UNIVERSIDAD SAN FRANCISCO DE QUITO

Sistemas de Control, Supervisión y Adquisición de datos de las plantas del laboratorio de control de la USFQ

Luis Alberto Caiza Chicaiza

Tesis de grado presentada como requisito
para la obtención del título de Ingeniero Electrónico

Quito

Mayo del 2009

Resumen

Este proyecto surge de una necesidad institucional, de la Universidad San Francisco de Quito, de implementar un laboratorio de control automático para uso de los estudiantes de Ingeniería Electrónica, especialización control. Para cumplir con esta expectativa, se procede a diseñar e implementar un Sistema de Adquisición, Supervisión y Control (SCADA) de las plantas involucradas en el laboratorio. El sistema SCADA se realizará sobre las siguientes plantas que se encuentran en desarrollo: tanque continuamente agitado (CSTR) presurizado con intercambio de calor, sistema de transporte basado en levitación electromagnética (MAGLEV), domótica aplicada a un recinto y prototipo de ascensor de carga. En el proyecto, se utiliza un controlador lógico programable Siemens S7-200 y cinco módulos de expansión para la automatización; y el Software de Visual Basic6 para la interfaz gráfica.

Abstract

This project arises from an institutional need, of the Universidad San Francisco de Quito to implement an automatic control laboratory for its students use of Electronic Engineering in control specialization. To meet this expectation is to design and implement an acquisition Monitoring and Control System (SCADA) of the plants involved in the laboratory. The SCADA system will be held on the following plants that are in development: continuous stirred tank (CSTR) with pressurized heat exchange, the transport system based on electromagnetic levitation (Maglev), home automation applied to an enclosure and freight elevator prototype. The project, using a programmable logic controller Siemens S7-200 and five expansion modules for automation, and Visual Software Basic to the graphical interface.

Tabla de contenido

Lista de figuras

CAPÍTULO 1: DEFINICIONES DE SISTEMAS SCADA, PROTOCOLOS DE COMUNICACIÓN Y REDES DE PETRI

1.1. Justificación del proyecto a desarrollar

Este proyecto surge de una necesidad institucional, de la Universidad San Francisco de Quito, de implementar un laboratorio de control automático para uso de los estudiantes de Ingeniería Electrónica, especialización control. Para cumplir con esta expectativa, se procede a diseñar e implementar un Sistema de Adquisición, Supervisión y Control (SCADA) de las plantas involucradas en el laboratorio. El SCADA se realizará sobre las siguientes plantas:

- Tanque continuamente agitado (CSTR) presurizado con intercambio de calor, en desarrollo.
- Sistema de transporte basado en levitación magnética MAGLEV, en desarrollo.
- Domótica aplicada a un recinto, en desarrollo.
- Prototipo de ascensor de carga, en desarrollo.

1.2. Definición de SCADA

Se pueden encontrar múltiples definiciones de lo que significa un Sistema SCADA, sin embargo todos tienen el mismo fundamento, así:

“SCADA es la recolección de la información, transmitiendo a un sitio central para realizar el análisis necesario y el control, y entonces mostrar la información sobre pantallas”[1].

Según la definición anterior, la información o datos provienen de los procesos de plantas remotas. Para este proyecto en particular provienen de las mencionadas en la sección1.1. La información puede ser analógica o digital; analógica como la presión, temperatura, volumen, flujo de líquido; y digitales como salidas o entradas de alarmas, sensores de presencia, pulsadores, etc. La adquisición de datos, por lo general, está a cargo de un PLC (Controlador Lógico Programable) o de una RTU (Unidad Terminal Remota), el mismo que los envía hacia las estaciones centrales para la interfaz gráfica, mediante protocolos de comunicación determinados.

Estos datos en un sitio central, un computador, son almacenados y procesados mediante software adecuados. Para una rápida visualización y análisis por parte del operario, los datos procesados se muestran en pantallas, permitiendo así observar la evolución temporal de los estados y alarmas de cada una de las plantas. De esta manera, es posible la toma de decisiones en tiempo real sobre las mismas. La pantalla del ordenador es configurada por el usuario y fácilmente modificable para una adecuada comunicación con los elementos de campo, como controladores autónomos, autómatas programables, máquinas de control numérico, etc. La información proveniente de las plantas anteriores se envían a diferentes departamentos en una empresa o sistema productivo: departamento de supervisión, ingeniería, control de calidad, mantenimiento, gerencia, etc.

Un software utilizado para realizar un Sistema SCADA debe ser capaz de:

- Crear paneles de alarma sonora o lumínica que permitan al operador vía Hardware o Software reconocer casos particulares de los procesos de las plantas.
- Realizar un registro de la evolución temporal de las variables, de modo que sea posible procesar datos en una hoja de cálculo y encontrar tendencias.
- Desarrollar programas que modifiquen o anulen los programas previamente establecidos en un autómata (PLC) en determinadas condiciones, por ejemplo alarmas.

Entre los software utilizados para tales propósitos están: C, Pascal, LabView, VBA (Visual Basic for Applications), etc. Se utilizará este último en el presente proyecto.

1.2.1. Funciones de un sistema SCADA

Las funciones de un sistema SCADA puede variar dependiendo de la aplicación, pero todos deben cumplir algunos requisitos básicos como los que se mencionan a continuación:

- "Supervisión remota de instalaciones y equipos"[2]. Con esto es posible saber el estado de la planta sin la necesidad de la presencia física del operario en la misma.

- “Control remoto de instalaciones y equipos” [2]. Es posible tener un control, ya sea manual o semiautomático sobre las variables remotas involucradas en el proceso desde una estación central.
- “Visualización gráfica dinámica” [2]. Mediante programas sencillos en el software debe permitir la visualización de la evolución temporal de las variables de la planta.
- “Representación de las señales de alarma” [2]. Cuando la planta está operando en ciertas condiciones críticas es imprescindible que el Sistema SCADA muestre tales estados activando alarmas sonoras y/o visuales.

Las maniobras o secuencias de mando son de crucial importancia para tener una comunicación Full Duplex (comunicación en ambos sentidos) con los elementos de campo. Se deben diseñar con minuciosidad tomando en cuenta las siguientes consideraciones.

- Para realizar una acción sobre alguna variable, el operador debe ser capaz de conocer el estado anterior de la misma y la razón del suceso.
- Cuando el mensaje del suceso es demasiado largo, se debe optar por símbolos ó abreviaturas relacionadas con el caso.
- La toma de acciones que involucran mucha responsabilidad y conocimientos como cambios de producción, cambio de parámetros de controladores PID, etc, sólo se realizarán con estricta seguridad bajo un password de identificación.
- Desde cualquier pantalla el operador debe tener la capacidad de actuar sobre señales de estados críticos de la planta, como alarmas, emergencias. Las acciones deben ser simples de accionar como

botoneras, las mismas que estarán en tamaños grandes y asignados símbolos gráficos.

1.3. Protocolos estándares

Los equipos conectados en un sistema SCADA pueden comunicarse gracias a protocolos establecidos. Un protocolo de comunicación se puede definir como conjunto de reglas y formatos de mensajes preestablecidos para establecer una comunicación entre un emisor y un receptor. Dentro de las reglas se especifican la forma de las temporizaciones, secuencia de bits, chequeo o corrección de errores de transmisión, etc.

Se incluyen tres elementos fundamentales dentro de un protocolo:

- **Sintaxis.-** corresponde al formato de los mensajes, el envío y recepción de datos y comandos.
- **Semántica.**- es el significado de los comandos para la respectiva interpretación.
- **Secuenciamiento y temporización.**- se refiere a la velocidad de transmisión y recepción.

Los protocolos utilizados están estructurados por capas. Cada una tiene una función específica a fin de dividir tareas complejas en sencillas. De esta manera estandarizar los componentes de red, facilitando la integración de protocolos desarrollados por diferentes industrias.

La función de cada capa es independiente de las demás lo que permite realizar cambios y mejoras en cualquiera de ellas sin afectar a las demás.

Dentro del modelo de capas tenemos dos estructuras estándar: modelo *TCP/IP*, y el modelo *OSI*.

1.3.1 Modelo TCP/IP

Este modelo representa un conjunto de protocolos, entre los cuales se menciona: HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), etc. El nombre TCP/IP proviene de las siglas, Transmission Control Protocol/ Internet Protocol (Protocolo de Control de Transmisión/Protocolo de Internet). Éste protocolo fue desarrollado para la comunicación Internet asignando a cada equipo de la red una dirección IP para enrutar paquetes de información.

El protocolo fue diseñado para cumplir con las siguientes tareas:

- Dividir mensajes en paquetes.
- Usar un sistema de direcciones.
- Enrutar datos por la red.
- Detectar errores en la transmisión de datos.

Las cuales están asignadas a las capas que se muestran en la figura1.1:

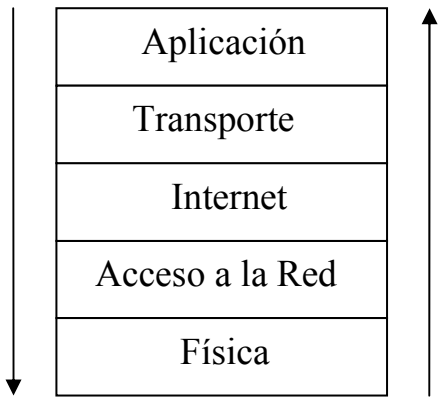


Figura 1.1. Capas del modelo TCP/IP

La capa de **aplicación**, incorpora aplicaciones de red estándar seleccionadas por el usuario.

La capa de **transporte**, se encarga de forma de transportar los datos.

La capa de **Internet**, es la encargada de administrar la información y relacionar con las direcciones IP. Dentro de esta capa se encuentran cinco protocolos: IP, ARP (Address Resolution Protocol), ICMP (Internet Control Message Protocol).

La capa de **Acceso a la Red,** define direcciones físicas. Entre las funciones que en ella se realizan son: coordinar una comunicación síncrona (mediante un reloj) o asíncrona (sin un reloj), establecer el formato de datos, detección y manejo de errores.

La capa **física,** define la conversión de señales análogas en digitales, ya sean mediante pulsos eléctricos, modulación de luz, etc.

1.3.2 Modelo OSI

La palabra OSI proviene de las siglas “Open Systems Interconnection” (modelo de referencia de Interconexión de Sistemas Abiertos). “En el modelo OSI cada capa agrupa algunas de las funciones requeridas para comunicar sistemas. Estas capas poseen estructura jerárquica. Cada capa se apoya en la anterior, realiza su función y ofrece un servicio a la capa superior. Este modelo posee la ventaja de poder cambiar una capa sin necesidad de modificar el resto”[3].

Este modelo comprende siete capas, a diferencia del anterior que solamente contiene cinco, pero muchas de ellas son similares ya que fueron desarrolladas a la par. Las siete capas se muestran en la figura1.2.

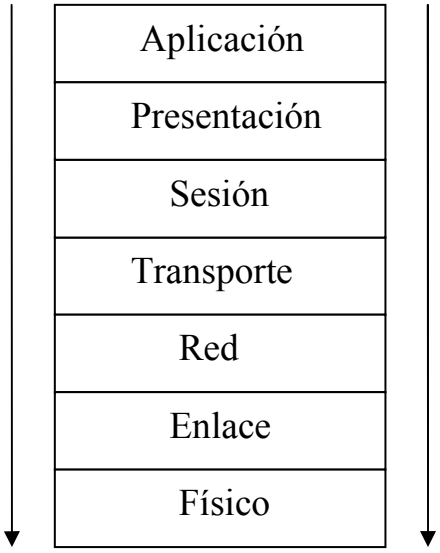


Figura 1.2. Capas del Modelo OSI

Capa Física. Define la manera en que se transmiten los datos. Dentro de estas, las características mecánicas, como conectores, pines; eléctricas como la amplitud y duración del voltaje de los bits.

Capa de Enlace. Permite la comunicación entre equipos dentro de la misma red. La información se encuentra en bloques llamados Tramas.

Esta capa dota de direcciones física, y también detecta y trata los errores de comunicación.

Capa de Red. A diferencia de la anterior, se encarga de la conexión de equipos distribuidos en diferentes redes. La información se encuentra contenida en los llamados paquetes que contienen las direcciones lógicas asociadas a cada puesto de origen y destino.

Capa de Transporte. Se encarga del transporte de datos de una manera eficiente. Los datos por lo general son divididos en paquetes y enviados

secuencialmente. Tanto los datos dentro de un paquete y los paquetes deben llegar en el mismo orden en que fueron enviados.

Capa de Sesión. Establece el inicio y la finalización de una comunicación entre equipos. Permite reestablecer un dialogo de una manera predeterminada.

Capa de Presentación. Establece el formato de los datos que maneja la capa de aplicación. El formato utilizado es un estándar, independiente del sistema operativo.

Capa de Aplicación. Es la interfaz con el usuario, ya que permite al mismo administrar el software.

1.3.3 Protocolo Modbus.

Es un protocolo desarrollado por Modicon, situado en el nivel 7 del Modelo *OSI*, para la comunicación entre PLC's. Por su sencillez y determinación pública es ampliamente utilizado por diferentes fabricantes de PLC's, RTU's, Drives, sensores y actuadores.

La comunicación entre dispositivos se basa en una estructura *Maestro/Esclavo*, para lo cual a cada esclavo se asigna una dirección entre 1 a 247.

1.3.4 Protocolo Profibus.

Es un estándar de comunicación abierto y robusto para bus de campo basado en el modelo OSI. Mediante una estructura Maestro/esclavo permite la comunicación de PLC, PC, sensores u actuadores.

Existen tres versiones: Profibus DP (Periférica Descentralizada) para la comunicación de sensores y actuadores a controladores automáticos y PC's; Profibus PA (Automatización de Procesos) para la automatización de procesos que requieren alta seguridad; y Profibus FMS (Especificación de Mensajes de Bus de Campo) para la comunicación entre equipos de automatización y dispositivos inteligentes. La versión *PD* es la más utilizada, una de las tecnologías es el protocolo RS-485.

1.3.5 Comunicación Serial.

La comunicación serial es utilizada para la conexión directa entre dos equipos en un instante dado, por ejemplo un PLC y un computador PC. Los bits son enviados y recibidos sucesivamente cada cierto intervalo de tiempo por el puerto serial de los equipos.

Los bits son enviados uno por uno, ya sea de manera **síncrona** o **asíncrona**. Cuando el envío y recepción está sincronizado por un reloj, tenemos una comunicación serial síncrona. En tanto que la comunicación serial asíncrona transmite bits en cualquier momento, se dice que opera en el modo UART ("receptor/transmisor asíncrono universal'').

La comunicación serial puede ser Simplex, Half Duplex y Full Duplex.

- **Simplex**. Si los datos fluyen en un solo sentido, por ejemplo la comunicación PC e impresora. Se dice que opera en modo maestro esclavo, donde el maestro envía y el esclavo sólo recibe.

- **Half Duplex**. La comunicación fluye en ambos sentidos pero uno a la vez. En un momento dado, un primer equipo envía mientras el segundo recibe, luego el segundo equipo envía y el primero recibe.
- **Full Duplex**. En todo momento es posible tener una comunicación simultánea en ambos sentidos. Para esto es necesario un mayor número de cables.

Los estándares de comunicación serial son RS-232, RS-422, RS-485, USB. Cada una presenta características especiales.

La comunicación RS-232 es una comunicación punto a punto (equipo a equipo) a una velocidad máxima de 20kbps (bits por segundo). Puede operar en modo half y full duplex y la máxima distancia del cable serial utilizado puede ser de 15 metros.

Con el RS-422, es posible comunicar equipos distantes hasta de 1200m, pero del modo half duplex. La ventaja de éste estándar es que permite conectar varios equipos (hasta10) a una misma red (multidrop). La velocidad máxima de transmisión es de 10Mbps.

El RS-485, es similar al anterior con la particularidad que es posible conectar en paralelo hasta 32 equipos y la velocidad máxima de transmisión es de 35Mbps

1.4. Sistemas SCADA de bajo costo versus sistemas SCADA industriales.

El grado de cumplimento de las funciones de un sistema SCADA mencionadas en la sección1.2.1 es función de los elementos de campo utilizados y más aún de los elementos utilizados para el software.

Se puede realizar un SCADA mediante computadores convencionales como, PC, computadoras portátiles y estaciones de trabajo, así como también utilizar arquitecturas más sofisticadas.

Un sistema SCADA de bajo costo involucra un miniordenador PC con sistema operativo DOS/Windows y/o Linux, y con software adecuados para el manejo de las interrupciones y la comunicación con los elementos de campo, por ejemplo Microcomputing de Siemens. El CPU del computador es el factor que limita la velocidad de respuesta en tanto que el sistema operativo Windows hace que sea monousuario aunque permita realizar multitareas pero en ámbito reducido.

