Universidad Don Bosco

Facultad de Ingeniería

Escuela de Electrónica

Simulador de plantas para automatismos usando LabVIEW

Trabajo de graduación

Para optar al título de:

Ingeniero en Automatización

Presentado por:

Zeus Bladimir Hernández Gavidia

Asesor:

Ing. Jorge Alonso López Salazar

Ciudadela Don Bosco, Diciembre de 2007

Universidad Don Bosco

Facultad de Ingeniería

Autoridades:

Ing. Federico Huguet Rivera

Rector

Pbro. Víctor Bermúdez sdb.

Vicerrector Académico

Lic. Mario Rafael Olmos

Secretario General

Ing. Ernesto Godofredo Girón

Decano Facultad de Ingeniería

Universidad Don Bosco

Tribunal Examinador

_____

Ing. Jorge Alonso López Salazar

Asesor de Tesis

_____ _____

Ing. Alexander Guzman Koch

Jurado Evaluador

Ing. Víctor Callejas

Jurado Evaluador

_____

Ing. Edgardo Cruz Zeledón

Jurado Evaluador

AGRADECIMIENTOS

Agradezco a Dios por todopoderoso por haber guiado mis pasos en este proceso educativo y darme fuerza para culminarlos satisfactoriamente.

A mis padres por todo el apoyo que me han brindado por todos los sacrificios que han hecho por mi superación como persona y como profesional, por ser la principal fuente de inspiración de mi proyecto de vida.

A mi familia en general porque cada uno a tenido que ver de alguna forma en la consecución de este triunfo.

A mi novia Jennifer por el apoyo y ánimos en esta última etapa del trabajo de graduación.

A todos mis amigos porque con nuestros esfuerzos conjuntos podemos ver el fruto de los sacrificios y desvelos que la carrera demandó.

A todos los maestros que ayudaron a formar mi carácter y me inspiraron a luchar por mis sueños y aspiraciones, en especial al Ing. Jorge López profesor y asesor gracias a el por todo el apoyo y consejos recibidos.

ÍNDICE

I. DEFINICIÓN DEL TEMA

Simulador de plantas[1] *para automatismos usando LabVIEW,* consiste en una aplicación en LabVIEW para Simulación de Plantas y procesos, para proporcionar al usuario el entorno necesario para el aprendizaje en el diseño y prueba de sistemas de control basados en autómatas programables.

[1] Con este término se hace referencia en el documento a una planta industrial.

II. DESCRIPCIÓN DE LA PROPUESTA DEL PROYECTO

El proyecto consiste en la elaboración de una plataforma gráfica que permita a los usuarios desarrollar simulaciones de procesos en una computadora, los cuales podrán ser controlados por un PLC[2] real.

Permite simular un buen número de elementos, necesarios para desarrollar una aplicación de automatización. Estos elementos van a ser de carácter discreto.

La forma que el usuario va a utilizar el simulador es de manera interactiva, tendrá una paleta de controles y un espacio de trabajo donde se dibujará la planta.

Estos objetos se van a relacionar entre sí y con una entrada/salida del PLC. Luego de haber dibujado por completo su planta y relacionado los objetos del simulador, se procederá a ejecutar la simulación

La aplicación tendrá la capacidad de interactuar con un PLC real, registrando los cambios que se efectúan en el proceso, y visualizando la información relevante con respecto al proceso simulado.

La parte operativa del simulador constará de dos etapas: una de diseño y otra de ejecución. En la parte de diseño se dibujará la planta y se relacionará los objetos de simulación con las entradas/salidas reales del PLC. En la etapa de ejecución se apreciará la visualización del comportamiento del sistema diseñado.

[2] **PLC** Controlador Lógico Programable

III. JUSTIFICACIÓN

El área de la automatización en nuestro entorno va tendiendo a aumentar, tiene como objetivo liberar al ser humano de operaciones rutinarias, disminuyendo así errores y a su vez aumentando la producción. La automatización hoy en día es más requerida que nunca para que una industria mantenga la competitividad.

Es por eso que como estudiante en esta área, es necesario conocer el mayor número posible de variantes y de técnicas que se aplican para construir soluciones a problemas que implican la automatización de un proceso industrial.

En la actualidad, los autómatas programables (PLC's) son los más utilizados para dar solución de problemas en automatización. Los procesos o plantas que se automatizan varían en dimensiones y obviamente realizar pruebas con elementos reales para un estudiante se hace de manera parcial en el laboratorio.

En el entorno de laboratorio de la universidad, no se dispone de una herramienta para simular plantas, pero si hay pequeñas plantas didácticas que son:

- Sistema Hidráulico.
- Sistema Térmico
- Motor didáctico

Las plantas didácticas mencionadas ayudan a realizar las prácticas, pero se limitan a estos sistemas en particular; por otro lado, con el simulador se posibilita contar una variedad de elementos para poder construir diversas plantas industriales discretas, enriqueciendo así los conocimientos y facilitando el aprendizaje.

Con esta aplicación se pretende dar al usuario las bases para poder observar el desempeño del autómata programable en conjunto con la simulación de la planta en LabVIEW.

Además, esta aplicación puede llegar más allá del ambiente del laboratorio, porque puede ser utilizada por integradores de sistemas de automatización, para hacer simulaciones en plantas a implementar sin tener la necesidad de hacer pruebas que puedan detener la producción de un sistema que esté en funcionamiento o que requiera del paro de alguna línea de producción solo para hacer pruebas, las cuales con dicho simulador fácilmente pueden ser efectuadas.

IV. OBJETIVOS

IV.1. OBJETIVO GENERAL

Desarrollar una aplicación usando LabVIEW que permita simular plantas y procesos industriales discretos, para facilitar al usuario la comprensión, prueba evaluación y corrección de sistemas automatizados basados en PLC's.

IV.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una aplicación en LabVIEW que permita interactuar con los PLC del laboratorio de instrumentación y control de la Universidad Don Bosco.
- Diseñar una plataforma gráfica que permita simular plantas y procesos industriales discretos, que pueda comunicarse con un PLC real para el control de éstas.
- Presentar la documentación completa sobre cómo efectuar la comunicación entre los PLC's y LabVIEW.

V. ALCANCES

- El sistema contará con la capacidad de simular algunos elementos de planta comunes (sensores, actuadores y dispositivos físicos). La cantidad y el tipo de elementos que se incluirán en el sistema será determinado tomando en cuenta la opinión de los usuarios potenciales del sistema (alumnos, profesores y profesionales), a través de una serie de consultas y entrevistas que se llevarán a cabo para tal efecto.
- Realizar la conexión de los autómatas programables con el software LabVIEW de National Instruments, ambos disponibles en el laboratorio de instrumentación y control. La documentación incluirá las bases para poder conectar cualquier autómata, incluso de otra marca.
- El usuario podrá dibujar la planta en la etapa de diseño de la misma. Posteriormente al diseño, podrá visualizarse el proceso para observar el comportamiento de la misma en conjunto con el PLC.
- Dibujo de planta de manera interactiva (gráfica) mediante cuadros de controles. Dichos cuadros incluyen todos los elementos de simulación por definir.
- El sistema tendrá la capacidad de simular los programas del PLC del usuario sin necesidad de modificarlos para que puedan interactuar con el simulador.
- Por otro lado, para la realización de estas aplicaciones se llevará a cabo una investigación para comunicar LabVIEW con los PLC's, la cual quedará documentada para que pueda ser retomada para cualquier otro tipo de aplicación que implique la integración de estos dos recursos.

VI. LIMITACIONES

- La aplicación solo podrá ser utilizada con los autómatas Simatic S5 y Simatic S7.
- La aplicación mostrará de forma básica, en el proceso de simulación, los cambios o las activaciones de los elementos dentro de la simulación. Es decir, no soportará animaciones complejas, como por ejemplo el movimiento de partes. El sistema se limitará a desplegar en pantalla la información visual necesaria para que el usuario comprenda qué es lo que está ocurriendo en la planta simulada.
- La variedad de sensores y actuadores que soportará el sistema, así como sus características o propiedades, se limitarán a proporcionar como mínimo una funcionalidad como la que se obtiene por medio del uso de las plantas industriales didácticas mencionadas arriba. Dicha funcionalidad se enriquecerá, en la medida de lo posible, tomando en cuenta la consulta realizada a los usuarios potenciales del sistema.
- La aplicación está orientada a la simulación, y no se trata de un sistema SCADA[3] de supervisión y control.
- El sistema soportará solamente los objetos de simulación discretos, es decir, sensores y actuadores cuyos valores pueden tener solo dos alternativas: activado y desactivado.

[3] SCADA Control supervisorio y adquisición de datos (*Supervisory Control and Data Acquisition*).

VII. VALIDACIÓN DE LOS RESULTADOS

La validación de resultados se llevará a cabo por medio de una demostración de las capacidades del sistema, en la que se simularán algunas plantas y procesos industriales y se controlarán por medio de los PLC's con los que se cuenta en el laboratorio de instrumentación y control, comprobando la correcta interacción entre el sistema y el PLC.

Se crearán grupos de prueba durante las últimas etapas del trabajo de graduación, con estudiantes interesados que tengan los conocimientos básicos de PLC que estén cursando o que ya hayan cursado la materia de Autómatas Programables, dejando constancia en video de las sesiones de prueba.

VIII. MARCO TEÓRICO

VIII.1. COMUNICACIONES INDUSTRIALES[4]

La estandarización de protocolos en la industria es un aspecto en constante discusión, donde intervienen problemas de carácter técnico y comercial. Los protocolos de comunicaciones digitales en la industria siguen, en general, el modelo estándar de interconexión de sistemas abiertos OSI. Sobre esta base y las recomendaciones de ISA (International Society for Measurement and Control) y la IEC (International Electrotechnic Committees) se ha establecido normas al respecto, en particular la IEC 1158 en desarrollo aun.

Cada protocolo está optimizado para diferentes niveles de automatización y en consecuencia responden al interés de diferentes proveedores de equipos; por ejemplo: Fieldbus Foundation, Profibus y Hart están diseñados para instrumentación de control de procesos. DevicetNet y SCD están optimizados para los dispositivos discretos (on-off) de detectores, actuadores e interruptores, donde el tiempo de respuesta y los procesos repetitivos son factores críticos.

Cada protocolo tiene un rango de aplicación, fuera del mismo disminuye el rendimiento y aumenta la relación coste/beneficio. En muchos casos no se trata de protocolos que compitan entre sí, sino que se complementan, cuando se trata de una arquitectura de un sistema de comunicación de varios niveles

[4] Tomado de [4]

VIII.2. ESTÁNDAR OPC[5]

VIII.2.1. Definición

OPC corresponde a un conjunto de especificaciones basadas en los estándares de Microsoft (COM[6], DCOM[7], OLE Automation y Active X[8]) que cubren los requerimientos de comunicación industrial entre aplicaciones y dispositivos, especialmente en lo que se refiere al tiempo real.

VIII.2.2. Generalidades

La reciente tecnología OPC (OLE for Process Control) es la nueva y exitosa apuesta de interconexión de dispositivos industriales ofreciendo una gran facilidad, seguridad, robustez y alto rendimiento. Cada vez son más los proyectos que se emprenden utilizando este nuevo paradigma de diseño por sus grandes ventajas y exitosos resultados.

OPC define un nuevo estándar de intercambio de información y las reglas de negociación entre dispositivos de distintos tipos. Así cualquier dispositivo que posea un software de control OPC podrá conectarse con cualquier software cliente OPC consiguiendo de esta manera una gran flexibilidad y conectividad, y la capacidad de agregar diferentes dispositivos a un software de control y adquisición de datos sin tener que modificar el mismo.

Por esto para la realización de este trabajo se utilizara OPC ya que permite la conexión de buena variedad de dispositivos para el caso particular PLC.

Tradicionalmente, cada diseñador de software necesitaba construir su propia interfaz o dispositivo “driver” para poder intercambiar información entre sus

[5] Tomado de [1]

[6] Es el modelo de objetos basados en componentes de Microsoft (*Component Object Model*), una tecnología para construir aplicaciones a partir componentes binarias de software.

[7] Modelo distribuido de objetos y componentes (DCOM), es un protocolo y método remoto de invocación de componentes COM.

[8] Componente de software que se puede insertar en una página Web para ofrecer una funcionalidad que no está disponible en HTML. Los controles ActiveX se pueden implementar en diferentes lenguajes de programación y deben descargarse al disco duro del computador para que los documentos que los utilizan puedan visualizarse.

dispositivos hardware. OPC elimina esta restricción definiendo una interfaz común y de alto rendimiento que permite que el trabajo sea hecho una sola vez y reutilizado en cualquier aplicación de control y monitoreo.

El trabajo del estándar OPC actualmente está avalado por más de las 200 compañías y asociaciones más importantes del sector como por ejemplo Microsoft, CERN, Compaq o National Instruments, lo que garantiza un soporte constante y continuas revisiones del estándar.

El OPC nace en 1996 cuando el grupo antecesor de la Fundación OPC, formado por las empresas Fisher-Rosemount, Rockwell Software, Opto 22, Intellution e Intuitive Technology, fue capaz de desarrollar una especificación básica en un solo año de trabajo. Esta especificación estaba basada en las tecnologías OLE/COM y DCOM, y fue nombrado como “Especificación OPC”.

El OPC fue diseñado para hacer puente entre aplicaciones basadas en Windows y aplicaciones de hardware y software para el control de procesos. Es un estándar abierto que permite un método consistente de acceso a datos de campo desde los dispositivos de la planta.

Fundación OPC

Buscando la opinión en la industria, se tomó la decisión de que la especificación OPC debía ser manejada por una organización independiente, no lucrativa llamada la Fundación OPC.

La Fundación OPC se presentó en ISA Show en el año 1996 en Chicago, con demostraciones de servidor OPC de varias empresas en la cabina Microsoft, y realizando la primera reunión de Asamblea general de miembros. Posteriormente demostraciones de colaboración se han mostrado en ferias de muestras principales en el mundo entero.

Productos comerciales que usan OPC comenzaron a aparecer a finales de 1996. A mediados de 1998, se confirma como el estándar de industria.

VIII.2.3. Estándares OPC

Actualmente son 7 los estándares desarrollados por La Fundación OPC. Estos siete estándares se nombran a continuación:

- OPC Data Access

Usado para mover datos en tiempo real desde PLC's, DCS's, y otros dispositivos de control hacia HMI's y otros clientes de visualización. La última versión de la especificación es la 3.0, la cual se apoya en versiones anteriores mientras mejora sus capacidades de búsqueda e incorpora el esquema *XML*-Data Access.

- OPC Alarms & Events

Provee notificaciones demandadas de alarmas y eventos. Esto incluye alarmas de procesos, acciones de operador, mensajes de información y mensajes auditivos.

- OPC Batch

Esta especificación lleva la filosofía de OPC hacia las necesidades específicas de los procesos de producción. Provee interfaces para el intercambio de capacidades de equipo y condiciones de operación actual.

- OPC Data Exchange

Esta especificación comunica a un servidor con otro servidor a través de redes Ethernet/*Fieldbus*. Esto provee interoperabilidad entre muchos vendedores, agrega configuraciones remotas y servicios de diagnósticos, monitoreo y administración.

- OPC Historical Data Access

Donde el OPC Data Access provee acceso en tiempo real, el OPC Historical Data Access provee acceso a datos previamente almacenados. Desde un simple sistema serial para el registro de datos hasta un sistema SCADA complejo, los archivos históricos pueden ser recuperados de una manera uniforme.

- OPC Security

Todos los servidores OPC proveen información que es valiosa para la empresa y si no es actualizada correctamente, podría tener consecuencias significativas para los procesos de planta. El OPC Security especifica cómo controlar el acceso de los clientes a estos servidores para proteger la información importante y cómo protegerse contra las modificaciones no autorizadas de los parámetros del proceso.

- OPC XML-DA

Provee reglas flexibles y consistentes, y formatos para exponer datos de la planta usando XML[9], apoyándose en el trabajo hecho por Microsoft y otros sobre SOAP y los servicios Web.

La Fundación OPC está trabajando actualmente en dos estándares más llamados OPC Complex Data y OPC Commands.

- OPC Complex Data

Una especificación compañera del OPC DA y OPC XML-DA que permite a los servidores exponer y describir tipos de datos complicados, así como estructuras binarias y documentos XML.

- OPC Commands

Un nuevo conjunto de interfaces, para servidores y clientes OPC, que definen comandos de identificación, envío y monitoreo, los cuales se ejecutan sobre un dispositivo.

[9] Extensible Markup Language, (Lenguaje de Marcas Ampliable) es una forma flexible de crear formatos de información y compartir tanto el formato como los datos en la World Wide Web, intranets y otras redes.

VIII.2.4. Aplicaciones del estándar OPC

OPC es un mecanismo uniforme para comunicar a numerosas fuentes de datos, o dispositivos en el proceso de la fábrica, o en una base de datos en un armario de control, algunas aplicaciones se mencionan a continuación.

- Administración de Campo. Con la llegada de dispositivos inteligentes, se puede proporcionar una variada información sobre dispositivos de campo que no estaban previamente disponibles. Esta información proporciona los datos de un dispositivo, sus parámetros de configuración, etc. Toda esta información debe ser presentada al usuario, y a cualquier aplicación que la use.
- Proceso de Administración. La instalación de Sistemas Distribuidos de Control (DCS) y sistemas SCADA para supervisar y controlar datos del proceso de fabricación disponibles electrónicamente que habían sido recopilados manualmente.
- Administración del negocio. Los beneficios pueden ser obtenidos instalando sistemas del control. Esto se consigue integrando la información recogida del proceso en los sistemas de negocio que maneja aspectos financieros de la fabricación. Proporcionar esta información eficazmente a aplicaciones de cliente aminora el esfuerzo requerido para proporcionar esta integración.

Para conseguir esto, los fabricantes necesitan conseguir acceso a los datos del piso de la planta e integrarlos en su sistema de negocio. Los fabricantes deben ser capaces de utilizar las herramientas (Paquetes de SCADA, las Bases de datos, etc.) para satisfacer sus necesidades. La clave es una arquitectura abierta y eficaz de comunicación en el acceso de datos, y no los tipos de datos.

Es por ello que hoy en día los sistemas SCADA son comunes en la industria donde los desarrolladores de este tipo de sistemas se enfocan principalmente en los siguientes aspectos: integración con el hardware, facilidad de uso, aplicación de herramientas innovadoras y apertura de comunicaciones.

VIII.2.5. Arquitectura OPC

Lo que se necesita fundamentalmente para las aplicaciones es una manera común de acceder a los datos de cualquier fuente, como un dispositivo de campo o una base de datos.

El servidor OPC en esta figura 1 y en secciones posteriores es sinónimo de cualquier servidor que proporciona una interfaz OPC.

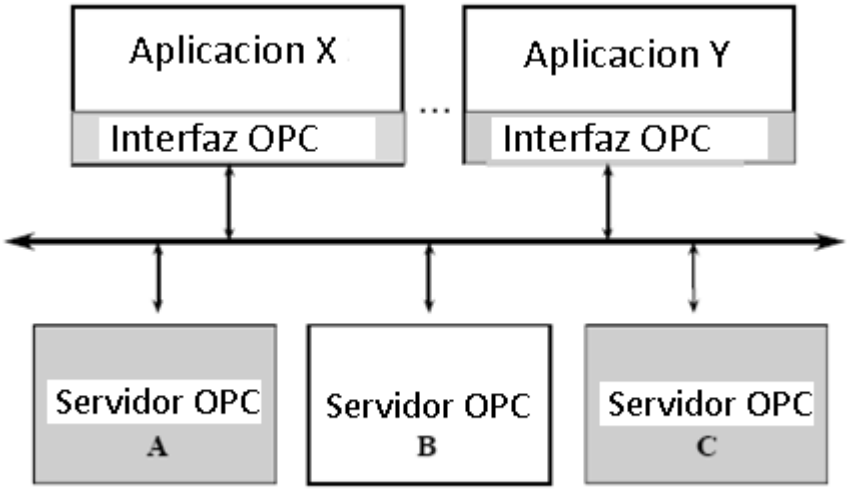

Figura 1 Conectividad de OPC

A un nivel alto, un Servidor de Datos OPC está compuesto por varios elementos: el servidor, el grupo y el ítem. El servidor OPC mantienes información sobre el servidor y sirve como contenedor para objetos del grupo OPC. El objeto del grupo OPC mantiene información sobre sí mismo y provee de mecanismos para contener y organizar lógicamente ítem de OPC.

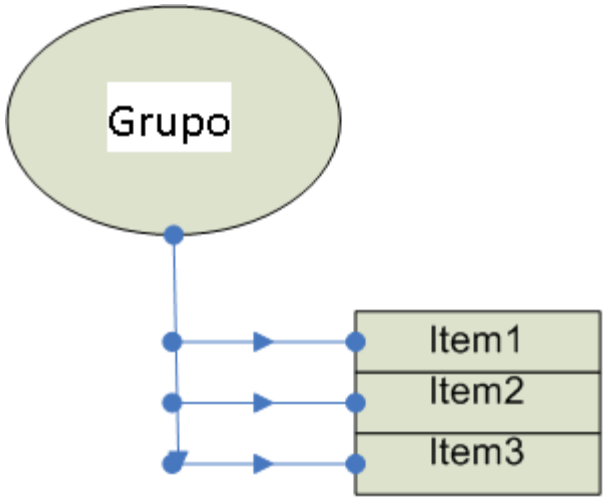

Figura 2 Acceso al Ítem en OPC

Los ítems OPC representan conexiones a las fuentes de datos dentro de un servidor.

Desde el punto de vista de una interfaz, un Ítem OPC, no es accesible como un objeto por parte de un Cliente OPC. Por lo tanto, no hay ninguna interfaz externa definida para un Ítem OPC. Todos los accesos a Ítems OPC se hacen a través de un objeto del Grupo OPC que contiene al Ítem OPC.

A cada Ítem se le asocian sellos de Valor, Calidad y Tiempo. El Valor se expresa en la forma VARIANT y la Calidad es parecida a lo especificado por Fieldbus[10].

Es importante entender que los ítems no son fuentes de datos, son únicamente conexiones a ellas. Por ejemplo, los ítems en un sistema DCS existes independientemente de si en ese momento hay un Cliente OPC accediendo a ellos.

[10] Las **redes de comunicaciones industriales** deben su origen a la fundación FieldBus (Redes de campo). La fundación FieldBus, desarrolló un nuevo protocolo de comunicación, para la medición y control de procesos donde todos los instrumentos puedan comunicarse en una misma plataforma.

Arquitectura OPC general e interfases.

Las especificaciones OPC contienen siempre dos tipos de interfaces; Interfaces Custom e interfaces de Automatización como aparece en la Figura 3.

