UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS

Carrera de Ingeniería en Sistemas Computacionales e Informáticos

TEMA:

DISEÑO DE UN SISTEMA SCADA PARA LA AUTOMATIZACIÓN DE UN PISO DE UNA EDIFICACIÓN, USANDO UNA TARJETA DE ADQUISICIÓN DE DATOS PARA LA EMPRESA M&B

Proyecto de Graduación modalidad Pasantía presentada como requisito previo a la obtención del Título de Ingeniero en Sistemas Computacionales e Informáticos.

AUTOR: Mario Fernando Torres Cortés

TUTOR: Ing. David Omar Guevara Aulestia Msc

Ambato – Ecuador

diciembre/2007

APROBACIÓN DEL TUTOR

En calidad de Tutor del Trabajo de Investigación sobre el tema:

"DISEÑO DE UN SISTEMA SCADA PARA LA AUTOMATIZACIÓN DE UN PISO DE UNA EDIFICACIÓN, USANDO UNA TARJETA DE ADQUISICIÓN DE DATOS PARA LA EMPRESA M&B" de Mario Fernando Torres Cortés, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos de la Facultad de Ingeniería en Sistemas, Universidad Técnica de Ambato, considero que dicho informe investigativo reúne los requisitos y méritos suficientes para ser sometidos a la evaluación de conformidad al Art. 68 del Reglamento de Pregrado de la Universidad Técnica de Ambato.

Ambato, diciembre 2007

EL TUTOR

_____

Ing. David Guevara

DEDICATORIA

A mis padres que con esfuerzo y ahínco han sabido apoyarme y orientarme en todas las etapas de mi vida, razón por la cual doy fe de su meritoria labor como padres y educadores. A mis hermanas y hermano por ser guías de mi camino.

AGRADECIMIENTO

Agradezco al Ser que me dio la oportunidad de vivir y compartir estos momentos, rodeado de grandes personalidades que me han enseñado que con sencillez y humildad en el trabajo y en la misma vida se puede llegar a ser una persona de bien, un ente constructivo para el medio en el que nos toca vivir, libre en todo el sentido de la palabra y con pensamientos que permitan llevar una vida integra y trascendente.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

Resumen ejecutivo

Después de haber analizado minuciosamente los distintos enfoques que han tomado los sistemas SCADA en la actualidad, y luego de haber recolectado las partes necesarias que comprenden dicho Sistema, se ha desarrollado un Sistema SCADA que permita automatizar espacios físicos fácilmente, gracias a las interfaces de usuario que el sistema contiene, por lo que el sistema permite implementar aplicaciones domóticas de una manera sencilla y transparente para el usuario.

El presente trabajo investigativo integra protocolos de comunicación, interfaces de usuario, manejo de base de datos y a breves rasgos se detallan tópicos del área electrónica. Se ha desarrollado el Sistema con herramientas libres, debido a sus grandes prestaciones que estas tienen al momento de realizar una investigación, obteniendo un desarrollo de un sistema SCADA con licencia GPL, el mismo que presta servicios que mejora la calidad de vida de los usuarios.

El presente trabajo permitirá tener una visión más clara de los alcances que tienen los Sistemas SCADA en la actualidad, sirviendo de esta manera como guía de aprendizaje a investigadores interesados en esta área.

Introducción

El compendio de información que existe en el mundo sobre sistemas SCADA es muy limitado, ya que muchos de estos sistemas se cierran a tecnologías propietarias, razón por la cual obliga a los usuarios a adquirir el software que a más de ser de uso exclusivo, tiene ligamientos en su uso, obligando a los usuarios a hacer uso de tecnologías que por lo general son costosas, motivo para que se decidiera realizar un sistema que sirva como base para futuras investigaciones y trabajos, el mismo que no es exclusivo de una tecnología o un problema específico, sino que globaliza y fusiona los diversos campos de la automatización, dotando de una herramienta útil a los futuros usuarios.

Pensando en que son pocas las empresas que han sido capaces de implantar sistemas exportados, extremadamente costosos, se busca dar alternativas a las empresas y o grupos sociales que promuevan el uso de tecnologías útiles para la vida humana.

CAPITULO I

PROBLEMA DE INVESTIGACIÓN

1.1 TEMA DE INVESTIGACIÓN

Diseño de un Sistema SCADA para la automatización de un piso de una edificación, usando una Tarjeta de Adquisición de Datos para la Empresa M&B.

1.1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.2 Contextualización

SCADA (Supervisory Control And Data Acquisition) es sinónimo de automatización en la actualidad. Las grandes Empresas han fijado su mirada en estos sistemas, debido a que actúan como Administradores de las actividades de producción y/o control, lo que convierte a los Sistema SCADA en uno de los Sistemas más cotizados.

En el Ecuador se está utilizando esta tecnología, pero es una área que no esta explotada en su totalidad, por lo que brinda nuevas oportunidades de trabajo.
Los Sistemas SCADA han tenido una gran acogida en la industria, pero también son utilizados en la automatización de edificios y hogares, debido a su crecimiento a nivel internacional y a las diferentes alternativas que prestan los Sistemas S CADA.
La automatización es una herramienta que hoy en día las empresas e industrias de la Provincia de Tungurahua están implantando, pero se ha creado una barrera entre las grandes empresas y empresas de menor envergadura, ya que implantar

Sistemas de Automatización requiere de una buena inversión de dinero que por más beneficios que esto contrae son egresos que sobrepasan de los límites de las pequeñas y medianas empresas.

La Empresa M&B carece de este Sistema, pero se ve envuelta en la necesidad de adquirir un Sistema SCADA, debido a las ventajas y oportunidades que este Sistema representa en la actualidad, con la finalidad de contribuir al progreso tecnológico del centro del país.

1.1.3 ANÁLISIS CRÍTICO

Las causas por las que no se cuenta con un diseño de un Sistema SCADA para la automatización de un piso de una edificación, usando una Tarjeta de Adquisición de Datos, es por el costo excesivo de los Sistemas SCADA, lo que imposibilita a la Empresa la compra, a corto plazo, de dicho Sistema.

Los Sistemas SCADA en su mayoría son genéricos, es decir tienen módulos que nunca se los utilizan, o carecen de módulos importantes, causando incertidumbre al momento de elegir un Sistema SCADA Comercial o Libre.

Los Sistemas SCADA actuales tienen una interfaz poco amigable para usuarios finales, los Sistemas están desarrollados para personas extremadamente técnicas, lo que impide comercializarlo fácilmente.

Otro factor es la indisponibilidad de tiempo de los trabajadores de la Empresa M&B, que por cuestiones de trabajo, más no de capacidad, están ligados a otras áreas de estudio, lo que provoca el desaprovechamiento de herramientas Informáticas para la creación de nueva soluciones.

1.1.4 PROGNOSIS

Si la Empresa M&B carece de este Sistema SCADA, dejaría de lado la oportunidad de comercializar una aplicación, que hoy en día es requerida en muchas edificaciones modernas, dando lugar a que la empresa no capte más mercado.

1.1.5 FORMULACIÓN DEL PROBLEMA

¿Qué incidencia tendría el Diseño de un Sistema SCADA para la automatización de un piso de una edificación, usando una Tarjeta de Adquisición de Datos?

1.1.6 DELIMITACIÓN DEL PROBLEMA

Este Proyecto está enfocado al diseño del Sistema SCADA, por lo que su parte práctica será desarrollada y simulada en la Empresa M&B, la parte electrónica no se profundizará en detalle, debido a que, la placa para la adquisición de datos, ya esta elaborada y lo que se busca es integrar lo electrónico con lo informático, por medio de este proyecto.

Se trabajará en el período comprendido entre Marzo y Julio del 2007, con una población conformada de 3 personas, quienes se dividen en dos grupos, el primero que colaborará en la parte electrónica y el segundo grupo en la parte informática.

La finalidad del Proyecto es desarrollar un Sistema SCADA, capaz de mantenerse comunicado con la Tarjeta de Adquisición de Datos.

1.2 JUSTIFICACIÓN

Este proyecto nace, en primera instancia, como iniciativa del Gerente de la Empresa M&B, quien se ve en la necesidad de desarrollar un Sistema que

Controle y Monitoree los Datos que se generan en una Tarjeta de Adquisición de Datos, previamente desarrollada.

Posteriormente, se ha realizado una investigación exhaustiva de las actuales tendencias informáticas, la misma que desvela la tendiente fusión de distintas áreas técnicas, tal es el caso de la Electrónica y la Informática, que sumadas y bien dirigidas, se convierten en un pilar para las actuales soluciones tecnológicas y técnicas que el medio requiere.

Los Sistemas SCADA, por sus características, son aptos para ser usados en la automatización de edificaciones, gracias a los diferentes protocolos de comunicación existentes, que permiten enlazar a las computadoras con dispositivos electrónicos, que en el caso de estudio es una tarjeta de adquisición de datos.

El Sistema propone hacer más placentera las tareas habituales que se realizan dentro del hogar y permitirá tener un control de las alarmas que se han generado, detallando claramente al usuario el origen de la misma.

1.3 OBJETIVO

1.3.1 OBJETIVO GENERAL

Diseñar un Sistema SCADA para la automatización de un piso de una edificación, usando una Tarjeta de Adquisición de Datos.

1.3.2 OBJETIVOS ESPECÍFICOS

- Analizar el funcionamiento del un Servidor OPC.
- Investigar el funcionamiento de un Cliente OPC.
- Establecer los requerimientos del Sistema SCADA.

- Definir el Lenguaje de Programación apropiado para el desarrollo del Sistema SCADA.
- Diseñar una base de Datos que almacene las variables requeridas por el Sistema.
- Seleccionar un motor de Base de Datos.
- Diseñar el Sistema.

CAPITULO II

MARCO TEÓRICO

2.1 ANTECEDENTES INVESTIGATIVOS

Después de buscar información en el Internet, se tiene como resultado la información siguiente;

2.1.1 SISTEMAS SCADA Y DISEÑO DE HMI CON INTOUCH, Dr. Luis Corrales.

En los Sistemas SCADA, usualmente existe una computadora que efectúa tareas de supervisión y gestión de alarmas, así como tratamiento de datos y control supervisado de procesos.

La comunicación en los sistemas SCADA se realiza mediante enlaces alámbricos, enlaces inalámbricos, buses especiales o redes LAN. Todo esto se ejecuta normalmente en tiempo que se puede considerar como real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos.

En este trabajo investigativo se utilizará enlaces alámbricos.

2.1.2 MANUAL DE LA TARJETA DE ADQUISICIÓN DE DATOS MB001a, M&B

CARACTERÍSTICAS PRINCIPALES

Fuente de Alimentación: 24 $V_{DC,}$ 1.5A.

Entradas / Salidas:

Entradas y Salidas	Cantidad
Entradas Digitales	16
Entradas Análogas	4
Salidas Digitales a Transistor	8
Salidas PWM a transistor	2

Tabla 2.1: Resúmen de Disposición de Variables de la Tarjeta de Adquisición de Datos.

Nota: Frecuencia de trabajo salidas PWM: 1.22 Khz.

Comunicación:

El protocolo de comunicación utilizado es MODBUS, cumple con las funciones de un esclavo MODBUS estándar.

Velocidad de Comunicación:

Para llevar a cabo la comunicación, la tarjeta dispone de un puerto de comunicaciones que cumple con las normas RS-232, capaz de intercambiar datos a las velocidades de:

- 2400 bps
- 4800 bps
- 9600 bps
- 19200 bps

La velocidad de comunicación deberá ser configurada mediante las diferentes posiciones del switch que se halla ubicado a la izquierda de las entradas digitales.

2.2 FUNDAMENTACIÓN LEGAL

Software:

- MySQL, motor de Base de Datos con licencia GPL (*General Public License* o licencia pública general, es una licencia creada por la Free Software Foundation, está orientada a la protección de la libre distribución, modificación y uso del software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios).

- Connector/Net 5.0, componente MySql, tiene licencia GPL.

- Sharp Develop 2.1 Beta, Aplicación que permite desarrollar software, tiene licencia LGPL(*Lesser General Public License* o *Library General Public License*, que permite el enlace dinámico de aplicaciones libres a aplicaciones no libres).

- Automated Solution Modbus (Sever OPC), el mismo que funciona como Servidor OPC, es un programa que se usa con período de prueba.

- Intellution's OPC Data Access Object (OpcData.ocx), que sirve para Cliente del Servidor OPC, es una librería gratuita.

Leyes y Acuerdos de M&B.

Reglamento a la Ley de Ejercicio Profesional de la Ingeniería, Acuerdo Ministerial 2, publicado en el Registro Oficial 257 de 18 de Enero de 1977:

Art. 4.- Están amparados por la Ley y sus Reglamentos los profesionales de las siguientes ramas de la Ingeniería: Agrícola, Agronómica, **Eléctrica y Electrónica**, Geología de Minas y de Petróleos, Industrial; Mecánica; Naval; Química; en Alimentos; Geográfica; Textiles; en Recursos Naturales Renovables; Agroindustriales; Administradores de Empresas Agropecuarias; e Industrias Agropecuarias, Forestal, Zootecnia; e Informática, Sistemas y Computación.

Art. 21.- Corresponde a los ingenieros legalmente autorizados para el ejercicio de su profesión, realizar en forma privativa las siguientes actividades:

a) Estudios de Ingeniería: preliminares, prefactibilidad, factibilidad y definitivos o diseño; anteproyectos, investigaciones, planificaciones, consultas, asesoría, mediciones, análisis, especificaciones, control de producción, establecimiento de métodos y sistemas, y transportes;

b) Planificación, presupuestos, avalúos, construcciones y asesoría en obras y trabajos de ingeniería;

c) Dirección, supervisión y fiscalización de estudios y obras de Ingeniería;

d) Instalación, operación y mantenimiento de equipos y sistemas;

e) Informes técnicos de Ingeniería, dirección, ejecución y control de producción;

f) Control de calidad y normalización;

g) Docencia, en las materias propias de la Ingeniería; y,

h) Cualquier otra actividad no especificada que, por su naturaleza u objetivo, requiera de conocimientos de profesionales ingenieros.

2.3 CATEGORÍAS FUNDAMENTALES

2.3.1 SISTEMA SCADA

Un sistema SCADA incluye una señal de entrada y salida, un hardware, controladores, interfase hombre-maquina, redes, comunicaciones y software.

El termino SCADA usualmente se refiere al sistema central que monitorea y controla un sitio completo o un sistema de despliegue de larga distancia. El sitio de control es diseñado automáticamente por una unidad-terminal remota (RTU) o

por un Controlador Lógico Programable (PLC). El control cíclico de retroalimentación es cerrado a través del RTU o el PLC; el sistema SCADA monitorea el desempeño en conjunto y su retorno.

La adquisición de datos inicia al nivel del RTU o del PLC e incluye lectores de medidores y equipo de estado que están comunicados con SCADA según su requerimiento. Los datos son recopilados y formateados de tal manera que un operador en el centro de control usando la interfase hombre-maquina IHM puede supervisar apropiadamente decisiones que pueden ser requeridas para ajustar o normalizar el sobre flujo en los controles RTU (o PLC).

Los sistemas SCADA, comúnmente implementados en bases de datos distribuidas que contienen elementos de datos llamados puntos. Un punto representa un valor de salida o entrada monitoreado o controlado por el sistema. Los puntos pueden ser "duros" o "blandos". Un punto duro es representativo de una entrada o salida actual conectada al sistema, mientras que un punto blando representa el resultado de operaciones lógicas o matemáticas aplicadas a puntos duros y blandos.

Los valores de los puntos normalmente son guardados como combinaciones de valores y tiempos; el tiempo (fecha y hora) cuando fue guardado o calculado el valor. Una serie de combinaciones valores-tiempos es la historia de un punto.

Interfase Hombre - Máquina

Una interfase Hombre - Máquina o HMI (*Human Machine Interface*) es el dispositivo que presenta datos a un operador (humano), quien controla el proceso.

Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnostico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o maquina en particular, incluso sistemas expertos con guía de resolución de problemas. Cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no

propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLCs, incluyendo la entrada al mercado de ingenieros mecánicos, eléctricos y técnicos para configurar estas interfases por sí mismos, sin la necesidad de un programa escrito por un desarrollador de software.

SCADA es popular debido a esta compatibilidad y seguridad. Ésta se usa desde aplicaciones pequeñas, como controladores de temperatura en un espacio, hasta aplicaciones muy grandes como el control de plantas nucleares.

Soluciones de Hardware

El uso de RTUs o PLCs sin involucrar computadoras maestras está aumentando, los cuales son autónomos ejecutando procesos de lógica simple. Frecuentemente se usa un lenguaje de programación funcional para crear programas que corran en estos RTUs y PLCs, siempre siguiendo lo estándares de la norma IEC 61131-3. La complejidad y la naturaleza de este tipo de programación hace que los programadores necesiten cierta especialización y conocimiento sobre los actuadores que van a programar. Aunque la programación de estos elementos es ligeramente distinta a la programación tradicional, también se usan lenguajes que establecen procedimientos, como pueden ser FORTRAN, C o ADA95. Esto les permite a los ingenieros de sistemas SCADA implementar programas para ser ejecutados en RTUs o un PLCs.

Componentes del Sistema

Los tres componentes de un sistema SCADA son:

1. Múltiples Unidades de Terminal Remota (también conocida como RTU o Estaciones Externas).
2. Estación Maestra y Computador con HMI.
3. Infraestructura de Comunicación.

Unidad de Terminal Remota (RTU)

La RTU se conecta al equipo físicamente y lee los datos de estado como los estados abierto/cerrado desde una válvula o un intercambiador, lee las medidas como presión, flujo, voltaje o corriente. Por el equipo el RTU puede enviar señales que pueden controlarlo: abrirlo, cerrarlo intercambiarlo la valvular o configurar la velocidad de la bomba.

El RTU puede leer el estado de los datos digital o medidas de datos análogos y envía comandos digitales de salida o puntos de ajuste análogos.

Una de las partes más importantes de la implementación de SCADA son las alarmas. Una alarma es un punto de estado digital que tiene cada valor NORMAL o ALARMA. La alarma se puede crear en cada paso que los requerimientos lo necesiten. El operador de SCADA pone atención a la parte del sistema que lo requiera, por la alarma. Pueden enviarse por correo electrónico o mensajes de texto con la activación de una alarma, alertando al administrador o incluso al operador de SCADA.

Estación Maestra

Se refiere a los servidores y el software responsable para comunicarse con el equipo del campo (RTUs, PLCs, etc.) en estos se encuentra el software HMI corriendo para las estaciones de trabajo en el cuarto de control, o en cualquier otro lado. En un sistema SCADA pequeño, la estación maestra puede estar en un solo computador, A gran escala, en los sistemas SCADA la estación maestra puede incluir muchos servidores, aplicaciones de software distribuido, y sitios de recuperación de desastres.

El Sistema SCADA usualmente presenta la información al personal operativo de manera gráfica. Esto significa que el operador puede ver un esquema que representa la planta que está siendo controlada. Por ejemplo un dibujo de una bomba conectada a la tubería puede mostrar al operador cuanto fluido esta siendo bombeado desde la bomba a través de la tubería en un momento dado. El operador

puede cambiar el estado de la bomba a apagado. El software HMI mostrara el promedio de fluido en la tubería decrementándose en tiempo real.

El paquete HMI para el sistema SCADA típicamente incluye un programa de dibujo con el cual los operadores o el personal de mantenimiento del sistema usan para cambiar la manera que estos puntos son representados en la interfase. Esta representaciones puede ser tan simple como una luces de tráfico en pantalla, los cuales representan el estado actual de un campo en el tráfico actual, o tan complejo como una pantalla de multiproyector representado posiciones de todos los elevadores en un rascacielos o todos los trenes de una vía férrea.

2.3.2 PROTOCOLOS DE COMUNICACIONES

El protocolo es “el idioma o formato de los mensajes que los diferentes elementos de control del sistema deben utilizar para entenderse unos con otros y que puedan intercambiar su información de una manera coherente”
Los protocolos de comunicaciones se pueden clasificar en:
Protocolos estándar. Desarrollados por fabricantes para que puedan ser utilizados abiertamente por empresas o terceras personas que empleen productos compatibles entre sí y por lo general están respaldados por diferentes organizaciones.
La ventaja principal que proporcionan estos protocolos es la capacidad para implementar o configurar una instalación domótica y su posible ampliación debido a la compatibilidad en el estándar que pueden poseer diversos equipos de distintos fabricantes pero resultan ser más costosos que los equipos de tecnología propietaria.

Algunos de los protocolos estándar más utilizados son: EIB, EHS, X-10,
Lonworks, Batibus, Modbus entre otros.

Protocolos propietarios. Son desarrollados por empresas en la que sus equipos solo son compatibles con otros productos y sistemas del mismo fabricante. Los

protocolos propietarios poseen ventaja frente a los estándar en cuanto a la economía y costo de los equipos pero resulta un riesgoso emplear un solo tipo de tecnología, pues si la empresa desaparece entonces no se puede seguir teniendo soporte técnico ni posibilidades para ampliaciones futuras.
Los protocolos más comunes son: Simon Vis, Domaike, Amigo, Biodom,
Cardio, entre otros.

2.3.3 MODBUS

Es un protocolo de comunicaciones situado en el nivel 7 del Modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs). Convertido en un protocolo de comunicaciones estándar de facto en la industria es el que goza de mayor disponibilidad para la conexión de dispositivos electrónicos industriales. Las razones por las cuales el uso de Modbus es superior a otros protocolos de comunicaciones son:

1. Es público.
2. Su implementación es fácil y requiere poco desarrollo.
3. Maneja bloques de datos sin suponer restricciones.

Modbus permite el control de una red de dispositivos, por ejemplo un sistema de medida de temperatura y humedad, y comunicar los resultados a un ordenador. Modbus también se usa para la conexión de un ordenador de supervisión con una unidad remota (RTU) en sistemas de supervisión adquisición de datos (SCADA). Existen versiones del protocolo Modbus para puerto serie y Ethernet (Modbus/TCP).

Modbus RTU es una representación binaria compacta de los datos. Modbus ASCII es una representación legible del protocolo pero menos eficiente. Ambas implementaciones del protocolo son serie. El formato RTU finaliza la trama con un suma de control de redundancia cíclica (CRC), mientras que el formato ASCII utiliza una suma de control de redundancia longitudinal (LRC). La versión

Modbus/TCP es muy semejante al formato RTU, pero estableciendo la transmisión mediante paquetes TCP/IP.

Modbus Plus (Modbus+ o MB+), es una versión extendida del protocolo que permanece propietaria de Modicon. Dada la naturaleza de la red precisa un coprocesador dedicado para el control de la misma. Con una velocidad de 1 Mbit/s en un par trenzado sus especificaciones son muy semejantes al estándar EIA/RS-485 aunque no guarda compatibilidad con este.

Cada dispositivo de la red Modbus posee una dirección única. Cualquier dispositivo puede enviar órdenes Modbus, aunque lo habitual es permitirlo sólo a un dispositivo maestro. Cada comando Modbus contiene la dirección del dispositivo destinatario de la orden. Todos los dispositivos reciben la trama pero sólo el destinatario la ejecuta (salvo un modo especial denominado "Broadcast"). Cada uno de los mensajes incluye información redundante que asegura su integridad en la recepción. Los comandos básicos Modbus permiten controlar un dispositivo RTU para modificar el valor de alguno de sus registros o bien solicitar el contenido de dichos registros.

Existe gran cantidad de modems que aceptan el protocolo Modbus. Algunos están específicamente diseñados para funcionar con este protocolo. Existen implementaciones para conexión por cable, wireless, SMS o GPRS. La mayoría de problemas presentados hacen referencia a la latencia y a la sincronización.

Variaciones

Todas las implementaciones presentan variaciones respecto al estándar oficial. Algunas de las variaciones más habituales son:

Tipos de Datos

- Coma Flotante IEEE.
- Entero 32 bits.
- Datos 8 bits.

- Tipos de datos mixtos.
- Campos de bits en enteros.
- Multiplicadores para cambio de datos a/de entero. 10, 100, 1000, 256 ...

Extensiones del Protocolo

- Direcciones de esclavo de 16 bits.
- Tamaño de datos de 32 bits (1 dirección = 32 bits de datos devueltos).

2.3.4 PUERTO SERIE

Es una interfaz de comunicaciones entre ordenadores y periféricos en donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits a la vez. Entre el puerto serie y el puerto paralelo, existe la misma diferencia que entre una carretera tradicional de un sólo carril por sentido y una autovía con varios carriles por sentido.

Puerto Serie Tradicional

El puerto serie por excelencia es el RS-232 (también conocido como COM) que utiliza cableado simple desde 3 hilos hasta 25 y que conecta ordenadores o microcontroladores a todo tipo de periféricos, desde terminales a impresoras y modems pasando por ratones.

La interfaz entre el RS-232 y el microprocesador generalmente se realiza mediante el integrado 82C50.

El RS-232 original tenía un conector tipo D de 25 pines, la mayoría de dichos pines no se utilizaban, por lo que IBM incorporó desde PS/2 un conector más pequeño de solamente 9 pines que es el que actualmente se utiliza.

Puertos Serie Actuales

Uno de los defectos de los puertos serie iniciales era su lentitud en comparación con los puertos paralelos, sin embargo, con el paso del tiempo, están apareciendo multitud de puertos serie de alta velocidad que los hacen muy interesantes ya que utilizan las ventajas del menor cableado y solucionan el problema de la velocidad. Por ello, el puerto RS-232 e incluso multitud de puertos paralelos están siendo reemplazados por nuevos puertos serie como el USB, el Firewire o el Serial ATA.

Un puerto de red puede ser puerto serie o puerto paralelo.

