Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

# PROPUESTA PARA LA APLICACIÓN DE FUNCIONES DE COMUNICACIÓN EN EL PLC DE UNA MÁQUINA LLENADORA

**Juan Carlos Sánchez Meyer**
Asesorado por el Ing. Julio César Solares Peñate

Guatemala, febrero de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA


FACULTAD DE INGENIERÍA

# PROPUESTA PARA LA APLICACIÓN DE FUNCIONES DE COMUNICACIÓN EN EL PLC DE UNA MÁQUINA LLENADORA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA

FACULTAD DE INGENIERÍA

POR

**JUAN CARLOS SÁNCHEZ MEYER**

ASESORADO POR EL INGENIERO JULIO CÉSAR SOLARES PEÑATE

AL CONFERÍRSELE EL TÍTULO DE

**INGENIERO ELECTRÓNICO**

GUATEMALA, FEBRERO DE 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

## NÓMINA DE JUNTA DIRECTIVA

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| VOCAL I | Inga. Glenda Patricia García Soria |
| VOCAL II | Inga. Alba Maritza Guerrero de López |
| VOCAL III | Ing. Miguel Ángel Dávila Calderón |
| VOCAL IV | Br. Kenneth Issur Estrada Ruiz |
| VOCAL V | Br. Elisa Yazminda Vides Leiva |
| SECRETARIA | Inga. Marcia Ivonne Véliz Vargas |

## TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

| | |
|---|---|
| DECANO | Ing. Murphy Olympo Paiz Recinos |
| EXAMINADOR | Dr. Juan Carlos Córdova Zeceña |
| EXAMINADOR | Inga. Ingrid Salomé Rodríguez García de Loukota |
| EXAMINADOR | Ing. Francisco Javier González López |
| SECRETARIA | Inga. Marcia Ivonne Véliz Vargas |

## HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

# PROPUESTA PARA LA APLICACIÓN DE FUNCIONES DE COMUNICACIÓN EN EL PLC DE UNA MÁQUINA LLENADORA,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, el 4 de septiembre de 2006.

Juan Carlos Sánchez Meyer

## AGRADECIMIENTOS

Te agradezco mi Dios, por haberme colocado en el seno de una familia que aprecia el conocimiento y no conoce la censura, donde he crecido en todo sentido hasta el día de hoy.

Te agradezco, también, la oportunidad que me has brindado siempre de cultivar el intelecto y la oportunidad de compartir con excelentes compañeros.

Además, te doy las gracias por los excelentes maestros de los que he disfrutado y aprendido desde pequeño. También, te agradezco por los malos maestros, que me has enseñado a reconocer y de los cuales he procurado aprender a no seguir sus pasos.

Te agradezco por los pocos, pero buenos amigos que me has dado. Pues, no hay ni uno solo que no valga su peso en oro. De los cuales, también, aprendo día a día y me han apoyado y reconfortado, aún sin darse cuenta de ello.

Te agradezco, porque sé que me amas y así como me has acompañado hasta hoy, tengo la seguridad que me seguirás acompañando, así como me has abierto y cerrado las puertas correctas, me las seguirás abriendo y cerrando para asegurarte que recorra el camino que me corresponde.

# ÍNDICE GENERAL

# ÍNDICE DE ILUSTRACIONES

## FIGURAS

## TABLAS

# GLOSARIO

**ActiveX** Tecnología que permite incorporar módulos preprogramados en aplicaciones nuevas.

**Neumática** Tecnología que utiliza aire comprimido para controlar y producir movimientos.

**PLC** Controlador lógico programable.

**Profibus** Tipo de bus de campo de comunicación industrial.

**TCP/IP** Protocolo de transmisión de paquetes de información utilizado en todo el mundo.

**TTL** Lógica transistor a transistor.

# RESUMEN

Se ha tomado a una máquina llenadora de pistón, una de las máquinas más comunes en la industria guatemalteca, para implementar en ella un sistema de control basado en un PLC. El sistema busca optimizar el funcionamiento de la máquina y adquirir información del proceso por medio del mismo.

Se inicia explicando cómo funciona una máquina de este tipo, cuáles son sus componentes, se explica su sistema eléctrico y neumático, se realiza el programa de control y, por último, se implementan funciones de comunicación entre el PLC y una hoja de Excel.

Al final, se ve y se comprende, en su totalidad, a una máquina llenadora de este tipo, lo que le da al lector la oportunidad de utilizar la información de este trabajo en la práctica.

# OBJETIVOS

## GENERAL

Diseñar un sistema de control basado en un controlador programable para una máquina llenadora de pistón, el cual permita la efectiva comunicación de información del proceso hacia una computadora.

## ESPECÍFICO

Demostrar que el conocimiento de la información de un proceso en el momento adecuado, en este caso, el de llenado, permite tomar decisiones certeras basadas en información real y no suposiciones.

# INTRODUCCIÓN

La información acerca de un proceso es crucial para la toma de decisiones, especialmente, en la actualidad, que el mundo se ha vuelto, extremadamente, competitivo. Muchas veces, la información está disponible demasiado tarde, cuando ésta no refleja la realidad del proceso. Si se integra un sistema eficiente y de bajo costo para la adquisición de información, idealmente integrado en el sistema de control, se obtiene una herramienta, extremadamente, poderosa para cualquier empresario.

Tradicionalmente, las funciones de un controlador programable en muchas máquinas se limitan a controlar la correcta operación de éstas y se desaprovecha, comúnmente, la oportunidad de adquirir información importante, tanto del proceso, como de los mismos componentes de las máquinas, utilizando el mismo sistema de control. Naturalmente, si se supone que es posible obtener información del proceso y de la máquina, a través del controlador programable, se debe disponer de algún mecanismo para transmitir esa información, rápidamente, hacia las personas encargadas de tomar las decisiones. Consecuentemente, si estas personas disponen de la información necesaria, oportunamente, los resultados de sus decisiones necesariamente deben ser acertados.

Muchas máquinas, en la industria guatemalteca, utilizan sistemas de control primitivos. Algunas otras utilizan sistemas de control electrónicos. Pero muy pocas, utilizan sistemas que implementen la adquisición de información del proceso en tiempo real.

El implementar sistemas de adquisición de información del proceso en conjunto con los sistemas de control, permite contar con una herramienta poderosa para basar la toma de decisiones en información real, la cual puede permitir a mediano plazo el crecimiento de la pequeña y mediana industria.

En este trabajo se pretende demostrar lo explicado, anteriormente, aplicando funciones de comunicación, para llevar la información de una simple máquina llenadora hacia una computadora.

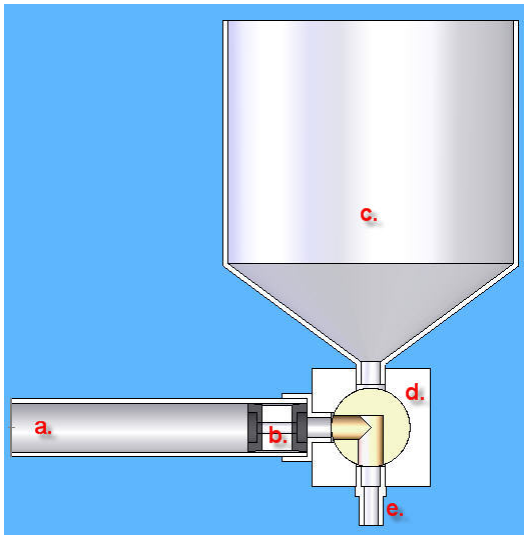
# 1. LA MÁQUINA LLENADORA DE PISTÓN

La máquina llenadora de pistón es una pieza de maquinaria muy común en la industria, que se encuentra presente en la mayoría de plantas donde se requiere llenar algún líquido. Es capaz de llenar desde un líquido hasta materiales viscosos como cremas. Presenta la característica de poder llenar un volumen prefijado con mucha precisión.

## 1.1. Partes principales

A continuación se presenta la imagen del mecanismo básico de una llenadora de pistón, donde se pueden apreciar sus componentes.

**Figura 1. Partes principales de la llenadora de pistón.**



Las partes principales para el funcionamiento básico de la llenadora son:

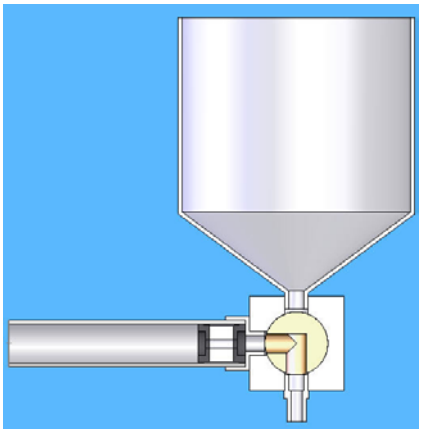
a. Cilindro.
b. Pistón o émbolo.
c. Tanque.
d. Válvula de 3 vías.
e. Boquilla de llenado.

## 1.2. Descripción del funcionamiento

El funcionamiento de la llenadora es realmente sencillo. Utiliza solamente dos piezas móviles, el émbolo y la válvula de 3 vías. El orden de los movimientos de estas dos piezas determina el correcto funcionamiento. A continuación se presentan cinco imágenes que representan los diferentes estados de los componentes a lo largo de un ciclo de llenado.

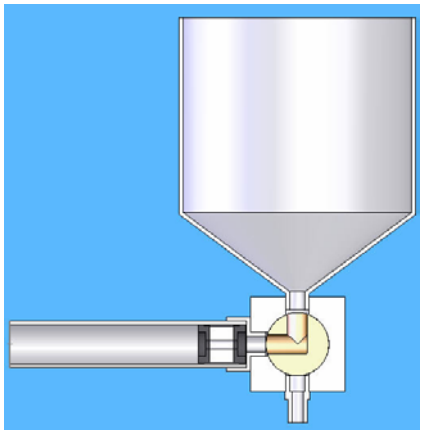
La figura 2 muestra el estado inicial de los componentes. El émbolo se encuentra en el extremo pegado a la válvula, y ésta última conecta el cilindro con la boquilla de llenado.

**Figura 2. Primer estado del ciclo de llenado.**



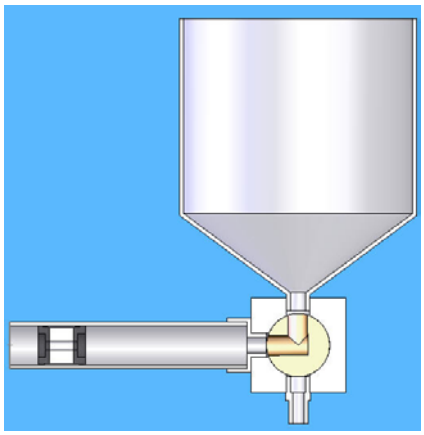
El primer movimiento es el de la válvula de 3 vías, que ahora pasa a conectar el tanque con el cilindro, como se ve en la figura 3. El tanque debe mantenerse siempre lleno del producto a llenar.

**Figura 3. Segundo estado del ciclo de llenado.**



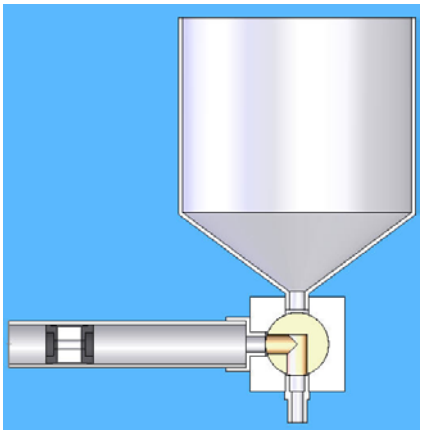
A continuación se mueve el émbolo creando una depresión en la recámara del cilindro. Esto provoca un efecto de succión que obliga al producto del tanque a llenar el volumen en el cilindro. Esto se aprecia en la figura 4.

**Figura 4. Tercer estado del ciclo de llenado.**



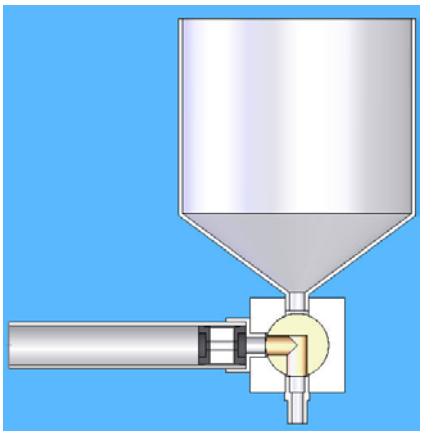
El siguiente paso en la secuencia de funcionamiento de la máquina contempla nuevamente el movimiento de la válvula de 3 vías. Ahora pasa a conectar el cilindro con la boquilla de llenado, como se ve en la figura 5.

**Figura 5. Cuarto estado del ciclo de llenado.**



Por último, el émbolo se desplaza nuevamente en dirección de la válvula, obligando al producto que se encuentra en la recámara del cilindro a salir expulsado por la boquilla de llenado (figura 6). Debajo de la boquilla se debe encontrar el recipiente a llenar. Con este paso finaliza un ciclo de llenado.

**Figura 6. Último estado del ciclo de llenado.**



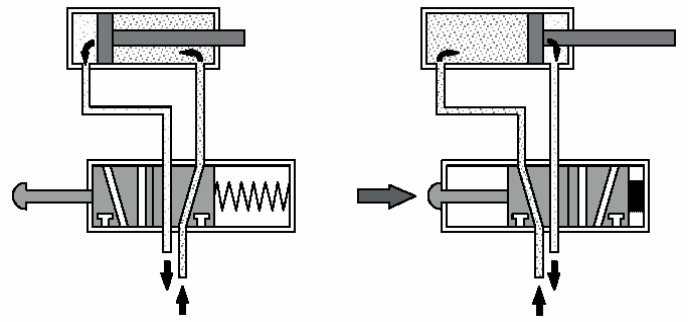
## 1.3. Actuadores neumáticos

Como hemos visto, el funcionamiento básico de una llenadora de pistón involucra dos movimientos, el del émbolo y el de la válvula de 3 vías. Es común en estas máquinas que estos movimientos sean hechos utilizando actuadores neumáticos.

La neumática es la técnica que se ocupa de la utilización del aire comprimido como fuente de energía para hacer trabajos. El actuador neumático más común es el cilindro de doble efecto.

El cilindro de doble efecto necesita de una válvula direccional para controlar el flujo del aire comprimido que le llega. La válvula por lo general es una válvula de 5 vías y 2 posiciones, y puede ser actuada manualmente, neumáticamente, mecánicamente o eléctricamente. A esta válvula se le conoce también como válvula 5/2.