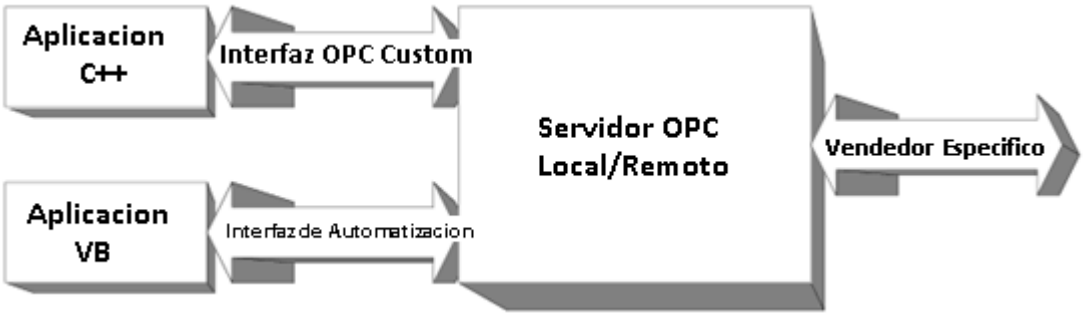

Figura 3 Interfaces OPC.

Hay varias consideraciones cuando se implementa un servidor OPC. El más importante es la frecuencia de transferencia de datos de paths de comunicación que no se pueden compartir a aparatos físicos. Por lo tanto se espera que el Servidor OPC sea un EXE local o remoto que incluya código que se encargue de recoger datos de aparatos físicos o de bases de datos.

Una aplicación Cliente OPC se comunica con un Servidor OPC mediante las interfaces Custom y Automation previamente mencionadas. Los servidores OPC han de implementar la interfaz de Custom y opcionalmente pueden implementar la de Automation.

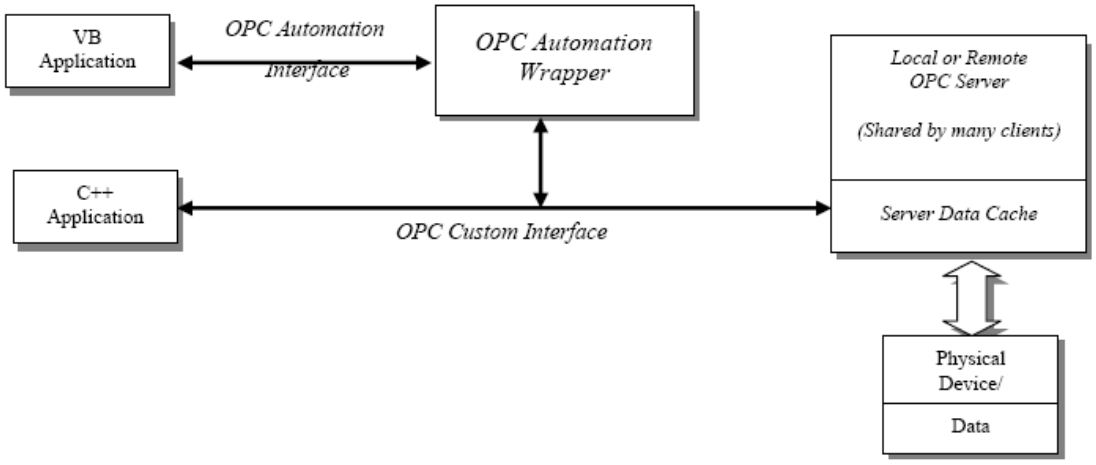

Figura 4 Comunicación entre aplicaciones

VIII.2.6. Método de aplicación del estándar OPC

Un número creciente de aplicaciones se desarrolla en ambientes como Visual Basic (VB), Delphy, Power Builder, LabVIEW etc. OPC debe tener en cuenta esta tendencia. Así, Microsoft diseña OLE/COM para permitir que componentes escritos en C y C + +, sean utilizados por programa de cliente (escritos en VB o Delphy para un dominio totalmente diferente). Los componentes software se escribirán en C y C ++ para encapsular la complejidad de acceso a datos de un dispositivo, de forma que permita a los promotores de aplicación de gestión escribir en VB y tener acceso a los datos de piso de planta. El objetivo de todas las especificaciones es el desarrollo de servidores OPC en C y C++, y así, facilitar el desarrollo de aplicaciones de cliente de OPC en el lenguaje escogido. La arquitectura y el diseño de los interfaces son pensados para apoyar el desarrollo de servidores OPC también en otros lenguajes.

OPC está diseñado para permitir a las aplicaciones de cliente el acceso a los datos de planta de una manera consistente. OPC proporcionará muchos beneficios:

- Los fabricantes de hardware sólo tienen que hacer un conjunto de componentes de programa para que los clientes los utilicen en sus aplicaciones.
- Los fabricantes de software no tienen que adaptar los drivers ante cambios de hardware.
- Con OPC, la integración de sistemas en un entorno heterogéneo se convertirá simple.

VIII.2.7. OPC Data Access (OPC DA)

En un servidor OPC Data Access se pueden diferenciar tres tipos de objetos:

- Objeto OPC Server.
- Objeto OPC Group
- Objeto OPC Item

El objeto OPC Server es el objeto COM al cual se conectan las aplicaciones. Debajo de él se encuentran una colección de objetos OPC Group. Estos objetos son creados por la aplicación de forma dinámica para mantener listas de tags[11] y atributos que en OPC son llamados ítems. Por ejemplo un interfaz HMI debería crear un grupo por cada imagen que tenga abierta. El contenido de los grupos e ítems puede variar en el tiempo dependiendo de las necesidades de las aplicaciones.

Topología de OPC DA

A continuación se muestra un diagrama en la que se representa la topología de los objetos donde se puede apreciar cada objeto y la relación que mantienen entre ellos.

[11] Ítems en OPC

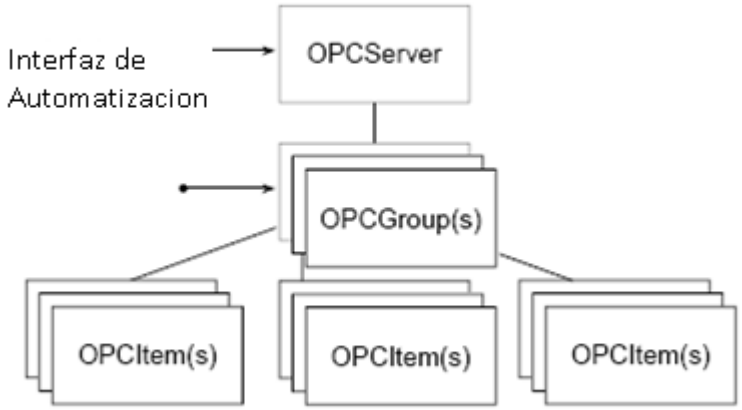

Figura 5. Relación entre objetos.

- Objeto OPC Server

El servidor OPC DA establece un objeto del tipo OPC Server para cada cliente y crea un canal de comunicación para cliente por separado. De esta forma se evita que el flujo de información entre cliente y servidor se vea disminuido a causa de otros clientes. El objeto OPC Server mantiene información del actual servidor y se usa como contenedor de objetos OPC Group.

- Objeto OPC Group

El objeto OPC Group tiene como finalidad el proporcionar a las aplicaciones un mecanismo para organizar los datos que necesitan. Diferentes grupos pueden ser usados por diferentes partes de la aplicación. El objeto OPC Group además de mantener información sobre sí mismo se encarga del mantenimiento y la organización lógica de los objetos OPC Ítem.

La transmisión de datos del servidor OPC DA actúa a nivel del OPC Group. Cada Grupo tiene que tener un nombre único relacionado con el cliente OPC. El cliente OPC puede cambiar más tarde este nombre, pudiendo especificar incluso el activo o inactivo del OPC Group

El servidor OPC DA es el encargado de generar grupos OPC. Además de crear grupos es posible suprimir un grupo, obtener el nombre de un grupo y enumerar los grupos. Los grupos contienen ítems, que corresponden a datos en el servidor.

Si el servidor es un supervisor los ítems se corresponden con tags. Si el servidor es un autómata los ítems se corresponden con registros (DB)

Los servidores OPC pueden leer sus datos de bases de datos o desde cualquier fuente de datos.

- OPC Ítem

El objeto OPC Ítem representa un punto de conexión entre el servidor y el valor real en el dispositivo físico. Esto significa que los Ítems OPC no representan las fuentes actuales de datos pero solo contienen la dirección al tag apropiado en la configuración del servidor. Se trata de un objeto transitorio que existe junto con el servidor OPC

Al contrario que el OPC Group y el OPC Server el Ítem OPC no soporta interfaces OPC y por eso no es un objeto COM. Es un objeto interno del servidor OPC que mantiene información importante de los necesidades solicitadas por del cliente OPC (por ejemplo los datos usados para actualizar valores, el estado activo o inactivo de los valores requeridos, etc.)

Desde la perspectiva del cliente OPC el Ítem OPC no representa la fuente de datos actual sino únicamente la conexión lógica a la fuente de datos. Usando el identificador del ítem (Ítem ID), el OPC Ítem está asociado unívocamente al ítem definido en la configuración física del servidor OPC.

Los Ítems se identifican por su nombre. El servidor debe utilizar una técnica para controlar la correspondencia de los nombres de los Ítems y los datos fuente que él maneja. Estos nombres son accesibles a través de la interfaz “Browse”.

Los ítems se encuentran dentro de los objetos OPC Group y se caracterizan por atributos y propiedades entre las cuales se puede resaltar:

- Nombre
- Valor
- Calidad
- Marca de tiempo

- Ítem ID

Ítem ID es un único identificador del tag y es usado por el cliente OPC para establecer la conexión con el servidor OPC.

VIII.3. LENGUAJE DE PROGRAMACIÓN

Una de las partes importantes de este proyecto será el software empleado desarrollar la aplicación. En este sentido se comenzará estableciendo un panorama general acerca de los lenguajes de programación.

VIII.3.1. Generalidades

Los lenguajes de programación son lenguajes especiales que ayudan al usuario a comunicarse con la computadora. Establecen una comunicación entre el humano que prefiere usar palabras (el sistema decimal) y la computadora que solo trabajan con valores discretos (0's y 1's).

Los lenguajes de programación pueden clasificarse como:

- Lenguaje de Máquina

Representa el más bajo nivel de la programación. Este lenguaje es en esencia números y códigos que solamente los sistemas a base de microprocesador comprenden. Este lenguaje es fácil de entender por la computadora, pero difícil para el usuario. El punto a su favor es que un programa escrito en lenguaje de maquina puede ejecutarse más rápido que los programas escritos en otros lenguajes y sus dimensiones son menores. Es el lenguaje original de la computadora el cual es generado por el "software" y no por el programador.

- Lenguaje de bajo nivel

Es un lenguaje de programación bien cercano al lenguaje de máquina. Es difícil de entender por las personas y requiere que los programadores codifiquen las instrucciones con muchos detalles. Ejemplo: lenguaje ensamblador.

- Lenguaje de alto nivel

Es un lenguaje que se asemeja más al lenguaje humano que a un lenguaje de maquina o ensamblador. Es más fácil escribir programas en este lenguaje, pero luego deben ser traducidos por compiladores o intérpretes para que la computadora los entienda.

En el caso de este proyecto se ha elegido como plataforma de desarrollo el software LabVIEW por las ventajas que posee en manejo de hardware e intercambio de datos con otras aplicaciones.

VIII.3.2. Lenguajes de alto nivel.

Los lenguajes de alto nivel más comunes son:

- BASIC (Beginners All-purpose Symbolic Instruction Code), fue el lenguaje de programación interactivo mas popular de la década de los 70. Es un lenguaje de propósito general. Desarrollado por John Kemeny y Thomas Kurtz en Dartmouth Collage en 1963. Existen numerosas versiones, algunas son compiladas y otras son interpretadas.
- COBOL (Common Business Oriented Language). Es un compilador diseñado para las aplicaciones de negocios. Desarrollado en 1959 por el gobierno federal de los Estados Unidos y fabricantes de computadoras bajo el liderazgo de Grace Hopper. Es el más utilizado por los “mainframe”.
- PASCAL**.** Este programa recibió su nombre en honor de Blaise Pascal. Fue desarrollado por el científico suizo Niklaus Wirth en 1970 y diseñado para enseñar técnicas de programación estructurada.
- FORTRAN (FORmula TRANslator). Es uno de los primeros lenguajes de alto nivel desarrollado en 1954 por John Backus y un grupo de programadores de IBM. Es un lenguaje compilador que se diseñó para expresar con facilidad las fórmulas matemáticas, resolver problemas científicos y de ingeniería.
- Lenguaje C. Fue desarrollado a principios de la década de los 70 en los laboratorios Bell por Brian Kernigham y Dennos Ritchie. Ellos necesitaban desarrollar un lenguaje que se pudiera integrar con UNIX permitiendo a los usuarios hacer modificaciones y mejorías fácilmente. Fue derivado de otro lenguaje llamado BCPL.
- Lenguaje C++. Fue desarrollado por Bjarne Stroustrup en los laboratorios Bell a principios de la década de los 80. C++ introduce la programación orientada a objeto en C. es un lenguaje extremadamente poderoso y eficiente.

VIII.3.3. Programación orientada a objetos.

Los objetivos de la programación orientada a objetos son mejorar la productividad de los programadores haciendo más fácil de reutilizar y de extender los programas y manejar sus complejidades. De esta forma, se reduce el costo de desarrollo y mantenimiento de los programas.

Para poder comprender la POO[12] se explican seis conceptos básicos:

- Objetos

Un objeto es una representación en computadora de alguna cosa o evento real. La figura 6 muestra cómo una computadora podría representar un auto. Por ejemplo, si se trata de un Toyota Yaris, la computadora podría guardar el nombre del modelo, el numero de ID del vehículo (#61Y61BG826341Y) y el tipo de motor (Cil). Los objetos pueden tener tanto atributos (tales como el modelo, VIN y tipo de motor) y comportamientos (tales como el modelo, VIN y tipo de motor) y comportamientos (tales como "encender la luz" y "apagar la luz").

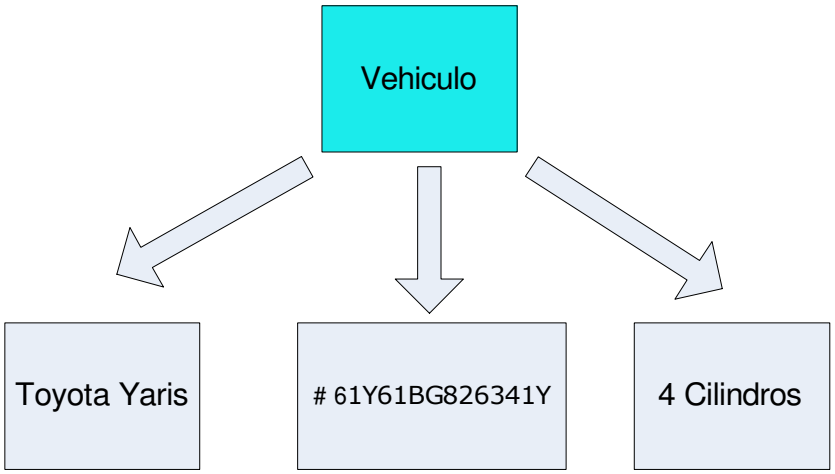

Figura 6 Atributos de objeto dibujo.

- Clases

Una clase es una categoría de objetos similares. Los objetos se agrupan en clases. La figura 7 muestra cómo un grupo de objetos que representan automóviles podrían estar formados en una clase llamada "Automóviles". Una clase define el conjunto de atributos y comportamientos compartidos que se encuentran en cada objeto de la clase. Por ejemplo, todo automóvil tendrá atributos para fabricante/modelo, VIN, y motor. El programador debe definir las clases en el programa. Cuando el programa ejecuta se pueden crear objetos a partir de las clases establecidas. Se usa el término "ocurrencia" cuando se crea un

[12] POO: Programación Orientada a Objetos

objeto de una clase. Por ejemplo, un programa podría tener una ocurrencia Toyota Yaris como un objeto de la clase automóviles.

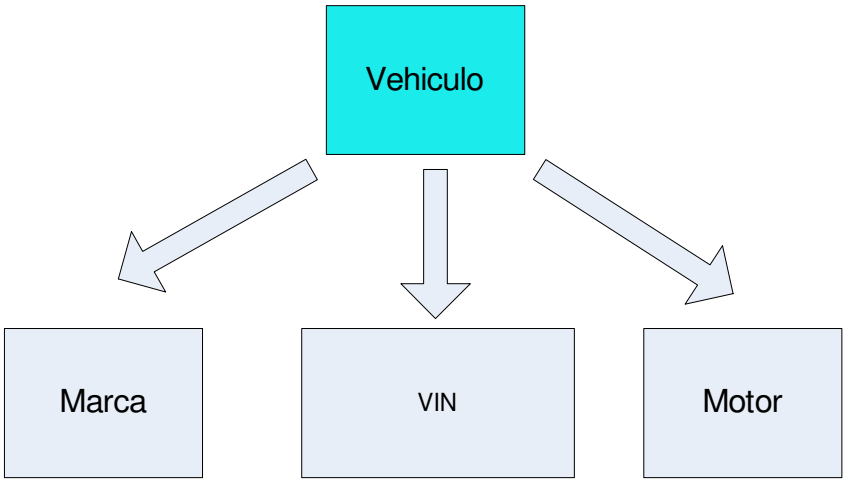

Figura 7 Vehículo objeto de la clase automóvil.

- Mensajes

Se puede enviar información de un objeto a otro. En la figura 8 un objeto (Juan Pérez) de la clase "operador" está enviando un mensaje a un objeto (Toyota Yaris) de la clase "automóviles" .El mensaje es "arrancar motor". Estos mensajes no son de forma libre en ningún sentido, ya que en vez de eso las clases de operador y automóviles han sido programadas cuidadosamente para enviar y recibir un mensaje de "arrancar motor". La clase operador ha sido programada para transmitir un mensaje arrancar motor bajo determinadas circunstancias. La clase automóviles ha sido programada para reaccionar a el mensaje "arrancar motor" en alguna forma.

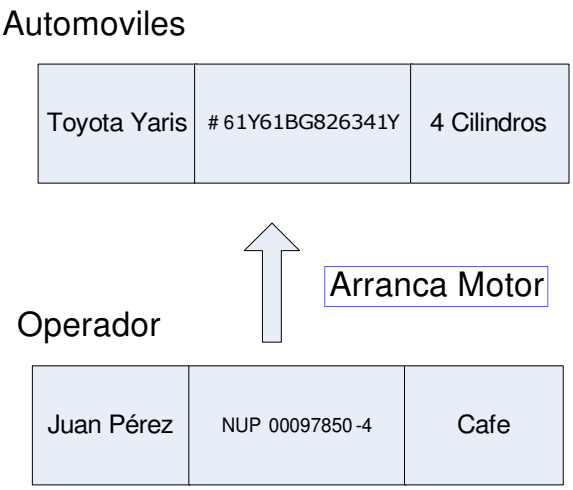

Figura 8. Mensajes entre objetos.

- Encapsulación

Típicamente la información acerca de un objeto está encapsulada por su comportamiento. Esto significa que un objeto mantiene datos acerca de cosas del mundo real a las que representa en un sentido verdadero. Típicamente, a un objeto se le debe "pedir" o "decir" que cambie sus propios datos con un mensaje, en vez de esperar que tales datos de procesos externos cambien la naturaleza de un objeto. En la figura 9 el objeto José Hernandez (clase mecánico) envía un mensaje quitar motor a el objeto Toyota (clase automóviles). La clase automóviles reacciona a este mensaje con un comportamiento (también llamado "método" o "procedimiento") que cambia el atributo de motor a ninguno. Este método es llamado quitar motor. Vemos que el objeto Toyota reacciona al mensaje cambiado uno de los atributos a ninguno. Puede parecer trivial el que un atributo de un objeto sea cambiado alterando directamente sus datos o enviando un mensaje a el objeto para que active el comportamiento interno que cambia ese dato. Pero esta diferencia es una característica extremadamente importante de los programas Orientados a Objetos. Los datos encapsulados pueden ser protegidos en forma tal que solamente el objeto mismo puede hacer tales cambios por medio de su propio comportamiento. Esta construcción facilita la construcción de objetos, que son muy confiables y consistentes debido a que ellos tienen control completo sobre sus propios atributos. También hace que sea mucho más fácil el mantenimiento y cambio de programas. Por ejemplo, la clase mecánico está aislada completamente de los detalles internos de la clase automóviles. La clase automóviles puede ser reprogramada totalmente si cambiar nada de la clase mecánica, siempre y cuando la clase automóviles continué recibiendo adecuadamente el mensaje quitar motor. Este aislamiento hace que sea mucho más fácil cambiar un parte del programa sin causar problemas en cascada a otra parte del programa.

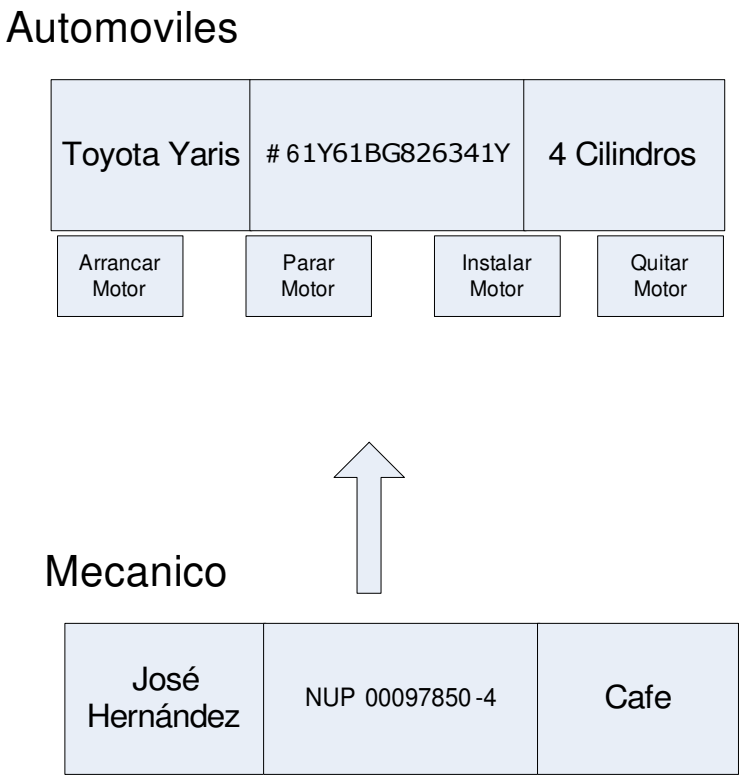

Figura 9. Atributos

- Herencia

Las clases pueden tener "hijos", quiere decir que una clase puede ser creada a partir de otra clase. La clase original, o madre, es llamada "clase base". La clase hija es llamada "clase derivada". Una clase derivada puede ser creada en forma tal que herede todos los atributos y comportamientos de la clase base. En la figura se crea una clase derivada (camión) en forma tal que hereda todos los atributos de la clase base automóviles. Una clase derivada puede tener también atributos y comportamientos adicionales. Por ejemplo, la clase camión no solo tiene atributos para fabricante/modelo, VIN y motor, sino también tiene atributos para cargar remolques y refrigeración. Los objetos automóviles no tienen estos nuevos atributos. La herencia reduce la labor de programación reutilizando fácilmente objetos antiguos. El programador solo necesita declarar que la clase camión hereda de la clase automóviles, y luego proporcionar cualquier detalle adicional acerca de nuevos atributos o comportamientos (mostrados en el cuadro de línea de la figura).Todos los atributos y comportamientos antiguos de la clase

automóviles son parte automática e implícitamente de la clase camión (se muestra en el cuadro de guiones) y no requiere ninguna nueva programación. Algunos lenguajes de programación Orientados a Objetos proporcionan herencia múltiple. En estos casos especiales se puede crear una clase derivada en forma tal que herede todos los comportamientos y los atributos de más de una clase base. Por ejemplo, si hubiera una clase llamada automóviles y una clase llamada bicicletas, podríamos crear una clase derivada llamada motocicletas que herede todos los atributos y comportamientos de automóviles y que herede todos los atributos y comportamientos de bicicletas.

