Universidad Nacional Autónoma de México

Facultad de Ingeniería

Implementación de un sistema SCADA en una red industrial DeviceNet® a través de LabVIEW®

TESIS

Que para obtener el título de:

INGENIERO MECATRÓNICO

PRESENTA

PABLO EDUARDO VEGA CRUZ

DIRECTOR DE TESIS

M.F. GABRIEL HURTADO CHONG

México, D F. Febrero de 2012.

Dedicatorias

A mi abuela, María Eugenia Palacios, que siempre creyó en mí y de la que siempre estaré agradecido, estoy seguro de que en donde quiera que esté está orgullosa de este logro.

A mis padres, Faustino Vega y María de Lourdes Cruz, por darme su apoyo incondicional, sus sabios consejos, su inagotable cariño e increíble comprensión, por estar siempre conmigo y por darme el mejor regalo que he recibido, mi familia.

Agradecimientos

Al M.F. Gabriel Hurtado Chong, por bien aconsejarme y auxiliarme a plantear, realizar y redactar este trabajo.

Al M.I. Serafín Castañeda Cedeño, por su gran ayuda y apoyo en la elaboración de este trabajo.

A mi familia, que siempre estuvo a mi lado y me dio soporte en todo.

A mi novia, que me motivó a siempre seguir mejorando.

A mis amigos, que siempre me apoyaron y tuvieron un consejo para mí.

Índice

Introducción 5
Formulación del problema……7
Objetivo……8
Justificación……8
Alcances……8

Capítulo 1. Marco teórico 10
1.1. Automatización industrial……11
1.2. Redes industriales……16
1.3. LabVIEW®……26
1.4. Sistemas SCADA……31

Capítulo 2. Descripción, características, configuración y conexión del hardware 36
2.1 Red DeviceNet®……37
2.1.1. Controladores lógicos programables……42
2.1.2. Electroválvulas……46
2.1.3. Servomotores……48
2.2 Sistema SCADA……54

Capítulo 3. Configuración, desarrollo e implementación del software 58
3.1. Red DeviceNet®……59
3.1.1. Configuración……59
3.1.2. Comunicación……63
3.1.3. Programación……74
3.2. LabVIEW®……77
3.2.1. Configuración……77
3.2.2. Comunicación……78
3.2.3. Programación……79
3.3. Sistema SCADA……81
3.3.1. Configuración……81
3.3.2. Comunicación……81
3.3.3. Programación……84

Capítulo 4. Resultados obtenidos 86

Capítulo 5. Conclusiones y recomendaciones 94

Referencias 97

Anexos 102
Anexo 1. Programas para el control de los PLCs SLC 500 y Micrologix 1000 y cuadros de distribución……103
Anexo 2. Programa para sistema SCADA en LabVIEW®……111
Anexo 3. Cables y conexiones eviceNet®……132
Anexo 4. Parámetros del servomotor y comando/respuesta de la comunicación……135

Introducción

El trabajo desarrollado tiene como objetivo principal implementar un sistema de control supervisor y de adquisición de datos (SCADA) en una red de comunicación conocida en la industria como DeviceNet®, usando el software de programación gráfica LabVIEW®. Como parte del trabajo se explica la importancia que tiene para una empresa el automatizar los procesos y el usar una red de comunicación entre sus líneas de producción ya que este trabajo se enfoca principalmente en crear un sistema flexible y de gran potencialidad para las industrias que utilicen PLCs (Controlador Lógico Programable), computadoras industriales, sensores y actuadores. Con la propuesta desarrollada se pretende que los estudiantes de ingeniería que optan por la rama de la automatización sean capaces de implementar una red de campo e interfaces Humano-Máquina, las cuales servirán para contribuir en su crecimiento profesional.

Se da una explicación general de las propiedades de la red DeviceNet®, el uso de LabVIEW® y las características que debe cumplir un sistema para poder ser llamado SCADA. Gran parte de los conocimientos utilizados son consecuencia de la implementación de la red, sin embargo, para la programación del SCADA se debe de conocer a nivel intermedio la programación de PLCs Allen Bradley® y la programación básica en LabVIEW®.

El sistema SCADA implementado integra dispositivos que interactúan en hardware y software, es por eso que el desarrollo en cada aspecto se encuentra en un capítulo diferente.

La conexión, configuración y características físicas de los elementos de la red son explicadas de tal forma que se pueda realizar una instalación después de leído el capítulo correspondiente al hardware, respecto a la implementación del software en la que se aborda la configuración, comunicación y programación, se hizo una división temática en 3 etapas, la primera consta en arrancar completamente la red industrial DeviceNet® de forma independiente y realizar una programación en los PLCs que demuestre el funcionamiento, la segunda se encarga de comunicar una computadora con un dispositivo de la red y se realiza un ejemplo en LabVIEW® que demuestra el funcionamiento; en la última etapa se tiene como tarea unir las dos etapas anteriores las cuales no son compatibles si son unidas tal y como se desarrollaron en su momento, es decir, se realiza una reconfiguración y programación de los PLCs y en LabVIEW® para crear el SCADA.

El sistema SCADA completo es el resultado final del trabajo y se explica con detalle una vez que se terminan las etapas de implementación, también se aportan recomendaciones que tienen como propósito general el instruir ingenieros con mayor conocimiento de los problemas que se enfrentan en la vida laboral y aportar una herramienta la cual pueda ser utilizada posteriormente en la automatización de la industria.

Formulación del problema

La necesidad de automatizar un sistema de producción o de servicios, se vuelve cada vez más importante para un eficiente y pleno desarrollo tecnológico, económico y empresarial. El beneficio que tiene un sistema automatizado en comparación con una flotilla de empleados, es muchas veces superior, empezando por saber que la máquina jamás hará una huelga, creará sindicatos, se quejará por lo mal que se le trata y tampoco cobrará las horas extras; además que en algunos casos hacen más rápido y certero el trabajo.

A pesar de que el desarrollo industrial es una maravilla con las máquinas, no todo está a su favor ya que no todos los procesos industriales pueden ser automatizados, necesitan de mantenimiento que quizá pueda ser comparable con los sueldos de algunos meses, se necesita de expertos que puedan manejar y supervisar su operación y sobre todo el costo por adquisición es sumamente elevado comparado con el costo inicial por contratar a una persona.

Las empresas de alto nivel económico optan por la inversión y adquieren sistemas automatizados, en la mayor parte de los casos, los sistemas que compran no son genéricos, sino se plantea a un equipo de especialistas el proceso o servicio que se quiere automatizar para que ellos generen propuestas de solución, este tipo de contratación puede llegar a ser el óptimo ya que el sistema a adquirir garantiza que estará dedicado especialmente para el cliente; claramente esto también conlleva un costo aún más elevado.

Actualmente, empresas de mediano y alto nivel económico, están buscando la forma de automatizar sus procesos, algunas de ellas acuden a empresas de automatización de renombre, centros de diseño especializado y a universidades, todo esto con el afán de encontrar la solución más económica y funcional.

Los alumnos de las universidades de México, y particularmente de la Universidad Nacional Autónoma de México, tienen la capacidad intelectual, creativa e innovadora suficiente para competir a nivel industrial, es por eso que se necesita estar preparados y conocer los sistemas que se utilizan fuera de las aulas de clase.

El desarrollo del alumno de ingeniería debería de estar íntimamente ligado con el desarrollo de las nuevas tecnologías, es decir, el crecimiento debe de ser al mismo tiempo, de tal forma que un ingeniero egresado conozca todo tipo de nuevas tecnologías o inclusive sea capaz de crearlas.

La ingeniería mecatrónica es una de las carreras que en su principio promovió la automatización industrial con base en brazos robóticos, sistemas de visión, electroneumática y software de alto nivel para desarrollo; con el paso del tiempo se le ha encontrado otros usos como la biomédica y la instrumentación.

El problema con el enfoque industrial es que no se vincula de gran manera al alumno con la industria, aunque sea cierto que en la universidad se dan las herramientas teóricas para poder resolver cualquier tipo de problema de ingeniería mecatrónica, sería de gran ayuda para los alumnos el poder conocer más sistemas industriales e integrar las tecnologías emergentes a las tecnologías existentes.

Objetivo

Conectar, configurar, comunicar y programar un sistema de control supervisor y adquisición de datos (SCADA) en una red de comunicación industrial usando nuevas tecnologías de software.

Justificación

El alumno de ingeniería mecatrónica de la UNAM compite con alumnos de diversas instituciones públicas y privadas en México, es por eso que sería recomendable no dar ventaja a los demás competidores, la principal misión de la Facultad de ingeniería de la UNAM es, forjar ingenieros competitivos con habilidades, actitudes y valores que ayuden al crecimiento, progreso y fortalecimiento de la universidad y del país [1], sin embargo, con la inversión en estos y muchos otros sistemas industriales, se puede ser líder en el conocimiento de las nuevas tecnologías, lo cual favorecería a los alumnos que deseen especializarse en la industria de la automatización.

Esta tesis da a conocer un sistema de uso común en la industria y la integración de un sistema de software de última tecnología. La unión sinérgica de estos dos sistemas da como resultado una innovación en la Facultad de Ingeniería de la UNAM y un nuevo sistema que es sustentable en la industria.

La posibilidad de conseguir trabajo aumenta en gran medida para los estudiantes de mecatrónica que sepan usar estos dos sistemas, ya que cualquier empresa en el área de la automatización optaría por un ingeniero que además de PLC's, neumática y electroneumática, domine y conozca sistemas industriales que se usan en la vida laboral como lo son las redes y los SCADA.

Alcances

El trabajo escrito en esta tesis cubre todo el entorno de conexión, configuración, programación y comunicación de la red industrial DeviceNet®.

El sistema SCADA implementado en el trabajo no es un caso real, es sólo un ejemplo planteado por el autor, si se quiere trasladar a un sistema físico es posible que no funcione tan eficientemente como la emulación debido a que no se consideran todos los parámetros del sistema, como la viscosidad de los fluidos, fricciones, inercias, entre otras. Para poder medir y considerar los parámetros restantes es necesario contar con sensores apropiados y modelar los sistemas críticos con base en los requerimientos del interesado.

El trabajo escrito aporta una guía rápida de cómo implementar la red de comunicación industrial y cómo programar la interfaz humano-máquina con el software LabVIEW®. Si no se conoce la programación básica con LabVIEW® y la programación avanzada con diagramas de escalera, no será de mucha utilidad el conocimiento aportado, el trabajo es altamente recomendado para alumnos y egresados de ingeniería mecatrónica, mecánica y electrónica.

Se aportan recomendaciones al usuario de la red a lo largo de la conexión, configuración, programación y comunicación, esto no asegura que sean los únicos problemas a los que se enfrentará, para mejor referencia de los problemas suscitados se recomienda consultar los manuales de los dispositivos utilizados.

Capítulo 1. Marco teórico

1.1 Automatización industrial

Para poder definir de una forma más certera a la automatización industrial, es necesario definir en primera instancia la palabra automatización, algunas definiciones son por ejemplo, "*Conversión de determinados procesos corporales o psíquicos en automáticos o involuntarios*" *[2]*, "*Es una amplia variedad de sistemas y procesos que operan con mínima o incluso nula intervención del ser humano*" *[3]*, "*Es la tecnología que trata de la aplicación de sistemas mecánicos, electrónicos y de bases computacionales para operar y controlar la producción*"[4], "*Aplicación de procedimientos automáticos a un aparato, proceso o sistema*" [5]. Cada individuo podría emitir un juicio de lo que significa la automatización, sin embargo, definiendo a la automatización más como un estado de operación humana y no de una máquina podría definir que "*Es una serie de eventos secuenciales y repetitivos sin uso de razón o conciencia*".

Todas de las definiciones obtenidas y aportadas podrían ser correctas ya que están enfocadas desde diferentes puntos de vista, es por eso que se debe acotar la definición con base en la segunda palabra de interés. La palabra industria está definida por la Real Academia Española como "*conjunto de operaciones materiales ejecutadas para la obtención, transformación o transporte de uno o varios productos naturales*". Sería vano pensar que la industria sólo realiza operaciones con materiales tangibles, la industria puede realizar prestación de servicios en la que no influye ningún recurso tangible; un ejemplo es la industria de las telecomunicaciones, donde el único flujo existente es el de información con base en la energía eléctrica.

La mejor definición para automatización industrial es la que mejor describe el enfoque deseado, el resultado de la suma sinérgica de las definiciones anteriores puede ser la siguiente.

Automatización industrial, es una variedad de sistemas y procesos mecánico-electrónicos programados con el fin de obtener, transformar o transportar productos, con la mínima intervención del ser humano.

Antes de realizar una inspección a lo que puede o pudo ser la definición se debe entender que cualquier definición es correcta si la vemos con el enfoque adecuado, esto no incluye que la opción elegida es la única y verdadera definición pero si es la que satisface el objetivo al que se quiere llegar.

La automatización ha sido usada desde más de 1500 años antes de cristo con objetivos de supervivencia más que de negocio, el claro ejemplo son las trampas de caza. Estaban dotadas de un elemento, a modo de trinquete primitivo, que sujetaba la trampa abierta. Cuando algún animal pisaba encima, el trinquete dejaba de sujetar la trampa, cerrándola y atrapando al animal. Acercándose al siglo III antes de cristo un famoso griego llamado Ktesibios creó un reloj de agua y postuló los primeros tratados del aire comprimido y sus usos, Ktesibios es conocido como el padre de la neumática, la cual es la técnica de producir o transformar la energía del aire comprimido en movimiento.

La neumática por mucho tiempo fue la técnica más usada para producir movimiento a partir de la energía almacenada en el aire comprimido, incluso después de haberse inventado la energía eléctrica, la neumática seguía siendo la técnica de automatizado más usada por las industrias mundiales.

En 1750 en pleno crecimiento de la revolución industrial, nace el término "automatización", el primer dispositivo aceptado como automatizado fue una máquina tejedora con tarjetas perforadas y se empiezan a crear los primeros circuitos neumáticos.

Los circuitos neumáticos están constituidos por actuadores que efectúan el trabajo y por aquellos elementos de señalización y de mando que gobiernan el paso del aire comprimido y, por lo tanto, la manipulación de los elementos de trabajo, estos dispositivos se denominan de manera genérica válvulas.

Para el tratamiento de la información de mando es preciso emplear aparatos que controlen y dirijan el fluido de forma preestablecida, lo que obliga a disponer de una serie de elementos que efectúen las funciones deseadas relativas al control y dirección del flujo del aire comprimido. Estos elementos tienen como finalidad mandar o regular la puesta en marcha o el paro del sistema, el sentido del flujo, así como la presión o el caudal del fluido procedente del depósito regulador.

Si se clasifican según su función, las válvulas se subdividen en los grupos siguientes:

- Válvulas de vías o distribuidoras. Estas válvulas son los componentes que determinan el camino que ha de seguir el aire en cada momento, manipulando el sentido de desplazamiento de los actuadores. Trabajan en dos o más posiciones fijas determinadas. En principio, no pueden trabajar en posiciones intermedias.
- Válvulas de bloqueo. Estas son válvulas destinadas a impedir, condicionar o dificultar el paso del flujo en uno u otro sentido.
- Válvulas de presión. Estas válvulas influyen principalmente sobre la presión, o están condicionadas por el valor que tome aquélla.
- Válvulas de caudal y de cierre. Estas válvulas tienen como finalidad regular el caudal que las atraviesa y con ello controlar la velocidad de los vástagos de los cilindros. Estas válvulas lo que producen es una pérdida de carga y ésta conduce a reducir el caudal.

Las válvulas de distribución pueden ser elegidas por las posiciones y el número de vías, las posiciones son la cantidad de conexiones posibles que puede hacer la válvula y las vías son el número de orificios que tiene la válvula.

Las válvulas monoestables, son válvulas que mantienen una sola posición en su estado de reposo, las válvulas biestables son válvulas que mantienen 2 posiciones por sí mismas.

Las válvulas en términos generales, tienen las siguientes misiones:

- Distribuir el fluido
- Regular caudal
- Regular presión

En resumen, las válvulas son elementos que mandan o regulan la puesta en marcha, el paro y la dirección, así como la presión o el caudal del fluido enviado por el compresor o almacenado en un depósito. Ésta es la definición de la norma DIN/ISO 1219 conforme a una recomendación del CETOP (Comité Europeo de Transmisión Oleohidráulica y Neumática).

Para finales de 1835 el estadounidense Joseph Henry inventaba un dispositivo electromecánico que funcionaba como interruptor controlado por un circuito eléctrico en el que, por medio de un electroimán se accionaban uno o varios contactos, a este dispositivo se le denominó relevador.

Las tecnologías para conversión de energía eléctrica a mecánica eran ya dominadas por la mayor parte de los científicos del siglo XIX. La gran eficiencia de los relevadores para generar movimiento mecánico a partir de energía eléctrica y la necesidad de nuevos pilotajes para las válvulas neumáticas, crearon lo que hoy conocemos como electroválvula.

Los accionamientos conocidos para la neumática eran sólo los mecánicos, ya sea por pedal, palanca o botones, sin embargo, la creación de la electroválvula daría un vuelco a la automatización. La electroválvula es un dispositivo de control que utiliza el principio de un relevador, el impulso mecánico provocado por el relevador ahora no cierra un switch eléctrico, sino un conducto para la apertura o cierre del flujo de un fluido.

En términos de automatización y tratando de dar una definición más formal, una electroválvula es un válvula la cual está accionada por un elemento eléctrico (solenoide).

Las electroválvulas revolucionaron la industria de la automatización, sin embargo, el control de los procesos estaba basado en bancos de relevadores interconectados lógicamente, este método de automatización era muy poco flexible ya que si se deseaba cambiar algún parámetro o función del proceso, era necesario cambiar todo el banco de relevadores y dejar el proceso parado hasta que se tuviera el nuevo proceso funcionando correctamente.

Poco a poco se fueron inventando nuevas formas de evitar perder el tiempo en las reconexiones de los relevadores, una de ellas era tener el banco de relevadores preparado para dejar fuera de servicio el viejo e inmediatamente conectar el nuevo. El mantenimiento de estos equipos era altamente costoso, si alguno de los relevadores en operación fallaba, era necesario dejar el proceso parado para poder analizar cuál pudiera ser el relevador que falló.

En busca de nuevas tecnologías más eficientes y menos costosas para la producción, a finales de la década de 1960, se crean los controladores lógicos programables (PLC), estos sistemas estaban basados en el mismo concepto de los circuitos combinacionales que usaban relevadores, de esta forma los programadores de los circuitos con relevadores no tendrían ningún problema para programar esta nueva tecnología.

El lenguaje de programación para los controladores lógicos fue conocido como lenguaje de escalera o diagramas de escalera. A pesar del tiempo que ha transcurrido desde la invención de este lenguaje, sigue siendo el más usado para autómatas programables.

Hoy en día, los procesos de casi todas las plantas industriales en el mundo son controlados por PLCs, que a diferencia con los primeros dispositivos creados, tienen gran capacidad de almacenamiento de información, pueden tener un alto número de salidas y entradas, son capaces de manipular variables continuas y discretas, y su mantenimiento y reposición es menos costoso para las empresas.

Los PLCs de nueva generación, contienen funciones especializadas para un desempeño óptimo de los procesos, por ejemplo la creación de perfiles de velocidad, controladores PID y contadores de alta velocidad, para este tipo de funciones los actuadores pueden ser motores y/o servomotores.

Los servomotores son motores que pueden tener un desplazamiento rotatorio o lineal y son capaces de informar a un servosistema la posición, la velocidad, la dirección en la que se encuentra su eje, y dependiendo del servosistema algunas veces la corriente de consumo, de esta forma se puede controlar el movimiento que se quiere producir con alta resolución, precisión y exactitud.

Un servomotor está compuesto por 5 elementos principales:

- Motor, es un elemento que generalmente es de corriente directa y es el encargado de convertir la energía eléctrica en mecánica.

- Mecanismo de engranes reductores, es el elemento que transforma la alta velocidad y bajo par del motor en baja velocidad y alto par.
- Sensor de posición, este elemento se encarga de medir la posición del eje del motor, regularmente se usa un codificador rotatorio (encoder) o un potenciómetro.
- Sensor de corriente, este elemento tiene la tarea de medir la corriente que se consume en el motor una vez que está operando, usualmente se utilizan sensores de efecto hall.
- Circuito de control, es el encargado de que la señal de entrada al circuito de control sea igual a la medida por el sensor de posición.

Los sistemas automatizados actuales cuentan con sistemas de control, adquisición y procesamiento de datos muy avanzados, la tecnología integrada en estos sistemas cada vez es mejor, integran dispositivos de alto nivel tecnológico como las electroválvulas, servomotores, variadores de frecuencia, autómatas programables y máquinas de control numérico.

1.2 Redes industriales

En una empresa industrial existen muchos sistemas automatizados, incluso hay más de un sistema automatizado para cada proceso que se ejecuta, muchas de las veces los procesos deben de estar mutuamente comunicados o al menos debe de existir algún sistema que los regule y supervise individualmente y que reporte a otro de un nivel jerárquico mayor.

Para tener una mejor organización de los sistemas que se están automatizando en una empresa se creó un diagrama con 4 jerarquías de organización, cada una tiene tareas, prioridades y responsabilidades diferentes, a este diagrama de jerarquías se le denominó pirámide de la automatización [6].

La pirámide de automatización está construida como se muestra en la Figura 1.1.

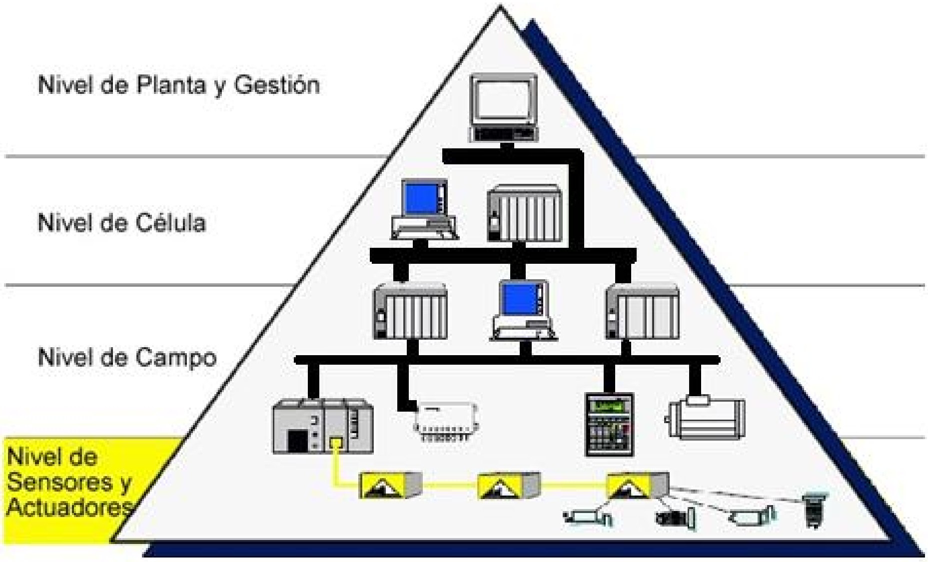


Figura 1.1. Pirámide de la automatización.
Fuente: http://www.uhu.es/antonio.barragan/book/export/html/125.

En el nivel inferior y como base de todo el diagrama, se muestra el nivel de sensores y actuadores, en este nivel se encuentran todos los dispositivos que operan directamente con el proceso, es decir, que son los sistemas que repercuten directamente en su funcionamiento, en este nivel están contenidos elementos tales como pistones, motores, variadores de frecuencia, electroválvulas, sensores de proximidad, sensores de nivel, controles musculares como botones, pedales y palancas, por mencionar algunos. La información en este nivel es a nivel de bits.

El segundo nivel es el de campo, en este nivel están todos los elementos que tienen el control de los dispositivos del nivel inferior, para su adecuado funcionamiento se debe asegurar que el nivel de sensores y actuadores funciona debidamente, en caso contrario, todo lo programado perjudicará a la planta. Los elementos que están contenidos en este nivel son PLCs, computadoras industriales y escáneres que tienen los automatismos específicos de cada una de las máquinas controladas. La información de este nivel es tratada por medio de bytes.

El nivel de célula es el tercer bloque de la pirámide, aquí se encuentran los dispositivos que gestionan a los controladores o autómatas que están inmersos en el nivel de campo. Los elementos que se encuentran en este nivel son uno o varios autómatas modulares de gran potencia. La información en este nivel es manejada por medio de paquetes de bytes.

El cuarto nivel de la pirámide es el más alto de los sistemas automáticos y se denomina nivel de planta y gestión. Está formado por una computadora que se encarga de la gestión total de la producción de la fábrica, es en este nivel donde se pueden modificar los parámetros de la planta. Este nivel puede tener un servidor en el cual una persona capacitada y con los permisos necesarios accede desde cualquier lugar sólo con el uso del internet.

Si se tienen múltiples procesos y se quiere gestionar toda la planta, el número de dispositivos en los dos primeros niveles incrementa, y en los dos últimos puede llegar a ser el mismo pero se necesitaría de más capacidad, además si la planta es de grandes dimensiones los procesos pueden estar muy lejos, lo cual haría que el gasto por cablear de manera convencional elevara el costo.

Por ejemplo, si se necesita que la salida analógica de un PLC que está controlando un proceso sea leída por otro PLC que está controlando otro proceso a 100 metros de distancia, se necesitaría gastar 100 metros de cable para poder comunicar la salida del primer PLC con la entrada del segundo, sin contar que se desperdiciaría una entrada y una salida física, las cuales podrían haber sido usadas por otro dispositivo. Para este tipo de eventos y problemas se crearon las redes industriales, 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 el control de procesos donde todos los instrumentos integrados en la planta puedan comunicarse en una misma plataforma creando un sistema uniforme, limpio y menos costoso.

Las comunicaciones entre los instrumentos de los procesos y el sistema de control se basan principalmente en señales analógicas. Existen instrumentos digitales capaces de manejar gran cantidad de datos y guardarlos históricamente; su resolución podría llegar a ser igual a la de una señal analógica. En vez de transmitir cada variable por un par de hilos para cada dispositivo, transmiten secuencialmente las variables por medio de un cable de comunicaciones llamado bus.

La arquitectura fieldbus conecta estos instrumentos con computadoras que se usan en diferentes niveles de coordinación y dirección de la planta.

Físicamente podemos considerar a un bus como un conjunto de conductores que conectan varios circuitos para permitir el intercambio de datos. Contrario a una conexión punto a punto donde sólo dos dispositivos intercambian información, un ejemplo de esta comunicación es la serial con protocolo RS-232, un bus consta normalmente de un número de usuarios superior, además que generalmente un bus transmite datos en modo serial, a excepción de algún protocolo de bus particular como SCSI o IEEE-488, utilizado para interconexión de instrumentos de medición, que no es el caso de los buses tratados como buses de campo.

Para una transmisión serial es suficiente un número de cables muy limitado, generalmente dos o tres conductores y la debida protección contra las perturbaciones externas para permitir su tendido en ambientes de ruido industrial.

Si una empresa pretende ser líder en su sector, necesita empezar por estar a la vanguardia e innovar en sus procesos, es sumamente problemático tener canaletas con grandes cantidades de cables por toda la planta, además que quita limpieza y estética, es importante dar una buena impresión ya que no sólo servirá para las certificaciones, también es más seguro y fácil de darle mantenimiento. En la figura 1.2 se muestra en un tablero industrial la diferencia entre un sistema con conexiones normales y uno con bus de campo.

a) b)

Figura 1.2 instalación en un tablero industrial, a) conexión normal b) conexiones con bus de campo.

Fuente: Sistema industriales distribuidos, Universidad de Valencia.

Un bus de campo tiene muchas propiedades favorables, algunas son que el intercambio puede llevarse a cabo por medio de un mecanismo estándar, conexión de módulos diferentes en una misma línea, distancias de operación superiores al cableado tradicional, simplificación de la puesta en marcha, reducción excesiva de cables y costos que estos producían.

A pesar de que todo parece estar resuelto con el uso de las redes de campo, también se pueden mencionar algunas desventajas como la necesidad de conocimientos superiores, la inversión de instrumentación y accesorios de diagnóstico y costos totales inicialmente superiores a los de un sistema típico.

El costo inicial por la instrumentación y sistemas de diagnóstico es parte de las inversiones que una empresa está dispuesta a hacer si el futuro de sus procesos está asegurado, no sólo por el control total de la planta, sino por el mantenimiento que a largo plazo es menor.

Existen muchos tipos de buses y la mayoría trabajan con interfaz RS 485, unas con interfaz RS-232 y otras más nuevas con USB, FireWire o Serial ATA. En grandes redes industriales, un simple cable no es suficiente para conectar el conjunto de todos los nodos de la red. Deben definirse topologías y diseños de redes para proveer un aislamiento y conocer los requerimientos de funcionamiento. Los elementos usados con más frecuencia o típicos en una red de campo son:

Bridge, es un puente de la conexión entre dos diferentes secciones de red, puede tener diferentes características eléctricas y protocolos; además puede enlazar dos redes diferentes.

Repetidor, es un dispositivo que intensifica las señales eléctricas para que puedan viajar grandes distancias entre nodos. Con este dispositivo se pueden conectar los nodos a la red a distancias muy grandes; además, se pueden adaptar a diferentes medios físicos como cable coaxial o fibra óptica.

Gateway, es similar a un puente, ya que suministra interoperabilidad entre buses y diferentes tipos de protocolos; además, las aplicaciones pueden comunicarse a través de él.

Enrutadores, son switches de paquetes de comunicación entre diferentes segmentos de red que definen la ruta hacia donde se transmite la información.

En resumen podríamos mencionar los siguientes beneficios de los buses de campo

- Simplificación de cableado
- Control distribuido
- Reducción del costo producido por las cajas de conexión y cables.
- Aplicable a todo tipo de sistema de manufactura

- Incremento de la confiabilidad de los sistemas de producción
- Optimización de los procesos existentes
- Reducción del mantenimiento

Los buses de campo son la mejor opción para el control de los procesos y seguramente seguirán siendo la mejor opción, sin embargo, no son la única forma de hacer una red industrial. Las soluciones basadas en Ethernet se están utilizando cada vez más en el sector de las tecnologías de automatización, donde las secuencias de procesos y producción son controladas por un modelo cliente/servidor con controladores, PLC y sistemas ERP (Planificación de los recursos de la empresa), teniendo acceso a cada sensor que se conecta a la red.

Si regresáramos a la pirámide de la automatización de la figura 1.1 podríamos observar que en el nivel de campo se usa un cable delgado para conectar sobre un solo cable a todos los dispositivos y para el nivel de célula y el nivel planta y gestión se usa un cable más grueso, la diferencia de grosores no es para hacer el diagrama llamativo, el cable delgado representa una red industrial usando un bus de campo y el cable grueso representa una red industrial usando EtherNet®.

DeviceNet®

La gran mayoría de los protocolos patentados para los buses de campo tienen una limitante y es que el fabricante no permite al usuario final la interoperabilidad de instrumentos, es decir, no es posible intercambiar los instrumentos de un fabricante por otro similar. Es claro que estas tecnologías cerradas tienden a desaparecer, ya que actualmente es necesaria la interoperabilidad de sistemas y aparatos y así tener la capacidad de manejar sistemas abiertos y estandarizados. Con la mejora de los protocolos de comunicación es ahora posible reducir el tiempo necesario para la transferencia de datos, asegurando la misma, garantizando el tiempo de sincronización, el tiempo real de respuesta determinística en algunas aplicaciones y sobre todo el costo de implementación.

La red DeviceNet® es una red digital multipunto para conectar sensores actuadores y sistemas de automatización en general, la red fue desarrollada para una máxima flexibilidad y operatividad entre equipos de campo de diferentes compañías.

Introducida originalmente en 1994 por la compañía de automatización estado unidense Allen Bradley® la cual transfirió su tecnología a la ODVA (Asociación Abierta de Vendedores DeviceNet®) en 1995. Es una organización sin fines de lucro creada por cientos de compañías de todo el mundo que mantienen, promueven y difunden la red DeviceNet® y redes basadas en CIP (Protocolo Industrial Común), actualmente 300 compañías son miembros registrados y más de 800 productos DeviceNet® son ofrecidos en todo el mundo.

La red DeviceNet® está clasificada como una red de dispositivos, cuyas características son: alta velocidad, comunicación a nivel de byte que incluye equipo de comunicaciones analógico y diagnóstico de alta potencia para los dispositivos [7].

En la figura 1.3 se muestra la tendencia de red y el espectro que abarca en los diferentes niveles de la automatización.

La tecnología DeviceNet® es un estándar abierto de automatización con el objetivo de transportar dos tipos de información.

- Datos cíclicos de sensores y actuadores directamente relacionados con el control y
- Los datos no cíclicos indirectamente relacionados con el control, como la configuración y el diagnóstico

Los datos cíclicos representan la información intercambiada periódicamente entre los equipos de campo y los controladores. Por otro lado, los datos no cíclicos son información intercambiada eventualmente durante la configuración o los diagnósticos del equipo de campo.

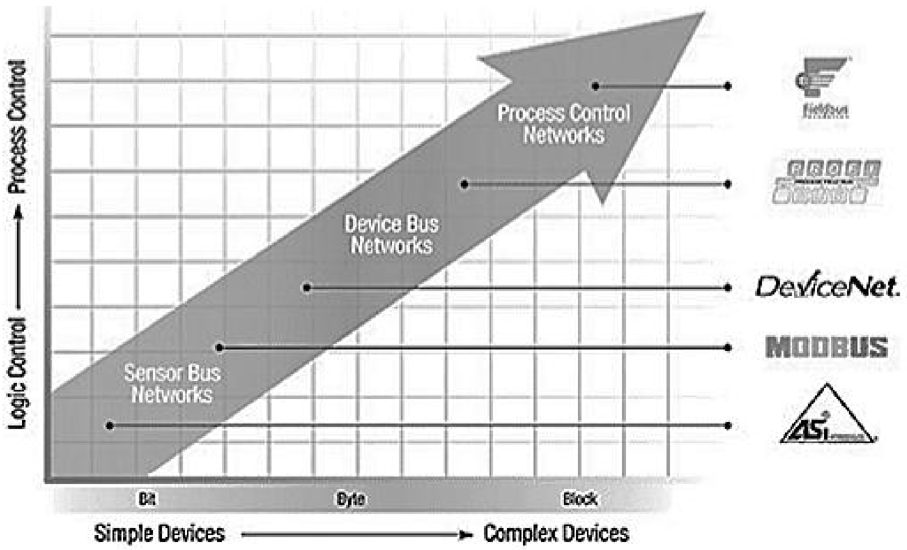


Figura 1.3 Tendencias tecnológicas.
Fuente: *ATAIDE, F.H. (2004).*

Los tipos de comunicación en el protocolo de la red DeviceNet® son dos: mensajes E/S cíclicos y los no cíclicos o mejor conocidos como mensajes explícitos. Ambos son usados para tipos particulares de datos.

Cyclic I/O: tipo de telegrama síncrono dedicado a desplazamiento de datos prioritarios entre un productor y uno o más consumidores. Se dividen de acuerdo con el método de cambio de datos. Los principales son:

- Polled: método de comunicación en que el maestro envía un telegrama a cada uno de los esclavos de su lista (scan list). Así que recibe la solicitud el esclavo y responde prontamente al maestro. Este proceso es repetido hasta que todos sean consultados, es decir, reiniciando el ciclo.
- Bit-strobe: método de comunicación de donde el maestro envía para la red un telegrama que contenie 8 bytes de datos. Cada bit de estos bytes representan un esclavo que responde de acuerdo a su programa, si es que está direccionado.
- Change of state: método de comunicación donde el cambio de datos entre maestro y esclavo ocurre apenas cuando existe un cambio en los valores monitoreados o controlados hasta un cierto límite de tiempo. Cuando este límite es alcanzado, la transmisión y recepción solicitarán la información, incluso sin alteración. La configuración de esta variable de tiempo es hecha en el programa de configuración de red.
- Cyclic: otro método de comunicación muy parecido al anterior. La única diferencia queda por cuenta de la producción y consumo de mensajes. En este tipo, todo cambio de datos ocurre en intervalos regulares de tiempo, independientemente de haber sido alterados o no. Este periodo también es ajustado en el software de configuración de red.

Explicit message: tipo de telegrama de uso general y no prioritario. Utilizado principalmente en tareas asíncronas tales como parametrización y configuración del equipamiento.

El protocolo de la trama de datos de la red es igual a la del protocolo CAN (Controller Area Network), parte del ancho de banda del protocolo es usado para transmitir mensajes CIP. En la figura 1.4 se muestra el formato de la trama de datos.

Un bit es usado para empezar la trama de datos, 11 bits crean el identificador del dispositivo origen, 1 bit para la inicialización de un frame remoto, 6 bits para el campo de control, 8 bytes para transmitir datos, 15 bits para detectar la secuencia del código de detección de error, 1 bit para delimitar el código de detección de error, 2 bits que no son usados y casi siempre son ceros lógicos, 7 bits para terminar la transmisión de datos y 3 para que vuelva a entrar otra trama.

1bit	11 bits	1 bit	6 bits	0-8 bytes	15 bits	1 bit	1 bit	1 bit	7 bits	3 bits
Start of Frame	Identifier	RTR bit	Control Field	Data Field (0...8 bytes)	CRC Sequence	CRC Delimiter	Ack Slot	Ack Delimiter	End of Frame	Interframe Space
Arbitration Field										

Figura 1.4 Formato de la trama de datos.
Fuente: http://www.smar.com/devicenet.asp.

La capa física y el acceso a la red DeviceNet® están basados en la tecnología CAN y las capas superiores con el protocolo CIP, la cual define una arquitectura basada en objetos y la conexión entre ellos.

El protocolo CAN fue desarrollado originalmente por BOSCH para el Mercado automovilístico europeo, sustituyendo el costoso cableado por una red de bajo costo para coches. Como resultado el protocolo CAN provee una rápida y fiable respuesta para aplicaciones principalmente usadas en el área automotriz [8].

Una red DeviceNet® puede tener hasta 64 dispositivos con un dispositivo en cada nodo direccionados de 0 a 63. Cualquiera de ellos puede ser usado, no hay ninguna restricción para su uso, aunque el nodo 63 no es recomendado ya que es usado típicamente para arrancar la red.

En la figura 1.5 se muestra un ejemplo de una red DeviceNet®.

A continuación se enlistan las características y capacidades de esta red.

- Topología basada en una línea troncal o línea principal (trunkline) con derivaciones (droplines).

  La trunkline básica usa solo un cable, los droplines son múltiples segmentos de cable y deben de estar separados no más de lo establecido por la velocidad de comunicación. En la figura 1.6 se muestra una topología con trunk y dropline.

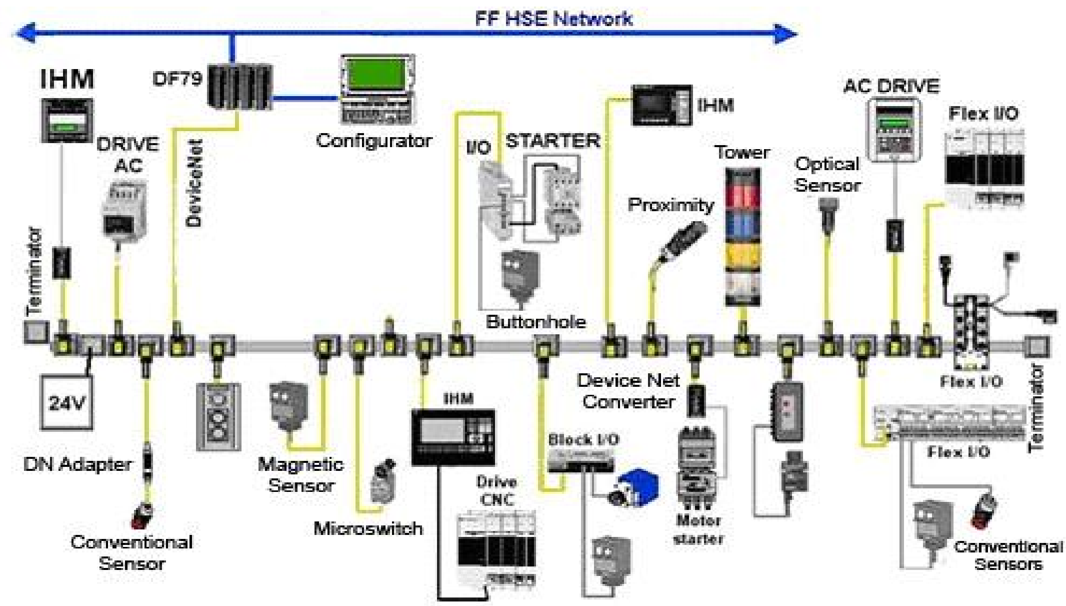


Figura 1.5 Red DeviceNet®.
Fuente: http://www.smar.com/devicenet.asp.

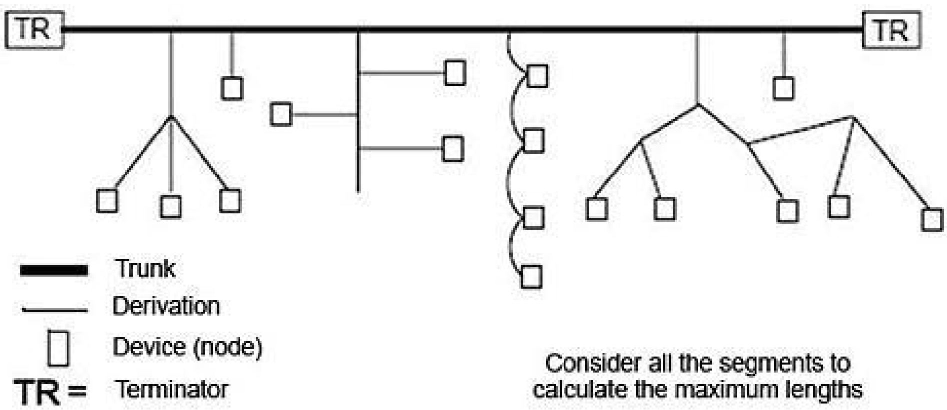


Figura 1.6 Topologia con trunkline y dropline.
Fuente: http://www.smar.com/devicenet.asp.

En topología usual de la red DeviceNet® existen dos elementos que no han sido mencionados y que son de suma importancia en la red, los terminadores son resistencias de 121 Ohm con 1% de tolerancia y un cuarto de watt. Sin el uso de estas resistencias en los extremos de la red, no se garantiza una buena comunicación en la red.

- La trunkline debe estar hecha de un cable grueso de DeviceNet® y las droplines con un cable delgado.

Hay 4 tipos de cables estándar, grueso, medio, delgado y plano. Para mejor referencia del tipo de cables se recomienda leer el anexo 3.

