INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY
CAMPUS ESTADO DE MÉXICO

SISTEMA PARA LA IMPLANTACIÓN DEL PROTOCOLO PROFIBUS DE REDES DE COMUNICACIÓN INDUSTRIALES PARA LA AUTOMATIZACIÓN DE SISTEMAS DE MANUFACTURA

TESIS QUE PARA OPTAR EL GRADO DE
MAESTRO EN CIENCIAS EN SISTEMAS DE MANUFACTURA
PRESENTA

ING. EDUARDO CALDERÓN PORRAS

Asesor:	Dr. ALEJANDRO VEGA SALINAS	
Comité de Tesis:	M.C. DAVID OLIVA URIBE	
	M.C. GUILLERMO SANDOVAL BENITEZ	
Jurado:	M.C. GUILLERMO SANDOVAL BENITEZ	Presidente
	M.C. DAVID OLIVA URIBE	Secretario
	Dr. ALEJANDRO VEGA SALINAS	Vocal

Atizapán de Zaragoza, Edo. De México., Junio de 2001

A Dios por estar conmigo en todos
los momentos de mi vida.

A mis padres por su gran apoyo
Incondicional.

RECONOCIMIENTOS

A La Universidad Autónoma de Bucaramanga por brindarme la beca para realizar esta maestría y por su gran apoyo en el transcurso de la misma.

Al Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México por darme la oportunidad de realizar la maestría.

Al Dr. Alejandro Vega Salinas por dirigir este proyecto y por su gran ayuda y apoyo en todo momento.

Al Dr. Emil Liebermann por todo su apoyo.

RESUMEN

El presente proyecto consiste en desarrollar un sistema para la implantación del protocolo PROFIBUS de redes de comunicación industriales para la automatización de sistemas de manufactura. Estas redes son implementadas con una metodología maestro esclavo, en donde los maestros son los administradores de la red y los esclavos realizan actividades de automatización y control y están siendo supervisados por su respectivo maestro. Para el sistema maestro se desarrolla un software en Visual Basic 6.0 y Access 2000. Las estaciones esclavas son implementadas con microcontroladores Intel 8051.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

1 INTRODUCCIÓN

La industria de las computadoras personales (PC) ha evolucionado rápidamente y, con ello, ha brindado poder de procesamiento de alto rendimiento, memoria y almacenamiento rápidos y baratos, E/S de alta velocidad y despliegues vívidos; todo junto en un paquete listo para usarse y a bajos costos.

En los pasados 25 años, el PC ha revolucionado la forma como las compañías hacen negocios, automatizan sistemas e incrementan productividad mientras reducen costos. En estos momentos las redes de comunicación están adicionando una nueva dimensión al PC, permitiendo a las compañías hacer negocios sobre una escala global. Con las tecnologías de redes, se puede ver remotamente y controlar un sistema que es localizado físicamente en otro sitio, o se puede monitorear el piso de producción desde la oficina [14].

Hacer la conexión física entre computadores usando tarjetas de interfaz y cables no asegura poder compartir información entres los computadores de la red. Para realizar este trabajo, las redes deben tener un juego de reglas o normas bien definidas, a estas normas se les conoce como protocolos, los cuales permiten el intercambio de información entre dos o más dispositivos. Los protocolos se encargan de especificar el formato del mensaje y las reglas necesarias para recibir e interpretar la información que se esta intercambiando. Un protocolo desarrolla muchas funciones, entre estas están: definir una dirección única para cada equipo conectado a la red, determinar como son transmitidos los datos entres los dispositivos y procesar la información cuando es recibida. Algunos protocolos usados para propósito general son: TCP/IP, NetBEUI, UDP,

AppleTalk, SNMP y LAT. Para redes industriales se encuentran: DeviceNet, ControlNet, PROFIBUS, Modbus y FOUNDATION Fieldbus [15].

1.1 REDES DE COMUNICACIÓN DE DATOS

En la década de los 70's hubo una revolución en la aplicación de dispositivos semiconductores, como microprocesadores y memorias. El costo de estos dispositivo ha descendido con gran rapidez, a su vez su poder y complejidad se han elevado en un grado similar. Como resultado, los sistemas construidos con este tipo de dispositivos han encontrado aplicación en todas las actividades de la vida y continúan haciéndolo cada vez más. Un sistema en particular, el computador, es una herramienta esencial en muchas industrias, negocios, universidades y otros lugares de trabajo. Debido a la creciente cantidad de computadores, se ha llegado a la necesidad de la comunicación entre ellos para el intercambio de datos, programas, mensajes, comunicación entre usuarios de la red y sobre todo para poder compartir entres sí los diferentes dispositivos periféricos (impresoras, escáner, dispositivos de almacenamiento, etc...). Las redes de computación llegaron para llenar esa necesidad, proporcionando caminos de comunicación entre los computadores enlazados a estas redes, [1,2].

Hay varias formas de intercomunicar las computadoras, esto depende de la cantidad de equipos (cualquier dispositivo basado en microprocesador) a comunicar y de las distancias que se desean cubrir. Por ejemplo si se desea comunicar dos computadoras es suficiente con establecer un enlace punto a punto entre estas dos computadoras, más sin embargo si se desea comunicar un número mayor de computadoras, se debe pensar en otra estrategia de intercomunicación ya que sería impractico tratar de conectar cada computadora con todas las restantes. Para resolver este problema se han desarrollados estándares y protocolos (conjunto de normas para poder controlar la comunicación entres los equipos de la red) para poder construir de una manera ordenada y practica una red de comunicación de datos.

Los protocolos básicamente se encargan de definir las siguientes características:

1. La forma física de enlace entre los diferentes equipos (cable, conectores, interfaces, etc..).

2. El lenguaje de comunicación.
3. La identificación de cada equipo de la red (Ésta identificación debe ser única).
4. Controlar el encaminamiento de cada mensaje enviado para que llegue al destinatario correcto.
5. Controlar que los mensajes se enviados y recibidos correctamente (control de errores).
6. Controlar de una manera ordenada el acceso a la red.
7. Iniciación, sincronización y reestablecimiento de la red.

Entre los protocolos más conocidos tenemos TCP/IP que se esta usando ampliamente sobre todo en Internet, TOP, Ethernet, PROFIBUS para redes industriales, MODBUS, entre otros.

Las redes de comunicación se pueden clasificar teniendo en cuanta las distancias a las que se encuentras los diferentes equipos a interconectar. Si los equipos a intercomunicar se encuentran en un mismo edificio se dice que son redes de área loca (LAN), cuando los equipos se encuentran en sitios fuera de la compañía son redes de área amplia (WAN). Puede existir una comunicación o enlace entre las LAN y las WAN, ya que se pueden comunicar dos edificios que tengan un LAN y establecer un WAN.

Las LAN son las más utilizadas en las compañías tanto en automatización de oficinas como en automatización de sistemas de manufactura. En este proyecto nos ocuparemos de las LAN ya que son las que se utilizan en las redes industriales.

1.2 REDES INDUSTRIALES, FIELDBUS

Las LAN inicialmente fueron pensadas para redes de computadores, pero con los avances tecnológicos en el área de automatización y control, se fueron incorporando dispositivos basados en microprocesador en estas áreas, posteriormente se fue viendo la necesidad de intercomunicar estos dispositivos de control entre sí y/o con computadoras centrales que administren el sistema total, esto para poder intercomunicar el área de producción de la compañía con las áreas administrativas, llegando a tener una empresa totalmente automatizada.

En los procesos de automatización y control industrial se utilizan diversos dispositivos tales como sensores, actuadores, PLC´s, microcontroladores, multiplexores, controladores PID, etc., encargados de la automatización y control de la producción. Estos dispositivos normalmente se encuentran separados y con cableados diferentes formando un conjunto de islas que controlan distintas áreas de la producción, esto lleva a una gran cantidad de cableado innecesario y a una difícil administración de todo el sistema de automatización, ya que cada vez que se quiere hacer una modificación, es necesario volver a cablear y a reconfigurar todos los dispositivos. Por esta razón muchas veces es necesario implementar una red industrial de comunicación en el área de automatización, con estas redes se logra una mejor administración, confiabilidad, reducción de costos y sobre todo poder compartir recursos de los diferentes dispositivos del sistema, [4,5,8].

Por definición, una red industrial consiste en la distribución geográfica de instrumentos de medida, actuadores y dispositivos de control que interactúan entres si para llevar a cabo una actividad de automatización y control, [12]. Normalmente a las redes industriales se les conoce como buses de campo o Fieldbus que es como se designará en este proyecto.

Fieldbus surgió por un trabajo de estandarización iniciado por la ISA (Instrumentations Society of America), [13].

La forma de comunicación de las redes industriales es digital, transfieren bits de información serialmente.

Fieldbus presenta muchas ventajas para la implantación de redes de automatización y control, algunas de esas ventajas son, [9,10,11,13]:

1. Interconectividad, ya que se pueden conectar equipos de diferentes fabricantes.
2. Gran reducción de cableado, ya que todos los equipos se conectan al mismo cable.
3. Más fácil la administración del sistema global en cuanto a cableado y software. Al ser comunicación digital se puede utilizar la capacidad de las computadoras personales para el manejo y control de información.
4. Más fácil adicionar o quitar dispositivos de la red.
5. Se pueden integrar al sistema dispositivos de control, alarmas, supervisión, visualización, administración, y en general cualquier dispositivo basado en microprocesador.
6. Conexión de dispositivos inteligente que ayudan a desarrollar actividades avanzadas de diagnostico.
7. Control distribuido. Entre otras.

Al igual que en las redes de computadores, en las redes industriales también se ha trabajado en la creación de protocolos de comunicación. Se han establecido diversos protocolos muy parecidos entre sí, aunque con pequeñas diferencias, [1,4,5,6], entre estos están: BITBUS de Intel, FIP de AFNOR, MIL-STD-1535 de ANSI, MODBUS de GOULD INC, Interbus-S de Phoenix, DeviceNet de Allen-Bradley, Ethernet de Intel y Xerox, PROFIBUS de DIN que es en el que nos centraremos en esta tesis ya que se esta teniendo una aceptación muy grande en todo el mundo y hay muchas empresas comprometidas con este producto como Siemens, Bosch, Klockner Moeller, ABB, AEG, Landis&Gir y algunas universidades alemanas, [9].

1.3 PROFIBUS

PROFIBUS (Process fieldbus) es un bus (red, en donde los dispositivos conectados utilizando el mismo medio físico de transmisión) de comunicación que implementa una metodología maestro / esclavo, [9], en donde los maestros son dispositivos con buena capacidad de procesamiento que controlan el acceso al medio de comunicación (hay que tener en cuenta que solo un dispositivo puede usar el medio de comunicación en un momento determinado, ya que todos se comunican por el mismo medio) y tienen a su cargo varios esclavos, que son los dispositivos que están realizando tareas de control (controladores, sensores, actuadores, etc...). El maestro viene realizando las actividades de supervisión y control de sus esclavos, comunicándose con estos periódicamente de la manera establecida por el protocolo, la figura 1 muestra esta estructura, en donde cada maestro tiene asignados unos esclavos así: el maestro 1 tiene el esclavo A, el maestro 2 tiene los esclavos B y C, el maestro 3 tiene los esclavos D y E, y todos se están comunicando por el mismo medio.

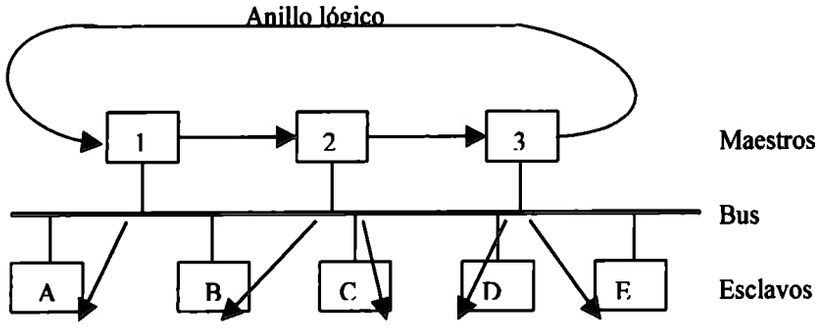


Figura 1. Esquema de arquitectura maestro / esclavo

PROFIBUS trabaja con una metodología orientada a objetos que facilita la integración de diferentes dispositivos en una misma red, [9,13]. En esta metodología, se maneja dispositivos virtuales, en donde cada dispositivo conectado a la red maneja un número determinado de variables dependiendo de la tarea de control que este realizando (estas variables podrían ser corriente, voltaje, peso, presión, etc...). El maestro implementa un software que trabaja con una base de datos en la cual tiene almacenada todas de las variables que maneja cada uno de sus esclavos. Para cada esclavo el maestro crea un dispositivo virtual que es con el que interactúa el usuario del sistema, sin necesidad de tener que modificar directamente el esclavo. Cada vez que el usuario quiere conocer o modificar alguna de las variables que maneja el dispositivo de control (esclavo), el usuario hace los cambios en el maestro y el maestro por medio de la red actualiza el esclavo. También cuenta con características que ayudan a la adición y eliminación a la red de nuevos dispositivos.

PROFIBUS esta definido por varios niveles que ayudan a interconectar las diferentes áreas de la compañía. Estos niveles son: PROFIBUS-DP que establece la comunicación entre maestros y esclavos, PROFIBUS-FMS que establece la comunicación entre maestros y PROFIBUS-PA que se utilizado en el área de automatización de procesos, [9].
PROFIBUS, además cuenta con todas las características de fieldbus mencionadas anteriormente, también vale la pena mencionar que el medio de comunicación física se basa en la norma RS-485

.

1.4 SISTEMAS DE ADMINISTRACIÓN DE REDES

En las redes es muy importante contar con un sistema o software que administre la red y los diversos dispositivos conectados a ésta. El software de red hace que la red sea totalmente transparente para los usuarios. Los software de red son llamados normalmente sistemas operativos de red. Estos sistemas tiene a cargo diversas actividades como:

1. Presentar al usuario de una manera lógica y ordenada el funcionamiento de la red.
2. Configuración de los parámetros de la red.

3. Ayuda a la comunicación con otros equipos de la red.
4. Transferencia de archivos e información entre los diferentes dispositivos.
5. Detección de fallas.
6. Manejo de bases de datos.
7. Enlazar con otro tipo de redes.
8. Configuración remota de dispositivos.
9. Generación de reportes de funcionamiento de la red.

Algunos ejemplos de estos sistemas para redes de computadores son: Windows NT, NetWare, Unix, LANtastic, entre otros.

1.5 PLANTEAMIENTO DEL PROBLEMA.

Gracias a los sistemas basados en microprocesadores, la automatización industrial ha tenido en los últimos tiempos un crecimiento nunca antes visto, muy de la mano de estos sistemas se encuentran las técnicas de intercomunicación entre ellos y con diferentes tipos de dispositivos en el área de control como son los sensores y actuadores, [1].

Como se menciono en el apartado anterior, actualmente existen varios protocolos para fieldbus que establecen una metodología para la intercomunicación de los diversos dispositivos utilizados en el área de automatización y control.

La idea de este proyecto es entrar en un estudio detallado del protocolo PROFIBUS y poder implantarlo en la automatización de sistemas de manufactura.
El proyecto consiste en el desarrollo de un sistema maestro / esclavo basado en PROFIBUS, ver figura 1. El maestro será un sistema basado en computadora personal hecho con el lenguaje de programación Visual Basic 6.0 bajo Windows de Microsoft, los maestros serán los administradores de la red, cada maestro tiene las siguientes características:

1. Tendrá a su cargo esclavos que son los que realizan las actividades de automatización y control.

2. Es responsable de la iniciación y configuración de la red
3. Designará quien puede utilizar el medio físico en un momento determinado.
4. Pasar el token o derecho a administrar la red al maestro siguiente.
5. Estar pendiente de la recepción del token del maestro antecesor.
6. Controlará la transmisión de datos (inicio, terminación, control)
7. Trabajará con dispositivos virtuales, que consiste en la representación virtual en el sistema de cada uno de los esclavos, en donde los usuarios podrán leer y modificar los parámetros de estos sin necesidad de acudir directamente al esclavo.
8. Contará con una base de datos hecha con Access 2000 de Microsoft, esta base de datos servirá de apoyo para la administración y control del sistema, tendrá información de: esclavos (índice, nombre, tipo de control, variables u objetos que maneja, tipos de datos de las variables, longitud de las variables, fabricante, parámetros de funcionamiento, etc...), también tendrá información de timers, errores, servicios y configuración.
9. Capacidad para detectar la adición o eliminación de un nuevo dispositivo a la red. Los esclavos tendrán drivers que son archivos que contienen toda la información referente al esclavo, cada vez que se adiciona un esclavo a la red, el maestro lee estos drivers y actualiza la base de datos.
10. Detectar que no haya dispositivos en la red con el mismo índice o dirección (Cada dispositivo tendrá un índice único que lo identificará en la red)
11. Generación del frame (mensaje para la comunicación que utiliza el protocolo, este mensaje tiene varios campos que son: delimitador de inicio, dirección destino, dirección origen, campo de control, datos, campo de verificación y delimitador de final).
12. Interpretación de los frames recibidos.
13. Detección de fallas de transmisión.
14. Detectar la ausencia de alguno de sus esclavos.
15. Detección de mal funcionamiento de algún esclavo.
16. Generación de reportes del funcionamiento de la red, de los esclavos, de los parámetros, etc...
17. Presentación visual de la información.
18. Ambiente amigable para usuarios inexpertos.
19. entre otras

Los esclavos, que son los dispositivos que realizan las actividades de automatización y control, en este caso serán microcontroladores Intel 8051. Para el microcontrolador se desarrollara un programa hecho en lenguaje ensamblador, este programa se encargara de interpretar el frame enviado por el maestro y de desarrollar actividades propias de automatización y control. Al recibir el frame enviado por el maestro, tomara las siguientes acciones:

1. Detectar si es un frame valido
2. Detectar si el frame va dirigido a él (leyendo la dirección destino).
3. Verificar que los datos se los esta pidiendo su propio maestro (verificando el campo de dirección de origen).
4. Detección de fallas en el frame.
5. Leer los datos solicitados por el maestro, estos datos pueden ser las variables o el estado del esclavo.
6. Generar y enviar el frame de respuesta con los datos solicitados por el maestro.
7. Activar alarmas de inactividad de su maestro.
8. Almacenar los valores de las variables que maneja, para poderlas comunicar a su maestro.
9. Desarrollar la actividad de control correspondiente
10. Entre otras.

En el proyecto también se definirán todos los parámetros relacionados con la implantación de la red, como son: cable, conectores, interfaz entre computador personal y el medio de comunicación usado por PROFIBUS que es RS-485. Se desarrollara de circuito electrónico para el funcionamiento del microcontrolador 8051 y la interfaz con RS-485.

1.6 JUSTIFICACIÓN

Una empresa para ser eficiente requiere de bases de datos y herramientas de planeación de requerimientos en el área de manufactura, hacer redes de comunicación es una característica crucial en las plantas modernas [14].

Las redes industriales remplazan los cableados convencionales punto a punto. Las redes industriales presentan muchos beneficios, entre estos están:

- Reducir cableado.
- Trabaja con dispositivos inteligentes.
- Control distribuido.
- Fácil mantenimiento.
- Bajan costos de instalación y mantenimiento.

Las redes industriales ofrecen la capacidad de encontrar las necesidades de expansión de las operaciones de manufactura de todos los tamaños [14].

La importancia de este proyecto radica en la tendencia que se esta viendo hacia la forma de hacer negocios. Las empresas de hoy en día deben ser ágiles y estar preparadas para afrontar los cambios del mercado, para poder realizar esto se debe contar con un sistema flexible de manufactura. Un sistema flexible de manufactura sería imposible si un sistema de comunicación de datos que intercomunique todas las áreas de la compañía.

Las redes de computo ya esta bastante avanzadas y actualmente es muy sencillo implementarlas, sin embargo el problema empieza cuando se quiere enlazar el piso de producción con el área administrativa de la empresa. Aunque existen muchos protocolos para redes industriales todavía existe cierto desconocimiento por parte de los ingenieros de manufactura en cuánto al uso y alcance de estos protocolos.

La presente tesis será de gran ayuda para tener un conocimiento de las redes de comunicación industrial en especial de PROFIBUS. Y será una herramienta de gran ayuda para la implementación de pequeños sistemas de automatización.

1.7 OBJETIVO GENERAL

El objetivo de esta tesis es el desarrollo de un sistema de comunicación maestro / esclavos, implementando el protocolo de comunicación redes industriales PROFIBUS, este sistema consta de un software para el maestro desarrollado con el lenguaje de programación Visual Basic 6.0 y un sistema para los esclavos basado en el microcontrolador Intel 8051.

1.7.1 OBJETIVOS ESPECIFICOS

1. Estudio de factibilidad
2. Estudio de los conceptos de redes de comunicación industriales (FieldBus).
3. Estudio del protocolo PROFIBUS.
4. Revisión de microcontroladores 8051.
5. Desarrollo del circuito electrónico para implementar el microcontrolador 8051 con PROFIBUS.
6. Análisis y diseño de los sistemas maestro y esclavo.
7. Desarrollo de Software con Visual Basic 6.0 para el sistema maestro y desarrollo de la base de datos en Access 2000.
8. Desarrollo de programa en lenguaje ensamblador para el microcontrolador 8051.
9. Implantación del protocolo PROFIBUS con el PC y con los microcontroladores 8051.
10. Pruebas del sistema.
11. Elaboración de trabajo escrito.

2 REDES DE ÁREA LOCAL (LAN)

A comienzos de los 80's el mainframe (Computadora con grandes capacidades de manejo de información) fue la pieza central de un centro de computo. Las terminales y varios tipos de procesos distribuidos fueron considerados como dispositivos con menor importancia en el sistema de computo corporativo. A finales de los 80's hubo un cambio radical en la percepción de este concepto. Los investigadores en las empresas empezaron a trabajar de manera diferente, cada trabajador contaba con una potente computadora personal y cada uno manejaba sus propias bases de datos. Después ellos fueron necesitando más servicios para soportar sus actividades de computo, entre estos servicios están: bases de datos centrales, bases de datos de cada departamento, impresoras, plotters, correo electrónico, accesos a otros computadores, acceso a grandes maquinas para procesamiento numérico, etc. [3]

Las redes de área local han cambiado la forma de trabajar con sistemas de computo, se paso del trabajo de una manera individual a un ambiente rico en recursos, en donde todos pueden utilizar la información y dispositivos existentes en la empresa.

La necesidad de comunicarse, enviar mensajes, compartir datos, acceder recursos de computo y compartir dispositivos periféricos, ha contribuido al desarrollo y crecimiento de las redes de área local (LAN). Una LAN es una red de computadores conectados en un área geográfica pequeña. Típicamente, una LAN conecta a usuarios localizados en la misma oficina, en el mismo piso o en el mismo edificio. El desafío para las redes de área local ha sido proveer a un costo razonable los servicios necesarios por los usuarios de la red. La compatibilidad es la principal variable para reducir costos, ya que no se esta ligado a una marca especifica, las LAN deben tener la capacidad

de conectar una variedad de productos hardware y software en la misma red, sin tener que construir complejas y costosas interfaces para cada producto. Para solucionar este problema varias organizaciones internacionales de estándares y vendedores de hardware y software han cooperado en el desarrollo de arquitecturas para redes de computadoras que permiten que una variedad de equipos sean interconectados y puedan interactuar correctamente. [1,3]

El IEEE (Institute of Electrical and Electronics Engineers) definió las LAN así: *"Un sistema de comunicación de datos que permite que un número de dispositivos independientes se comuniquen directamente unos con otros, en un área física moderada y sobre un canal físico a velocidades moderadas"* [3]

Para tener un mejor entendimiento de la tesis, en este capitulo se dará una descripción de los conceptos y técnicas usadas en las redes de área local. En el anexo 1 hay un diccionario de los términos utilizados comúnmente en las redes de comunicación de datos

2.1 COMPONENTES DE LAS LAN

Las redes de área local son implementadas típicamente por una combinación de componentes hardware y software, entre estos componentes están: dispositivos de computo, tarjetas de interfaz de red, sistema de cableado, concentradores (hubs) y software de red.

2.1.1 DISPOSITIVOS DE COMPUTO

Una LAN es usada típicamente para interconectar dispositivos de computo de propósito general, tales como computadoras personales, estaciones de trabajo, impresoras, unidades de almacenamiento, etc.

2.1.2 TARJETAS DE INTERFAZ DE RED (NIC)

Cada dispositivo de computo debe tener una tarjeta de interfaz de red (NIC), las NIC son las que interactúan entre el dispositivo de computo y la red. Las NIC desarrollan las funciones de

hardware requeridas para proveer al dispositivo de computo de las capacidades físicas de comunicación.

2.1.3 SISTEMA DE CABLEADO

Se refiere al cableado necesario para interconectar las NIC instaladas en los dispositivos de computo. En las LAN se pueden implementar varios tipos de cables y fibra óptica. El sistema de cableado incluye los diferentes conectores utilizados. Algunas veces el sistema de cableado es remplazado por otras formas de comunicación, como: radio, microondas o señales infrarrojos.

2.1.4 CONCENTRADORES (HUBS)

Algunas LAN implementan dispositivos llamados concentradores, unidades de acceso o hubs que permiten que diversos dispositivos sean conectados al sistema de cableado a través de un punto central. Esto simplifica la instalación y mantenimiento de la red.

2.1.5 SOFTWARE DE RED

Las NIC desarrollan funciones de bajo nivel que permiten la comunicación física entre los diferentes dispositivos. Las funciones de alto nivel que emplea el usuario final para hacer su trabajo, son implementadas por el software de red. Estos software son llamados normalmente sistemas operativos de redes. Los sistemas operativos de red proveen al usuario muchas facilidades, como acceder archivos e impresoras.

2.2 MODELO DE REFERENCIA OSI / ISO

En el área de sistemas computacionales y comunicaciones existen muchas empresas proveedoras de productos para todo tipo de necesidades en la industria, más sin embargo cada empresa tiene su propia metodología de desarrollo de productos y como tal tiene dispositivos que se programan y comunican de manera diferente unos con otros. Esto ha creado serios problemas a la hora de querer intercomunicar entre sí dispositivos de diferentes marcas.

Por lo cual ha habido la necesidad de crear una norma de referencia que deben cumplir todos los desarrolladores de productos para comunicaciones.

El modelo OSI (Open System Interconnection) creado en 1984 por la ISO (International Standard Organization) es la referencia oficial para comunicaciones, pensada para abarcar desde redes locales hasta las grandes redes de paquetes conmutados, [1,2,3].

2.2.1 DESCRIPCIÓN DEL MODELO OSI

El modelo OSI es un modelo general para comunicación digital entre dos o más participantes. Éste provee la estructura para la elaboración de estándares para sistemas abiertos de comunicación, [1,2,3,4,11]

El modelo esta basado en los siguientes principios:

1. Las funciones de comunicación son divididas en capas.
2. Cada capa provee servicios específicos.
3. La capa N + 1 usa los servicios de las capas anteriores sin necesidad de cómo funcionen estas capas.
4. La comunicación entre la capa N y la terminal es especificada por el protocolo ISO.

El modelo esta conformado por 7 capas, figura 1.2. Donde cada capa puede consistir de varias subcapas, las capas 7, 6 y 5 se encargan de dar soporte al usuario y las otras capas 4, 3 ,2 y 1 se encargan de facilitar el flujo de información digital entre terminales y/o maquinas, ver figura 2.1

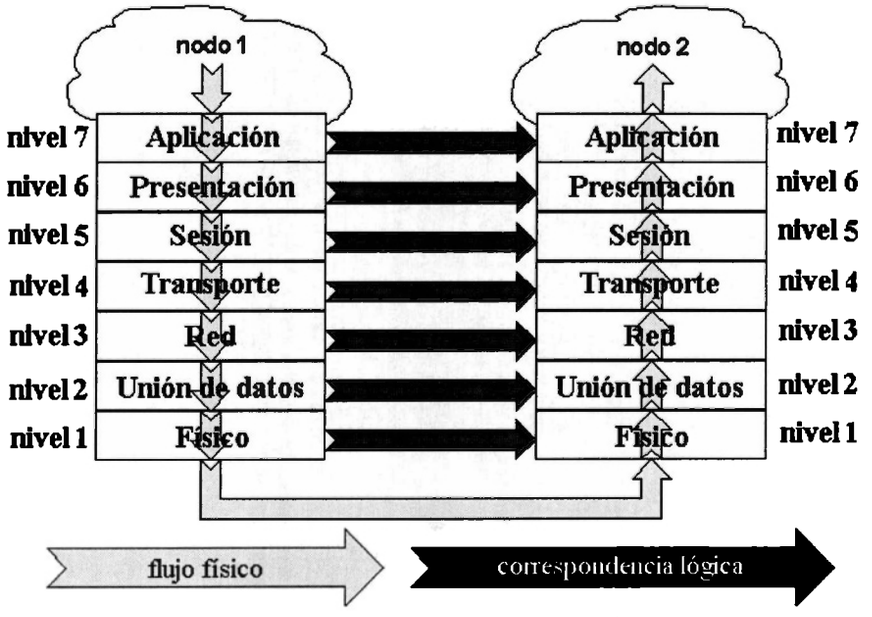


Figura 2.1. Niveles OSI: Flujo de dialogo, [11]

2.2.2 CAPAS DEL MODELO OSI

Como dijimos anteriormente el modelo OSI esta dividido en 7 niveles o capas, a continuación se dará una breve descripción de cada una.

Capa 1: Física

Define las características eléctricas y mecánicas del medio para la transmisión de datos, las tareas más importantes de esta capa son:

1. Conversión Paralela/Serial, multiplexación.
2. Interfase física del medio de transmisión (Fibra óptica, cable coaxial).
3. Sincronización de bits.
4. Definición de señales validas y líneas de conexión.

Capa 2: Enlace

Provee el acceso al medio de transmisión y asegura la transmisión de bloques individuales de datos, sus tareas son:

1. Activación y desactivación de acceso al medio de transmisión
2. Sincronización de bloques
3. Monitoreo de la secuencia de bloques de datos
4. Detección de errores y corrección si es necesario

Funcionalmente puede ser dividida en dos capas MAC (Médium Access Control) y LLC (Logical Link Control), las cuales describiremos más adelante.

Capa 3: Red

Esta capa es la responsable real del encaminamiento de mensajes entre nodo y nodo, a través de un medio físico, si importar el medio ni el contenido del mensaje. Otras tareas son:

1. Formación de paquetes.
2. Control del intercambio de paquetes entre las terminales y la red.
3. Estabilizar la conexión virtual entre las terminales.
4. Transporte de datos entre dos terminales.

5. Ruteo con la red.

Capa 4: Transporte

Esta capa es la responsable de establecer un medio de comunicación y garantizar la transferencia de información sin errores en ambos sentidos. Apoyándose en los niveles inferiores, actúa como un administrador capaz de interpretar las direcciones, fraccionar, otras funciones son:

1. Estabilizar la conexión de transporte.
2. Control de flujo de punto a punto

Capa 5: Sesión

Es un dialogo interactivo, controla la comunicación, decidiendo quien debe transmitir y quien debe recibir, también señala el inicio y final de la comunicación, otras funciones son:

1. Conversión de direcciones simbólicas en direcciones reales
2. Declaración de los parámetros de sesión (tipo de comunicación, control de flujo)
3. Reestabilizar las interrupciones

Capa 6: Presentación

Se encarga de facilitar la comunicación en el ámbito de lenguaje y formato de presentación, que puede ser negociado entre las terminales que se están tratando de comunicar.

Capa7: Aplicación

Es la capa más importante desde el punto de vista del usuario, se encarga de proporcionar un entorno que facilite el entendimiento entre usuarios de distintas máquinas digitales en el ámbito temático, sin importarle medios ni protocolos de comunicación.

2.3 TOPOLOGÍAS DE REDES

Las topologías de redes se refieren a la disposición física de las distintas terminales que componen la red y la forma en que se encuentran enlazadas por el medio físico. Existen básicamente 4 tipos de topologías que son: Estrella, Anillo, Bus y árbol, en la figura 2.2 se pueden observar más claramente estas disposiciones.