- Polimorfismo

El termino polimorfismo se refiere a comportamientos alternos entre clases derivadas relacionadas. Cuando varias clases heredan atributos y comportamientos, puede haber casos donde el comportamiento de una clase derivada deba ser diferente del de su clase base o de sus clases derivadas parientes. Esto significa que un mensaje puede tener diferentes efectos, dependiendo de exactamente qué clase de objeto recibe el mensaje.

VIII.3.4. Programación orientada a eventos.

Los lenguajes visuales orientados al evento y con manejo de componentes dan al usuario que no cuenta con mucha experiencia en desarrollo, la posibilidad de construir sus propias aplicaciones utilizando interfaces gráficas sobre la base de ocurrencia de eventos.

Para soportar este tipo de desarrollo interactúan dos tipos de herramientas, una que permite realizar diseños gráficos y, un lenguaje de alto nivel que permite codificar los eventos. Con dichas herramientas es posible desarrollar cualquier tipo de aplicaciones basadas en el entorno.

VIII.3.5. LabVIEW.

Generalidades

LabVIEW es una herramienta gráfica de test, control y diseño mediante la programación. El lenguaje que usa se llama lenguaje G.

Este programa fue creado por National Instruments (1976) para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986. Ahora está disponible para las plataformas Windows, UNIX, MAC y Linux y va por la versión 8.20 (desde 2006).

Los programas hechos con LabVIEW se llaman VI (*Virtual Instrument*), lo que da una idea de uno de sus principales usos: el control de instrumentos. El lema de LabVIEW es: *"La potencia está en el Software"*. Esto no significa que la empresa haga únicamente software, sino que busca combinar este software con todo tipo de hardware, tanto propio -tarjetas de adquisición de datos, PAC, Visión, y otro Hardware- como de terceras empresas.

Principales usos

Es usado principalmente por ingenieros y científicos para tareas como:

- Adquisición de datos
- Control de instrumentos
- Automatización industrial o PAC (Controlador de Automatización Programable)
- Diseño de control: prototipos rápidos y hardware-en-el-bucle (HIL)

Principales características

Su principal característica es la facilidad de uso, personas con pocos conocimientos en programación pueden hacer programas relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. También es muy rápido hacer programas con LabVIEW y cualquier programador, por experimentado que sea, puede beneficiarse de él. Para los amantes de lo complejo, con LabVIEW pueden crearse programas de miles de VIs (páginas de

código) para aplicaciones complejas, programas de automatizaciones de decenas de miles de puntos de entradas/salidas, etc. Incluso existen buenas prácticas de programación para optimizar el rendimiento y la calidad de la programación.

Presenta facilidades para el manejo de:

- Interfaces de comunicaciones:
    - Puerto serie
    - Puerto paralelo
    - GPIB
    - PXI
    - VXI
    - TCP/IP, UDP, DataSocket
    - Irda
    - Bluetooth
    - USB
    - OPC
- Capacidad de interactuar con otras aplicaciones:
    - ***DLL***
    - ActiveX
    - Matlab
    - Simulink
- Herramientas para el procesado digital de señales.
- Visualización y manejo de gráficas con datos dinámicos.
- Adquisición y tratamiento de imágenes.
- Control de movimiento.
- Tiempo Real estrictamente hablando.
- Programación de FPGAs.
- Sincronización.

VIII.3.6. Librerías de enlace dinámico DLL[13]

[13] Tomado de [3]

DLL es el acrónimo de *Dynamic Linking Library* (*Bibliotecas de Enlace Dinámico*), término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda del programa por parte del sistema operativo. Esta denominación se refiere a los sistemas operativos Windows siendo la extensión con la que se identifican los ficheros, aunque el concepto existe en prácticamente todos los sistemas operativos modernos.

Las DLL's *pueden* verse como la evolución de las bibliotecas estáticas y de forma análoga contienen funcionalidad o recursos que utilizan otras aplicaciones. Sin embargo, su uso proporciona algunas ventajas:

- **Reducen el tamaño de los archivos ejecutables:** Gran parte del código puede estar almacenado en bibliotecas y no en el propio ejecutable lo que redunda en una mejor optimización de código.
- **Pueden estar compartidas entre varias aplicaciones:** Si el código es suficientemente genérico, puede resultar de utilidad para múltiples aplicaciones (por ejemplo, la MFC es una biblioteca dinámica con clases genéricas que recubren la API gráfica de Windows y que usan gran parte de las aplicaciones).
- **Facilitan la gestión y aprovechamiento de la memoria del sistema:** La carga dinámica permite al sistema operativo aplicar algoritmos que mejoren el rendimiento del sistema cuando se carguen estas bibliotecas. Además, al estar compartidas, basta con mantener una copia en memoria para todos los programas que la utilicen.
- **Brindan mayor flexibilidad frente a cambios:** Es posible mejorar el rendimiento o solucionar pequeños errores distribuyendo únicamente una nueva versión de la biblioteca dinámica. Nuevamente, está corrección o mejora será aprovechada por todas las aplicaciones que compartan la biblioteca.

Las DLL es una técnica de implementación de forma dinámica es decir que la computadora solo la carga cuando el programa que la usa la llama.

VIII.4. COMUNICACIÓN DE LABVIEW CON PLC.

LabVIEW posee diversas maneras de comunicación con hardware de distinto tipo. Para la aplicación, LabVIEW se va a conectar a un PLC a través de un servidor OPC.

VIII.5. CONFIGURACIÓN DEL SERVIDOR OPC DELTALOGIC S7/S5.

Para configurar el servidor OPC DELTALOGIC se accede al menú inicio/DELTALOGIC/S7-OPC-SERVER/*Configure S7-OPC-Server* como lo indica la siguiente figura 10.

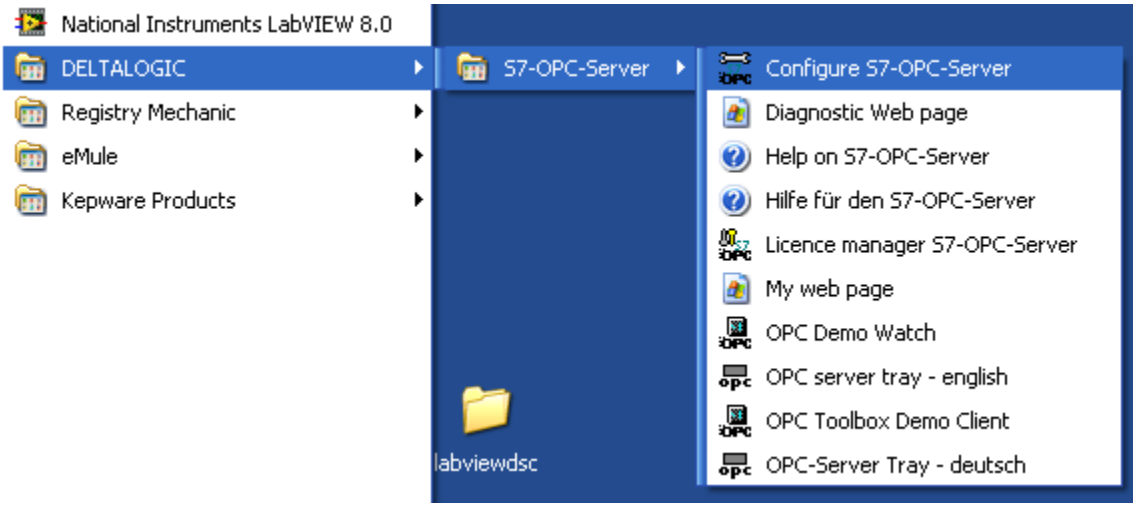

Figura 10. Acceso a OPC Server

Aparece el siguiente cuadro que contiene varias pestañas. A continuación se muestra la pestaña Conexiones en blanco.

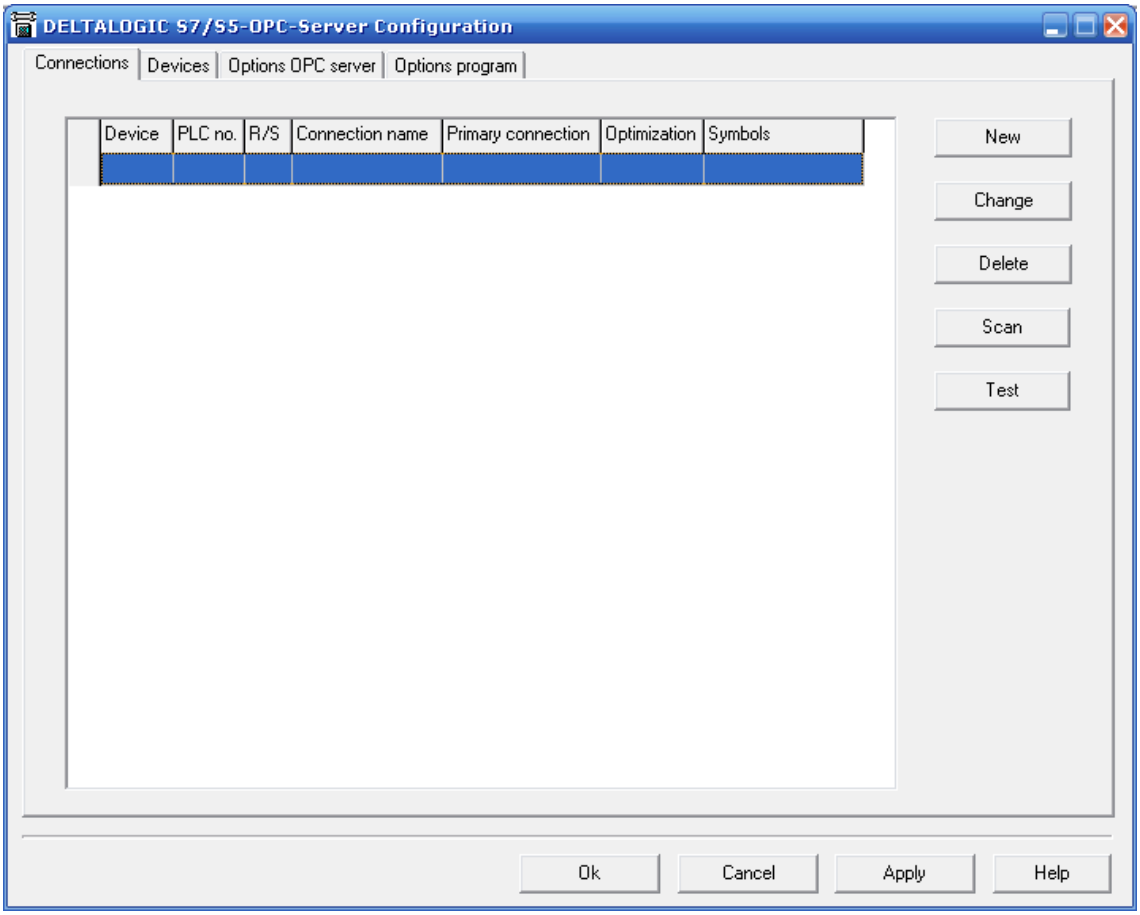

Figura 11. Pantalla de configuración.

Se procede a realizar la comunicación entre un PLC Siemens S7 200 CPU222.

En la pestaña *Devices* (Dispositivos), procedemos a configurar el PLC y el tipo de comunicación que va a tener con la computadora. Para el caso en cuestión, éste tiene una interfaz de comunicación PPI conectada al COM 1 de la computadora y una velocidad de transmisión de 9600 kbaudios.

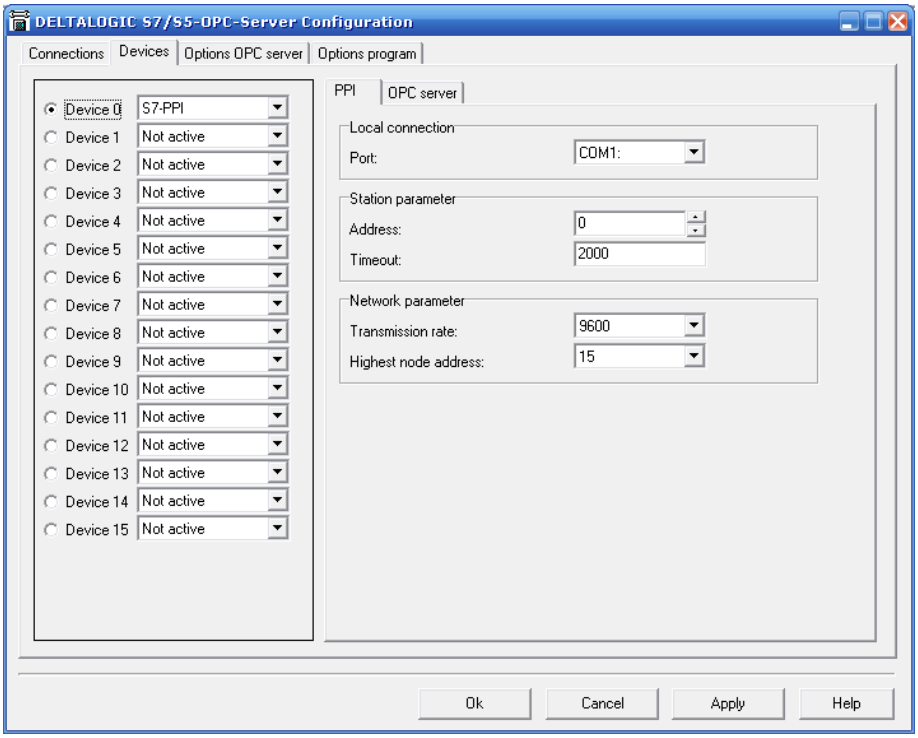

Figura 12 Configuración de la interfaz de comunicación física

Luego en la pestaña *Connections* (Conexiones) seleccionamos el botón *New* para agregar dicho PLC al servidor OPC. Introducimos un nombre a la conexión, para este caso es CPU222; y la ubicación de la conexión, para este caso *Device* 0 PLC 2

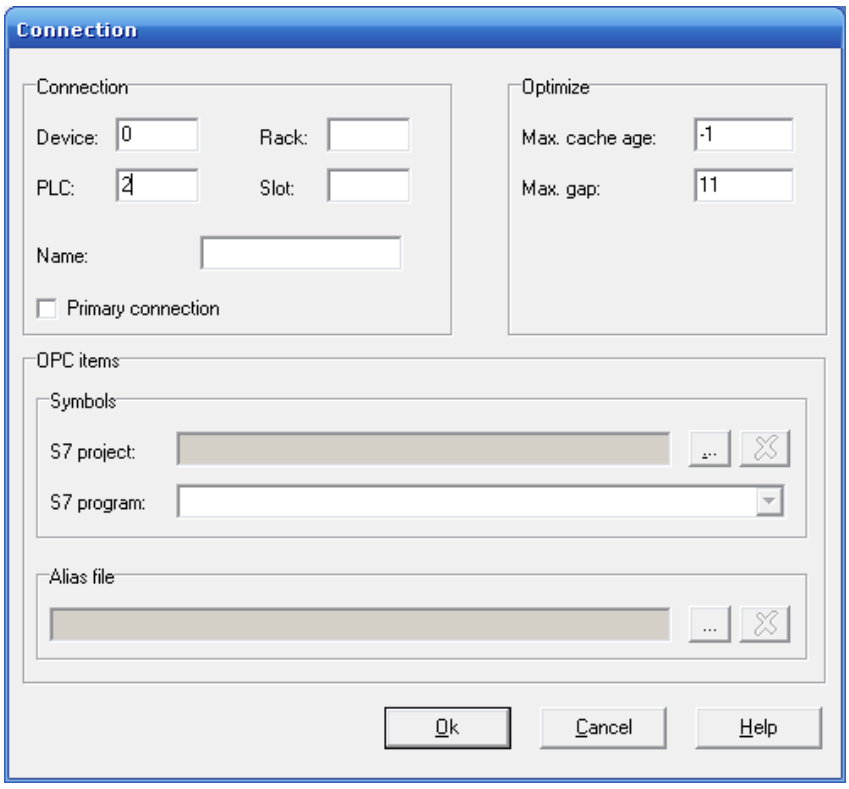

Figura 13 Ubicación del dispositivo

A continuación se realiza una prueba de la conexión. El programa muestra en este punto si la conexión ha sido o no exitosa; en un cuadro de diálogo muestra el dispositivo conectado, caso contrario muestra un código de error.

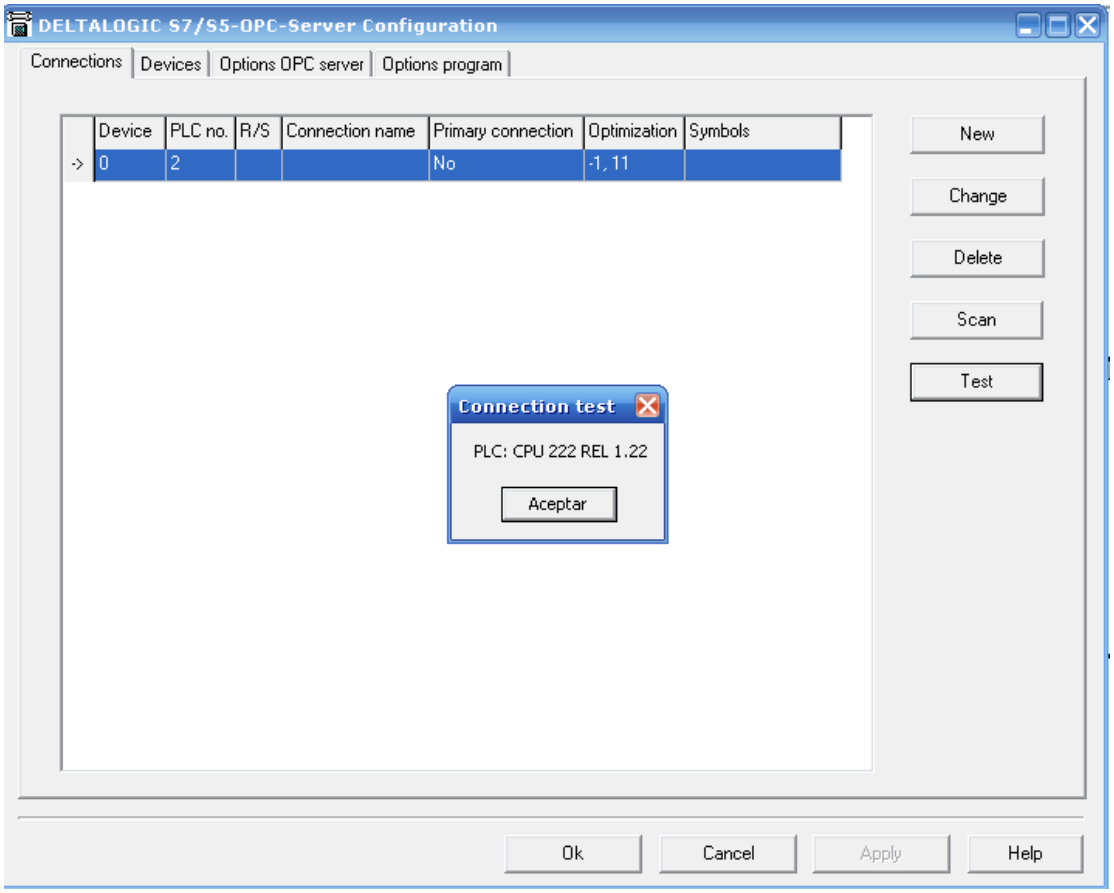

Figura 14 prueba de acceso al PLC

Hasta este punto, podemos decir que la comunicación entre el PLC y la computadora es exitosa; solo falta agregar los ítems a monitorear.

El siguiente paso es ejecutar el OPC Server seleccionando *OPC Server tray*.

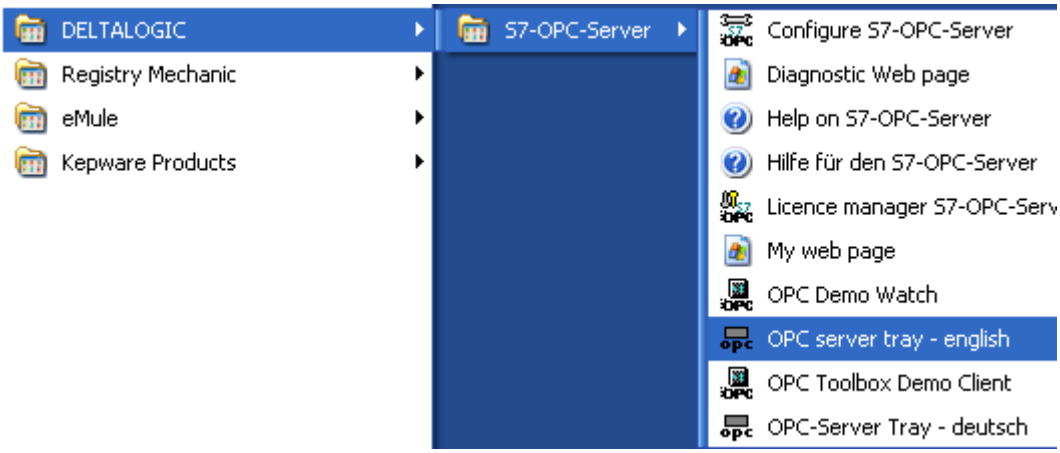

Figura 15 Inicialización de servidor OPC.

