, se usa una variable, en cualquier parte de la aplicación, podemos usar esta aplicación. Para ello la abriremos desde el menú *Ver* de la aplicación.



Primero debemos seleccionar el **Servidor** que nos interese, una vez seleccionado, automáticamente aparecerán los **grupos** de variables de ese servidor en la columna siguiente, seleccionaremos el grupo que nos interese y en la siguiente columna aparecerán los **registros** que pertenecen a ese grupo. De nuevo debemos seleccionar el registro deseado y en la última columna aparecerán los controles (formulario y control) que tienen asignado ese registro. Si lo deseamos podemos ver el formulario directamente pulsando sobre el botón '*Ir a*'.

En el caso que queramos cambiar el registro asignado a uno o varios controles podemos usar la opción 'Reasignar…'. Para usar esta funcionalidad, primero debemos seleccionar los controles a los que queramos cambiar el origen (admite selección múltiple), y pulsamos el botón *'Reasignar…'*, una nueva pantalla aparecerá:



En esta pantalla debemos seleccionar el nuevo registro que queremos usar en los controles seleccionados. Esto causará que los controles seleccionados en la pantalla anterior cambien la información de su registro asignado.

APÉNDICE

TABLA DE ALMACENAMIENTO DE ALARMAS

A continuación mostramos la definición de la tabla de almacenamiento de Alarmas tal y como se define en SQL Server.

```
CREATE TABLE [Alarmas] (
     [ID] [int] IDENTITY (1, 1) NOT NULL,
     [Servidor] [varchar] (50) NULL,
     [Grupo] [varchar] (50) NULL ,
     [Variable] [varchar] (50) NULL ,
     [TextoAlarma] [varchar] (250) NULL ,
     [Accion] [varchar] (50) NULL ,
     [Valor] [varchar] (50) NULL ,
     [Usuario] [varchar] (50) NULL ,
     [FechaHora] [datetime] NULL ,
     [Numerico] [int] NULL ,
     [Sencillo] [float] NULL
) ON [PRIMARY]
GO

ALTER TABLE [Alarmas] WITH NOCHECK ADD
     CONSTRAINT [PK_Alarmas] PRIMARY KEY  CLUSTERED
     (
            [ID]
     )  ON [PRIMARY]
GO

 CREATE     INDEX  [IX_Alarmas]  ON  [Alarmas]([Servidor],  [Grupo],  [Variable],
[TextoAlarma]) ON [PRIMARY]
GO

 CREATE  INDEX [IX_Alarmas_1] ON [Alarmas]([FechaHora]) ON [PRIMARY]
GO
```

TABLA DE ALMACENAMIENTO DE AUDITORÍA

A continuación mostramos la definición de la tabla de almacenamiento de para los registros de Auditoría tal y como se define en SQL Server.

```
CREATE TABLE [dbo].[Auditoria] (
            [ID] [int] IDENTITY (1, 1) NOT NULL ,
            [Computer] [varchar] (50) NULL ,
            [UserName] [varchar] (50) NULL ,
            [ActionAudited] [varchar] (50) NULL ,
            [Parameter1] [varchar] (250) NULL ,
            [Parameter2] [varchar] (250) NULL ,
            [Parameter3] [varchar] (250) NULL ,
            [DateTime] [datetime] NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Auditoria] ADD
            CONSTRAINT [PK_Auditoria] PRIMARY KEY  CLUSTERED
            (
                        [ID]
            )  ON [PRIMARY]
GO
```

RANGO DE REGISTROS MODBUS

Rangos de Registros Modbus

Autómatas y dispositivos Modbus utilizan registros para almacenar datos. Los registros se dividen en bloques de 65536 unidades. Monitoriza utiliza rangos entre 0 y 65535. Diferentes tipos de datos se almacenan en cada bloque.

Cuando se desea leer o escribir en un registro Modbus, se debe tener información específica, sobre el módulo o autómata, para saber a qué grupo Modbus pertenece la información y por lo tanto qué tipo de grupo deberemos utilizar en Monitoriza.

Los registros Modbus son de 16 bits (una palabra), mientras que las marcas (coils) son datos de 1 bit. En todo caso Monitoriza provee la posibilidad de leer de forma directa palabras dobles y de tipo single (float).

Vemos a continuación una tabla de los diferentes rangos de registros Modbus y el tipo de grupo a utilizar en Monitoriza.

Modbus Register Range	Modbus	Start Digit	Monitoriza Modbus Function Code
000001 – 065536	**Output Coils (Rango de 0 - 65535)** (For example, Digital input points and digital output points.)	0	ReadCoilStatus WriteCoil
100001 – 165536	**Input Discretes (Rango de 0 – 65535)** (For example, Digital input points.)	1	ReadInputStatus (Solo lectura)
300001 – 365536	**Input Registers (Rango de 0 - 65535)** (For example, Digital input points and analog input points.)	3	ReadInputRegisters (Solo lectura)
400001 – 465536	**Holding Registers (Rango de 0 - 65535)** (For example, Digital input points, digital output points, analog inputs, and analog output points.)	4	ReadHoldingRegisters WriteMultipleRegisters

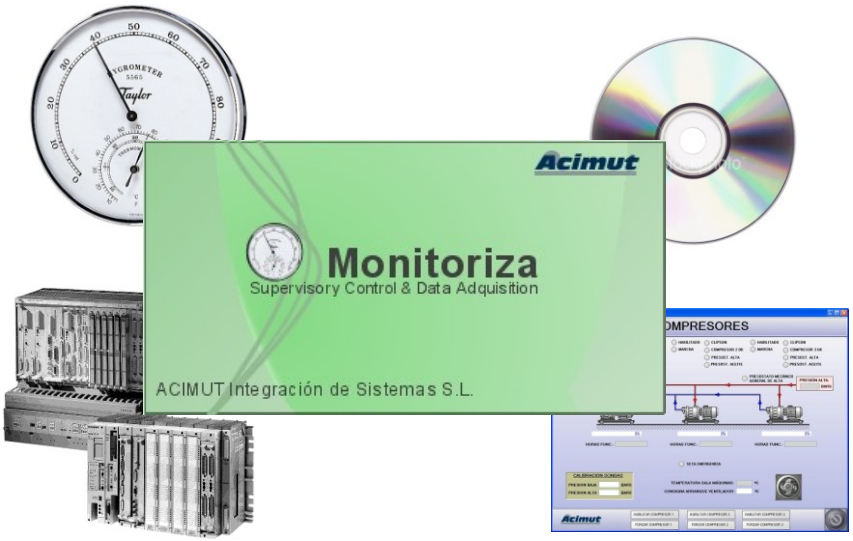
Funciones ModBUS usadas por Monitoriza

El protocolo Modbus usa diferentes funciones para cada tipo de registro, las funciones que se usan en Monitoriza se encuentran en la siguiente tabla:

Modbus Function Code	Funcionalidad
FC1	Read Coil Status
FC2	Read Input Status
FC3	Read Holding Registers
FC4	Read Input Registers
FC5	Write Coil
FC16	Write Multiple Registers

Las funciones que no se utilzan son:

FC6	Write Single Register
FC15	Write Multiple Coils



Revisión Octubre 2012
Acimut Integración de Sistemas