- Se permite usar repetidores, bridges, routers y gateways.
- Soporta hasta 64 nodos incluyendo el maestro (MAC ID del 0 al 63).
- Doble par de cable uno para la línea de voltaje de 24 V y otro para comunicación.
- Inserción y remoción de dispositivos en pleno proceso sin causar disturbios a la red.
- Soporta la energización de equipos a 24 V o permite dispositivos que tengan su propia fuente. La fuente de alimentación de la red debe de ser de 24 V en corriente directa.
- Utiliza conectores abiertos o cerrados
- Protección contra la inversión de conexiones y cortos circuitos.
- Capacidad de corriente de hasta 16 A.
- Muchas fuentes pueden ser usadas en la misma red para satisfacer las necesidades de carga y longitud de cable.
- Velocidad de comunicación seleccionable de 125, 250 y 500 kbps.

Esta velocidad depende de la relación plasmada en el cuadro 1.1.

Velocidad de datos	125 Kbps	250 Kbps	500 Kbps
Distancia del bus principal con el cable grueso	500 m	250 m	100 m
Distancia del bus principal con el cable delgado	100 m	100 m	100 m
Distancia máxima entre la línea principal y la derivación más larga	6 m	6 m	6 m
Distancia acumulada desde la línea principal hasta las derivaciones	156 m	78 m	39 m

Cuadro 1.1 Velocidades de comunicación.
Fuente: http://www.smar.com/devicenet.asp.

- Diagnóstico para cada equipo y para la red en general.
- Eficiente transporte de datos de control digital y analógico.
- Detección de nodos duplicados.
- Mecanismo de comunicación extremadamente robusto a la interferencia magnética.

1.3 LabVIEW®

LabVIEW® constituye un revolucionario sistema de programación gráfica para aplicaciones que involucren adquisición, control, análisis y presentación de datos.

El lenguaje que usa se llama lenguaje G, donde la G simboliza que es lenguaje totalmente gráfico.

Este programa fue creado por National Instruments®, salió al mercado por primera vez en 1986 y ahora está disponible para las plataformas Windows, UNIX, MAC y GNU/Linux en su última versión 2011.

Los programas desarrollados con LabVIEW® se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control, sino también a su programación embebida. Un lema tradicional de LabVIEW® es: *"La potencia está en el Software"*, que con la aparición de los sistemas multinúcleo se ha hecho aún más potente. Un sistema multinúcleo es un microprocesador que combina dos o más procesadores independientes en un solo paquete, a menudo un solo circuito integrado.

Entre los objetivos de este poderoso software está el reducir el tiempo de desarrollo de aplicaciones (prueba, instrumentación, control y diseño) y permitir la entrada a la informática a profesionales de cualquier otro campo. LabVIEW® consigue combinarse con todo tipo de software y hardware del propio fabricante, como tarjetas de adquisición de datos, sistemas de visión, FPGAs (Field Programmable Gate Array, Campo Matricial de Compuertas Programables), PLCs y otros dispositivos de hardware creados por diferentes fabricantes.

Las ventajas que proporciona el empleo de LabVIEW® se podrían resumir en las siguientes:

- Se reduce el tiempo de desarrollo de las aplicaciones al menos de 4 a 10 veces, ya que es muy intuitivo y fácil de aprender.
- Dota de gran flexibilidad al sistema, permitiendo cambios y actualizaciones tanto del hardware como del software.
- Da la posibilidad a los usuarios de crear soluciones tan completas y complejas como se necesiten.
- Con un único sistema de desarrollo se integran las funciones de adquisición, análisis y presentación de datos.
- El sistema está dotado de un compilador gráfico para lograr la máxima velocidad de ejecución posible.
- Tiene la posibilidad de incorporar aplicaciones escritas en otros lenguajes, como C, C++ y C#.

LabVIEW® es un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan el *lenguaje C* o *BASIC*. Sin embargo, LabVIEW® se diferencia de dichos programas en un importante aspecto: los citados lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW® emplea la programación gráfica para crear programas basados en diagramas de bloques.

Para el uso de LabVIEW® no se requiere gran experiencia en programación, ya que se emplean iconos, términos e ideas familiares a científicos e ingenieros, y se apoya sobre símbolos gráficos en lugar de lenguaje escrito para construir las aplicaciones. Por ello resulta mucho más intuitivo que el resto de lenguajes de programación convencionales.

Posee extensas bibliotecas de funciones y subrutinas. Además de las funciones básicas de todo lenguaje de programación, LabVIEW® incluye bibliotecas específicas para la adquisición de datos, control de instrumentación a través de plataformas VXI (VMEbus eXtensions for Instrumentation) o el bus de datos GPIB (Bus de Instrumentación de Propósito General), comunicación serie, análisis, presentación y almacenamiento de datos. También proporciona potentes herramientas que facilitan la depuración de los programas.

LabVIEW® tiene una de las comunidades más grandes de programadores, lo cual ayuda a consultar, adquirir y aprender programas con nuevos enfoques.

Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs.

Todos los VIs tienen un panel frontal y un diagrama de bloques.

En el panel frontal se colocan todos los sistemas de entrada y salida, este panel es el que interactuará con el usuario una vez que se esté ejecutando el programa, es decir, se trata de la interfaz gráfica del VI con el usuario. Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el diagrama de bloques. Está conformado por una serie de botones, pulsadores, potenciómetros, gráficos, etc.

En la ventana de diagrama de bloques se colocan los bloques que harán las operaciones con las entradas del panel frontal. El diagrama de bloques constituye el código fuente del VI. En el diagrama de bloques es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesamiento de las entradas y salidas que se crearon en el panel frontal.

Incluye funciones y estructuras integradas en las bibliotecas que incorpora LabVIEW®. En el lenguaje G las funciones y las estructuras son nodos elementales. Son análogas a los operadores o bibliotecas de funciones de los lenguajes convencionales.

Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante terminales de lectura o control.

El diagrama de bloques se construye conectando los distintos objetos entre sí, como si se tratara de un circuito electrónico. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos, cada cable tiene diferente color y apariencia, esto depende del tipo de dato, se pueden manejar datos convencionales como binarios, enteros, caracteres, arreglos y hasta clusters.

LabVIEW® posee una extensa biblioteca de funciones, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, análisis de señales, etc. Las estructuras son similares a los usados en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle for, while o case).

Para poder crear entradas, salidas, bloques y conexiones se necesita de la paleta de herramientas, la paleta de controles y la paleta de funciones.

La paleta de herramientas (*Tools palette*) se emplea tanto en el panel frontal como en el diagrama de bloques. Contiene las herramientas necesarias para editar y depurar los objetos tanto del panel frontal como del diagrama de bloques.

Las opciones más destacadas de esta paleta son las de cambiar el valor de los controles, agregar texto como etiqueta, cablear los bloques, cambiar el color del fondo, crear puntos de prueba, etc.

La paleta de controles (*Controls palette*) se utiliza únicamente en el panel frontal, contiene todos los controles e indicadores que se emplearán para crear la interfaz del VI con el usuario. El menú *Controls* de la ventana correspondiente al panel frontal contiene opciones como la de crear controles booleanos (leds, pulsadores, palancas), numéricos (perillas analógicas, arreglos), agrega cajas para inserción o despliegue de texto, gráficas, tablas y decoraciones.

Al seleccionar objetos desde el menú *Controls* estos aparecen sobre el panel frontal, pueden colocarse donde convenga, y además tienen su propio menú desplegable que permite la configuración de algunos parámetros específicos de cada tipo de control. En la figura 1.7 se puede mostrar el panel frontal, las opciones principales de la paleta de control y la paleta de herramientas.

La paleta de funciones (*functions palette*) se emplea en el diseño del diagrama de bloques. La paleta de funciones contiene todos los objetos que se emplean en la implementación del programa del VI, ya sean funciones aritméticas, de entrada/salida de señales, entrada/salida de datos en un archivo, adquisición de señales, temporización de la ejecución del programa, estructuras case, while o for, bloques de análisis de señales, comparaciones, entre muchas más. En la figura 1.8 se puede mostrar el diagrama de bloques, algunas de las opciones de la paleta de funciones y la paleta de herramientas.

La programación en LabVIEW® es una de las nuevas enseñanzas para los ingenieros mecatrónicos de la UNAM, sus grandes alcances y poderes tecnológicos ayudan a sus estudiantes a tener una ventaja tecnológica y competitiva. Además de instruirlos en sistemas de desarrollo que son ocupados hasta en las mejores empresas de la industria mundial.

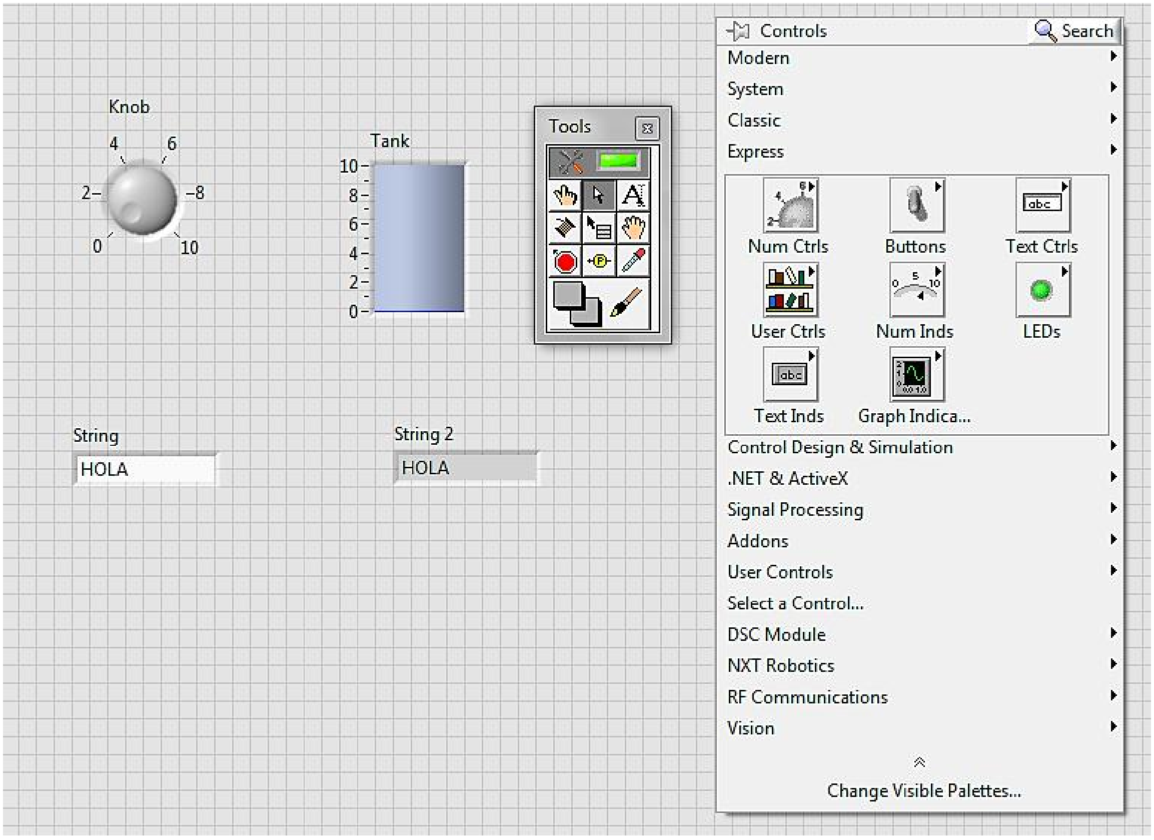


Figura 1.7. Panel frontal, paleta de control y paleta de herramientas.

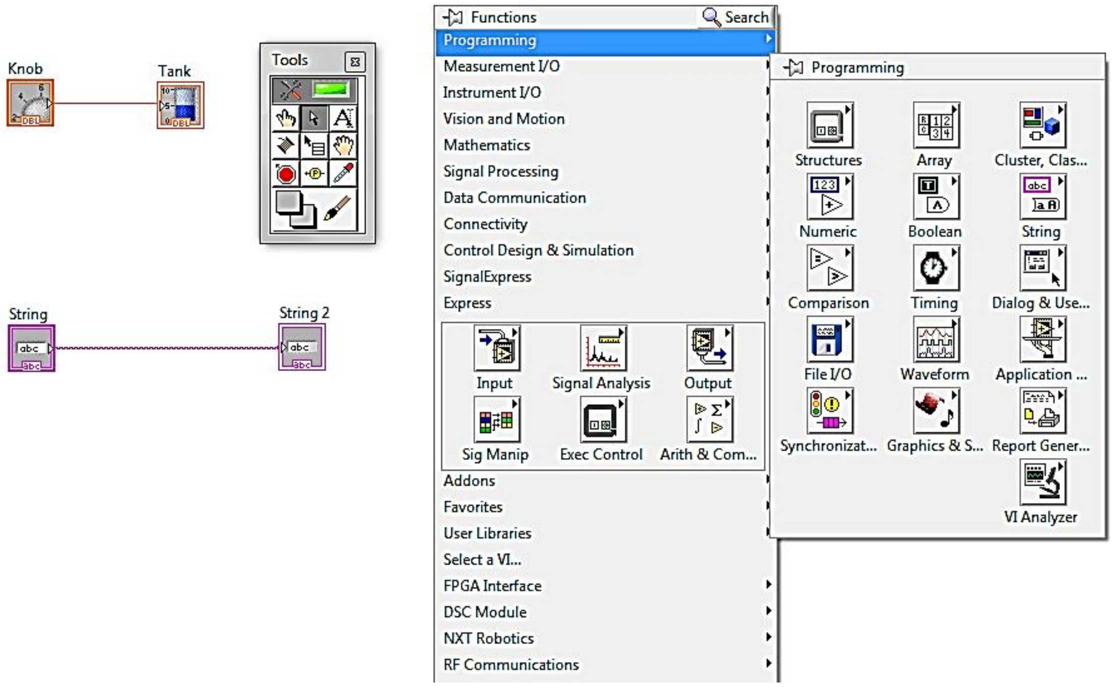


Figura 1.8. Diagrama de bloques, paleta de funciones y paleta de herramientas.

1.4 Sistemas SCADA

Los sistemas SCADA (*Supervisory Control And Data Adquisition*) son aplicaciones de software, diseñadas con la finalidad de controlar y supervisar procesos a distancia. Se basan en la adquisición de datos de los procesos remotos.

Es una plataforma de software, especialmente diseñada para funcionar sobre computadoras en el control de la producción, proporcionando comunicación con los dispositivos de campo como controladores autónomos, autómatas programables y escáneres de red, además de estar controlando el proceso de forma automática desde una computadora. Envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, que permite la participación de otras áreas como por ejemplo: control de calidad, supervisión, mantenimiento y toda aquella área vinculada con el proceso.

Cada uno de los items de SCADA involucran muchos subsistemas, por ejemplo, la adquisición de los datos puede estar a cargo de un controlador lógico programable, el cual toma las señales y las envía a las estaciones remotas usando un protocolo determinado, otra forma podría ser que una computadora realice la adquisición de datos y luego esa información la transmita hacia un equipo, pueden existir muchas alternativas para implementación de SCADAs.

Las tareas de supervisión y control generalmente están más relacionadas con el software SCADA, porque en él, el operador puede visualizar en la pantalla de la computadora, cada una de las estaciones remotas que conforman el sistema, los estados de éste, las situaciones de alarma y tomar acciones físicas sobre algún equipo lejano. La comunicación se realiza mediante buses especiales o redes LAN (Red de Área Local). Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos.

Estos sistemas actúan sobre los dispositivos instalados en la planta, como son los controladores, autómatas, sensores, actuadores, registradores, etc. Además permiten controlar el proceso desde una estación remota, para ello el software brinda una interfaz gráfica que muestra el comportamiento del proceso.

Generalmente se vincula el software al uso de una computadora o de un PLC, la acción de control es realizada por los controladores de campo, pero la comunicación del sistema con el operador es necesariamente vía computadora. Sin embargo, el operador puede gobernar el proceso en un momento dado si es necesario.

Un software SCADA debe ser capaz de ofrecer al sistema:

- Posibilidad de crear paneles de alarma, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.
- Generación de datos históricos de las señales de planta, que pueden ser volcados para su proceso sobre una hoja de cálculo.
- Ejecución de programas, que modifican la ley de control, o incluso anular o modificar las tareas asociadas al autómata, bajo ciertas condiciones.
- Posibilidad de programación numérica, que permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador.

Existen diversos tipos de sistemas SCADA dependiendo del fabricante y sobre todo de la finalidad con que se va a hacer uso del sistema, por ello antes de decidir cuál es el más adecuado hay que tener presente si cumple o no ciertos requisitos básicos:

- Todo sistema debe tener arquitectura abierta, es decir, debe permitir su crecimiento y expansión, así como deben poder adecuarse a las necesidades futuras del proceso y de la planta.
- La programación e instalación no debe presentar mayor dificultad, debe contar con interfaces gráficas que muestren un esquema básico y real del proceso.
- Deben permitir la adquisición de datos de todo equipo, así como la comunicación a nivel interno y externo (redes locales y de gestión)
- Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables para el usuario.

Además de los requerimientos específicos del sistema, también hay funciones principales que el sistema debe cumplir:

- Supervisión remota de instalaciones y equipos: Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.
- Control remoto de instalaciones y equipos: Mediante el sistema se puede activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, prender motores, etc.), de manera automática y también manual. Además es posible ajustar parámetros, valores de referencia y algoritmos de control.
- Procesamiento de datos: El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada, y comparada con datos anteriores, y con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.

- Visualización gráfica dinámica: El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente dentro de una planta real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.
- Generación de reportes: El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.
- Representación de señales de alarma: A través de las señales de alarma se logra alertar al operador frente a una falla o la presencia de una condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.
- Almacenamiento de información histórica: Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o del autor del programa.
- Programación de eventos: Está referido a la posibilidad de programar subprogramas que brinden automáticamente reportes, estadísticas, gráfica de curvas, activación de tareas automáticas, etc.

La información de todos los sistemas SCADA debe transmitirse vía red, puertos GPIB, telefónica o satélite, es necesario contar con computadoras remotas que realicen el envío de datos hacia una computadora central, ésta a su vez será parte de un centro de control y gestión de información.

Para realizar el intercambio de datos entre los dispositivos de campo y la estación central de control y gestión, se requiere un medio de comunicación, existen diversos medios que pueden ser cableados (cable coaxial, fibra óptica, cable telefónico) o no cableados (microondas, ondas de radio, comunicación satelital).

Cada fabricante de equipos para sistemas SCADA emplea diferentes protocolos de comunicación y no existe un estándar para la estructura de los mensajes. Un protocolo de comunicación es un conjunto de reglas y procedimientos que permite a las unidades remotas y central, el intercambio de información.

El sistema SCADA puede ser tan complejo como el programador o el usuario lo requieran, es claro que el mejor sistema aplicado es el que cumpla con las necesidades del proceso y no el más complejo.

En la figura 1.9 se puede observar una interfaz de usuario de un sistema SCADA.

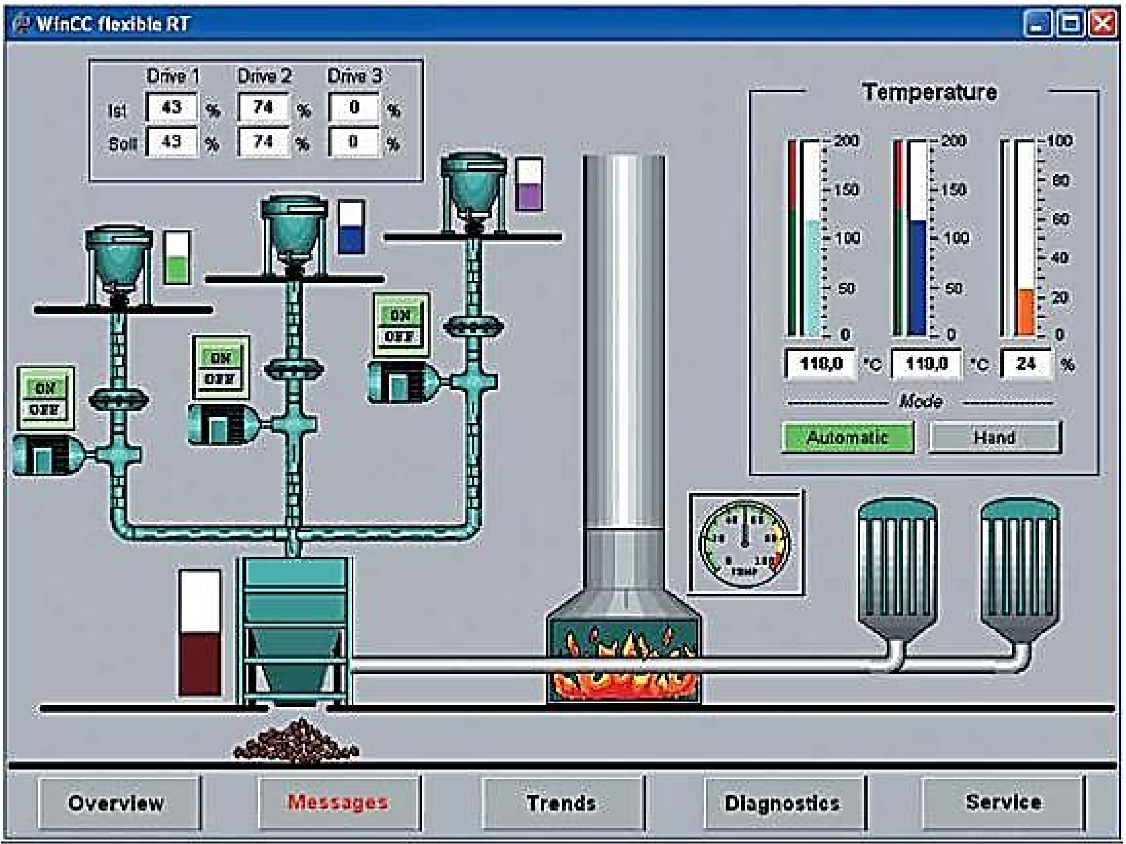


Figura 1.9. Interfaz de usuario para control de un proceso usando el software de programación WinCC de SIEMENS.

Fuente: http://www.plm-engineering.com/english/competences/operating-automation/index.php

Los requerimientos para la implementación de un sistema SCADA no sólo son de programación, también son físicos, los elementos físicos mínimos del sistema son:

- Pantalla para interfaz del Operador: Es el entorno visual que brinda el sistema para que el operador se adapte al proceso desarrollado por la planta. Permite la interacción del ser humano con los medios tecnológicos implementados.
- Unidad Central (MTU): Conocido como Unidad Maestra. Ejecuta las acciones de mando (programadas) en base a los valores actuales de las variables medidas. La programación se realiza por medio de bloques de programa en lenguaje de alto nivel (como C, Basic, LabVIEW®, etc.). También se encarga del almacenamiento y proceso ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.
- Unidad Remota (RTU): Lo constituye todo elemento que envía algún tipo de información a la unidad central. Es parte del proceso productivo y necesariamente se encuentra ubicada en la planta.

- Sistema de Comunicaciones: Se encarga de la transferencia de información del punto donde se realizan las operaciones, hasta el punto donde se supervisa y controla el proceso. Lo conforman los transmisores, receptores y medios de comunicación.
- Transductores: Son los elementos que permiten la conversión de una señal física en una señal eléctrica y viceversa. Su calibración es muy importante para que no haya problema con los datos interpretados.

Capítulo 2. Descripción, características, configuración y conexión del hardware

2.1 Red DeviceNet®

El hardware de la red DeviceNet® está compuesto en esencia de una trunkline, múltiples droplines, nodos, fuente(s) de alimentación y terminadores. A pesar de que estos componentes son los mínimos para poder conformar la red, existen otros dispositivos que sirven para poder gestionarla y ponerla en marcha eficientemente. La selección y uso de estos dispositivos dependerán del diseñador de la red así como del objetivo para el que será implementada la red.

En el laboratorio de automatización industrial de la Facultad de Ingeniería de la UNAM, existen productos que son para uso exclusivo de la red DeviceNet®, estos dispositivos tienen objetivos específicos y diferentes capacidades. La configuración y disposición se eligió después de conocer sus características, configuración y conexión.

Fuente de poder

Fuente de alimentación con salida regulada a 24 Volts en directa y una capacidad de corriente de hasta 20 Amperes para poder alimentar la red completamente. La tensión de entrada es de 120/240 Volts en alterna a 4/2 Amperes respectivamente, la carcasa es de aluminio con el propósito de disipar el calor generado por la fuente y en caso de conexiones erróneas es capaz de proteger la red y cortar el flujo de corriente. La fuente es mostrada en la figura 2.1.

Figura 2.1. Fuente de poder.

Power-Tap

Es un derivador de Allen Bradley® que ayudó a conectar la fuente de alimentación y la trunkline de la red DeviceNet® para conseguir una conexión más segura, este derivador tiene un circuito de protección contra sobre-corriente y un diodo para poder utilizar diferentes Power Taps en cada una de las fuentes de alimentación conectadas. Como entrada existe un conector para la línea a 24 Volts proveniente de la fuente de alimentación y dos salidas usadas para la alimentación completa de la red. El derivador se muestra en la figura 2.2, en la zona izquierda de la imagen está la entrada proveniente de la fuente y a la derecha las 2 salidas directas a la red.

Figura 2.2. Power-Tap DeviceNet®.

T-port

Es un puerto T de Allen Bradley® donde se conectó un módulo de válvulas, este puerto T es una derivación de la trunkline, además que hace la reducción física de micro-sealed a mini- sealed, en la figura 2.3 se muestra el puerto de entrada en la zona izquierda de la imagen (micro- sealed) y la derivación al módulo de electroválvulas en la zona inferior (mini- sealed), para mejor referencia en los conectores ver el anexo 3.

Figura 2.3. T-port DeviceNet®.

Terminador

Es una resistencia terminal de Allen Bradley® que se utilizó tal y como lo especifican los requerimientos de la red DeviceNet®, esta resistencia es de 121 ohm con 1% de error, es necesario colocarla al inicio y final de la red para tener una mayor estabilidad dentro de los conductores de comunicación, tiene una sola conexión micro-sealed. En la figura 2.4 se muestra un terminador el cual se recomienda instalar en una zona visible y de fácil acceso para posteriores mantenimientos.

Figura 2.4. Terminador DeviceNet®.

1770 KFD

Es una interfaz para la gestión de arranque de Allen Bradley®, este dispositivo permitió el acceso y la configuración de la red mediante una computadora, en el mercado es conocido como módulo de interfaz RS-232 1770 KFD. Las conexiones de comunicación son 2, la primera es la del conector DeviceNet® que va directamente a la red formando un nodo y la segunda es un conector macho para puerto RS-232 el cual va conectado a la computadora. El módulo tiene la opción de conectarse a una fuente de 9V a 1 A o conmutar la alimentación y hacerlo a través del mismo cable de la DeviceNet®.

El módulo contiene 3 leds indicadores que informan del estado actual de la red.

En la figura 2.5 se observa la parte frontal del módulo, los tres leds indicadores, la conexión de la red en la zona superior de la imagen y la conexión RS-232 en la zona inferior.

Figura 2.5. Módulo 1770-KFD.

Network Status. Led que informa del estado actual de la DeviceNet®.

- Verde parpadeante. La red está activa pero no está transmitiendo datos.
- Verde constante. La red se encuentra transmitiendo o recibiendo datos.
- Rojo constante. El módulo detectó una incongruencia en la red, como una duplicidad de nodos o error en la velocidad de comunicación.

Module Status. Led encargado de representar el estado actual del módulo.

- Verde parpadeante. El módulo está en línea pero no ha encontrado la velocidad de comunicación correcta.
- Verde constante. El módulo está en línea y la velocidad de comunicación es correcta.
- Rojo parpadeante. El módulo detectó una falla como un corto circuito, el módulo debe de ser reseteado.
- Rojo constante. El módulo está en una falla irrecuperable, el módulo ya no puede operar.

RS-232 Status. Led encargado de Informar de la comunicación a través de la computadora.

- Verde parpadeante. El módulo está transfiriendo o recibiendo datos de la computadora.
- Rojo parpadeante. El módulo detectó una falla como velocidad de comunicación y debe de mandarse la información otra vez.
- Rojo constante. El módulo está en una falla general y es incapaz de comunicarse, se recomienda reiniciar la conexión general del módulo.

1761 DNI

Es un módulo de interfaz Allen Bradley® con el cual, a través de un puerto RS-232 se permite conectar los PLCs a la red, los PLCs por sí solos no pueden acceder completamente a la red ya que no cuentan con entradas físicas DeviceNet®. Con esta interfaz el PLC además de comunicarse con otros sistemas, adquiere la capacidad de ser programado por la computadora sin necesidad de hacer alguna conexión extra. En el mercado es conocido como interfaz 1761 NET DNI, tiene 3 leds indicadores y operan similar a la interfaz de gestión de arranque.

Network Status o NET. Informa del estado actual de la red.

- Verde parpadeante. La red está activa pero no le transmite datos al módulo.
- Verde constante. La red se encuentra transmitiendo o recibiendo datos del módulo.

- Rojo parpadeante. El módulo detectó corto circuito en la red, es necesario verificar conexiones.
- Rojo constante. El módulo tiene un error irreparable, se recomienda remplazarlo.

Module Status o MOD. Encargado de representar el estado real del módulo.

- Verde parpadeante. El módulo está en línea pero no ha encontrado la velocidad de comunicación correcta.
- Verde constante. El módulo está en línea y la velocidad de comunicación es correcta.
- Rojo parpadeante. El módulo detectó una falla como un corto circuito, el módulo debe de ser reiniciado.
- Rojo constante. El módulo tiene una falla irrecuperable, el módulo ya no puede operar.

RS-232 Status. Led que indica que la comunicación RS-232 está activa

- Verde parpadeante. El módulo está transfiriendo o recibiendo datos de la red.

Las conexiones físicas de este dispositivo son 2, la primera es para conectarse directamente en la red y alimentar de energía al módulo y la segunda es para conectarse al PLC mediante un cable convertidor RS-232.

En la figura 2.6 se muestra una de las dos interfaces 1761 DNI existentes, el conector macho de la red se encuentra en la zona superior del módulo y la conexión RS-232 en la zona inferior con el cable convertidor (1761-CBL-AM00) conectado al PLC Micrologix 1000 (en el caso del SLC500 se conecta a través del cable 1761-CBL-PM02).

Figura 2.6 Módulo de interfaz 1761 NET DNI empotrado en la estación de trabajo del PLC al que interpretará.

2.1.1. Controladores lógicos programables

Hasta este momento se han descrito 6 dispositivos que conforman la red, sin embargo, ninguno de ellos puedes realizar una tarea por sí solo y tampoco genera un propósito a la red. Los elementos realmente necesarios en las redes de campo son los PLCs ya que su autonomía y facilidad de soportar ambientes severos es culminante para su uso en los procesos.

PLC modular SLC500

Controlador lógico programable modular con capacidad de 4 slots y una fuente de poder incluida con salidas en tensión directa a 24 y 5 Volts hasta 0.46 y 2 Amperes respectivamente. El PLC modular se muestra en la figura 2.7.

Sólo cuenta con una entrada de tensión monofásica 120 Volts o bifásica a 240 Volts provenientes de la acometida.

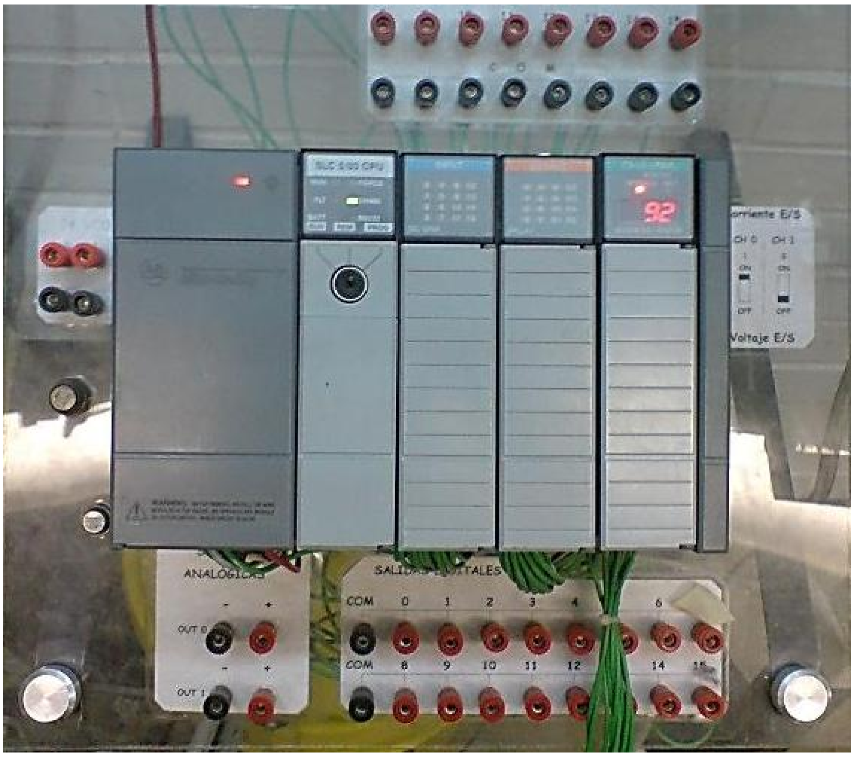


Figura 2.7. PLC SLC 500 montado en su estación de trabajo con todos los módulos conectados.

Los slots contenidos en el chasis del PLC SLC500 están distribuidos de la siguiente forma

Slot 0

Procesador SLC 5/03 con capacidad de memoria de 8000 palabras, control de 4096 entradas/salidas discretas, 480 entradas/salidas analógicas y tiempo de escaneo cada 1 ms.

Obligatoriamente el procesador debe de estar ubicado en el primer slot, independientemente de la cantidad de módulos o la capacidad que pueda procesar.

Las conexiones de este módulo son dos, independientemente de la alimentación y comunicación la cual es interna a través de un conector universal (backplane) que se encuentra dentro del chasis SLC500 y es conectado al momento de introducir el módulo al chasis.

La primer entrada es la conexión para la red DH-485 y la segunda es para comunicación RS-232 la cual es usada típicamente para la programación del PLC y en este caso es usada para conectarse a la interfaz 1761 NET-DNI.

Slot 1

Módulo de entradas digitales 1746-IB16 con capacidad de 16 puntos de acceso binario a 24 Volts de tensión directa y hasta 0.008 Amperes, las entradas son para configuraciones *sinking*.

Slot 2

Módulo de salidas digitales 1746-OW16 con capacidad de 16 puntos de salida binaria, las salidas son a relevador, es decir, se puede conectar tensión alterna o directa sin problemas. Para la tensión alterna con conexiones de 120 o 240 Volts, la capacidad de corriente es de 2.5 Amperes y para la conexión en tensión directa de 24 o 125 Volts, la capacidad de corriente es de 2 Amperes y 1 Amper respectivamente.

Slot 3

Módulo 1747 SDN Scanner Allen Bradley® para la red DeviceNet®, este dispositivo se encarga de consultar y editar los parámetros de cada uno de los dispositivos que sean sus esclavos dentro de la red, puede trabajar como dispositivo maestro o esclavo. Se alimenta con la energía que proporciona el chasis SLC 500 a 24 V (backplane). El escáner tiene un único conector macho para la conexión directa a la red, sin embargo, esto no significa que el PLC pueda acceder directamente a la red, para que esto suceda debe de conectarse al slot 0 la interfaz 1761 NET-DNI.

El escáner es capaz de comunicarse con el procesador del SLC500 y con los dispositivos de campo que estén operando dentro de la red, si el SLC500 desea comunicarse con los dispositivos de campo que estén a cargo del escáner, tiene que dar la orden al escáner para que éste interprete la información a los dispositivos de campo. A pesar de que los PLCs puedan tener una interfaz 1761 NET-DNI que les permita enviar datos a la red, éstos no son capaces de manipular u obtener datos de los dispositivos de campo por sí solos, únicamente pueden compartir y editar datos entre PLCs.

El módulo escáner facilita su interpretación utilizando 2 leds indicadores y un doble display de 7 segmentos para notificar al usuario los errores ocurridos durante la operación, arranque y paro.

Module Status Indicator. Led encargado de informar al usuario el estado actual del módulo de escaneo.

- Verde parpadeante. El módulo no está configurado.
- Verde constante. El módulo se encuentra operando normalmente.
- Rojo parpadeante. Existe una configuración inválida, se recomienda verificar la configuración.
- Rojo constante. El módulo tiene una falla irrecuperable necesita reemplazarse.

Network Status Indicator. Led asignado a informar el estado actual del nodo en la red.

- Verde parpadeante. Indica que se debe checar el código de error desplegado en el display para solucionar el problema ocurrido.
- Verde constante. El módulo está operando normalmente.
- Rojo parpadeante. Indica que un error grave ha pasado y que es necesario revisar el código desplegado en el display.
- Rojo constante. El canal de comunicación falló y es necesario revisar el código desplegado en el display para corregir el problema.

Address/Error. Doble display de 7 segmentos que indica alternadamente la dirección de los nodos problemáticos en la red y el código de error que están cometiendo. En caso de que se cometa un error en los nodos, sólo se identifica el error de los nodos que estén registrados como esclavos en la memoria del escáner.

En la figura 2.8 se muestra el módulo desinstalado del chasis.

Figura 2.8. 1747-SDN Scanner DeviceNet®.

Micrologix 1000 series E

Controlador lógico programable Allen Bradley® de 1000 palabras de memoria, capacidad de 20 entradas digitales a 24 Volts de directa y 0.0025 Amperes, 12 salidas digitales a relevador con capacidad máxima de corriente de 2 Amperes a 240 Volts de alterna, 2.5 Amperes a 125 Volts de directa y 1 Ampere a 24 volts de directa.

La alimentación de este PLC es de 120 Volts de alterna y una demanda de corriente de 0.250 Amperes provenientes de la acometida.

La única conexión independiente de las entradas y salidas digitales, es la asignada para programación y comunicación con protocolo RS-232 configurada en un conector especial para PLCs Allen Bradley®.

El PLC micrologix 1000 serie E tiene la posibilidad de comunicarse con otros dispositivos ya sea como maestro o cómo esclavo a través de redes como DH+ (Data Highway), DH-485 y Devicenet®.

En la figura 2.9 se muestra el micrologix 1000 conectado mediante un cable (1761-CBL-AM00) RS-232 a una interfaz 1761 del lado izquierdo de la imagen.

Figura 2.9 Micrologix 1000 Series E.

2.1.2. Electroválvulas

Los elementos de control y gestión pueden dar un propósito a la red, sin embargo, se debe de conocer los dispositivos de campo que serán utilizados. Para la tarea de campo el laboratorio cuenta con 2 módulos de electroválvulas Parker® de la red Devicenet®.

Módulos de electroválvulas monoestables y biestables P2M2HBVD21600

Módulo de electroválvulas Parker con direccionamiento simple (monoestable) o doble (biestable) con capacidad de 16 válvulas simples u 8 dobles, presiones de hasta 10 bares, alimentación a 24 Volts de tensión directa y consumo de 1.5 Watts para el módulo y alimentación a 24 Volts de tensión directa y 0.24 Amperes de corriente en los solenoides.

El módulo contiene una entrada (BUS IN) para la red DeviceNet® y una salida (BUS OUT) para conectar a otro módulo de electroválvulas. El cable que se conecta al bus de entrada es el mini-sealed proveniente de la reducción y derivación del T-port, este cable sirve únicamente para comunicarse y alimentar el módulo.

Además de las conexiones para comunicación y alimentación del módulo existe una conexión (POWER) que sirve para alimentación de los solenoides la cual es independiente de la alimentación del módulo, todo esto con el propósito de aislar los sistemas de comunicación del de potencia.

En la figura 2.10 se muestra la disposición esquemática de los conectores del módulo, en la imagen se nota que sólo se comparte el cable de tierra física. La tensión de las conexiones BUS IN, BUS OUT y POWER es a 24 Volts, por lo que puede usarse la misma fuente de alimentación; si la fuente de alimentación tiene un abastecimiento limitado de corriente es recomendable usar una fuente independiente para la conexión POWER de los solenoides.

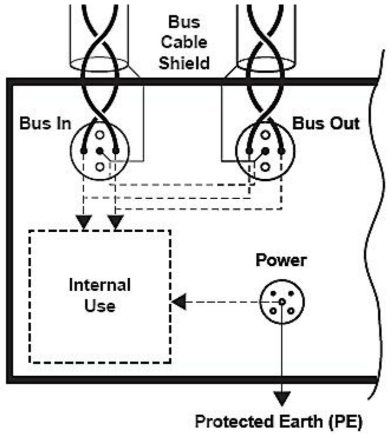


Figura 2.10. Disposición de las conexiones del módulo.

El módulo de válvulas es un sistema fijo y con pocas posibilidades de corrupción mediante software, tal muestra de ello es, que el módulo debe de ser configurado manualmente según las características y descripciones de su panel de interfaz con leds y dials.

En la figura 2.11 se muestra la distribución esquemática real del panel de las electroválvulas, en la imagen no se ven los leds de pilotaje ya que estos leds no son fijos, depende de la cantidad de válvulas que estén contenidas en el módulo.

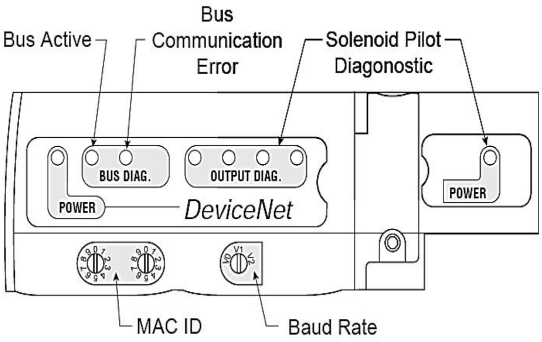


Figura 2.11. Disposición del panel de interfaz del módulo de electroválvulas.

Power. Led que indica si está alimentado. Existen 2 leds, uno para indicar la alimentación del módulo y otro para la alimentación las electroválvulas que es independiente.

- Verde constante. Módulo o solenoides alimentadas adecuadamente.
- Rojo constante. El módulo o solenoides tienen grave error de alimentación, es necesario desconectar inmediatamente el módulo ya que existe un corto en la red o inversión de polaridad.

BUS DIAG. Leds que dan un diagnóstico de la red desde el punto de vista del módulo.

Bus Active. Indica si el módulo ya está en línea.

- Verde parpadeante. En busca de la red.
- Verde constante. Módulo en línea
- Rojo constante. Error en la red.

Bus communication error. Indica si el módulo tiene problemas con la red.