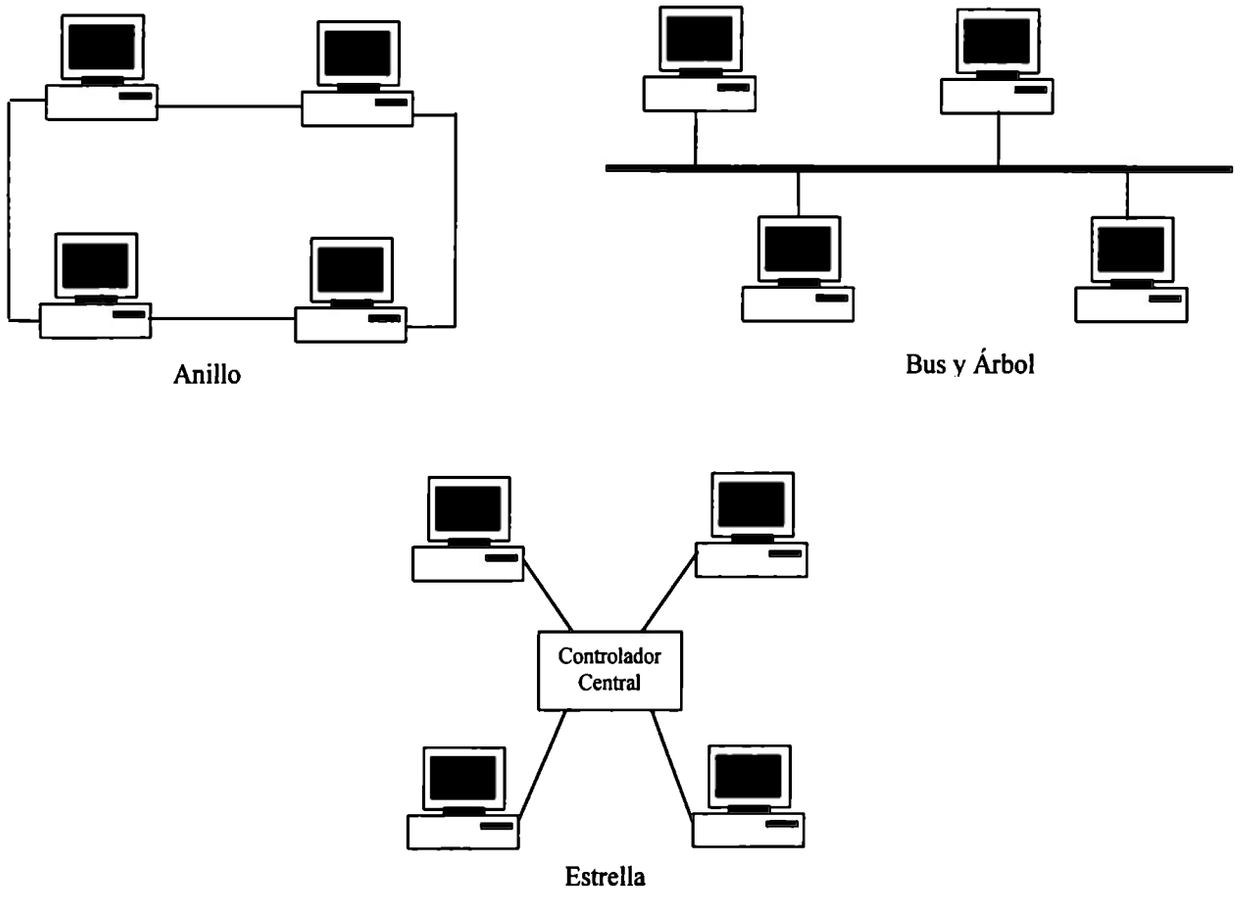


Figura 2.2. Topologías de red

2.3.1 ESTRELLA

En esta topología todas las estaciones están conectadas a ana estación central o maestra, esta es la más vieja y simple implantación de una red, un ejemplo son los PBX de las redes de telefonía. Por el nodo central pasan todos los datos, cada estación debe comunicarse primero con el nodo central para poder enviar y recibir información.

La principal ventaja es la facilidad de añadir nuevos nodos y que si se daña un nodo solo se afecta el trafico en este nodo, y su principal desventaja es que cualquier fallo en el nodo central deshabilita todo el sistema

2.3.2 ANILLO

En esta topología las estaciones están conectadas en forma de lazo cerrado, los datos circulan en una única dirección de tal manera que cada estación revisa los datos que están en circulación y si son para ella los toma y si no los envía a la estación siguiente.
Puede haber una estación maestra o puede haber control distribuido entre las diferentes estaciones. La red puede crecer indefinidamente, aunque que cada vez que se adicione un nodo se va perdiendo velocidad.

2.3.3 BUS

En esta topología las estaciones están unidas entre sí a través de líneas comunes. Su principal desventaja es que solo una línea puede utilizar el bus a la vez, para esto se debe utilizar algún método de control de acceso, del cual hablaremos más adelante.
Este tipo de red es el más utilizado en las redes industriales.

2.3.4 ÁRBOL

Es similar a la topología en bus, solo que esta permite subdivisiones en los nodos.

2.4 NIVEL FÍSICO DE LA RED

El nivel físico se refiere al conjunto de elementos hardware utilizados para transmitir la señal eléctrica u óptica entre los diversos nodos de la red. Existen varios medios físicos de comunicación, entre los cuales están: cables eléctricos, fibra óptica, microondas, radio, señales infrarrojas.

El nivel físico también es el encargado de definir las interfaces de la red, estas interfaces son dispositivos que desarrollan funciones de apoyo a las comunicaciones.

2.4.1 MEDIOS DE TRANSMISIÓN

Los principales medios de transmisión son el cable coaxial, el par trenzado y la fibra óptica. El tipo se selecciona dependiendo de la frecuencia de transmisión deseada.
La fibra óptica se caracteriza por evitar las interferencias por ruido eléctrico, los cables de fibra óptica están formados por conductores ópticos, la velocidad de la luz en su interior es del orden de 200.000 Km/s.

2.4.2 INTERFACES

Las interfaces son los diferentes dispositivos con funciones de apoyo a las comunicaciones.

2.4.2.1 Modem

Para la transmisión de señales a distancias muy grandes, se usan las líneas telefónicas, para poder hacer esta transmisión es necesario un dispositivo intermedio entre la línea telefónica y el sistema digital, este dispositivo es el Modem que se encarga de la Modulación y demodulación de las señales.

2.4.2.2 Repetidor

Es un dispositivo que se usa para amplificar la señal eléctrica permitiendo que ésta viaje distancias más grandes entre los nodos. También puede ser utilizado para adaptar diferentes medios físicos, tal como RS-485 con fibra óptica. En relación con el modelo OSI, este dispositivo solo trabaja con la capa 1. ver figura 2.3

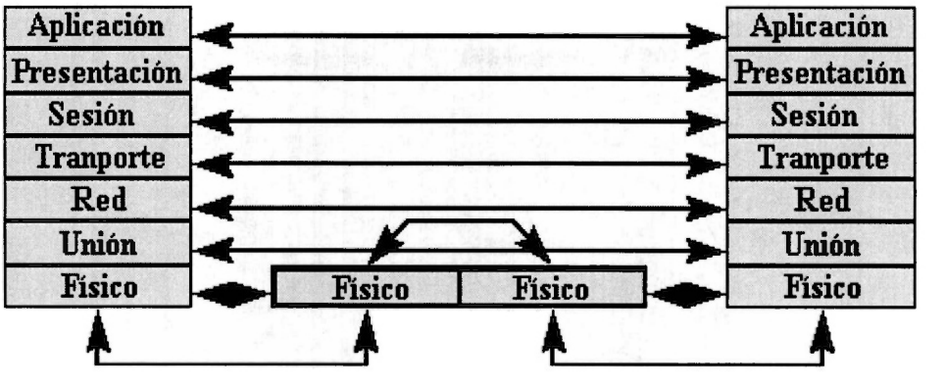


Figura 2.3 Repetidor, [11]

2.4.2.3 Puente

El puente enlaza a las capas 1 y 2 del modelo OSI. Este permite la conexión entre dos secciones diferentes de una red que tiene diferentes características eléctricas y de codificación. En general puede enlazar dos tipos diferentes de redes. Ver figura 2.4

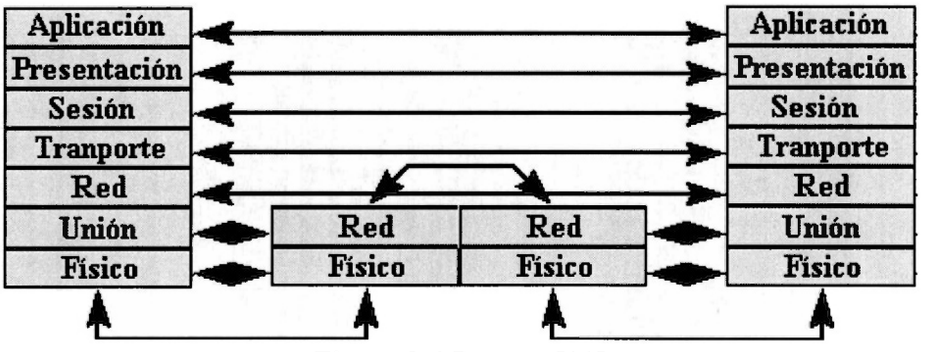


Figura 2.4 Puente, [11]

2.4.2.4 Ruteador

Como su nombre lo dice, el ruteador define el camino a seguir entre diferentes segmentos de red, este debe interpretar las señales de la capa 3 del modelo OSI. Ver figura 2.5

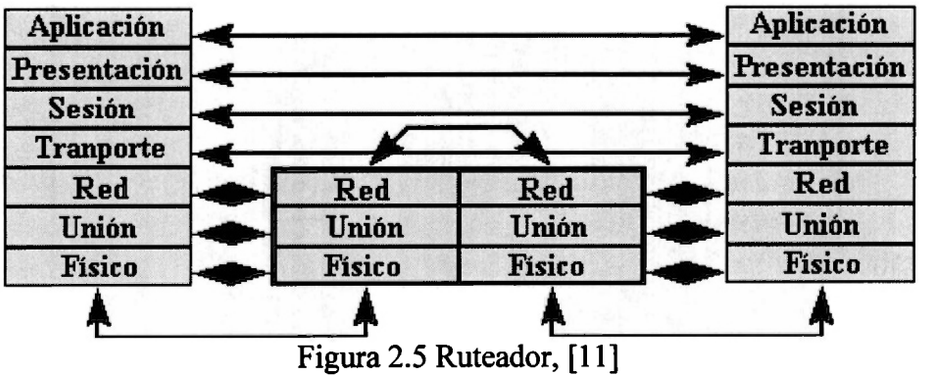


Figura 2.5 Ruteador, [11]

2.4.2.5 Gateway

Es parecido a un puente pero con suficiente inteligencia para decodificar las señales, maneja las 7 capas del modelo OSI. El gateway permite la comunicación de buses de diferentes tipos de características. Ver figura 2.6.

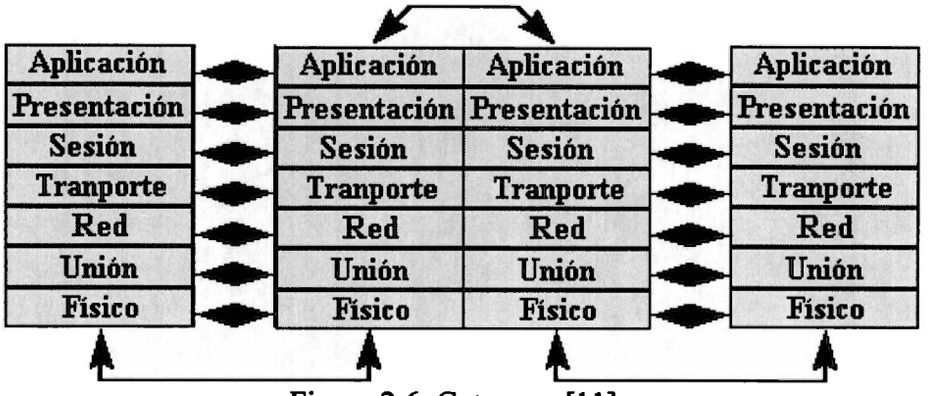


Figura 2.6. Gateway, [11]

2.4.3 TRANSMISIÓN DE DATOS

El termino dato normalmente se utiliza para describir un grupo de uno o más caracteres y números digitalmente codificados que se utilizaran para intercambiar entre dos dispositivos. La idea principal de las redes es precisamente el intercambio de datos que serán interpretados por cada uno de los participantes en la comunicación y tomarán las debidas acciones indicadas en estos datos.

En comunicaciones digitales los datos son enviados en forma binaria (unos y ceros). Los bytes que esta formados por 8 bits (cada bit puede ser un uno o cero) representan números y caracteres codificados en alguno de los estándares de codificación (eje, código ASCII o EBCDIC), la unión de bytes representan palabras, y las palabras representan información.

Cuando se establece una comunicación entre dos dispositivos, estos deben indicar el lenguaje o código que van a utilizar para entenderse mutuamente, posteriormente deberán codificar y decodificar la información cada vez que envíen o reciban datos.

Existen dos formas de enviar la información, estas dos formas son: paralela y serial. En la transmisión paralela se envía la información por caracteres, ésta transmisión es más rápida que la

serial, pero tiene el inconveniente de que utiliza más líneas físicas de transmisión, una por cada bit. En la transmisión serial la información se envía bit por bit, es más lenta que la anterior, pero tiene la ventaja de utilizar solo dos o tres líneas físicas, para controlar el inicio y final de los caracteres se utiliza un bit de inicio y bit de parada. Esta transmisión es la más utilizada, sobre todo para grandes distancias, ya que se requiere mucho menos cableado.

En la comunicación en serie que consiste en una secuencia de unos y ceros, codificados mediante niveles de tensión, por flancos o por señales moduladas en frecuencia, existen dos problemas básicos a tratar que son:

1. Sincronización
    a. Sincronización de bits
    b. Sincronización de caracteres
    c. Sincronización de mensaje
2. Codificación de los bits

2.4.3.1 Sincronización de bits

Existen dos grandes métodos de transmisión en serie que son: transmisión asíncrona y transmisión síncrona

2.4.3.1.1 Transmisión Asíncrona

En este tipo de transmisión no se transmite la señal del reloj y para sincronizar la transmisión, se utilizan bits de inicio y bits de parada con los valores de 0 y 1 respectivamente. Generalmente la transmisión tiene 11 o 12 bits donde 1 es de inicio, 8 de datos, 1 o 2 de parada y 1 de paridad.

2.4.3.1.2 Transmisión Síncrona

En este tipo de transmisión si se transmite la señal de reloj, esto se puede hacer por una línea adicional ó de forma que el receptor pueda extraer dicha señal de la línea de datos, esto se hace seleccionando una forma de codificación apropiada, como por ejemplo modulación de fase, de frecuencia, Manchester, etc.

2.4.3.2 Sincronización de los caracteres

2.4.3.2.1 En comunicación Asíncrona

Como en la sincronización de bits se realiza enviando caracteres de inicio y de parada. Cuando no hay comunicación la línea se mantiene en nivel alto, hasta que llega el primer carácter de inicio.

2.4.3.2.2 En Comunicación Síncrona

Para velocidades de transmisión más altas y optimizar la información útil se recurre a este tipo de transmisión, en esta transmisión se evita los bits de inicio y de parada, pero se necesitan formas de codificación que permitan resincronizar el reloj a cada bit. La forma de sincronizar el carácter consiste en intercalar periódicamente unos caracteres especiales denominados de sincronismo (SYNC).

2.4.3.3 Codificación

Los códigos de codificación de bits más utilizados son: NRZ (Non Return to Zero), NRZI (Non Return to Zero Inverted), Manchester, AMI (Alternate Mark Inversión), HDBn (High Density Bipolar of order n), CMI (Coded Mark Inversión).
A nivel de byte los más utilizados son: ASCII y EBCDIC.

2.4.4 ENLACES MÁS USADOS

En esta sección se dará una breve descripción de los tipos de enlace más usados en las redes industriales.

2.4.4.1 RS-232C, V.24

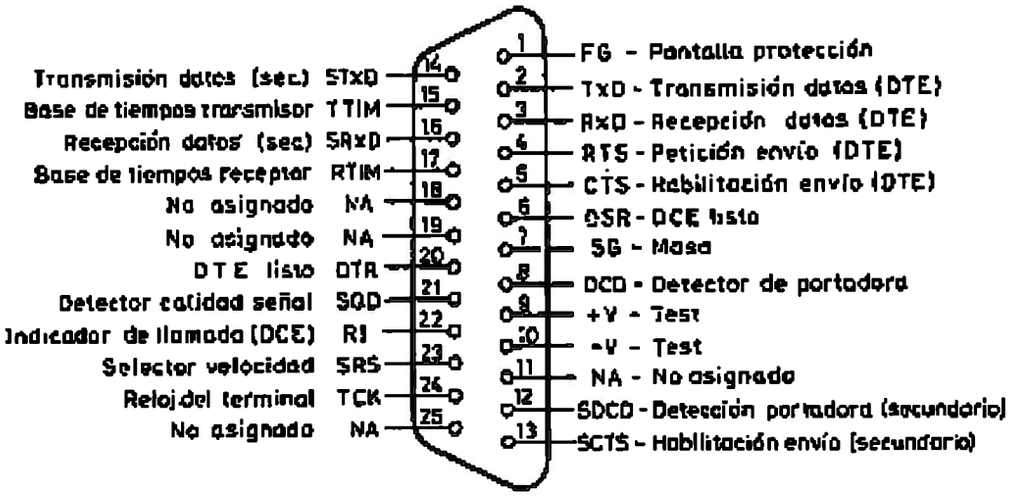
Propuesto por EIA (Electrical Industries Association) en 1960, es equivalente al estándar europeo V.24 de CCITT. Es la interfaz utilizada para conectar una computadora con un modem. La especificación mecánica establece el uso de un conector de 25 pines conocido como DB-25 para el uso total de todas sus funciones, pero se puede usar también conectores con menos de 25 pines

como el DB-9 o el RJ-45 (ISO 8877 de 8 pines) ya que no todos los circuitos se necesitan en todas las aplicaciones.

La especificación eléctrica define que las señales en los circuitos de intercambio se transmiten como voltajes aislados con referencia a una misma tierra (pin 7) y que este circuito se conecta a las tierras lógicas en cada equipo. También se especifica que cualquier voltaje menor a –3V es identificado como un 1 lógico y cualquier voltaje mayor que 3V es un 0 lógico, y la magnitud de los voltajes están entre 5 y 15V.

La figura 2.7 se muestra la configuración de los conectores DB-25 y DB-9 y la tabla 2.1 describe las señales más importantes de la interfaz. Dependiendo de la aplicación se seleccionan las señales a utilizar.

Las principales limitaciones de la norma RS-232C son su velocidad máxima de transmisión y su alcance máximo: la norma especifica que la velocidad de transmisión debe ser menor a 20 Kbps y su distancia menor a 15m.



a) Conector DB-25 según norma básica EIA RS-232C

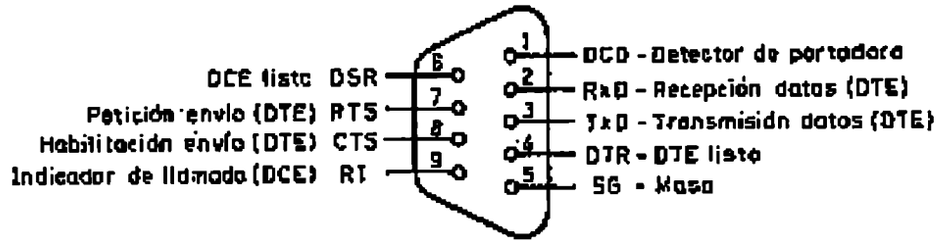


Figura 2.7 Conectores empleados en la conexión RS-232C, DB-25 y DB-9

Señal	*Descripción*
FG	(Field Ground).Pantalla de protección contra ruido electromagnético. Común a ambos extremos
TxD	Pin de transmisión de datos (Salida)
RxD	Pin de recepción de datos (Entrada)
RTS	(Request to send). Pin de salida de un terminal que indica que éste esta dispuesto para enviar un dato. Debería interpretarse como <<dispuesto para transmitir>>
CTS	(Clear to send). Pin de entrada de un terminal que indica habilitación para transmitir. Debe interpretarse como una señal de receptor que indica <<Listo para recibir>>
DSR	(Data Set Ready). Para un terminal es una entrada que indica que el otro terminal interlocutor ésta dispuesto. Para un MODEM será un pin de salida que indica que esta dispuesto para la comunicación.
DTR	(Data Terminal Ready), Para un terminal es una salida que indica que éste esta dispuesto para transmitir o recibir.
DCD	(Data Carrier Detect). Esta señal es propia de un MODEM, en el cual sería una salida que indicaría que éste esta recibiendo señal portadora. En un terminal debe interpretarse como una entrada que indica que el interlocutor esta dispuesto y se suele emplear como alternativa de DSR.
RI	(Ring Indicator). Esta señal es propia de un MODEM e indica que éste esta recibiendo una llamada por línea telefónica. No suele emplearse en enlace entre terminales.
SG	(Signal Ground). Línea de masa o cero señal. Es común a ambos lados del enlace.

Tabla 2.1 Descripción de las señales más importantes de RS-232C

2.4.4.2 RS-485

El RS-485 es un medio físico de transmisión y no un protocolo como la norma RS-232, por lo tanto en el RS-485 pueden conectarse buses de distintas configuraciones pero con características físicas semejantes como niveles de tensión y tipo de conexión. Los niveles de tensión empleados van entre $\pm 5V$ y $\pm 6V$ para el nivel lógico 1 y 0V para el nivel lógico 0, y las distancias y frecuencias admitidas en el bus son (1200m a 1500m y 2400 baudios a 19200 baudios). Estas

líneas deben cargarse a la salida con resistencias de terminación de línea entre 100 Ω y 250 Ω, dependiendo del tipo de cable empleado.

Es un enlace XON, XOFF (que existen solo líneas de datos y a lo sumo una línea tierra),en este caso solo usa una misma línea para transmitir y recibir ya que es semidúplex, esto requiere un software de control de enlace (capa OSI 2) que haga conmutar la línea para que la terminal transmita o reciba datos, la figura 2.8 muestra el principio de enlace.

Este tipo de enlace se suele utilizar en topología de bus como lo muestra la figura 2.9. En la conexión en red, el número máximo de terminales conectadas suele estar limitado a 32 por razones de carga, sin embargo pueden utilizarse repetidores o amplificadores de bus para tener más terminales.

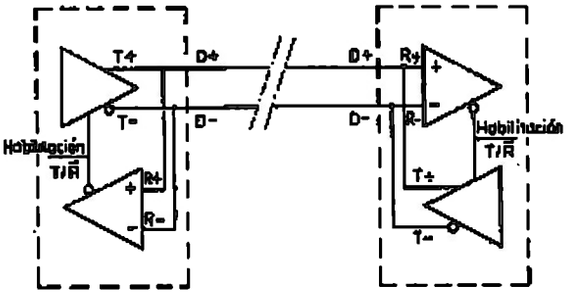


Figura 2.8 Enlace punto a punto RS-485

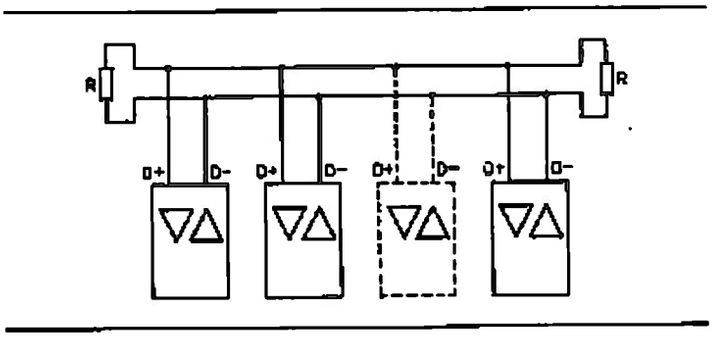


Figura 2.9 Enlace red mediante bus RS-485

2.4.5 ESTRUCTURA LÓGICA DE LAS LAN

Esta estructura se refiere a la capa 2 (Enlace) del modelo OSI, esta capa se divide en dos grupos MAC (Medium Access Control) y LLC (Logic Link Control), el conjunto de estas tareas y considerando cierto hardware es lo que se conoce como protocolo.

líneas deben cargarse a la salida con resistencias de terminación de línea entre 100 Ω y 250 Ω, dependiendo del tipo de cable empleado.

Es un enlace XON, XOFF (que existen solo líneas de datos y a lo sumo una línea tierra),en este caso solo usa una misma línea para transmitir y recibir ya que es semidúplex, esto requiere un software de control de enlace (capa OSI 2) que haga conmutar la línea para que la terminal transmita o reciba datos, la figura 2.8 muestra el principio de enlace.

Este tipo de enlace se suele utilizar en topología de bus como lo muestra la figura 2.9. En la conexión en red, el número máximo de terminales conectadas suele estar limitado a 32 por razones de carga, sin embargo pueden utilizarse repetidores o amplificadores de bus para tener más terminales.

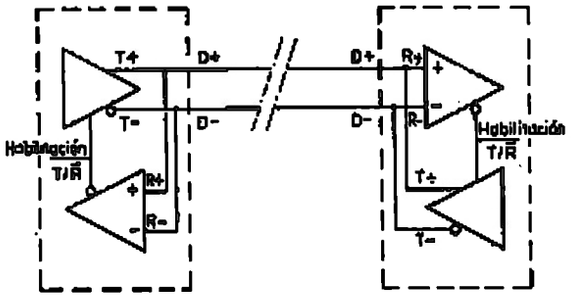


Figura 2.8 Enlace punto a punto RS-485

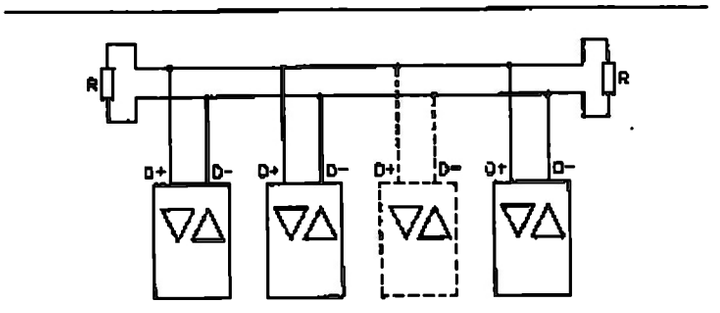


Figura 2.9 Enlace red mediante bus RS-485

2.4.5 ESTRUCTURA LÓGICA DE LAS LAN

Esta estructura se refiere a la capa 2 (Enlace) del modelo OSI, esta capa se divide en dos grupos MAC (Medium Access Control) y LLC (Logic Link Control), el conjunto de estas tareas y considerando cierto hardware es lo que se conoce como protocolo.

2.4.5.1 Control de Acceso al Medio (MAC)

En las redes industriales el medio físico más utilizado es un bus compartido por todas las estaciones de la red. Esto hace que sea necesario el control de transmisión para evitar las transmisiones simultaneas, hay dos formas de hacer este control, uno es control centralizado en donde una estación maestra controla el acceso de todas las demás estaciones al bus, el problema de este tipo de configuración es que si se daña el maestro toda la red deja de funcionar. La otra es control descentralizado en donde no existe una estación maestra sino que cada estación toma el bus cada vez que lo necesita, tiene el inconveniente de que varias pueden tomar el bus simultáneamente, por lo tanto se debe establecer un método de control de acceso al bus, uno de estos métodos es:

Paso de Testigo (Token Passing) descrito en la recomendación IEEE-802.4, el cual consiste en crear un mensaje llamado testigo que recorre todas las estaciones y le asigna el derecho a cada estación según la prioridad o secuencia asignada, después de que dicha estación toma el mensaje, solo esta estación esta habilitada para transmitir, mientras las otras estaciones están en espera o recepción. La forma de asignar el testigo a las distintas estaciones se hace por uno de los siguientes criterios: Limitación del número a transmitir en cada turno, reparto equitativo de tiempos, reparto del tiempo con ciertas prioridades, o algún otro criterio que sea conveniente.

El protocolo debe incluir además de las condiciones del nivel de enlace las siguientes funciones: Inicialización del anillo lógico, adición o supresión de estaciones al anillo, recuperación de errores en caso de pérdida de testigo. El tiempo de respuesta estaría dado por:

$$T_r = (n-1)T_0$$

En donde n es el número de estaciones activas, y T_o el tiempo máximo de ocupación asignado a cada estación.

También se permite modificar las prioridades de acceso.

2.4.5.2 Control Lógico de Enlace (LLC)

Esta definido por la recomendación IEEE-802.2 y básicamente el LLC se encarga de controlar que terminal se esta comunicando con que otra terminal, y cuando empieza y termina el mensaje, es decir se encarga de hacer un protocolo de comunicación de tal manera que cuando una estación mande un mensaje, se sepa hacia que estación va dirigido este, y cuando la estación

reciba el mensaje sepa de que estación proviene este. También se encarga del control de inicio y fin del mensaje enviando caracteres de control.

Los mensajes pueden ir dirigidos de tres formas: Enlace punto a punto en donde una estación envía un mensaje solo a un destinatario, enlace con grupo en donde una estación envía un mensaje a un grupo concreto de destinatarios y enlace difundido en donde una estación envía un mensaje a todas las estaciones de la red.

Es bueno aclarar que para las redes WAN (Red de área extensa) la capa 3 del modelo OSI se encarga de esto, pero para redes LAN (Red de área local) se encarga la capa 2 del modelo OSI.

3 REDES INDUSTRIALES, FIELDBUS

Las LAN inicialmente fueron pensadas para redes de computadores, pero con los avances tecnológicos en el área de automatización y control, se fueron incorporando dispositivos basados en microprocesador en estas áreas, posteriormente se fue viendo la necesidad de intercomunicar estos dispositivos de control entre sí y/o con computadoras centrales que administren el sistema total, esto para poder intercomunicar el área de producción de la compañía con las áreas administrativas, llegando a tener una empresa totalmente automatizada.

En los procesos de automatización y control industrial se utilizan diversos dispositivos tales como sensores, actuadores, PLC's, microcontroladores, multiplexores, controladores PID, etc., encargados de la automatización y control de la producción. Estos dispositivos normalmente se encuentran separados y con cableados diferentes formando un conjunto de islas que controlan distintas áreas de la producción, esto lleva a una gran cantidad de cableado innecesario y a una difícil administración de todo el sistema de automatización, ya que cada vez que se quiere hacer una modificación, es necesario volver a cablear y a reconfigurar todos los dispositivos. Por esta razón muchas veces es necesario implementar una red industrial de comunicación en el área de automatización, con estas redes se logra una mejor administración, confiabilidad, reducción de costos y sobre todo poder compartir recursos de los diferentes dispositivos del sistema, [4,5,8].

Por definición, una red industrial consiste en la distribución geográfica de instrumentos de medida, actuadores y dispositivos de control que interactúan entre sí de forma digital para llevar a

cabo una actividad de automatización y control, [12]. Normalmente a las redes industriales se les conoce como buses de campo o Fieldbus que es como se designará en este proyecto.

Fieldbus surgió por un trabajo de estandarización iniciado por la ISA (Instrumentations Society of America), [13].

Fieldbus desarrolla una estrategia de control distribuido, esto es posible ya que los dispositivos conectados a la red, son dispositivos inteligentes (basados en microprocesador) y esto dispositivos tienen la habilidad de comunicarse unos con otros de una manera rápida y segura. Con esta estrategia de control distribuido se orienta a una aplicación cliente / servidor o maestro / esclavo como se le llama en redes industriales, en este tipo de estrategia el maestro es un equipo con mayor capacidad de procesamiento que los esclavos, este equipo se encarga de administrar cada uno de sus esclavos y almacenar la información generado por estos.

Los dispositivos pueden ser configurados de acuerdo a las necesidades del usuario, cada dispositivo desarrolla una actividad de automatización y control y la comunica periódicamente a los otros dispositivos de más alto nivel que tiene derecho de conocer esta información.

Fieldbus presenta muchas ventajas para la implantación de redes de automatización y control, algunas de esas ventajas son, [9,10,11,13]:

1. Interconectividad, ya que se pueden conectar equipos de diferentes fabricantes.
2. Gran reducción de cableado, ya que todos los equipos se conectan al mismo cable.
3. Conexión de dispositivos inteligentes.
4. Más fácil la administración del sistema global en cuanto a cableado y software. Al ser comunicación digital se puede utilizar la capacidad de las computadoras personales para el manejo y control de información.
5. Cada dispositivo puede operar múltiples variables
6. Más fácil adicionar o quitar dispositivos de la red.
7. Se pueden integrar al sistema dispositivos de control, alarmas, supervisión, visualización, administración, y en general cualquier dispositivo basado en microprocesador.
8. Conexión de dispositivos inteligente que ayudan a desarrollar actividades avanzadas de diagnostico.
9. Facilidad de mantenimiento, ya que cada dispositivo puede reconocer sus fallas y las comunica.

10. Control distribuido. Entre otras.

Fieldbus cubre solo una parte del modelo OSI de la ISO, esta parte son los niveles: nivel 1 (físico), nivel 2 (enlace) y nivel 7 (aplicación), algunas funciones que son propias de los niveles de red y sesión se añaden a los niveles 2 y 7.

Al igual que en las redes de computadores, en las redes industriales también se ha trabajado en la creación de protocolos de comunicación. Se han establecido diversos protocolos muy parecidos entre sí, aunque con pequeñas diferencias, [1,4,5,6], entre estos están: BITBUS de Intel, FIP de AFNOR, MIL-STD-1535 de ANSI, MODBUS de GOULD INC, Interbus-S de Phoenix, DeviceNet de Allen-Bradley, Ethernet de Intel y Xerox, PROFIBUS de DIN que es en el que nos centraremos en esta tesis ya que se esta teniendo una aceptación muy grande en todo el mundo y hay muchas empresas comprometidas con este producto como Siemens, Bosch, Klockner Moeller, ABB, AEG, Landis&Gir y algunas universidades alemanas, [9].

3.1 REGLAS GENERALES DE FIELDBUS SEGÚN IEC

La IEC por medio del comité TC65C-WG6 ha definido unas reglas generales para tratar de formar un estándar para fieldbus, estas reglas son:

1. Nivel físico: Bus serie controlado por un maestro, semiduplex, comunicación en banda base
2. Transmisión digital y serialmente.
3. Velocidades: para distancias cortas 1 Mbit/s o valores inferiores, entre 250 kbits/s y 64 kbits/s para distancias largas.
4. Longitudes: 40 m para la máxima velocidad y 350 m a velocidades más bajas
5. Número de periféricos: Máximo de 30 nodos, con posibles ramificaciones hasta un máximo de 60 elementos.
6. Tipo de cable: Pares de cables trenzados y pantalla.