2.3.5 SHARPDEVELOP 2.2.1

Entorno integrado de desarrollo libre para los lenguajes de programación C#, Visual Basic .NET y Boo (programación).

Hay disponible un sitio para Mono/Gtk#, llamado MonoDevelop, el cual funciona en otros sistemas operativos.

Para el completado automático de código, la aplicación incorpora sus propios parsers.

Figura 2.1: Logo de Sharp Develop

Características principales

Incorpora un diseñador de Windows forms.

Completado de código. Soporta el uso de la combinación de teclas Ctrl + Espacio.
Depurador incorporado.
Herramientas para "Ir a Definición", "Encontrar referencias" y "renombrado".
Títulos para títulos y para depuración.
Conversor bidireccional entre C# y Visual Basic .NET, y unidireccional hacia Boo.
Escrito enteramente en C#.
Compilación de código directamente dentro del entorno de desarrollo integrado.
Complementos para ILAsm y C++.
Integración con herramientas de pruebas unitarias NUnit y MbUnit.
Analizador para ensamblado FxCop.
Previsualización de documentación XML.
Gran integración con plantillas a la hora de añadir o crear ficheros, proyectos o compiladores.
Escritura de código C#, ASP.NET, ADO.NET, XML y HTML.
Coloreado de sintáxis para los lenguajes C#, HTML, ASP, ASP.NET, VBScript, Visual Basic .NET, y XML.
Llaves inteligentes en la escritura de código.
Gestión de marcadores (favoritos).
Soporte para plantillas de código.
Extensible mediante herramientas externas, o complementos.

2.3.6 MYSQL 5.0

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado lo ofrece bajo la GNU GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. Está desarrollado en su mayor parte en ANSI C.

Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson, y Michael Widenius.

SQL (Lenguaje de Consulta Estructurado) fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde ese entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL:92, SQL:99, SQL:2003. MySQL es una idea originaria de la empresa opensource MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius. El objetivo que persigue esta empresa consiste en que MySQL cumpla el estándar SQL, pero sin sacrificar velocidad, fiabilidad o usabilidad.

Michael Widenius en la década de los 90 trató de usar mSQL para conectar las tablas usando rutinas de bajo nivel ISAM, sin embargo, mSQL no era rápido y flexible para sus necesidades. Esto lo conllevó a crear una API SQL denominada MySQL para bases de datos muy similar a la de mSQL pero más portable.

Existen varias APIs que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby, REALbasic (Mac), FreeBASIC, y Tcl; cada uno de estos utiliza una API específica. También existe un interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL.

Plataformas

MySQL funciona sobre múltiples plataformas, incluyendo AIX, BSD, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell Netware, OpenBSD, OS/2 Warp, QNX, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP,

Windows Vista y otras versiones de Windows. También existe MySQL para OpenVMS.

Figura 2.2: Logo de MySql

Características de la versión 5.0.22

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte a multiplataforma
- Procedimientos almacenados
- Triggers
- Cursors
- Vistas actualizables
- Soporte a VARCHAR
- INFORMATION_SCHEMA
- Modo Strict
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas, InnoDB para transacciones e integridad referencial)
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos de recuperación(savepoints) con InnoDB
- Soporte para SSL
- Puero caching
- Sub-SELECTs (o SELECTs anidados)
- Replication with one master per slave, many slaves per master, no automatic support for multiple masters per slave.

- Indexing y buscado campos de texto completos usando el motor de almacenamiento MyISAM
- Embedded database library
- Soporte completo para Unicode
- Conforme a las reglas ACID usando los motores InnoDB, BDB y Cluster
- Shared-nothing clustering through MySQL Cluster

Características adicionales

- Usa GNU Automake, Autoconf, y Libtool para portabilidad
- Uso de multihilos mediante hilos del kernel.
- Usa tablas en disco b-tree para búsquedas rápidas con compresión de índice
- Tablas hash en memoria temporales
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL
- Completo soporte para operadores y funciones en cláusulas select y where.
- Completo soporte para cláusulas group by y order by, soporte de funciones de agrupación
- Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).
- Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando named pipes y en sistemas Unix usando ficheros socket Unix.
- En MySQL 5.0, los clientes y servidores Windows se pueden conectar usando memoria compartida.

- MySQL contiene su propio paquete de pruebas de rendimiento proporcionado con el código fuente de la distribución de MySQL

Características distintivas

Las siguientes características son implementadas únicamente por MySQL:

- Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example en 5.x), permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

Tipos de compilación del servidor

Hay tres tipos de compilación del servidor MySQL:

- Estándar: Los binarios estándar de MySQL son los recomendados para la mayoría de los usuarios, e incluyen el motor de almacenamiento InnoDB.
- Max (No se trata de MaxDB, que es una cooperación con SAP): Los binarios incluyen características adicionales que no han sido lo bastante probadas o que normalmente no son necesarias.
- MySQL-Debug: Son binarios que han sido compilados con información de depuración extra. No debe ser usada en sistemas en producción porqué el código de depuración puede reducir el rendimiento.

Especificaciones del código fuente

MySQL está escrito en una mezcla de C y C++. Hay un documento que describe algunas de sus estructuras internas en http://dev.mysql.com/doc/internals/en/

2.3.7 LENGUAJE LADDER

También denominado lenguaje de contactos o en escalera, es un lenguaje de programación gráfico muy popular dentro de los autómatas programables debido a que está basado en los esquemas eléctricos de control clásicos. De este modo, con los conocimientos que todo técnico eléctrico posee, es muy fácil adaptarse a la programación en este tipo de lenguaje.

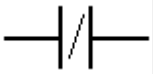
Elementos de programación

Para programar un autómata con **LADDER**, además de estar familiarizado con las reglas de los circuitos de conmutación, es necesario conocer cada uno de los elementos de que consta este lenguaje. A continuación se describen de modo general los más comunes.

Elementos básicos

En la siguiente tabla podemos observar los símbolos de los elementos básicos junto con sus respectivas descripciones.

Elementos básicos en LADDER

Símbolo	**Nombre**	**Descripción**
	Contacto NA	Se activa cuando hay un uno lógico en el elemento que representa, esto es, una entrada (para captar información del proceso a controlar), una variable interna o un bit de sistema.
	Contacto NC	Su función es similar al contacto NA anterior, pero en este caso se activa cuando hay un cero lógico, cosa que deberá de tenerse muy en cuenta a la hora de su utilización.
	Bobina NA	Se activa cuando la combinación que hay a su entrada (izquierda) da un uno lógico. Su activación equivale a decir que tiene un uno lógico. Suele representar elementos de salida, aunque a veces puede hacer el papel de variable interna.

	Bobina NC	Se activa cuando la combinación que hay a su entrada (izquierda) da un cero lógico. Su activación equivale a decir que tiene un cero lógico. Su comportamiento es complementario al de la bobina NA.

Tabla 2.2: Elementos Básicos de LADDER

Variables internas y bits de sistema

Se suele indicar mediante los caracteres B ó M y tienen tanto bobinas como contactos asociados a las mismas de los tipos vistos en el punto anterior. Su número de identificación suele oscilar, en general, entre 0 y 255. Su utilidad fundamental es la de almacenar información intermedia para simplificar esquemas y programación.

Los bits de sistema son contactos que el propio autómata activa cuando conviene o cuando se dan unas circunstancias determinadas. Existe una gran variedad, siendo los más importantes los de arranque y los de reloj, que permiten que empiece la ejecución desde un sitio en concreto y formar una base de tiempos respectivamente. Su nomenclatura es muy diversa, dependiendo siempre del tipo de autómata y fabricante.

Temporizadores

Permite activar bobinas pasado un cierto tiempo desde la activación. El esquema básico de un temporizador varía de un autómata a otro, pero siempre podemos encontrar una serie de señales fundamentales.

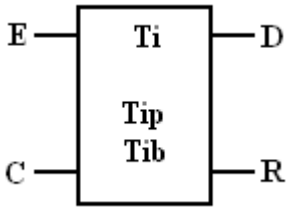


Figura 2.3: Imagen de un Temporizador de Ladder.

Podemos observar, en la figura de la derecha, el esquema de un temporizador, **Ti**, con dos entradas (E y C a la izquierda) y dos salidas (D y R a la derecha con las siguientes características:

Entrada Enable (E): Tiene que estar activa (a 1 lógico) en todo momento durante el intervalo de tiempo, ya que si se desactiva (puesta a cero lógico) se interrumpiría la cuenta de tibia (puesta a cero temporal).

Contadores

El contador es un elemento capaz de llevar el cómputo de las activaciones de sus entradas, por lo que resulta adecuado para memorizar sucesos que no tengan que ver con el tiempo pero que se necesiten realizar un determinado número de veces.

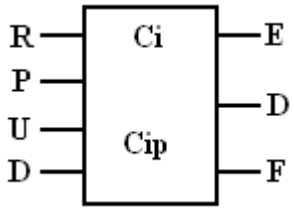


Figura 2.4: Contador Ladder

En la figura de la derecha puede verse el esquema de un contador, **Ci**, bastante usual, donde pueden distinguirse las siguientes entradas y salidas:

- **Entrada RESET** (R): Permite poner a cero el contador cada vez que se activa. Se suele utilizar al principio de la ejecución asignándole los bits de arranque, de modo que quede a cero cada vez que se arranca el sistema.

- **Entrada PRESET** (P). Permite poner la cuenta del contador a un valor determinado distinto de cero, que previamente se ha programado en Cip.

- **Entrada UP** (U): Cada vez que se activa produce un incremento en una unidad de la cuenta que posea en ese momento el contador.

- **Entrada DOWN** (D): Cada vez que se activa produce un decremento en una unidad de la cuenta que posea en ese momento el contador.

- **Salida FULL** (F): Se activa al producirse un desbordamiento del valor del contador contando en sentido ascendente.

- **Salida DONE** (D): Se activa cuando el valor del contador se iguala al valor preestablecido Cip.

- **Salida EMPTY** (E): Se activa al producirse un desbordamiento del valor del contador contando en sentido descendente.

Monoestables

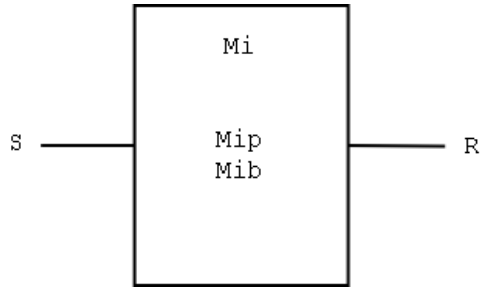


Figura 2.5: Contador Monoestable

El monoestable es un elemento capaz de mantener activada una salida durante el tiempo con el que se haya programado, desactivándola automáticamente una vez concluido dicho tiempo. Una de sus principales ventajas es su sencillez ya que sólo posee una entrada y una salida como podemos observar en la siguiente figura.

- **Entrada STAR** (S): Cuando se activa o se le proporciona un impulso comienza la cuenta que tiene programada.

- **Salida RUNNING** (R): Se mantiene activada mientras dura la cuenta y se desactiva al finalizarla. Al igual que con el temporizador, para programar la cuenta hay que introducir los valores de Mip y Mib.

Programación

El siguiente esquema representa la estructura general de la distribución de todo programa LADDER, contactos a la izquierda y bobinas y otros elementos a la derecha.

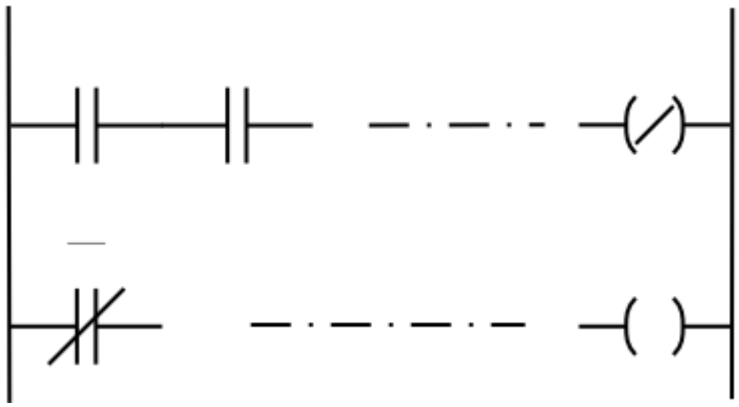


Figura 2.6: Distribución de un programa Ladder.

En cuanto a su equivalencia eléctrica, podemos imaginar que la línea vertical de la izquierda representa el terminal de alimentación, mientras que la línea vertical de la derecha representa el terminal de masa.

El orden de ejecución es generalmente de arriba a bajo y de izquierda a derecha, primero los contactos y luego las bobinas, de manera que al llegar a éstas ya se conoce el valor de los contactos y se activan si procede. El orden de ejecución puede variar de un autómata a otro, pero siempre se respetará el orden de introducción del programa, de manera que se ejecuta primero lo que primero se introduce.

Sistemas Combinacionales

Aunque en los sistemas industriales la programación se centra en procesos secuenciales, no teniendo demasiado interés los procesos combinacionales, es necesario conocer la lógica combinacional ya que en muchas ocasiones es necesaria en la programación secuencial.

Una vez obtenida la función lógica de un problema combinaciónal, el paso a LADDER o esquema de contactos es muy sencillo. De acuerdo con el álgebra de Boole aplicada a la conmutación, las sumas serán contactos en paralelo, los productos contactos en serie y las negaciones contactos normalmente cerrados. En la siguiente figura se muestra un ejemplo de esquema LADDER para una determinada ecuación.

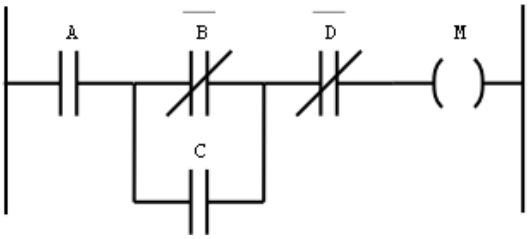


Figura 2.7: LADDER para la función M = A(B'+C)D'

Elementos de memoria

La conexión tradicional para realizar una función de memoria en los circuitos con relés, es el circuito con autoalimentación. Esto se consigue mediante la conexión de un contacto NA del relé (o contactor) en paralelo con el pulsador de marcha. A continuación puede observarse las dos variantes de este circuito: con prioridad a la conexión (figura 2.8) y con prioridad a la desconexión (figura 2.9).

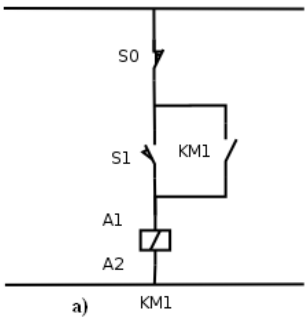


Figura 2.8: Prioridad a la conexión

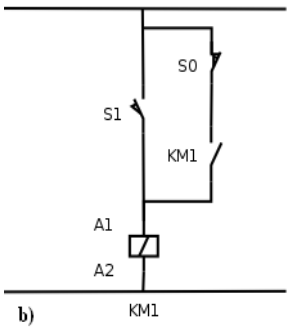


Figura 2.8: Prioridad a la desconexión

En la siguiente figura se pueden observar los esquemas equivalente en LADDER:

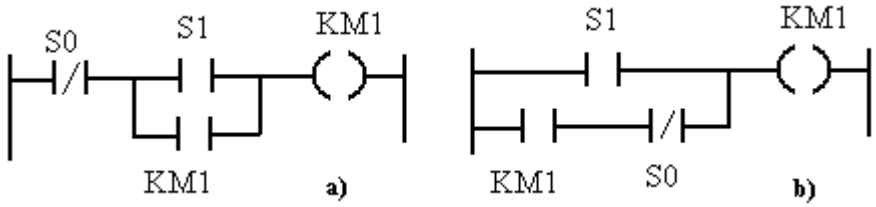


Figura 2.9: Circuitos LADDER con autoalimentación

Sin embargo, con LADDER el esquema puede quedar mucho más sencillo si empleamos las bobinas de SET para la marcha y RESET para paro:

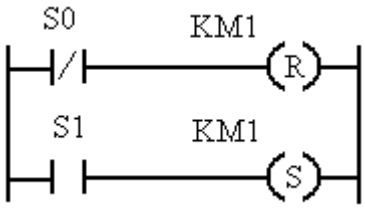


Figura 2.10: Circuito de marcha y paro con bobinas SET y RESET

En este caso la prioridad dependerá del PLC utilizado, aunque usualmente la función RESET tiene prioridad sobre la SET.

Elementos de tiempo

Los dos elementos básicos de tiempo son el temporizador y el monoestable. A continuación veremos un ejemplo de programación de un automatismo temporizado.

El esquema siguiente se corresponde con el mando de un motor con marcha temporizada.

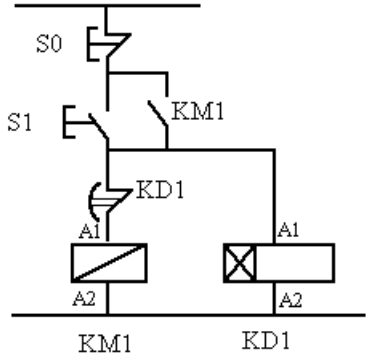


Figura 2.11: Automatismo temporizado.

Un posible programa equivalente en LADDER podría ser el siguiente:

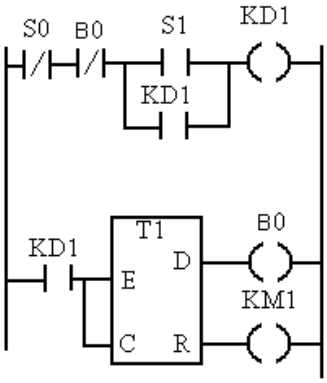


Figura 2.12: Aplicación de un temporizador en LADDER.

Elementos de cómputo

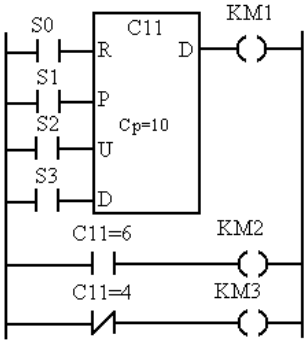


Figura 2.13: Ejemplo de programa LADDER de cómputo

Para aclarar la programación con elementos de cómputo, se explicará el funcionamiento del esquema de la derecha:

Como se puede observar, el programa consta de un contador C11 que ha sido programado con el valor 10 (Cp=10). Con la entrada S0 ponemos a cero el contador y con la entrada S1 se preselecciona con el valor de Cp, esto es, 10. Cada impulso dado en S2 incremaenta en una unidad el contador y cada impulso en S3 lo decrementa.

Las bobinas KMI y KM2 se activan cuando el contador posee el valor 10 y 6 respectivamente, en cambio, la bobina KM3 está continuamente activada excepto cuando el contador se encuentra con el valor 4.

2.3.8 FUNDAMENTACIÓN DE LA EMPRESA

En el año (1994) el ingeniero Fernando Vinicio Muñoz con una visión futurista, creyendo en sus capacidades y habilidades y con una misión preponderante, dar solución a problemas presentados en las distintas industrias ambateñas pero fundamentalmente demostrar que en el la ciudad de Ambato existen personas que pueden desarrollar proyectos de investigación, modernistas e innovadores que irán en beneficio del desarrollo de la productividad de empresas aportando de manera directa con el desarrollo de la ciudad por esta razón crea M&B Automatización La misma que desde sus inicios se dedico a dar solución a problemas de mantenimiento de máquinas cuyos procesos eran electrónicos, sin embargo el objetivo fundamental fu y es diseñar y construir sistemas eléctricos-electrónicos-informáticos para facilitar los diferentes procesos de producción, de esta manera M&B Automatización ha ido cumpliendo paso a paso su misión demostrando de que no solo la gente del extranjero o de las grandes ciudades ecuatorianas inventan o diseñan cosas nuevas permanentemente.

2.4 DETERMINACIÓN DE VARIABLES

2.4.1 VARIABLE INDEPENDIENTE

Sistema SCADA.

2.4.2 VARIABLE DEPENDIENTE

Empresa M&B.

2.5 HIPÓTESIS

El Diseño del Sistema SCADA para la Empresa M&B, permitirá automatizar un piso de una edificación, usando una tarjeta de adquisición de datos.

CAPITULO III

METODOLOGÍA

3.1 ENFOQUE

Delimitado el problema, establecido los objetivos y variables, es necesario seleccionar los métodos y técnicas a emplearse en este proyecto.

La forma de investigación que se ha llevado hasta el momento es de carácter cualitativo, debido a la existencia de cualidades de normativa, explicativa y realista.

3.2 MODALIDAD DE INVESTIGACIÓN

El presente Proyecto esta enmarcado en dos metodologías de investigación:

- Bibliográfico – Documental, debido a la información que se necesita recopilar, entre textos, obras, libros e Internet.

- Especial, debido a que el Desarrollo del Sistema SCADA es también de tipo creativo, puesto se encuentra dentro del área de Desarrollo de Software.

3.3 NIVELES O TIPO DE INVESTIGACIÓN

El tipo de Investigación predominante en este Proyecto es de tipo exploratorio, descriptivo y analítico para facilitar la comprobación de la hipótesis.

3.4 POBLACIÓN Y MUESTRA

Por ser el proyecto de características eminentemente de Diseño, no consta de una muestra, significativa para realizar el cálculo pertinente.

3.5 TÉCNICAS E INSTRUMENTOS DE INVESTIGACIÓN

Se utilizará la observación como técnica fundamental en este proceso investigativo. Además se utilizará a la entrevista para reforzar el conocimiento ya adquirido por otra persona.

3.6 PROCESAMIENTO DE INFORMACIÓN

Una vez aplicadas las técnicas de investigación, se dio paso a la integración del conocimiento y lineamiento del proyecto, lo que dio como resultado la solución al problema planteado.

Posteriormente se elaboraron las respectivas conclusiones y recomendaciones de este proyecto investigativo.

Parte importante al presente trabajo es la formulación de un tema de investigación que es el Diseño de un Sistema SCADA para la automatización de un piso de una edificación, usando una Tarjeta de Adquisición de Datos.

CAPITULO IV

ADMINISTRATIVO

4.1 Recursos

Se utilizaron los siguientes recursos humanos y materiales en el desarrollo del trabajo investigativo.

4.1.1 Institucionales

- Empresa M&B.
- UTA - FISEI, en casos excepcionales.

4.1.2 Humanos

- Personal Especializado en Tarjetas de Adquisición de Datos.
- Tutor M&B y UTA.
- Desarrollador del Sistema.

4.1.3 Físicos

- Laboratorio de Investigación de la Empresa M&B.
- Biblioteca de la Universidad Técnica de Ambato, Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

4.1.4 Materiales

- Bloc de notas.
- Bolígrafos
- Equipos de cómputo.
- Internet.
- Flash memory.
- Tarjeta de Adquisición de Datos

- CD´s
- Cuadernos

4.1.5 Económicos

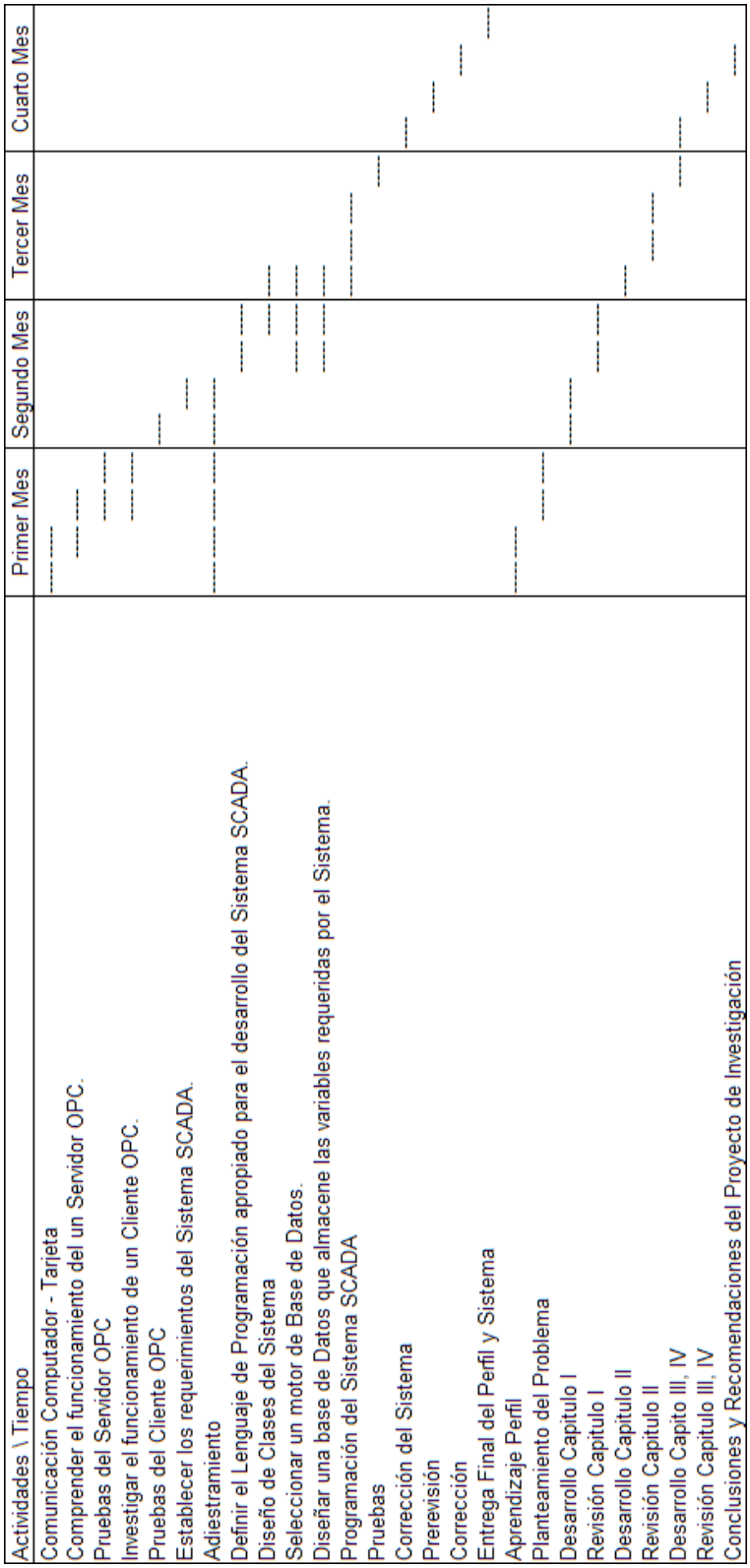
GASTOS DE OPERACIÓN	
DETALLE	**VALOR**
Anillado	3.40
Bloc de notas	1.00
Computadora	1500.00
Copias	5.00
Flash memory	21.00
Impresiones	18.00
Internet(100h)	200.00
Tarjeta de Adquisición de Datos	600.00
Transporte	6.00
Subtotal:	664.40
Imprevistos 10%	66.44
TOTAL:	**3085.24**

Tabla 4.1: Gastos de Operación

4.1.6 Bibliográficos

- Biblioteca FIS.
- Internet.