Un sistema como el anterior presenta limitaciones, si se desea tener una mejor funcionalidad se emplean sistemas operativos complejos (VAX, VMS, Unix) y "arquitecturas cliente-servidor aptas para compartir recursos informáticos (datos y aplicaciones) que permiten ofrecer programas capaces de entender simultáneamente a varios servicios, por ejemplo, un operador puede estar viendo informes de incidencias desde un ordenador industrial con un sistema operativo Unix en tanto que otro está modificando la evolución del proceso desde una estación de trabajo y un tercero monitorizando la situación en la planta desde un PC"[4]. En el presente caso estamos en sistemas SCADA industrial, su principal

característica es la utilización de un sistema operativo robusto que soporta multitareas y varios usuarios que estando separados trabajan sobre un mismo sistema al enlazar informaciones mediante redes locales.

Dependiendo del tipo del sistema SCADA se puede configurar de dos maneras, como se describe a continuación:

- En sistemas SCADA de bajo costos se acostumbra a utilizar paquetes de software genéricos que son fáciles de adecuar a la aplicación. Los Drivers de comunicación y el software para las pantallas son configuradas y parametrizadas por el usuario. Así para realizar pantallas, el usuario utiliza las librerías del software para crear botones, editores, funciones, etc. El inconveniente radica principalmente en la limitación de los paquetes y también de la licencia dada por el proveedor.
- En sistemas SCADA industriales se acostumbra a utilizar paquetes de software específicos, como el software *ABB* utilizado para la administración de la distribución de tensiones eléctricas y el software *Indusoft* utilizado para el SCADA de procesos de tratamiento de aguas y generación de energía. En dichos software las pantallas, controles, comunicación y demás son preestablecidas y las adecuadas para la empresa. De esta manera no es necesario un diseñador del sistema, con la certeza que el sistema responde a las expectativas de la industria. Pero, no permite realizar mejoras futuras como cambio de protocolos de comunicación, mejoras en la visualización, etc.

1.5. Sistemas a eventos discretos (DES)

Antes de dar una definición formal, es necesario dar la definición de algunos términos.

- Sistema: parte del universo compuesto por cuerpos que interaccionan entre sí, pero que no tienen interacción externa.
- Evento Discreto: representación del cambio en el tiempo de los componentes del sistema.

Con lo anterior en mente, los sistemas a eventos discretos (DES) son sistemas que dependen de variables que cambian sólo en un número finito (contable) de puntos en el tiempo (Discreto) y cuyas variables cambian continuamente con respecto al tiempo (Dinámica) [5].

Los DES son adecuados para describir procesos interconectados secuenciales. En la automatización de líneas de producción, redes de comunicación, semáforos, procesos de prensado, galvanizado, etc.

Dentro de las metodologías de análisis y tratamientos de los sistemas a eventos discretos se encuentran otros sistemas como: Redes de Petri, Grafcet, Statecharts, Grafos de eventos, Máquinas de estados finitos y Álgebra.

1.6. Red de Petri

Una Red de Petri (*Petri Nets PN*) es un modelo gráfico, estructurado y formal utilizado para modelar y analizar el flujo de información en un proceso. La

información dinámica más relevante de los sistemas modelados es mostrada por una PN, para lo cual utiliza un conjunto de ecuaciones matemáticas.

Las Redes de Petri son utilizadas en diseños de sistemas de hardware y software para simulación de amplios problemas de ingeniería, en especial para procesos concurrentes, procesos que requieren una apropiada sincronización, procesos cíclicos, etc. La utilización de dicha herramienta gráfica, garantiza seguridad, eficiencia y optimización de recursos en el diseño e implementación. Además, facilita el mantenimiento y posibles mejoras a futuro del modelo.

Es menester hacer una breve reseña histórica del aparecimiento y evolución de las PN. Las Redes de Petri nacen en 1962, con Carl Adam Petri de la Universidad Darmstadt en Alemania cuando él presenta su trabajo doctoral “Kommunikation mit Automaten” (Comunicaciones con Autómatas). Luego, un grupo de investigadores del Applied Data Research Inc, desarrolla la teoría Systemics, con lo cual proporcionan la notación y representación de las Redes de Petri lo que hace posible una amplia divulgación en 1968. En 1970, Holt y Commoner en sus trabajos “Events and Conditions”, muestran como esta herramienta gráfica puede ser utilizada para modelar y analizar sistemas que involucren procesos concurrentes.

Las Redes de Petri fueron objeto de estudio del grupo “The Computation Structures Group”, del Massachussets Institute of Technology (MIT) que bajo la dirección de Jack Dennis presentaron los eventos “Project MAC Conference on Concurrent Systems and Parallel Computation” y “Conference on Petri Nets and Related Methods”. Posteriormente aquel grupo ha publicado innumerables tesis doctorales sobre las PN. Desde entonces el MIT y otros centros de investigación americanos han concentrado la aplicación de dicha herramienta hacia la teoría de

autómatas. De la misma manera, las universidades europeas tienen el mismo enfoque, con lo cual se tiene una misma corriente en pro del desarrollo de la ciencia y la tecnología.

1.6.1 Ventajas de las Redes de Petri

- “Formalismo matemático, que facilita la especificación formal de procesos y sistemas en tiempo discreto, evitando ambigüedades.
- Sintaxis y semántica bien definida.
- Herramienta gráfica para modelar DES. Lo que permite dar un seguimiento del proceso de automatización en todo el tiempo.
- Provee técnicas de análisis de eventos.
- Permite la construcción sistemática de sistemas a eventos discretos y controladores autómatas.
- Facilita el mantenimiento y modificaciones posteriores en los modelos. Ésta metodología permite un fácil entendimiento hasta para quienes no han diseñado el modelo, permitiendo así realizar correcciones y mejoras.
- Garantiza el desempeño del modelo en términos de: No Bloqueos, Ciclicidad, Estabilidad, Alcanzabilidad de estados deseados”[6].

1.6.2 Componentes fundamentales de una Red de Petri

Las Redes de Petri están formadas básicamente por un conjunto de plazas o lugares *P*, un conjunto de transiciones *T,* arcos dirigidos y las funciones de entrada *I,* y de salida *O.*

- **Lugares y transiciones.** Representan *eventos* conectados por arcos. Gráficamente los lugares *p* se representan por circunferencias y las transiciones *t* por pequeños segmentos rectilíneos. Un lugar representa un estado en el que puede estar parte del sistema.
- **Arcos.** Un arco une un lugar con una transición, o viceversa e indica la dirección de la secuencia de eventos. Nunca es posible unir un lugar con otro lugar, o una transición con otra transición. Además, n arcos de igual dirección son representados con un solo arco etiquetado con el número n, dicho número se conoce como peso del arco, cuando no existe ninguna etiqueta se asume que el arco es de peso uno.
- **Marca.** Se representa por un punto dentro de la circunferencia que representa al lugar. Las marcas nos permiten visualizar el estado del sistema.
- **Tokens.** Son números naturales o puntos colocados en circunferencias que representan a los lugares. Las transiciones son las responsables de moverlos a través de la red.

Una definición matemática de la estructura de las redes de Petri se muestra a continuación:

$$R = (P\ T\ I\ O\ M_0)$$

Donde:

- P = {p1, p2,…..pm} Conjunto finito y no vacío de lugares.
- T = {t1, t2,…..tm} Conjunto finito y no vacío de transiciones.
- I: (PxT) es la función de entrada representados por arcos que van desde los lugares a las transiciones.
- O: (PxT) es la función de salida representados por arcos que van desde las transiciones a los lugares.
- Mo: es la marca inicial de la red.

Una red de Petri con sus elementos básicos se muestra en la figura 1.3:

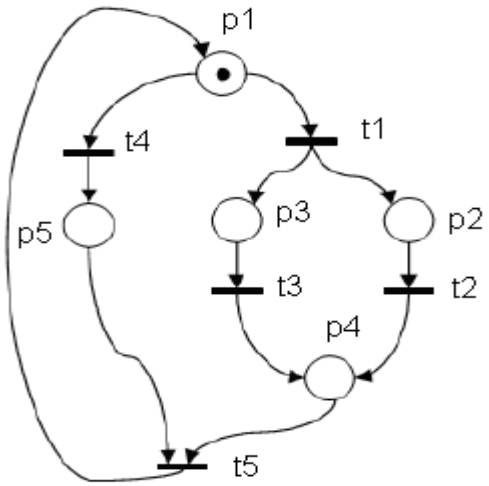


Figura 1.3. Red de Petri

1.6.3 Ejecución de una Red de Petri

En la ejecución de una PN, se dan dos eventos: la habilitación y el disparo de las transiciones. Una transición está habilitada cuando existen en los lugares de entrada un número de tokens igual al peso del arco que las une.

Cuando se da el disparo de una transición, se remueven los tokens de los lugares precedentes y se asignan a los lugares a los que apuntan los arcos de salida.

En el presente proyecto se utilizan las redes de Petri para modelar el proceso de alarmas a fin de tener un seguimiento sistemático de su evolución. Para lo cual los lugares y transiciones se correlacionan con las posiciones de memoria del correspondiente autómata utilizado.

Como una aplicación de la Red de Petri se tiene el tratamiento de la alarma de humo de la DOMÓTICA. Dada la entrada de alarma de humo se debe activar un estado de alarma y una salida sonora y una visual, las cuales deben permanecer activas aún cuando la entrada de alarma ya no lo esté. El usuario debe ser capaz de apagarlas mediante una interfaz gráfica como Visual Basic. Además, es necesario conocer si la entrada de alarma estaba o no activada por más de un cierto tiempo, lo cual nos permite estimar las posibles causas de la alarma o desechar las falsas alarmas.

Para el proceso anterior se utiliza la Red de Petri de la figura1.4, cuya secuencia es la siguiente:

- El lugar *Operación Normal* contiene la marca inicial del proceso, cuando se activa la entrada de alarma *AH* se remueve el Token de *Operación Normal* y se coloca en los lugares *A_AH y T37*. El símbolo *A_AH* representa la dirección de memoria de un bit y *T37* un temporizador del PLC. El bit llamado *A_AH* permanece en alto si dispone de tokens. El temporizador toma el tiempo en que la entrada de alarma está en activa y pone en alto el bit de temporización T37 de la PLC si supera un valor de tiempo determinado por el usuario.

- La transición *OK_AH* representa una dirección de un bit en la memoria del PLC. Cuando la transición *OK_AH* está deshabilitada toma tokens del lugar *A_AH* y los coloca en *Luz* y *Sonido* que hace que se activen las salidas de alarma.
- El estado *Reconocimiento* está representado por dos bits del PLC, el bit llamado *OK_AH* y el *Forzar_AH*, ambos remueven tokens de los lugares *Luz* y *Sonido*. El reconocimiento *OK_AH* remueve los tokens de los lugares *Luz, Sonido*, *A_AH y T37* y los coloca en *Operación Normal*, si el bit *T37* no esta activado, apagando así las salidas de alarma y el estado de alarma *A_AH*. Si T37 esta en alto se remueven los tokens de *Luz* y *Sonido* y se coloca en *Reconocimiento* y se deshabilita la transición *OK_AH,* por lo que las salidas de alarma se apagan pero no el estado de alarma *A_AH.*
- El Reconocimiento *Forzar_AH* remueve los tokens de los lugares *Luz, Sonido*, *A_AH* y los coloca en *Operación Normal.* Con esto se apagan las salidas de alarma y también el estado de alarma *A_AH.* Se fuerza la alarma, cuando no es posible apagar el estado de alarma *A_AH* con el normal reconocimiento *OK_AH.*
- El token de T37 es removido gracias a que la transición *A_AH* está habilitada. Remover un token del temporizador significa establecer en cero el conteo del temporizador.

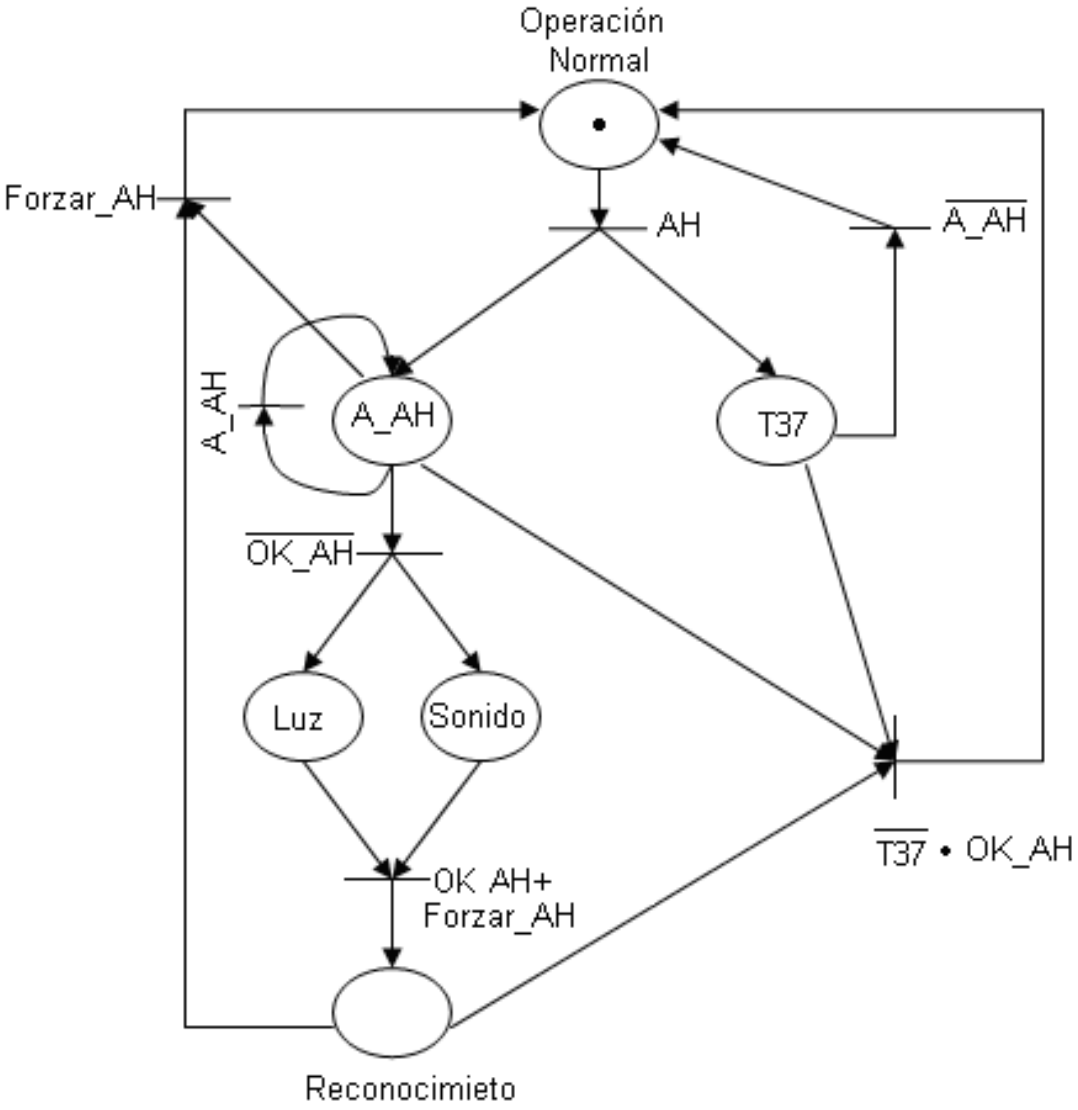


Figura 1.4. Red de Petri para la alarma de humo.

CAPÍTULO 2

AUTÓMATA S7-200

2.1 Características de la CPU 226 y de los módulos de ampliación.

El autómata programable utilizado para la automatización de las plantas del laboratorio de control es un Micro-PLC S7-200 del tipo CPU226, este tipo de PLC es lo que se dispone en el mencionado laboratorio de la USFQ. Su capacidad de expansión para incorporar tanto módulos de entradas y/o salidas digitales como analógicos, y su eficiencia en la solución de tareas de automatización sencillas hace que sea adecuado para la presente aplicación.

Los módulos de expansión utilizados son cinco. Un módulo de 16 entradas y 16 salidas digitales, dos módulos de 4 entradas analógicas y 1 salida analógica, y dos módulos de 2 salidas analógicas.

A continuación se describen las características técnicas del autómata programable y de los módulos de expansión.

2.1.1 Datos técnicos del Autómata Siemens S7-200 CPU 226

- 24 Entradas/16 Salidas digitales.
- 4K Palabras de capacidad de programa del usuario (EEPROM) y 2560 2.5K Palabras de bloque de datos (EEPROM).
- 2 puertos de comunicación RS-485 y PPI.

- 256 temporizadores y contadores.
- 256 marcas internas.

Figura 2.1. CPU 226

2.1.2 Áreas de memoria de la CPU 226.

Memoria de programa. Se encuentra almacenada en una EEPROM no volátil, por lo que en una interrupción de energía eléctrica no se borra el contenido. En esta memoria se almacena el contenido y la secuencia de instrucciones de la aplicación.