7. Conectores: Bornes de tipo industrial o conectores DB9 o DB25
8. Conexión / desconexión (on line): la conexión o desconexión de algún nodo no debe interferir el tráfico de datos.
9. Topología: Bus físico con posibles derivaciones.
10. Longitud de ramificación: Máxima longitud de 10m.
11. Aislamientos: 500 Vc.a. permanentes entre elementos de campo y bus. Tensión de prueba 1500 Vc.a./1 minuto.
12. Seguridad intrínseca: opción a conectar elementos de campo con tensiones reducidas para atmósferas explosivas.
13. Longitud de mensaje: Mínimo 16 bytes por mensaje.
14. Transmisión de mensaje: posibilidad de dialogo entre cualquier par de nodos sin repetidor.
15. Maestro flotante: Posibilidad de maestro flotante entre diversos nodos, es decir que cualquier estación pueda ser maestro en determinado momento, según normas establecidas.
16. Implantación de protocolo: los circuitos integrados que implementen el protocolo deben estar disponibles comercialmente y ser de dominio publico.

3.2 MAP (MANUFACTURING AUTOMATION PROTOCOL)

MAP es uno de los intentos más serios de estandarización de redes industriales, desde el nivel de red local hasta el nivel de WAN para enlazar incluso varias fabricas. Fue impulsado por General Motors y IEEE desde 1980 [5]. Las razones que impulsaron esta iniciativa fueron que GM preveía que para la década de los noventa debía integrar en red unos 400.000 sistemas más o menos, basados en microprocesador y de distintas marcas [4,8].

El advenimiento de MAP ha tenido un gran efecto en el desarrollo y implantación de redes de computadores y aplicaciones distribuidas.
MAP es un conjunto de protocolos seleccionados por el grupo de usuarios de MAP.

MAP cumple con los 7 niveles del modelo OSI, como se dijo anteriormente, MAP se basa en protocolos ya existentes para cada nivel. La Figura 3.1 describe estos protocolos.

Nivel	*Protocolo*
7 (Aplicación)	MMS, FTAM, MAP CASE
6 (Presentación)	ASCII y Codificación binaria
5 (Sesión)	ISO sesión (IS 8327)
4 (Transporte)	ISO transporte (IS 8073)
3 (Red)	ISO Internet (DIS 8473), opcional X.25
2 (Enlace)	ISO (DIS 8802/2), IEEE 802.2, opcional X.25.
1 (Físico)	IEEE 802.4 token passing bus MAC

Tabla 3.1 Protocolos en MAP [5]

MAP esta pensado para redes WAN, Para redes industriales aplicar el protocolo MAP resulta bastante complicado y costoso, en vista de estoe el grupo de MAP creo un subconjunto del protocolo pensado para redes industriales de área local, este protocolo simplificado se conoce como MINIMAP. MINIMAP usa las mismas características de MAP con la diferencia de que solo opera los niveles 1, 2 y 7 del modelo OSI, ver figura 3.1.

En este proyecto no entraremos en detalle sobre este protocolo, Para un conocimiento más detallado de MAP y MINIMAP puede consultar la referencia [5].

4 PROFIBUS

PROFIBUS (Process fieldbus), fue iniciado en 1987 por las firmas alemanas Bosch, Klockner Moeller y Siemens, a este grupo de trabajo se integraron otras grandes compañías como: ABB, AEG, Landis&Gir y algunas universidades alemanas. El objetivo del proyecto era crear un sistema abierto de comunicaciones, apto para integrar todos los dispositivos usados en la automatización industrial. En los niveles de red jerárquicamente más altos se adopto la red MAP, con lo cual el objeto era solo el diseño de un bus de campo con una estructura abierta y un protocolo compatible para enlazar con MAP.
El resultado de estos desarrollos fue el proyecto de norma DIN-19245 parte 1 que define las técnicas de acceso al medio (MAC) y enlace lógico (LLC) y la parte 2 que define el nivel de aplicación a través del protocolo denominado FMS (Fieldbus Message Specification) basado en un subjuego del protocolo MMS de MAP según ISO IS-9506, [8,9].

A partir de 1990 se abrió la posibilidad para cualquier usuario o empresa de integrarse en un consorcio denominado PROFIBUS Nutzerorganisation [8], que a través de diversos comités sigue desarrollando y dando soporte al nivel de aplicación y certificación de productos acordes con la norma DIN-19245.

PROFIBUS intenta definir toda una red de comunicación industrial, desde el nivel físico hasta el de aplicación. Una de sus virtudes estriba en el hecho de aplicar e integrar al máximo las técnicas de comunicación previamente definidas y consolidadas en algunas normas, tales como el propio modelo OSI de ISO, la EIA RS-485 a nivel físico, Token passing a nivel de enlace lógico (MAC y

LLC). Sin embargo tiene un inconveniente y es la poca disponibilidad de información en lenguaje diferente al alemán [9].

PROFIBUS solo implementa tres niveles del modelo OSI, estos niveles son: Nivell (físico), Nivel 2 (Enlace), nivel 7 (Aplicación).

4.1 RESUMEN DE PROFIBUS

PROFIBUS (Process fieldbus) es un bus (red, en donde los dispositivos conectados utilizando el mismo medio físico de transmisión) de comunicación que implementa una metodología maestro / esclavo [9], en donde los maestros son dispositivos con buena capacidad de procesamiento, que controlan el acceso al medio de comunicación (hay que tener en cuenta que solo un dispositivo puede usar el medio de comunicación en un momento determinado, ya que todos se comunican por el mismo medio) y tienen a su cargo varios esclavos, que son los dispositivos que están realizando tareas de control (controladores, sensores, actuadores, etc...). El maestro viene realizando las actividades de supervisión y control de sus esclavos, comunicándose con estos periódicamente de la manera establecida por el protocolo, la figura 4.1 muestra esta estructura, en donde cada maestro tiene asignados unos esclavos así: el maestro 1 tiene el esclavo A, el maestro 2 tiene los esclavos B y C, el maestro 3 tiene los esclavos D y E, y todos se están comunicando por el mismo medio.

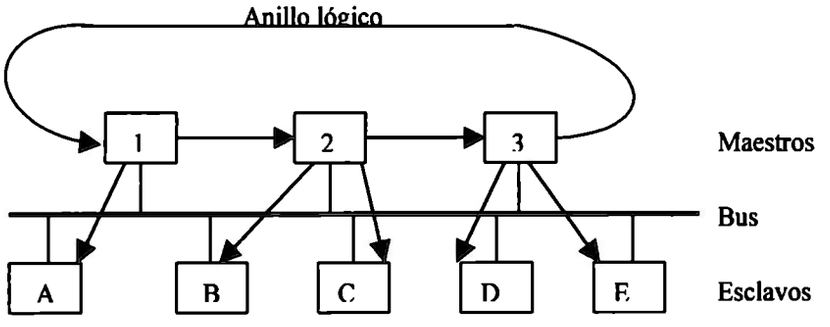


Figura 4.1 Esquema de arquitectura maestro / esclavo

PROFIBUS trabaja con una metodología orientada a objetos que facilita la integración de diferentes dispositivos en una misma red, [9,13]. En esta metodología, se maneja dispositivos

virtuales, en donde cada dispositivo conectado a la red maneja un número determinado de variables dependiendo de la tarea de control que este realizando (estas variables podrían ser corriente, voltaje, peso, presión, etc...). El maestro implementa un software que trabaja con una base de datos en la cual tiene almacenada todas de las variables que maneja cada uno de sus esclavos. Para cada esclavo el maestro crea un dispositivo virtual que es con el que interactúa el usuario del sistema, sin necesidad de tener que modificar directamente el esclavo. Cada vez que el usuario quiere conocer o modificar alguna de las variables que maneja el dispositivo de control (esclavo), el usuario hace los cambios en el dispositivo virtual del maestro y el maestro por medio de la red actualiza el esclavo. También cuenta con características que ayudan a la adición y eliminación a la red de nuevos dispositivos.

PROFIBUS esta definido por varios niveles que ayudan a interconectar las diferentes áreas automatización. Estos niveles son: PROFIBUS-DP que establece la comunicación entre maestros y esclavos, PROFIBUS-FMS que establece la comunicación entre maestros y PROFIBUS-PA que se utilizado en el área de automatización de procesos, A nivel físico, define RS-485 y IEC 1158-2 para procesos industriales [9].La arquitectura del protocolo PROFIBUS es mostrada en la figura 4.2. Es importante tener en cuenta que PROFIBUS esta basado en MINIMAP.

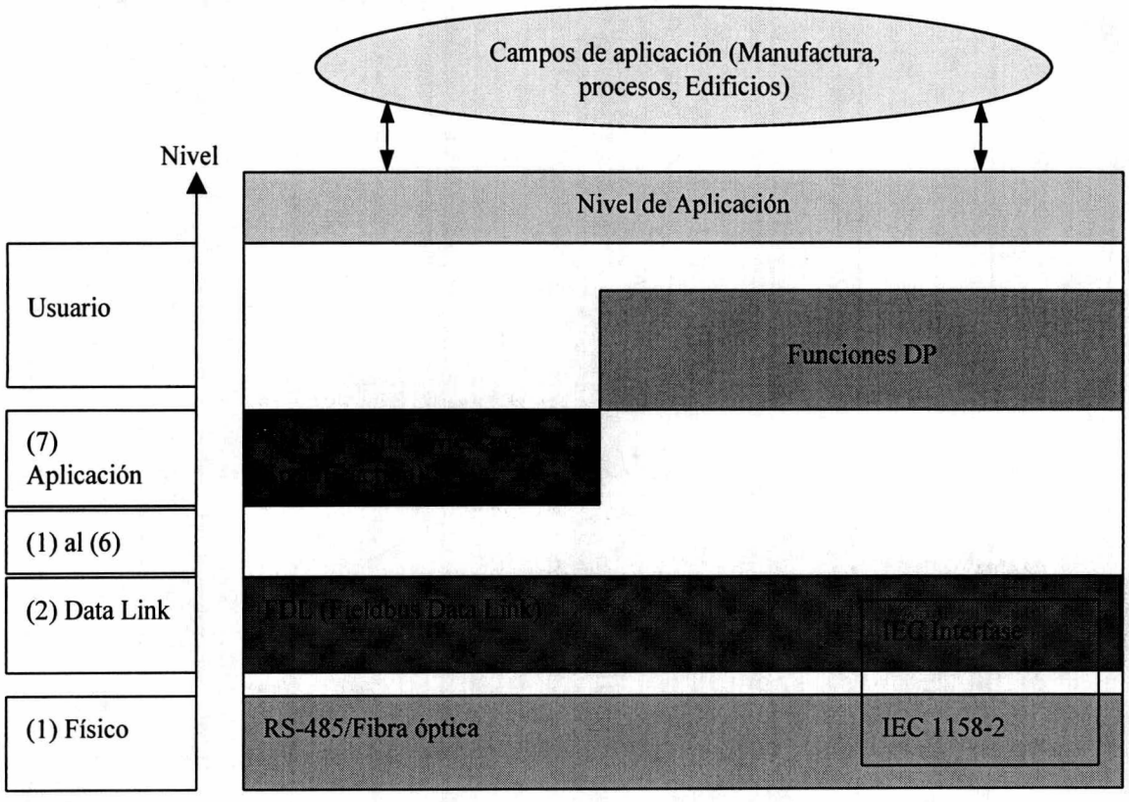


Figura 4.2 Arquitectura de PROFIBUS

4.2 NIVEL FÍSICO EN PROFIBUS

Este nivel es el encargado de seleccionar las tecnologías de transmisión. También establece las características generales que requiere el bus, como: confiabilidad de transmisión, larga distancias y altas velocidades. Se deben tener en cuenta las condiciones del área de trabajo como ruido, transmitir datos y energía por un mismo cable.

Los medios de transmisión disponibles para PROFIBUS son:

- RS-485 para aplicaciones de automatización de la manufactura. El medio de comunicación suele ser un simple par diferencial, previsto para comunicación semiduplex.
- IEC 1158-2 para automatización de procesos
- Fibra óptica para evitar interferencia en distancias grandes

4.2.1 ELEMENTOS DEL BUS

Los elementos esenciales del bus son los nodos, existen dos tipos de nodos, que son: activos que serían los maestros que tienen el control del bus, pasivos que únicamente pueden actuar como esclavos, solo se pueden comunicar con sus maestros y cuando estos lo solicitan.
Otro elemento importante es el repetidor que se utiliza para ampliar y ramificar la red.

4.2.2 TOPOLOGÍA

La topología puede ser un bus línea o árbol, en el que los repetidores constituyen el nodo de partida de una expansión del bus.
El número máximo de nodos conectados a cada tramo del bus, sin repetidores es de 32. Cuando se usan repetidores, estos se cuentan como nodos. Con ramificaciones el número máximo de estaciones de 126, de los cuales un máximo de 32 pueden ser nodos activos. La figura 4.3 muestra esta estructura.

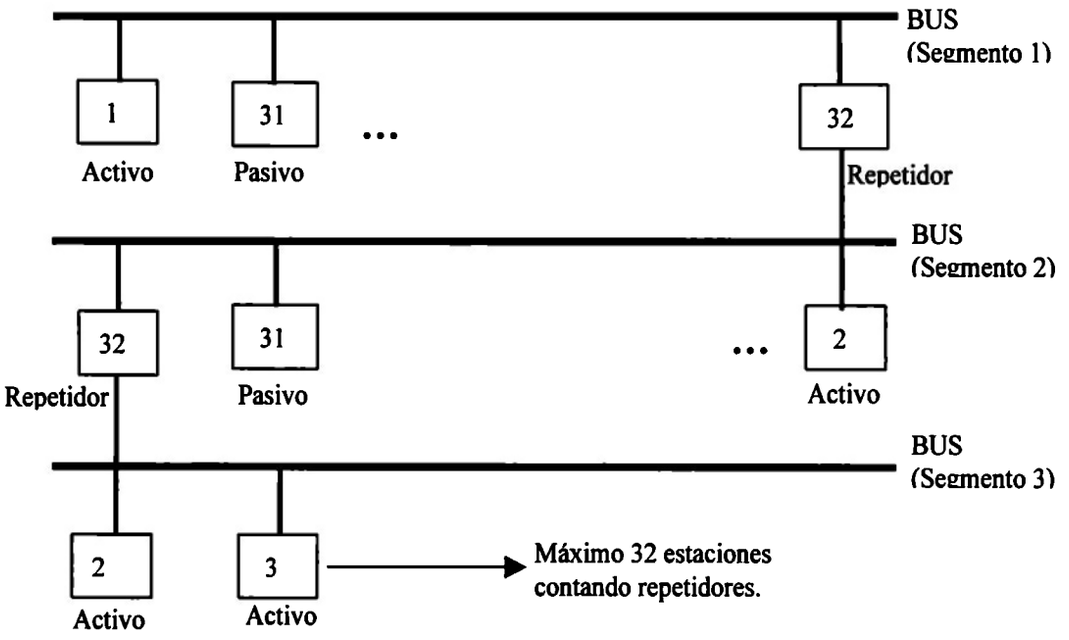


Figura 4.3 Topología de PROFIBUS [8].

4.2.3 ESTRUCTURA LÓGICA

La estructura lógica es combinada, las estaciones activas o maestras implementan un anillo lógico, como lo muestra la figura 4.1, los maestros se rotan el token o control de la red de acuerdo a los parámetros establecidos por el protocolo.

Esta estructura admite la posibilidad de que solo haya un maestro en la red, con lo cual se convierte en una estructura maestro / esclavo.

4.2.4 RS-485

En el capitulo 2 se hablo de RS-485, en este capitulo se profundizara más sobre este tema.

RS-485 es el medio de transmisión usado más frecuente en PROFIBUS. Este medio se utiliza en áreas donde se necesitan altas velocidades de transmisión.

Algunas de las ventajas que presenta RS-485 con relación a RS-232 son:

1. Bajo costo: Los drivers (transmisores) y receivers (receptores) utilizados son económicos y solo requieren $\pm$ 5 V.
2. Habilidad de red: Se puede utilizar para múltiples dispositivos en la red a diferencia de RS-232 que solo se usa para enlace punto a punto. La estructura del bus permite

adicionar o quitar estaciones o hacer cambios sin necesidad de parar las otras estaciones.

3. Grandes distancias: Se pueden alcanzar distancias de hasta 1200m, RS-232 soporta máximo 30 metros.
4. Rapidez: Soporta hasta 10 Mb/s. Se debe seleccionar la misma velocidad de transmisión para todos los dispositivos de la red.

La principal razón por la que RS-485 puede transmitir a grandes distancias es por el uso de líneas balanceadas o diferenciales. Cada señal tiene un par trenzado, en donde el voltaje de una línea es el complemento de la otra. El receptor responde de acuerdo a la diferencia entre estos voltajes. Una gran ventaja de las líneas diferenciales es su inmunidad al ruido, en la figura 4.4 se muestra esa estructura.

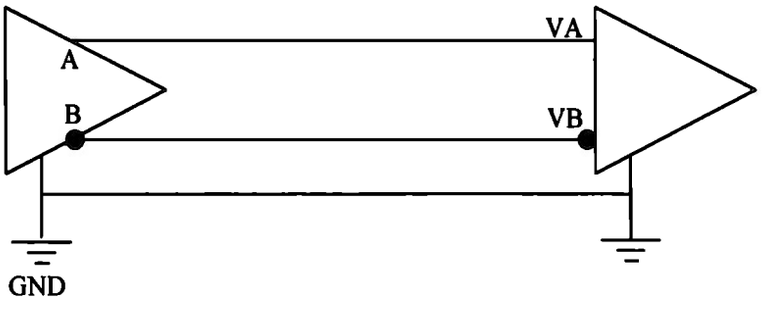


Figura 4.4 Líneas diferenciales

El RS-485 designa dos líneas en un par diferencial A y B. El driver y el receiver toman un uno lógico cuando A > B y un cero lógico cuando A < B.

Las líneas diferenciales son rápidas por que las dos señales son casi iguales, con corrientes opuestas. Estos reduce el ruido ya que los ruidos de voltajes son casi iguales en ambas líneas y un ruido que este subiendo es cancelado por un voltaje opuesto de la otra línea.

Cuando se trabaja con más de dos estaciones se debe hacer la transmisión en modo semiduplex (Solo se puede enviar o recibir pero no las dos cosas simultáneamente), por lo tanto se debe controlar cuando se transmite y cuando se recibe, esto lo hace con un bit de control que trae el chip para la interfase RS-485.

El bus debe ser terminado e iniciado por un terminador activo para evitar errores de transmisión, en la figura 2.9 se puede ver claramente como se hace una conexión en red y como quedan los terminadores que serían resistencias de 120 Ω.

Los tipos de conectores usados son: conector D-Sub de 9 pines para IP 20M12 (preferido para RS-485), HAN-BRID, Conector híbrido de Siemens para IP65/67.

La longitud del cable de transmisión depende de las velocidades requeridas, en la tabla 4.1 se muestran diferentes velocidades para diferentes longitudes. Las especificaciones son basadas sobre el cable Tipo-A que tiene los siguientes parámetros:

- Impedancia: 135 a 165 W
- Capacitancia: <30 pf/m
- Resistencia: 110 W/km
- Calibre: 0.64 mm
- Área conductora: >0.34 mm^2

Kbits/s	9.6	19.2	93.75	187.5	500	1200	1500
Longitud(m)	1200	1200	1200	1000	400	200	100

Tabla 4.1 Longitudes para diferentes velocidades del cable Tipo-A

4.2.4.1 Chips para RS-485

Existen muchos chips para implementar la interfase RS-485. Algunos chips tienen una velocidad de transmisión máxima de 10 Mbps. Otros permiten más nodos en la red, protección contra descargas eléctricas, circuitos de control de fallas, y otras características. Algunos de estos chips son: MAX232, SN75176, DS3695, LTC485.

En la figura 4.5 se muestra la estructura de estos chips, en donde:

Ro = Salida de recepción, Si A > B entonces Ro = 1, Si A < B entonces Ro = 0.

RE = Habilita salida de recepción si RE = 0, si Re = 1, Ro es alta impedancia.

DE = Habilita salida de transmisión si DE = 1, si DE = 0, alta impedancia.

DI = Entrada de transmisión (dato de transmisión).

B , A = Líneas diferenciales.

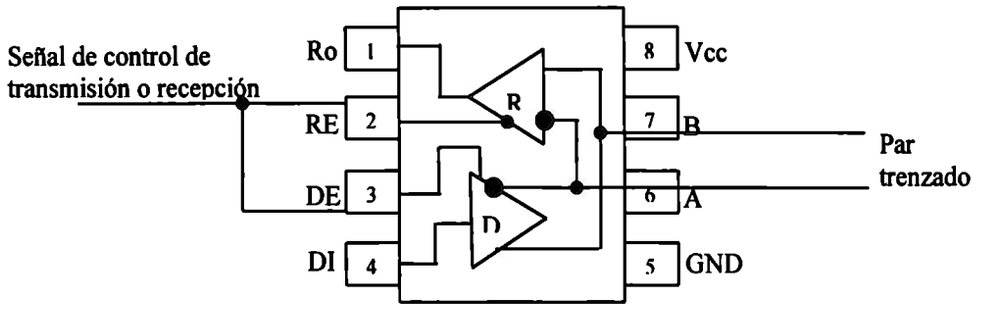


Figura 4.5 Estructura del chip para interfaz RS-485

4.2.4.2 Interfase entre RS-232 y RS-485

Para conectar el computador personal al la red PROFIBUS, se debe hacer una interfaz que convierta las señales RS-232 a RS-485 y viceversa, ya que los computadores personales normalmente solo traen la interfase RS-232.

La figura 4.6 muestra un esquema para una posible interfaz, en este esquema se usan 3 líneas de RS-232, estas líneas son TxD (Transmisión), RxD (Recepción) y RTS (Control). Un chip MAX233 convierte las señales de RS-232 a los niveles de TTL y las señales TTL se conectan a un 75176 que provee la interfase RS-485.

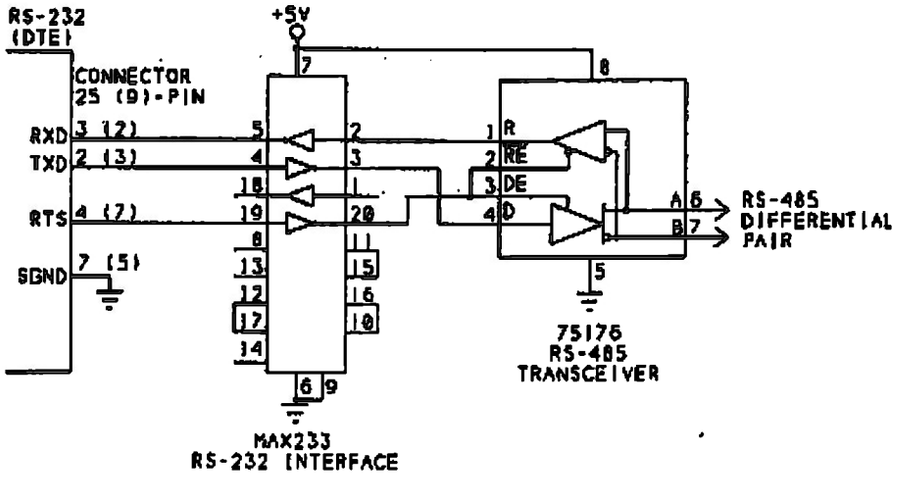


Figura 4.6. Interfaz RS-232 – RS-485

4.2.5 IEC 1158-2

Se usa para procesos industriales en transmisiones asincrónicas con una velocidad de transmisión definida de 31.25 kbits/s. Cumple con importantes requerimientos de seguridad, especialmente en industrias químicas y petroquímicas [9].

Los requerimientos son definidos por FISCO (Fieldbus Intrinsacally Safe Concept) desarrollado en Alemania por PTB y hoy es reconocido como modelo básico para Fieldbus in áreas peligrosas.

La transmisión de acuerdo con IEC 1158-2 y FISCO, esta basada en los siguientes principios [9]:

- Cada segmento tiene solo una fuente de poder
- No hay fuente de poder alimentando cuando una estación esta enviando
- Todos los dispositivos consumen una corriente constante básica
- Son permitidas topologías en bus, árbol y estrella

En estado estable cada estación consume una corriente básica de 10 mA. Con la potencia del bus, esta corriente sirve para entregar energía a los dispositivos.

Para cambiar de medio RS-485 a IEC 1158-2 y viceversa se utilizan convertidores que son transparentes desde el punto de vista del protocolo y si se utiliza un convertidor la velocidad de transmisión de RS-485 será restringida a un máximo de 93.75 kbits/s.

El número de estaciones que pueden ser conectadas a un segmento es de 32.

La tabla 4.2 muestra las características principales de la transmisión IEC 1158.2.

Transmisión de datos	Digital, sincrónico, codificación Manchester
Velocidad de transmisión	31.25 kbits/s
Seguridad de datos	Delimitadores de inicio y final
Cable	Dos pares de cables protegidos
Potencia remota	Opcional, por medio de las líneas de datos
Protección de explosión	Encapsulamiento (EEx d/m/p/q)
Topología	bus, árbol y estrella
Número de estaciones	32 por segmento, máximo de 126
Repetidores	Se puede expandir con 4 repetidores

Tabla 4.2 Características de IEC 1158-2

El tipo de cable usado en la transmisión IEC 1158-2 debe cumplir las siguientes características:

- Diseño: Par trenzado
- Área de conducción: 0.8 mm^2 (AWG 18)
- Resistencia: 44 Ohm/km
- Impedancia a 31.25 kHz: 100 Ohm $\pm$20%
- Atenuación a 39 Khz: 3 dB/km
- Capacitancía: 2 nF/km

4.3 MAC (MEDIUM ACCESS CONTROL)

El MAC (Control de acceso al medio) representa el nivel 2 del modelo OSI, este nivel define cómo los usuarios transmiten sus datos en serie por un mismo medio de transmisión, incluye seguridad de datos, manejo de los protocolos de transmisión y telegramas. En PROFIBUS, este nivel es llamado Fieldbus Data Link (FDL) que se encarga de establecer el orden de circulación del testigo una vez inicializado el bus, adjudicando el testigo en el arranque, en caso de pérdida del mismo, o en caso de adición o eliminación de estaciones activas. El MAC (Medium Access Control) especifica el procedimiento para que a una estación se le permita transmitir datos y se encarga básicamente de dos funciones que son:

- Durante la comunicación entre estaciones maestras, debe asegurar que cada una de las estaciones tenga suficiente tiempo para desarrollar sus tareas de comunicación con un tiempo preciso y ya definido.
- Para comunicación entre estaciones maestras y esclavas, el tiempo de transmisión necesita ser implementado tan rápido y simple como sea posible.

Para estas tareas existen dos procedimientos que son:

- El token passing que asegura que el derecho de acceso al bus (el token) sea asignado a cada maestro con un tiempo preciso ya definido, el mensaje token (derecho de acceso al bus), es pasado a cada maestro cíclicamente, el token passing es usado solamente entre maestros [9].
- El otro procedimiento es el maestro-esclavo que permite al maestro que actualmente posee el token, acceder las estaciones esclavas, esto habilita al maestro a enviar y recibir mensajes con los esclavos.

Un token anillo significa la organización de estaciones maestras en forma de un anillo lógico. El token es pasado de un maestro a otro con una secuencia ya predefinida, cuando una estación maestra recibe el token, esta puede desarrollar el rol de maestro por un periodo de tiempo y comunicarse con todos sus esclavos. Al terminar el tiempo establecido, esta estación cede el token a la siguiente estación maestra.

La tarea del MAC de la estación activa es detectar el asignamiento lógico del token de acuerdo a un orden ya predefinido. Durante la operación cuando una estación falla o es desactivada, esta estación debe ser quitada del anillo y cuando se activa debe ser adicionada al anillo. Además el MAC asegura que el token es pasado de una estación maestra a otra de acuerdo al orden establecido.
Otra tarea importante de este nivel es la seguridad en la transmisión, donde cada parte del mensaje tienen un bit de inicio y un bit de parada, un bit de paridad y un byte de chequeo, Todos tienen un Hamming Distance = 4.

El PROFIBUS opera en dos modos que son broadcast y multicast que vamos a definir:
Comunicación broadcast: Significa que una estación maestra puede enviar un mensaje sin respuesta a las demás estaciones maestras o esclavas.
Comunicación multicast: Significa que una estación maestra envía un mensaje sin respuesta a un predeterminado grupo de estaciones maestras o esclavas.

A continuación se dará una descripción mas detallada de MAC y de la norma token passing IEEE 802.4 [6], en la tabla 4.3 se describen los términos más utilizados por la norma.

4.3.1 ESTRUCTURA INTERNA DE MAC

El MAC desarrolla muchas funciones que están encapsuladas y relacionadas entre sí. Tiene 5 "maquinas" lógicas asincrónicas que desarrollan funciones del MAC [6], la figura 4.7 muestra esta arquitectura.
A continuación se describirá cada una de estas maquinas:

4.3.1.1 Maquina de Interfaz (IFM)

Esta maquina actúa como una interfaz entre el LLC y el MAC, entre el administrador de estaciones y el MAC. Interpreta todos los mensajes MA_DATA. Desarrolla el reconocimiento de direcciones, aceptando solo las direcciones que van dirigidas a esta estación.

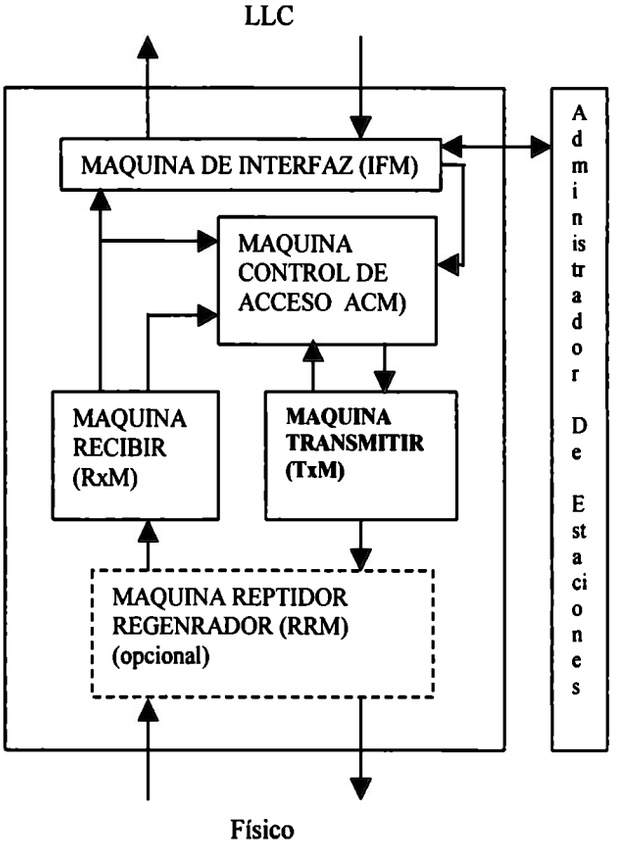


Figura 4.7 Estructura de MAC [6].

4.3.1.2 Maquina de control de acceso (ACM)

Esta maquina es la más critica y las más compleja, es el mecanismo principal para el método de acceso al bus. ACM coopera con las otras ACM de todas las estaciones en el bus, su función es manejar el token para controlar el acceso a la transmisión en el bus compartido. El ACM es el responsable de la iniciación y mantenimiento del anillo lógico, adición y eliminación de estaciones en la red. También tiene la responsabilidad de detectar fallas y si es posible recobrarse de las fallas en la red token bus.

4.3.1.3 Maquina recibir (RxM)

Esta maquina es la encargada de aceptar los símbolos del medio físico, ensambla los símbolos recibidos en la forma del frame valido para la norma validándolo y pasa al ACM y IFM. También es función de RxM reconocer los delimitadores de inicio y final del frame (SD y ED), revisar la secuencia de control del frame (FCS) y validar la estructura del frame.

4.3.1.4 Maquina enviar (TxM)

Esta maquina generalmente acepta un frame del ACM y lo transmite al medio físico como una secuencia de símbolos. El TxM también esta encargado de anexar la secuencia de control de frame (FCS) , el delimitador inicial y final (SD y ED).

4.3.1.5 Maquina repetidor regenerador (RRM)

Esta maquina es un componente opcional del MAC y esta presente solo en estaciones repetidoras especiales.

4.3.2 ESPECIFICACIÓN DE TOKEN PASSING (IEEE 802.4)

Como se ha dicho anteriormente, PROFIBUS implemente un token ring lógico con los maestros de la red, es importante aclarar que es un anillo solo lógico ya que token ring esta diseñado para redes con topología en anillo y PROFIBUS implementa una topología en bus. El mensaje y la implantación del protocolo es especificado por token passing bus.

Token passing esta especificado en el estándar ANSI/IEEE 802.4 para redes de área local [7]. Este estándar define el formato y protocolo usado por el control de acceso al medio (MAC), el nivel físico y el significado del mensaje del token passing bus

4.3.2.1 Resumen del token passing

El token (testigo) controla el derecho de acceso al medio físico de la red; la estación que tiene el token en determinado momento es la que tiene derecho de usar el medio enviando y recibiendo mensajes, las únicas estaciones que pueden tener el token son las estaciones maestras, las estaciones esclavas se limitan a enviar la información que le solicita el maestro al que pertenece.
El token es pasado a todas las estaciones maestras de la red, esto lo hace en forma de anillo lógico, donde cada estación maestra es identificada por un índice consecutivo, y el token es pasado de una estación maestra a otra en orden decreciente, como lo muestra la figura 4.1.

El anillo posee funciones para inicialización, recobrar el token en caso de perdida, adicionar nuevas estaciones, y en general para mantener el buen funcionamiento de la red.

Es importante tener claro que la conexión física no necesariamente deben ser secuencial, no importa el orden como esten conectadas las estaciones, lo importante es el identificador de cada estación.