4.1.7 CRONOGRAMA DE TRABAJO



CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

- Se ha logrado integrar en el presente estudio el área de Sistemas Computacionales e Informáticos con la Electrónica, demostrando la complementariedad que las mismas tienen en la actualidad.
- Se ha demostrado el objetivo propuesto al inicio de la investigación, por lo que se concluye que el presente trabajo permite automatizar una planta de una edificación utilizando una tarjeta de adquisición de datos, para la Empresa M&B.
- Se presta al medio estudiantil una herramienta que sirve como guía para realizar automatizaciones o instalaciones domóticas, además de recalcar que son de gran ayuda los programas con licencia GPL, ya que los mismos sirven de guía en el proceso de investigación.
- Los conocimientos que la Universidad Técnica de Ambato, Facultad de Ingeniería en Sistemas son de aplicación práctica en el sector empresarial.
- Este trabajo se obtuvo como resultado de fusionar la investigación de varias áreas del conocimiento, por lo que se ha logrado desarrollar un sistema que sobrepasan los lineamientos iniciales propuestos.

5.2. RECOMENDACIONES

- Utilizar el software como una guía de aplicación práctico-educativo, en ámbitos relacionados con el objeto de estudio.
- Verificar si el medio electrónico que se vaya a poner en funcionamiento con el sistema cumpla con las características detalladas en este documento, para evitar posibles daños.

- No instalar el Sistema SCADALA 1.0 en una computadora que tuviese programas que trabajen con MySql, ya que el sistema usa exclusivamente la Base de Datos de MySQL, además se debe utilizar el sistema en máquinas Pentium 2 o superiores, con 256 en memoria RAM.
- Impulsar este tipo de convenios con las diferentes empresas, ya que permiten reafirmar y poner en práctica los conocimientos que entrega la Facultad a los futuros profesionales.

5.3. Bibliografía

- Empresa M&B (2007), MANUAL DE LA TARJETA DE ADQUISICIÓN DE DATOS MB001a.
- VÁSCONEZ DIEGO (2006), Perfil de Diseño de un Sistema Domótico con acceso remoto vía Internet para la Universidad Técnica de Ambato.
- SIDE (Sociedad de Ingenieros del Ecuador), Ley de Ejercicio Profesional de la Ingeniería y sus Reglamentos.

INTERNET

- http://es.wikipedia.org/wiki/Modbus
  Funcionamiento e Historia del protocolo Modbus.
- http://es.wikipedia.org/wiki/Puerto_serie
  Evolución del puerto de Comunicación.
- http://www.es.gnu.org/modules/content/index.php?id=8
  Leyes de licencias GNU – GPL y GNU – LGPL.
- http://dev.mysql.com/downloads/connector/net/5.0.html
  Sitio de descarga para el componente que funciona como cliente de MySql.
- http://www.mysql-hispano.org/page.php?id=41&pag=1
  Información del funcionamiento del Motor de Base de Datos MySql.
- http://www.sharpdevelop.com
  Información referente a Sharp Develop, descargas e información.
- http://www.automatedsolutions.com/products/opcmodbusrtu.asp
  Sitio de descargas para el Servidor OPC.
- http://www.modbus.org/viewdevice.php?id=342
  Información del protocolo modbus.
- http://www.mintrab.gov.ec/MinisterioDeTrabajo//Institucional/..%5C%5CDocumentos%5C306.pdf
- http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_LADDER
  Información del Lenguaje de Programación LADDER.

Las direcciones fueron consultadas durante el período de desarrollo del Proyecto.

CAPITULO VI

PROPUESTA

6.1 ANÁLISIS DEL SISTEMA

El Sistema ha sido analizado y diseñado en el Lenguaje de Modelado Unificado. El mismo que representa las diferentes etapas de desarrollo del Proyecto.

6.1.1 DIAGRAMA UML

6.1.1.1 DIAGRAMA DE COMPONENTES

Se utiliza para modelar la vista estática del Sistema SCADALA.

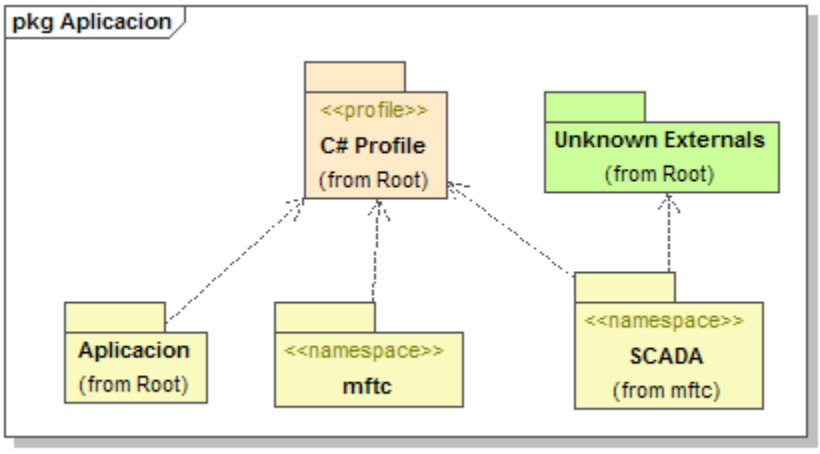


Figura 6.1: Diagrama de Componentes

6.1.1.2 DIAGRAMA DE CASOS DE USO

Se resume la funcionalidad completa de un sistema, presentando las respectivas interacciones con los agentes externos. A continuación los diferentes Casos de Uso:

Usuario del Sistema

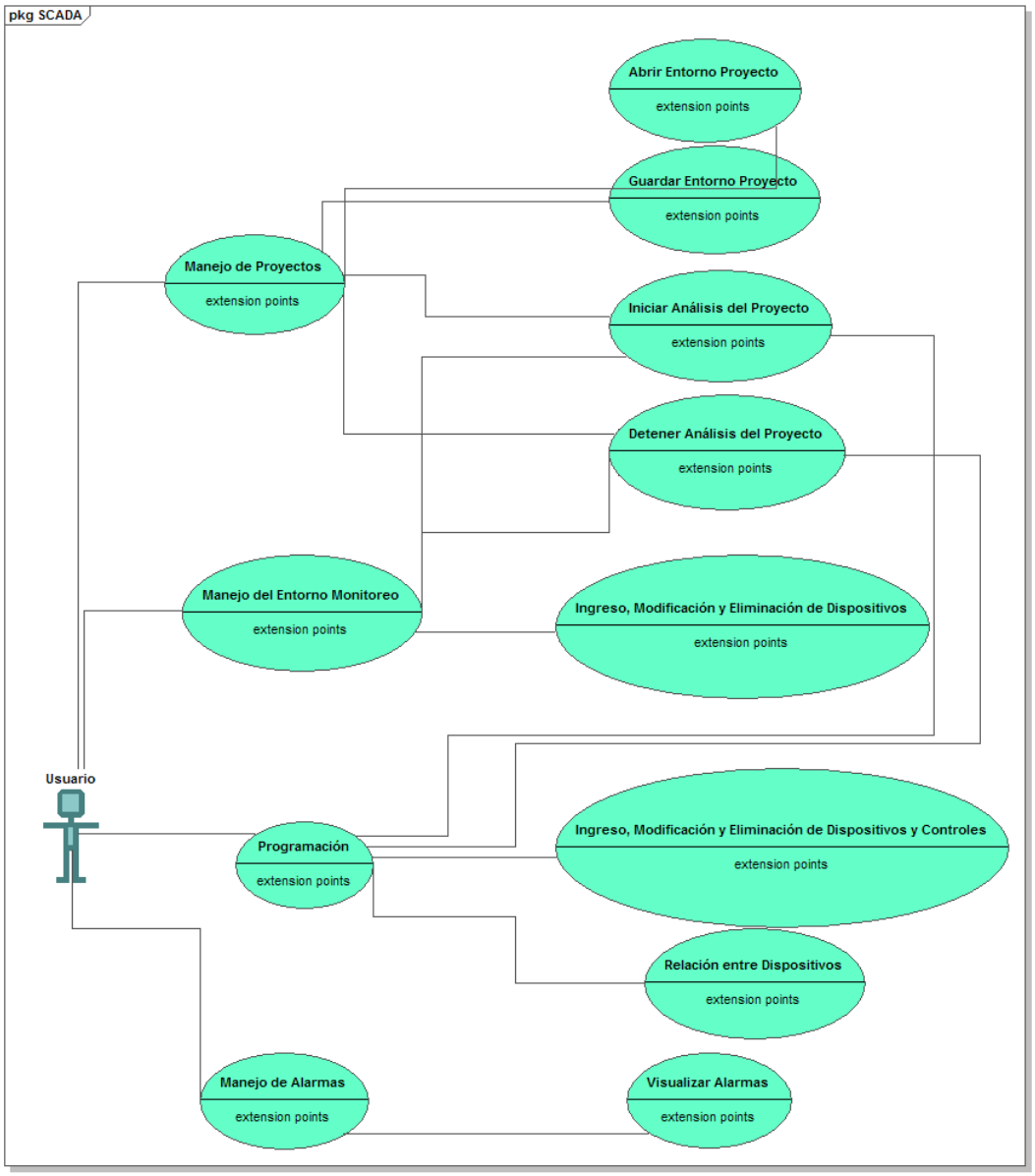


Figura 6.2: Diagrama de Caso de Usos

6.1.1.3 DIAGRAMA DE SECUENCIA

Se detalla la forma en que se intercambian mensajes o datos entre las respectivas entidades del Sistema, identificando el tiempo de vida de ciertas actividades. A continuación los diagramas de Secuencia:

Inicio del Sistema

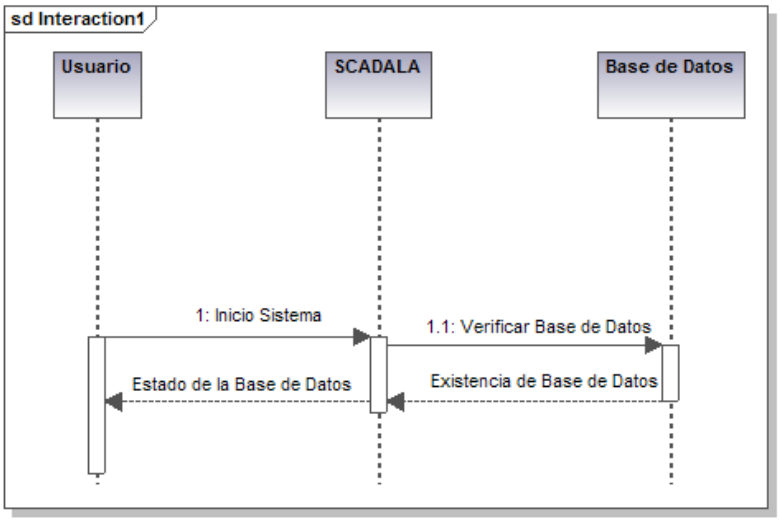


Figura 6.3: Diagrama de Secuencia – Inicio del Sistema

Abrir Proyecto

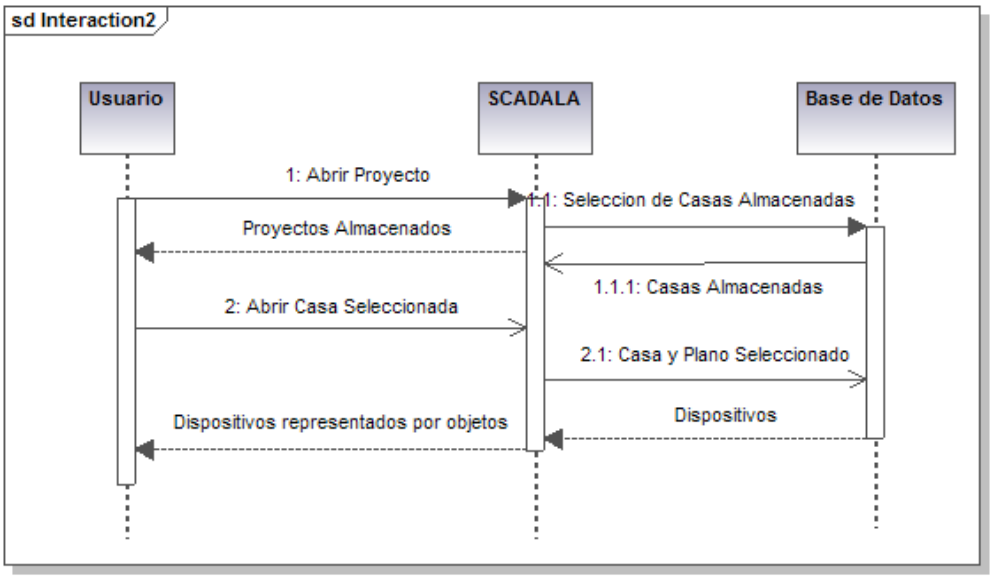


Figura 6.4: Diagrama de Secuencia – Abrir Proyecto

Guardar Proyecto

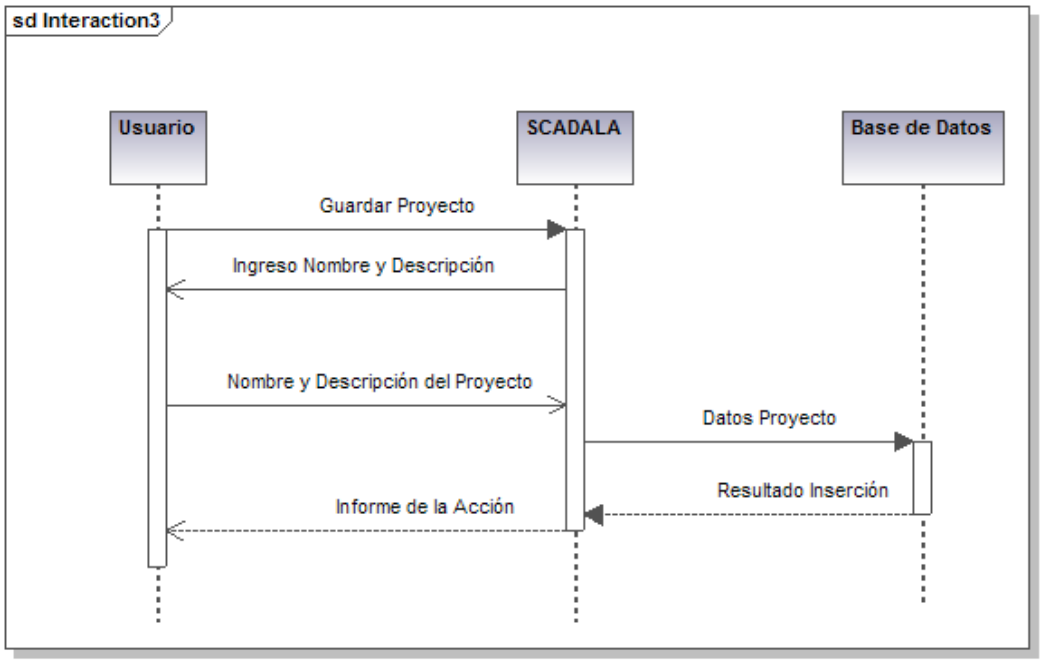


Figura 6.5: Diagrama de Secuencia – Guardar Proyecto

Iniciar y Detener Proyecto

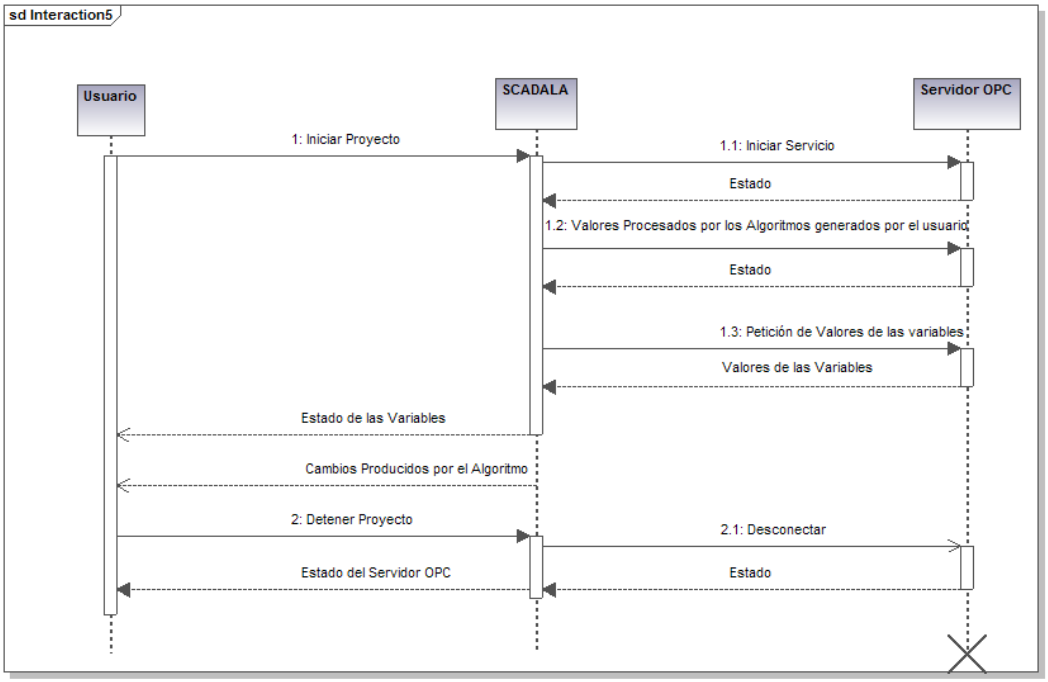


Figura 6.6: Diagrama de Secuencia – Detener Proyecto

Visualización de Alarmas

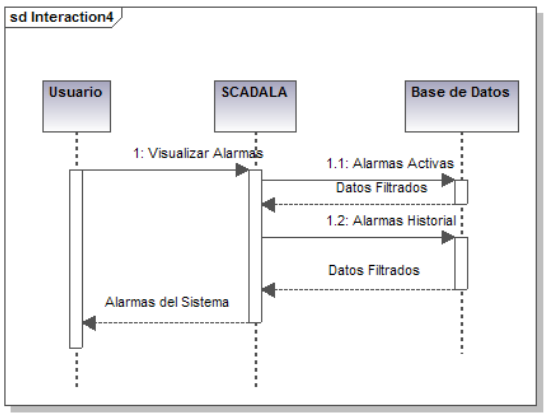


Figura 6.7: Diagrama de Secuencia – Visualización de Alarmas

6.2 DISEÑO DEL SISTEMA

6.2.1 DIAGRAMA DE CLASES

El Sistema consta de las siguientes clases, las mismas que se resumen en el Anexo 4.

6.2.2 DISEÑO DE LA INTERFAZ DE USUARIO

Se presenta al usuario una interfaz amigable, la misma que le permite monitorear y programar las variables disponibles en la tarjeta de adquisición de datos. Así como también se permite tener un control de alarmas que el sistema ha generado en la rutina de Inicialización del Proyecto.

Se generó una interfaz de usuario que es accesible con la distribución del espacio entre las áreas de trabajo, dando al usuario final la posibilidad de tener espacios predeterminados por el mismo,

Las ventanas generadas para interactuar con el usuario son las siguientes:

Ventana Inicial (Monitoreo)

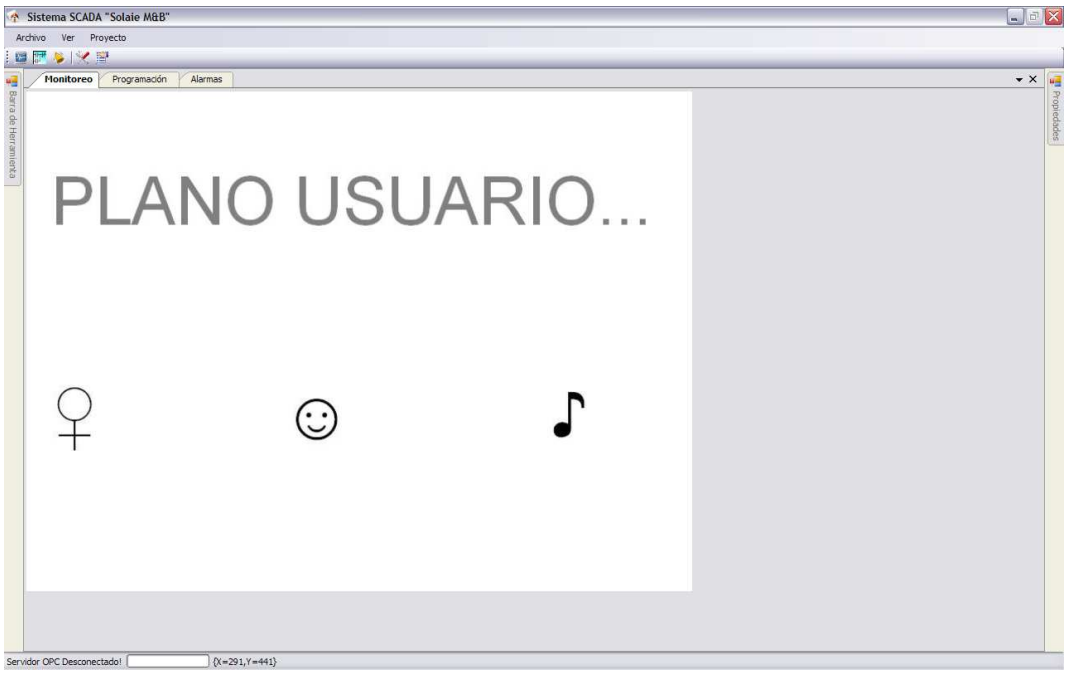


Figura 6.8: GUI - Ventana Inicial

Barras de Herramientas y Barra de Propiedades (Monitoreo)

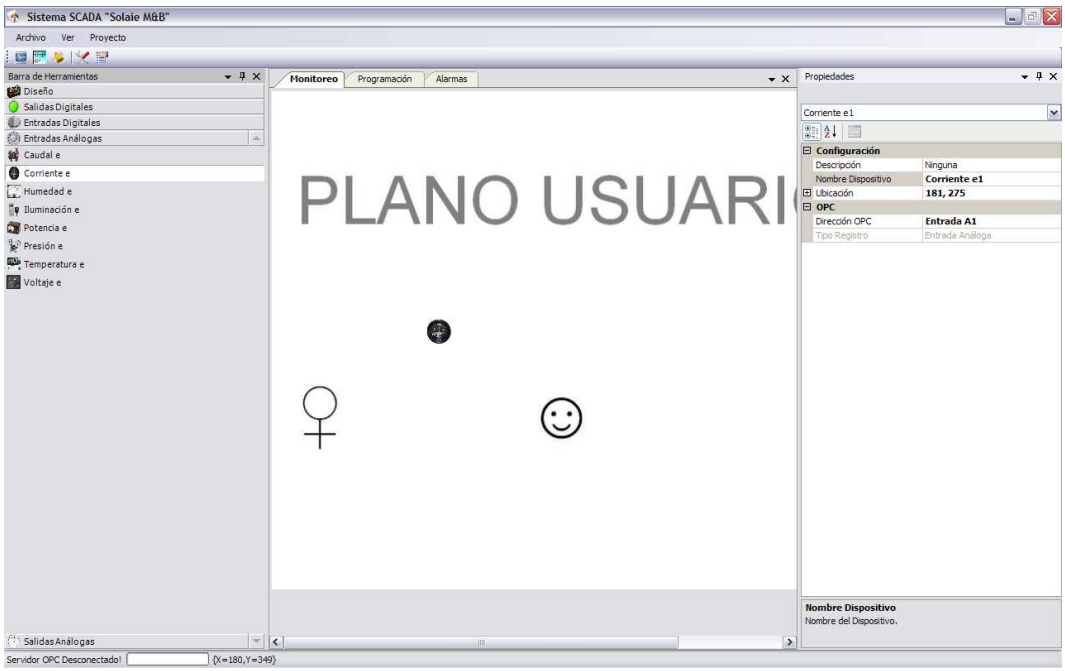


Figura 6.9: GUI - Barras de Herramientas y Barra de Propiedades (Monitoreo)