Memoria de datos. Es el espacio donde se guarda las direcciones para los cálculos, valores de resultados intermedios, valores de temporizadores y contadores. También están los estados de las entradas y salidas digitales y analógicas.

Estos datos pueden ser guardados en una memoria EEPROM no volátil y en una RAM. El acceso a los valores se realiza por bits, bytes, palabras o palabras dobles.

El área de datos contiene las siguientes áreas: Memoria de Variables, Imagen de procesos de entrada y salida, marcas especiales, temporizadores, contadores,

acumuladores, entradas y salidas analógicas, memoria local. A continuación se describen las utilizadas en el proyecto.

- ***Memoria de variables V.*** Ésta se puede utilizar para almacenar valores intermedios de cálculos durante el programa. Ej. V1.0, donde 1 es la dirección del byte y 0 la dirección del bit.
- ***Imagen del proceso de entradas I y salidas Q digitales***. El autómata ejecuta el programa por ciclos, al empezar un ciclo lee el estado de cada entrada y almacena en la imagen del proceso de las entradas, al final del ciclo escribe en las salidas físicas el valor almacenado en la imagen del proceso de las salidas.
- ***Área de Marcas M.*** Se puede utilizar como relés de control que guardan estados intermedios de un proceso.
- ***Marcas Especiales SM.*** Se utiliza para intercambiar información entre la CPU y el programa. Ciertas funciones de la CPU pueden ser controladas por medio de estas marcas.

Tanto las marcas **M** y las marcas especiales **SM** son las transiciones y lugares en la ejecución de una Red de Petri.

- ***Entradas analógicas AI***. Los valores reales analógicos son convertidos en formato digital de palabra (16 bits) y guardados en el área AI. Se acceden a cada uno con bytes pares como AIW0, AIW2,....
- ***Salidas analógicas AQ.*** Los valores digitales son convertidos en formato real analógico. Se acceden a cada uno con bytes pares como AQW0, AQW2,....

- **Área de memoria local (L)**. Tiene un alcance local, lo que significa que solo es posible acceder a su contenido dentro del programa principal, subrutina o interrupción en la que es declarado.

2.1.3 Módulo de ampliación EM 223.

El módulo de ampliación digital EM 223 acoplado a la CPU 226, tiene 16 entradas digitales/16 salidas digitales de relé X DC24V.

El área de memoria en la CPU del PLC correspondiente a la imagen del proceso de entradas va de I3.0 a I4.7, mientras que el área de la imagen de salida va de Q2.0 a Q3.7.

2.1.4 Módulo de ampliación EM 235.

También se conectan a la CPU dos módulos analógicos EM 235 con 4 entradas analógicas IQ y una salida analógica AQ de 12 bits.

Las salidas se configuran a un margen de corriente de 4-20mA mediante los interruptores DIP incluidos en el módulo

Las imágenes de entrada toman las direcciones AIW0, AIW2, AIW4, AIW6, AIW8, AIW10, AIW12 y AIW14 en la CPU226. Mientras que las salidas, toman las direcciones AQW0 y AQW2.

2.1.5 Módulo de ampliación EM 231.

Se utiliza dos módulos de ampliación EM 231 con dos salidas analógicas AQ de 12 bits.

Las imágenes de salida, en la CPU226, están en las direcciones AQW4, AQW6, AQW8 y AQW10

2.2 Software de programación del autómata

La programación se realiza con STEP-7 Micro/WIN 32 versión 3.2 que se ejecuta bajo Windows y permite programar el PLC, configurar el sistema y supervisar en tiempo de ejecución.

Para conectar al ordenador con el PLC se utiliza un cable conversor de norma RS-485 a RS-232. Una vez establecida la comunicación, se utilizó el editor KOP o esquema de contactos para realizar la programación y descargar en el autómata.

2.2.1 Configuración de la comunicación

Se tiene una comunicación punto a punto entre la PC y el PLC, para lo cual se utiliza un cable PC/PPI. El extremo RS-232 (PC) del cable es conectado al puerto de comunicaciones de la PC, y el otro extremo RS-485 al puerto de comunicaciones del PLC.

Los interruptores DIP del cable son ajustados para una transmisión a 9.6kbps. También se debe configurar la comunicación desde el software de Micro/WIN32 en el icono de comunicaciones del programa del PLC.

2.2.2 Operaciones en el lenguaje de programación KOP utilizadas.

El lenguaje de programación KOP o esquema de contactos se caracteriza por ser un lenguaje gráfico. A continuación se describen los elementos básicos que componen el programa.

- ***Contactos*:** representan interruptores eléctricos por los que circula corriente cuando están cerrados. Éstos pueden ser normalmente abiertos o normalmente cerrados.

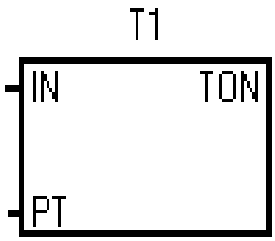
- ***Bobinas**:* son relés que se activan cuando se aplica una corriente.
- ***Marcas especiales (SM)*:** son funciones de estado y control ofrecidas por el autómata. En el proyecto se utiliza las marcas SM0.0, SM0.1 y SM8 a SMB21.

SM0.0: es un bit que permanece activado siempre. Se utiliza para habilitar permanentemente la bobina M0.0. La marca M0.0 es utilizada para habilitar todas las acciones que el programa ejecuta en cada ciclo.

SM0.1: es un bit que se activa solo en el primer ciclo del programa. Es utilizado, en combinación con secuencias, para comprobar el estado de los módulos de expansión, en el presente proyecto.

A demás se utilizarán las marcas SMB8 a SMB21 que están organizados en pares de bytes y contienen la información de comunicación y error para los módulos de ampliación de 0 a 6.

- ***Temporizador con retardo a la conexión.***

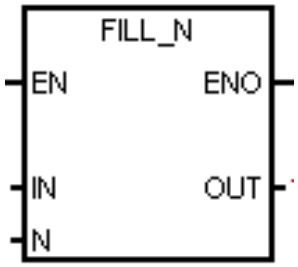


La función TON cuenta el tiempo en que la entrada IN está activada. Si el tiempo es mayor o igual al valor preestablecido PT, el bit de temporización se activa y se borra cuando se desactiva la entrada IN.

T1 representa el nombre del temporizador, otro temporizador podría tomar cualquier nombre comprendido de T0 a T255.

Cualquier temporizador entre T37-T63 es utilizado en el proyecto para el proceso de las entradas de alarma, su resolución es 100ms.

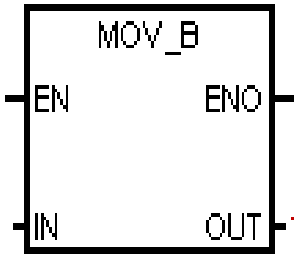
- ***Inicializar memoria.***



Es utilizada para escribir N palabras consecutivas empezando en la dirección OUT, con el valor de la palabra contenida en IN.

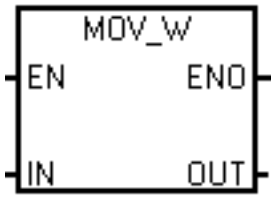
En el proyecto se utiliza este bloque para inicializar con ceros las direcciones en el área de memoria V, donde se guardan las entradas analógicas.

- ***Transferir bytes.***



MOV_B transfiere el byte de entrada (IN) a la salida (OUT). En el proyecto se emplea para transferir el byte que contiene la información de conexión de los módulos.

- ***Transferir palabras (MOV_W)***

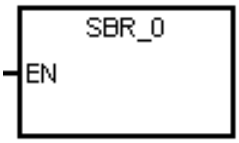


Transfiere la palabra de entrada (IN) a la dirección de salida (OUT) cuando está habilitada la entrada (EN).

Se utiliza para transferir el contenido de las entradas analógicas a las direcciones VW16, VW18,…, VW30 en cada ciclo de operación, cuando los módulos están comunicando correctamente.

- ***Comparadores:*** Son utilizados para comparar dos valores, un IN1 con un IN2. Cuando la condición es verdadera el contacto está cerrado. Los valores de las entradas pueden estar formato bit, byte, reales, palabras o palabras dobles.

- ***Llamada a subrutinas.***



Son funciones que transfieren el control a las subrutinas (SBR_n). Éstas pueden llamar con o sin parámetros.

En el proyecto se utiliza una subrutina sin parámetros de entrada para los PIDs, y dos subrutinas con parámetros de entrada para la comprobación del estado de los módulos de expansión y para el escalado de las entradas análogas.

Se llama a la subrutina cuando la entrada EN está habilitada.

2.3 Programa implementado en la CPU siemens s7-200

El programa implementado en el PLC es tal que permita la interacción con la interfaz gráfica implementada en Visual Basic. Se ha estructurado de la siguiente manera:

- Asignación de las direcciones de memoria M a las entradas digitales del PLC.
- Verificación de los módulos de expansión y asignación de las entradas analógicas a marcas en la CPU del PLC.
- Escalado de las entradas analógicas.
- Procesamiento de alarmas.
- Asignación de los parámetros de los PIDs.
- Activar y desactivar marcas M correspondientes al ASCENSOR

2.3.1 Asignación de las direcciones M a las entradas digitales del PLC.

La marca especial SM0.0 es un bit que permanece activado siempre y se utiliza para habilitar permanentemente la bobina M0.0. El bit M0.0 es utilizado para habilitar contactos, bloques, funciones en cada ciclo.

En la primera parte del programa se hace un tratamiento de las entradas digitales del PLC. A cada variable de entrada se asigna una ubicación en el área de memoria M de acuerdo a la tabla1 del **Anexo1**. Por ejemplo, en la figura 2.2,

el valor de la variable LSHH de CSTR conectada a la entrada I0.1 se guarda en la dirección M0.1 a la cual se hará referencia cuando se necesite de tal valor. Lo mismo se realiza con todas las entradas digitales de I0.2 a I4.7.

Figura 2.2. Este segmento asigna a la marca M0.1 el valor de I0.1

2.3.2 Verificación de los módulos analógicos de expansión y asignación de las entradas analógicas a marcas en la CPU.

Se empieza con cargar la marca especial SM0.1 (activo solo en el primer ciclo), como muestra la figura 2.3, para llenar 8 palabras con ceros a partir de la dirección VW16. Estos espacios de memoria se utilizarán para guardar las imágenes de las entradas analógicas. A continuación se llama a una subrutina llamada “COMPRUEBA _ ERROR”.

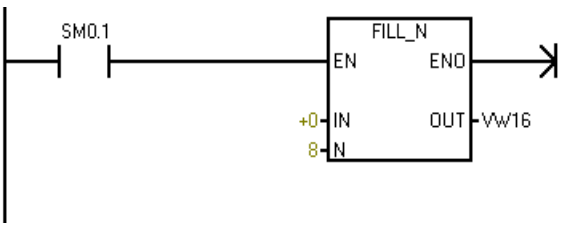


Figura 2.3. Se llena de ceros 8 palabras a partir de la dirección VW16.

Subrutina COMPRUEBA _ ERROR. La subrutina tiene un parámetro de entrada llamado “slot” y cinco salidas: energía, rango, identificación, configuración y OK, que indican el estado del módulo.

Los parámetros de entrada y de salida son declarados dentro de la subrutina. El parámetro “slot” se declara como byte y se ubica en la dirección de memoria local LB0, las salidas energía, rango, identificación, configuración y OK como boolenas en las direcciones L1.0, L1.1, L1.2 y L1.3, respectivamente. También se declara una variable interna “identerror” como palabra en la dirección LW2 que no es ni entrada ni salida de la subrutina. Ésta variable contendrá la información de error de cada uno de los módulos como se verá mas adelante.

La primera llamada a “COMPRUEBA _ ERROR”, como muestra la figura 2.4, es para verificar el módulo analógico EM235 ubicado en el slot0.

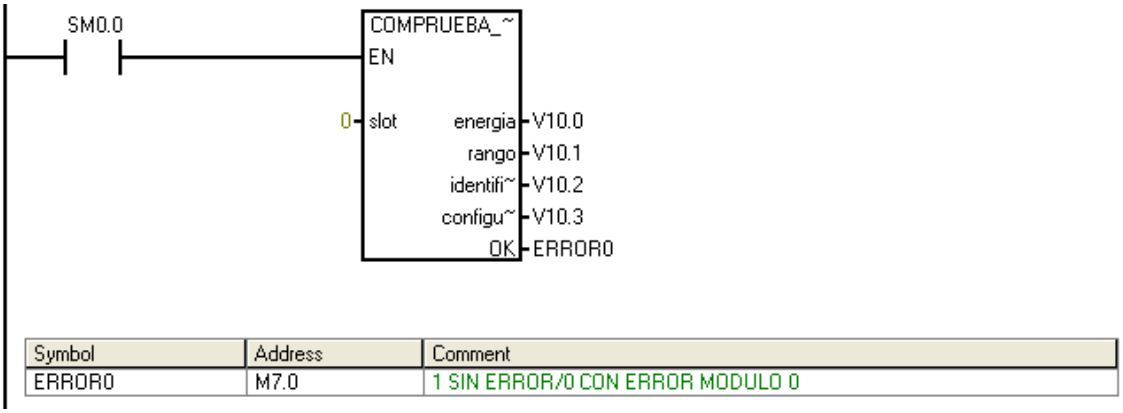


Figura 2.4. Llama a la subrutina COMPRUEBA _ ERROR para verificar el modulo0.

Dentro del programa de la subrutina, la primera línea (figura2.5) compara si el valor del parámetro de entrada es igual a “0”, que para la primera llamada a la subrutina es afirmativo por lo que el bloque *MOB _W* mueve el contenido de la marca SMW8 que contiene la identificación y error del primer módulo a la

dirección LW2. LB2 contiene la identificación del módulo y LB3 el error del módulo. Los siguientes cuatro segmentos la subrutina comparan “slot” con 1, 2, 3 y 4; y el bloque MOV_W tiene de entrada en IN las palabras SMW10, SMW12, SMW14, SMW16 que informan el estado de los módulos1, modulo2, módulo3 y módulo4, respectivamente. Todas se guardan en la dirección interna LW2.

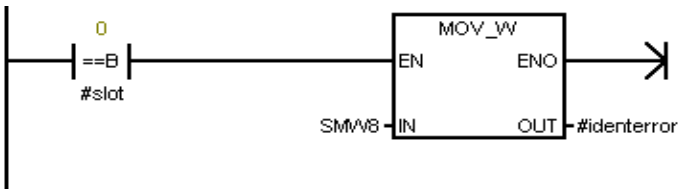


Figura 2.5. Mueve la información del modulo 0 contenida en SMW8 a la variable interna *identerror*.

Las líneas de la figura 2.6 comprueban el estado de energización del módulo, si el área es excedida o no, la identificación y configuración. Toda la información se encuentra en la dirección LW2 y se interpreta según la tabla2.1.

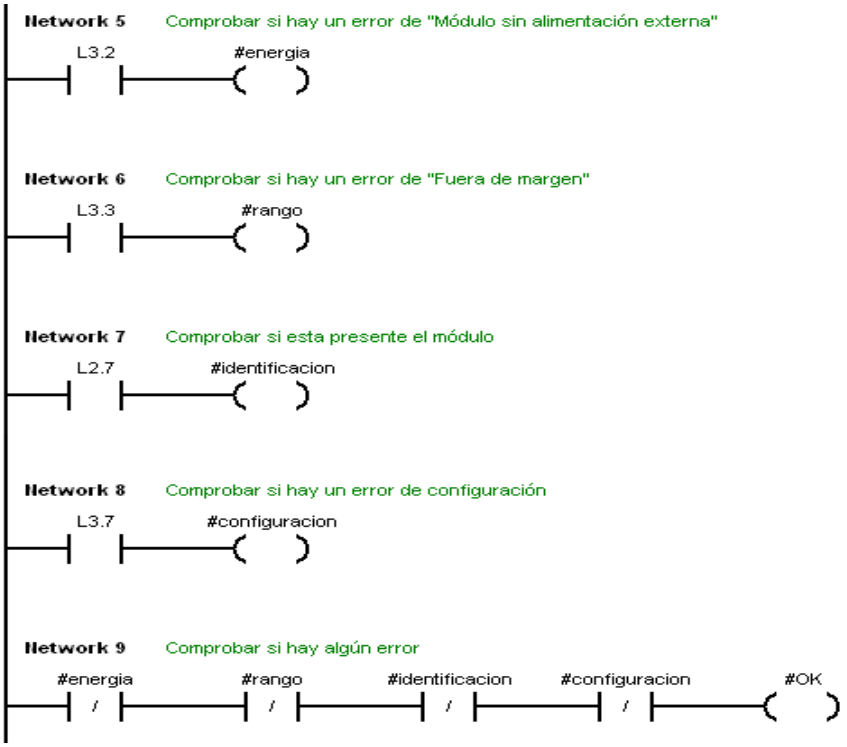


Figura 2.6. Tratamiento de la información del módulo analógico.