Aparece el cuadro mostrado en la figura 15 seleccionamos *Start OPC Server* para que la computadora comience a interactuar con el PLC. Al cerrar dicha ventana, el servidor queda corriendo en segundo plano. Se puede acceder a la configuración del mismo desde la barra de aplicaciones situada en la esquina inferior derecha como lo muestra la Figura 17.

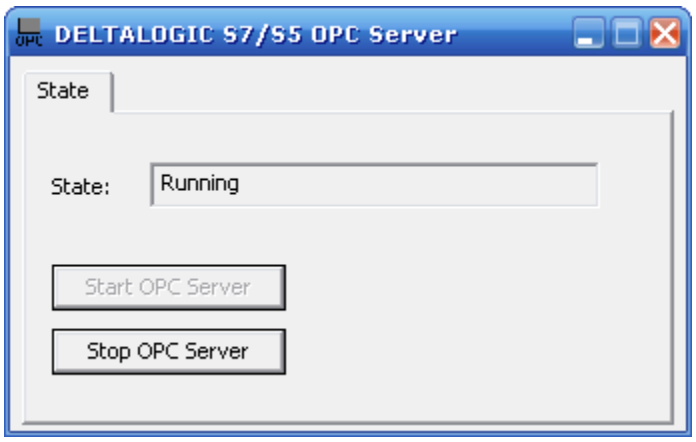

Figura 16 Estado de la conexión con el dispositivo

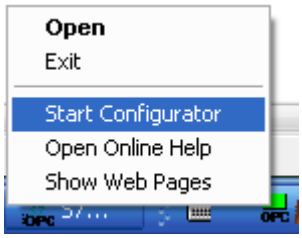

Figura 17 Arranque del visor de Ítems

La configuración final de los tags se hace desde una página Web que se muestra seleccionando *Show Web Pages*, con lo que despliega la pantalla que se muestra en la siguiente figura.

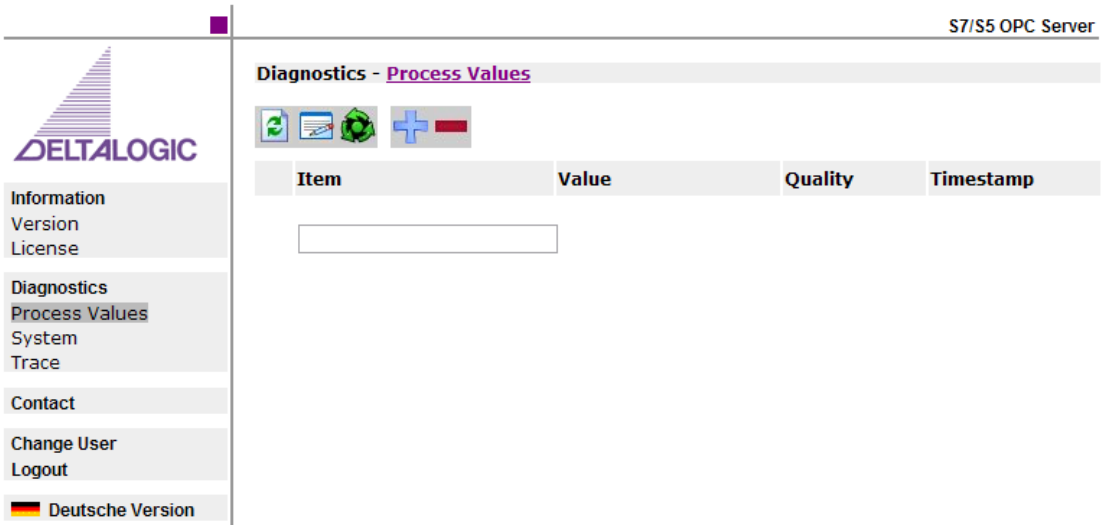

Figura 18 Agregar/quitar ítems

Esta página sirve para agregar/quitar los ítems. Además sirve para observar en tiempo real el comportamiento de los mismos.

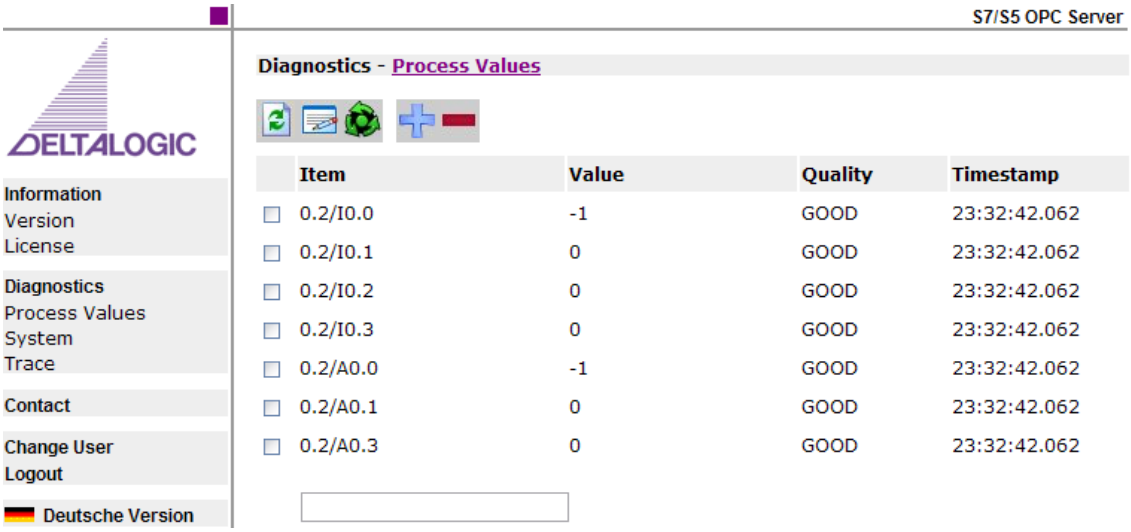

Figura 19. Visor de ítems en tiempo real.

En esta figura se puede observar los ítems ya configurados, interactuando con el PLC. Se han configurado cuatro entradas: I0.0, I0.1, I0.2, I0.3; tres salidas: A0.0, A0.1, A0.3. La página muestra los valores en tiempo real de las variables del PLC. Para este caso el 0 representa una entrada/salida apagada y -1 una entrada/ salida activada.

De esta manera queda configurado el servidor OPC, teniendo la capacidad de seguir agregando ítems, ya sean entradas/salidas físicas o marcas internas.

VIII.6. CONEXIÓN LABVIEW-OPC.

La forma de comunicación entre el PLC y LabVIEW es a través del estándar OPC, el mismo va a estar configurado con los tags respectivos. LabVIEW va a acceder a cada uno de los ítems por medio de DATASOCKET.

DATASOCKET va a brindar una conectividad entre LabVIEW y la variable compartida, en este caso va a ser el ítem configurado en el servidor OPC.

VIII.6.1. ¿Qué es Datasocket?[14]

Datasocket es una nueva tecnología de programación basada en el estándar industrial TCP/IP. Intercambia datos entre diferentes aplicaciones en una o entre varias computadoras interconectadas en una red.

Aunque una variedad de diversas tecnologías existe hoy para compartir los datos entre las aplicaciones, tales como TCP/IP y DDE[15], la mayor parte de estas herramientas no reúnen las condiciones para la transferencia de datos de forma dinámica. DataSocket pone en ejecución una interfaz de programación fácil de utilizar, de alto rendimiento diseñado para compartir y publicar datos de forma dinámica.

DataSocket consiste en dos partes:

- El DataSocket API; El DataSocket API presenta una sola interfaz para comunicarse con los tipos de datos múltiples de lenguajes de programación múltiples
- El servidor de DataSocket. Simplifica la comunicación de Internet manejando el TCP/IP.

DataSocket es un sencillo API basado en URLs para conectar con los datos localizados dondequiera, a nivel local dentro de una computadora o dondequiera en el Internet. Es un protocolo-independiente, y de sistema operativo

[14] Tomado de la ayuda de software Labview 8.0

[15] Dynamic Data Exchange (Networked DDE) [Intercambio Dinámico de Datos (DDE en red)]. Es un protocolo de software muy similar a un protocolo de PLC, diseñado para transferir bytes individuales de datos entre programas diferentes ya sea en la misma computadora (DDE), o bien a través de una red (NetDDE).

independiente. EL Datasocket API[16] se pone en ejecución como un control de ActiveX, así que puede ser utilizado en cualquier ambiente de programación.

El DataSocket API convierte automáticamente los datos de la medición del usuario en una serie de bytes que se envían a través de la red.

La aplicación cliente de Datasocket convierte esa serie de bytes en su forma original. Esta conversión automática elimina la complejidad de la red, que reduce una cantidad substancial de código que se debe escribir al usar bibliotecas del TCP/IP.

Básicamente los eventos que implica la utilización de Datasocket son cuatro.

- Abrir
- Leer
- Escribir
- Cerrar

En el ambiente real de LabVIEW, se cuenta con una librería Datasocket que tiene los siguientes componentes.

[16] Una API (Application Programming Interface - Interfaz de Programación de Aplicaciones, interfaz de programación de la aplicación) es un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software.

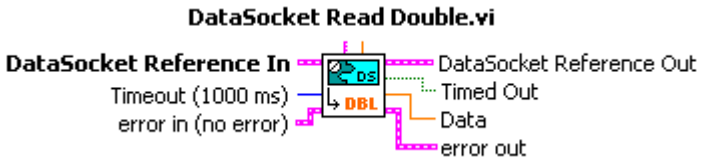

Figura 20 Lectura de Datasocket

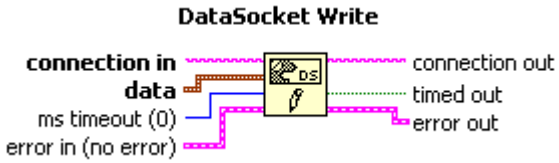

Figura 21 Escritura en Datasocket

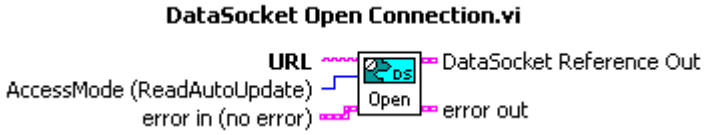

Figura 22 Abrir una conexión datasocket

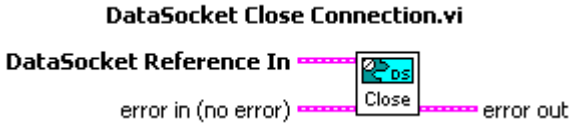

Figura 23 cerrar conexión Datasocket

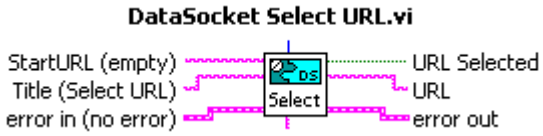

Figura 24 Seleccionar ubicación desde una lista

Además posee otro componente para realizar la búsqueda de las variables compartidas (ítems) de forma dinámica.

A continuación se muestra un pequeño ejemplo de cómo monitorear el estado de una salida del PLC.

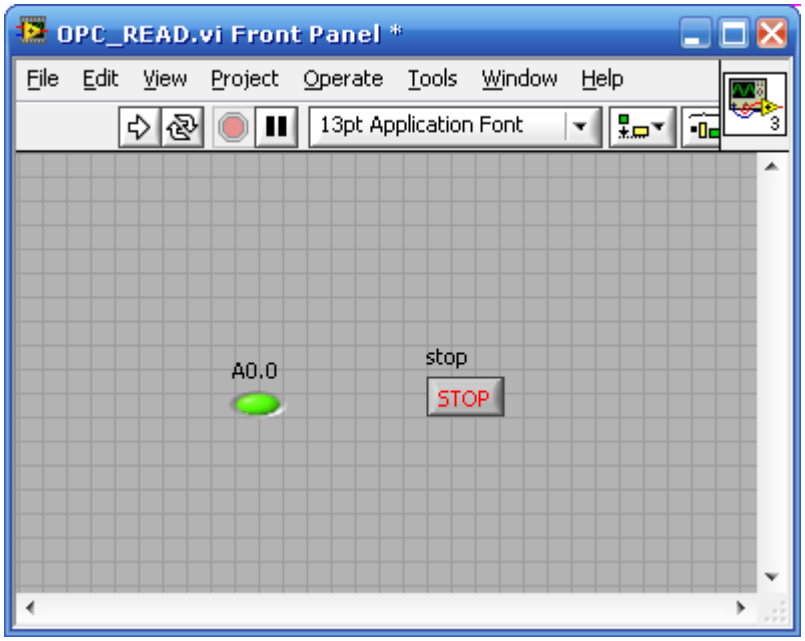

Figura 25 Panel frontal de ejemplo

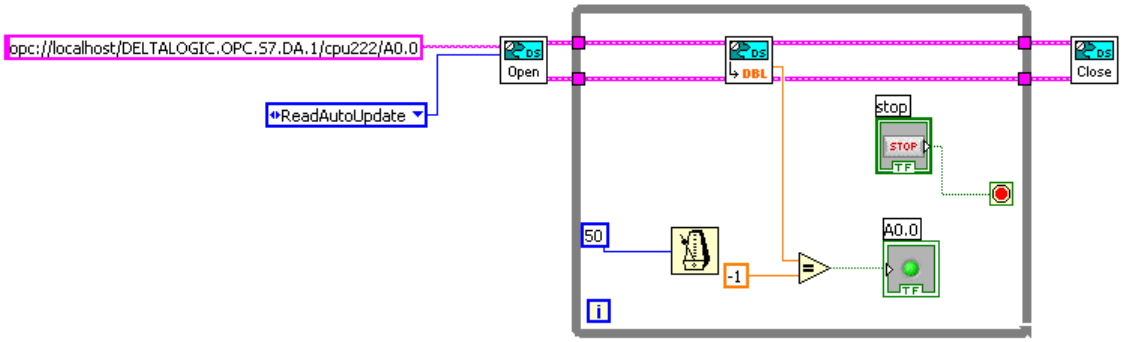

Figura 26. Diagrama de LabVIEW para efectuar comunicación

Para efectuar esta pequeña aplicación se utilizan tres eventos de Datasocket.

Primero se debe abrir la conexión a una URL especificada. Específicamente el ítem configurado en el servidor OPC.

Establecida la conexión con el tag, se procede a leer dicho valor. El programa queda en un lazo condicional while hasta que aparezca la orden *stop del push button*.

Cuando el programa salga del lazo condicional, se cierra inmediatamente la conexión.

VIII.7. INTERFAZ GRÁFICA.

La interfaz gráfica se refiere al entorno visual que un usuario dispone para actuar con las funcionalidades de una aplicación o programa.

Puede haberse invertido mucho en la aplicación, pero estaría fallando si los usuarios finales no encuentran un entorno fácil e intuitivo con el cual operar para conseguir sus metas.

La interfaz gráfica del usuario, o GUI, es el conjunto de elementos gráficos (ventanas, menús, botones, etc.) que permiten la interacción entre el usuario y la aplicación informática. Es el conjunto de componentes empleados por los usuarios para comunicarse con las computadoras. El usuario dirige el funcionamiento de la máquina mediante instrucciones, denominadas genéricamente entradas.

VIII.8. VISUAL BASIC.

Aparece como uno de los lenguajes de programación que más interés despiertan entre los programadores de computadoras, tanto expertos como novatos. En el caso de los programadores expertos por la facilidad con la que desarrollan aplicaciones complejas en poco tiempo (comparado con lo que cuesta programar en ***Visual C++***, por ejemplo). En el caso de los programadores novatos por el hecho de ver de lo que son capaces a los pocos minutos de empezar su aprendizaje.

Visual Basic es un lenguaje de programación visual, también llamado lenguaje de 4ta generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla.

Los programas *interactivos* exigen la intervención del usuario en tiempo de ejecución, bien para suministrar datos, bien para indicar al programa lo que debe

hacer por medio de menús. Los programas interactivos limitan y orientan la acción del usuario.

VIII.8.1. Eventos.

Ya se ha dicho que las acciones del usuario sobre el programa se llaman *eventos*. Son eventos típicos el clic sobre un botón, el hacer doble clic sobre el nombre de un fichero para abrirlo, el arrastrar un icono, el pulsar una tecla o combinación de teclas, el elegir una opción de un menú, el escribir en una caja de texto, o simplemente mover el ratón. Más adelante se verán los distintos tipos de eventos reconocidos por *Windows* y por *Visual Basic*. Cada vez que se produce un evento sobre un determinado tipo de control, *Visual Basic* arranca una determinada *función* o *procedimiento* que realiza la acción programada por el usuario para ese evento concreto. Estos procedimientos se llaman con un nombre que se forma a partir del nombre del objeto y el nombre del evento, separados por el carácter (_), como por ejemplo txtBox_click, que es el nombre del procedimiento que se ocupará de responder al evento *click* en el objeto *txtBox*.

VIII.8.2. Propiedades y métodos.

Además de los eventos, la mayor parte de los objetos, como los formularios y los controles, son suministrados con propiedades y métodos.

Son conceptos fundamentales e importantes.

VIII.8.3. Propiedades.

Una propiedad es una asignación que describe algo sobre un objeto como un formulario. Dependiendo de la propiedad, se la puede asignar en tiempo de diseño usando la ventana Propiedades y/o en tiempo de ejecución al programar.

A continuación se describen dos ejemplos de las propiedades del formulario de Visual Basic:

MinButton. Esta propiedad puede asignarse como TRUE (verdadero) o FALSE (falso). Dependiendo de la asignación, el formulario tendrá o no tendrá un botón minimizar.

BackColor. Asignando esta propiedad a un valor expresado como hexadecimal RGB (Rojo Verde Azul) o como una constante se cambia el color del fondo del formulario.

VIII.8.4. Métodos.

Los métodos son funciones que también son llamadas desde programa, pero a diferencia de los procedimientos no son programadas por el usuario, sino que vienen ya pre-programadas con el lenguaje. Los métodos realizan tareas típicas, previsibles y comunes para todas las aplicaciones. De ahí que vengan con el lenguaje y que se libere al usuario de la tarea de programarlos. Cada tipo de objeto o de control tiene sus propios métodos.

En general solo pueden ser ejecutados en tiempos de ejecución no en tiempo de diseño. Algunos ejemplos de métodos de formularios son el método MOVE, que mueve un formulario en un espacio de dos dimensiones en la pantalla.

Los métodos son invocados dando nombres al objeto y cuyo método se está llamando, listando el operador punto (**.**), y después listando el nombre del método. Como cualquier rutina los métodos pueden incorporar argumentos

VIII.8.5. Programas para el entorno Windows.

Visual Basic está orientado a la realización de programas para *Windows*, pudiendo incorporar todos los elementos de este entorno informático: ventanas, botones, cajas de diálogo y de texto, botones de opción y de selección, barras de desplazamiento, gráficos, menús, etc. Prácticamente todos los elementos de interacción con el usuario de los que dispone *Windows* pueden ser programados en Visual Basic de un modo sencillo. En ocasiones bastan unas pocas operaciones con el ratón y la introducción a través del teclado de algunas

sentencias para disponer de aplicaciones con todas las características de ***Windows***.

VIII.8.6. Modo de diseño y modo de ejecución.

La aplicación Visual Basic de Microsoft puede trabajar de dos modos distintos: en modo de diseño y en modo de ejecución. En modo de diseño el usuario construye interactivamente la aplicación, colocando controles en el formulario, definiendo sus propiedades, desarrollando funciones para gestionar los eventos. La aplicación se prueba en modo de ejecución. En ese caso el usuario actúa sobre el programa (introduce eventos) y prueba cómo responde el programa. Hay algunas propiedades de los controles que deben establecerse en modo de diseño, pero muchas otras pueden cambiarse en tiempo de ejecución desde el programa escrito en Visual Basic, en la forma en que más adelante se verá. También hay propiedades que sólo pueden establecerse en modo de ejecución y que no son visibles en modo de diseño.

VIII.8.7. Formularios y controles

Cada uno de los elementos gráficos que pueden formar parte de una aplicación típica de Windows es un tipo de control: los botones, las cajas de diálogo y de texto, las cajas de selección desplegables, los botones de opción y de selección, las barras de desplazamiento horizontales y verticales, los gráficos, los menús, y muchos otros tipos de elementos son controles para Visual Basic. Cada control debe tener un nombre a través del cual se puede hacer referencia a él en el programa. Visual Basic proporciona nombres por defecto que el usuario puede modificar.

En la terminología de Visual Basic se llama formulario (form) a una ventana. Un formulario puede ser considerado como una especie de contenedor para los controles. Una aplicación puede tener varios formularios, pero un único formulario puede ser suficiente para las aplicaciones más sencillas. Los formularios deben también tener un nombre, que puede crearse siguiendo las mismas reglas que para los controles.

VIII.8.8. Objetos y propiedades.

Los formularios y los distintos tipos de controles son entidades genéricas de las que puede haber varios ejemplares concretos en cada programa. En programación orientada a objetos (más bien basada en objetos, habría que decir) se llama clase a estas entidades genéricas, mientras que se llama objeto a cada ejemplar de una clase determinada. Por ejemplo, un programa puede tener varios botones, cada uno de los cuales es un objeto del tipo de control command button, que sería la clase.

Cada formulario y cada tipo de control tienen un conjunto de propiedades que definen su aspecto gráfico (tamaño, color, posición en la ventana, tipo y tamaño de letra, etc.) y su forma de responder a las acciones del usuario (si está activo o no, por ejemplo). Cada propiedad tiene un nombre que viene ya definido por el lenguaje.

Por lo general, las propiedades de un objeto son datos que tienen valores lógicos (true, false) o numéricos concretos, propios de ese objeto y distintos de las de otros objetos de su clase. Así pues, cada clase, tipo de objeto o control tiene su conjunto de propiedades, y cada objeto o control concreto tiene unos valores determinados para las propiedades de su clase.

Casi todas las propiedades de los objetos pueden establecerse en tiempo de diseño y también -casi siempre- en tiempo de ejecución. En este segundo caso se accede a sus valores por medio de las sentencias del programa, en forma análoga a como se accede a cualquier variable en un lenguaje de programación. Para ciertas propiedades ésta es la única forma de acceder a ellas. Por supuesto Visual Basic permite crear distintos tipos de variables, como más adelante se verá.

Se puede acceder a una propiedad de un objeto por medio del nombre del objeto a que pertenece, seguido de un punto y el nombre de la propiedad, como por ejemplo optColor.objName.

VIII.8.9. Orden de disparo de eventos.