Ventana Programación

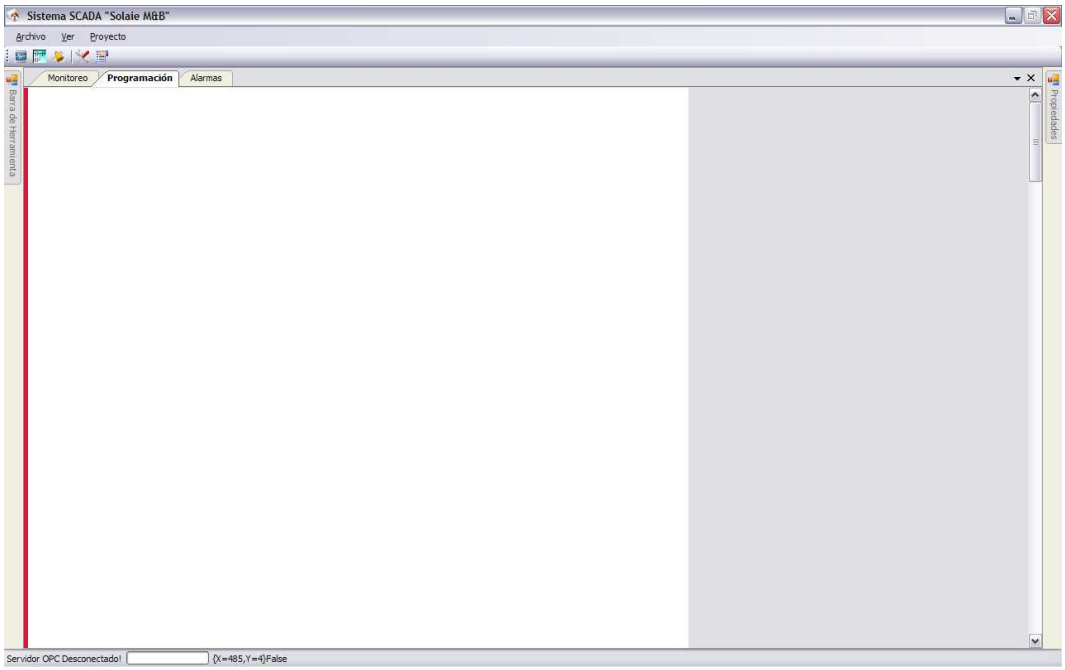


Figura 6.10: GUI - Ventana Programación

Barra de Herramientas y Barra de Propiedades (Programación)

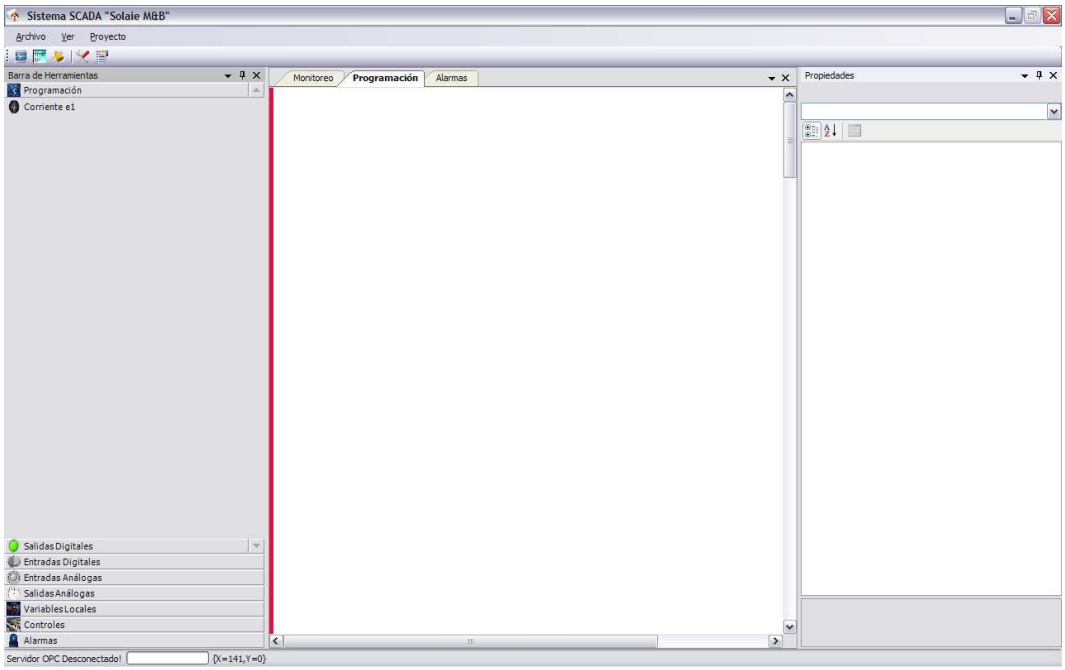


Figura 6.11: GUI - Barras de Herramientas y Barra de Propiedades (Programación)

Ventana Alarmas

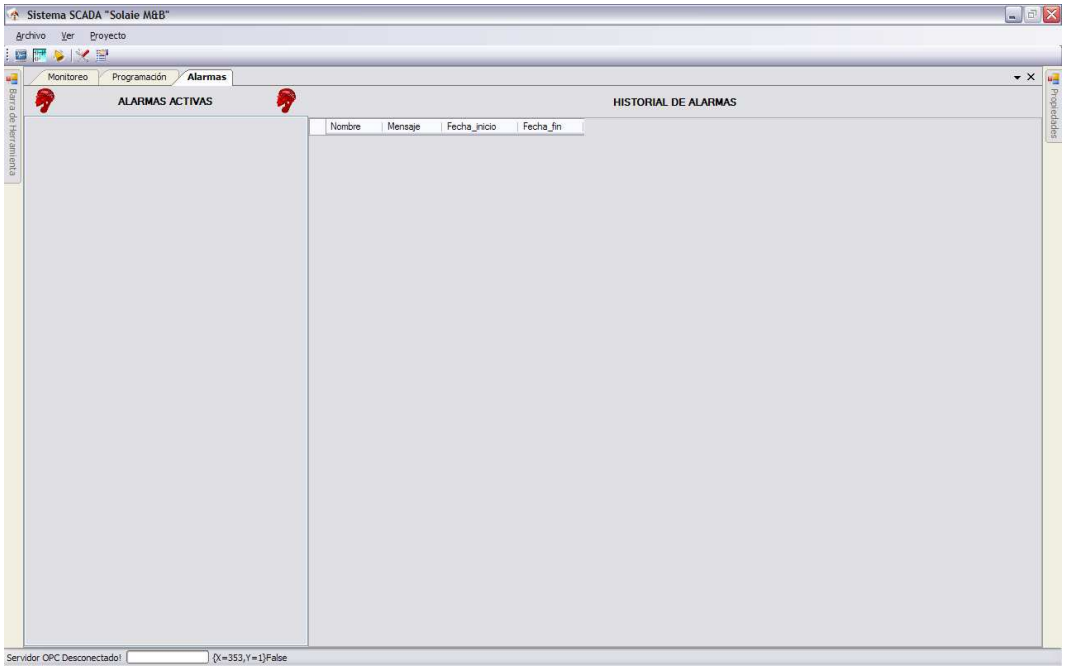


Figura 6.12: GUI – Ventana Alarmas

Detalle de los controles utilizados en el diseño de la interfaz de usuario:

- **TextBox:** Caja de Texto que permite ingresar información directamente al usuario.
- **ListBox:** Permite presentar de manera ordenada la información requerida por el usuario.
- **ComboBox:** Al igual que el ListBox, despliega la información requerida por el usuario, con la diferencia que la lista no es visible en su totalidad, sino que mantiene a un elemento visible. Distribuyendo el área de trabajo de una manera más adecuada.
- **ToolBox:** Barra de Herramientas que permite seleccionar al usuario de entre varios elementos de una forma organizada e intuitiva.
- **PropertyGrid:** Control que permite extraer y manipular las propiedades de los objetos de una manera transparente y sencilla.

- **DockingControl:** Facilita al usuario la distribución del área de trabajo, pudiendo de esta manera personalizar y acomodar el diseño del sistema a sus requerimientos.
- **Label:** Permite ingresar nombres que describen a los controles, nombres cortos que guían al usuario en el recorrido por la aplicación, así también existen labels que sirven como indicativos del estado y/o funcionamiento del Sistema, manteniendo al usuario informado de las actividades que el sistema realiza.
- **OpenFileDialog:** Ventana que manipula las exploraciones entre los diversos directorios que el sistema posee.
- **ControlErrores:** Es un control diseñado para mantener dos niveles de usuarios, los mismos que recibirán la información necesaria sobre algún fallo que el sistema reporte.
- **ToolTip:** Usualmente requeridos en los objetos creados por el usuario, que prestan una ayuda básica sobre la funcionalidad u operatividad del objeto.
- **MenuStrip:** Menú que contiene agrupadas las funciones del sistema accesibles por el usuario.

6.2.3 DISEÑO DE BASE DE DATOS

Diseño lógico

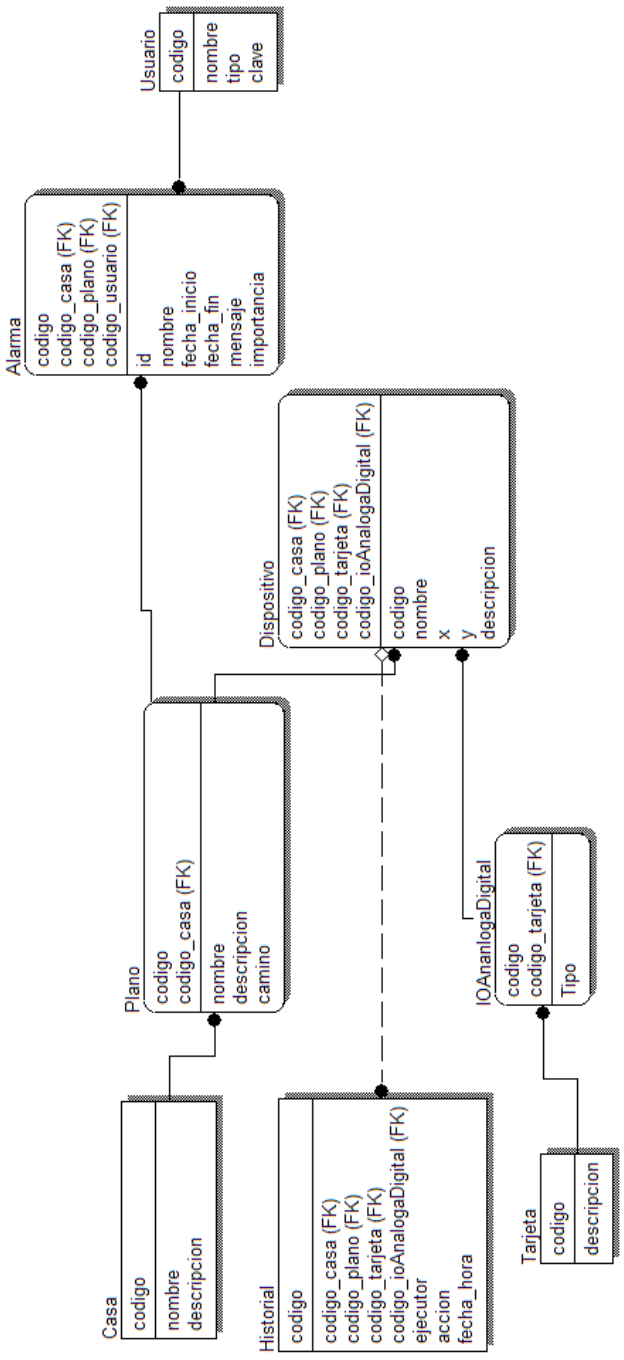


Figura 6.13: Diseño lógico de la base de datos.

Diseño físico

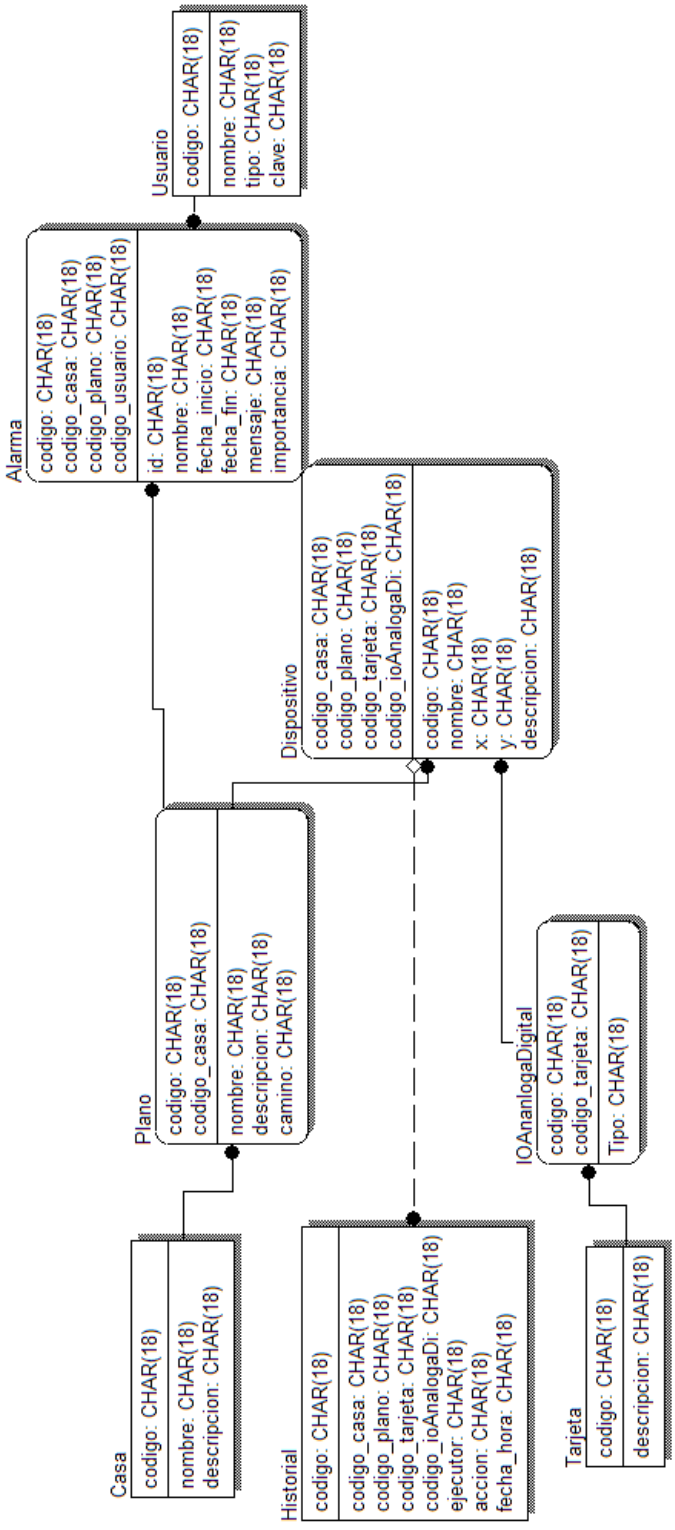


Figura 6.14: Diseño físico de la base de datos.

6.3 IMPLEMENTACIÓN

El Sistema fue implementado en un caldero, el mismo que hacía las secuencias requeridas por el mismo. Las características del equipo de cómputo son:

- Sistema Operativo Windows XP, Service Pack 2, Pentium 4 de 2.8 Ghz con 1 GB en RAM.

6.3.1 PASOS PARA IMPLEMENTAR EL SISTEMA SCADALA

Doble click en el archivo de instalación y esperar hasta que el sistema adquiera los archivos requeridos por este.

La primera interfáz que el sistema presta es una pantalla de bienvenida:

Figura 6.15: Bienvenida Instalación de SCADALA 1.0.

A continuación se presenta la licencia del Sistema, la misma que indica la naturaleza y condiciones que el Sistema tiene.

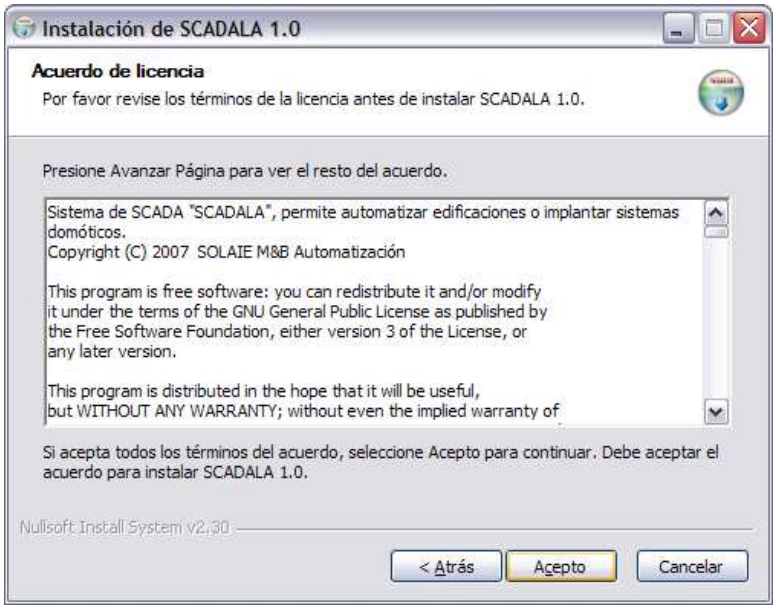


Figura 6.16: Aceptación de Términos de Licencia GPL.

Una vez leído y aceptado las condiciones del acuerdo de licencia el sistema comienza a instalar los componentes necesarios.

Uno de estos es el Servidor OPC de Automated Solution, el mismo que es visible para el usuario por las condiciones de licencia que este tiene. Las interfaces de instalación que el Servidor contiene empiezan por la Bienvenida.

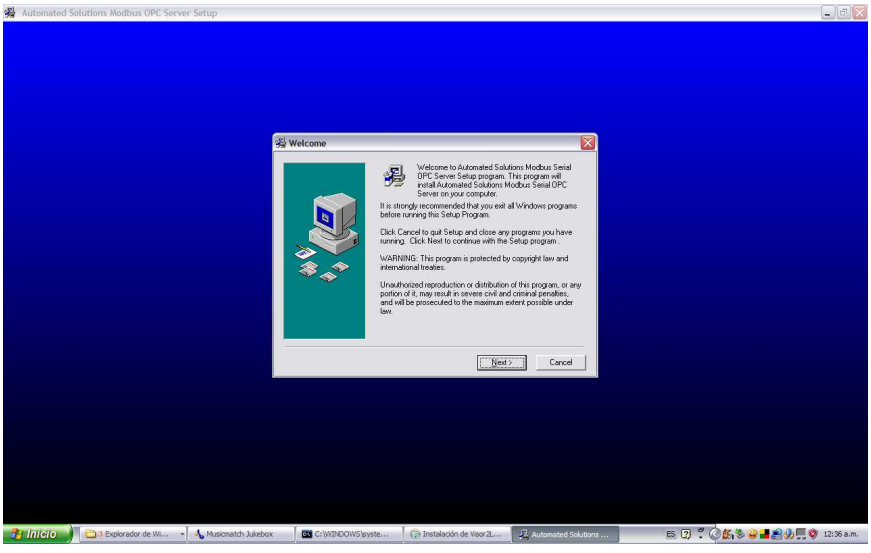


Figura 6.17: Aceptación Instalación del Servidor OPC de Automated Solution.

Posteriormente se puede elegir el directorio de instalación del servidor OPC.

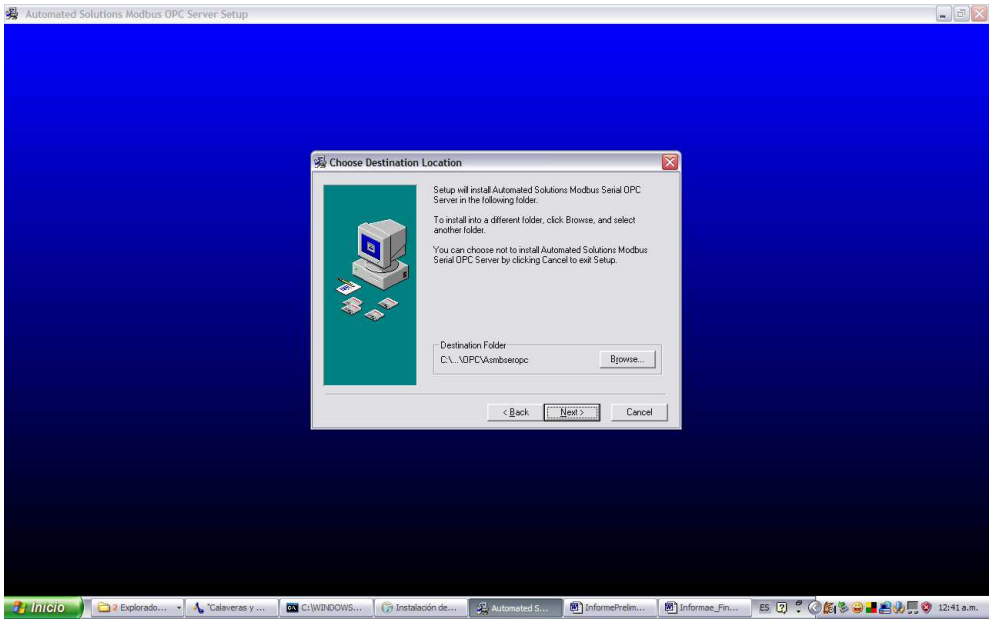


Figura 6.18: Directorio de Instalación del Servidor OPC de Automated Solution.

Y con la confirmación de la inicialización del Sistema se inicializa el proceso de instalación.

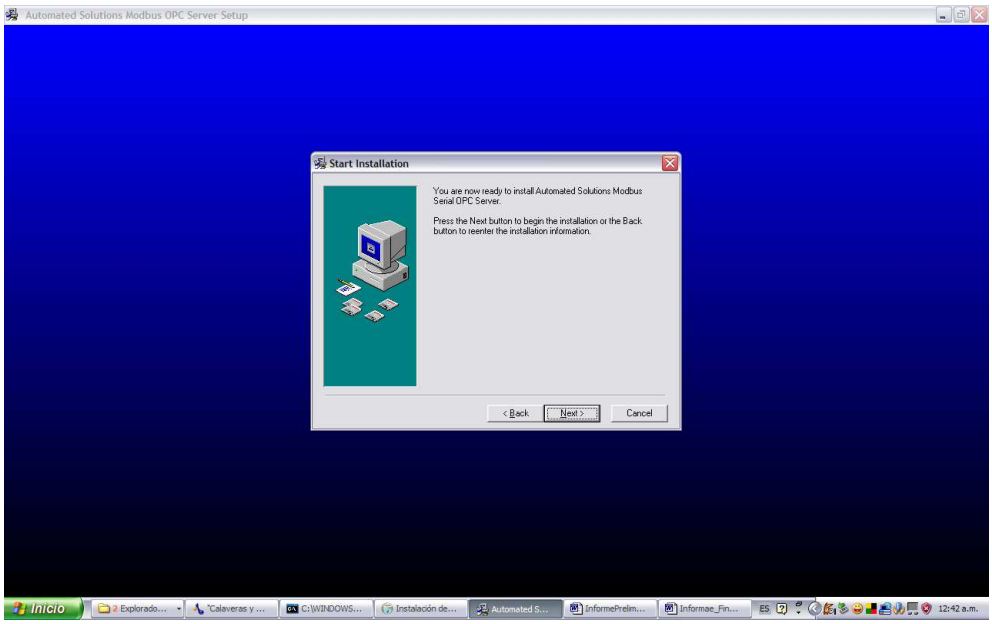


Figura 6.19: Confirmación de Instalación del Servidor OPC de Automated Solution.

Con estos pasos y si todo transcurrió sin novedades el sistema presentará la pantalla del final de la instalación del Servidor OPC.

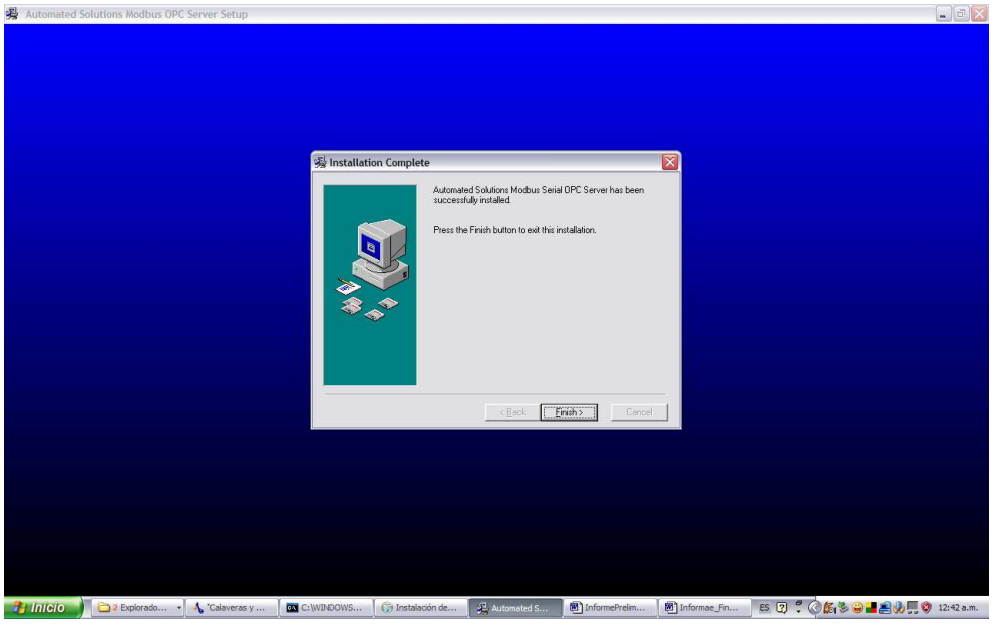


Figura 6.20: Finalización de Instalación del Servidor OPC de Automated Solution.