Tabla 2.1. Información del módulo.

Byte LB2

MSB 7							LSB 0
m	t	t	a	i	i	q	q

m: Módulo presente 0 = presente, 1 = no presente

tt: Tipo de módulo

00 Módulo de ampliación no inteligente.

01 Módulo inteligente.

10 Reservado

11 Reservado

a: Tipo de E/S 0 = digital, 1 = analógica

ii: Entradas

00 Sin entradas

01 4 AI o 8 DI

10 8 AI o 16 DI

11 16AI o 32 DI

qq: Salidas

00 Sin salidas

01 2 AQ o 8 DQ

10 8 AQ o 16 DQ

11 16AQ o 32 DQ

Byte LB3

MSB 7							LSB 0
c	0	0	b	r	p	f	t

c: Error de configuración 0 = sin error , 1 = con error.

b: Fallo de bus o error de paridad.

r: Área excedida.

p: Error de alimentación.

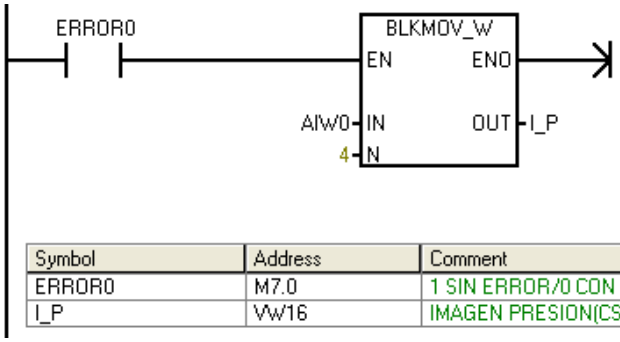
f: Fusible fundido.

t: Bloque de terminales suelto

Según lo anterior, los bits de interés son: L3.2 que informa el estado de energización, L3.3 que contiene la información del rango o área, bit L2.7 que informa la identificación del módulo, L3.7 que contiene el error de configuración. Basta con que uno de los anteriores esté activo para decir que el módulo tiene error.

El programa de la subrutina “COMPRUEBA _ ERROR”, como muestra la figura 2.6, termina con un segmento donde se activa la bobina llamada “OK” de no existir error de energía, rango, identificación y configuración, de existir un error la bobina “OK” se desactiva. El estado de “OK” es uno de los valores de retorno de la subrutina. Ya en el programa principal dicho estado se guarda en la variable “ERROR0”.

En el programa principal, cuando la variable “ERROR0” es igual a uno, en la figura 2.7 (es decir no existe error del módulo 0) se mueven las entradas análogas AIW0, AIW2, AIW4, AIW6 a los espacios de memoria VW16, VW18, VW20, VW22 respectivamente.



Symbol	Address	Comment
ERROR0	M7.0	1 SIN ERROR/0 CON ERROR MODULO 0
I_P	VW16	IMAGEN PRESION(CSTR)

Figura 2.7. Mueve el contenido de las cuatro primeras entradas analógicas.

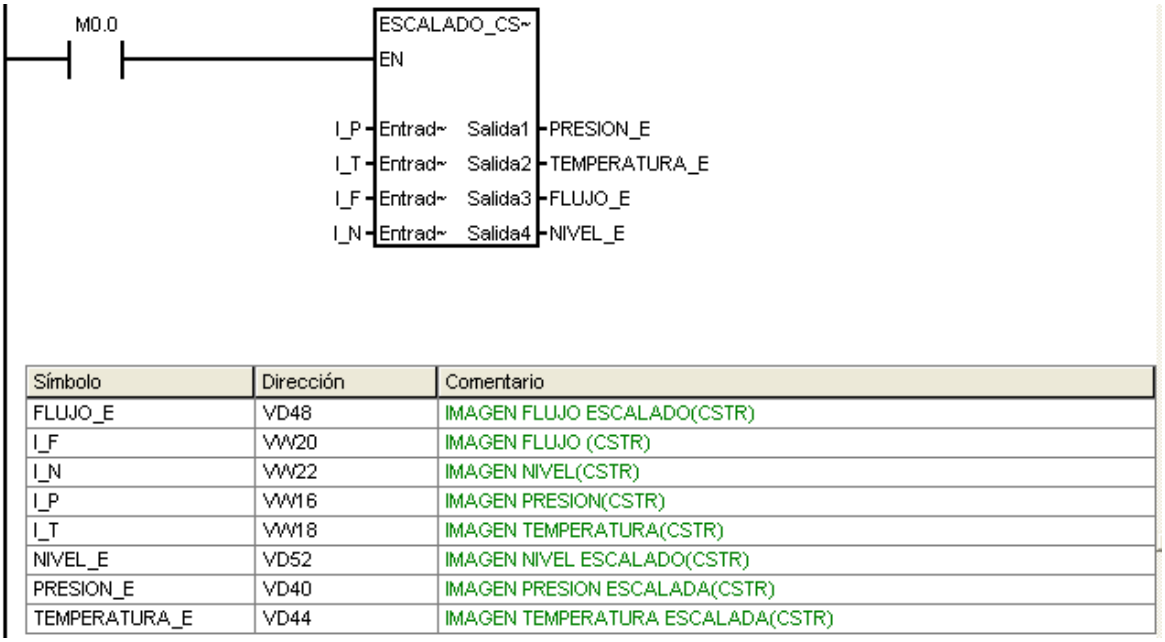
La verificación de los tres restantes módulos analógicos y el digital se realiza de la misma manera. Para el segundo módulo (EM235), el parámetro de entrada con que se llama a la subrutina “COMPRUEBA _ ERROR” es 1, para el tercer

módulo (EM 232) es “2” para el cuarto módulo (EM 232) es “3” y para el último módulo (EM223) es “4”. Los estados de error (energía, rango, identificación y configuración) se guardan en las direcciones VB10, VB11 y VB 12 para mediante la interfaz gráfica guardar el estado de errores en un archivo de texto.

2.3.3 Escalado de las entradas analógicas

Los módulos analógicos EM235 y EM232 están configurados de manera que acepten entradas de corriente entre 4-20mA, que en la memoria del PLC representan números enteros entre 6400- 32000. Cada entrada corresponde a diferentes variables como presión, temperatura, flujo, nivel de líquido que tienen su propia escala. En esta sección se limita a convertir la escala entera de 6400-32000 a escala real de 0-1 que será mostrado en la interfaz gráfica. Una escala real (0-1) permite al usuario una rápida apreciación del nivel de la variable.

Subrutina “ESCALADO _ CSTR”. Esta subrutina tiene cuatro parámetros de entrada: Entrada1, Entrada2, Entrada3, Entrada4, y cuatro de salida: Salida1, Salida2, Salida3, Salida4. A los parámetros de entrada se asignan las imágenes de presión, temperatura, nivel y flujo del líquido del CSTR. Las salidas corresponden a las entradas anteriores pero escaladas, y se guardan en las direcciones VD40, VD44, VD48 y VD52, como muestra la figura 2.8, con los nombres PRESIÓN_E, TEMPERATURA_E, FLUJO_E y NIVEL_E, respectivamente.



Símbolo	Dirección	Comentario
FLUJO_E	VD48	IMAGEN FLUJO ESCALADO(CSTR)
I_F	VW20	IMAGEN FLUJO (CSTR)
I_N	VW22	IMAGEN NIVEL(CSTR)
I_P	VW16	IMAGEN PRESION(CSTR)
I_T	VW18	IMAGEN TEMPERATURA(CSTR)
NIVEL_E	VD52	IMAGEN NIVEL ESCALADO(CSTR)
PRESION_E	VD40	IMAGEN PRESION ESCALADA(CSTR)
TEMPERATURA_E	VD44	IMAGEN TEMPERATURA ESCALADA(CSTR)

Figura 2.8. Llamada a la subrutina *ESCALADO_CSTR*

Las entradas escaladas, en real, deben estar en un rango de 0 a 1, de esta manera 0 corresponde al número entero 6400 y 1 a 32000. Por lo tanto lo primero que hace el programa de la figura2.9 es restar 6400 de la entrada entera, luego convertir de entero a entero doble, de entero doble a real y luego multiplicar por un factor de escala, figura2.9. El factor de escala se obtiene con la siguiente fórmula:

$$factor\_de\_escala = \frac{Vmáx\_real - Vmín\_real}{32000 - 6400} \qquad (1)$$

Donde $Vmáx\_real$ es 1 y $Vmín\_real$ es 0 para todas las entradas analógicas del CSTR, dando un factor de escala igual a 3.9063E-5. Si se desea cambiar de rango, son los dos valores los que hay que modificar, por ejemplo para una escala entre 0-15, $Vmáx\_real$ es 15 y $Vmín\_real$ es 0

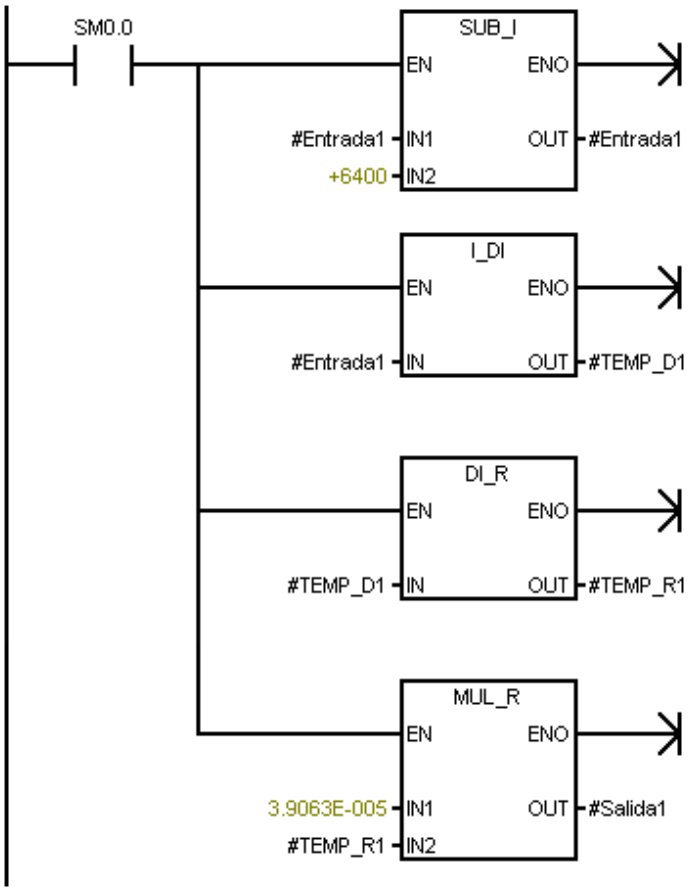


Figura 2.9. Convierte la imagen entera de la entrada de nivel a un valor real escalado.

En la figura 2.9 los símbolos TEMP_D1 Y TEMP_R1 son variables locales de la subrutina que no son ni entradas ni salidas, solo ayudan a la conversión.

Además, dentro del programa de la subrutina “ESCALADO _ CSTR”, existen tres bloques adicionales similares al anterior, para la conversión de la imagen de temperatura, flujo y nivel.

En el programa principal, se tiene una segunda subrutina para el escalado de las 4 restantes entradas analógicas, dos entradas analógicas del MAGLEV y dos de la DOMÓTICA. Para ello se utiliza la subrutina *“ESCALADO _MAGLEV_ Y_ DOMOTICA”*. El programa de dicha subrutina es exactamente el mismo que el

correspondiente a la subrutina *“ESCALADO _ CSTR”.* Los parámetros de entrada son las imágenes de las entradas analógicas: IMAGEN MAGLEV1 (I_M1), IMAGEN MAGLEV2(I_M2), IMAGEN LUMINOSIDAD(I_L) y IMAGEN VENTILACION(I_V) y las salidas son las anteriores escaldas entre 0-1, llamadas MAGLEV1_E, MAGLEV2_E, LUMINOSIDAD_E, Y VENTILACION _E.

2.3.4 Procesamiento de alarmas.

Subrutina *PROCESA _ ALARMAS*. En esta subrutina se procesan dos alarmas de la DOMÓTICA, una alarma del ASCENSOR, cuatro alarmas del CSTR y dos alarmas del MAGLEV.

Alarmas de la DOMÓTICA. La alarma de humo correspondiente a la variable AH del programa del PLC y se procesa de acuerdo a la Red de Petri de la figura1.4, del capítulo anterior.

El diagrama de la Red de Petri de la figura 1.4, es llevada al programa de la subrutina PROCESA _ ALARMAS en el Siemens, figura 2.10.

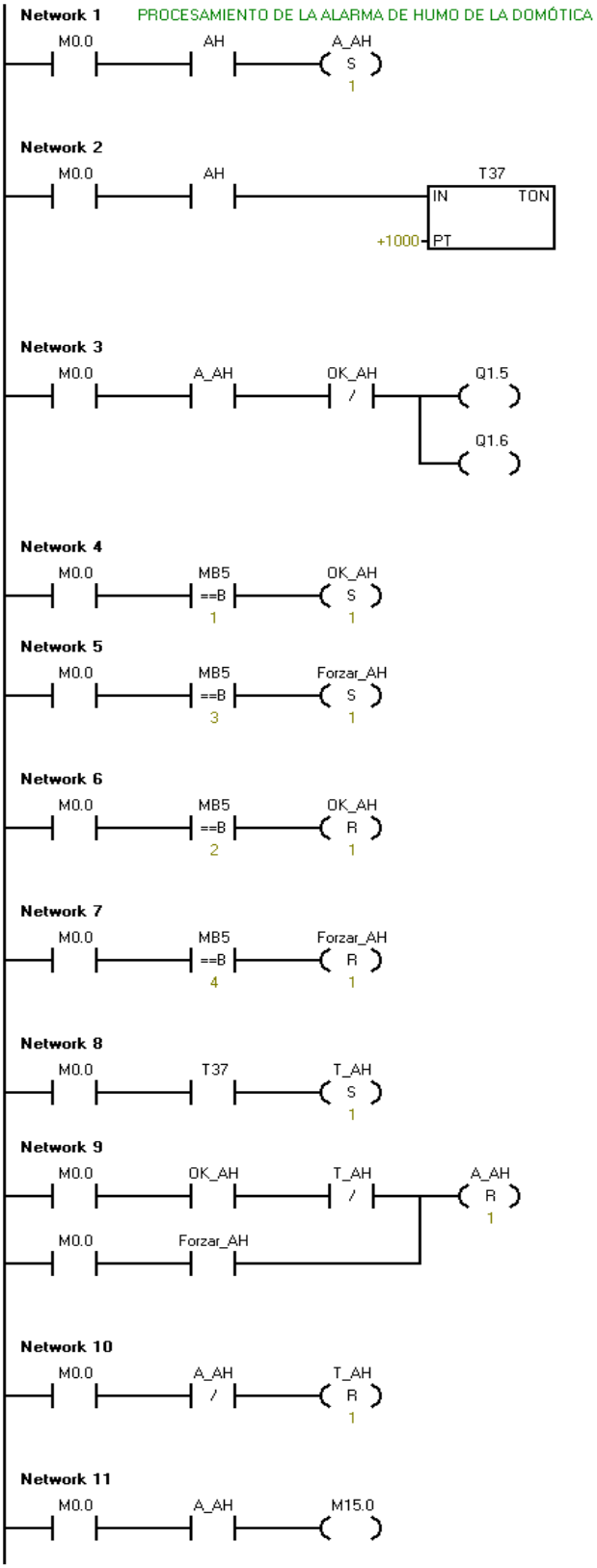


Figura 2.10. Programa del tratamiento de la alarma humo

Cuando se activa la entrada de alarma de humo (AH), en el primer segmento en la figura 2.10, se activa el estado de alarma (A_AH) mediante un seteo a 1. El segundo segmento inicia la cuenta en el temporizador T37. El tercer segmento activa tanto la salida sonora (Q1.5) como la visual (Q1.6). Los siguientes cuatro segmentos, son utilizados para el reconocimiento de las alarmas, así, si MB5 es igual a “1” (valor ingresado desde Visual Basic), se activa el reconocimiento OK_AH, esto apaga las salidas y también regresa al sistema a “Operación Normal” si la bobina T_AH no está activa, caso contrario únicamente es posible mediante Forzar_AH como muestra el segmento 9. Para desactivar el reconocimiento OK_AH se debe escribir en MB5 desde Visual Basic el número “2”, y para activar y desactivar Forzar_AH el número 3 y 4 respectivamente, mediante los botones de comando del **Anexo2**.

Como se verá en el siguiente capítulo, la escritura de dichos valores se logra pulsando botones de comando programados. La tabla del **Anexo 2**, contiene los valores de MB5, los botones de comando que permiten asignar y el evento que se produce.