Para controlar con éxito la aparición y el comportamiento de los formularios (y también de los controles) en tiempos de ejecución, debe comprenderse en qué orden se disparan los eventos. Las consideraciones del orden de disparo de los eventos deciden generalmente por el usuario donde debe ser colocada una parte determinada de código de respuesta de un evento. Los eventos de formularios pueden ser divididos en los grupos siguientes:

- Inicio.
- Respuesta a una acción (de usuario).
- Vinculación.
- Cierre.

Es importante también comprender que un evento inicia automáticamente con frecuencia a otro evento, produciendo un efecto en cascada. Por ejemplo un evento *KeyPress* no puede ser disparada sin disparar también lo eventos *KeyUp y KeyDown*. El secreto para trabajar con esta clase de situaciones es una comprensión clara de que es lo que dispara cada evento en la secuencia; el peligro de la codificación es iniciar una cadena sin fin de llamada a eventos circulares recursivos.

VIII.8.10. Eventos relacionados con el ratón.

Clic y DblClick.

El evento Click se activa cuando el usuario pulsa y suelta rápidamente uno de los botones del ratón. También puede activarse desde código (sin tocar el ratón) variando la propiedad Value de uno de los controles. En el caso de un formulario este evento se activa cuando el usuario clica sobre una zona del formulario en la que no haya ningún control o sobre un control que en ese momento esté inhabilitado (propiedad Enabled = False). En el caso de un control, el evento se activa cuando el usuario realiza una de las siguientes operaciones:

- Hacer clic sobre un control con el botón derecho o izquierdo del ratón. En el caso de un botón de comando, de un botón de selección o de un botón de opción, el evento sucede solamente al darle un clic con el botón izquierdo.
- Seleccionar un registro de alguno de los varios tipos listas desplegables que dispone Visual Basic.
- Pulsar la barra espaciadora cuando el foco está en un botón de comando, en un botón de selección o en un botón de opción.
- Pulsar la tecla Return cuando en un formulario hay un botón que tiene su propiedad Default = True.
- Pulsar la tecla Esc cuando en un formulario hay un botón que tiene su propiedad Cancel = True.
- Pulsar una combinación de teclas aceleradoras (Alt + otra tecla, como por ejemplo cuando de despliega el menú File de Word con Alt+F) definidas para activar un determinado control de un formulario.
- También se puede activar el evento Click desde código realizando una de las siguientes operaciones:
- Hacer que la propiedad Value de un botón de comando valga True.
- Hacer que la propiedad Value de un botón de opción valga True
- Modificar la propiedad Value de un botón de selección.
- El evento DblClick sucede al hace clic dos veces seguidas sobre un control o formulario con el botón izquierdo del ratón.

MouseDown, MouseUp y MouseMove.

El evento *MouseDown* sucede cuando el usuario pulsa cualquiera de los botones del ratón, mientras que el evento *MouseUp* sucede al soltar un botón que había sido pulsado. El evento *MouseMove* sucede al mover el ratón sobre un control o formulario.

Los eventos *MouseUp* y *MouseDown* tienen algunos argumentos que merecen ser comentados. El argumento *Button* indica cuál de los botones del ratón ha sido pulsado o soltado, y el argumento *Shift* indica si además alguna de las teclas *Alt*, *Shift* o *Ctrl* está también pulsada.

DragOver y DragDrop.

El evento *DragOver* sucede mientras se está arrastrando un objeto sobre un control. Suele utilizarse para variar la forma del cursor que se mueve con el ratón dependiendo de si el objeto sobre el que se encuentra el cursor en ese momento es válido para soltar o no. El evento *DragDrop* sucede al concluir una operación de arrastrar y soltar. El evento *DragOver* requiere de los argumentos que se muestran a continuación:

Private Sub Text1_DragOver (Source As Control, _X As Single, Y As Single, State As Integer

...

End Sub

Los argumentos de este evento son *Source* que contiene el objeto que está siendo arrastrado, *X* e *Y* que indican la posición del objeto arrastrado dentro del sistema de coordenadas del objeto sobre el que se está arrastrando y *State* (que es propio del *DragOver*, pero no aparece en el *DragDrop*) que vale 0, 1 ó 2 según se esté *entrando*, *saliendo* o *permaneciendo dentro* del mismo objeto, respectivamente. Es importante señalar que el evento *DragOver* es propio del objeto sobre el que se arrastra, no del objeto que es arrastrado

VIII.9. Microcontroladores PIC.

VIII.9.1. Generalidades

Los microcontroladores PIC[17] son una familia de microcontroladores tipo RISC[18] fabricados por Microchip Technology Inc.

[17] **PIC,** controlador de interfaz de periféricos.

[18] **RISC,** computadora con set de instrucciones reducido.

El PIC usa un juego de instrucciones tipo RISC, cuyo número varia, desde 35 para PIC's de gama baja a 70 para los de gama alta. Las instrucciones son de los siguientes tipos:

- Las que realizan operaciones entre el acumulador y una constante.
- Las que lo hacen entre el acumulador y una posición de memoria
- Instrucciones de condicionamiento y de salto/retorno.
- Implementación de interrupciones.
- Implementación del modo de bajo consumo de energía.

La compañía fabricante proporciona software de desarrollo para PC gratuito, este se llama MPLAB, incluye un simulador software y un ensamblador. Otras empresas desarrollan compiladores C y BASIC, entre ellas están Mikroelektronica, CCS, SDCC y otras.

Todos los PIC's manejan datos en paquetes de 8 bits, todos menos los dsPIC[19], por lo que se deberían llamar microcontroladores de 8 bits. Pero a diferencia de la mayoría de CPUs, la arquitectura del PIC permite que el tamaño de las instrucciones pueda ser distinto del de la palabra de datos.

Las familias de los microcontroladores PIC se pueden ver en el siguiente diagrama de Funcionalidad y Desempeño.

[19] **dsPIC**, PIC con capacidad de procesamiento digital de señales.

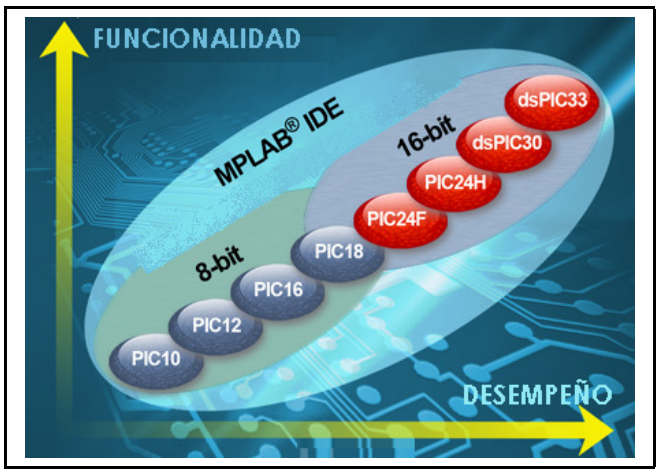

Figura 27. Familias de microcontroladores PIC, funcionalidad y desempeño.

Las características más comunes de los PIC's son las siguientes:

- Puertos de E/S (típicamente 0 a 5,5 voltios)
- Memoria Flash y ROM disponible desde 256 bytes a 256 kilobytes
- Núcleos de CPU de 8/16 bits.
- Temporizadores de 8/16 bits
- Tecnología para modos de ahorro de energía
- Periféricos serie síncronos y asíncronos: USART (Transmisor/Receptor Síncrono/Asíncrono Universal), , EUSART (Transmisor/Receptor Síncrono/ Asíncrono Universal Mejorado)
- Convertidores analógico/digital de 10-12 bits
- Comparadores de tensión
- Módulos de captura y comparación PWM (Modulación por ancho de pulso)
- Controladores LCD (Display de cristal liquido)
- Periférico MSSP para comunicaciones I^2C (bus de comunicación serie de circuitos Inter-Integrados), SPI (Interfaz serial de Periféricos), y I^2S (Inter-IC sound, es una interfaz de bus usada para conectar dispositivos de audio digital)
- Memoria EEPROM interna con duración de hasta un millón de ciclos de lectura/escritura
- Periféricos de control de motores

- Soporte de interfaz USB
- Soporte de controlador Ethernet
- Soporte de controlador Irda

VIII.9.2. Programación en C para PIC's usando el compilador CCS[20]

VIII.9.3. Descripción de los compiladores PCB, PCM, y PCH

PCB está hecho para los opcodes[21] de 12 bits, PCM esta hecho para los opcodes de 14 bits, y PCH está hecho para los opcodes de 16 bits, todos son para opcodes de microcontroladores PIC. Debido a que son muy similares, es la misma información para los tres, estos compiladores están específicamente diseñados para cumplir con las necesidades únicas de los microcontroladores PIC, esto permite a los desarrolladores diseñar rápidamente aplicaciones de software de una manera más fácil, en lenguaje de alto nivel.

Cuando se compara con compiladores C más tradicionales, PCB, PCM y PCH tienen algunas limitantes. Como ejemplo de estas limitaciones tenemos, que la recursión de funciones no está permitida. Esto debido al hecho de que los PICs no tienen pilas para guardar los valores de las variables actuales. Los compiladores pueden implementar de manera eficiente construcciones, operaciones de entrada/salida, y operaciones de manejo de bits. Todos los tipos de datos de C son soportados así como también punteros, arreglos de constantes, decimales de punto fijo, y arreglos de bits.

VIII.9.4. Estructura principal

Un programa está formado de los siguientes cuatro elementos en un archivo:

- Comentario
- Directiva pre-procesador

[20] Toda la sección ha sido adaptada y traducida de la ayuda del compilador CCS para C. http://www.ccsinfo.com descargado diciembre del 2006.

[21] **Opcodes,** código de operación, es la porción de una palabra de instrucción que especifica que tipo de instrucción es.

- Definición de datos
- Definición de función

Todo programa en C debe contener una función “main”, que es el punto de inicio del programa en ejecución. El programa puede ser partido en múltiple funciones de acuerdo con su propósito y las funciones pueden ser llamadas desde la función “main” o desde otras subfunciones. En grandes funciones de proyecto las subfunciones pueden ser colocadas en diferentes archivos de C o encabezado de archivos, y estos pueden ser incluidos en el archivo “main” de C. CCS C también requiere la inclusión del archivo de dispositivo apropiado, usando la directiva #include para incluir las funcionalidades del dispositivo a usar. También hay unas directivas de procesador como #fuses para especificar los fusibles del chip, y #use para especificar la velocidad del clock. Las funciones contienen las declaraciones de datos, definiciones, declaraciones, y expresiones. El compilador también provee un gran número de librerías Standard de C, así como también drivers para otros dispositivos que pueden ser incluidos y usados en los programas. CCS también provee un gran número de funciones incorporadas para acceder a varios periféricos incluidos en los microcontroladores PIC.

VIII.9.5. Declaraciones

Las principales declaraciones del compilador CCS para C, se muestran en la siguiente tabla:

Declaracion	Ejemplo

if (expr) stmt; [else stmt;]	`if (x==25)`
	`  x=1;`
	`else`
	`  x=x+1;`

while (expr) stmt;	`while (get_rtcc()!=0)` `  putc('n');`

do stmt while (expr);	`do {`
	`  putc(c=getc());`
	`} while (c!=0);`

for (expr1;expr2;expr3) stmt;	`for (i=1;i<=10;++i)` `  printf("%u\r\n",i);`

switch (expr) {
expr de caso: stmt; //uno o mas casos [default:stmt]
... }
	`switch (cmd) {`
	`  case 0:  printf("cmd 0");`
	`  break;`
	`  case 1: printf("cmd 1");`
	`  break;`
	`  default: printf("bad cmd");`
	`  Break; }`

return [expr];	`return (5);`
goto label;	`goto loop;`
label: stmt;	`loop: I++;`
break;	`break;`
continue;	`continue;`
expr;	`i=1;`
;	`;`
{[stmt]}	`{a=1;` ` b=1;}`

Cero o mas

Tabla 1. ***Declaraciones del compilador CCS***

VIII.9.6. Operadores

Los operadores principales del compilador CCS para C, se muestran en la siguiente tabla:

Operador	Definición
+	Operador suma
+=	Operación de suma y resultado, x+=y
&=	Operador Bitwise y resultado
&	Operador de dirección
&	Operador Bitwise
^=	Bitwise exclusive or assignment operator, x^=y
^	Bitwise exclusive or operator
l=	Bitwise inclusive or assignment operator, xl=y
l	Bitwise inclusive or operator
?:	Operador de expresión condicional
- -	Decremento
/=	Operador División y resultado, x/=y
/	Operador División
==	Igualdad
>	Operador Mayor que
>=	Operador Mayor o igual que

++	Incremento
*	Operador de Indirección
!=	Inigualdad
<<=	Operador de rotación a la izquierda y resultado
<	Operador Menor que
<<	Operador de rotación a la izquierda
<=	Operador Menor o igual que
&&	Operador lógico AND
!	Operador Lógico de Negación
ll	Operador Lógico OR
%=	Operador de asignación de módulos x%=y
%	Operador de Módulos
*=	Operador de Multiplicación y resultado
*	Operador de Multiplicación
~	Operador de Complemento a Uno
>>=	Operador de rotación a la derecha y resultado
>>	Operador de rotación a la derecha
->	Operación de puntero de Estructura
-=	Operador de resta y resultado
-	Operador de Resta
sizeof	Determina el tamaño en bytes del operador

Tabla 2. ***Operadores del compilador CCS***

VIII.9.7. Definiciones de datos

Los Principales tipos de datos del compilador CCS para C, se muestran en la siguiente tabla:

Dato	Definición
int1	Define un numero de 1 bit
int8	Define un numero de 8 bits
int16	Define un numero de 16 bits
int32	Define un numero de 32 bits
char	Define un carácter de 8 bits
float	Define un numero de punto flotante de 32 bits
short	Por predeterminación es igual que int1
Int	Por predeterminación es igual que int8
long	Por predeterminación es igual que int16
void	Indica un tipo no especifico

Tabla 3. ***Tipos de datos del compilador CCS***

VIII.9.8. Declaración de variables

Una declaración especifica un tipo de calificador y un tipo de especificador, y es seguido por una lista de uno o más variables de ese tipo, por ejemplo:

int a,b,c,d;

mybit e,f;

mybyte g[3][2];

*char *h;*

colors j;

struct data_record data[10];

static int i;

extern long j;

VIII.9.9. Uso de la memoria de programa para datos

El compilador C de CCS permite usar la memoria de datos de varias maneras. Estas se discuten a continuación:

Datos Constantes:

un calificador constante coloca las variables en la memoria de programa. La sintaxis es “const type especif id [cexpr] ={value}” si la palabra clave CONST es usada antes de identificador, el identificador es tratado como constante. Las constantes deben ser inicializadas y no deben ser cambiadas en la ejecución.

Por ejemplo:

const char cstaring[6]={"hello"}	*Para colocar una cadena en la memoria ROM*
*const char *cptr;*	*Para crear punteros de constantes*
cptr = string;	
#ORG 0x1C00, 0x1C0F	*El comando #org puede ser usado para colocar una constante en bloques de direcciones especificados, para el ejemplo, esta ID estará en la dirección 1C00.*
CONST CHAR ID[10]= {"123456789"};	

La función “label_address” puede usarse para obtener la dirección de una constante. La variable constante puede ser acezada en el código. Esta es una buena manera de guardar datos constantes en programas grandes. Cadenas constantes de tamaño variado pueden ser guardadas en la memoria de programa.

Para PIC18 el compilador permite de una manera no standard en C, la implementación de un arreglo constante de cadenas de tamaño variable. La sintaxis es la siguiente:

const char id[n] [*] = { "strint", "string" ...};

Donde n es opcional, y id es el identificador de la tabla. Como por ejemplo:

const char colors[] [*] = { "Red", "Green", "Blue"};

Directiva #ROM:

Otro método es usar #rom para asignar datos a la memoria de programa, el uso es #rom address={data,data,…,data}. Por ejemplo:

#rom 0x1000={1,2,3,4,5} //pondra 1,2,3,4,5 en rom iniciando en 0x1000

#rom address={"hello"} // pondrá hello en la memoria rom

Funciones Incorporadas:

El compilador también provee funciones incorporadas para colocar datos en la memoria de programa, las funciones son las siguientes:

VIII.9.10. Funciones Incorporadas del compilador CCS

El compilador CCS tiene funciones incorporadas que sirven para utilizar de manera rápida los subsistemas que estén disponibles en el modelo de PIC que se esté usando, los subsistemas y las funciones incorporadas para el uso de estos, son los siguientes:

RS232 I/O	ASSERT( )	GETCH( )	PUTC( )
	FGETC( )	GETCHAR( )	PUTCHAR( )
	FGETS( )	GETS( )	PUTS( )
	FPRINTF( )	KBHIT( )	SET_UART_SPEED( )
	FPUTC( )	PERROR( )	SETUP_UART( )

	FPUTS( )	PRINTF( )		
SPI DOS CABLES I/O	SETUP_SPI( )	SPI_DATA_IS_IN( )	SPI_READ( )	SPI_WRITE( )
	SPI_XFER( )			
I/O DISCRETAS	GET_TRISx( )	INPUT_K( )	OUTPUT_FLOAT( )	SET_TRIS_B( )
	INPUT( )	INPUT_STATE( )	OUTPUT_G( )	SET_TRIS_C( )
	INPUT_A( )	INPUT_x( )	OUTPUT_H( )	SET_TRIS_D( )
	INPUT_B( )	OUTPUT_A( )	OUTPUT_HIGH( )	SET_TRIS_E( )
	INPUT_C( )	OUTPUT_B( )	OUTPUT_J( )	SET_TRIS_F( )
ESCLAVO PARALELO I/O	PSP_INPUT_FULL( )		PSP_OVERFLOW( )	
	PSP_OUTPUT_FULL( )		SETUP_PSP( )	
I²C I/O	I²C_ISR_STATE()	I²C_SlaveAddr( )	I²C_WRITE( )	
	I²C_POLL( )	I²C_START( )		
	I²C_READ( )	I²C_STOP( )		
CONTROLES DE PROC.	CLEAR_INTERRUPT( )	GOTO_ADDRESS( )	RESET_CPU( )	
	DISABLE_INTERRUPTS( )	INTERRUPT_ACTIVE( )	RESTART_CAUSE( )	
	ENABLE_INTERRUPTS( )	JUMP_TO_ISR	SETUP_OSCILLATOR( )	
	EXT_INT_EDGE( )	LABEL_ADDRESS( )	SLEEP( )	
	GETENV( )	READ_BANK( )	WRITE_BANK( )	
MANIPULACION BIT/BYTE	BIT_CLEAR( )	MAKE8( )	_MUL( )	SHIFT_LEFT( )
	BIT_SET( )	MAKE16( )	ROTATE_LEFT( )	SHIFT_RIGHT( )
	BIT_TEST( )	MAKE32( )	ROTATE_RIGHT( )	SWAP( )
MATEMATICA STANDARD C	ABS( )	COSH( )	LABS( )	SIN( )
	ACOS( )	DIV( )	LDEXP( )	SINH( )
	ASIN( )	EXP( )	LDIV( )	SQRT( )
VOLTAJE DE REF	SETUP_LOW_VOLT_D		SETUP_VREF( )	

	ETECT( )
CONVERSION A/D	SET_ADC_CHANNEL( ) SETUP_ADC( ) SETUP_ADC_PORTS( ) READ_ADC( )
CHAR STANDARD C	ATOF( ) ATOI( ) ATOI32( ) ISLOWER(char) ISPRINT(x) ISPUNCT(x) STRCMP( ) STRCOLL( ) STRCPY( ) STRRCHR( ) STRSPN( ) STRSTR( )
TIMERS	GET_TIMER0( ) GET_TIMER1( ) GET_TIMER2( ) GET_TIMERx( ) RESTART_WDT( ) SET_RTCC( ) SET_TIMER0( ) SET_TIMER1( ) SET_TIMER5( ) SETUP_COUNTERS( ) SETUP_TIMER_0 ( ) SETUP_TIMER_1 ( ) SETUP_TIMER_2 ( ) SETUP_WDT ( )
MEMORIA STANDARD C	CALLOC( ) FREE( ) LONGJMP( ) MALLOC( ) MEMCHR( ) MEMCMP( ) MEMCPY( ) MEMMOVE( ) MEMSET( ) OFFSETOF( ) OFFSETOFBIT( ) REALLOC( ) SETJMP( )
CAPTURA/COMPARA/PWM	SET_POWER_PWM_OVERRIDE( ) SET_POWER_PWMX_DUTY( ) SET_PWM1_DUTY( ) SET_PWM2_DUTY( ) SETUP_CCP2( ) SETUP_CCP3( ) SETUP_CCP4( ) SETUP_CCP5( )
EEPROM INTERNA	ERASE_PROGRAM_EEPROM( ) READ_CALIBRATION( ) READ_EEPROM( ) READ_EXTERNAL_MEMORY( ) READ_PROGRAM_EEPROM( ) READ_PROGRAM_MEMORY( ) SETUP_EXTERNAL_MEMORY( ) WRITE_CONFIGURATION_MEMORY( ) WRITE_EEPROM( ) WRITE_EXTERNAL_MEMORY( ) WRITE_PROGRAM_EEPROM( ) WRITE_PROGRAM_MEMORY( )
RTOS	RTOS_AWAIT( ) RTOS_DISABLE( ) RTOS_MSG_SEND( ) RTOS_OVERRUN( ) RTOS_TERMINATE( ) RTOS_WAIT( )

	RTOS_ENABLE( )	RTOS_RUN( )	RTOS_YIELD( )
	RTOS_MSG_POLL( )	RTOS_SIGNAL( )	
	RTOS_MSG_READ( )	RTOS_STATS( )	
LCD	LCD_LOAD( )	LCD_SYMBOL( )	SETUP_LCD( )

Tabla 4.

VIII.10. Bus serie universal (USB)

Antes de definir el control OCX utilizado para el control del puerto USB, se dará una pequeña introducción de lo que es el Puerto USB y la clase que se utilizará, que es el estándar HID.

VIII.10.1. Descripción del USB

El USB o Universal Serial Bus es una interfaz para la transmisión serie de datos y distribución de energía desarrollado por empresas líderes del sector de las telecomunicaciones y de los ordenadores, y que ha sido introducida en el mercado de las computadoras personales y periféricos para mejorar las lentas interfaces serie rs-232 y paralelo. Provee una mayor velocidad de transferencia (de hasta 100 veces más rápido) comparado con el puerto paralelo de 25-pin y el Serial DB-9, DB-25, rs-232 que son los puertos que se encuentran en la mayoría de los computadores. Tenía en principio como objetivo el conectar periféricos relativamente lentos (ratones, impresoras, cámaras digitales, unidades zip, etc.) de una forma realmente sencilla, rápida y basada en comunicaciones serie, aunque por sus características también podía conectarse hasta discos duros.