Concluyendo la instalación del sistema SCADALA satisfactoriamente se tendrá la siguiente pantalla de finalización.

Figura 6.21: Finalización de Instalación del Sistema SCADALA 1.0.

Nota: Al primer inicio del Sistema este genera automáticamente la Base de Datos, proceso para el cual se pedirá la aceptación del usuario.
Es necesario que se abra el archivo scada.tdb con el OPC Server de Automated Solutions. Una vez abierto el mismo, se debe guardar.

6.4 PRUEBAS

6.4.1 PRUEBAS DE CAJA BLANCA

Se realizaron varias pruebas al código del sistema, cubriendo en más de una ocasión los diversos caminos generados, ya sea en cuanto a bucles como en condiciones, optimizando en cada pasada las líneas de programación.

6.4.2 PRUEBAS DE CAJA NEGRA

Se hizo el seguimiento paso a paso y entre funciones del código, con la finalidad de verificar las funciones, revisando al mismo tiempo que los valores no excedan de sus dimensiones previamente analizadas.

6.4.3 PRUEBAS DE VALIDACIÓN

El Sistema fue probado en un entorno que permite validar la funcionalidad del mismo, logrando satisfacer las metas planteadas al inicio del proyecto en un mayoritario porcentaje.

ANEXOS

ANEXO 1

GNU GENERAL PUBLIC LICENSE

Es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

TÉRMINOS Y CONDICIONES PARA LA COPIA, DISTRIBUCIÓN Y MODIFICACIÓN

0. La licencia se aplica a cualquier programa u otro trabajo que contenga un aviso del dueño del copyright diciendo que el software debe ser distribuido bajo los términos y condiciones de esta licencia General Public License (GPL). El "programa", desde ahora, se refiere tanto a el Programa como a cualquier trabajo derivado bajo la ley del copyright: es decir, un trabajo que contenga el programa o una porción de el, tanto copia como con modificaciones y/o traducido a otra lengua (más abajo, la traducción está incluida sin limitaciones en el término "Modificación".) Cada licencia está asignada a usted.

Otras actividades distintas de la copia, distribución y modificación no están cubiertas por esta licencia y están fuera de su objetivo. Ejecutar el programa no está restringido, y el resultado del programa está cubierto por la licencia solo si su contenido contribuye parte de un trabajo derivado del Programa (independiente de la ejecución del programa). Incluso si esto es verdad, depende de lo que haga el programa

1. Usted puede copiar y distribuir copias exactas del código fuente del Programa tal y como lo recibió, usando cualquier medio, a condición de, adecuadamente y de forma bien visible, publique en cada copia una nota de copyright y un repudio de garantía; mantenga intactas todas las notas que se refieran a esta licencia y a la

exención de garantía; y proporcione a los receptores del Programa una copia de esta Licencia junto al programa.

Usted puede cobrar unos honorarios por la transferencia física de la copia, y puede a su criterio ofrecer una garantía adicional por un precio.

2. Usted puede modificar su copia o copias del Programa o cualquier porción de ella, obteniendo así un trabajo derivado del Programa, y copiar y distribuir estas modificaciones o trabajo derivado bajo los mismos términos de la Sección 1, antedichos, cumpliendo además las siguientes condiciones:

a) Debe hacer que los ficheros modificados contengan información visible de que ha modificado el fichero y la fecha de cualquier cambio.

b) Debe hacer que cualquier trabajo que distribuya o publique y que en su totalidad o en parte contenga o sea derivado del Programa o de cualquier parte de el, sea licenciado como un todo, sin carga alguna a las terceras partes, bajo los términos de esta licencia.

c) Si la modificación del programa normalmente interpreta comandos interactivos en su ejecución, debe, cuando comience su ejecución para ese uso interactivo de la forma más habitual, imprimir o mostrar un aviso de exención de garantía (o por el contrario que sí ofrece garantía) y de como los usuarios pueden redistribuir el programa bajo estas condiciones, e informando a los usuarios de como pueden obtener una copia de esta Licencia. (Excepción: Si el programa es interactivo pero normalmente no muestra este anuncio, no es necesario en un trabajo derivado mostrar este aviso).

Estos requisitos se aplican a las modificaciones como un todo. Si secciones identificables del trabajo no están derivadas del Programa, pueden ser razonablemente consideradas independientes y trabajos separados en si mismos, por tanto esta Licencia, y sus términos, no se aplican a estas secciones cuando usted las distribuye como trabajos independientes. Pero cuando usted distribuye las mismas secciones como parte de un todo que es un trabajo derivado del

Programa, la distribución del todo debe respetar los términos de esta licencia, cuyos permisos para otros licenciatarios se extienden al todo, y por lo tanto a todas y cada una de sus partes, con independencia de quién la escribió.

Por lo tanto, no es la intención de esta sección reclamar derechos o desafiar sus derechos sobre trabajos escritos totalmente por usted mismo. Por el contrario, la intención es ejercer el derecho a controlar la distribución de trabajos derivados o colectivos basados en el Programa.

Además, el mero acto de agregar otro trabajo no basado en el Programa con el Programa (o con otro trabajo derivado del Programa) en un volumen de almacenamiento o un medio de distribución no consigue que el otro trabajo se encuentre bajo los términos de esta licencia.

3. Usted puede modificar su copia y distribuir el Programa (o un trabajo derivado, cubierto bajo la Sección 2) en formato objeto o ejecutable bajo los términos de las Secciones 1 y 2 antedichas proporcionado con el al menos una de las siguientes:

a) Acompañando el Programa con el código fuente completo, legible por un ordenador, correspondiente a la arquitectura correspondiente, que debe ser distribuido bajo los términos de las secciones 1 y 2 usando un medio físico habitual en el intercambio de software; o,

b) Acompañando el Programa con una oferta por escrito, valida al menos por tres años, de facilitar a cualquier tercera parte, sin un cargo mayo del coste del medio físico, una copia completa legible por un ordenador del código fuente de la arquitectura elegida, que será distribuido bajo los términos de las Secciones 1 y 2 usando un medio físico habitual en el intercambio de software; o,

c) Acompañando el Programa con un la información que recibió, ofreciendo distribuir el código fuente correspondiente. (Esta opción se permite sólo para distribuir software gratuito -no comercial- y sólo si recibió el programa como código objeto o en formato ejecutable con esta misma oferta, de acuerdo con el apartado b anterior).

El código fuente de un trabajo se entiende la forma óptima para realizar modificaciones en el. Para un programa ejecutable, el código fuente completo se refiere a todo el código fuente para todos los módulos que contiene, más cualquier fichero asociado de definición de interfaces, más el script utilizado para la compilación y la instalación del ejecutable. Sin embargo, como una excepción especial, el código fuente distribuido no necesita incluir nada que no sea normalmente distribuido (ni en código fuente o en forma binaria) con los componentes mas importantes (compiladores, kernels y demás) del sistema operativo donde corre el ejecutable, salvo que el componente acompañe al ejecutable.

Si la distribución de ejecutables o compilado se realiza ofreciendo acceso a un sitio para la copia, entonces ofrecer un acceso equivalente de copia desde un sitio para el código fuente cuenta como una distribución de código fuente, incluso aunque terceras partes no estén obligadas a copiar el código fuente con el código compilado.

4. Usted no debe copiar, modificar, sublicenciar o distribuir el Programa excepto como está permitido expresamente en esta Licencia. Cualquier intento de copiar, modificar, sublicenciar o distribuir el Programa que no esté incluido en la Licencia está prohibido, y anulará automáticamente los derechos otorgados por esta licencia. Sin embargo, las partes que hayan recibido copias, o derechos, por usted bajo esta Licencia no verán sus licencias terminadas mientras estas partes continúen cumpliendo los términos de esta licencia.

5. Usted no está obligado a aceptar esta licencia, ya que usted no la ha firmado. Sin embargo, nada más le garantiza los derechos de modificación o distribución del programa o de sus trabajos derivados. Estas acciones están prohibidas por la ley si usted no acepta esta Licencia. En cualquier caso, por modificar o distribuir el programa (o cualquier trabajo derivado del programa), usted indica su aceptación implícita de esta Licencia, ya que la necesita para hacerlo, y todos sus términos y condiciones de copia, distribución o modificación del Programa o trabajos derivados.

6. Cada vez que usted redistribuya el Programa (o cualquier trabajo derivado del Programa), el receptor automáticamente recibe la licencia por parte del licenciatario original para copiar, distribuir o modificar el Programa sujeto a estos términos y condiciones. Usted no puede imponer ninguna otra restricción a los receptores limitando los derechos garantizados en esta Licencia. Usted no es responsable de asegurar el cumplimiento de terceras partes sobre la Licencia.

7. Si, como consecuencia de una decisión judicial o una acusación de infracción de patentes o por cualquier otra razón (no limitada a una causa de patentes), le son impuestas condiciones (ya sea por una orden judicial, por un acuerdo o cualquier otra forma) que contradiga los términos y condiciones d esta Licencia, no le exime de cumplir los términos y condiciones de dicha Licencia. Si usted no puede distribuir el Programa cumpliendo simultáneamente tanto los términos y condiciones de la Licencia como cualquier otra obligación que le haya sido impuesta, usted consecuentemente no puede distribuir el Programa bajo ninguna forma. Por ejemplo, si una patente no permite la redistribución gratuita del Programa por parte de todos aquellos que reciben copias directas o indirectamente a través de usted, entonces la única forma de satisfacer tanto esa condición como los términos y condiciones de esta Licencia sería evitar completamente la distribución del Programa.

Si alguna porción de esta sección es inválida o imposible de cumplir bajo una circunstancia particular, el resto de la sección tiene que intentar aplicarse y la seccionó completa debe aplicarse en cualquier otra circunstancia.

El propósito de esta sección no es inducir a infringir ninguna patente o otros derechos de propiedad o impugnar la validez de estos derechos; esta sección tiene le único propósito de proteger la integridad del sistema de distribución del Software Libre, que está implementado bajo practicas de licencias públicas. Mucha gente ha realizado generosas contribuciones a la gran variedad de software distribuido bajo este sistema con confianza en una aplicación consistente del sistema; será el autor/donante quien decida si quiere distribuir software mediante cualquier otro sistema y una licencia no puede imponer esa elección.

Este apartado pretende dejar completamente claro lo que se cree que es una consecuencia del resto de esta Licencia.

8. Si la distribución y/o el uso del Programa está restringido en ciertos países, ya sea por patentes o por interfaces bajo copyright, el propietario del Copyright original que pone el Programa bajo esta Licencia debe añadir unos límites geográficos específicos excluyendo esos países, por lo que la distribución solo estará permitida en los países no excluidos. En este caso, la Licencia incorpora la limitación de escribir en el cuerpo de esta Licencia.

9. La Free Software Foundation puede publicar versiones revisadas y/o nuevas de la Licencia GPL de cuando en cuando. Dichas versiones serán similares en espíritu a la presente versión, pero pueden ser diferentes en detalles para considerar nuevos problemas o situaciones.

Cada versión tiene un número de versión propio. Si el Programa especifica un número de versión de esta Licencia que hace referencia a esta o "cualquier versión posterior", usted tiene la opción de seguir los términos y condiciones de o bien la versión referenciada o bien cualquiera de las versiones posteriores publicadas por la Free Software Foundation. Si el Programa no especifica ningún número de versión, usted debe elegir cualquiera de las versiones publicadas por la Free Software Foundation.

10. Si usted quiere incorporar parte del Programa en otros programas libres cuyas condiciones de distribución son diferentes, escriba al autor para pedir permiso. Para el software con el copyright bajo la Free Software Foundation, escriba a la Free Software Foundation; algunas veces hacemos excepciones en esto. Nuestra decisión estará guiada por los objetivos de preservar la libertad del software y de los trabajos derivados y el de promocionar el intercambio y reutilización del software en general.

SIN GARANTÍA

11. DADO QUE ESTE PROGRAMA ESTÁ LICENCIADO LIBRE DE COSTE, NO EXISTE GARANTÍA PARA EL PROGRAMA, EN TODA LA EXTENSIÓN PERMITIDA POR LA LEY APLICABLE. EXCEPTO CUANDO SE INDIQUE POR ESCRITO, LOS DUEÑOS DEL COPYRIGHT Y/O OTRAS PARTES PROVEEDORAS FACILITAN EL PROGRAMA "TAL CUAL" SI GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO, PERO NO LIMITANDO, LAS GARANTÍAS APLICABLES MERCANTILES O DE APLICABILIDAD PARA UN PROPÓSITO PARTICULAR. USTED ASUME CUALQUIER RIESGO SOBRE LA CALIDAD O LAS PRESTACIONES DEL PROGRAMA. SI EL PROGRAMA TIENE UN ERROR, USTED ASUME EL COSTE DE TODOS LOS SERVICIOS NECESARIOS PARA REPARARLO O CORREGIRLO.

12. EN NINGÚN CASO, EXCEPTO CUANDO SEA REQUERIDO POR LA LEGISLACIÓN APLICABLE O HAYA SIDO ACORDADO POR ESCRITO, NINGÚN PROPIETARIO DEL COPYRIGTH NI NINGUNA OTRA PARTE QUE MODIFIQUE Y/O REDISTRIBUYA EL PROGRAMA SEGÚN SE PERMITE EN ESTA LICENCIA SERÁ RESPONSABLE POR DAÑOS, INCLUYENDO CUALQUIER DAÑO GENERAL, ESPECIAL ACCIDENTAL O RESULTANTE PRODUCIDO POR EL USO O LA IMPOSIBILIDAD DE USO DEL PROGRAMA (CON INCLUSIÓN PERO SIN LIMITACIÓN DE LA PÉRDIDA DE DATOS O A LA GENERACIÓN INCORRECTA DE DATOS O A LAS PÉRDIDAS OCASIONADAS O POR USTED O POR TERCERAS PARTES, O A UN FALLO DEL PROGRAMA AL FUNCIONAR EN COMBINACIÓN CON CUALQUIER OTRO PROGRAMA), INCLUSO SI DICHO PROPIETARIO U OTRA PARTE HA SIDO ADVERTIDO SOBRE LA POSIBILIDAD DE DICHOS DAÑOS.

ANEXO 2

MANUAL DE USUARIO

Introducción

El Sistema SCADALA, permite realizar el control y programación de Variables del mundo real, por medio de una Tarjeta de Adquisición de Datos. A continuación se detallan las principales funcionalidades del Sistema.

Monitoreo de las variables de la Tarjeta de Adquisición de Datos.

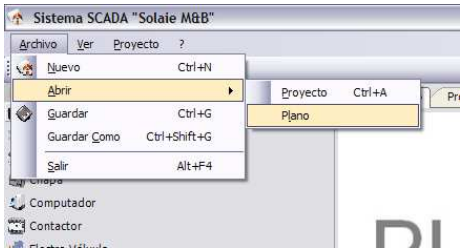
En esta interfaz se puede añadir controles predeterminados para su respectiva monitorización.

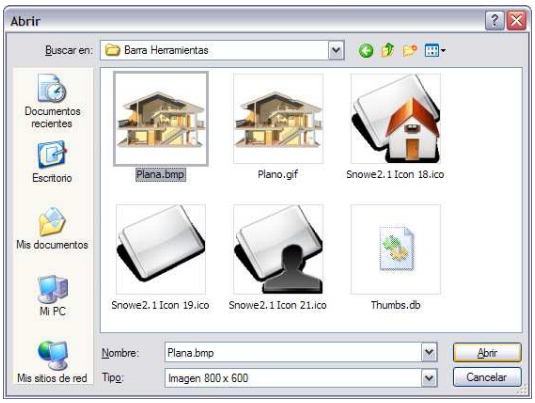
La parte de monitoreo esta dividido en:

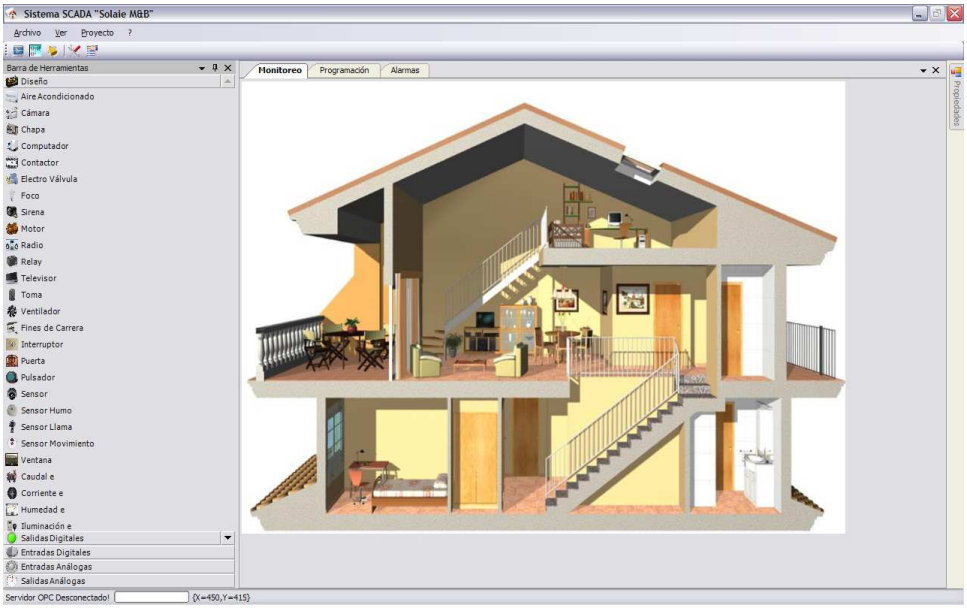
- Área de Visualización de Elementos.
- Barra de Herramientas.
- Barra de Propiedades

Área de Visualización de Elementos.

Se debe escoger una imagen del plano de la edificación para situarla en el Área de Visualización, la misma que servirá de guía en la distribución de los elementos.







Barra de Herramientas

Control que contiene los diversos dispositivos que el Sistema puede Monitorear.

Para el monitoreo se usa funciones que permiten adquirir y enviar los estados de las variables de la tarjeta de adquisición de datos. Las mismas que son:

Entradas Digitales

Lectura de Variables.

Entradas Análogas

Lectura de Variables y representación gráfica de los datos en tiempo real.

Salida Digital

Lectura/Escritura de variables.

Salida Análoga

Lectura/Escritura de variables y representación gráfica de los datos en tiempo real.

Para la comunicación y la interacción entre el Sistema SCADA y la Tarjeta de Adquisición de Datos se usa el servidor Modbus de Automated Solutions, el mismo que soporta las comunicaciones por medio del protocolo Modbus.

El Sistema SCADA usa la librería OPCDataLib, la misma que se encarga de mantener las comunicaciones entre la aplicación y el Servidor OPC.

Los controles disponibles están agrupados en:

Diseño

- Entradas Digitales.
- Salidas Digitales.
- Entradas Análogas.
- Salidas Análogas.

Entradas Digitales

- Pulsador de Entrada.

- Sensores de Movimiento.
- Señales de Alarma.
- Pulsadores de Iluminación.
- Control Equipos.
- Sensores de Puerta y Ventana.
- Switchs.
- Pulsadores.
- Sensores movimiento
- Sensores humo
- Sensores de llama
- Sensores
- Fines de carrera.

Salidas Digitales

- Sirena.
- Televisor.
- Equipos.
- Chapas Eléctricas.
- Chapa.
- Luces.
- Motor.
- Televisión.
- Radio.
- Toma corrientes.
- Ventilador.
- Aire acondicionado.
- Contactor.
- Relay.
- Electro Válvulas.
- Computador.
- Cámara.

Entradas/Salidas Análogas.

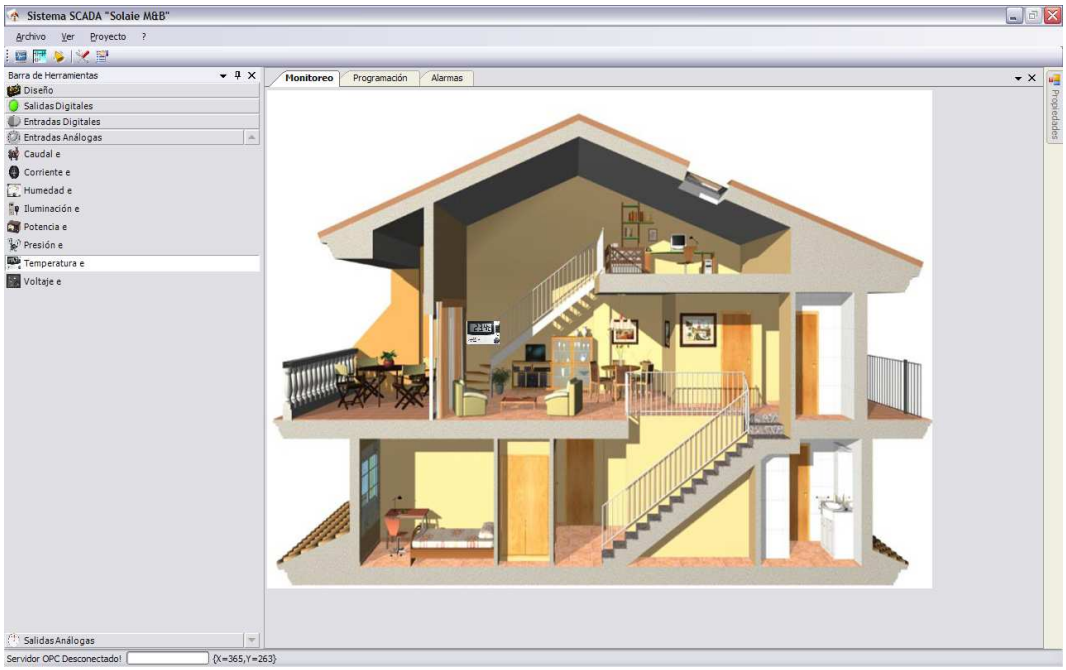
- Sensor Temperatura.
- Sensor Humedad.
- Análogas I/O.
- Presión.
- Voltaje.
- Corriente.
- Potencia.
- Caudal.
- Iluminación.

Estos controles pueden ser adheridos al Área de Visualización, arrastrándolos desde su lugar de origen, hasta el espacio deseado.

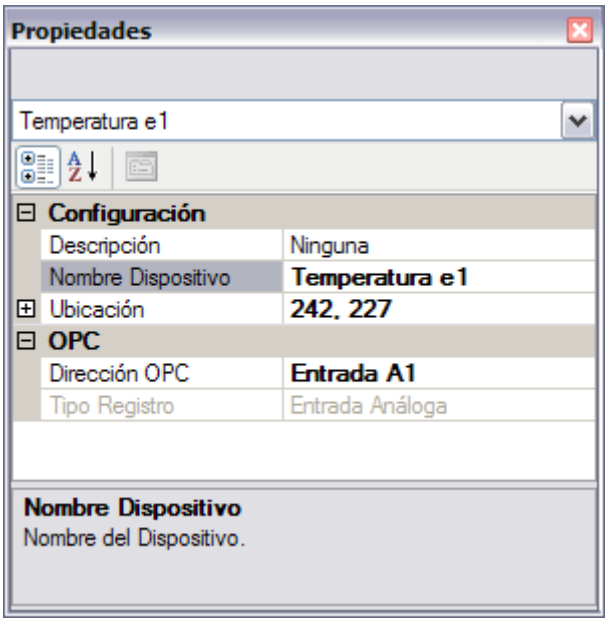
Por ejemplo, si se desea monitorear las variaciones de la temperatura que se presenten en un determinado sensor se debería seguir los siguientes pasos:

1. Identificar el tipo de dispositivo. En este caso de la temperatura es una entrada análoga. Entonces dar un clic en Entradas Análogas.

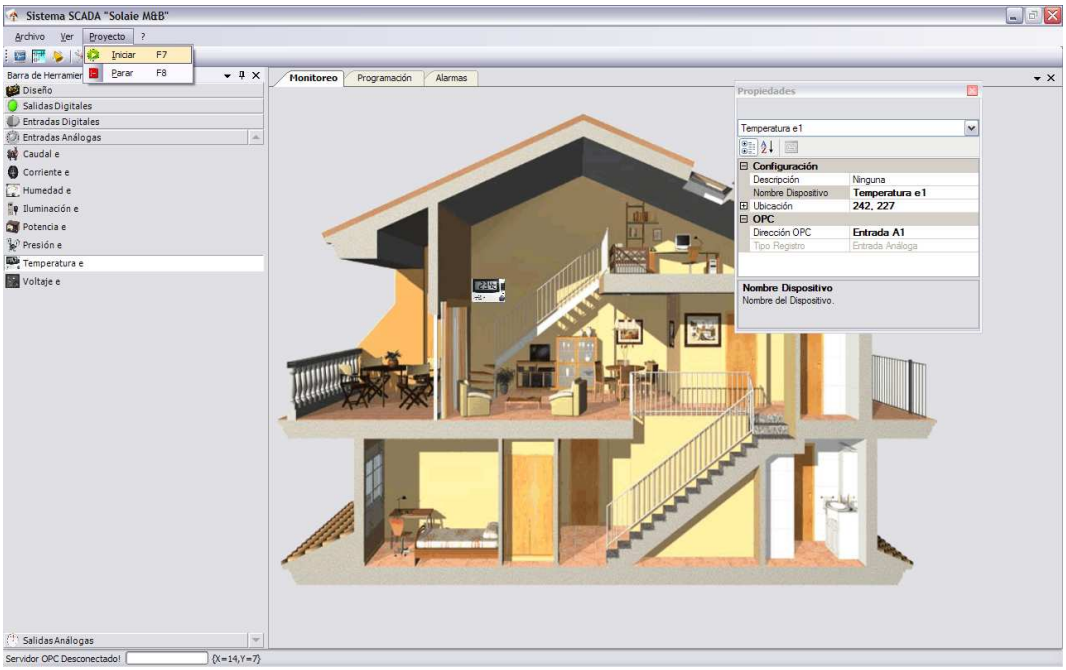
2. Arrastrar el control Temperatura e al Área de Visualización.