- Verde parpadeante. Módulo en espera de instrucción.
- Verde constante. Módulo recibiendo o enviando datos a la red.
- Rojo constante. Error en la comunicación, es necesario reconectar.

MAC ID. Selectores tipo dial para colocar el número de nodo deseado para el módulo, este número dependerá de los demás dispositivos ya que no debe haber duplicidad de nodos en la red y de la capacidad de la red (1-63, el 0 es exclusivo para los gestores de arranque de la red)

- Selector1. Selecciona las decenas del nodo.
- Selector2. Selecciona las unidades del nodo.

BAUD RATE. Selector tipo dial que selecciona la velocidad a la que se quiere comunicar el módulo. Sólo existen 3 posiciones.

- V0. 125 Kbaudios.
- V1. 250 Kbaudios.
- V2. 500 Kbaudios.

OUTPUT DIAG. Son 4 leds que indican si los solenoides tienen un corto circuito.

- Led 1. Da el diagnóstico de cortocircuito para las válvulas 0-3.
- Led 2. Da el diagnóstico de cortocircuito para las válvulas 4-7.
- Led 3. Da el diagnóstico de cortocircuito para las válvulas 8-11.
- Led 4. Da el diagnóstico de cortocircuito para las válvulas 12-15.

PILOT. Estos leds están colocados en las electroválvulas e indican si ha sido encendida una válvula.

En el laboratorio hay un módulo de direccionamiento simple y otro de direccionamientos doble, cada uno con 3 válvulas.

En la figura 2.12 se muestran los dos módulos instalados en una mesa de pruebas del laboratorio.

Figura 2.12. Módulos de electroválvulas monoestables (izquierda) y biestables (derecha).

2.1.3. Servomotores

El servomotor desde su invención y aplicación en el área de la automatización ha desarrollado un papel importante en los procesos gracias a su control de posición, velocidad, dirección y algunas veces torque. Es importante conocer estos sistemas ya que es muy fácil encontrar este elemento en un proceso automatizado.

En el laboratorio de automatización se cuenta con este potente dispositivo de campo el cual podría ser operado sin necesidad de la red ya que dentro de sus características está la de operación manual.

Módulo de servomotor NS300

El NS300 es un módulo versátil de Yaskawa® que puede controlar servomotores de hasta alto par o servomotores de alta velocidad. Este módulo tiene una memoria interna reprogramable con la cual es capaz de guardar datos y parámetros como perfiles de velocidad, puntos de home y regímenes de velocidades.

Para que el módulo pueda operar debidamente es necesario que esté acompañado de 3 sistemas más, que son acondicionamiento, potencia y actuador. Con intención de acotar el sistema que se empleará, se señalarán únicamente las conexiones, características y configuración de los sistemas correspondientes a la red DeviceNet®.

Acondicionamiento. Etapa utilizada para transformar la tensión alterna de la acometida y obtener una tensión acondicionada para los elementos que requieran de ella.

Es una fuente de alimentación NLS2 MURRELEKTRONIK® de media fase, con regulación lineal, tensión de entrada a 230 Volts de alterna con 0.46 Amperes de corriente y salida regulada de 24 Volts de directa a 2 Amperes.

Potencia. Etapa utilizada para convertir las señales obtenidas del módulo de procesamiento (módulo NS300) y aplicarlas al motor directamente, esta etapa también está encargada de transferir los datos obtenidos del codificador al módulo NS300 para que los pueda reenviar a su dispositivo maestro.

La etapa de potencia SGDH 01AE Yaskawa® tiene múltiples entradas que se describen a continuación:

- Panel display. Quíntuple Display de 7 segmentos que informan al usuario en forma de código, el estado del módulo NS300 y de la propia etapa.
- Charge indicator. Led que se enciende si el dispositivo ha recibido la alimentación correcta y está preparado para usarse.

- Panel Keys. Panel de 4 pushbuttons utilizados principalmente para manipular el módulo sin necesidad de tener conectada una red.
- Power ON indicator. Led que se enciende cuando el módulo de control está alimentado.
- CN10. Conector que permite la comunicación con el módulo de control. El módulo de control puede cambiar, en el laboratorio hay dos módulos de control, uno es el NS300 y el otro es el MP940 ambos de Yaskawa.
- CN3. Conector para operación o reprogramación de parámetros con una computadora.
- CN1. Conector de entrada y salida utilizado para reenviar los datos recibidos a otro módulo.
- CN2. Conector para la adquisición de datos del codificador del servomotor.
- Ground terminal. Terminal para conexión a tierra.
- MCPST. Terminal para conectar la fuente de poder principal, para este caso la conexión principal de tensión es de 200 Volts de alterna.
- CPST. Terminal para la conexión del sistema de control, en el caso del NS300 se conecta a 24 Volts de tensión directa proveniente de la fuente de alimentación NLS 2.
- Servomotor terminal. Conexión directa para la alimentación del servomotor.

En la figura 2.13 se muestra la disposición de los conectores, botones e indicadores del módulo SGDH 01AE.

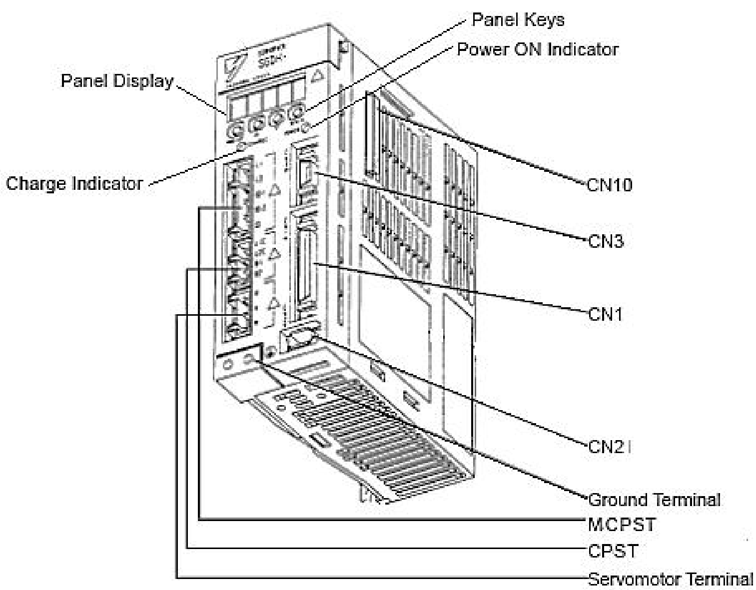


Figura 2.13. Módulo de potencia SGDH 01AE.

Actuador. Elemento final donde se ve reflejado el funcionamiento correcto de las etapas anteriores y del módulo NS300. Es un motor al cual se le controla la velocidad, dirección, posición y torque mediante transductores.

El servomotor de tensión alterna SGMAH 01AAF, es un motor con capacidad nominal de 3000 revoluciones por minuto, tensión alterna de 200 Volts, un consumo de corriente de 0.91 Amperes y un par máximo de 0.318 Nm.

Este motor tiene 2 conectores, el primero es para la alimentación y control, y el segundo es para la adquisición de datos provenientes del codificador rotatorio (*encoder*).

Una vez conocidos estos sistemas se puede definir el módulo NS300 más formalmente. Su alimentación es a través de un conector CN10 proveniente del SGDH 01AE a 24 Volts de directa y con un consumo de corriente máximo de 0.25 Amperes.

El módulo tiene conexiones parecidas a las de un dispositivo DeviceNet de Allen Bradley.

Rotary Switches. Son 3 selectores manuales para la configuración de velocidad de comunicación y elección de nodo del módulo NS300.

- MAC ID. Usa dos selectores para configurar el nodo en cualquier número del 1 al 63, al igual que en las válvulas se debe evitar duplicidad de nodos.
    1. X10. Selector para elegir las decenas del número de nodo.
    2. X1. Selector para elegir las unidades del número de nodo.
- DR. Selector único para configurar la velocidad a la que se quiere comunicar con la red.
    1. 0 - Velocidad de 125 KBaudios.
    2. 1 - Velocidad de 250 KBaudios.
    3. 2 - Velocidad de 500 KBaudios.
    4. 3..9 Ocurre un error

CN11. Conector de gestión y monitoreo del módulo a través de una computadora.

Module status. Led que indica el comportamiento del módulo respecto a la red.

- Verde parpadeante. Indica que el módulo está sobrecalentado.
- Verde constante. Operación normal.
- Rojo parpadeante. Error de bajo nivel como dirección errónea.
- Rojo constante. Error de alto nivel como corto circuito.
- Rojo y verde alternado. Se inicia un autodiagnóstico.

Network Status. Indica el comportamiento de la red respecto del módulo.

- Verde parpadeante. Se encuentra en línea pero en estado de reposo.
- Verde constante. Se encuentra en línea y transfiriendo datos.
- Rojo constante. Error fatal en la comunicación con la red.

CN6. Conector utilizado para la conexión directa a la red DeviceNet®.

CN4. Salida óptica aislada opcional para comunicación con otros módulos.

En la figura 2.14 se muestra la disposición de cada uno de los indicadores, selectores y conectores del módulo.

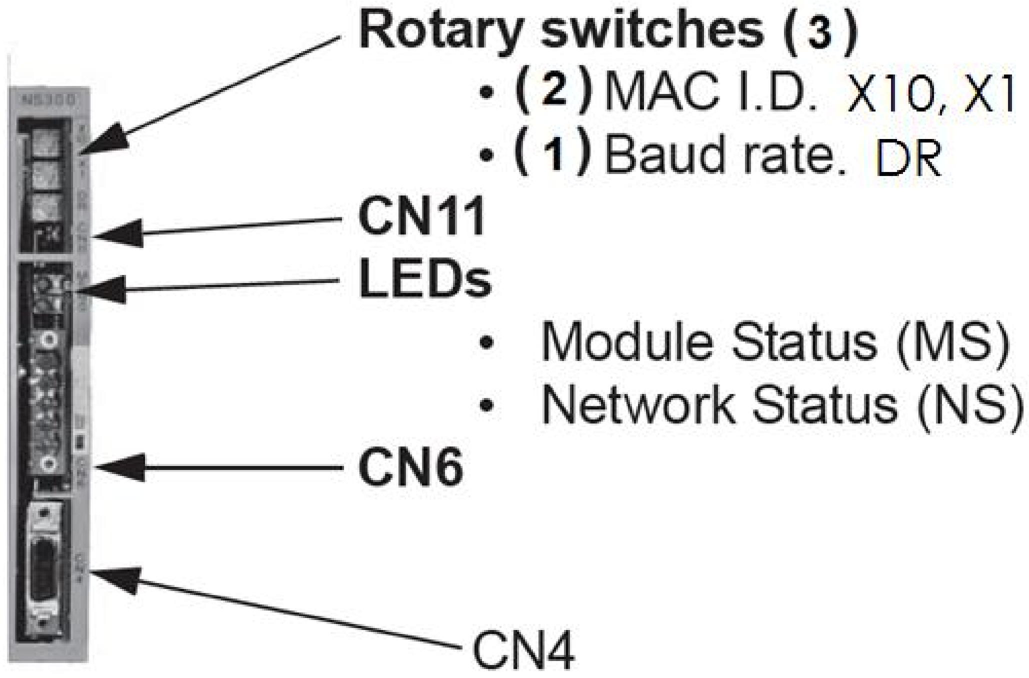


Figura 2.14. Módulo de servomotores NS300.

En la figura 2.15 se muestra el conjunto de sistemas incluidos en un gabinete, iniciando de arriba a abajo y de izquierda a derecha se encuentra la fuente de alimentación NLS 2, a continuación está un switch para alimentar a 220 Volts, en medio se encuentra el módulo MP940 con su etapa de potencia SGDH 01AE los cuales no se usarán, al final está ubicado el módulo NS300 con la etapa de potencia SGDH 01AE.

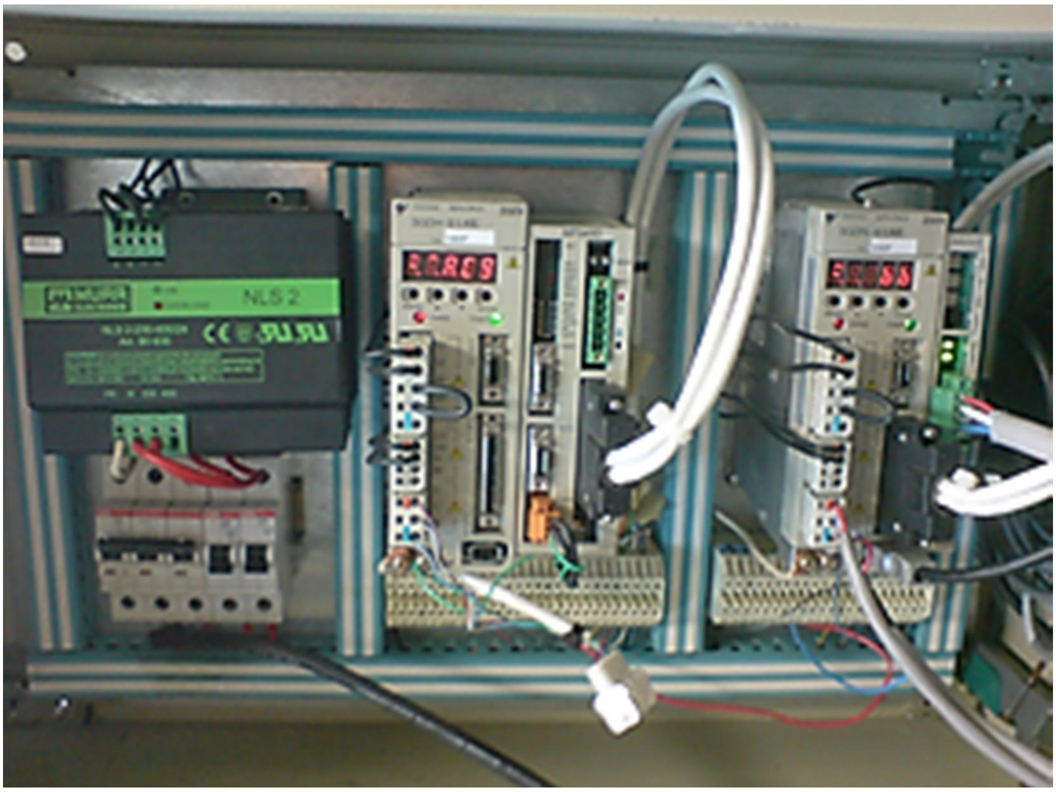


Figura 2.15. Gabinete completo del módulo de servomotores.

En la figura 2.16 se muestra el servomotor SGMAH 01AAF.

Figura 2.16. Servomotor SGMAH 01AAF.

2.2 Sistema SCADA

Un sistema SCADA es una herramienta de software que ayuda a la supervisión y control de los datos en un proceso o línea de producción automatizada, sin embargo, existen elementos físicos dedicados especialmente para el mejor desempeño de esta tarea. En el laboratorio de automatización existen dos elementos de este tipo.

- Pantalla touch Siemens. Puede entrar a una red como dispositivo remoto sin necesidad de otro dispositivo.
- Tarjeta PCI National Instruments®. Accede a la red DeviceNet® y es controlada a través de una computadora que contenga este tipo de puerto y LabVIEW®.

Tarjeta National Instruments® PCI DNET

Tarjeta diseñada para uso específico de la red DeviceNet® y configurada para ser conectada desde un puerto PCI (Peripheral Component Interconnect) la cual es una expansión de la placa madre de la computadora, en este puerto se puede también conectar tarjetas de sonido y video.

La tarjeta sólo tiene dos conectores de comunicación, el conector para el puerto PCI y el directo a la red. La tarjeta trabaja como maestro, es decir, desde la computadora se pueden editar y obtener los parámetros de otros dispositivos esclavos, contiene un microprocesador de alto desempeño Intel 80386EX.

Se pueden usar más de 2 tarjetas en la misma computadora, y gracias a su desempeño se puede hacer un control en tiempo real en cada una.

La velocidad de comunicación, MAC ID y nombre de dispositivo es editable mediante software, en el hardware sólo se puede realizar la conexión a la red.

En la figura 2.17 se puede ver el conector para la red DevieNet® de la tarjeta PCI montada en la computadora.

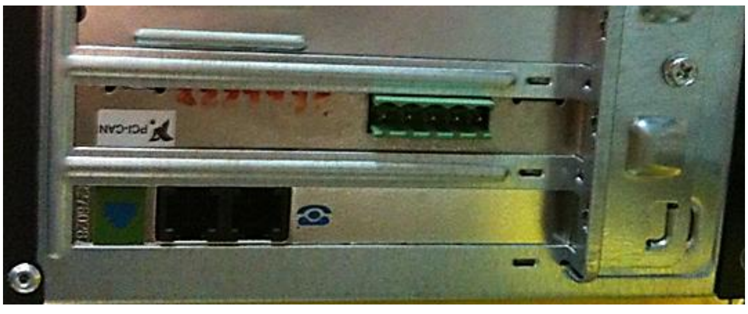


Figura 2.17. Tarjeta NI PCI-DNET instalada en una computadora.

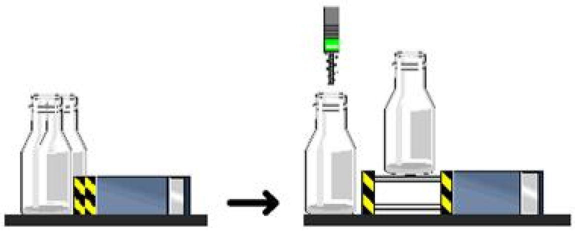
Para la creación de un sistema SCADA es necesario tener un proceso o línea de producción, es por eso que se propuso un proceso productivo para dosificar mediante un pistón botellas de agua vacías, llenarlas de agua utilizando una válvula de paso, transportarlas a un subsistema de sellado y finalmente sellarlas.

Se diseñó una topología de red con base en los dispositivos que se ocuparían en el proceso, el cual está compuesto de 4 funciones principales:

Dosificar, llenar, transportar y sellar.

Dosificar

Esta función coloca las botellas vacías provenientes de algún recipiente o proceso anterior, en posición para su llenado. El sistema simulado para esta tarea fue un pistón de doble efecto controlado por una electroválvula biestable, colocado estratégicamente de tal forma que cuando el pistón se encuentre retraído dejará pasar la botella vacía, cuando se extienda colocará la botella para su llenado, en cuanto el pistón vuelva a su estado retraído otra botella se colocará en el lugar de la anterior. En la figura 2.18 se muestran las posiciones del pistón de doble efecto.



a) b)

Figura 2.18. Pistón de dosificado. a) retraído, b) extendido.

Llenar

En la función llenar, una vez que la botella está en la posición correcta, un sistema deberá inyectar la botella vacía. El sistema simulado es el de una electroválvula monoestable la cual permite el paso y obstrucción del líquido, debido a que esta operación necesita de un sensor de nivel el cual indique que ya se ha llenado la botella, se sustituirá el sensor por una fórmula programada dentro del programa de control.

Transportar

Para que esta función pueda operar se necesita que la función llenar haya concluido, para así poder llevar la botella al sistema de sellado. El sistema simulado es el compuesto por un servomotor que mueve el rodillo de una banda, la precisión del servomotor colocará la botella en el lugar adecuado para poder sellar.

Sellar

En esta última función la botella es cerrada a presión por una tapa. El sistema simulado es el de un pistón controlado por una electroválvula monoestable el cual tiene un mecanismo que apila las tapas, cuando el pistón se extiende la tapa sella a presión con la boca de la botella, cuando regresa al estado retraído otra tapa se posiciona para ser empujada por el pistón a la siguiente botella. Este sistema debe de estar encima de la botella y el vástago del pistón alineado concéntricamente con la boca de la botella para asegurar su sellado.

En resumen el proceso productivo contiene los siguientes elementos existentes:

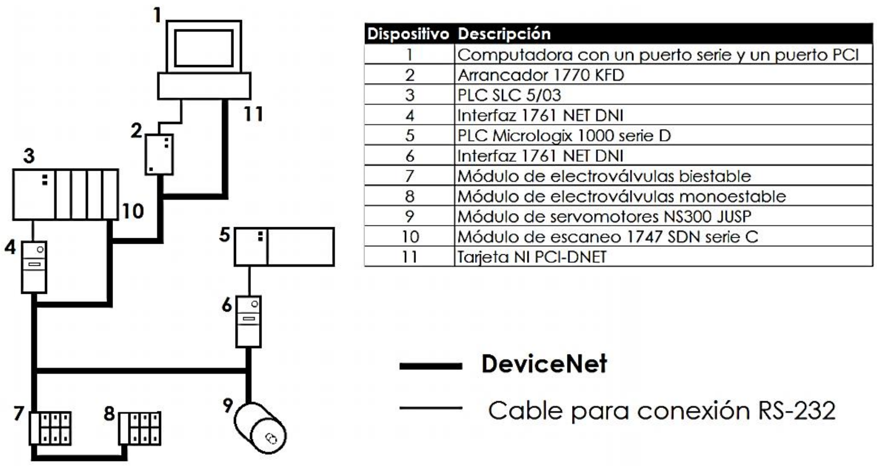
- 3 Válvulas. 2 monoestables (pistón de sellado y válvula de llenado) y una biestable (pistón de dosificado).
- 2 Pistones doble efecto. Dosificado y sellado.
- 4 Detectores de posición. 2 para pistón de dosificado y 2 para pistón de sellado.
- 1 Servomotor. Banda de transporte.

Los dispositivos de control de los elementos de campo se propusieron de la siguiente forma:

- PLC Micrologix 1000. Controló el encendido de la electroválvula para el pistón de sellado y sus 2 sensores de posición, además mantuvo comunicación punto a punto con el PLC SLC500 para leer y editar los parámetros necesarios que garanticen el correcto funcionamiento del proceso.
- PLC SLC500. Se comunicó con el PLC Micrologix 1000 y controló al escáner modular.
- Escáner 1747. Operó directamente a los módulos de electroválvulas y el módulo de servomotores con base en los parámetros establecidos por el SLC500, también fue el dispositivo que comunicó a la tarjeta PCI con el proceso.
- Tajerta PCI. Supervisa, adquiere y edita los datos críticos que fluyen a través de la red. La comunicación es punto a punto con el escáner.

El uso de los dispositivos fue realizado de tal forma que se pudiera demostrar la capacidad de cada uno de los elementos existentes en el laboratorio, a pesar de que el proceso pudo haber sido controlado con menos dispositivos.

La topología BUS diseñada para la red es la mostrada en la figura 2.19, donde se establece como maestro de la red a la tarjeta PCI, la cual está conectada a la computadora, la tarjeta tiene comunicación de punto a punto con el escáner 1747 que comparte información con el SLC500, el PLC Micrologix 1000 está habilitado para leer y editar información únicamente con el SLC500, los dispositivos de campo son controlados por los PLCS aunque son operados directamente por el escáner 1747.



Dispositivo	Descripción
1	Computadora con un puerto serie y un puerto PCI
2	Arrancador 1770 KFD
3	PLC SLC 5/03
4	Interfaz 1761 NET DNI
5	PLC Micrologix 1000 serie D
6	Interfaz 1761 NET DNI
7	Módulo de electroválvulas biestable
8	Módulo de electroválvulas monoestable
9	Módulo de servomotores NS300 JUSP
10	Módulo de escaneo 1747 SDN serie C
11	Tarjeta NI PCI-DNET

Figura 2.19. Topología de red diseñada y tabla de elementos integrados

Existen elementos que no están contemplados en la topología ya que son elementos que son propios de la red y sin ellos la red no podría operar, sin embargo, en la figura 2.20 se muestran todos los elementos conectados en la red.

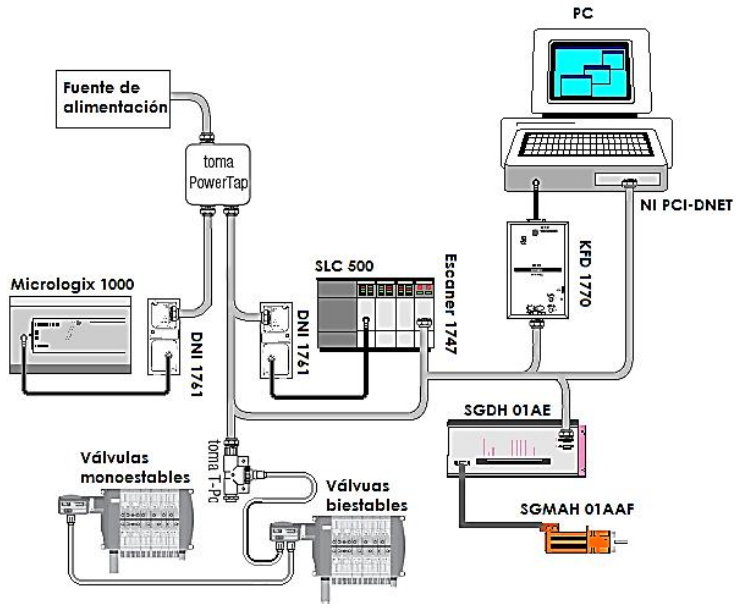


Figura 2.20. Elementos incluidos en la red DeviceNet®.

Capítulo 3. Configuración, desarrollo e implementación del software

El diseño y conexión de la red con base en la capacidad de cada uno de los elementos, es una etapa importante en la implementación, sin embargo, al tratarse de un sistema en el que fluyen datos, el uso del software y programación del firmware es una etapa vital para el correcto funcionamiento del sistema SCADA.

La implementación de esta etapa se dividió en 3 partes principales, una es la puesta en marcha y preparación de la red DeviceNet® como elemento independiente, la segunda es la programación en LabVIEW® para la comunicación con la tarjeta PCI como maestro de un dispositivo DeviceNet y por último se hizo la programación y reconfiguración de los PLCs y la tarjeta PCI para conformar el sistema SCADA.

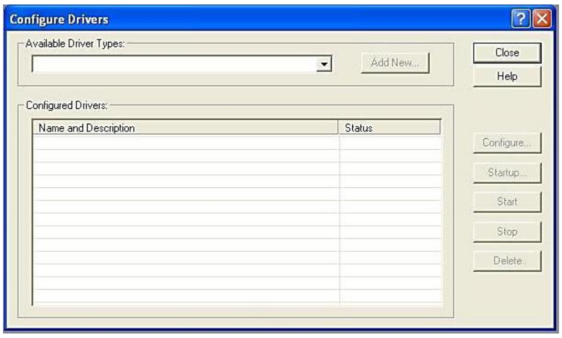
3.1. Red DeviceNet®

3.1.1. Configuración

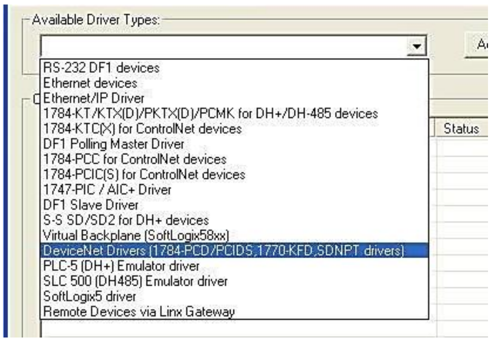
La primera parte de la implementación del software inició una vez que se instaló la red. Se encendió la fuente de poder y después cada uno de los dispositivos que tienen alimentación independiente. Inmediatamente el arrancador 1770 KFD conectado a la computadora a través de un cable de comunicación RS-232 DB9 y alimentado por el cable de la red, encendió los leds conforme a lo especificado en las características del arrancador, en la computadora se ejecutó el software RSLinx de Rockwell Automation® V2.43. Este software nos dio de alta la red en el sistema, se encargó de configurar la velocidad de la transferencia de datos, habilitar los puertos de la computadora que se usaron y asignar el número de nodo del arrancador de la siguiente forma.

En la computadora se dio clic en Inicio > Programas > Rockwell Software > RSLinx y desplegó una nueva pantalla.

En la barra de menú principal estaba disponible la opción "Communications" > "Configure Drivers", apareció una ventana (figura 3.1 a) en la que se dio clic a la pestaña "Available Driver Types", donde se desplegó una lista de controladores a las que RSLinx da soporte (figura 3.1 b), entre ellas se encontró la opción "DeviceNet Drivers", se seleccionó con un solo clic y después se presionó el botón "Add New".



a)



b)

Figura 3.1 a) Ventana para configurar controladores, b) Lista de controladores.

Al presionar “Add New” se desplegó una nueva ventana la cual contenía el controlador Allen Bradley® 1770-KFD, al seleccionarla desplegó una nueva ventana para configurar los parámetros de inicialización tal y como la muestra la figura 3.2.

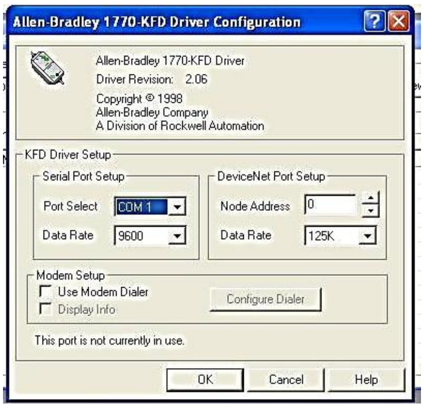


Figura 3.2. Ventana de configuración.

Los parámetros para primera ejecución son principalmente la configuración del puerto de la computadora y la del puerto de la red.

- Serial port
  - i) Port select. Se seleccionó el puerto de comunicaciones usado en la computadora. COM1.
  - ii) Data rate. Se seleccionó la velocidad de transferencia entre la computadora y el 1770-KFD. 9600 Baudios.
- DeviceNet port
  - i) Node address. Se seleccionó el número de nodo dentro de la red. Nodo 0 recomendado por los diseñadores de la red DeviceNet®.
  - ii) Data rate. Se seleccionó la velocidad de transferencia dentro de la red. 125 Kilo baudios.

Se deben tener en cuenta los siguientes puntos:

- El puerto COM elegido en la configuración no puede ser compartido por otra aplicación ya que se vuelve lenta cuando se transfieren datos.
- Existe la alternativa de emplear un puerto COM empleando un convertidor USB/Serial, el cual también fue probado.

Al aceptar los parámetros configurados, se inició la comunicación y arranque de la red, apareció una ventana emergente donde se dio un nombre al controlador de la red el cual fue "DEVICENET". En la ventana "Configure Drivers" en la zona de "Configured Drivers" apareció el controlador "DEVICENET" activado, con el identificador MAC seleccionado (0) y a la velocidad configurada (125 K), figura 3.3.

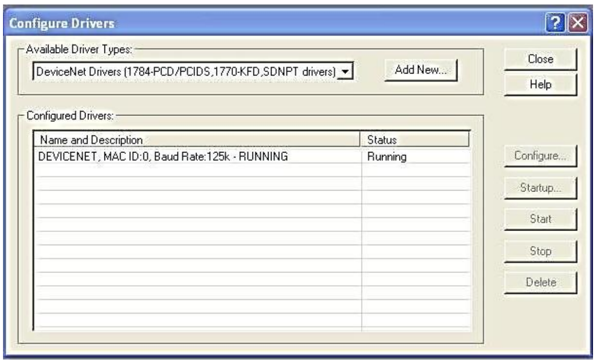


Figura 3.3. Controlador dado de alta.

Si no aparece la ventana emergente para colocar el nombre del controlador se debe de verificar lo siguiente:

- Se conectó al menos uno de los dispositivos que entrará a la red, si sólo se conecta el 1770-KFD la inicialización no podrá concluir.
- Se seleccionó en la configuración el puerto de comunicación adecuado, si se conectó el puerto COM1 y el que se seleccionó fue el COM2 el problema persistirá.
- Antes de configurar el controlador de la red, verificar que en el panel de "Configured Drivers" en la ventana "Configure Drivers" no exista ningún controlador ejecutándose, en caso contrario, seleccionar el controlador en ejecución y dar clic en la opción "STOP" del panel derecho, posteriormente seleccionar la opción "DELETE".
- Los leds indicadores de "Status Network" y "Status Module" del arrancador 1770-KFD no señalen ningún error.

Por último se cerró la ventana "Configure Drivers" y se verificó que la inicialización fue realizada satisfactoriamente, en la opción "Communications" > "RSWho" del menú principal, se desplegó una ventana donde se mostró del lado izquierdo las conexiones de red existentes o que reconoce por defecto la computadora, entre las redes reconocidas se encontró "DEVICENET", se dio clic en ella y del lado derecho después de una búsqueda automática se desplegó una lista con los dispositivos que se encontraban conectados a la red, figura 3.4.

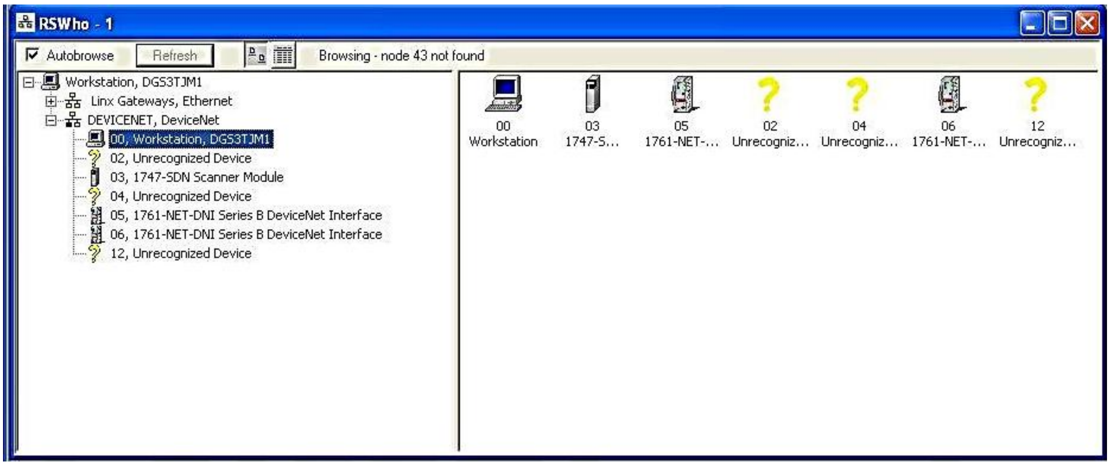


Figura 3.4. Dispositivos detectados por RSLinx.

Con excepción de la tarjeta PCI, si no se detectan todos los dispositivos que fueron conectados físicamente a la red, se debe revisar lo siguiente:

- Duplicidad de nodos, es decir, que no hay dispositivos con el mismo número de nodo. En el caso de los dispositivos con selector de nodo, se recomienda desconectarlo, cambiar el número de nodo manualmente y volver a conectar sin necesidad de volver a hacer los pasos anteriores.
- Que la velocidad de transmisión de la red sea la misma en cada uno de los dispositivos, En el caso de los dispositivos con selector de velocidad, sin necesidad de desconectar, se recomienda poner el selector en autorate o en la velocidad seleccionada para la red.
- Que el nodo se encuentre bien conectado al dispositivo.
- Que los dispositivos estén energizados, cuando los dispositivos dependen de la fuente, se verifica la tensión con un multímetro en los 2 pines correspondientes, si no hay tensión se revisa la tensión desde la fuente para conocer si es por corto o por falso contacto del cable asociado al dispositivo. En caso que sea alimentación externa revisar la fuente de suministro.

3.1.2. Comunicación

La configuración de la red concluyó con la ubicación automática de los dispositivos conectados en la red, queda mencionar que la tarjeta PCI aunque estuvo conectada físicamente a la red y por lo tanto utilizó uno de los nodos, no se identificó como dispositivo manipulable en la red, esto es porque la tarjeta PCI es usada únicamente como maestro, por lo tanto no se pueden obtener datos, a menos que se autorice el acceso.

El software RSLinx permitió la configuración de la red pero no tuvo la capacidad de comunicarse directamente con los dispositivos que estaban conectados, para esta tarea existe otro software llamado RSNetworx de Rockwell Automation® V5.11, este software ayudó a ordenar la red, se cambiaron algunos números de nodo y se configuraron los dispositivos esclavos de la red.

La ejecución comenzó en Inicio > Programas > Rockwell Software > RSNetworx y desplegó la siguiente pantalla, figura 3.5.

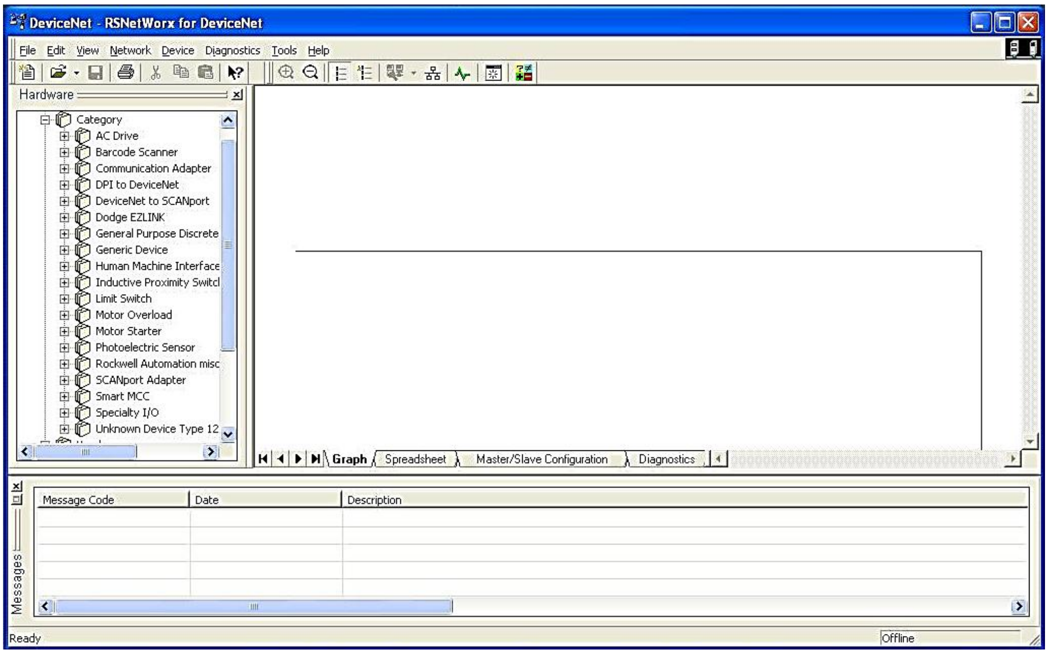


Figura 3.5. Ventana principal de RSNetworx.

En esa pantalla se visualizarán 3 zonas diferentes “Hardware”, “Topology” y “Messages”, la primera está ubicada en la zona izquierda, la segunda ubicada del lado derecho y la última abarcando la parte inferior.

En la zona de “Hardware” se localiza cada uno de los dispositivos con los que el programa se puede comunicar, en la zona de “Topology” se muestran gráficamente cada uno de los dispositivos que están conectados en la red, y por último en la zona de “Messages” se publican cada una de las actividades realizadas por el programa, así como los eventos que requieren de atención y los eventos realizados satisfactoriamente.

Para visualizar gráficamente los dispositivos de la red se tiene que sincronizar el programa RSNetworx con la red DeviceNet®, en el menú principal se seleccionó la opción Network > On line, se desplegó una ventana con un árbol similar al de RSLinx; con un clic en la opción de la red "DEVICENET", los dispositivos conectados empezaron a visualizarse, figura 3.6, al aceptar la red seleccionada, se lanzó una advertencia la cual mencionaba que la red necesitaba ser cargada o descargada para poder ser sincronizada, se seleccionó la opción de aceptar ya que lo que se necesitaba era cargar los datos de los dispositivos.

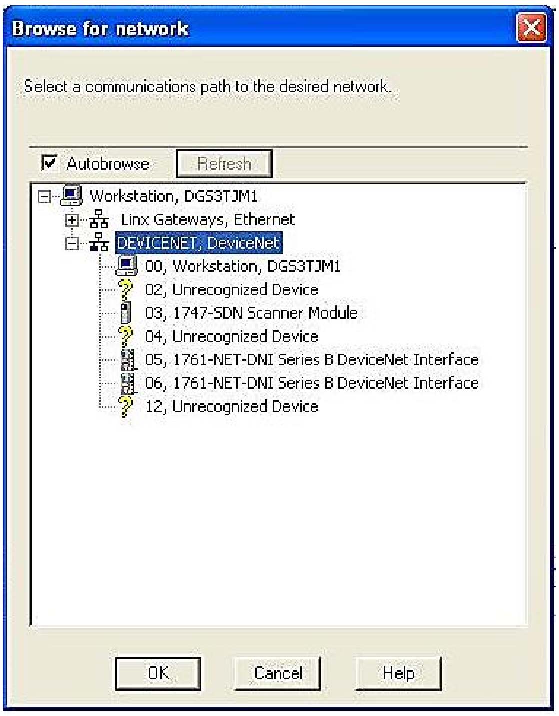


Figura 3.6. Ventana para la búsqueda de dispositivos.

Una vez terminada la sincronización y carga, en la zona de "Topology" aparecieron cada uno de los dispositivos (excepto la tarjeta PCI), así como en la zona de "Messages" se notificó de algunos dispositivos que no contaban con los controladores necesarios y por lo tanto no podían comunicarse con el programa, figura 3.7.

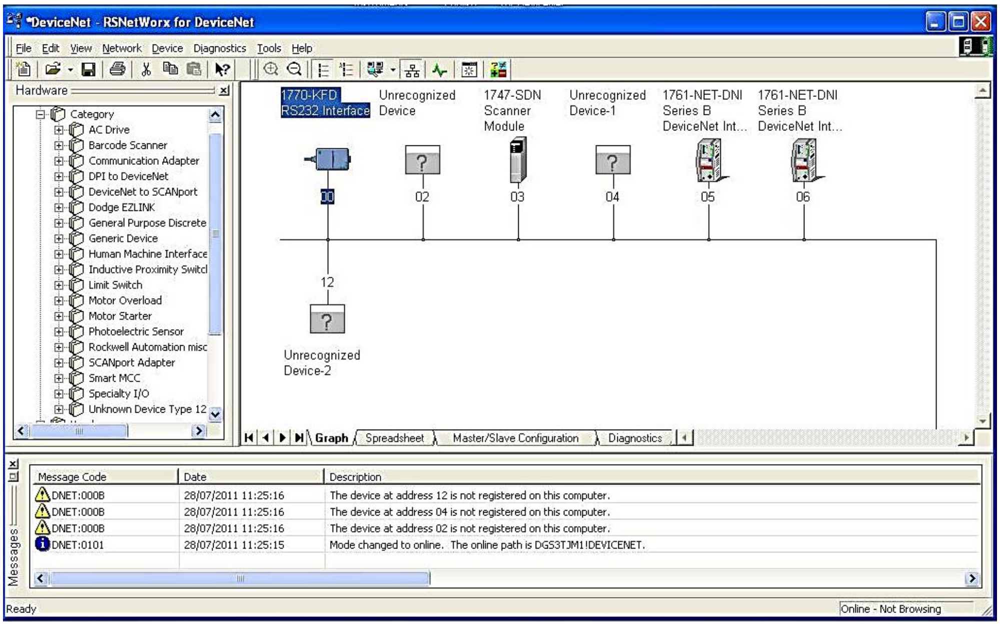


Figura 3.7. Dispositivos reconocidos y mensajes de dispositivos no registrados.