En este tipo de topología todas las estaciones reciben el token o frame enviado, pero solo lo lee la o las estaciones a las que va dirigido éste.

El MAC provee acceso secuencial al bus por pasar el control del medio de una estación a otra en orden secuencial, MAC determina cuando la estación debe pasar el token a la siguiente estación.

4.3.2.2 Funciones generales de la norma

Las funcione generales de la norma son:

- Encapsulamiento del frame y del token
- Timer para perdida de token
- Inicialización distribuida
- Timer para tiempo con el token
- Limite de buffer para datos

- Reconocimiento de las direcciones
- Generar y revisar FCS (secuencia de control de frame)
- Reconocimiento de token valido
- Adicción de nueva estación al anillo
- Quitar estación del anillo
- Recobrarse de errores

4.3.2.3 Especificación del formato del frame

En esta sección se definirá el formato del frame usado en MAC, el termino frame es usado entendiendo que es una unidad de dato del protocolo (PDU). El formato del frame se muestra a continuación:

Donde:

SD = Delimitador de inicio (1 octeto)

FC = Control del frame (1 octeto)

DA = Dirección destino (2 a 6 octetos)

SA = Dirección fuente (2 a 6 octetos)

INFO = Información (0 o más octetos)

FCS = Secuencia de control del frame (4 octetos)

ED = Delimitador final (1 octeto)

4.3.2.3.1 Secuencia de aborto

Esta secuencia es usada cuando se desea terminar una transacción prematuramente.

4.3.2.3.2 Delimitador de Inicio (SD)

Donde:

N = Sin datos

0 = Cero lógico

4.3.2.3.3 Control del frame (FC)

Tiene dos posibles formatos. El FC determina que clase de frame esta siendo enviado, que puede ser:

1. Control MAC
2. Dato LLC
3. Administración de datos de la estación
4. Propósitos especiales, para usos futuros.

Formato 1 de FC

0 0 C C C C C C

Donde:

CCCCCC = Tipo de frame MAC, que puede ser:

C	C	C	C	C	C	Definición
3	4	5	6	7	8	BIT
0	0	0	0	0	0	Pedir el token
0	0	0	0	0	1	Solicitud sucesor 1 (tiene 1 ventana de respuesta)
0	0	0	0	1	0	Solicitud sucesor 3 (tiene 2 ventanas de respuesta)
0	0	0	0	1	1	¿Quién sigue?
0	0	0	1	0	0	Resolver competencia (2 ventanas de respuesta)
0	0	1	0	0	0	Token
0	0	1	1	0	0	Poner sucesor

Las ventanas de respuesta se tiempos de espera, cada ventana es un tiempo especificado.

Formato 2 de FC

FF MMM PPP CC

Donde:

FF = Tipo de frame

12 -> bits

01 = Dato LLC

10 = Administración de datos de la estación

11 = Propósitos especiales, reservado para futuras modificaciones

MMM = Acción de MAC

345 -> bits

000 = Solicitar sin respuesta

001 = Solicitar con respuesta

010 = Respuesta

PPP = Prioridad

678 -> bits

111 = Prioridad más alta

110

101

100

011

010

001

000 = Prioridad más baja

4.3.2.3.4 Dirección destino (DA)

Es la dirección a la que va dirigido el frame, su estructura es como sigue:

I/G dirección

Donde:

I/G = Indica si va dirigido a una o a varias estaciones, 0 = individual y 1 = grupo.

Dirección = La o las direcciones destino.

4.3.2.3.5 Dirección fuente (SA)

Dirección de la estación maestra que envía el mensaje, el formato es igual a DA, el valor del bit I/G siempre es igual a 0.

4.3.2.3.6 Información (INFO)

Dependiendo de los bits de control especificados en el FC, el campo INFO puede tener:

- Un dato de administración de MAC que puede ser intercambiado entre diferentes MAC's
- Un valor especifico de uno del los frames de control MAC.

4.3.2.3.7 Secuencia de control del frame (FCS)

Se refiere al control del frame de datos.

4.3.2.3.8 Delimitador final (ED)

Indica el final del frame, tiene la siguiente estructura:

NN 1 NN 1 I E

Donde:

N = No dato

1 = Uno lógico

I = Bit Intermedio, 1 = se va a transmitir más, 0 = fin de transmisión

E = Bit de error, 1 = No hubo error, 0 = Hubo error

4.3.2.4 Operación básica de MAC

La operación en estado estable (Condición de la red en donde el anillo lógico ha sido estabilizado y no se presentan condiciones de error) simplemente requiere el envío del token a una estación especifica.

Otra tarea esencial y más difícil es estabilizar el anillo lógico (Inicializar y reinicializar en caso de errores) y mantener el anillo lógico (permitiendo a las estaciones entrar y salir del anillo sin causar problemas en el funcionamiento de la red).

Cada estación maestra participante conoce la dirección lógica de su predecesor (estación que tenia el token antes que él, referida como estación previa (PS)) , conoce la dirección lógica de su sucesor (estación a la que le enviara el token (NS)) y por supuesto conoce su propia dirección (TS).

A continuación se definirán algunos parámetros y términos importantes para poder manejar el protocolo, por compatibilidad con la norma se trabajar con los términos en ingles.

4.3.2.4.1 Slot time

Se refiere al tiempo máximo que una estación necesita para una respuesta de otra estación, este tiempo lo debe saber la estación antes de enviar un frame a la red. Todas las estaciones en la red deben usar el mismo valor de slot_time para que opere correctamente el MAC.

4.3.2.4.2 Derecho de transmisión

El token (derecho para transmitir) es pasado de una estación a otra en orden numérico descendente. Cuando una estación detecta un token frame direccionado a ésta, la estación toma el token y puede comenzar a transmitir frames de datos, Cuando termina de transmitir pasa el token a la próxima estación en el anillo lógico.

4.3.2.4.3 Token passing

Después de que cada estación ha completado la transmisión de frames de datos y ha completado otras funciones de mantenimiento, la estación pasa el token al sucesor enviándole un frame de token MAC_control. Después de enviar este token, la estación queda en espera de un frame enviado por la otra estación receptora notificándole que ya tomo el token. Si la estación no recibe el frame de respuesta, intenta nuevamente y si vuelve a fallar, debe notificar un problema en el estado de la red y envía un frame Who_follows con la dirección de la estación que no respondió, cada estación compara este valor con el valor de su propio predecesor y si coinciden, esta estación responde con un frame Set_successor.

Si cuando la estación envía el frame de Who_follows y no responde ninguna estación , entonces vuelve a intentar y si no responden entonces implementa otra estrategia para reestabilizar el anillo lógico enviando un frame solicit_successor_2 con su propia dirección, esto se discutirá más adelante.

Si todas las solicitudes de sucesor fallan, esto significa que hay un problema grave en la red (todas las estaciones fallaron, el cable esta roto o la misma estación puede estar fallando por que no puede recibir los mensajes) y la estación debe enviar una alarma de mantenimiento de la red.

4.3.2.4.4 Ventanas de respuesta

Las ventanas de respuesta se usan cuando nuevas estaciones son adicionadas al anillo lógico. Una ventana de respuesta es un intervalo de tiempo controlado (igual al slot_time) después de la transmisión de un frame MAC_control en el que la estación hace una pausa y espera respuesta. Si la estación detecta que le están enviando datos durante la ventana de respuesta, la estación continua capturando los datos hasta que le terminen de enviar o hasta que el tiempo de la ventana de respuesta termine. En otras palabras las ventanas de respuesta definen el tiempo durante el cual una estación esperará la respuesta del frame.

Los dos tipos de frames son: solicit_successor_1 y solicit_successor_2 que dan la libertad de que nuevas estaciones puedan entrar al anillo lógico. Este frame indica un rango de dirección en los

campos DA y SA, las estaciones cuya dirección éste en este rango y deseen entrar al anillo lógico son las que pueden responder al frame.

La estación que envía el frame solicit_successor esperan respuesta en el intervalo de ventana de respuesta, si le responde una estación, entonces toma la dirección de esta estación y la pone como su nuevo sucesor y le envía el token a éste.

Puede ocurrir que dos estaciones simultáneamente respondan al frame solicit_successor, esto se soluciona poniendo el rango de tal manera que solo una estación pueda responder.

El solicit_successor_1 tiene una ventana de respuesta, éste es enviado cuando las direcciones de las estaciones sucesoras son menores que la dirección de la estación que esta haciendo la petición de sucesor.

4.3.2.4.5 Tiempo de rotación del token

Determina el máximo tiempo que una estación espera para ganar acceso a la red, cuando hay muchas estaciones tratando de adicionar nuevas estaciones al anillo, hay un tiempo de espera grande. Si este tiempo sobrepasa el tiempo establecido en le parámetro (max_ring_maintenance_rotation_time) la estación no desarrolla el procedimiento de solicitud de sucesor.

4.3.2.4.6 Iniciación

La iniciación es un caso especial de adicionar nuevas estaciones, éste es activado por la expiración de un timer de inactividad (bus_idle) en una estación. Cuando este timer expira, la estación envía un frame claim_token (solicitud de token). Se debe asumir que muchas estaciones podrían tratar de inicializar el anillo, esto es resuelto por establecer un orden para la inicialización.

Una estación puede salir del anillo lógico no respondiendo a un frame que le es enviado.

4.3.2.5 Estados de la maquina ACM (Maquina de control de acceso)

El acceso lógico al medio de una estación es descrito como una maquina computacional que tiene una secuencia a través de pasos llamados estados (etapa en la cual se encuentra en un momento determinado). Token passing cuenta con 11 estados los cuales se definirán a continuación.

4.3.2.5.1 Estado Offline (fuera de línea)

En este estado la ACM esta siguiendo el power-on (encendido) o la detección por la subcapa MAC de muchas condiciones de falla. Después de power-on, una estación se examina ella misma y no transmite al medio. Después de haber completado los procedimientos de power-on, la estación esta en estado Offline hasta que inicializa todos los parámetros internos necesarios y esta lista para ir al estado online.

4.3.2.5.2 Estado Idle (ocio)

En este estado la estación esta escuchando el medio y no transmite. Si es recibido un frame MAC_frame para que la estación deba tomar alguna acción, esta cambia al estado apropiado.

Si la estación tiene un periodo largo de tiempo (Un múltiplo definido del slot_time) sin escuchar una actividad del medio, ésta puede tratar de recobrar el anillo lógico, entra en estado claim_token (solicitud de token) y reinicializa el anillo.

4.3.2.5.3 Estado Demand In (reclamar)

Se entra a este estado desde el estado Idle si un frame de solicitud de sucesor es recibido por la estación y ésta quiere entrar al anillo. En este estado la estación envía un frame set_successor para ser sucesor y entra al estado Demand_Delay para esperar respuesta.

Si la estación intenta responder en la primera ventana de respuesta a un frame solicit_successor o a un frame who_follows, la estación entra al estado Demand_in desde el estado Idle inmediatamente, transmite un frame Set_successor y va al estado Demand-delay.

Mientras la estación esta en el estado Demand_in, si recibe una transmisión, la estación asume que otra estación con un número de dirección más grande esta solicitando el token y debe regresar al estado idle.

4.3.2.5.4 Estado Demand Delay

En este estado la estación entra después de haber enviado un frame Set_successor en el estado Demand_In. En este estado la estación espera a que le envíen:

1. Un token desde la estación que solicita sucesor, diciendo que recibió el frame y que le da el derecho de poseer el token, la estación toma el token y va al estado Use_token e inicia la transmisión.
2. Un frame Resolve_contection desde la estación que solicito el sucesor, diciendo que todas las estaciones que aspiran ser sucesoras deben desarrollar otro paso de la contienda para resolver el proceso de selección. Las estaciones antes de responder activan un timer y retornan al estado Demand_in y escuchan a las otras estaciones que quieren ser sucesoras, si no se escucha ninguna otra estación y el timer expira entonces la estación envía un frame Set_successor a la estación que esta solicitando sucesor.

La contienda de todas las estaciones por ser sucesoras es resuelta utilizando delays (tiempos de espera), donde cada estación demandante tiene un timer en el estado Demand_In y cuando el timer expira ésta envía el frame Set_successor. El intervalo del delay es asignado en base de la dirección de cada estación (que debe ser única) usando los 2 bits más significativos de la dirección y este valor se multiplica por el Slot_time, la primera vez que inicia la contienda, la estación usa los primeros dos bits, la segunda los próximos dos bits y así sucesivamente. Si dos estaciones tienen igual los 2 bits, ocurriría que ambas transmitan a la vez y esto provocaría un frame erróneo, cuando la estación que pide el sucesor recibe el frame, notará que no es un frame valido y volverá a iniciar la contienda.

Cuando muchas estaciones quieren ser sucesoras, la estación que busca el sucesor selecciona la que tenga la dirección más grande que vendría siendo la primera que responda.

Si la estación no escucha nada o recibe algún frame diferente a los mencionados a 1,2 o 3, la estación debe dejar el estado Demand_delay y retorna al estado idle.

4.3.2.5.5. Estado Claim Token (Reclamar el token)

Se entra a este estado desde el estado Idle después de que el timer de inactividad expira (periodo de tiempo en el que la estación no escucha nada). En este estado la estación intenta inicializar o reinicializar el anillo lógico enviando un frame Claim_token.

Para resolver el problema cuando hay muchas estaciones enviando un frame Claim_token, cada estación espera por un tiempo Slot_time después de haber enviado el frame Claim_token, y entonces monitorea el medio como se ha descrito anteriormente. Si después de este periodo de tiempo no recibe respuesta envía otro frame Claim_token.

Si la estación envía un frame Claim_token con Max_pass_count (La mitad del número de bits de la dirección de la estación más uno) sin escuchar las otras estaciones, la estación ha reclamado el token exitosamente y pasa al estado Use_token.

4.3.2.5.6 Estado Use Token (Usar el token)

Se entra a este estado justo después de haber recibido o reclamado el token. Este es el estado en que la estación puede enviar frames de datos.

Cuando la estación entra a este estado inicia el timer Token_holding, que establece el tiempo que la estación tendrá el token. El valor inicial de este timer es uno de los parámetros de iniciación del sistema.

Cuando expira el timer y una transmisión en progreso es completada, la estación entra al estado Check_access_class.

Cuando una estación envía un frame de datos, ésta activa el timer de ventana de respuesta con un valor de 3 Slot_time y entra al estado Await_IFM_response .

4.3.2.5.7 Estado Await IFM Response (Espera de respuesta de la maquina de interfaz)

Se entra a este estado cuando un frame de datos es enviado. Éste estado espera la respuesta del frame enviado.

Si el frame enviado en el estado Use_token fue un frame Request_with_no_response (frame sin respuesta), no se espera ninguna respuesta, se entra al estado Use_token para enviar otros frames y revisar si se activo el timer Token_holding.

Si el frame enviado era un frame Resques_whit_response_frame (frame con respuesta), la estación espera en este estado por alguno de las siguientes acciones:

1. Un frame de respuesta, la estación entra al estado Use_token para enviar otros frames o para revisar el timer. El estado Await_IFM_response pasa el frame recibido a la función encargada de procesarlo.
2. Otro frame valido pero no el de respuesta, acá ha ocurrido un error, la estación retorna al estado Idle y procesa el frame recibido.
3. Expiración del timer, si expira el timer de espera del frame, la estación vuelve a enviar el frame Resques_whit_response_frame. Si no recibe respuesta, repite el número de veces especificado por el parámetro Max_retry_limit_parameter y si no responde es abanada la solicitud. El IFM notifica este problema, se entra al estado Use_token para enviar otros frames o revisar si expiro el timer token_holding.

4.3.2.5.8 Estado Check Access Class (Revisar clases de acceso)

Este estado controla la transmisión de frames para diferentes clases de acceso. Si no son implementadas opciones de prioridad todos los frames son considerados con prioridad alta y el estado check_access_class sirve para controlar la entrada al token passing.

4.3.2.5.9 Estado Pass Token (Pasar el token)

Es el estado en que la estación intenta pasar el token a su sucesor.

Cuando el valor Inter_solicit_count es cero y a la vez permanece sobre el Ring_maintenance_timer, la estación permite a nuevas estaciones entrar al anillo lógico antes de pasar el token.

Si la dirección del sucesor (NS) es reconocida, la estación envía un frame pasándole el token y si recibe respuesta la estación ha completado sus obligaciones de pasar el token.

Si la dirección del sucesor (NS) no es reconocida, la estación que tiene el token hace las respectivas actividades de busca de sucesor. Este procedimiento ya fue descrito anteriormente.

4.3.2.5.10 Estado Check Token Pass (Revisar el paso del token)

En este estado la estación espera la reacción de la estación a la que le paso el token. La estación envía el token, espera un Slot_time para que la estación que recibió el token le envié un frame confirmándole. Si el frame es escuchado la estación asume que el token fue tomado correctamente, y el frame es procesado como si hubiera sido recibido en el estado Idle.

Si no es escuchado nada en un Slot_time, la estación retorna al estado Pass_token y trata de enviarlo nuevamente o implementa otra estrategia.

4.3.2.5.11 Estado Await Response (Esperar respuesta)

En este estado la estación intenta establecer un sucesor hasta que es recibido un frame Set_successor. Se entra a este estado desde el estado Pass_token, la estación determina el tiempo para la ventana de respuesta. La estación espera en el estado Await_response el tiempo establecido en la ventana de respuesta. Si no escucha nada en tiempo que la ventana esta abierta, la estación va al estado Pass_token, e implementa alguna estrategia.

Si es recibido el frame Set_successor, la estación espera que el tiempo de la ventana de respuesta sea completado. Cuando el tiempo termina la estación entra al estado Pass_token y envía el token al nuevo sucesor.

Si se recibe un frame diferente, la estación deja el token y entra al estado Idle.

Si le llega un frame con ruido o defectuoso, la estación entra a un ciclo y envía un frame Resolve_contention y espera respuesta. El ciclo repite un máximo de Max_pass_count veces.

4.4 PROFUBUS DP (DECENTRALIZED PERIPHERY)

PROFIBUS DP esta optimizado para velocidad, eficiencia y bajos costos de conexión y es diseñado especialmente para comunicación entre sistemas de automatización y periféricos distribuidos. PROFIBUS DP es usado para la comunicación entre maestros y esclavos.. En DP los datos son transmitidos con el servicio SRD (Send And Request Data With Reply) del nivel 2.

En DP existen tres niveles de diagnostico capaces de localizar las fallas, estos mensajes de fallas son almacenados en el maestro del grupo, los niveles son:

1. **Diagnostico a la estación:** Estos mensajes se refieren al estado general de la estación (temperatura, bajo voltaje, etc).
2. **Diagnostico del modulo:** Diagnostico a las entradas y salidas de un modulo de 8 bits.

3. **Diagnostico del canal:** En este caso la causa de la falla es especificada con relación a entradas y salidas individuales, ósea de un bit (canal).

DP permite sistemas monomaster y multimaster, en monomaster existe un solo maestro en el bus, en multimaster pueden existir varios maestros tipo DPM1, donde a cada maestro se le asignan un número de esclavos. Un máximo de 126 estaciones (maestras o esclavas) pueden ser conectadas en un bus. Existen tres tipos de dispositivos en DP que son:

1. DPM1 (DP Master Class 1): DPM1 Se refiere a un controlador central que cíclicamente intercambia información con los esclavos en un ciclo de mensaje ya definido, típicos dispositivos de esta clase son: PLC, PC, Microcontroladores.
2. DPM2 (DP Master Class 2): Los dispositivos de este tipo son: equipos de configuración o operación, equipo de ingeniería, que son usados para mantenimiento y diagnostico, evaluar medidas de valores y parámetros, y revisar el estado de los dispositivos.3.
3. Esclavos: Es un dispositivo periférico (I/O, driver, válvula, sensor) que captura o envía información a los periféricos

La cantidad de información de entrada y salida depende del tipo de dispositivo, es permitido un máximo de 246 bytes de datos de entrada y 246 bytes de datos de salida.

Las entradas y salidas de los esclavos pueden ser leídas por todos los maestros pero solo un maestro asignado puede escribir las salidas.

El DPM1 puede ser controlado localmente o por el bus por el dispositivo de configuración. Hay tres estados principales del DPM1 que son:

1. Stop (Parada): En este estado no hay transmisión de datos entre el DPM1 y los esclavos.
2. Clear: En este estado el DPM1 lee las entradas de información de los esclavos y almacena las salidas en estado a prueba de fallos.
3. Operate (Operar): En este estado el DPM1 esta en la fase de transferencia de datos. En el ciclo de comunicación de datos, las entradas de los esclavos son leídas y las salidas son escritas a los esclavos.

El DPM1 envía cíclicamente su estado a todas las estaciones asignadas a él, esto lo hace usando un comando multicast.

La reacción del sistema a un error durante la fase de transferencia de DPM1 es determinada por el parámetro de configuración, "auto-clear". Si éste parámetro es verdadero, el DPM1 cambia las salidas de todos los esclavos asignados al estado fail-safe (a prueba de fallos) tan pronto como un esclavo no esta listo para transmitir. El DPM1 cambia al estado Clear.

Si el parámetro "auto-clear" es falso, entonces el DPM1 se queda en estado Operate hasta que ocurra un fallo y el usuario puede especificar la reacción del sistema.

4.4.1 TRANSMISIÓN CICLICA DE DATOS ENTRE EL DPM1 Y LOS ESCLAVOS

La transmisión de datos entre el DPM1 y los esclavos asignados a éste, es ejecutada automáticamente por el DPM1 en un orden definido.

La transmisión de datos entre el DPM1 y los esclavos esta dividida en 3 fases que son: parametrización, configuración y transferencia de datos. Antes de que un DP esclavo entre a la fase de transmisión de datos, éste debe pasar por las fases de parametrización y configuración en donde se revisan datos como: Tipo de dispositivo, formato y longitud de la información, número de entradas y salidas que debe trabajar y otros parámetros que el usuario puede configurar.

4.4.2 MODOS SYNC (SINCRONÍA) Y FREEZE (CONGELAR)

El maestro también puede enviar comandos de control a un esclavo, a un grupo de esclavos, o a todos los esclavos simultáneamente. Estos comando de control son transmitidos con comandos multicast que permiten el uso de los modos sync y freeze para el control de eventos de sincronización de los esclavos.

Los esclavos inician el modo sync cuando reciben un sync comando del maestro asignado.

Las salidas de todos los esclavos están normalmente en modo frozen (Congelado). Durante las transmisiones de usuario, los datos de salida son almacenados en los esclavos, pero los estados de salida permanecen sin cambiar. Los datos de salida se envían cuando los esclavos reciben el próximo mensaje de sync. El modo sync es concluido con el comando unsync.

Similarmente un comando de control freeze causa que los esclavos entren al modo freeze. En este modo los estados de las entradas son congelados con el valor actual. Los datos de entrada son actualizados cuando el maestro envía el próximo comando freeze. El modo freeze es concluido con un comando unfreeze.

4.4.3 MECANISMOS DE PROTECCIÓN

Para la seguridad y confiabilidad, DP implementa mecanismos de monitoreo, estos mecanismos de monitoreo son implementados en los maestros y en los esclavos en intervalos específicos, los intervalos son definidos durante la fase de configuración.

El DPM1 monitorea los esclavos con un parámetro que se llama Data_Control_Timer, se usa un control timer para cada eslavo, el tiempo de monitoreo es activado cuando no hay transmisión de datos en el intervalo de monitoreo especificado y el usuario es informado. Cuando se habilita el parámetro de reacción automática a errores (Auto_Clear=True), el DPM1 sale del estado OPERATE, cambia al estado CLEAR y cambia todas las salidas de sus esclavos al estado fail-safe.

Los esclavos usan el watchdog para detectar fallas de la transmisión del maestro, si en el intervalo de tiempo definido con el watchdog no hay comunicación con el maestro, el esclavo automáticamente cambia sus salidas al estado fail-safe.
Además los esclavos cuentan con protección de acceso para las salidas y las entradas asegurando que solo los maestros autorizados tienen derecho de acceso. Para los demás maestros, los esclavos ofrecen una imagen de sus entradas y salidas que pueden ser leídas pero sin derechos de acceso.

4.5 PROFIBUS FMS (NIVEL 7 DE OSI)

FMS (Fieldbus Message Specification) es el nivel de comunicación universal. FMS ofrece muchas funciones sofisticadas para comunicación entre dispositivos inteligentes. FMS es diseñado para comunicación a nivel de célula, en este nivel PLC's, PC y Microcontroladores se comunican unos con otros. En este nivel es más importante la funcionalidad que la velocidad de reacción del sistema. El FMS esta conformado básicamente por: FMS y LLI (Lower Layer Interface). FMS busca la fácil integración del bus de campo con la estructura jerárquica de automatización de la empresa [8].

El FMS permite que aplicaciones distribuidas de procesos sean unificadas en un proceso común usando relaciones de comunicación. Para eso se utilizan los VFD (Virtual Field Device) que son la parte del dispositivo real que es visible para la comunicación, en la figura 4.8 se muestra la relación entre el dispositivo real y los VFD. Los VFD tienen ciertas variables que pueden ser leídas o escritas por las relaciones de comunicación entre los VFD, entre otras están: cantidad, velocidad de operación, etc.

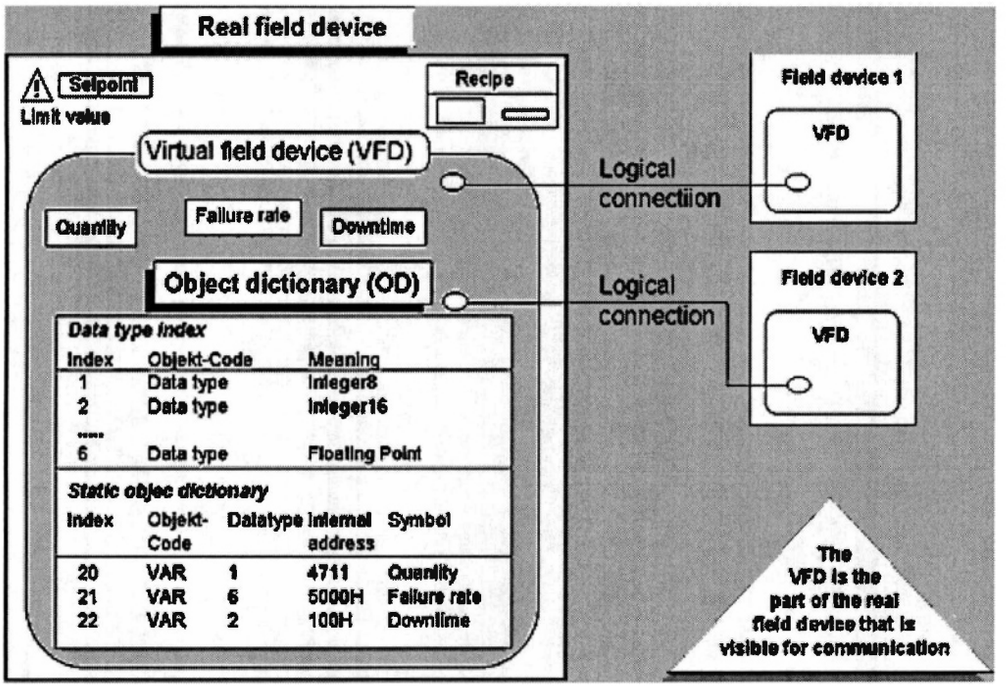


Figura 4.8 VFD (Virtual Field Device) con Diccionario de objetos (OD) [9]

Los VFD están formados por objetos, lo objetos se refieren a las variables que maneja cada uno de los dispositivos de automatización y control.

Cada vez que se adiciona un nuevo dispositivo al bus, el dispositivo debe tener un driver (archivo con toda la información del dispositivo, como nombre, tipo de control, objetos que maneja, características de seguridad, fabricante, velocidad máxima, etc.), estos drivers pueden estar almacenados en un disquete o un CD. El maestro lee estos drivers y crea el dispositivo virtual para ese esclavo, actualiza la bases de datos (VFDs y OD) y establece comunicación con este esclavo para verificar su buen funcionamiento. Posteriormente estará interrogándolo periódicamente y actualizando el estado actual de sus variables.

Todos los objetos de comunicación de un FMS son almacenados en el diccionario de objetos (OD), el diccionario de objetos contiene: descripción, tipo de dato, estructura, relación entre la dirección interna del dispositivo y su designación en el bus (Índice / nombre).

El OD debe ser almacenado en una base de datos relacional, que contendrá toda la información pertinente para su correcto funcionamiento. Se cuenta con servicios que permiten leer, escribir, borrar, generara reportes y en general poder manipular toda esta información.

Los objetos de comunicación se dividen en estáticos y dinámicos:

Los objetos de comunicación estáticos son almacenados en el diccionario de objetos estáticos, estos objetos son configurados una vez y no pueden ser modificados durante la operación. Hay cinco tipos de objetos que son:

1. Variable simple
2. Array ó vector (serie de variables de un mismo tipo)
3. Registro (Serie de variables de diferentes tipos)
4. Dominio
5. Evento

Los objetos de comunicación dinámicos son almacenados en el diccionario de objetos dinámicos, estos objetos pueden ser modificados durante la operación.
El direccionamiento de los objetos es el direccionamiento lógico, en donde el acceso es desarrollado con una dirección corta (Índice) que es un número de longitud 16. Cada objeto tiene un índice único, otra opción es direccionar los objetos por nombre.

Los objetos de comunicación deben ser protegidos contra accesos no autorizados y se deben ceder derechos de acceso, como solo lectura o lectura y escritura.

4.5.1 SERVICIOS DE FMS

Los servicios de FMS son un subconjunto de MMS (Manufacturing Message Specification de MAP, ISO 9506) que han sido optimizados para aplicaciones fieldbus y que ha sido expandidos con funciones para administración de objetos de comunicación y administración de redes [9]. La figura 4.9 muestra estos servicios disponibles para FMS y son:

Administración de contexto: Este servicio sirve para iniciar y terminar conexión lógica.
Acceso a variables: Es usado para el acceso a variables, registros, vectores o lista de variables.
Administración de dominio: Es usado para transmitir gran cantidad de áreas de memoria, los datos deben ser divididos por el usuario en segmentos.
Administración de eventos: Son usados para transmitir mensajes de alarma, estos mensajes pueden ser enviados en transmisión broadcast o multicast.
Soporte a VFD: Son usados para identificación.
Administración OD: Son usados para leer y escribir en el diccionario de objetos.
Control de programa: Son usados para el control del programa

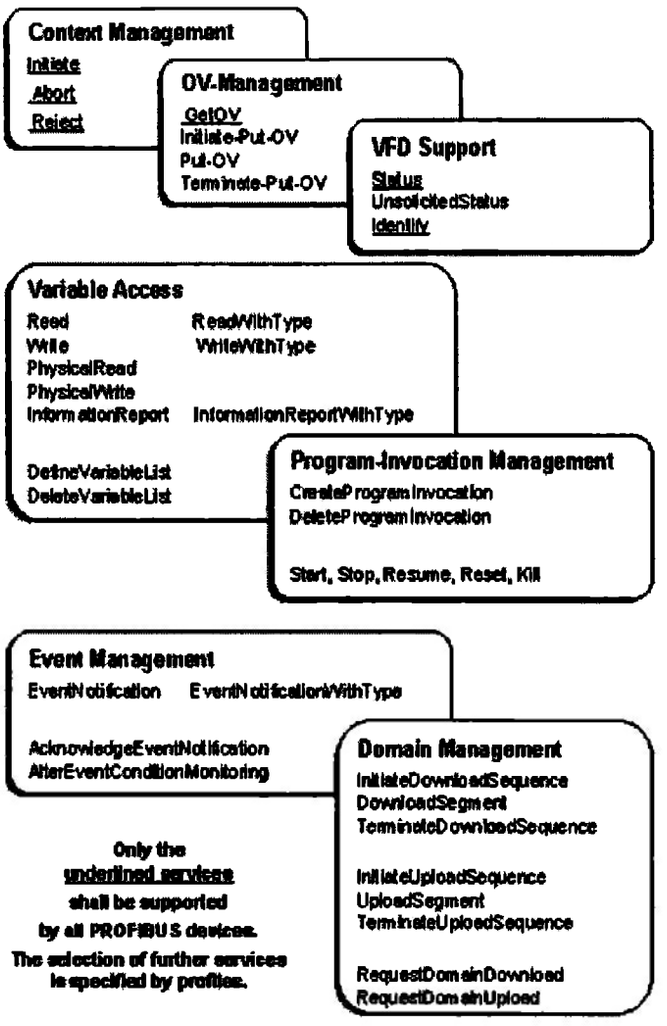


Figura 4.9 Servicios de FMS

4.5.2 LLI (LOWER LAYER INTERFACE)

Las tareas del LLI son control de flujo y monitoreo, los usuarios se comunican con los otros procesos por el canal de relaciones de comunicación. El LLI soporta diferentes tipos de comunicaciones como son: monitoreo, transmisión y demanda de otros equipos.

5 DESARROLLO DEL SISTEMA

El sistema consiste en una red distribuida maestro / esclavo basado en el protocolo PROFIBUS, ver figura 5.1. En este caso los maestros serán computadoras personales, cada maestro tendrá un sistema desarrollado con el lenguaje de programación Visual Basic 6.0 bajo Windows de Microsoft, los maestros serán los administradores de la red y los esclavos serán los que realizan las actividades de automatización y control.

