ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO EN INGENIERÍA

“Diseño e Implementación de un Controlador Programable para la Estación Neumática PN–2800”

AUTORES:

Rommel David Fernández Granja

Christian Alejandro Sánchez Carranza

SANGOLQUÍ – ECUADOR

2007

CERTIFICACIÓN

Certificamos que el presente proyecto fue realizado en su totalidad por los señores Rommel David Fernández Granja y Christian Alejandro Sánchez Carranza, como requisito parcial para la obtención del título en INGENIERÍA ELECTRÓNICA.

Ing. Víctor Proaño.	Ing. Hugo Ortiz.
DIRECTOR	CODIRECTOR

AGRADECIMIENTO

Agradecemos, de manera especial a la Escuela Politécnica del Ejército por habernos dado la solvente preparación académica acorde a los avances científicos y tecnológicos contemporáneos. Individualmente agradecemos a nuestros padres por su permanente apoyo en todos estos años, a nuestras hermanas y hermanos por su oportuna colaboración; al Director y Codirector de nuestro Proyecto de Grado, señores ingenieros: Víctor Proaño y Hugo Ortiz respectivamente, por sus generosos aportes y gentiles sugerencias.

DEDICATORIA

A la persona que le sea útil el contenido de este trabajo.

PRÓLOGO

El presente proyecto titulado “Diseño e Implementación de un Sistema de Control Programable para la Estación Neumática PN2800”, perteneciente al Laboratorio de Robótica del Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejército, tiene por objetivo la implementación de un sistema sencillo y económico para la programación de rutinas de control de la estación neumática PN-2800. Estas rutinas de control serán cargadas en la memoria de programa del microcontrolador. El microcontrolador ejecutará las rutinas programadas y a través de sus puertos de salida comandará las electro-válvulas para ejecutar las rutinas de la estación neumática.

Además el presente proyecto diseñará el hardware de entradas y salidas del controlador lógico programable tanto para acoplar los niveles de tensión a los sensores y actuadores de la estación neumática como también para controlar y monitorear todas las señales que requiere la estación neumática.

Por otro lado se desarrollará las librerías que faciliten la escritura de programas de las rutinas de control de la Estación neumática en un lenguaje amigable de alto nivel provista mediante comunicación serial que permitirá la carga y descarga de programas.

Finalmente la meta principal del presente proyecto es elaborar un conjunto de prácticas de laboratorio para verificar las bondades del prototipo desarrollado.

ÍNDICE GENERAL

CAPÍTULO I

INTRODUCCIÓN

El laboratorio de Robótica del Departamento de Eléctrica y Electrónica de la Escuela Politécnica del Ejército dispone de una estación neumática adquirida a la firma DEGEM SYSTEMS. La denominación asignada a la estación neumática es PN-2800. La Estación Neumática PN-2800 es un Sistema de suministro y clasificación de materia prima, sus elementos finales de control son electro-válvulas de relativa baja potencia que efectúan un control tipo ON/OFF sobre los actuadores neumáticos.

La estación neumática en su diseño original estaba provista de un PLC Modicon AEG 984 con los respectivos módulos de Entrada/Salida discreta que se utilizaban para la adquisición de información que proviene de sensores de posición y la correspondiente actuación de los elementos finales de control.

Si bien es cierto que existe en la actualidad gran variedad de controladores lógicos programables es no menos cierto que la cantidad de entradas y salidas que la estación necesita, impide obtener un PLC de bajo costo que se ajuste a estos requerimientos. El fabricante original Degem Systems colocó el PLC Modicon AEG 984 que está subutilizado en la estación neumática. Existe un desperdicio en el sistema original puesto que no se justifica usar un PLC que incluye múltiples capacidades de programación si estas no son utilizadas nunca.

El sistema que se propone en este proyecto será dedicado y por tanto se ajustará exactamente a lo que la estación neumática requiere.

Un controlador lógico programable PLC basa su funcionalidad en un dispositivo microprocesador. La meta final constituye realizar un PLC dedicado para la

estación neumática. El sistema que se ha desarrollado es de bajo costo y por lo mismo puede quedar ligado definitivamente a la estación neumática. La estación neumática tendrá entonces un desempeño muy cercano al que originalmente tuvo con el PLC Modicon AEG 984.

El proyecto permite que el PLC que originalmente estuvo ligado a la estación pueda ser utilizado en aplicaciones más exigentes entre las que se incluyen las aplicaciones de la asignatura de PLC. Otro aspecto que justificó la realización del proyecto es la cantidad de entradas y salidas a ser manejadas que exceden la capacidad que puede obtenerse en forma directa de un microcontrolador. También, se presentó la necesidad de acoplar los niveles de tensión de la estación neumática para la interconexión al microcontrolador.

El sistema permite que el estudiante programe en lenguaje de alto nivel las rutinas a realizar por parte de la estación PN-2800. Mediante una interface RS-232 el programa puede ser descargado al microcontrolador sin necesidad de equipos adicionales.

Esto significa que el estudiante podrá diseñar algoritmos de control para la estación neumática y comprobar el funcionamiento. Mediante el prototipo diseñado el estudiante tendrá la capacidad de operar la estación neumática en forma manual desde una consola de operación en el computador y la posibilidad de explorar las funciones básicas del protocolo Modbus-RTU. Además estará habilitado para realizar prácticas de HMI's (Human Machine Interface) mediante Lookout.

1.1 CONTROLADOR

"Los microcontroladores están conquistando el mundo. Están presentes en nuestro trabajo, en nuestra casa y en nuestra vida, en general. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadores, en los teléfonos, en los hornos microondas y los televisores de

nuestro hogar. Pero la invasión acaba de comenzar y en el transcurso del siglo XXI seremos testigos de la conquista masiva de estos diminutos computadores, que gobernarán la mayor parte de los aparatos que usamos los humanos.

El concepto de controlador ha permanecido invariable a través del tiempo, su implementación física ha variado frecuentemente. Hace tres décadas, los controladores se construían exclusivamente con componentes de lógica discreta, posteriormente se emplearon los microprocesadores, que se rodeaban con chips de memoria y Entradas/Salidas sobre una tarjeta de circuito impreso. En la actualidad, todos los elementos del controlador se han podido incluir en un chip, el cual recibe el nombre de microcontrolador. Realmente consiste en un sencillo pero completo computador contenido en el corazón (chip) de un circuito integrado." [1]

El microcontrolador puede caracterizarse como:

- Una microcomputadora en un solo chip.
- Optimizado para aplicaciones de control (Real Time).

PIC es el nombre con que el fabricante MICROCHIP, nombra a sus microcontroladores. Los PIC tienen "algo" que fascina a los diseñadores, puede ser la velocidad, el precio, la facilidad de uso, la información, las herramientas de apoyo. Quizás un poco de todo eso es lo que produce esa imagen de sencillez y utilidad. Es probable que en un futuro próximo otra familia de microcontroladores le arrebate ese "algo".

Se desea constatar que para las aplicaciones más habituales (casi un 90%) la elección de una versión adecuada de PIC es la mejor solución; sin embargo, dado su carácter general, otras familias de microcontroladores son más eficaces en

[1] MICROCONTROLADORES.pdf incluido en el CD de documentación.

aplicaciones específicas, especialmente si en ellas predomina una característica concreta, que puede estar muy desarrollada en otra familia.

Los detalles más importantes que buscan los profesionales de la microelectrónica y microinformática y las razones de la excelente acogida que tienen los PIC son los siguientes:

- Sencillez de manejo: Tienen un juego de instrucciones reducido; 35 en la gama media.

- Buena información, fácil de conseguir y económica.

- Precio: Su costo es comparativamente inferior al de sus competidores.

- Poseen una elevada velocidad de funcionamiento. Buen promedio de parámetros: velocidad, consumo, tamaño, alimentación, código compacto, etc.

- Herramientas de desarrollo fáciles y baratas. Muchas herramientas software se pueden recoger libremente a través de Internet desde Microchip.

- Existe una gran variedad de herramientas hardware que permiten grabar, depurar, borrar y comprobar el comportamiento de los PIC.

- Diseño rápido.

- La gran variedad de modelos de PIC permite elegir el que mejor responde a los requerimientos de la aplicación.

1.2 MANDOS NEUMÁTICOS

La Neumática es la rama de la técnica que se dedica al estudio y aplicaciones prácticas del aire comprimido. El aire comprimido es aire tomado de la atmósfera y confinado a presión en un espacio reducido. Por ejemplo cuando inflamos un globo y posteriormente lo soltamos sin cerrar, la energía acumulada por el aire lo hace revolotear rápidamente por la habitación. Se produce una transformación de la energía almacenada en trabajo útil en mover el globo.

Hoy en día son muchos los sistemas técnicos que basan su funcionamiento en este tipo de energía. Por ejemplo, las puertas de algunos autobuses y trenes se accionan con aire comprimido; en la industria son muy útiles los sistemas neumáticos porque proporcionan movimiento lineal y desarrollan grandes fuerzas, utilizándose para empujar y levantar cargas pesadas, en cadenas de montajes automatizados, etc.

En los sistemas neumáticos, el aire comprimido se produce en un elemento llamado compresor, que es una bomba de aire comprimido accionada normalmente por un motor eléctrico. Este aire se almacena en un depósito denominado receptor. Desde éste, el aire es conducido a través de válvulas a los cilindros, que son los componentes encargados de realizar el trabajo. Cuando el aire comprimido fluye en el interior de un cilindro, aumenta la presión y obliga a desplazarse a un émbolo situado en su interior, y proporcionando un movimiento lineal y realizando un trabajo.

Las válvulas tienen como misión controlar el flujo de aire comprimido que entra y sale de los cilindros. Las válvulas son los elementos de control del circuito.

Existen válvulas direccionales que son de mando, que distribuyen el aire comprimido hacia los elementos de trabajo. Dos de sus características principales que posibilitan su clasificación son las vías y las posiciones que poseen.

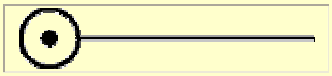
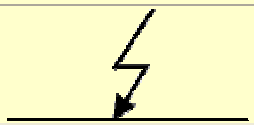
Las vías son los orificios de Entrada/Salida y las posiciones representan las conexiones posibles entre las vías. Estas válvulas podrán ser también mono o bi-estables de acuerdo a las posiciones que tome por acción de mando. El mando puede estar dado por acción mecánica, eléctrica o neumática.[2]

1.2.1 Simbología Neumática.

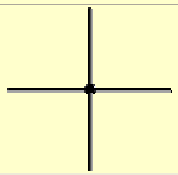
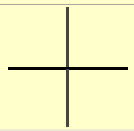
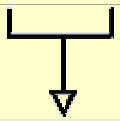
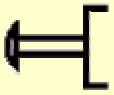
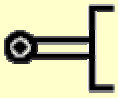
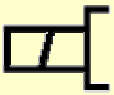
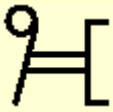
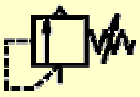
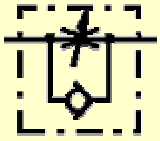
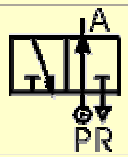
En la representación de los circuitos neumáticos se utiliza una simbología específica, siguiendo las normas establecidas por los organismos correspondientes UNE (Una Norma Española), ISO (Organización Internacional para la Estandarización), DIN (Instituto Alemán de Normalización). Los esquemas neumáticos son una representación de las instalaciones neumáticas reales.

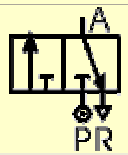
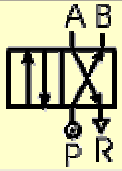
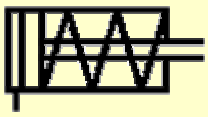
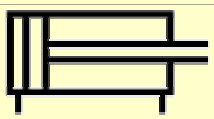
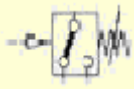
En la Tabla. 1.1 se puede ver la simbología neumática de conexiones, accionamientos, válvulas y cilindros.

Tabla. 1.1. Simbología.[3]

Simbología de conexiones	
Símbolo:	Descripción:
	Conducto de trabajo
	Conducto flexible
	Origen de presión. P
	Conducción eléctrica
	Conducto de mando
	Conducto de escape

[2] http://www.sociedadelainformacion.com/20011204/neumatica/neumatica.htm
[3] http://proton.ucting.udg.mx/temas/control/nares/simbolos/sim_conex.html

	Conexiones fijas
	Cruce de conexiones
	Escape recuperable
	Pulsador de emergencia. Seta.
	Accionamiento por rodillo.
	Electro-válvula, solenoide SOV.
	Accionamiento por Palanca.
	Accionamiento por Pedal.
	Retorno por muelle.
	Regulador de presión.
	Regulador de caudal unidireccional RV.
	Válvula 3 vías 2 posiciones (3/2) normalmente abierta

	Válvula 3 vías 2 posiciones (3/2) normalmente cerrada
	Válvula 4 vías 2 posiciones (4/2)
	De simple efecto. Retorno por muelle.
	De simple efecto. Retorno por fuerza externa.
	De doble efecto.
	Presostato, aparato que conmuta con una presión determinada, regulable.

Nomenclatura de conexiones: las conexiones en las vías de las válvulas responden de acuerdo con la siguiente nomenclatura para letras y números:

A, B, C (2, 4, 6): Se utilizan en orden ascendente conforme el número de salidas existentes.

P (1) : Indica alimentación o inicio de presión.

R, S, T (3, 5, 7): Escapes de aire comprimido

En la Figura. 1.1 se puede ver la conexión electro-neumática de la estación, que contiene las siguientes partes:

- Fuente Neumática (P)

- Válvula reductora de presión
- Electro-válvula (SOV) 4 vías 2 posiciones (4/2), accionamiento de palanca con retorno con muelle
- Dos reguladores de caudal unidireccional (RV1,RV2) y
- Actuador un cilindro de doble efecto (C).

Los elementos de mando identifican la forma en que será accionada, una válvula, es decir por solenoide y palanca con retorno con muelle. Este símbolo acompañado de la válvula permite en un momento dado ubicar y realizar mantenimiento a una válvula con el seguimiento de un diagrama. El diagrama indica el efecto de los accionamientos de la válvula sobre el actuador.

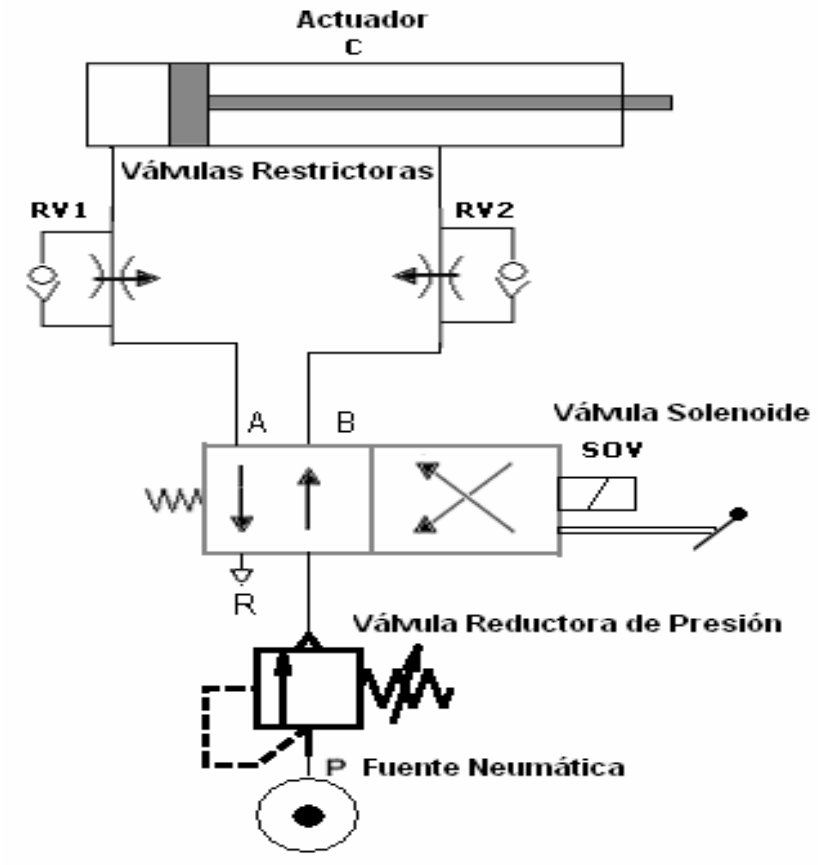


Figura. 1.1. Ejemplo de conexión electro-neumática.

CAPÍTULO II

MARCO TEÓRICO

2.1 FUNCIONALIDAD DE LA ESTACIÓN NEUMÁTICA.

La Estación Neumática PN-2800 (Figura. 2.1.), es un sistema de suministro y clasificación de materia prima, utilizada para alimentar con una variedad de materias primas y patrones:

1. Base rectangular
2. Barra cilíndrica de 26mm de diámetro
3. Barra cilíndrica de 20mm de diámetro
4. Pallet Standard, en el cual se deposita el material

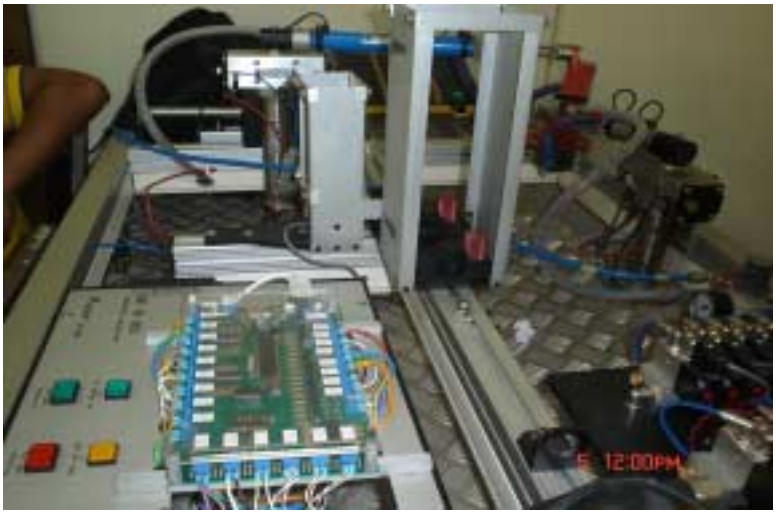


Figura. 2.1. Estación Neumática PN-2800.

"La estación consiste en 6 dispositivos principales:

1. Un depósito para pallets, con una capacidad máxima de 8 pallets, equipado con un manipulador de carga y dispositivos sensores.

2. Un depósito para las bases rectangulares, que puede contener hasta 13 bases, equipado con un manipulador de carga, dispositivos sensores y un contador para las bases existentes en el inventario.

3. Dos depósitos para barras cilíndricas, cada uno de los cuales con capacidad para un máximo de 10 barras, equipado con un carril y con un manipulador de carga, dispositivos sensores y un contador para los tipos de barras existentes en el inventario.

4. Un robot eléctrico, utilizado para cargar pallets, bases y barras para el sistema. El robot está equipado con agarradores capaces de percibir el diámetro apropiado de las barras cilíndricas.

5. Una célula para recolectar restos, utilizada para reunir barras imperfectas que fueron removidas durante el proceso de alimentación.

6. Un sistema de control basado en PC, con controladores lógicos programables ligados a la red de comunicación.

La manipulación de robots o cilindros de la estación se consigue físicamente a través de electro-válvulas, sin embargo, en lo que se refiere a la materia prima, se utiliza sensores de proximidad que se encargan de verificar la presencia o no de materia prima.

La estación realiza varias verificaciones durante el ingreso de materiales al proceso de alimentación:

A. Para los pallets:

La estación verifica si existen pallets en el área de carga.

La tarea es realizada por medio de dos sensores:

1. Un sensor localizado en la parte inferior del depósito de los pallets, que siente si hay o no pallets disponibles en el mismo.

2. Un sensor de proximidad que ayuda al manipulador a empujar el pallet para el local apropiado de la estación de carga.

B. Para las bases rectangulares:

Un sensor localizado en el almacén de las bases rectangulares, que siente si hay o no bases disponibles en el depósito. No suministra el número exacto de bases existentes en el inventario.

C. Para las barras cilíndricas.

Son efectuadas tres verificaciones:

1. Sensores localizados en los dos depósitos de barras cilíndricas, que detectan si hay ó no barras disponibles en el depósito.

2. Luego que una barra cilíndrica fue trasladada al carril de alimentación, un sensor de proximidad indica si el manipulador empujó la barra al lugar apropiado.

3. La estación realiza también una función de inspección de las barras cilíndricas, durante la entrada de las mismas: sensores localizados en los agarradores del robot indican si fue cargada una barra de diámetro apropiado. En caso contrario, el robot removerá la barra errada y la pondrá en una célula especial.

El proceso de alimentación es iniciado por el operador local.

La estación alimenta un pallet vacío, después lo carga con la materia prima deseada y finalmente mueve el pallet cargado a un vagón". [4]

2.1.1 Elementos de la Estación.

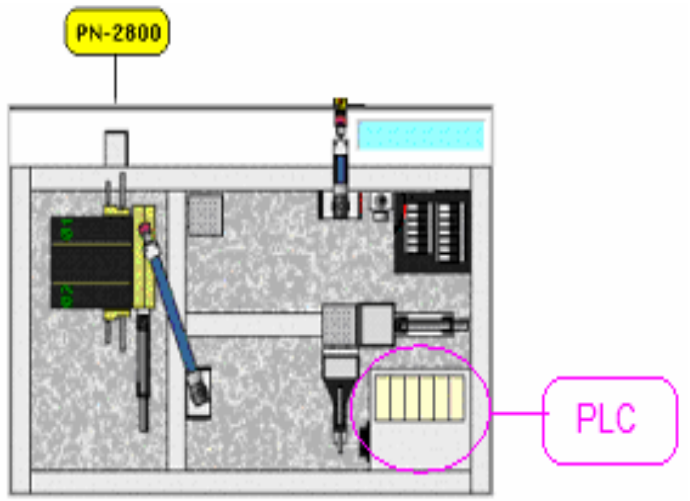


Figura. 2.2. Ubicación del PLC en la Estación Neumática PN-2800.

Para la correcta operación de toda la estación neumática, se utiliza un PLC (Figura. 2.2.), el cual activa y desactiva todos los elementos de la estación, este PLC tiene:

- 2 módulos DEP 216 (Entradas discretas a 24VDC)
- 1 módulo DAP 216 (Salidas discretas a 24VDC)
- 1 módulo DAP 212 (Entradas/Salidas)

Estos módulos de entradas y salidas permiten que el PLC tenga una comunicación con cada actuador y sensores.

[4] En CIM2000.doc Incluido en el CD de documentación.

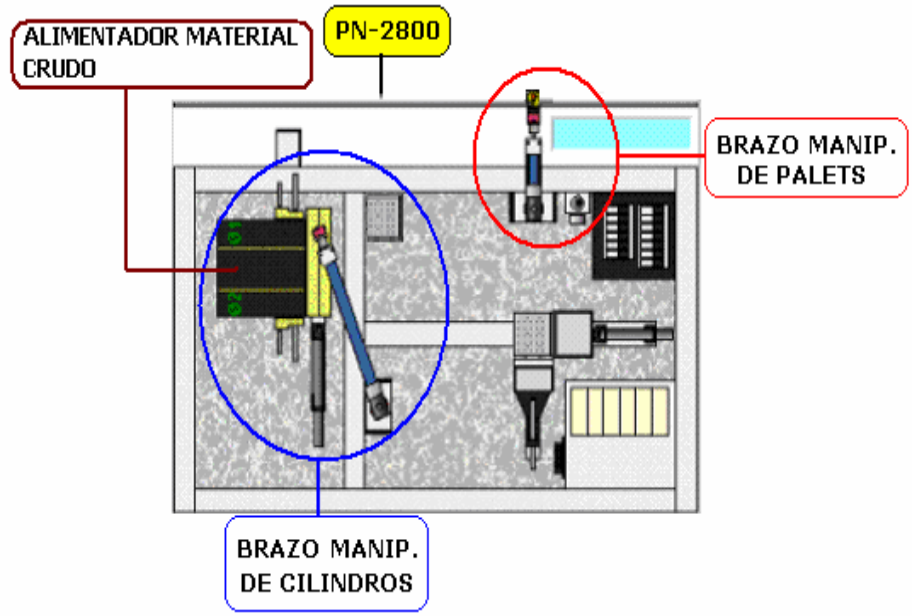


Figura. 2.3. Ubicación de Brazos Robots en la Estación Neumática PN-2800.

El alimentador de material crudo consiste en dos alimentadores para las barras cilíndricas (Figura. 2.3.). Cada uno de ellos es capaz de contener hasta diez barras cilíndricas, está provisto con sensores de proximidad y dos actuadores neumáticos (Figura. 2.4. (a)). El alimentador de material crudo incluye una barra cargante con el sensor a la posición cargante y un actuador neumático que se usan para el transporte de la barra a la posición cargante. Se reciclan las barras impropias que fueron descartadas durante el proceso, hacia una célula que recolecta la materia prima defectuosa.

(a) **(b)**

Figura. 2.4. Brazos robots: a) Brazo Manipulador de Cilindros; b) Brazo Manipulador de Pallets.

Los elementos motrices o actuadores del brazo son neumáticos emplean el aire comprimido como fuente de energía y son muy indicados en el control de movimientos rápidos, pero de precisión limitada como se puede ver en Figura. 2.4. (b).

La función de los brazos es la de manipular tanto cilindros como pallets, es decir, la materia prima, desde el área de alimentación hasta la banda transportadora, éstos son neumáticos y controlados por electro-válvulas y éstas a su vez por el PLC y PC. Cada brazo manipulador consta de las siguientes partes:

- Brazo.
- Actuadores o cilindros neumáticos.
- Gripper: una herramienta, que será la encargada de materializar el trabajo previsto.
- Mangueras de conexiones (RIL SAN)

En la Figura. 2.5 se muestra el diagrama esquemático del manipulador de materia prima.

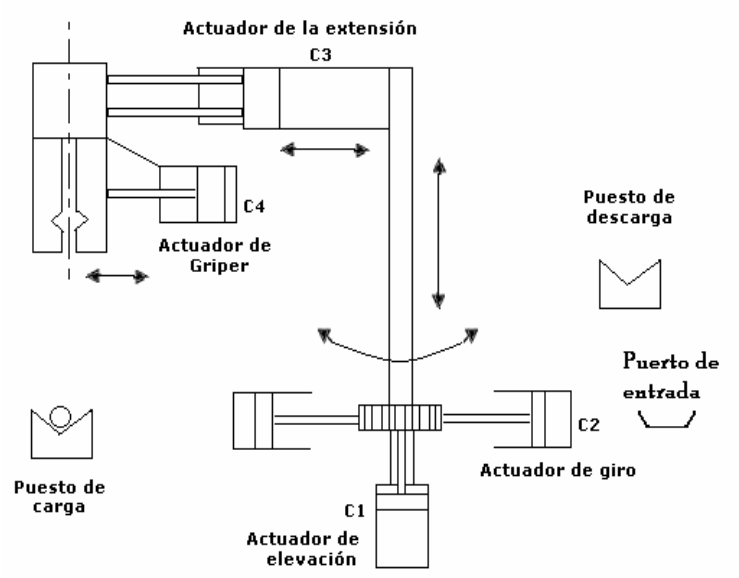


Figura. 2.5. Diagrama del brazo manipulador de cilindros.

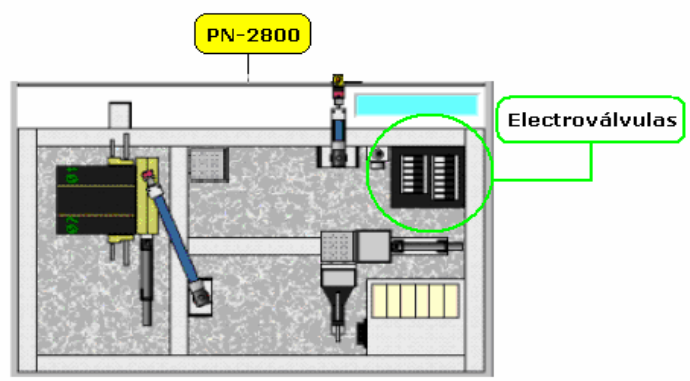


Figura. 2.6. Ubicación de las válvulas Solenoide en la Estación Neumática PN-2800.

En la sección señalada en la Figura. 2.6 están ubicadas las electro-válvulas con las siguientes características:

- Cantidad de electro-válvulas: 15
- Son válvulas solenoides 4/2 (4 vías y 2 posiciones) de tipo directo, es decir, al ser energizada mueve el pistón o actuador a la posición extendida y cuando es desactivada se invierte el sentido de flujo en la válvula y el pistón se mueve hacia la posición retraída.
- Son controladas por señales enviadas desde el PLC

En la Figura. 2.7 se observa la operación neumática de las válvulas.

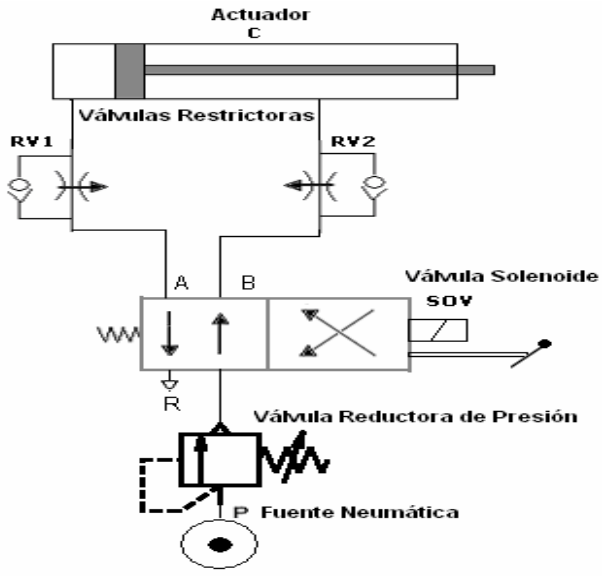


Figura. 2.7. Diagrama neumático de las válvulas.

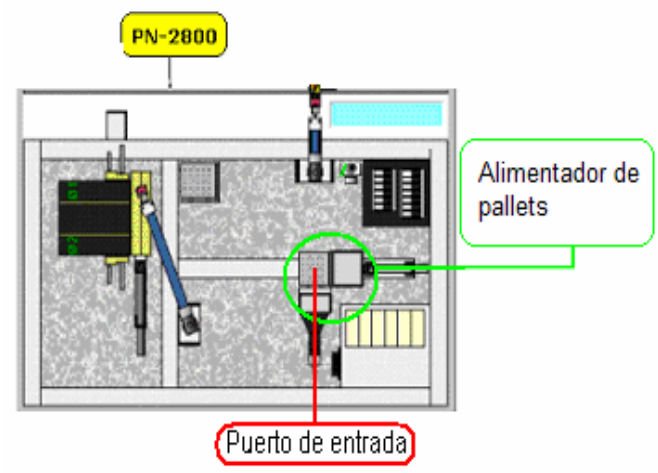


Figura. 2.8. Ubicación del Alimentador de Pallets en la Estación Neumática PN-2800.

El alimentador de pallets incluye un almacén de pallets vertical, capaz de contener ocho pallets normales y un alimentador corredizo operado por un actuador neumático. Este actuador se usa para recibir el pallet fuera del almacén al puerto de entrada como se puede ver en la Figura.2.8.

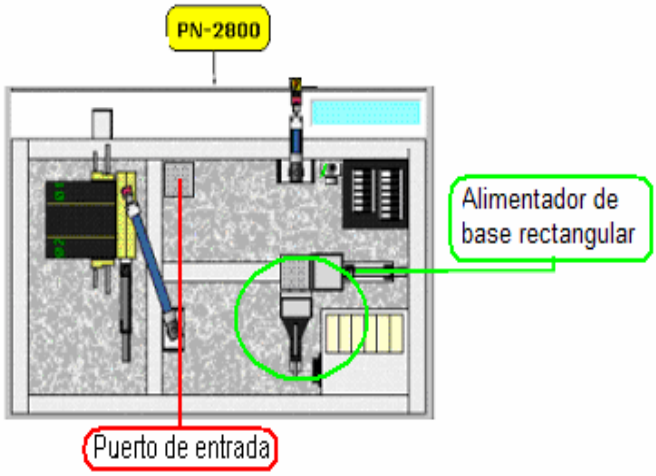


Figura. 2.9. Ubicación del Alimentador de Base Rectangular Estación Neumática PN-2800.

El alimentador de bases rectangulares incluye un almacén vertical capaz de contener 13 bases plásticas, y el alimentador corredizo opera por un actuador neumático. Este actuador se usa para cargar la base en un pallet como se puede ver en la Figura.2.9.

Figura. 2.10. Tablero de Control de la Estación Neumática PN-2800.

El tablero de control (Figura. 2.10.) tiene el cableado y conexiones entre fuente de poder eléctrica, protecciones, electro-válvulas y demás dispositivos que conforman la estación PN-2800, además contiene:

- Borneras – Bornera para riel de 16mm.
- Relés de propósito general – OMRON MY4 I2N/ 5A /24VAC.
- Fuente 220 V trifásica - Tipo Switching ETA Electric 3,2A@24V 100/200 V, cable de poder a toma monofásica-estándar europeo.
- Conexiones con cable No 18 - varios colores.
- Manguera flexible de protección 105º C VW-1.
- Fusible Térmico Siemens SSQ22 C2 400 Vn.
- Porta-fusibles monofásico Hager 3A 500 Vn.
- Dos Pulsadores Breter de 24 V, un pulsador con enclavamiento mecánico y tres luces indicadoras de 24 V.

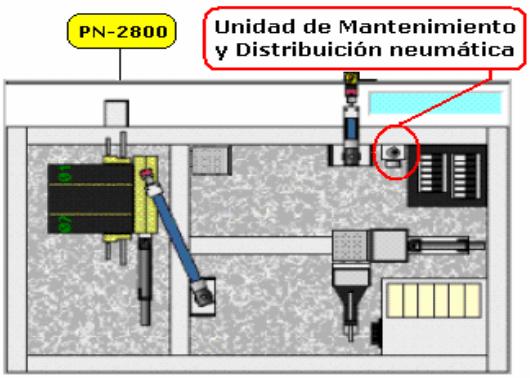


Figura. 2.11. Ubicación de la Unidad de Mantenimiento en la Estación Neumática PN-2800.

Para la correcta operación de los actuadores neumáticos debe ser instalada una unidad de mantenimiento (Figura. 2.11.), para garantizar que el aire que estamos utilizando sea el más adecuado y no produzca averías en los cilindros o en los demás dispositivos.

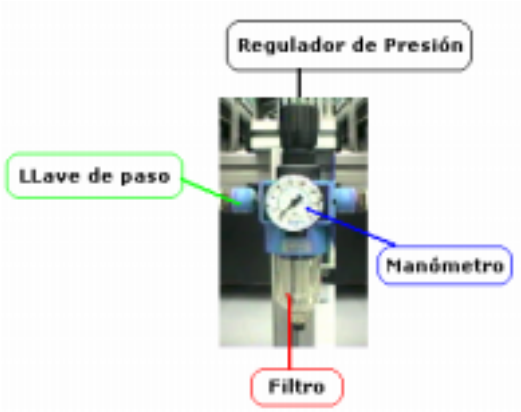


Figura. 2.12. Partes que conforman la Unidad de Distribución.

La unidad de mantenimiento dispone de los siguientes elementos:

- Estación Neumática lista para trabajar con una presión de 70 lb. /pulg., que indica el manómetro del regulador de presión.
- Regulador de presión con seguridad de lock (0 a 8 bares).
- Manómetro (Figura. 2.12.) - medición en: (0 a 10) bares o (0 a 140) lb. /pulg.

Los cilindros se encuentran ubicados alrededor de toda la estación neumática, estos son de simple y doble acción (Figura. 2.13.). Los cilindros de doble acción son los que están ubicados en el brazo manipulador de pallets y manipulador de cilindros.

El vástago de un cilindro de doble efecto debe salir o entrar según se accione una válvula.

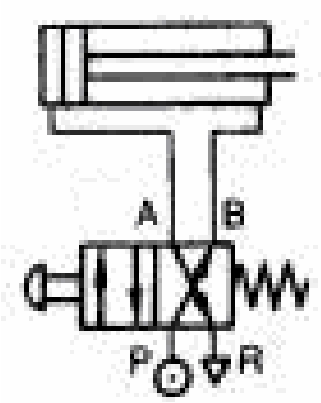


Figura. 2.13. Mando de un cilindro de doble efecto.

Este mando de cilindro puede realizarse tanto con una válvula distribuidora 4/2 como con una 5/2. La unión de los conductos de P hacia B y de A hacia R en la 4/2 mantiene el vástago retraído. Al accionar el botón de la válvula se establece la unión de P hacia A y de B hacia R.

El vástago del cilindro se extiende hasta la posición final de carrera. Al soltar el botón, el muelle recuperador de la válvula hace regresar ésta a la posición inicial. El vástago del cilindro vuelve a entrar.

Los cilindros de simple acción (Figura. 2.14.) son los que se encuentran en el almacén de barras rectangulares o pallets y en el almacén de materia prima. El vástago de un cilindro de simple efecto debe salir al accionar un pulsador y regresar inmediatamente al soltarlo.

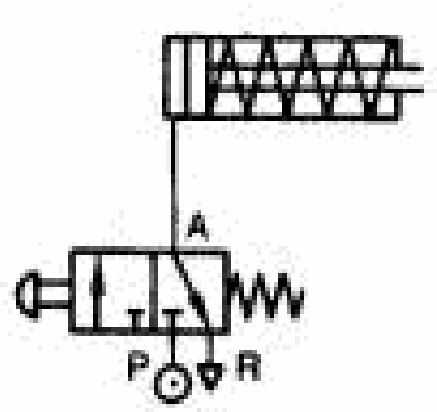


Figura. 2.14. Mando de un cilindro de simple efecto.

Se tiene los siguientes cilindros en la estación neumática:

- 2 Cilindros de doble vástago KOGANET SD40x150-N-180
- 2 Cilindros HUMPHREY 8 – S ½
- 2 Cilindros HUMPHREY 8 – D ½
- 1 Cilindro HUMPHREY 7D8
- 1 Cilindro HUMPHREY 8D4

El centro de entrenamiento se encuentra normalizado bajo normas DIN VDE 0641.