Los dispositivos que no se reconocieron por RSNetworx fueron los módulos de las válvulas y el módulo del servomotor, esto fue de esperarse ya que ninguno de ellos pertenece a Rockwell Automation®, para arreglar este problema se necesita cargar los controladores de cada dispositivo.

La forma de identificar los módulos de electroválvulas y servomotores, fue por el número de nodo seleccionado que tenía el dispositivo antes de conectarlas, se dio clic con el botón derecho sobre un módulo de electroválvulas y apareció una ventana emergente de la cual se seleccionó “Register Device”, como consecuencia se desplegó un wizard de instalación de EDS. Un EDS es un archivo en formato electrónico necesario para configurar los dispositivos que no estén en la base de datos del programa RSNetworx, EDS es el acrónimo de “Electronic Data Sheet”, esto significa que con base en estos archivos podemos obtener los datos y configuraciones del dispositivo para poder manipularlo, cada dispositivo o familia de dispositivos tiene un EDS, estos fueron descargados de las páginas electrónicas de los fabricantes.

En el wizard aparecieron varias opciones para dar de alta el EDS, en caso de saber los parámetros del dispositivo se podía crear el EDS manualmente, sin embargo, se contaba con los archivos EDS, por lo que se seleccionó la primera opción; en la siguiente pantalla del wizard apareció la opción “Browse” donde se colocó la ruta en la que estaba almacenado el archivo EDS del módulo de válvulas, la siguiente pantalla mostró el EDS seleccionado figura 3.8, al pasar a la siguiente pantalla se mostró el EDS instalado, por defecto el wizard seleccionó un icono para las electroválvulas pero fue cambiado por uno más representativo.

Figura 3.8. Ventana con el EDS instalado.

Al cerrar el wizard se actualizaron los íconos de los dos módulos de electroválvulas por efecto de que los dos son iguales, para dar de alta el EDS del módulo de servomotores se repitió el mismo procedimiento que en el módulo de electroválvulas.

Una vez dados de alta todos los EDS, se dispuso a nombrar y asignar un número de nodo a cada dispositivo. Para cambiar el nombre de los dispositivos, en la zona de “Topology” se dio doble clic en el nombre que contiene el dispositivo por defecto y se coloca el nuevo nombre, de igual forma para cambiar el número de nodo se dio doble clic sobre el número del dispositivo que se quería cambiar, a diferencia del cambio de nombre, este paso se realizó con precaución por dos cosas; una es que sólo se puede cambiar el número de nodo si el dispositivo no cuenta con un selector de nodo físico y la otra es por la duplicidad de nodos.

A los dispositivos sin selector físico se les asignó un número de nodo a través de RSNetworx y a los otros se les configuró el nodo deseado manualmente, si se cambia el nodo manualmente se debe de desconectar el dispositivo y sin alimentación se selecciona el nuevo nodo.

El orden por número de nodo y nombre de los dispositivos quedó de la siguiente forma:

0. Arrancador 1770-KFD
1. Módulo válvulas mónoestables
2. Módulo válvulas biestables
3. Escáner 1747-SDN
4. Tarjeta PCI
5. PLC SLC500
6. PLC Micrologix 1000
7. Módulo servomotor NS-300

El orden de los dispositivos puede o no existir pero es conveniente arreglarlos para tener un mejor reconocimiento. En la zona de "Topology" nunca aparecerá la tarjeta PCI o si llega a aparecer será por un periodo corto y después desaparecerá, sin embargo, esto no quiere decir que se pueda utilizar en otro dispositivo el nodo asignado para la tarjeta. El acomodo jerárquico requerido para implementar la topología diseñada es realizado en este programa, según la pirámide de automatización y el diseño propuesto para ejecutar el proceso productivo, en el nivel de campo se colocó a la Tarjeta PCI, el escáner 1747 y los PLCs SLC500 y Micrologix 1000, a nivel de sensores y actuadores se tiene a los dos módulos de electroválvulas y el módulo de servomotores.

En el capítulo anterior se explicó el objetivo de cada uno de los dispositivos y, por lo tanto, las dependencias entre ellos. La dependencia más importante es la que tienen los actuadores con el escáner, ya que el escáner será el encargado de dar el acceso a estos dispositivos. Los módulos de actuadores (electroválvulas y servomotor) deben de ser asignados como esclavos del escáner.

Antes de asignar a los módulos como esclavos del escáner, se debe de saber cuáles son sus formas de comunicación.

Electroválvulas

Se dio doble clic sobre el ícono de uno de los módulos de las válvulas y desplegó una ventana de datos generales, se dio clic sobre la pestaña "Parameters" y desplegó un mensaje con opción de cargar o descargar la información del módulo, se seleccionó la opción de cargar y aparecieron los parámetros de la válvula y el valor del estado actual en el que se encontraban, figura 3.9.

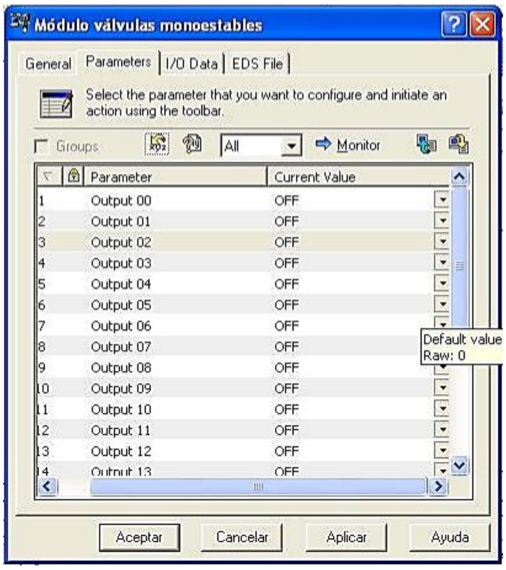


Figura 3.9. Parámetros del módulo de electroválvulas.

Cada vez que se seleccionó un dispositivo para editar u obtener datos de él, se desplegó este mensaje, fue importante siempre seleccionar la carga y no la descarga de información a los dispositivos ya que la descarga podría borrar los datos almacenados en ellos.

En esta pestaña se observó que cada módulo de válvulas tiene la capacidad de controlar 16 salidas y 4 entradas discretas, las cuales están repartidas como lo especifica el cuadro 3.1.

I/O Tables Common to All Device Bus Modules

Input Data Table								
Byte	**Bit 0**	**Bit 1**	**Bit 2**	**Bit 3**	**Bit 4**	**Bit 5**	**Bit 6**	**Bit 7**
0	Discrete Input 0 (Diagnostic LED 0-3)	Discrete Input 1 (Diagnostic LED 4-7)	Discrete Input 2 (Diagnostic LED 8-11)	Discrete Input 3 (Diagnostic LED 12-15)	—	—	—	—
Output Data Table								
Byte	**Bit 0**	**Bit 1**	**Bit 2**	**Bit 3**	**Bit 4**	**Bit 5**	**Bit 6**	**Bit 7**
0	Discrete Output 0	Discrete Output 1	Discrete Output 2	Discrete Output 3	Discrete Output 4	Discrete Output 5	Discrete Output 6	Discrete Output 7
1	Discrete Output 8	Discrete Output 9	Discrete Output 10	Discrete Output 11	Discrete Output 12	Discrete Output 13	Discrete Output 14	Discrete Output 15

Cuadro 3.1. Distribución de bits de entrada y salida en electroválvulas.
Fuente. Catálogo de sistemas Fieldbus - Moduflex

De acuerdo con la tabla y documentación de módulo de electroválvulas, a la salida se cuenta con dos bytes para el direccionamiento de 16 electroválvulas, las primeras 8 (bit 0 al 7) en el byte 0 y las últimas 8 (bit 8 al 15)en el byte 1, de tal forma que la primera electroválvula está ubicada en el bit 0 del byte 0 y la última en el bit 7 del byte 1; a la entrada se cuenta con un byte completo de información, sin embargo, sólo se utilizan los bits 0,1, 2 y 3, donde cada uno indica si existe un corto circuito en las electroválvulas 0-3, 4-7, 8-12 y 13-15 respectivamente, es decir, si existe un corto circuito en la válvula 9, el bit 2 del byte 0 se encenderá.

En la pestaña "I/O Data" se identifica la forma de comunicación de las válvulas en la red, figura 3.10, en este módulo se puede elegir la forma de comunicación, por defecto el módulo se comunicó en tipo Polled, aunque podría cambiarse si es que el diseñador así lo prefiere.

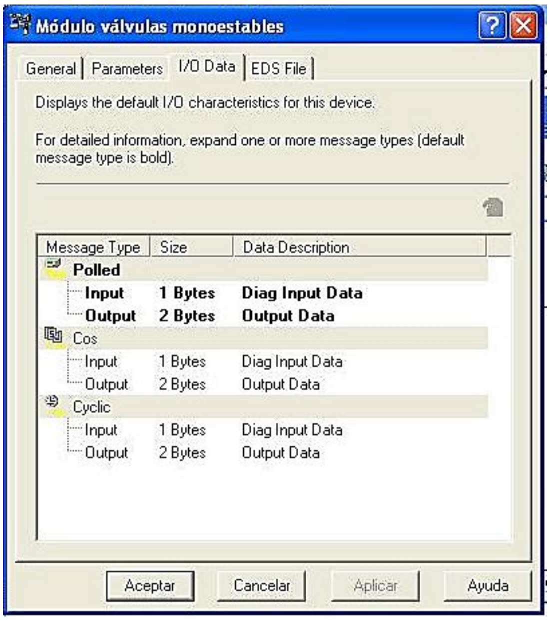


Figura 3.10. Información de la pestaña "I/O Data" en el módulo de válvulas monoestables.

Servomotor

Se dio doble clic en el ícono del módulo del servomotor y se desplegó una ventana con datos idéntica a la desplegada en las electroválvulas, al seleccionar la pestaña "Parameters" y cargar los datos del módulo, se identificaron más de 280 parámetros. El módulo del servomotor es un sistema avanzado el cual puede ser manipulado manualmente o remotamente; es precisamente por éste último que es de vital importancia el que los 282 parámetros del servomotor estén debidamente guardados ya que un parámetro erróneo provocaría problemas en la operación del módulo, los parámetros clave programados para este servomotor son.

Tipo de búsqueda del home, velocidad de búsqueda de home, constante de aceleración y desaceleración, velocidad de avance, máxima velocidad de avance, tiempo de aceleración, porcentaje de par y velocidad de aproximación, para mayor referencia de los parámetros ver el anexo 4.

El servomotor ocupa 8 bytes de entrada y 8 bytes de salida, a diferencia de las electroválvulas las entradas y salidas no siempre son discretas, para el módulo los bytes de entrada son bytes de respuesta y los de salida son bytes de comando, la distribución de los bytes no es constante, la respuesta proveniente del módulo y comandos de entrada al módulo son cambiados si los alcances del módulo así lo requieren, en el cuadro 3.2 se puede ver la distribución para los bytes de comando y en el cuadro 3.3 los bytes correspondientes a la respuesta.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	MOD	0	ALRST	ESTP	0	0	SVON	C_STRT
1	Response type				Command code			
2	HOME	PTBL	STN	STEP	FEED	0	HOLD	CANCEL
3	0	0	0	0	0	0	DIR	INC
4	Command data							
5								
6								
7								

Cuadro 3.2. Distribución de los bytes de comando en el servomotor.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	Response type				Command code			
2	HOME_R	PTBL_R	STN_R	STEP_R	FEED_R	0	HOLD_R	PRGS
3	POT	NOT	INPOS	NEAR	HOME_P	0	DIR_R	INC_R
4	Command message							
5								
6								
7								

Cuadro 3.3. Distribución de los bytes de respuesta en el servomotor.

Todos los bits y datos de comando son importantes en el servomotor por lo que se debe de estar bien documentado, algunos bits y datos de comando son:

- C_STRT (Bit 0, byte 0). Permite la entrada de comandos.
- SVON (Bit 1, byte 0). Permite la alimentación al servomotor.
- Command Code (Bit 0-3, byte 1). Indica el tipo de comando a utilizar.
- FEED (Bit 3, byte 2). Permite el movimiento constante del servomotor de acuerdo al dato marcado en el “Command data”.
- DIR (Bit 1, byte 0). Indica la dirección de giro del servomotor.
- Command data (byte 4-byte 7). Utilizado para proporcionar al módulo el dato correspondiente al comando elegido.

Para mayor referencia de los comandos y respuestas del módulo del servomotor, ver el anexo 4.

Este módulo de servomotores sólo puede comunicarse en modo Polled a través de los 8 bytes de comando y 8 bytes de respuesta.

Escáner

Se dio doble clic en el ícono del escáner 1747 y se desplegó una ventana de edición, en esta ventana aparecieron 7 pestañas.

General. Nombre, número de nodo e identificadores del módulo.

Module. Edición de operación del módulo (esclavo o maestro), restauración de configuración y carga o descarga de información al escáner.

Scanlist. Adición o eliminación de los dispositivos esclavos del escáner.

Input. Distribución de las entradas de los dispositivos esclavos del escáner.

Output. Distribución de las salidas de los dispositivos esclavos del escáner.

ADR. Reemplazo automático de dispositivos.

Summary. Información del estado del escáner y sus dispositivos esclavos.

Se dio clic sobre la pestaña “Scanlist” para poder agregar los dispositivos esclavos al escáner, en la ventana aparecieron dos listas, una era la de los dispositivos disponibles y la otra era de los que estaban dentro de la lista de escaneo, en la lista de dispositivos disponibles aparecieron los dos módulos de válvulas (nodo 1 y 2), las interfaces 1761 de los PLCs SLC500 y Micrologix1000 (nodo 5 y 6) y el módulo del servomotor (nodo 7), mientras que en la lista de escaneo no existía ningún elemento, se deseleccionó la opción de “Automap on Add” ubicada debajo de la primer lista y se seleccionó uno de los módulos de electroválvulas de la lista, se dio clic sobre el botón de agregar “>”, figura 3.11.

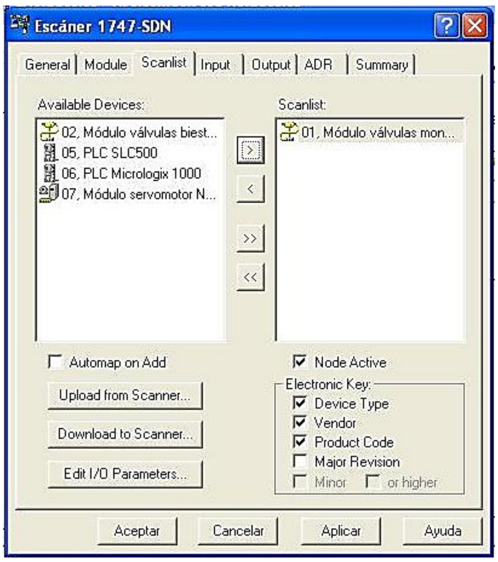


Figura 3.11. Pestaña “Scanlist” para agregar dispositivos esclavos.

De esta forma se agregó el primer elemento a la lista de escaneo; para el segundo módulo de válvulas y el módulo del servomotor se hizo lo mismo, se seleccionó y se agregó cada uno por separado. Las interfaces 1761 no pueden ser agregadas como dispositivos esclavos por 2 restricciones, una es que el PLC SLC500 no puede ser esclavo de su propio módulo (Escáner 1747) y la segunda es que para que un PLC pueda ser esclavo, el maestro debe de ser un sistema de alto nivel, tal como una computadora industrial u otro PLC.

El escáner permite el acceso a las entradas y salidas de los dispositivos esclavos desde el PLC SLC500. Es decir, si el SLC500 no da la orden al escáner de escribir o leer un dispositivo, el escáner no podrá generar ningún cambio porque es un sistema simple.

Una vez agregados los 3 dispositivos esclavos se dio clic sobre la pestaña "Input", en esa pestaña se mostraron en una lista las entradas de los 3 dispositivos esclavos del escáner, el tipo de comunicación, tamaño en bytes de la entrada y mapeo, aunque estos 3 módulos ya son esclavos del escáner aún no pueden comunicarse con él debido a que sus entradas y salidas no están mapeadas en la memoria del escáner.

El escáner cuenta con una memoria de 600 bytes distribuidos en 150 palabras de entrada y 150 palabras de salida, considerando que cada palabra es de 2 bytes da como resultado que el PLC SLC500 además de toda la capacidad de memoria nominal, podrá manejar 150 palabras de entrada y salida adicionales, sólo que estas palabras serán para el control de los dispositivos esclavos del escáner.

La memoria asignada en el escáner para la escritura y lectura de las entradas y salidas de los dispositivos esclavos son denominadas archivos M, de los cuales M0 representan a las salidas y M1 a las entradas, por lo tanto, se tienen las palabras de salida desde el M0.0 hasta el M0.149 y en el caso de las entradas desde el M1.0 hasta el M1.149.

En la pestaña apareció la opción para mapear las entradas de los esclavos en memoria, debajo de la lista de entradas de los dispositivos esclavos se seleccionó la opción "M File", se dio clic sobre el primer módulo de válvulas de la lista y después sobre el botón "Automap", automáticamente el byte usado para leer la entrada del primer módulo de válvulas apareció mapeado en la memoria del escáner como lo muestra la figura 3.12, la primer válvula ocupa la mitad de la primer palabra (M1:1.0), es decir, desde el bit 0 (M1:1.0/0) hasta el bit 7 (M1:1.0/7). Los módulos restantes fueron añadidos a la memoria del escáner de la misma forma.

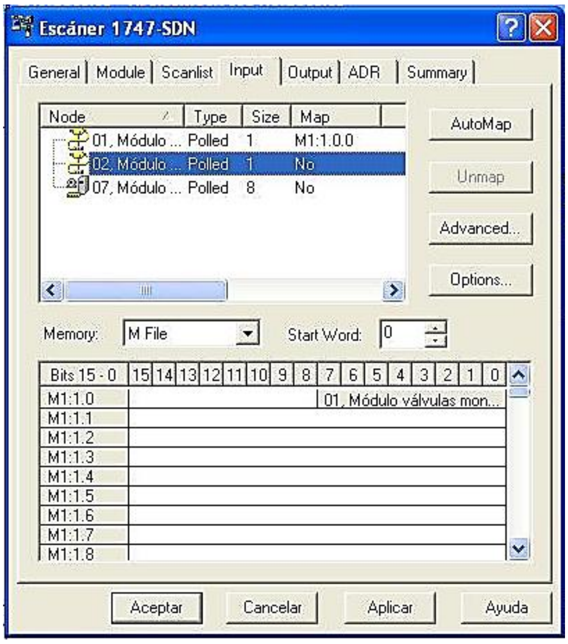


Figura 3.12. Mapeo de los bytes de entrada en la memoria del escáner.

Se aplicó el mismo procedimiento utilizado en la pestaña “Input” para agregar las salidas de los dispositivos esclavos en la pestaña “Output”, con la diferencia que en las salidas se manejan los archivos M0 y el tamaño de las salidas en los módulos de válvulas son diferentes (2 bytes cada uno). La distribución en las palabras de entradas y salidas de los módulos en cada pestaña quedó ordenada como lo muestra la figura 3.13.

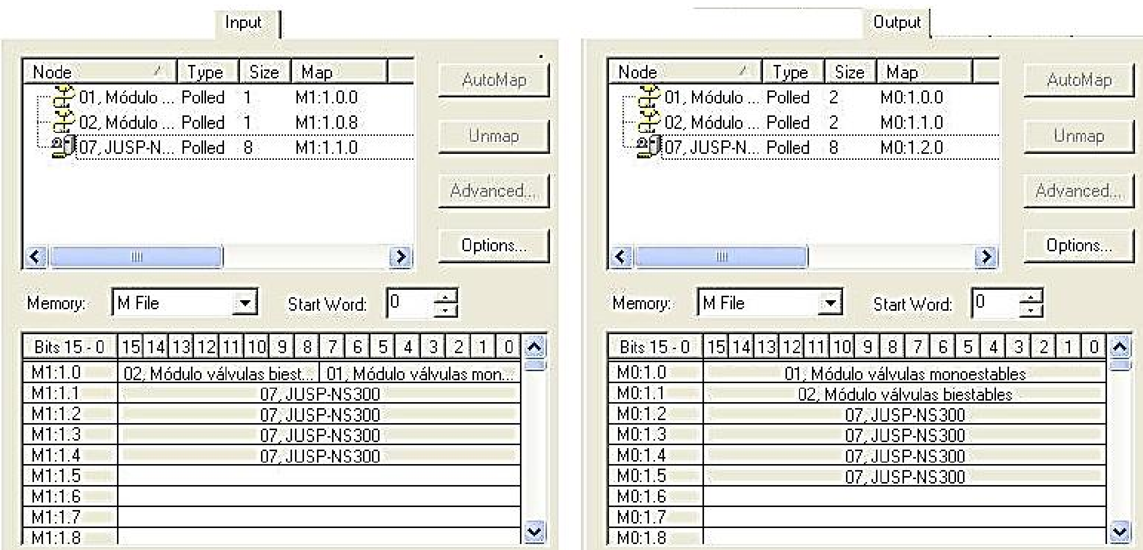


Figura 3.13. Distribución de entradas y salidas en los archivos M del escáner.

Se aplicaron los cambios en el módulo y se cerró la ventana para la edición del escáner.

PLC Micrologix 1000

Este PLC tiene la capacidad de comunicarse con otros PLCs de mayor o menor nivel utilizando una función avanzada de mensajes explícitos, el mensaje envía una palabra de información (2 bytes) a otros PLCs ya sea para escribir o leer información, en la función existen 4 parámetros importantes:

- Read/Write. Elige el modo de acceso al procesador con el que se comunicará (lectura o escritura).
- Target device. Tipo de procesador del PLC objetivo, 500CPU o 485CIF (los PLC Allen Bradley® utilizan el 500CPU).
- Control block. Bloque de información con el cual se controlará el correcto funcionamiento del mensaje.
- Block length. Tamaño en palabras del bloque de información, por defecto son 7 palabras (cada palabra es de 2 bytes).

En el bloque de control se tiene información para la prevención y corrección de errores en el mensaje, sin esa información podrían generarse loops infinitos o paro total del programa en ejecución. En la primer palabra del bloque se encuentran algunos bits de control de errores, entre los bits importantes y usados para la programación se encuentra:

- Time Out Bit (bit 8). Se activa temporalmente cuando la instrucción MSG falló.
- Enabled and Waiting Bit (bit 10). Se activa en cuanto se entra al peldaño de la función de mensaje.
- Error Bit (bit 12). Se activa cuando la transmisión del mensaje falló.
- Done Bit (bit 13). Se activa cuando el mensaje fue transmitido satisfactoriamente.
- Start Bit (bit 14). Se activa cuando se recibe un dato desconocido proveniente del otro PLC, esto indica que el PLC ya ha empezado a procesar el mensaje, se resetea cuando el bit 8, 12 o 13 son activados.
- Enable Bit (bit 15). Se activa si el buffer de transmisión de mensajes está disponible.

3.1.3. Programación

El establecimiento de la comunicación concluyó con la parametrización del escáner, hasta ese momento la red podía comenzar a operar sin problemas; como prueba del correcto procedimiento se programó un ejemplo en el cual se debía de encender el segundo pilotaje del módulo de válvulas monoestables utilizando una entrada física desde el micrologix 1000, el programa del micrologix 1000 es mostrado en la figura 3.14.

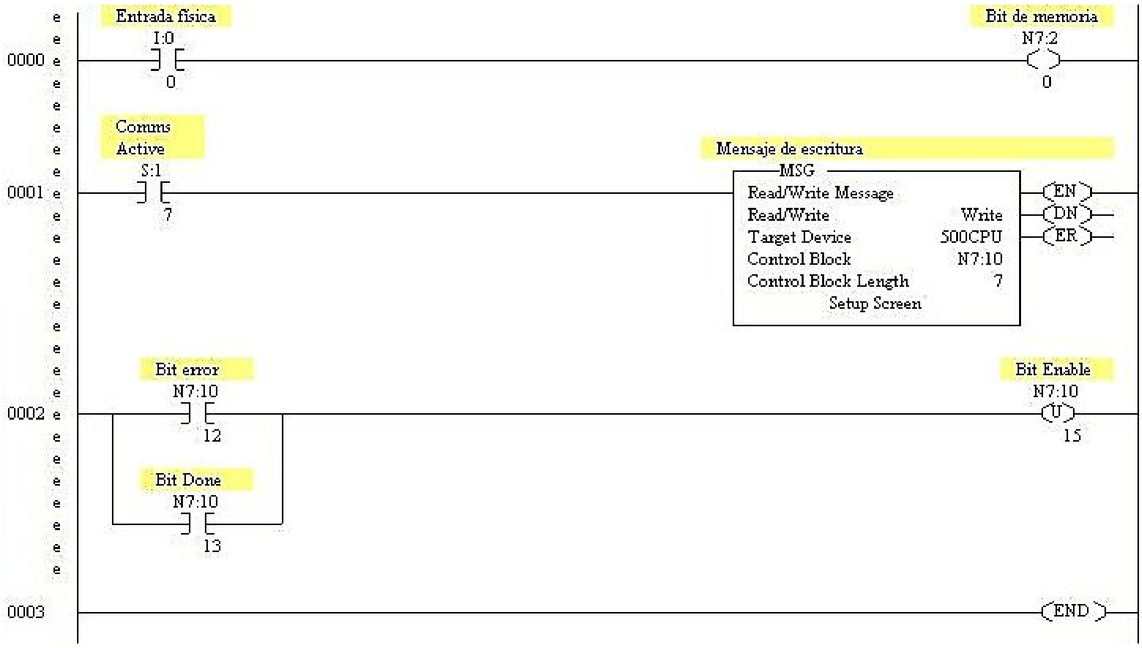


Figura 3.14. Programa para prueba de comunicación en el Micrologix 1000.

En este programa se asignó el valor booleano del primer bit (I:0/0) de entradas discretas (I:0), al primer bit (N7:2/0) de la tercer palabra (N7:2) de números enteros (N7), el transferir el valor de una entrada física a una palabra en memoria permitió enviar información al PLC SLC500.

El bloque de control comenzó desde la palabra N7:10 hasta N7:16 y la información del PLC fuente (Micrologix 1000, nodo 6) y el PLC objetivo (SLC500, nodo 5) se muestra en la figura 3.16.

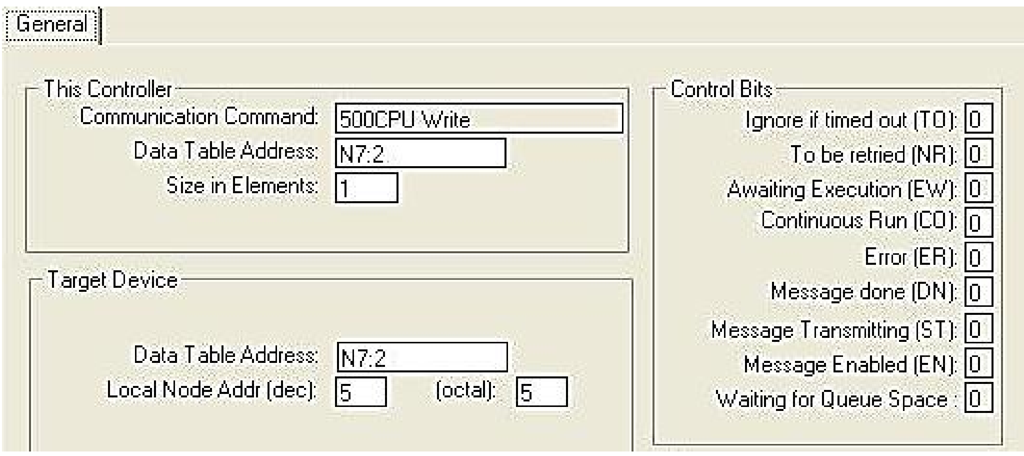


Figura. 3.15. Bloque de control para el envío del mensaje al SLC500.

El programa para el SLC500 es mostrado en la figura 3.16.

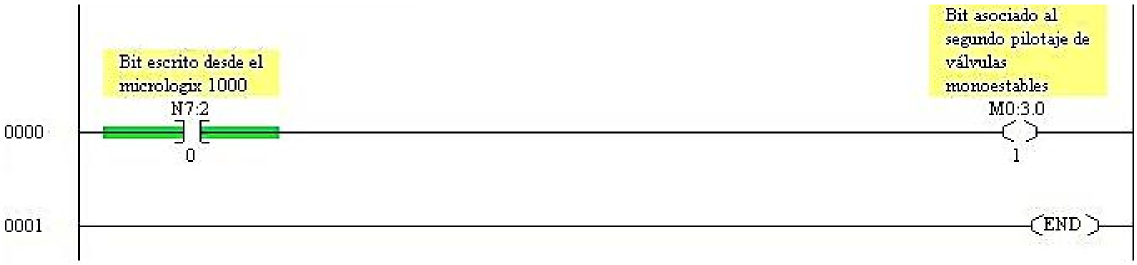


Figura 3.16. Programa para prueba de comunicación en el SLC500.

La función de mensaje escribió la tercer palabra de enteros (N7:2) del Micrologix 1000 en la tercer palabra de enteros del SLC500 (N7:2), con esto se leyó el bit en cuestión (N7:2/0) y se asoció el valor de este bit al pilotaje de la segunda válvula del módulo de monoestables (M0:3.0/1, donde M0 representa los archivos de salida, el 3 es el número del slot en el que está el escáner, el 0 indica la palabra y el 1 el bit del pilotaje, según la parametrización realizada para el escáner).

En la figura 3.17 se muestra el flujo y transferencia de datos entre los 2 PLCs.

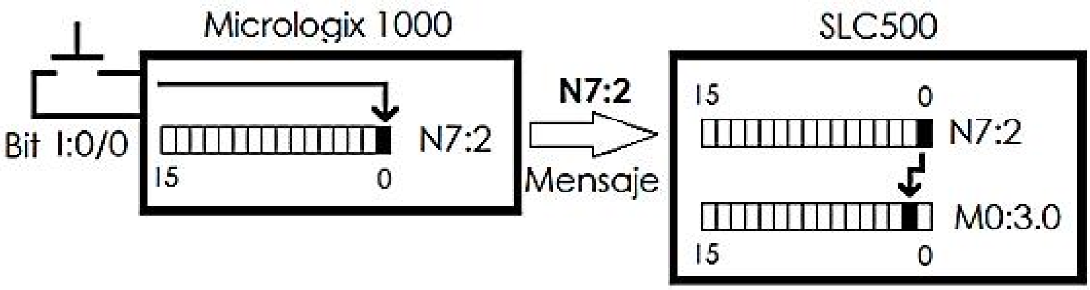


Figura 3.17. Flujo y transferencia de datos entre el Micrologix 1000 y el SLC 500.

3.2. LabVIEW®

El utilizar la red DeviceNet® como sistema independiente es una etapa para la integración total del sistema SCADA, por lo tanto, para conseguir la segunda etapa y poder integrarlas, falta la comunicación de la tarjeta PCI con un dispositivo de la red. El dispositivo utilizado para servir como ejemplo es el módulo de válvulas biestables.

3.2.1. Configuración

Se dio clic en Inicio>Programas>National Instruments>Measurement & Automation y se desplegó una ventana con un árbol de elementos, en la raíz “My system” se dio clic sobre el elemento “Devices and Interfaces”, se desplegaron más elementos de los cuales se seleccionó PCI-CAN, en la misma ventana aparecieron datos habilitados para ese elemento, se seleccionó el botón “Protocol” y se abrió una nueva ventana de edición en la cual se solicitaba seleccionar el protocolo de comunicación de la tarjeta de donde se eligió “DeviceNet”, tal como lo muestra la figura 3.18.

Figura 3.18. Selección del protocolo de comunicación en la tarjeta PCI.

En el árbol de elementos se activó un nuevo dispositivo dentro de la rama PCI-CAN llamado “DNET0”, al seleccionarlo se mostraron en la ventana botones y propiedades de la tarjeta, para la configuración, se seleccionó “Configurator” el cual desplegó una ventana en la que se solicitaba el nombre de la interfaz (DNET0), el número de nodo (4) y la velocidad de transferencia (125 Kbps) como lo muestra la figura 3.19.

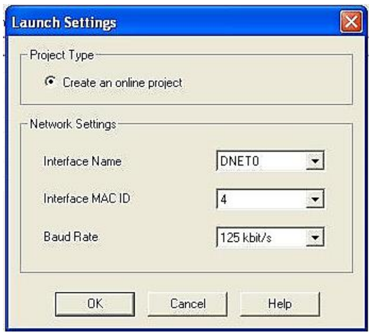


Figura 3.19. Ventana para configuración de la tarjeta.

Al aceptar los datos de configuración y después de haber escaneado cada uno de los dispositivos dentro de la red se envió un mensaje en el que se informaba del escaneo satisfactorio, al igual que en los programas de Rockwell Automation®, se solicitó una carga o descarga de datos pero para este caso se dio clic sobre la opción "Cancel". Se cerró la ventana de configuración y en la ventana donde se encontraba el árbol de elementos, al dar clic sobre el elemento DNET0 aparecieron los datos proporcionados en el configurador.

3.2.2. Comunicación

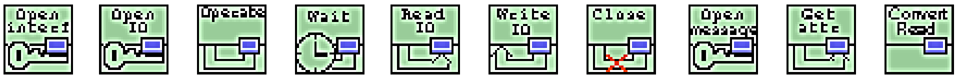
En LabVIEW® se puede establecer comunicación con dispositivos que sean conectados a través de los puertos de la computadora para manipularlos o ser manipulados según sea el objetivo; para la comunicación con la tarjeta PCI-DNET existe un módulo con bloques el cual se instaló en la versión 2010 de LabVIEW®, este módulo es llamado NI-DNET, localizado en "Instrument I/O" en la paleta de funciones.

El módulo de bloques NI-DNET permitió programar en LabVIEW® la configuración de dispositivo fuente (Tarjeta PCI) al dispositivo objetivo (módulo de válvulas biestables), velocidades de transferencia, tamaño de las entradas y salidas, entre otras, sin embargo, lo más importante fue la rutina que se debió ejecutar antes, durante y después de establecida la comunicación.

Los principales bloques para la comunicación de LabVIEW® son los siguientes, figura 3.20.

a) Open DeviceNet interface. Abre y configura una interfaz identificable en la red DeviceNet®.
b) Open DeviceNet IO. Abre y configura un objeto con entradas y salidas de la red DeviceNet®.
c) Operate DeviceNet Interface. Controla la comunicación con la red DeviceNet®.
d) Wait for state. Espera que ocurran uno o más eventos en el objeto elegido.

e) Read DeviceNet IO. Lee el objeto de entradas y salidas elegido.
f) Write DeviceNet IO. Escribe el objeto de entradas y salidas elegido.
g) Close Object. Cierra el objeto elegido.
h) Open DeviceNet explicit messaging. Abre y configura un objeto para utilizar la comunicación con mensajes explicitos.
i) Get DeviceNet attribute. Obtiene atributos de un dispositivo de la red DeviceNet® usando mensajes explicitos.
j) Convert from DeviceNet read. Convierte los datos leidos de la red DeviceNet® en datos procesables para LabVIEW®.



a) b) c) d) e) f) g) h) i) j)

Figura 3.20. Bloques de comunicación NI-DNET.

Todos estos bloques son VIs (subprogramas) que ayudan a controlar el correcto funcionamiento de la configuración, comunicación, programación y ejecución del programa. Cada bloque tiene un fin diferente y depende del programador proporcionar un eficiente procesamiento y control del flujo de datos.

3.2.3. Programación

El programa en diagrama de bloques que se desarrolló para establecer comunicación y operar al menos 2 válvulas del módulo de válvulas biestables, es el mostrado en la figura 3.21.

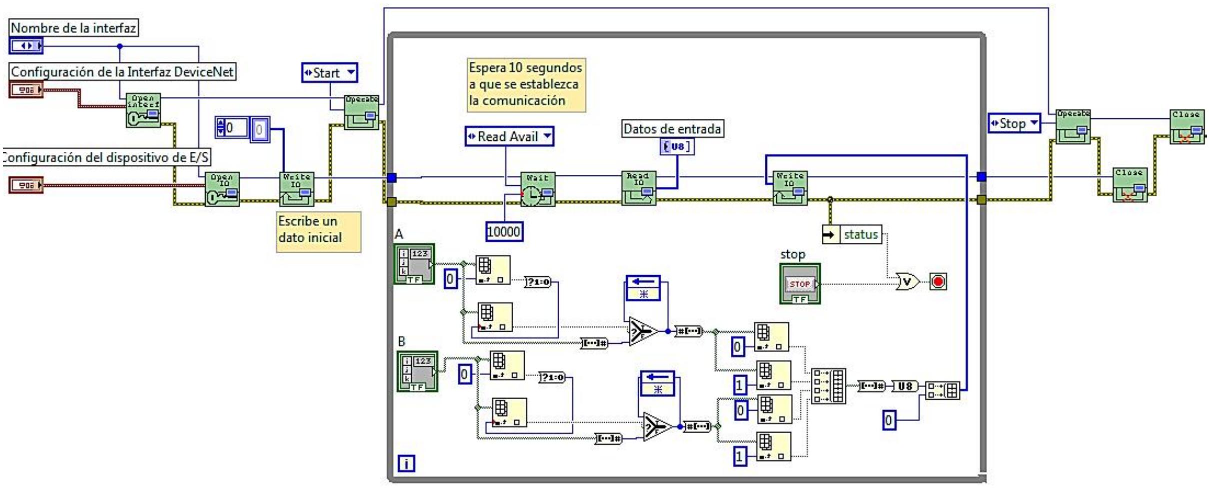


Figura 3.21. Programa en diagrama de bloques para el control de dos válvulas biestables.

En este programa se introdujo el nombre de la interfaz (dnet 0), número de nodo de la interfaz (nodo 4), velocidad de comunicación en la red (125 Kbps) y modo de acceso (automático) y para el dispositivo objetivo se introdujo el número de nodo (nodo 2), tipo de comunicación (Polled) y tamaño de las entradas y salidas (1 y 2 bytes respectivamente); cuando se estableció la comunicación se habilitaron dos arreglos booleanos con los que se simuló el pilotaje de las válvulas, es decir, cada arreglo tiene dos controles, al encender un control, el pilotaje de la válvula asociada a ese arreglo, se encendió. Si después de 10 segundos no se hubiera establecido la comunicación, el programa se hubiera cerrado automáticamente.

3.3. Sistema SCADA

3.3.1. Configuración

El uso de la red DeviceNet® como elemento independiente y el control de un dispositivo de la red teniendo a la tarjeta PCI como maestro, fueron 2 partes necesarias para poder implementar el sistema SCADA, sin embargo, la configuración para trabajar en conjunto no fue la misma que la que se hizo para trabajo independiente.

La tarjeta PCI es un elemento que se configuró para trabajar de punto a punto, es decir, no se puede comunicar al mismo tiempo con todos los elementos de la red, en teoría, con esta restricción no podría implementarse el sistema SCADA porque se necesita conocer la mayoría de los datos que estén fluyendo a través de la red, es por eso que se tiene que crear un punto por el cual todos los datos críticos deban de pasar forzosamente y ese mismo punto también debe de dar acceso a la tarjeta PCI.

El "punto de acceso" lo logró el escáner 1747 ya que es el elemento con el que se pueden comunicar un dispositivo del nivel de campo (SLC500) y los dispositivos del nivel de actuadores y sensores; de esta forma la tarjeta PCI sólo se comunicó con un dispositivo y no atrofió el funcionamiento general de la red.

3.3.2 Comunicación

La comunicación en el escáner fue la única que cambió respecto a las configuraciones y comunicaciones anteriores, los esclavos de este dispositivo eran los dos módulos de electroválvulas y el módulo del servomotor, sin embargo, la red estaba compuesta también por dos PLCs, el propio módulo escáner y la tarjeta PCI.

El proceso productivo elegido no sólo controlaba pistones y el servomotor, existían datos importantes que no tenían una interpretación física, como lo son las alarmas, los bits de registro usados en los PLCs o los sensores de final de carrera. El flujo de los datos críticos dentro de la red en el escáner se hizo con una nueva distribución en la memoria, es decir, los archivos M no sólo gestionarían a los dispositivos físicos (válvulas y servomotor) sino también a los datos controlados por el SLC500, el Micrologix 1000 y la tarjeta PCI.

La tarjeta PCI puede comunicarse con el escáner cuando éste actúa como esclavo, en cambio, cuando el escáner está como maestro la tarjeta no reconoce ninguna de sus propiedades de entrada y salida, para el proceso se necesita que sea esclavo de la tarjeta y maestro de los dispositivos de control al mismo tiempo.

El escáner tiene la capacidad de comunicarse como maestro o como esclavo dentro de la red, cuando actúa como maestro basta con agregar los dispositivos esclavos a su lista de escaneo y mapearlos en su memoria, pero cuando actúa como esclavo, el escáner aporta parte de sus archivos M para que sean utilizados para fines diferentes al control de los dispositivos esclavos. La distribución de datos y la cantidad de bytes asignados para ser esclavos depende del programador del sistema SCADA, si se utilizan pocos datos, la red tendrá menos tráfico y por lo tanto evitará retrasos de tiempo en la lectura y escritura de los datos.

Si se quiere encender una válvula desde el SLC500, se deben de utilizar los archivos "M0 maestros" para poder escribir en la salida del módulo de válvulas pero para poder encenderla desde la tarjeta PCI se necesitarían seguir 2 pasos, primero, escribir sobre los archivos "M1 esclavos" y después dar la orden al SLC500 de que lea el archivo M1 y lleve la información al archivo "M0 maestro" donde esté localizada la válvula. La tarjeta no puede dar la orden de encender directamente la válvula porque la válvula es esclava del escáner y nadie puede acceder a ella, a menos que el escáner esté fuera de servicio o la válvula no se encuentre en la lista de escaneo.

El flujo de información en el escáner se explica en la figura 3.22.