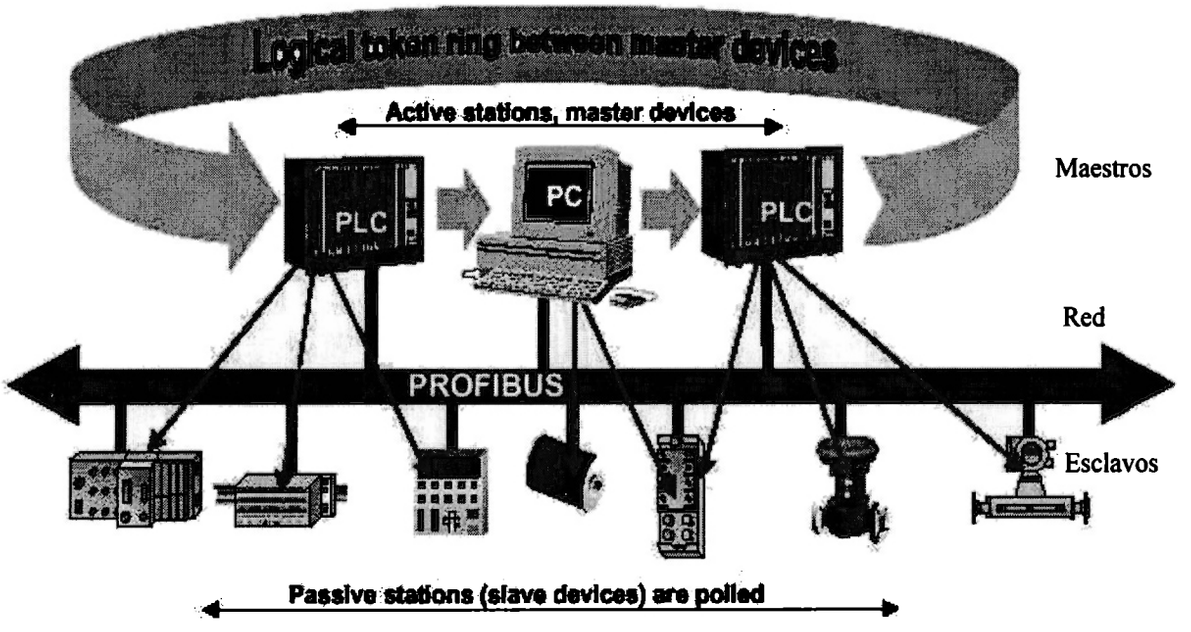


Figura 5.1 Estructura de una red Maestro / esclavo basada en PROFIBUS [9].

En la figura se puede ver de manera global el funcionamiento del sistema. Cabe recordar que la red es un bus de datos y que solamente puede haber una estación transmitiendo en un momento determinado. El software de cada maestro es el que se encarga de la sincronización de la red y de evitar que una estación envié un mensaje cuando no le corresponde. En las siguientes secciones se describirá más a fondo el funcionamiento del sistema.

5.1 ANÁLISIS DEL SISTEMA MAESTRO

Hace cuatrocientos años Maquiavelo dijo: "...no hay nada más difícil de conseguir, más arriesgado de mantener ni más inseguro de tener éxito, que estar a la cabeza en la introducción de un nuevo orden de cosas..." [16].

Durante el último cuarto de siglo veinte, los sistemas basados en computadora están introduciendo un nuevo orden de cosas. Aunque la tecnología ha hecho grandes avances desde Maquiavelo, sus palabras siguen siendo ciertas [16].

La ingeniería del software y la ingeniería del hardware entran dentro de la amplia categoría que llamaremos ingeniería de sistemas computacionales. Cada una de estas disciplinas representa un intento de establecer un orden de desarrollo de sistemas basados en computadora. Las técnicas de ingeniería para el hardware de computadoras provienen del diseño electrónico y han alcanzado un estado de relativa madurez [16]. En los sistemas basados en computadora, el software tiende a reemplazar al hardware.

La palabra "sistema" es posiblemente el termino más sobreutilizado y del que más se ha abusado en el léxico técnico. Hablamos de sistemas políticos y educativos, de sistemas de fabricación, de sistemas bancarios, de sistemas electrónicos, sistemas de cómputo, etc. El diccionario Webster describe la palabra sistema así:

"1. un conjunto u ordenación de cosas relacionadas de tal manera que formen una unidad o un todo orgánico. 2. Un conjunto de hechos, principios, reglas, etc... clasificados y ordenados de tal

manera que muestran un plan lógico uniendo las diferentes partes. 3. Un método o plan de clasificación u ordenación; 4. Una forma establecida de hacer algo; un método; un procedimiento..."

Con base a la definición anterior, definimos un sistema basado en computadora como:

"Un conjunto u ordenación de elementos organizados para llevar a cabo algún método, procedimiento o control mediante el procesamiento de información" [16].

Los elementos del sistema se combinan de muchas formas para transformar la información. Por ejemplo, un robot transforma un archivo de órdenes, que contiene instrucciones concretas, en un conjunto de señales de control que producen alguna acción física concreta.

Una característica de los sistemas computacionales es que un sistema puede pertenecer a otro sistema mayor.
Categóricamente los sistemas computacionales se pueden dividir en [17]:

- Sistemas en línea: Es aquel que acepta material de entrada directamente del área de donde se creó. El material de salida se devuelve directamente a donde es requerido.
- Sistemas en tiempo real: Es aquel que controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho ambiente en ese momento. Este es el tipo de sistema que se trabaja en este proyecto.
- Sistemas de apoyo a decisiones: Ayudan a tomar decisiones en la organización con base en la información que procesan.
- Sistemas basados en el conocimiento: Sistemas del campo de la inteligencia artificial.

Una parte muy importante en el desarrollo de sistemas computacionales es el modelado de estos sistemas, los modelos nos ayudan a ver de una manera más clara el funcionamiento general del sistema y es más fácil entenderlo para una persona que no este relacionada con el desarrollo de software o que no conozca el lenguaje de programación con el que se desarrolla el sistema, actualmente existen diversas técnicas para hacer estos modelos. Las técnicas que se utilizarán en este proyecto son: diagrama jerárquico, análisis estructurado moderno, análisis orientado a objetos (Modelo entidad – relación) y diagrama de transición de estados.

5.1.1 RESUMEN DEL FUNCIONAMIENTO DEL SISTEMA MAESTRO

Como se ha dicho en capítulos anteriores los maestros forman un anillo lógico por medio del cual se están rotando el token o administración de la red. Cuando el maestro tiene el token, éste tendrá el control total de la transmisión.

Cada vez que una estación maestra entra a la red, ésta debe realizar una etapa de inicialización del sistema, en esta etapa el maestro escucha la red durante un tiempo establecido para verificar si hay otra estación maestra que en ese momento tenga el token, si terminado el tiempo de verificación la estación no escucha ningún mensaje, el maestro asumen que no hay ningún otra estación maestra en la red con el token, entonces la estación envía en mensaje anunciando que él va a poseer el token en ese momento y empieza a realizar sus tareas administrativas. Si por el contrario la estación escucha mensajes, significa que otro maestro tiene el token, entonces debe quedarse esperando a que le llegue un mensaje enviado por su maestro predecesor diciéndole que pude poseer el token.

En esta etapa de iniciación el maestro debe leer de la base de datos la información de cada uno de sus esclavos (índice, variables u objetos, estado actual,...) y almacenarla en una matriz para posteriormente cuando tenga el token comunicarse con sus esclavos. También presentará en pantalla todos los dispositivos conectados a la red y la información con la que cuenta de cada uno de sus esclavos .

Si es la primera vez que el maestro entra a la red; el usuario debe configurar ciertos parámetros del maestro, como: timers, adicionar esclavos (aunque también se pueden adicionar esclavos en cualquier momento), registros, banderas y variables de control.
Una vez que la estación maestra obtenga el token, empieza la comunicación con cada uno de sus esclavos, esta comunicación la hace de una manera ordenada de acuerdo al índice de cada esclavo (Hay que recordar que todos los dispositivos conectados a la red deben tener un índice o identificador único, preferiblemente un número consecutivo sin importar si es un maestro o un esclavo). La comunicación con el esclavo puede ser de solicitud de información y de envío de información para cambiar el valor de algunas de las variables del esclavo, cuando le envía el mensaje (el formato del mensaje es especificado por el protocolo, ver capitulo 4) al esclavo, el maestro pasa a un estado de espera de respuesta por parte del esclavo. En este estado el maestro

espera un tiempo determinado, si pasado el tiempo el esclavo no responde, el maestro vuelve a intentar el número de veces especificado en la configuración, si no responde debe mostrar en pantalla un mensaje de alarma, notificar la falla, actualizar la base de datos y pasar a enviar información al siguiente esclavo. La falla puede ser por que la estación esclava tiene problemas de comunicación o por que la estación maestra tiene problemas para escuchar mensajes, el usuario del sistema se encargará de verificar esto. Si recibe respuesta del esclavo, el maestro verificará que el mensaje venga correctamente, si el mensaje es correcto actualizara en pantalla y en la base de datos los datos recibidos y pasará a enviar mensajes al siguiente esclavo. Si el mensaje es erróneo el maestro vuelve a intentar el número de veces especificado en la configuración y si no recibe respuesta correcta notificará la falla y pasará a enviar información al siguiente esclavo.

Cuando el maestro termine de comunicarse con todos sus esclavos o termine el tiempo a que tiene derecho de poseer el token, éste envía un mensaje al maestro sucesor notificándole que le va a pasar el token, al enviar el mensaje pasa a un estado de espera de respuesta por parte del maestro sucesor notificándole que recibió el mensaje y que va a poseer el token. En este estado el maestro espera un tiempo determinado, si al pasar ese tiempo no recibe respuesta, el maestro vuelve a intentar el número de veces especificado en la configuración, si no recibe repuesta, asume que el maestro sucesor esta desconectado o tiene problemas de comunicación, por lo tanto notifica la falla e intenta pasar el token a otra estación maestra.

A grandes rasgos, lo anterior fue resumen del funcionamiento del sistema maestro, a continuación se mostrara cada una de sus funciones específicas:

1. Tendrá a su cargo esclavos que son los que realizan las actividades de automatización y control.
2. Es responsable de la iniciación y configuración de la red.
3. Designará quien puede utilizar el medio físico en un momento determinado.
4. Pasar el token o derecho a administrar la red al maestro siguiente.
5. Estar pendiente de la recepción del token del maestro antecesor.
6. Controlará la transmisión de datos (inicio, terminación, control)

7. Trabajará con dispositivos virtuales, que consiste en la representación virtual en el sistema de cada uno de los esclavos, en donde los usuarios podrán leer y modificar los parámetros de estos sin necesidad de acudir directamente al esclavo.
8. Contará con una base de datos hecha con Access 2000 de Microsoft, esta base de datos servirá de apoyo para la administración y control del sistema, tendrá información de: esclavos (índice, nombre, tipo de control, variables u objetos que maneja, tipos de datos de las variables, longitud de las variables, fabricante, parámetros de funcionamiento, etc...), también tendrá información de timers, errores, servicios y configuración.
9. Tendrá capacidad para detectar la adición o eliminación de un nuevo dispositivo a la red. Los esclavos tendrán drivers que son archivos que contienen toda la información referente al esclavo, cada vez que se adiciona un esclavo a la red, el maestro lee estos drivers y actualiza la base de datos.
10. Detectar que no halla dispositivos en la red con el mismo índice o dirección (Cada dispositivo tendrá un índice único que lo identificará en la red)
11. Generación del frame (mensaje para la comunicación que utiliza el protocolo, este mensaje tiene varios campos que son: delimitador de inicio, dirección destino, dirección origen, campo de control, datos, campo de verificación y delimitador de final).
12. Interpretación de los frames recibidos.
13. Detección de fallas de transmisión.
14. Detectar la ausencia de alguno de sus esclavos.
15. Detectar la ausencia de alguno de los maestros.
16. Detección de mal funcionamiento de algún esclavo.
17. Presentación visual de la información.
18. Generación de reportes del funcionamiento de la red, de los esclavos, de los parámetros, etc...
19. Ambiente amigable para usuarios inexpertos.
20. Entre otras

5.1.2 DIAGRAMA JERÁRQUICO

El diagrama jerárquico muestra el funcionamiento del sistema de una manera jerárquica como su nombre lo indica. Es similar al árbol de estructura organizacional de una compañía. Cada función

del árbol es descrita con más detalle en las funciones hijas del árbol. En la figura 5.2 se muestra el árbol del sistema maestro.

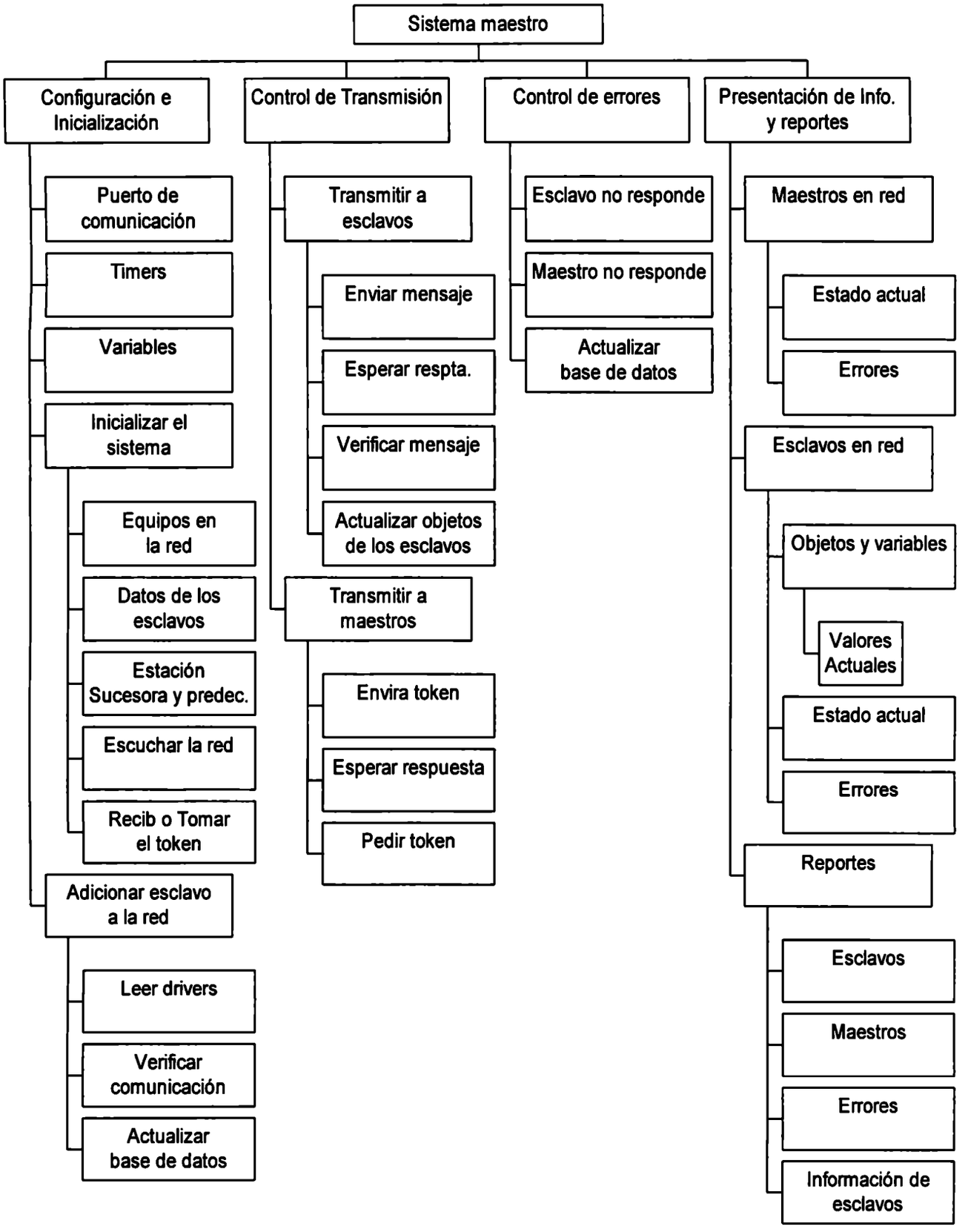


Figura 5.2 Diagrama jerárquico del sistema maestro

5.1.3 ANÁLISIS ESTRUCTURADO MODERNO

Para considerar un sistema de cómputo de una manera exitosa es muy importante tener en cuenta tanto los procesos como los datos que se manejan [17]. La herramienta de modelado que utilizaremos para describir la transformación de la información de entradas a salidas es un diagrama de flujo de datos. Es importante aclarar que este diagrama de flujo de datos es diferente a los diagramas de flujo que se utilizan para modelar el funcionamiento de un algoritmo, el diagrama de flujo de datos que se utilizará es un diagrama mucho mas versátil, que muestra el funcionamiento del sistema desde una manera global a una manera específica. Este diagrama sirve para que personas sin conocimientos en computación entiendan el funcionamiento del sistema.

A medida que la información se mueve a través del software, es modificada por una serie de transformaciones. El diagrama de flujo de datos es una técnica gráfica que representa el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida. Se puede usar un diagrama de flujo de datos para representar un sistema o un software a cualquier nivel de abstracción [16]. Los diagramas de flujo de datos pueden ser presentados en niveles que representan un mayor flujo de información y un mayor detalle funcional. El nivel 0 que también es llamado diagrama de contexto, representa el software total en un solo proceso, todas las entradas y salidas que están involucradas en el sistema. Para mostrar más detalle se pasa al nivel 1 y así sucesivamente. En la figura 5.3 se muestra la notación utilizada en los diagramas de flujo de datos

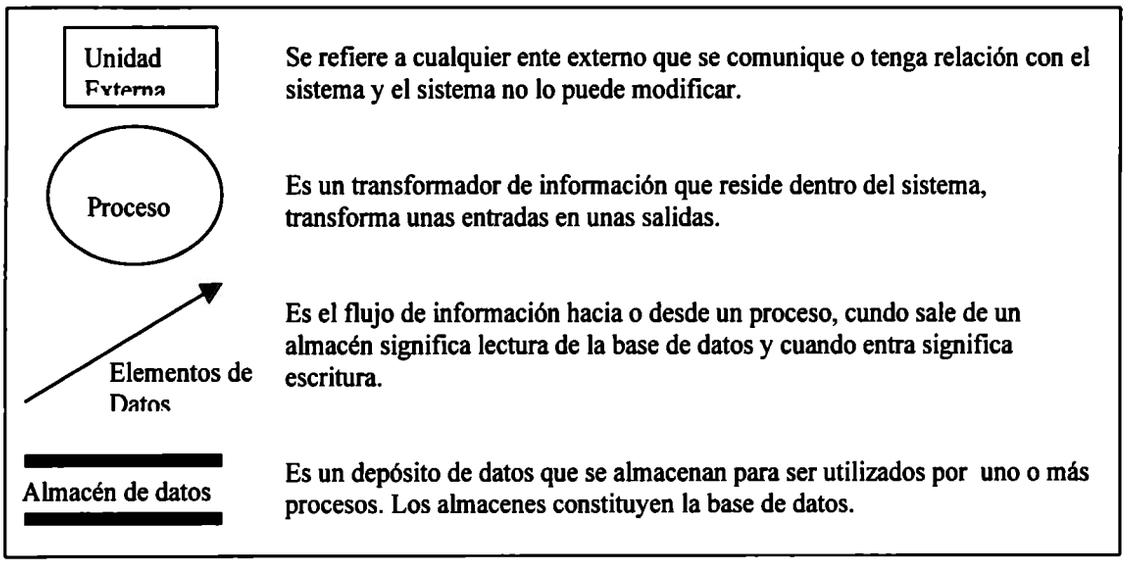


Figura 5.3 Notación utilizada en los diagramas de flujo de datos.

5.1.3.1 Diagrama de Contexto

Como se dijo anteriormente el diagrama de contexto muestra de manera general el sistema y su comunicación con el mundo externo. La figura 5.4 muestra el diagrama de contexto del sistema maestro.

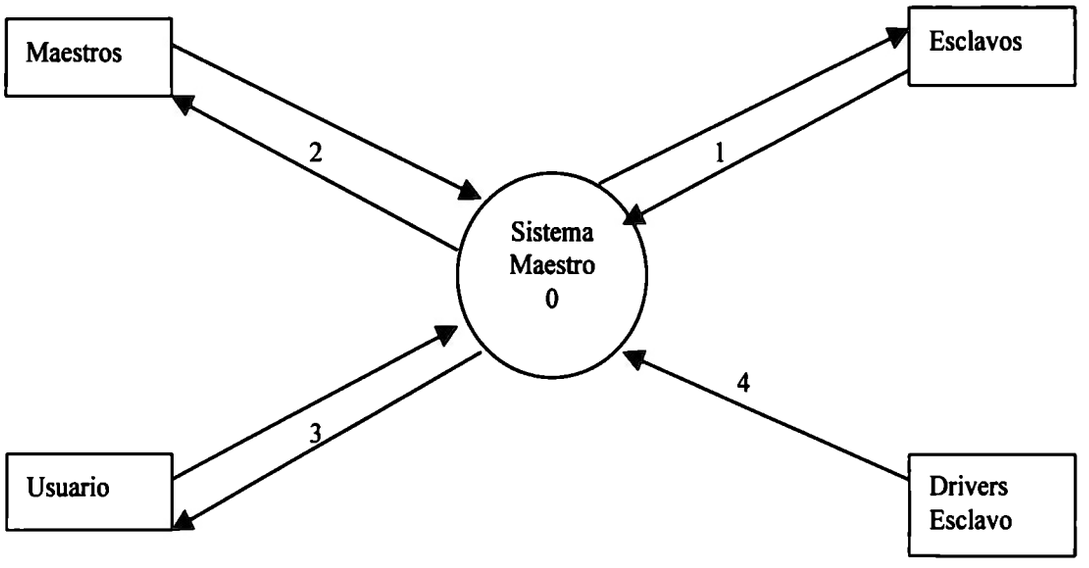


Figura 5.4 Diagrama de contexto del sistema maestro

1. Transmisión y recepción de datos
2. Token
3. Admón. del sistema, reportes, etc..
4. Información de un esclavo nuevo (Índice, objetos, etc)

5.1.3.2 Nivel 1

En este nivel ya se mostrará de una manera un poco más profunda el sistema total, vendría siendo el primer nivel del árbol de la figura 5.2, solo que se presenta la interacción de los procesos con las entidades externas y con la base de datos. Si se quiere llegar a un nivel de detalle mayor se pasará al nivel 2. En este nivel los procesos se enumerarán en orden consecutivo (1,2,3...). En la figura 5.5 se presenta el nivel 1 para sistema maestro.

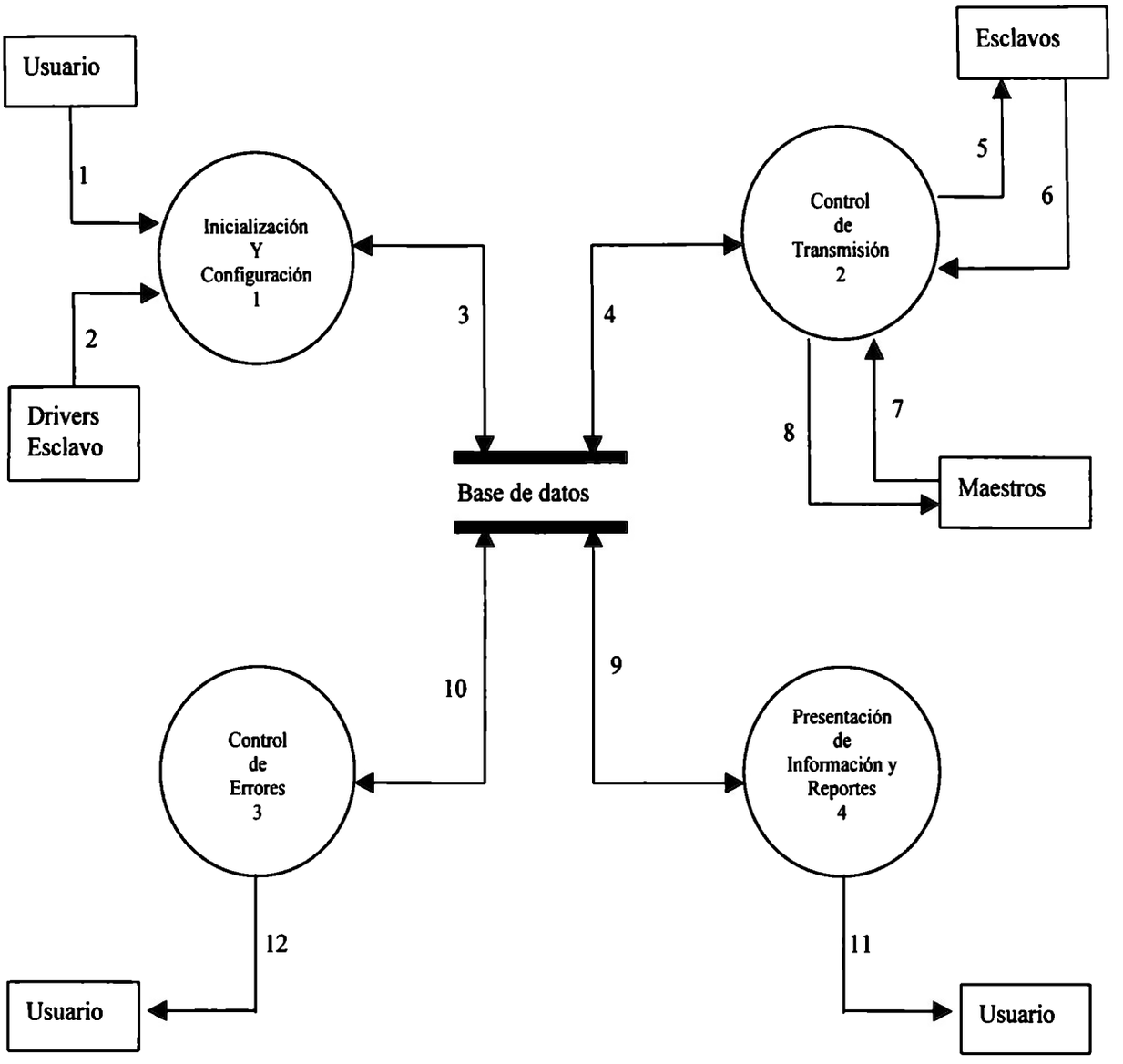


Figura 5.5 Nivel 1 del sistema maestro

1. Datos de configuración del sistema, como: timers, variables, estado, etc..
2. Datos del esclavo: índice, objetos, nombre, etc.
3. Lectura y escritura a la base de datos, datos de los esclavos, maestros y timers.
4. Datos del esclavo, (índice, objetos o variables), datos de los maestros.
5. Envió de mensaje, solicitando o actualizando las variables del esclavo.
6. Respuesta del esclavo con la información solicitada.
7. Recepción del token del maestro predecesor.
8. Envío del token al maestro sucesor.
9. Datos de los esclavos, maestros, errores, funcionamiento del sistema.
10. Errores del sistema.
11. Información y reportes.
12. Notificación de errores.

5.1.3.2 Nivel 2

En este nivel se entrar a especificar cada uno de los procesos del nivel 1, estos proceso son: 1. Iniciación y configuración, 2. Control de transmisión, 3. Control de errores, 4. Presentación de información y reportes. Cada uno de estos procesos se descompondrá en otros subprocesos, para identificarlos se enumeran con el número del proceso más un consecutivo, ejemplo, para el proceso 1, quedara 1.1, 1.2, 1.3...

5.1.3.2.1 Nivel 2. Proceso 1. (Iniciación y configuración).

En este nivel se presentara detalladamente el funcionamiento del proceso Inicialización y configuración. En la figura 5.6 se muestra este diagrama.

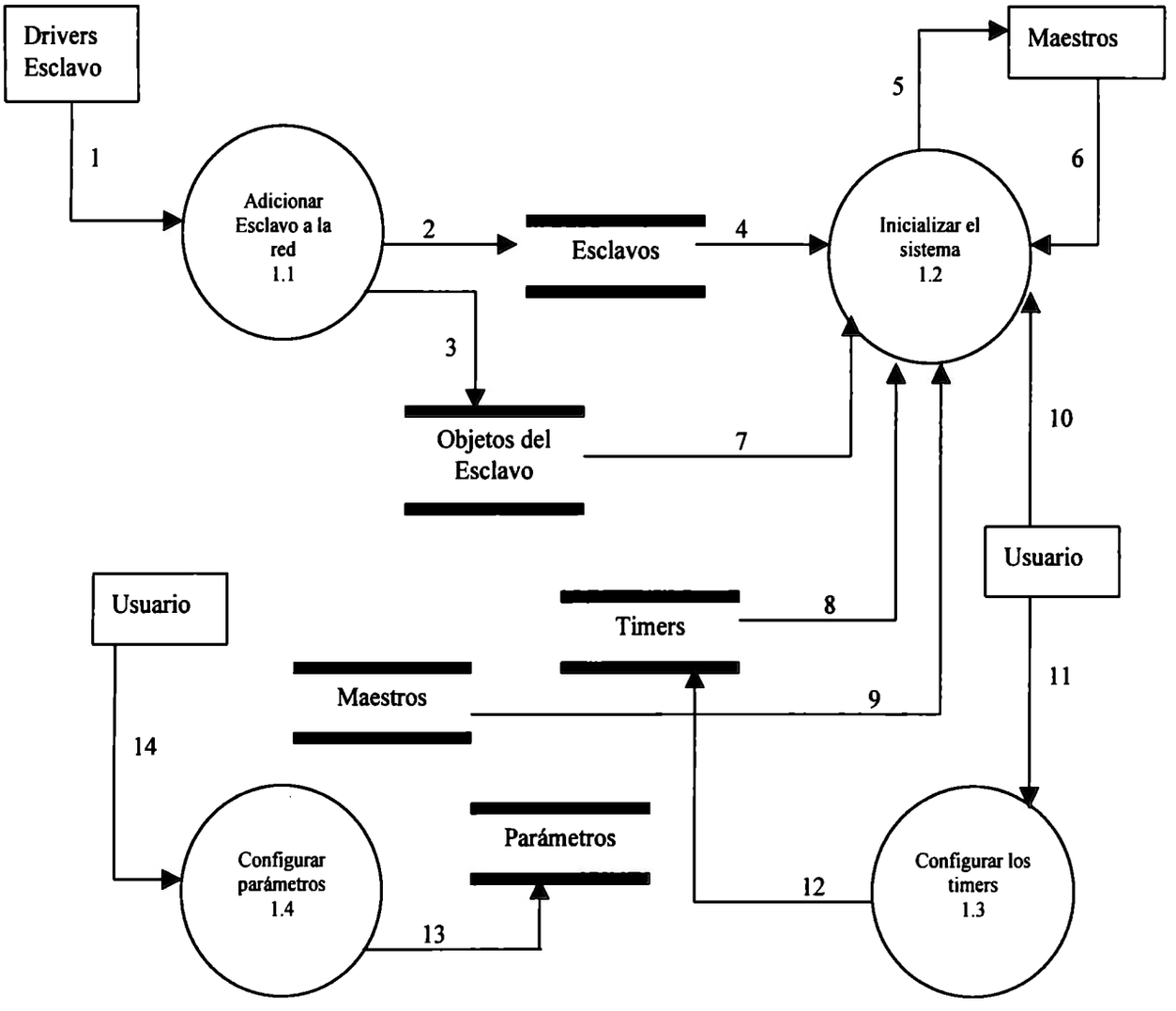


Figura 5.6 Nivel 2. Proceso Inicialización y configuración

1. Datos del esclavo, (Índice, nombre, objetos, etc..).
2. Datos del esclavo leídos de los drivers.
3. Objetos del esclavo leídos de los drivers.
4. Datos del esclavo.
5. Mensaje al maestro indicando que va a tomar el token.
6. Escuchando mensajes de los maestros existentes, verificando si tienen el token.
7. Objetos que maneja cada esclavo.
8. Valor de los timers para poder inicializarlos.
9. Datos del maestro sucesor y predecesor.
10. Iniciación del sistema.
11. Datos de los valores de los timers.
12. Datos de los valores de los timers.
13. Parámetros generales de funcionamiento del sistema.
14. Parámetros generales de funcionamiento del sistema.

5.1.3.2.2 Nivel 2. Proceso 2. (Control de transmisión)

En este nivel se presentara detalladamente el funcionamiento del proceso control de transmisión. En la figura 5.7 se muestra este diagrama. Este proceso necesita un nivel más de descomposición, este nivel se trabajará con un diagrama de transición de estados que se presentara más adelante.