2.2 ELEMENTOS DEL SISTEMA DE CONTROL.

2.2.1 Microcontrolador.

Recibe el nombre de controlador (Figura. 2.15.) al dispositivo que se emplea para el gobierno de uno o varios procesos. Por ejemplo, el controlador que regula el funcionamiento de un horno dispone de un sensor que mide constantemente su temperatura interna y, cuando traspasa los límites prefijados, genera las señales adecuadas que accionan los efectores que intentan llevar el valor de la temperatura dentro del rango estipulado.[5]

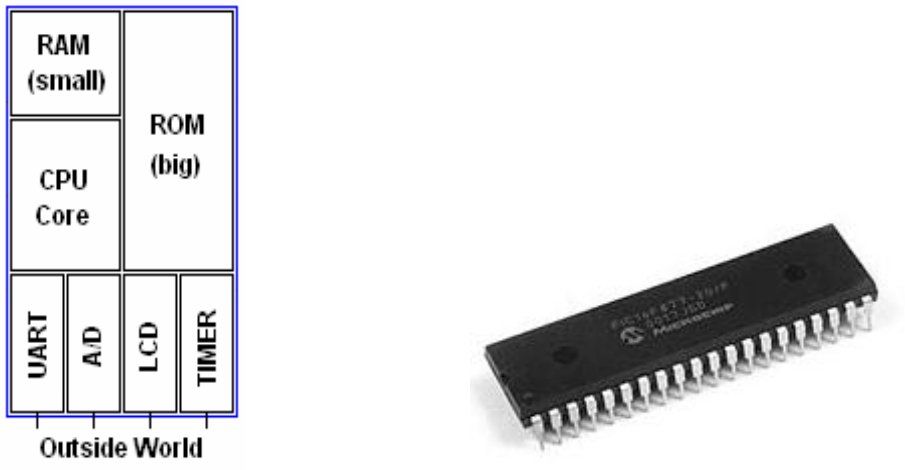


Figura. 2.15. Microcontrolador.

Un microcontrolador es un circuito integrado de alta escala de integración que incorpora la mayor parte de los elementos que configuran un controlador.

2.2.1.1 Arquitectura básica.

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de von Neumann, en el momento presente se impone la arquitectura Harvard.

[5] MICROCONTROLADORES.pdf incluido en el CD de documentación.

La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control).

La arquitectura Harvard (Figura. 2.16.) dispone de dos memorias independientes una, que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.

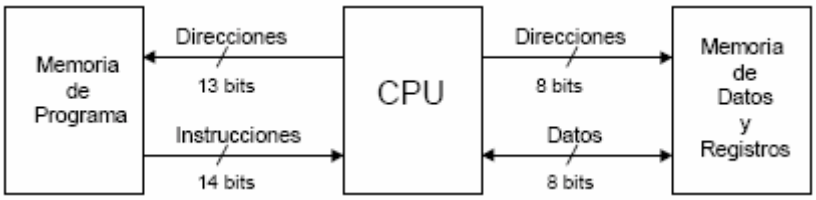


Figura. 2.16. Arquitectura Harvard.

RISC: Tanto la industria de los computadores comerciales como la de los microcontroladores están tomando la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo.

La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

2.2.2 La Norma RS-232.

Ante la gran variedad de equipos, sistemas y protocolos que existen surgió la necesidad de un acuerdo que permitiera a los equipos de varios fabricantes comunicarse entre si. La **EIA (Electronics Industry Association)** elaboró la

norma RS-232,[6] la cual define la interface mecánica, los pines, las señales y los protocolos que debe cumplir la comunicación serial. Todas las normas RS-232 cumplen con los siguientes niveles de voltaje:

- Un “1” lógico es un voltaje comprendido entre –5V y –15V en el transmisor y entre -3V y –25V en el receptor.
- Un “0” lógico es un voltaje comprendido entre +5V y +15V en el trasmisor y entre +3V y +25V en el receptor.

El envío de niveles lógicos (bits) a través de cables o líneas de transmisión necesita la conversión a voltajes apropiados. En los microcontroladores para representar un 0 lógico se trabaja con voltajes inferiores a 0.8V, y para un 1 lógico con voltajes mayores a 2.0V. En general cuando se trabaja con familias TTL se asume que un “0” lógico es igual a 0V y un “1” lógico es igual a 5V.

La importancia de conocer esta norma, radica en respetar los niveles de voltaje que maneja el puerto serial del ordenador, ya que son diferentes a los que utilizan los microcontroladores y los demás circuitos integrados. Por lo tanto se necesita de una interface que haga posible la conversión de los niveles de voltaje a los estándares manejados por los circuitos integrados TTL.

2.2.2.1 El Circuito MAX-232.

Este circuito soluciona los problemas de niveles de voltaje cuando se requiere enviar señales digitales sobre una línea RS-232. Este chip se utiliza en aquellas aplicaciones donde no se dispone de fuentes dobles de +12 y –12 voltios.

El MAX-232 (Figura. 2.17) necesita solamente una fuente de +5V para su operación, internamente tiene un elevador de voltaje que convierte el voltaje de

[6] SERIAL RS-232.pdf incluido en el CD de documentación.

0V a de doble polaridad de +12V y –12V. Cabe mencionar que existe una gran variedad de circuitos integrados que cumplen con esta función como lo son: MAX220, DS14C232, MAX233, LT1180A.

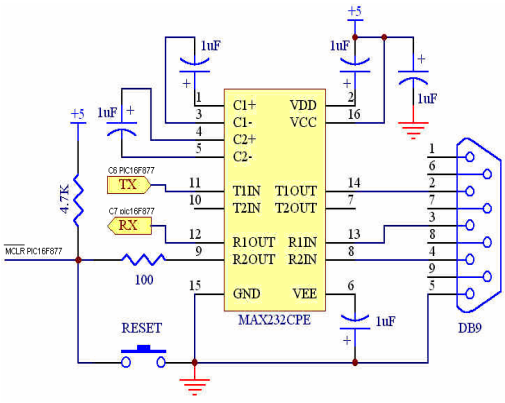


Figura. 2.17. Conexión del Max-232.

2.2.3 Buffer 74LS244.

Es un buffer tri-estado (Figura. 2.18.) tiene un puerto de entrada de 8 bits, trabaja como un paso de datos cuando los habilitadores están activados. El momento en que los habilitadores se desactivan el buffer trabaja en alta impedancia, impidiendo el paso de datos a través de este. Tiene 2 habilitadores los mismos que permiten el paso de datos en dos grupos de cuatro bits, siendo el primer habilitador el que controla los cuatro bits de salida menos significativos y el segundo de los cuatro restantes como se muestra en la Tabla. 2.1.

Tabla. 2.1. Tabla de funcionamiento del 74LS244.

ENTRADAS		SALIDA
$\overline{OE}$	An	Yn
L	L	L
L	H	H
H	X	Z

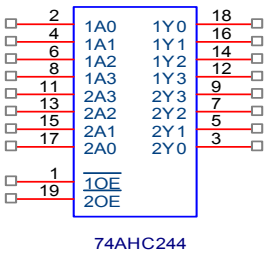


Figura. 2.18. Diagrama de conexión de los pines.

CAPÍTULO III

DISEÑO DE HARDWARE

3.1 DIAGRAMA DE BLOQUES DEL SISTEMA

El sistema de control realiza funciones de monitoreo de señales discretas. Cuenta con un circuito acoplador de tensiones de 24 VDC a 5 VDC. El sistema además debe manejar elementos finales de control de baja potencia relativa tipo ON/OFF. Para la programación del controlador y el manejo de la estación PN-2800 mediante PC el sistema de control propuesto cuenta con un puerto de comunicación de tipo serial.

La Figura. 3.1 ilustra la relación entre los bloques que conforman el sistema de control para la estación PN-2800.

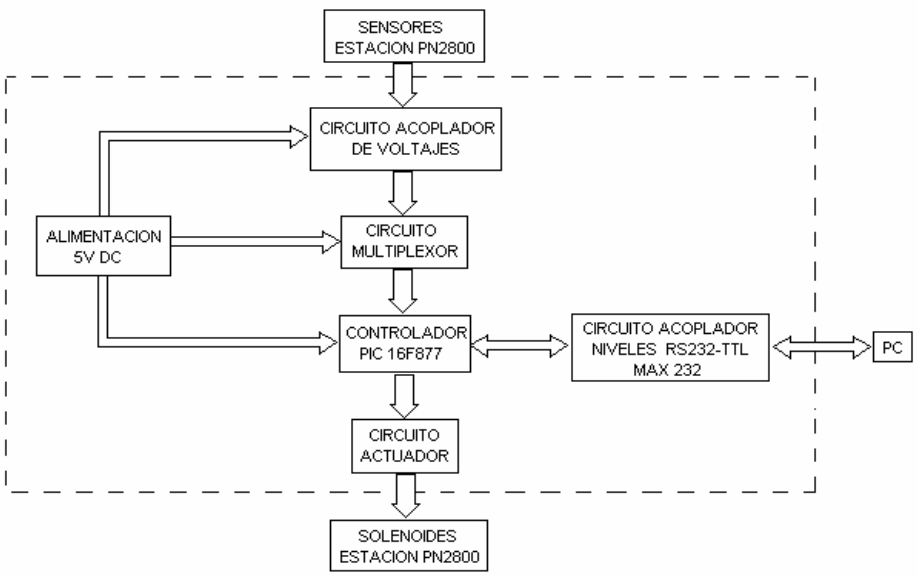


Figura. 3.1. Bloques del sistema.

3.2 INGENIERÍA DE DETALLE DEL SISTEMA COMPLETO

3.2.1 Acoplamiento de voltajes.

Para que el microcontrolador pueda monitorear el estado de los sensores de la estación, se requiere acoplar los niveles de voltaje de 24 V a 5V, esto mediante la conexión en serie de una resistencia con un diodo zener. Cuando la salida del sensor excita este circuito, la tensión existente en el cátodo del diodo zener es muy cercana a los 5VDC, así mismo cuando el sensor no proporciona tensión en el circuito, la tensión a través del zener es casi nula.

De esta manera el voltaje inverso a través del zener cumple con los valores requeridos de una señal de entrada TTL.

3.2.1.1 Criterios para el diseño.

La resistencia en serie Rs de la Figura. 3.3 es utilizada para mantener la corriente Is a través del zener dentro de su rango nominal de trabajo. Como muestra la Figura. 3.2, la magnitud de Is debe permanecer entre Izt *e* Izm. Para el diodo zener 1N4733A se tiene: Izt = 49mA *e* Izm = 178mA.

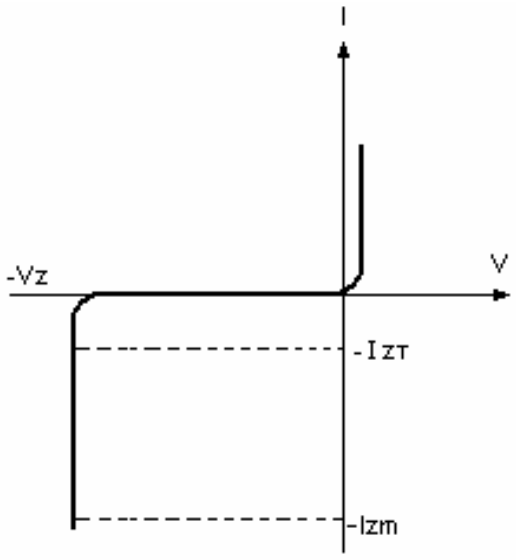


Figura. 3.2. Comportamiento Zener.

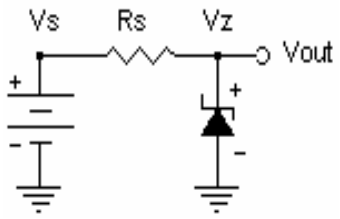


Figura. 3.3. Circuito acoplador básico.

$$I_S = \frac{V_S - V_Z}{R_S}$$

Donde:

Vs = voltaje que entrega el sensor = 24v

Vz = voltaje zener = 5.1v

$$Izm > Is > Izt$$

$$\therefore 178 > Is > 49 \ [mA]$$

$$Rs_{\min} = \frac{24 - 5.1}{178mA} \approx 106\Omega$$

$$Rs_{\max} = \frac{24 - 5.1}{49mA} \approx 386\Omega$$

$$\therefore$$

$$386 > Rs > 106 \ [\Omega]$$

Donde:

V_{out} es la señal de entrada al buffer.

3.2.1.1.1 Resistencia para Pull Down.

Cuando los sensores no están entregando energía no necesariamente ponen sus salidas a 0V, sus estados quedan en estado de alta impedancia. Una resistencia *(Pull Down) (RA)* con sus terminales conectados entre la salida del sensor y tierra tal como indica la Figura. 3.4 pone a 0V la diferencia de tensión a través del sensor cuando este se encuentra en estado de alta impedancia.

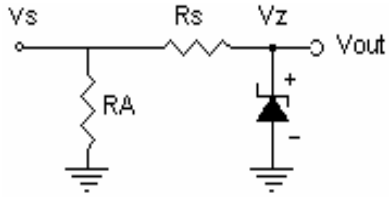


Figura. 3.4. Circuito acoplador con resistencia de Pull Down.

En la práctica para este proyecto, se determinó que la resistencia para Pull Down (RA) debe tener un valor inferior a 15kΩ. La Figura. 3.5 ilustra el circuito acoplador de tensiones para un total de 32 entradas discretas.

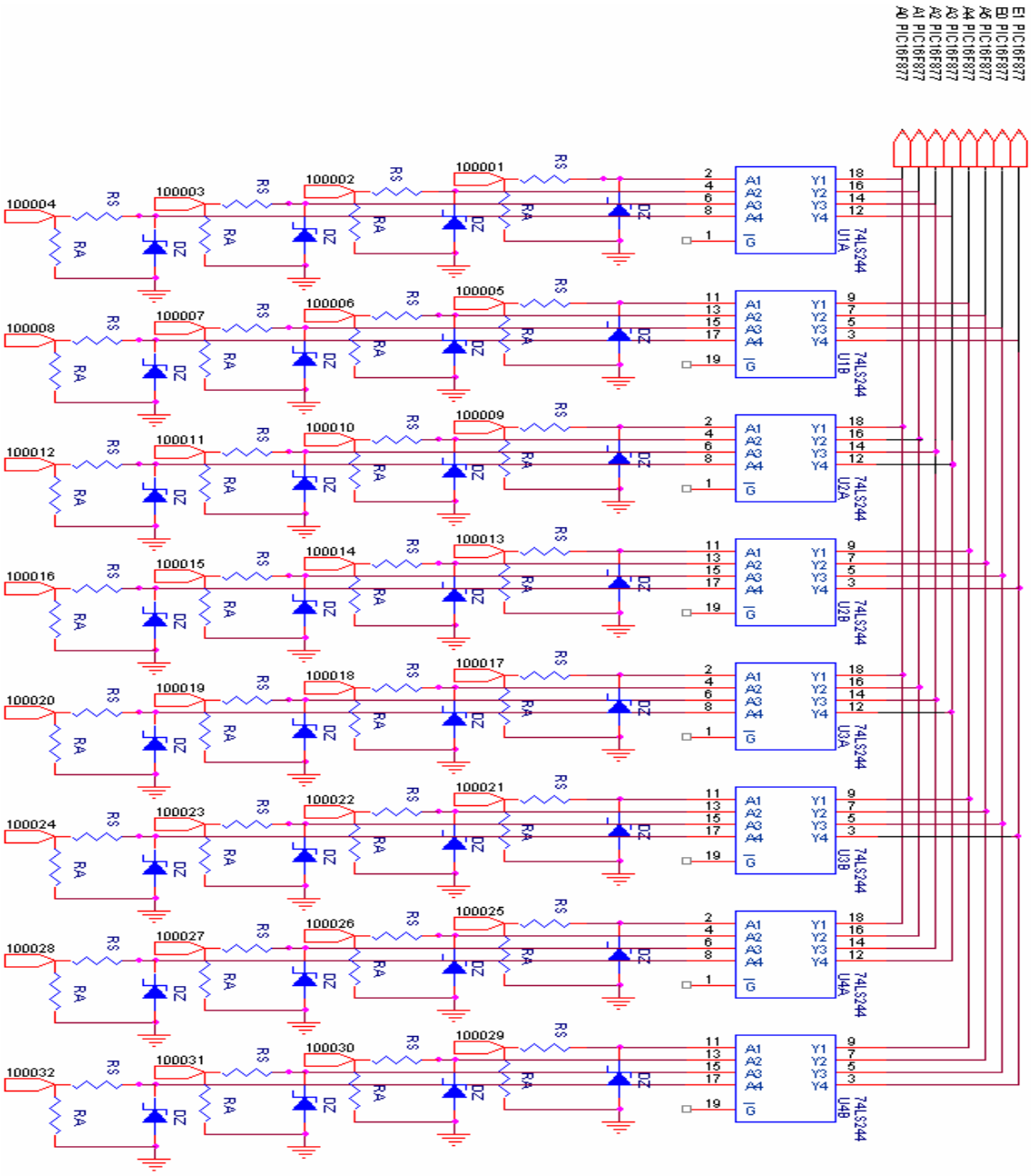


Figura. 3.5. Circuito acoplador de tensiones.

3.2.2 Circuito multiplexor.

Una vez que se ha realizado el acoplamiento de voltajes, es necesario proveer al microcontrolador de un circuito que permita leer hasta 32 sensores con únicamente un puerto de 8 entradas. Dicho circuito basa su funcionalidad en la técnica de multiplexación por división de tiempo, la cual es lograda cortocircuitando entre si las salidas de los 4 buffer SN74LS244 tal como muestra la Figura. 3.6.

En el momento de efectuar la lectura de los sensores, solo un buffer es habilitado, mientras los demás permanecen en estado de alta impedancia. Realizando esta operación en cada uno de los cuatro buffer, es posible leer los 26 sensores discretos de la estación PN-2800 en relativo corto intervalo de tiempo.

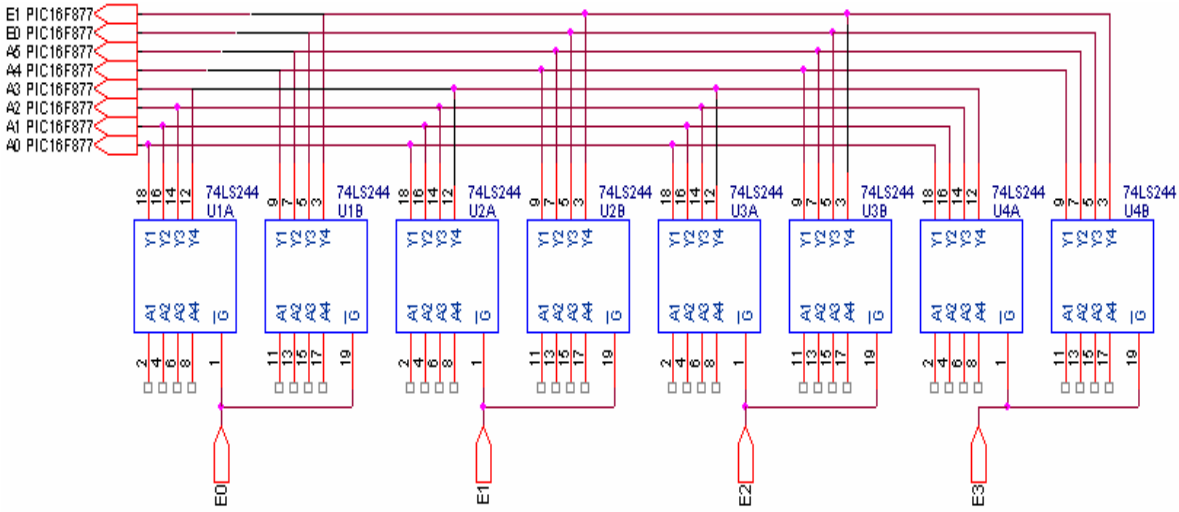


Figura. 3.6. Circuito multiplexor.

Donde:

A_0, A_1, A_3..... Van conectados a los respectivos cátodos de los 32 zener.
E_0, E_1, E_2, E_3. Son las salidas del circuito habilitador (7404).

3.2.2.1 Circuito habilitador

Este circuito puede poner en sus salidas únicamente los siguientes valores binarios (Tabla. 3.1).

Tabla. 3.1. Valores válidos para la habilitación.

E3	E2	E1	E1	Enable Buffer
1	1	1	0	1
1	1	0	1	2
1	0	1	1	3
0	1	1	1	4

A simple vista se podría pensar que el microcontrolador puede realizar esta tarea. Eso es verdad siempre y cuando no se utilice un bootloader para cargar los programas al microcontrolador. Mientras el computador está enviando serialmente el nuevo firmware al microcontrolador, este pone a nivel lógico de “0” todas sus I/0. Esto causaría la habilitación de todos los buffer a la vez. Para evitar esto, se dispone de una compuerta lógica 74LS04 entre las salidas del microcontrolador destinadas a la habilitación de los $\overline{OE_S}$ de los buffer (Figura. 3.7). Con esto, mientras el bootloader realiza la transferencia del firmware al PIC, todas las salidas de los cuatro buffer permanecen en alta impedancia.

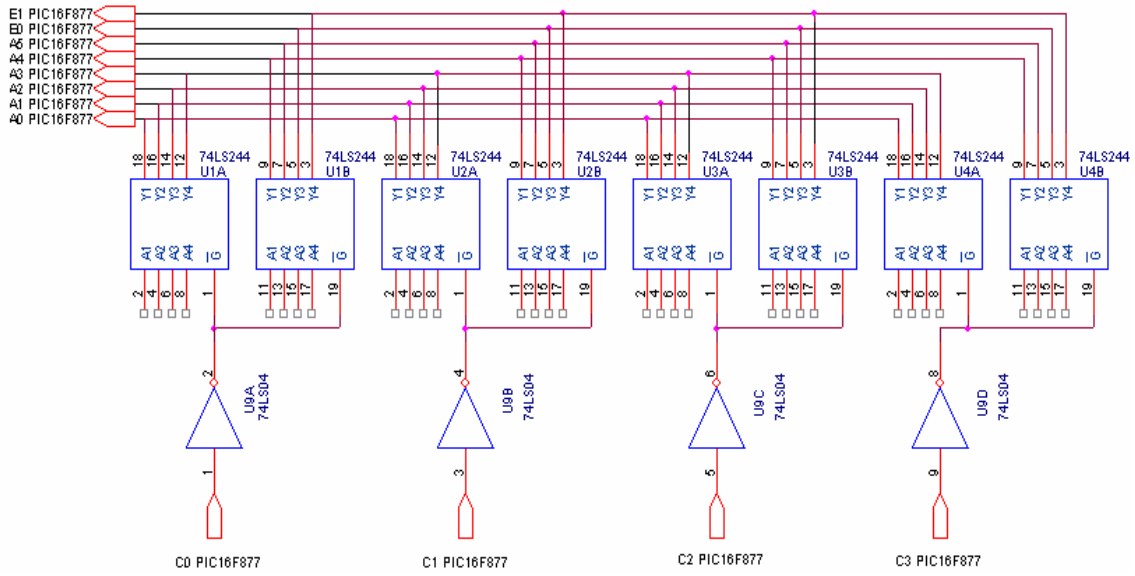


Figura. 3.7. Circuito multiplexor completo.

La Figura. 3.8 constituye el hardware completo para manejar las señales discretas de entrada.

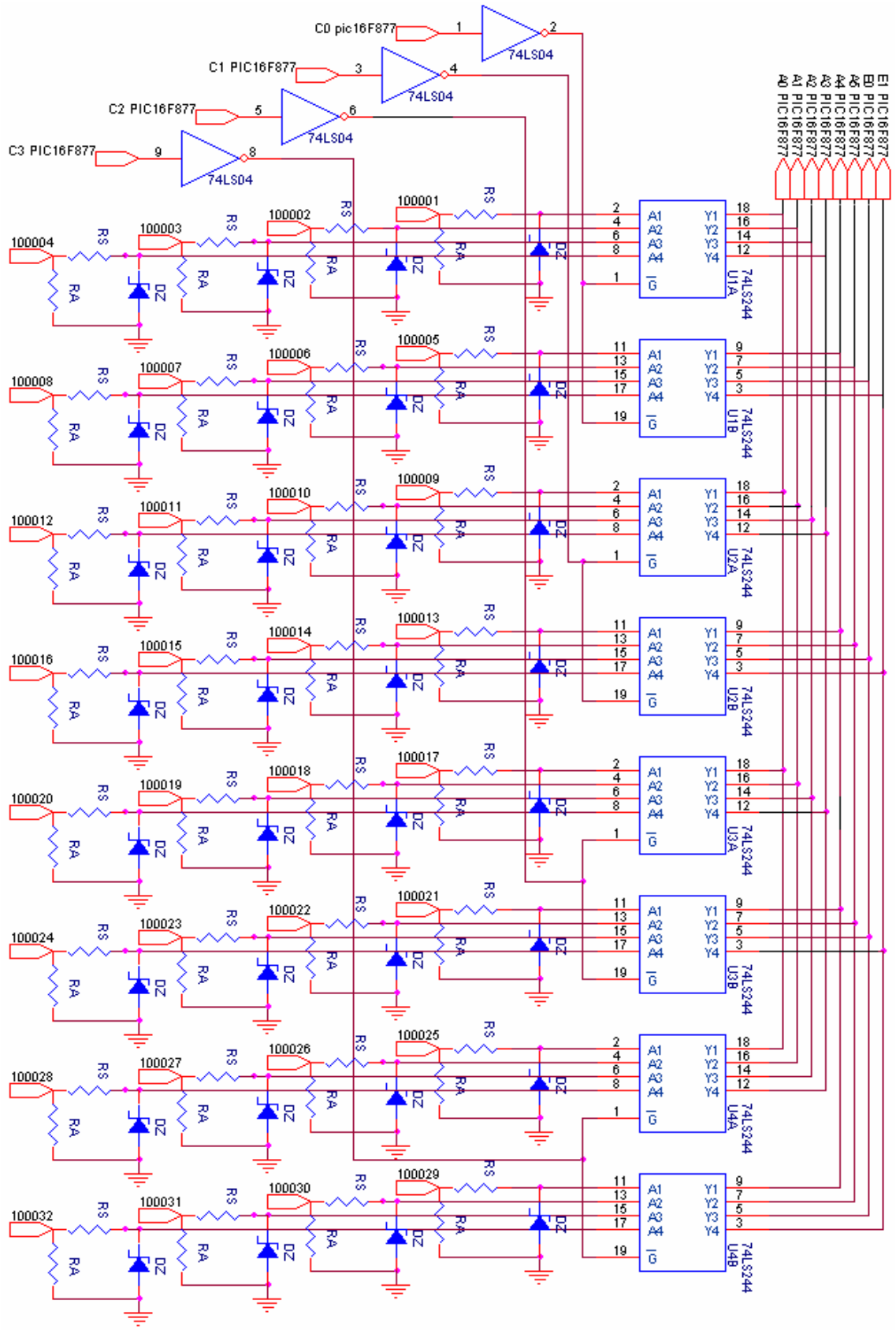


Figura. 3.8. Diagrama completo del Hardware para manejar las señales de entrada.

3.2.3 Circuito actuador.

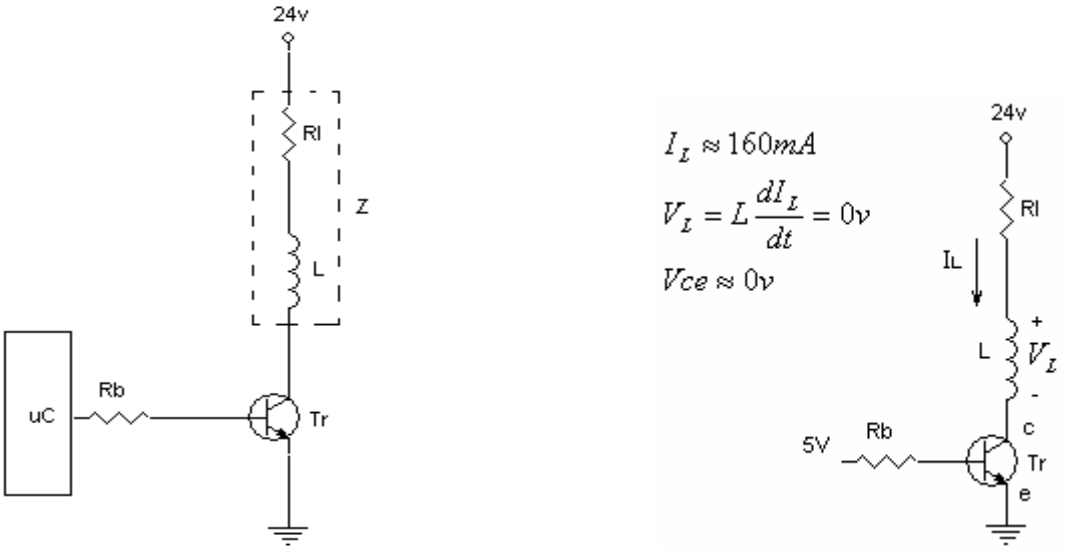
Las válvulas neumáticas de la estación requieren de una corriente de alrededor de 160mA para operar, la misma que puede ser manejada holgadamente por un transistor de uso general. El transistor 2N2222A posee un β ≈160, esto indica que se requiere de alrededor de 1mA a través de la base del transistor para activar una válvula.

$$I_B = 1mA = \frac{5 - 0.7}{R_B}$$

$$\therefore$$

$$R_B \approx 4k\Omega$$

Las impedancias de estas válvulas neumáticas tienen una relativa gran componente inductiva, la cual destruirá al transistor si no se usa la correspondiente protección. La Figura. 3.9. (a) representa un circuito con carga inductiva sin protección. La Figura. 3.9. (b) muestra el comportamiento del circuito sin protección cuando el transistor se encuentra en estado de saturación.



(a) **(b)**

Figura. 3.9. a) Circuito sin protección; b) Circuito con Tr en saturación.

La Figura. 3.10 muestra el comportamiento del circuito sin protección cuando el transistor ha pasado del estado de saturación al de corte. Se observa que la pendiente de la corriente en esa transición tiende a menos infinito, esto implica que en esa misma transición el voltaje colector-emisor tienda a infinito.

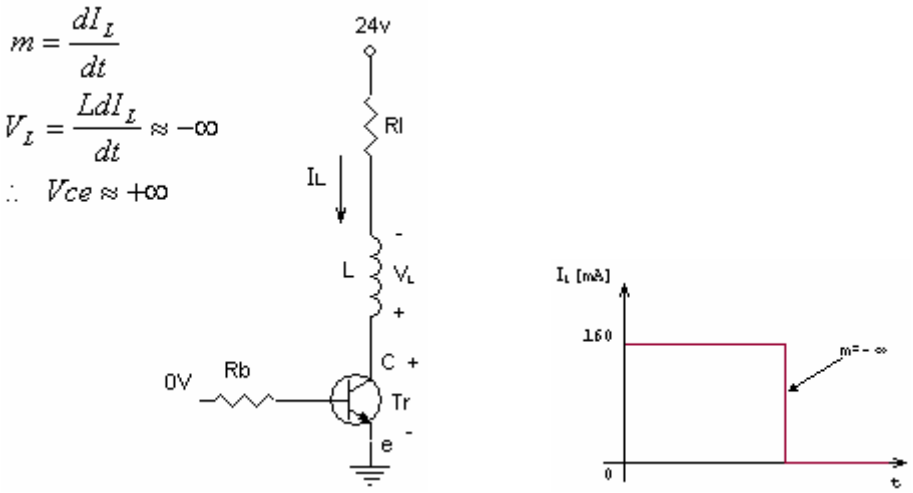
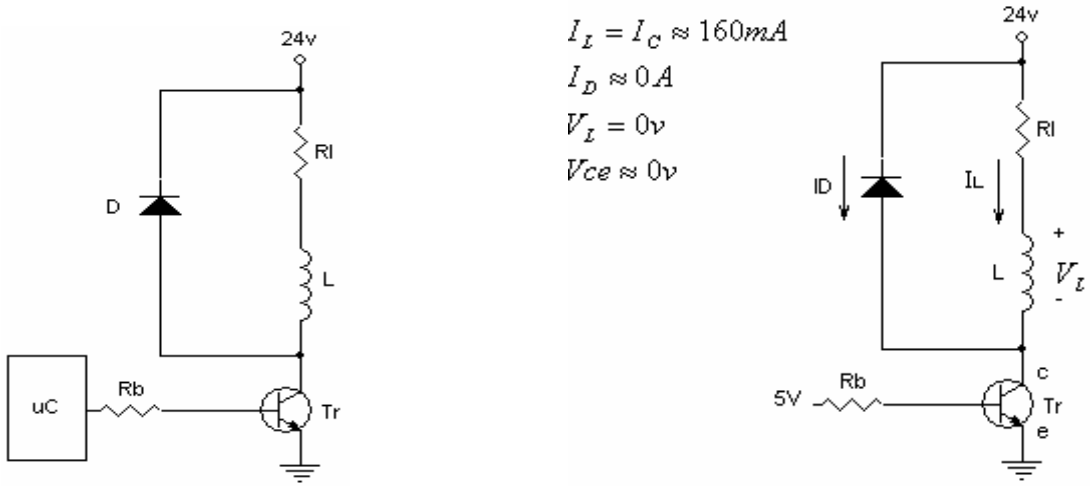


Figura. 3.10. Circuito sin protección en transición de saturación a corte.

Un diodo de propósito general actúa como elemento protector del transistor cuando este pasa del estado de saturación al estado de corte (Figura. 3.11. (a)). La idea es disipar la energía almacenada en el solenoide a través del diodo y no a través del transistor. La Figura. 3.11. (b) muestra el comportamiento del circuito con protección cuando el transistor se encuentra en estado de saturación.



(a) **(b)**

Figura. 3.11. a) Circuito con protección; b) Circuito con Tr en saturación.

La Figura. 3.12 muestra el comportamiento del circuito con protección cuando el transistor está pasando del estado de saturación al de corte. Aquí se nota la disminución de la pendiente de la corriente, esto asegura la desaparición del voltaje infinito en el colector-emisor al momento de la transición.

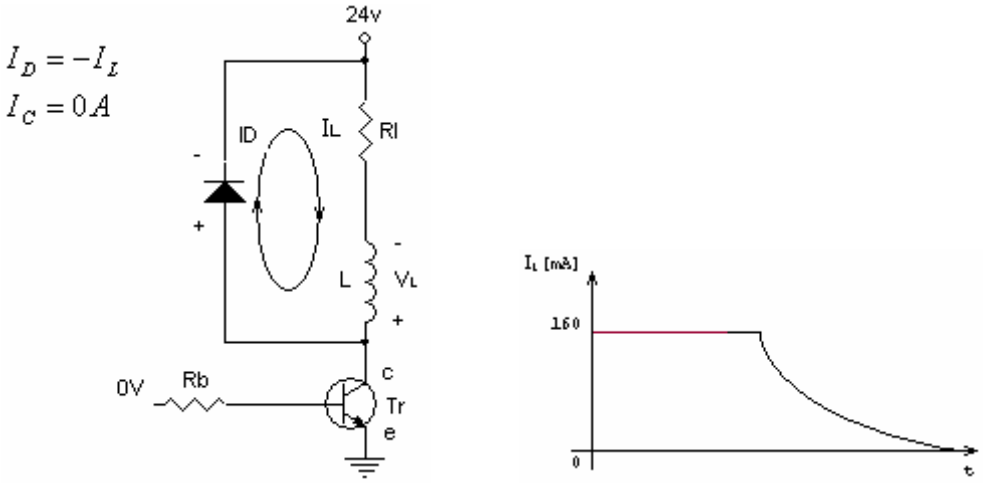


Figura. 3.12. Circuito con protección en transición de saturación a corte.

La Figura. 3.13 ilustra el circuito para manejar las electro-válvulas de la estación PN-2800.

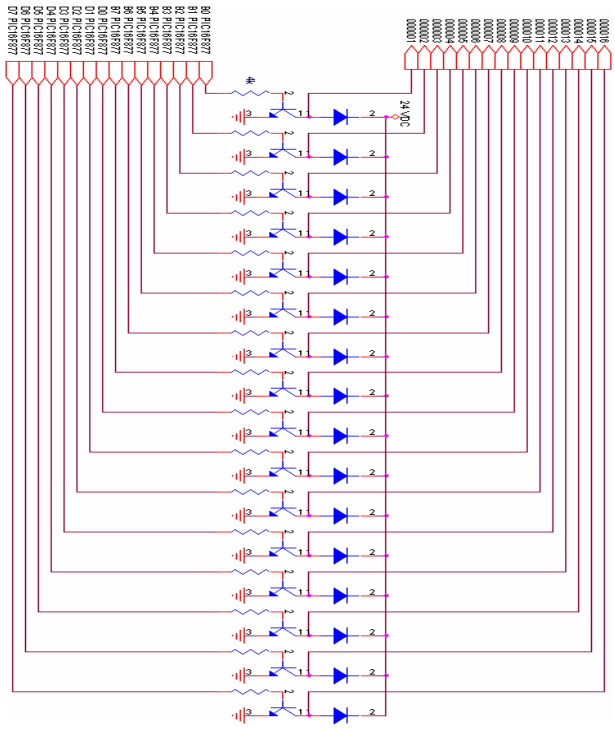


Figura. 3.13. Esquema del hardware para el manejo de electro-válvulas.

3.2.4 Controlador.

Como elemento controlador se ha utilizado un microcontrolador PIC16F877, cuyos puertos A, E y C han sido destinados a realizar la lectura de los sensores. El uso de los puertos se describe a continuación:

Port A y Port E → Lectura (8 bits) $E_1E_0A_5A_4A_3A_2A_1A_0$

$Donde:$
$A_0 = bit\ \ menos\ \ significativo$
$E_1 = bit\ \ mas\ \ significativo$

La estación PN-2800 posee 42 I/O: 26 sensores y 16 solenoides. Todos los sensores pueden ser leídos con la ayuda de la multiplexación en el tiempo por un solo puerto de 8 bits. Para realizar esta operación, se ha usado un puerto adicional de 4 bits *(selector de 1 de los 4 buffer. Tabla.3.2.).* Lo cual indica que para monitorear a los 26 sensores de la estación, es preciso contar con al menos 12 I/Os del elemento controlador: 8 para leer y 4 para seleccionar.

Port C → Selección (4 bits) $C_3C_2C_1C_0$

Tabla. 3.2. PortC → Selección (4 bits).

C3	C2	C1	C0	Enable Buffer
0	0	0	1	1
0	0	1	0	2
0	1	0	0	3
1	0	0	0	4

Para efectuar la conmutación de los 16 transistores, que conmutan los 16 solenoides, se dispone de una palabra de 16 bits conformada por los puertos B y D del microcontrolador.

$D_7D_6D_5D_4D_3D_2D_1D_0B_7B_6B_5B_4B_3B_2B_1B_0$
$Donde:$
$D_7D_6D_5D_4D_3D_2D_1D_0 = byte\ \ mas\ \ significativo$
$B_7B_6B_5B_4B_3B_2B_1B_0 = byte\ \ menos\ \ significativo$

3.2.5 Alimentación.

La estación PN-2800 proporciona una fuente de 24VDC. Para alimentar al controlador y demás circuitos TTL es necesario reducir el voltaje a 5VDC. Para ello se ha utilizado un circuito reductor de voltaje basado básicamente en un LM350 (Figura. 3.14).