3. Posteriormente se debe definir las respectivas propiedades de cada elemento, para lo cual se debe seleccionar el elemento o a su vez desplazarse entre los dispositivos que se despliegan en el comboBox de las Propiedades.



4. Luego de Haber definido las propiedades se debe ir a Proyecto-Iniciar.



5. Y con esto el Sistema empieza a adquirir los datos de la Tarjeta de Adquisición de Datos, así como también se tiene un control de las variables de salida.

Programación del Sistema SCADA

En esta sección se puede verificar la presencia de un entorno de programación tipo LADDER, el mismo que facilita la creación de secuencias de programación.

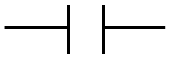
Tipos de controles del Sistema SCADA (Programación).

- Entrada Digital
- Entrada Análoga
- Salida Digital
- Salida Análoga.
- Marca.
- Marca Salida Análoga.
- Marca Salida Digital.

- Marca Marca.
- Marca Temporizador TON.
- Alarma.
- Temporizador TON.
- Temporizador TOFF.
- Contador.

Entrada Digital

Lectura de la variable.



Entrada Análoga

Lectura y comparación del valor de la entrada con un valor constante definido por el usuario.

Salida Digital

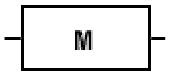
Escritura de valores lógicos que dependen del resultado de un análisis lógico.

Salida Análoga

Escritura de un valor constante definido por el usuario en la variable.

Marca

Referencia un espacio de memoria, el mismo que adquiere un valor booleano dependiendo de un análisis lógico previo.



Marca Salida Análoga

Apunta al estado de la salida análoga padre.

Marca Salida Digital.

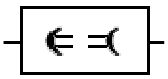
Permite leer el valor de una salida digital previamente definida.

Marca Marca.

Permite leer el valor de una marca previamente definida.

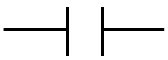
Marca Temporizador TON.

Obtiene el tiempo del temporizador, el mismo que se compara con un valor constante para obtener un resultado lógico.



Marca Contador

Se pone en alto cuando el Contador haya llegado al valor establecido.



Alarma

Permite almacenar en la base de datos el resultado de una operación lógica definida por el usuario.

Temporizador TON

Da una pausa de encendido a la salida del mismo, siempre y cuando el resultado lógico previo sea el requerido por la lógica de programación del usuario.

Temporizador TOFF

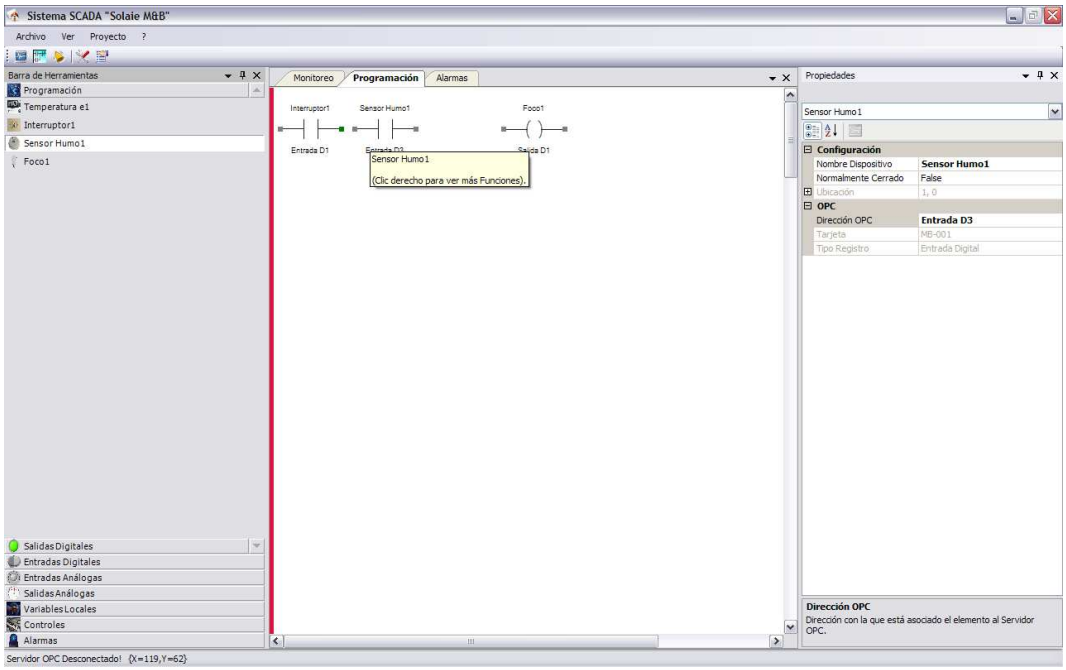
Da una pausa de apagado a la salida del control, pausa que se inicia con el resultado lógico previsto por el usuario que ingresará por la entrada del control.

Contador

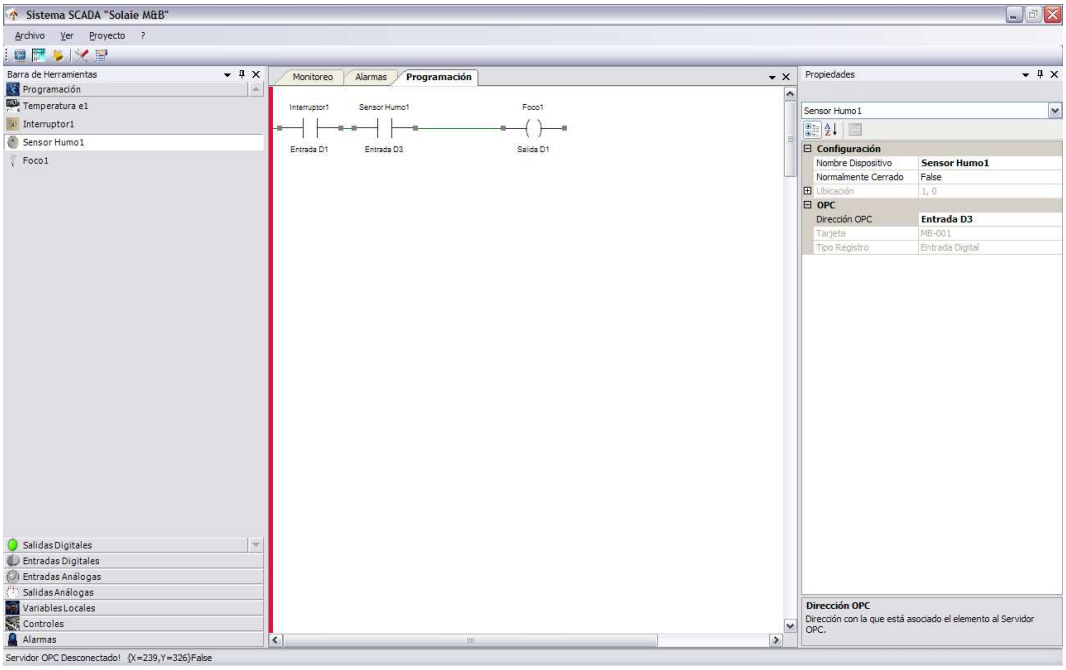
Control que se incrementa en uno por flancos, ya sea en 0 o en 1.

Para realizar las respectivas comparaciones lógicas se deben seguir los siguientes pasos:

1. Haber asignado al menos un elemento en el Área de Visualización o Monitoreo.
2. Arrastrar los elementos que se deseen considerar hacia el Área de Programación.



3. Definir sus respectivas propiedades.
4. Unir los elementos desde la barra de energía hasta interconectarlos entre ellos, de izquierda a derecha.

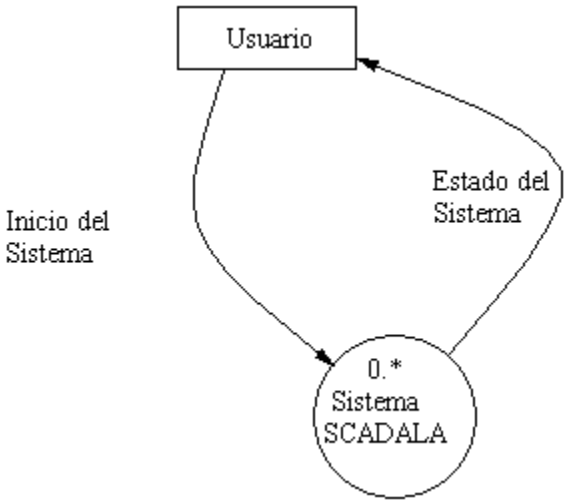


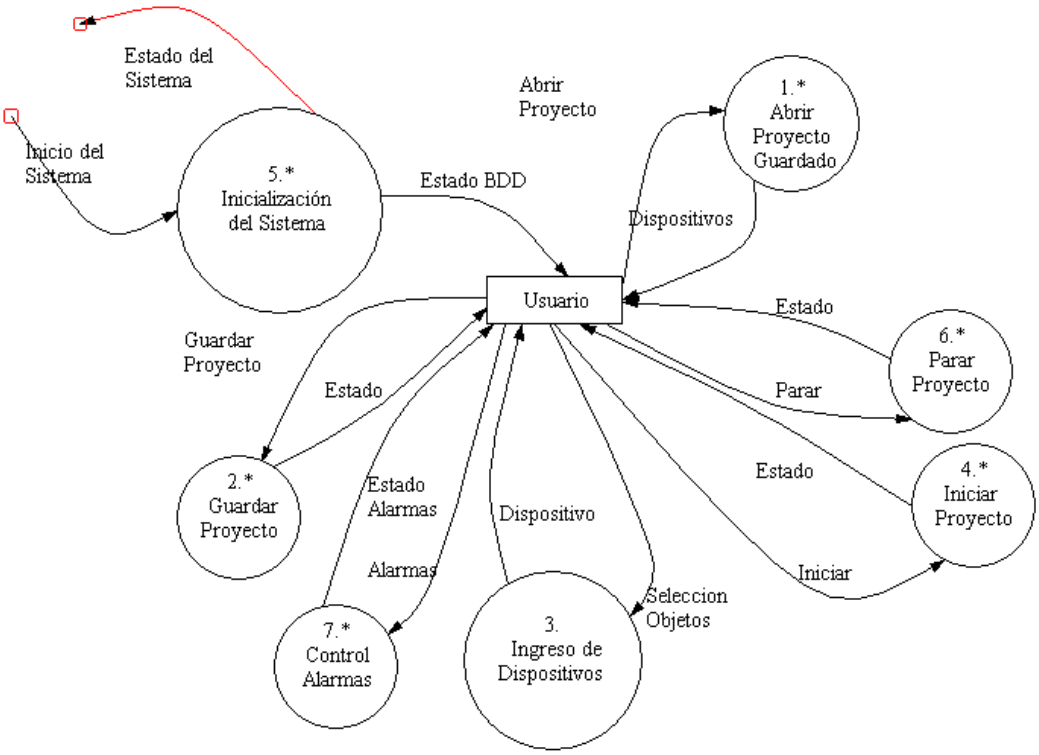
ANEXO 3

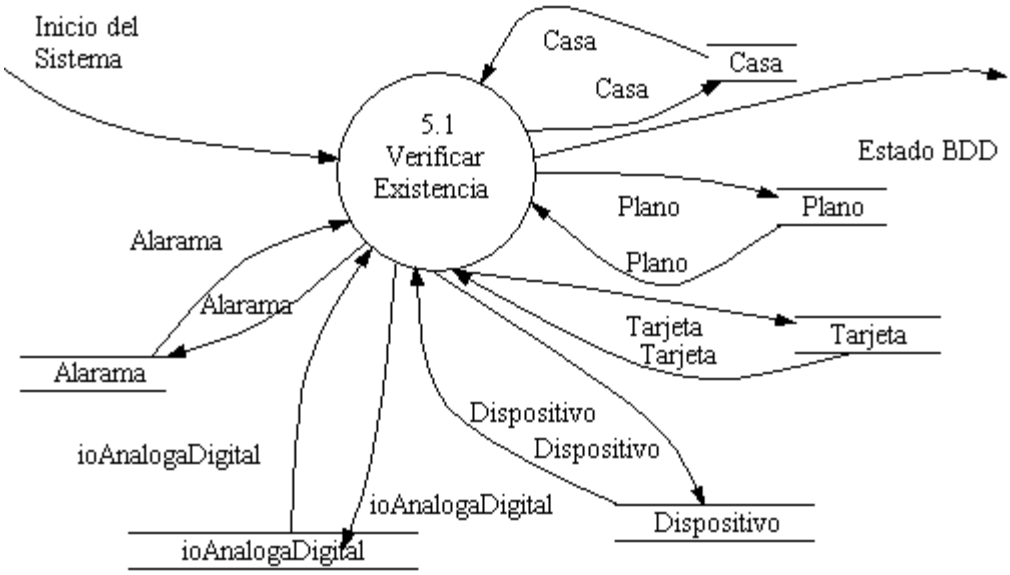
MANUAL TÉCNICO

DIAGRAMA DE FLUJO DE DATOS

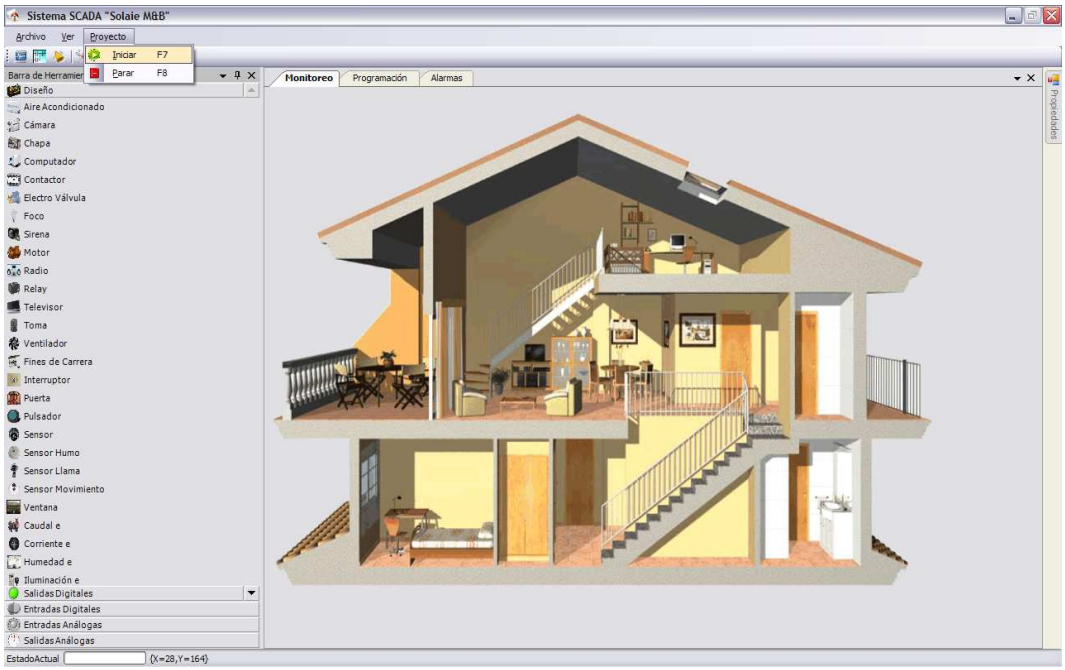
Pantalla Inicial

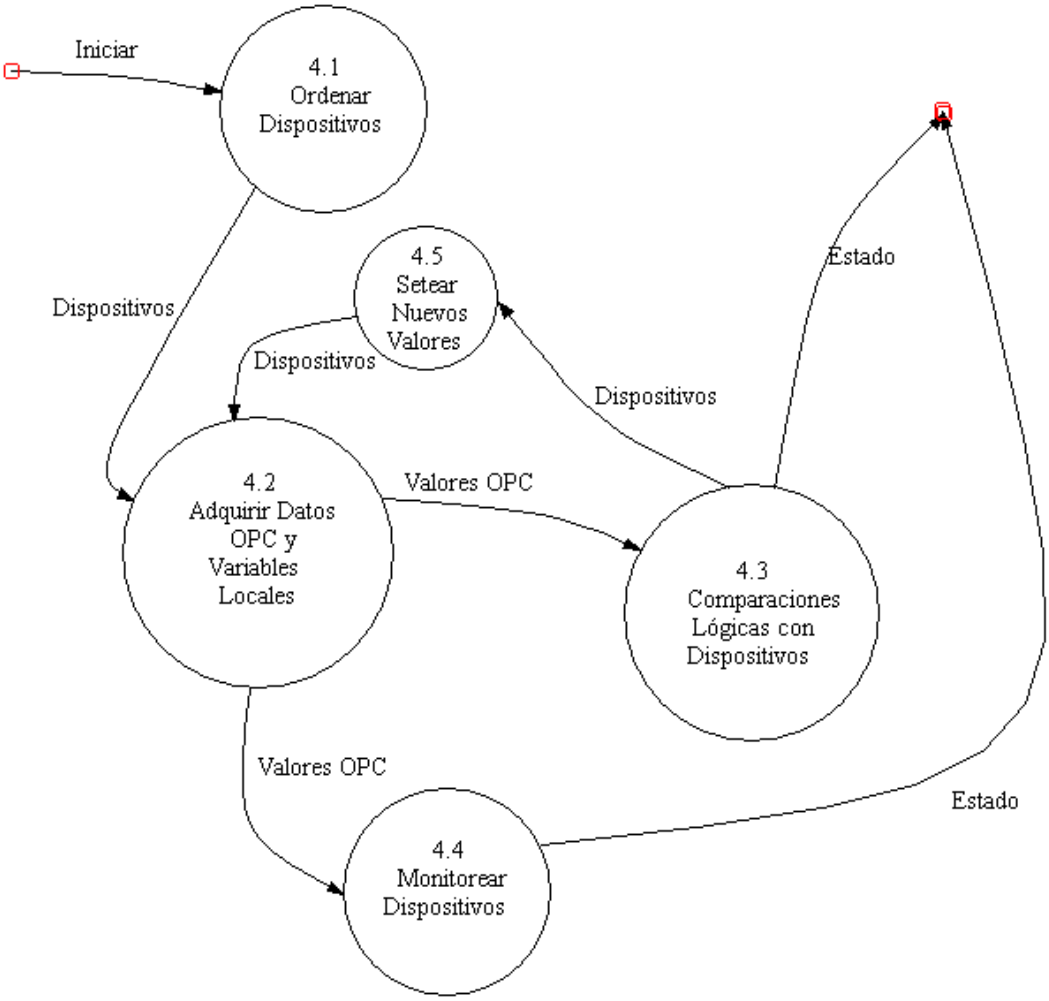



Estado del Sistema
Inicio del Sistema
5.* Inicialización del Sistema
Estado BDD
Abrir Proyecto
1.* Abrir Proyecto Guardado
Dispositivos
Usuario
Estado
6.* Parar Proyecto
Parar
Guardar Proyecto
Estado
2.* Guardar Proyecto
Estado Alarmas
Alarmas
Dispositivo
Estado
4.* Iniciar Proyecto
Iniciar
7.* Control Alarmas
3. Ingreso de Dispositivos
Seleccion Objetos

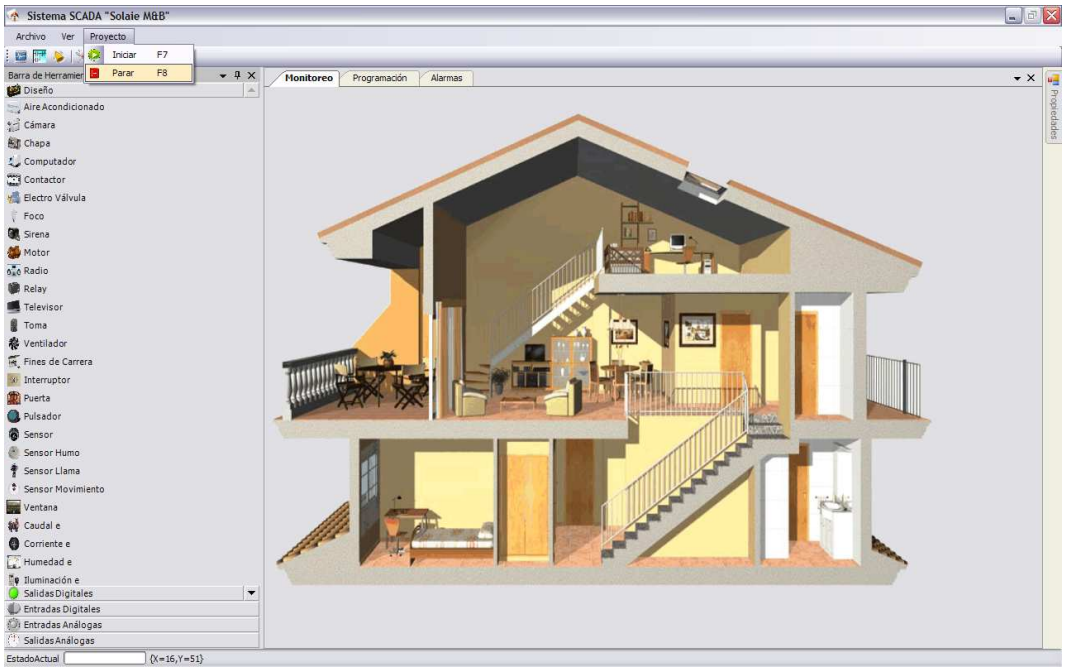

Inicio del Sistema
Casa
Casa
Casa
5.1 Verificar Existencia
Estado BDD
Plano
Plano
Plano
Alarama
Alarama
Alarama
Tarjeta
Tarjeta
Tarjeta
Dispositivo
Dispositivo
ioAnalogaDigital
ioAnalogaDigital
Dispositivo
ioAnalogaDigital