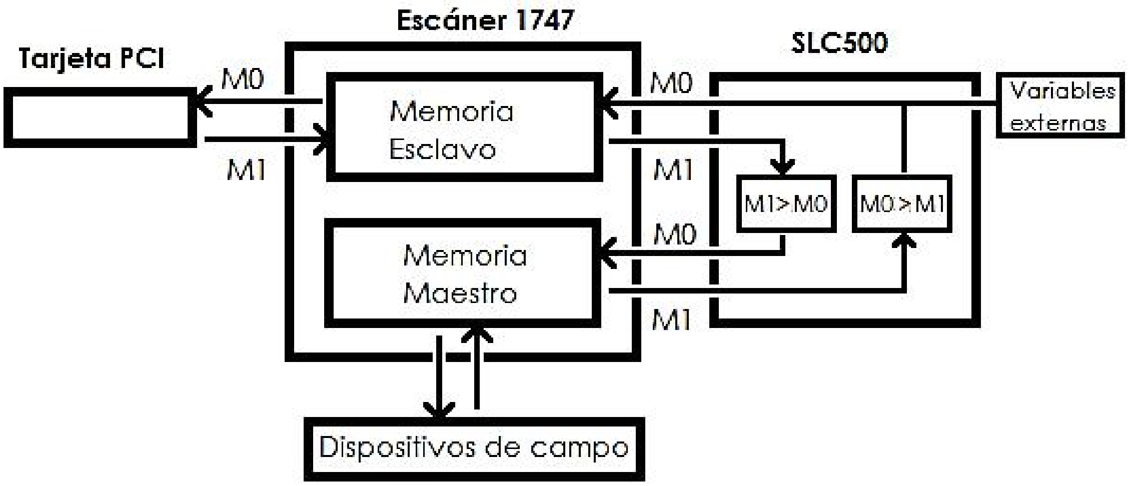


Figura 3.22. Flujo de información y comunicación a través del escáner 1747.

Los datos que se adquirieron de la tarjeta no sólo fueron de los dispositivos de campo, también intervino información de control para los programas del SLC500 y el micrologix 1000 y variables externas a la red como lo fueron los sensores de final de carrera y el botón de paro de emergencia, la distribución de palabras asignadas para ser esclavas y maestras tanto de entrada como de salida, se muestra en el cuadro 3.4.

Word	M1		M0	
0	Escritura bits de control del proceso		Lectura bits de control del proceso	
1	Servomotor Escritura		Monoestables Lectura	Biestables Lectura
2	Servomotor Escritura		Servomotor Lectura	
3	Servomotor Escritura		Servomotor Lectura	
4	Servomotor Escritura		Servomotor Lectura	
5	Monoestables Lectura	Biestables Lectura	Servomotor Lectura	
6	Servomotor Lectura		Monoestables escritura	
7	Servomotor Lectura		Biestables escritura	
8	Servomotor Lectura		Servomotor Escritura	
9	Servomotor Lectura		Servomotor Escritura	
10			Servomotor Escritura	
11			Servomotor Escritura	

Cuadro 3.4. Distribución de palabras en el escáner para sistema SCADA.

Archivos M1 esclavos (sombreado tenue).

- 2 bytes en donde se escribió la información trascendente para el control en los programas del SLC500 y el micrologix 1000.
- 8 bytes en donde la tarjeta escribió los parámetros necesarios para controlar el servomotor y los cuales el SLC500 trasladó a los archivos “M0 maestros” correspondientes al servomotor.

Archivos M1 maestros.

- 2 bytes de donde se leyeron las entradas de cada módulo de válvulas.
- 8 bytes de donde se leyó la información del módulo del servomotor.

Archivos M0 esclavos (sombreado tenue).

- 2 bytes de donde se leyeron las variables externas a la red y la información trascendente con la que se llevó a cabo el control de los programas del SLC500 y micrologix1000.
- 2 bytes de donde se leyeron las entradas de cada módulo de válvulas, trasladadas por el SLC500 desde los archivos “M1 maestros” correspondientes a las válvulas.
- 8 bytes de donde la tarjeta leyó la información resultante de los parámetros introducidos al servomotor, los cuales fueron trasladados por el SLC500 desde los archivos “M1 maestros” correspondiente al servomotor.

Archivos M0 maestros.

- 2 bytes en donde se escribieron las salidas del módulo de válvulas monoestables.
- 2 bytes en donde se escribieron las salidas del módulo de válvulas biestables.
- 8 bytes en donde se escribieron los parámetros del servomotor, transferidos por el SLC500 desde los archivos “M1 esclavos” correspondientes al servomotor.

La distribución se hizo en RSNetworx; una vez inicializada y configurada la red como se hizo en la primera etapa, se abrió la ventana de edición para el escáner 1747, en la pestaña “Module” se mostró el botón de “Slave Mode…”, al dar clic se desplegó una ventana en la que se seleccionó “Enable Slave Mode”, en la zona de “Polled” se eligió el tamaño de 10 bytes de entrada y 12 bytes de salida tal y como se planteó en el cuadro 3.4., en la figura 3.23 se muestra la ventana de edición.

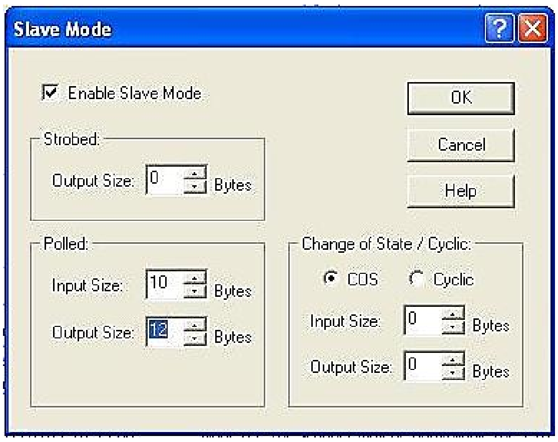


Figura 3.23. Ventana de edición de entradas y salidas esclavo en el escáner.

Al regresar a la ventana de edición del escáner, en la pestaña “Input” aparecieron todos los dispositivos habilitados contando los nuevos 10 bytes proporcionados por el escáner para ser esclavos; para crear un orden idéntico al planteado en el cuadro 3.4, se dio clic sobre una válvula y después sobre el botón “Unmap”, de esta forma se eliminaba el espacio en memoria asignado, lo mismo se hizo para la segunda válvula y el servomotor; una vez que todos los elementos fueron retirados, se mapeo nuevamente en los archivos M1 pero esta vez iniciando con los 10 bytes esclavos del escáner y después todos los demás dispositivos en el orden planteado anteriormente, se hizo lo mismo en la pestaña “Output” con los nuevos 12 bytes esclavos.

3.3.3. Programación

Para comenzar la programación, se creó una lista de características con las cuales el programa pudiera cumplir con los requerimientos de un sistema SCADA, las características elegidas fueron:

- Interfaz Humano – Máquina amigable.
- Visualización gráfica dinámica.
- Configuración y puesta en marcha rápida.
- Representación de señales de alarma.
- Almacenamiento de información histórica.

- Programación de eventos.
- Procesamiento de datos de alta resolución.
- Control remoto de instalaciones y equipo.
- Arquitectura abierta.
- Generación de datos históricos de señales.
- Cambio de parámetros que modifiquen tareas asociadas al autómata.

Siguiendo estas características se programó en LabVIEW® con ayuda de los módulos de DSC (Datalogging and Supervisory Control) y NI-DNET, el diagrama de bloques y el panel de control de LabView® para acondicionar los datos de la tarjeta PCI, así como también la programación en RSLogix del firmware para el PLC SLC500 y el PLC micrologix 1000. El flujo de información a través de todos los dispositivos del sistema SCADA es el que muestra la figura 3.24.

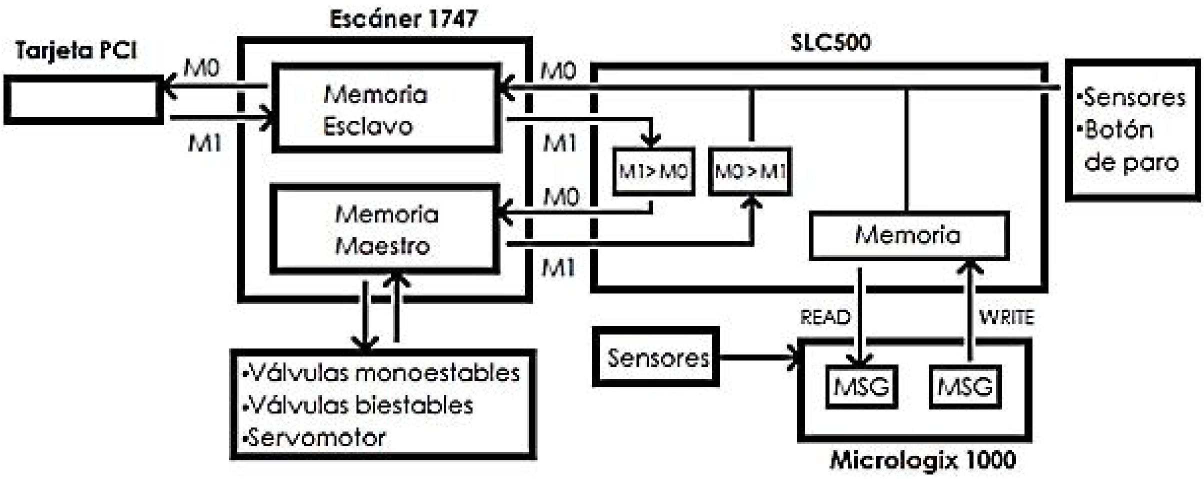


Figura 3.24. Flujo de información para el sistema SCADA.

Capítulo 4. Resultados

La implementación del sistema de control supervisor y de adquisición de datos en la red DeviceNet® usando el software LabVIEW® no sólo dio como resultado un sistema SCADA formado por el hardware y software de Rockwell Automation® y National Instruments®, aportó también el conocimiento teórico y sobre todo experimental en la implementación de una red industrial moderna, así como la instalación completa de la red en el laboratorio de automatización de la facultad de ingeniería, se aportó una innovación en la integración de dos tecnologías contemporáneas y de alto índice de desarrollo, creando de esta forma una posible solución a nivel industrial.

Cada característica del sistema SCADA que se desarrolló está descrita a continuación.

Interfaz Humano–Máquina amigable

Con el propósito de ayudar al usuario a usar el sistema, se colocaron elementos visualmente identificables como pistones, botellas de agua, sensores de final de carrera, un servomotor, bandas de transporte, indicadores tipo “gauge” de velocidad y posición e indicadores booleanos. Se creó un menú en el panel frontal con el que se pueden seleccionar diferentes submenús y posteriormente aplicaciones de propósito específico, el panel frontal se muestra en la figura 4.1.

Figura 4.1. Panel del sistema SCADA programado.

- Iniciar. Submenú que se encargó de contener a las aplicaciones de parametrización y puesta en marcha.

- Parametrizar. Aplicación donde se introdujeron los datos necesarios para el arranque del SCADA, la ventana de dicha aplicación se muestra en la figura 4.2.

Figura 4.2. Ventana de parametrización.

- Arrancar. Aplicación que inicializó, comunicó y puso en marcha la supervisión y adquisición del proceso.
- Desconectar. Desconectó la comunicación con el proceso sin salir de la aplicación SCADA.
- Salir. Cerró la aplicación SCADA completamente.

• Alarmas. Submenú en el cual se almacenaron los accesos a las aplicaciones para la supervisión y control de las alarmas.
  - Panel de alarmas. Aplicación en la que se supervisó y manipuló el estado de las alarmas dentro del proceso, la ventana se puede observar en la figura 4.3.

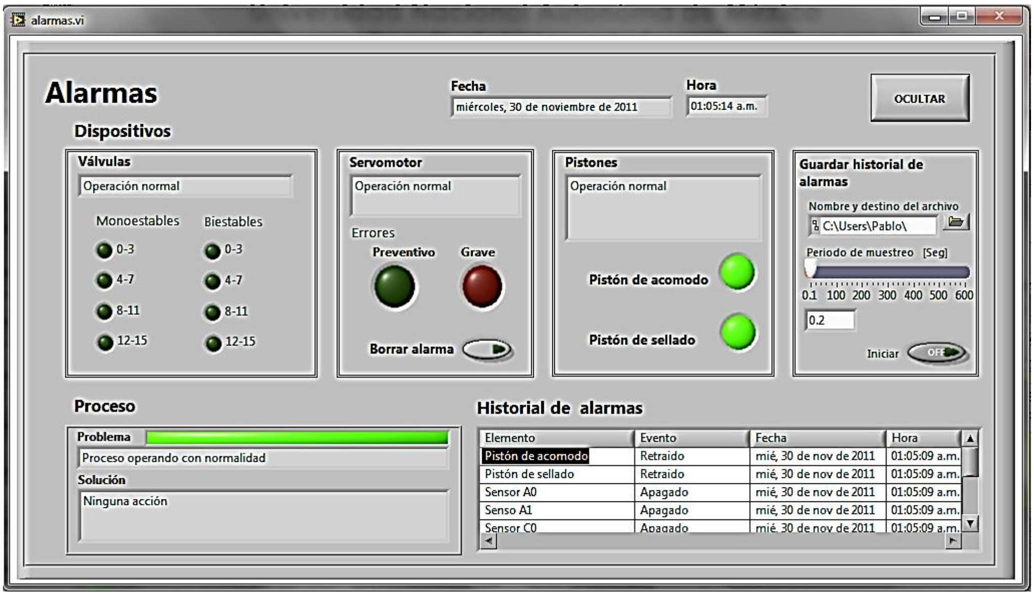


Figura 4.3. Ventana del panel de alarmas.

El panel de alarmas supervisó el estado de los dispositivos de campo los cuales están más propensos a fallar por el ambiente severo en el que están inmersos, no obstante, también se supervisó la comunicación entre los PLCs y el estado de la comunicación, se activó y temporizó el registro histórico de eventos dentro del proceso el cual pudo ser abierto en una hoja de cálculo, se desplegó una lista a detalle editada en tiempo real de cada uno de los elementos activos durante el proceso y en cuanto surgió alguna alarma, el panel se desplegó automáticamente en el panel frontal indicando el problema detectado y aportando al usuario una solución para el problema.

- Graficas de tendencia. Aplicación en la que se graficaron en tiempo real las variables numéricas del proceso.

- Mantenimiento. Submenú el cual contuvo las aplicaciones hechas para poder dar mantenimiento a los dispositivos de la red, siempre y cuando el elemento no estuviera actuando dentro del proceso.
  - Válvulas monoestables. Aplicación en la que se verificó el correcto funcionamiento del módulo de válvulas monoestables, la ventana se observa en la figura 4.4.

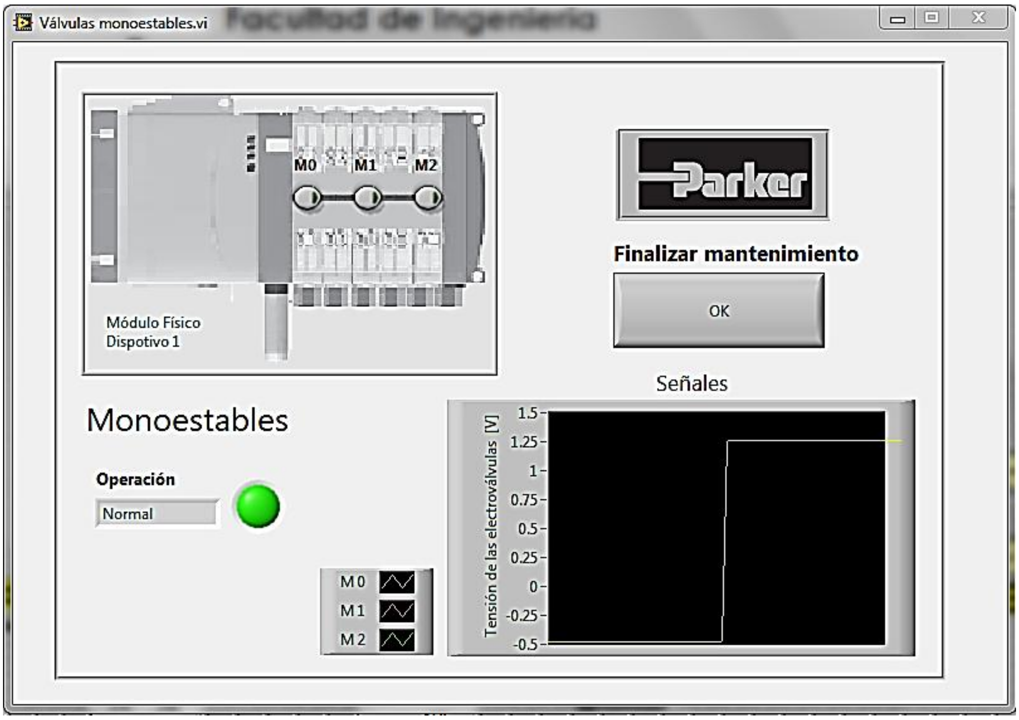


Figura 4.4. Ventana de mantenimiento para las válvulas monoestables.

  - Válvulas biestables. Aplicación en la que se verificó el correcto funcionamiento del módulo de válvulas biestables, a diferencia de la ventana de monoestables, ésta se programó con doble pilotaje y direccionamiento uniforme, es decir, no se podía encender los 2 pilotajes de una válvula simultáneamente, la ventana se observa en la figura 4.5.

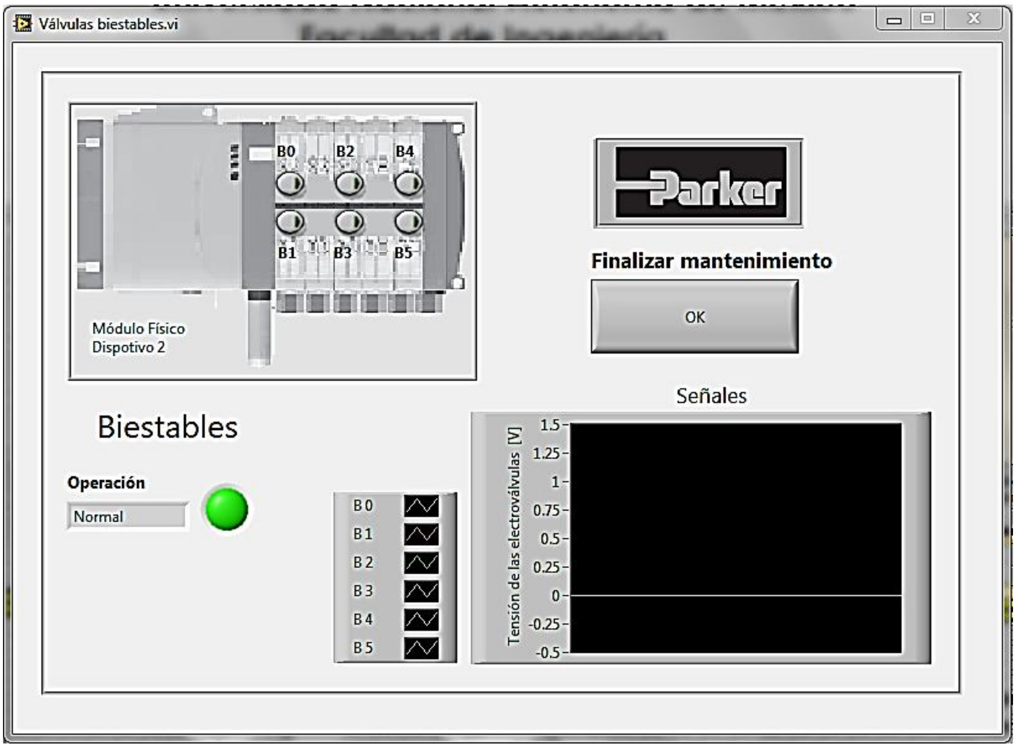


Figura 4.5. Ventana de mantenimiento para las válvulas biestables

- Servomotor. Aplicación en la que se dio mantenimiento especializado al módulo de servomotor, el panel se observa en la figura 4.6.
-

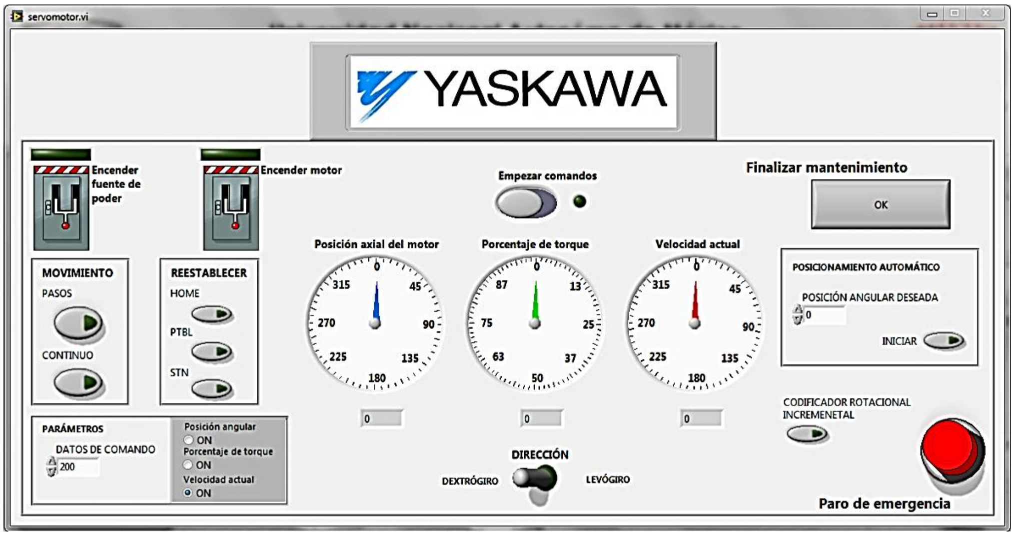


Figura 4.6. Ventana de mantenimiento especializado al servomotor.

Esta ventana comprobó el funcionamiento del servomotor, se usó para comprobar la correcta comunicación en los movimientos por pasos o continuo al introducir datos de comando, se realizaron mediciones con indicadores tipo “Gauge”, con “radio buttons” se seleccionó el tipo de medición a realizar, velocidad en RPM, porcentaje de torque utilizado o posición axial del motor en grados, se realizó un subprograma de posicionamiento con el que se introdujo un ángulo y el servomotor se posicionó automáticamente.

- Red. Aplicación que verificó la comunicación con la red DeviceNet® y los atributos de sus dispositivos, la ventana se muestra en la figura 4.7.

Figura 4.7. Ventana para comprobar el funcionamiento de la red.

La aplicación de esta ventana se encontró a todos los dispositivos que estuvieran dentro de la red y comprobó antes de poner en marcha el proceso que se cumpliera la comunicación en todos los elementos, se hizo un subprograma para obtener con base en mensajes explícitos los atributos de los dispositivos.

Visualización gráfica dinámica

Se agregó movimiento a los pistones de dosificado y de sellado cuando están retraídos o cuando están extendidos con ajustes avanzados, a las botellas de agua con los nodos de propiedad y ajustes avanzados se les agregó la capacidad de llenarse y de moverse sobre la banda de transporte, las tapas del dispositivo de sellado son móviles, para que durante el transcurso del proceso el usuario se sienta como si estuviera realmente en la planta.

Configuración y puesta en marcha rápida

Con la aplicación “Parametrizar” del submenú “Iniciar” en el panel frontal se facilitó la inicialización, teniendo la programación de guardar la última información utilizada con fines de uso posterior y evitar tiempo en la puesta en marcha, la aplicación “Arrancar” del mismo submenú, integra una fácil y sistemática puesta en marcha evitando al usuario intervenir en la comunicación.

Representación de señales de alarma

Tanto en el panel de alarma como en el panel frontal se programaron indicadores booleanos y de caracteres, los cuales avisan al usuario que ha surgido un problema en el proceso o en algún elemento del proceso, dependiendo del color del indicador es la gravedad del problema.

Almacenamiento de información histórica

En el panel de alarmas se programó una rutina en la que se pudo activar un registro histórico de la información crítica del proceso, esta rutina dio la oportunidad al usuario de decidir el tiempo de muestreo y la ubicación donde se guardaría el archivo para posteriormente ser abierto en una hoja de Excel.

Programación de eventos

Se programaron eventos los cuales son activados únicamente cuando es necesario, como es el caso de las alarmas, se despliega una ventana automáticamente cuando surge un problema.

Procesamiento de datos de alta resolución

Se realizó un perfil de velocidad el cual primero obtuvo información de hasta 32 bits (Datos de comando del Servomotor), se comparó con los datos introducidos y se reajustó hasta dar la respuesta esperada en los dispositivos, también se obtuvieron datos en crudo desde la red y se regresaron procesados para mejorar la respuesta del sistema en el tiempo.

Control remoto de instalaciones y equipo

Gracias a las capacidades de la red DeviceNet® el sistema cumplió con esta característica, ya que se puede estar alejado del proceso y no afectar el funcionamiento, esta capacidad se generó por las capacidades de la red mas no por el desarrollo del sistema SCADA, de no haber existido una red de esta capacidad la opción hubiera sido sistemas de radiofrecuencia o internet.

Arquitectura abierta

La programación en LabVIEW® y en los PLCs se hizo de tal forma que el sistema puede seguir creciendo, la mayor parte de la programación se hizo modular, así que se requiere agregar funciones o características se puede crear un módulo y adherirlo al ya existente.

Generación de datos históricos de señales

Se programó un visualizador que generó gráficas históricas de las señales de alta resolución las cuales están en continuo cambio durante el proceso, las gráficas son la velocidad generada por el servomotor, el gasto del fluido por la tobera de inyección y la posición del servomotor en el corrimiento de la banda, la ventana se muestra en la figura 4.8.

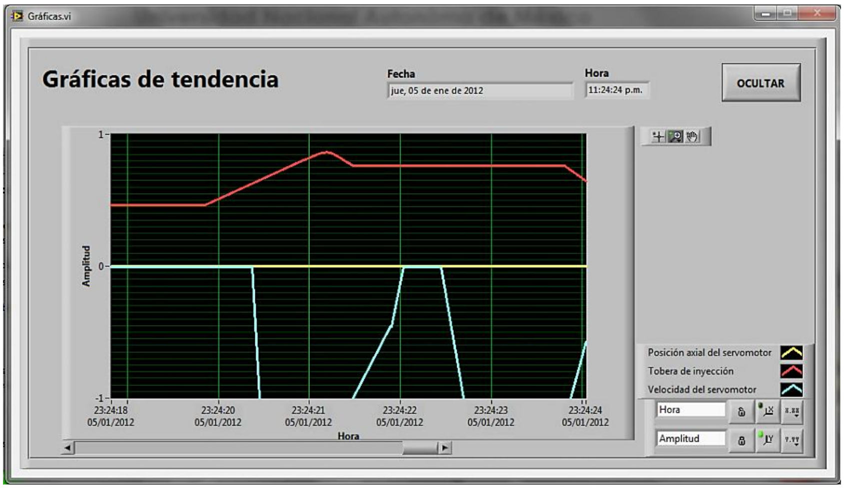


Figura 4.8. Ventana de gráficas de tendencia.

Cambio de parámetros que modifiquen acciones asociadas al autómata

El programa realizado en los PLCs y en la tarjeta PCI fue de tal forma que existen parámetros editables los cuales son capaces de influir en la velocidad, capacidad y cantidad de procesar botellas de agua.

Para ver detalladamente los programas elaborados en los PLCs SLC500 y micrologix 1000 a través de RSLogix y la tabla de distribución de memoria ver el anexo 1.

Para ver detalladamente el programa en LabVIEW®, los diagramas jerárquicos y la lista de elementos, se recomienda ver el anexo 2.

A pesar de parecer intuitiva la idea de unir dos sistemas como lo son la red DeviceNet® y LabVIEW®, la problemática suscitada en la unión no lo fue, se trató de hacer conforme al desarrollo natural de cada sistema y la integración de la tarjeta como si fuera un elemento más de la red, sin embargo, ninguno de los dos sistemas estaban habilitados para ser integrados, cada se ejecutaba correctamente pero de manera independiente, la idea del uso del escáner como recolector de información fue el punto culminante para el desarrollo del sistema SCADA, ya que de no haberse hecho de esta forma el alcance del trabajo sería menor.

Capítulo 5. Conclusiones y recomendaciones

Conclusiones

La implementación de una red industrial desde la conexión de hardware hasta la comunicación con el software, fue realizada satisfactoriamente, la red implementada es de gran uso en la industria de la automatización, los conceptos teóricos explicados facilitan la comprensión de la red, y la instalación completa de la red junto con la guía estructurada consolida el aprendizaje.

La tarjeta PCI fue creada para el control de dispositivos en la red DeviceNet®, por lo que su configuración, conexión y comunicación fue realizada satisfactoriamente.

El diseño del sistema SCADA cumplió con las características establecidas en el desarrollo del trabajo, se elaboró detalladamente cada una de ellas con el objetivo de satisfacer al sistema y apegarlo lo más posible a la realidad.

La conexión de la red DeviceNet® y la tarjeta PCI en el laboratorio de automatización industrial se realizó de tal forma que pudiera usarse no sólo para desarrollar el trabajo de tesis sino para posteriores enseñanzas de la materia, el trabajo desarrollado contiene material suficiente como para impartir teoría y desarrollar prácticas en la materia de Automatización Avanzada.

La configuración se realizó correctamente en cada uno de los elementos elegidos para entrar a la red, en el caso de las válvulas y el servomotor fue manualmente y para el arrancador, las interfaces DNI, el escáner y la tarjeta PCI fue a través de software.

La comunicación entre los sistemas implementados no fue una tarea fácil, el hecho de que los dispositivos utilizados fueran especialmente para aplicaciones con la red DeviceNet® no conllevó a que la comunicación fuera inmediata, cada dispositivo cumple con los mismos criterios de configuración, número de nodo, velocidad de comunicación, etc., sin embargo, la comunicación entre ellos es diferente, lo que presentó un problema en la integración y jerarquización de los dispositivos; las intensas búsquedas en manuales ayudaron a comunicar gran parte de la red, pero las características de los dispositivos y la necesidad del sistema fueron las que realmente abrieron las puertas a la innovación del SCADA.

La programación en diagrama de escalera del PLC SLC500 y el micrologix 1000 con el uso del software RSLogix presentó complicación cuando se introdujo la función avanzada MSG del micrologix 1000, esta función implicó más que sólo colocarla en el peldaño de programación, los bytes del bloque de control y los cambios físicos que ocurren dentro del PLC durante la ejecución perjudican a la comunicación y por lo tanto a los resultados programados en los PLCs.

La programación en la tarjeta PCI también tuvo dificultades ya que los módulos de NI-DNET y DSC, son módulos de programación que necesitan de cursos especializados en LabVIEW®, la investigación, exploración e iteración de los bloques del software ayudó a realizar una programación que cumpliera las características del SCADA, aunque el potencial del sistema se hubiera incrementado con un mejor conocimiento de estos módulos.

Recomendaciones

Debido a que uno de los resultados que se obtuvieron con el desarrollo del sistema SCADA fue la implementación de la red industrial DeviceNet®, se recomienda a la Facultad de Ingeniería que invierta más en estos tipos de sistemas, los cuales son ampliamente usados en el mundo laboral, los ingenieros de la Facultad de Ingeniería en el UNAM que seleccionan la rama de automatización deben de ser capaces de implementar y desarrollar sistemas como el que se plantea en este trabajo; estar a la vanguardia es un propósito de toda institución de educación, la UNAM no se puede eximir de esta responsabilidad y debe procurar que sus alumnos tengan conocimientos que tienen alumnos de otras instituciones educativas en el mundo.

La tecnología que se implementa durante el desarrollo no es tan compleja en comparación con la que se podría desarrollar en la industria, sin embargo, la esencia del sistema nunca cambia, es por eso que se recomienda a la Facultad de Ingeniería que consiga cursos especializados a los académicos que estén interesados en este tema para que se transfiera la información a los alumnos y no pasen los problemas que se presentaron en este trabajo.

La constante demanda de tecnología para facilitar las tareas es uno de los problemas que se deben de abordar para lograr cambios notables en la industria y es este motivo por el que hace altamente recomendable el implementar estos conocimientos en un proceso real de la industria para ser desarrollado en un trabajo de tesis.

Referencias

Enciclopedias, revistas y publicaciones electrónicas

- [1] Facultad de Ingeniería, UNAM: Misión y visión. Consultado el 17 de agosto de 2011. http://www.ingenieria.unam.mx/paginas/misionVision.htm.

- [2] Real Academia Española: Automatizar. Consultado el 17 de agosto de 2011.http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=automatizaci%C3%B3n.

- [3] Mundo HVACR: Automatización. Consultado el 17 de agosto de 2011. http://issuu.com/dogmedia/docs/mundo-hvacr-abril-2010.

- [4] SALAZAR, Alejandro: Automatizació. ITESM. Consultado el 17 de agosto de 2011. http://sifunpro.tripod.com/automatizacion.htm.

- [5] ALEGSA: Automatización. Consultado el 18 de agosoto de 2011. http://www.alegsa.com.ar/Definicion/de/automatizar.php.

- [6] BARRAGAN, Antonio: Bus de Sensores y Actuadores AS-interface. UHU. Consultado el 29 de Agosto de 2011. http://www.uhu.es/antonio.barragan/book/export/html/125.

- [7] ODVA. Tutorial DeviceNet. SMAR. Consultado el 31 de Agosto de 2011. http://www.smar.com/devicenet.asp.

- [8] BOSCH, Robert: Can Specification. Bosch. Consultado el 31 de agosto de 2011. http://esd.cs.ucr.edu/webres/can20.pdf

- BALCELLS, Joseph: Autómatas Programables. Alfaomega. Consultado el 23 de agosto de 2011. http://books.google.com/books?id=xfsSjADge70C&pg=PT127&hl=es&source=gbs_toc_r&cad=4#v=onepage&q&f=false.

- GEA, José Manuel: Circuitos Básicos de Ciclos Neumáticos y Electroneumático. Alfaomega. Consultado el 25 de Agosto de 2011.

- Bruchi Alejandro: Automatización. Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México. Consultado el 26 de agosto. http://sifunpro.tripod.com/automatizacion.htm.

- ITESM: Historia de la Automatización. Instituto Tecnológico y de Estudios Superiores de Monterrey. Consultado el 26 de agosto de 2011. www.itescam.edu.mx/principal/sylabus/fpdb/recursos/r14059.DOC.

- UNIVERSIDAD DEL PAIS VASCO: Neumática y Electroneumática. EHU. Consultado el 28 de Agosto de 2011. http://www.ehu.es/inwmooqb/NEUMATICA/Neumatica%20y%20electroneumatica/CAP4.%20Valvulas%20neumaticas.pdf.

- FIROOZIAN, Riazollah: Servo Motors and Industrial Control Theory. Springer. Consultado el 29 de Agosto de 2011. http://books.google.com.mx/books?id=MnOOnHJwVVQC&printsec=frontcover&dq=servomotors&hl=es&ei=ZYncTprfDMuGsgLBlcXzDQ&sa=X&oi=book_result&ct=result&resnum=1&ved=0CDEQ6AEwAA#v=onepage&q&f=false.

- AULA ELECTRONICA: Automatismo Cableados, Jerarquía de la Automatización Industrial. Consultado el 30 de Agosto de 2011. http://guindo.pntic.mec.es/rarc0002/all/aut/dat/ace.jerarquia.aut.pdf.

- UNIVERSIDAD CENTRAL DE VENEZUELA. Redes Industriales. UCV. Consultado el 30 de Agosto de 2011. http://neutron.ing.ucv.ve/revista-e/No4/RCI.html.

- UNIVERSIDAD DE VALENCIA. Sistemas Industriales Distribuidos, Redes de comunicación Inndustriales. UV. Consultado el 30 de Agosto de 2011. http://www.uv.es/rosado/sid/Capitulo3_rev0.pdf.

- GRUPO DE TECNOLOGIA ELECTRONICA. Tutorial LabVIEW. Escuela Superior de Ingenieros de Sevilla. Consultado el 2 de Septiembre de 2011. http://www.gte.us.es/ASIGN/IE_4T/Tutorial%20de%20Labview.pdf.

- MENDIBURU, Henry: SISTEMAS SCADA. El Galeón. Consultado el 5 de Septiembre de 2011. http://www.galeon.com/hamd/pdf/scada.pdf.

- TARRAGA, Federico: Diseño de infraestructuras informáticas. Wordpress. Consutlado el 5 de septiembre de 2011. http://ftarraga.wordpress.com/que-hago/.

Manuales y catálogos

- ALLEN BRADLEY: Sistema de cables DeviceNet (Núm. cat. DN-6.7.2ES). Rockwell Automation. Consultado el 23 de Marzo de 2011. http://samplecode.rockwellautomation.com/idc/groups/literature/documents/um/dn-um072_-es-p.pdf

- INDUSTRIAL AUTOMATION: DeviceNet Media. Turck. Consultado el 23 de Marzo de 2011. http://www.clrwtr.com/PDF/TURCK/TURCK-DeviceNet-Cables.pdf

- ALLEN BRADLEY: 1747-SDN DeviceNet Scanner Module (Catalog Number 1747-SDN, Series C). Rockwell Automation. Consultado el 26 de Marzo de 2011.

- ALLEN BRADLEY: DeviceNet Manager Software User Manual (Catalog Number 1787-MGR). Rockwell Automation. Consultado el 28 de Marzo de 2011.

- ALLEN BRADLEY: DeviceNet Scanner Configuration Manual(Catalog Number 1784-PCIDS-CPCIDS). Rockwell Automation. Consultado el 28 de Marzo de 2011.

- ALLEN BRADLEY: DeviceNet RS_232 Interface Module (Catalog Number 1770-KFD and 1770-KFDG). Rockwell Automation. Consultado el 5 de Abril de 2011.

- ALLEN BRADLEY: Start Up Guide for UD77 DeviceNet Systems. Rockwell Automation. Consultado el 15 Abril de 2011.

- ALLEN BRADLEY: SLC 500 Systems Selection Guide (Bulletin 1746 and 1747). Rockwell Automation. Consultado el 21 Abril de 2011

- ALLEN BRADLEY: MicroLogix_ 1000 Programmable Controllers, User manual (Bulletin 1761 Controllers). Rockwell Automation. Consultado el 21 de Abril de 2011.

- ALLEN BRADLEY: DeviceNet RS_232 Interface Module (Catalog Number 1770-KFD and 1770-KFDG). Rockwell Automation. Consultado el 24 de Abril de 2011.

- FIELDBUS SYSTEMS: Moduflex Fieldbus System. Parker Automation. Consultado el 17 de Mayo de 2011.

- JUSP-NS300 INDEXER: Sigma II DeviceNet Module - NS300, Connectivity for Single-Axis Positioning. Yaskawa Electric. Consultado 25 de Mayo de 2011.

- JUSP-NS300 INDEXER: Sigma II DeviceNet Communication. Yaskawa Electric. Consultado 25 de Mayo de 2011.

- JUSP-NS300 INDEXER: DeviceNet Interface Unit, User's Manual. Yaskawa Electric. Consultado 28 de Mayo de 2011.

- JUSP-NS300 INDEXER: Sigma II Indexer User's Manual. Yaskawa Electric. Consultado 30 de Mayo de 2011.

- SIGMA CATALOG: Sigma II Servo System Product Catalog Supplement. Yaskawa Electric. Consultado el 30 de Mayo de 2011.

- SGMH SERVOMOTORS: 100/200V Single-phase Sigma II Servo Systems. Yaskawa Electric. Consultado el 3 de Junio de 2011.

- SGMH SERVOMOTORS: Super High Power Rate Series SGMAH Servomotors. Yaskawa Electric. Consultado el 3 de Junio de 2011.

- DEVICENET MASTER INTERFACES: NI PCI-DNET, NI PXI-8461, NI PCMCIA-DNET. National Instruments. Consultado el 18 de Junio de 2011. http://www.ni.com/pdf/products/us/4ic751-752.pdf.

- NI-DNET: Programmer Reference Manual. National Instruments. Consultado el 20 de Junio de 2011. http://www.ni.com/pdf/manuals/370376c.pdf.

- NI-DNET: User Manual. National Instruments. Consultado el 3 de Julio de 2011. http://www.ni.com/pdf/manuals/370375b.pdf.

- LABVIEW: W Datalogging and Supervisory Control Module Developer's Manual. National Instruments. Consultado el 5 de julio de 2011.

- LABVIEW: Getting Started with the LabVIEW Datalogging and Supervisory Control Module. National Instruments. Consultado el 6 de julio de 2011.