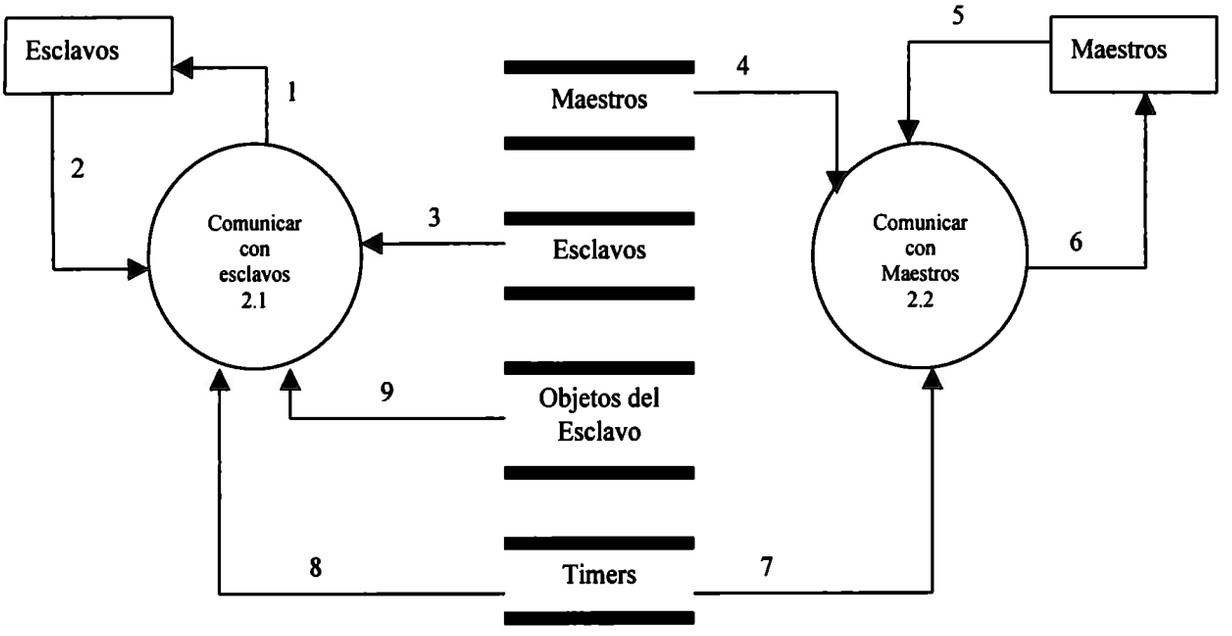


Figura 5.7 Nivel 2. Proceso control de transmisión

1. Mensaje para cada uno de los esclavos que le pertenecen al maestro.
2. Respuesta del mensaje enviado por el maestro.
3. Datos del esclavo.
4. Datos del maestro sucesor y predecesor.

5. Respuesta del maestro diciendo que recibió el token.
6. Envío del token al maestro sucesor.
7. Timers de espera de respuesta de los maestros.
8. Timers de espera de respuesta de los esclavos.
9. Objetos de cada esclavo.

5.1.3.2.3 Nivel 2. Proceso 3. (Control de Errores)

En este nivel se presentara detalladamente el funcionamiento del proceso control de Errores. En la figura 5.8 se muestra este diagrama

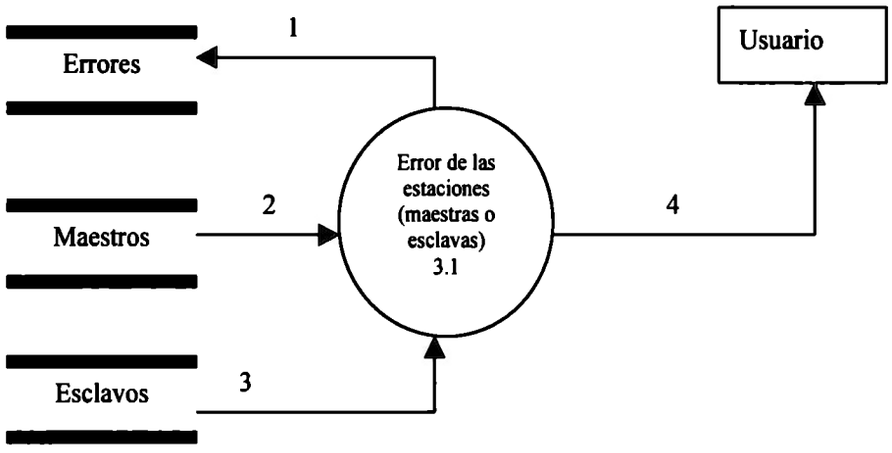


Figura 5.8 Nivel 2. Proceso control de errores

1. Datos de la estación que cometió el error, fecha y hora, posible causa del error.
2. Datos del maestros, si no respondió.
3. Datos del esclavo, si no respondió.
4. Notificación del error.

5.1.3.2.4 Nivel 2. Proceso 4. (Presentación de información y reportes)

En este nivel se presentará detalladamente el funcionamiento del proceso presentación de información y reportes.

En la figura 5.9 se muestra este diagrama

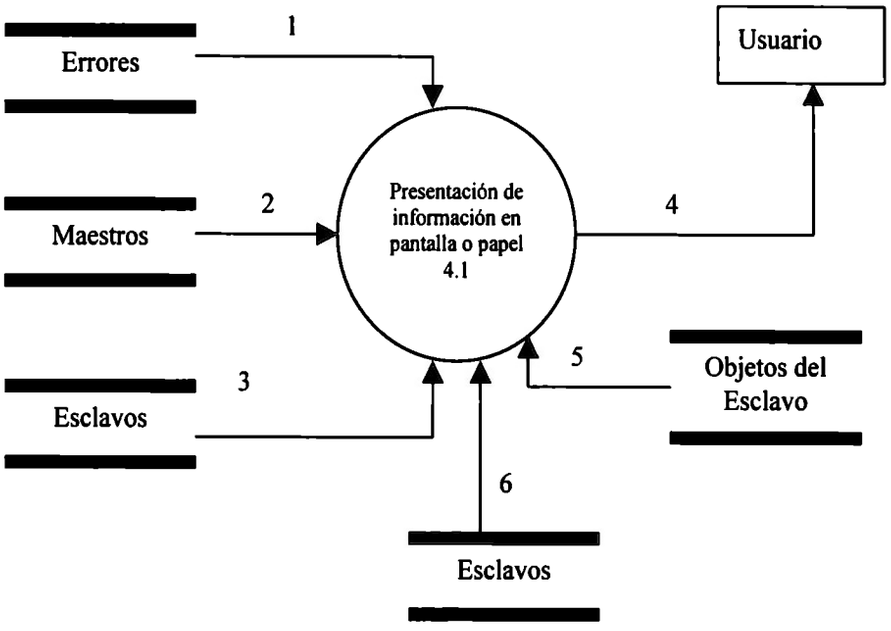


Figura 5.9 Nivel 2. Proceso presentación de información y reportes

1. Errores del sistema.
2. Maestros en la red.
3. Esclavos en la red.
4. Presentación en pantalla o impresora.
5. Valores de las variables de cada esclavo
6. Esclavos en la red.

5.1.4 ANÁLISIS ORIENTADO A OBJETOS (MODELO ENTIDAD – RELACIÓN) Y DESARROLLO DE BASE DE DATOS EN ACCESS 2000

Actualmente, el análisis orientado a los objetos va progresando poco a poco como método de análisis de requisitos por derecho propio y como complemento de otros métodos de análisis. Coad y yourdan definen el análisis orientado a objetos así:

"El análisis orientado a objetos, se basa en conceptos que una vez aprendimos en la guardería: objetos y atributos, clases y miembros, todo y parte. Nadie sabe por qué hemos tardado tanto tiempo en aplicar estos conceptos en el análisis y la especificación de los sistemas de información, quizás hemos estado demasiado ocupados siguiendo el flujo durante nuestros análisis diarios, para ponernos a considerar otras alternativas" [16].

Para poder entender el concepto orientado a objetos haremos un ejemplo: Consideremos una silla, la silla es un miembro de una clase de objetos mucho mayor que denominamos mueble. Se puede asociar un conjunto de atributos genéricos a cada objeto de la clase mueble. Por ejemplo, todo mueble tiene precio, dimensiones, peso, situación y color, entre muchos atributos posibles. Se aplican estemos hablando de una mesa o de una silla, de un sofá o de una cómoda. Dado que silla es miembro de la clase mueble, hereda todos los atributos definidos para la clase. En la figura 5.10 se ilustra este concepto.

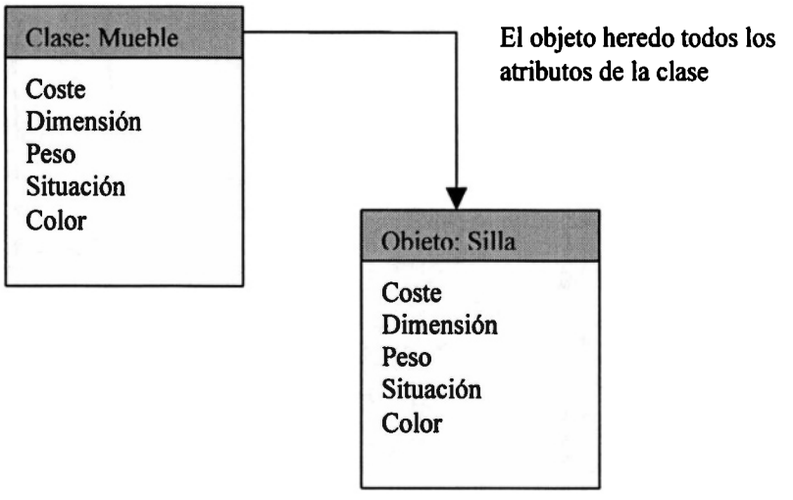


Figura 5.10 Herencia de clase a objeto

5.1.4.1 Diagrama entidad - relación

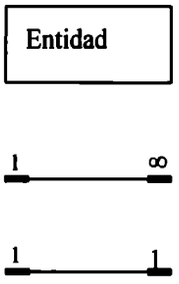
El diagrama entidad – relación es una técnica para definir las necesidades de información del sistema. El diagrama entidad – relación identifica las cosas importantes del sistema (Entidades u objetos), las propiedades de esas cosas (atributos) y como están relacionados entre sí esas cosas [19]. Lo más importante de este diagrama es poder ver las relaciones que existen entre los objetos del sistema. Éste se usa para tener un conocimiento de la base de datos que maneja el sistema sin necesidad de ser un experto en bases de datos o sin necesidad de conocer Access.

El diagrama entidad – relación posteriormente pasara a ser la base de datos del sistema.

La notación entidad – relación es relativamente sencilla. Los objetos de datos se representan como rectángulos etiquetados. Las relaciones se indican mediante flechas y las conexiones

mediante variadas formas especiales de conexión. En la figura 5.11 se muestran la notación utilizada.

En el apartado anterior se presentaron los diagramas de procesos, estos procesos normalmente toman datos de los almacenes, estos almacenes pasarán a ser las entidades u objetos en el diagrama entidad relación.



Representa las entidades, cada entidad tiene ciertos atributos. Cuando se cree la base de datos, las entidades pasaran a ser tablas y los atributos serán los campos.

Relación una a muchas.

Relación una a una.

Figura 5.11 Notación utilizada por el diagrama entidad - relación

Para entender mejor este concepto haremos un ejemplo: Una compañía de ventas quiere tener almacenado en una base de datos sus clientes (de sus clientes quiere tener un código, el nombre la dirección y el teléfono) y las compras realizadas por cada cliente (de las compras quiere tener concepto, cantidad y valor). En la figura 5.12 se muestra el respectivo modelo.

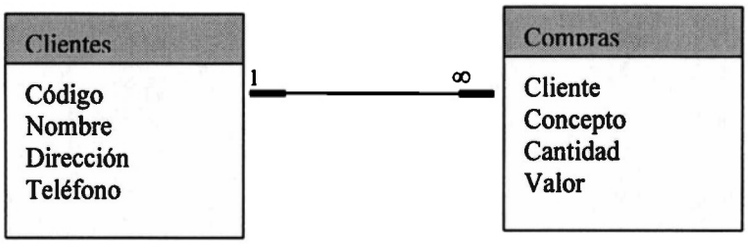
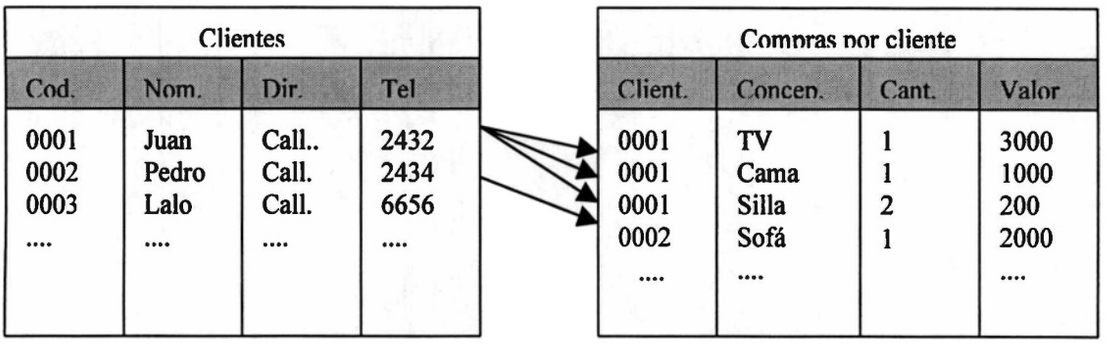


Figura 5.12 Ejemplo de entidad - relación

En el ejemplo se ve una relación una a muchas del cliente con las ventas respectivamente, esto significa que el cliente puede tener 0,1 o muchas compras, pero una compra específica sólo pertenecer a un cliente. La relación entre las dos tablas es el código del cliente. Con esto se evita redundancia en la información. En la base de datos quedaría algo como se muestra en la figura 5.13. En la figura se puede ver que el cliente 0001 hizo 3 compras, el cliente 0002 hizo 1 compra y el cliente 0003 no hizo compras. Si se hubiera utilizado una sola tabla en lugar de dos

relacionadas, sería mucho más difícil de administrar. En la figura 5.14 se muestra como quedaría con usa tabla sin relacionar.



Clientes			
Cod.	Nom.	Dir.	Tel
0001	Juan	Call..	2432
0002	Pedro	Call.	2434
0003	Lalo	Call.	6656
....	....	....	....

Compras por cliente			
Client.	Concep.	Cant.	Valor
0001	TV	1	3000
0001	Cama	1	1000
0001	Silla	2	200
0002	Sofá	1	2000
....	....		....

Figura 5.13. Como quedaría el ejemplo en una base de datos.

Compras por clientes						
Cod.	Nom.	Dir.	Tel	Concep.	Cant.	Valor
0001	Juan	Call..	2432	TV	1	3000
0001	Juan	Call.	2432	Cama	1	1000
0001	Juan	Call.	2432	Silla	2	200
0002	Pedro	Call.	2434	Sofá	1	2000
0003	Lalo	Call.	6656			
....	....	....	....			

Figura 5.14 Como quedaría el ejemplo si usar relaciones.

En la figura anterior se puede ver que cuando no se usan relaciones hay redundancia de información ya que las tres primeras filas y las cuatro primeras columnas tienen la misma información. Por cada compra que el cliente haga se repetiría la misma información.

5.1.4.1.1 Diagrama entidad – relación del sistema maestro

A continuación se presentará el diagrama entidad – relación del sistema maestro. Los objetos que están sin relaciones es por que no se considera necesario relacionarlos con otros objetos. Este diagrama consta de 7 entidades que son:

1. Esclavos: Esclavos que pertenecen al maestro
2. DiccionarioObjetosEsclavos: Las variables que maneja cada esclavo
3. Objetos: El valor de los objetos de cada esclavo, este valor es leído periódicamente cada vez que el maestro se comunica con el esclavo.

4. Maestros: Maestros en la red.
5. Timers: Timers utilizados en el sistema
6. Errores: Errores de transmisión de las estaciones maestras y esclavas
7. Parámetros: Parámetros generales de funcionamiento del sistema.

Cada entidad tiene atributos que son presentados en el modelo. En la figura 5.15 se muestra el diagrama del sistema maestro. Este diagrama esta hecho en Access 2000 y para cada entidad fue creada una tabla con los respectivos campos.

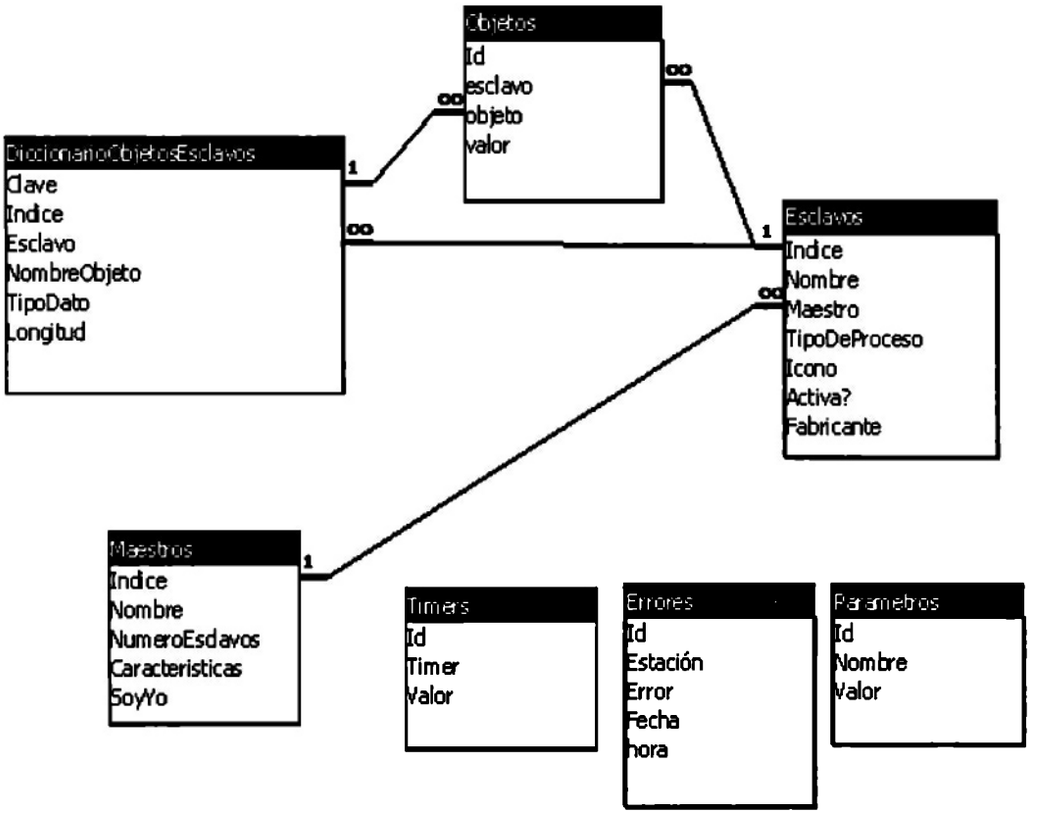


Figura 5.15 Diagrama Entidad – Relación del sistema maestro

En la figura 5.16 se muestra la base de datos y las tablas creadas en Access 2000.

Figura 5.16 Base de datos en Access 2000

5.1.5 DIAGRAMA DE TRANSICIÓN DE ESTADOS

Este tipo de diagrama se utiliza para modelar sistemas que trabajan en tiempo real. Como cualquier sistema basado en computadora, un sistema de tiempo real debe integrar hardware, software, hombres y elementos de una base de datos, para conseguir adecuadamente un conjunto de requisitos funcionales y de rendimiento [16].

Los sistemas de tiempo real generan alguna acción en respuesta a sucesos externos. Para realizar esta función, ejecutan una adquisición y control de datos a alta velocidad bajo varias ligaduras de tiempo y fiabilidad. Debido a que estas ligaduras son muy rigurosas, los sistemas de tiempo real están frecuentemente dedicados a una única aplicación [16,17].

En la figura 5.7 se mostró el diagrama de flujo de datos del proceso de control de transmisión, se dijo que sería conveniente descomponer este diagrama a otro nivel, más sin embargo considero

que la mejor opción es representar esta descomposición con un diagrama de transición de estados, ya que los procesos que trabaja este nivel son dependientes del tiempo.

Este proceso es el encargado de enviar mensajes a los esclavos y esperar la respuesta durante cierto tiempo determinado, lo mismo sucede cuando envía el token a los maestros.

Los principales componentes de un diagrama de transición de estados son los estados que se representan por rectángulos, condiciones, acciones, y las flechas que representan el cambio de un estado a otro. En la figura 5.17 se muestra un ejemplo.

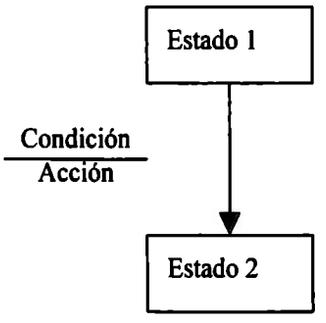


Figura 5.17 Ejemplo de diagrama de transición de estados

5.1.5.1 Diagrama de transición del control de transmisión

El proceso de control de transmisión realiza diversas actividades, entre estas están:

1. Escuchar el medio, para ver si están transmitiendo.
2. Si no transmiten, tomar e token y comunicar a los maestros.
3. Enviar mensajes a cada uno de sus esclavos.
4. Esperar respuesta del esclavo.
5. Verificar que la respuesta sea correcta.
6. Reenviar si es necesario.
7. Pasar el token al maestro siguiente.
8. Esperar respuesta del maestro siguiente notificándole que recibió el token.
9. Esperar que le envíe el token el maestro predecesor.
10. Comenzar el ciclo nuevamente.

Todas estas actividades son sincronizadas con timers, cada estado tiene un timer, se sale del estado cuando realiza su labor o cuando el timer expira. El tiempo de los timers es especificado en la etapa de configuración del sistema.

La figura 5.18 muestra el diagrama de transición de estados de los procesos de control de transmisión. En la etapa de programación del sistema, cada estado de la figura 5.18 pasará a ser un procedimiento o un algoritmo que realiza esa actividad y estará enlazado a los timers de control de comunicación, estos timers se explicarán más adelante.

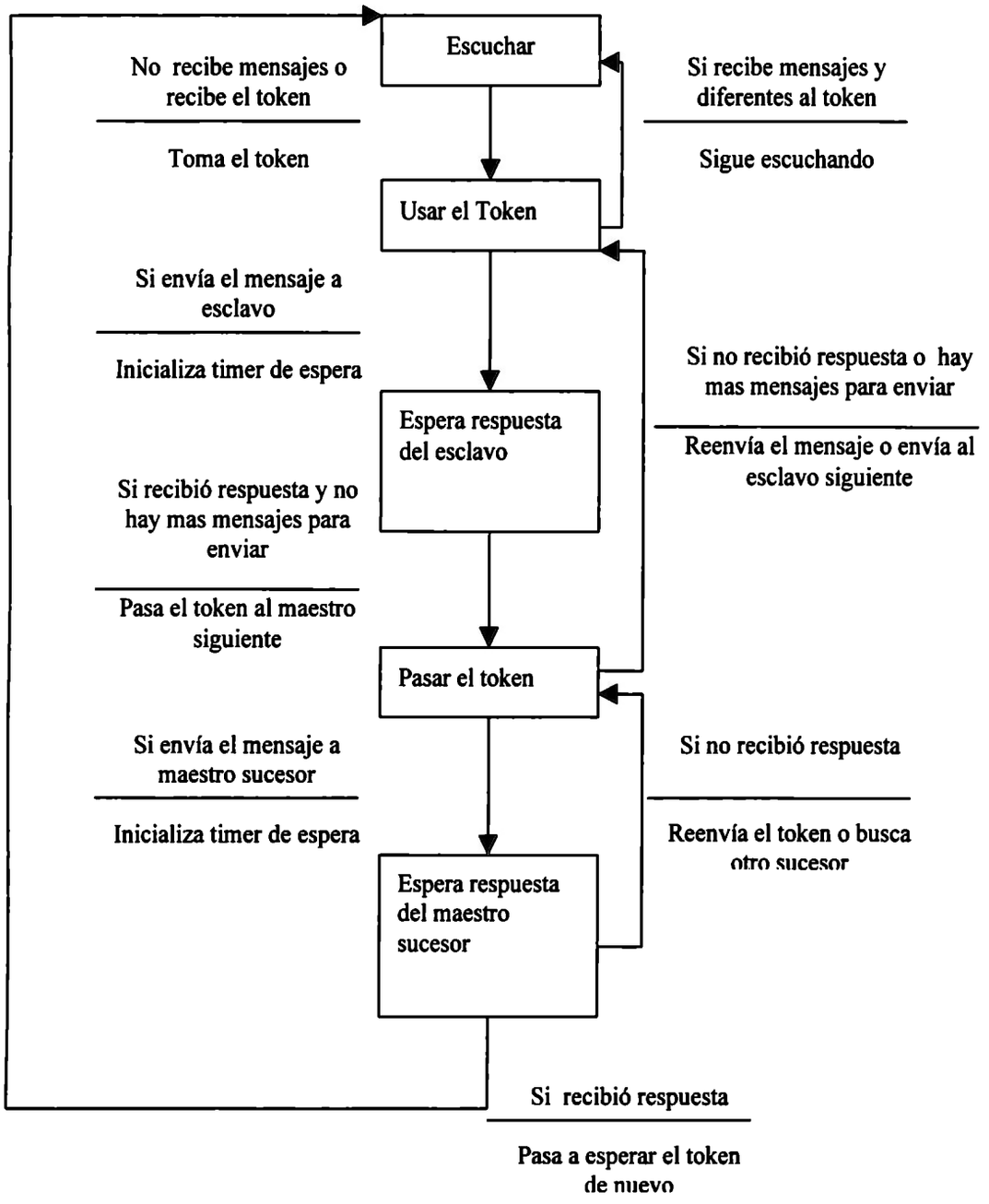


Figura 5.18 Diagrama de transición de estados del control de la transmisión.

5.2 DISEÑO Y PROGRAMACIÒN DEL SISTEMA MAESTRO

Se puede definir diseño de sistemas como: "...el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física" [16].

Habiendo realizado el análisis del sistema se pasa a la etapa de diseño y programación del mismo, en esta etapa se realizan las siguientes actividades:

1. Se crean las pantallas que tendrá el sistema, esto involucra: menús, vistas, Barras de herramientas, ventanas, reportes, etc. Es en si es la interfaz del usuario.
2. Codificación y especificación de cada uno de los procedimientos y funciones indicados en el análisis del sistema.

5.2.1 INTERFAZ DEL USUARIO

El sistema se ha diseñado de tal manera que el usuario pueda ver toda la información del funcionamiento de la red en una sola pantalla. Esta pantalla se divide en 5 secciones principales que son:

Menús: Esta sección contiene cada una de las opciones que puede realizar el sistema, El menú cuenta con diferentes opciones principales que son: Configurar, transmitir, servicios, reportes, ver y ayuda. La figura 5.19 muestra estos menús y la tabla 15.19 describe cada opción y subopción del menú. Cada opción principal tiene otras subopciones que se describirán en la tabla 5.1.

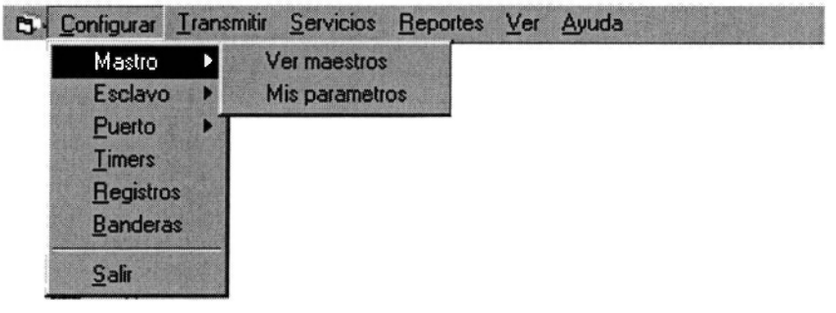


Figura 5.19 Menús del sistema

Opción	*Descripción*
Maestro	Sirve para ver los parámetros del maestro.
Esclavo	Sirve para adicionar esclavo nuevo, configurar uno existente, ver objetos del esclavo. Cuando se adiciona un nuevo esclavo a la red, el sistema debe leer los drivers del esclavo, estos drivers son un archivo de texto. La figura 5.21 muestra la ventana para seleccionar los drivers.
Puerto	Sirve para seleccionar el puerto serial de la computadora (COM1 o COM2)
Timers	Con esta opción puedo ver o modificar los valores de los timers que usa el sistema.
Registros	Registros del sistema.
Salir	Salir del sistema.
Transmitir	Sirve para transmitir frames o tokens.
Reportes	Para ver reportes de maestros, esclavos y fallas.
Ver	Para elegir si quiero ver o no la barra de herramientas.

Tabla 5.1 Descripción de las opciones del menú.

La figura 5.20 muestra la pantalla principal del sistema.

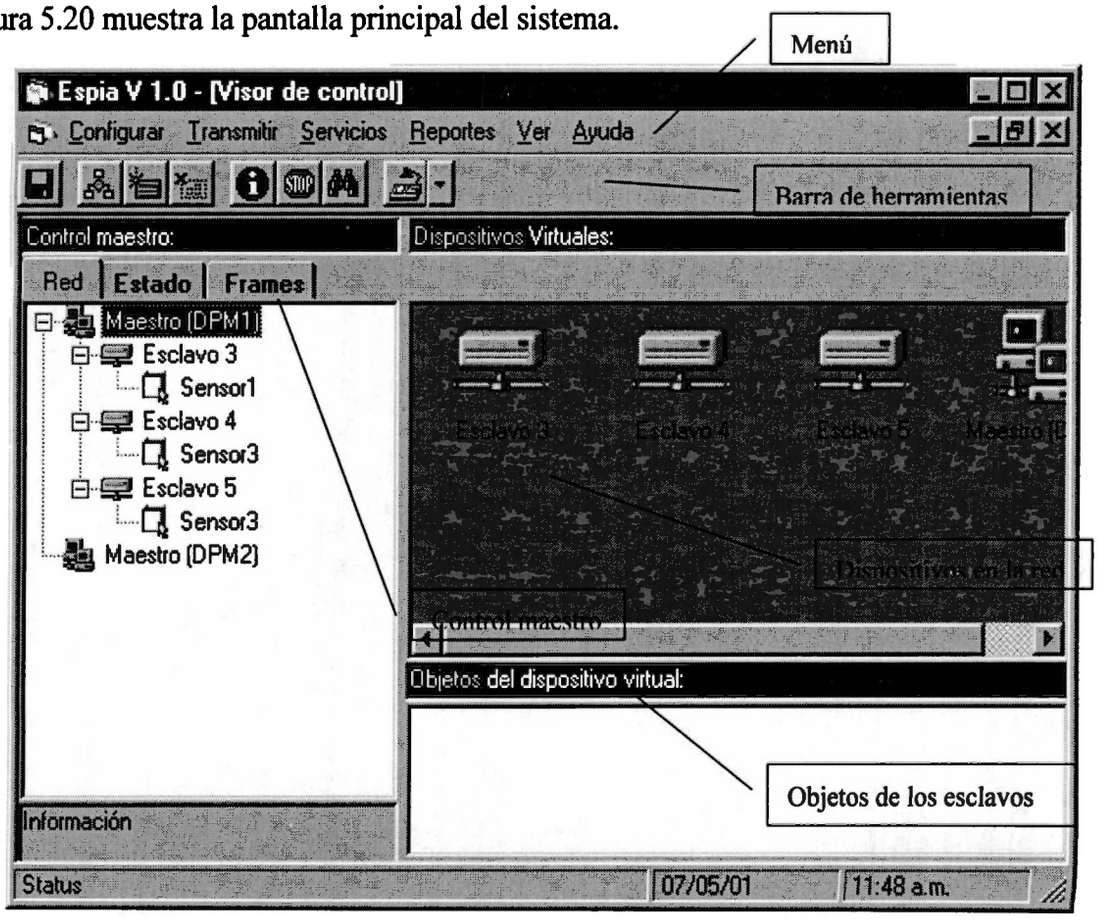


Figura 5.20 Pantalla principal del sistema

Barra de herramientas: Son accesos rápidos para realizar algunas de las opciones del sistema. La figura 5.21 muestra la barra de herramientas del sistema y especifica las opciones más importantes.

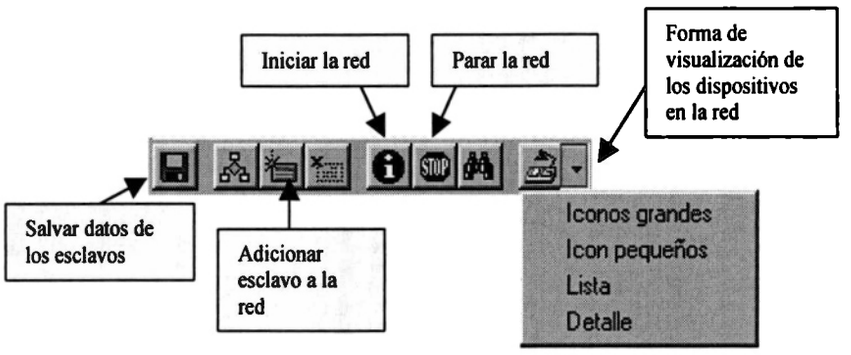


Figura 5.21 Barra de herramientas del sistema

Control maestro: Esta ventana se divide a su vez en tres ventanas, las cuales se describirán a continuación:

Red: Muestra en forma de árbol los dispositivos conectados a la red en determinado momento y también cada una de las propiedades u objetos de los esclavos que pertenecen al maestro. Cuando se adiciona un nuevo esclavo a la red, este aparecerá de inmediato en esta ventana junto con las variables que maneja, igualmente cuando se elimina un esclavo este desaparecerá de la ventana. Cuando un esclavo no responde o tiene problemas de comunicación, en le icono del esclavo aparecerá un X de color rojo indicando que el esclavo tiene problemas de comunicación. La figura 5.22 muestra esta ventana

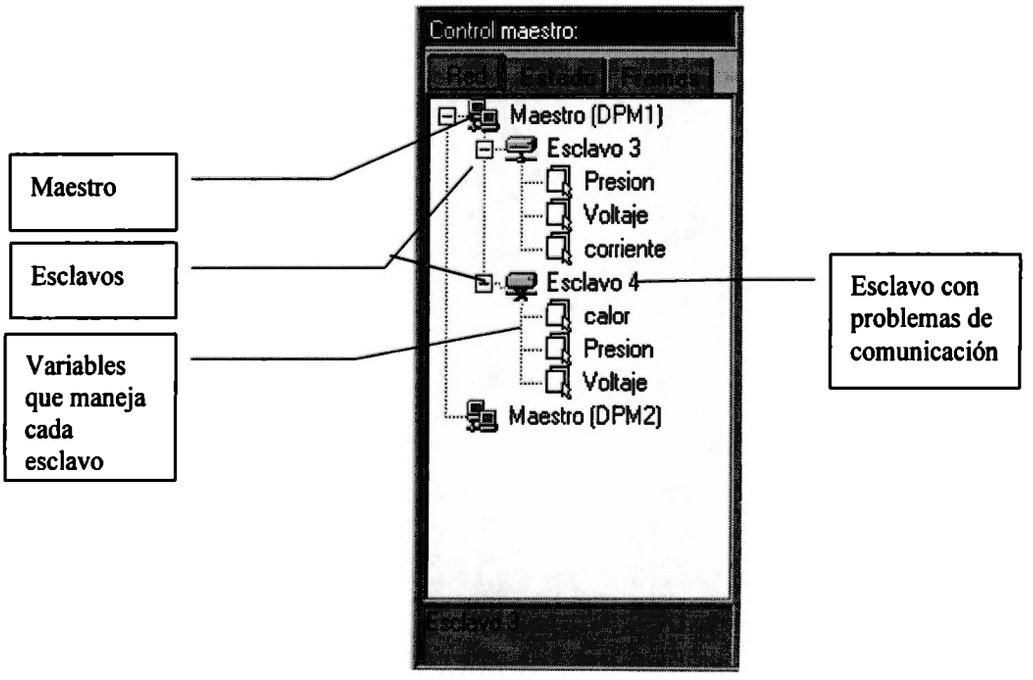


Figura 5.22 Ventana de red

Estado: Muestra el estado de cada uno de los dispositivos conectados en la red. Al dar clic al dispositivo (el cual aparece en la ventana dispositivos virtuales) me mostrara su estado y describirá posible error si existe. Los estados son mostrados por un semáforo donde rojo es alarma, amarillo pausa y verde correcto. La figura 5.23 muestra esta ventana.