La bobina T_AH se pone en alto, al activar el bit de temporización, lo cual ocurre cuando la entrada de alarma permanece activa por más de un determinado tiempo (establecido por el usuario desde la interfaz gráfica). El segmento 10 borrar la marca T_AH, únicamente cuando el estado de alarma está desactivado. Finalmente se mueve el estado de alarma (A_AH) a la dirección M15.0 para la visualización en el SCADA.

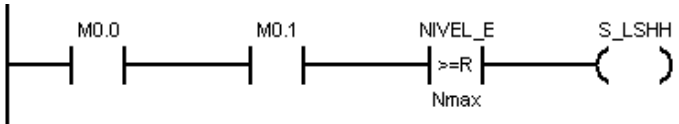
La alarma de seguridad (AS) se procesa de la misma manera que la correspondiente de humo. En este caso, el estado de alarma se llama A_AS, el temporizador T38, la salida sonora y visual Q1.7 y Q2.0 (de acuerdo al **Anexo1)**,

respectivamente. Los reconocimientos se realizan mediante OK_AS ó Forzar_AS según el **Anexo2**.

Alarma del ASCENSOR. El sensor de paracaídas representa una alarma llamada SPCAIDAS. El tratamiento es similar a tratamiento de las alarmas de la DOMOTICA de la sección anterior. En este caso la salida sonora y visual están en Q2.4 y Q2.5 (tabla3 del **Anexo1**).

Alarmas de CSTR. El tratamiento de alarmas de nivel alto (LSHH), nivel bajo (LSLL), alarma de sobre presión (PT) y de sobre temperatura (TT) del líquido merecen un tratamiento adicional al aplicado a las alarmas del ASCENSOR y de la DOMÓTICA. Así, además de conocer el tiempo en que las entradas de alarma están activas, es necesario saber si el nivel de líquido, presión o temperatura están dentro de un rango predeterminado para regresar al estado normal luego de haber ocurrido una alarma.

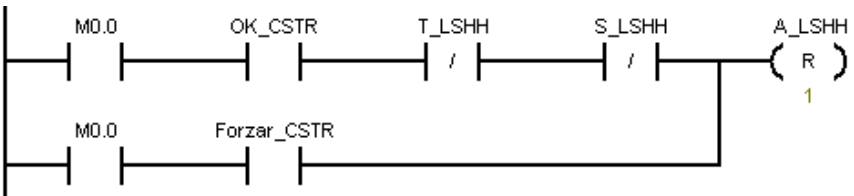
El tratamiento adicional implica un par de segmentos adicionales, donde se compara el valor de las entradas analógicas con ciertos niveles. Por ejemplo, cuando se activa la alarma LSHH en la figura2.11, se compara el valor de la entrada de nivel de líquido escalada con una constante establecida por el usuario desde la interfaz gráfica, por lo general 0.9.



Símbolo	Dirección	Comentario
NIVEL_E	VD52	IMAGEN NIVEL ESCALADO(CSTR)
Nmax	VD692	Contiene el nivel alto del nivel de una variable para el tratamiento de alarma
S_LSHH	M12.3	Indica si el nivel de liquido del CSTR es mayor a un cierto valor

Figura 2.11. Compara el nivel de líquido escalado con una constante establecido por el usuario.

Si está sobre ese nivel se activa la bobina (S_LSHH) que impide regresar al estado “Operación Normal” con el reconocimiento de alarmas llamado OK_CSTR del formulario CSTR del Visual Basic. Solo es posible regresar, en la condición anterior, mediante una acción forzada, llamada reconocimiento Forzar_CSTR (Figura2.12).



Símbolo	Dirección	Comentario
A_LSHH	M12.0	Estado de alarma de nivel alto del CSTR
Forzar_CSTR	M12.1	Regresa al estado normal luego de haber activado el LSHH
OK_CSTR	M12.2	Apaga las salidas sonoras y visuales de la alarma del LSHH
S_LSHH	M12.3	Indica si el nivel de liquido del CSTR es mayor a un cierto valor
T_LSHH	M12.4	Indica si el switch LSHH estaba activo mayor a un tiempo establecido

Figura 2.12. Es posible desactivar A_LSHH con OK_CSTR ó Forzar_CSTR.

El tratamiento de sobre presión y sobre temperatura se realiza exactamente igual al correspondiente de sobre nivel LSHH.

2.3.5 Asignación de los parámetros de los PIDs.

Se necesitan de PIDs para el control de flujo, nivel, presión y temperatura en el CSTR y ventilación de la DOMÖTICA. Cada uno de los PID tiene parámetros como SetPoint, constante proporcional (K), tiempo integral (Ti) y tiempo derivativo (Td). Los dos primeros están en una escala de 0-1, mientras que los últimos en minutos.

Los valores reales de los parámetros de los PIDs se ingresan desde la interfaz gráfica del Visual Basic a las direcciones de memoria VD716, VD720, VD724 y VD728 del PLC. Los valores de SetPoint y K son números reales entre 0-100. La asignación se realiza mediante una subrutina PARÁMETROS_PID, que es llamada desde el programa principal en cada ciclo del programa.

Al hacer un clic sobre el botón de control *PRESION* del formulario *Principal* de Visual Basic, se habilita la marca M8.0 de la figura 2.13 y se habilitan los cuatro bloques del segmento de la subrutina PARÁMETROS_PID. Las dos primeras funciones, dividen el contenido de VD716 (SetPoint) y VD720 (K) entre 100 y los guardan en las direcciones VD740 y VD744. Las dos últimas funciones mueven el contenido de VD724 (Ti) y VD728 (Td) a las direcciones VD748 y VD752. De esta manera se tiene los valores de los parámetros del PID de la presión en las direcciones VD740, VD744, VD748 y VD752 para ser utilizados cuando se programen los controladores.

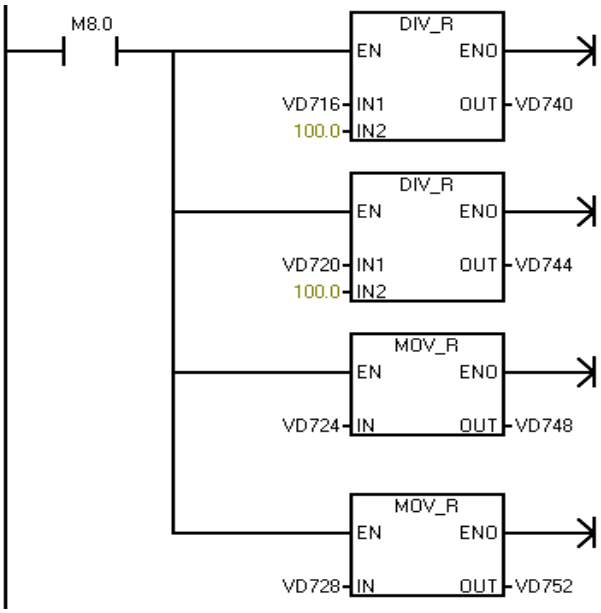


Figura 2.13. Programa en la Subrutina *PARAMETROS_PID,* para asignar los parámetros a la variable PRESIÓN

Los restantes segmentos de la subrutina PARÁMETROS_PID son similares al anterior y se utilizan para asignar parámetros de los PIDs de las variables restantes.

2.3.6 Subrutina PULSADORES _ ASCENSOR

Esta subrutina permite al usuario modificar ciertas localidades de memoria del PLC desde el formulario ASCENSOR de Visual Basic. En la subrutina *PULSADORES _ ASCENSOR*, las dos primeras líneas (figura2.14) se utilizan para activar y desactivar la marca M16.0.

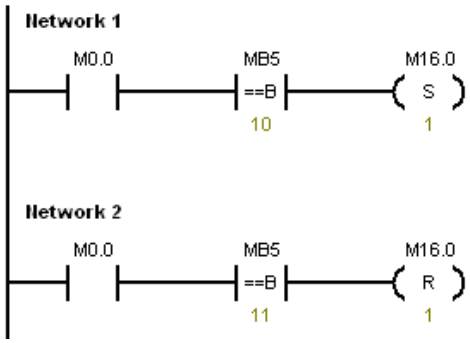


Figura 2.14. Segmentos en la subrutina *PULSADORES _ ASCENSOR,* que permiten activar y desactivar M16.0

Con un clic sobre el botón de comando *PCP1* del formulario ASCENSOR en Visual Basic se asigna el valor 10 a la dirección MB5 del PLC. De acuerdo a la figura2.14, cuando MB5 es 10 se pone en alto la marca M16.0. Ahora, el botón de comando *PCP1* del bloque *Desactivar* desactiva M16.0 al asignar el valor 11 a MB5 cuando se hace un clic sobre él. En el programa principal del PLC, el

segmento 28 tiene PCP1 en paralelo con M16.0 conectados a la marca M3.4, figura2.15. Por lo que activar M16.0 desde la interfaz gráfica es igual que activar el pulsador de cabina del piso1. En la programación del programa del ASCENSOR debe utilizarse la marca M3.4, mas no PCP1 ni M16.0.

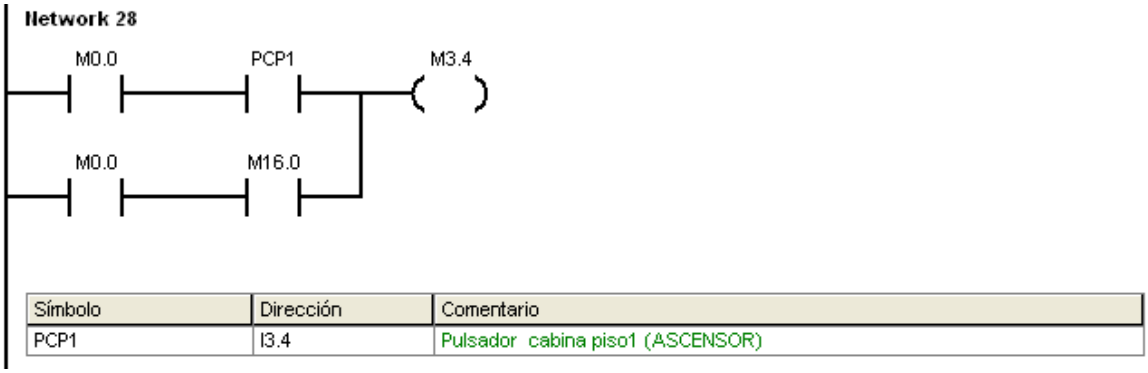


Figura 2.15. En el programa principal, PCP1 está en paralelo con M16.0

Para el resto de pulsadores del ASCENSOR se aplica el mismo principio. Los números que se escriben y botones de comando correspondientes se detallan en el **Anexo2**.

CAPÍTULO 3

SISTEMAS SCADA: LABORATORIO DE CONTROL DE PROCESOS

3.1 Diseño del sistema SCADA

El ordenador es el encargado de la supervisión de las entradas y salidas de las plantas, mediante las ventanas abiertas en Visual Basic. Cada cambio en la planta debe ser representado. Para un mejor entendimiento por parte del usuario las pantallas se configuran en tres áreas: Proceso global, partes significativas del proceso, y área con asignación de mando para acciones sobre parámetros de la planta.

En el diseño de las pantallas se toman en cuenta las siguientes consideraciones:

- La organización de las pantallas y parte de las mismas, se realiza de tal manera que muestre áreas diferenciales fáciles de entender. Por ejemplo, botoneras de entradas, alarmas del sistema, subpantallas, etc.
- Las señales de controles son agrupadas por funciones, así mismo los nombres asignados están relacionadas con la función.
- La utilización de colores, es otro aspecto muy importante. Desde el color del fondo de las pantallas hasta el color de cada botonera, permite una mejor apreciación e interpretación de la información. Así por ejemplo, asignar colores más vivos como el rojo o verde a los botones correspondientes a las alarmas.

El sistema SCADA es desarrollado mediante el software de Visual Basic. Para la supervisión del proceso se realiza mediante varias pantallas dentro de una misma aplicación. Se tiene una pantalla principal a partir de la cual se puede acceder al resto de pantallas.

3.2 Definiciones de Visual Basic 6

Visual Basic 6 representa una versión mejorada del lenguaje de programación Visual Basic de Microsoft. Visual Basic reduce el trabajo por parte del usuario, de tal manera que muchas tareas son simples como arrastrar con el ratón objetos gráficos en la pantalla. El lenguaje de programación que utiliza es Basic.

Visual Basic 6 es más que un lenguaje de programación. Como su nombre (*Visual*) lo indica, durante la creación de un programa es posible colocar objetos gráficos en la pantalla y la configuración fácil de sus parámetros y atributos. De esta manera el usuario tiene la capacidad de interactuar gráficamente con la pantalla mediante el ratón, teclado e impresora.

Un programa de Visual Basic puede estar en uno de dos modos: en el modo de diseño o en el modo de ejecución. En el modo de diseño se colocan controles en el formulario con sus respectivas propiedades como también funciones. En el modo de ejecución se prueba el programa creado.

Cuando se trabaja en una aplicación de Visual Basic se encuentra con los siguientes términos:

- **Controles:** son cada uno de los botones, cajas de texto, barras de desplazamiento, gráficos, menús, etc. A cada control se debe asignar un nombre para llamar dentro del programa.
- **Formularios:** Un formulario es una ventana que contiene elementos de control. Una aplicación puede contener uno o más formularios en función con la complejidad del proyecto. Así, en el presente trabajo se necesita de múltiples formularios.
- **Objeto:** es un elemento de la aplicación, como un control o formulario que contiene instrucciones programadas.
- **Propiedades:** establece el aspecto de los controles y formularios, como colores, tamaños, posición y la manera de responder a acciones del usuario.
- **Proyecto:** Un proyecto es un conjunto de archivos. Estos archivos contienen formularios con sus respectivas descripciones.

3.3 Crear un proyecto en Visual Basic6

Cuando se inicia Visual Basic desde el menú Inicio de Windows, aparece el cuadro de dialogo *Nuevo proyecto* que se muestra en la figura3.1.

Figura 3.1. Cuadro de dialogo Nuevo Proyecto

Seleccionando el icono etiquetado *EXE estándar* se crea una aplicación que se puede compilar o ejecutar de la forma tradicional. Se puede seleccionar otros iconos dependiendo de la aplicación que se requiera.

Una vez seleccionado uno de los iconos, se muestra una pantalla con varias partes: barra de herramientas, cuadro de herramientas, ventana formulario, ventana posición del formulario, ventana proyecto y ventana propiedades, como muestra la figura3.2.

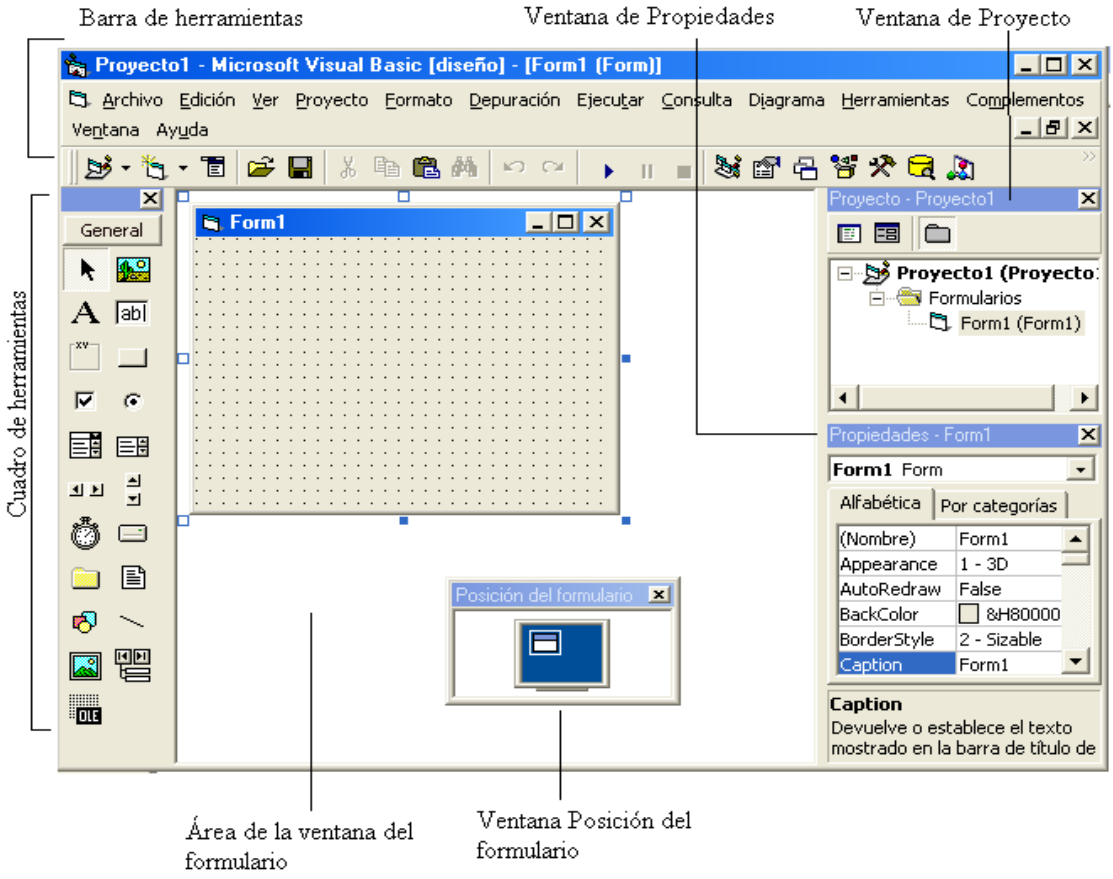


Figura 3.2. Componentes de un proyecto

La barra de herramientas se encuentra justo debajo de la barra de menús. Ésta contiene iconos como agregar formularios, guardar proyecto, etc. Para propósitos de la SCADA de este trabajo se agregan cuatro formularios, cuyos nombres son asignados mas adelante.