**Figura 7. Control de un cilindro neumático de doble efecto.**

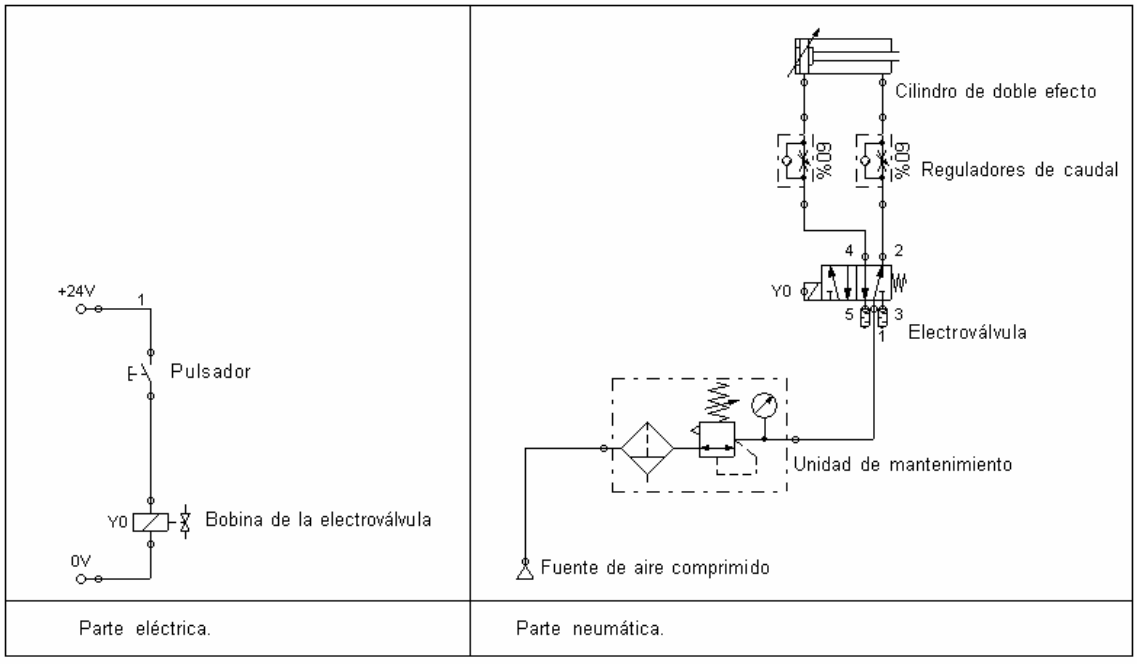


Fuente: Croser, Peter y Ebel, Frank, ***Pneumatics***, pág. 37.

En la figura 7 se puede observar como una válvula 5/2 actuada manualmente controla el movimiento de un cilindro doble efecto. Del lado izquierdo aparece la válvula en su posición de reposo, donde por la disposición

de las conexiones obliga al aire comprimido a llenar la recámara anterior y al mismo tiempo permite que el aire en la recámara posterior sea evacuado. Esto ocasiona que el cilindro se mueva hacia dentro. Del lado derecho, se ve la válvula cuando es activada, lo que permite que el aire comprimido llene la recámara posterior y el aire de la recámara anterior escape. Ahora el cilindro se mueve hacia fuera.

**Figura 8. Diagrama eléctrico y neumático de un sistema electroneumático.**




Los símbolos utilizados en los diagramas representan claramente el funcionamiento de cada dispositivo. En la figura 8 se presenta un ejemplo de un diagrama para el accionamiento de un cilindro de doble efecto. Ahora la válvula es accionada eléctricamente y no manualmente, por lo que existe un diagrama eléctrico. Además, ahora aparecen dos reguladores de caudal, que tienen la función de regular la velocidad del movimiento del cilindro. También aparece

una unidad de mantenimiento que regula la presión del aire comprimido, lo filtra, le quita el condensado y lo lubrica.

Cuando se presiona el pulsador se cierra el circuito de la bobina de la electroválvula, marcada como Y0. Ésta provoca que la electroválvula cambie de posición, obligando al cilindro de doble efecto a salir. Cuando el pulsador se suelta, el circuito de la bobina se abre, se desactiva la electroválvula y el cilindro regresa a su posición inicial.

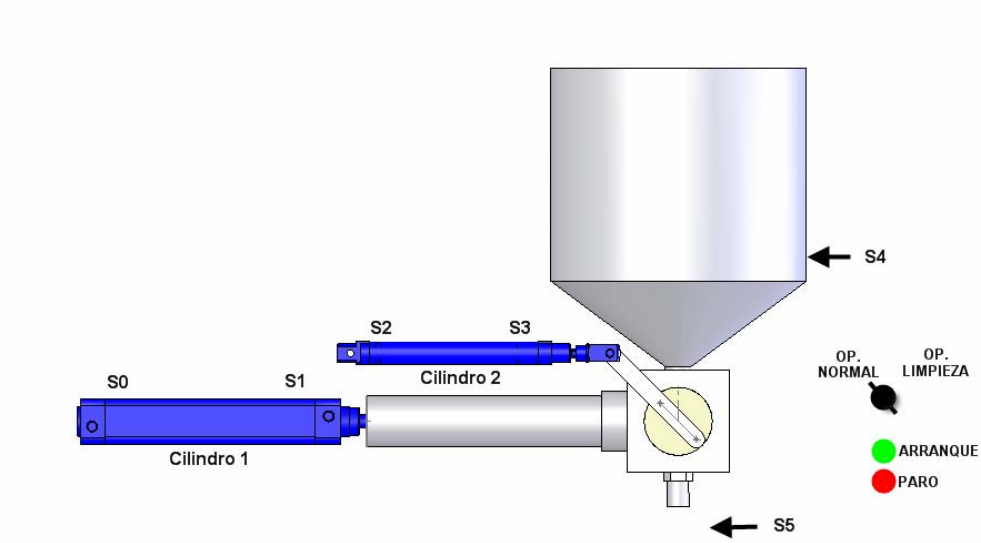
### 1.4. Disposición final de la máquina

Como ya se ha mencionado, la máquina llenadora de pistón necesita un cilindro de doble efecto neumático por cada uno de sus movimientos. Si se observa la figura 9, se pueden ver ambos cilindros. El cilindro 1 mueve el émbolo del cilindro de la máquina para poder succionar y expulsar producto. El cilindro 2 mueve la posición de la válvula 3/2. Cada uno de estos cilindros neumáticos necesita reguladores de caudal para poder controlar la velocidad del movimiento. También es necesario tener una electroválvula para cada cilindro neumático.

Para poder tener información de retroalimentación acerca de la posición de cada cilindro neumático, se utilizan sensores diseñados para detectar un campo magnético causado por un imán montado en el émbolo de cada cilindro neumático. Si se ve nuevamente la figura 9, se ven las posiciones de los sensores identificadas con S0, S1, S2 y S3. Cuando el cilindro 1 esté dentro, el estado del sensor S0 será alto. Cuando esté fuera, el sensor S1 generará una señal alta. De igual forma tendrán los sensores S2 y S3 señales altas, al momento de que el cilindro 2 esté fuera o adentro, respectivamente.

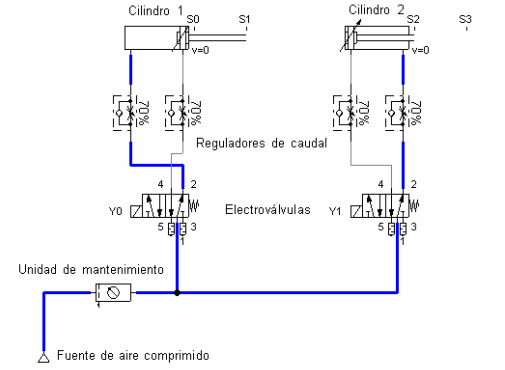
Adicionalmente, es necesario un sensor que detecte cuando el nivel de producto haya descendido por debajo de un nivel mínimo. Este sensor está identificado como S4 y se muestra su posición en la figura 9. En esta figura también se observa el sensor S5, que se trata de un sensor optoelectrónico para detectar la presencia de envase para llenar.

**Figura 9. Posición de cilindros y sensores en la máquina.**




Por último, se ve en la figura 9, que se cuenta con dos pulsadores, uno para el arranque del sistema y otro para el paro. Un selector indica el modo de operación, que puede ser operación normal u operación en modo de limpieza.

**Figura 10. Diagrama neumático de la máquina llenadora.**



El diagrama neumático de la máquina llenadora se presenta en la figura 10. Se observan los cilindros neumáticos 1 y 2 sobre los cuales se señalan las posiciones de los sensores S0, S1, S2 y S3. Ambos cilindros tienen sus reguladores de caudal para regulación de velocidad de los movimientos. También se observan las electroválvulas, cada una con su respectiva bobina Y0 y Y1. Este es el diagrama neumático de la máquina llenadora al cual estaremos haciendo referencia en el resto del trabajo.

# 2. EL CONTROLADOR LÓGICO PROGRAMABLE O PLC COMO UNIDAD DE CONTROL

El primer controlador lógico programable fue desarrollado por un grupo de ingenieros de *General Motors* en 1968, cuando la compañía buscaba una alternativa para reemplazar los complejos sistemas de control basados en relevadores.

El nuevo sistema de control tenía que cumplir con los siguientes requisitos:

- Programación sencilla.
- Cambios de programación sin necesidad de intervenir la configuración física del sistema (sin modificar el alambrado).
- Más pequeño, más barato y más confiable que los sistemas basados en relevadores.
- Mantenimiento sencillo y de bajo costo.

El resultado del desarrollo fue un sistema, que permitía la simple conexión de señales binarias. Los requisitos de como y cuando debían de conectarse estas señales se especificaban en el programa. Con estos nuevos sistemas, por primera vez fue posible graficar las señales en una pantalla y archivarlas en memorias electrónicas.

Desde entonces, más de tres décadas han pasado, durante las cuales se han hecho grandes progresos en microelectrónica que se han reflejado también en la tecnología de controladores lógicos programables.

El rango de funciones de un PLC ha crecido considerablemente. Hace 15 años, la visualización de procesos, el procesamiento analógico o el simple hecho de utilizar un PLC como un controlador, eran considerados utópicos. Hoy en día, estas funciones son parte integral de muchos PLCs.

## 2.1. Aplicaciones

Cada sistema o máquina tiene un controlador. Dependiendo del tipo de tecnología usada, los controladores se dividen en neumáticos, hidráulicos, eléctricos y electrónicos. Frecuentemente, una combinación de diferentes tecnologías es usada. Además, se diferencian también en controladores de programación por medio de alambrado (interconexión o alambrado de elementos electromecánicos o componentes electrónicos) y controladores lógicos programables. El primero se utiliza por lo general, en lugares donde no se necesita reprogramación del sistema y donde el tamaño del trabajo permite el desarrollo de un controlador especializado en esa tarea. Aplicaciones típicas para ese tipo de controladores, se encuentran en lavadoras de ropa, cámaras de video, carros, etc.

Sin embargo, si el tamaño del trabajo no permite el desarrollo de un controlador especializado, o si el usuario necesita la característica de hacer cambios en el programa, o de modificar tiempos y contadores, entonces la alternativa de utilizar un controlador universal, donde el programa se almacene en memoria electrónica, es la alternativa preferida. El PLC es un ejemplo de controlador universal. Puede ser usado para diferentes aplicaciones y permite el cambio, la ampliación y la optimización del programa de control del proceso.

La tarea original de un PLC envolvía la interconexión de las señales de entrada de acuerdo a un programa específico, donde si el resultado era

“verdadero”, conmutaba la salida correspondiente. El álgebra de Boole forma la base matemática para este tipo de operaciones, donde solamente se reconocen dos posibles estados para una variable, “0” y “1”. Una salida, por lo tanto, solamente puede tener alguno de estos dos estados. Por ejemplo, un motor conectado a una salida, puede estar solamente, ya sea encendido o apagado.

Esta función le ganó el nombre de PLC o controlador lógico programable. Sin embargo, las tareas de un PLC se multiplicaron rápidamente: temporizadores y contadores, memoria, operaciones matemáticas, las cuales son ejecutadas por cualquiera de los PLCs disponibles hoy en día.

Las redes de varios PLCs así como la de un PLC y una computadora maestra son creadas utilizando interfaces especiales de comunicación. Para realizar esto, muchos de los PLCs modernos son compatibles con estándares de buses de comunicación abiertos, como es el caso de Profibus. Además, debido a la gran capacidad de algunos PLCs, estos pueden funcionar directamente como maestros en una red.

Para el final de la década de los setenta, se agregaron entradas y salidas analógicas, lo que permitió controlar procesos con variables analógicas (medición de fuerza, ajuste de velocidades, sistemas de posicionamiento servoneumáticos). Al mismo tiempo, fue posible realizar funciones para controles de lazo cerrado.

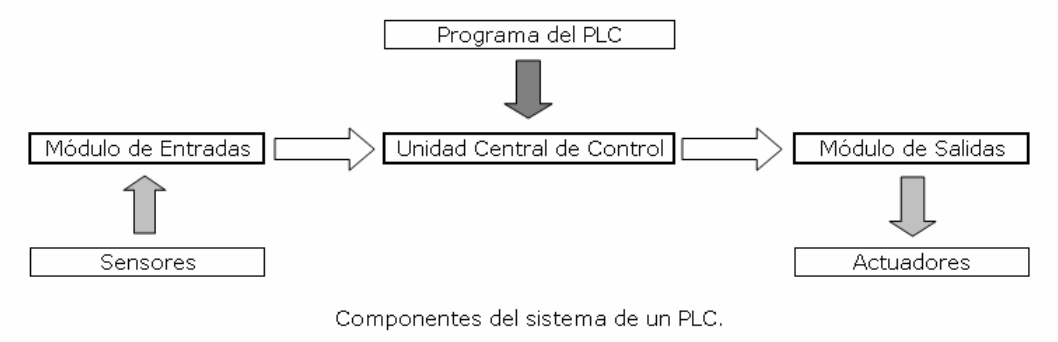
En el mercado hay PLCs que se adaptan a prácticamente cualquier necesidad del usuario. Por ejemplo, hay PLCs miniatura con pocas entradas y salidas. También hay PLCs más grandes con 28 entradas / salidas, 256 entradas / salidas e incluso más.

Hay PLCs que pueden ser expandidos con entradas / salidas digitales y analógicas, con módulos de posicionamiento y módulos de comunicación. Existen PLCs especiales para tecnología de seguridad, para aplicaciones marítimas y tareas de minería. Por si fuera poco, hay PLCs que pueden procesar varios programas de control simultáneamente (*multitasking*).

## 2.2. Diseño

Un controlador lógico programable no es más que una computadora, diseñada específicamente para ciertas tareas de control, que incorpora entradas y salidas, digitales y analógicas, para interactuar con el proceso que controla. Los componentes de un sistema de un PLC se aprecian en la figura 11.

**Figura 11. Componentes del sistema de un PLC.**




Componentes del sistema de un PLC.

La función de un módulo de entradas es convertir las señales entrantes en señales, que puedan ser procesadas por el PLC, y que puedan así ser pasadas a la unidad central de control. La tarea inversa es efectuada por un módulo de salidas. Éste convierte las señales de un PLC en señales adecuadas para los actuadores. El procesamiento de las señales se lleva a cabo en la unidad central de control de acuerdo al programa de control almacenado en la memoria.

El programa de un PLC puede ser creado en varias formas: utilizando comandos de lenguaje ensamblador (listado de instrucciones), lenguajes de alto nivel orientados a la solución de problemas (texto estructurado) o diagramas de flujo. En Europa, el uso de diagramas de bloques basados en símbolos gráficos y compuertas lógicas es ampliamente usado, mientras que en América el que más se utiliza es el diagrama de escalera basado en un diagrama eléctrico.

Se puede hacer una diferenciación entre PLCs compactos y modulares, dependiendo como esté conectada la unidad central de control a los módulos de entradas y salidas. En la figura 12 se pueden ver algunos ejemplos.

**Figura 12. Ejemplos de PLCs compactos y modulares.**

El diseño del hardware de un controlador lógico programable está hecho para soportar las condiciones típicas de un ambiente industrial como determinados niveles de señales, calor, humedad, fluctuaciones en la fuente de poder e impactos mecánicos.