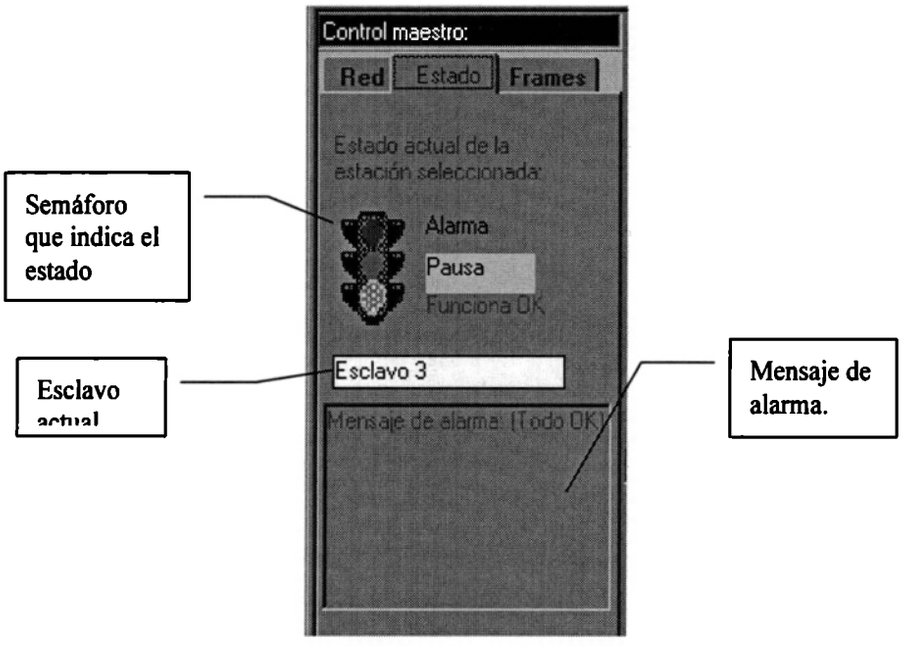


Figura 5.23 Ventana de estado

Frames: Esta ventana sirve para enviar un mensaje a cualquier esclavo de la red que pertenezca al maestro y que sea actuador en alguna actividad de control. La figura 5.24 muestra esta ventana.

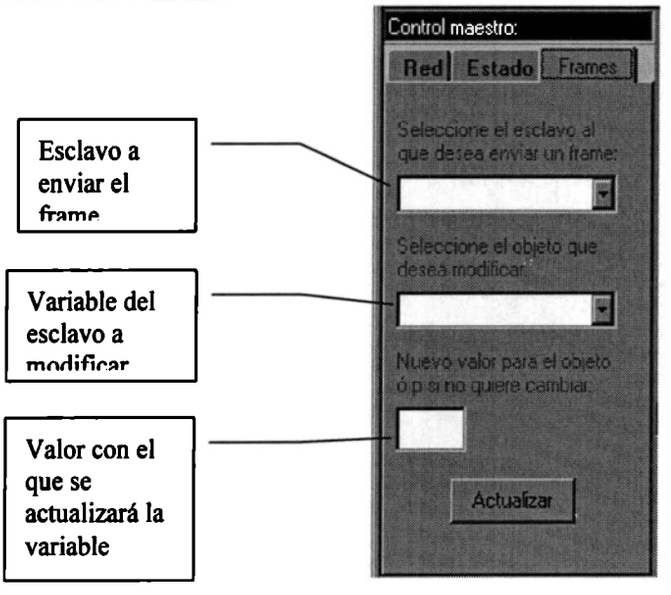


Figura 5.24 Ventana de frames

Dispositivos virtuales: Esta ventana muestra de manera gráfica cada uno de los dispositivos conectados a la red. Los íconos con los cuales muestra estos dispositivos son diferentes dependiendo si es un esclavo, un maestro, si esta activado o desactivado. Al darle clic al dispositivo se vera en la ventana objetos del dispositivo el valor actual de cada uno de los objetos que tiene el esclavo y también se verá en la ventada estado el estado actual del dispositivo. En la figura 5.25 se muestra claramente esta ventana.

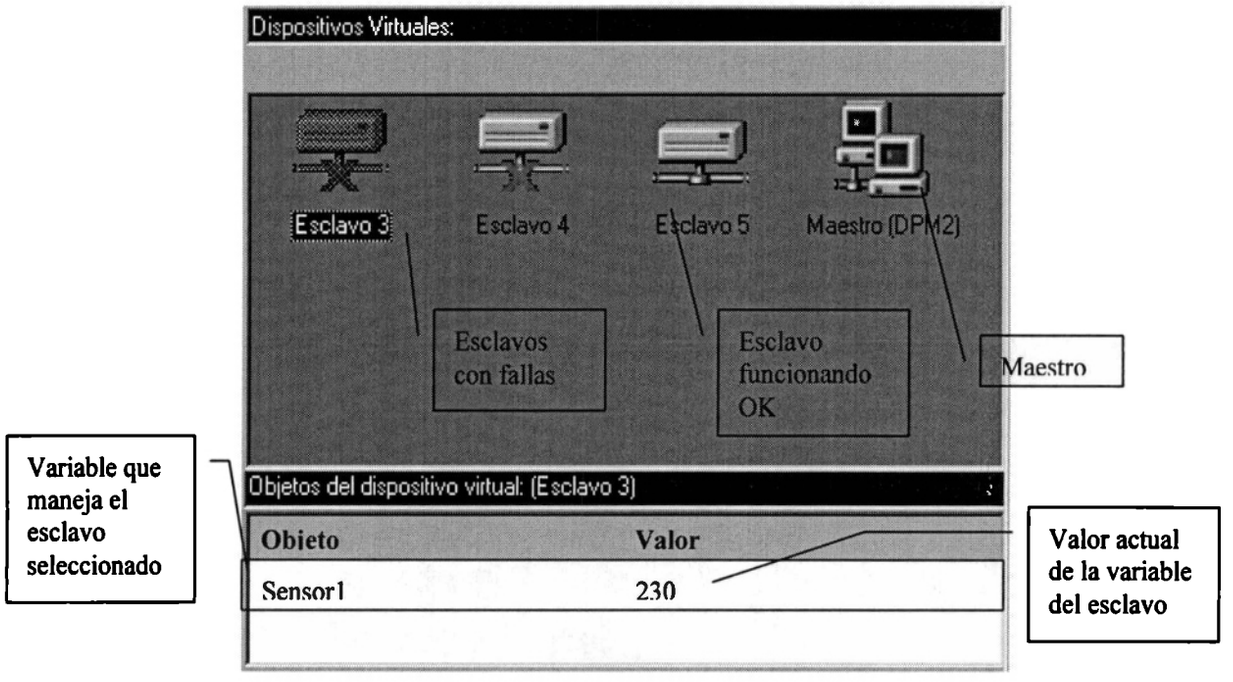


Figura 5.25 Ventana de dispositivos virtuales y los objetos de cada dispositivo

Objetos de los dispositivos virtuales: Esta ventana muestra en forma de tabla cada uno de los objetos que maneja el esclavo y el valor actual, el cual esta leyendo periódicamente por medio de la red. La 5.25 muestra esta ventana

Otra ventana importante es cuando se adiciona un nuevo esclavo a la red. Cada vez que se adiciona un nuevo esclavo, el sistema maestro debe leer un archivo con los drivers del esclavo, este archivo es un archivo de texto que cuenta con la siguiente información:

- Identificador del esclavo.
- Nombre del esclavo.
- Descripción.
- Fabricante.
- Variables que maneja.
- Longitud de cada variable y tipo de valor.

Una vez leído este archivo el sistema almacena esta información en la base de datos actualizando las tablas: esclavos y objetos del esclavo. La 5.26 muestra la ventana para leer los drivers de los esclavos.

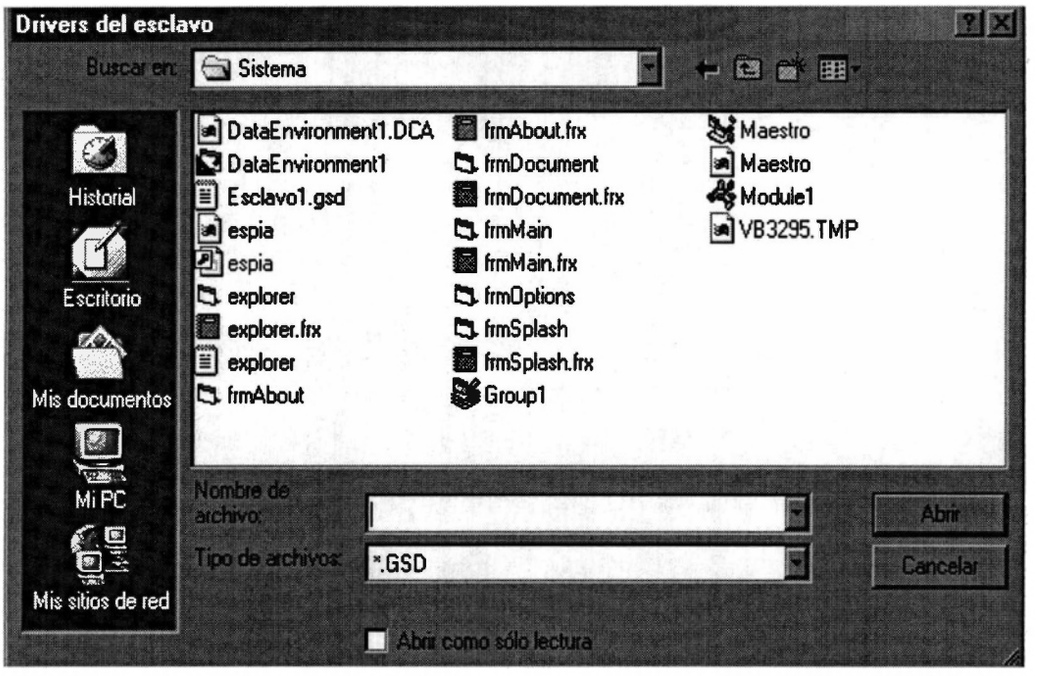


Figura 5.26 Ventana para leer los drivers del esclavo.

5.2.2 PROGRAMACIÒN DEL SISTEMA

El sistema se desarrolló con el lenguaje de programación Visual Basic 6.0 bajo Windows 95 o superior. A continuación se definirán los procedimientos y funciones más importantes del sistema, estos procedimientos y funciones realizan las actividades citadas en la sección 4.3. En el anexo 2 se encuentran los algoritmos desarrollados en Visual Basic. Los procedimientos y funciones (algoritmos que realizan una actividad especifica) son:

Inicio: Se encarga de la iniciación de sistema, la inicialización involucra, inicializar base de datos, timers, variables, constantes, banderas, registros, ventanas, etc. En esta parte el sistema se enlaza con Access para leer los datos de cada una de las tablas, estas tablas son: esclavos del maestro, variables de cada esclavo, maestros, timers y parámetros. Leídos estos datos, los actualiza en la pantalla principal y los muestra al usuario. En este procedimiento se crea una

matriz con toda la información de los esclavos, para posteriormente usarla en la construcción del frame que le envía a cada esclavo.

Control_De_Comunicaciones: Es el procedimiento que esta controlando la comunicación con los dispositivos de la red. Éste selecciona el estado al que debe saltar el programa, estos estados son los mostrados en la figura 5.18.

Estado_Idle: Una vez inicializada la red, el maestro entra en el estado idle, que es el estado en el cual el maestro escucha la red durante un tiempo especificado por el timer idle, si durante este tiempo el maestro no recibe ningún mensaje significa que no hay otro maestro con el control de la red o nadie posee el token, entonces este maestro toma el token y pasa al estado use_token.

Armar_Frame: Esta función es la que arma el frame que se va a enviar. El frame se arma de la siguiente manera:

SD	FC	DA	SA	INFO...	FCS	ED

Donde:

SD = Delimitador de inicio = 11011000

FC = Control del frame = Tipo de frame enviado, los tipos de frames son:

- Pasar_Token: Se indica con el número decimal 8 y quiere decir que se le va a enviar el token al maestro siguiente.
- Frame_sin_respuesta: Se indica con el número decimal 71, y se usa cuando se va a enviar un frame que no requiere respuesta.
- Frame_con_respuesta: Se indica con el número decimal 79. Es el más utilizado y se usa para enviar un frame solicitando respuesta.
- Frame_de_respuesta: Se indica con el número decimal 87 e indica que se esta respondiendo al frame de solicitud, es el que utilizan los esclavos cuando se le piden datos.
- Token_recibido: Se indica con el número decimal 49 y se usa cuando se responde al maestro diciendo que se recibió el token correctamente.

DA = Dirección destino = 1 byte con la dirección destino.

SA = Dirección fuente = 1 byte con la dirección del maestro.

INFO = Información = En este campo se pone el índice de cada variable del esclavo y el valor a enviar o una "p" si es solicitud de datos.

FCS = Secuencia de control

ED = Delimitador final = 11111100

Por ejemplo, si el maestro actual que tiene el índice 1, le va a enviar un frame a la estación con índice 3, solicitándole el valor actual de la variable 1, el frame quedara así:

SD =11011000

FC = Frame_con_respuesta = 79

DA = Dirección destino = 3

SA = Dirección fuente = 1

INFO = 1p

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	79	3	1	1p	255	252

Cuando el esclavo lea el frame verificará que el SD=216, DA=3 y SA=1, cuando lee que FC = 79 sabe que tiene que enviarle respuesta al maestro y el campo INFO=1p, indica que le esta pidiendo información de la variable 1. Entonces el esclavo arma el frame de respuesta de una manera similar la cual se explicará mas adelante.

La respuesta esperada por el maestro se especificara más adelante.

Estado_Use_Token: Cuando el maestro tiene el token entra a este estado, en este estado el maestro se comunica con cada uno de sus esclavos. El orden en que se comunica con los esclavos es de acuerdo al índice del esclavo, iniciando de menor a mayor. Cada vez que el maestro le envía un mensaje debe armar el frame que va a enviar, este frame lo arma llamando a la función armar frame. Después de enviado el frame el maestro pasa a un estado de espera de respuesta del esclavo. El maestro esta en el estado use token un tiempo determinado por el timer token_hold_timer.

Estado_Await_IFM_Response: En este estado el maestro entra a esperar respuesta cuando le envía un frame a un esclavo solicitándole información, si en determinado tiempo especificado por el timer Response_window_timer el maestro no recibe respuesta, intenta 2 veces más y si no recibe respuesta el maestro asume que el esclavo tiene problemas de comunicación y notifica con un mensaje de alarma. Si recibe respuesta pasa al procedimiento Verificar_frame_recibido

Verificar_Frame_Recibido: Si el maestro recibe respuesta, éste debe verificar que el frame venga correctamente. Esto lo hace verificando cada uno de los bytes del frame, siguiendo con el ejemplo mostrado en armar frame, la respuesta esperada debe ser la siguiente:

SD =11011000
FC = Frame_de_respuesta = 87
DA = Dirección destino = 1
SA = Dirección fuente = 3
INFO = 1x; donde x es el valor actual de la variable
FCS = 11111111
ED = 11111100

Pasando todo a números decimales el frame será:

216	87	1	3	1x	255	252

Si vemos detalladamente se puede ver que los campos DA y SA se invierten, el valor de FC = 87 ya que es respuesta del esclavo, y en INFO cambia el valor de "p" por el valor actual de la variable 1.

Estado_Pass_Token: Una vez se termina el tiempo de uso del token, el maestro debe pasar a enviar el frame al maestro siguiente notificándole que puede tomar el token o control de la red. Por ejemplo si el índice del maestro siguiente es 2, el frame a enviar es el siguiente:

SD =11011000
FC = Pasar_token = 8
DA = Dirección destino = 2

SA = Dirección fuente = 1

INFO =

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	8	2	1		255	252

Estado_Check_Token_Pass: Una vez el maestro envía el token al siguiente maestro, éste debe entrar a un estado de espera de respuesta, en este estado esta un tiempo especificado por el timer Response_window_timer, si durante este tiempo el maestro no recibe respuesta, éste asume que el otro maestro esta fuera de la red y pasa el token a otro maestro o lo toma él nuevamente si no existen más maestros en al red. Si recibe respuesta verificará que venga correctamente si viene correctamente pasa nuevamente al Estado_idle, siguiendo con el ejemplo anterior la respuesta debe ser la siguiente:

SD =11011000

FC = Token_recibido = 49

DA = Dirección destino = 1

SA = Dirección fuente = 2

INFO =

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	49	1	2		255	252

Esos son los procedimientos más importantes del sistema, existen otros que realizan actividades como: actualizar información en pantalla, actualizar información de los esclavos, reportar errores, leer drivers, etc...

5.2.3 TIMERS

El sistema cuenta con diversos timers que están monitoreando el correcto funcionamiento del sistema, por compatibilidad con la norma los nombres de los timers se dejan en inglés. Estos timers son:

1. Recibir: Timer que esta periódicamente revisando si ha recibido datos.
2. Token_hold_timer: Tiempo que la estación maestra dura con el token.
3. Bus_Idle_Timer: Tiempo que la estación esta en estado de espera antes de entrar al estado de reclamar o tomar el token.
4. Claim_Timer: Tiempo que una estación escucha antes de enviar un frame pidiendo el token.
5. Response_Window_Timer: Tiempo que una estación espera antes de transmitir el próximo frame.
6. Token_Pass_Timer: Tiempo que la estación tiene el token antes de pasarlo al sucesor.

Los timers se implementan con el objeto timer de Visual Basic, los timers usados en el proyecto se muestran en la figura 5.27 donde están en el orden enumerados anteriormente.

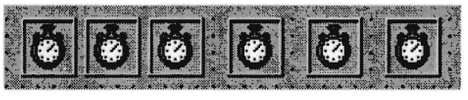


Figura 5.27 timers del sistema.

El valor de cada timer esta almacenado en la tabla timers de la base de datos. Cuando el sistema se inicializa, estos timers son actualizados leyéndolos de la base de datos. El valor de estos timers puede ser cambiado por el usuario del sistema de acuerdo a sus necesidades.
Los timer se activan al cumplirse el tiempo especificado en la propiedad intervalo, al activarse el timer se realiza la actividad para la cual esta pensado, por ejemplo cuando se activa el timer Token_hold_timer el maestro sabe que se termino el tiempo que puede usar el token y pasa al procedimiento de pasar el token al maestro siguiente.

5.3 INTERFAZ FÍSICA ENTRE RS-232 Y RS-485

Las computadoras normalmente vienen con un puerto de comunicación serial RS-232 y Visual Basic tiene un objeto que se llama Mscomm, este objeto es el encarga de enviar y recibir los frames construidos en la etapa de programación explicada anteriormente. En la figura 5.28 se muestra este objeto y las propiedades que maneja.

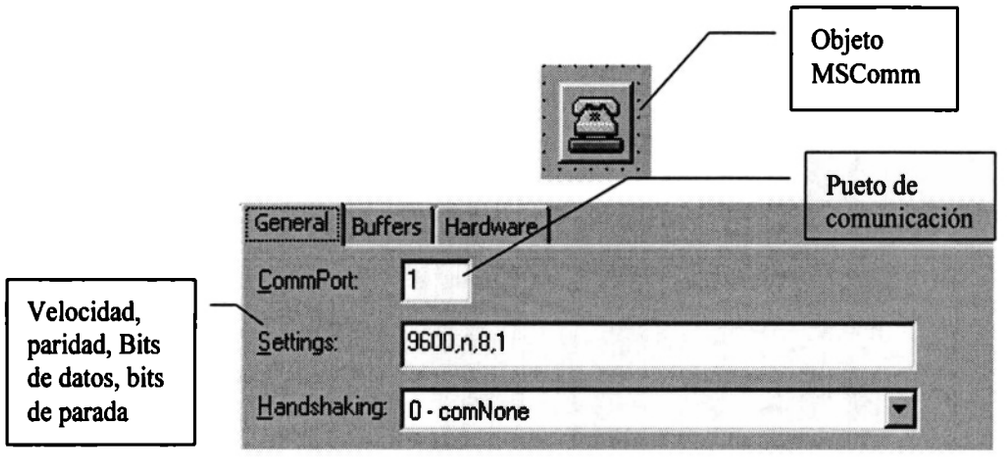


Figura 5.28 Objeto MSComm y sus propiedades.

Es importante aclarar que todos los dispositivos conectados a la red deben cumplir con las mismas características de velocidad de transmisión que en este caso es 9600 baudios, paridad, bits de datos que son 8 y bits de parada.

El timer recibir esta periódicamente interrogando el buffer de este objeto para verificar si le han llegado datos, si hay datos nuevos, estos datos son tomados y el buffer del objeto es vaciado.

Como se ha dicho en capítulos anteriores, PROFIBUS opera con la norma RS-485, por lo tanto es necesario desarrollar un dispositivo que convierta las señales RS-232 a RS-485 y viceversa. En la sección 4.2.4.2 se presento una alternativa para solucionar este problema, sin embargo a lo largo del proyecto se ha encontrado otra alternativa más practica y más fácil de controlar. En la figura 5.29 se muestra el esquema de este convertidor, y en la figura 5.30 se muestra el dispositivo desarrollado. Este convertidor esta formado de un circuito integrado MAX232 que toma las señales del puerto serial RS-232 del computador (Estas señales son de –12V a 12V) y las

convierte a voltajes TTL (5V), una vez convertidas estas señales, se las envía al circuito integrado 75176 que es el que las convierte a las señales diferenciales manejadas por la norma RS-485 como se explico en la sección 4.2.4. También se usa un inversor 7404 que es el que se encarga de conmutar el 75176 para que reciba o envíe datos según sea necesario. Los conectores finales se modificaron para que fueran ambos DB9, inicialmente era un DB9 y un header de 3 pines, de los cuales un extremo va al computador y el otro extremo a la red PROFIBUS. También se desarrollo una caja para esta interfaz, la cual ayuda a proteger el prototipo del ambiente exterior. Este prototipo opera a un voltaje de 5V.

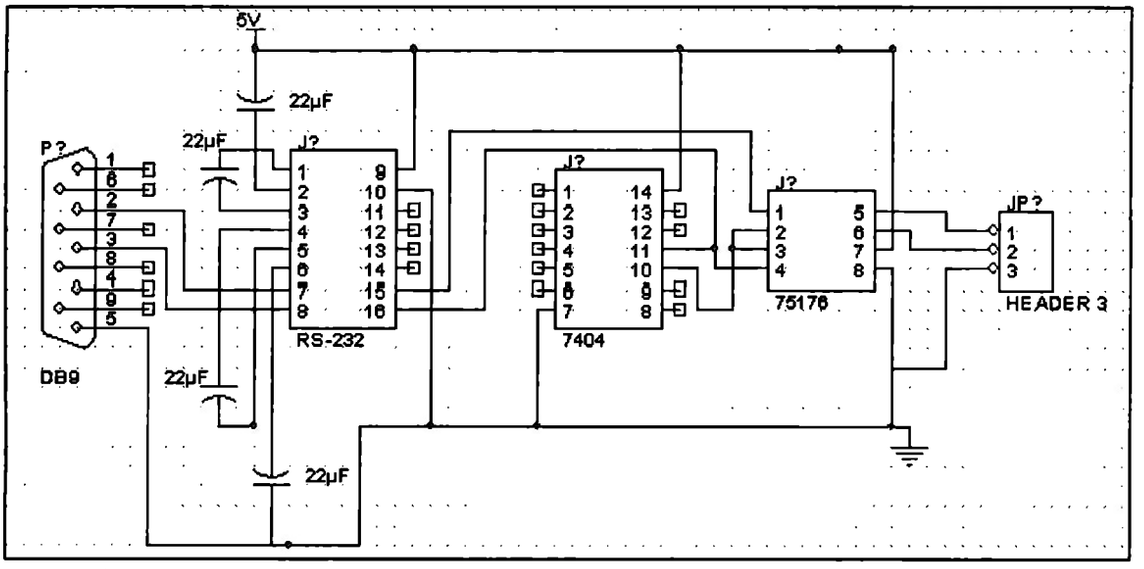


Figura 5.29 Esquema del convertidor de RS-232 – RS-485 y viceversa

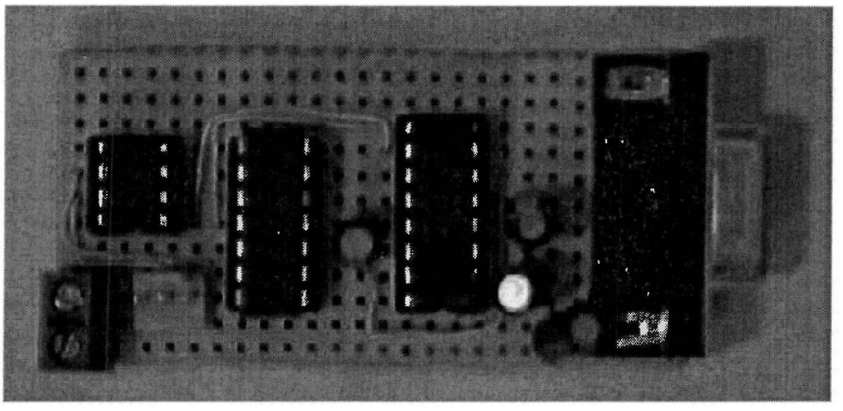


Figura 5.30 Prototipo del convertidor RS-232 – RS-485 y viceversa.

5.3.1 CABLE Y CONECTORES UTILIZADOS

Inicialmente se estaba utilizando un alambre común y conectores header de 3 pines y la red presentaba ciertos errores de comunicación. Para mejoras esto se opto por utilizar cable M4X24 y conectores DB9 y se vio una mejora notoria en el funcionamiento de la red. En la figura 5.31 se muestra el cable utilizado anteriormente y el cable usado actualmente con los conectores DB9.

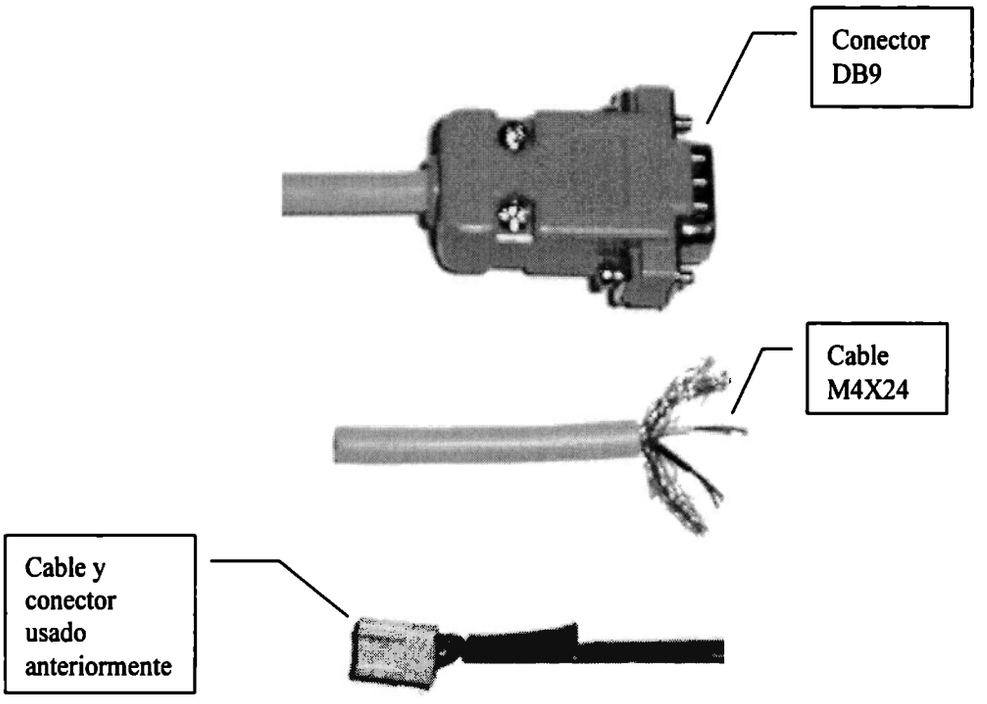


Figura 5.31 Foto con los cables y conectores

5.4 DESARROLLO DEL SISTEMA ESCLAVO

Los esclavos, que son los dispositivos que realizan las actividades de automatización y control, en este caso serán microcontroladores Intel 8051. Para el microcontrolador se desarrollo un programa en lenguaje ensamblador, este programa se encargara de interpretar el frame enviado por el maestro, enviar respuesta si es solicitada y de desarrollar actividades propias de automatización y control. Al recibir el frame enviado por el maestro, el esclavo tomará las siguientes acciones:

1. Detectar si es un frame valido
2. Detectar si el frame va dirigido a él (leyendo la dirección destino).
3. Verificar que los datos se los esta pidiendo su propio maestro (verificando el campo de dirección de origen).
4. Detección de fallas en el frame.
5. Leer los datos solicitados por el maestro, estos datos pueden ser las variables o el estado del esclavo.
6. Generar y enviar el frame de respuesta con los datos solicitados por el maestro.
7. Activar alarmas de inactividad de su maestro.
8. Almacenar los valores de las variables que maneja, para poderlas comunicar a su maestro.
9. Desarrollar la actividad de control correspondiente.
10. Entre otras.

5.4.1 MICROCONTROLADORES (INTEL 8051)

Un microcontrolador es todo un "sistema mínimo" dentro de un sólo dispositivo, lo cual ofrece un enorme panorama hacia el mundo de la compatibilidad. Este dispositivo contiene: Un CPU (basado principalmente en un microprocesador de 4, 8 ò 16 bits), puertos paralelos de entrada y salida, puerto serie, timers, contadores, memorias y en algunos casos hasta convertidores analógicos digitales, todo esto dentro de un solo chip [20]. Un microcontrolador esta encaminado básicamente hacia aplicaciones concretas en donde, el espacio, y el número de componentes es mínimo, además, los cambios a ampliaciones futuras del sistema son casi nulos. A diferencia con

un microprocesador que es destinado a sistemas donde su expansión a corto o mediano plazo es factible. A pesar de que un microprocesador es más rápido que un microcontrolador para la ejecución de sus instrucciones, en la mayoría de los casos es necesario interconectarlo con dispositivos periféricos [20], lo cual complica un poco el trabajo con estos.

En este proyecto se opto por trabajar con el microcontrolador Atmel AT89S8252 que es una versión mejorada del modelo Intel 8051. Es importante aclarar que el sistema funciona correctamente con cualquier tipo de microcontrolador de cualquier familia.

El microcontrolador Atmel AT89S8252 tiene las misma características que el Intel 8051, pero adiciona otras características muy importantes que son: 8K bytes de memoria flash reprogramable que es donde residirá el programa, 2K bytes de memoria EEPROM, 128 bytes más de memoria RAM interna, un timer más. Las características de este microcontrolador son [21]:

- Compatible con el Intel 8051.
- Microcontrolador de 8 bits.
- 8K bytes de memoria flash.
- 2K bytes de memoria EEPROM.
- Voltaje de operación de 4V a 6V.
- 256 bytes de memoria RAM interna.
- 32 líneas I/O programables.
- 3 Timers/Counters de 16 bits.
- Nueve fuentes de interrupción.
- Puerto serial programable.

En la figura 5.31 se muestra un diagrama de bloques del microcontrolador AT89S8252. Para ver con más profundidad las características de funcionamiento de este microcontrolador puede consultar el manual de la referencia [21].