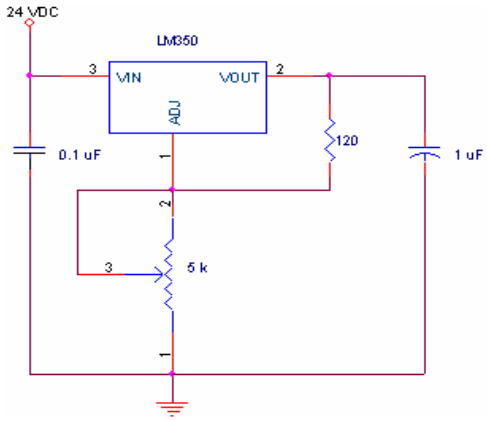


Figura. 3.14. Fuente DC ajustable.

3.2.6 Comunicación serial.

El microcontrolador PIC16F877 dotado con un puerto USART, posee la capacidad de comunicarse con el computador mediante un circuito basado en un Max-232 (Figura. 3.15). Este circuito acopla niveles de voltajes de interface RS-232 a niveles TTL. Mediante la interface RS-232 del controlador, se cubre los requerimientos físicos necesarios para ejecutar el Bootloader y realizar la monitorización y control de la estación PN-2800 mediante HMI.

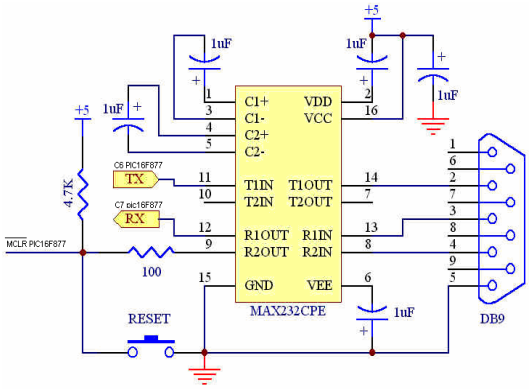


Figura. 3.15. Conexión Max-232.

3.2.7 Asignación de borneras.

La Figura. 3.16 muestra la distribución y asignación de borneras para I/Os y alimentación de energía.

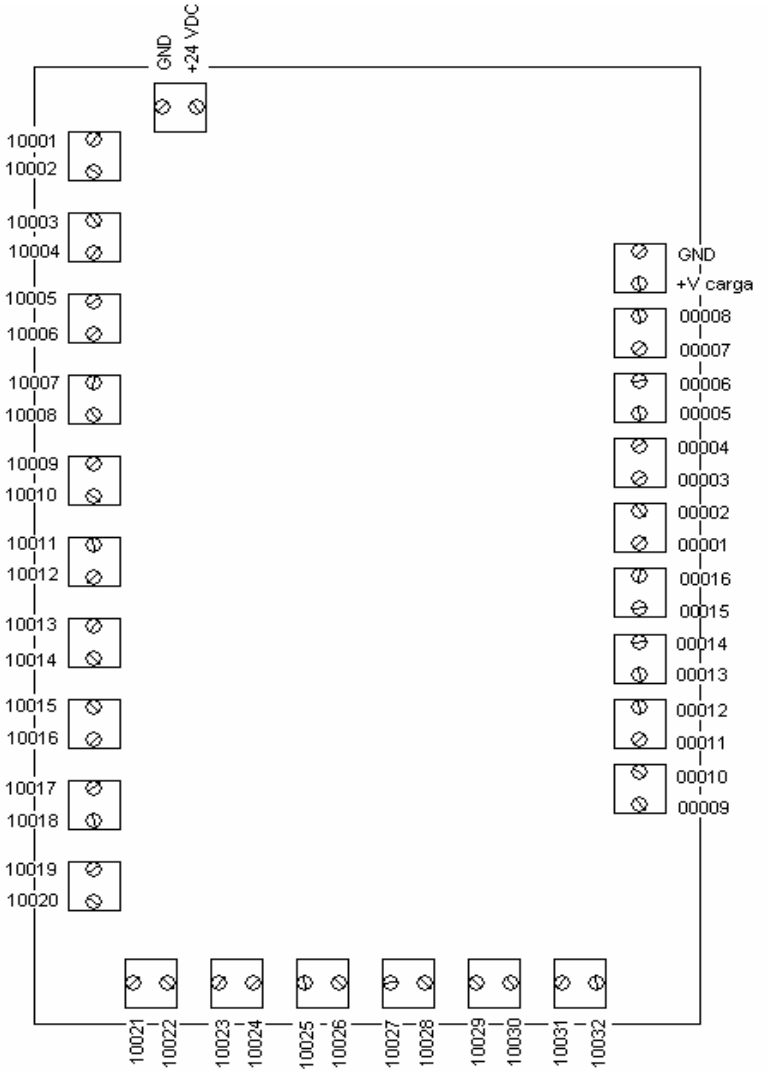


Figura. 3.16. Asignación de borneras.

CAPÍTULO IV

DISEÑO DE SOFTWARE

El sistema de control desarrollado tiene dos modos de funcionamiento:

1) Con HMI mediante PC y

2) Auto soportado

En el presente Capítulo se hace una descripción de los diferentes elementos de programación que permiten dotar de estas funcionalidades al prototipo implementado. Estos elementos son: Bootloader, Microcode Loader, protocolo Modbus y la interface hombre-maquina Lookout.

Mediante el Microcode Loader en la PC y un programa residente en el PIC16F877 denominado Bootloader, se puede descargar los programas de usuario desde la PC hacia el microcontrolador sin la necesidad de extraer el microcontrolador de su lugar de trabajo. Las descargas se realizan por medio del puerto serial de la PC y para ello, basta con embocar a Microcode Loader y ejecutar un reset en el microcontrolador. La Figura. 4.1 muestra la relación general entre los elementos de software de la PC y del microcontrolador que hacen posible esta tarea.

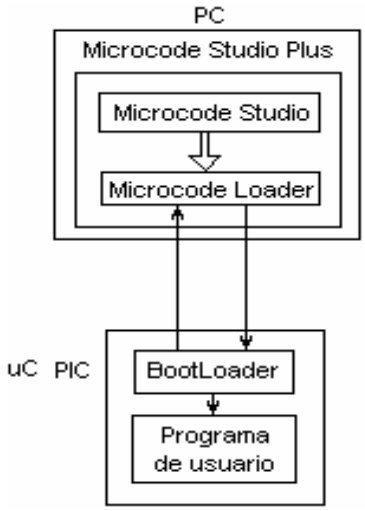


Figura. 4.1. Flujo de información entre Microcode Studio Plus y Bootloader.

Para la implementación del protocolo Modbus y el monitoreo de la estación PN-2800 por medio de Lookout, se ha desarrollado un programa de usuario denominado PLC_2800. Cuando el programa PLC_2800.HEX ha sido descargado hacia el microcontrolador, se permite al sistema implementado la comunicación con otros dispositivos bajo el protocolo de comunicación Modbus modo RTU. En esta sección se describe las estructuras de las tramas Modbus, así como las funciones y modos de detección de errores implementados para este proyecto.

Finalmente se presenta aspectos referentes al manejo del programa Lookout, el mismo que ha sido utilizado en la elaboración de las HMIs para este proyecto. Lookout es una poderosa herramienta creada por la firma National Instruments y es muy utilizada en el control y monitoreo de procesos industriales. Posee todos los elementos necesarios para la elaboración de HMIs. Lookout por medio del protocolo Modbus opera como master en redes formadas por PLCs. Más adelante, se describe los procedimientos a seguirse en la elaboración de HMIs así como la descripción de los objetos de Lookout utilizados.

4.1 SISTEMA DE CARGA DE PROGRAMAS BOOTLOADER[7]

El bootloader es un programa que ocupa una pequeña porción de la memoria flash del PIC. El objetivo del bootloader es que, una vez que esté grabado en el PIC, el usuario pueda emplearlo para reprogramar el PIC las veces que quiera usando sólo el puerto serial (y no el grabador PIC).

Con el fin de maximizar el espacio disponible para los programas descargados, es necesario hacer el bootloader tan pequeño como sea posible.

Como el bootloader precisa operar después del reset, significa que necesita utilizar el vector de reset. Por facilidad, es mejor si el bootloader se coloca enteramente al principio o al final de la memoria de programa. En los PICs de alcance medio, el vector de interrupción está situado en la dirección 0x4. Si el bootloader fuera a ser colocado enteramente al principio de memoria, entonces se tendría que volver a dirigir el vector de interrupción.

Debido a que esto agregaría complejidad al código y estado latente al vector, los diseñadores han decidido colocar el bootloader en el final de la memoria de programa.

Ya que el vector de reset *(4 primeras localidades de memoria)* será utilizado para saltar hacia el código del bootloader, es necesario que el vector de reset de programa descargado sea puesto en otra parte.

La Figura. 4.2 representa el mapa de memoria (uControlador) de un programa con y sin el bootloader instalado.

[7] http://www.HI-TECH Software Design Tip Give Your Project the Boot!.htm

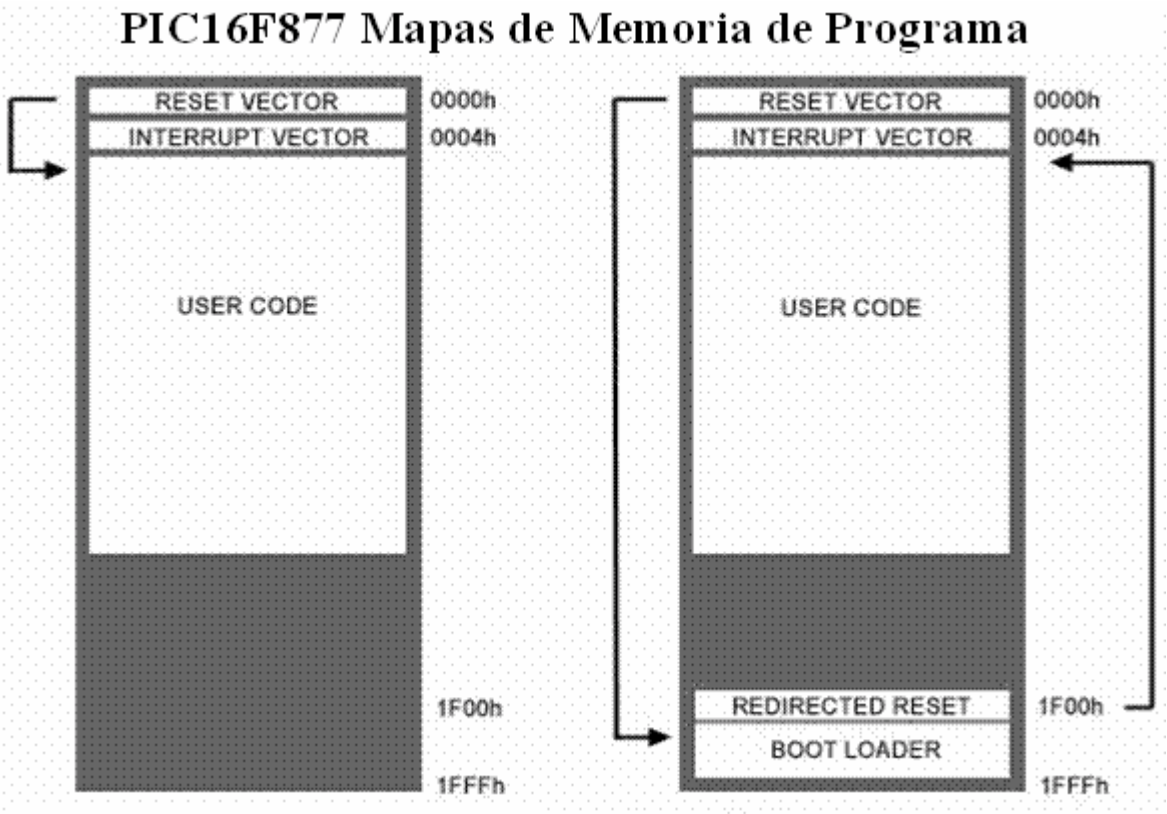


Figura. 4.2. a) Típico mapa de memoria; b) Mapa de memoria con bootloader instalado.

Según las indicaciones de la Figura. 4.2. (a) el vector de reset contiene típicamente instrucciones de saltar al programa principal. Puesto que el bootloader necesita del vector de reset, se debe mover el vector de reset del programa descargado a otra parte. La Figura. 4.2. (b) demuestra cómo el vector de reset ahora apunta al código del bootloader y que el vector de reset del programa descargado será colocado a una dirección antes del bootloader.

4.1.1 Modo de Trabajo.

Instantes posteriores al reset, el bootloader tomará el control y esperará a que el usuario envíe un archivo con extensión hexadecimal. Si no se envía ningún archivo válido, el bootloader asumirá que no se requiere ninguna actualización y saltará al vector redireccionado de reset de programa, con lo cual se ejecutará el programa previamente descargado (Figura. 4.3).

Si no se ha descargado ningún programa, el vector redireccionado de reset simplemente efectuará un salto al principio del bootloader y el proceso entero comenzará otra vez. Si un archivo válido se envía al bootloader dentro del período de espera, comenzará a interpretar los datos y a escribirlos en la memoria del programa. El bootloader buscará direcciones mayores a 0x4 (vector de reset) y menores a 0x1F00 (inicio del programa bootloader) para escribir el programa descargado.

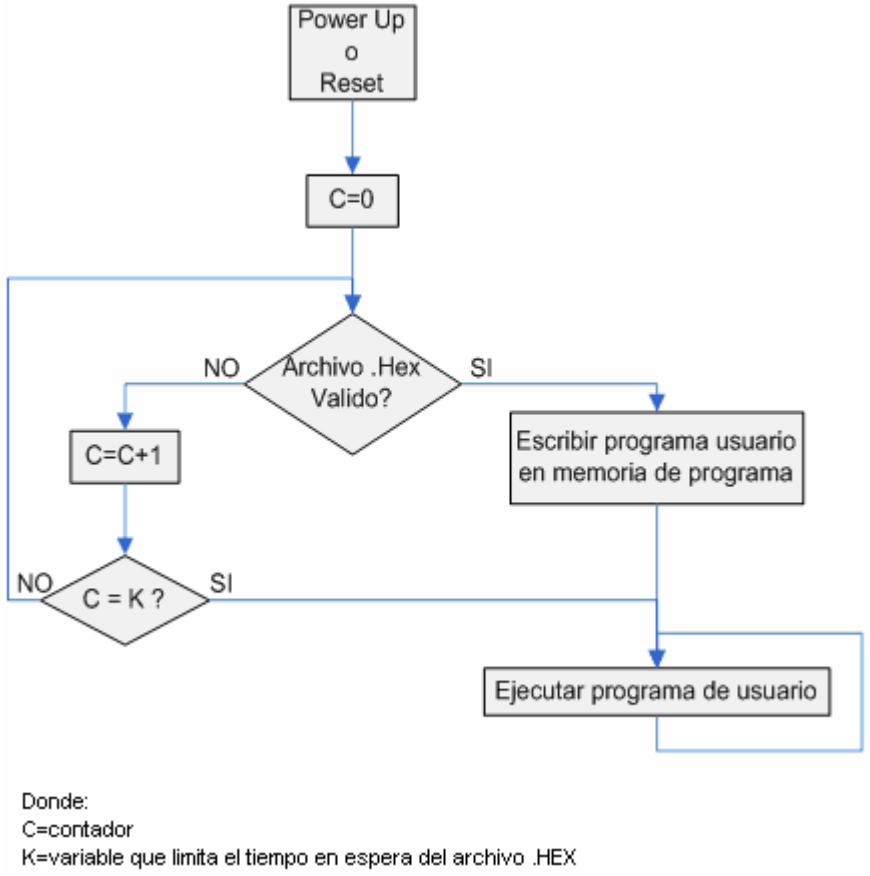


Figura. 4.3. Diagrama de flujo que describe la operación básica del bootloader.

Existen varios programas Loader, así como Bootloader, pero se debe tener en cuenta que un loader será capaz de comunicarse y descargar programas al microcontrolador solo si trabaja con su apropiado bootloader.

Para este proyecto fue utilizado como Loader el software *MicroCode Loader*, cuyas características se detallan a continuación.

4.2 DESCRIPCIÓN DE MICROCODE LOADER.[8]

El MicroCode Loader es un software que reside en la computadora. Juntos, el Microcode Loader y el programa residente en el microcontrolador *(Bootloader)* permiten a un usuario programar, verificar y leer el programa así como los datos de la memoria EEPROM.

Las series PIC 16F87x y 18Fxxx (x) de microcontroladores tienen la capacidad de escribir en su propia memoria del programa, sin la necesidad de un programador de hardware. Un pedazo pequeño de software (un bootloader) reside en el microcontrolador, el cual permite que el código de usuario y los datos de EEPROM sean transmitidos vía un cable serial y escritos en el dispositivo.

Antes de usar El MicroCode Loader, es necesario asegurar que el software del bootloader se haya programado en el microcontrolador. Después de lo cual se puede comenzar a programar el microcontrolador bajo una conexión serial.

Así mismo es necesario cerciorarse de que el archivo del bootloader *.hex concuerde con la velocidad de reloj del microcontrolador a usarse. Por ejemplo, el archivo 16F877_04.hex se debe utilizar para programar un dispositivo que esté funcionando a 4MHz. Si no se hace esto, el MicroCode Loader no podrá comunicarse con el microcontrolador. No es necesario fijar la velocidad de transmisión del MicroCode Loader, pues esto será detectado automáticamente.

Cuando la energía es aplicada al microcontrolador (o este es reseteado), el bootloader chequea primero si el MicroCode Loader tiene algo para realizar (por ejemplo, programar un código en el microcontrolador). Si esto sucede, el bootloader da el control al MicroCode Loader hasta que la operación esté culminada. Sin embargo, si el bootloader no recibe ninguna instrucción durante unos pocos cientos de milisegundos posteriores al reseteo del dispositivo, el

[8] http://www. MicroCode Studio - UK - MicroCode Loader Overview.htm

bootloader cesará y el código de usuario escrito previamente en el dispositivo comenzará a ejecutarse.

El software del bootloader reside en las 256 palabras superiores de la memoria del programa (336 palabras para los dispositivos 18Fxxx (x)), con el resto del espacio disponible para el código de usuario. Toda la memoria EEPROM, así como los registros del microcontrolador están disponibles para el programa de usuario. Cabe destacar que solamente el espacio de código de programa y el espacio de los datos de EEPROM se pueden programar, verificar y leer por MicroCode Loader. Los bits de configuración del microcontrolador no están disponibles en el proceso del Loader, estos deben por lo tanto ser fijados cuando el software del bootloader se programa en el microcontrolador.

4.2.1 Requisitos de hardware. [9]

El MicroCode Loader se comunica con el microcontrolador usando el hardware Universal-Síncrono-Asíncrono-Receptor-Transmisor (USART). La familia de PIC 16F87x de microcontroladores tiene el pin de recepción RX situado en PORTC.7 y el pin de transmisión TX situados en PORTC.6. Por lo tanto, para utilizar el Loader es necesario disponer de un circuito que soporte comunicación serial RS-232.

4.2.2 Requisitos de programación.

Debido a que el bootloader reside en la parte superior de la memoria de programa, es necesaria una manera de saltar al comienzo del código del bootloader cuando la energía es inicialmente aplicada, o cuando es reseteado el microcontrolador. Para hacer esto, se utiliza las primeras cuatro palabras del programa (llamadas el vector de reset) para ejecutar un salto al código del bootloader.

[9] http://www. MicroCode Studio - UK - MicroCode Loader Overview.htm

Cuando se programa el microcontrolador usando MicroCode Loader, las instrucciones de programa que ocupan las primeras cuatro localizaciones son vueltas a poner automáticamente por el bootloader. Si el bootloader no volviera a poner estas instrucciones, el “salto importante al bootloader” sería sobrescrito y el software del bootloader no se ejecutaría cuando el microcontrolador fuera nuevamente reseteado.

Por tanto, para lograr esto; es necesario poner al inicio de la página de programación la siguiente expresión:

```
DEFINE ONINT_USED 1
```

4.3 COMUNICACIÓN MODBUS.[10]

El protocolo Modbus proporciona el estándar interno que los controladores Modicon usan para el análisis de los mensajes. Durante la comunicación sobre una red Modbus, el protocolo determina cómo cada controlador conocerá su dirección de dispositivo, reconocerá un mensaje direccionado a él, determinará el tipo de acción a tomar y extraerá cualquier dato u otra información contenida en el mensaje. Si se requiere una repuesta, el controlador construirá el mensaje respuesta y lo enviará utilizando el protocolo Modbus.

Los controladores se comunican usando una técnica maestro – esclavo, en la cual sólo un dispositivo (el maestro) puede iniciar transacciones (llamadas ‘peticiones’). Los otros dispositivos (los esclavos) responden suministrando al maestro el dato solicitado, o realizando la acción solicitada en la petición.

Entre los dispositivos maestros típicos se incluyen los procesadores centrales y los paneles de programación. Esclavos típicos son los PLC’s.

[10] PI_MBUS_300.pdf incluido en el CD de documentación.

El maestro puede direccionar esclavos individualmente o puede generar un mensaje en modo difusión a todos los esclavos. Los esclavos devuelven un mensaje (llamado 'respuesta') a las peticiones que les son direccionadas individualmente. No se devuelven respuestas a peticiones en modo difusión enviadas desde el maestro.

4.3.1 Contenidos de las tramas Petición-Respuesta.

Tabla. 4.1. Petición Maestro.

Dirección Esclavo	Función	Datos	Comprobación de Error

Tabla. 4.2. Respuesta Esclavo.

Dirección Esclavo	Función	Datos	Comprobación de Error

4.3.1.1 Dirección Esclavo.

Contiene la dirección asignada a un esclavo, la dirección válida asignada a un esclavo individual debe estar entre 1 y 247. La dirección 0 es utilizada para dirección de broadcast.

4.3.1.2 Función.

Este código es el mismo tanto para la petición como para la respuesta, en el primer caso el maestro indica al esclavo direccionado la acción a realizar, en el segundo caso el esclavo direccionado indica al maestro cual acción ha sido ejecutada.

4.3.1.3 Datos.

Los bytes de datos en la petición, contienen información adicional que el dispositivo esclavo necesita para ejecutar la función encomendada. De la misma manera en la respuesta, el Esclavo en el campo de datos envía información

adicional que el Maestro requiere para interpretar las acciones realizadas por el esclavo.

4.3.1.4 Comprobación de Error.

El campo de comprobación de error proporciona un código realizado en *función* de los campos: Dirección, Función y Datos, con el cual el Esclavo o el Maestro validarán la integridad del contenido del mensaje recibido.

4.3.1.5 Modos de Transmisión Serie.

Los controladores sobre redes Standard Modbus pueden ser configurados para comunicar utilizando cualquiera de los dos modos de transmisión: ASCII o RTU. Los usuarios seleccionan el modo deseado, junto con los parámetros de comunicación del puerto serie (velocidad, paridad, etc.), durante la configuración de cada controlador. El modo y los parámetros serie *deben ser los mismos para todos los dispositivos conectados a una red Modbus*.

La selección del modo ASCII o RTU tiene que ver únicamente con redes Modbus Standard.

Los modos ASCII y RTU definen los bits contenidos en los campos del mensaje transmitido en forma serie en la red, así cómo también la forma en que debe ser empaquetada y decodificada la información en los campos del mensaje.

4.3.2 Métodos de comprobación de error.

Las redes series Standard Modbus utilizan dos tipos de comprobación de error. La comprobación de paridad (par o impar) puede ser aplicada opcionalmente a cada carácter. La comprobación de la trama [LRC *(ASCII)* o CRC *(RTU)*] es aplicada al mensaje completo. Ambas comprobaciones, de carácter y de trama de mensaje son generadas en el dispositivo transmisor y aplicadas a los contenidos

del mensaje antes de la transmisión. El dispositivo receptor comprueba cada carácter y la trama del mensaje completo durante la recepción.

4.3.3 Trama RTU.

En modo RTU los mensajes comienzan y terminan con un intervalo silencioso de al menos 3.5 tiempos de carácter, mostrado como T (Tabla. 4.3).

Tabla. 4.3. Trama RTU.

Arranque	Dirección	Función	Datos	CRC	Final
T	8 bits	8 bits	Nx8 bits	16 bits	T

4.3.3.1 Respuesta de excepción.

En el caso de ocurrir algún tipo de error, el esclavo devuelve el código de función original pero con su bit más significativo puesto a valor de 1. Por Ejemplo: Si el maestro quiere saber el estado de 16 salidas binarias de un determinado esclavo, pero este solo posee 4, se producirá una respuesta de excepción. El maestro envía en el campo de función el valor binario 00000001*(leer coils),* pero como se genera una respuesta de excepción, el esclavo en el campo de función devuelve al maestro el valor binario de 10000001 en lugar de 00000001.

4.3.3.2 Secuencia de transmisión RTU.

El formato para cada byte en modo RTU es: (Tabla. 4.4 y Tabla. 4.5)

Tabla. 4.4. Formato con paridad.

Start	1	2	3	4	5	6	7	8	Par	Stop

Tabla. 4.5. Formato sin paridad.

Start	1	2	3	4	5	6	7	8	Stop	Stop

4.3.3.3 Método de chequeo de error CRC.

La información que entrega este campo de comprobación de error en modo RTU está constituida por 16 bits, el valor de esta comprobación de error es el resultado de un cálculo Comprobación Cíclica Redundante (CRC) realizado sobre el contenido de los campos esclavo, función y datos. El campo CRC es añadido al mensaje como último campo del mensaje. La forma de hacerlo es, enviar primero el byte de orden menos significativo y luego el byte de orden más significativo del CRC.

El valor CRC es calculado por el dispositivo emisor, que añade el CRC al mensaje. El dispositivo receptor calcula el CRC durante la recepción del mensaje y compara el valor calculado con el valor recibido en el campo CRC. Si los dos valores no son iguales, resulta un error.

4.3.3.4 Cálculo de CRC.

Para calcular el valor CRC Modbus se precarga un registro de 16 bits, todos ellos a 1. Luego comienza un proceso que toma los sucesivos bytes de los campos y los opera con el contenido del registro y actualiza éste con el resultado obtenido. Sólo los 8 bits de datos de cada carácter son utilizados para generar el CRC. Los bits de arranque, paro y paridad, no se tienen en cuenta para el cálculo del CRC.

Durante la generación del CRC, se efectúa una operación booleana XOR a cada carácter de 8 bits con el contenido del registro. Entonces al resultado se le aplica un desplazamiento de 1 bit en la dirección de bit menos significativo (LSB), rellenando la posición del bit más significativo (MSB) con un cero. El LSB es extraído y examinado. Si el LSB extraído fuese un 1, se realiza un XOR entre el

registro y un valor binario fijo preestablecido (1010 00000000 0001). Si el LSB fuese un 0, no se efectúa la operación XOR.

Este proceso es repetido hasta haber cumplido 8 desplazamientos. Después del último desplazamiento (el octavo), el próximo byte es operado XOR con el valor actual del registro y el proceso se repite con ocho desplazamientos más, después de que todos los bytes del mensaje han sido procesados, el resultado es el valor del CRC.

4.3.3.5 Ejemplo para generación de CRC.

1) Cargar un registro de 16 bits denominado registro CRC, con FFFF (todos 1).

2) XOR del primer byte del mensaje *(esclavo)* con el byte de orden bajo del registro CRC, colocar el resultado en el registro CRC.

3) Extraer *(guardar)* el valor del bit menos significativo del CRC.

4) Desplazar el registro CRC un bit a la derecha (LSB), rellenando con un cero el MSB y examinar el bit LSB extraído en el paso 3.

5) (Si el LSB es 0): Repetir el paso 3. (Si el LSB es 1): Hacer XOR entre el registro CRC y el valor hexadecimal A001.

6) Repetir los pasos 3, 4 y 5 hasta que se hayan efectuado 8 desplazamientos. Una vez hecho esto, se habrá procesado un byte completo.

7) repetir los paso 3, 4, 5 y 6 con los Bytes de los campos Función y Datos.

4.3.4 Código de funciones Modbus.

4.3.4.1 Códigos de función soportados por los controladores.

El listado de abajo muestra los códigos soportados por algunos controladores Modicon. Los códigos están en decimal.

01 Leer Estados de Bobinas
02 Leer Estados de Entradas
03 Leer Registros Mantenidos
04 Leer Registros de Entradas
05 Forzar una única Bobina
06 Preestablecer un único Registro
07 Leer Status de Excepción
08 Diagnósticos
09 Programar 484
10 Selección 484
11 Buscar Contador de Eventos de Comunic.
12 Buscar Anotac de Eventos de Comunic.
13 Programar Controlador
14 Selección Controlador
15 Forzar Múltiples Bobinas
16 Preestablecer Múltiples Registros
17 Reportar Identificación de Esclavo
18 Programar 884/M84
19 Resetear Enlace de Comunicaciones
20 Leer Referencia General
21 Escribir Referencia General
22 Escribir con máscara en Registros 4X
23 Leer/Escribir Registros 4X
24 Leer Cola FIFO

Para la monitorización y control de la estación PN-2800 se requiere únicamente de las funciones 01, 02, 05 y 15, así como de algunas respuestas de excepción. Es por ello que a continuación se da una explicación más detallada de las mismas.

4.3.4.2 Leer bobinas (Función 01).

Lee el estado on/off de salidas discretas (0X referencia) del dispositivo esclavo.

Pregunta:

El mensaje de pregunta especifica la bobina inicial a ser leída y la cantidad de bobinas que deben ser leídas a partir de esta.

Si se quiere leer el estado de las bobinas 1 *a* 16, estas son direccionadas como 0 *a* 15. Aquí está un ejemplo de pregunta para leer las bobinas 20 a 56 del esclavo No 17:

Field Name	(Hex)
Slave Address	11
Function	01
Starting Address Hi	00
Starting Address Lo	13
No. of Points Hi	00
No. of Points Lo	25
Error Check (LRC or CRC)	——

Respuesta:

El estado de las bobinas está dado como una bobina por bit en el campo de datos. El estado es indicado como: 1 = ON; 0 = OFF.

El LSB del primer byte de datos *(*CD_{hex}*)* contiene el estado de la bobina 20. El estado de las subsiguientes bobinas sigue hacia la dirección más significativa del byte. El mismo orden es seguido para las restantes bobinas en los subsiguientes bytes.

Si la cantidad de bobinas a ser leídas no es un múltiplo de 8, los bits sobrantes en el último byte de datos son puestos a 0. El *Byte count* especifica la cantidad de bytes de datos. Aquí está un ejemplo de respuesta a la pregunta del ejemplo anterior:

Field Name	(Hex)
Slave Address	11
Function	01
Byte Count	05
Data (Coils 27–20)	CD
Data (Coils 35–28)	6B
Data (Coils 43–36)	B2
Data (Coils 51–44)	0E
Data (Coils 56–52)	1B
Error Check (LRC or CRC)	—

4.3.4.3 Leer estado de entradas (Función 02).

Lee el estado on/off de entradas discretas (1X referencia) del dispositivo esclavo.

Pregunta:

El mensaje de pregunta especifica la entrada inicial a ser leída y la cantidad de entradas que deben ser leídas a partir de esta.

Si el maestro desea leer el estado de las entradas 1 *a* 16, estas son direccionadas como 0 *a* 15. Aquí está un ejemplo de pregunta para leer el estado de las estradas 10197 *a* 10218 del dispositivo esclavo No 17.

Field Name	(Hex)
Slave Address	11

Function	02
Starting Address Hi	00
Starting Address Lo	C4
No. of Points Hi	00
No. of Points Lo	16
Error Check (LRC or CRC)	—

Respuesta:

El estado de las entradas está asignado a una bobina por bit en el campo de datos. El estado es indicado como: 1 = ON; 0 = OFF. El LSB del primer byte de datos *(C4)* contiene el estado de la entrada 10197. El estado de las subsiguientes entradas sigue hacia la dirección más significativa del byte. El mismo orden es seguido para las restantes entradas en los subsiguientes bytes.

Si la cantidad de entradas a ser leídas no es un múltiplo de 8, los bits sobrantes en el último byte de datos son puestos a 0.

El *Byte count* especifica la cantidad de bytes de datos. Aquí está un ejemplo de respuesta a la pregunta del ejemplo anterior:

Field Name	(Hex)
Slave Address	11
Function	02
Byte Count	03
Data (Inputs 10204–10197)	AC
Data (Inputs 10212–10205)	DB
Data (Inputs 10218–10213)	35
Error Check (LRC or CRC)	—

4.3.4.4 Forzar estado de individual bobina (Función 05).

Forza el estado de una individual bobina (0X referencia) al estado de ON/OFF.

Pregunta:

El mensaje de pregunta especifica la bobina a ser forzada. Bobina 1 es direccionada como 0, bobina 8 es direccionada como 7, etc. El estado ON/OFF al que se debe poner la bobina es especificado mediante una constante en el campo de datos en la pregunta. Un valor de FF 00 hex refiere a que la bobina debe ser energizada *(ON),* un valor 00 00 hex refiere a que la bobina debe ser desactivada *(OFF).*

Otros valores distintos a estos son ilegales y no tendrán efecto sobre el estado de la bobina. Aquí está un ejemplo de petición para forzar la bobina 173 al estado ON en el dispositivo esclavo No 17:

Field Name	(Hex)
Slave Address	11
Function	05
Coil Address Hi	00
Coil Address Lo	AC
Force Data Hi	FF
Force Data Lo	00
Error Check (LRC or CRC)	—

Respuesta:

Una respuesta normal, es una replica exacta de la pregunta; la misma que es retornada después de que la bobina ha sido forzada al estado solicitado por el dispositivo maestro. Aquí está un ejemplo de respuesta a la pregunta en el ejemplo anterior.

Field Name	(Hex)
Slave Address	11
Function	05
Coil Address Hi	00

Coil Address Lo	AC
Force Data Hi	FF
Force Data Lo	00
Error Check (LRC or CRC)	—

4.3.4.5 Forzar múltiples bobinas (Función 15 ($0F_{hex}$)).

Forza el estado de dos o más bobinas *(0X referencia)* a estados de ON/OFF.

Pregunta:

Para forzar el estado de las bobinas, El mensaje de pregunta especifica primeramente en el campo de datos la bobina que servirá como referencia. Bobina 1 será direccionada como 0, bobina 7 será direccionada como 6, etc.

Los estados ON/OFF a los que deben ser puestas las bobinas son especificadas en el contenido del campo de datos. Un “1” lógico en algún bit del campo de datos, concierne a que la correspondiente bobina debe ser puesta a ON. Un “0” refiere a que la correspondiente bobina debe ser puesta a OFF. El siguiente ejemplo muestra una petición para forzar una serie de 10 bobinas, tomando como referencia la bobina 20 *(direccionadas como 19, o 13 hex)* en el esclavo No 17.

El contenido de los datos son dos bytes: CD 01 hex (1100 1101 0000 0001 binario) (Tabla. 4.6) y 01 hex (00000001 binario) (Tabla. 4.7).

Tabla. 4.6. Estado de bobinas 20-27.

Bit	1	1	0	0	1	1	0	1
Coil	27	26	25	24	23	22	21	20

Tabla. 4.7. Estado de bobinas 28-29.

Bit	0	0	0	0	0	0	0	1
Coil	-	-	-	-	-	-	29	28

El primer byte trasmitido *(CD)* direcciona las bobinas 27-20, con el bit menos significativo direccionando a la bobina 20.

El siguiente byte transmitido (01 hex) direcciona las bobinas 29-28, con el bit menos significativo direccionando a la bobina 28. Los bits sobrantes en este byte son puestos a “0”.

Field Name	(Hex)
Slave Address	11
Function	0F
Coil Address Hi	00
Coil Address Lo	13
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Byte Count	02
Force Data Hi (Coils 27-20)	CD
Force Data Lo (Coils 29-28)	01
Error Check (LRC or CRC)	—

Respuesta:

Una respuesta normal retorna la dirección esclavo, código de función, dirección de inicio *(referencia),* y cantidad de bobinas forzadas*.* Aquí está un ejemplo de respuesta a la pregunta en el ejemplo anterior.

Field Name	(Hex)
Slave Address	11
Function	0F

Coil Address Hi	00
Coil Address Lo	13
Quantity of Coils Hi	00
Quantity of Coils Lo	0A
Error Check (LRC or CRC)	—

4.3.5 Respuestas de Excepción.

Cuando un dispositivo maestro envía una petición a un dispositivo esclavo, espera una respuesta normal.

Uno de cuatro posibles eventos puede ocurrir desde la petición del maestro:

- Si el dispositivo esclavo recibe la petición sin error de comunicación y puede manejar la petición normalmente, devuelve una respuesta normal.
- Si el esclavo no recibe la petición debido a un error de comunicación, no hay devolución de respuesta. El programa del maestro eventualmente procesará una condición de tiempo excedido - time out -, para la petición.
- Si el esclavo recibe la petición, pero detecta un error de comunicación (paridad, LRC, o CRC), no hay devolución de respuesta. El programa del maestro eventualmente procesará una condición de tiempo excedido – time out -, para la petición.
- Si el esclavo recibe la petición sin error de comunicación, pero no puede manejarla (por ejemplo, si la solicitud es leer un una bobina o registro inexistente), el esclavo devolverá una respuesta de excepción informando al maestro de la naturaleza del error.

El mensaje de respuesta de excepción tiene dos campos que lo diferencian de una respuesta normal.

4.3.5.1 Campo de Código de Función.

En una respuesta normal el esclavo pone el código de función de la petición original en el campo del código de función de la respuesta. Todos los códigos de función tienen el bit más significativo (MSB) a 0 (sus valores están por debajo de 80 hexadecimal).

En una respuesta de excepción, el esclavo establece el MSB del código de función a 1. Esto hace que el código de función en una respuesta de excepción resulte 80 hexadecimal más alto de lo que sería para una respuesta normal. Con el MSB del código de función activado, el programa de aplicación del maestro puede reconocer la respuesta de excepción y puede examinar en el campo de datos el código de excepción.

4.3.5.2 Campo de Datos.

En una respuesta normal, el esclavo puede devolver datos o estadísticas en el campo de datos (cualquier información que fuera solicitada en la petición).

En una respuesta de excepción, el esclavo devuelve un código excepción en el campo de datos. Esto define la condición del esclavo que causó la excepción.

A continuación se muestra un ejemplo de una petición del maestro y una respuesta de excepción por parte de un esclavo. Los campos del ejemplo se muestran en hexadecimal.