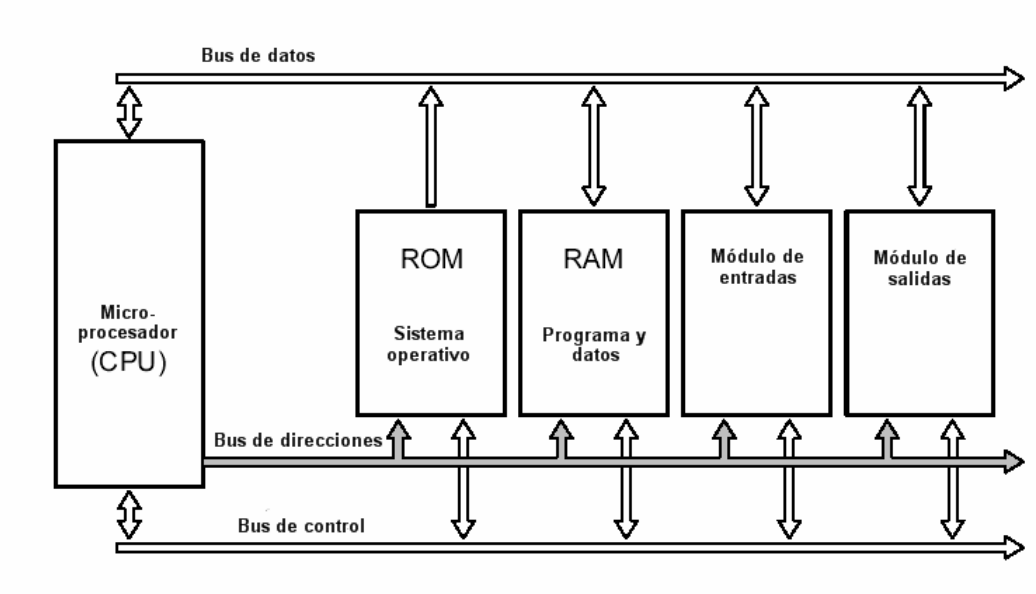
En sistemas de computadoras, generalmente se diferencia entre *hardware*, *firmware* y *software*. Esto mismo se aplica para un PLC, que está basado en una microcomputadora.

El *hardware* está formado por los elementos físicos, como los circuitos impresos, los circuitos integrados, los alambres, baterías, carcasa, etc.

El *firmware* es la parte de software, que está permanentemente instalada y es proveída por el fabricante del PLC. En él se incluye rutinas fundamentales del sistema, utilizadas para iniciar el procesador cuando se le conecta la energía. Adicionalmente, está el sistema operativo, que se almacena generalmente en una memoria ROM o EPROM.

Finalmente, el *software* es el programa de control escrito por el usuario del PLC. Los programas son normalmente almacenados en memoria RAM o EEPROM, donde son fácilmente modificables.

**Figura 13. Diseño de un microcomputador.**




Fuente: Bliesener, Ebel, Loeffler, Plagemann, Regber, Terzi, Winter, ***Programmable logic controllers***, pág. B-33.

El *hardware* de un PLC está basado en un sistema de bus, como se ve en la figura 13. Éste está conformado de un número de líneas eléctricas divididas en líneas de dirección, de datos y de control. Las líneas de dirección son usadas para seleccionar la dirección de una estación conectada al bus, y las líneas de datos sirven para transmitir la información necesaria. Las líneas de control son necesarias para activar la estación correcta ya sea como un transmisor o un receptor de información.

Las estaciones principales conectadas al sistema de bus son el microprocesador y la memoria. La memoria puede estar dividida en memoria para el *firmware* y memoria para el programa del usuario e información.

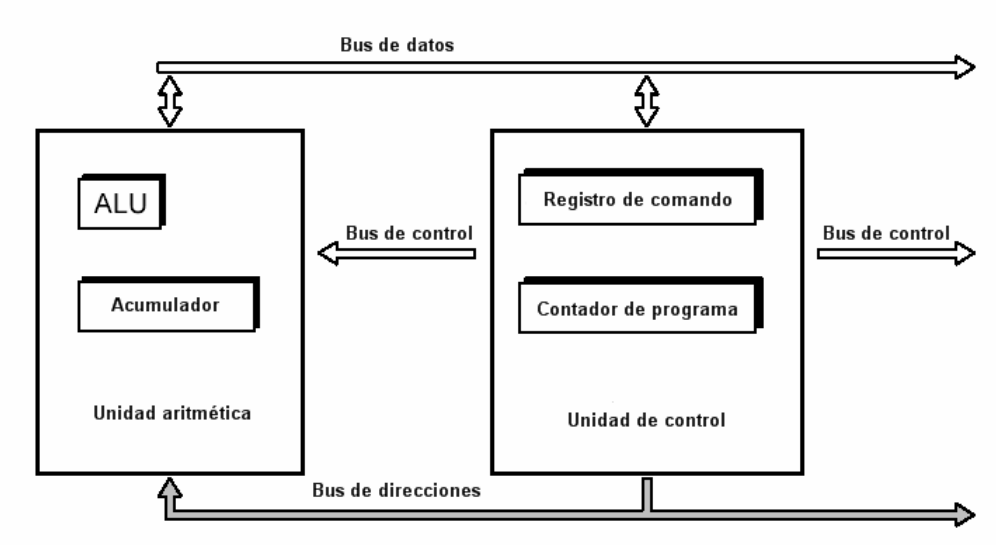
Dependiendo de la estructura del PLC, los módulos de entradas y salidas pueden estar conectados a un único bus común o, con la ayuda de alguna interfase, a un bus externo. Por último, se necesita una conexión hacia un dispositivo de programación o una computadora personal, que por lo general es una interfase serial.

## 2.3. Unidad central de control

En esencia, la unidad central de control de un PLC consiste de una microcomputadora. El sistema operativo del fabricante convierte una computadora de uso universal en un PLC, optimizándola específicamente para tareas de control.

Un microprocesador es el corazón de una microcomputadora. Éste consiste en una unidad aritmética, una unidad de control y un pequeño número de unidades de memoria, llamadas registros. La figura 14 ilustra una versión simplificada de uno.

**Figura 14. Esquema simplificado de un microprocesador.**




Fuente: Bliesener, Ebel, Loeffler, Plagemann, Regber, Terzi, Winter, ***Programmable logic controllers***, pág. B-35.

La tarea de una unidad aritmética es ejecutar operaciones de aritmética y lógica con la información disponible.

El acumulador, abreviado AC, es un registro especial asignado directamente a la unidad aritmética. Él almacena información para ser procesada al igual que el resultado de alguna operación.

El registro de instrucciones guarda un comando llamado de la memoria de programa hasta que sea decodificado y ejecutado.

Un comando consiste en una parte de operación y otra de dirección. La parte de operación indica cual operación debe llevarse a cabo. La parte de dirección indica a cual operando debe aplicarse esta operación.

El contador de programa es un registro, que contiene la dirección del próximo comando a ser procesado.

La unidad de control regula y controla completamente la secuencia de las operaciones requerida para la ejecución de un comando.

## 2.4. Memoria

Los programas desarrollados para resolver tareas específicas necesitan memoria de programa para ser almacenados, y de donde se puedan leer cíclicamente por la unidad central de control.

Los requerimientos para esta memoria son relativamente simples:

- La modificación del programa o el almacenamiento de programas nuevos debe ser simple y llevarse a cabo con el dispositivo programador o computadora personal.
- El programa no debe perderse por una falla en el suministro de energía o por interferencia en el voltaje.
- El costo de la memoria debe ser bajo.
- La memoria de programa debe ser suficientemente rápida para no retrasar las operaciones de la unidad central de control.

Hoy en día, tres tipos de memoria son utilizadas:

- RAM
- EPROM
- EEPROM

## 2.5. Entradas

El módulo de entradas de un PLC es donde los sensores son conectados. Las señales de los sensores son transmitidas de éste a la unidad central de control. Las funciones del módulo de entradas son:

- Detección confiable de señales.
- Cambio de niveles de voltaje de control a niveles de voltaje lógicos.
- Protección de la electrónica de voltajes externos.
- Filtrado de señales.

Un componente utilizado que cumple con lo anterior es el optoacoplador.

El optoacoplador transmite la información del sensor utilizando luz, al mismo tiempo que crea un aislamiento eléctrico entre los circuitos de control y lógica. Esto protege la electrónica de voltajes potencialmente dañinos. Existen optoacopladores tecnológicamente avanzados que garantizan protección hasta aproximadamente 5 kilovoltios, lo que es adecuado para aplicaciones industriales.

El filtrado de señales emitidas por sensores es crítico en la automatización industrial. En la industria, las líneas eléctricas son afectadas fuertemente por voltajes de interferencia de origen inductivo, que crean impulsos en las líneas de señales. Las líneas de señales pueden ser apantalladas con blindaje o los mismos módulos de entradas pueden encargarse del filtrado de la señal utilizando retardos.

Esto último requiere que la señal de entrada sea aplicada durante un período suficientemente largo, antes de que sea reconocida como una señal de entrada. Los pulsos de interferencia, por su naturaleza inductiva, son señales

de corta duración, que pueden ser suprimidas con retardos de algunos milisegundos.

Estos retardos para las señales de entrada son ejecutados por medio de hardware, por ejemplo, circuitos RC. En algunos casos, también se puede producir retardos con software.

La duración del retardo para las señales de entrada es aproximadamente de 1 a 20 milisegundos, dependiendo del fabricante y el tipo. Muchos fabricantes ofrecen entradas rápidas especiales, cuando este retardo es demasiado largo para reconocer ciertas señales de corta duración.

Existen dos posibilidades para conectar sensores a los módulos de entradas de un PLC. Puede hacerse conexiones positivas (PNP) o negativas (NPN), o lo que es lo mismo, tipo fuente de corriente o sumidero de corriente.

## 2.6. Salidas

El módulo de salida transmite las señales de la unidad central de control a los elementos finales de control, que actúan de acuerdo a la tarea a realizar. Las funciones del módulo de salidas son las siguientes:

- Cambia los niveles de voltaje de la lógica interna al nivel de voltaje de control.
- Proteger a la electrónica de interferencias externas.
- Amplificación de potencia para accionar los elementos finales de control.
- Protección contra cortos circuitos o sobrecarga.

La protección contra cortos circuitos, la protección contra sobrecarga y la amplificación de potencia son hechas con módulos integrales. La sección de

protección contra cortos circuitos mide el flujo de corriente con una resistencia, de manera de poder desactivar la salida en el caso de un corto circuito. Un sensor de temperatura se encarga de proteger contra sobrecargas. Un transistor Darlington u otro transistor de potencia proveen la amplificación de potencia.

Si se utiliza relevadores para las salidas, entonces estos asumen prácticamente todas las funciones de un módulo de salidas. Los contactos de un relevador y su bobina están aislados eléctricamente, además el relevador representa un buen amplificador de potencia y es resistente a las sobrecargas. La protección contra cortos circuitos debe ser implementada con fusibles adicionales.

Las salidas por relevador tienen la ventaja de poder ser utilizadas con diferentes voltajes de salida. Las salidas electrónicas, por otro lado, tienen velocidades de conmutación mucho más altas y tiempos de vida más largos.

## 2.7. Funcionamiento

Los programas para el procesado convencional de información son procesados solamente una vez de principio a fin, y después son terminados. En cambio, el programa de un PLC se procesa en un ciclo continuamente.

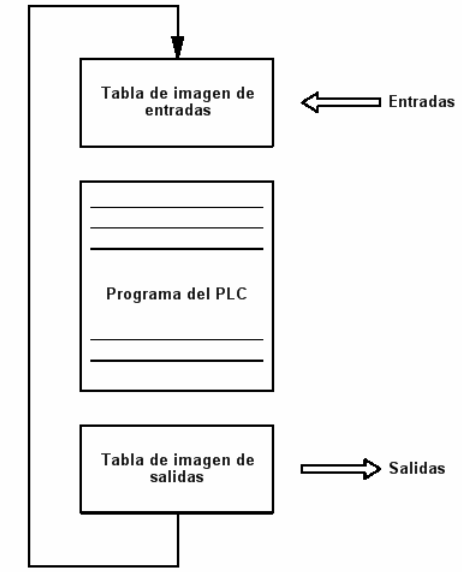
Las características del procesamiento cíclico, tal y como se observa en la figura 15, son:

- Tan pronto como el programa ha sido ejecutado una vez, este automáticamente salta al inicio y se ejecuta nuevamente.
- Al inicio de cada ciclo, el estado de las entradas es almacenado en una tabla de imagen. Esta imagen del proceso es un área separada de

memoria que es utilizada durante el ciclo de programa. El estado de las entradas permanece constante durante cada ciclo aún si han cambiado físicamente.

- Al igual que para las entradas, las salidas no son activadas o desactivadas inmediatamente, sino lo que se modifica es la tabla imagen de salidas. Solamente al final de cada ciclo, es cuando las salidas son alteradas de acuerdo al estado de la tabla imagen de salidas almacenada en memoria.

**Figura 15. Procesamiento cíclico de un programa en un PLC.**




El procesado de una línea de programa en un PLC toma algún tiempo, que puede ser de algunos microsegundos hasta algunos milisegundos, dependiendo del PLC y la operación llevada a cabo.

El tiempo requerido por un PLC para la ejecutar una vez el programa, incluyendo la actualización de las tablas imagen, es denominado tiempo de ciclo. El tiempo de ciclo depende directamente de la longitud del programa y del

tiempo de procesado de una línea de programa. Estos están entre uno a cien milisegundos.

Las consecuencias del procesamiento cíclico de un programa de PLC utilizando tablas imagen son las siguientes:

- Señales de entrada más cortas que el tiempo de ciclo posiblemente no puedan ser reconocidas.
- En algunos casos, puede que haya un retardo de dos tiempos de ciclo entre la llegada de una señal de entrada y la reacción de la correspondiente señal de salida.

Como los comandos son procesados secuencialmente, el comportamiento específico de la secuencia de un programa de PLC puede llegar a ser crucial para su buen funcionamiento.

Para algunas aplicaciones es esencial que el acceso a las entradas y salidas sea durante el ciclo. Este tipo de procesamiento del programa es soportado por algunos sistemas de PLC.

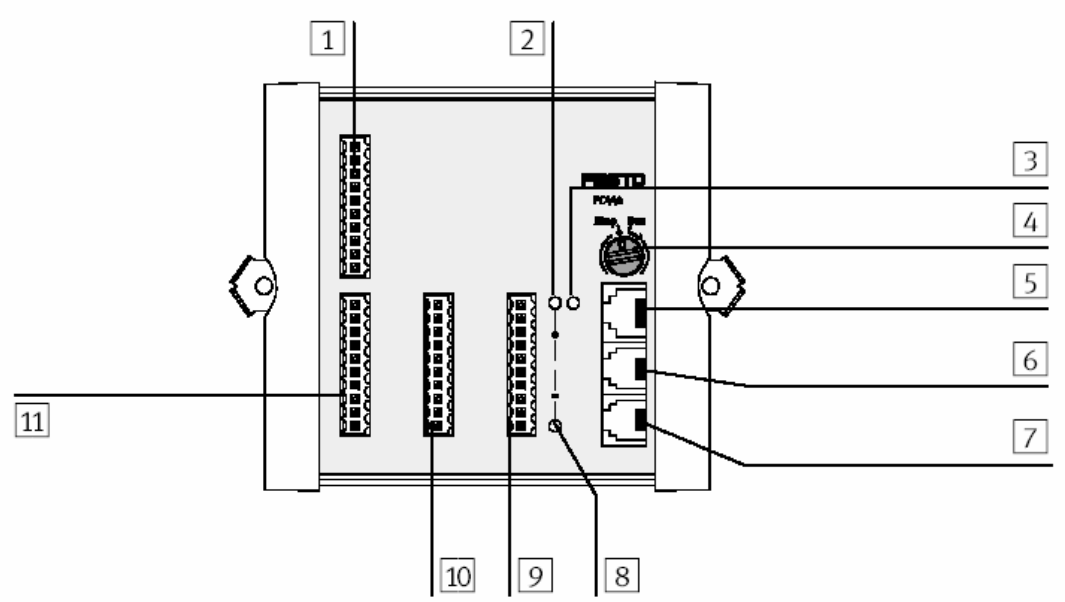
## 2.8. El PLC FEC-FC440-FST de Festo

El PLC FEC-FC440-FST de Festo tiene las siguientes características:

- Alimentación de 24 VDC.
- 16 entradas digitales PNP.
- 8 salidas digitales por transistor, 400 mA máximo por salida.
- 2 puertos seriales TTL.
- 1 puerto de red Ethernet 10BaseT.
- Procesador 80186.
- Memoria de trabajo de 512 kBytes.