Esta interfaz de 4 hilos distribuye 5V para la alimentación y puede transmitir datos a una velocidad de hasta 480 Mbps en su versión 2.0. Es un bus serie que hace posible la conexión de hasta 127 periféricos a una única puerta de un PC,

con detección y configuración automáticas, siendo esto posible con el PC encendido, sin tener que instalar software adicional, y sin tener que reiniciar el ordenador (plug and play), algo que con los puertos convencionales serie y paralelo no sucedía. Tampoco hay que preocuparse por conflictos de IRQ's[22] o instalar tarjetas de adaptador para cada periférico, estos periféricos pueden ser: Ratones, teclados, impresoras, escáneres, grabadoras, discos duros, módems, cámaras digitales, etc.

VIII.10.2. Características generales del USB

La especificación del USB proporciona una serie de características que pueden ser distribuidas en categorías. Estas características son comunes para todas las versiones (desde la 1.0 hasta la 2.0).

- Fácil uso para los usuarios
- Modelo simple para el cableado y los conectores
- Detalles eléctricos aislados del usuario (terminaciones del bus)
- Periféricos auto-identificativos
- Periféricos acoplados y reconfigurados dinámicamente (hot Swappable[23])
- Flexibilidad
- Amplio rango de tamaños de paquetes, permitiendo variedad de opciones de buffering[24] de dispositivos.

---

[22] Pedido de Interrupción **IRQ** (Interrupt Request). También conocida como interrupción hardware, es una señal recibida por el microprocesador de una computadora, indicando que debe "interrumpir" el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación.

[23] El término "**hot swap" o "hot swappable"** hace referencia a la capacidad de algunos componentes de hardware para sufrir su instalación o sustitución sin necesidad de detener o alterar la operación normal de la PC donde se alojan.

[24] Almacenamiento temporal; una ubicación de la memoria en una computadora o en un instrumento digital reservada para el almacenamiento temporal de información digital, mientras que está esperando ser

- Gran variedad de tasas de datos de dispositivos acomodando el tamaño de buffer para los paquetes y las latencias.
- Control de flujo para el manejo del buffer construido en el protocolo
- Ancho de banda isócrono[25]
- Se garantiza un ancho de banda y bajas latencias apropiadas para telefonía, audio, etc.
- Cantidad de trabajo isócrono que puede usar el ancho de banda completo del bus.
- Amplia gama de aplicaciones y cargas de trabajo
- Adecuando el ancho de banda desde unos pocos kbs hasta varios Mbs
- Soporta tanto el tipo de transferencia isócrono como el asíncrono[26] sobre el mismo conjunto de cables
- Conexiones múltiples, soportando operaciones concurrentes de varios dispositivos
- Soporta hasta 127 dispositivos físicos
- Soporta la transferencia de múltiples datos y flujos de mensajes entre el host y los dispositivos
- Robustez
- Manejo de errores y mecanismos de recuperación ente fallos implementados en el protocolo
- Inserción dinámica de dispositivos

procesada.

[25] Sinónimo de sincrónico. Que tiene un intervalo de tiempo constante entre cada evento.

[26] Que no tiene un intervalo de tiempo constante entre cada evento.

- Soporte para la identificación de dispositivos defectuosos
- Implementación de bajo costo
- Sub canal de bajo coste a 1.5Mbs
- Conectores y cables de bajo coste
- Adecuado para el desarrollo de periféricos de bajo coste.

VIII.10.3. Arquitectura del bus USB

Para lograr un mínimo grado de comprensión del presente trabajo, es necesario analizar los elementos del bus USB desde el punto de vista de los sistemas de comunicaciones.

VIII.10.4. Elementos del USB

El USB es un bus ideado para intercambio de datos entre un computador anfitrión (Host), y dispositivos conectados a él (Esclavos). Los periféricos conectados al USB comparten el ancho de banda del bus mediante un protocolo basado en mensajes (tokens[27]).

Un sistema USB consta de 3 partes:

Anfitrión USB (USB Host o Host).

Dispositivos USB (USB devices).

Interconexión USB (USB interconnect).

[27] Es un bloque de texto categorizado. Por ejemplo una marca de puntuación, un operador, un identificador, un número, etc.

Existe sólo un Host en un sistema USB. Los dispositivos USB proveen servicios o funciones al Host. La interconexión USB es la que soporta el tráfico entre el Host y los dispositivos USB, es el canal de comunicación.

VIII.10.5. Topología

La topología física del USB es de tipo estrella jerarquizada. Con un máximo de 7 niveles de jerarquía.

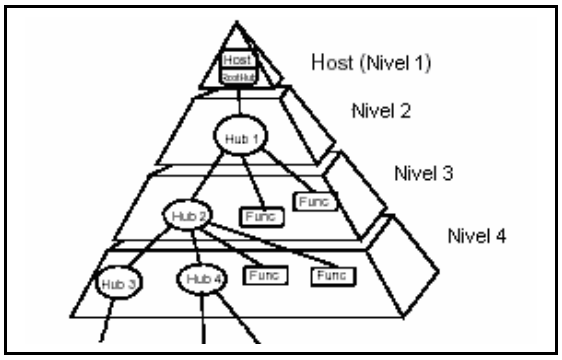

Figura 28. Capas del USB

La topología lógica del USB es de tipo estrella. Lo que implica que en un dispositivo físico puede haber implementado más de un dispositivo lógico (por ej.: un teclado con un mouse incluido).

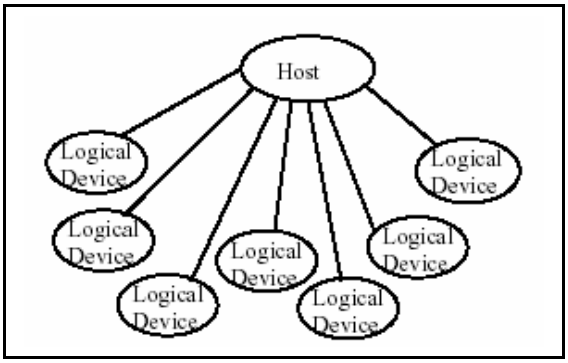

Figura 29. Topologia USB

VIII.10.6. Transmisión y codificación

Los datos son transmitidos en forma serie, en 2 líneas de datos complementarias denominadas D+ y D-. Además se proveen 2 líneas de alimentación y de masa respectivamente, las cuales pueden servir para que el dispositivo tome alimentación del Host (5 V, 500mA máx.). Para transmitir los datos en forma serie se utiliza la codificación Non-Return-To-Zero-Inverted o NRZI. En este tipo de codificación, un 0 (cero) se representa sin un cambio de nivel en la tensión, y un 1 (uno) se representa con un cambio de nivel en la tensión. Conjuntamente, se utiliza el bit stuffing[28], técnica que consiste en insertar un 0 (cero) cada 6 (seis) 1s (unos) consecutivos en el flujo de bits. Además, del bit stuffing y de la codificación NRZI, se utilizan CRCs[29]. Los CRCs se generan después del bit stuffing.

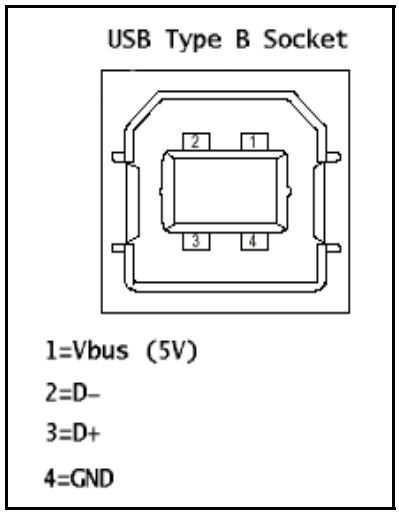

Figura 30. Conector USB de tipo B

[28] Técnica de inserción de bit o de relleno de bit

[29] **CRC** o control de redundancia cíclica, es un mecanismo de detección de errores en sistemas digitales.

VIII.11. Componentes HID.

Los dispositivos de la clase Human Interface Device (HID). Estos poseen las características necesarias para que el usuario interactúe con una PC. Además, los drivers para los mismos están ya incluidos en el sistema operativo.

La ventaja que tiene el HID es que el sistema operativo Windows ya trae los controladores para dispositivos clase HID, eso ayuda que para hacer un dispositivo se requiriera un mínimo de firmware, lo que vuelve más simple la utilización del USB. Algunas desventajas que tiene el HID es que solo se puede trasmitir datos a 64KB/s eso es mucho menor de la capacidad del USB en full-speed que son de 12Mbits/s, pero aun así es más rápido que los puertos serial (RS232) y paralelo.

Se seleccionó un circuito integrado controlador USB para construir el hardware necesario para la realización del trabajo. La función del mismo es, por un lado proveer los niveles eléctricos de las señales que intervienen en la comunicación, y por otro llevar el control de la misma acorde al protocolo, facilitando así la conexión con un microcontrolador de uso habitual. Hecho esto se pasó a diseñar las rutinas básicas del software del microcontrolador para lograr la correcta interconexión con el controlador USB. Esta es la primera etapa para lograr una comunicación con la PC. Cuando un dispositivo USB se conecta a una PC comienza un dialogo entre ellos conocido como *enumeración*, en el cual la PC interroga al dispositivo para identificarlo y conocer sus características, por ejemplo: tipo de dispositivo, forma en que envía la información, etc. Se amplió el software para que lleve a cabo la enumeración de forma tal que el dispositivo fuera reconocido por la PC.

Una vez logrado esto quedó sentada la base para agregar funcionalidad al dispositivo, ya que a grandes rasgos se puede decir que esta parte es común para una amplia gama de los mismos.

VIII.11.1. Clase HID

El nombre HID es la abreviatura de “Human Interface Devices”. Esta Clase, cuya versión actual es el estándar HID 1.11 fue ideada con el propósito de englobar a dispositivos que permitan la interacción del usuario (ser humano) con el Host. Por lo tanto, los requerimientos de ancho de banda son mínimos, y la transferencia de datos debe ser confiable.

Los datos que los dispositivos HID envían al Host son interpretados por el “HID Class Driver” del sistema operativo, para luego poder ser utilizados por la aplicación que los requiera (Client Software). Los requisitos para la implementación de un dispositivo HID son:

- Control Endpoint (Endpoint0): obligatorio
- Interrupt IN Endpoint: obligatorio
- Interrupt OUT Endpoint: opcional

VIII.11.2. Facilidades para el desarrollo de dispositivos USB

Microchip, unos de los más importantes fabricantes de microcontroladores tiene entre su gama alta a la familia 18F2455/2550/4455/4550 dispositivos que entre sus múltiples periféricos cuentan con una interfaz USB 2.0 con compatibilidad para LS y FS que permite una complejidad moderada a la hora de realizar una interfaz utilizando este puerto.

Estos dispositivos cuentan con el hardware necesario para realizar una conexión USB a baja velocidad (Low speed o LS) o velocidad completa (full speed o FS), las operaciones de protocolo deben ser programadas y consumen importantes recursos del sistema como lo son algunas interrupciones y muchos ciclos de máquina de tal forma que sería necesario incluir en el programa un

complejo set de rutinas para controlar el puerto USB, esta situación incrementa la complejidad al desarrollar aplicaciones donde se requiere alto rendimiento y es crítico un control exacto del tiempo, por ejemplo en un sistema de adquisición temporizado internamente, no obstante no se debe descartar este dispositivo a la hora de desarrollar pues se deben tener en cuenta la enormes ventajas que ofrece este, entre las cuales encontramos un bajo costo, altísimas calidad, flexibilidad y el invaluable respaldo y soporte técnico de su fabricante Microchip.

VIII.12. Control de puertos USB para dispositivos con microcontroladores PIC[30]

VIII.12.1. Control OCX[31] HIDComm para Visual Basic 6.0

El control de ActiveX[32] HIDComm esta diseñado para facilitar el proceso de comunicación de la PC a los dispositivos HID[33] USB. Sin el control de ActiveX HIDComm, lo usuarios deberán utilizar formas mas complicadas para comunicarse con el dispositivo.

VIII.12.2. Instalación

El Control de ActiveX HIDComm viene en un archivo fácil de correr "SETUP.EXE" para instalar solo hay que correr este ejecutable, el programa de instalación creara unos cuantos accesos directos en el botón de inicio, para usar el control ActiveX simplemente se debe iniciar Visual Basic 6.

Cuando se aparezca la ventana de nuevo proyecto, se debe elegir "Standard.EXE". Visual Basic creará el proyecto, en la parte izquierda de la pantalla se encuentra una ventana donde están todos los controles ActiveX instalados, como se muestra a continuación.

---

[30] Adaptado y traducido de la ayuda del control OCX HidComm para Visual Basic. http://www.microchip.com descargado en junio del 2004, ya no está disponible.

[31] **OCX,** módulo de programa independiente que puede ser accesado por otros programas en un ambiente de Windows.

[32] **ActiveX,** es un conjunto de tecnologías desarrolladas por Microsoft que permiten a los componentes de software interactuar entre sí en un ambiente conectado de red, como Internet, independientemente del lenguaje de desarrollo en que fueron creados. Los elementos fundamentales de la tecnología ActiveX son el Modelo de objetos componentes (COM) y el Modelo de objetos componentes distribuido (DCOM).

[33] **HID,** dispositivo de interfaz de humano.

Figura 31. ***Controles ActiveX instalados en VB 6.0***

Se debe hacer click derecho en un punto vacío y elegir "Components". Una ventana aparecerá, hay un ítem con el nombre "HIDComm ActiveX Control" se debe poner un check en la cajita y hacer click en OK para cerrar la ventana. Ahora el control ActiveX debe aparecer en la ventana de componentes, para usarlo simplemente hay que arrastrarlo a un formulario que este siendo diseñado.

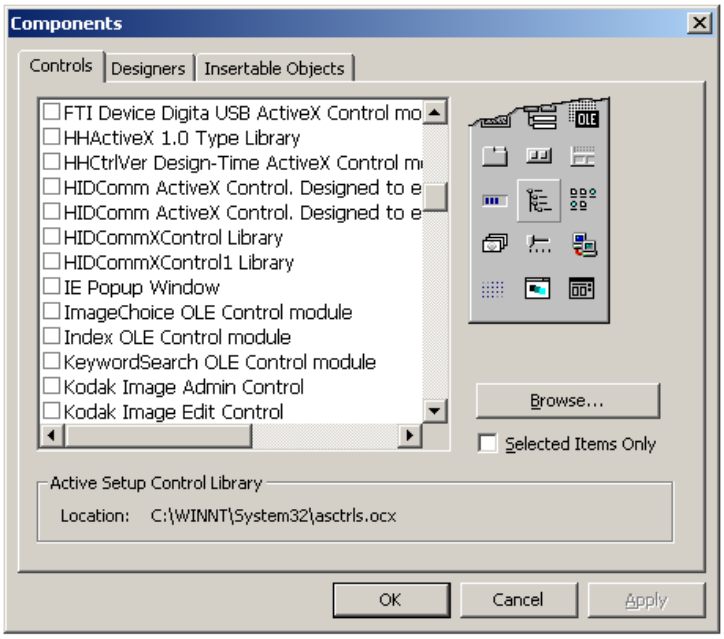

Figura 32. ***Ventana de componentes disponibles en VB 6.0***

VIII.12.3. Preparando el dispositivo USB

El control de ActiveX HIDComm está diseñado para facilitar la comunicación entre la PC y un dispositivo USB HID usando microcontroladores PIC, para mas detalles de cómo programar estos dispositivos se puede ver la sección de este trabajo dedicado a los microcontroladores PIC y referirse a la pagina web del fabricante Microchip.

VIII.12.4. Ajustando los criterios de concordancia

La caja de criterios de concordancia, como se muestra a continuación, habilita al usuario para que éste pueda ajustar de manera rápida los criterios de búsqueda para encontrar el dispositivo HID. El usuario puede llenar cada campo con sus criterios de búsqueda, y habilitar dicho criterio poniendo un cheque en la checkbox de al lado. El usuario puede también buscar a través de los dispositivos HID conectados a la PC haciendo clic en el botón “browse”, y escogiendo el dispositivo deseado.

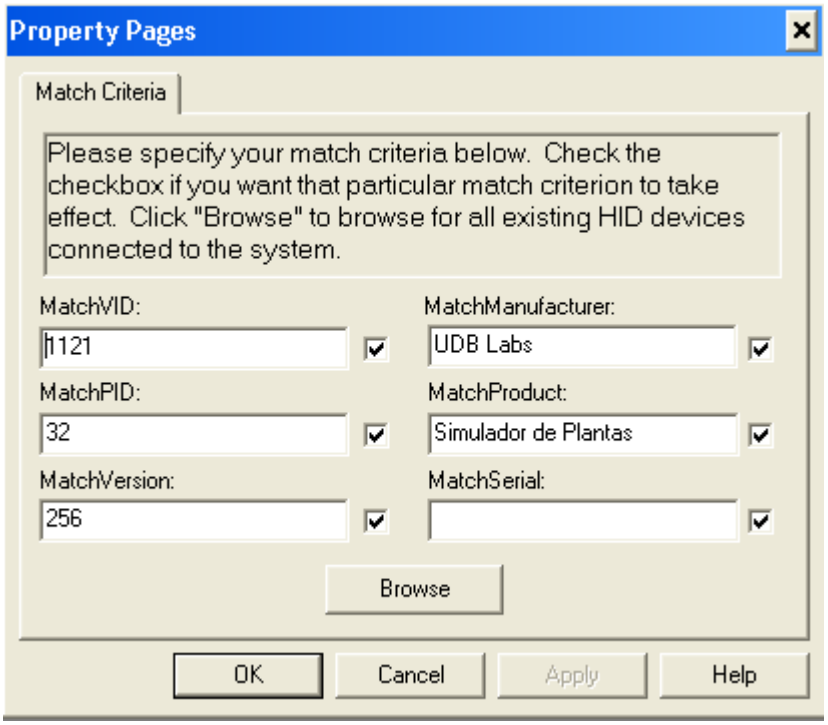

Figura 33.

2.4.4 Criterios de concordancia

El significado de cada campo en la caja de criterios es el siguiente:

MatchPID: es uno de los criterios que se pueden especificar para localizar un dispositivo HID, por ejemplo si se quiere conectar a un dispositivo cuyo Product ID es 1234, se debe especificar 1234 en este campo.

MatchVID: es otro de los criterios que se pueden especificar para localizar un dispositivo HID, por ejemplo si se quiere conectar a un dispositivo cuyo Vendor ID es 1234, se debe especificar 1234 en este campo.

Match Version: es otro de los criterios que se pueden especificar para localizar un dispositivo HID, por ejemplo si se quiere conectar a un dispositivo cuya versión es 1.00, se debe especificar 256 en este campo porque la versión esta expresada en un numero BDC.

MatchManufacturer: es otro de los criterios que se pueden especificar para localizar un dispositivo HID, por ejemplo si se quiere conectar a un dispositivo cuyo fabricante es “Microchip”, se debe especificar “Microchip” en este campo.

MatchProduct: es otro de los criterios que se pueden especificar para localizar un dispositivo HID, por ejemplo si se quiere conectar a un dispositivo que tiene por nombre de Producto “Simulador de Plantas”, se debe especificar “Simulador de Plantas” en este campo.

MatchSerial: es otro de los criterios que se pueden especificar para localizar un dispositivo HID, por ejemplo si se quiere conectar a un dispositivo cuyo serial es 1234ABCD, se debe especificar 1234ABCD en este campo.

VIII.12.5. Ejemplo de uso de entradas y salidas.

Después de colocar el control de ActiveX HIDComm en un formulario vacío se debe revisar la ventana de propiedades.

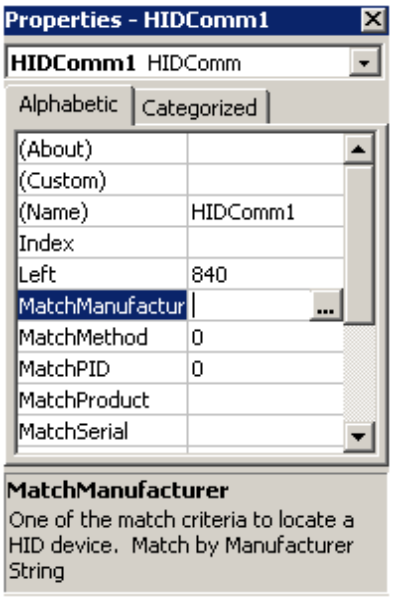

Figura 34. ***Ventana de Propiedades***

Se puede observar que todas las propiedades de tiempo de diseño se muestran aquí, abajo se puede observar una breve descripción de la propiedad siendo editada actualmente. En este caso el MatchManufacturer.

Se debe hacer click en el set de propiedades de MatchManufacturer, después hacer click en el botón de 3 puntos; una interfaz de usuario se mostrara en donde se puede ajustar los criterios de match.

Posteriormente se debe conectar el PIC programado a la PC, cuando el dispositivo USB es conectado por primera vez, la PC desplegara el aviso de "Nuevo Hardware Encontrado" y se buscaran automáticamente los drivers necesarios, no hay necesidad de un driver especial ya que el que trae Windows funciona.

Se debe hacer click en el botón "Browse" en la ventana de criterios de Match, el dispositivo debe ser presentado. Se debe elegir el dispositivo y hacer click en "OK", todos los criterios de Match deberían ser llenados automáticamente. Se debe tener en cuenta que un control ActiveX solo puede comunicarse con un dispositivo a la vez.

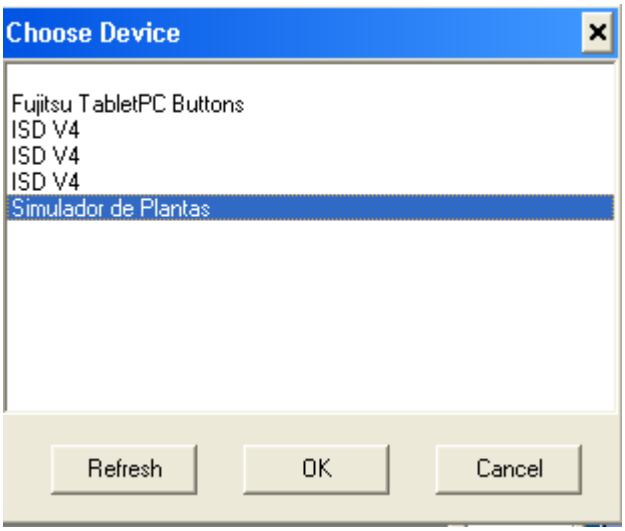

Figura 35

Selección de dispositivo

Los comandos más importantes en el código fuente del formulario son:

"HIDComm1.Connect": le dice a HIDComm que se conecte al dispositivo HID. HIDComm usará los criterios de match para localizar el dispositivo.