PETICION		Datos del Ejemplo
Byte	Nombre del Campo	(Hex)
1	Dirección del Esclavo	0A
2	Función	01
3	Dirección Comienzo Alto	04
4	Dirección Comienzo Bajo	A1
5	No. de Bobinas Alto	00
6	No. de Bobinas Bajo	01
7	LRC	4F

RESPUESTA DE EXCEPCION		Datos del Ejemplo
Byte	Nombre del Campo	(Hex)
1	Dirección del Esclavo	0A
2	Función	81
3	Código de Excepción	02
4	LRC	4F

En el ejemplo, el maestro direcciona una petición al dispositivo esclavo 10 (0A hex).

El código de función (01) es correspondiente a una operación Leer el Estado de una Bobina. Si la dirección de la bobina es inexistente en el dispositivo esclavo, el esclavo devolverá la respuesta de excepción con el código de excepción mostrado (02). Eso especifica un dato de dirección ilegal para el esclavo.

La Tabla. 4.8, contiene los posibles códigos de excepción que podría generar un dispositivo esclavo.

Tabla. 4.8. Listado de códigos de excepción.

Código	Nombre	Significado
01	FUNCIÓN ILEGAL	El código de función recibido en la petición no es una acción permitida para el esclavo.
02	DATO DE DIRECCIÓN ILEGAL	El dato de dirección recibido en la petición no es una dirección permitida para el esclavo.
03	DATO DE VALOR ILEGAL	Un valor contenido en el campo de datos de la petición no es un valor admisible para el

		esclavo.
04	FALLO DISPOSITIVO ESCLAVO	Ha ocurrido un error no recuperable mientras el esclavo estaba intentando ejecutar la acción solicitada.
05	RECONOCIMIENTO	El esclavo ha aceptado la petición y está procesándola, pero requerirá un tiempo largo para hacerlo. Esta respuesta se devuelve para prevenir que ocurra un error de tiempo excedido en el maestro. El maestro puede notificar a continuación un mensaje Completar Selección de programa para determinar si se ha completado el procesamiento.
06	DISPOSITIVO ESCLAVO OCUPADO	El esclavo está ocupado procesando un comando de programa de larga duración. El maestro debería retransmitir el mensaje más tarde cuando el esclavo esté libre.
07	RECONOCIMIENTO NEGATIVO	El esclavo no puede efectuar la función de programa recibida en la petición. Este código es devuelto por una petición de programación fallida, utilizando código de función 13 ó 14 decimal. El maestro debería pedir al esclavo diagnóstico o información de error.
08	ERROR DE PARIDAD EN MEMORIA	El esclavo ha intentado leer memoria extendida, pero ha detectado un error de paridad en la memoria. El maestro puede reintentar la petición, pero el servicio debe ser requerido al dispositivo esclavo.

4.4 LOOKOUT INTERFACE HOMBRE – MÁQUINA HMI

La versión Lookout 5.1 de National Instruments[11] es una versión del HMI (interface del usuario) y SCADA (sistema de supervisión y control distribuido) más fácil de usar disponible en el mercado de software actual. Con Lookout se puede construir aplicaciones de automatización industriales más fácil, rápidamente y a un costo sustancialmente inferior a otro software. Introducido por primera vez en 1989, Lookout continúa siendo el líder en el software de automatización industrial, combinando el poder de la tecnología distribuida orientada a objetos, controles ActiveX y capacidades de Internet para que se pueda obtener más rápidos beneficios de la inversión en el software.

Lookout de National Instruments es el software HMI/SCADA más fácil de usar en el mercado. Lookout es un software que permite fácilmente crear poderosas aplicaciones de monitoreo y control de procesos. Con Lookout, el desarrollo de la interface hombre-máquina toma menos tiempo permitiendo ahorrar sustancialmente en el costo total del proyecto. Estas son algunas de las características que lo hacen especial:

Lookout utiliza una arquitectura basada en objetos, con lo cual elimina completamente la programación, scripts o compilación separada. Solamente tiene que configurar y conectar objetos para desarrollar aplicaciones de monitoreo y control. La arquitectura basada en objetos permite fácilmente desarrollar y mantener aplicaciones, reduciendo aún más el costo total del proyecto.

Lookout es una poderosa herramienta de software (HMI y SCADA) de fácil uso para la automatización industrial. Se ejecuta bajo Windows y se comunica con Entradas/Salidas ubicadas en campo mediante PLCs, RTUs y otros dispositivos. Proyectos típicos de Lookout incluyen control monitoreo y supervisión continua de procesos, fabricación discreta, aplicaciones batch, y sistemas de telemetría remota.

[11] http://www.ni.com

Con Lookout, se puede crear representaciones gráficas sobre la pantalla de una computadora de dispositivos reales tales como interruptores (switchs), escalas gráficas, registradores de eventos, botones pulsadores (push-buttons), preillas (knobs), etc. y después enlazar sus imágenes a los actuales instrumentos de campo usando PLCs, RTUs, tarjetas DAQ, u otros dispositivos de Entradas/Salidas.

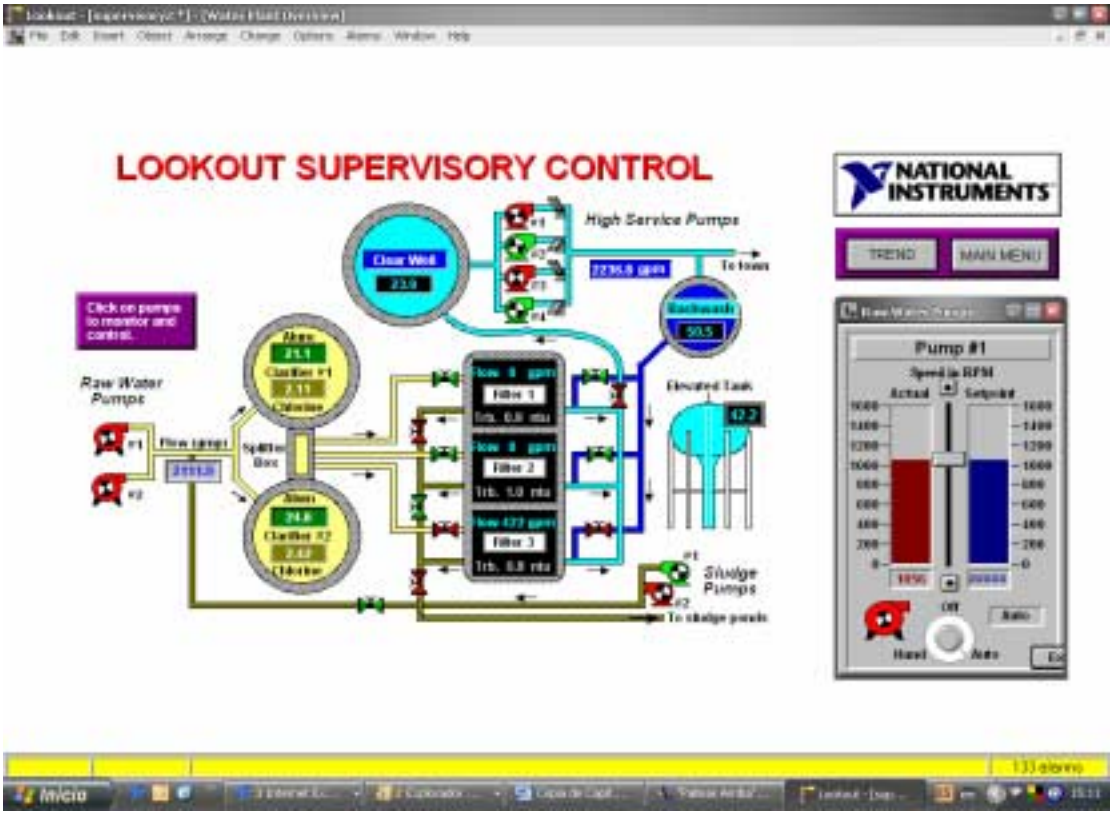


Figura. 4.4. Control y supervisión en Lookout.

En la Figura. 4.4 se puede ver la estructura de un objeto en Lookout, un objeto encapsula datos, parámetros y funcionalidad en un solo paquete. Un objeto es una unidad de programa que se auto contiene, y que tiene una base de datos predefinida, un grupo de parámetros y funcionalidad empotrada.

Se piensa en un objeto como un modelo programado de algo físico. Por ejemplo, un interruptor de luz es algo físico. Se puede prender y apagar. En Lookout un objeto Switch representa el interruptor físico. También puede prenderse o apagarse.

Los parámetros definen los límites de la funcionalidad del objeto. Por ejemplo, en el objeto Switch el parámetro Security Level determina quien puede prender o apagarlo, luego una base de datos puede almacenar información indicando la posición actual del interruptor.

4.4.1 Procedimiento de desarrollo de la Interface HMI.

Para el desarrollo de la interface se empieza por definir el proceso que se va a realizar, para esto es necesario abrir Lookout de la siguiente manera: **Inicio\Programas\National Instruments Lookout 5.1**, como se muestra en la Figura. 4.5.

Figura. 4.5. Software seleccionado para la interface HMI.

Una vez abierto, se procede a crear el proceso (Figura. 4.6.), en este caso, el proceso se llamará pn2800, que es el correspondiente a la estación neumática.

Figura. 4.6. Crear proceso en Lookout.

El proceso se realiza mediante conexiones de objetos. Las conexiones del objeto modbus con otros objetos que son importantes para la realización del HMI se describen a continuación.

El objeto modbus se comunica con el microcontrolador en el que previamente se ha cargado el programa PN_2800.bas, con la configuración que se muestra en la Figura. 4.7.

Figura. 4.7. Configuración del objeto Modbus.

En Lookout se puede conectar objetos permitiendo pasar señales entre los objetos, para la creación y conexiones de objetos se puede seguir los siguientes pasos:

Paso1. En el Icono Object se despliega un cuadro de opciones, se selecciona Create (Figura. 4.8.).

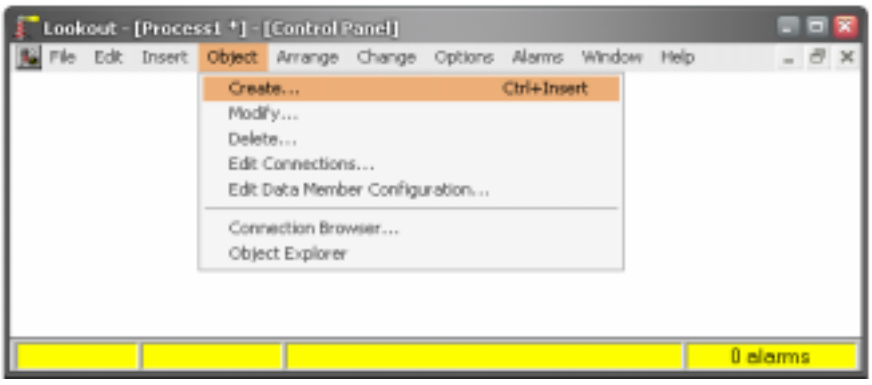


Figura. 4.8. Pantalla de creación del Objeto.

Paso 2. Aparece la pantalla de seleccionar el objeto, en este caso se selecciona el objeto Switch (Figura. 4.9.).

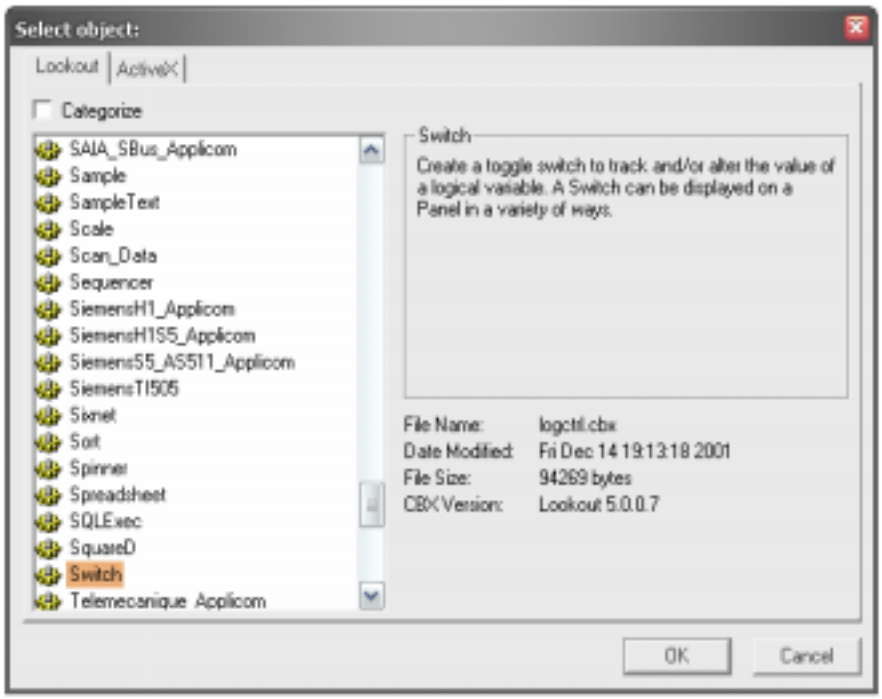


Figura. 4.9. Seleccionar objeto Switch.

Paso 3. Se despliega el panel referente al objeto switch, que se lo llamará Switch1 (Figura. 4.10.).

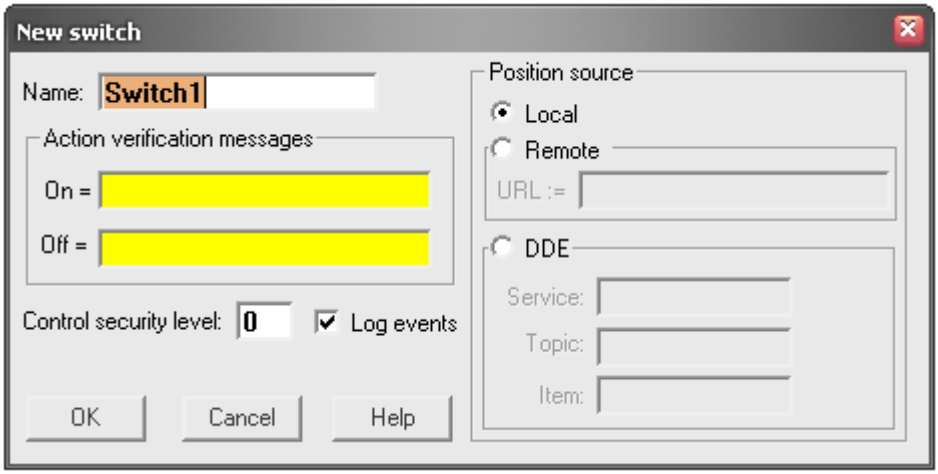


Figura. 4.10. Nombrar el objeto.

Paso 4. Escogemos qué gráfica se desea establecer para el Switch1 como se muestra en la Figura. 4.11.

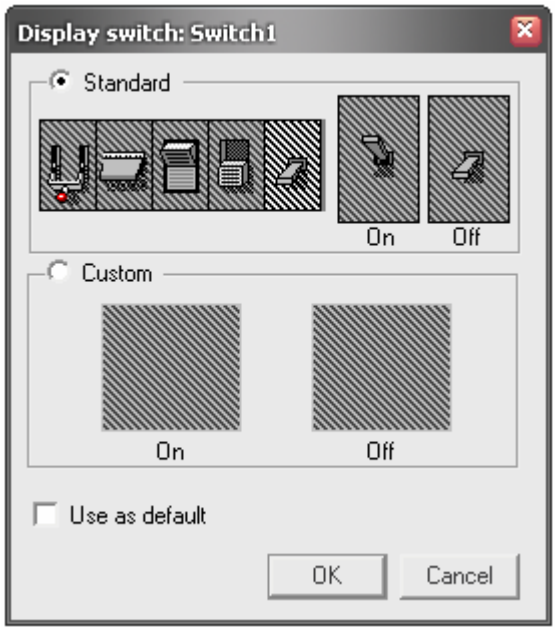


Figura. 4.11. Tipo de gráfica.

Paso 5. Aparece el switch en Panel de control y se procede a crear una expresión para este objeto (Figura. 4.12.).

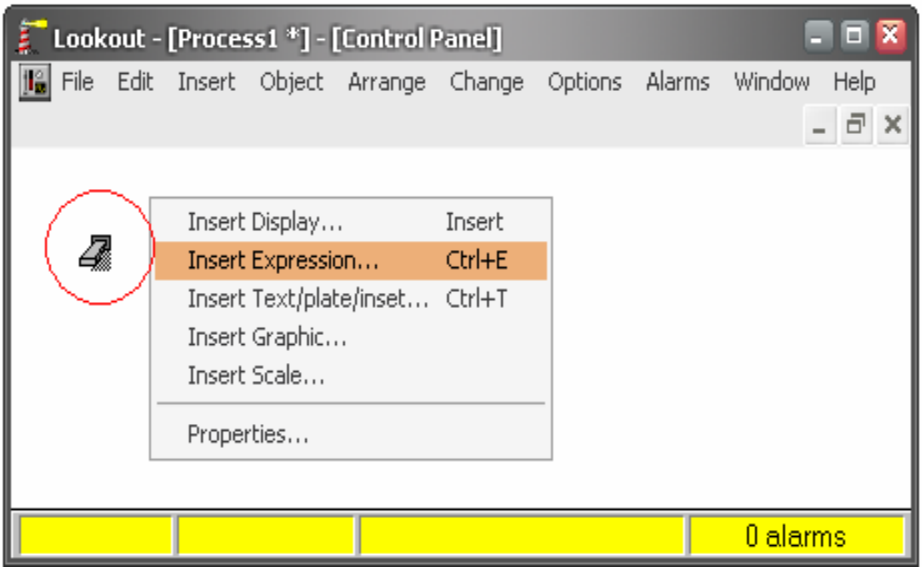


Figura. 4.12. Insertar expresión.

Paso 6. Se procede a escribir Modbus1., que es el objeto y 00004, que es la bobina que se va a controlar (Figura. 4.13.).

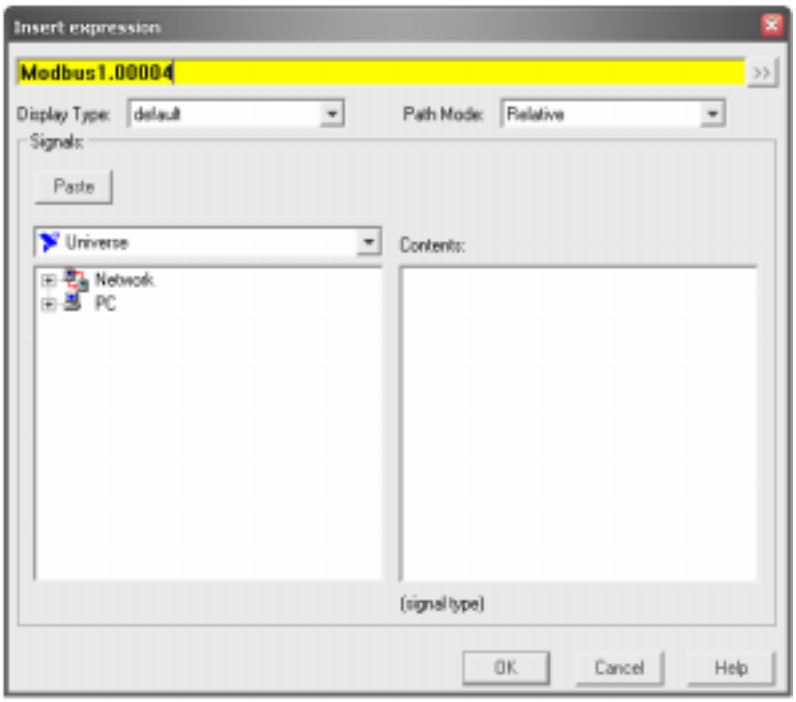


Figura. 4.13. Nombrar expresión.

Paso 7. Se escoge la gráfica que se desea establecer para la señal lógica como se muestra en la Figura. 4.14.

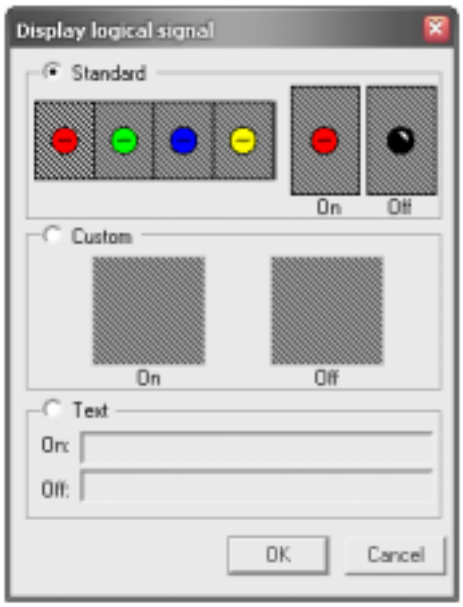


Figura. 4.14. Tipo de gráfica.

Paso 8. Se conecta los objetos que se creó (Figura. 4.15.).

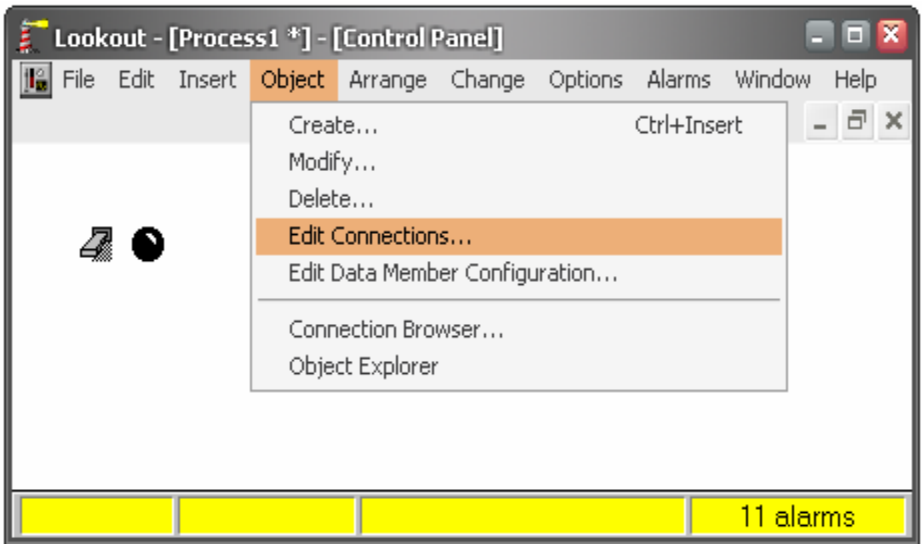


Figura. 4.15. Pantalla de conexión.

Paso 9. Se selecciona la bobina 00001 y se conecta con el switch1 (Figura. 4.16.).

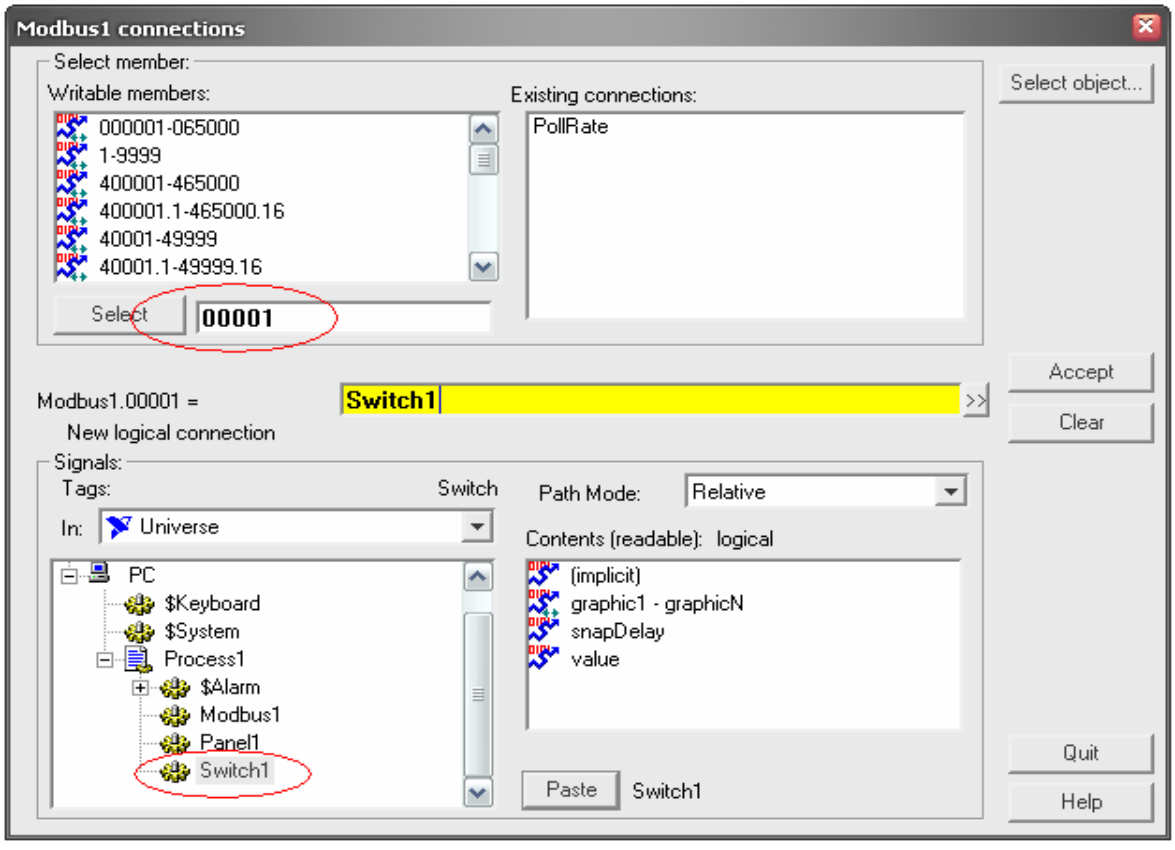


Figura. 4.16. Pantalla de conexión del Modbus1.

Paso 10. Se despliega el Panel de control con la conexión del objeto con su respectiva expresión, apagado (Figura. 4.17.) y encendido (Figura. 4.18.).

Figura. 4.17. Panel principal con Switch Off.

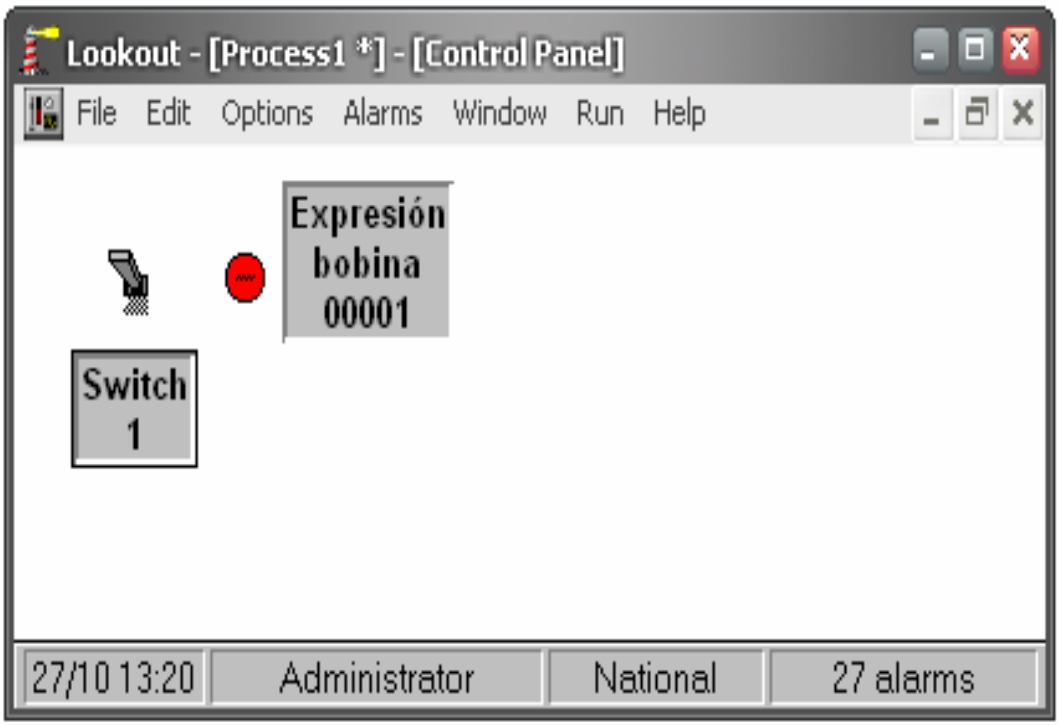


Figura. 4.18. Panel principal con Switch On.

4.4.2 Esquema de conexión Modbus para la automatización de la estación neumática.

Se han utilizado dos objetos de Lookout principalmente, Sequencer y Multi-state. El objeto Sequencer genera una secuencia de estados asociados con las salidas, se usa para implementar bucles o para árboles de decisiones complicadas, se puede trabajar hasta 100 estados diferentes con su respectivo límite de tiempo, los estados se pueden asociar con un máximo de 26 salidas que tienen un control ON/OFF. El Sequencer se quedará en el estado en curso hasta que se haya agotado el límite de tiempo o hasta que se interrumpa con los comandos Goto o Jump solo ahí pasa al siguiente estado. El objeto Multi-state exhibe una ilustración gráfica única para cada uno de hasta seis condiciones diferentes sobre la base de la lógica de if-then-else avanzada, las expresiones condicionales son asociadas con las entradas. Si algunas expresiones condicionales son verdaderas inmediatamente el objeto de Multi-state exhibe la ilustración gráfica relacionada con la primera expresión verdadera.

En el Control Panel se han conectado diez interruptores para la selección de bobinas del Manipulador de Pallets y Manipulador de Cilindros de forma manual y un interruptor para el modo semi-automático del Manipulador de Pallets y Cilindros respectivamente como se observa en la Figura. 4.19.

Se ha conectado los objetos de Multi-state que sirven para monitorear el estado de las bobinas con su respectivo gráfico y el Sequencer que sirve para los procesos de forma semi-automática del Manipulador de Cilindros y Pallets como se nota en la Figura. 4.20. El Multi-state 1 está asociado con las entradas 10001 hasta 10006 que están asignadas al brazo manipulador de pallets, el Multi-state 2 está asociado con las entradas 10010, 10011, 10013, 10014 y 10020 que están asignadas al manipulador de cilindros. El Sequencer 1 está asociado con las salidas 00005 hasta 00008 que están asignadas al manipulador de cilindros y el Sequencer 2 está asociado con las salidas 00001 hasta 00004 que están asignadas al manipulador de pallets.

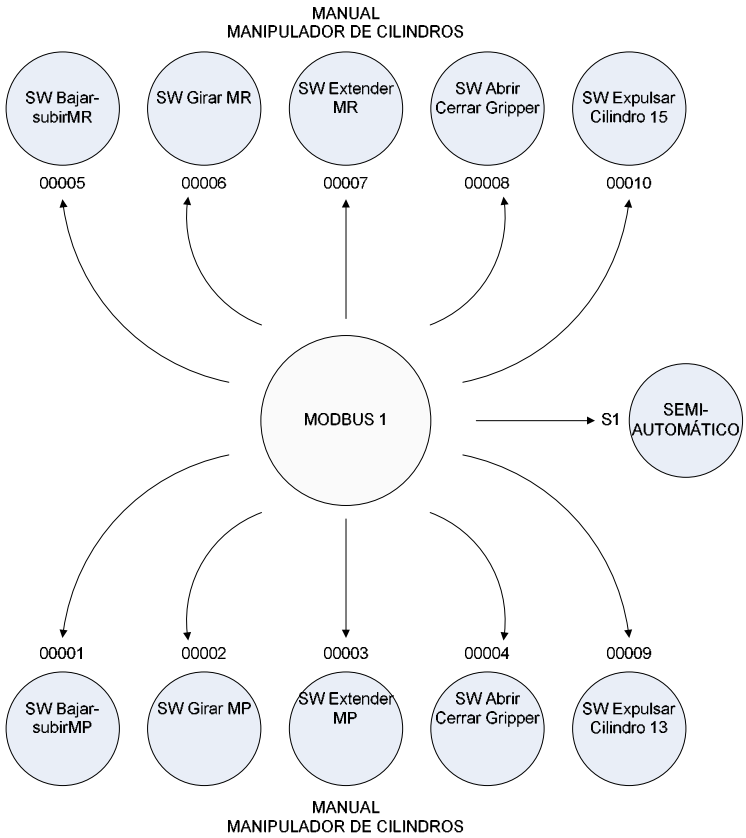


Figura. 4.19. Conexiones del objeto Modbus en el Control Panel de Interruptores.

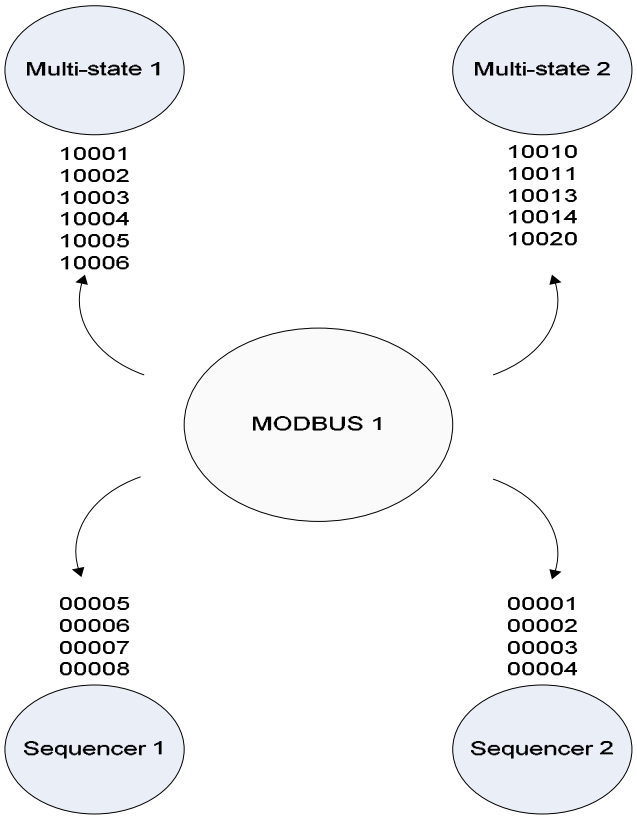


Figura. 4.20. Conexiones del objeto Modbus en el Control Panel del Multi-state y Sequencer.

Para las conexiones en el Multi-state se tiene en cuenta las entradas que se accionan y las entradas que no se accionan como condiciones de programación, mediante esta lógica se procede a realizar el respectivo gráfico el cual aparece señalado en la Figura. 4.21 y Figura. 4.22.

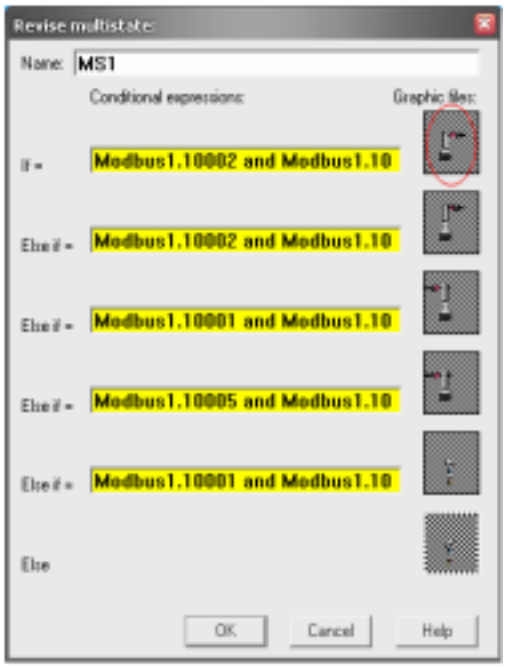


Figura. 4.21. Creación del Multi-state para el Manipulador de pallets.

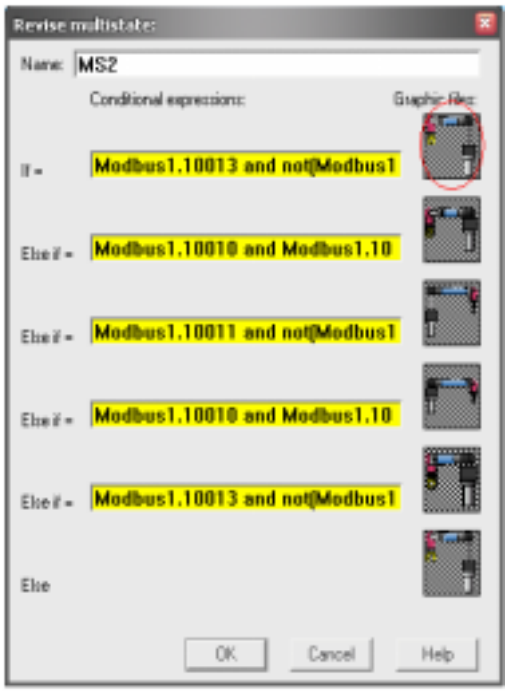


Figura. 4.22. Creación del Multi-state para el Manipulador de cilindros.

Para las conexiones en el Sequencer se tiene en cuenta las bobinas que se accionan y bobinas que no se accionan como condiciones de programación, siguiendo las condiciones de secuencia como se observa en la Figura. 4.23 y Figura. 4.24.

En los Check Box de las salidas; A corresponde a la bobina 00001, B corresponde a la bobina 00002, etc.

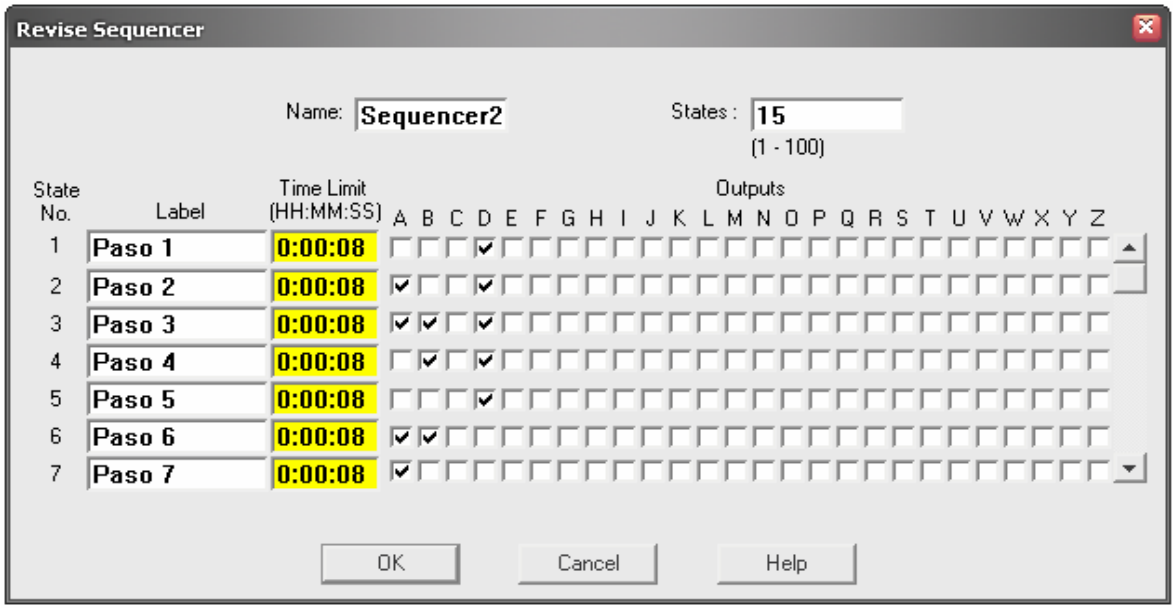


Figura. 4.23. Creación del Sequencer para el Manipulador de Pallets.