- Memoria de programa de 512 kBytes flash.
- Consumo de corriente aproximado de 200 mA a 24VDC.
- Sistema operativo Festo FST.
- Dimensiones físicas de 105 mm x 35 mm x 114.2 mm.

**Figura 16. Elementos del FEC-FC440-FST.**




En la figura 16 se aprecian las siguientes partes del FEC-FC440-FST:

1. Entradas I0.0 a I0.7.
2. LED indicador de estado de energía.
3. LED indicador del estado de tráfico en la red.
4. Interruptor para arrancar o detener el programa.
5. Conexión a red 10BaseT.
6. Interfase de comunicación serial COM.
7. Interfase de comunicación serial EXT.
8. LED del estado del sistema (corriendo, detenido, error).
9. Alimentación de energía.
10. Salidas O0.0 a O0.7.
11. Entradas I1.0 a I1.7.

El PLC FEC-FC440-FST tiene los siguientes registros:

- Entradas como palabras IW0 e IW1, o como bits de I0.0 a I0.7 y de I1.0 a I1.8.
- Salidas como palabra OW0, o como bits de O0.0 a O0.7.
- Banderas como palabra de FW0 a FW9999, o como bits de Fx.0 a Fx.15.
- Registros de R0 a R255.
- Temporizadores de T0 a T255 (para cada uno también registros TWx y TPx).
- Contadores de C0 a C255 (para cada uno también registros CWx y CPx).
- Programas de P0 a P63.
- Módulos de CMP0 a CMP99 y CFM0 a CFM99.

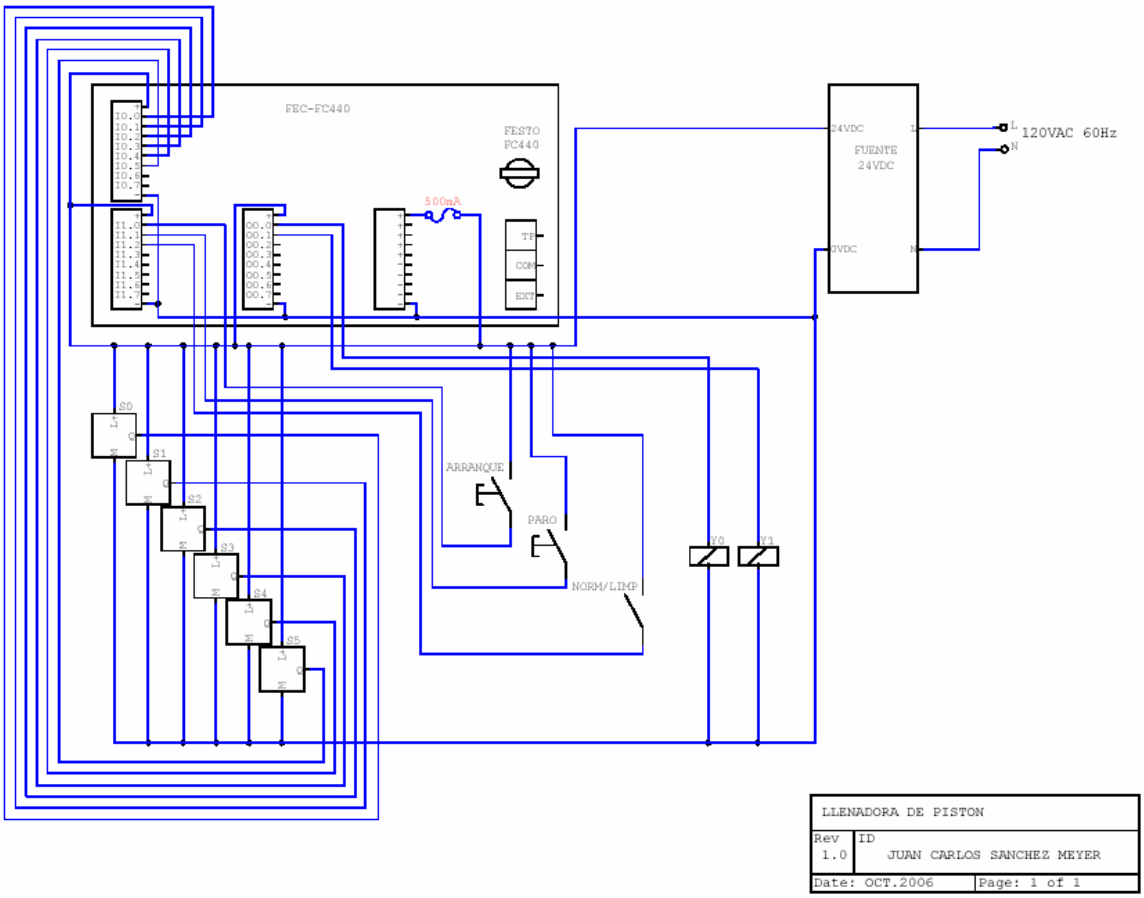
De estos registros solamente los siguientes son remanentes:

- R0 al R127.
- TP0 al TP127.
- C0 al C127.
- CP0 al CP127.
- CW0 al CW127.
- FW0 al FW255.

## 2.9. Conexiones eléctricas del PLC

En la figura 17 tenemos el diagrama de lo que es la instalación eléctrica con el PLC como elemento central. Esta instalación corresponde al diagrama de la instalación mecánica de la figura 9 y al diagrama de la instalación neumática de la figura 10. Este seguirá siendo el diagrama al que haremos referencia en el resto de este trabajo.

**Figura 17. Diagrama eléctrico de la instalación.**




## 2.10.Programación básica

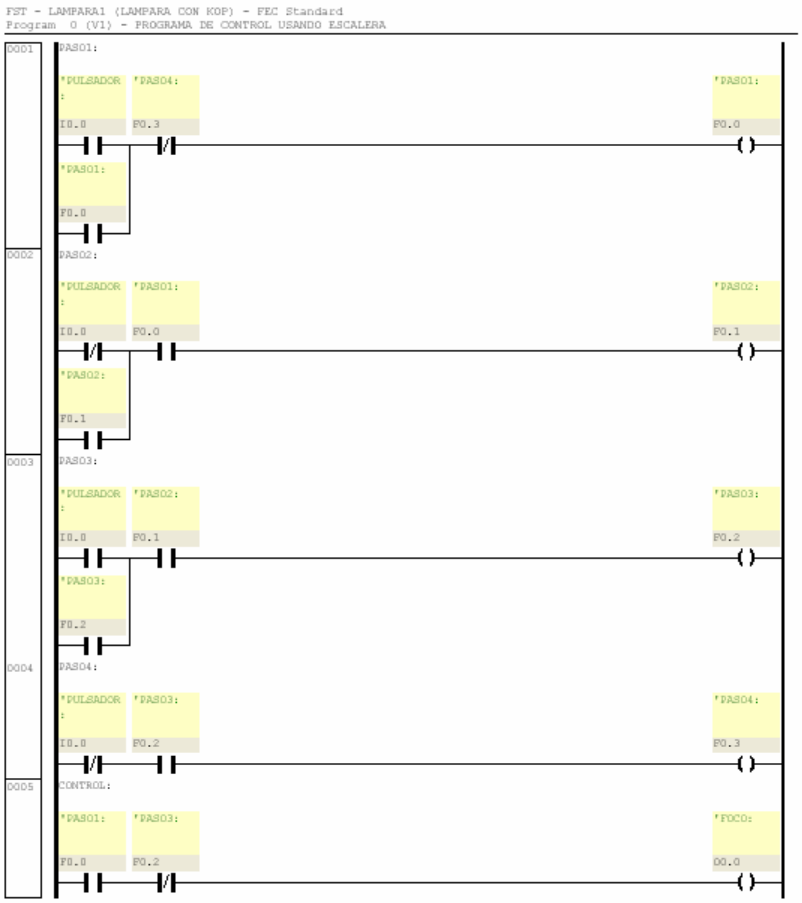
Los controladores de Festo necesitan un paquete de software para ser programados que tiene el nombre de FST4.10. Por medio de este software se puede elaborar programas en dos tipos de lenguajes diferentes, en lenguaje escalera KOP y en listado de instrucciones STL.

El lenguaje escalera KOP es de tipo gráfico y asemeja los diagramas eléctricos que se utilizan en las instalaciones de sistemas de control. Por otro lado, el listado de instrucciones STL, permite la programación de secuencias complicadas en poco tiempo, y es más adecuado para el uso de un ingeniero.

Hay investigaciones que demuestran que el 80% de los sistemas automatizados son procesos secuenciales, que necesitan de programas secuenciales. En la mayoría de casos donde se utiliza un controlador lógico programable es muy probable que se trate de un proceso secuencial.

Utilizando un simple y bien conocido ejemplo, se demostrará a continuación como un programa secuencial puede ser escrito usando el lenguaje escalera (ver figura 18). Un control para una lámpara será programado, donde la lámpara se encienda con un pulso de un pulsador, y se apague con otro pulso del mismo pulsador.

**Figura 18. Programa escrito en lenguaje escalera.**



Se puede ver como una tarea aparentemente sencilla se complica si se utiliza este lenguaje de programación. Se han utilizado las banderas, con nombres simbólicos de PASO1, PASO2, PASO3 Y PASO4, para crear la secuencia, haciendo que habiliten únicamente lo importante en el paso actual del proceso.

Ahora pasaremos a analizar la programación utilizando listado de instrucciones. Los programas en listado de instrucciones son hechos utilizando varios elementos importantes. No todos los elementos son requeridos, y la forma en la que se combinen, influencia en gran medida la operación del programa.

La jerarquía de los elementos de un programa en listado de instrucciones es la siguiente:

Programa.

Paso.

Oración.

Condición.

Acción.

Aunque el uso de la palabra STEP es opcional, la mayoría de programas en STL la utilizan. Esta palabra se usa para marcar el inicio de un bloque de código del programa.

Se le agrega una etiqueta a la palabra STEP, si este paso va a servir de destino de alguna instrucción de salto.

La oración consiste de una parte de condiciones y otra de acciones. La parte de condiciones sirve para listar una o más condiciones que deben ser

evaluadas antes de ejecutar la parte de acciones. La parte de condiciones siempre inicia con la palabra IF y la de acciones con la palabra THEN.

Un paso puede tener varias oraciones. Cuando la última oración de un paso se cumple, las acciones correspondientes son ejecutadas, y se continúa al siguiente paso. Si la última oración no se cumple, entonces el programa retorna a la primera oración del mismo paso.

**Figura 19. Programa escrito en listado de instrucciones.**

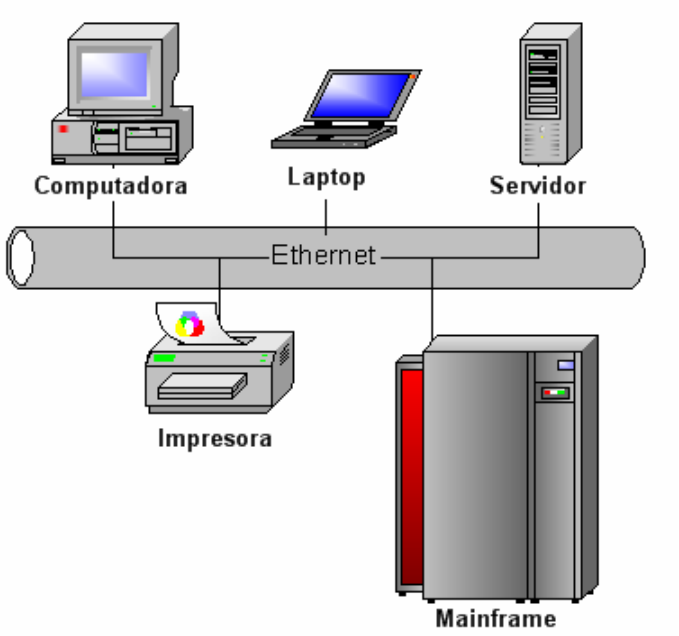
```
FST - LAMPARA1 (LAMPARA CON KOP) - FEC Standard
Program  0 (V2) - PROGRAMA DE CONTROL USANDO STL
0001 ""SE PRESIONA EL PULSADOR Y SE ACTIVA EL FOCO
0002 STEP PASO1
0003  IF                  I0.0          'PULSADOR:
0004  THEN  SET           O0.0          'FOCO:
0005
0006 ""SE SUELTA EL PULSADOR
0007 STEP PASO2
0008  IF            N     I0.0          'PULSADOR:
0009  THEN                NOP
0010
0011 ""SE PRESIONA EL PULSADOR Y SE APAGA EL FOCO
0012 STEP PASO3
0013  IF                  I0.0          'PULSADOR:
0014  THEN  RESET         O0.0          'FOCO:
0015
0016 ""SE SUELTA EL PULSADOR
0017 STEP PASO4
0018  IF            N     I0.0          'PULSADOR:
0019  THEN  JMP TO PASO1
0020
```

En la figura 19 se muestra el programa en listado de instrucciones para controlar el encendido y apagado de la lámpara usando un sólo pulsador. Se puede ver, como utilizando lenguaje inglés, se pueden resolver tareas de control de manera sencilla. Por la naturaleza de este lenguaje, es posible resolver tareas complicadas de forma sencilla y simultáneamente documentada.

# 3. CONCEPTOS BÁSICOS DE REDES

Una red es un sistema de comunicación que permite a los usuarios de computadoras compartir equipo de cómputo, aplicaciones de software, y transmitir información, voz y video. Las redes pueden unir a usuarios que se encuentren en un mismo salón hasta usuarios que se encuentren en diferentes continentes. La información de la red normalmente se transmite por cables, fibra óptica u ondas electromagnéticas, como las microondas.

**Figura 20. Una red con diferentes tipos de nodos.**



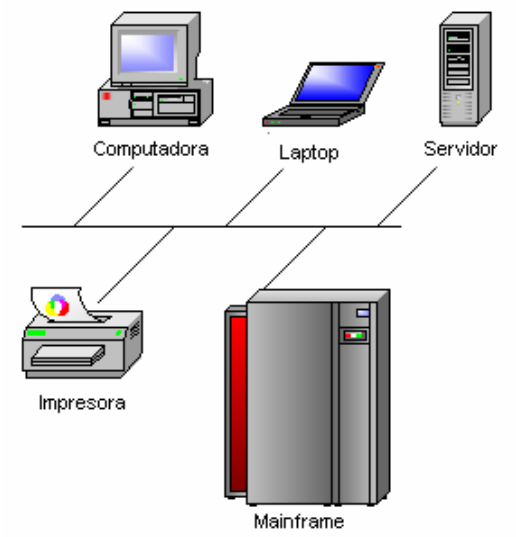

Una red se forma interconectando varios equipos, como estaciones de trabajo, servidores, impresoras, hubs, PLC´s, etc. Cada equipo en la red recibe el nombre de nodo. En la figura 20 se muestra un ejemplo de una red con varios tipos de nodos.