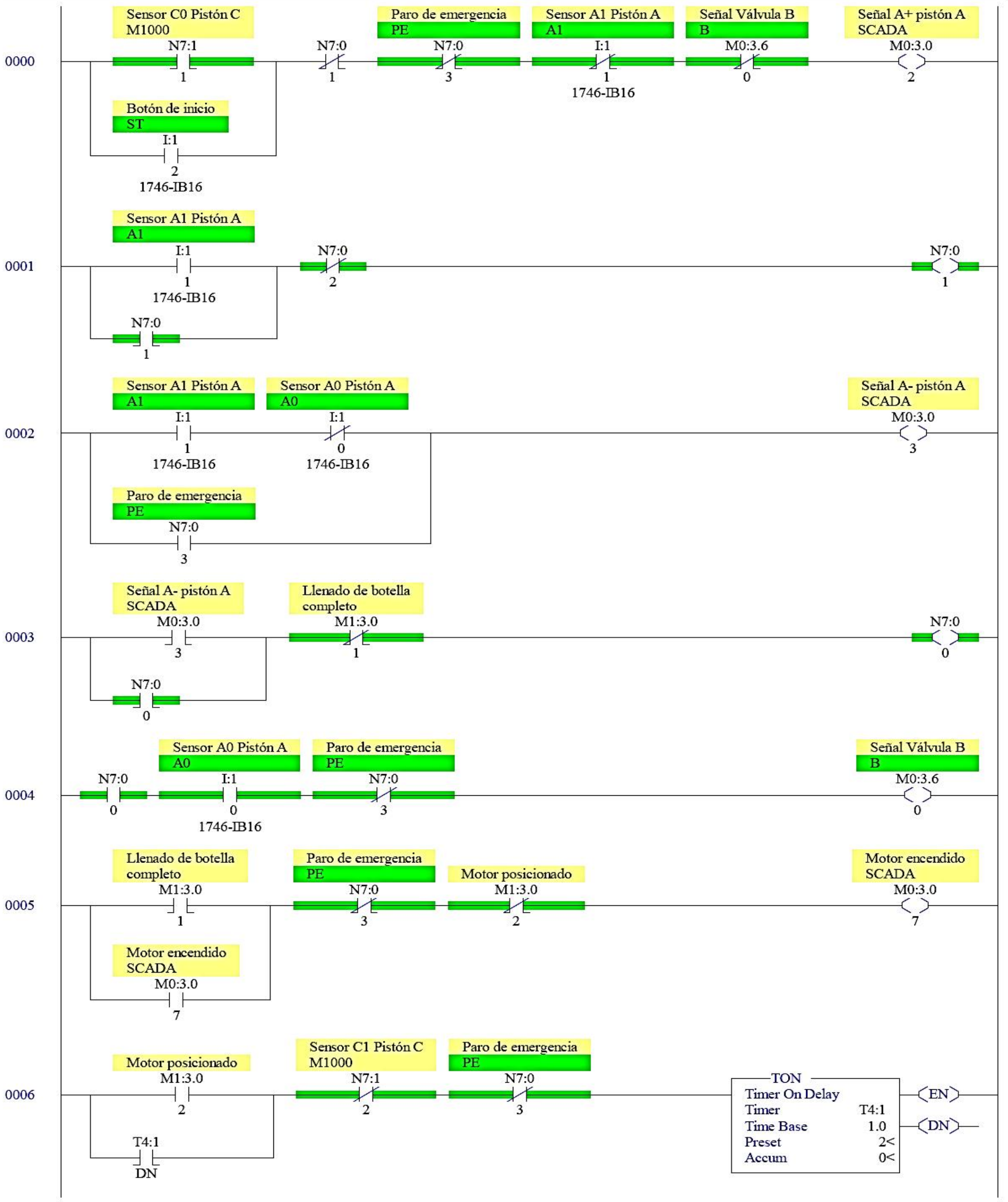
Anexos

Anexo 1. Programas para el control de los PLCs SLC 500 y Micrologix 1000 y cuadros de distribución de memoria

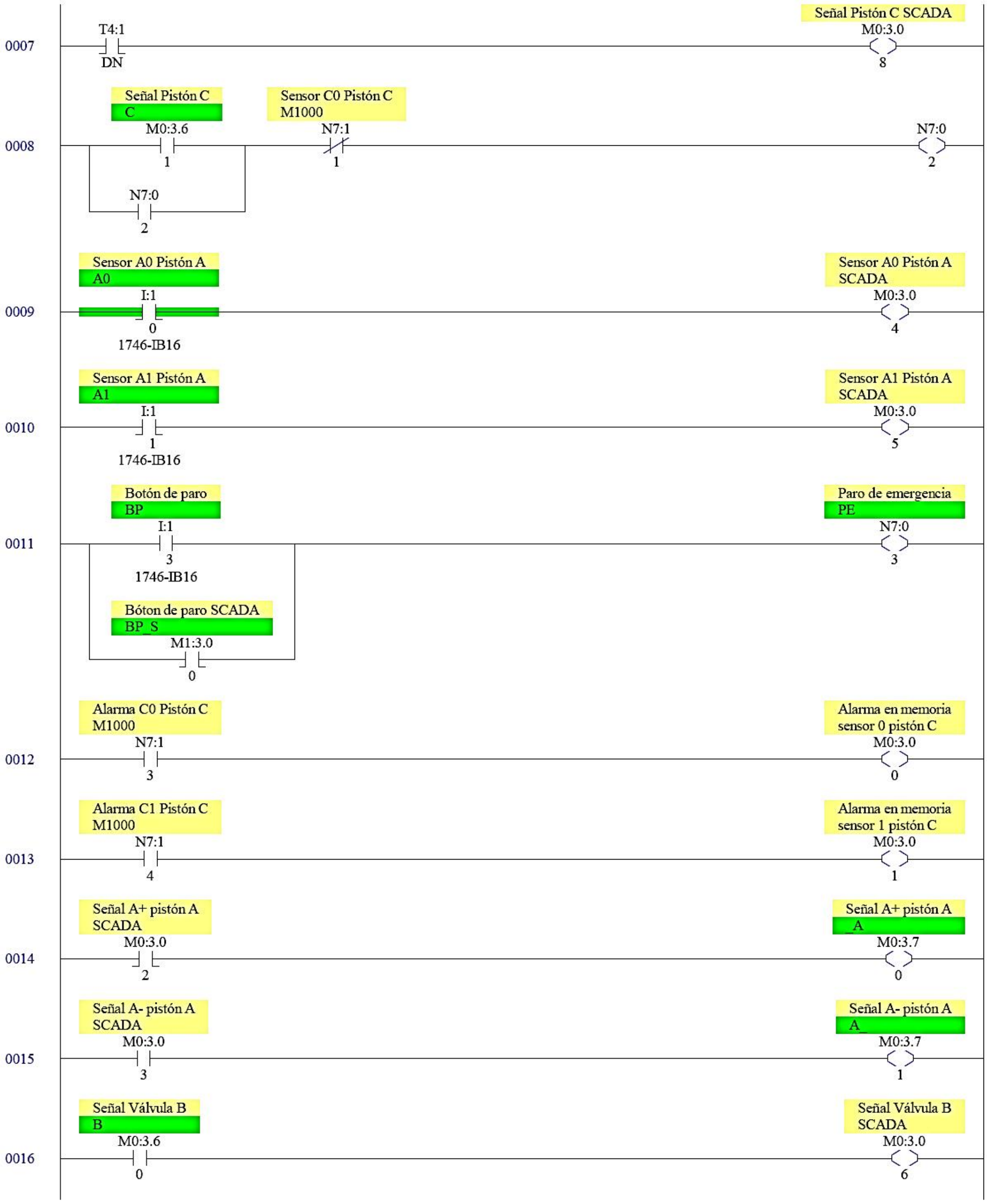
Programa desarrollado en RSLogix para el PLC SLC500

SLC500 FINAL

LAD 2 - --- Total Rungs in File = 34

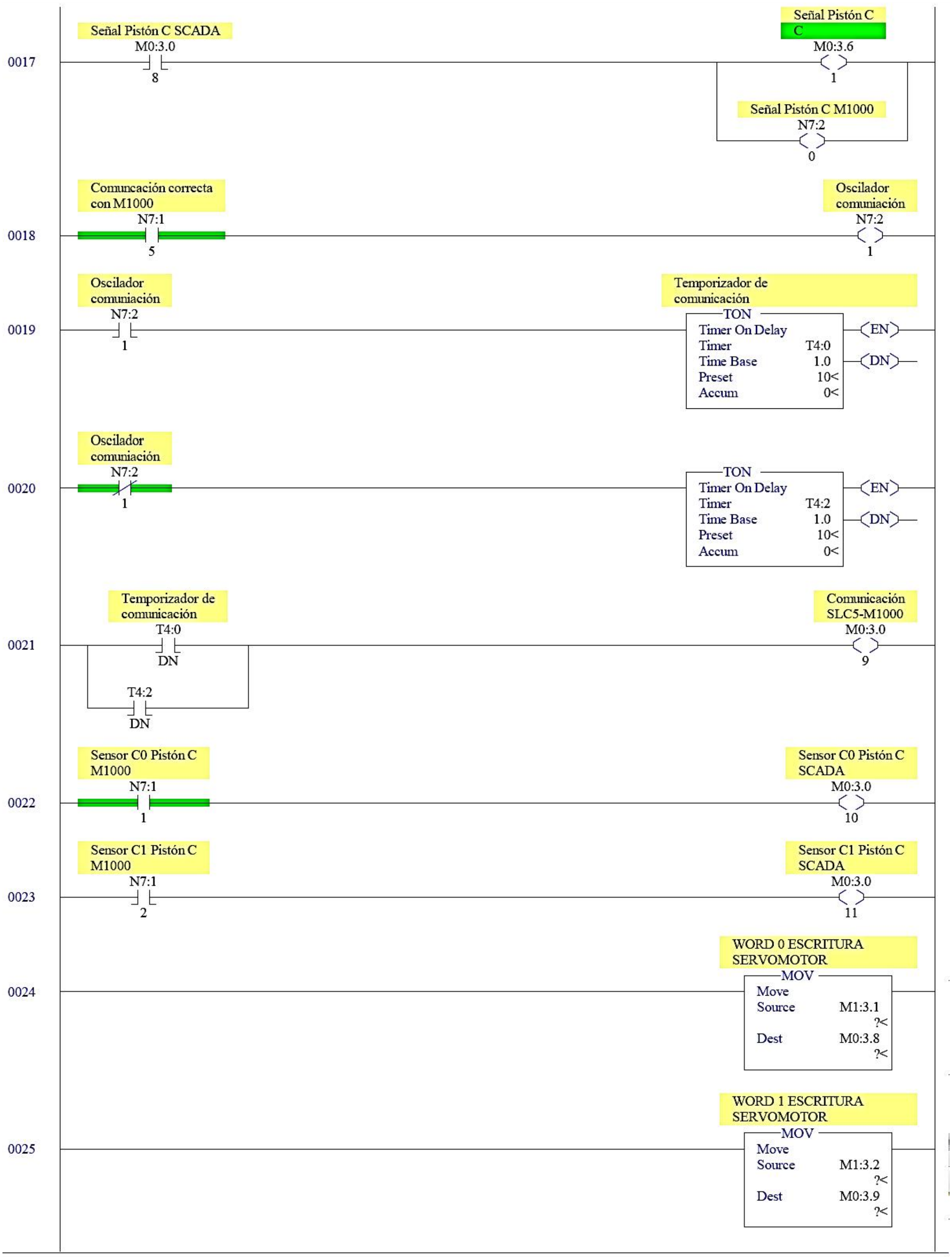


Page 1 Tuesday, November 22, 2011 - 17:46:42

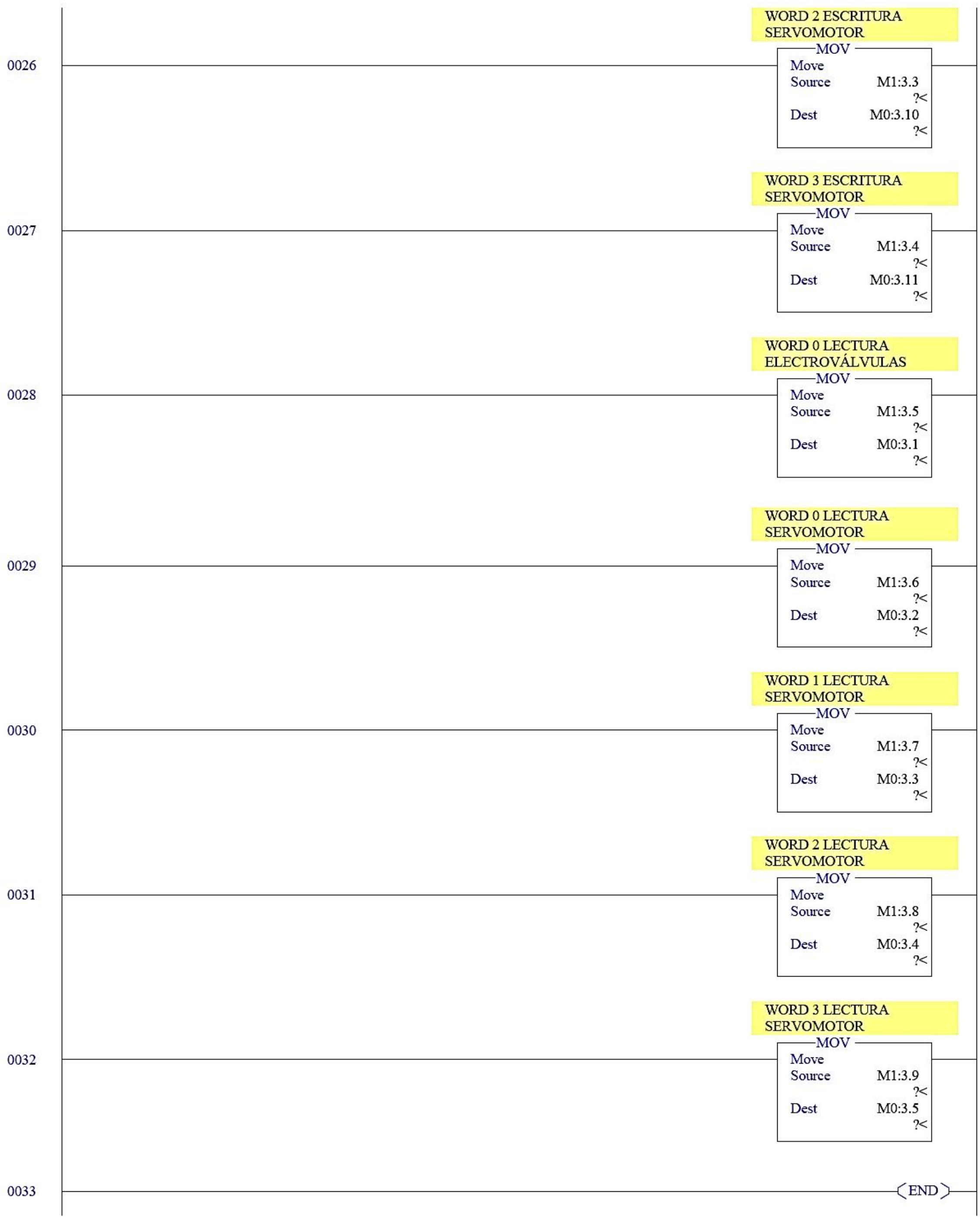

0007
T4:1
DN
Señal Pistón C SCADA
M0:3.0
8
0008
Señal Pistón C
C
M0:3.6
1
N7:0
2
Sensor C0 Pistón C
M1000
N7:1
1
N7:0
2
0009
Sensor A0 Pistón A
A0
I:1
0
1746-IB16
Sensor A0 Pistón A
SCADA
M0:3.0
4
0010
Sensor A1 Pistón A
A1
I:1
1
1746-IB16
Sensor A1 Pistón A
SCADA
M0:3.0
5
0011
Botón de paro
BP
I:1
3
1746-IB16
Bóton de paro SCADA
BP_S
M1:3.0
0
Paro de emergencia
PE
N7:0
3
0012
Alarma C0 Pistón C
M1000
N7:1
3
Alarma en memoria
sensor 0 pistón C
M0:3.0
0
0013
Alarma C1 Pistón C
M1000
N7:1
4
Alarma en memoria
sensor 1 pistón C
M0:3.0
1
0014
Señal A+ pistón A
SCADA
M0:3.0
2
Señal A+ pistón A
_A
M0:3.7
0
0015
Señal A- pistón A
SCADA
M0:3.0
3
Señal A- pistón A
A_
M0:3.7
1
0016
Señal Válvula B
B
M0:3.6
0
Señal Válvula B
SCADA
M0:3.0
6

SLC500 FINAL

LAD 2 - --- Total Rungs in File = 34



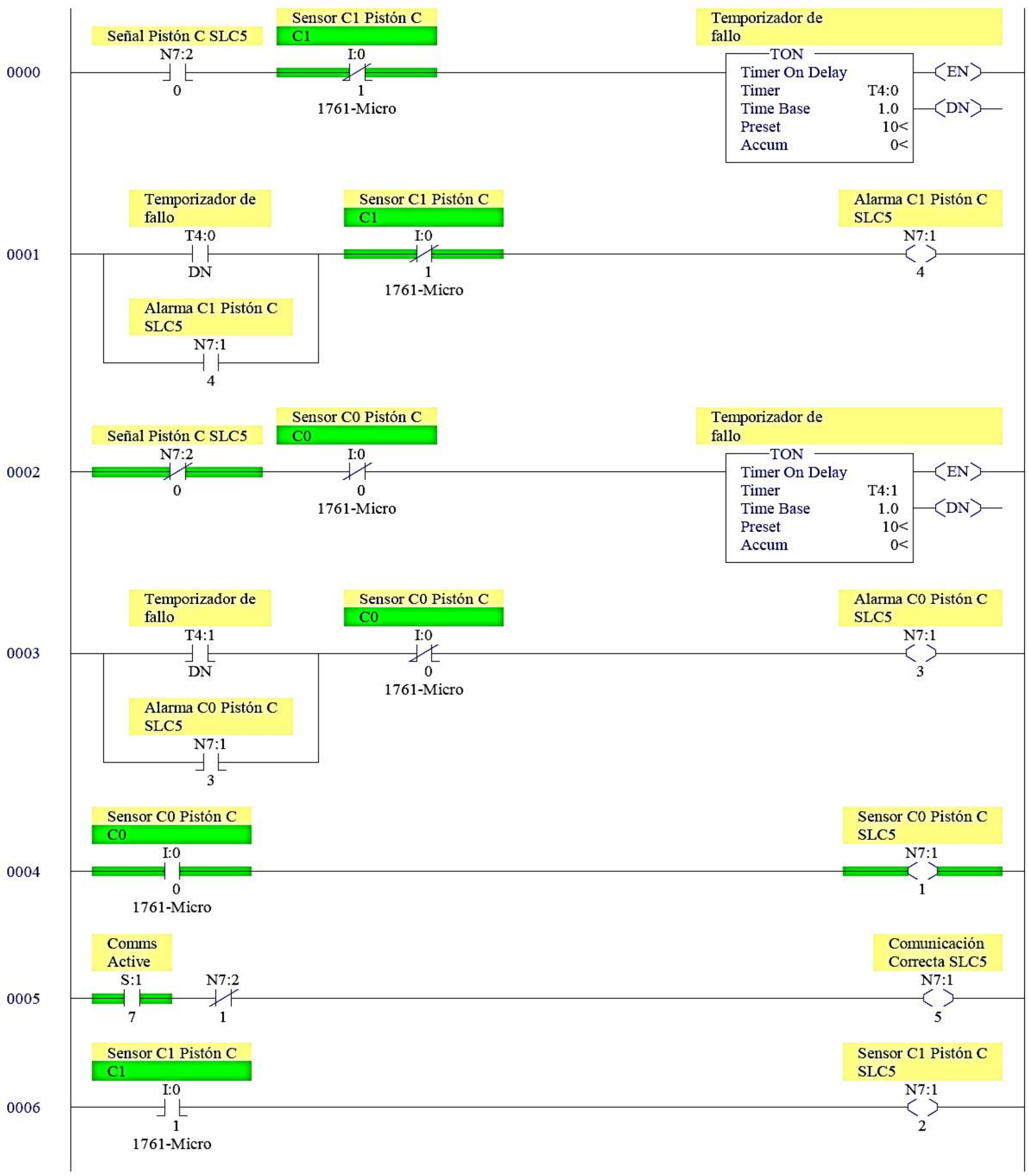
Page 3 Tuesday, November 22, 2011 - 17:46:42


0026
WORD 2 ESCRITURA SERVOMOTOR
MOV
Move
Source M1:3.3
?<
Dest M0:3.10
?<
0027
WORD 3 ESCRITURA SERVOMOTOR
MOV
Move
Source M1:3.4
?<
Dest M0:3.11
?<
0028
WORD 0 LECTURA ELECTROVÁLVULAS
MOV
Move
Source M1:3.5
?<
Dest M0:3.1
?<
0029
WORD 0 LECTURA SERVOMOTOR
MOV
Move
Source M1:3.6
?<
Dest M0:3.2
?<
0030
WORD 1 LECTURA SERVOMOTOR
MOV
Move
Source M1:3.7
?<
Dest M0:3.3
?<
0031
WORD 2 LECTURA SERVOMOTOR
MOV
Move
Source M1:3.8
?<
Dest M0:3.4
?<
0032
WORD 3 LECTURA SERVOMOTOR
MOV
Move
Source M1:3.9
?<
Dest M0:3.5
?<
0033
END

Programa desarrollado en RSLogix para el PLC Micrologix 1000

M1000 FINAL

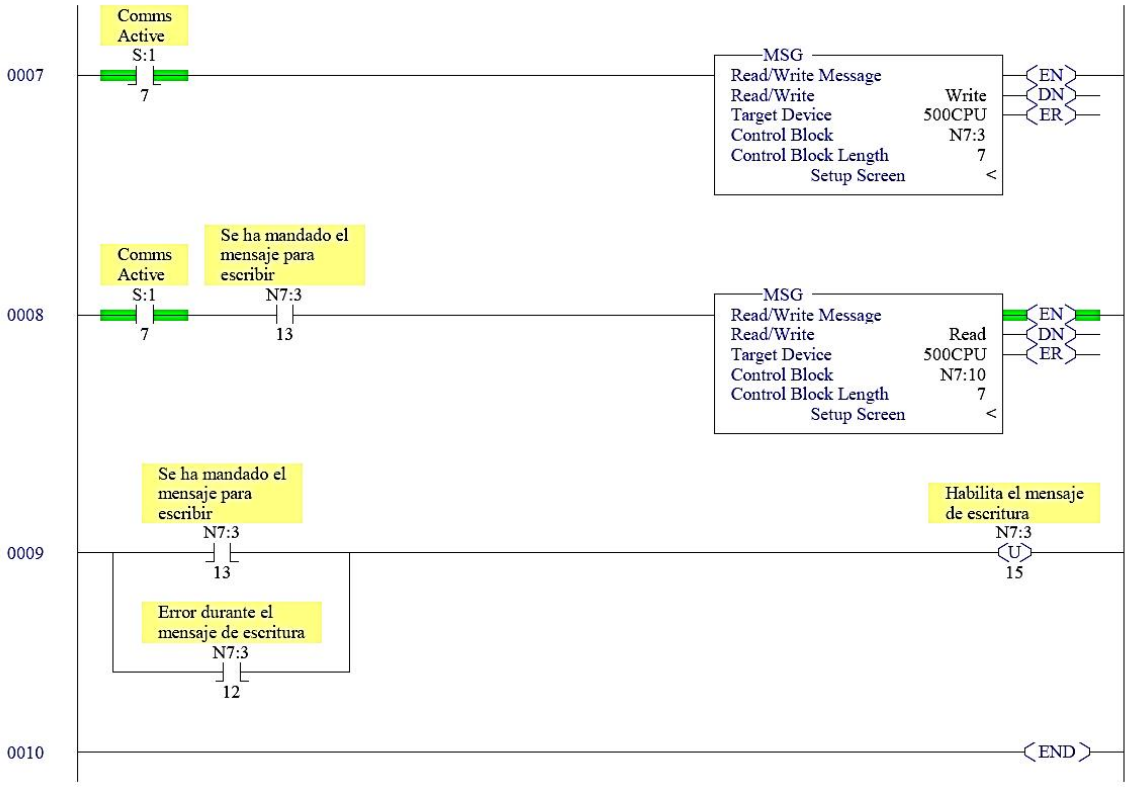
LAD 2 - MAIN_PROG --- Total Rungs in File = 11



Page 1 Tuesday, November 22, 2011 - 17:42:13

M1000 FINAL

LAD 2 - MAIN_PROG --- Total Rungs in File = 11



Page 2 Tuesday, November 22, 2011 - 17:42:13

Cuadro de distribución de memoria para el programa del SLC500

SLC500				
Descripción	Entradas físicas	Descripción	Memoria	Observación
Sensor A0 Pistón A	I:1/0	Bits de registro programa	N7:0/0	
Sensor A1 Pistón A	I:1/1	Bits de registro programa	N7:0/1	
Botón de inicio	I:1/2	Bits de registro programa	N7:0/2	
Botón de paro	I:1/3	Paro de emergencia	N7:0/3	
		Sensor C0 Pistón C M1000	N7:1/1	Lectura
		Sensor C1 Pistón C M1000	N7:1/2	Lectura
		Alarma C0 Pistón C M1000	N7:1/3	Lectura
		Alarma C1 Pistón C M1000	N7:1/4	Lectura
		Señal Pistón C M1000	N7:2/0	Escritura
		Oscilador Comunicación	N7:2/1	Escritura

Cuadro de movimiento de palabras dentro del SLC500 para configuración adaptada en el sistema SCADA

Movimiento de registros				
Word	Descripción		Word	Descripción
M1:3.1	MOT WRITE desde PCI	>	M0:3.8	MOT WRITE DIRECTO
M1:3.2	MOT WRITE desde PCI	>	M0:3.9	MOT WRITE DIRECTO
M1:3.3	MOT WRITE desde PCI	>	M0:3.10	MOT WRITE DIRECTO
M1:3.4	MOT WRITE desde PCI	>	M0:3.11	MOT WRITE DIRECTO
M1:3.5	VALVULAS READ	>	M0:3.1	VAL READ para PCI
M1:3.6	SERVOMOTOR READ	>	M0:3.2	MOT READ para PCI
M1:3.7	SERVOMOTOR READ	>	M0:3.3	MOT READ para PCI
M1:3.8	SERVOMOTOR READ	>	M0:3.4	MOT READ para PCI
M1:3.9	SERVOMOTOR READ	>	M0:3.5	MOT READ para PCI

Cuadro de distribución de memoria para el programa del Micrologix 1000

Micrologix 1000				
Descripción	Entradas físicas	Descripción	Memoria	Observación
Sensor C0 Pistón C	I:1/0	Sensor C0 Pistón C SLC5	N7:1/1	Escritura
Sensor C1 Pistón C	I:1/1	Sensor C1 Pistón C SLC5	N7:1/2	Escritura
		Alarma C0 Pistón C SLC5	N7:1/3	Escritura
		Alarma C1 Pistón C SLC5	N7:1/4	Escritura
		Comunicación correcta con SLC5	N7:1/5	Escritura
		Señal Pistón C SLC5	N7:2/0	Lectura
		Oscilador comunicación	N7:2/1	Lectura
		Bloque de control Escritura	N7:3-9	
		Bloque de control Lectura	N7:10-16	

Cuadro de distribución de los bits de control del proceso SCADA en el escáner 1747

Escáner						
Descripción	M0		Descripción	M1		Observación
	WORD	BIT		WORD	BIT	
Alarma sensor 0 pistón C SCADA	0	0	Botón de paro SCADA	0	0	Enclavado
Alarma sensor 1 pistón C SCADA	0	1	Llenado completo	0	1	Pulso
Seña A+ Pistón A SCADA	0	2	Motor posicionado	0	2	Pulso
Señal A- Pistón A SCADA	0	3				
Sensor A0 Pistón A SCADA	0	4				
Sensor A1 Pistón A SCADA	0	5				
Válvula B extendido SCADA	0	6				
Motor encendido SCADA	0	7				
Señal Pistón C SCADA	0	8				
Comuncación SLC5-M1000	0	9				
Sensor C0 Pistón C SCADA	0	10				
Sensor C1 Pistón C SCADA	0	11				

Anexo 2. Programa para sistema SCADA en LabVIEW

Panel principal

Lista de VIs

Write DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Write DeviceNet IO.vi

Read DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Read DeviceNet IO.vi

Wait For State.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Wait For State.vi

Simple Error Handler.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\error.llb\Simple Error Handler.vi

Close Object.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Close Object.vi

Operate DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Operate DeviceNet Interface.vi

Open DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet IO.vi

DnetIOConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIOConfig.ctl

Open DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet Interface.vi

DnetIntfConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIntfConfig.ctl

Red.vi
C:\Users\Pablo\Documents\Tesis\SCADA\Red.vi

alarmas.vi
C:\Users\Pablo\Documents\Tesis\SCADA\alarmas.vi

servomotor.vi
C:\Users\Pablo\Documents\Tesis\SCADA\servomotor.vi

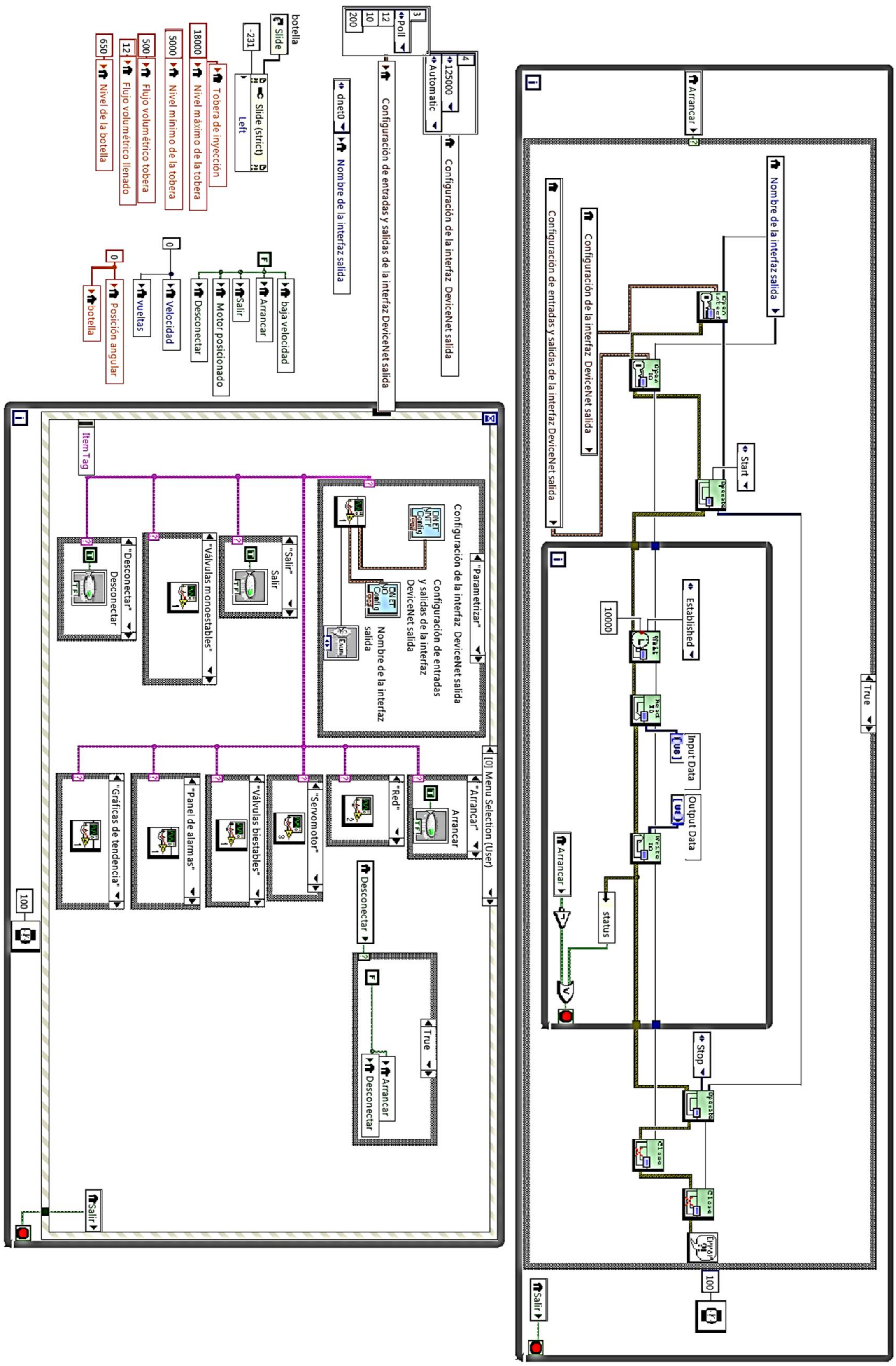
Válvulas biestables.vi
C:\Users\Pablo\Documents\Tesis\SCADA\Válvulas biestables.vi

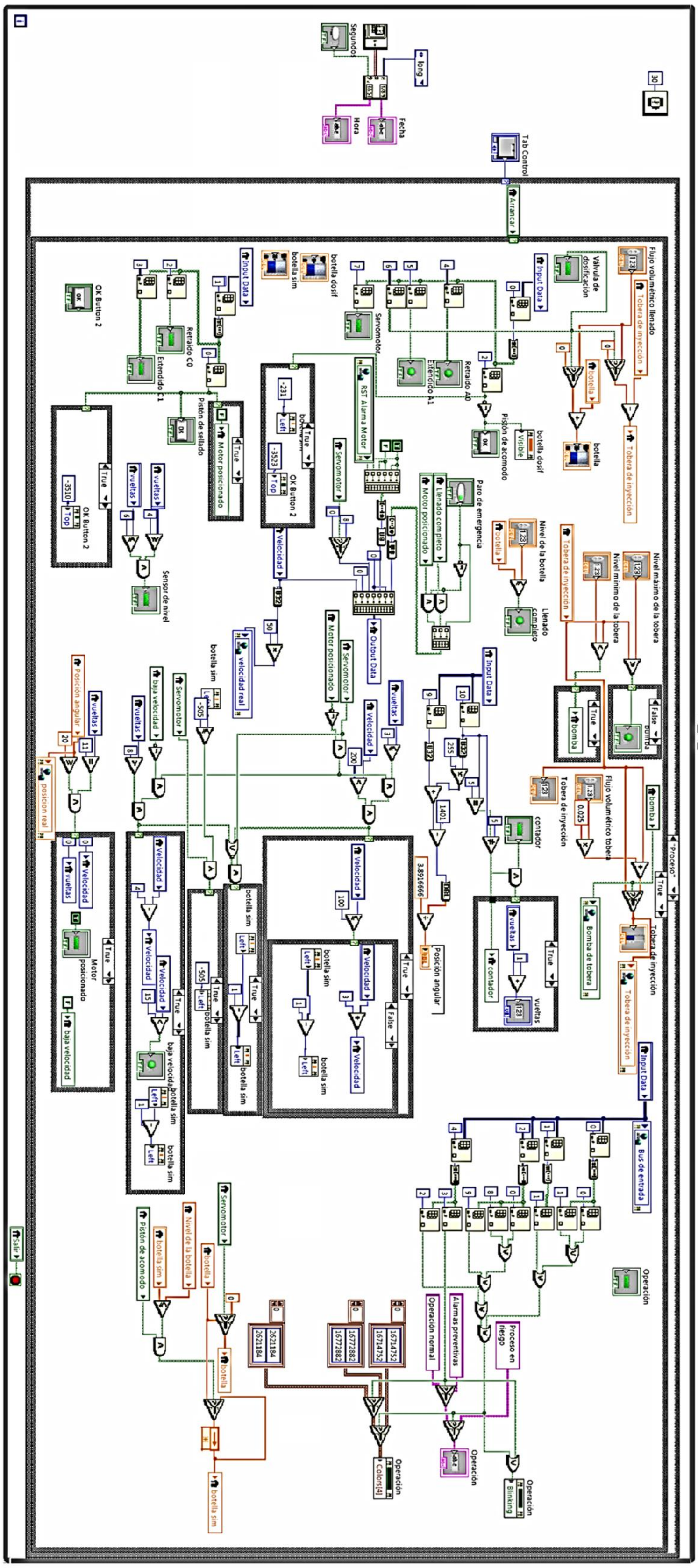
Válvulas monoestables.vi
C:\Users\Pablo\Documents\Tesis\SCADA\Válvulas monoestables.vi

Ventana de parametrización.vi
C:\Users\Pablo\Documents\Tesis\SCADA\Ventana de parametrización.vi

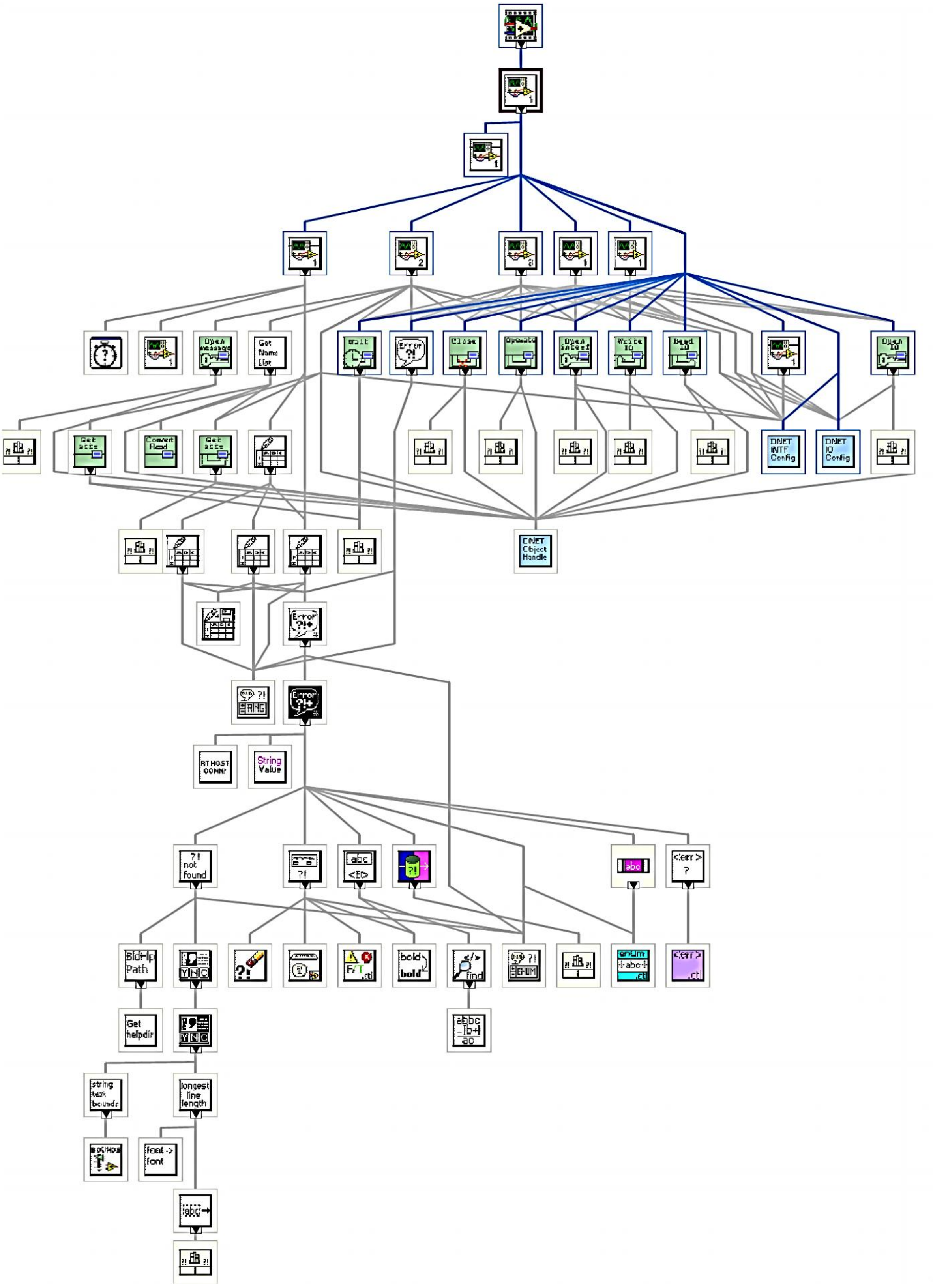
Gráficas.vi
C:\Users\Pablo\Documents\Tesis\SCADA\Gráficas.vi

Diagrama de bloques

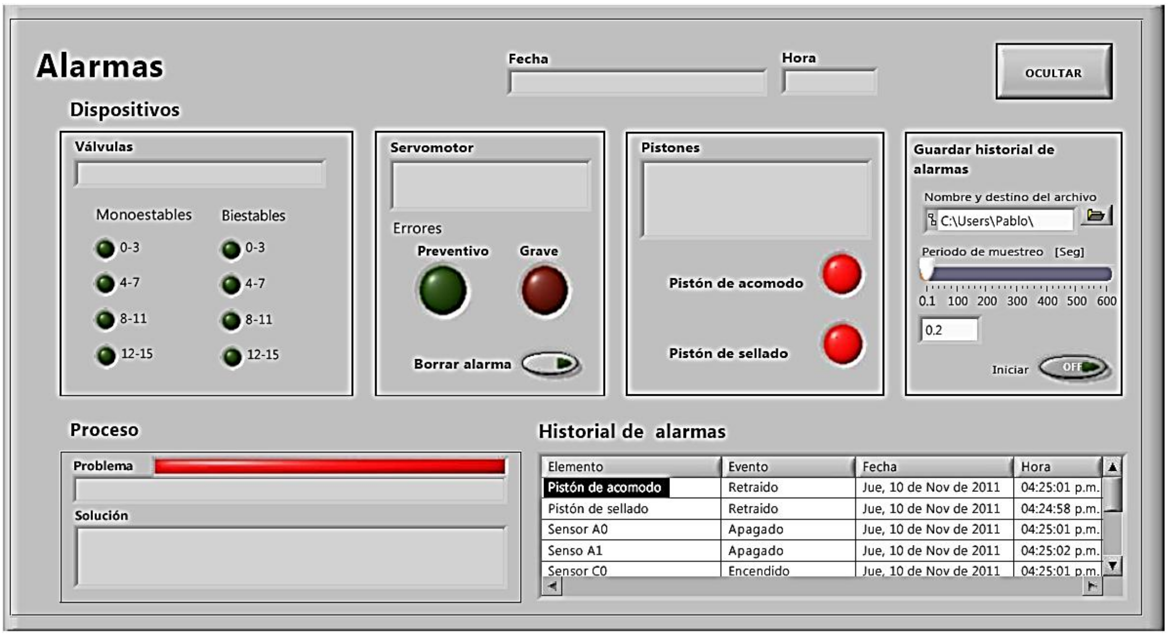



Segundos
long
Fecha
Hora
Tab Control
Avanzar
Flujo volumétrico llenado
Válvula de dosificación
Servomotor
OK Button 2
Paro de emergencia
Nivel de la botella
Llenado completo
Nivel máximo de la tobera
Nivel mínimo de la tobera
Motor posicionado
Sensor de nivel
Pistón de sellado
Pistón de acomodo
Bomba de tobera
contador
vueltas
Posición angular
botella sim
Bus de entrada
Operación
Operación normal
Alarmas preventivas
Proceso en riesgo

Jerarquía de VIs



Panel de alarmas



Lista de VIs

Write To Spreadsheet File (string).vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\file.llb\Write To Spreadsheet File (string).vi

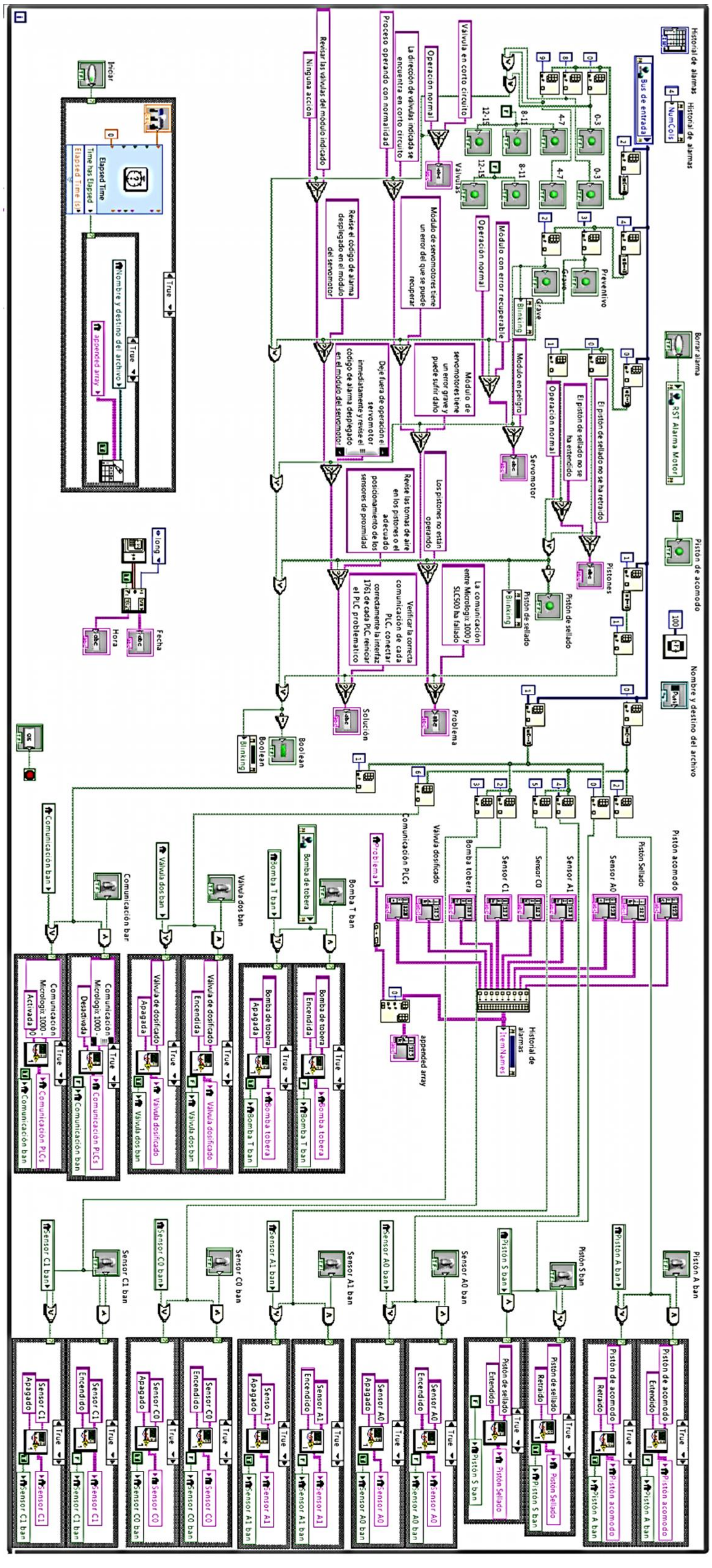
Write To Spreadsheet File.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\file.llb\Write To Spreadsheet File.vi

Elapsed Time
Elapsed Time
Indicates the amount of time that has elapsed since the specified start time.