En el cuadro de herramientas se encuentran los controles que se pueden agregar al formulario con tan solo arrastrar con el ratón.

En el área de la ventana de formulario se encuentran todos los formularios creados, y es el lugar donde se desarrolla la mayor parte del proyecto.

La ventana Posición del formulario indica una vista previa de la ubicación en la pantalla en tiempo de ejecución. Ésta ventana permite cambiar la posición de los formularios con tan solo arrastrar con el ratón.

La ventana de proyecto muestra los componentes del mismo en forma de árbol, como formularios, módulos.

La ventana de propiedades permite asignar nombres y propiedades a controles y formularios. Así, a los formularios creados anteriormente (Form1, Form2, Form3, Form4 y Form5) se asignan los siguientes nombres Principal, CSTR1, MAGLEV, DOMÓTICA y ASCENSOR, respectivamente.

3.4 Simatic Microcomputing.

Sematic MicroComputing es un software de la tecnología Microsoft que permite acceder y manipular datos de máquinas de control y automatización, como el PLC S7-200 de CPU226, a través de los controles ActiveX.

Para insertar los controles de SIMATIC en el cuadro de herramientas del Visual Basic se procede de la siguiente manera: **Proyecto>Componentes,** y se selecciona *Siemens SIMATIC Data control* y *Siemens_SIMATIC_UserControls,* de esta manera se agregan en el cuadro de herramientas los controles: Data, Button, Edit, Label y Slider.

Data. Permite la conexión con el PLC Siemens S7-200.

Button. Permite acceder por bits a los datos de la memoria de la CPU de PLC Siemens. Tiene dos valores 0 ó 1, por defecto 0 = OFF de color rojo y 1=ON de color verde.

Edit. Se utiliza para acceder y modificar datos en la memoria del PLC Siemens en formato de bytes, palabras, palabras dobles, con tan solo escribir un número.

Label. Permite leer valores de la memoria del PLC Siemens, los cuales son mostrados en formato String.

Slider. Permite acceder a la memoria del PLC Siemens en bytes, palabras o palabras dobles. Ajustando el nivel en el Slider se puede modificar valores de la memoria.

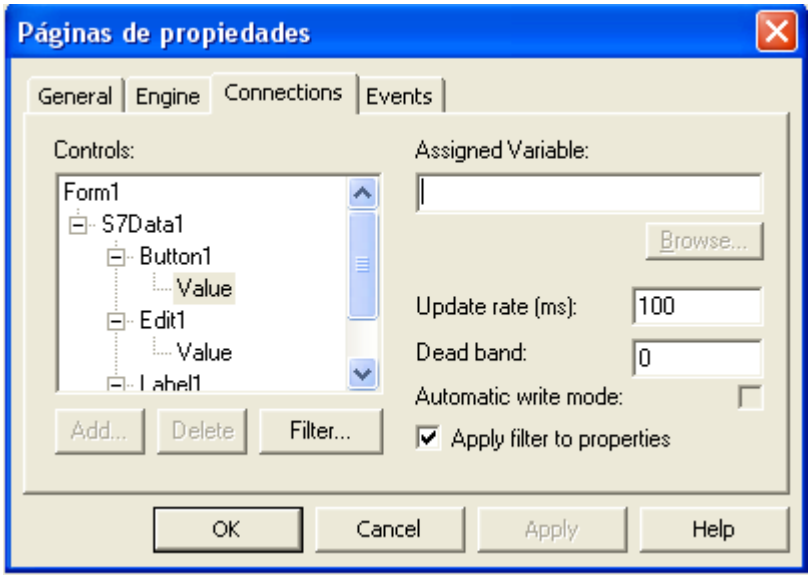


Figura 3.3. Ventana de propiedades del control S7Data de Microcomputing

Para conectar un control del Microcomputing con el Siemens se va a la ventana de propiedades del control S7Data, en *Conexiones* se selecciona el control y en el cuadro de Variable Asignada se coloca la dirección de memoria del PLC que se desea leer o escribir. La figura 3.3 muestra las instrucciones anteriores.

3.5 Diseño de los formularios.

A continuación se describe la funcionalidad de cada uno de los formularios que conforman la Interfaz gráfica del sistema SCADA.

3.5.1 Formulario: Principal.

Por defecto, el formulario Principal, en tiempo de ejecución tiene la apariencia de la figura3.4.

En este y todos los formularios, se utiliza un botón de comando etiquetado como *IniciarSCADA,* mediante un clic sobre el mismo se habilita los temporizadores o relojes del Visual Basic que por defecto estaban inhabilitados. La razón de inhabilitarlos por defecto, es que en el transiente de ejecución, muchas operaciones en el Visual Basic pueden ejecutarse antes que se logre una comunicación con el PLC y ocasionar conflictos; por esta razón todos los formularios tienen dicho botón de comando para habilitar los temporizadores

Figura 3.4. Formulario Principal en tiempo de ejecución

En la figura3.4, los cuatro botones de comando etiquetados como *DOMOTICA, ASCENSOR, CSTR* y *MAGLEV*, sirven para mostrar los formularios que llevan el mismo nombre. Por ejemplo, con un clic sobre *DOMÓTICA* se carga el formulario *DOMOTICA.frm* que ocupa toda la pantalla del computador.

En el bloque *Estados de conexión de módulos de los módulos de expansión,* se dispone de cinco botones de control para mostrar el estado de conexión de los módulos acoplados al PLC. Los botones tienen un color rojo cuando hay un error de identificación y error en los módulos, y un color verde cuando no existen dichos errores. De esta manera los botones etiquetados como *Error en Modulo0, Error en Modulo1, Error en Modulo2, Error en Modulo3, Error en Modulo4*

permiten conocer el estado de error del primer módulo EM235, del segundo EM235, del primer EM232, del segundo EM232 y del módulo digital EM223 acoplados al PLC.

El botón de comando y el cuadro de texto del bloque *Modificar Parámetros de Alarmas* permiten mostrar ciertos controles cuando se escribe “2009” en el cuadro de texto, como muestra la figura3.5.

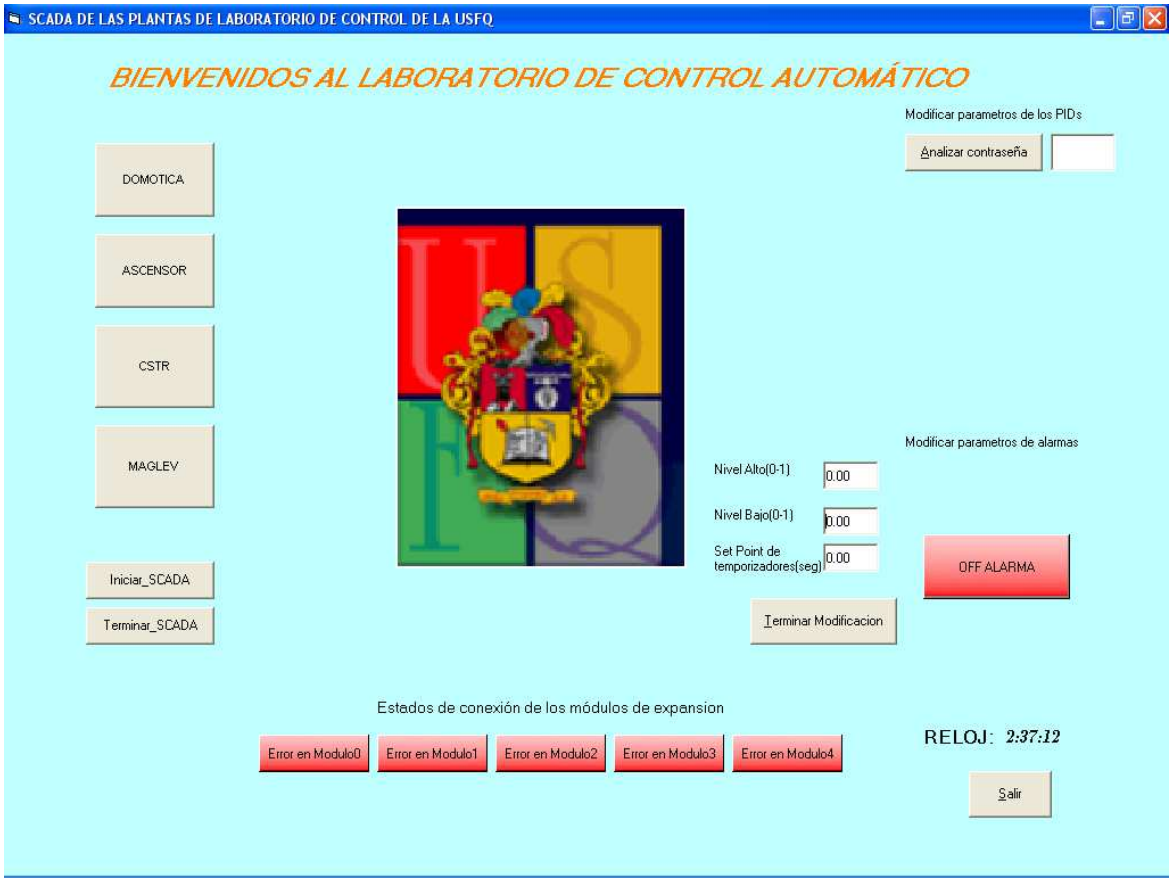


Figura 3.5. Formulario *Principal* en tiempo de ejecución, con el bloque de los parámetros de alarmas.

El bloque desplegado contiene tres editores de control que permiten asignar valores *Nivel Alto* y *Nivel Bajo* entre 0-1, y el SetPoint en segundos de los temporizadores, utilizados en la subrutina de tratamiento de alarmas del PLC. El

Nivel Alto y *Nivel Bajo* representan valores de las variables de presión, temperatura, flujo, nivel y ventilación; el primero por lo general 0.9 y el segundo 0.1. Estos valores, como se verá en los siguientes formularios, influyen en el tratamiento de alarmas. Una vez terminada la modificación, con un clic sobre *Terminar Modificación* se ocultan los controles que fueron puestos como visibles.

También se dispone de un botón de control para mostrar si alguna de las entradas de alarmas de las plantas involucradas se activó. Éste botón está etiquetado como *OFF ALARMA* (color rojo) si ningún estado de alarma está activado y como *ON ALARMA* (color verde) si alguno lo está.

Debajo de la etiqueta *Modificar Parámetros de los PIDs* se dispone de un botón de comando *Analizar contraseña* y un cuadro de texto. Si en el cuadro de texto se escribe “11461”, y se hace un clic sobre el botón de comando, se carga el formulario PIDs que permite al usuario conocer los parámetros de los PIDs de *PRESIÓN, TEMPERATURA, FLUJO, NIVEL, VENTILACIÓN* y cambiar los mismos.

Para inhabilitar los temporizadores utilizados en el programa de Visual Basic, se utiliza otro botón de comando llamado *TerminarSCADA*. Es necesario inhabilitar los temporizadores antes de cerrar un formulario, ya que podría ser que algunas operaciones aún se estén realizando dentro del Visual Basic mientras que la conexión con el PLC ya se terminó. Todos los formularios tienen un botón de comando con la misma funcionalidad.

Para terminar la ejecución del programa, basta hacer un clic sobre el botón de comando etiquetado como *Salir.*

3.5.2 Formulario: PIDs

Este formulario, permite al usuario asignar parámetros a los PIDs de las variables de presión, temperatura, flujo, nivel y ventilación. Para asignar parámetros a una variable, se hace un clic sobre el botón correspondiente y se escriben los valores de SetPoint, K, Ti y Td en los editores de control correspondientes. El SetPoint y la gananica proporcional (K) son expresados en porcentaje, y el tiempo integral (Ti) y tiempo derivativo (Td) en minutos. Cuando se termina la modificación, se hace un clic sobre el botón de la variable modificada para que vuelva al color rojo original y se puede modificar los parámetros de otra variable.

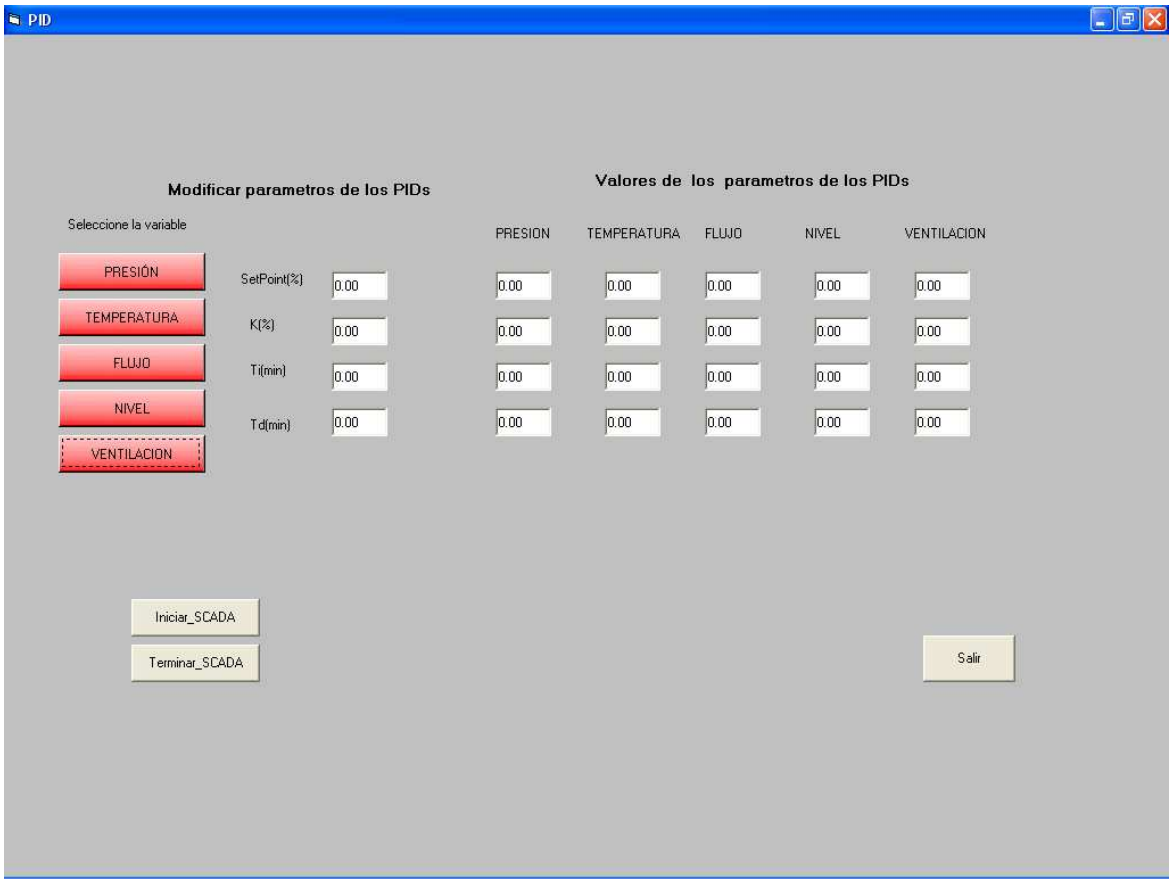


Figura 3.6. Formulario PIDs.

El bloque derecho, *Valores de los parámetros de los PIDs*, muestra los valores de los parámetros de los PIDs de las variables de control. El valor del SetPoint y la constante K están en porcentajes, y el tiempo integral y tiempo derivativo en segundos.

3.5.3 Formulario: DOMÓTICA

Éste y los demás formularios contienen botones de control etiquetados con los nombres de la variable cuyo estado se desea conocer, así por ejemplo, para visualizar el estado de la variable puerta cerrada (PC, tabla1 del **Anexo1**) del programa del PLC, se crea un botón etiquetado como PC. Del mismo modo, los botones de comando están etiquetados con los nombres de las variables en el PLC que se desea modificar.

Para comunicar la interfaz gráfica con el PLC, a través del formulario principal, se debe hacer un clic sobre el botón de comando etiquetado como *Iniciar_SCADA* para activar el temporizador. Antes de cerrar el formulario es necesario desactivar el temporizador con un clic sobre el botón de comando *Terminar_SCADA,* las razones se describen en la sección anterior. Como se verá mas adelante todos los formularios requieren de proceso anterior.