## 3.1. Topología de redes

El término topología de red se refiere a la disposición física de la conexión de la red. Existen tres topologías básicas: en bus, en anillo y en estrella. Cada una tiene sus ventajas y desventajas.

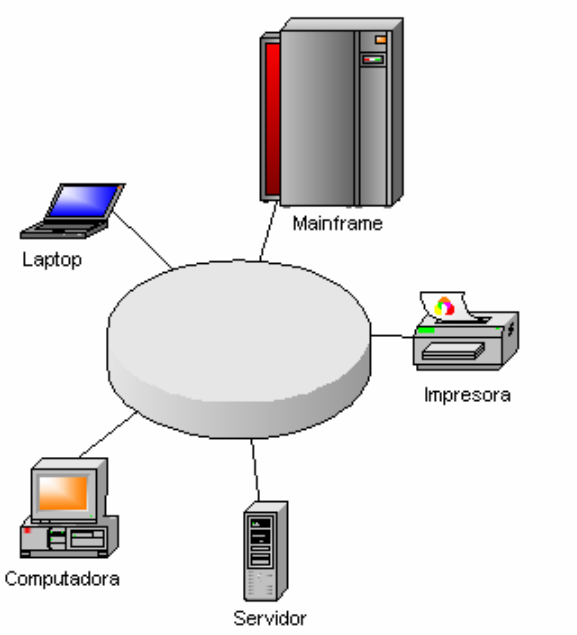
La topología de red tipo bus consiste en un cable que va de un nodo al siguiente nodo como se ve en la figura 21.

**Figura 21. Topología de red tipo bus.**



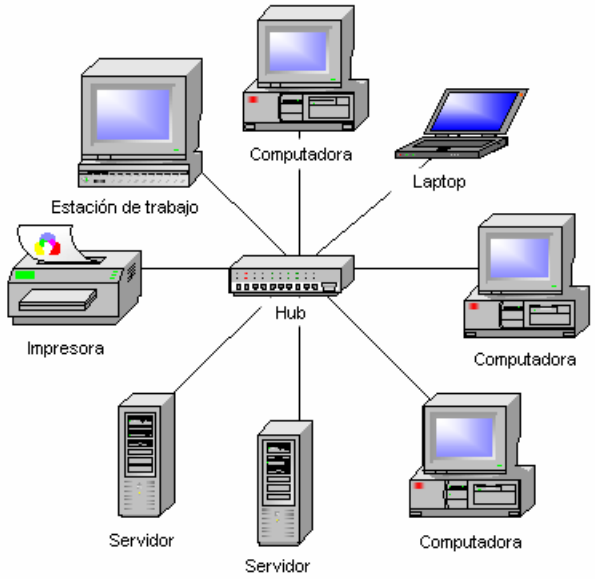

La topología de red tipo anillo es una red con forma de un anillo o un círculo, con sus nodos conectados alrededor del mismo tal y como se ve en la figura 22.

**Figura 22. Topología de red tipo anillo.**



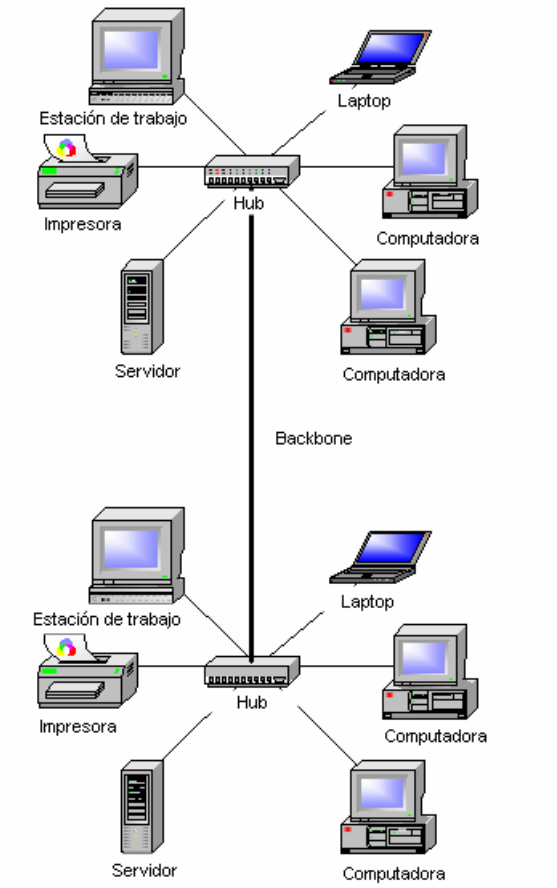

En una topología de red tipo estrella los nodos están conectados a un concentrador o hub central, de manera que la conexión se asemeja a una estrella. Se ilustra un ejemplo en la figura 23.

**Figura 23. Topología de red tipo estrella.**



La redes modernas combinan la topología tipo bus con la topología tipo estrella mediante el uso de backbones, tal y como se observa en la figura 24.

**Figura 24. Topología de red tipo bus - estrella.**




## 3.2. TCP/IP

El protocolo TCP/IP (*Transmisión Control Protocol / Internet Protocol*) es un protocolo de transmisión de paquetes utilizado en todo el mundo.

La parte TCP fue originalmente desarrollada para asegurar conexiones confiables para las redes del gobierno, ejército y educativas. En ella se aplica

un minucioso control de errores para asegurar que la información sea entregada exitosamente y sin alteraciones.

La parte IP provee el direccionamiento en la red para asegurar que los paquetes de información lleguen al destino correcto. Aquí se utiliza un sistema de notación para la dirección que consiste de cuatro números separados por un punto, por ejemplo 192.164.10.15.

Las direcciones IP son números binarios de 32 bits que contienen una dirección de subred y una dirección de host. Por ejemplo, la siguiente es una dirección IP:
11000001 00001010 00011110 00000010

No es fácil de comprender este número. Por ello, se ha agrupado en 4 secciones de 8 bits cada una, que comúnmente son llamadas octetos. Cada grupo puede ser convertido a su equivalente en sistema decimal y separado por un punto:
193.10.30.2

Este formato es el que comúnmente se utiliza para una dirección IP.

Cada dirección IP consiste de dos campos. Uno es el campo de identificación de red, que es el la dirección lógica de la red o subred a la cuál la computadora está conectada. El otro es el cambo de identificación de host, que es la dirección lógica del dispositivo, la cual identifica a cada host o nodo en una subred.

Juntas, la identificación de red y la identificación de host, permiten que cada nodo en la red tenga una dirección IP única que lo identifique.

Cuando el protocolo TCP/IP fue desarrollado, se pensó que las redes de computadora podrían caer en cualquiera de las siguientes tres categorías: Clase A, Clase B y Clase C.

Las direcciones Clase A utilizan solamente el primer octeto del número IP para identificar una subred. Los tres octetos restantes son usados para identificación de nodos. Originalmente se pensó que una red Clase A consistiría de un pequeño número de subredes con gran número de nodos en cada una.

Las direcciones Clase B utilizan los primeros dos octetos para identificar una subred. Los dos octetos restantes se usan para direcciones de nodos.

Las direcciones Clase C usan los primeros tres octetos para identificar una red. El octeto restante es usado para direcciones de nodos.

Una máscara de subred es un patrón de bits que define que porción de la dirección IP representa una dirección de subred. Por ejemplo, considere la siguiente dirección IP Clase B: 170.203.93.5. En formato binario se ve de la siguiente forma:
10101010 11001011 01011101 00000101

La máscara de subred predeterminada para una dirección Clase B es:
11111111 11111111 00000000 00000000

En notación decimal esta máscara de subred es:
255.255.0.0

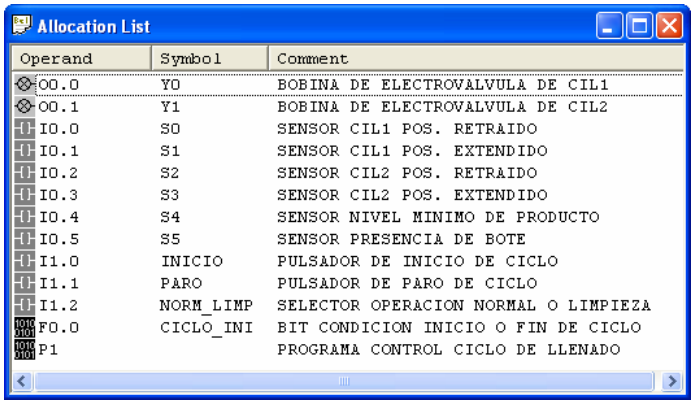
El software de la red combina la máscara de subred con la dirección IP para identificar la identificación de subred y la identificación de host para cada nodo dentro de la red. La máscara de subred hace esta operación fácil y rápida.

La información de la red como la dirección IP para un nodo, las máscaras de subred y demás características son administradas por el administrador de la red.

# 4. PROGRAMA DE CONTROL BÁSICO

El programa de control para la llenadora de pistón es relativamente sencillo. El programa se basa en lo explicado hasta ahora sobre la llenadora de pistón y sobre la instalación eléctrica. Para mayores detalles se puede regresar a consultar las figuras incluidas en los capítulos 1 y 2. En la figura 25 se ha incluido una tabla que incluye los elementos eléctricos y electrónicos utilizados, las direcciones de las entradas y salidas a las cuales están conectados y sus respectivos símbolos.

**Figura 25. Tabla de direcciones y símbolos para las entradas y salidas.**




El programa de control se ha dividido en 2 partes o programas, el programa P0 y el programa P1. Esto se hace para resolver la tarea más fácilmente y para aprovechar la capacidad de ejecución simultánea de hasta 64 programas del PLC que estamos utilizando.

El programa P0 es corto como se puede ver en la figura 26. Lo primero que hace es arrancar la ejecución del programa P1, y después se queda monitoreando el estado de los pulsadores de arranque y paro, para activar o desactivar el bit CICLO_INI que es una condición para iniciar el ciclo de la máquina.

**Figura 26. Código del programa P0.**

```
STEP
""ARRANCA PROGRAMA DE CONTROL DE CICLO DE LLENADO
 IF                  NOP
 THEN  SET           P1            'PROGRAMA CONTROL CICLO DE LLENADO

STEP LOOP
""SI SE PRESIONA EL PULSADOR DE INICIO ACTIVA BIT DE INICIO DE CICLO
 IF                  INICIO        'I1.0: PULSADOR DE INICIO DE CICLO
       AND     N     PARO          'I1.1: PULSADOR DE PARO DE CICLO
 THEN  SET           CICLO_INI     'F0.0: BIT CONDICION INICIO O FIN DE CICLO

""SE SE PRESIONA EL PULSADOR DE PARO DESACTIVA BIT DE INICIO DE CICLO
 IF                  PARO          'I1.1: PULSADOR DE PARO DE CICLO
 THEN  RESET         CICLO_INI     'F0.0: BIT CONDICION INICIO O FIN DE CICLO

""SALTA INCONDICIONALMENTE A LOOP
 IF                  NOP
 THEN  JMP TO LOOP
```

En la figura 27 se puede apreciar el código del programa P1. En este caso el programa es más largo que el programa P0, pero de igual manera permanece siendo sencillo de comprender. En el paso con la etiqueta STEP 10 se verifican las condiciones necesarias para iniciar el ciclo ya sea en modo de operación normal o en modo de operación de limpieza. Estas condiciones incluyen las señales de algunos sensores de posición, nivel y presencia, el estado del selector de modo de función y el estado del bit de inicio de ciclo que se activa y desactiva en el programa P0. Los pasos del STEP 20 al STEP 40 son el resto de la rutina para el modo de operación normal. Los pasos del STEP 50 al STEP 70 son la continuación de la rutina del modo de operación de limpieza.

**Figura 27. Código del programa P1.**

```
STEP 10
""INICIO CICLO NORMAL - MUEVE VALV. CONECTA TANQUE CON PISTON
 IF                   S1             'I0.1: SENSOR CIL1 POS. EXTENDIDO
       AND            S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND            S4             'I0.4: SENSOR NIVEL MINIMO DE PRODUCTO
       AND            S5             'I0.5: SENSOR PRESENCIA DE BOTE
       AND            CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
       AND     N      NORM_LIMP      'I1.2: SELECTOR OPERACION NORMAL O LIMPIEZA
 THEN  SET            Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2
       JMP TO 20

""INICIO CICLO LIMPIEZA - MUEVE VALV. CONECTA TANQUE CON PISTON
 IF                   S1             'I0.1: SENSOR CIL1 POS. EXTENDIDO
       AND            S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND            CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
       AND            NORM_LIMP      'I1.2: SELECTOR OPERACION NORMAL O LIMPIEZA
 THEN  SET            Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2
       JMP TO 50

STEP 20
""CICLO NORMAL - CARGA PISTON CON PRODUCTO
 IF                   S3             'I0.3: SENSOR CIL2 POS. EXTENDIDO
 THEN  SET            Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1

STEP 30
""CICLO NORMAL - MUEVE VALV. CONECTA PISTON CON BOQUILLA DE LLENADO
 IF                   S0             'I0.0: SENSOR CIL1 POS. RETRAIDO
 THEN  RESET          Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2

STEP 40
""CICLO NORMAL - DESCARGA PISTON, DOSIFICA PRODUCTO
 IF                   S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
 THEN  RESET          Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
       JMP TO 10

STEP 50
""CICLO LIMPIEZA - CARGA PISTON
 IF                   S3             'I0.3: SENSOR CIL2 POS. EXTENDIDO
 THEN  SET            Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1

STEP 60
""CICLO LIMPIEZA - MUEVE VALV. CONECTA PISTON CON BOQUILLA DE LLENADO
 IF                   S0             'I0.0: SENSOR CIL1 POS. RETRAIDO
 THEN  RESET          Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2

STEP 70
""CICLO LIMPIEZA - DESCARGA PISTON
 IF                   S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
 THEN  RESET          Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
       JMP TO 10
```

# 5. MICROSOFT EXCEL COMO HERRAMIENTA PARA DESPLEGAR INFORMACIÓN DEL PROCESO

El controlador FEC-FC440-FST puede ser conectado a una red utilizando el driver TCP/IP, permitiendo el acceso a través de otras computadoras que usen el protocolo TCP/IP. Como resultado de esto, el proceso puede ser remotamente operado, monitoreado y configurado.

Para acceder al PLC por la red, es necesario cargarle el driver TCP/IP y asignarle una dirección IP.

Para entablar comunicación con el PLC se utilizarán la colección de controles de Festo, que son controles ActiveX que permiten entablar comunicación entre plataformas PC y controladores de Festo. Estos controles permiten el desarrollo de aplicaciones de administración y monitoreo de procesos utilizando una plataforma Microsoft Windows. Algunos ejemplos de ambientes que soportan el uso de controles ActiveX son: Microsoft Office que incluye Excel, Visual Basic, Visual C++ y Delphi.