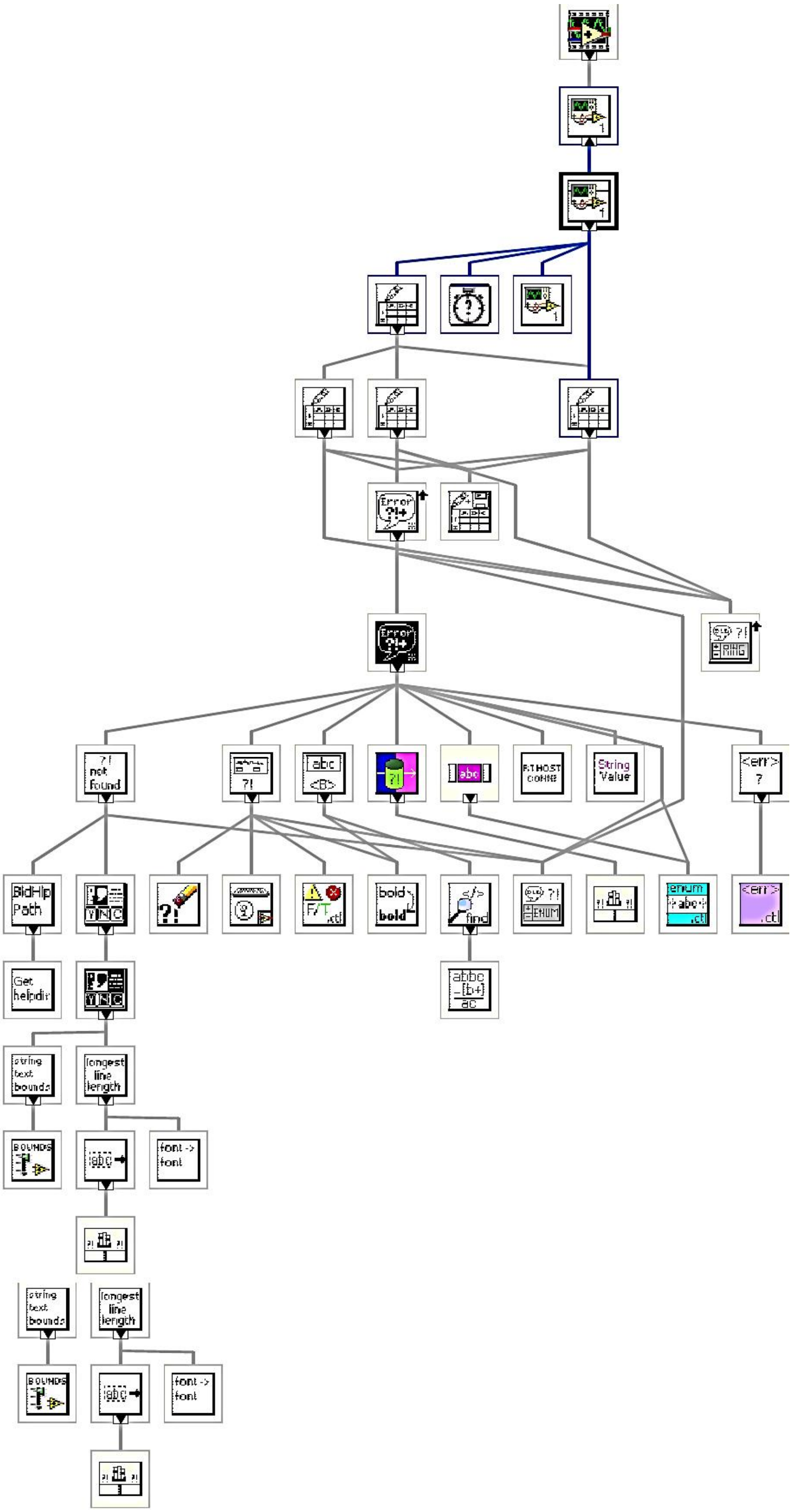
This Express VI is configured as follows:

Time Target: 1 s
Auto Reset: On

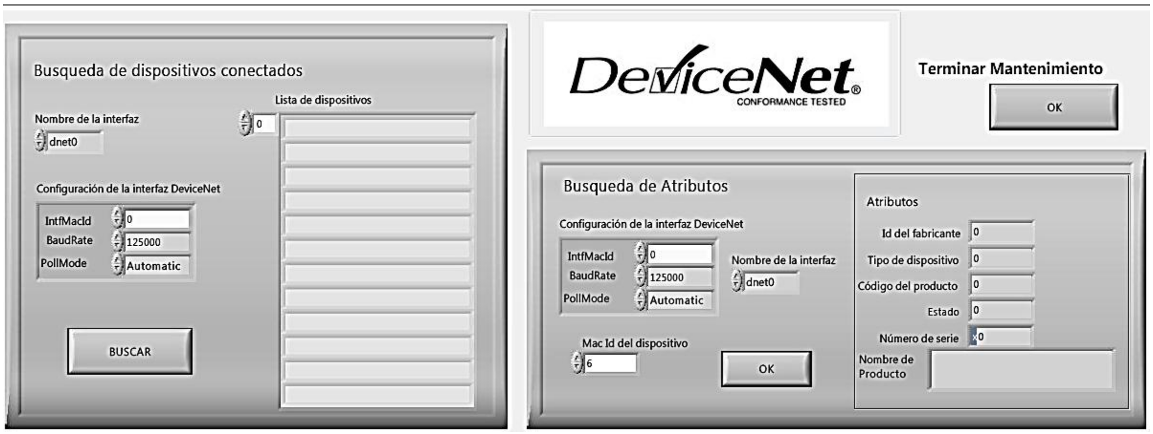
Diagrama de bloques



Jerarquía de VIs



Panel de red

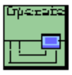


Lista de VIs

DnetIntfConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIntfConfig.ctl

xobjhandle.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\xobjhandle.ctl

Open DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet Interface.vi



Operate DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Operate DeviceNet Interface.vi

Close Object.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Close Object.vi

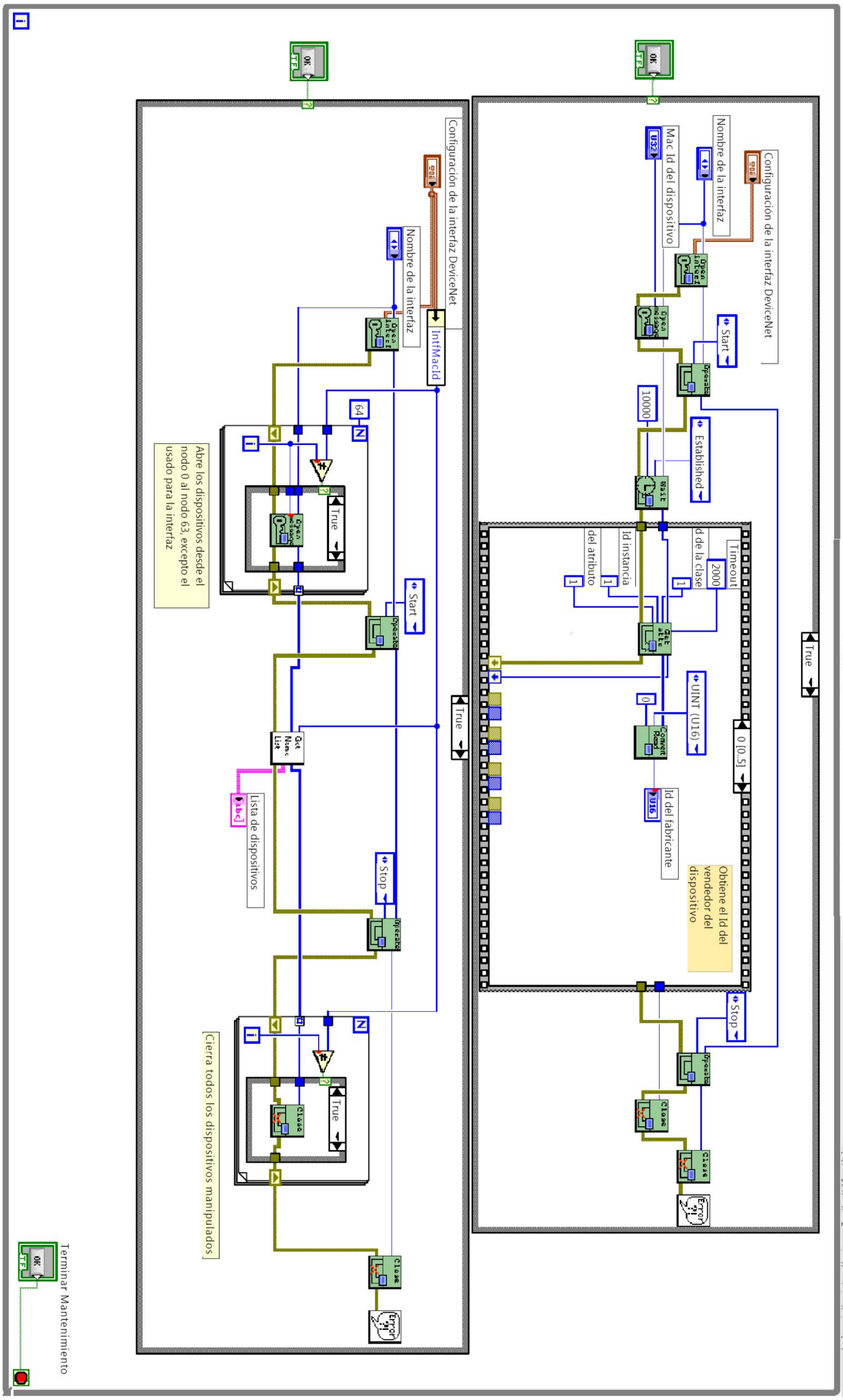
Simple Error Handler.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\error.llb\Simple Error Handler.vi

Wait For State.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Wait For State.vi

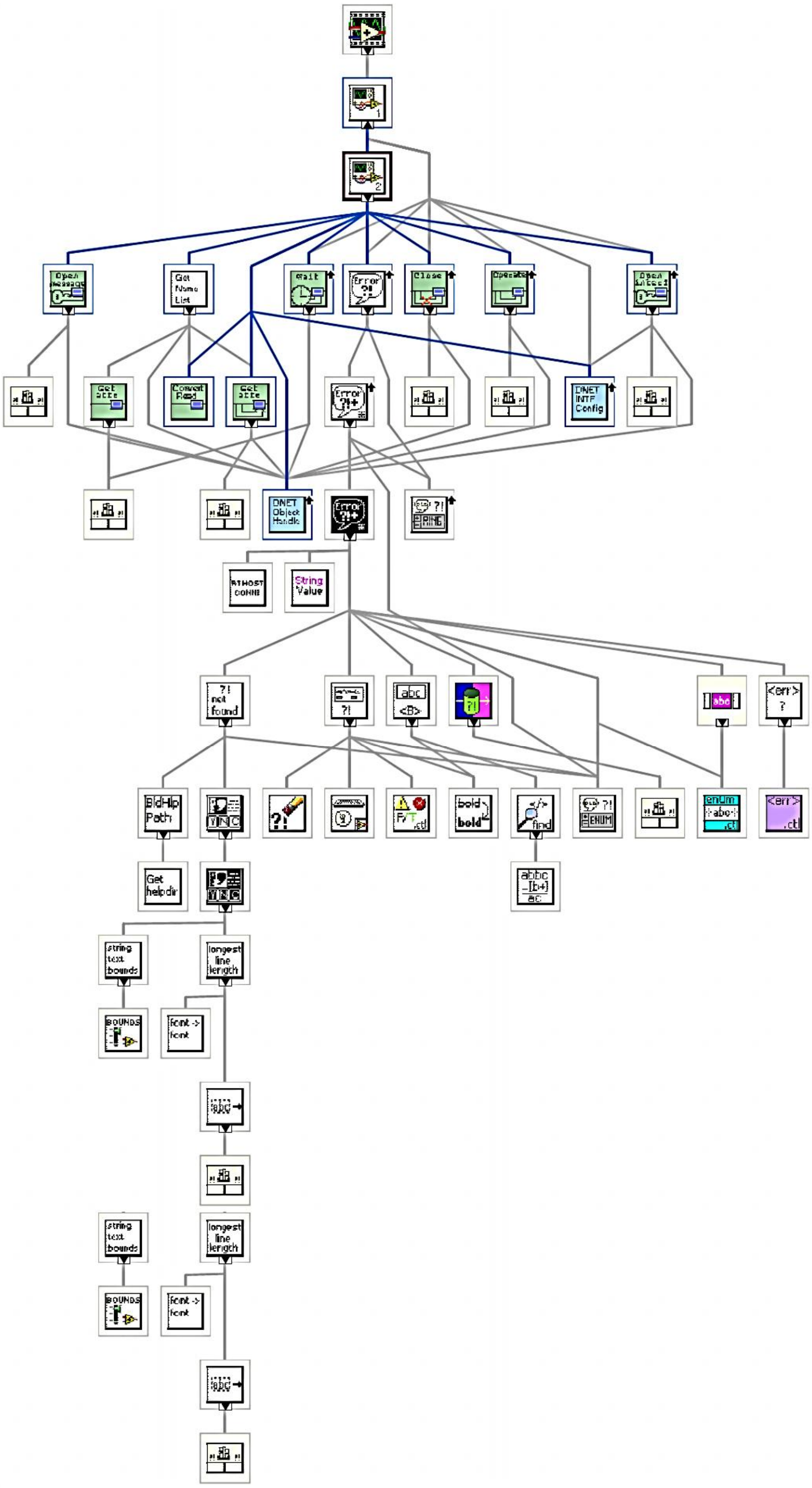
Convert From DeviceNet Read.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Convert From DeviceNet Read.vi

Get DeviceNet Attribute.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Get DeviceNet Attribute.vi

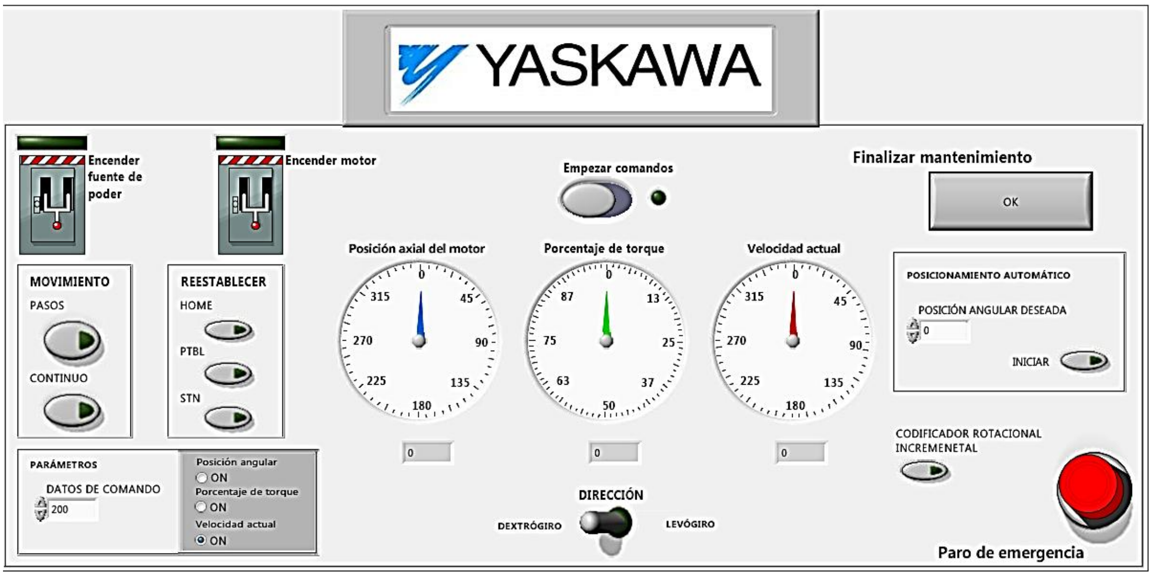
Diagrama de bloques



Jerarquía de VIs



Panel del servomotor



Lista de VIs

DnetIntfConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIntfConfig.ctl

DnetIOConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIOConfig.ctl

Open DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet Interface.vi

Open DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet IO.vi

Operate DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Operate DeviceNet Interface.vi

Close Object.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Close Object.vi

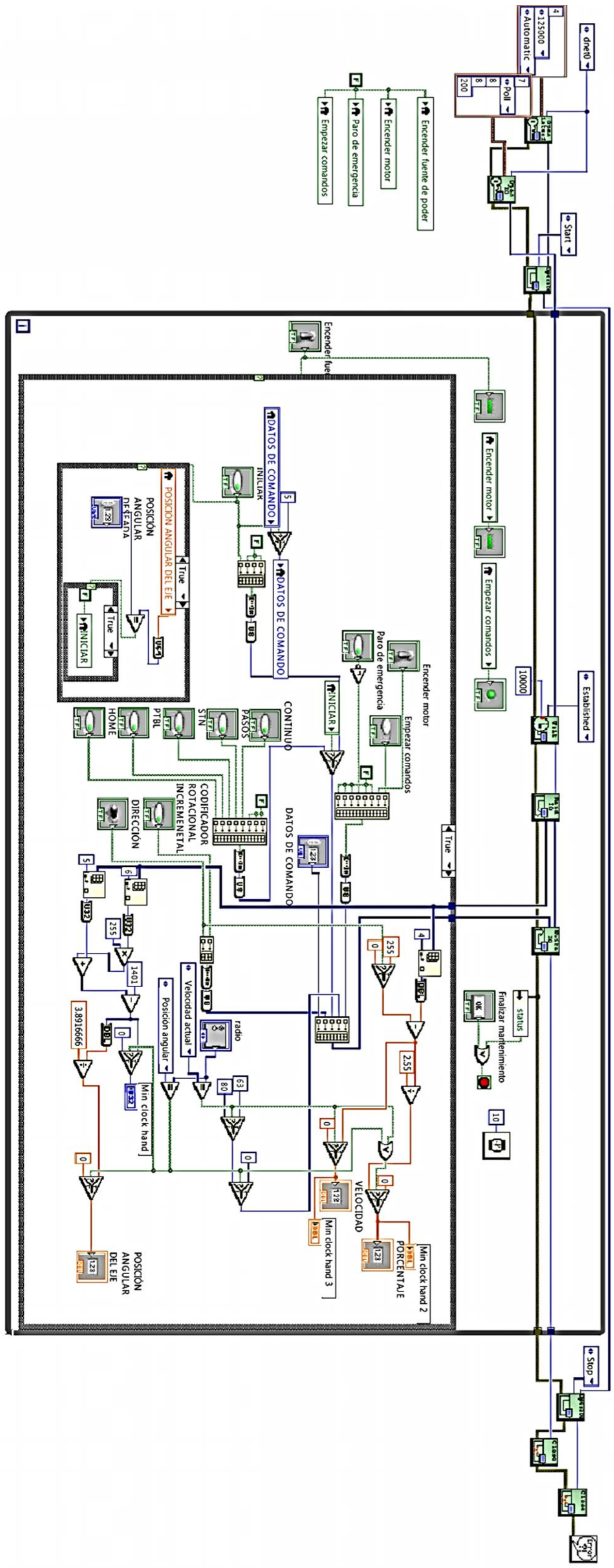
Simple Error Handler.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\error.llb\Simple Error Handler.vi

Wait For State.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Wait For State.vi

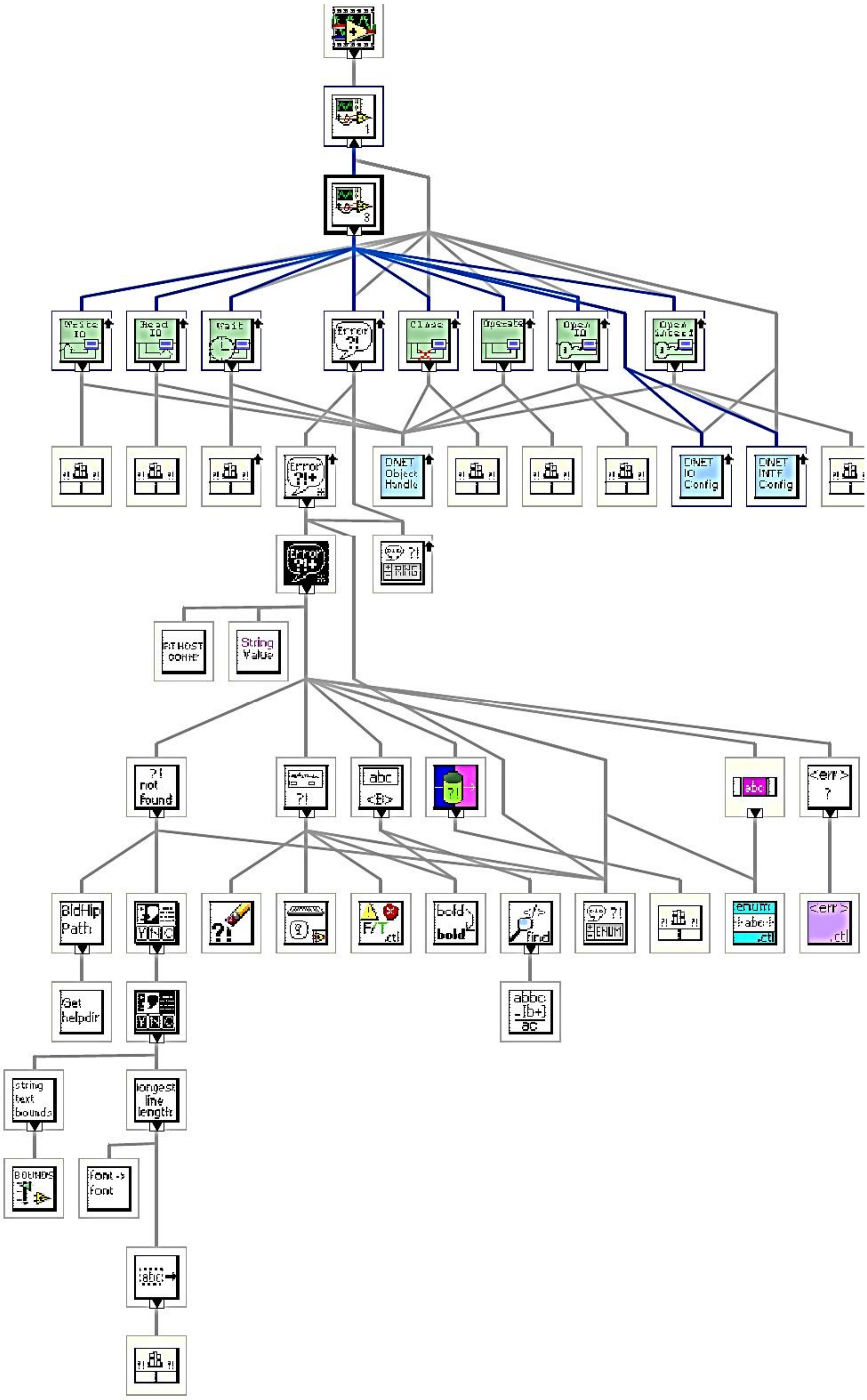
Read DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Read DeviceNet IO.vi

Write DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Write DeviceNet IO.vi

Diagrama de bloques



Jerarquía de VIs



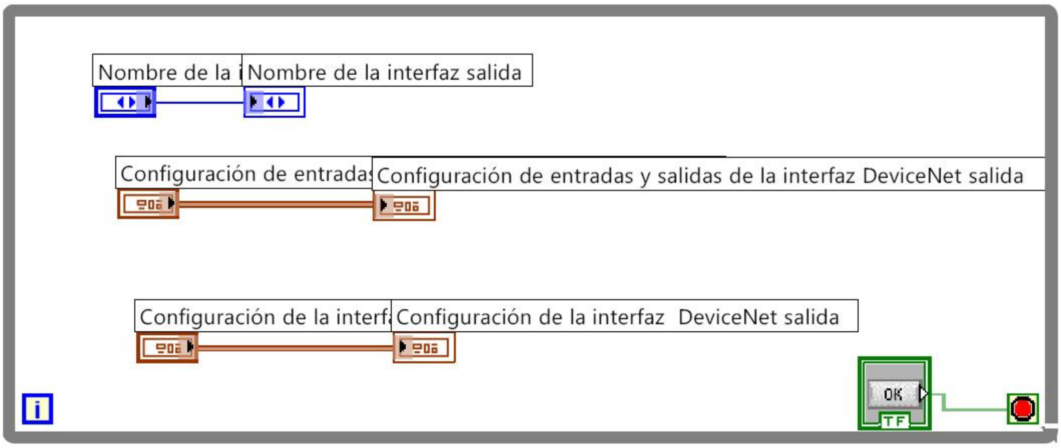
Panel de parámetros de arranque

Lista de VIs

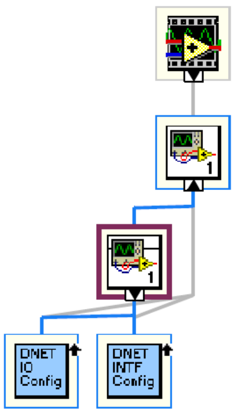
DnetIOConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIOConfig.ctl

DnetIntfConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIntfConfig.ctl

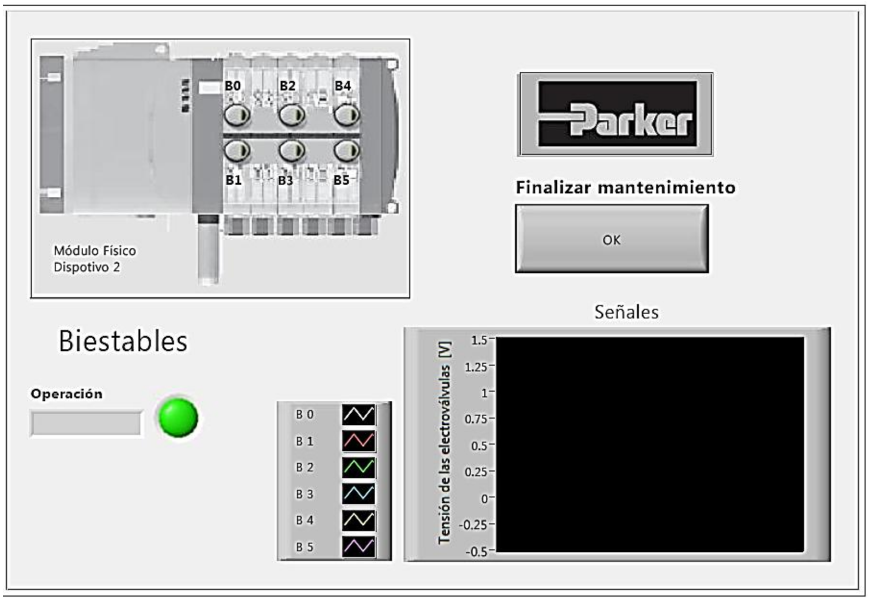
Diagrama de bloques



Jerarquía de VIs



Panel de válvulas biestables



Lista de VIs

DnetIOConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIOConfig.ctl

DnetIntfConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIntfConfig.ctl

Open DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet Interface.vi

Open DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet IO.vi

Operate DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Operate DeviceNet Interface.vi

Close Object.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Close Object.vi

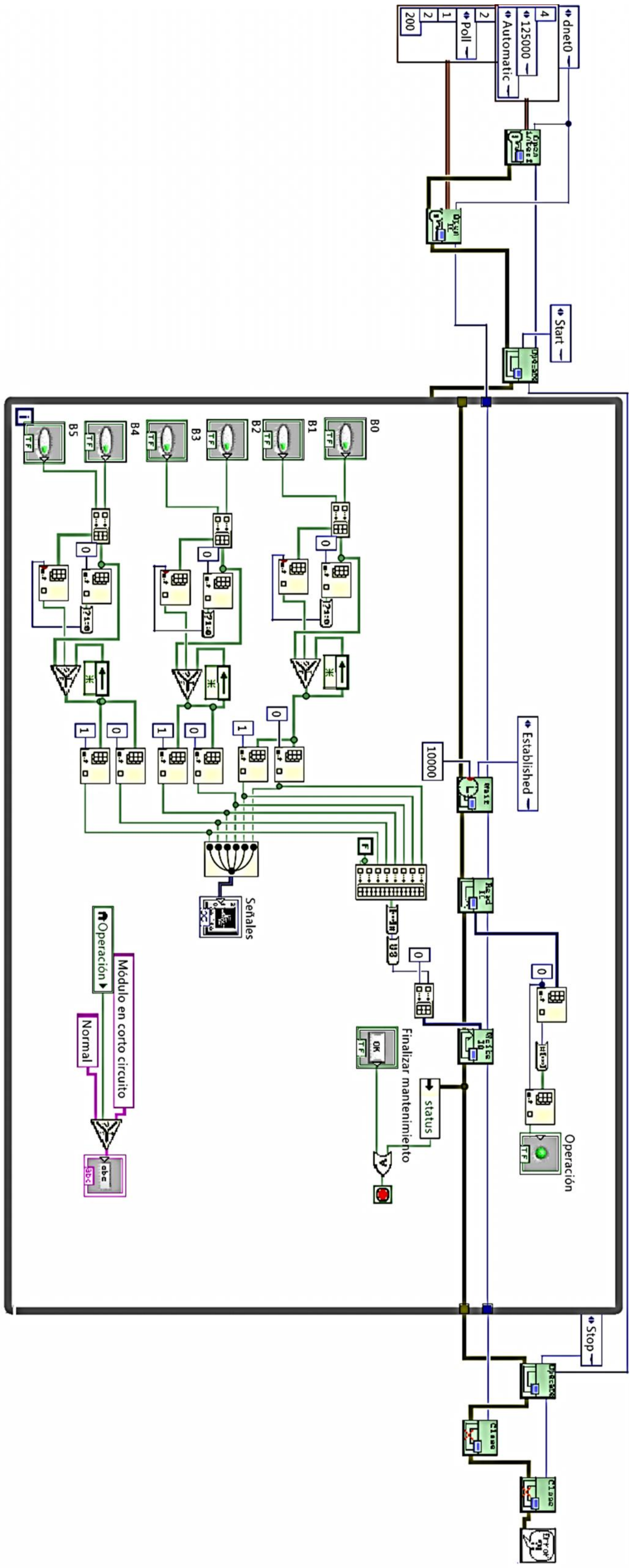
Simple Error Handler.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\error.llb\Simple Error Handler.vi

Wait For State.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Wait For State.vi

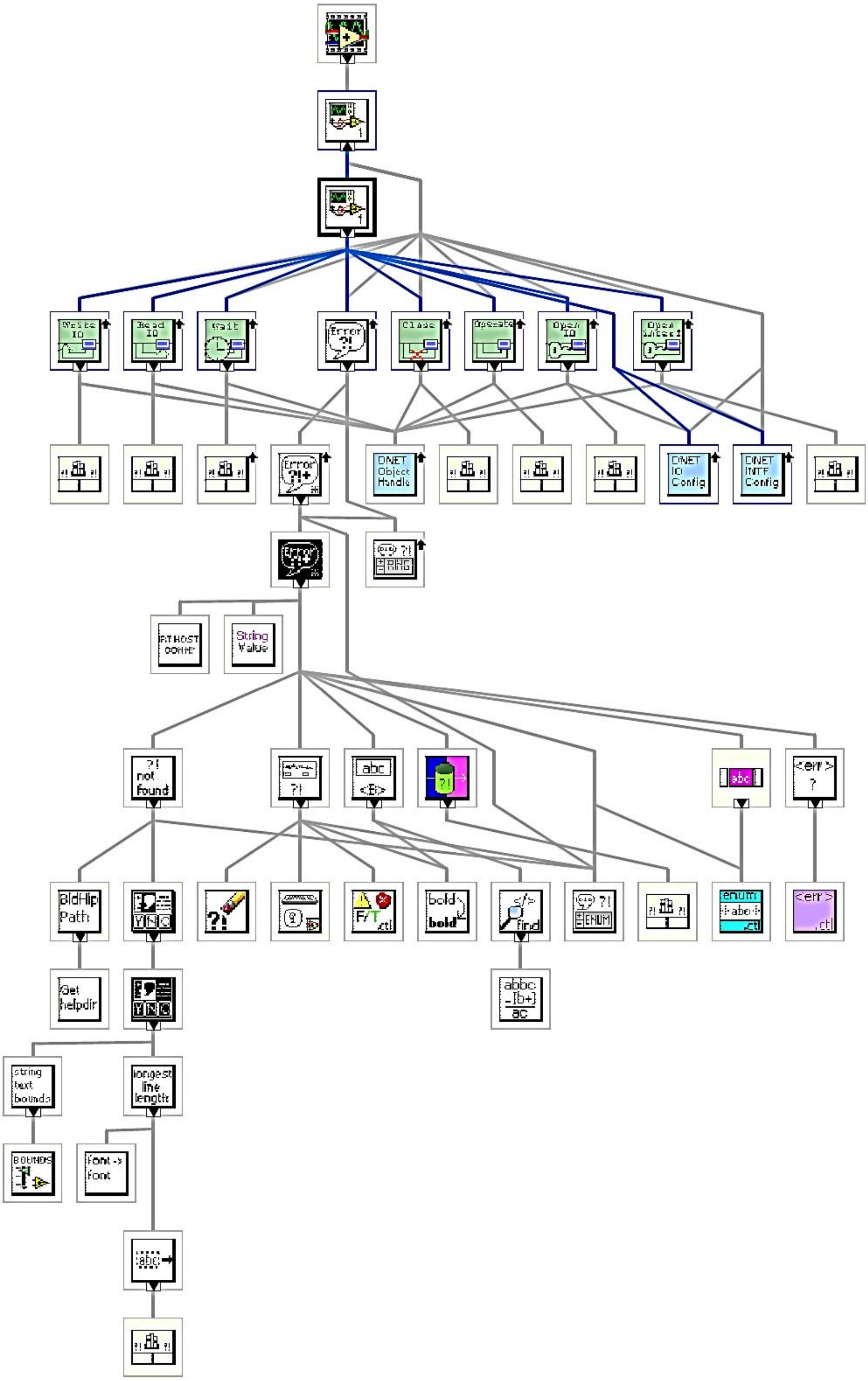
Read DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Read DeviceNet IO.vi

Write DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Write DeviceNet IO.vi

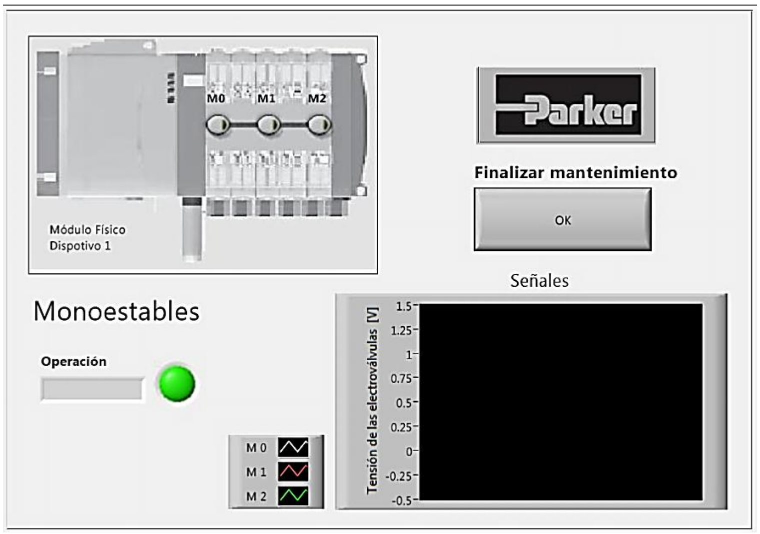
Diagrama de bloques



Jerarquía de VIs



Panel de válvulas monoestables



Lista de VIs

DnetIOConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIOConfig.ctl

DnetIntfConfig.ctl
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\DnetIntfConfig.ctl

Open DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet Interface.vi

Open DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Open DeviceNet IO.vi

Operate DeviceNet Interface.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Operate DeviceNet Interface.vi

Close Object.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Close Object.vi

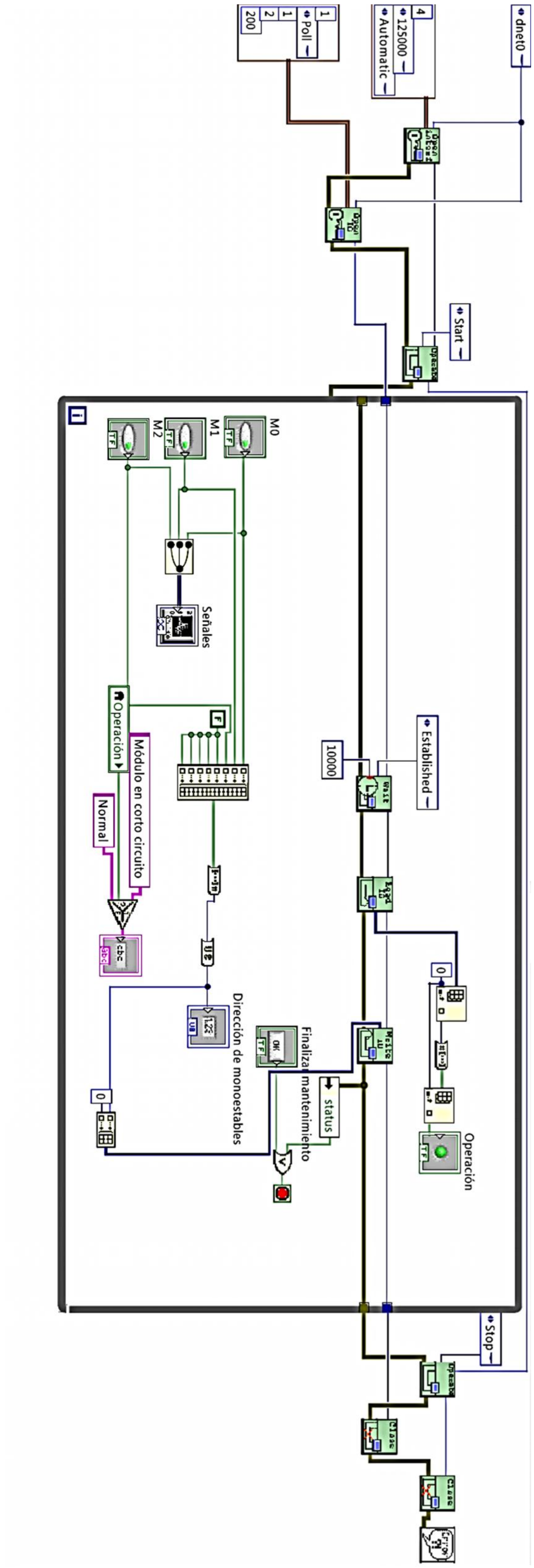
Simple Error Handler.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\Utility\error.llb\Simple Error Handler.vi

Wait For State.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Wait For State.vi

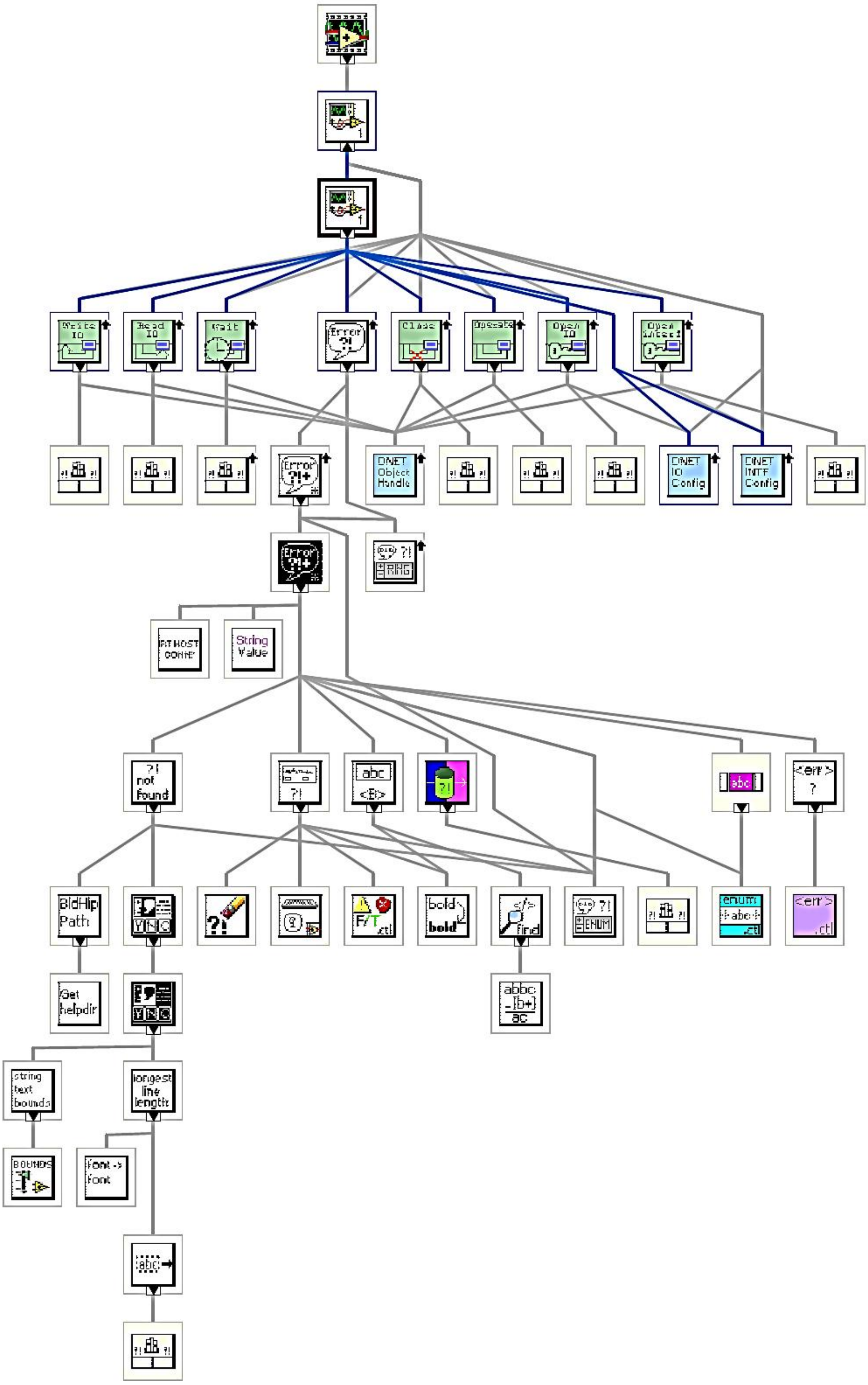
Read DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Read DeviceNet IO.vi

Write DeviceNet IO.vi
C:\Program Files (x86)\National Instruments\LabVIEW 2010\vi.lib\nidnet\Nidnet.llb\Write DeviceNet IO.vi

Diagrama de bloques



Jerarquía de VIs



Panel de gráficas de tendencia

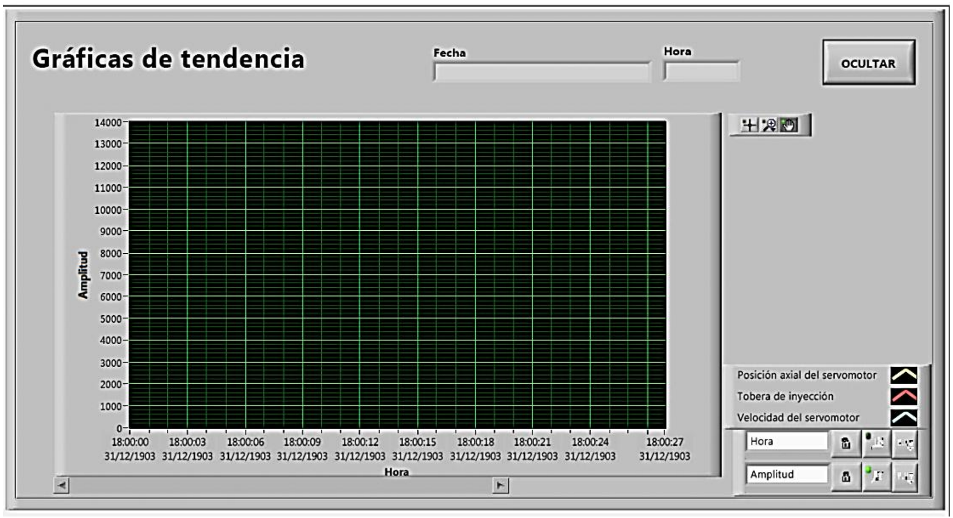
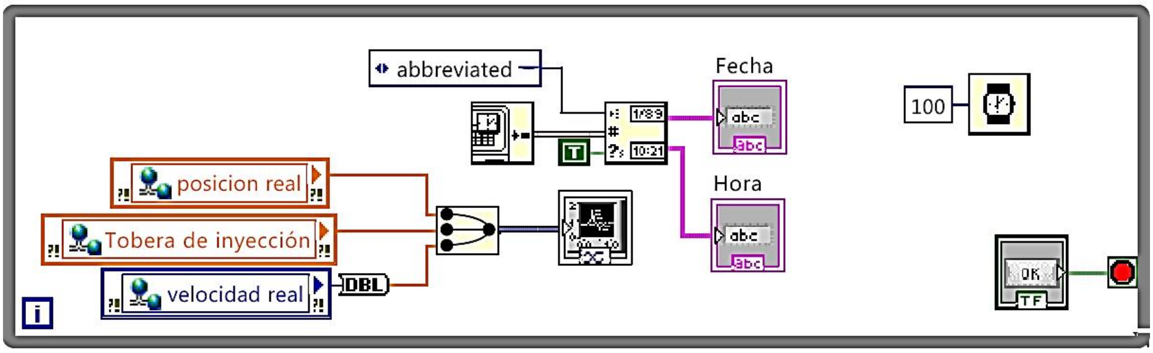


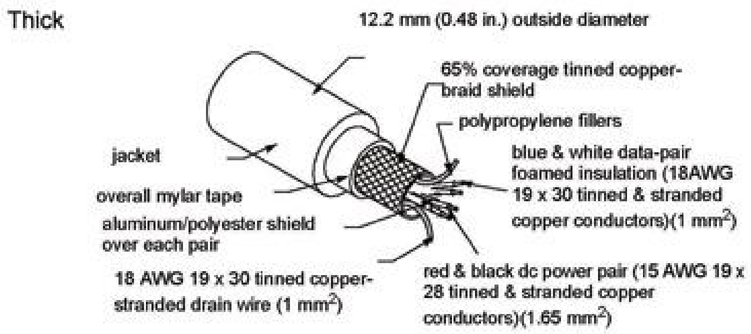
Diagrama de bloques.