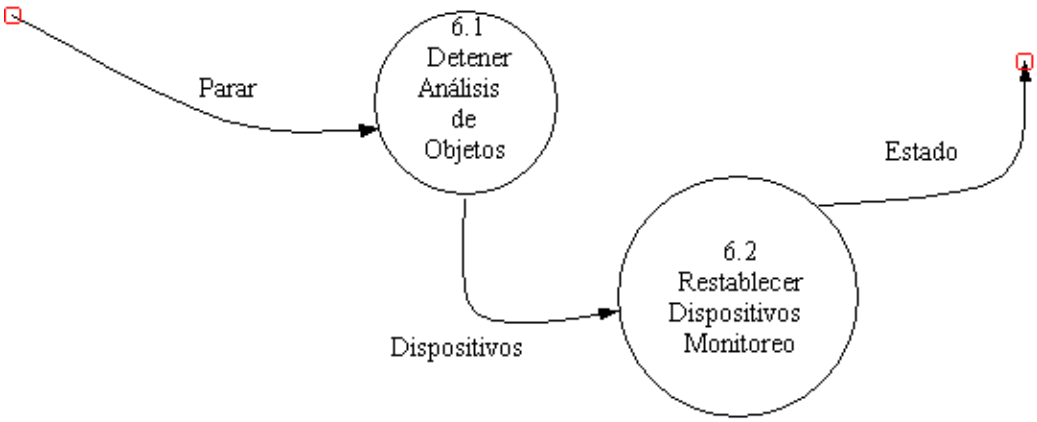
Iniciar Proyecto



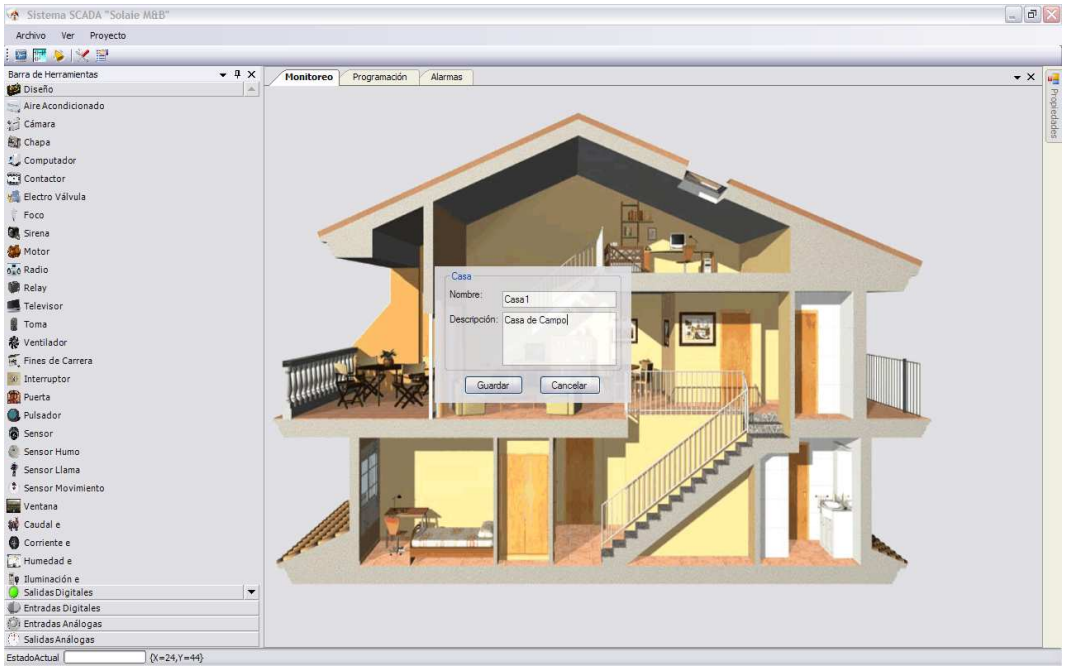


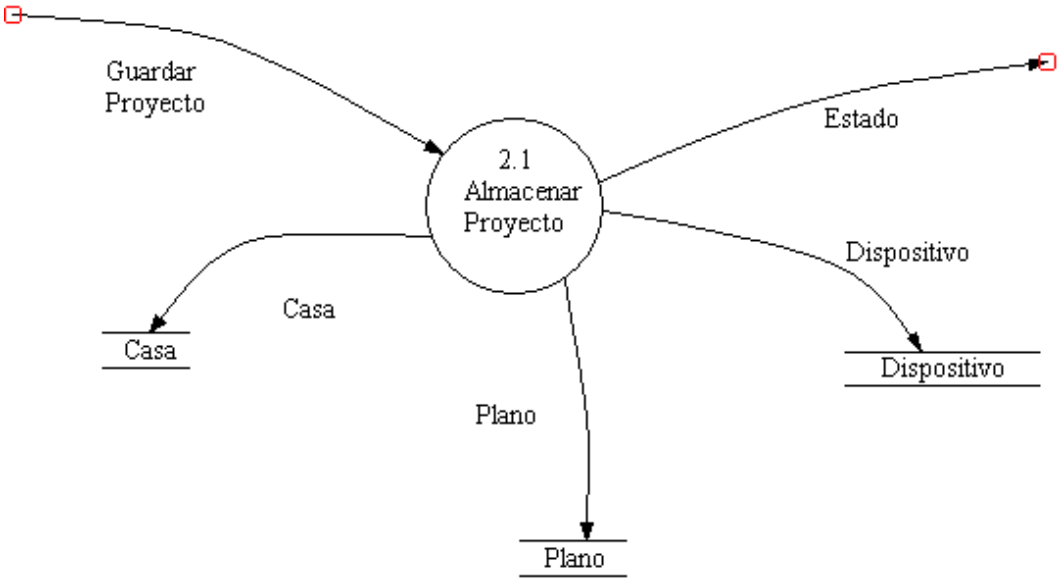
Parar Proyecto



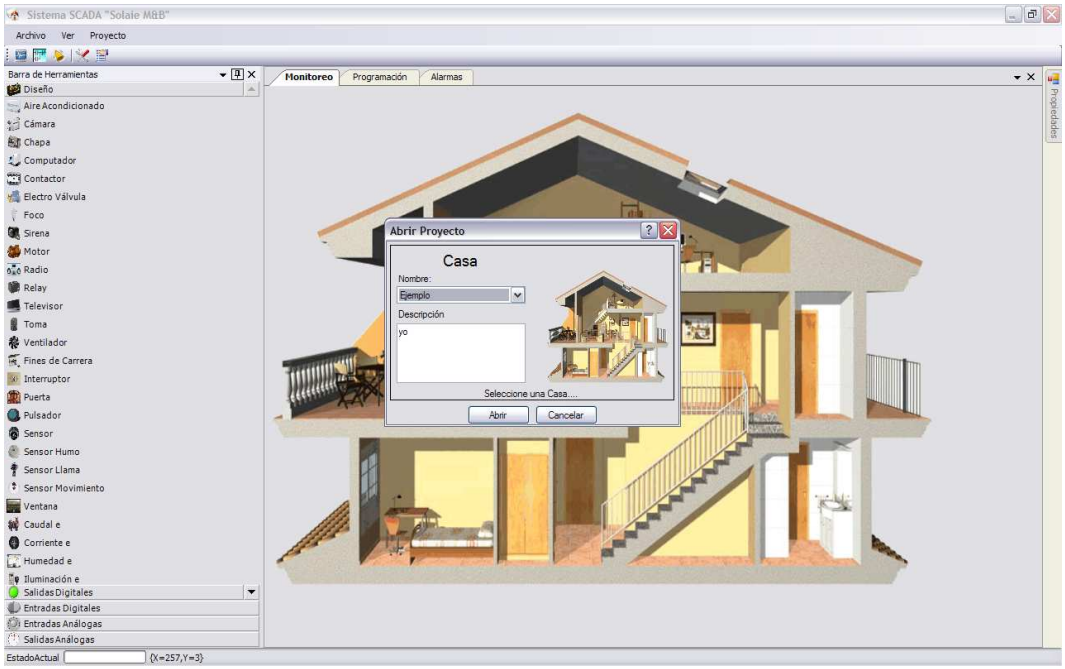


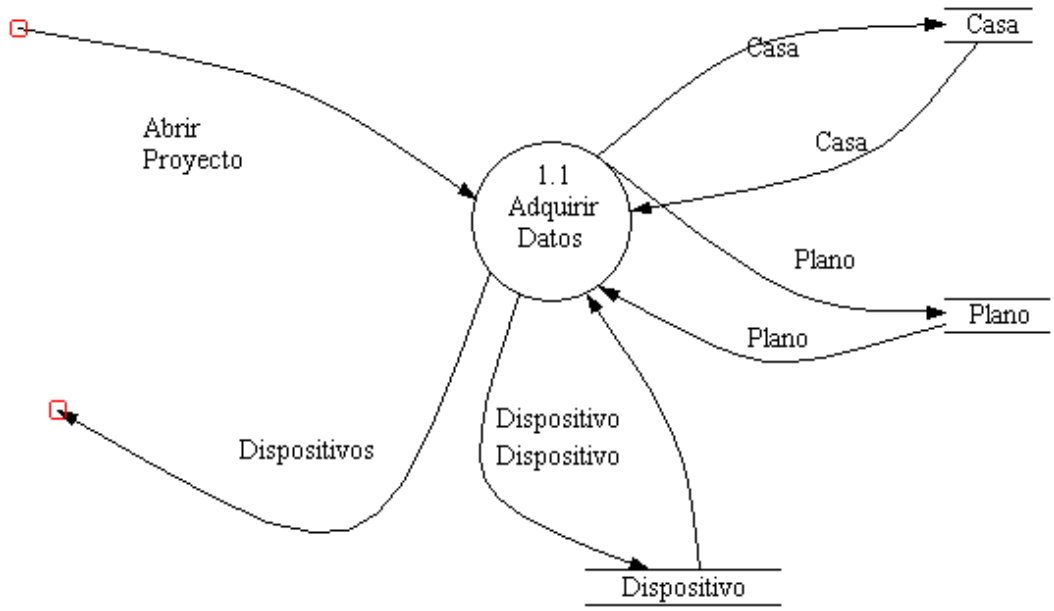
Almacenar Proyecto



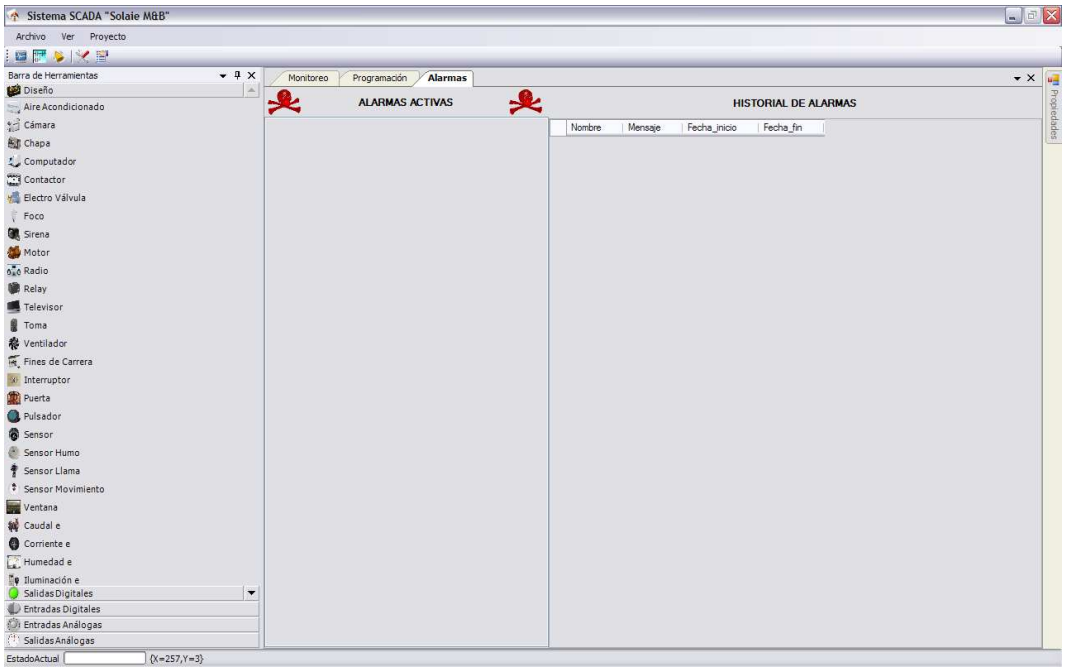


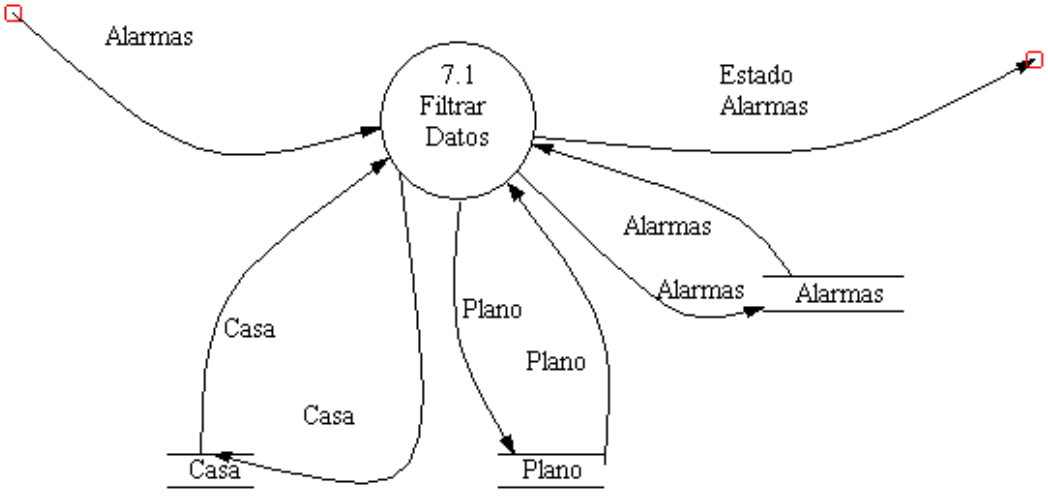
Abrir Proyecto





Alarmas Proyecto





DOCUMENTACIÓN DE CLASES

Lista de Espacios de Nombres

Los espacios de nombres especificados en el documento son:

Namespace	Assembly
mftc.SCADA	SCADALA

Namespace : mftc.SCADA
mftc.SCADA Tipo Lista

Enumerations

Tipo	Resumen
csAlarma.enumAlarma	
csContador.AumentarEn	
csControles.operador	Tipo de operador Lógico.
csTabProgramación.operador	
EstadoActualiza	
PlaySoundFlags	
signo	

Clases

Tipo	Resumen
Conexión	Maneja todos los parámetros de conexión con la Base de Datos de MySql.
csAlarma	Inicializa dispositivos con características del registro Alarma.
csCasa	Referencia los datos necesarios de la casa.
csContador	Inicializa dispositivos con características del registro Contador.
csControles	Descripción de csControles.
csControles.ContrastEditor	Clase que implementa los temporizadores para desplegar los controles de Tiempo.
csControlMarca	Inicializa dispositivos con características del registro de Marcas.
csDatos	Crea la Base de datos e ingresa los datos por defecto a la Base de Datos.
csEntorno	Clase para referenciar las variables necesarias del Sistema SCADALA.
csEntradaDigital	Instancia dispositivos con características de Entrada Digital.
csMarca	Clase que permite trabajar al código generado como marcas.
csMarcaPropia	Inicializa dispositivos con características del registro de Marcas.
csMarcaTemporizador	Inicializa dispositivos con características del registro de Marcas de Temporizador.
csOpc	Establece las variables necesarias entre el sistema y el servidor OPC.
csPlano	Referencia los datos necesarios para el Plano
csSalidaAnáloga	Inicializa dispositivos con características del registro de Salidas Análogas.
csSalidaDigital	Inicializa dispositivos con características del registro de Salidas Digitales.
csTabProgramación	Descripción de csTabProgramación.
csTemporizador	Inicializa dispositivos con características del registro Temporizador.
csTemporizadorDias	No implementado. Inicializa dispositivos con características del registro Temporizador Días.
csTemporizadorOff	Inicializa dispositivos con características del registro Temporizador OFF.
csUsuario	Clase que hace referencia a la Información del Usuario.
DateTimeSlicker	Represents an enhanced Windows date-time picker control.
fmrControlErrores	Controla los errores que se produscan en el Sistema.
frmAbrirProyecto	Descripción de frmAbrirProyecto.
frmAcercaDe	Descripción breve del desarrollo del Sistema SCADALA.
frmAlarma	Descripción de frmAlarma.
frmAlarmaInteractiva	Descripción de frmAlarmaInteractiva.

frmContrast	Forma que permite ingresar los tiempos a la propiedad del dispositvo.
frmGuardar	Descripción de frmGuardar.
frmMonitoreo	Descripción de frmDiseño.
frmProgramación	Descripción de frmProgramación.
frmPropiedades	Descripción de frmPropiedades.
frmToolBox	Descripción de frmToolBox.
MainForm	Descripción de MainForm.
RuleConverter	Clase para desplegar una lista en la propiedad de tipo de registro de los objetos de la clase csControles.
Sound	Reproduce Sonido, útil para las alarmas. Utilizando la librería de Windows.

mftc.SCADA Enumerations

csControles.operador Enumeration

Resumen

nestedPublic enumeration csControles.operador

Tipo de operador Lógico.

Enumeration Members

Campo
Diferente
Igual
Mayor
Menor
Ninguno
O
XOR
Y

EstadoActualiza Enumeration

Resumen

public enumeration EstadoActualiza

Enumeration Members

Campo
Apaga
Ninguno
Prende

csTabProgramación.operador Enumeration

Resumen

nestedPublic enumeration csTabProgramación.operador

Enumeration Members

Campo
Diferente
Igual

Mayor
Menor
Ninguno
O
XOR
Y

signo Enumeration

Resumen

public enumeration signo

Enumeration Members

Campo
Igual
Mayor
MayorIgual
Menor
MenorIgual

csContador.AumentarEn Enumeration

Resumen

nestedPublic enumeration csContador.AumentarEn

Enumeration Members

Campo
Off
On

csAlarma.enumAlarma Enumeration

Resumen

nestedPublic enumeration csAlarma.enumAlarma

Enumeration Members

Campo
comienza
inactiva
termina

PlaySoundFlags Enumeration

Resumen

public enumeration PlaySoundFlags

Enumeration Members

Campo
SND_ASYNC
SND_FILENAME

SND_LOOP
SND_NODEFAULT
SND_NOSTOP
SND_NOWAIT
SND_RESOURCE
SND_SYNC

mftc.SCADA Classes

Conexión Class

Resumen

public class Conexión

Maneja todos los parámetros de conexión con la Base de Datos de MySql.

Constructores

Nombre	Acceso	Resumen
Conexión()	public	Constructor que instancia la Conexión con el Motor de Base de Datos.

Propiedades

Nombre	Acceso	Resumen
BaseDatos : String	public	Nombre de la Base de Datos a la que se desea acceder.
CadenaConexion : String	public	Mantiene todos los parámetros de Conexión.
Clave : String	public	Contraseña asignada al usuario de BDD.
Servidor : String	public	Nombre o Dirección IP del equipo que es Servidor de MySql, en este caso es localhost.
Usuario : String	public	Usuario de la Base de Datos.

Métodos

Nombre	Acceso	Resumen
AbrirConexion() : Void	public	Abre la conexión con el Motor de BDD, verificando si la conexión esta cerrada.
CerrarConexion() : Void	public	Cierra la Conexión con la BDD.
Conexion() : MySqlConnection	public	Método que devuelve el objeto de Conexión con la BDD.

csControles Class

Resumen

public class csControles : System.IDisposable

Descripción de csControles.

Constructores

Nombre	Acceso	Resumen
csControles()	public	Permite crear un objeto de tipo csControl, el cual contiene propiedades y objetos que sirven para representar visualmente e internamente a los dispositivos. Este se debe usar si es la primera vez que se crea el dispositivo, ya que a este dispositivo hay que determinar el Nombre único.
csControles()	public	Permite crear un objeto de tipo csControl, el cual contiene propiedades y objetos que sirven para representar visualmente e internamente a los dispositivos. Este se debe usar si se tiene conocimiento del id del dispositivo, es decir cuando se desea abrir un elemento guardado.

Campo Members

Nombre	Acceso	Resumen

p1 : PictureBox	public	Elementos Picture para realizar la interfáz del Control en el Panel de Diseño.
prpMenuConf : String	protected	Nombre del grupo de Propiedades de Configuración
prpMenuGen : String	protected	Nombre del grupo de Propiedades Generales.
prpMenuOp : String	protected	Nombre del grupo de Propiedades del Servidor OPC.

Propiedades

Nombre	Acceso	Resumen
Descripción : String	public	Breve descripción de la utilidad del dispositivo.
DirecciónOPC : String	public	Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Id : String	public	Identificador único que el sistema usa para manipular el objeto.
IdColección : Int32	public	Posición del elemento en la colección Padre.
IdOpc : Int32	public	Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
LiActivo : String	public	Sirve para determinar la lista de Entradas io Salidas Análogas-Digitales se las puede utilizar para que se visualicen en la propiedad de tipo de registro de los dispositivos.
LocDispositivo : Point	public	Ubicación del dispositivo en el Área de Diseño.
NombreDispositivo : String	public	Nombre que el usuario da al dispositivo.
NumPicture : Int32	public	Hace referencia al Número de la imágen del Dispositivo.
OnDrag : Boolean	public	Permite saber si algún control esta siendo reubicado.
TipoRegistro : String	public	Característica del dispositivo, dependiente de la variable del Servidor OPC.

Métodos

Nombre	Acceso	Resumen
Completar() : String	public	Método que completa las horas, minutos y segundos con 0.
Dispose() : Void	public	Inicializa la Liberación de Memoria.
Dispose() : Void	protected	Libera memoria cuando el objeto se destruye.
Finalize() : Void	protected	Destructor de la Clase. Utilizado para liberar memoria.
FocusTextBox() : Void	public	Establece el foco del cursor en el recuadro para escribir algún valor en las Salidas Análogas.
Imagen() : PictureBox	public	Sirve como conexión gráfica entre el Área de Diseño y el Objeto.
MuevePicture() : Void	public	Permite manipular la ubicación del dispositivo.
TipoSeñal() : String	public	Basandose en el nombre del dispositivo, devuelve el tipo de registro o señal que esta almacenada para cada uno de ellos.
ToString() : String	public	Visualiza el Nombre del Dispositivo, cuando se haga referencia al objeto

csControles.ContrastEditor Class

Resumen

nestedPublic class csControles.ContrastEditor : UITipoEditor

Clase que implementa los temporizadores para desplegar los controles de Tiempo.

Constructores

Nombre	Acceso	Resumen
csControles.ContrastEditor()	public	Initializes a new instance of the class.

Métodos

Nombre	Acceso	Resumen
EditValue() : Object	public	Setea el valor a la propiedad.
GetEditStyle() : UITipoEditorEditStyle	public	Visualiza el boton del dropdown en la propiedad.

RuleConverter Class

Resumen

public class RuleConverter : StringConverter

Clase para desplegar una lista en la propiedad de tipo de registro de los objetos de la clase csControles.

Constructores

Nombre	Acceso	Resumen
RuleConverter()	public	Initializes a new instance of the class.

Métodos

Nombre	Acceso	Resumen
GetStandardValues() : TipoConverter#StandardValuesCollection	public	Valores que se almacenan en la lista al momento de ser desplegado el combo.
GetStandardValuesExclusive() : Boolean	public	Limita la lista
GetStandardValuesSupported() : Boolean	public	Despliega el Combo Box.

csOpc Class

Resumen

public class csOpc : System.IDisposable

Establece las variables necesarias entre el sistema y el servidor OPC.

Constructores

Nombre	Acceso	Resumen
csOpc()	public	Declara las variables que se comunican con el OPC. Si una variable falla o no esta especificada en el OPC, todos los valores del OPC fallan.

Propiedades

Nombre	Acceso	Resumen
CntListaDirección : Container	public	Lista que contiene los caminos entre los dispositivos y el Servidor OPC.

Métodos

Nombre	Acceso	Resumen
Dispose() : Void	public	Libera memoria cuando la clase ya no sea necesaria.

fmrControlErrores Class

Resumen

public class fmrControlErrores : Form, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl

Controla los errores que se produscan en el Sistema.

Constructores

Nombre	Acceso	Resumen
fmrControlErrores()	public	Control de Errores al nivel de Base de Datos.
fmrControlErrores()	public	Controla los errores al nivel de Aplicación.

Campo Members

Nombre	Acceso	Resumen
frm : MainForm	public	Variable necesaria para apuntar a la forma Principal.

Métodos

Nombre	Acceso	Resumen
Dispose() : Void	protected	Disposes resources used by the form.

csDatos Class

Resumen

public class csDatos

Crea la Base de datos e ingresa los datos por defecto a la Base de Datos.

Constructores

Nombre	Acceso	Resumen
csDatos()	public	Constructor de la clase csDatos.
csDatos()	public	Construye la clase csDatos.

frmAcercaDe Class

Resumen

public class frmAcercaDe : Form, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl

Descripción breve del desarrollo del Sistema SCADALA.

Constructores

Nombre	Acceso	Resumen
frmAcercaDe()	public	Constructor de la Forma.

Métodos

Nombre	Acceso	Resumen
Dispose() : Void	protected	Disposes resources used by the form.

frmAlarma Class

Resumen

public class frmAlarma : DockContent, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl, WeifenLuo.WinFormsUI.Docking.IDockContent

Descripción de frmAlarma.

Constructores

Nombre	Acceso	Resumen
frmAlarma()	public	Constructor de la Forma.

Propiedades

Nombre	Acceso	Resumen
AllowEndUserDocking : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
AutoHidePortion : Double	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
CloseButton : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DgvHistorial : DataGridView	public	Grid de historial de alarmas.
DockAreas : DockAreas	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockHandler : DockContentHandler	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockPanel : DockPanel	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
HideOnClose : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsActivated : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsFloat : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsHidden : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
LbxAlarmasActivas : ListBox	public	Lista de Alarmas Activas.
Pane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PanelPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ShowHint : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenu : ContextMenu	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenuStrip : ContextMenuStrip	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ToolTipText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
VisibleState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

Métodos

Nombre	Acceso	Resumen
Activate() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Dispose() : Void	protected	Disposes resources used by the form.
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatAt() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
GetPersistString() : String	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Hide() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsDockStateValid() : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
OnDockStateChanged() : Void	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

frmAlarmaInteractiva Class

Resumen

public class frmAlarmaInteractiva : DockContent, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl, WeifenLuo.WinFormsUI.Docking.IDockContent

Descripción de frmAlarmaInteractiva.

Constructores

Nombre	Acceso	Resumen
frmAlarmaInteractiva()	public	Inicializador de la Forma.

Propiedades

Nombre	Acceso	Resumen
AllowEndUserDocking : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
AutoHidePortion : Double	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
CloseButton : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockAreas : DockAreas	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockHandler : DockContentHandler	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockPanel : DockPanel	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
HideOnClose : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsActivated : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsFloat : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsHidden : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
LbxAlarmas : ListBox	public	Lista de Alarmas Interactivas
Pane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PanelPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ShowHint : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenu : ContextMenu	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenuStrip : ContextMenuStrip	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ToolTipText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
VisibleState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

Métodos

Nombre	Acceso	Resumen
Activate() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Dispose() : Void	protected	Disposes resources used by the form.
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatAt() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
GetPersistString() : String	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Hide() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsDockStateValid() : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
OnDockStateChanged() : Void	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

frmMonitoreo Class

Resumen

public class frmMonitoreo : DockContent, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke,

System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl, WeifenLuo.WinFormsUI.Docking.IDockContent

Descripción de frmDiseño.

Constructores

Nombre	Acceso	Resumen
frmMonitoreo()	public	Constructor de la Forma.

Propiedades

Nombre	Acceso	Resumen
AllowEndUserDocking : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
AutoHidePortion : Double	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
CloseButton : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockAreas : DockAreas	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockHandler : DockContentHandler	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockPanel : DockPanel	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
HideOnClose : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsActivated : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsFloat : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsHidden : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Pane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PanelPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PnlDiseño : Panel	public	Panel que sirve de contenedor oara los objetos de Monitoreo del Sistema SCADALA.
ShowHint : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenu : ContextMenu	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenuStrip : ContextMenuStrip	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ToolTipText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
VisibleState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

Métodos

Nombre	Acceso	Resumen
Activate() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Dispose() : Void	protected	Disposes resources used by the form.
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatAt() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
GetPersistString() : String	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Hide() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsDockStateValid() : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
OnDockStateChanged() : Void	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

frmProgramación Class

Resumen

public class frmProgramación : DockContent, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl, WeifenLuo.WinFormsUI.Docking.IDockContent

Descripción de frmProgramación.

Constructores

Nombre	Acceso	Resumen
frmProgramación()	public	Constructor de la Forma.