Los controles ActiveX pueden considerarse como pequeños componentes de software que implementan funcionalidades específicas y que pueden ser incorporados a aplicaciones que soporten el uso de estos. Estas aplicaciones son llamadas contenedores de ActiveX. Un programador no necesita conocer la implementación interna de un control para poderlo integrar exitosamente en su ambiente de programación favorito. Un control correctamente diseñado ofrece una interfase fácil de utilizar que consta de propiedades, métodos y eventos. Las propiedades son variables que pueden ser leídas o escritas por la

aplicación y que frecuentemente se usan para especificar como debe comportarse el control. Los métodos son las funciones que ofrece el control y que pueden ser llamadas por la aplicación para controlar alguna operación. Los eventos son señales generadas por la aplicación para informar de eventos ocurridos dentro del control. Las propiedades, los métodos y eventos proporcionan comunicación de doble vía entre la aplicación y el control ActiveX. Los ambientes de programación modernos hacen muy fácil el uso de controles ActiveX en algún proyecto, de manera que se puede hacer aplicaciones de administración y monitoreo por una pequeña fracción del costo asociado a sistemas SCADA o aplicaciones que implementen la comunicación por si solas.

## 5.1. Controles ActiveX de Festo

La colección de controles ActiveX de Festo consiste de tres. El control discoverX puede identificar PLCs en una red y puede recabar de ellos información básica, como direcciones IP, máscara de red, tipo de CPU y nombre de proyecto. El control commandX provee a la aplicación donde se implemente acceso al intérprete de comandos CI del PLC. El control exchangeX intercambia bloques de registros con el PLC, por ejemplo el estado de banderas, entradas, salidas y programas.

## 5.2. El control discoverX

El control discoverX puede ser usado para localizar e identificar PLCs en una red. El control transmite paquetes de mensajes a la red, a los cuales todos los PLCs responden con información de ellos.

Este control ofrece los métodos, propiedades y eventos listados en las tablas I, II y III.

**Tabla I. Propiedades del control discoverX.**

| Propiedades | |
|---|---|
| PacketCount | El número de paquetes a enviar. El valor predeterminado es 3. |
| Interval | El tiempo a esperar entre paquetes, dado en milisegundos. El valor predeterminado es 300. |

**Tabla II. Métodos del control discoverX.**

| Métodos | |
|---|---|
| Discover() | Inicia envío de paquetes. |
| CancelDiscover() | Detiene envío de paquetes. |
| IsDiscovering() | Interroga si el control se encuentra enviado paquetes o no. |
| GetIpcCount() | Retorna el número de PLCs detectados en la red. |
| GetIp(número) | Retorna la dirección de uno de los PLCs descubiertos. |
| GetNetMask(número) | Retorna la máscara de red de uno de los PLCs descubiertos. |
| GetMAC(número) | Retorna la dirección del hardware de uno de los PLCs descubiertos. |
| GetProject(número) | Retorna el nombre del proyecto de uno de los PLCs descubiertos. |
| GetKernel(número) | Retorna la versión de kernel de uno de los PLCs descubiertos. |
| GetDriver(número) | Retorna la versión del driver TCP/IP de uno de los PLCs descubiertos. |
| GetCPU(número) | Retorna información del CPU de uno de los PLCs descubiertos. |
| GetData(número, campo) | Retorna la información de uno de los PLCs descubiertos.<br>Campo puede tomar los siguientes valores:<br>0 – dirección IP 1 – máscara de red<br>2 – dirección MAC 3 – nombre de proyecto<br>4 – versión de kernel 5 – versión del driver TCP/IP<br>6 – información del CPU |
| About() | Retorna información del control. |

**Tabla III. Eventos del control discoverX.**

| Eventos | |
|---|---|
| SendAttemp(cuenta) | Se ejecuta cada vez que un paquete es enviado. |
| Ready(cuenta) | Se ejecuta cuando todos los paquetes han sido enviados. |
| Discovery(IP, máscara, MAC, proyecto, kernel) | Se ejecuta cuando un nuevo PLC ha sido descubierto. |

## 5.3. El control commandX

El control commandX puede ser usado para acceder al intérprete de comandos CI de un PLC. Con este control, se puede controlar la mayoría de funciones de un PLC. Las respuestas de todos los comandos enviados al PLC, son retornados, de manera que se puede utilizar este control para hacer consultas.

El control commandX ofrece los métodos, propiedades y eventos detallados en las tablas IV, V y VI.

**Tabla IV. Propiedades del control commandX.**

| Propiedades | |
|---|---|
| PacketCount | El número máximo de paquetes a enviar. El valor predeterminado es 3. |
| Interval | El tiempo a esperar antes de retransmitir un paquete en milisegundos. |
| Ip | La dirección IP del PLC al que se le envía el comando. |

**Tabla V. Métodos del control commandX.**

| Métodos | |
|---|---|
| SendCommand() | Envía un paquete con comandos. |
| CancelCommand() | Detiene el envío de paquetes. |
| IsSending() | Consulta si se está enviando paquetes con comandos o no. |
| HasResult() | Consulta si se ha recibido respuesta al comando enviado. |
| GetResult() | Retorna el resultado recibido al comando enviado. |
| About() | Retorna información del control. |

**Tabla VI. Eventos del control commandX.**

| Eventos | |
|---|---|
| SendAttempt(cuenta) | Se ejecuta cuando se envía un paquete con comandos. |
| ResultReceived(resultado) | Se ejecuta cuando se ha recibido respuesta a un comando enviado. |
| Timeout() | Se ejecuta cuando no se ha recibido respuesta a un comando enviado después de cierto tiempo. |

## 5.4. El control exchangeX

El control exchangeX permite el intercambio de bloques de datos entre un PLC y una computadora. Es posible intercambiar valores de entradas, salidas, banderas, programas, registros y temporizadores. Cada paquete puede contener información para hasta 255 elementos.

El control exchangeX ofrece los métodos, propiedades y eventos detallados en las tablas VII, VIII y IX.

**Tabla VII. Propiedades del control exchangeX.**

| Propiedades | |
|---|---|
| PacketCount | El número máximo de paquetes a enviar. El valor predeterminado es 3. |
| Interval | El tiempo a esperar antes de retransmitir un paquete en milisegundos. |
| Ip | La dirección IP del PLC al que se le envía el comando. |
| OperandOffset | La posición inicial de los elementos de memoria a enviar o recibir. |
| OperandCount | El número de elementos de memoria a intercambiar. (1 a 255) |
| Mode | La dirección de intercambio de información.<br>0 – lectura 1 – escritura |
| Type | El tipo de información a intercambiar.<br>0 – entradas 1 – salidas<br>2 – banderas 3 – programas<br>4 – registros 5 – temporizadores |

**Tabla VIII. Métodos del control exchangeX.**

| Métodos | |
|---|---|
| Exchange() | Intercambia la información. |
| CancerExchange() | Detiene el intercambio de información. |
| IsExchanging() | Consulta si el control está o no intercambiando información. |
| HasResponse() | Consulta si se ha recibido respuesta o no. |
| GetOperand(número) | Retorna el valor de un elemento. |
| SetOperand(número, valor) | Ajusta el valor de un elemento. |
| About() | Retorna información del control. |

**Tabla IX. Eventos del control exchangeX.**

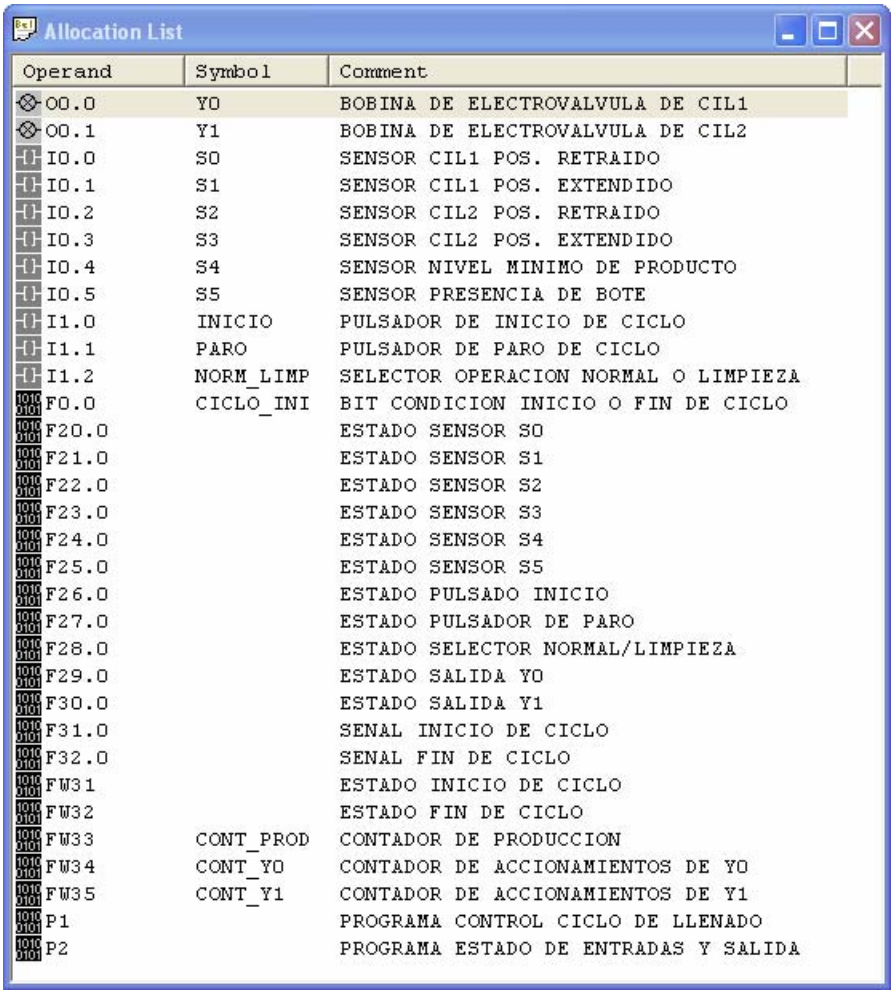
| Eventos | |
|---|---|
| SendAttempt(cuenta) | Se ejecuta cada vez que se manda un paquete. |
| ResultReceived() | Se ejecuta cuando un bloque de elementos o alguna confirmación son recibidos. |
| Timeout() | Se ejecuta cuando no se recibe respuesta después de un determinado tiempo. |

## 5.5. Modificaciones al programa del PLC para implementar funciones de comunicación

El programa de control de la máquina permanece muy parecido al explicado en el capítulo 4. Se ha agregado ahora algunos bits para facilitar el intercambio de datos con Excel, algunos contadores para conteo de accionamientos de los elementos y de producto producido. También se ha agregado un tercer programa que copia el estado de las entradas y salidas a

algunos bits, con el propósito de acceder más fácilmente a esta información desde Excel. En la figura 28 se muestra el listado de elementos utilizados en el nuevo programa, que incluye entradas, salidas y banderas.

**Figura 28. Lista de direcciones y símbolos para las entradas y salidas.**




| Operand | Symbol | Comment |
|---|---|---|
| O0.0 | Y0 | BOBINA DE ELECTROVALVULA DE CIL1 |
| O0.1 | Y1 | BOBINA DE ELECTROVALVULA DE CIL2 |
| I0.0 | S0 | SENSOR CIL1 POS. RETRAIDO |
| I0.1 | S1 | SENSOR CIL1 POS. EXTENDIDO |
| I0.2 | S2 | SENSOR CIL2 POS. RETRAIDO |
| I0.3 | S3 | SENSOR CIL2 POS. EXTENDIDO |
| I0.4 | S4 | SENSOR NIVEL MINIMO DE PRODUCTO |
| I0.5 | S5 | SENSOR PRESENCIA DE BOTE |
| I1.0 | INICIO | PULSADOR DE INICIO DE CICLO |
| I1.1 | PARO | PULSADOR DE PARO DE CICLO |
| I1.2 | NORM_LIMP | SELECTOR OPERACION NORMAL O LIMPIEZA |
| F0.0 | CICLO_INI | BIT CONDICION INICIO O FIN DE CICLO |
| F20.0 | | ESTADO SENSOR S0 |
| F21.0 | | ESTADO SENSOR S1 |
| F22.0 | | ESTADO SENSOR S2 |
| F23.0 | | ESTADO SENSOR S3 |
| F24.0 | | ESTADO SENSOR S4 |
| F25.0 | | ESTADO SENSOR S5 |
| F26.0 | | ESTADO PULSADO INICIO |
| F27.0 | | ESTADO PULSADOR DE PARO |
| F28.0 | | ESTADO SELECTOR NORMAL/LIMPIEZA |
| F29.0 | | ESTADO SALIDA Y0 |
| F30.0 | | ESTADO SALIDA Y1 |
| F31.0 | | SENAL INICIO DE CICLO |
| F32.0 | | SENAL FIN DE CICLO |
| FW31 | | ESTADO INICIO DE CICLO |
| FW32 | | ESTADO FIN DE CICLO |
| FW33 | CONT_PROD | CONTADOR DE PRODUCCION |
| FW34 | CONT_Y0 | CONTADOR DE ACCIONAMIENTOS DE Y0 |
| FW35 | CONT_Y1 | CONTADOR DE ACCIONAMIENTOS DE Y1 |
| P1 | | PROGRAMA CONTROL CICLO DE LLENADO |
| P2 | | PROGRAMA ESTADO DE ENTRADAS Y SALIDA |

El único cambio que se le ha hecho al programa P0 se muestra en la figura 29. Ahora al inicio además de arrancar el programa P1, también se arranca el programa P2 y se inicializan a cero los valores de algunos registros.

**Figura 29. Código del programa P0 modificado.**

```
STEP
""ARRANCA PROGRAMA DE CONTROL DE CICLO DE LLENADO Y AJUSTE INICIAL
 IF                  NOP
 THEN  SET           P1            'PROGRAMA CONTROL CICLO DE LLENADO
       SET           P2            'PROGRAMA ESTADO DE ENTRADAS Y SALIDA
       LOAD          V0
         TO          CONT_PROD     'FW33: CONTADOR DE PRODUCCION
         TO          FW31          'ESTADO INICIO DE CICLO
         TO          FW32          'ESTADO FIN DE CICLO

STEP LOOP
""SI SE PRESIONA EL PULSADOR DE INICIO ACTIVA BIT DE INICIO DE CICLO
 IF                  INICIO        'I1.0: PULSADOR DE INICIO DE CICLO
       AND     N     PARO          'I1.1: PULSADOR DE PARO DE CICLO
 THEN  SET           CICLO_INI     'F0.0: BIT CONDICION INICIO O FIN DE CICLO

""SE SE PRESIONA EL PULSADOR DE PARO DESACTIVA BIT DE INICIO DE CICLO
 IF                  PARO          'I1.1: PULSADOR DE PARO DE CICLO
 THEN  RESET         CICLO_INI     'F0.0: BIT CONDICION INICIO O FIN DE CICLO

""SALTA INCONDICIONALMENTE A LOOP
 IF                  NOP
 THEN  JMP TO LOOP
```

En el programa P1, mostrado en la figura 30, ahora se ha incluido al inicio un nuevo paso STEP 0 que se encarga de activar un bit y poner a cero el contador de producto fabricado en el período que permanece encendida la máquina. Este bit que se ha mencionado notifica a la aplicación en Excel que se ha iniciado la operación de la máquina. Se han incorporado ahora tres contadores, uno para el número de operaciones del cilindro 1, otro para el número de operaciones del cilindro 2 y el tercero, que ya se mencionó, para contar el producto fabricado. Se puede ver a lo largo del programa los puntos donde se incrementan los valores de los contadores. Al final del programa, para el momento en el que se detiene la operación de la máquina, se activa otro bit, con el fin de notificarle a Excel de este evento.