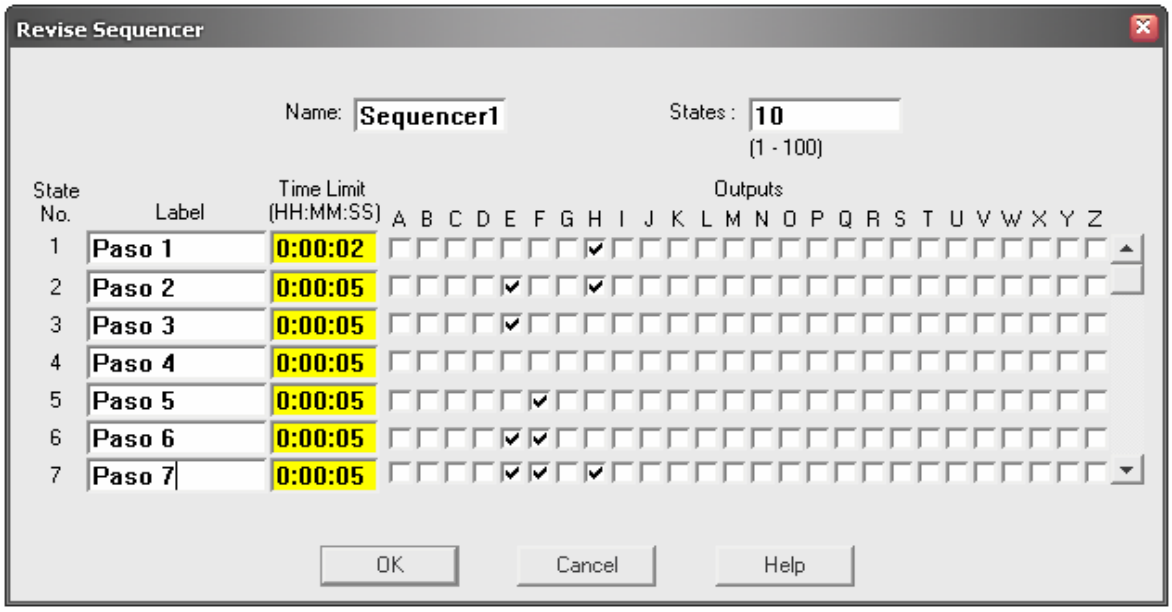


Figura. 4.24. Creación del Sequencer para el Manipulador de cilindros.

CAPÍTULO V

PRUEBAS Y RESULTADOS

5.1 PRUEBAS

En pruebas iniciales, para la generación del código que permita la comunicación bajo el protocolo Modbus RTU, se recurrió a herramientas de software como: Mbus, Lookout y Microcode Studio Plus *(bootloader).*

5.1.1 Mbus.

Es un software examinador del protocolo Modbus, el cual permite variar ciertos parámetros de este protocolo, así como monitorear gráficamente las tramas enviadas y recibidas. La Figura. 5.1 muestra la pantalla en la que el usuario puede ajustar parámetros como: baud rate, paridad, bits de stop…etc. Cabe mencionar que el programa permite únicamente trabajar con la opción Modbus RTU, ya que las demás opciones se hallan en proceso de desarrollo.

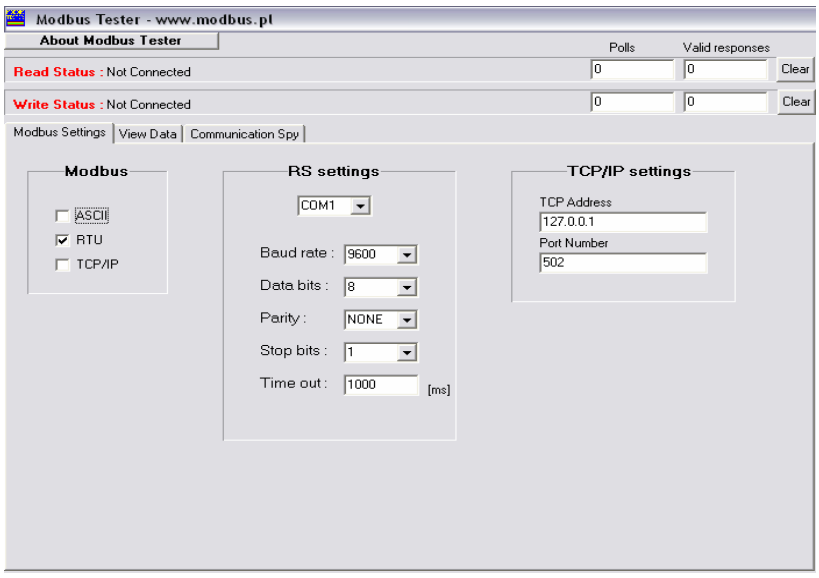


Figura. 5.1. Parámetros Modbus Tester.

En la pantalla que ilustra la Figura. 5.2 se escoge la dirección del dispositivo esclavo, así como el tipo de función de trabajo *(leer bobinas, leer entradas,….etc.)*

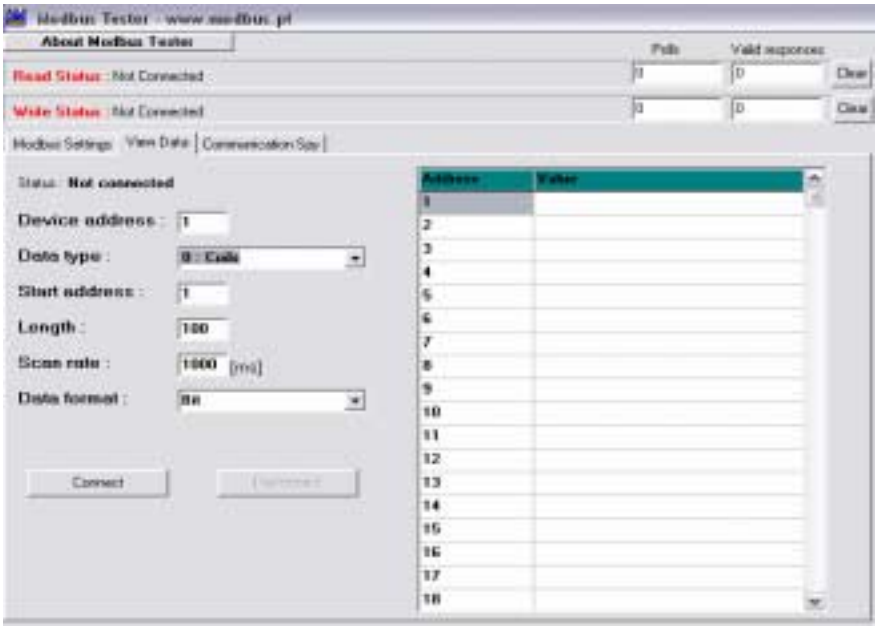


Figura. 5.2. Funciones Modbus Tester.

La pantalla de comunicación (Figura. 5.3) despliega tanto las tramas enviadas y recibidas como el número de respuestas válidas o erróneas.

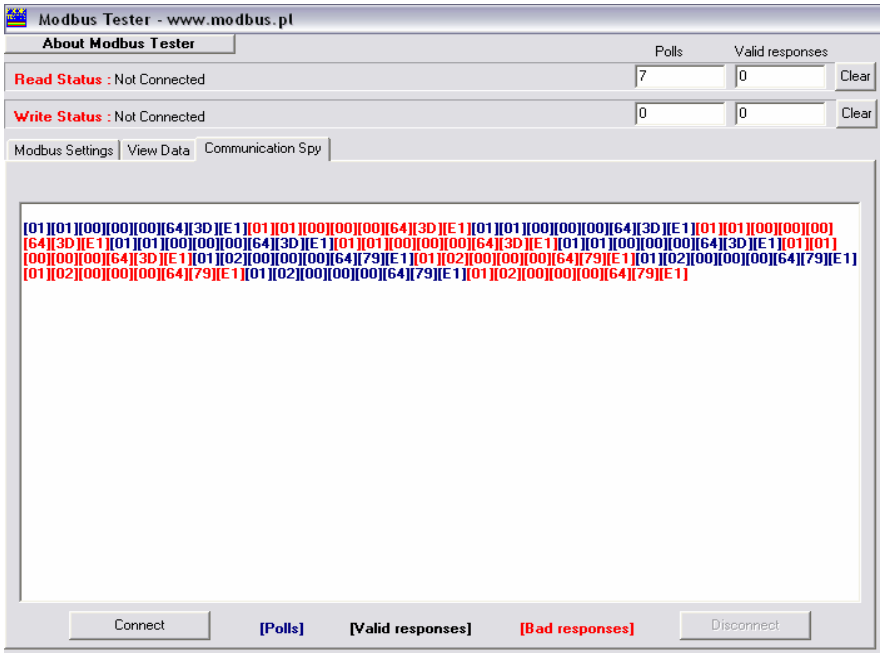


Figura. 5.3. Comunicación Modbus Tester.

5.1.2 Monitoreo de tramas con Lookout.

Lookout permite generar y monitorear tramas Modbus con las que no cuenta el software Mbus, como por ejemplo: *forzar coils*. Para ello, en el panel principal es preciso realizar la secuencia que se muestra en los gráficos de la Figura 5.4.

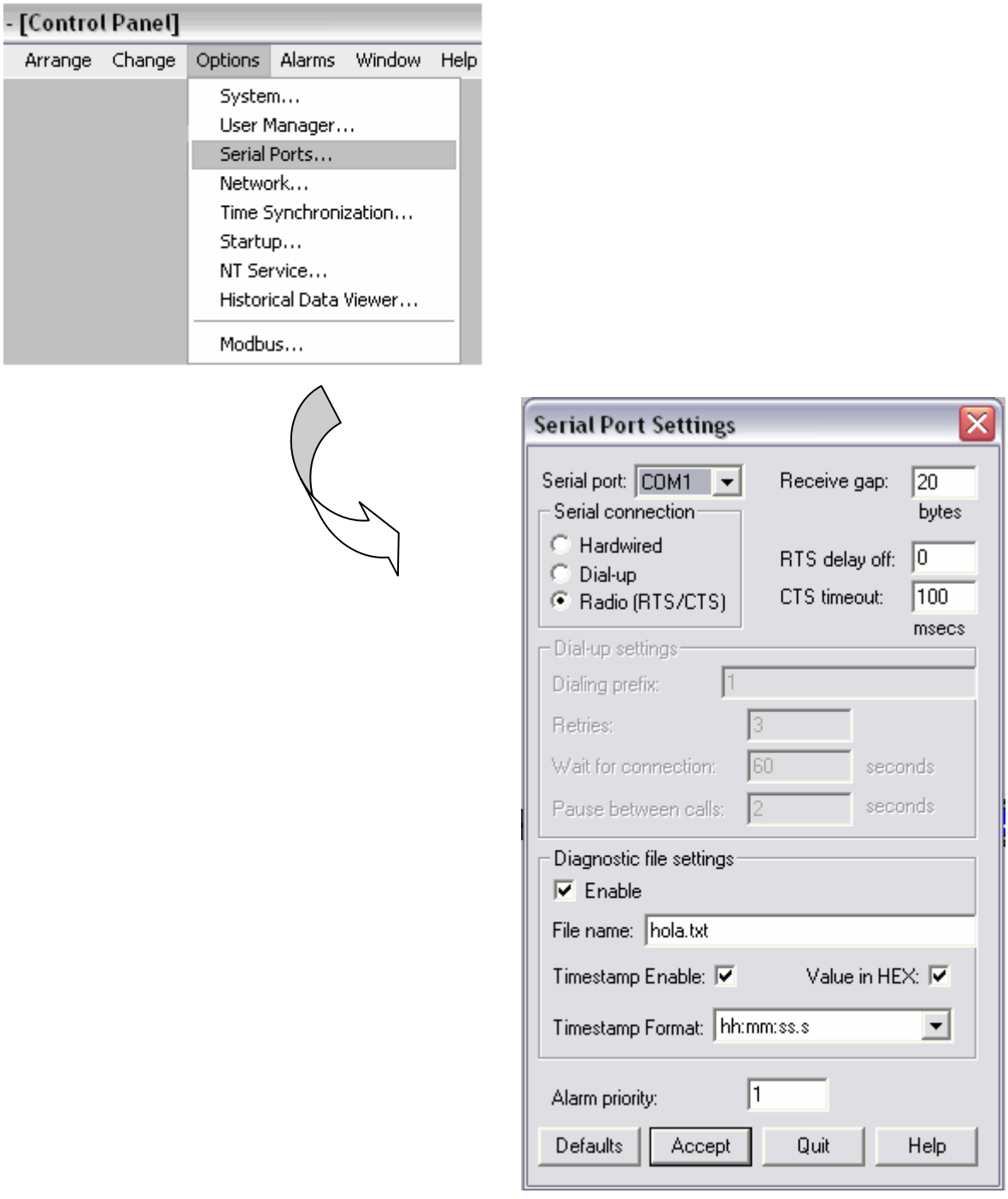


Figura. 5.4. Creación del archivo de monitoreo de tramas.

Este proceso generará un documento *hola* con extensión .txt en la carpeta donde se halla los archivos de Lookout en uso.

Tendrá una apariencia semejante a esta:

23:56:31.0 - Modbus1 ->

[01][04][00][00][00][01][31][CA]

23:56:31.3 - Modbus1 <-

[01][04][00][00][00][01][31][CA]

23:56:31.3 - Modbus1 ->

[3A][30][31][30][34][30][30][30][30][30][30][30][31][46][41][0D][0A]

23:56:31.6 - Modbus1 <-

[3A][30][31][30][34][30][30][30][30][30][30][30][31][46][41][0D][0A]

23:56:31.6 - Modbus1 ->

[01][04][00][00][00][01][31][CA]

23:56:31.8 - Modbus1 <-

[01][04][00][00][00][01][31][CA]

Las flechas hacia la derecha indican la trama que el maestro ha enviado al esclavo, mientras que las flechas hacia la izquierda muestran la trama que el esclavo ha enviado como respuesta a la petición del maestro.

5.1.3 Microcode Studio Plus.

La funcionalidad de la herramienta de Microcode Studio Plus *(Microcode Loader)* eliminó la engorrosa tarea de extraer el microcontrolador de su lugar de trabajo para su programación. De esta forma, se pudo realizar de manera rápida y eficiente las pruebas necesarias para la comprobación del correcto

funcionamiento del hardware implementado, así como para la elaboración de los programas de aplicación.

La Figura. 5.5 muestra a Microcode Loader en pleno proceso de descarga del programa usuario hacia el microcontrolador usando el puerto serial de la PC.

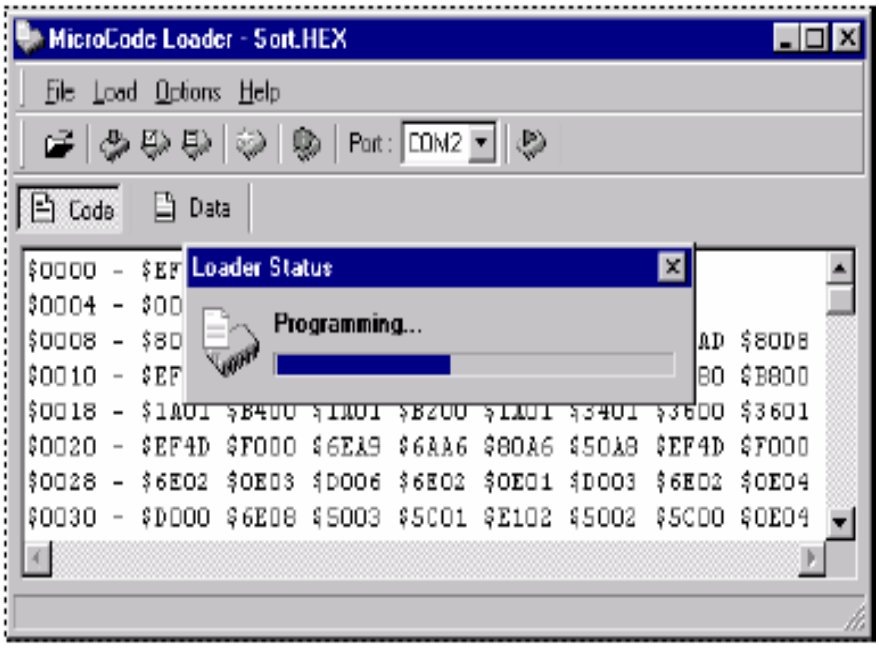


Figura. 5.5. Carga de firmware vía bootloader.

5.2 RESULTADOS

Posterior a la carga del programa PLC_2800.hex en el microcontrolador vía bootloader, y la ejecución de los programas *Mbus* o *LooKout.* Se obtuvieron los siguientes resultados.

5.2.1 Mbus.

Se puede ver que las respuestas por parte del microcontrolador son válidas, ya que las mismas tienen un color negro (Figura. 5.6). Si las respuestas no fuesen válidas, estas tendrían un color rojo.

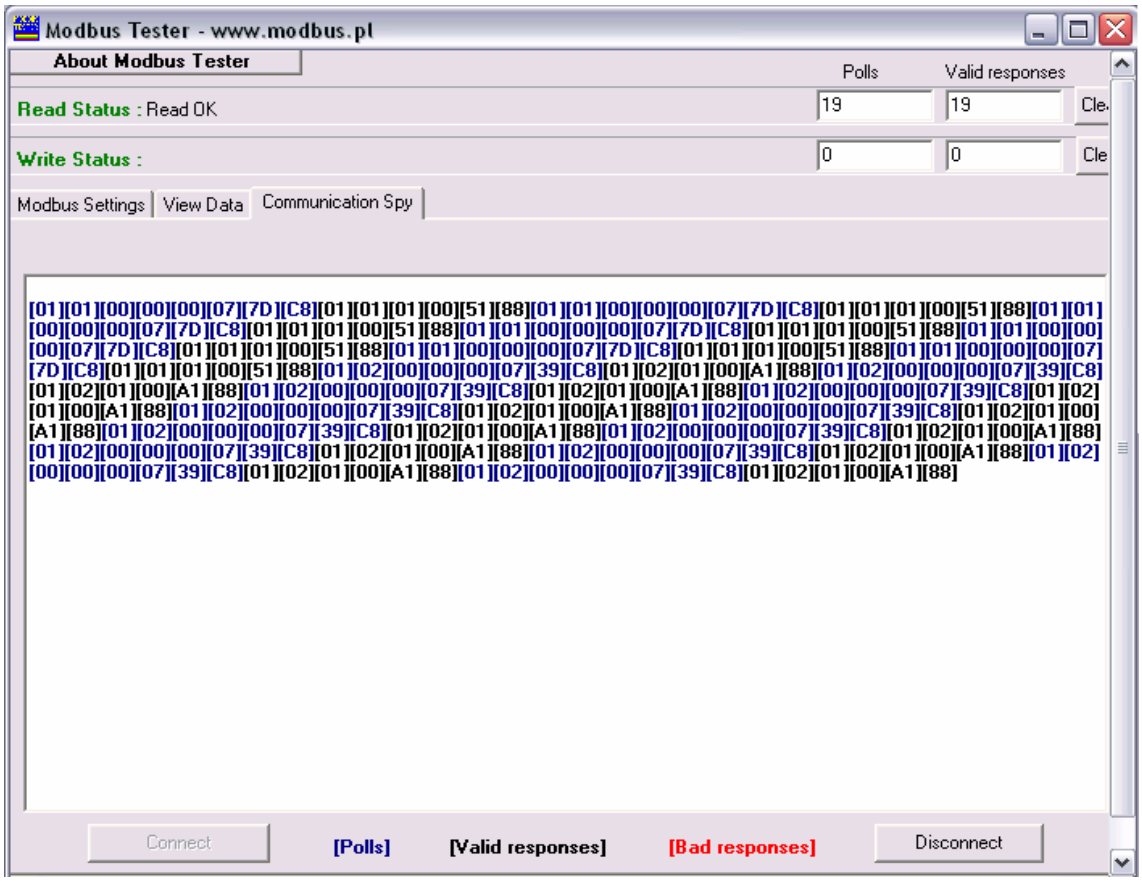


Figura. 5.6. Respuesta válida del microcontrolador.

5.2.2 Estadísticas de la comunicación Modbus en Lookout.

Lookout posee una herramienta que muestra las estadísticas de la comunicación Modbus, tales como: respuestas válidas, errores, mal CRC, etc. Para ello es necesario realizar los pasos que indican los gráficos de la Figura. 5.7.

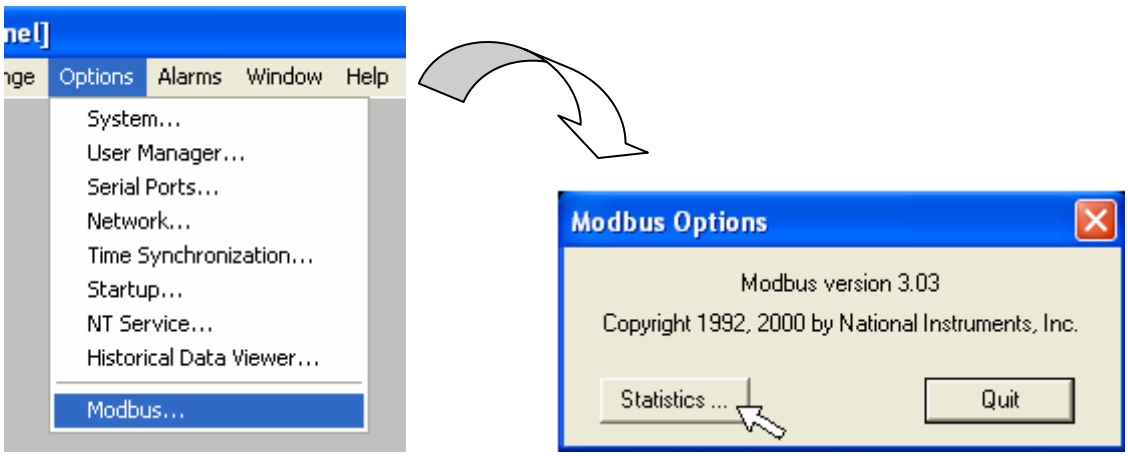


Figura. 5.7. Pasos para mostrar estadísticas Modbus.

Posterior a estos pasos se despliega el cuadro de las estadísticas Modbus (Figura. 5.8), el cual confirma que el programa PLC_2800 desarrollado funciona de manera óptima y libre de errores.

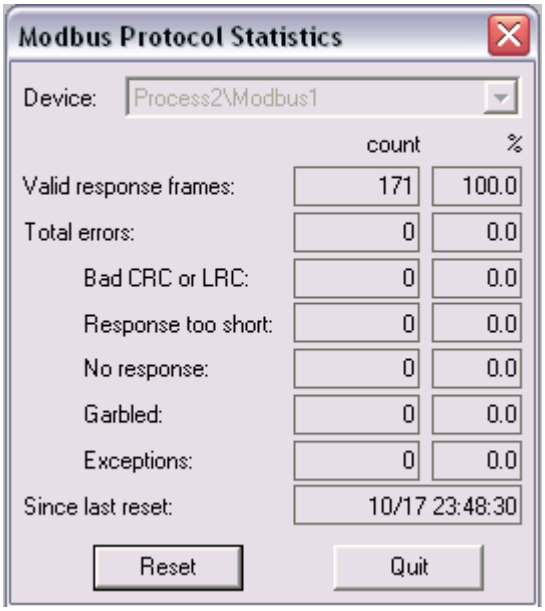


Figura. 5.8. Cuadro estadístico Modbus.

El protocolo Modbus implementado, además de cumplir con un objetivo del proyecto, también sirvió para verificar el correcto funcionamiento del hardware propuesto.

CAPÍTULO VI

PRÁCTICAS

6.1 PROCEDIMIENTO PARA REALIZAR LA CARGA DE PROGRAMAS AL MICROCONTROLADOR.

Previo al uso de MicroCode Loader, con la ayuda del programa Icprog, es necesario cargar por única vez en el microcontrolador el programa bootloader 16F877_04.Hex (Figura. 6.1).

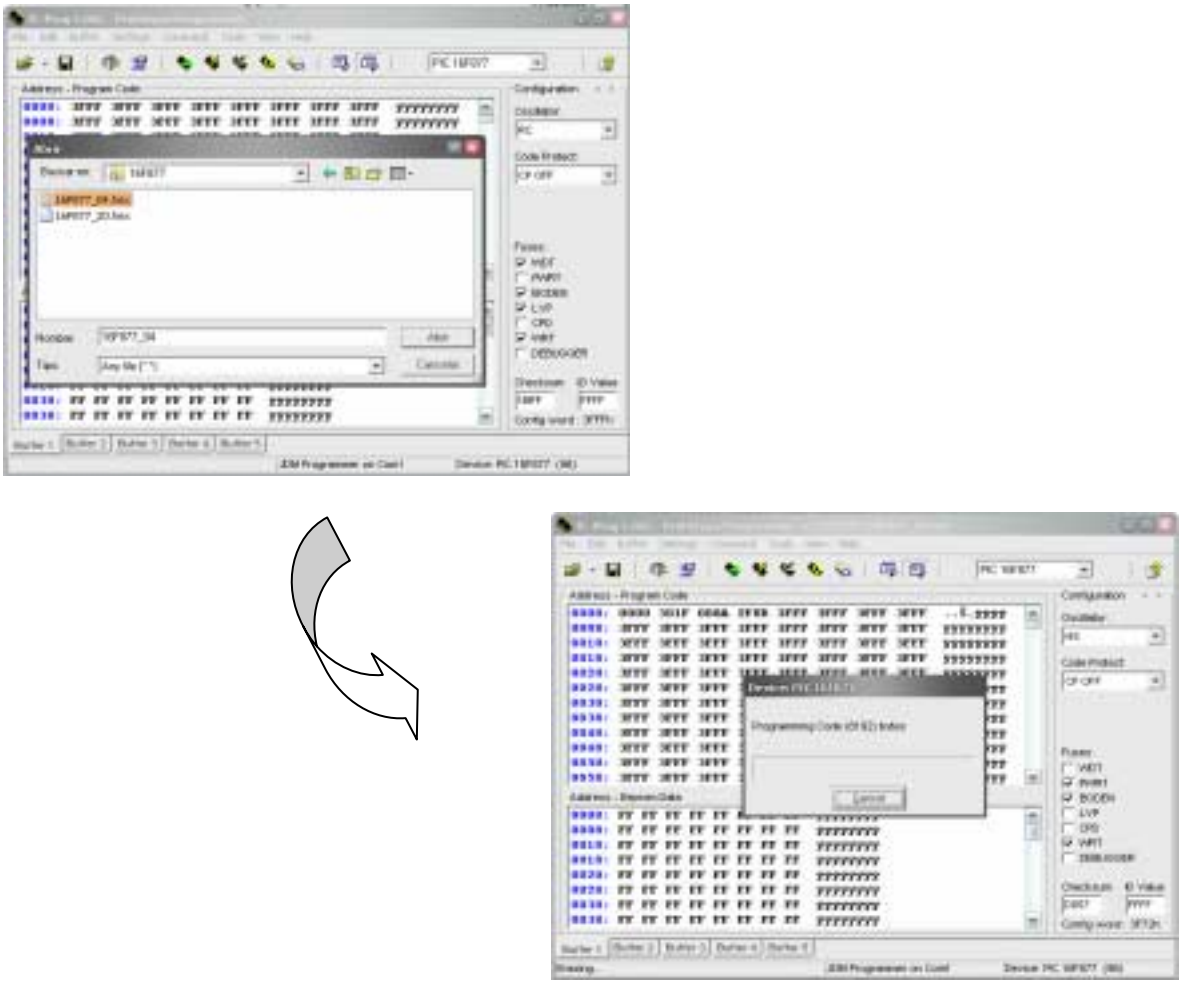


Figura. 6.1. Pasos para la carga del Bootcode.

La elaboración de los programas se realiza en la hoja de programación que ofrece el programa MicroCode Studio Plus (Figura. 6.2).

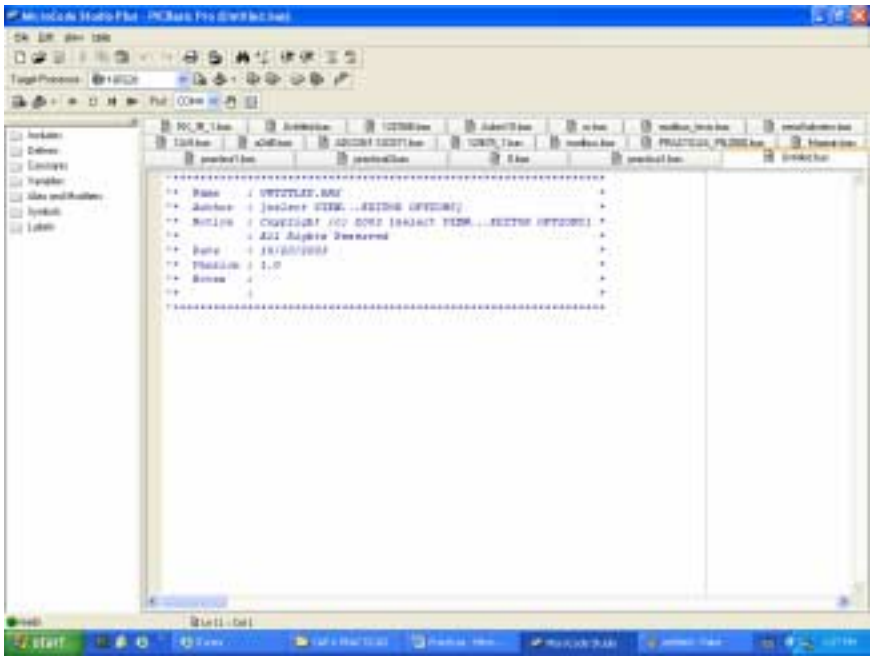


Figura. 6.2. Página de Programación en MicroCode.

Posterior a la escritura del código de programación en MicroCode Studio Plus, y la correspondiente conexión del dispositivo de control con la PC mediante cable serial, se debe ejecutar un clic en el icono señalado en la Figura. 6.3. Esta acción compilará el código de programación y el loader tomará el mando del programa escrito por el usuario en el microcontrolador.

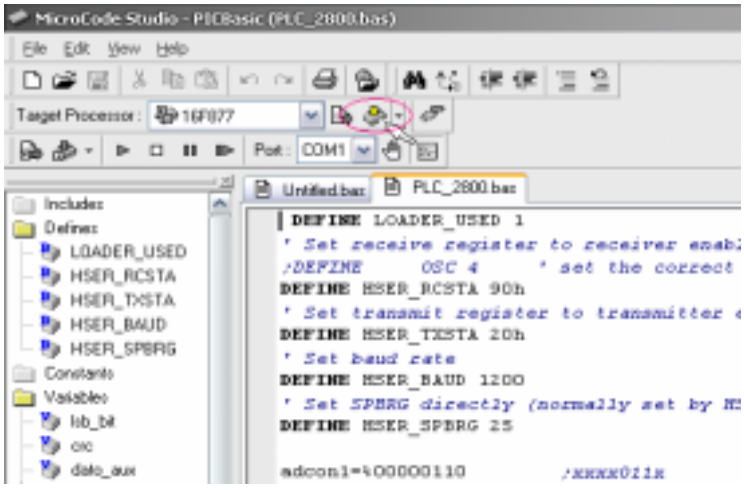


Figura. 6.3. Compilación y evocación del bootloader.

Microcode Loader solicitará resetear el microcontrolador (Figura. 6.4).

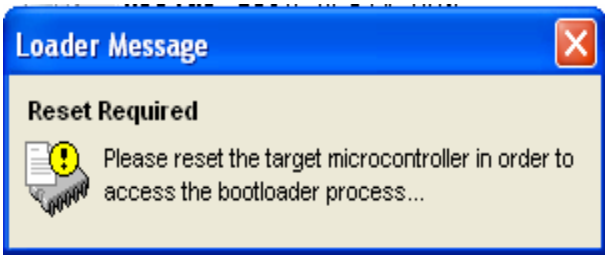


Figura. 6.4. Requerimiento de Reset.

Posterior al reset manual en el dispositivo de control, la carga del programa usuario se iniciará (Figura. 6.5).

Cabe mencionar que únicamente es necesario fijar el puerto COM con el que se trabajará, ya que las velocidades de transmisión son manejadas automáticamente por MicroCode Loader.

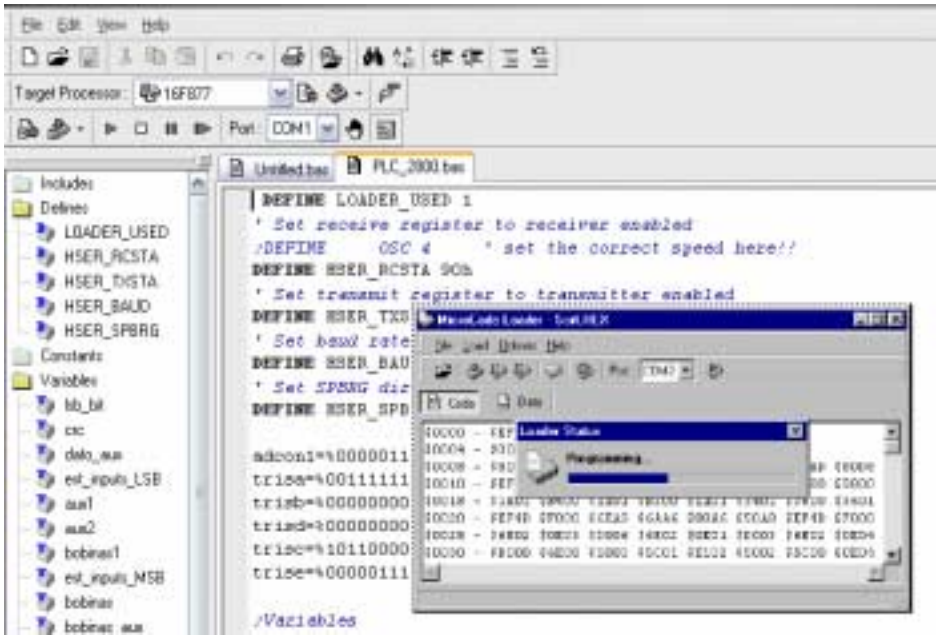


Figura. 6.5. Carga del programa al microcontrolador en proceso.

Tan pronto haya finalizado la carga del programa usuario, este será ejecutado por el microcontrolador.

Al realizar las prácticas se debe tener en cuenta los siguientes aspectos:

1. Definición de usuario bootloader.

   Para que la carga del nuevo firmware sea puesta en el microcontrolador vía bootloader, se debe poner en el inicio de la página de programación la siguiente expresión:

   ```
   DEFINE LOADER_USED 1                ; habilita el registro de recepción
   ```

2. La asignación de I/Os de la estación PN-2800 con los puertos del PIC16F877, así como el procedimiento para leer el estado de los 32 posibles sensores.

Los puertos B y D del microcontrolador han sido designados para trabajar como salidas mediante la siguiente disposición (Tabla. 6.1).

Tabla. 6.1. Disposición de entradas y salidas.

Puerto PIC16F877	OUT	Electro válvula
B.0	000001	SOV1/MP
B.1	000002	SOV2/MP
B.2	000003	SOV3/MP
B.3	000004	SOV4/MP
B.4	000005	SOV5/MP
B.5	000006	SOV6/MP
B.6	000007	SOV7/MP
B.7	000008	SOV8/MP
D.0	000009	SOV9/MP
D.1	000010	SOV10/MP
D.2	000011	SOV11/MP
D.3	000012	SOV12/MP
D.4	000013	SOV13/MP
D.5	000014	SOV14/MP
D.6	000015	SOV15/MP
D.7	000016	Indicador de error

6.1.1 Ejemplos.

6.1.1.1 Ejemplo 1.

Si se desea activar las bobinas 000002 y 000013, con las restantes apagadas, se debe escribir en la página de programación:

Portb = %00000010
Portd = %00010000

Los puertos A y E del microcontrolador han sido designados para trabajar como entradas.

La asignación de las entradas es dependiente del estado de los puertos C3, C2, C1 y C0, ya que los mismos son los encargados de seleccionar el grupo de 8 entradas que serán leídas.

Para los valores que se indica en la Tabla. 6.2., los puertos A y E quedan asignados a las entradas que muestra la Tabla. 6.3.

Tabla. 6.2. Selección del primer bloque de 8 entradas.

C3	C2	C1	C0
0	0	0	1

Tabla. 6.3. Asignación de entradas a Puertos A y E.

Puerto PIC16F877	IN	Sensor
A0	100001	LS1/MPU
A1	100002	LS2/MPD
A2	100002	LS3/MPR
A3	100004	LS4/MPL
A4	100005	LS5/MPF
A5	100006	LS6/MPB
E0	100007	LS7/PBUF
E1	100008	

Para los valores que se indica en la Tabla. 6.4., los puertos A y E quedan asignados a las entradas que muestra la Tabla. 6.5.

Tabla. 6.4. Selección del segundo bloque de 8 entradas.

C3	C2	C1	C0
0	0	1	0

Los puertos A y E quedan asignados a las siguientes entradas (Tabla. 6.5):

Tabla. 6.5. Asignación de entradas a Puertos A y E.

Puerto PIC16F877	IN	Sensor
A0	100009	LS1/MRU
A1	100010	LS2/MRD
A2	100011	LS3/MRR
A3	100012	LS4/MRL
A4	100013	LS5/MRF
A5	100014	LS6/MRB
E0	100015	Reserva
E1	100016	Reserva

Para los valores que se indica en la Tabla. 6.6., los puertos A y E quedan asignados a las entradas que muestra la Tabla. 6.7.

Tabla. 6.6. Selección del tercer bloque de 8 entradas.

C3	C2	C1	C0
0	1	0	0

Tabla. 6.7. Asignación de entradas a Puertos A y E.

Puerto PIC16F877	IN	Sensor
A0	100017	Modulo Set Point
A1	100018	Modulo Set Point
A2	100019	Modulo Set Point
A3	100020	Modulo Set Point
A4	100021	LS/PLT
A5	100022	LS1/BATCH
E0	100023	LS/S1
E1	100024	LS/S2

Para los valores que se indica en la Tabla. 6.8., los puertos A y E quedan asignados a las entradas que muestra la Tabla. 6.9.