“HIDComm1.WriteTo Buffer BufferSize": envía el buffer al dispositivo USB. El BufferSize especifica el número de bytes a ser enviados, este se debe declarar como variable Long.

"HIDComm1.ReadFrom(BufferSize)": lee de el dispositivo USB. El parámetro BufferSize especifica el máximo número de bytes que se puede recibir. Los bytes recibidos son guardados en el buffer, y el número de bytes leídos son guardados en BufferSize.

“HIDComm1.Uninit”: es crítico agregar esta llamada a subrutina en el evento de “Unload” del formulario para desinicializar el control ActiveX

IX. DISEÑO DE LA APLICACIÓN.

IX.1. Diagrama de bloques del sistema

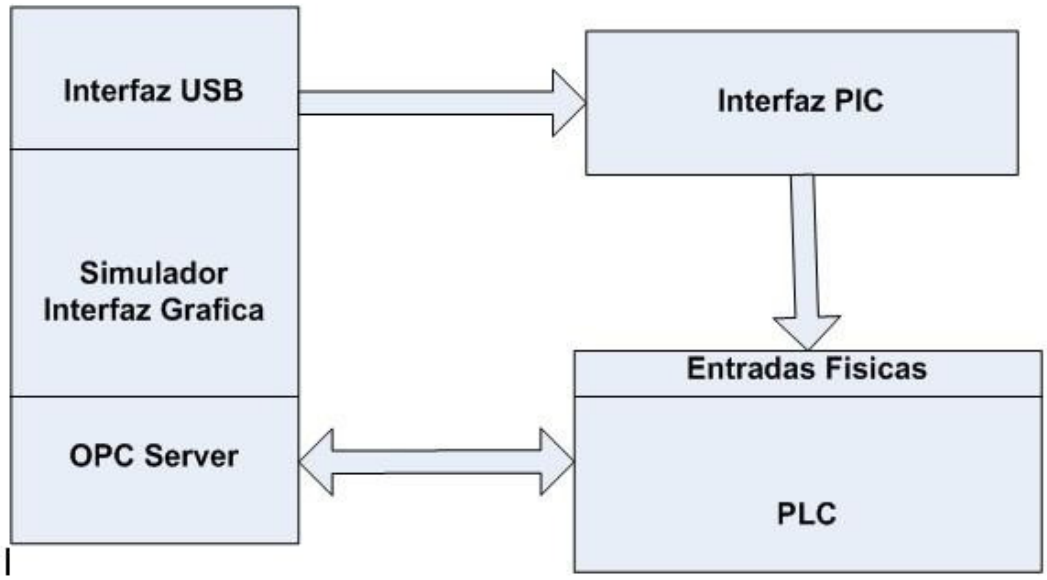

Figura 36.

IX.2. Diseño de interfaz de hardware con microcontrolador PIC

Para poder proveer las entradas de activación al PLC es necesario de diseñar una tarjeta electronica, la comunicación es a través del puerto USB teniendo como salida elementos conmutadores, para este caso el diseño es para 8 salidas a rele con voltaje de conmutación de 24 voltios debido a que los PLC a controlar necesitan este voltaje para operar.

Para el diseño de la interfaz de hardware, tiene como componente principal el microcontroladores PIC18F2550 dicho integrado va a operar con el voltaje que provee el USB, el circuito de protección sirve para proteger al microcontrolador de ruidos provenientes de los circuitos conmutadores(reles) y de la etapa de potencia en si.

El diagrama de bloques de la tarjeta electrónica se muestra a continuación:

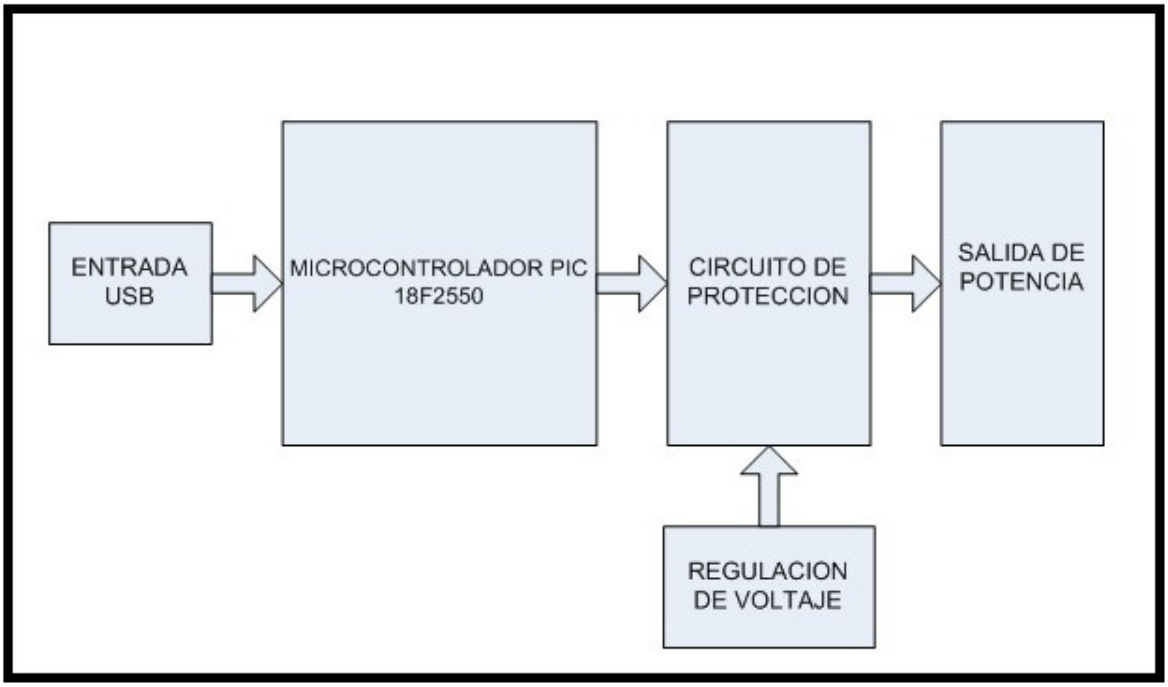

Figura 37

IX.2.1. Diseño de la etapa de regulación de voltaje.

La tarjeta tiene como entrada un voltaje de 24 voltios, este voltaje necesita el PLC para poder activar sus entradas físicas, en su mayoría las demás etapas de la tarjeta trabajan con 12V, por lo tanto, se procedió a realizar un circuito regulador de voltaje, cuya finalidad es mantener el voltaje constante de 12 a su salida, siempre y cuando su entrada sea mayor que este valor de voltaje.

El circuito integrado que se utilizó para realizar esta función es el 7812, cuyas característica principales son las siguientes:

- Voltaje de entrada de 12 a 35 voltios máximo.
- Voltaje de salida de 12V (siempre que en la entrada se mantenga el rango de 12V - 35V).
- Corriente de salida de hasta 1A.

- Protección por sobrecarga térmica.
- Protección por corto circuito.
- Circuito integrado de tres pines. 1. Entrada, 2. Tierra o GND, 3. Salida.

A continuación se presentan las figuras del encapsulado del circuito integrado, asi como también el diagrama de bloques y el diagrama de aplicación usado para la tarjeta.

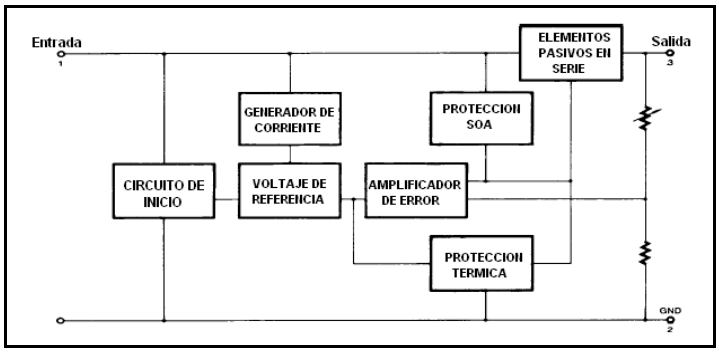

Figura 38. Diagrama de bloques interno del regulador de voltaje 7812.

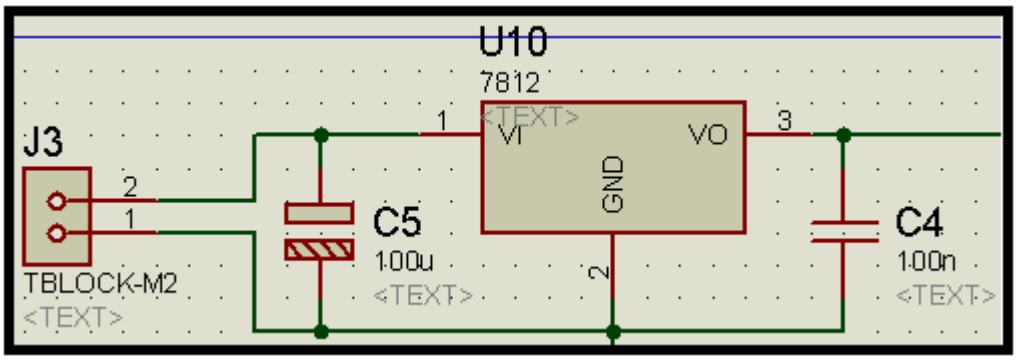

Figura 39. Diagrama esquemático del circuito regulador de voltaje utilizado para alimentar el sistema

IX.2.2. Diseño de circuito de protección y de potencia

La tarjeta posee salidas de potencia para poder manejar el voltaje adecuado para el correcto funcionamiento del PLC ; la interfaz está aislada eléctricamente con la finalidad de protección de sus propios elementos y del propio sistema en sí. Para el aislamiento eléctrico se hace uso de circuitos integrados 4N35 que son dispositivos optoacopladores.

Optoacopladores 4N35: Este circuito integrado se utiliza para acoplar las señales del microcontrolador con el exterior, para evitar sobretensiones, ruido y posibles daños al microcontrolador y a la computadora.

Las salidas de potencia de la interfaz están compuestas por transistores 2N3904 y relevadores de 12V, como se puede observar en la figura abajo; lo cual le permite al sistema un control conmutar el voltaje de 24 voltios a las entradas del PLC.

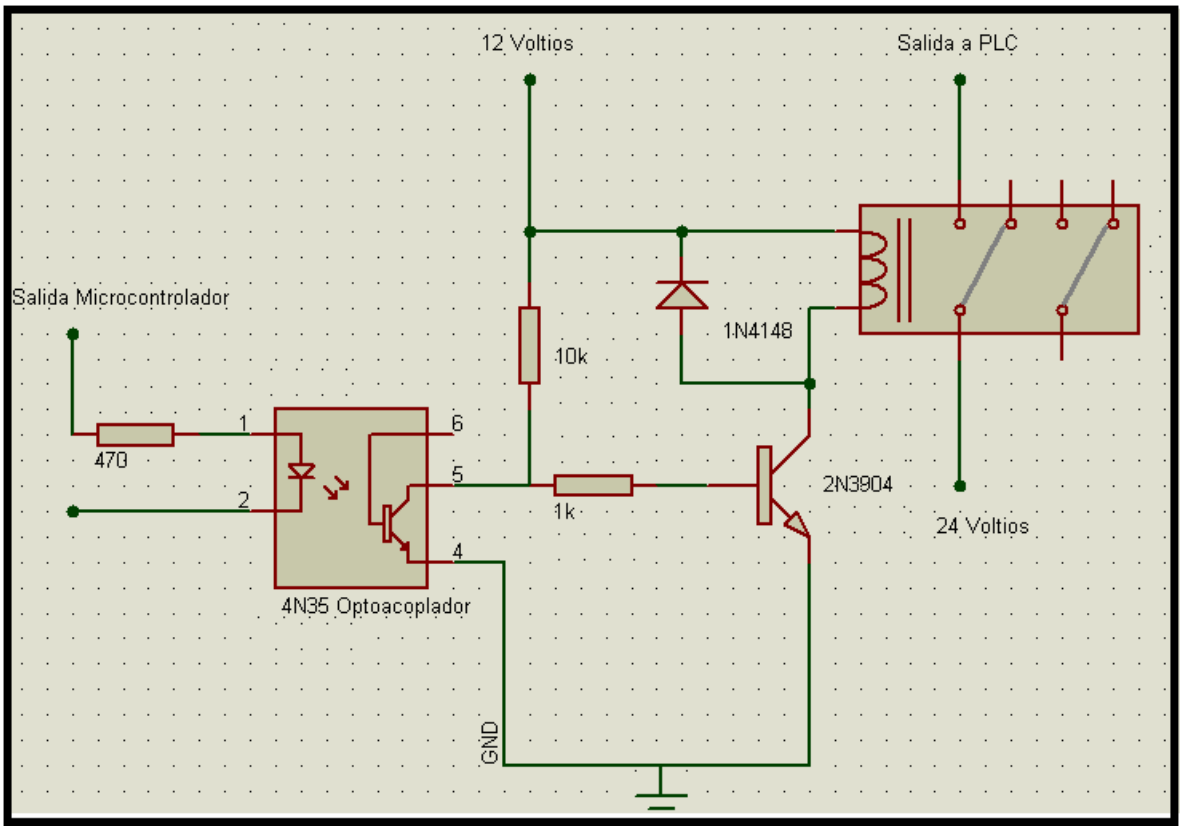

Figura 40. Diagrama Esquemático del circuito de protección y de potencia.

IX.2.3. Diseño de la etapa de procesamiento PIC.

Es la parte central de la tarjeta es la encargada de gestionar la conmutación de los dispositivos de salida hacia el PLC, la comunicación a través de USB.

PIC18F2550: Este circuito integrado microcontrolador, está conectado directamente a la computadora y se alimenta del voltaje que provee el puerto USB, la entrada son las señales de D+ y D- del USB, y se va a destinar un puerto de 8 bits como salida, estas van a ser aisladas eléctricamente y van a ser la señal de activación para los relés que conmutan el voltaje de activación del PLC.

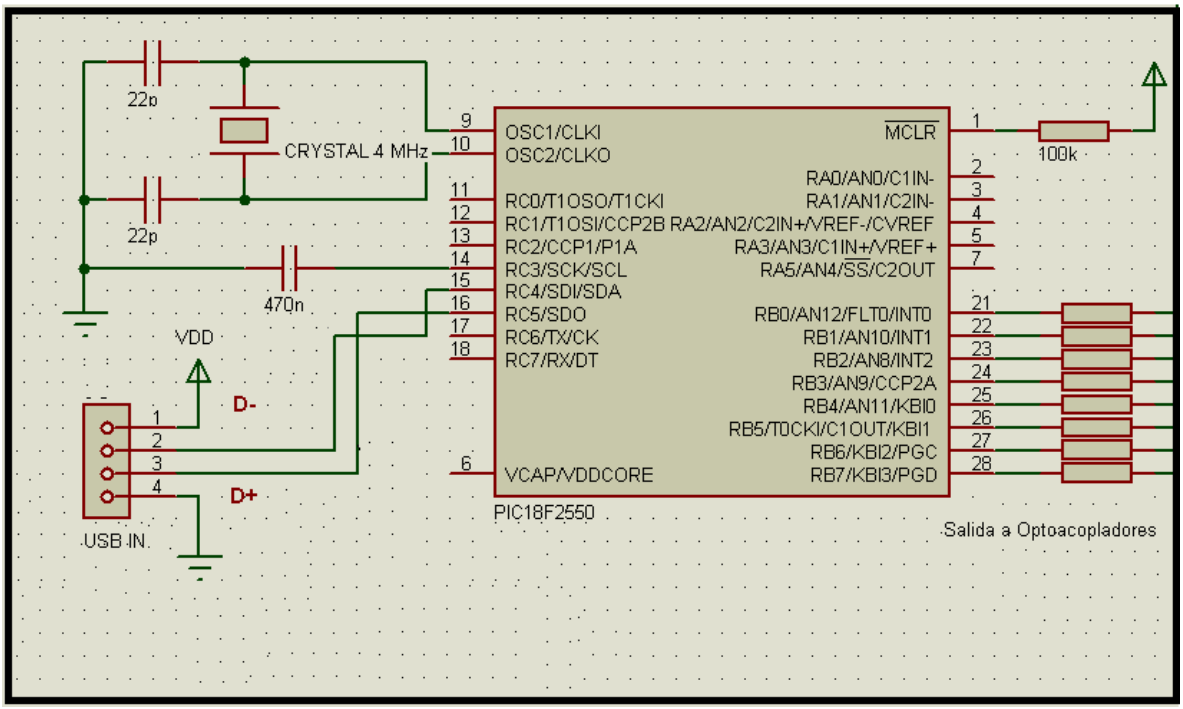

Figura 41

IX.3. Diseño de las pistas del circuito impreso

Para el diseño del circuito impreso se utilizó software especializado para la realización de circuitos impresos a partir del esquema, el software utilizado es Proteus 7 Profesional, que tiene una interfaz para diseño de circuitos esquemáticos y partiendo de ese diseño, automáticamente genera el diseño de impreso.

Con respecto a las dimensiones de las pistas y puntos de conexión, el software tiene en sus librerías los dispositivos y los tamaños de puntos de conexión de una gran variedad de dispositivos electrónicos.

El Software posee la característica de poder variar el tamaño y forma de las pistas, para lo cual se procedió a especificar un ancho de pista de 0.048, este programa puede generar las pistas y ubicaciones de forma automática y manual, la forma en la que se procedió es la distribución de los componentes se hizo de forma manual y impreso.

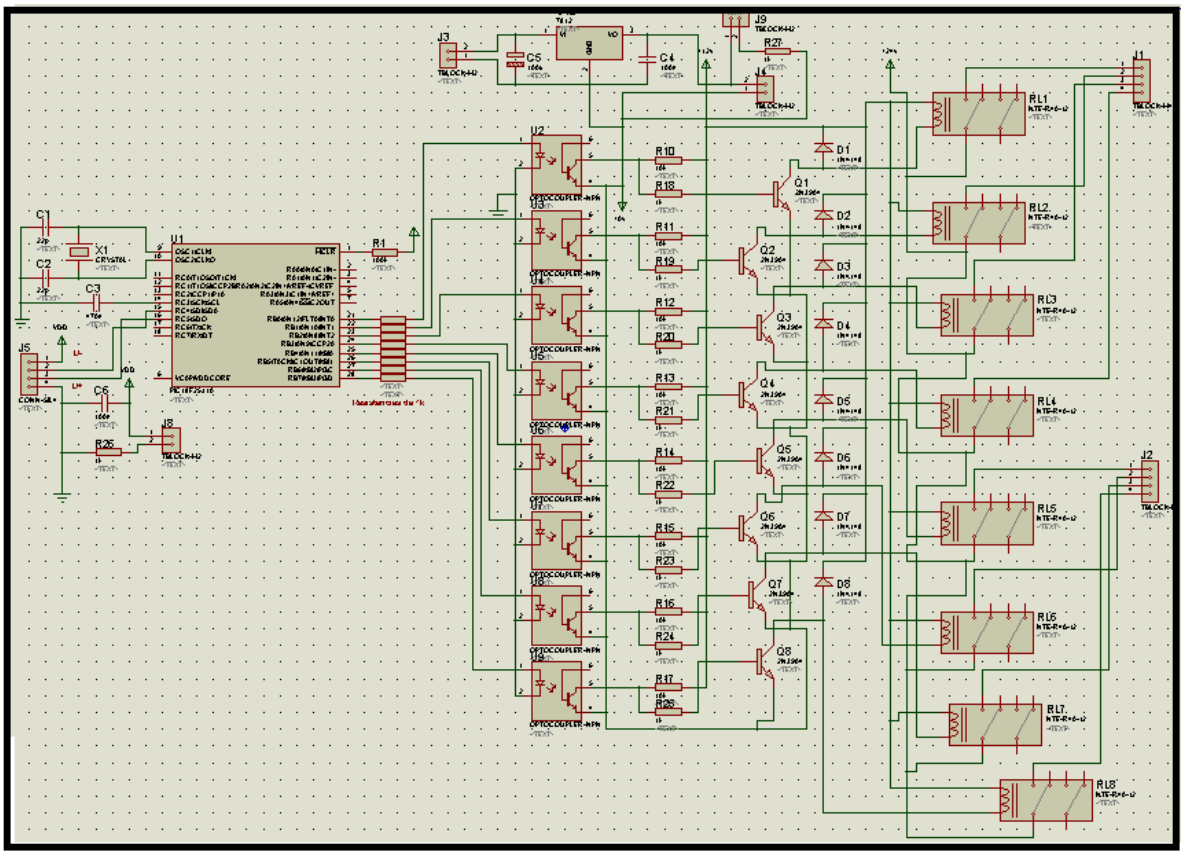

Figura 42. Circuito completo de la interfaz

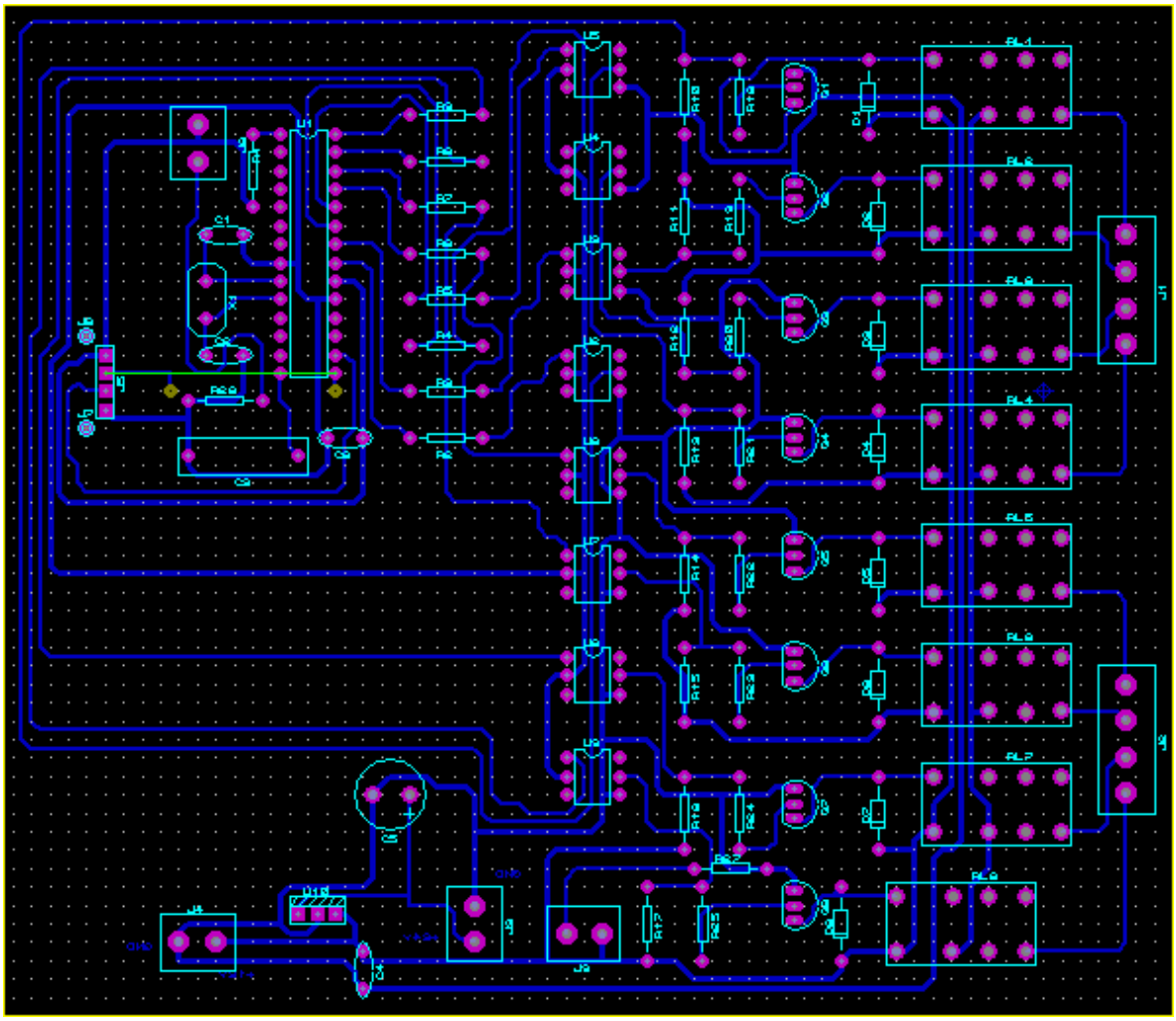

Figura 43. Pistas del circuito impreso de la interfaz de hardware.