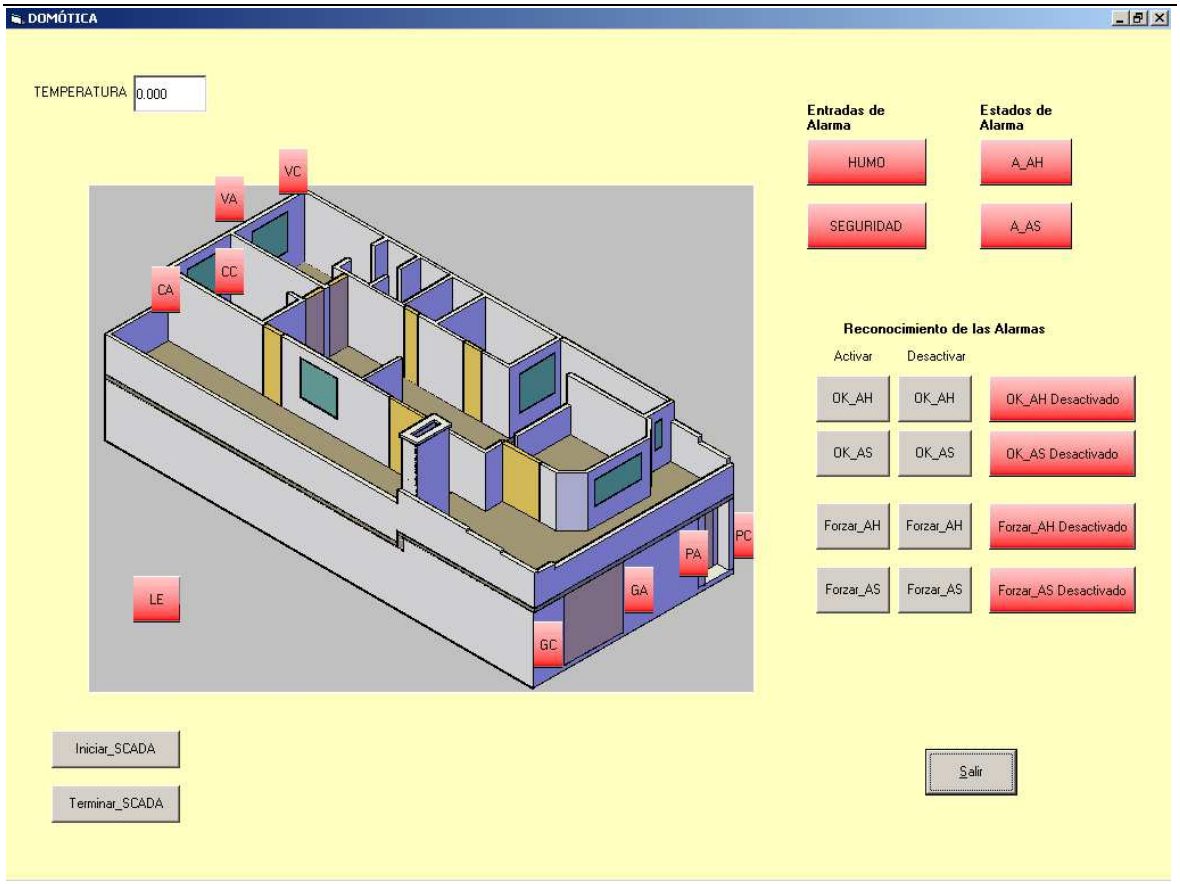


Figura 3.7. Formulario DOMÓTICA en tiempo de ejecución

En la figura3.7, la mayoría de los botones son para visualizar el estado de las variables del PLC correspondientes a la DOMOTICA (Tabla1 **Anexo1**). Estos botones tienen un color rojo cuando las entradas respectivas del PLC están desactivadas y verdes cuando están activadas.

En el bloque Reconocimiento de alarmas, al hacer un clic sobre los botones de comando se tiene un determinado evento. Por ejemplo, con un clic en *OK_AH* se apaga las salidas de alarma pero no siempre el estado de alarma, el estado de alarma se apaga si el tiempo que en que estaba activada la entrada de alarma de humo es menor a un determinado tiempo (valor dado asignado por el usuario en el formulario Principal), caso contrario solo es posible apagar dicho estado

mediante el botón *Forzar_AH.* Se tiene el mismo evento con el botón de comando *OK_AS y Forzar_AS* para la alarma de seguridad.

Un editor de control muestra un número real entre 0-1 que representa la temperatura de una habitación. Dicho rango puede ser modificado fácilmente a pedido del usuario en el programa del PLC o directamente en Visual Basic.

Para terminar la ejecución basta hacer un clic sobre el botón de comando etiquetado como *Salir.*

3.5.4 Formulario: ASCENSOR

La figura3.8 contiene botones que muestran el estado de las entradas digitales del PLC correspondientes al ASCENSOR, un bloque llamado *Pulsadores* y otro de *Reconocimiento de Alarmas*.

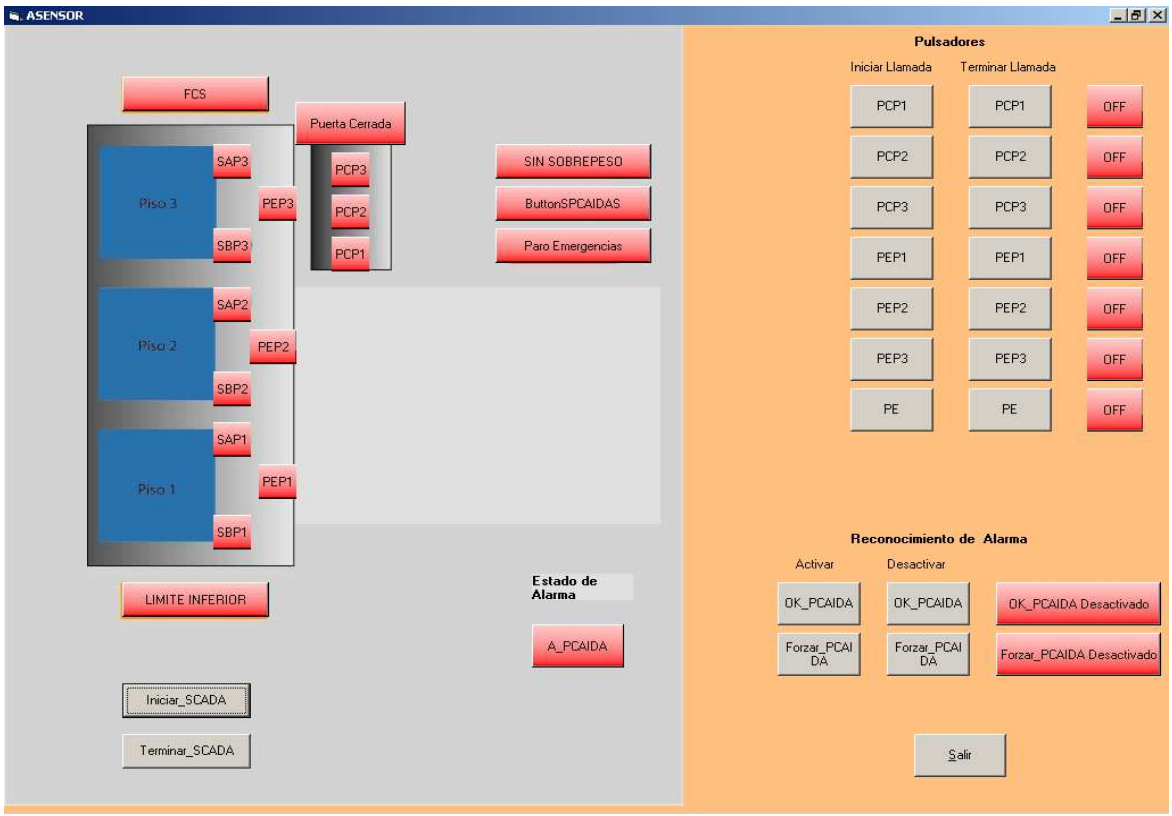


Figura 3.8. Formulario ASCENSOR en tiempo de ejecución

Las entradas digitales I2.6 a I4.7 del PLC corresponden al *ASCENSOR*, y muestran es estado de los switch y pulsadores de acuerdo a la tabla1 del **Anexo1**. Los botones tienen un color rojo cuando las respectivas entradas están desactivadas y un color verde cuando están activadas.

La entrada I4.7 representa una entrada de alarma del sensor de paracaídas llamada *SPCAIDAS* que al activarse, activa el botón *A_CAIDA* que representa el estado de alarma y las salidas Q2.4 y Q2.5 (sonora y visual).

Los botones de comando *OK_PCAIDA* y *Forzar_PCAIDA* que están debajo de la etiqueta *Activar* son para activar el reconocimiento de las alarmas. Con un clic sobre *OK_PCAIDA* se desactivan las salidas Q2.4 y Q2.5 del PLC, y se desactiva el estado de alarma si el tiempo en que la entrada que produjo la alarma estaba activada por un intervalo de tiempo menor a un cierto valor, el cual es determinado por el usuario el formulario Principal. Con un clic sobre *Forzar_PCAIDA*, se desactivan tanto las salidas como el estado de alarma.

Ciertas variables del ascensor se pueden activar desde el SCADA, por ejemplo la variable pulsador cabina piso1 (PCP) puede activarse con un clic sobre el botón *cmdIPCP1* y desactivar con un clic sobre *cmdTPCP1 (***Anexo2)***.* Esto permite al usuario realizar llamadas del ascensor y direccionar a la cabina desde la interfaz gráfica.

3.5.5 Formulario: CSTR1

La figura3.9 muestra el valor de cuatro entradas analógicas y cuatro digitales del PLC, los estados de alarma y el bloque de reconocimiento de las alarmas.

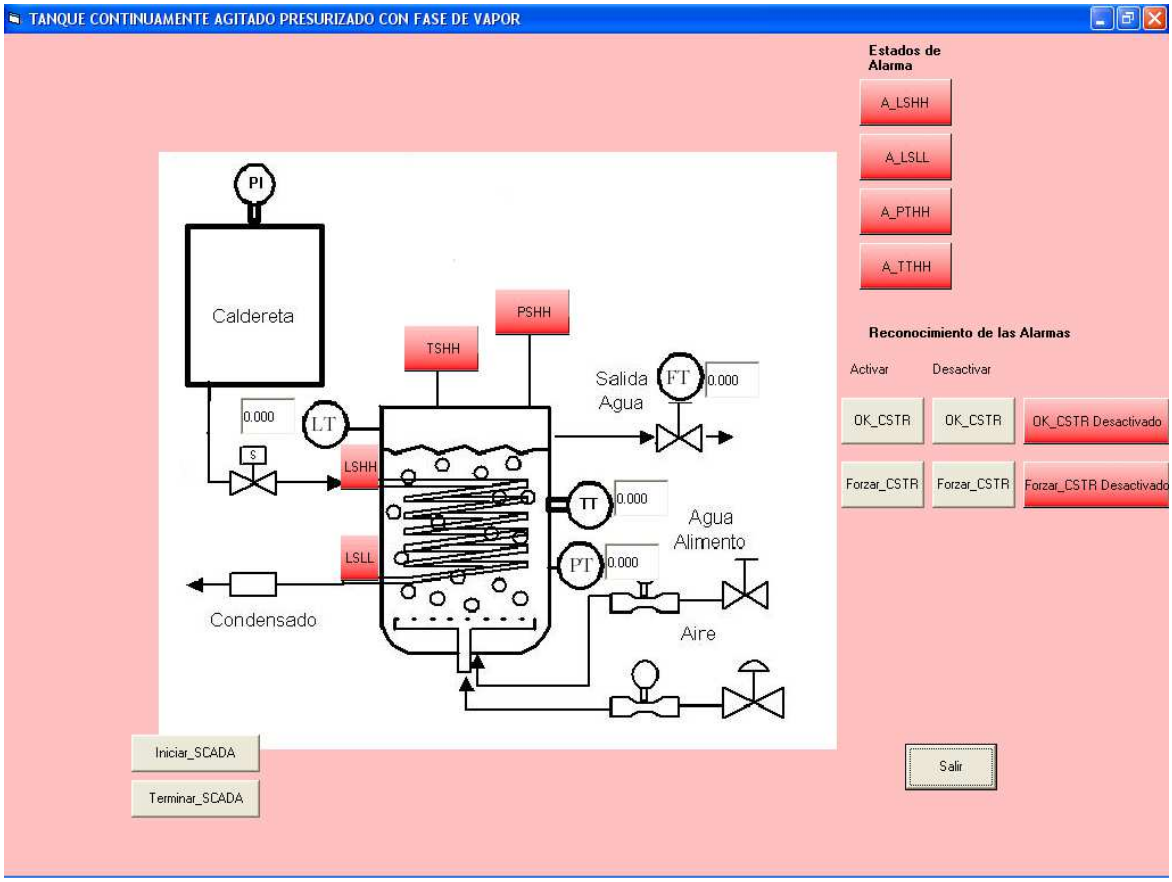


Figura 3.9. Formulario CSTR en tiempo de ejecución

Se dispone de cuatro editores de control que permiten visualizar los valores de las entradas analógicas AIW0, AIW2, AIW4 y AIW6 del PLC que se serán conectados a los sensores de PRESIÓN(PT), TEMPERATURA(TT), FLUJO(FT) Y NIVEL(LT) del tanque continuamente agitado. El rango de los controles es de 0 a 1, siendo lineal para una entrada de corriente de 4mA a 20mA. Para menos de 4mA, por defecto los controles muestran cero.

Cuatro botones muestran el estado de las entradas I0.1, I0.2, I0.3 y I0.4 del PLC que se serán conectados a los Switch de nivel alto (LSHH), nivel bajo (LSLL), sobre presión (PSHH), sobre temperatura (TSHH) del CSTR. Los

botones tienen un color rojo cuando las entradas están desactivadas y un color verde cuando se activan.

Al activar una entrada de alarma se producen dos eventos: se activa un botón del bloque *Estado de Alarma* correspondiente y también dos salidas Q0.4 y Q0.5 del PLC (tabla2 del **Anexo1**). Existe un botón que muestra el estado de alarma para cada entrada de alarma.

En el bloque de reconocimiento de alarmas, se dispone de cuatro botones de comando, dos para activar el reconocimiento *OK _CSTR* y *Forzar_CSTR* y dos para desactivar. Al activar el reconocimiento con *OK _CSTR* se apagan las salidas Q0.4 y Q0.5 (sonara y visual) del PLC, pero no siempre el estado de alarma.

El botón de comando *OK _CSTR* desactiva el estado de alarma si se cumplen dos condiciones. Primero, si el tiempo en que la entrada de alarma estaba activada por un intervalo de tiempo menor a un cierto valor establecido por el usuario en el formulario principal. Segundo, si el valor de la variable analógica que produjo la alarma se encuentra dentro de un rango también establecido por el usuario.

Con el botón de comando *Forzar_CSTR* , es posible apagar las salidas Q0.4 y Q0.5 y también el estado de alarma independientemente de las condiciones del párrafo anterior.

Para salir del formulario, primero se hace un clic sobre el botón de comando *Terminar_SCADA* y luego sobre el botón etiquetado *Salir.*

3.5.6 Formulario: MAGLEV

En este formulario se muestran las tres entradas de alarma, el estado de alarma, la posición vertical y el bloque de reconocimiento de las alarmas.