Tabla. 6.8. Selección del cuarto bloque de 8 entradas.

C3	C2	C1	C0
1	0	0	0

Tabla. 6.9. Asignación de entradas a Puertos A y E.

Puerto PIC16F877	IN	Sensor
A0	100025	LS/SH
A1	100026	Presostato
A2	100027	CP5
A3	100028	LS/PLTS
A4	100029	CP4
A5	100030	Reserva
E0	100031	Reserva
E1	100032	Reserva

La Tabla. 6.10, describe la representación de los sensores utilizados para las prácticas.

Tabla. 6.10. Descripción de electro – válvulas.

Electro - válvula	Descripción
SOV1/MP	Arriba/Abajo manipulador de pallets
SOV2/MP	Gira manipulador de pallets
SOV3/MP	Extender/Contraer brazo manipulador de pallets
SOV4/MP	Abrir/Cerrar gripper
SOV5/MP	Arriba/Abajo manipulador de cilindros
SOV6/MP	Gira manipulador de cilindros
SOV7/MP	Extender/Contraer brazo manipulador de cilindros
SOV8/MP	Abrir/Cerrar gripper
SOV9/MP	Reserva
SOV10/MP	Reserva
SOV11/MP	Reserva
SOV12/MP	Reserva
SOV13/MP	Pistón de almacén de pallets
SOV14/MP	Reserva
SOV15/MP	Pistón para el canal de cilindros

Tabla. 6.11. Descripción de sensores.

Sensor	Descripción
L4 ERROR	Indicador de error
LS1/MPU	Manipulador de pallets arriba
LS2/MPD	Manipulador de pallets abajo
LS3/MPR	Manipulador de pallets en la derecha
LS4/MPL	Manipulador de pallets en la izquierda
LS5/MPF	Manipulador de pallets
LS6/MPB	Mano de manipulador de pallets atrás
LS1/MRU	Subir manipulador de cilindros
LS2/MRD	Bajar manipulador de cilindros
LS3/MRR	Rotar manipulador de cilindros a la derecha
LS4/MRL	Rotar manipulador de cilindros a la izquierda
LS5/MRF	Mano de manipulador de cilindros adelante
LS6/MRB	Mano de manipulador de cilindros atrás
LSG/MR	Agarradera del manipulador del cilindros (set1)
LSG/MR	Agarradera del manipulador del cilindros (set2)
LSG/MR	Agarradera del manipulador del cilindros (set3)
LSG/MR	Cerrar agarradera del manipulador del cilindros
LS/PLT	Detección de pallet fuera
LS/BATCH	Presencia de prismas en almacén de prismas
LS/S1	Cilindro 1 en carril 1
LS/S1	Cilindro2 en carril 2
LS/CH	Detección de cilindro
PS 201	Detección de presión baja
READY	Encendido
LS/PLTS	Detección de pallet en almacén

6.1.1.2 Ejemplo 2.

Si se desea almacenar el estado de las entradas 9 a 16 en una variable tipo byte llamada X, se debe poner en la página de programación lo siguiente:

```
Port c = %00000010          ; habilita buffer1
Y = porte                   ; guarda estado de puerto E en Y
Y = Y << 6                  ; desplaza el contenido de X 6 bits hacia la izquierda
X = porta                   ; guarda estado de puerto A en X
X = X &│%00111111          ; pone a cero los dos bits MSB
X = X │ Y                  ; X OR Y  el resultado es guardado en X
```

Si se quiere almacenar el estado de las entradas 24 a 32 en una variable tipo byte llamada X, se debe poner en la página de programación lo siguiente:

```
Port c = %00001000          ; habilita buffer1
Y = porte                   ; guarda estado de puerto E en Y
Y = Y << 6                  ; desplaza el contenido de X 6 bits hacia la izquierda
X = porta                   ; guarda estado de puerto A en X
X = X &│%00111111          ; pone a cero los dos bits MSB
X = X │ Y                  ; X OR Y  el resultado es guardado en X
```

6.2 PRÁCTICA No.1 RUTINAS DE CONTROL BÁSICAS

Esta práctica tiene como objetivo, colocar en el puerto de entrega del *brazo manipulador de pallets* un pallet con su correspondiente cilindro, para lo cual, es preciso primero trasladar el cilindro desde el puerto de entrada del *brazo manipulador de cilindros* hacia el puerto de entrada del *brazo manipulador de pallets.*

Aquí dicho cilindro es ubicado en el interior de un pallet y trasladado hacia el puerto de entrega de material de la estación PN-2800.

Esta práctica se puede realizar de dos maneras:

1) Actuadores en función de los sensores.

En esta manera de realizar secuencias, las válvulas permanecen en cierto estado mientras un determinado sensor o sensores no cambien de estado.

En el diagrama de flujo de la Figura. 6.6 se puede ver la lógica utilizada para esta práctica.

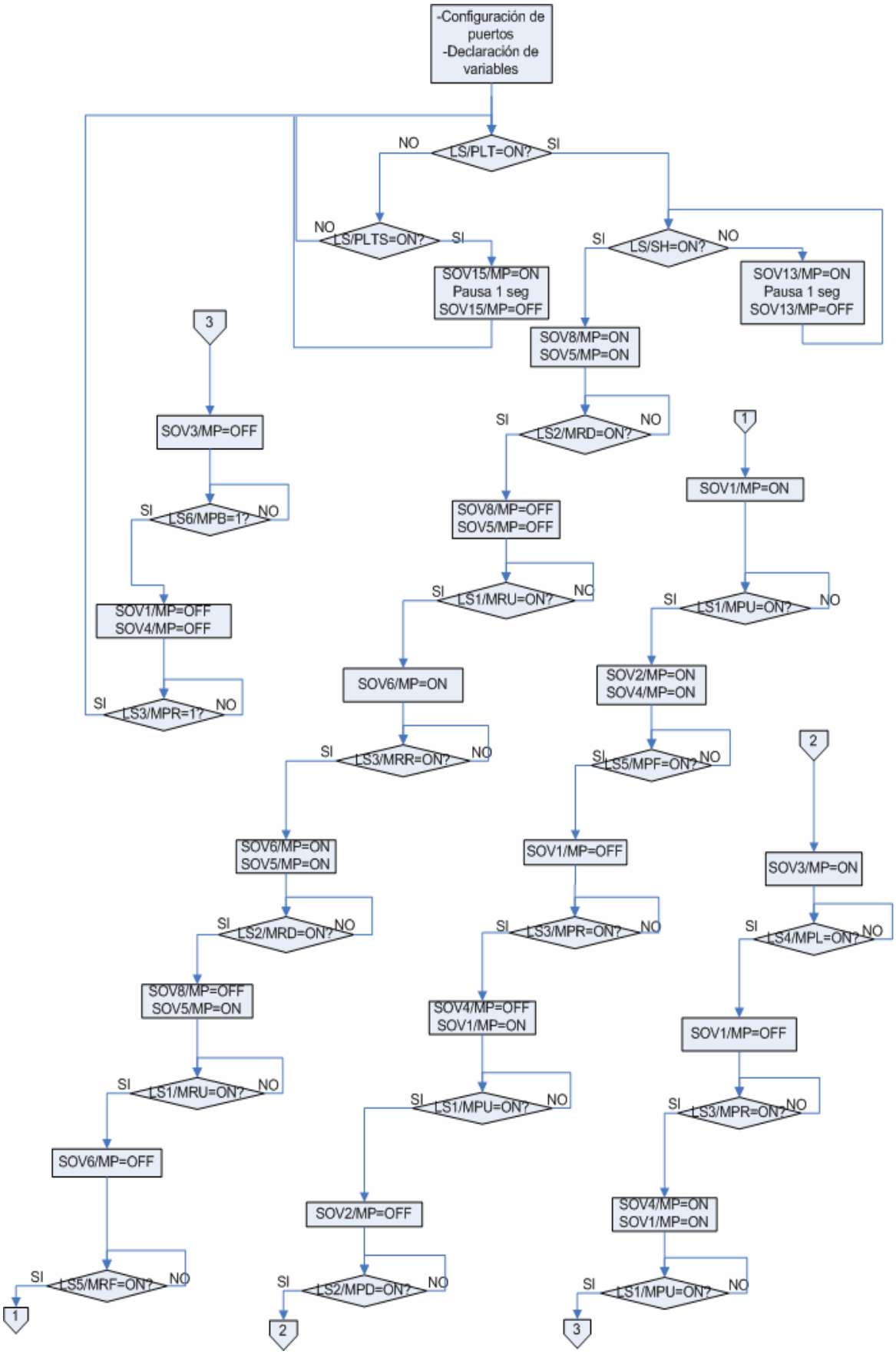


Figura. 6.6. Diagrama de Flujo Práctica No.1.1.

Código de programa en anexo IV

2) Actuadores temporizados.

Aquí las válvulas son activadas y desactivadas en un instante posterior al tiempo aproximado que les tomaría a los brazos neumáticos realizar un movimiento completo en dirección de uno de sus grados de libertad. La Figura. 6.7 ilustra la lógica utilizada en esta práctica.

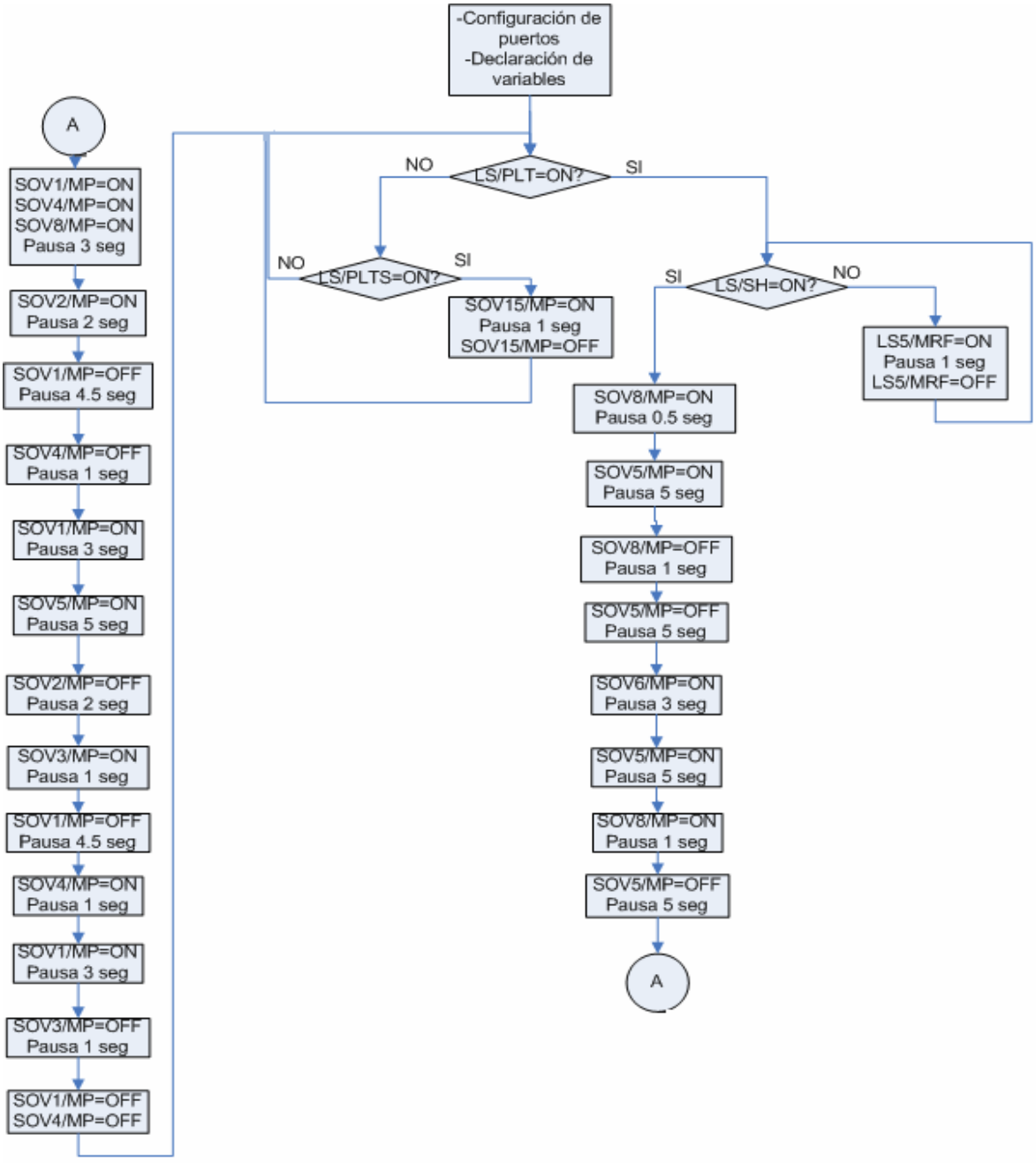


Figura. 6.7. Diagrama de Flujo Práctica No.1.2.

Código de programa en anexo IV

6.3 PRÁCTICA No.2 EL PROTOCOLO DE COMUNICACIÓN MODBUS

Esta práctica tiene como fin implementar en el sistema de control las funciones básicas del protocolo Modbus. Con lo cual se hace posible el manejo de la estación PN-2800 mediante HMI.

La Tabla. 6.12 muestra los parámetros establecidos para esta práctica.

Tabla. 6.12. Parámetros Modbus.

Dirección esclavo	1
Data Rate [bits/seg.]	2400
paridad	Sin paridad
Bits stop	1
Operación	Modbus serial
Data Bits	8

El siguiente diagrama de flujo (Figura. 6.8) muestra la lógica utilizada por el programa destinado a esta práctica.

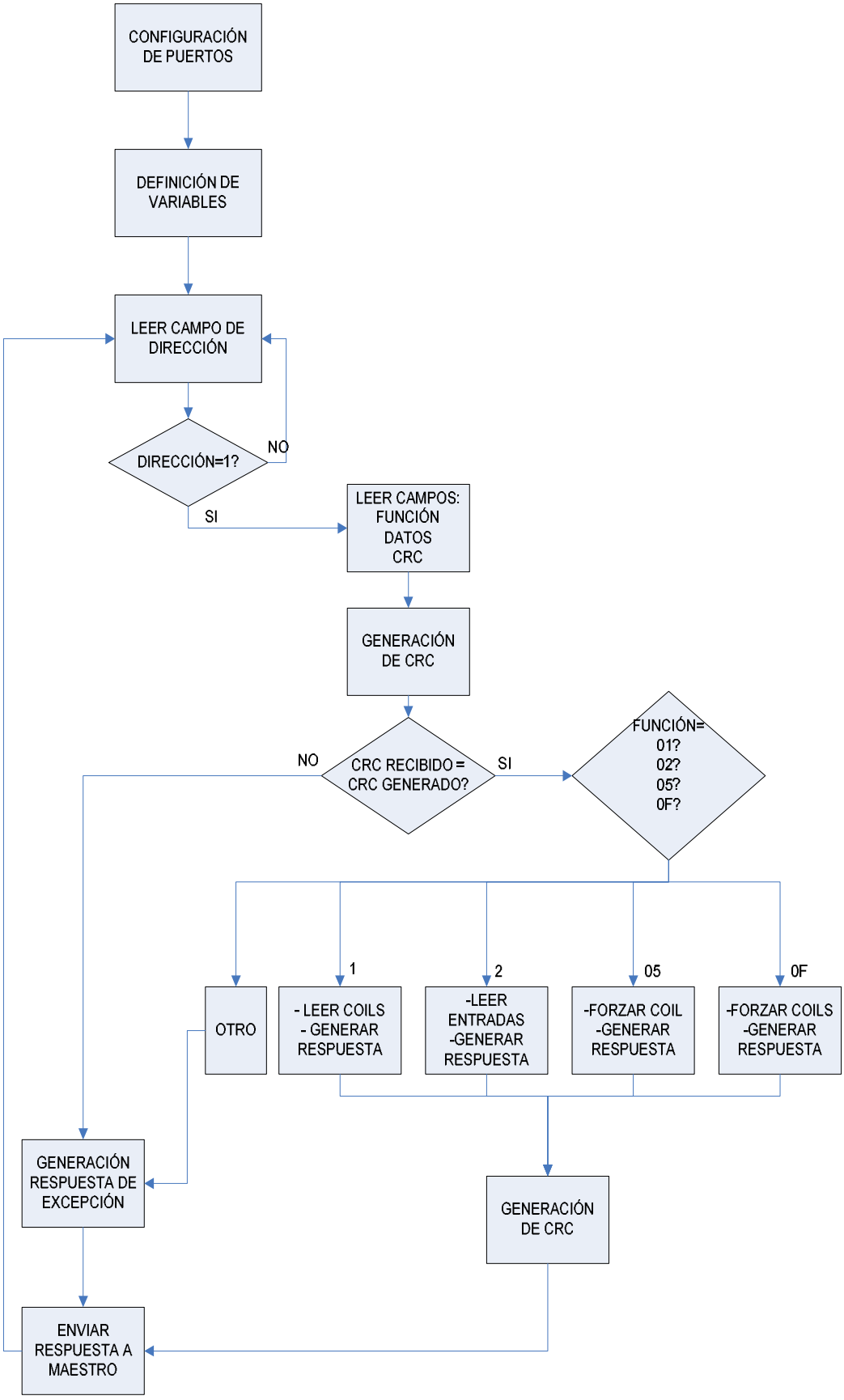


Figura. 6.8. Diagrama de Flujo Práctica No.2.

Código de programa en anexo IV

La comprobación de esta práctica se la puede hacer con Lookout o con el examinador del protocolo modbus Mbus.

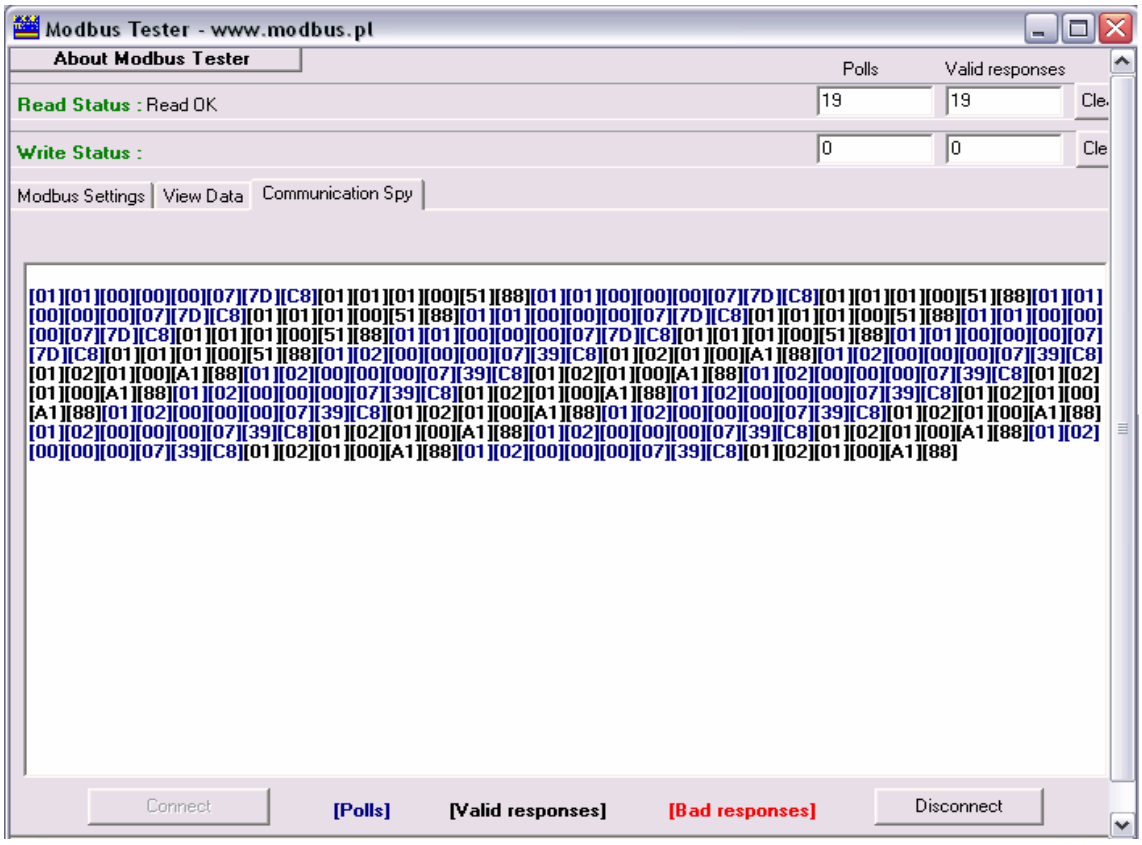


Figura. 6.9. Comprobación de la práctica con Modbus Tester.

En las primeras expresiones hexadecimales en azul de la Figura.6.9 se puede apreciar que el programa Mbus envía al microcontrolador la siguiente trama:

[01][01][00][00][00][07][7D][C8]

Según la estructura de las tramas modbus ([esclavo] [función] [datos] [CRC]), esto significa que el master desea saber el estado de las 8 primeras salidas discretas del dispositivo esclavo No.1.

[01]→ Esclavo No 1
[01]→ Función No 1 (leer salidas discretas)
[00][00]→ empezar desde salida 00001
[00][07]→ finalizar en la salida 00008
[7D][C8]→ CRC

Las primeras expresiones en negro en la Figura.6.9 son la trama que el microcontrolador envía como respuesta a la petición del maestro.

[01][01][01][00][51][88]

Según la siguiente interpretación, esto significa que todas las 8 primeras salidas del dispositivo esclavo No.1 están a nivel lógico de “0”.

[01]→ Esclavo No 1

[01]→ Función No 1 (leer salidas discretas)

[01]→ # 1 byte de datos

[00]→ Estado de 8 primeras entradas en hexadecimal

[51][88]→ CRC

El color negro de la trama que el microcontrolador envía como respuesta al programa Mbus, indica que el microcontrolador está enviando al dispositivo maestro tramas válidas y sin error. Si existiera algún tipo de error en las respuestas, estas tendrían una coloración roja.

En subsiguientes tramas de pregunta en la misma Figura.6.9 se puede ver que estas se hallan involucradas con la función 02hex (leer entradas discretas), y que para todas ellas el microcontrolador envía una correspondiente válida respuesta.

Para la comprobación en Lookout 5.1 se procede a crear un switch que está conectado a una expresión, se puede ver en la Figura. 6.10 que el switch está en posición de OFF y en la Figura. 6.11 el switch cambia de posición a ON y se activa la bobina 00001 con su indicador.

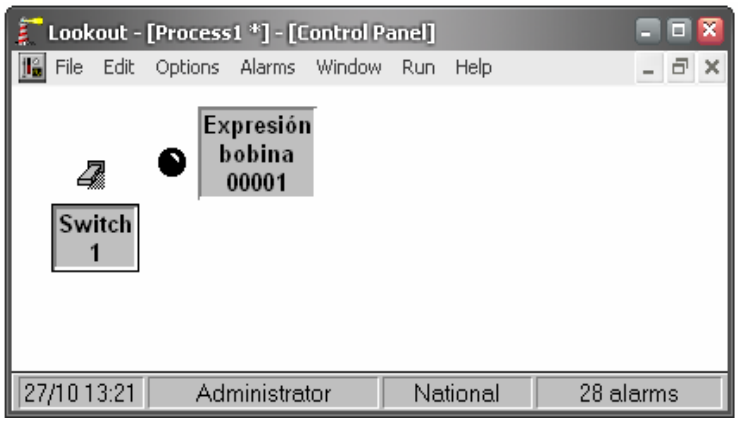


Figura. 6.10. Panel principal con Switch Off.

Figura. 6.11. Panel principal con Switch On.

6.4 PRÁCTICA No.3 CONTROL DESDE PC A TRAVÉS DE INTERFACE HOMBRE – MÁQUINA HMI

La estación neumática PN-2800 maneja la parte encargada de entregar materia prima. Como ya se ha explicado con anterioridad, esta estación es manejada por medio de un PC y el sistema propuesto por comunicación Modbus. Para la implementación de la interface, se utilizó el software Lookout 5.1 versión de desarrollo. Esta herramienta permite desarrollar una interface capaz de monitorear y controlar la estación en cuestión.

Para el desarrollo de la interface se empieza por definir el proceso que se va a realizar, para esto es necesario abrir Lookout 5.1 de la siguiente manera: **Inicio\Programas\National Instruments Lookout 5.1** (Figura. 6.12.).

Figura. 6.12. Software seleccionado para la interface HMI.

Una vez abierto, se procede a determinar el proceso, en este caso, el proceso es pn2800.l4p que es el correspondiente a la estación neumática (Figura. 6.13.).

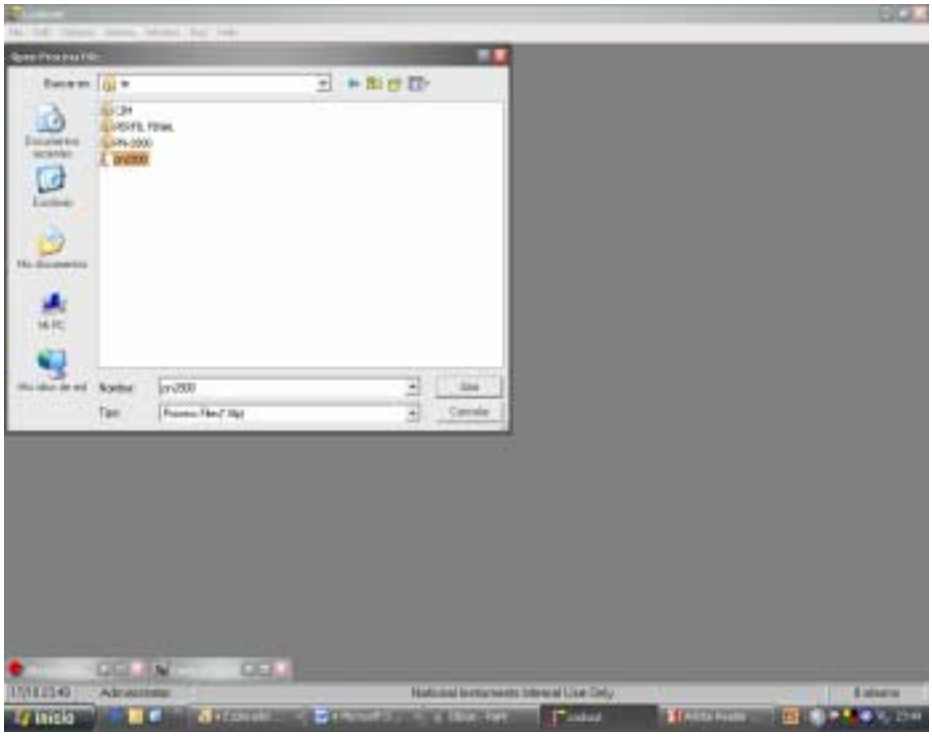


Figura. 6.13. Abrir programa PN-2800.

6.4.1 Panel de Control.

Figura. 6.14. Panel de control del HMI.

En el Panel de control (Figura. 6.14.) se aprecia el nombre de la Escuela, Facultad, especialización, sello de la Facultad y el nombre de la estación. El panel está dividido en dos partes:

- Manipulador de cilindros.
- Manipulador de pallets.

6.4.2 Manipulador de Cilindros.

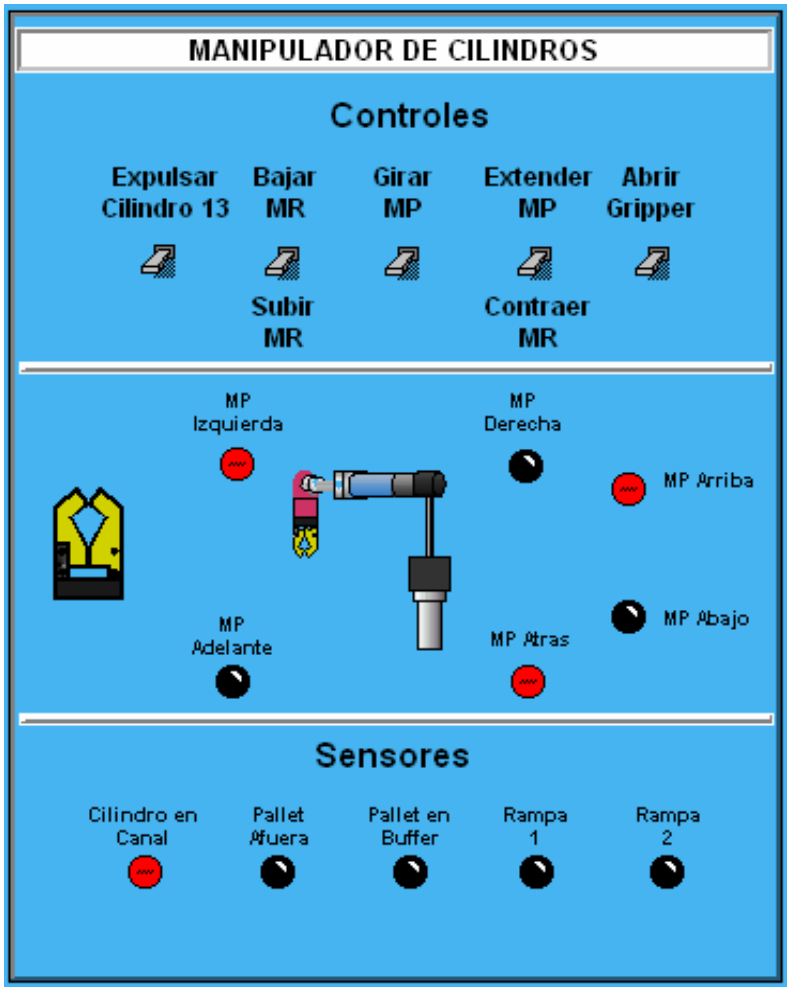


Figura. 6.15. Control y monitoreo Manual para el Manipulador de Cilindros.

El manipulador de cilindros es el encargado de entregar material hacia el manipulador de pallets, en la Figura. 6.15. se encuentran tres secciones, la primera sección muestra la manipulación del brazo neumático encargado de la entrega del material y manejo en forma manual del manipulador de cilindros mediante switchs, estos moverán el brazo de acuerdo a los requerimientos del usuario, una segunda sección encargada del monitoreo del área en cuestión con multiestado para observar el cambio de posición del brazo robot y la última sección que se encarga del monitoreo de los sensores y nos muestra la información correspondiente a la presencia o no de cilindros en el canal 1, en las dos rampas que acumulan el material en bruto, así como también si existe pallets en el almacén ó en el buffer de salida.

6.4.3 Manipulador de Pallets.

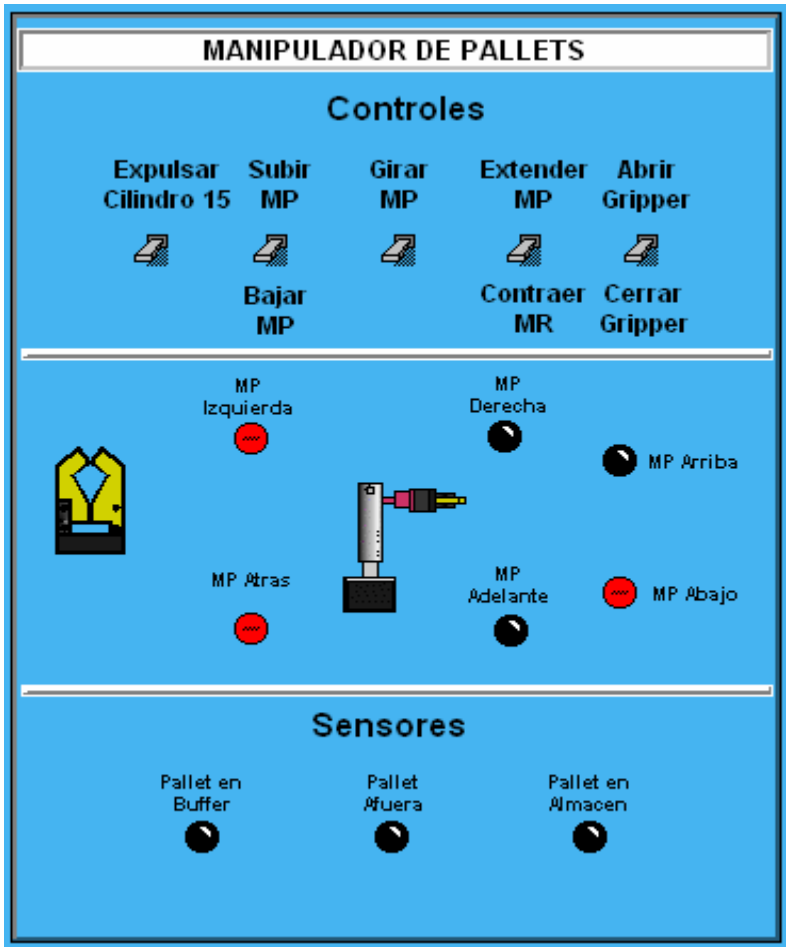


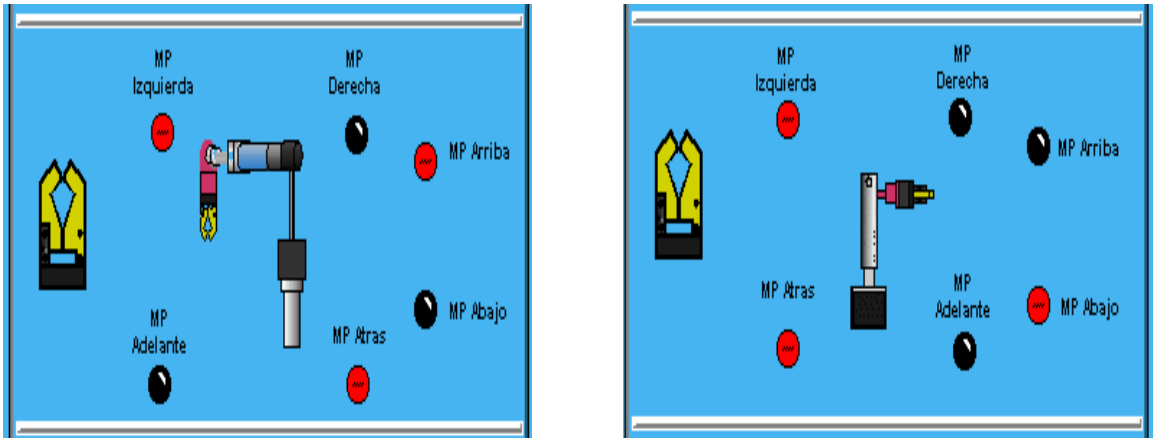
Figura. 6.16. Control y monitoreo Manual para el Manipulador de Pallets.

El manipulador de pallets es el encargado de entregar material y viceversa, en esta parte del panel de control se encuentran tres secciones, una que muestra la manipulación del brazo neumático encargado de la entrega de material desde la estación y manejo en forma manual del manipulador de pallets, esta moverá el brazo de acuerdo a los requerimientos del usuario, una segunda sección encargada del monitoreo del área en cuestión, con multiestado para observar el cambio de posición del brazo robot, y la última sección que se encarga del monitoreo de los sensores y nos muestra la información correspondiente a la presencia o no de pallets en el almacén, fuera del almacén ó en el buffer de salida como se muestra en la Figura. 6.16.

6.4.4 Ciclo semi-automático.

Figura. 6.17. Control y monitoreo Semi-automático para el Manipulador de Cilindros y Pallets.

El Panel de Control ciclo semi-automático (Figura. 6.17.) se encarga de realizar procesos sin necesidad de la manipulación manual del usuario. La estación cuenta con 2 semi-ciclos que se encargan de realizar la entrega de material en forma parcial o total. El ambiente en cuestión se conforma de 2 partes, una primera sección que acciona cada semi-ciclo como el manipulador de cilindros o manipulador de pallets y una segunda sección de monitoreo del material en toda la estación como se muestra en la Figura. 6.18.



a) b)

Figura. 6.18. Monitoreo semi-automático a) Manipulador de Cilindros; b) Manipulador de Pallets.

Para los ciclos semi-automáticos debemos tener en cuenta las siguientes consideraciones de la Tabla.6.13.

Tabla. 6.13. Condiciones de trabajo para el modo semi-automático.

Ciclo Manipulador de cilindros:	DEBE EXISTIR: PALLET AFUERA DEL ALMACÉN. CILINDRO EN EL CANAL
Ciclo Manipulador de pallets:	DEBE EXISTIR: PALLET EN EL ALMACÉN. PALLET AFUERA DEL ALMACÉN.

CAPÍTULO VII

ANÁLISIS COSTO – BENEFICIO

7.1 OBJETIVO

El objetivo general de este estudio es el de conocer en profundidad cuanto nos cuesta en términos financieros y de trabajo, diseñar e implementar un Controlador Programable para la Estación Neumática PN-2800 en relación con un Controlador Lógico Programable Modicon AEG 984 de características semejantes al Proyecto que hemos realizado.

7.2 ANÁLISIS COSTO – BENEFICIO

El objetivo del proyecto es la implementación de un sistema sencillo y económico para la programación de rutinas de control de la estación neumática PN-2800. Estas rutinas de control serán cargadas en la memoria de programa del microcontrolador. El microcontrolador ejecutará las rutinas programadas y a través de sus puertos de salida comandará electro-válvulas para ejecutar las rutinas de la estación neumática.

Los beneficios del sistema se pueden observar desde el comienzo de su diseño e implementación, con un ahorro de aproximadamente $400.00.

7.2.1 Sistema Actual.

La siguiente es una descripción de las características y costos (Tabla. 7.1) del sistema actual basado con un PLC Modicon AEG 984 (Figura. 7.1) durante los anteriores años en la Estación Neumática PN-2800.

7.2.1.1 Características PLC Modicon.

Figura. 7.1. PLC Modicon Compact CPU: 984-145.

Tabla. 7.1. Características del Controlador del Sistema Actual.

Características PLC	**PLC Modicon Compact CPU: 984-145**
Entradas y Salidas	24 Entradas discretas a 24VDC 20 Salidas discretas a 24VDC
Total User Memory	8K instrucciones (words)
Total State Ram	2K instrucciones (words)
Memoria Total	106K instrucciones (words) Slot para insertar memoria EEPROM de 8 a 16 bytes
Scan Time	4,25ms…..6ms/K nodos de instrucción. 8ms…..11ms para 64 I/O y 1K memoria lógica
Comunicaciones	1 Modbus y Modbus Plus, puerto RS-232C
Condiciones de operación	0 – 60 ºC Humedad de 0 a 93% no condensado
Resistencia a vibraciones	Transitoria 2KV en la fuente de alimentación y I/O Ringwave 2,5KV en la fuente de alimentación y I/O
Descargas estáticas	+/- 8KV aire, 10 descargas +/- 4KV contacto, 10 descargas
Condiciones de almacenamiento	40 a 85ºC Humedad de 0 a 93% no condensado
Software	Modicon State Language, ProWORX, Concept
Costo Unitario	**$ 499**

7.2.2 Sistema Propuesto.

El siguiente es un detalle de las características y costos (Tabla. 7.2.), que se incurrió durante la implementación del nuevo sistema (Figura. 7.2) basado en Microcontrolador 16F877, describiendo cada uno de sus elementos con sus respectivos precios (Tabla. 7.3.).