IX.4. Interfaz Gráfica

La interfaz grafica se ha desarrollado en Visual Basic dado que es un software especializado en programación visual y orientada a eventos que es un requisito principal de la aplicación.

Posee un formulario principal donde está el espacio de trabajo y formularios secundarios para establecer valores a las propiedades de los objetos del menú principal.

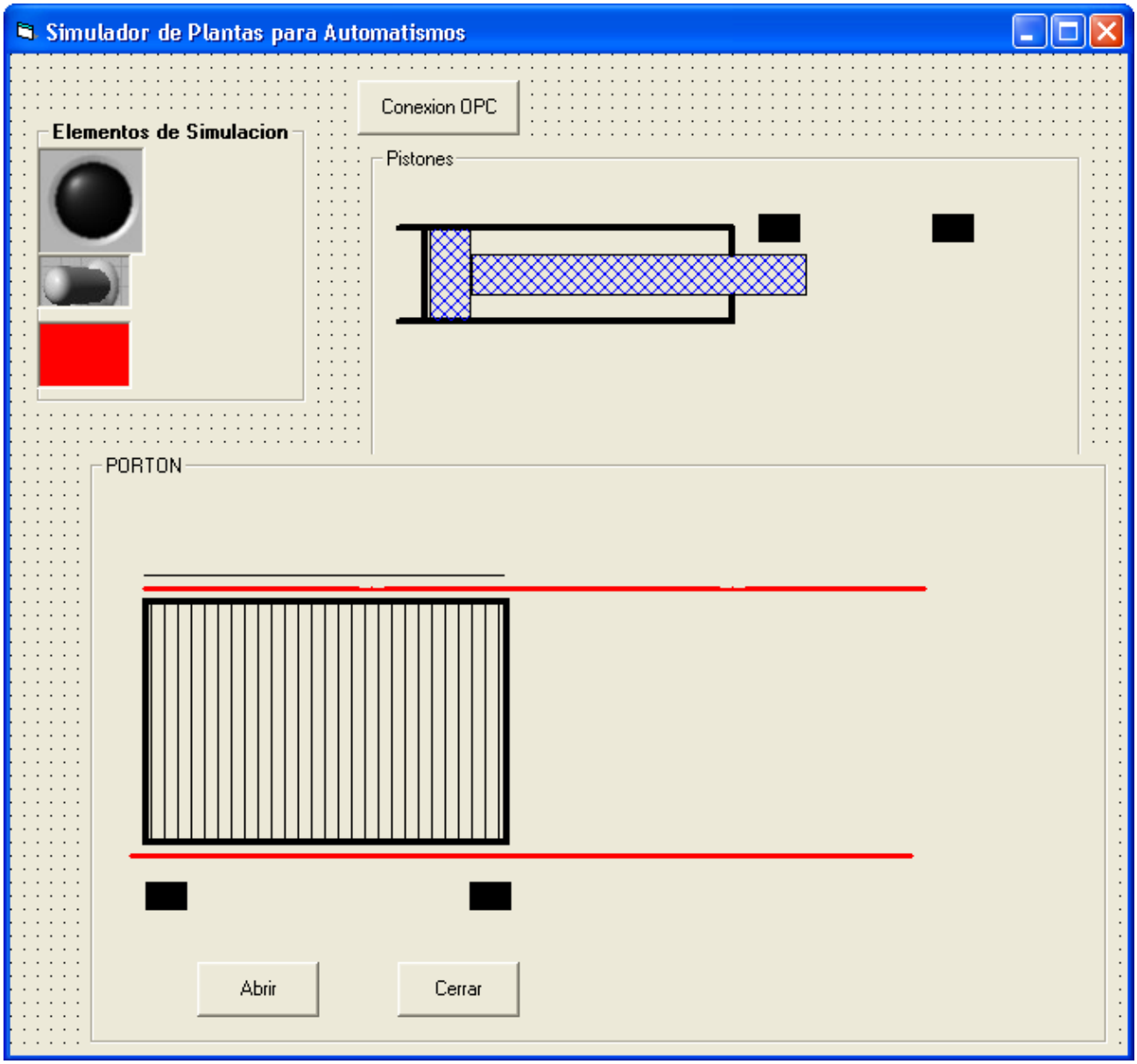

Figura 44. Interfaz Gráfica

La Figura 44 muestra el aspecto de la interfaz posee 4 elementos de simulación,

Interruptores, Indicadores, pistones, portón eléctrico.

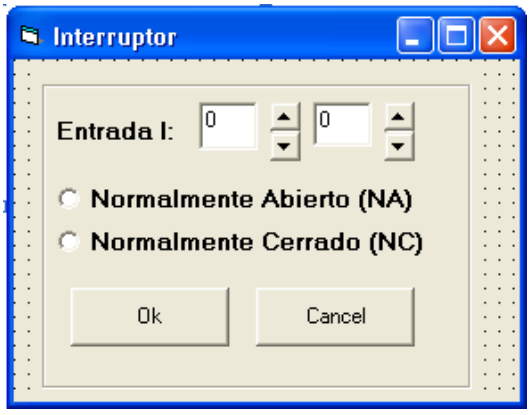

Figura 45. Cuadro de propiedades de Interruptor.

La Fig. 45 muestra el cuadro de propiedades para el objeto interruptor allí se le indica la entrada o marca intena del PLC a la cual va a estar asociada.

Figura 46. Cuadro de propiedades de Led.

La Fig. 46 muestra el cuadro de propiedades para el objeto Led(Indicador) allí se le indica la salida o marca interna del PLC a la cual va a estar asociado.

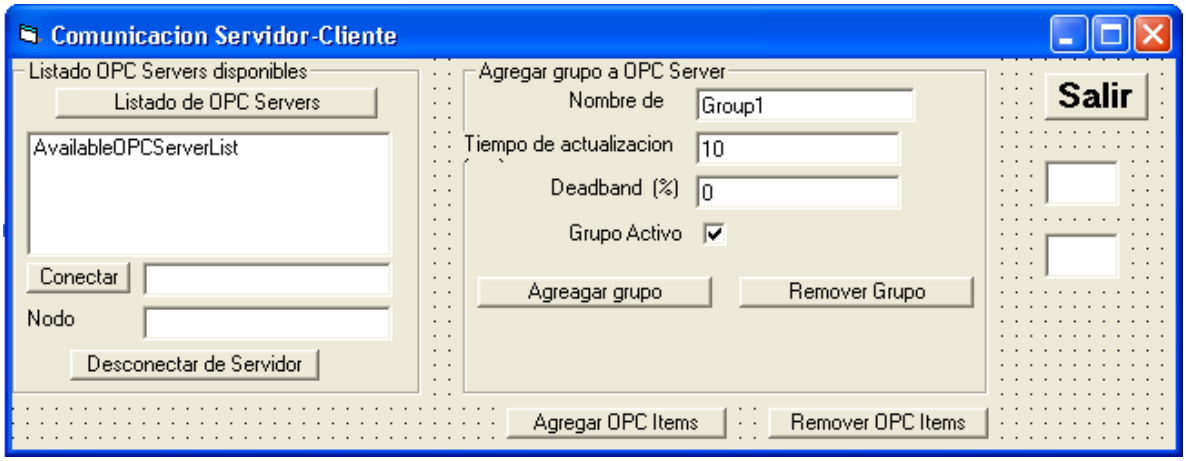

Figura 47. Cuadro de configuración del OPC Server.

La figura 47 muestra el cuadro de configuración del servidor OPC el formulario se llega a través del botón conexión OPC del formulario principal aquí se le indica al programa con que servidor OPC se quiere conectar al PLC dependiendo el numero de servidores OPC instalados en la computadora, además se configura el grupo por defecto se le asigna un nombre de grupo y los Ítems OPC que son las entradas/salidas o marcas internas configuradas en el formulario principal.

XII. CONCLUSIONES

- Un simulador es una herramienta importante en el área académica porque posibilita tener una percepción rápido de las limitaciones y alcances que poseen los diseños que se desarrollan en esa etapa.

- Las interfaces graficas poseen un gran valor porque es la manera con que los humanos se comunican con las maquinas para poder operarlas .

- Los microcontroladores PIC's pueden utilizarse en muchas aplicaciones y los alcances dependen del programador dado que poseen muchos subsistemas y son compatibles con muchas interfaces y protocolos facilitando con esto adaptarlos a nuestras necesidades para este caso se utilizó como interfaz entre computadora-PLC y no hubo necesidad de utilizar más circuitería que la protección que todo circuito electrónico debe poseer.

- OPC permite observar y modificar el comportamiento del PLC en tiempo real, esta característica se puede utilizar para diversas aplicaciones como de supervisión y control de sistemas que tengan elementos que soporten este estándar y permite integrar mucho hardware que no sea del mismo proveedor.

- USB ha venido a facilitar la conexión de periféricos a las computadoras provee una forma sencilla y robusta de comunicar dispositivos se puede adaptar a cualquier necesidad y existen actualmente muchas herramientas para poder programar aplicaciones usando USB.

- LabVIEW es una poderosa herramienta de simulación dado que provee una alta interoperabilidad con otras plataformas de software y hardware

- Visual Basic es el lenguaje de programación que más facilidades provee para desarrollar aplicaciones graficas y es el lenguaje ideal para la construcción de interfaces Humano-Maquina.

XII. REFERENCIAS

[1] Documento sobre comunicaciones industriales.

http://www.disa.bi.ehu.es/spanish/ftp/material_asignaturas/Laboratorio%20de%20comunicaciones%20industriales/Documentaci%F3n/OPC%20conceptos%20fundamentales.pdf

[2] Documento sobre OPC

http://www.monografias.com/trabajos/progeventos/progeventos.shtml

Documento de la programación orientada a eventos

[3] Conceptos sobre Librerías de enlace dinámico

http://es.wikipedia.org/wiki/DLL

[4] Documento de comunicaciones en entornos industriales

http://fing.uncu.edu.ar/catedras/archivos/electronica/tema12r.pdf

[5] Ayuda de LabVIEW versión 8.0

[6] Jiménez, Carlos Fernando. Desarrollando para el puerto USB con la familia 18F2455/2550/4455/4550 y la PICDEM FS USB DEMOSTRATION:

http://usuarios.lycos.es/charlytospage/Articulos/Desarrollando_USB_18F2455_2550_4555_4550_PICDEM_.PDF

X. ANEXOS

Codigo Fuente de formulário principal

```
'Public indice_led As Integer
Dim Ultimo, Nuevo As PictureBox
Dim modo_captura As Boolean
Dim MouseX, MouseY As Single
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim l As Integer
Dim m As Integer
Dim n As Integer
Dim nx As Integer
Dim nn As Integer
Dim p As Integer
Dim oo As Integer
Dim limit As Integer
Dim cont As Integer
Dim suma_arreglo As Integer  'suma de indices maximos de tablas
'Dim arreglo(30) As String
Dim indice_led As Integer
Dim Buffer() As Byte    'tarjeta usb
Dim BufferSize As Long  'tarjeta usb


Private Sub Command1_Click()
```

```
main.MSFlexGrid2.Col = 1
main.MSFlexGrid2.Row = 0
main.MSFlexGrid3.Col = 1
main.MSFlexGrid3.Row = 0
nx = Val(main.MSFlexGrid2.Text)   'numero de salidas creadas
nn = Val(main.MSFlexGrid3.Text)  'numero de entradas creadas
suma_arreglo = nx + nn
'------------abrir opc configurator------------
SimpleOPCInterface.Show
main.MSFlexGrid1.Col = 1
main.MSFlexGrid1.Row = 0
limit = suma_arreglo
limittxt.Text = limit


'Limit = Val(main.MSFlexGrid1.Text)
For k = 1 To nx
main.MSFlexGrid2.Col = 1
main.MSFlexGrid2.Row = k
main.MSFlexGrid1.Col = 1
main.MSFlexGrid1.Row = k
main.MSFlexGrid1.Text = main.MSFlexGrid2.Text
Next k
cont = 1
For k = (nx + 1) To (nx + nn)
main.MSFlexGrid3.Col = 1
```

```
main.MSFlexGrid3.Row = cont
main.MSFlexGrid1.Col = 1
main.MSFlexGrid1.Row = k
main.MSFlexGrid1.Text = main.MSFlexGrid3.Text
cont = cont + 1 'Contador  indice entradas
Next k


'Transferir Nombre de ItemIP a conectar con servidor OPC.
main.MSFlexGrid1.Col = 1
SimpleOPCInterface.MSFlexGrid1.Col = 1
 For j = 0 To limit
    main.MSFlexGrid1.Row = j
    SimpleOPCInterface.MSFlexGrid1.Row = j
    SimpleOPCInterface.MSFlexGrid1.Text = main.MSFlexGrid1.Text
    Next j

'Agregar Marca SMB28.
SimpleOPCInterface.MSFlexGrid1.Col = 1
SimpleOPCInterface.MSFlexGrid1.Row = (limit + 1)
SimpleOPCInterface.MSFlexGrid1.Text = "SMB28"

End Sub


Private Sub control_Click()
m = m + 1
```

```
    Load led(m)
    led(m).Visible = True
    led(m).Left = 3000
    led(m).Top = 2000
End Sub



Private Sub Form_Click()
'Timer1.Enabled = True

HIDComm1.TimeOut = 10

End Sub



Private Sub HIDComm1_ConnectSuccess(ByVal Status As Long)
   main.Visible = True
   Form2.Visible = False
   End Sub
Private Sub Form_Load()
i = 0
j = 0
k = 0
l = 0
m = 0
```

```
n = 0
p = 0
var = 0
'Timer1.Enabled = False
HIDComm1.TimeOut = 10
End Sub

Private Sub Form_Terminate()
HIDComm1.Uninit
End Sub

Private Sub interruptor_Click(Index As Integer)
Dim var As Integer
Dim status_arreglo(29) As Integer
Dim ii As Integer
  txtinterruptor(0).Text = 1
  'var = 1
  'ElseIf var = 1 Then
Else
txtinterruptor(0).Text = 0
'var = 0
End If

'Else
'End If
'codigo 3
```

```
status_arreglo(Index) = Val(txtinterruptor(0).Text)



''Descarga de Datos arreglo -- Tabla Led
 main.MSFlexGrid4.Col = 1

 'For ii = 0 To 29
 main.MSFlexGrid4.Row = Index
 main.MSFlexGrid4.Text = status_arreglo(Index)
 'Next ii



End Sub

Private Sub interruptor_DblClick(Index As Integer)
main.Text2.Text = Index
interruptor_propiedades.Show
End Sub

Private Sub interruptor_MouseDown(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
modo_captura = True
    MouseX = X
    MouseY = Y
End Sub

Private Sub interruptor_MouseMove(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    If modo_captura Then
        interruptor(Index).Left = interruptor(Index).Left - (MouseX - X)
        interruptor(Index).Top = interruptor(Index).Top - (MouseY - Y)
    End If
End Sub

Private Sub interruptor_MouseUp(Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)
    If modo_captura Then
        modo_captura = False
        interruptor(Index).Left = interruptor(Index).Left - (MouseX - X)
        interruptor(Index).Top = interruptor(Index).Top - (MouseY - Y)
    End If
End Sub

Private Sub interruptor1_Click()
n = n + 1
    Load interruptor(n)
    interruptor(n).Visible = True
    interruptor(n).Left = 3000
    interruptor(n).Top = 2000

End Sub

Private Sub led_DblClick(Index As Integer)
main.Text1.Text = Index
```

```
    led_propiedades.Show
    End Sub


    Private Sub led_MouseDown(Index As Integer, Button As Integer, Shift As
Integer, X As Single, Y As Single)
    modo_captura = True
        MouseX = X
        MouseY = Y
    End Sub


    Private Sub led_MouseMove(Index As Integer, Button As Integer, Shift As
Integer, X As Single, Y As Single)
    If modo_captura Then
          led(Index).Left = led(Index).Left - (MouseX - X)
          led(Index).Top = led(Index).Top - (MouseY - Y)
       End If
    End Sub


    Private Sub led_MouseUp(Index As Integer, Button As Integer, Shift As Integer,
X As Single, Y As Single)
    If modo_captura Then
          modo_captura = False
          led(Index).Left = led(Index).Left - (MouseX - X)
          led(Index).Top = led(Index).Top - (MouseY - Y)
       End If
    End Sub


    Private Sub Timer1_Timer()
    If Not HIDComm1.Connected Then
```

```
        'Form1.Visible = False
        Form2.Visible = True
        Form2.Label1.Caption = "Tarjeta desconectada"
        HIDComm1.Connect
        Exit Sub
    End If
End Sub

Private Sub Timer2_Timer()
Dim contador As Integer
contador = 0

main.MSFlexGrid2.Col = 1
main.MSFlexGrid2.Row = 0

For mm = 1 To Val(main.MSFlexGrid2.Text)
main.MSFlexGrid1.Col = 2
main.MSFlexGrid1.Row = mm

If main.MSFlexGrid1.Text = "True" Then
led(mm).BackColor = vbGreen
Else
led(mm).BackColor = vbBlack
End If
Next mm
'-----------------codigo 1-------------------
main.MSFlexGrid3.Col = 1
```

```
main.MSFlexGrid3.Row = 0

For mm = 1 To Val(main.MSFlexGrid3.Text)
main.MSFlexGrid3.Col = 1
main.MSFlexGrid3.Row = mm
main.MSFlexGrid4.Col = 1
main.MSFlexGrid4.Row = mm

var1 = Val(Right(main.MSFlexGrid3.Text, 1))
var2 = Val(main.MSFlexGrid4.Text)
contador = var2 * (2 ^ (var1)) + contador

Next mm

 'enviar dato a tarjeta
 contador = 255 - contador
 ReDim Buffer(16)
    Buffer(0) = CByte(contador)
    Buffer(1) = CByte(0)
    BufferSize = 2
    HIDComm1.WriteTo Buffer, BufferSize



End Sub
```

Codigo Fuente Del formulário Led Propiedades

```
Dim indice_led As Integer
```

```
Dim kk As Integer

Dim arreglo(30) As String

Dim Indice_max As Integer

Dim i As Integer

Dim j As Integer



Private Sub Command1_Click()

arreglo(indice_led) = Text1.Text & "." & Text2.Text

'transferir datos.

'----------------------cargar datos--------------------

''Descarga de Datos arreglo -- Tabla Led

 led_propiedades.MSFlexGrid1.Col = 1

 For i = 0 To 29

 led_propiedades.MSFlexGrid1.Row = i

 led_propiedades.MSFlexGrid1.Text = arreglo(i)

 Next i

'Transferir Datos.
```

```
    main.MSFlexGrid2.Col = 1

    led_propiedades.MSFlexGrid1.Col = 1

    For i = 0 To Indice_max

    main.MSFlexGrid2.Row = i

    led_propiedades.MSFlexGrid1.Row = i

    main.MSFlexGrid2.Text = "Q" & led_propiedades.MSFlexGrid1.Text ' Se
agrega la Q de salida

    Next i

    main.MSFlexGrid2.Col = 1

    main.MSFlexGrid2.Row = 0

    main.MSFlexGrid2.Text = Indice_max



    '-------------------------fin cargar datos-----------------

    Unload Me ' Cerrar Menu de propiedades.

    End Sub

    Private Sub Command2_Click()

    ''Descarga de Datos arreglo -- Tabla Led

     led_propiedades.MSFlexGrid1.Col = 1
```

```
 For i = 0 To 29
 led_propiedades.MSFlexGrid1.Row = i
 led_propiedades.MSFlexGrid1.Text = arreglo(i)
 Next i
'Transferir Datos.
main.MSFlexGrid2.Col = 1
led_propiedades.MSFlexGrid1.Col = 1
For i = 0 To Indice_max
main.MSFlexGrid2.Row = i
led_propiedades.MSFlexGrid1.Row = i
main.MSFlexGrid2.Text = "Q" & led_propiedades.MSFlexGrid1.Text ' Se agrega la Q de salida
Next i
main.MSFlexGrid2.Col = 1
main.MSFlexGrid2.Row = 0
main.MSFlexGrid2.Text = Indice_max
 Unload Me
End Sub
```

```
Private Sub Command3_Click()
'Button de prueba.
End Sub


Private Sub Form_Load()
indice_led = Val(main.Text1.Text)
'Text1.Text = Left$(arreglo(Index), 1)
'Text2.Text = Right$(arreglo(Index), 1)
Text1.Text = Left$(arreglo(indice_led), 2)
Text2.Text = Right$(arreglo(indice_led), 2)
'Text2.Text = arreglo(indice_led)
'Codigo para determinar el número de salidas generadas.
   If indice_led >= Indice_max Then
      Indice_max = indice_led
   Else
      Indice_max = Indice_max
```

```
    End If

    End Sub

Private Sub VScroll1_Change()

Text1.Text = VScroll1.Value

End Sub

Private Sub VScroll2_Change()

Text2.Text = VScroll2.Value

End Sub
```