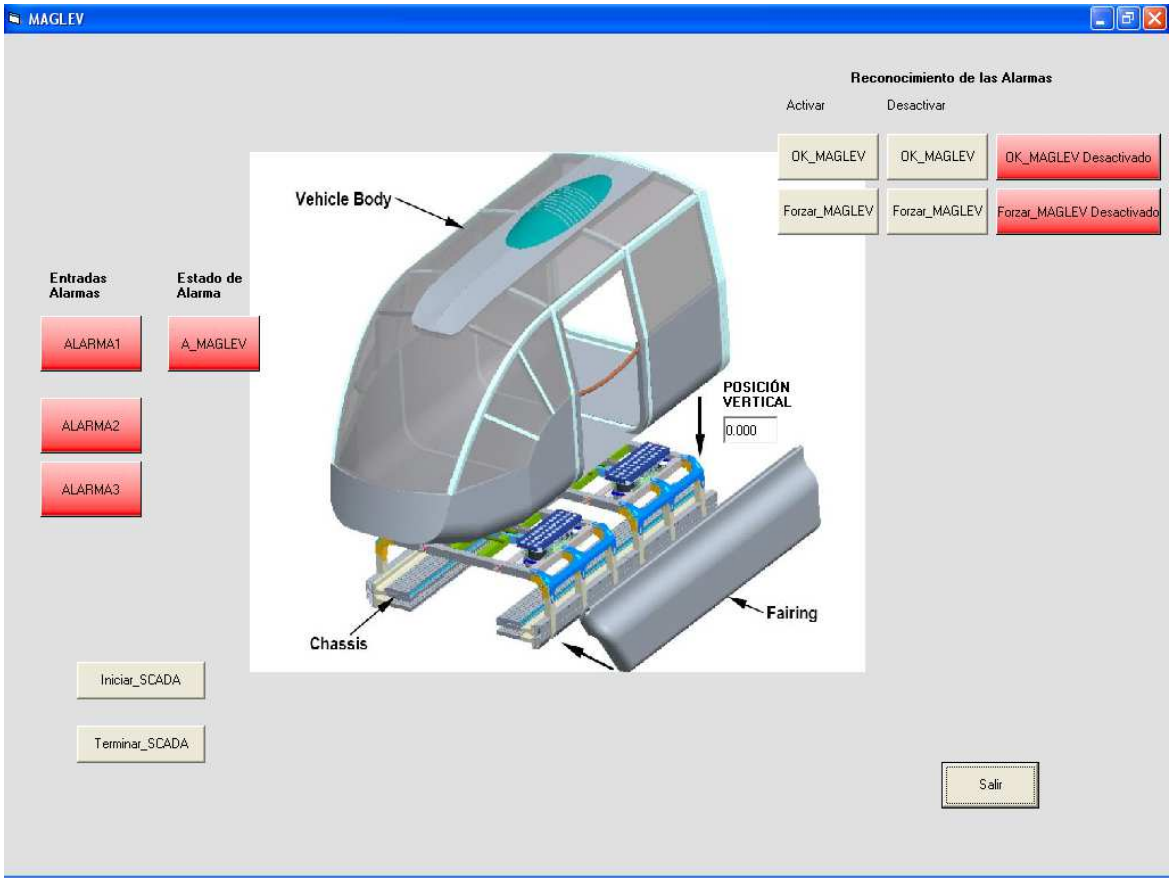


Figura 3.10. Formulario MAGLEV en tiempo de ejecución

Los botones etiquetados como ALARMA1, ALARMA2 y ALARMA3 muestran los estados de las entradas I0.6, I0.7 y I1.0 del PLC correspondientes al MAGLEV. Dichos botones, tienen un color rojo cuando las entradas están desactivadas y un color verde cuando están activadas. El programa implementado en el PLC, permite que se activen dos salidas Q0.6 y Q0.7 para la alarma sonora y visual, cuando alguna de ellas se active.

Un botón etiquetado como A_MAGLEV, cambia de rojo a verde tan pronto como se active una entrada de alarma. Mantiene en dicho estado (en verde) independientemente del estado de las entradas de alarma. El usuario puede realizar el reconocimiento de la alarma mediante dos maneras.

Los botones de comando OK_MAGLEV y Forzar_MAGLEV que están debajo de la etiqueta *Activar* son para activar el reconocimiento de las alarmas. Con un clic sobre OK_MAGLEV se desactivan las salidas Q0.6 y Q0.7 del PLC, y se desactiva el estado de alarma si el tiempo en que la entrada que produjo la alarma estaba activada por un intervalo de tiempo menor a un cierto valor, el cual es determinado por el usuario el formulario Principal. Con un clic sobre Forzar_MAGLEV, también es posible desactivar las salidas, pero es más utilizado para apagar el estado de alarma cuando no es posible desactivar mediante OK_MAGLEV, o sea cuando la entrada de alarma estuvo activada por un lapso de tiempo mayor a un cierto valor.

Los botones de comando OK_MAGLEV y Forzar_MAGLEV que están debajo de la etiqueta *Desactivar* son utilizados para desactivar el reconocimiento de alarmas.

Los dos botones restantes del bloque de reconocimiento de las alarmas, solo muestran si está o no activada el reconocimiento de las alarmas.

Finalmente se tiene un editor de control que muestra un número real entre 0-1 para representar la posición vertical del MAGLEV. Dicho rango puede ser modificado fácilmente a pedido del usuario en el programa del PLC o directamente en Visual Basic.

Para salir del formulario, primero se hace un clic sobre el botón de comando *Terminar_SCADA* y luego sobre el botón etiquetado *Salir*

4. Conclusiones

- El presente proyecto Sistemas de Control, Supervisión y Adquisición de datos de las plantas del laboratorio de Control de la USFQ, cumple con los objetivos planteados en el perfil de tesis. El Sistema SCADA desarrollado permite la supervisión y control remoto de las instalaciones: DOMÓTICA aplicada a un recinto, prototipo de ASCENSOR, tanque continuamente agitado presurizado (CSTR) y transporte basado en levitación electromagnética (MAGLEV).
- La capacidad del PLC S7-200 para incorporar tanto módulos de entradas y/o salidas digitales como analógicos, y su eficiencia en la solución de tareas de automatización sencillas hace que sea adecuado para el presente proyecto.
- La programación del PLC S7-200 fue realizado en el lenguaje de contactos KOP. Esté lenguaje es bastante amigable para la programación, aún para programadores que no tienen mucha experiencia en la materia.
- Las Redes de Petri demostraron ser una herramienta efectiva para la construcción sistemática de sistemas a eventos discretos, como por ejemplo el tratamiento de las alarmas.
- Se mostró que es posible construir un Sistema SCADA de bajo costo utilizando software convencionales como Visual Basic. Con la ventaja que permite mejoras a futuro y con la desventaja de la limitada velocidad de respuesta dada por la CPU de computador.
- La nomenclatura utilizada para representar botones, botones de comando, editores de control están asociadas con el nombre de la variable física a la

cual representa. De esta manera, se requiere solo un adiestramiento básico para entender y hacer uso de las pantallas de la interfaz gráfica.

- El Sistema SCADA desarrollado permite realizar modificaciones a futuro. Por ejemplo se pueden añadir nuevas plantas para el control, la supervisión y adquisición de datos de la misma.
- El Sistema SCADA genera un archivo de texto que permite conocer el estado de las entradas y salidas del PLC, como también de ciertas variables de las plantas involucradas, como parámetros de los PIDs.

Bibliografía

[1] [1]Bailey, David y Wrigth, Edwin, Practical SCADA for Industry.

[2] [5] Bustos Farias, Eduardo. INSTITUTO POLITÉCNICO NACIONAL. *SIMULACIÓN.* Obtenido en línea el 2 de marzo del 2009. Disponible en: http://www.angelfire.com/ak6/ilb/5_1.pdf

[3] García, Javier. Universidad de Navarra. Aprenda Visual Basic6 como si estuviera en primero. San Sebastian, agosto 1999. Obtenido en línea el 23 de marzo del 2009. Disponible en: http://mat21.etsii.upm.es/ayudainf/aprendainf/VisualBasic6/vbasic60.pdf

[4] [4]Jiménez Macías, Emilio. *Técnicas de automatización avanzadas en procesos industriales*. Obtenido en línea el 5 de marzo del 2009. Disponible en: http://www.google.com.ec/search?q=emilio+jimenez+macias

[5] Microsoft Corporation. Microsoft Visual Basic 6.0. *Aprendiendo Visual Basic en 21 Dias*. EditorialPrentice Hall.

[6] Siemens. *SIMATIC MicroComputing*. User Manual. Edición 08/2001. A5E00077133-02

[7] Siemens. *SIMATIC Sistemas de automatización S7-200*. Manual del sistema. Tercera Edición 2002. 6ES7 298-8FA22-8DH0.

[8] [2]*SISTEMAS SCADA*, Obtenido en línea el 10 de febrero del 2009. Disponible: en http://www.alfinal.com/Temas/sistemascada.shtml

[9] [3]Tolosa, Gabriel. *Protocolos y modelo OSI.* Obtenido en línea el 15 de febrero del 2009. Disponible en: http://www.tyr.unlu.edu.ar/cms/files/02-Protocolos%20y%20OSI_0.pdf

[10] [6]UNIVERSIDAD NACIONAL DE COLOMBIA,2007. Zapata, Germán.*DISEÑO DE AUTOMATISMOS SECUENCIALES PARA CONTROLADORE LÓGICOS PROGRAMABLES*

[11] Vazquez, Carlos. *Diseño y construcción de un laboratorio virtual utilizando Redes de Petri*.Toluca, México, Diciembre 2002.

Anexo1. Tablas de las entradas y salidas del PLC.

Tabla1. Descripción de las variables digitales de entrada del sistema

VARIABLE	DIRECCIÓN EN MEMORIA	DESCRIPCIÓN	UBICACIÓN
LSHH	I0.1	Switch de nivel alto (CSTR)	M0.1
LSLL	I0.2	Switch de nivel bajo (CSTR)	M0.2
PSHH	I0.3	Switch de sobre presión (CSTR)	M0.3
TSHH	I0.4	Switch de sobre temperatura (CSTR)	M0.4
A1	I0.6	Alarma1 del MAGLEV	M0.6
A2	I0.7	Alarma2 del MAGLEV	M0.7
A3	I1.0	Alarma3 del MAGLEV	M1.0
SPC	I1.2	Switch Puerta Cerrada(DOMOTICA)	M1.2
SGC	I1.3	Switch Garaje Cerrado(DOMOTICA)	M1.3
SCC	I1.4	Switch Cortina Cerrada(DOMOTICA)	M1.4
SVC	I1.5	SwitchVentana Cerrada(DOMOTICA)	M1.5
SPA	I1.6	Switch Puerta Abierta(DOMOTICA)	M1.6
SGA	I1.7	Switch Garaje Abierto(DOMOTICA)	M1.7
SCA	I2.0	Switch Cortina Abierta(DOMOTICA)	M2.0
SVA	I2.1	Switch Ventana Abierta(DOMOTICA)	M2.1
AH	I2.2	Alarma de Humo(DOMOTICA)	M2.2
AS	I2.3	Alarma de Seguridad(DOMOTICA)	M2.3
LE	I2.4	Luces Exteriores(DOMOTICA)	M2.4
SPUERTA	I2.6	Switch puerta de cabina(ASC.)	M2.6
SPESO	I2.7	Switch de sobrepeso (ASC.)	M2.7
PEP1	I3.0	Pulsador exterior piso1 (ASC.)	M3.0
PEP2	I3.1	Pulsador exterior piso2 (ASC.)	M3.1
PEP3	I3.2	Pulsador exterior piso3 (ASC.)	M3.2
PE	I3.3	Paro de emergencies (ASC.)	M3.3
PCP1	I3.4	Pulsador cabina piso1 (ASC.)	M3.4
PCP2	I3.5	Pulsador cabina piso2 (ASC.)	M3.5
PCP3	I3.6	Pulsador cabina piso3 (ASC.)	M3.6
SAP1	I3.7	Switch A piso1 (ASC.)	M3.7
SBP1	I4.0	Switch B piso1 (ASC.)	M4.0
SAP2	I4.1	Switch A piso2 (ASC.)	M4.1
SBP2	I4.2	Switch B piso2 (ASC.)	M4.2
SAP3	I4.3	Switch A piso3 (ASC.)	M4.3
SBP3	I4.4	Switch B piso3 (ASC.)	M4.4
FCS.	I4.5	Final de carrera superior (ASC.)	M4.5
FCI.	I4.6	Final de carrera inferior (ASC.)	M4.6
SPCAIDAS	I4.7	Sensor paracaídas (ASC.)	M4.7

Tabla2. Descripción de las variables analógicas de entradas del sistema

VARIABLE	DIRECCIÓN EN MEMORIA	DESCRIPCIÓN	UBICACIÓN
PRESIÓN	AIW0	CSTR	VW16
TEMPERATURA	AIW2	CSTR	VW18
FLUJO	AIW4	CSTR	VW20
NIVEL	AIW6	CSTR	VW22
SIN ASIGNAR	AIW8	MAGLEV	VW24
SIN ASIGNAR	AIW10	MAGLEV	VW26
SIN ASIGNA	AIW12	DOMÓTICA	VW28
VENTILACION	AIW14	DOMOTICA	VW30

Tabla3. Descripción de las variables digitales de salida del sistema

VARIABLE	DIRECCIÓN EN MEMORIA	DESCRIPCIÓN
Salida rápida1	Q0.0	MAGLEV
Salida rápida2	Q0.1	MAGLEV
Sentido giro válvula1	Q0.2	CSTR
Sentido giro válvula2	Q0.3	CSTR
Alarma sonora	Q0.4	CSTR
Alarma visual	Q0.5	CSTR
Alarma sonora	Q0.6	MAGLEV
Alarma visual	Q0.7	MAGLEV
Apertura/Cierre Puerta	Q1.1	DOMÓTICA
Apertura/Cierre garaje	Q1.2	DOMÓTICA
Apertura/Cierre cortina	Q1.3	DOMÓTICA
Apertura/Cierre ventana	Q1.4	DOMÓTICA
Encendido/Apagado Alarma Sonora	Q1.5	DOMÓTICA
Encendido/Apagado Alarma Visual incen	Q1.6	DOMÓTICA
Encendido/Apagado Alarma Sonora Segur.	Q1.7	DOMÓTICA
Encendido/Apagado Alarma Visual Segur.	Q2.0	DOMÓTICA
Encendido/Apagado Luces	Q2.1	DOMÓTICA
Encendido/Apagado Ventilador	Q2.2	DOMÓTICA
Alarma sonora	Q2.4	ASCENSOR
Alarma visual	Q2.5	ASCENSOR
Sin asignar	Q2.6	ASCENSOR
Sin asignar	Q2.7	ASCENSOR

Tabla4. Descripción de las variables analógicas de salidas del sistema

VARIABLE	DIRECCIÓN EN MEMORIA	DESCRIPCIÓN
Sin asignar	AQW0	CSTR
Sin asignar	AQW2	CSTR
Luces interiores	AQW4	DOMOTICA

Anexo2. descripción de los botones de comando utilizados en la interfaz gráfica.

Valor en MB5	Nombre del Botón/ formulario	Descripción
1	cmdActOK_AH(DOMOTICA)	Activa la variable OK_AH
2	cmdDActOK_AH(DOMOTICA)	Desactiva la variable OK_AH
3	cmdActForzar_AH(DOMOTICA)	Activa la variable Forzar_AH
4	cmdDActForzar_AH(DOMOTICA)	Desactiva la variable Forzar_AH
5	cmdActOK_AS(DOMOTICA)	Activa la variable OK_AS
6	cmdDActOK_AS(DOMOTICA)	Desactiva la variable OK_AS
7	cmdActForzar_AS(DOMOTICA)	Activa la variable Forzar_AS
8	cmdDActForzar_AS (DOMOTICA)	Desactiva la variable Forzar_AS
10	cmdIPCP1(ASCENSOR)	Activa la variable PCP1
11	cmdTPCP1(ASCENSOR)	Desactiva la variable PCP1
12	cmdIPCP2(ASCENSOR)	Activa la variable PCP2
13	cmdTPCP2(ASCENSOR)	Desactiva la variable PCP2
14	cmdIPCP3(ASCENSOR)	Activa la variable PCP3
15	cmdTPCP3(ASCENSOR)	Desactiva la variable PCP3
16	cmdIPEP1(ASCENSOR)	Activa la variable PEP1
17	cmdTPEP1(ASCENSOR)	Desactiva la variable PEP1
18	cmdIPEP2(ASCENSOR)	Activa la variable PEP2
19	cmdTPEP2(ASCENSOR)	Desactiva la variable PEP2
20	cmdIPEP3(ASCENSOR)	Activa la variable PEP3
21	cmdTPEP3(ASCENSOR)	Desactiva la variable PEP3
22	cmdIPE (ASCENSOR)	Activa la variable PE
23	cmdcmdTPE (ASCENSOR)	Desactiva la variable PE
31	cmdActOK_PCAIDA (ASCENSOR)	Activa la variable OK_PCAIDA
32	cmdDActOK_PCAIDA (ASCENSOR)	Desactiva la variable OK_PCAIDA
33	cmdActForzar_PCAIDA (ASCENSOR)	Activa la variable Forzar_PCAIDA
34	cmdDActForzar_PCAIDA (ASCENSOR)	Desactiva la variable Forzar_PCAIDA
41	cmdActOK_CSTR (CSTR1)	Activa la variable OK_CSTR
42	cmdDActOK_CSTR (CSTR1)	Desactiva la variable OK_CSTR
43	cmdActForzar_CSTR (CSTR1)	Activa la variable Forzar_CSTR
44	cmdDActForzar_CSTR (CSTR1)	Desactiva la variable Forzar_CSTR

51	cmdActOK_MAGLEV (MAGLEV)	Activa la variable OK_ MAGLEV
52	cmdDActOK_ MAGLEV (MAGLEV)	Desactiva la variable OK_ MAGLEV
53	cmdActForzar_ MAGLEV (MAGLEV)	Activa la variable Forzar_ MAGLEV
54	cmdDActForzar_ MAGLEV(MAGLEV)	Desactiva variable Forzar_ MAGLEV