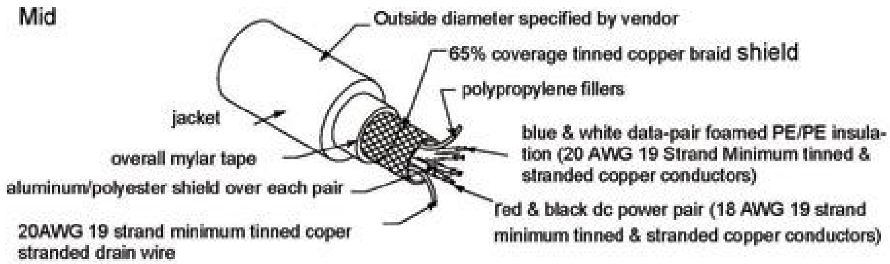


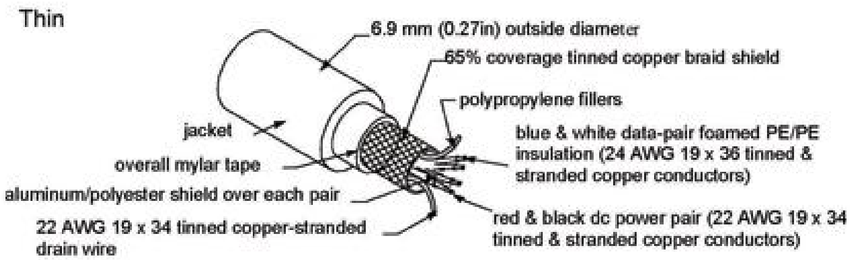
Jerarquía de VIs

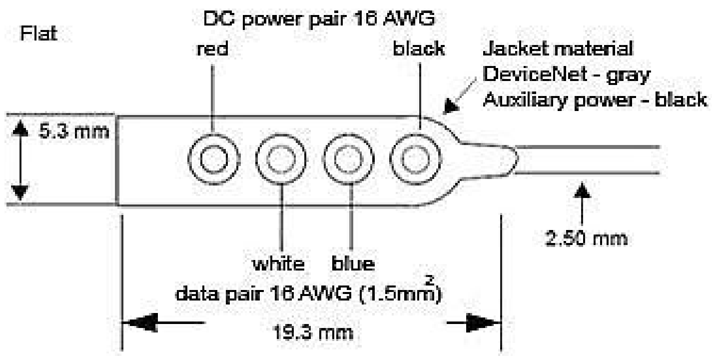
Anexo 3. Cables y conexiones DeviceNet.

Cables



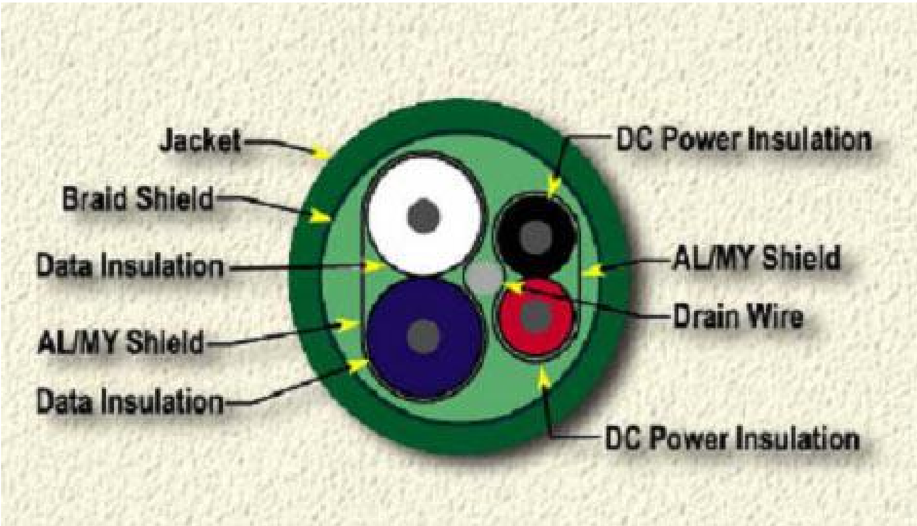






El más usado son los cables grueso y delgado los cuales tiene 5 conductores identificados de acuerdo a la siguiente tabla.

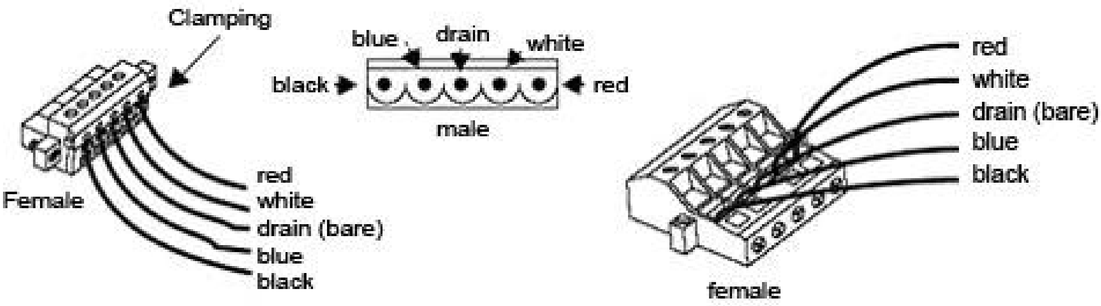
Wire Color	Signal	Round Cable	Flat Cable
White	CAN_H	DN Signal	DN Signal DN
Blue	CAN_L	DN Signal	DN Signal
Naked wire	Drain	Shield	Not used
Black	V-	Power	Power
Red	V+	Power	Power



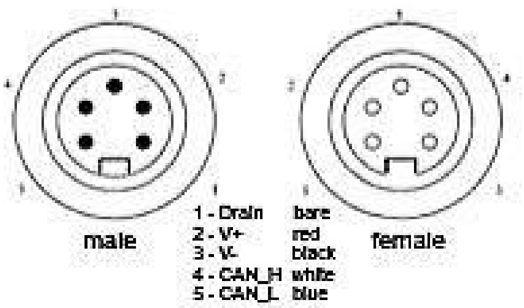
Vista de un cable estandar de la DeviceNet.

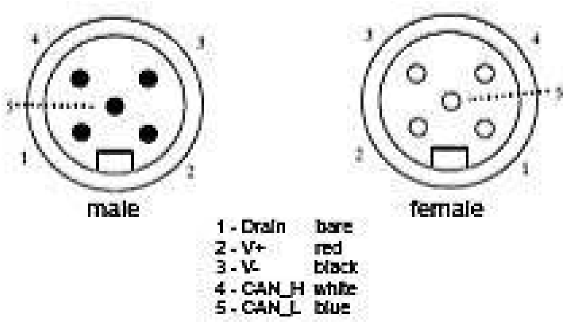
Conectores

Existen básicamente 3 tipos de conectores: abierto, mini-sealed y micro sealed. Su elección depende en la aplicación y características del equipo a conectar o de la necesidad de la conexión. En las siguientes figuras se muestra el código de conexión para cada tipo.



Conector abierto





Conector Mini-sealed Conector Micro-sealed

Anexo 4. Parámetros del servomotor y comando/respuesta de la comunicación

Tablas con parámetros del módulo del servomotor

4.2 Parameter Tables

The following tables list the parameters.

If using the NSxxx Setup Tool or reading/writing using a command message, edit parameters using Pn□□□. If editing via DeviceNet explicit messages, edit using the object number and attribute number. Refer to 5.6 *Changing Parameters* or the host controller manual for details.

4.2.1 Unit Parameters

The unit parameter table is shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#30	Pn810	Electronic Gear Ratio (Numerator)	1 to 10,000,000	---	Power-up	1	B
	#31	Pn811	Electronic Gear Ratio (Denominator)	1 to 10,000,000	---	Power-up	1	B

4.2.2 Zero Point Return Parameters

The table of zero point return parameters are shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#10	Pn800	Zero Point Return Mode	0 to 3	---	Immediate	0	B
	#11	Pn801	Zero Point Return Function Selection	0 to 7	---	Power-up	1	B
	#12	Pn802	Feed Speed for Zero Point Return	1 to 240,000	1000 steps/min	Immediate	10,000	B
	#13	Pn803	Approach Speed for Zero Point Return	1 to 240,000	1000 steps/min	Immediate	1,000	B
	#14	Pn804	Creep Speed for Zero Point Return	1 to 240,000	1000 steps/min	Immediate	500	B
	#15	Pn805	Final Travel Distance for Zero Point Return	0 to 99,999,999	Steps	Immediate	0	B
	#16	Pn806	Output Width for Zero Point Return	0 to 32, 767	Step	Immediate	100	B

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#17	Pn809	Zero Point Offset	-99,999,999 to 99,999,999	Steps	Immediate	0	C
	#18	Pn80A	Accel/Decel Time for Zero Point Return	1 to 10,000	ms	Immediate	100	B

Note: 1. "Steps" means "reference unit." For reference unit details, refer to *4.3.1 Unit Parameters*.

2. If you set the reference unit to 0.001 mm, 1,000 steps/min becomes mm/min.

4.2.3 Machine System and Peripheral Device Parameters

The machine system and peripheral device parameter table is shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#32	Pn812	Coordinate Type	0, 1	---	Immediate	0	C
	#33	Pn813	Reference units per Machine Rotation	1 to 1,500,000	---	Immediate	360,000	C
	#34	Pn814	Backlash Compensation	0 to 32,767	Steps	Immediate	0	C
	#35	Pn815	Backlash Direction	0, 1	Steps	Immediate	0	C
	#36	Pn816	Positive Software Limit	±99,999,999	---	Power-up	99999999	B
	#37	Pn817	Negative Software Limit	±99,999,999	Steps	Power-up	−99999999	B
	#38	Pn818	Machine Function Selection	0 to 3	---	Immediate	0	B
	#39	Pn819	Hardware Limit Signal Function Selection	0 to 3	---	Immediate	1	B
	#40	Pn81A	Hardware Limit Action Selection	0, 1, 2	---	Immediate	0	B
	#41	Pn81B	Emergency Stop Signal Function Selection	0 to 3	---	Immediate	1	B

Note: 1. "Steps" means "reference unit." For reference unit details, refer to *4.3.1 Unit Parameters*.

2. If you set the reference unit to 0.001 mm, 1,000 steps/min becomes mm/min.

4.2.4 Speed, Acceleration, and Deceleration Parameters

A table of speed, acceleration, and deceleration parameters is shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#51	Pn821	Feed Speed for Positioning	1 to 240,000	1000 steps/min	Immediate	24,000	B
	#52	Pn822	Acceleration Time for Positioning	1 to 10,000	ms	Immediate	100	B
	#53	Pn823	Deceleration Time for Positioning	1 to 10,000	ms	Immediate	100	C
	#54	Pn824	Switch Speed for Second Accel/Decel for Positioning	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#55	Pn825	Accel/Decel Time for Second Accel/Decel for Positioning	1 to 10,000	ms	Immediate	200	B
	#56	Pn826	Accel/Decel Type for Positioning	0 to 3	---	Immediate	0	B
	#57	Pn827	Feed Speed for External Positioning	1 to 240,000	1000 steps/min	Immediate	24,000	B
	#58	Pn829	Filter Selection	0 to 3	---	Immediate	0	B
	#59	Pn830	Constant Feed Reference Unit Selection	0, 1	---	Immediate	0	B
	#60	Pn831	Constant Feed Speed	1 to 240,000	1000 steps/min	Immediate	24,000	B
	#61	Pn832	Acceleration Time for Constant Feed	1 to 10,000	ms	Immediate	100	B
	#62	Pn833	Deceleration Time for Constant Feed	1 to 10,000	ms	Immediate	100	C
	#63	Pn834	Switch Speed for Constant Feed Second Accel/Decel	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#64	Pn835	Accel/Decel Time for Constant Feed Second Accel/Decel	1 to 10,000	ms	Immediate	200	C

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#65	Pn836	Accel/Decel Type for Constant Feed	0, 1, 2, 3	---	Immediate	0	B
	#70	Pn840	Time Constant for Exponential Accel/Decel	4 to 10,000	ms	Immediate	25	C
	#71	Pn841	Bias Speed for Exponential Accel/Decel	1 to 240,000	1000 steps/min	Immediate	0	C
	#72	Pn842	Time Constant of Travelling Average	4 to 10,000	ms	Immediate	25	C
	#73	Pn843	Maximum Feed Speed	1 to 240,000	1000 steps/min	Immediate	24,000	B
	#74	Pn844	Step Distance 1	0 to 99,999,999	Steps	Immediate	1	B
	#75	Pn845	Step Distance 2	0 to 99,999,999	Steps	Immediate	10	B
	#76	Pn846	Step Distance 3	0 to 99,999,999	Steps	Immediate	100	B
	#77	Pn847	Step Distance 4	0 to 99,999,999	Steps	Immediate	1,000	B

Note: 1. "Steps" means "reference unit." For reference unit details, refer to *4.3.1 Unit Parameters*.

2. If you set the reference unit to 0.001 mm, 1,000 steps/min becomes mm/min.

4.2.5 Positioning Parameters

The positioning parameter table is shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#90	Pn850	Positioning Deadband	0 to 10,000	Steps	Immediate	5	A
	#91	Pn851	Positioning Timeout	0 to 100,000	ms	Immediate	0	A
	#92	Pn852	Positioning Proximity Detection Width	0 to 32,767	Steps	Immediate	10	B
	#93	Pn853	Direction for Rotation System	0, 1	---	Immediate	0	B
	#94	Pn854	Approach Speed for External Positioning	1 to 240,000	1,000 steps/min	Immediate	24,000	B
	#95	Pn855	Travel Distance for External Positioning	0 to 99,999,999	Steps	Immediate	0	B
	#96	Pn856	Function Selection for External Positioning	0 to 1	---	Power-up	1	B
	#100	Pn85A	Number of Stations	1 to 32,767	---	Immediate	1	B

Note: 1. "Steps" means "reference unit." For reference unit details, refer to *4.3.1 Unit Parameters*.

2. If you set the reference unit to 0.001 mm, 1,000 steps/min becomes mm/min.

4.2.6 Multi-speed Positioning Parameters

A table of multi-speed positioning parameters is shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#111	Pn861	Number of Points for Speed Switching	0 to 16	–	Immediate	0	C
	#112	Pn862	Initial Feed Speed for Multi-speed Positioning	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#113	Pn863	Speed Switching Position 1	0 to 99,999,999	Steps	Immediate	0	C
	#114	Pn864	Speed Switching Position 2	0 to 99,999,999	Steps	Immediate	0	C
	#115	Pn865	Speed Switching Position 3	0 to 99,999,999	Steps	Immediate	0	C
	#116	Pn866	Speed Switching Position 4	0 to 99,999,999	Steps	Immediate	0	C
	#117	Pn867	Speed Switching Position 5	0 to 99,999,999	Steps	Immediate	0	C
	#118	Pn868	Speed Switching Position 6	0 to 99,999,999	Steps	Immediate	0	C
	#119	Pn869	Speed Switching Position 7	0 to 99,999,999	Steps	Immediate	0	C
	#120	Pn86A	Speed Switching Position 8	0 to 99,999,999	Steps	Immediate	0	C
	#121	Pn86B	Speed Switching Position 9	0 to 99,999,999	Steps	Immediate	0	C
	#122	Pn86C	Speed Switching Position 10	0 to 99,999,999	Steps	Immediate	0	C
	#123	Pn86D	Speed Switching Position 11	0 to 99,999,999	Steps	Immediate	0	C
	#124	Pn86E	Speed Switching Position 12	0 to 99,999,999	Steps	Immediate	0	C
	#125	Pn86F	Speed Switching Position 13	0 to 99,999,999	Steps	Immediate	0	C
	#126	Pn870	Speed Switching Position 14	0 to 99,999,999	Steps	Immediate	0	C
	#127	Pn871	Speed Switching Position 15	0 to 99,999,999	Steps	Immediate	0	C
	#128	Pn872	Speed Switching Position 16	0 to 99,999,999	Steps	Immediate	0	C
	#129	Pn873	Switching Speed 1	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#130	Pn874	Switching Speed 2	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#131	Pn875	Switching Speed 3	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#132	Pn876	Switching Speed 4	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#133	Pn877	Switching Speed 5	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#134	Pn878	Switching Speed 6	1 to 240,000	1000 steps/min	Immediate	24,000	C

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#135	Pn879	Switching Speed 7	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#136	Pn87A	Switching Speed 8	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#137	Pn87B	Switching Speed 9	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#138	Pn87C	Switching Speed 10	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#139	Pn87D	Switching Speed 11	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#140	Pn87E	Switching Speed 12	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#141	Pn87F	Switching Speed 13	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#142	Pn880	Switching Speed 14	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#143	Pn881	Switching Speed 15	1 to 240,000	1000 steps/min	Immediate	24,000	C
	#144	Pn882	Switching Speed 16	1 to 240,000	1000 steps/min	Immediate	24,000	C

Note: 1. "Steps" means "reference unit." For reference unit details, refer to *4.3.1 Unit Parameters*.

2. If you set the reference unit to 0.001 mm, 1,000 steps/min becomes mm/min.

4.2.7 Notch Output Parameters

The notch output parameter table is shown below.

Object	Attribute	No.	Name	Range	Units	Effective Timing	Default Value	Type
0x64	#160	Pn890	Notch Signal Output Position Setting	0, 1	–	Immediate	0	C
	#161	Pn891	Notch Signal Output Setting	0 to 3	–	Immediate	0	C
	#162	Pn892	Notch 1 Output Position Lower Limit	±99,999,999	Steps	Immediate	0	C
	#163	Pn893	Notch 1 Output Position Upper Limit	±99,999,999	Steps	Immediate	0	C
	#164	Pn894	Notch 2 Output Position Lower Limit	±99,999,999	Steps	Immediate	0	C
	#165	Pn895	Notch 2 Output Position Upper Limit	±99,999,999	Steps	Immediate	0	C

Note: 1. "Steps" means "reference unit." For reference unit details, refer to *4.3.1 Unit Parameters*.

2. If you set the reference unit to 0.001 mm, 1,000 steps/min becomes mm/min.

5.3 Command/Response Format

This section explains command messages sent to an NS300 Unit from the master device and the response messages sent from the NS300 Unit.

5.3.1 Command Format

This section explains the basic format of command messages sent to an NS300 Unit from the master device and the response messages sent from the NS300 Unit to the master device. Command and response messages are in an 8-byte data format.

Command

Byte		
[0]	0/1	General command bits
[1]	Command-dependant data	
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		

Response

Byte		
[0]	0/1	General status
[1]	Command-dependant data	
[2]		
[3]		
[4]		
[5]		
[6]		
[7]		

Both the command and response data are separated into two sections. Byte 0 is a general area and bytes 1 to 7 make up the command-dependant data area. The command-dependant data area depends on the two types of commands, move commands and set/read commands. The type of command is defined by the most-significant bit of byte 0.

■ Format for Move Commands

Command

Byte		
[0]	0	General command bits
[1]	Response type	Command code
[2]	Command bit	
[3]		
[4]	Command data	
[5]		
[6]		
[7]		

Response

Byte		
[0]	0	General status
[1]	Response type	Command code
[2]	Status	
[3]		
[4]	Response data	
[5]		
[6]		
[7]		

■ Format for Set/Read Commands

Command

Byte		
[0]	1	General command bits
[1]	0	Command code
[2]	Command number	
[3]		
[4]	Command data	
[5]		
[6]		
[7]		

Response

Byte		
[0]	1	General status
[1]	0	Command code
[2]	Command number	
[3]		
[4]	Response data	
[5]		
[6]		
[7]		

5.3.2 General Command Bits and Status

■ General Command Bits

The general command bit area is detailed below.

Table 5.2 General Command Bits

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	MOD	0	ALRST	ESTP	0	0	SVON	C_STRT

Mode: MOD

Use the MOD bit to specify the data format for bytes 1 to 7.

- 0: Move command format
- 1: Set/read command format

INFO The MOD bit alters the data format for bytes 1 to 7. Set it carefully.

Alarm Reset Command: ALRST

Set the ALRST bit to 1 to reset the current alarm/warning. When an alarm or warning occurs in the NS300 Unit or SGDH, remove the cause of the alarm before setting this bit to 1. The alarm/warming will be cleared.

Always make sure this bit is set to 0 during normal operation and after an alarm has been cleared.

Emergency Stop Command: ESTP

When the ESTP bit is changed from 1 to 0, a move command is canceled and the SGDH servo is turned OFF. If the axis is travelling, axis travel is stopped immediately and the SGDH servo is turned OFF as soon as the servomotor stops.

The ESTP Command has negative logic to confirm that DeviceNet communications have been established. Therefore, set the bit to 1 for normal operation and set it to 0 for emergency stops.

The emergency stop status will continue while this bit is set to 0. To release the emergency stop status, set the bit to 1. To turn ON the servo after releasing an emergency stop, set the Servo ON Command bit to 0 and then set it to 1.

Servo ON Command: SVON

Set the SVON bit to 1 to turn ON the SGDH servo. When the leading edge of the bit is detected, the SGDH servo is turned ON and remains ON while the command bit is set to 1. When the command bit setting changes to 0, the servo is turned OFF.

If an alarm turns OFF the SGDH servo, the command bit must be set to 0 and then set to 1 again.

Command Start Command: C_STRT

Set the C_STRT bit to 1 to start execution of the command specified by the command code. Always set the command code and command area data before (or at the same time as) setting the C_STRT bit to 1.

Refer to *5.3.3 Move Command Messages* and *5.3.4 Set/Read Command Messages* for details on using command codes.

■ General Status

Details on the general status area are shown below.

Table 5.3 General Status Bits

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	MOD_R	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R

Mode: MOD_R

The MOD_R bit specifies the data format of bytes 1 to 7.

This bit will be the same as the mode given in the command message.

- 0: Response format for move commands
- 1: Response format for set/read commands

Unit Ready: READY

The READY bit will be set to 1 when the NS300 Unit is ready to receive commands from the host device. The READY status will be 0 when the power is turned ON, and when the Unit Reset Command in the command message has been received and the NS300 Unit is initializing.

Main Power Supply Status: PWRON

The PWRON bit will be set to 1 when the SGDH main power supply is turned ON. If the main power supply is turned OFF, the bit will be 0 and the Servo ON and other commands cannot be executed.

Emergency Stop: ESTP_R

The ESTP_R bit will be set to 0 when the Emergency Stop Command in the command message has been set to 0 and the NS300 Unit is in emergency stop status. Set the Emergency Stop Command in the command message to 1 to clear the emergency stop status, and this bit will change to 1.

This status has negative logic.

Alarm: ALRM

The ALRM bit will be set to 1 when the NS300 Unit has detected an alarm. When all alarms have been cleared by the Alarm Reset Command in the command message, this bit will change to 0.

Warning: WARN

The WARN bit will be set to 1 when the NS300 Unit has detected a warning. When all warnings have been cleared by the Alarm Reset Command in the command message, this bit will change to 0.

When a warning has occurred, the command that generated the warning and commands other than data setting commands can still be executed normally.

Servo ON: SVON_R

The SVON_R bit will be set to 1 when the Servo ON Command in the command message is set to 1 and the SGDH servo is ON.

The SVON_R bit will be 0 in the following circumstances.

- When the Servo ON Command in the command message has been set to 0
- When the Emergency Stop Command has been set to 0
- When the Unit Reset Command has been set to 1
- When an alarm has occurred

Command Start Response: C_STRT_R

The C_STRT_R bit will be set to 1 when the Command Start Command in the command message has been set to 1. The host device can recognize that the NS300 Unit has received a command from the host device by checking that this bit is 1.

5.3.3 Move Command Messages

■ Command Messages

Details on command messages for move commands are shown below.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	ALRST	ESTP	0	0	SVON	C_STRT
1	Response type				Command code			
2	HOME	PTBL	STN	STEP	FEED	0	HOLD	CANCEL
3	0	0	0	0	0	0	DIR	INC
4	Command data							
5								
6								
7								

Command Codes

Command codes are used to specify positioning and other commands. To start execution of a command, set the command code and command data first (or at the same time), and then change the Command Start Command from 0 to 1.

Command Codes	Description
0000	No operation
0001	Simple positioning
0010	External positioning
0011	Positioning with notch signal outputs
0100	Multi-speed positioning

Response Types

The response type in the command message specifies the type of data that will be stored as the response data in the response message. The NS300 Unit creates response data in the response messages based on the specified response type.

Response Type	Response Data
0000	Command position (reference units)
0001	Current position (reference units)
0010	Position error (reference units)
0011	Command speed (1000 reference units/min)
0100	Current speed (1000 reference units/min)
0101	Torque (%)
1010	Station number
1011	Point table number

Cancel Command: CANCEL

If the Cancel Command is set to 1 during execution of a move command, the execution of the move command will be stopped and the servomotor will decelerate to a stop. The remaining travel distance will be canceled.

Hold Command: HOLD

If the Hold Command is set to 1 during execution of a move command, the execution of the move command will be held and the servomotor will decelerate to a stop. The NS300 Unit will wait for command execution to be restarted. Set the Hold Command to 0 again to restart the execution of a move command.

Constant Feed Command: FEED

The NS300 Unit will start feeding at a constant speed when it detects the leading edge of the FEED bit. Constant feeding will continue while this bit is set to 1.

When the FEED bit is set to 0, the servomotor will decelerate to a stop. The direction for feeding is determined by the Movement Direction set in the command data area.

An override can be set for this command in the command data area. The override can be from 0% to 200% of the parameter speed or a specific speed can be set in the command data. Which method is used depends on Pn830 (Constant Feed Reference Unit Selection).

Settings Data Area	Description	
Movement direction	0: Forward 1: Reverse	
Command data	When Pn830 = 0	Set an override value (0 to 200). Set to 100 when not using the override function.
	When Pn830 = 1	Set the feed speed.

Step Command: STEP

The NS300 Unit will start step operation when it detects the leading edge of the STEP bit. While the STEP bit is set to 1, the axis will travel only the distance set in the specified parameter. If the STEP bit is set to 0 during step operation, the servomotor will decelerate to a stop and the step operation will end. The remaining travel distance will be canceled.

The direction of movement for step operations is determined by the Movement Direction set in the command data area. The number of the step travel distance (0 to 3) is also set in the command data area. The parameter data set in Pn844 to Pn847 will be used for the step travel distance.

Settings Data Area	Description
Movement direction	0: Forward 1: Reverse
Command data	Set the selection number for step travel distance. 0: Uses Pn844 data. 1: Uses Pn845 data. 2: Uses Pn846 data. 3: Uses Pn847 data.

Station Command: STN

The NS300 Unit will start station operation when it detects the leading edge of the *STN* bit. If this command is set to 0 while the axis is travelling, the servomotor will decelerate to a stop and the station operation will end. The remaining travel distance will be cancelled.

Settings Data Area	Description
Movement direction	0: Forward 1: Reverse
Absolute/incremental value	Specify whether the station number is an absolute value or incremental value.
Command data	Specify the target station number.

Point Table Command: PTBL

The NS300 Unit will start point table operation when it detects the leading edge of the PTBL bit. If this command is set to 0 while the axis is travelling, the servomotor will decelerate to a stop and the point table operation will end. The remaining travel distance will be cancelled.

Settings Data Area	Description
Absolute/incremental value	Specify whether the position data in the point table is an absolute value or incremental value.
Command data	Specify the point table number to be used.

Zero Point Return Command: HOME

The NS300 Unit will start a zero point return when it detects the leading edge of the HOME bit. If this command is set to 0 while the axis is travelling, the servomotor will decelerate to a stop and the zero point return operation will end. The zero point return operation will not restart even if the HOME bit is set to 1 again.

The type of zero point return depends on the zero point return mode setting in Pn800.

Incremental Specification: INC

The INC bit specifies whether the data that indicates a position is used as an absolute value or an incremental value. Set this bit to 0 to specify an absolute position and to 1 to specify an incremental position.

This setting is used for the following commands.

- Station Command
- Point Table Command
- Positioning Command

Movement Direction: DIR

The DR bit specifies the movement direction. Set this bit to 0 for forward and to 1 for reverse operation.

This specification is used for the following commands.

- Feed Command
- Step Command
- Station Command

The movement direction specification is disabled during normal positioning.

Response Messages

The response messages for move commands are shown below.

Table 5.4 Responses for Move Commands

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	Response type				Command code			
2	HOME_R	PTBL_R	STN_R	STEP_R	FEED_R	0	HOLD_R	PRGS
3	POT	NOT	INPOS	NEAR	HOME_P	0	DIR_R	INC_R
4	Command message							
5								
6								
7								

Progressing Flag: PRGS

The PRGS bit is set to 1 during the execution of a command. For move commands, this flag will be set to 1 while outputting to the SGDH.

When command execution has been completed or when a Cancel Command or other stop command has been received, the Progressing Flag is set to 0.

Holding Flag: HOLD_R

The HOLD_R bit is set to 1 when a Hold Command is received from the host device and for the duration of the hold.

The host device can confirm that the NS300 Unit correctly received the Hold Command by checking that the Holding Flag is set to 1.

Constant Feed Flag: FEED_R

The FEED_R bit is set to 1 when a Constant Feed Command is received from the host device and while constant feeding is being executed. When the Constant Feed Command is set to 0, this flag is also set to 0. The host device can confirm that the NS300 Unit correctly received the Constant Feed Command by checking that the Feed Flag is set to 1.

This flag is set to 1 even if constant feeding cannot be executed because of a Servo OFF status, for example. The user must monitor for alarms during constant feeding for the Constant Feed Command.

The movement direction during constant feeding can be checked using the Movement Direction Flag: DIR_R.

Step Flag: STEP_R

The STEP_R bit is set to 1 when a Step Command has been received from the host device and during step operation. This flag is set to 0 when the step operation has been completed normally or cancelled. The host device can confirm that the NS300 Unit correctly received the Step Command by checking that the Step Flag is set to 1.

This flag is set to 1 even if the step operation cannot be executed because of a Servo OFF status, for example. The user must monitor for alarms during step operation.

Station Flag: STN_R

The STN_R bit is set to 1 when a Station Command has been received from the host device and during station operation. This flag is set to 0 when the station operation has been completed normally or cancelled. The host device can confirm that the NS300 Unit correctly received the Station Command by checking that the Station Flag is set to 1.

This flag is set to 1 even if the station operation cannot be executed because of a Servo OFF status, for example. The user must monitor for alarms during station operation.

Point Table Flag: PTBL_R

The PTBL_R bit is set to 1 when a Point Table Command has been received from the host device and during point table operation. This flag is set to 0 when the point table operation has been completed normally or cancelled. The host device can confirm that the NS300 Unit correctly received the Point Table Command by checking that the Point Table Flag is set to 1.

This flag is set to 1 even if the point table operation cannot be executed because of a Servo OFF status, for example. The user must monitor for alarms during point table operation.

Zero Point Return Flag: HOME_R

The HOME_R bit is set to 1 when a Zero Point Return Command has been received from the host device and during zero point return. This flag is set to 0 when the zero point return has been completed normally or cancelled. The host device can confirm that the NS300 Unit correctly received the Zero Point Return Command by checking that the Zero Point Return Flag is set to 1.

This flag is set to 1 even if the zero point return cannot be executed because of a Servo OFF status, for example. The user must monitor for alarms during zero point return.

Incremental Specification Flag: INC_R

The INC_R bit reflects the status of the Incremental Specification in the command message. The host device can confirm by the change of status of this flag that the NS300 Unit has correctly received change in the incremental specification.

Movement Direction Flag: DIR_R

The DIR_R bit indicates the current command rotation direction for the servomotor. If the servomotor has stopped, this flag indicates the last command rotation direction. This flag is set to 0 to indicate forward, and to 1 to indicate reverse.

Zero Point Flag: HOME_P

The HOME_P bit is set to 1 when the servomotor is within the zero point range. The zero point range is set in Pn806 (Zero Point Return Output Width).

When an incremental position detection system is used, this flag cannot be set to 1 for the period from when power is turned ON to the SGDH until the initial zero point return has been completed.

Near Signal Flag: NEAR

The NEAR bit is set to 1 when the current position is within the On-target position range. When the current position is outside the On-target position range, the flag is set to 0. The On-target position range is set in Pn852 (Positioning Proximity Detection Width).

In-position Flag: INPOS

The INPOS bit is set to 1 when the current position is within the positioning completed range of the target position. The flag is set to 0 when the current position is outside the positioning completed range. The On-target position range is set in Pn850 (Positioning Deadband).

Negative Overtravel Flag: NOT

The NOT bit indicates the status of the negative overtravel signal for the external input connected to CN1 on the SGDH.

Positive Overtravel Flag: POT

The POT bit indicates the status of the positive overtravel signal for the external input connected to CN1 on the SGDH.

5.3.4 Set/Read Command Messages

■ Command Messages

Details on bytes 1 to 7 of the command messages for set/read commands (MOD = 1) are shown below.

The response type does not need to be specified for set/read commands.

Table 5.5 Set/Read Commands

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	ALRST	ESTP	0	0	SVON	C_STRT
1	0				Command code			
2	Command number							
3								
4	Command message							
5								
6								
7								

Command Codes

A list of command codes is shown in the following table. Set "No operation" for commands that will not be executed.

The command codes in the response messages will basically be a copy of the command codes in the command messages. A warning for parameter setting error will be returned when the parameter number is different or the data is outside the setting range for the parameter.

Table 5.6 Command Codes

Command Code	Description
0000	No operation
1000	Read parameter
1001	Write parameter
1010	Set current position
1011	Set zero point
1100	Read alarm
1110	Reset Unit

■ Response Messages

Details on bytes 1 to 7 of the response messages for set/read commands (MOD = 1) are shown below.

Table 5.7 Responses for Set/Read Commands

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	0				Command code			
2	Command number							
3								
4	Response data							
5								
6								
7								

Parameter Read Command

The Parameter Read Command reads SGDH and NS300 Unit parameters.

To use the Parameter Read Command, make the following settings and then change the Command Start Command from 0 to 1.

- Command code
- Parameter number

Table 5.8 Parameter Read Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	ALRST	ESTP	0	0	SVON	C_STRT
1	0				8			
2	Parameter number							
3								
4	0							
5								
6								
7								

Table 5.9 Response for Parameter Read Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1		0				8		
2	Parameter number							
3								
4	Parameter data							
5								
6								
7								

Parameter Write Command

The Parameter Write Command writes SGDH and NS300 Unit parameters.

To use the Parameter Write Command, make the following settings and then change the Command Start Command from 0 to 1.

- Command code
- Parameter number
- Parameter data

Table 5.10 Parameter Write Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	ALRST	ESTP	0	0	SVON	C_STRT
1	0				9			
2	Parameter number							
3								
4	Parameter data							
5								
6								
7								

Table 5.11 Response for Parameter Write Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	0				9			
2	Parameter number							
3								
4	Parameter data							
5								
6								
7								

■ Current Position Setting Command

The Current Position Setting Command sets the specified value as the current position of the servomotor.

To use the Current Position Setting Command, make the following settings and then change the Command Start Command from 0 to 1.

- Command code
- Current position data

Table 5.12 Current Position Setting Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	ALRST	ESTP	0	0	SVON	C_STRT
1	0				10 (decimal)			
2	0							
3								
4	Current position data							
5								
6								
7								

Table 5.13 Response for Current Position Setting Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	0				10 (decimal)			
2	0							
3								
4	Current position data							
5								
6								
7								

■ Alarm Read Command

The Alarm Read Command reads the last four alarms that have occurred on the SGDH and the NS300 Unit.

To use the Unit Reset Command, make the command code settings and then change the Command Start Command from 0 to 1.

Table 5.14 Alarm Read Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	ALRST	ESTP	0	0	SVON	C_STRT
1	0				12 (decimal)			
2	0							
3								
4	0							
5								
6								
7								

Table 5.15 Response for Alarm Read Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	0				12 (decimal)			
2	0							
3								
4	Last alarm code							
5	Second last alarm code							
6	Third last alarm code							
7	Fourth last alarm code							

■ Unit Reset Command

The Unit Reset Command restarts the SGDH and NS300 Unit software. When this command is executed, the NS300 Unit parameters are stored in flash ROM and then the NS300 Unit is restarted.

To use the Unit Reset Command, make the command code settings and then change the Command Start Command from 0 to 1.

When the NS300 Unit is resetting the Unit, the Unit Ready Flag is set to 0. When the reset has been completed, the flag is set to 1.

Table 5.16 Unit Reset Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	0	ALRST	ESTP	0	0	SVON	C_STRT
1	0				14 (decimal)			
2	0							
3								
4	0							
5								
6								
7								

Table 5.17 Response for Unit Reset Command

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	1	READY	PWRON	ESTP_R	ALRM	WARN	SVON_R	C_STRT_R
1	0				14 (decimal)			
2	0							
3								
4	0							
5								
6								
7								