7.2.2.1 Características PICmicro 16F877.

Figura. 7.2. Sistema propuesto.

Tabla. 7.2. Características del Controlador del Sistema Propuesto.

Características PICmicro ™	**PIC16F877**
Frecuencia operativa	DC - 20 MHz
RESETS (y Delays)	POR, BOR (PWRT, OST)
Memoria de programa FLASH (Palabras de 14 bits)	8K
Memoria de datos RAM (bytes)	368
Memoria de datos de EEPROM (bytes)	256
Interrupciones	14
Puertos de E/S	Ports A,B,C,D,E
Timers	3
Módulos de captura / comparación / PWM	2
Comunicaciones Seriales	MSSP, USART
Comunicaciones paralelas	PSP
10-bit Módulo de análogo - Digital	8 canales de entrada
Conjunto de instrucciones	35 instrucciones

Tabla. 7.3. Lista de Elementos del Sistema Propuesto.

CANT.	DESCRIPCIÓN	V. UNIT	V.TOTAL	
1	PIC 16F877	10	10	INTEGRADOS
1	74LS04	0,45	0,45	
1	MAX 232	0,5	0,5	
4	74LS244	0,6	2,4	
1	Zócalo 40 PINES	1,3	1,3	ZÓCALOS
1	Zócalo 14 PINES	0,1	0,1	
1	Zócalo 16 PINES	0,1	0,1	
4	Zócalo 20 PINES	0,12	0,48	
1	DIODO LED	0,07	0,07	UNITARIOS
1	MICRO PULSADOR NA 2P	0,12	0,12	
1	CONECTOR Db-9 MACHO P/PLACA	0,36	0,36	
1	FUSIBLE EUROPEO 0,8 A	0,04	0,04	
1	PLACA	51	51	
1	CRISTAL 4 MHZ	0,9	0,9	CRISTAL
2	CONDENSADORES 22PF 50V	0,08	0,16	
26	BORNERAS 2 CONTACTOS AZUL	0,45	11,7	BORNERAS
16	RESISTENCIAS 4.7 KΩ	0,05	0,8	ENTRADAS
16	ECG TRANSISTOR 123A	0,1	1,6	
16	DIODO RECTIFICADOR 1A, 600V	0,05	0,8	
32	DIODO ZENER 5,1 V 1W	0,12	3,84	SALIDAS
32	RESISTENCIAS 1/2W 1KΩ	0,02	0,64	
32	RESISTENCIAS 82 KΩ	0,02	0,64	
32	RESISTENCIAS 10 KΩ	0,02	0,64	
1	LM350	4	4	FUENTE
1	CONDENSADORES 0,1uF 50V	0,06	0,06	
1	CONDENSADORES 1uF 50V	0,06	0,06	
1	RESISTENCIA 1W 120Ω	0,09	0,09	
1	POTENCIÓMETRO 5KΩ	0,15	0,15	
5	CONDENSADORES 1uF 50V	0,06	0,3	MAX 232
1	RESISTENCIAS 4.7 KΩ	0,05	0,05	
1	CARCASA	16,65	16,65	CARCASA
		TOTAL	USD 110	

- **Costo Total: $110 USD.**

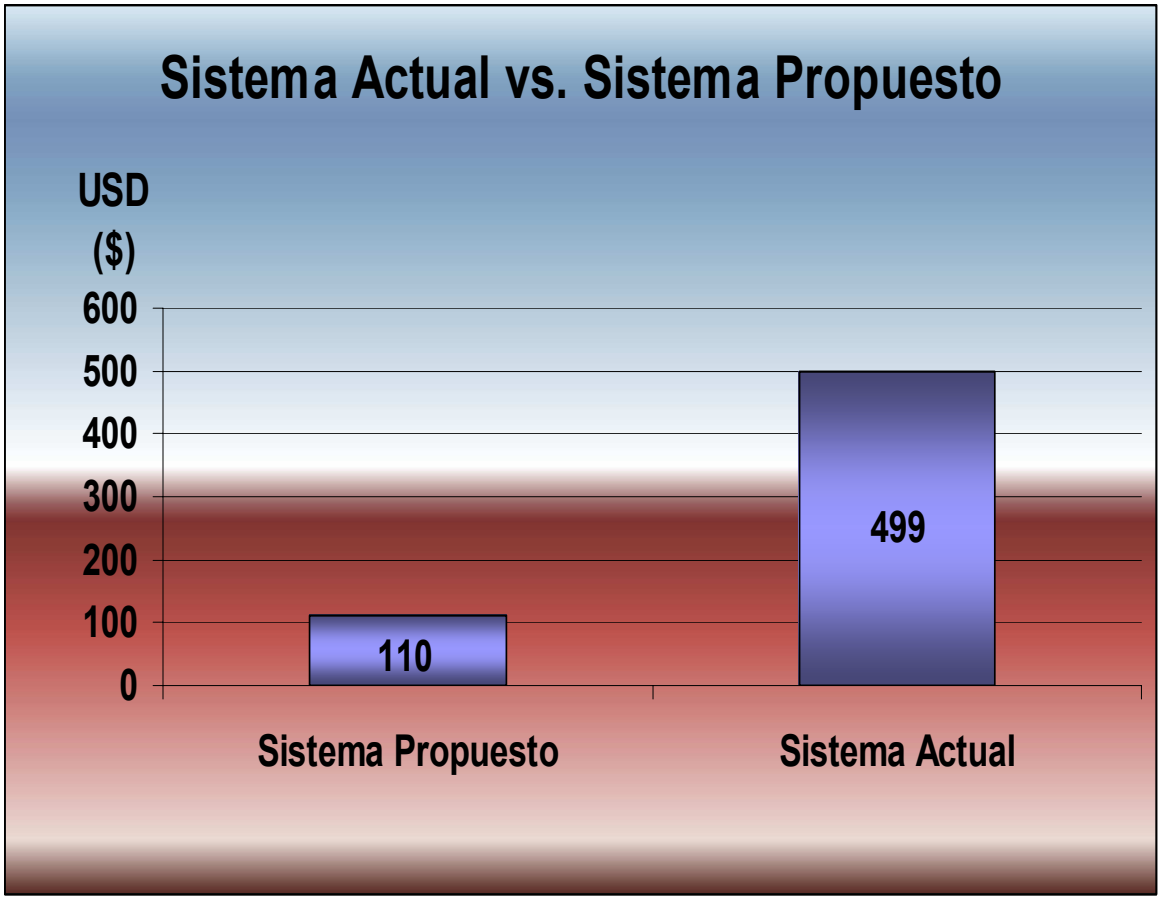


Figura. 7.3. Costo del Sistema Actual vs. Sistema Propuesto.

La Figura. 7.3 representa la comparación de los costos de los sistemas y demuestra la reducción de costo del sistema propuesto. Se debe notar, sin embargo, que no se ha considerado en el análisis el costo de diseño y fabricación del prototipo. Frente a este aspecto tampoco se ha considerado el beneficio económico obtenido al liberar el PLC Modicon para realización de prácticas.

CAPÍTULO VIII

CONCLUSIONES Y RECOMENDACIONES

8.1 CONCLUSIONES

- Con elementos generales y de relativo bajo costo se ha logrado construir un sistema de control que cubre con los requerimientos de potencia, autonomía y comunicación de la estación PN-2800.

- Con ciertos cambios en el hardware y la apropiada configuración y programación en el microcontrolador del sistema propuesto de control, es posible disponer no solo de entradas y salidas discretas, sino también de entradas y salidas análogas.

- El estudio e implementación del protocolo Modbus modo RTU, nos ha brindado técnicas de software que garantizan alta integridad en la comunicación. Lo cual sin duda aumentará la confiabilidad de nuestros futuros proyectos.

- El sistema implementado es flexible a nivel de software, puesto que el microcontrolador utilizado en este proyecto es muy popular y existe en el mercado una diversidad de programas utilizados en la elaboración de programas para la mayoría de microcontroladores PIC.

- Este equipo está limitado a comunicarse únicamente con una PC, si se deseara conectar a una red de dispositivos esclavos se debería previamente dotar al equipo con un hardware diseñado para ello. Como por ejemplo hardware RS-485.

- Al cargar el programa PLC_2800.hex en el microcontrolador del sistema de control, el control de la estación PN-2800 es posible únicamente mediante un HMI. Si se desea dar autonomía y comunicación Modbus a la vez, es preciso trabajar en funciones Modbus (23, 03) que el sistema propuesto actualmente no dispone.

8.2 RECOMENDACIONES

- No cambiar arbitrariamente la configuración de los puertos del microcontrolador, ya que esto conllevaría al daño del hardware del dispositivo de control.

- Realizar prácticas apoyándose primeramente en el contenido de los ejemplos de rutinas básicas de programación en el capítulo de prácticas, anexos y manual de usuario.

- No realizar conexiones, desconexiones o cambio en el hardware del equipo de control mientras este se encuentra trabajando.

- Antes de realizar cambios en el firmware del microcontrolador, es recomendable que la estación PN-2800 se halle en estado inactivo.

REFERENCIAS BIBLIOGRÁFICAS

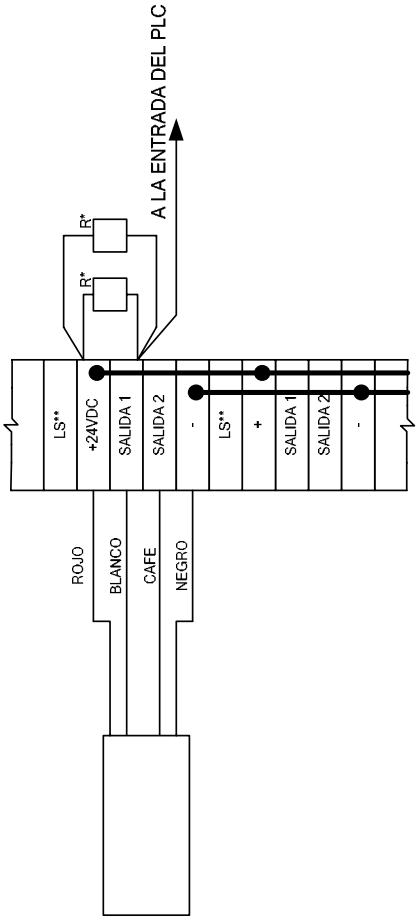
- ORTIZ, Hugo R., Instrumentación y Sistemas de Control, 1998.
- DEGEM SYSTEM, Curso PN-2800, Manual del Estudiante, 1994.
- DEGEM SYSTEM, Curso PN-2800, Manual del Instructor, 1994.
- SMITH, Carlos A., Control Automático de Procesos, 1997.
- www.microchip.com
- www.modbus.com
- www.ni.com

ANEXOS

ANEXO I

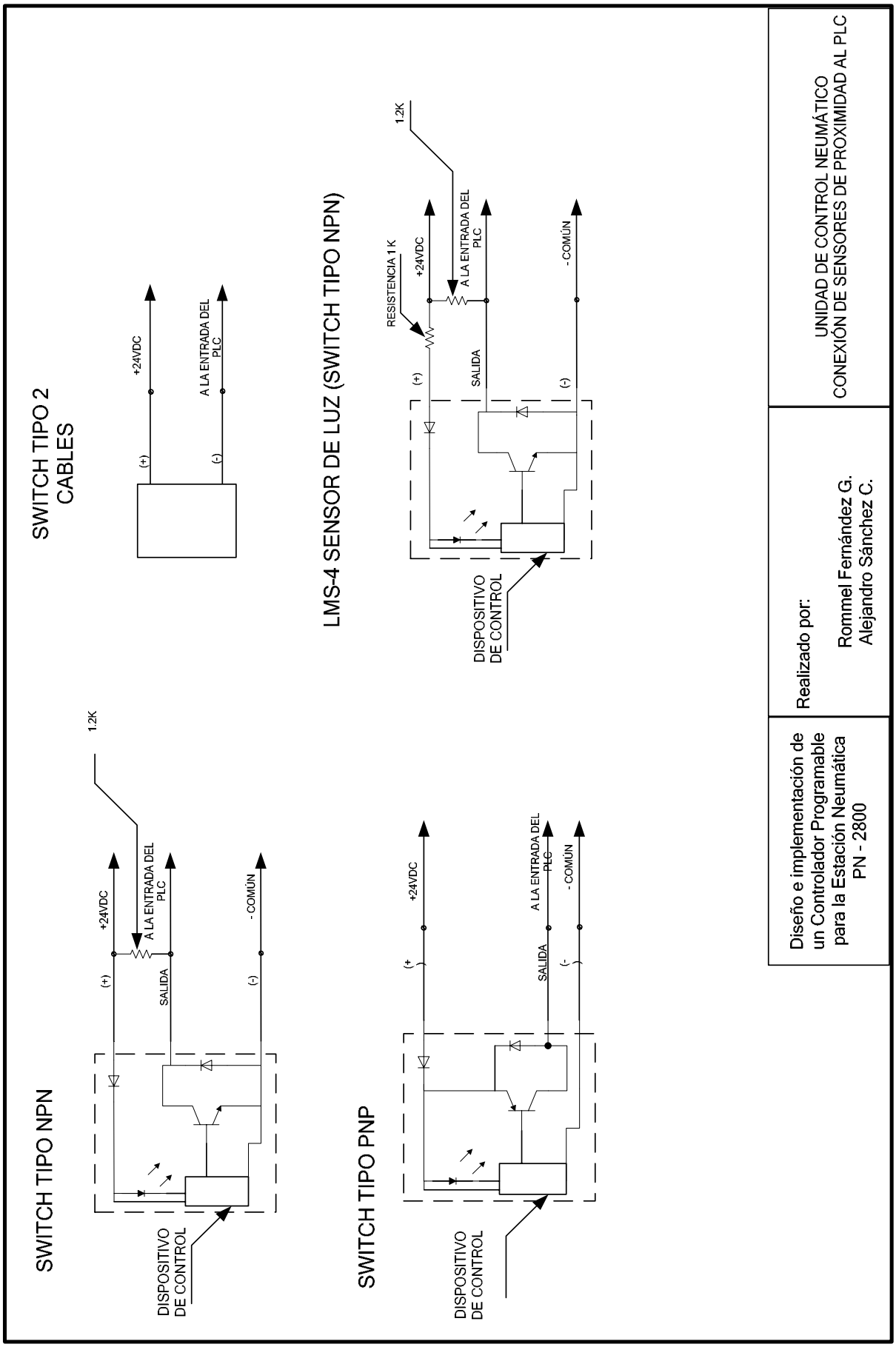
DIAGRAMA ELÉCTRICO Y NEUMÁTICO DE LA ESTACIÓN NEUMÁTICA PN-2800

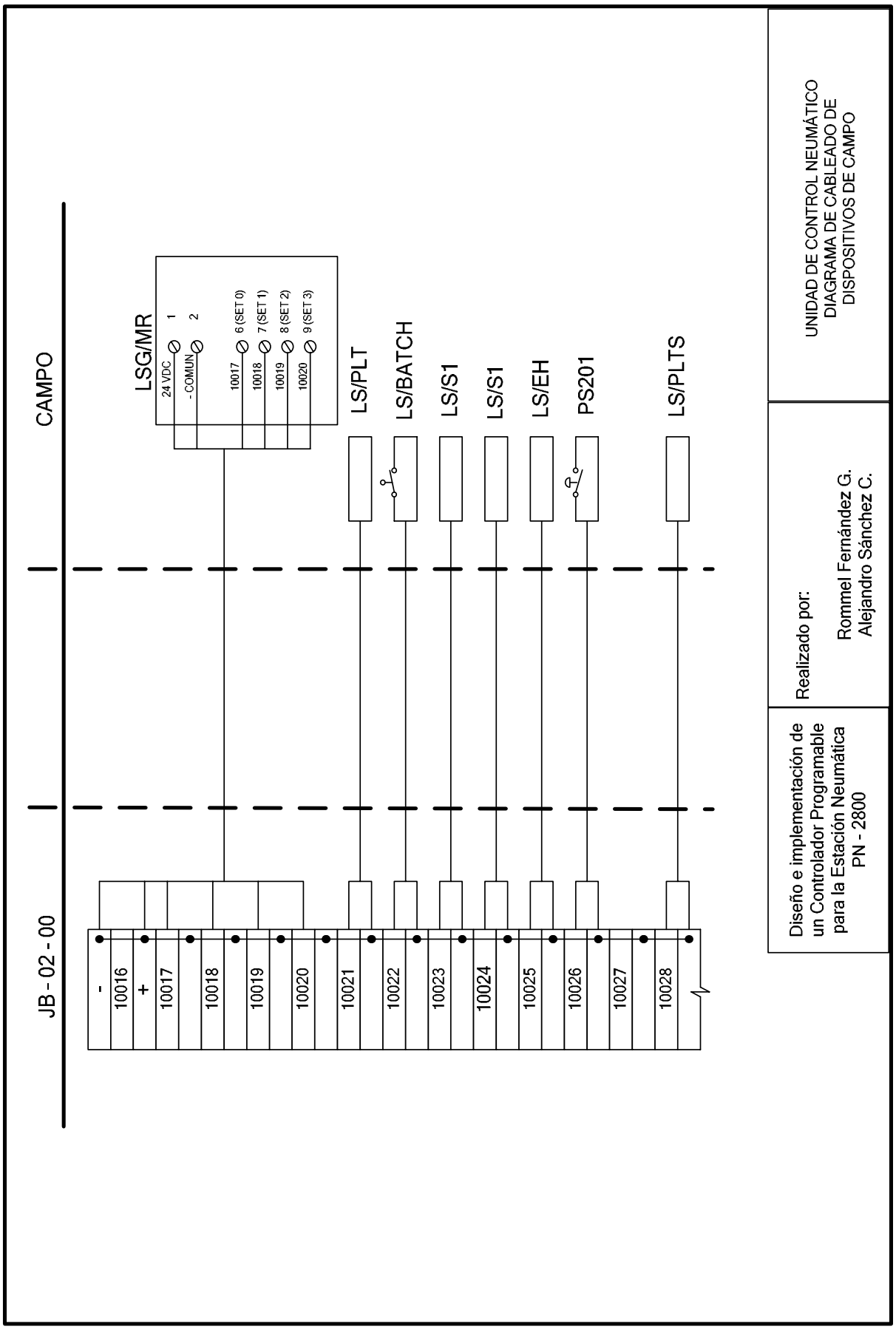
EZ - 8M, EZ - 16M, SENSORES DE PROXIMIDAD

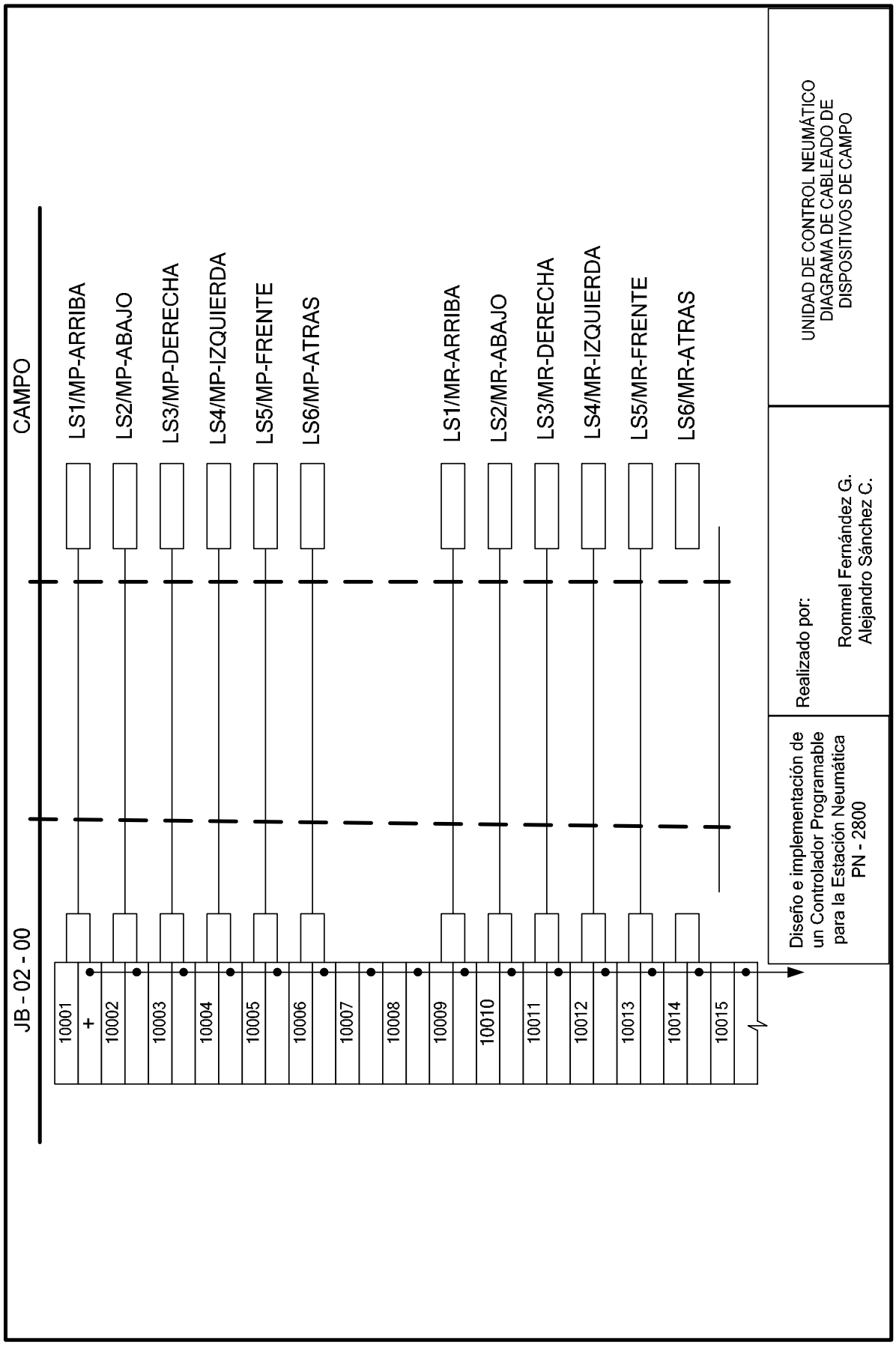


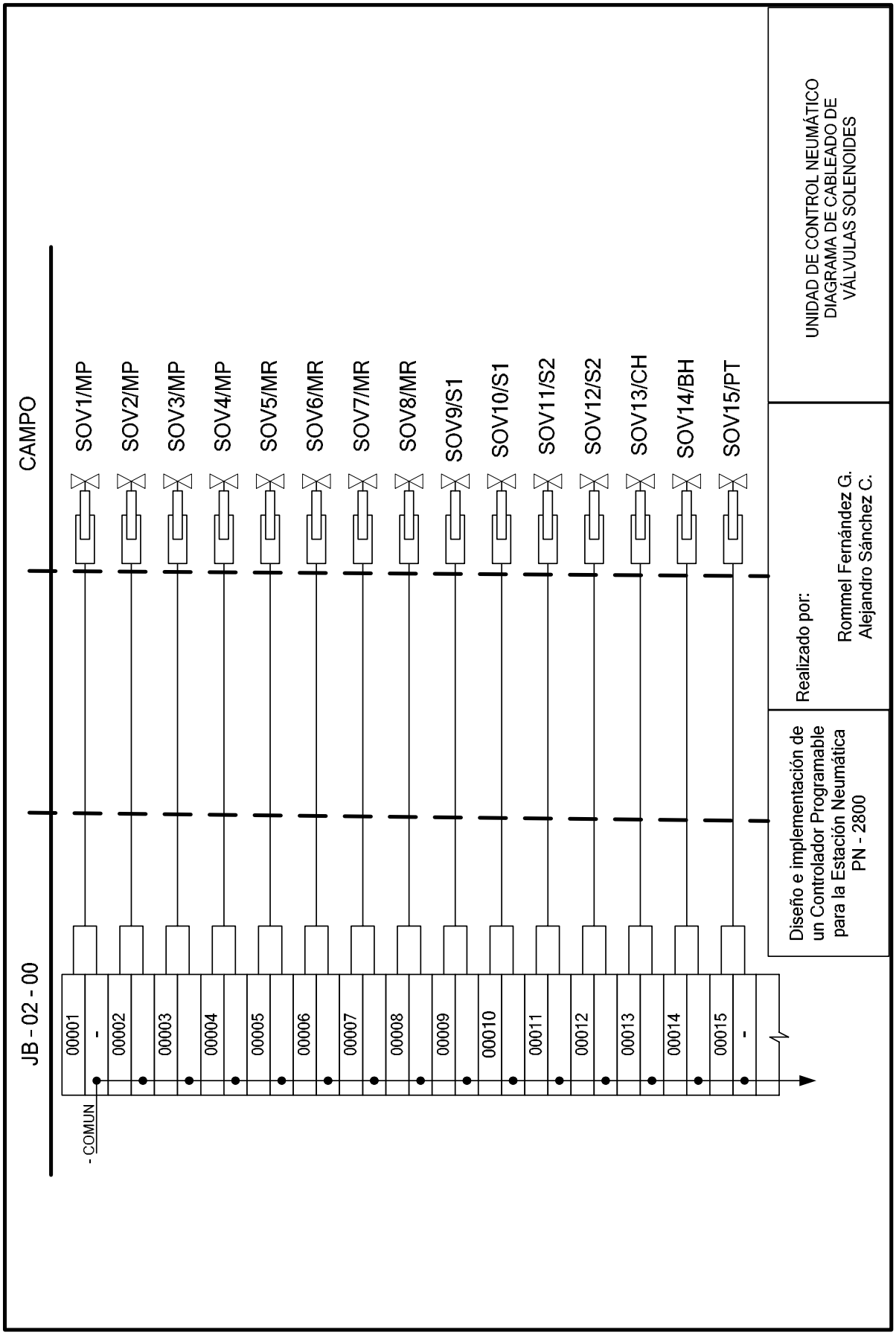
R* RESISTOR DE 4,7K
** NUMERO DEL SWITCH

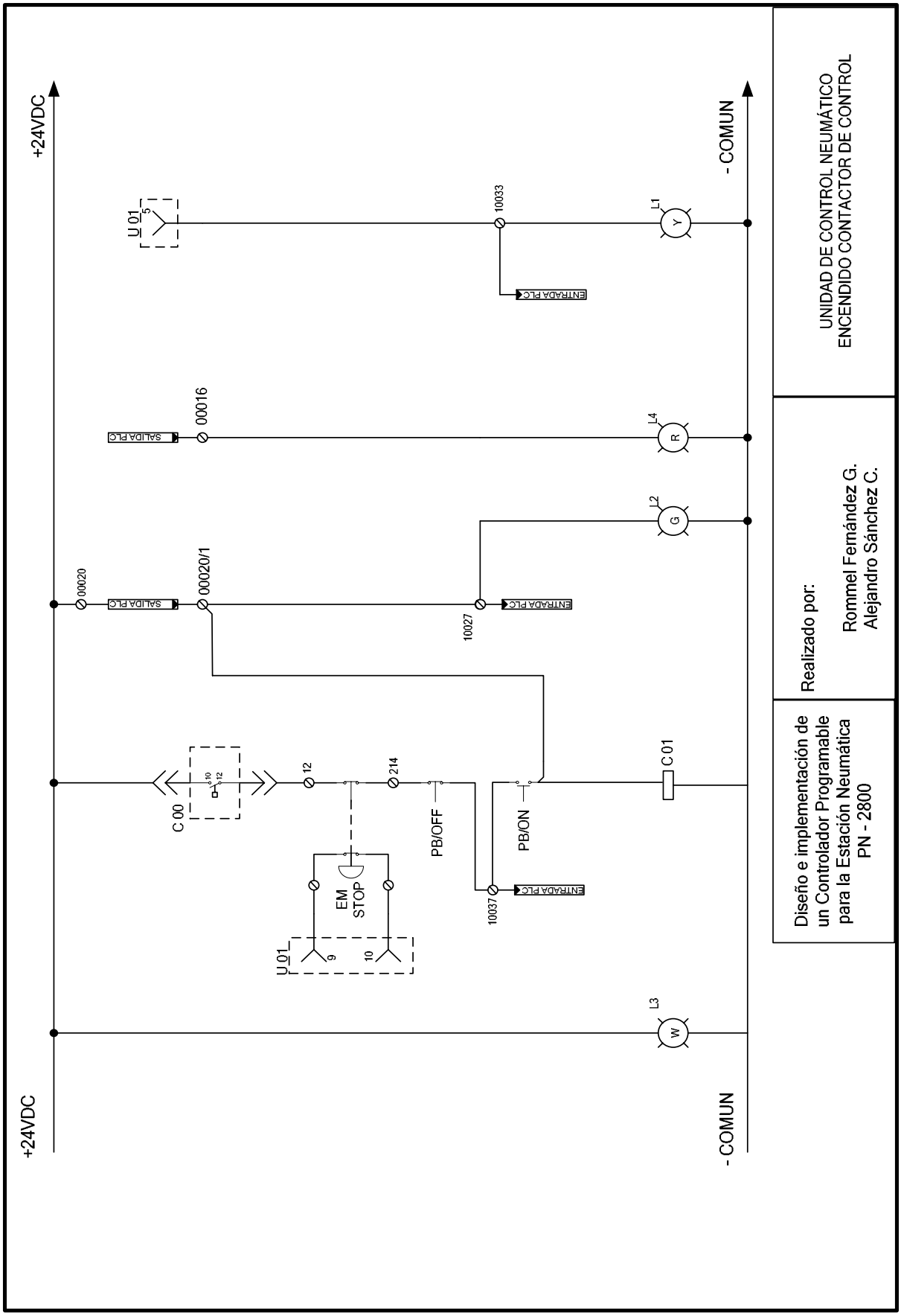
Diseño e implementación de un Controlador Programable para la Estación Neumática PN - 2800	Realizado por: Rommel Fernández G. Alejandro Sánchez C.	UNIDAD DE CONTROL NEUMÁTICO CONEXIÓN DE SENSORES DE PROXIMIDAD

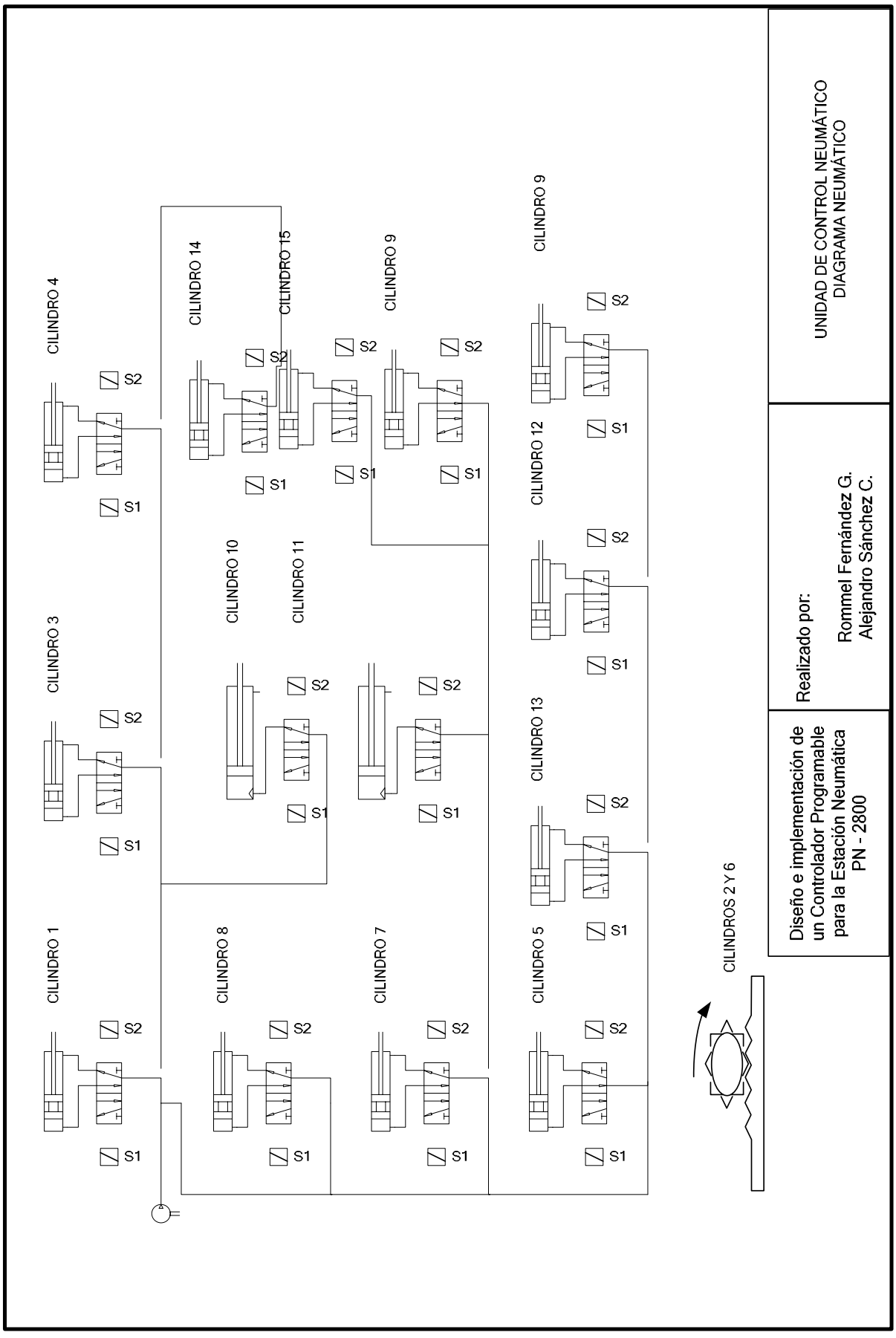

SWITCH TIPO 2
CABLES
+24VDC
A LA ENTRADA DEL PLC
(+)
(-)
LMS-4 SENSOR DE LUZ (SWITCH TIPO NPN)
1.2K
RESISTENCIA 1 K
+24VDC
A LA ENTRADA DEL PLC
SALIDA
(+)
(-)
- COMÚN
DISPOSITIVO DE CONTROL
SWITCH TIPO NPN
1.2K
+24VDC
A LA ENTRADA DEL PLC
SALIDA
(+)
(-)
- COMÚN
DISPOSITIVO DE CONTROL
SWITCH TIPO PNP
+24VDC
A LA ENTRADA DEL PLC
SALIDA
(+)
(-)
- COMÚN
DISPOSITIVO DE CONTROL
UNIDAD DE CONTROL NEUMÁTICO
CONEXIÓN DE SENSORES DE PROXIMIDAD AL PLC
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de un Controlador Programable para la Estación Neumática PN - 2800

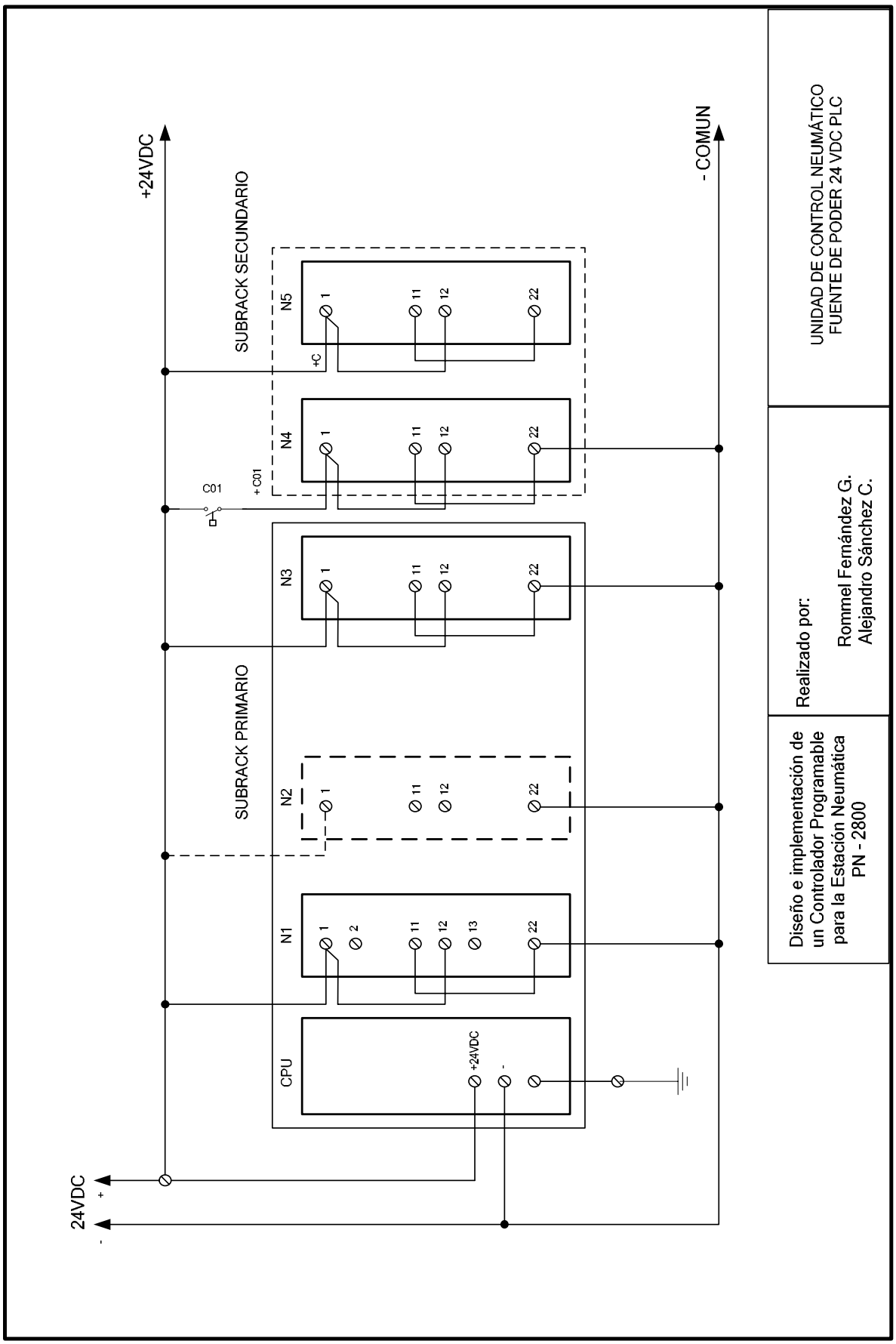

CAMPO
JB - 02 - 00
LSG/MR
24 VDC
- COMUN
1
2
10017
10018
10019
10020
6 (SET 0)
7 (SET 1)
8 (SET 2)
9 (SET 3)
LS/PLT
LS/BATCH
LS/S1
LS/S1
LS/EH
PS201
LS/PLTS
-
+
10016
10017
10018
10019
10020
10021
10022
10023
10024
10025
10026
10027
10028
UNIDAD DE CONTROL NEUMÁTICO
DIAGRAMA DE CABLEADO DE
DISPOSITIVOS DE CAMPO
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de
un Controlador Programable
para la Estación Neumática
PN - 2800