Propiedades

Nombre	Acceso	Resumen
AllowEndUserDocking : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
AutoHidePortion : Double	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
CloseButton : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockAreas : DockAreas	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockHandler : DockContentHandler	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockPanel : DockPanel	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DwaProgramación : DrawArea	public	Espacio para realizar el diseño de los gráficos.
FloatPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
HideOnClose : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsActivated : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsFloat : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsHidden : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Pane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PanelPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PbxBarraEnergía : PictureBox	public	Barra de Energía del Área de Programación.
ShowHint : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenu : ContextMenu	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenuStrip : ContextMenuStrip	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ToolTipText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
VisibleState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

Métodos

Nombre	Acceso	Resumen
Activate() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Dispose() : Void	protected	Disposes resources used by the form.
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
EliminaRelación() : Void	public	Elimina las relaciones que el dispositivo tiene con los demás dispositivos.
FloatAt() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
GetPersistString() : String	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Hide() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsDockStateValid() : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
OnDockStateChanged() : Void	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PosiciónDrag() : MouseEventArgs	public	Obtiene la posición del Mouse, cuando se esta desplazando los objetos.

Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

frmPropiedades Class

Resumen

public class frmPropiedades : DockContent, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl, WeifenLuo.WinFormsUI.Docking.IDockContent

Descripción de frmPropiedades.

Constructores

Nombre	Acceso	Resumen
frmPropiedades()	public	Constructor de la Forma.

Propiedades

Nombre	Acceso	Resumen
AllowEndUserDocking : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
AutoHidePortion : Double	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
CbxPropiedad : ComboBox	public	Combo Box que almacena los objetos de las Áreas de Trabajo (Monitoreo y Programación).
CloseButton : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockAreas : DockAreas	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockHandler : DockContentHandler	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockPanel : DockPanel	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
HideOnClose : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsActivated : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsFloat : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsHidden : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Pane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PanelPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PrpDetalle : PropertyGrid	public	Property Grid, para modificar las propiedades de los elementos de Programación y Monitoreo.
ShowHint : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenu : ContextMenu	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenuStrip : ContextMenuStrip	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ToolTipText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
VisibleState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

Métodos

Nombre	Acceso	Resumen
Activate() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Dispose() : Void	protected	Disposes resources used by the form.

DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatAt() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
GetPersistString() : String	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Hide() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsDockStateValid() : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
OnDockStateChanged() : Void	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

frmToolBox Class

Resumen

public class frmToolBox : DockContent, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl, WeifenLuo.WinFormsUI.Docking.IDockContent

Descripción de frmToolBox.

Constructores

Nombre	Acceso	Resumen
frmToolBox()	public	Constructor de la Forma

Propiedades

Nombre	Acceso	Resumen
AllowEndUserDocking : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
AutoHidePortion : Double	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
CloseButton : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockAreas : DockAreas	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockHandler : DockContentHandler	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockPanel : DockPanel	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
HideOnClose : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsActivated : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsFloat : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsHidden : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Pane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
PanelPane : DockPane	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
ShowHint : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenu : ContextMenu	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabPageContextMenuStrip : ContextMenuStrip	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TabText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
TbxScada : ToolBox	public	Barra de Herramientas.
ToolTipText : String	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
VisibleState : DockState	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

Métodos

Nombre	Acceso	Resumen
Activate() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Dispose() : Void	protected	Disposes resources used by the form.
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
DockTo() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
FloatAt() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
GetPersistString() : String	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Hide() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
IsDockStateValid() : Boolean	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
OnDockStateChanged() : Void	protected	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*
Show() : Void	public	*(desde WeifenLuo.WinFormsUI.Docking.DockContent)*

frmAbrirProyecto Class

Resumen

public class frmAbrirProyecto : Form, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl

Descripción de frmAbrirProyecto.

Constructores

Nombre	Acceso	Resumen
frmAbrirProyecto()	public	Constructor de clase.

Métodos

Nombre	Acceso	Resumen
Dispose() : Void	protected	Disposes resources used by the form.

frmContrast Class

Resumen

public class frmContrast : Form, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl

Forma que permite ingresar los tiempos a la propiedad del dispositvo.

Constructores

Nombre	Acceso	Resumen
frmContrast()	public	Constructor.

Campo Members

Nombre	Acceso	Resumen
_wfes : IWindowsFormsEditorService	public	Variable que permite visualizar la forma dentro de la propiedad.

nudDecseg : NumericUpDown	public	Cantidad de decenas de segundos.
nudHora : NumericUpDown	public	Cantidad de horas.
nudMinutos : NumericUpDown	public	Cantidad de minutos.
nudSegundos : NumericUpDown	public	Cantidad de segundos.

Métodos

Nombre	Acceso	Resumen
Dispose() : Void	protected	Libera los recursos utilizados por la clase.

frmGuardar Class

Resumen

public class frmGuardar : Form, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl

Descripción de frmGuardar.

Constructores

Nombre	Acceso	Resumen
frmGuardar()	public	Constructor de la Forma Guardar.

Métodos

Nombre	Acceso	Resumen
Dispose() : Void	protected	Disposes resources used by the form.

MainForm Class

Resumen

public class MainForm : Form, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent, System.Windows.Forms.IContainerControl

Descripción de MainForm.

Constructores

Nombre	Acceso	Resumen
MainForm()	public	Constructor de la Forma Principal.

Propiedades

Nombre	Acceso	Resumen
BanCable : Boolean	public	Para conectar entre los elementos.
CbxDiseño : ComboBox	public	Combo de Propiedades.
CllProgra : ArrayList	public	Colección que guardará los objetos de tipo csTabProgramación.
DgvHistorial : DataGridView	public	Se accede al Data Grid que contiene el historial de alarmas.
DocPanel : DockPanel	public	Conenedor de Formas.
DwaProgramación : DrawArea	public	Para acceder al DrawArea.
FlagRed : Boolean	public	Bandera que permite establecer los niveles de relación entre los objetos ingresados en el Área de Programación.
FrmAlama : frmAlarma	public	Accede a la forma que contiene las Forma de Alarmas.
FrmAlarmaI : frmAlarmaInteractiva	public	Forma Alarma Interactiva.

FrmBarraHerramientas : frmToolBox	public	Accede a la forma que contiene las Forma Barra de Herramientas.
FrmMonitoreo : frmMonitoreo	public	Accede a la forma que contiene las Forma Monitoreo.
FrmPrograma : frmProgramación	public	Accede a la forma que contiene las Forma de Programación.
FrmPropiedades : frmPropiedades	public	Accede a la forma que contiene las Propiedades del Sistema
LbxAlarmas : ListBox	public	Lista de Alarmas.
Linea : DrawLine	public	Linea que sirve de diseño para la programación.
MnuProyectoParar : ToolStripMenuItem	public	Menú Parar Proyecto.
MstSCADA : MenuStrip	public	Propiedad de Menú Principal.
ObjRelación : Object	public	Objeto que se va a relacionar.
Padre : Object	public	Objeto Padre que permite establecer los niveles de relación entre los objetos ingresados en el Área de Programación.
PnlDiseño : Panel	public	Panel de Diseño.
PrpDiseño : PropertyGrid	public	Propiedades del Sistema.
Situación : ToolStripStatusLabel	public	Etiqueta para enseñar el estado del Sistema.
TssPosiciónMouse : ToolStripStatusLabel	public	Accede a la posición del Puntero del Ratón.
TstHerramientas : ToolStrip	public	Barra que contiene las diferentes Herramientas del Sistema.

Métodos

Nombre	Acceso	Resumen
CllDiseño() : ArrayList	public	Colección de Dispositivos de tipo csControles.
CtxMenuImage() : ContextMenuStrip	public	Menú de Contexto.
CuentaIdColección() : Int32	public	Cuenta los elementos que se hayan puesto en el tab Programación.
CuentaNombre() : Int32	public	Devuelve el Índice siguiente de la colección deseada. Si esta un registro intermedio eliminado, obtiene el índice de este.
Dispose() : Void	protected	Disposes resources used by the form.
EliminaLineas() : Void	public	Elimina las líneas de la anterior relación.
EliminaRelaciones() : Void	public	Elimina las relaciones que el objeto tuviera con los elementos de programación.
FocoDocumento() : Void	public	Inicializa las propiedades y o características nesesarias de cada Área de Trabajo.
Hijos() : Void	public	Permite establecer los niveles de relación entre los objetos ingresados en el Área de Programación.
ImlDiseño() : ImageList	public	Colección de Imágenes de Programación.
ImlProgramación() : ImageList	public	Colección de Imágenes de Programación.
Main() : Void	public	Inicio del Sistema.
MnuProyectoPararClick() : Void	public	Detiene el Proceso de Monitoreo y Programación del Sistema.
NumImagen() : Int32	public	Retorna el Número de la Imagen que le corresponde, de acuerdo al nombre.
OpcMonitoreo() : AxOpcdata	public	Objeto del Cliente OPC.
Pausa() : Void	public	Genera una pausa del sistema, no exacto pero bastante aproximado en centecimas de nano segundos. Es decir 1 Pausa es 100 nano segundos.
PosiciónDrag() : MouseEventArgs	public	Obtiene la posición del Mouse, cuando se esta desplazando los objetos.
Select() : DataSet	public	Ingresa código Sql para realizar las consultas con la BDD.
Signos() : String	public	
TtpPictureNombre() : ToolTip	public	Mensaje de Ayuda para setear los objetos.

csEntorno Class

Resumen

public abstract class csEntorno

Clase para referenciar las variables necesarias del Sistema SCADALA.

Propiedades

Nombre	Acceso	Resumen
Casa : csCasa	public	Clase de Tipo Casa.
Plano : csPlano	public	Clase de Tipo Plano.
Usuario : csUsuario	public	Clase de Tipo Usuario.

csCasa Class

Resumen

public class csCasa

Referencia los datos necesarios de la casa.

Constructores

Nombre	Acceso	Resumen
csCasa()	public	Constructor Básico, instancia las variables con valores String.Empty.
csCasa()	public	Constructor que referencia a la Casa.

Propiedades

Nombre	Acceso	Resumen
Codigo : String	public	Código de la casa que se esta usando.
Nombre : String	public	Nombre de la Casa que se esta usando.

csPlano Class

Resumen

public class csPlano

Referencia los datos necesarios para el Plano

Constructores

Nombre	Acceso	Resumen
csPlano()	public	Constructor Básico de la clase.
csPlano()	public	Constructor de clase.

Propiedades

Nombre	Acceso	Resumen
Camino : String	public	Camino en el cuál esta ubicado la imágen del Plano.
Codigo : String	public	Codigo del Plano.
Nombre : String	public	Nombre del Plano.

csUsuario Class

Resumen

public class csUsuario

Clase que hace referencia a la Información del Usuario.

Constructores

Nombre	Acceso	Resumen
csUsuario()	public	Constructor que instancia las variables de Usuario.

Propiedades

Nombre	Acceso	Resumen
Codigo : String	public	Código de Usuario.
Nombre : String	public	Nombre de Usuario.

csMarca Class

Resumen

public class csMarca

Clase que permite trabajar al código generado como marcas.

Constructores

Nombre	Acceso	Resumen
csMarca()	public	Instancia las marcas con el valor que se indique.

Propiedades

Nombre	Acceso	Resumen
Actualiza : Object	public	Valor que se inicializa con el valor de Valor, luego sirve como bandera para comprobar si hay cambios en esta para posteriormente escribir los cambios al Servidor OPC.
IndDirecciónOPC : Int16	public	Indice de la Dirección de la variable en la Colección.
UnionDeObjetos : String	public	Envia unida las especificaciones de las variables locales.
Valor : Object	public	Valor de la variable del Servidor OPC.

csTabProgramación Class

Resumen

public class csTabProgramación

Descripción de csTabProgramación.

Constructores

Nombre	Acceso	Resumen
csTabProgramación()	public	Constructor Genérico.
csTabProgramación()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Campo Members

Nombre	Acceso	Resumen
prpMenuConf : String	protected	Nombre del grupo de Propiedades de Configuración
prpMenuGen : String	protected	Nombre del grupo de Propiedades Extras.
prpMenuOp : String	protected	Nombre del grupo de Propiedades del Servidor OPC.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	Representa el estado del dispositivo.
FrmMain : MainForm	public	Forma Principal.
Id : String	public	Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	Índice de los dispositivos en la colección Padre.

IdOpc : Int32	public	Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	Invierte el resultado de las operaciones.
LocDispositivo : Point	public	Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	Relación de los objetos con este dispositivo.
Tarjeta : String	public	Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	Añade Controles al Picture
ClickPicture() : Void	public	Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	Empieza el Drag.
DragPictureRelación() : Void	public	Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	Evento que permite analizar el comportamiento del dispositivo.
MouseDownPicture() : Void	public	Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	Cuando coje el foco el mouse del control.
ToString() : String	public	Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csEntradaDigital Class

Resumen

public class csEntradaDigital : csTabProgramación

Instancia dispositivos con características de Entrada Digital.

Constructores

Nombre	Acceso	Resumen
csEntradaDigital()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Evento que permite analizar el comportamiento del dispositivo.

MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csSalidaDigital Class

Resumen

public class csSalidaDigital : csTabProgramación

Inicializa dispositivos con características del registro de Salidas Digitales.

Constructores

Nombre	Acceso	Resumen
csSalidaDigital()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.

Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Evento que permite analizar el comportamiento del dispositivo.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csSalidaAnáloga Class

Resumen

public class csSalidaAnáloga : csTabProgramación

Inicializa dispositivos con características del registro de Salidas Análogas.

Constructores

Nombre	Acceso	Resumen
csSalidaAnáloga()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.

PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Evento que permite analizar el comportamiento del dispositivo.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csMarcaPropia Class

Resumen

public class csMarcaPropia : csTabProgramación

Inicializa dispositivos con características del registro de Marcas.

Constructores

Nombre	Acceso	Resumen
csMarcaPropia()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto,

		necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Evento que permite analizar el comportamiento del dispositivo.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csControlMarca Class

Resumen

public class csControlMarca : csTabProgramación

Inicializa dispositivos con características del registro de Marcas.

Constructores

Nombre	Acceso	Resumen
csControlMarca()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Signo : signo	public	Permite hacer comparaciones lógicas.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
IFOperador() : Boolean	public	
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csMarcaTemporizador Class

Resumen

public class csMarcaTemporizador : csTabProgramación

Inicializa dispositivos con características del registro de Marcas de Temporizador.

Constructores

Nombre	Acceso	Resumen
csMarcaTemporizador()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.

IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Signo : signo	public	
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : TimeSpan	public	

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
IFOperador() : Boolean	public	
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	Tiempo inicial que se seteo el timer.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csContador Class

Resumen

public class csContador : csTabProgramación

Inicializa dispositivos con características del registro Contador.

Constructores

Nombre	Acceso	Resumen
csContador()	public	Este control se energizará cuando cumpla un cierto número de eventos on/off. Posteriormente el contador se resetea si se desea o deja de funcionar, quedando activo el paso de energía.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Incremento : Int32	public	
InteractuarEn : csContador#AumentarEn	public	
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxReset : PictureBox	public	
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
RelaciónReset : ArrayList	public	
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.

DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureRelaciónIzquierdaReset() : Void	public	Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	Aumenta el contador.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MouseDownPictureIzquierdaReset() : Void	public	
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
NuevosPuntillos() : Point[]	public	
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csTemporizador Class

Resumen

public class csTemporizador : csTabPrograмación

Inicializa dispositivos con características del registro Temporizador.

Constructores

Nombre	Acceso	Resumen
csTemporizador()	public	Este control se energizará cuando cumpla un cierto número de eventos on/off. Posteriormente el contador se resetea si se desea o deja de funcionar, quedando activo el paso de energía.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Intervalo : String	public	
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo :	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.

String		
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
TmrInterno : TimeSpan	public	
Valor : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	Aumenta el contador.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csTemporizadorOff Class

Resumen

public class csTemporizadorOff : csTabProgramación

Inicializa dispositivos con características del registro Temporizador OFF.

Constructores

Nombre	Acceso	Resumen
csTemporizadorOff()	public	Este control se energizará cuando cumpla un cierto número de eventos on/off. Posteriormente el contador se resetea si se desea o deja de funcionar, quedando activo el paso de energía.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Intervalo : String	public	
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
TmrInterno : TimeSpan	public	
Valor : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.

DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	Aumenta el contador.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

csAlarma Class

Resumen

public class csAlarma : csTabPrograмación

Inicializa dispositivos con características del registro Alarma.

Constructores

Nombre	Acceso	Resumen
csAlarma()	public	Genera el objeto con todos los requerimientos necesarios para poderlo enlazar, simulando grafos, con otros objetos. Este permite setear las propiedades necesarias para poder iniciar un objeto.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
Estado : csAlarma#enumAlarma	public	
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
Mensaje : String	public	
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Valor del Dispositivo.

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
CargarAlarmas() : Void	public	
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	
InsertaAlarma() : csAlarma#enumAlarma	public	Ingresa información almacenada en la BDD.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	
Ventana() : Void	public	

csTemporizadorDias Class

Resumen

public class csTemporizadorDias : csTabProgramación

No implementado. Inicializa dispositivos con características del registro Temporizador Días.

Constructores

Nombre	Acceso	Resumen
csTemporizadorDias()	public	Este control se energizará cuando cumpla un cierto número de eventos on/off. Posteriormente el contador se resetea si se desea o deja de funcionar, quedando activo el paso de energía.

Propiedades

Nombre	Acceso	Resumen
DirecciónOPC : String	public	*(desde mftc.SCADA.csTabProgramación)* Ruta a la que el dispositivo apunta, haciendo referencia a una variable del Servidor OPC.
Energizado : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Representa el estado del dispositivo.
FrmMain : MainForm	public	*(desde mftc.SCADA.csTabProgramación)* Forma Principal.
Id : String	public	*(desde mftc.SCADA.csTabProgramación)* Identificador único, que permite al sistema ubicar al

		dispositivo.
IdColección : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Índice de los dispositivos en la colección Padre.
IdOpc : Int32	public	*(desde mftc.SCADA.csTabProgramación)* Representación entera en la Colección Padre del camino de la dirección OPC que el dispositivo hace referencia.
Intervalo : DateTime	public	
Invertido : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Invierte el resultado de las operaciones.
LocDispositivo : Point	public	*(desde mftc.SCADA.csTabProgramación)* Ubicación del dispositivo en el Área de Programación.
NombreDispositivo : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre que el usuario da al dispositivo.
P1 : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Elementos Picture para realizar la interfáz del Control.
PbxDerecha : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
PbxIzquierda : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Cuadro de Dibujo que sirve para relacionar al elemento mediante métodos de drag and drop.
RelaciónDerecha : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación no implementada de los objetos con este dispositivo.
RelaciónIzquierda : ArrayList	public	*(desde mftc.SCADA.csTabProgramación)* Relación de los objetos con este dispositivo.
Tarjeta : String	public	*(desde mftc.SCADA.csTabProgramación)* Nombre de la Tarjeta de Adquisición de Datos.
TipoRegistro : String	public	*(desde mftc.SCADA.csTabProgramación)* Característica del dispositivo, dependiente de la variable del Servidor OPC.
Valor : Int32	public	

Métodos

Nombre	Acceso	Resumen
AñadeControl() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Añade Controles al Picture
ClickPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
ControlYA() : Boolean	public	*(desde mftc.SCADA.csTabProgramación)* Comprueba si existe un objeto ya relacionado.
Dispose() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento para Liberar Memoria.
Dispose() : Void	protected	*(desde mftc.SCADA.csTabProgramación)* Libera los recursos utilizados por este.
DragComienzaPictureIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando el objeto entra al área del control, habilita o deshabilita el efecto del drag.
DragPictureComienza() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Empieza el Drag.
DragPictureRelación() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Elimina la línea de diseño de programación si se ha soltado el drag sobre este.
DragPictureRelaciónDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* No implementada, Relaciona al objeto con el objeto que esta siendo arrastrado.
DragPictureRelaciónIzquierda() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se suelta el objeto previamente arrastrado sobre el área del Picture Izquierda, este crea la relacion entre los elementos.
DragPictureRelaciónIzquierdaAnteriro() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando finaliza el arrastrado del control.
DragPictureSobre() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando se esta arrastrando un objeto, necesita redibujar la línea de diseño del Área de Programación.
Imagen() : PictureBox	public	*(desde mftc.SCADA.csTabProgramación)* Sirve como conexión gráfica entre el Área de Programación y el Objeto.
Iniciar() : Boolean	public	Aumenta el contador.
MouseDownPicture() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Permite seleccionar al elemento cuando se presiona sobre el cuadro de imagen.
MouseDownPictureDerecha() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Inicializa el evento Drag, que permite relacionar entre objetos.
MousePictrueEnter() : Void	public	*(desde mftc.SCADA.csTabProgramación)* Cuando coje el foco el mouse del control.
ToString() : String	public	*(desde mftc.SCADA.csTabProgramación)* Visualiza el nombre del Dispositivo cuando se hace referencia al objeto.

DateTimeSlicker Class

Resumen

public class DateTimeSlicker : DateTimePicker, System.ComponentModel.IComponent, System.IDisposable, System.Windows.Forms.IDropTarget, System.ComponentModel.ISynchronizeInvoke, System.Windows.Forms.IWin32Window, System.Windows.Forms.IBindableComponent

Represents an enhanced Windows date-time picker control.

Constructores

Nombre	Acceso	Resumen
DateTimeSlicker()	public	Initializes a new instance of the class.

Propiedades

Nombre	Acceso	Resumen
Checked : Boolean	public	Gets or sets a value indicating whether the property has been set with a valid date-time value and the displayed value is able to be updated.
CustomFormat : String	public	Gets or sets the custom date-time format string.
Format : DateTimePickerFormat	public	Gets or sets the format of the date and time displayed in the control.
Text : String	public	Gets or sets the text associated with this control.
Value : Object	public	Gets or sets the date-time value assigned to the control.

Métodos

Nombre	Acceso	Resumen
OnCheckedChanged() : Void	protected	Raises the event.
OnCustomFormatChanged() : Void	protected	Raises the event.
OnFormatChanged() : Void	protected	Raises the event.
OnKeyDown() : Void	protected	Raises the event.
WndProc() : Void	protected	

Sound Class

Resumen

public class Sound

Reproduce Sonido, útil para las alarmas. Utilizando la librería de Windows.

Constructores

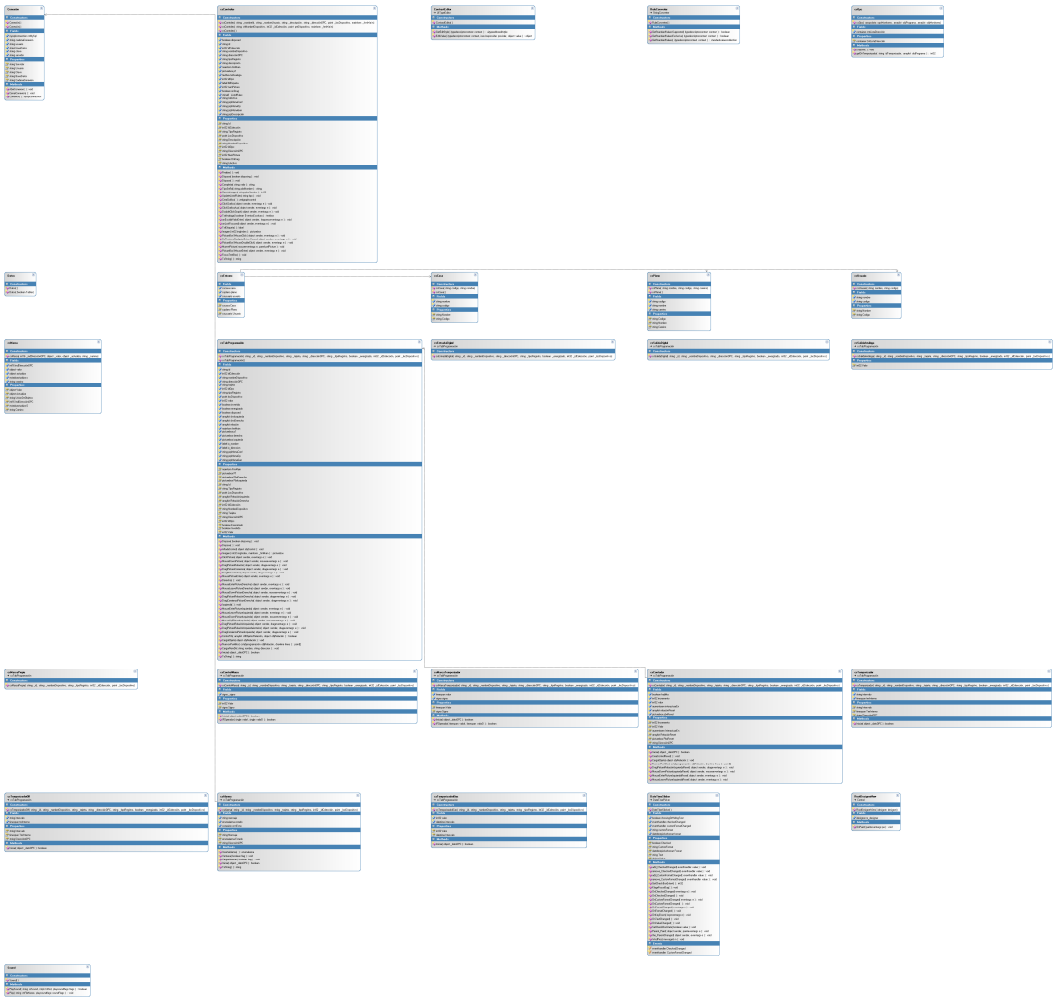
Nombre	Acceso	Resumen
Sound()	public	Initializes a new instance of the class.

Métodos

Nombre	Acceso	Resumen
Play() : Void	public	Inicia reproducción de Sonido.

ANEXO 4

DIAGRAMA DE CLASES



Fin… ☻ 10/9/2007