## Figura 30. Código del programa P1 modificado.

```
""INICIO DE CICLO
STEP 0
""ACTIVA SENAL DE INICIO DEL CICLO PARA INTERACCION CON EXCEL
 IF                  CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
 THEN  SET           F31.0          'SENAL INICIO DE CICLO
       LOAD          V0
         TO          CONT_PROD      'FW33: CONTADOR DE PRODUCCION

STEP 10
""INICIO CICLO NORMAL - MUEVE VALV. CONECTA TANQUE CON PISTON
 IF                  S1             'I0.1: SENSOR CIL1 POS. EXTENDIDO
       AND           S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND           S4             'I0.4: SENSOR NIVEL MINIMO DE PRODUCTO
       AND           S5             'I0.5: SENSOR PRESENCIA DE BOTE
       AND           CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
       AND     N     NORM_LIMP      'I1.2: SELECTOR OPERACION NORMAL O LIMPIEZA
 THEN  SET           Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2
                                    "INCREMENTA CONTADOR DE ACCIONAMIENTOS DE Y1
       INC           CONT_Y1        'FW35: CONTADOR DE ACCIONAMIENTOS DE Y1
       JMP TO 20

""INICIO CICLO LIMPIEZA - MUEVE VALV. CONECTA TANQUE CON PISTON
 IF                  S1             'I0.1: SENSOR CIL1 POS. EXTENDIDO
       AND           S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND           CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
       AND           NORM_LIMP      'I1.2: SELECTOR OPERACION NORMAL O LIMPIEZA
 THEN  SET           Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2
                                    "INCREMENTA CONTADOR DE ACCIONAMIENTOS DE Y1
       INC           CONT_Y1        'FW35: CONTADOR DE ACCIONAMIENTOS DE Y1
       JMP TO 50

STEP 20
""CICLO NORMAL - CARGA PISTON CON PRODUCTO
 IF                  S3             'I0.3: SENSOR CIL2 POS. EXTENDIDO
 THEN  SET           Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
                                    "INCREMENTA CONTADOR DE ACCIONAMIENTOS DE Y0
       INC           CONT_Y0        'FW34: CONTADOR DE ACCIONAMIENTOS DE Y0

STEP 30
""CICLO NORMAL - MUEVE VALV. CONECTA PISTON CON BOQUILLA DE LLENADO
 IF                  S0             'I0.0: SENSOR CIL1 POS. RETRAIDO
 THEN  RESET         Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2

STEP 40
""CICLO NORMAL - DESCARGA PISTON, DOSIFICA PRODUCTO
 IF                  S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND           CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
 THEN  RESET         Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
                                    "INCREMENTA CONTADOR DE UNIDADES PRODUCIDAS
       INC           CONT_PROD      'FW33: CONTADOR DE PRODUCCION
       JMP TO 10

 IF                  S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND     N     CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
 THEN  RESET         Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
                                    "INCREMENTA CONTADOR DE UNIDADES PRODUCIDAS
       INC           CONT_PROD      'FW33: CONTADOR DE PRODUCCION
       SET           F32.0          'SENAL FIN DE CICLO
       JMP TO 0

STEP 50
""CICLO LIMPIEZA - CARGA PISTON
 IF                  S3             'I0.3: SENSOR CIL2 POS. EXTENDIDO
 THEN  SET           Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
                                    "INCREMENTA CONTADOR DE ACCIONAMIENTOS DE Y0
       INC           CONT_Y0        'FW34: CONTADOR DE ACCIONAMIENTOS DE Y0

STEP 60
""CICLO LIMPIEZA - MUEVE VALV. CONECTA PISTON CON BOQUILLA DE LLENADO
 IF                  S0             'I0.0: SENSOR CIL1 POS. RETRAIDO
 THEN  RESET         Y1             'O0.1: BOBINA DE ELECTROVALVULA DE CIL2

STEP 70
""CICLO LIMPIEZA - DESCARGA PISTON
 IF                  S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND           CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
 THEN  RESET         Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
       JMP TO 10

 IF                  S2             'I0.2: SENSOR CIL2 POS. RETRAIDO
       AND     N     CICLO_INI      'F0.0: BIT CONDICION INICIO O FIN DE CICLO
 THEN  RESET         Y0             'O0.0: BOBINA DE ELECTROVALVULA DE CIL1
                                    "ACTIVA SENAL DE FIN DE CICLO PARA INTERACCION CON EXCEL
       SET           F32.0          'SENAL FIN DE CICLO
       JMP TO 0
```

En la figura 31 se puede ver el nuevo programa P2 que se ha agregado. Este programa copia el estado de cada entrada y cada salida a varios bits, para poder leer los valores desde Excel más fácilmente.

**Figura 31. Código del programa P2 modificado.**

```
STEP LOOP
 IF                  NOP
 THEN  LOAD          S0            'IO.0: SENSOR CIL1 POS. RETRAIDO
         TO          F20.0         'ESTADO SENSOR S0
       LOAD          S1            'IO.1: SENSOR CIL1 POS. EXTENDIDO
         TO          F21.0         'ESTADO SENSOR S1
       LOAD          S2            'IO.2: SENSOR CIL2 POS. RETRAIDO
         TO          F22.0         'ESTADO SENSOR S2
       LOAD          S3            'IO.3: SENSOR CIL2 POS. EXTENDIDO
         TO          F23.0         'ESTADO SENSOR S3
       LOAD          S4            'IO.4: SENSOR NIVEL MINIMO DE PRODUCTO
         TO          F24.0         'ESTADO SENSOR S4
       LOAD          S5            'IO.5: SENSOR PRESENCIA DE BOTE
         TO          F25.0         'ESTADO SENSOR S5
       LOAD          INICIO        'I1.0: PULSADOR DE INICIO DE CICLO
         TO          F26.0         'ESTADO PULSADO INICIO
       LOAD          PARO          'I1.1: PULSADOR DE PARO DE CICLO
         TO          F27.0         'ESTADO PULSADOR DE PARO
       LOAD          NORM_LIMP     'I1.2: SELECTOR OPERACION NORMAL O LIMPIEZA
         TO          F28.0         'ESTADO SELECTOR NORMAL/LIMPIEZA
       LOAD          Y0            'OO.0: BOBINA DE ELECTROVALVULA DE CIL1
         TO          F29.0         'ESTADO SALIDA Y0
       LOAD          Y1            'OO.1: BOBINA DE ELECTROVALVULA DE CIL2
         TO          F30.0         'ESTADO SALIDA Y1
       JMP TO LOOP
|
```

## 5.6. Diseño de las hojas de Excel para la comunicación al PLC

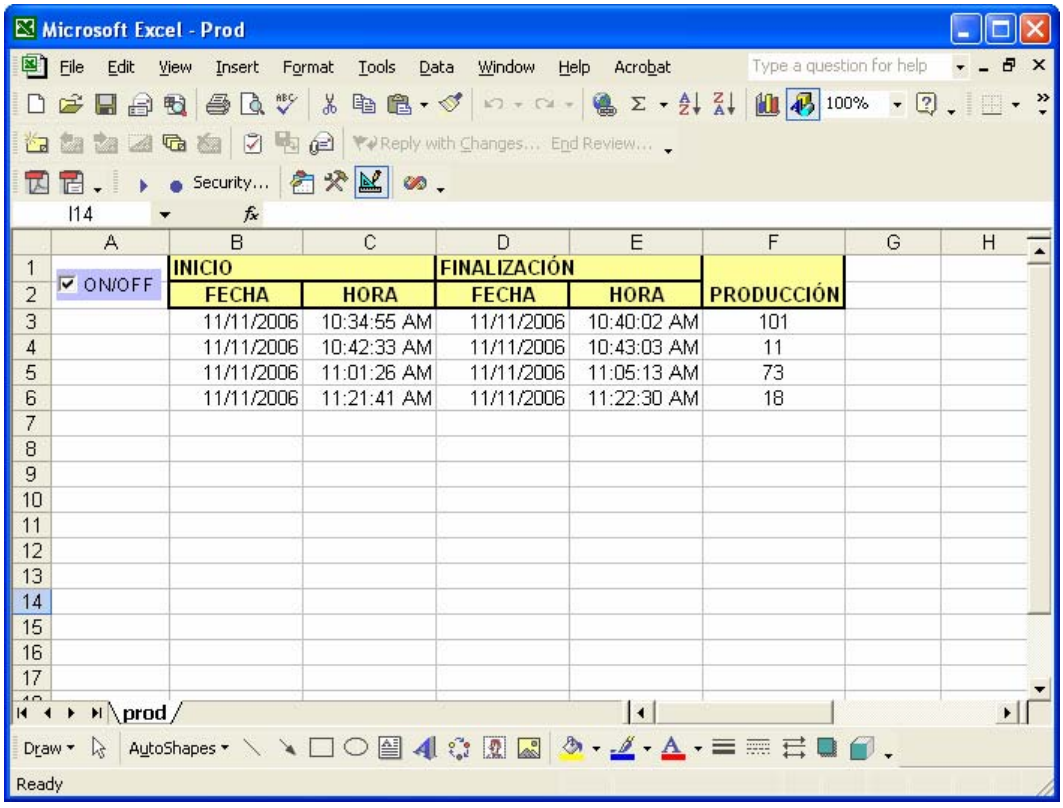
El diseño de las hojas de Excel para implementar la comunicación con el PLC es sencillo. Se basa en lo que cualquier persona, que ya ha utilizado Excel para trabajar hojas de cálculo, conoce, sumado a lo que se ha explicado de los controles ActiveX y a conocimientos de programación básica aplicados al lenguaje Visual Basic.

La hoja de Excel se utiliza para desplegar información en sus celdas, y para trabajarla posteriormente de manera convencional. La comunicación al PLC se implementa utilizando los controles ActiveX de Festo y programando cierta lógica en el ambiente de programación Visual Basic que Excel

normalmente ya incluye. Esto transforma a Excel en una poderosa herramienta para este tipo de aplicaciones.

Para nuestra llenadora se ha diseñado dos diferentes hojas de Excel. Una tiene la finalidad de proveer al departamento de producción con información básica sobre el desempeño de la máquina, y la otra se ha hecho pensando en una ayuda al departamento de mantenimiento para mantenerla trabajando en buen estado. A continuación se examinarán ambas a detalle.

**Figura 32. Vista de la hoja de Excel para el departamento de producción.**




En la figura 32 se muestra como se ve la hoja diseñada para el departamento de producción. A simple vista se ve muy sencilla, pero no deja

por eso de contener información muy importante de la cual, se pueden derivar varios indicadores. Vemos que esta hoja al estar en comunicación con el PLC, automáticamente tabula en la columna B y C la fecha y hora de inicio de la actividad de la máquina. De igual forma, al terminarse la operación de la máquina, tabula en la columna D y E la fecha y hora de finalización de actividades, y además tabula en la columna F la cantidad de unidades producidas. De esta forma se puede tener los registros de la forma en que ha estado trabajando la máquina, y permite al departamento de producción calcular indicadores a partir de la información básica que se ha adquirido. Al hablar de indicadores, se habla por ejemplo de poder calcular cuantas unidades se producen por unidad de tiempo, cuanto tiempo diario se aprovecha productivamente la máquina, o cuánto tiempo permanece parada.

En las figuras 33, 34 y 35 se incluye todo el código con comentarios del programa escrito en Visual Basic correspondiente a esta hoja de Excel.

**Figura 33. Primera parte del código de Visual Basic detrás de la hoja de Excel para el departamento de producción.**

```
Private Sub CheckBox1_Click()
'Habilita o deshabilita Timer, que es el que se encarga de refrescar
'la información
    If CheckBox1.Value = True Then
        IeTimer1.Interval = 1000    'Tiempo en milisegundos
        IeTimer1.Enabled = True
    Else
        IeTimer1.Enabled = False
    End If

End Sub

Private Sub IeTimer1_Timer()
    'Solicita información al PLC cada periodo de tiempo determinado por el timer
    PideDatosAlPlc

End Sub

Sub PideDatosAlPlc()

    'Pide datos de FW31, FW32 y FW33
    ExchangeX1.Ip = "10.10.10.5"    'Direccion IP del PLC
    ExchangeX1.OperandOffset = 31   'Numero de palabra de inicio de los registros a intercambiar
    ExchangeX1.OperandCount = 3     'Cantidad de palabras a intercambiar a partir de la palabra de inicio
    ExchangeX1.Mode = 0             'Modo 0=lectura 1=escritura
    ExchangeX1.Type = 2             'Tipo de registro 2=Flag

    ExchangeX1.Exchange

End Sub
```

**Figura 34. Segunda parte del código de Visual Basic detrás de la hoja de Excel para el departamento de producción.**

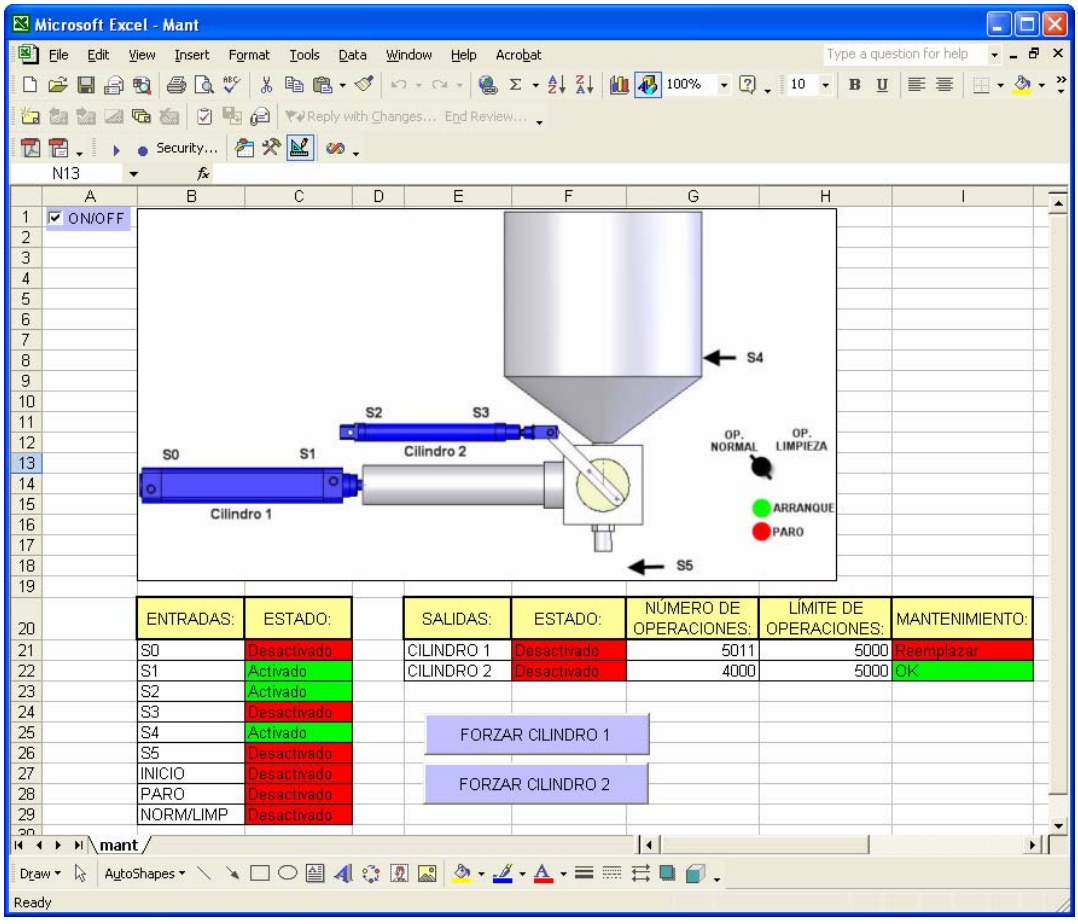
```
Private Sub ExchangeX1_ResultReceived()

    Dim SenalIni As Integer
    Dim SenalFin As Integer
    Dim ContProd As Integer

    If ExchangeX1.Mode = 0 Then
        'Recibe la información del PLC
        SenalIni = ExchangeX1.GetOperand(0) 'Valor de FW31
        SenalFin = ExchangeX1.GetOperand(1) 'Valor de FW32
        ContProd = ExchangeX1.GetOperand(2) 'Valor de FW33

        'Si ha iniciado operación
        If SenalIni = 1 Then
            CeldaLibre = 1
            BuscaCeldaLibre
            Sheets("prod").Cells(CeldaLibre, 2).Value = Date    'Imprime fecha de inicio
            Sheets("prod").Cells(CeldaLibre, 3).Value = Time    'Imprime hora de inicio
            SenalIniACero
        End If

        'Si ha finalizado operación
        If SenalFin = 1 Then
            Sheets("prod").Cells(CeldaLibre, 4).Value = Date    'Imprime fecha de finalización
            Sheets("prod").Cells(CeldaLibre, 5).Value = Time    'Imprime hora de finalización
            Sheets("prod").Cells(CeldaLibre, 6).Value = ContProd  'Imprime cantidad de elementos producidos
            SenalFinACero
        End If

    End If

End Sub
```