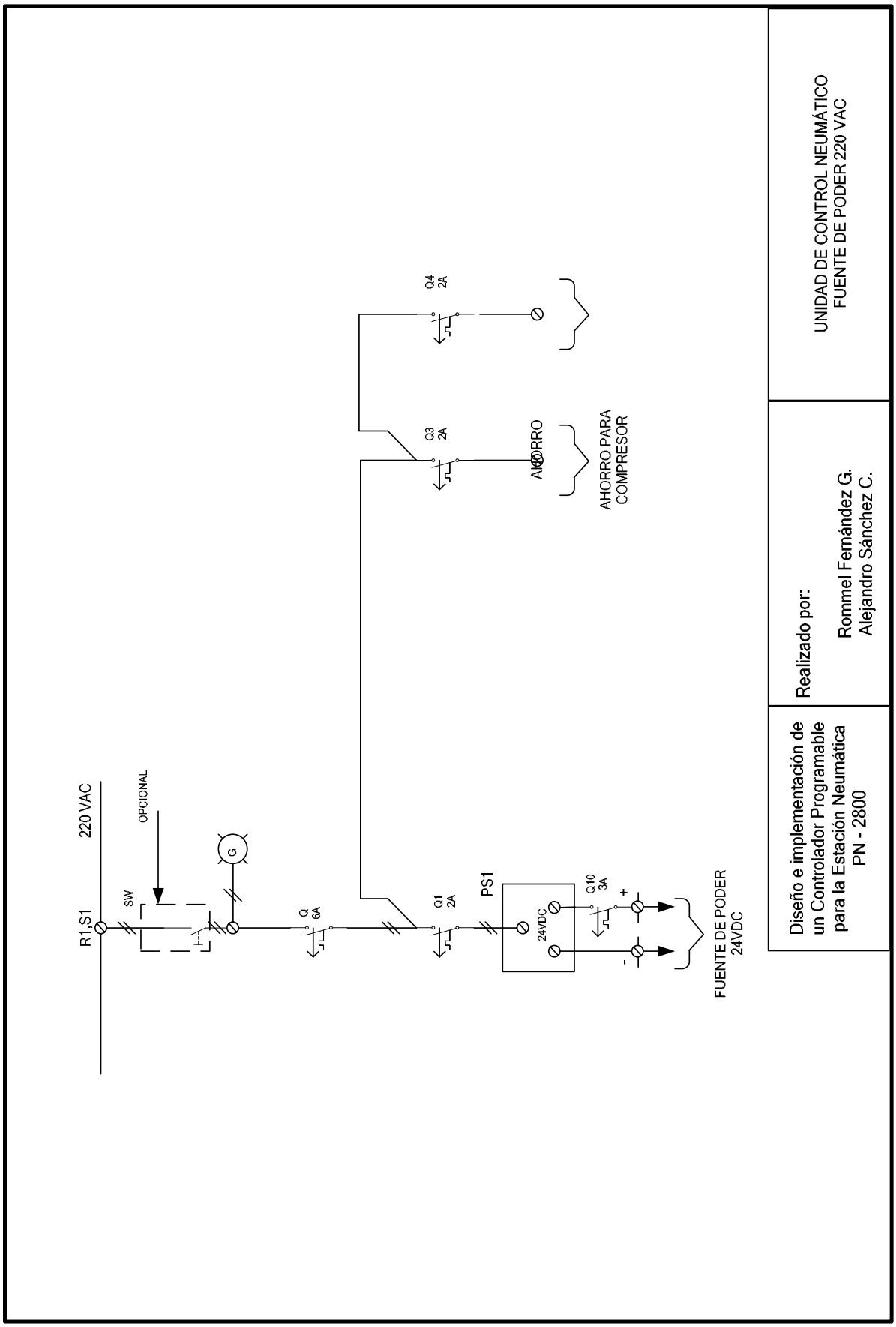

CAMPO
JB - 02 - 00
LS1/MP-ARRIBA
LS2/MP-ABAJO
LS3/MP-DERECHA
LS4/MP-IZQUIERDA
LS5/MP-FRENTE
LS6/MP-ATRAS
LS1/MR-ARRIBA
LS2/MR-ABAJO
LS3/MR-DERECHA
LS4/MR-IZQUIERDA
LS5/MR-FRENTE
LS6/MR-ATRAS
10001
+
10002
10003
10004
10005
10006
10007
10008
10009
10010
10011
10012
10013
10014
10015
UNIDAD DE CONTROL NEUMÁTICO
DIAGRAMA DE CABLEADO DE
DISPOSITIVOS DE CAMPO
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de
un Controlador Programable
para la Estación Neumática
PN - 2800


JB - 02 - 00
CAMPO
- COMUN
00001
-
00002
00003
00004
00005
00006
00007
00008
00009
00010
00011
00012
00013
00014
00015
-
SOV1/MP
SOV2/MP
SOV3/MP
SOV4/MP
SOV5/MR
SOV6/MR
SOV7/MR
SOV8/MR
SOV9/S1
SOV10/S1
SOV11/S2
SOV12/S2
SOV13/CH
SOV14/BH
SOV15/PT
UNIDAD DE CONTROL NEUMÁTICO
DIAGRAMA DE CABLEADO DE
VÁLVULAS SOLENOIDES
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de
un Controlador Programable
para la Estación Neumática
PN - 2800


+24VDC
- COMUN
U 01
5
10033
L1
Y
ENTRADA PLC
00016
SALIDA PLC
L4
R
L2
G
00020
SALIDA PLC
00020/1
10027
ENTRADA PLC
C 00
10
12
12
214
C 01
PB/OFF
PB/ON
EM
STOP
U 01
9
10
10037
ENTRADA PLC
L3
W
+24VDC
- COMUN
UNIDAD DE CONTROL NEUMÁTICO
ENCENDIDO CONTACTOR DE CONTROL
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de un Controlador Programable para la Estación Neumática PN - 2800


UNIDAD DE CONTROL NEUMÁTICO
DIAGRAMA NEUMÁTICO
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de un Controlador Programable para la Estación Neumática PN - 2800
CILINDRO 4
CILINDRO 14
CILINDRO 15
CILINDRO 9
CILINDRO 9
CILINDRO 3
CILINDRO 10
CILINDRO 11
CILINDRO 12
CILINDRO 1
CILINDRO 8
CILINDRO 7
CILINDRO 13
CILINDRO 5
CILINDROS 2 Y 6
S1
S2

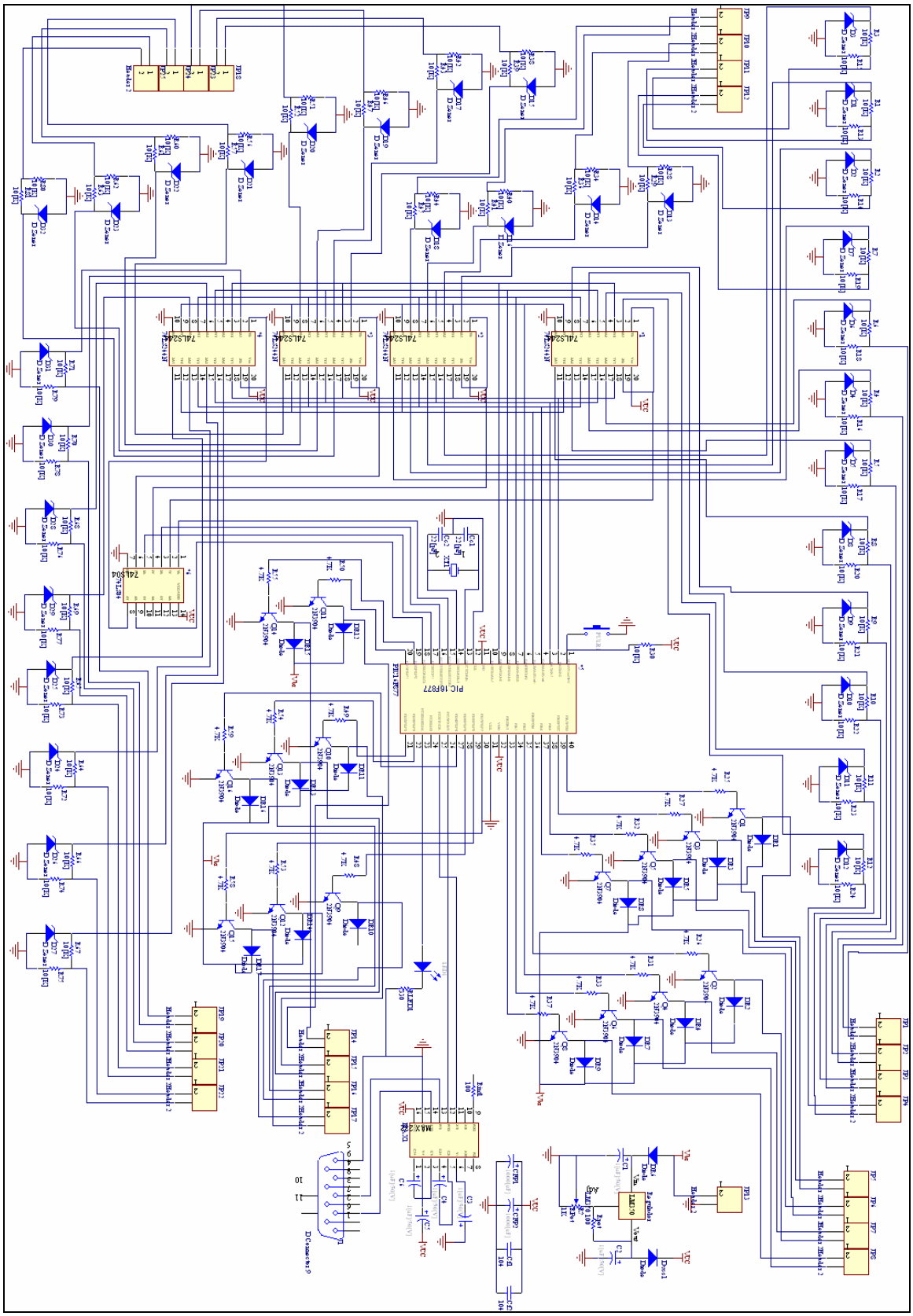

+24VDC
- COMUN
SUBRACK SECUNDARIO
SUBRACK PRIMARIO
N5
N4
N3
N2
N1
CPU
C01
+C01
+C
+24VDC
24VDC
UNIDAD DE CONTROL NEUMÁTICO
FUENTE DE PODER 24 VDC PLC
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de un Controlador Programable para la Estación Neumática PN - 2800


UNIDAD DE CONTROL NEUMÁTICO
FUENTE DE PODER 220 VAC
Realizado por:
Rommel Fernández G.
Alejandro Sánchez C.
Diseño e implementación de un Controlador Programable para la Estación Neumática PN - 2800
220 VAC
R1,S1
SW
OPCIONAL
G
Q
6A
Q1
2A
PS1
24VDC
Q10
3A
+
-
FUENTE DE PODER
24VDC
Q3
2A
AHORRO
AHORRO PARA
COMPRESOR
Q4
2A

ANEXO II

DIAGRAMA ESQUEMÁTICO DEL CIRCUITO

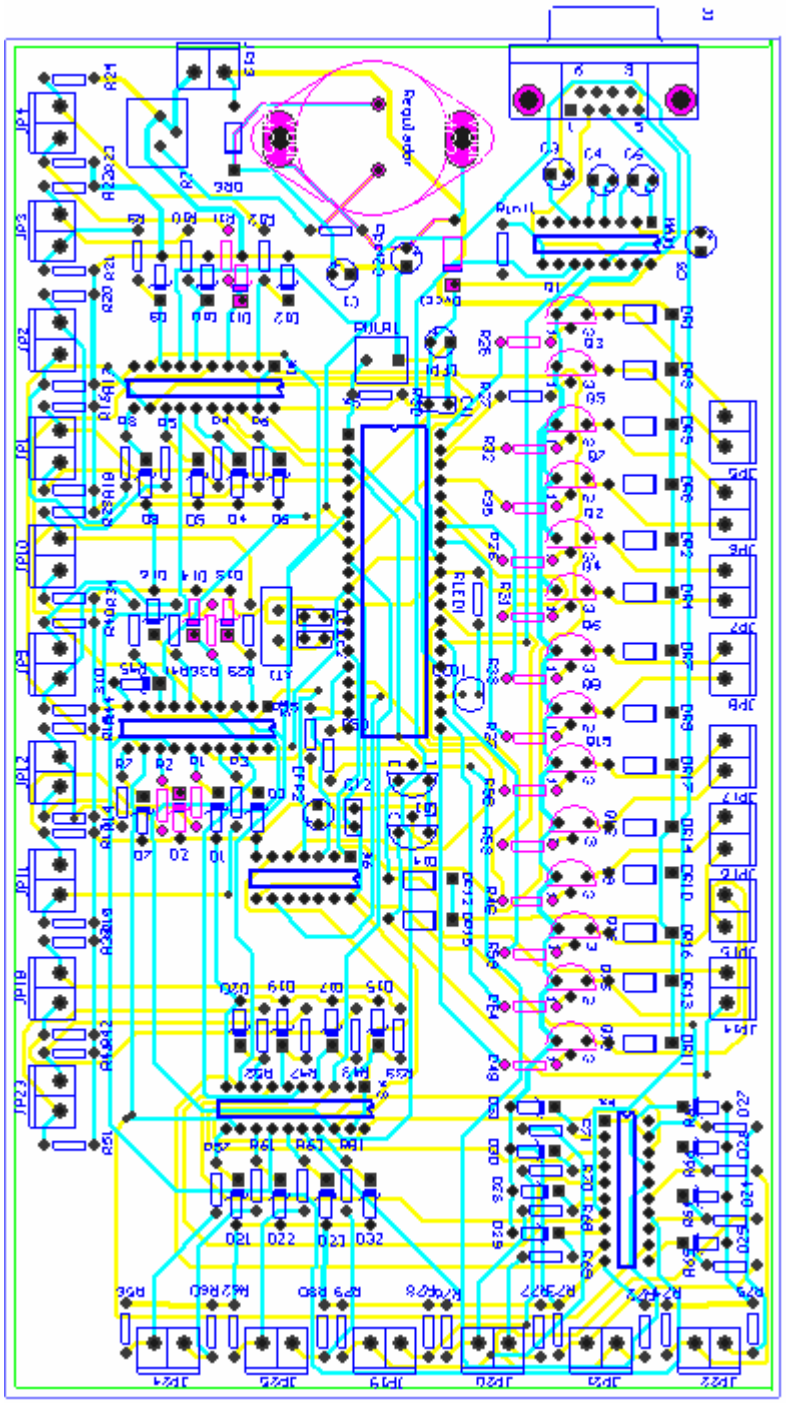
ESQUEMA DEL CIRCUITO COMPLETO



ANEXO III

DIAGRAMA DEL RUTEADO DE LA PLACA

PLACA A DOBLE LADO



ANEXO IV

CÓDIGO DE PROGRAMACIÓN DE LAS PRÁCTICAS EN MICROCODE STUDIO

Práctica 1.1

```
DEFINE LOADER_USED 1
DEFINE    OSC 4                          ' set the correct speed here!!
DEFINE HSER_BAUD 2400
adcon1=%00000110                         ;xxxx011x Puerto A, E digital
trisa=%00111111                          ;xx111111 Puerto A como IN
trisb=%00000000                          ; Puerto B como OUT
trisd=%00000000                          ; Puerto D como OUT
trisc=%10110000                          ;10xx0000
trise=%00000111       ;xxxxx111          ;  Puerto E como IN
                                         ; Variables
n var byte
i var byte
j var byte
dato var byte [14]
In var   byte
G var byte[4]                            ; declaración de Array de 4 localidades de byte
A var byte
portb=0
portd=0
pause 5000                               ; Espera 5 seg

loop:                                    ; Inicio de lazo
gosub Leer_Sensores
portb=%00000001                          ; SOV1/MP=ON
A=G[0]
while A.0==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00000101                          ; SOV3/MP=ON
A=G[0]
while A.3==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00001100                          ; SOV4/MP=ON , SOV0/MP=OFF
A=G[0]
while A.2==0
gosub Leer_Sensores
A=G[0]
wend
pause 1000
portb=%00000100                          ; SOV4/MP=OFF
pause 1000
portb=%00000101                          ; SOV1/MP=ON
A=G[0]
while A.0==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00000001                          ; SOV3/MP=OFF
pause 1000
portb=%00000011                          ; SOV2/MP=ON
A=G[0]
while A.4==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00000010                          ; SOV1/MP=OFF
```

```
A=G[0]
while A.2==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00001011                    ; SOV1/MP=ON,  SOV4/MP=ON
A=G[0]
while A.0==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00001001                    ; SOV2/MP=OFF
A=G[0]
while A.1==0
gosub Leer_Sensores
A=G[0]
wend
portb=%00000000                    ;SOV1/MP=OFF,  SOV4/MP=OFF
pause 5000
goto loop
Leer_Sensores:
;lee grupo0
portc=%00000001
gosub Leer
G[0]= In
                                   ;lee grupo1
portc=%00000010
gosub Leer
G[1]= In
                                   ;lee grupo2
portc=%00000100
gosub Leer
G[2]= In
                                   ;lee grupo3
portc=%00001000
gosub Leer
G[3]= In
return
Leer:
pauseus 500
In = porta
In.6 = porte.0
In.7 = porte.1
return
End
```

Práctica 1.2

```
DEFINE LOADER_USED 1
DEFINE    OSC 4                          ' set the correct speed here!!
DEFINE HSER_BAUD 2400
adcon1=%00000110                         ;xxxx011x Puerto A, E digital
trisa=%00111111                          ;xx111111 Puerto A como IN
trisb=%00000000                          ; Puerto B como OUT
trisd=%00000000                          ; Puerto D como OUT
trisc=%10110000                          ;10xx0000
trise=%00000111       ;xxxxx111          ;  Puerto E como IN
                                         ; Variables
n var byte
i var byte
j var byte

In var   byte
G var byte[4]                            ; Declaración de Array
A var byte
B var byte
C var byte
D var byte
portb=0
portd=0
pause 5000
loop:                                    ;Inicio de lazo
gosub Leer_Sensores
C=G[2]
D=G[3]
while D.0==0                             ; LS/PLTS=ON?
portd=%00010000
pause 1000
portd=%00000000
pause 3000
gosub Leer_Sensores
C=G[2]
D=G[3]
wend
while C.4==0                             ; LS/SH=ON?
if D.3==1 then
portd=%01000000
pause 1000
portd=%00000000
endif
gosub Leer_Sensores
C=G[2]
D=G[3]
wend
portb=%10000000                          ; SOV8/MP=ON
pause 500
portb=%10010000                          ; SOV5/MP=ON
pause 5000
portb=%00010000                          ; SOV8/MP=OFF
pause 1000
portb=%00000000                          ; SOV5/MP=OFF
pause 5000
portb=%00100000                          ; SOV6/MP=ON
pause 3000
portb=%00110000                          ; SOV5/MP=ON
pause 5000
portb=%10110000                          ; SOV8/MP=ON
pause 1000
portb=%10100000                          ; SOV5/MP=OFF
pause 5000
portb=%10000000                          ; SOV6/MP=OFF
pause 2000
                                         ; Robot pequeño
```

```
portb=%10001001                    ; SOV1/MP=ON, SOV4/MP=ON, SOV8/MP=ON
pause 3000
portb=%10001011                    ; SOV2/MP=ON
pause 2000
portb=%10001010                    ; SOV1/MP=OFF
pause 4500
portb=%10000010                    ; SOV4/MP=OFF
pause 1000
portb=%10000011                    ; SOV1/MP=ON
pause 3000
portb=%10000001                    ; SOV2/MP=OFF
pause 2000  ;
portb=%10000101                    ; SOV3/MP=ON
pause 1000
portb=%10000100                     ; SOV1/MP=OFF
pause 4500
portb=%10001100                    ; SOV4/MP=ON
pause 1000
portb=%10001101                    ; SOV1/MP=ON
pause 3000
portb=%10001001                    ; SOV3/MP=OFF
pause 1000
portb=%10000000                    ; SOV4/MP=OFF, SOV1/MP=OFF
goto loop
Leer_Sensores:
;lee grupo0
portc=%00000001                    ; SOV8/MP=OFF
gosub Leer
G[0]= In
;lee grupo1
portc=%00000010                    ; SOV1/MP=OFF, SOV2/MP=ON
gosub Leer
G[1]= In
;lee grupo2
portc=%00000100                    ; SOV3/MP=ON, SOV2/MP=OFF
gosub Leer
G[2]= In
;lee grupo3
portc=%00001000                     ; SOV1/MP=OFF, SOV4/MP=OF, SOV3/MP=OFF
gosub Leer
G[3]= In
return
Leer:
pauseus 500
In = porta
In.6 = porte.0
In.7 = porte.1
return
End
```

Código Modbus

```
DEFINE LOADER_USED                          ;Set receive register to receiver enabled
DEFINE    OSC 4                             ' set the correct speed here!!
adcon1=%00000110       ;xxxx011x            ; puerto A, E como digital
trisa=%00111111        ;xx111111            ; puerto A como in
trisb=%00000000                             ; puerto B como out
trisd=%00000000                             ; puerto D como out
trisc=%10110000        ;10xx0000
trise=%00000111        ;xxxxx111            ; puerto E como in

                                            ; declaración de variables
lsb_bit var bit
crc var word
dato_aux var word
est_inputs_LSB var word
aux1 var word
aux2 var word
bobinas1 var word
est_inputs_MSB var word
bobinas var word
bobinas_aux var word
num_coils_a_leer var word
wh var word
start_coil var word
n var byte
i var byte
j var byte
crcL var byte
crcH var byte
crc_in_H var byte
crc_in_L var byte
dato var byte [14]
In var   byte
G var byte[4]
portb=0
portd=0

Campo_DIRECCION:
hSERIN [dato[0]]                            ; lee dirección
if dato[0]!=1 then                          ; salir si dirección ≠ 1
goto Campo_DIRECCION
endif
HSERIN [dato[1]]                            ; lee función
HSERIN [dato[2]]                            ; lee dato1
HSERIN [dato[3]]                            ; lee dato2
HSERIN [dato[4]]                            ; lee dato3
HSERIN [dato[5]]                    ; lee dato4
n=5
if dato[1]= $0f  then               ; si función=1 es necesario  leer mas datos
 HSERIN [dato[6]]                   ; lee dato5
 HSERIN [dato[7]]                   ; lee dato6
 n=7
 if dato[6]==2 then                 ; se necesita leer otro dato
  HSERIN [dato[8]]                  ; lee dato3
  n=8
 endif
endif
HSERIN [crc_in_L]                   ; lee LSB de CRC
HSERIN [crc_in_H]                   ; lee MSB de CRC
```

```
gosub Cálculo_CRC                                   ; ir hacia el bucle que calcula el CRC
if  (crc_in_L != crcL)or(crc_in_H != crcH)  then    ;comparación de los crc
  dato[1]=dato[1] | $80                             ;error
  n=5
  goto respuesta                                    ; respuesta de excepción
endif
if dato[1]=1 then                                   ;leer bobinas
 num_coils_a_leer = dato[4]                         ; numero de bobinas a ser leídas
 num_coils_a_leer = num_coils_a_leer  << 8          ; desplaza contenido 8 bits a la izquierda
 num_coils_a_leer = num_coils_a_leer | dato[5]      ; operación OR

 start_coil = dato[2]
 start_coil = start_coil  << 8                      ; desplaza contenido 8 bits a la izquierda
 start_coil = start_coil | dato[3]                  ; operación OR
 bobinas = portd
 bobinas = bobinas << 8
 bobinas = bobinas | portb
 bobinas = bobinas >> start_coil
 bobinas_aux=%1111111111111111
 bobinas_aux= bobinas_aux >> (16 - num_coils_a_leer)
 bobinas = bobinas & bobinas_aux



 if  ((start_coil + num_coils_a_leer )> 7)and (start_coil  <7)  then
  dato[2]=%00000010     ; contador de datos =2
  dato[3]= bobinas
  dato[4]= bobinas >> 8
  n=4
 else
  dato[2]=%00000001                                 ; contador de datos =1
  dato[3]= bobinas
  n=3
 endif




endif
if dato[1]=2 then                                   ;leer entradas
 start_coil = dato[2]
 start_coil = start_coil  << 8
 start_coil = start_coil | dato[3]
 num_coils_a_leer = dato[4]
 num_coils_a_leer = num_coils_a_leer  << 8   ; desplaza contenido 8 bits a la izquierda
 num_coils_a_leer = num_coils_a_leer | dato[5]
 aux1 = num_coils_a_leer / 8                        ; parte entera de división en aux1
 aux2 = num_coils_a_leer // 8                       ; parte decimal de división en aux2
 if aux2 > 0  then
 aux1 = aux1 + 1                                    ; # de bytes a enviar
 endif

 gosub Leer_Sensores                                ; ir a bucle que lee sensores
 est_inputs_LSB =G[1]
 est_inputs_LSB = est_inputs_LSB << 8          ; desplaza contenido 8 bits a la izquierda
 est_inputs_LSB = est_inputs_LSB | G[0]
 est_inputs_MSB =G[3]
 est_inputs_MSB = est_inputs_MSB << 8
 est_inputs_MSB = est_inputs_MSB | G[2]
```

```
 wh = G[2]
 wh = wh << 8
 wh = wh | G[1]

  est_inputs_LSB = est_inputs_LSB >> start_coil
  wh = wh >> start_coil
  est_inputs_MSB = est_inputs_MSB >> start_coil
  dato[2]= aux1
  dato[3]= est_inputs_LSB
  dato[4]= wh
  dato[5]= est_inputs_MSB
  est_inputs_MSB = est_inputs_MSB >> 8
  dato[6]= est_inputs_MSB
  n= 2 +  dato[2]
endif
if dato[1]== $0f then                                   ;forzar bobinas
 aux1=%1111111111111111
 aux2=%1111111111111111
 start_coil = dato[2]
 start_coil = start_coil  << 8
 start_coil = start_coil | dato[3]
 num_coils_a_leer = dato[4]      ;
 num_coils_a_leer = num_coils_a_leer  << 8
 num_coils_a_leer = num_coils_a_leer | dato[5]
 num_coils_a_leer = num_coils_a_leer +  start_coil
 num_coils_a_leer=16 - num_coils_a_leer
 aux1= aux1 >>num_coils_a_leer
 aux2= aux2 <<  start_coil
 aux1=aux1 & aux2
 aux2 = ~ aux1
 bobinas_aux = portd
 bobinas_aux = bobinas_aux << 8
 bobinas_aux = bobinas_aux | portb
 bobinas_aux = bobinas_aux & aux2
 bobinas=0
 if dato[6]==%00000010 then
 bobinas = dato[8]
 bobinas = bobinas  << 8
 bobinas = bobinas | dato[7]
 else                                                   ; count = 1  o error count mayor a 3
 bobinas = dato[7]
 endif
 bobinas = bobinas << start_coil
 bobinas = bobinas & aux1
 bobinas = bobinas_aux | bobinas
 portb = bobinas
 bobinas = bobinas >> 8
 portd = bobinas
 n=5
endif
if dato[1]== $05 then                                   ;forzar una bobina
bobinas_aux=%0000000000000001
 ;if dato[3]>0 then     ojo
  bobinas_aux=bobinas_aux << dato[3]
 ;endif               ojo
   bobinas = portd
   bobinas = bobinas << 8
   bobinas = bobinas | portb
 if dato[4]== $ff then                                 ;encender
```

```
  bobinas = bobinas | bobinas_aux          ; OR
else                                       ;apagar
  bobinas_aux = ~ bobinas_aux              ; operación NOT
  bobinas = bobinas & bobinas_aux          ; operación AND
 endif
  portb = bobinas
  bobinas = bobinas >> 8
  portd = bobinas
  n=5
endif
gosub Cálculo_CRC                          ; ir a bucle que calcula CRC
RESPUESTA:
for i=0 to n
hSEROUT [dato[i]]
next
hSEROUT [crcL]
hSEROUT [crcH]
goto Campo_DIRECCION                       ; ir a campo dirección

Leer_Sensores:                                          ;lee grupo0
portc=%00000001
gosub Leer
G[0]= In
                                           ;lee grupo1
portc=%00000010
gosub Leer
G[1]= In
                                           ;lee grupo2
portc=%00000100
gosub Leer
G[2]= In                                        ;lee grupo3
portc=%00001000
gosub Leer
G[3]= In
return
Leer:
pauseus 500                                ; pausa de 500 useg
In = porta
In.6 = porte.0
In.7 = porte.1
return
Cálculo_CRC:
crc=$ffff
for i=0 to n
dato_aux= dato[i] & $00ff
crc= crc ^ dato_aux
 for j=0 to 7
 lsb_bit=crc.0
 crc=crc >> 1
  if lsb_bit == 1 then
  crc=crc ^ $A001
  endif
 next j
next i
crcL=crc & $00ff
crc=crc >> 8
crc=crc & $00ff
crcH=crc
return
End
```

ANEXO V

MANUAL DE USUARIO

Este manual, incluye todas las ilustraciones y procedimientos para el manejo del software el cual controla el sistema realizado.

Requerimientos del Sistema

Intel Pentium II/III/IV, Celeron y MMX

Windows 98, Me, XP

Requerimientos del Software

Microcode Studio Plus

Icprog

Mbus

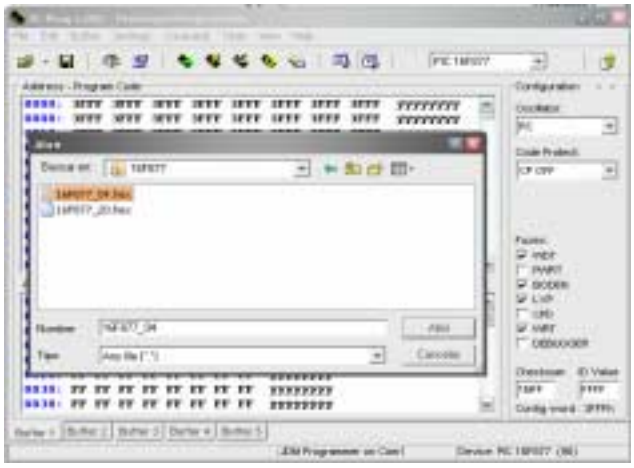
Lookout 5.1

Requerimientos del Hardware

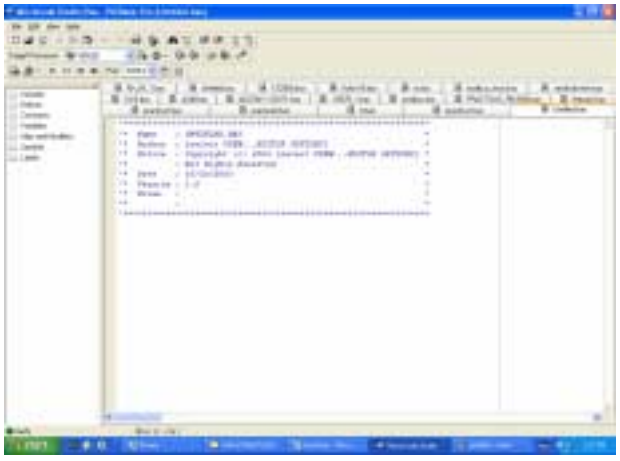
Sistema de Control Programable

Fecha de Realización © 2006

Paso 1. Con la ayuda del programa Icprog, es necesario cargar por única vez en el microcontrolador el programa bootloader 16F877_04.Hex[12] (Funciona con cristal de 4Mhz).



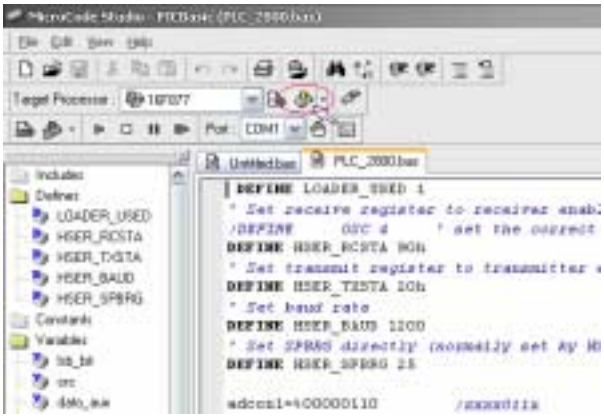
Paso 2. La elaboración de los programas se realiza en la hoja de programación que ofrece el programa MicroCode Studio Plus.[13]



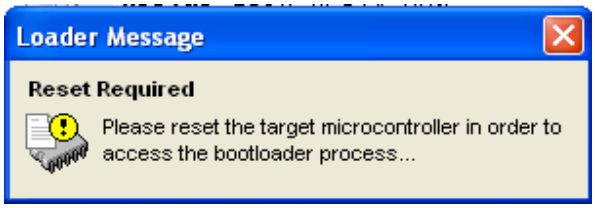
Paso 3. Se debe tener en cuenta las siguientes características de hardware para proceder a programar como se muestra en el CAPÍTULO VI en los ejemplos.

[12] 16F877_04.Hex incluido en el CD de documentación.
[13] MicroCode Studio Plus incluido en el CD de documentación.

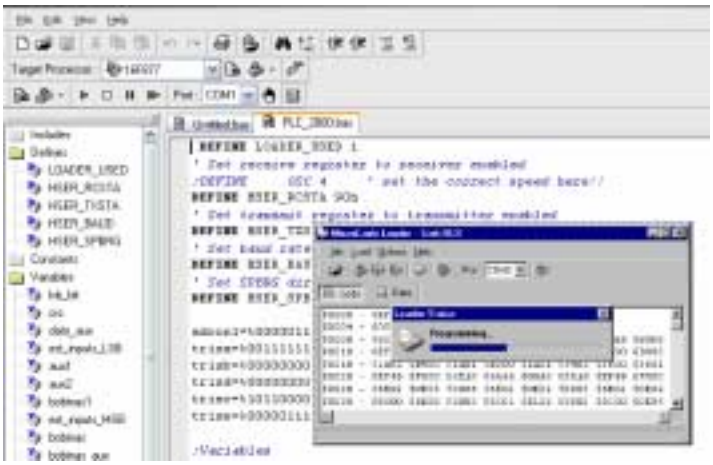
Paso 4. Se da un clic en el icono señalado. Esta acción compilara el código de programación y el loader tomará el mando del programa escrito por el usuario en el microcontrolador.



Paso 5. Microcode Loader solicitará resetear el microcontrolador.



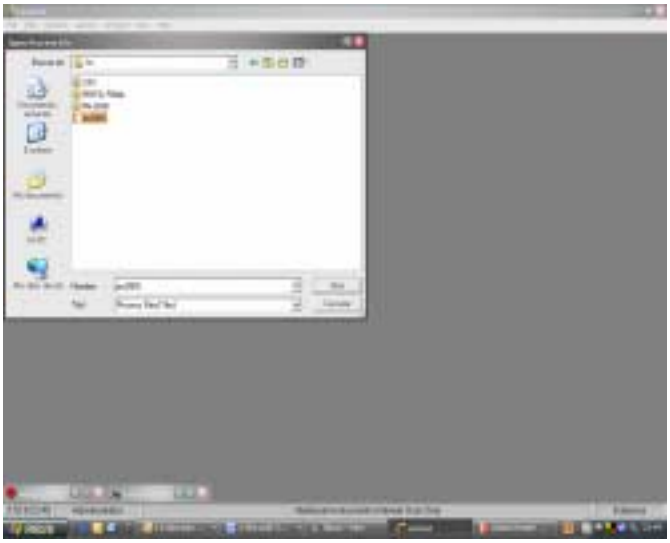
Paso 6. Posterior al reset manual en el dispositivo de control, la carga del programa usuario se iniciará.



La estación trabajará sin necesidad del cable serial conectado al Sistema de Control Programable

Para trabajar con la interface HMI, se carga el programa PLC_2800.hex[14] comenzando desde el Paso 2 hasta el Paso 6.

Con el software Lookout 5.1 se procede la realización de la HMI tomando en cuenta las consideraciones que se muestran en el CAPÍTULO IV en la parte de INTERFACE HOMBRE – MÁQUINA y se procede al control y monitoreo de la Estación Neumática PN – 2800, en este caso, el proceso es ***pn2800.L4P*** [15], que es el correspondiente a la estación.



Se despliega el Panel de Control realizado en Lookout 5.1 que su procedimiento está detallado en el CAPÍTULO VI en la Práctica No. 3 tanto para el manejo de forma manual y semi-automático, para este último se lo activa abriendo en Lookout 5.1 el archivo ***processauto.L4P***.[16]

[14] PLC_2800.hex incluido en el CD de documentación.
[15] pn2800.L4P incluido en el CD de documentación.
[16] processauto.L4P incluido en el CD de documentación.

ÍNDICE DE FIGURAS

CAPÍTULO I INTRODUCCIÓN

CAPÍTULO II MARCO TEÓRICO

CAPÍTULO III DISEÑO DE HARDWARE

CAPÍTULO IV DISEÑO DE SOFTWARE

CAPÍTULO V PRUEBAS Y RESULTADOS

CAPÍTULO VI PRÁCTICAS

CAPÍTULO VII ANÁLISIS COSTO - BENEFICIO

ÍNDICE DE TABLAS

CAPÍTULO I INTRODUCCIÓN

CAPÍTULO II MARCO TEÓRICO

CAPÍTULO III DISEÑO DE HARDWARE

CAPÍTULO IV DISEÑO DE SOFTWARE

CAPÍTULO VI PRÁCTICAS

CAPÍTULO VII ANÁLISIS COSTO - BENEFICIO

FECHA DE ENTREGA

Sangolquí,

AUTORIDADES:

_____

Sr. Ing. Víctor Proaño

Coordinador de Carrera de Ingeniería Electrónica y Control Automático

_____

Sr. Dr. Jorge Carvajal

Unidad de Admisión y Registro

ELABORADO POR:

_____

Sr. Rommel David Fernández Granja

_____

Sr. Christian Alejandro Sánchez Carranza