**Figura 35. Tercera parte del código de Visual Basic detrás de la hoja de Excel para el departamento de producción.**

```
Sub BuscaCeldaLibre()
    'Busca la primer celda libre en la hoja de cálculo
    Do While Sheets("prod").Cells(CeldaLibre, 2).Value <> ""
        CeldaLibre = CeldaLibre + 1
    Loop

End Sub

Sub SenalIniACero()
    'Le carga el valor 0 a FW31
    CommandX1.Ip = "10.10.10.5"     'Direccion IP del PLC
    CommandX1.Command = "mmw31=0"   'Comando CI a enviar para poner a cero FW31

    CommandX1.SendCommand

End Sub

Sub SenalFinACero()
    'Le carga el valor 0 a FW32
    CommandX1.Ip = "10.10.10.5"     'Direccion IP del PLC
    CommandX1.Command = "mmw32=0"   'Comando CI a enviar para poner a cero FW32

    CommandX1.SendCommand

End Sub
```

La hoja de Excel diseñada para el departamento de mantenimiento es mostrada en la figura 36. En ella se incluye, en primer lugar, una imagen de la máquina, con la localización física de cada elemento, para mejor interpretación de la información. En las columnas B y C se agrupan todas las entradas del PLC y se indica su estado. Cuando una entrada está activa, se imprime en la celda la palabra “Activado” y la celda se torna color verde. Cuando está inactiva, se imprime la palabra “Desactivado” y la celda se torna color rojo. De igual manera funciona para las salidas, incluidas en las columnas E y F.

**Figura 36. Vista de la hoja de Excel para el departamento de mantenimiento.**




| ENTRADAS: | ESTADO: |
|---|---|
| S0 | Desactivado |
| S1 | Activado |
| S2 | Activado |
| S3 | Desactivado |
| S4 | Activado |
| S5 | Desactivado |
| INICIO | Desactivado |
| PARO | Desactivado |
| NORM/LIMP | Desactivado |

| SALIDAS: | ESTADO: | NÚMERO DE OPERACIONES: | LÍMITE DE OPERACIONES: | MANTENIMIENTO: |
|---|---|---|---|---|
| CILINDRO 1 | Desactivado | 5011 | 5000 | Reemplazar |
| CILINDRO 2 | Desactivado | 4000 | 5000 | OK |

Además, en la columna G aparece el valor del contador de accionamientos para cada cilindro, que es comparado con el límite de accionamientos máximo de la columna H. Cuando el valor del contador supera el límite establecido, se imprime en la columna I, un mensaje que indica que es necesario cambiar el elemento. Para terminar, se han incluido dos botones que tienen la función de conmutar el estado de los cilindros. Es decir, si el cilindro se encuentra desactivado, al presionar el botón, este se activa, o si se encuentra ya activo, entonces se desactiva.

**Figura 37. Primera parte del código de Visual Basic detrás de la hoja de Excel para el departamento de mantenimiento.**

```
Private Sub CheckBox1_Click()
'Habilita o deshabilita Timer, que es el que se encarga de refrescar
'la información
    If CheckBox1.Value = True Then
        IeTimer1.Interval = 1000    'Tiempo en milisegundos
        IeTimer1.Enabled = True
    Else
        IeTimer1.Enabled = False
    End If

End Sub

Private Sub CommandButton1_Click()
    'Forzar Cilindro 1
    'Si esta desactivado lo activa
    If CILINDRO1 = 0 Then
        CommandX1.Ip = "10.10.10.5"     'Direccion IP del PLC
        CommandX1.Command = "ma0.0=1"     'Comando CI a enviar
        CommandX1.SendCommand
    End If
    'Si esta activado lo desactiva
    If CILINDRO1 = 1 Then
        CommandX1.Ip = "10.10.10.5"     'Direccion IP del PLC
        CommandX1.Command = "ma0.0=0"     'Comando CI a enviar
        CommandX1.SendCommand
    End If

End Sub

Private Sub CommandButton2_Click()
    'Forzar Cilindro 2
    'Si esta desactivado lo activa
    If CILINDRO2 = 0 Then
        CommandX1.Ip = "10.10.10.5"     'Direccion IP del PLC
        CommandX1.Command = "ma0.0=1"     'Comando CI a enviar
        CommandX1.SendCommand
    End If
    'Si esta activado lo desactiva
    If CILINDRO2 = 1 Then
        CommandX1.Ip = "10.10.10.5"     'Direccion IP del PLC
        CommandX1.Command = "ma0.0=0"     'Comando CI a enviar
        CommandX1.SendCommand
    End If

End Sub
```

**Figura 38. Segunda parte del código de Visual Basic detrás de la hoja de Excel para el departamento de mantenimiento.**

```
Private Sub IeTimer1_Timer()
    'Solicita información al PLC cada período de tiempo determinado por el timer
    PideDatosAlPlc

    'Imprime valores de contadores de accionamientos
    Workbooks("Mant.xls").Sheets("mant").Cells(21, 7).Value = CONT_Y0
    Workbooks("Mant.xls").Sheets("mant").Cells(22, 7).Value = CONT_Y1

    'Actualiza estado de S0
    If S0 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(21, 3).Value = "Desactivado"
        Worksheets("mant").Range("C21").Interior.ColorIndex = 3 'rojo
    End If

    If S0 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(21, 3).Value = "Activado"
        Worksheets("mant").Range("C21").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de S1
    If S1 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(22, 3).Value = "Desactivado"
        Worksheets("mant").Range("C22").Interior.ColorIndex = 3 'rojo
    End If

    If S1 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(22, 3).Value = "Activado"
        Worksheets("mant").Range("C22").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de S2
    If S2 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(23, 3).Value = "Desactivado"
        Worksheets("mant").Range("C23").Interior.ColorIndex = 3 'rojo
    End If

    If S2 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(23, 3).Value = "Activado"
        Worksheets("mant").Range("C23").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de S3
    If S3 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(24, 3).Value = "Desactivado"
        Worksheets("mant").Range("C24").Interior.ColorIndex = 3 'rojo
    End If

    If S3 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(24, 3).Value = "Activado"
        Worksheets("mant").Range("C24").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de S4
    If S4 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(25, 3).Value = "Desactivado"
        Worksheets("mant").Range("C25").Interior.ColorIndex = 3 'rojo
    End If

    If S4 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(25, 3).Value = "Activado"
        Worksheets("mant").Range("C25").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de S5
    If S5 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(26, 3).Value = "Desactivado"
        Worksheets("mant").Range("C26").Interior.ColorIndex = 3 'rojo
    End If

    If S5 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(26, 3).Value = "Activado"
        Worksheets("mant").Range("C26").Interior.ColorIndex = 4 'verde
    End If
```

**Figura 39. Tercera parte del código de Visual Basic detrás de la hoja de Excel para el departamento de mantenimiento.**

```
    'Actualiza estado de pulsador INICIO
    If INICIO = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(27, 3).Value = "Desactivado"
        Worksheets("mant").Range("C27").Interior.ColorIndex = 3 'rojo
    End If

    If INICIO = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(27, 3).Value = "Activado"
        Worksheets("mant").Range("C27").Interior.ColorIndex = 4 'verde
    End If
    'Actuliza estado de pulsador PARO
    If PARO = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(28, 3).Value = "Desactivado"
        Worksheets("mant").Range("C28").Interior.ColorIndex = 3 'rojo
    End If

    If PARO = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(28, 3).Value = "Activado"
        Worksheets("mant").Range("C28").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de selector NORM/LIMP
    If NORM_LIMP = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(29, 3).Value = "Desactivado"
        Worksheets("mant").Range("C29").Interior.ColorIndex = 3 'rojo
    End If

    If NORM_LIMP = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(29, 3).Value = "Activado"
        Worksheets("mant").Range("C29").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de Cilindro 1
    If CILINDRO1 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(21, 6).Value = "Desactivado"
        Worksheets("mant").Range("F21").Interior.ColorIndex = 3 'rojo
    End If

    If CILINDRO1 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(21, 6).Value = "Activado"
        Worksheets("mant").Range("F21").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza estado de Cilindro 2
    If CILINDRO2 = 0 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(22, 6).Value = "Desactivado"
        Worksheets("mant").Range("F22").Interior.ColorIndex = 3 'rojo
    End If

    If CILINDRO2 = 1 Then
        Workbooks("Mant.xls").Sheets("mant").Cells(22, 6).Value = "Activado"
        Worksheets("mant").Range("F22").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza mensaje de mantenimiento para Cilindro 1
    If CONT_Y0 > Workbooks("Mant.xls").Sheets("mant").Cells(21, 8).Value Then
        Workbooks("Mant.xls").Sheets("mant").Cells(21, 9).Value = "Reemplazar"
        Worksheets("mant").Range("I21").Interior.ColorIndex = 3 'rojo
    End If

    If CONT_Y0 <= Workbooks("Mant.xls").Sheets("mant").Cells(21, 8).Value Then
        Workbooks("Mant.xls").Sheets("mant").Cells(21, 9).Value = "OK"
        Worksheets("mant").Range("I21").Interior.ColorIndex = 4 'verde
    End If
    'Actualiza mensaje de mantenimento para Cilindro 2
    If CONT_Y1 > Workbooks("Mant.xls").Sheets("mant").Cells(22, 8).Value Then
        Workbooks("Mant.xls").Sheets("mant").Cells(22, 9).Value = "Reemplazar"
        Worksheets("mant").Range("I22").Interior.ColorIndex = 3 'rojo
    End If

    If CONT_Y1 <= Workbooks("Mant.xls").Sheets("mant").Cells(22, 8).Value Then
        Workbooks("Mant.xls").Sheets("mant").Cells(22, 9).Value = "OK"
        Worksheets("mant").Range("I22").Interior.ColorIndex = 4 'verde
    End If

End Sub
```

**Figura 40. Cuarta parte del código de Visual Basic detrás de la hoja de Excel para el departamento de mantenimiento.**

```
Sub PideDatosAlPlc()

    'Pide datos del FW20 al FW35
    ExchangeX1.Ip = "10.10.10.5"    'Direccion IP del PLC
    ExchangeX1.OperandOffset = 20   'Numero de palabra de inicio de los registros a intercambiar
    ExchangeX1.OperandCount = 16    'Cantidad de palabras a intercambiar a partir de la palabra de inicio
    ExchangeX1.Mode = 0             'Modo 0=lectura 1=escritura
    ExchangeX1.Type = 2             'Tipo de registro 2=Flag

    ExchangeX1.Exchange

End Sub
```

```
Private Sub ExchangeX1_ResultReceived()

    If ExchangeX1.Mode = 0 Then
        'Recibe la información del PLC
        S0 = ExchangeX1.GetOperand(0) 'Valor de FW20
        S1 = ExchangeX1.GetOperand(1) 'Valor de FW21
        S2 = ExchangeX1.GetOperand(2) 'Valor de FW22
        S3 = ExchangeX1.GetOperand(3) 'Valor de FW23
        S4 = ExchangeX1.GetOperand(4) 'Valor de FW24
        S5 = ExchangeX1.GetOperand(5) 'Valor de FW25
        INICIO = ExchangeX1.GetOperand(6)   'Valor de FW26
        PARO = ExchangeX1.GetOperand(7)     'Valor de FW27
        NORM_LIMP = ExchangeX1.GetOperand(8)    'Valor de FW28
        CILINDRO1 = ExchangeX1.GetOperand(9)    'Valor de FW29
        CILINDRO2 = ExchangeX1.GetOperand(10)   'Valor de FW30
        CONT_Y0 = ExchangeX1.GetOperand(14)    'Valor de FW34
        CONT_Y1 = ExchangeX1.GetOperand(15)   'Valor de FW35

    End If

End Sub
```

En las figuras 37, 38, 39 y 40 se ha incluido el código de Visual Basic para esta nueva hoja de Excel. Los comentarios incluidos explican su funcionamiento.

# CONCLUSIONES

1. Es posible mediante la aplicación de tecnología de punta mejorar el funcionamiento de una máquina como la llenadora de pistón.

2. Un controlador lógico programable o PLC permite, en la actualidad, la implementación de funciones avanzadas como las de comunicación, además de las clásicas funciones de control.

3. Una herramienta de software, como lo es Excel, permite la comunicación con el PLC que controla un proceso y permite la visualización y adquisición de información del mismo.

4. El contar con información básica del proceso en tiempo real permite mejorar la toma de decisiones a diferentes niveles, por ejemplo, a nivel producción o mantenimiento.

# RECOMENDACIONES

1. Es conveniente, a nivel nacional, reemplazar los sistemas de control antiguos de la maquinaria industrial, por modernos sistemas de control que permitan aumentar la productividad de las mismas.

2. Cuando se cuenta con sistemas modernos de control, normalmente, se puede implementar con el mismo equipo funciones adicionales que permiten optimizar el proceso.

3. El aprender y conocer de las nuevas tendencias de la tecnología no es suficiente, es necesario saber aplicar estos conocimientos.

# BIBLIOGRAFÍA


1. Bliesener, R. y otros. ***Programable logic controllers.*** 8ª ed. Alemania: Festo Didactic, 2002. 214 pp.

2. Croser, Peter y Frank Ebel. ***Pneumatics.*** 7ª ed. Alemania: Festo Didactic, 1999. 274 pp.

3. Hesse, Steffan. ***Sensors in production engineering.*** Alemania: Festo AG & Co., 2001. 134 pp.

4. Plagemann, B. ***IPC recipe book.*** Alemania: Festo AG & Co., 2004. 109 pp.

5. Prede, G. y D. Scholz. ***Electropneumatics.*** Alemania: Festo Didactic, 2002. 294 pp.

6. Tocci, Ronald. **Sistemas Digitales.** 6ª ed. México: Prentice Hall, 1996. 833 pp.

7. Weiss, Marvin. ***Microprocessors in industrial measurement and control.*** Estados Unidos: *Tab Professional and Reference Books*, 1987. 436 pp.