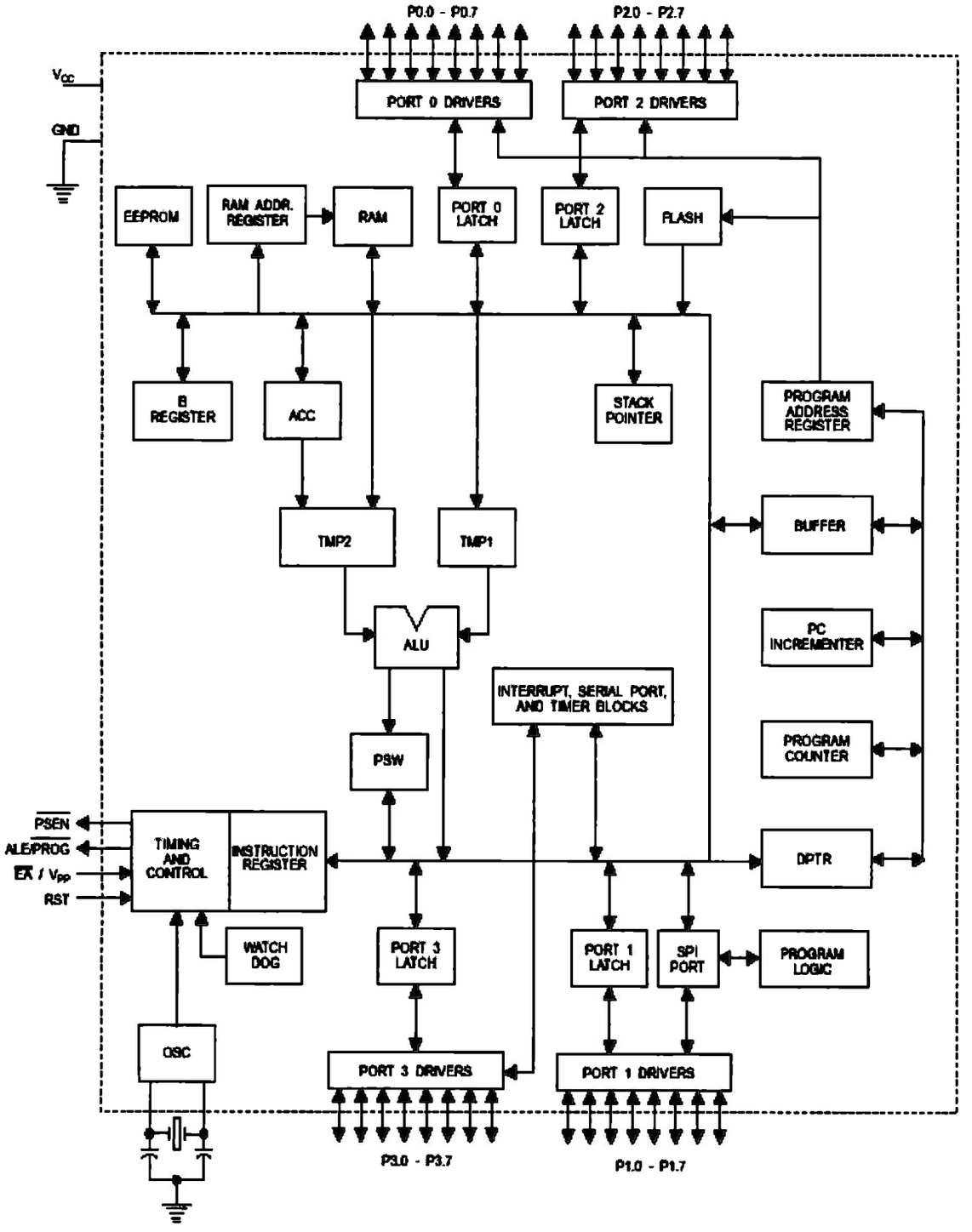


Figura 5.31 Diagrama de bloques del microcontrolador Atmel AT89S8252 [21].

5.4.2 PROTOTIPO DEL SISTEMA ESCLAVO

El esclavo aparte de estar realizando una actividad de control, debe estar constantemente recibiendo y enviando información al sistema maestro, el microcontrolador Atmel AT89S8252 cuenta con un puerto de comunicación serial que servirá para realizar esta actividad, mas sin

embargo se debe hacer la interfaz entre el microcontrolador y la norma RS-485 que es la utiliza PROFIBUS, para esto se utiliza el circuito integrado 75176. La figura 5.32 muestra un esquema general del funcionamiento del sistema esclavo.

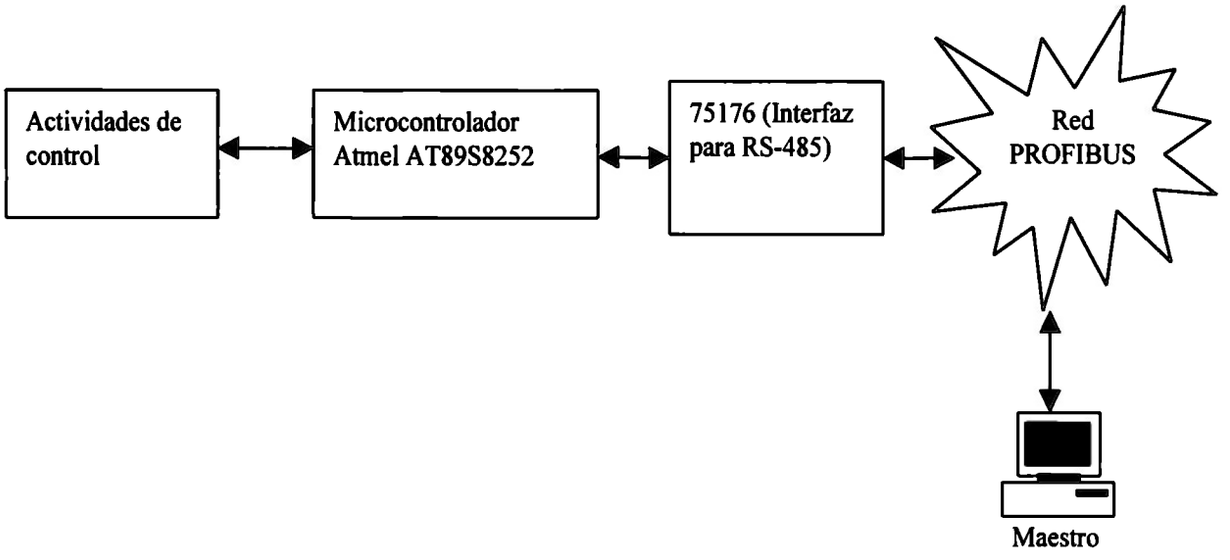


Figura 5.32 Esquema general de la interfaz del microcontrolador.

El circuito electrónico del sistema esclavo se montará en una tableta de baquelita con pistas de cobre para prototipos. El microcontrolador esta conectado directamente al circuito integrado 75176 que como ya hemos dicho es el que convierte las señales de transmisión de microcontrolador a las señales diferenciales manejadas por RS-485 como se explico en la sección 4.2.4. El puerto p1.7 es el encargado de conmutar el 75176 para que envíe o reciba datos según sea necesario, para realizar esta actividad el puerto p1.7 se debe conmutar por medio de software. La conexión para la red se hace con conectores DB9, estos van directamente conectados al 75176. Este circuito opera a un voltaje de 5V. En la figura 5.33 se muestra un esquema detallado de las conexiones del sistema esclavo. Como pede verse en el esquema, el sistema queda prácticamente con los cuatros puertos de I/O libres para realizar las actividades de control asignadas. En la figura 5.34 se muestra el prototipo creado para el sistema esclavo.

5.4.2 PROGRAMACIÒN DEL SISTEMA ESCLAVO

Todos los esclavos contaran con un programa que se encargara de mantener la comunicación con el sistema maestro. Este programa se desarrolló en lenguaje ensamblador, las líneas de código del programa se encuentran en el anexo 3. La figura 5.35 muestra el diagrama de flujo del programa.

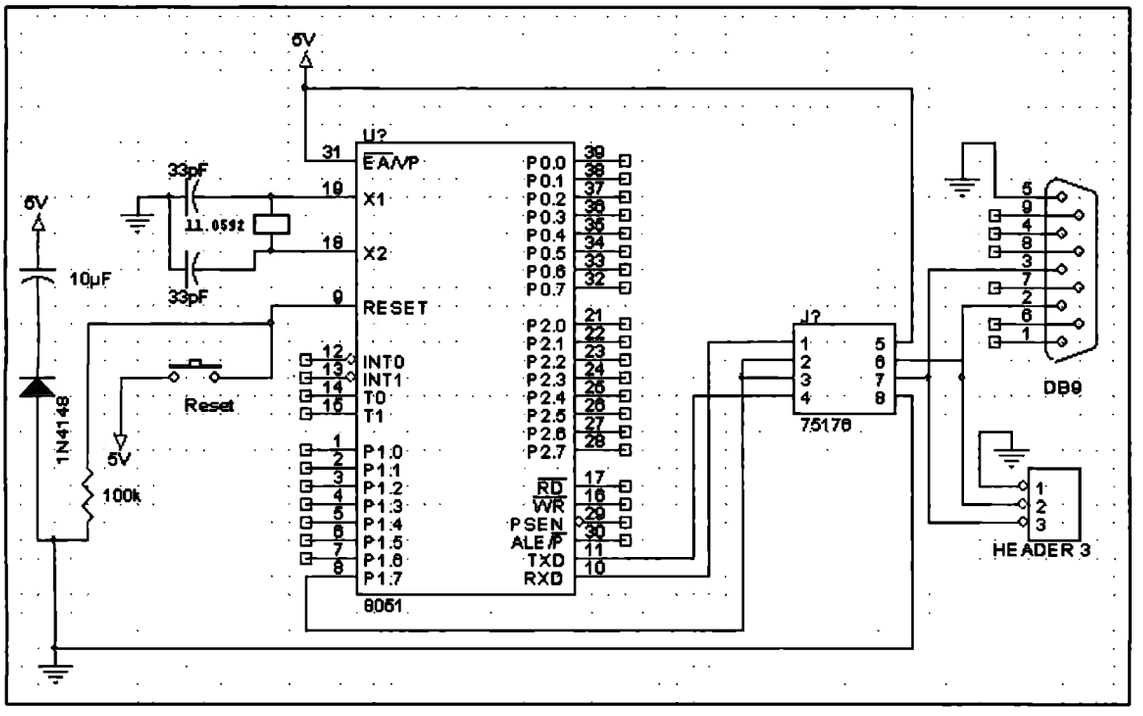


Figura 5.33 Esquema del sistema esclavo

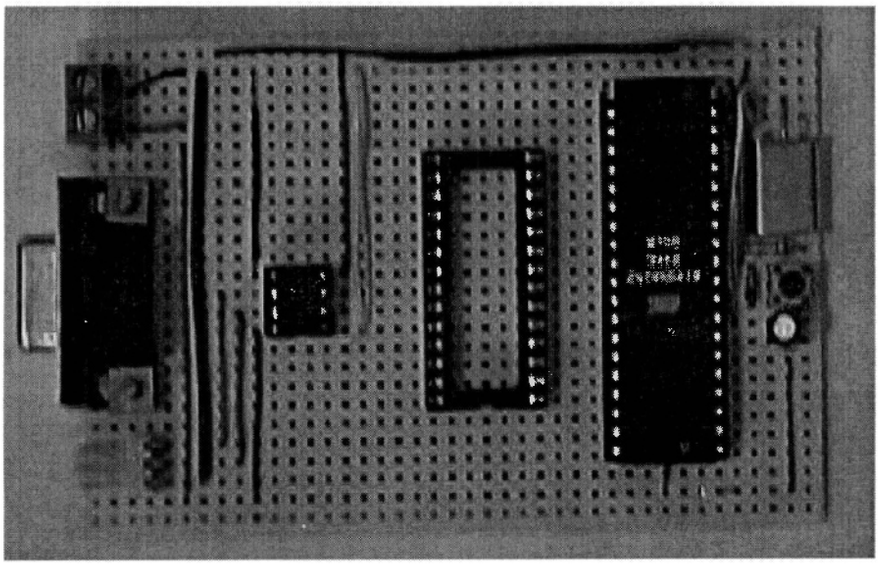


Figura 5.28 Prototipo del sistema esclavo

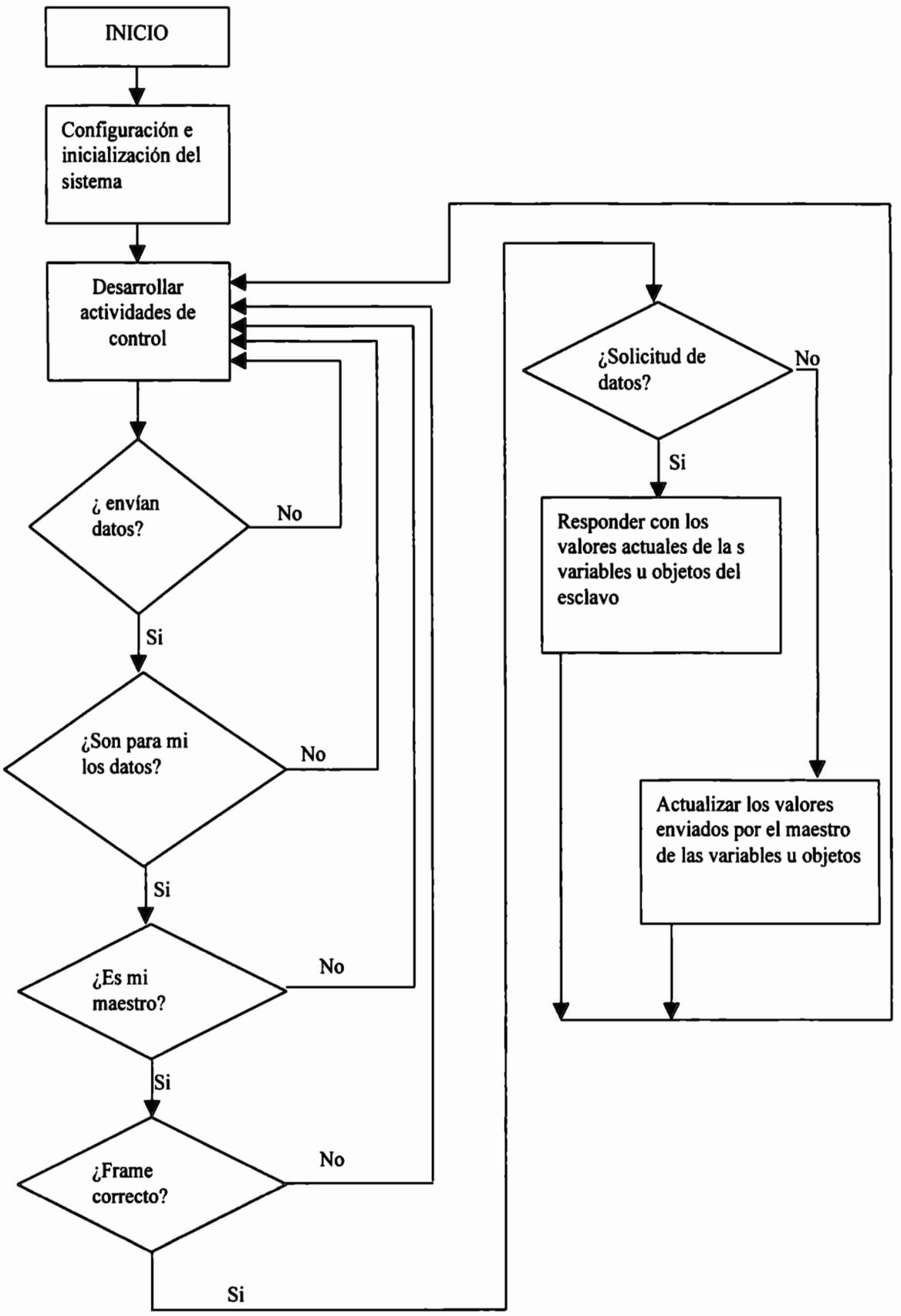


Figura 5.35 Diagrama de flujo del programa de los sistemas esclavos.

A continuación se describirá el funcionamiento del programa. Cuando el microcontrolador es encendido lo primero que debe hacer es configurar el sistema, esta configuración incluye:

- Asignación de un índice único para el esclavo. Este índice es el que lo identificará en la red.
- Asignación de direcciones de memoria para los objetos que maneja el esclavo, estos objetos son las variables que esta controlando el esclavo (como velocidad de un motor, temperatura, etc...). Cada objeto debe tener un índice iniciando de 1 en adelante.
- Configuración del puerto serial. La configuración involucra definir la velocidad de comunicación, bits de datos, bit de paridad, bit de inicio, bit de parada. Estos parámetros deben ser los mismos para todos los dispositivos conectados a la red. En nuestro caso estos parámetros son:
    - Velocidad: 9600 b/s.
    - Bits de datos: 8 bits.
    - Bit de paridad: 0 bit.
    - Bit de inicio: 1 bit.
    - Bit de parada: 1 bit.
- Configuración de timers para la velocidad de comunicación. Se usara el timer 1 en modo 2.
- Inicializar la interrupción para el puerto serial y el timer
- Enviar señal al circuito 75176 para que esta recibiendo o transmitiendo, esto lo hace conmutando el puerto p1.7.

Una vez que ha inicializado el sistema, el programa pasa a realizar la actividad de control especificada. Estando realizando las actividades de control, cuando reciba datos por el puerto serial se produce una interrupción, esta interrupción llama al subprograma que verifica el mensaje recibido, este subprograma verifica que el mensaje venga correctamente. Revisa el campo SD para verificar que es un frame, el campo FC para ver si es un mensaje que requiere respuesta, el campo DA que indica la dirección destino y lo compara con su índice, si coinciden puede tomar el frame, el campo SA para ver si pertenece a la dirección del maestro, si cumple todo lo citado anteriormente, toma la información enviada y actualiza el sistema. Siguiendo con el ejemplo mostrado en le procedimiento Armar_frame, en donde el maestro le envía un frame al esclavo 3

pidiéndole información sobre el valor actual de la variable 1. Entonces el frame que envía el maestro es el siguiente:

SD =11011000

FC = Frame_con_respuesta = 79

DA = Dirección destino = 3

SA = Dirección fuente = 1

INFO = 1p

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	79	3	1	1p	255	252

El esclavo al recibir este frame almacena cada byte de la dirección 20h del microcontrolador en adelante. Posteriormente pasa a revisar cada uno de los componentes del frame. Lo primero que revisa es el campo SD que debe ser igual a 216 si no es igual abandona el procedimiento y vuelve a realizar sus actividades de control, si es igual revisa el campo DA revisando si el frame va dirigido a él, en este caso DA debe ser igual a 1, si es así mira el campo SA viendo que el frame venga de su maestro que debe ser 1, revisado esto el esclavo ya ha verificado que es un frame correcto y que viene para él y que lo envía su maestro designado, después revisa si el campo FC = 79 quiere decir que debe enviar respuesta a su maestro, El campo INFO le indica que le esta pidiendo información de la variable 1. Entonces el esclavo procede a armar el frame de respuesta, para armarlo intercambia las direcciones de memoria DA con SA, ya que ahora la dirección destino será su maestro y la dirección fuente será el mismo, cambia el campo FC con el valor 87 que significa Frame_de_respuesta. El frame quedara así:

SD =11011000

FC = Frame_de_respuesta = 87

DA = Dirección destino = 1

SA = Dirección fuente = 3

INFO = 1x

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	87	1	3	1x	255	252

En el campo INFO el valor de x será el valor actual de la variable 1. Una vez armado el frame, el programa conmuta el p1.7 para indicarle al 75176 que va a empezar a transmitir e inicia la transmisión del frame.

Por el lado del maestro, este sería el frame que estaría esperando, si llega un frame diferente, el maestro lo asume como un error de comunicación y vuelve a enviar el frame 2 veces más y si no llega este mensaje esperado, el maestro asume que el esclavo tiene problemas de comunicación y reporta esto con un mensaje de alarma.

Si el frame que envío el maestro tuviera el campo INFO = 1B, y el esclavo esta trabajando como actuador, el esclavo asumiría que el maestro le esta diciendo que actualice el valor de la variable 1 con el valor B. Entonces el esclavo hace la actualización y le envía un frame de respuesta similar al anterior pero con el campo INFO = 1B.

En el anexo 3 se muestra detalladamente el programa, para un mayor entendimiento cada línea de código tiene comentarios sobre su función.

6 PRUEBAS DEL SISTEMA

Para comprobar el funcionamiento del sistema se hicieron pruebas con dos maestros y cada maestro con 2 esclavos. Los esclavos trabajarán como sensores o como actuadores, los que actúan como sensores leen un número de 1 byte por el puerto 2 del microcontrolador y los que actúan como actuadores muestran en un a pantalla de cristal líquido el valor de un número enviado por el maestro al esclavo. La figura 6.1 muestra un esquema de la prueba a realizada.

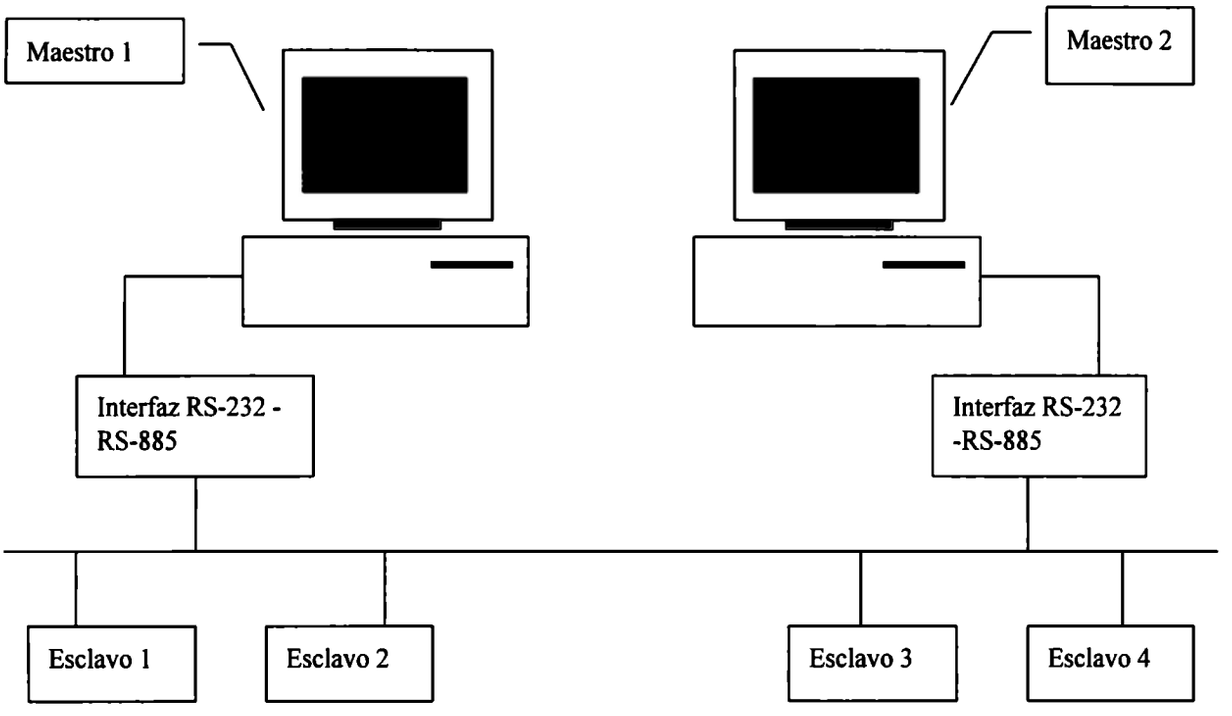


Figura 6.1 Esquema de la prueba realizada

Cada microcontrolador lee por el puerto 2 un número que es seleccionado por medio de un DIP de 8 pines. La figura 6.2 muestra una foto de la prueba hecha.

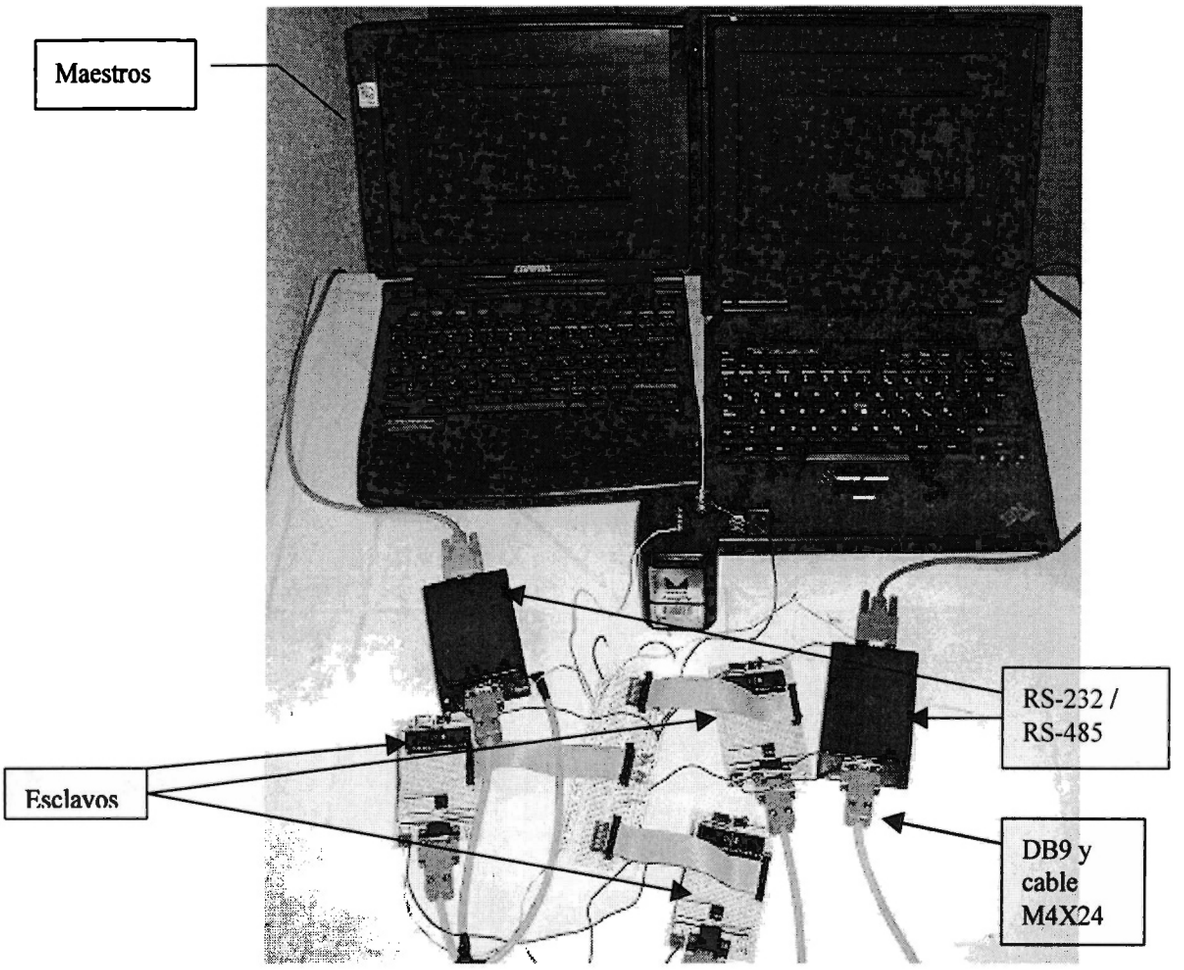


Figura 6.2 Foto de la prueba hecha

En la figura 6.3 se muestra el prototipo para los sensores que van conectados al puerto de cada microcontrolador.

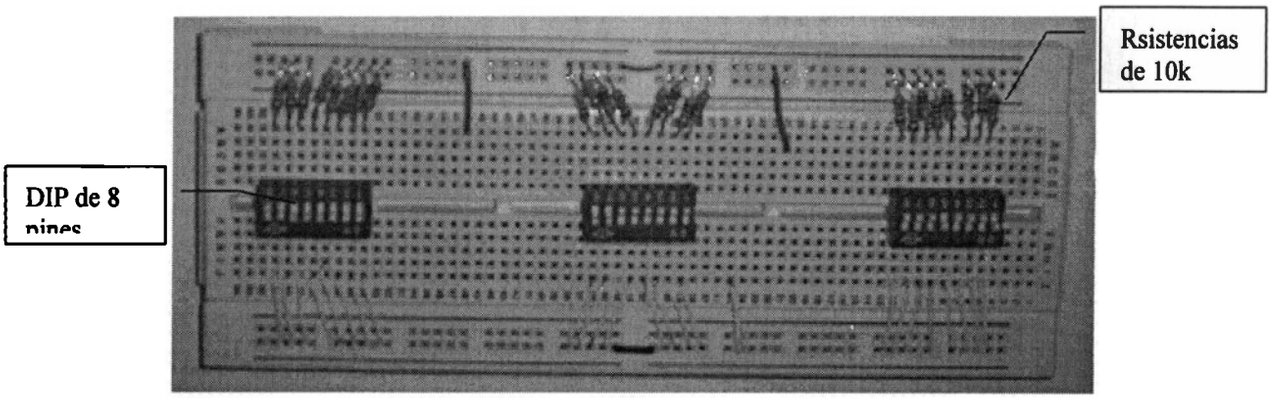


Figura 6.3 Foto de los sensores usados

El microcontrolador esta periódicamente revisando el valor del puerto 2 que cambia al modificar los valores del DIP, posteriormente almacena este valor en una dirección de memoria y cuando se comunique con el maestro le envía este valor..

Haciendo la prueba, el sistema funciona así:

Los identificadores de los dispositivos conectados a la red son:

Maestro 1 = 1
Maestro 2 = 2
Esclavo 1 = 3
Esclavo 2 = 4
Esclavo 3 = 5
Esclavo 4 = 6

Los esclavos están haciendo las actividades de control asignadas y están escuchando la red para verificar si les están enviando información. Los dos maestros inicializan el sistema leyendo la base de datos actualizando timers, variables, pantalla, etc... después de inicializar la red entran a un estado de espera especificado por el timer Bus_idle_timer, en este tiempo de espera los maestros están escuchando la red para ver si alguien esta transmitiendo, si alguien esta transmitiendo, el maestro asume que ya hay otro maestro con el token y por lo tanto seguirá esperando a que le llegue un frame notificándole que ya puede tomar el token. Si terminado el tiempo del timer Bus_idle_timer y el maestro no escucha nada, esto quiere decir que no hay otro maestro en la red con el token, por lo tanto el maestro toma el control de la red y empieza a transmitir a cada uno de sus esclavos. Entonces si por el ejemplo el control lo toma el maestro 1, éste entra al estado Use_token y empezará a transmitirle a los esclavos que tiene asignados que son los esclavos 1 y 2 respectivamente. La secuencia de transmisión la hace en orden ascendente de acuerdo al índice de cada esclavo, entonces empieza enviándole un frame al esclavo 1 que tiene índice 3, el frame que envía será:

SD =11011000
FC = Frame_con_respuesta = 79
DA = Dirección destino = 3

SA = Dirección fuente = 1

INFO = 1p

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	79	3	1	1p	255	252

Cuando el esclavo 3 recibe este frame procederá a responder con el frame:

SD =11011000

FC = Frame_de_respuesta = 87

DA = Dirección destino = 1

SA = Dirección fuente = 3

INFO = 1x (x es el valor actual de la variable 1)

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	87	1	3	1x	255	252

Una vez el maestro recibe este frame verifica que viene correctamente, actualiza los datos en pantalla y actualiza la base de datos. Posteriormente procede a enviar un frame al esclavo 2 que tiene el índice 4, el frame enviado sería:

SD =11011000

FC = Frame_con_respuesta = 79

DA = Dirección destino = 4

SA = Dirección fuente = 1

INFO = 1p

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	79	3	4	1p	255	252

Cuando el esclavo 4 recibe este frame procederá a responder con el frame:

SD =11011000

FC = Frame_de_respuesta = 87

DA = Dirección destino = 1

SA = Dirección fuente = 4

INFO = 1x (x es el valor actual de la variable 1)

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	87	1	4	1x	255	252

Una vez el maestro recibe este frame, verifica que viene correctamente, actualiza los datos en pantalla y actualiza la base de datos. Si el maestro 1 tuviera mas esclavos seguiría comunicándose con ellos de la misma manera, en este caso no tiene más esclavos por lo tanto procede a pasarle el token al maestro 2, el frame para pasar el token al maestro es el siguiente:

SD =11011000

FC = Pasar_token = 8

DA = Dirección destino = 2

SA = Dirección fuente = 1

INFO =

FCS = 11111111

ED = 11111100

Pasando todo a números decimales el frame será:

216	8	2	1		255	252

Cuando el maestro 2, que debe estar en estado de espera, recibe este frame sabe que le están enviando el token, por lo tanto responde al maestro 1 diciéndole que ya tomo el token y empieza y comunicarse con los esclavos 3 y 4 de la misma manera que lo hizo el maestro 1. El maestro 1 recibe este mensaje y asume que el maestro 2 ya tiene el token y entra en estado de espera hasta que el maestro 2 le envía el token nuevamente. El frame de respuesta del maestro 2 es el siguiente:

SD =11011000
FC = Token_recibido = 49
DA = Dirección destino = 1
SA = Dirección fuente = 2
INFO =
FCS = 11111111
ED = 11111100

Pasando todo a números decimales el frame será:

216	49	1	2		255	252

Este ciclo se repite indefinidamente.

Cuando se adiciona un nuevo esclavo a la red, se actualiza la base de datos y el maestro la próxima vez que tenga el token empieza a comunicarse con ese esclavo de la misma forma mostrada anteriormente.

A continuación se mostrarán los resultados de las pruebas hechas. Para realizar las pruebas se insertaron 2 variables en el código del programa, una va contando las comunicaciones correctas y otra cuenta las comunicaciones erradas. Se realizaron 3 pruebas cada una con una duración de 5 y 6 horas aproximadamente.

Prueba 1

Esta prueba se realizó con un maestro y tres esclavos comunicándose con éste. La prueba se hizo por un tiempo de 6 horas aproximadamente y se tomaron 5841 muestras. Los resultados de esta prueba se muestran en la figura 6.4

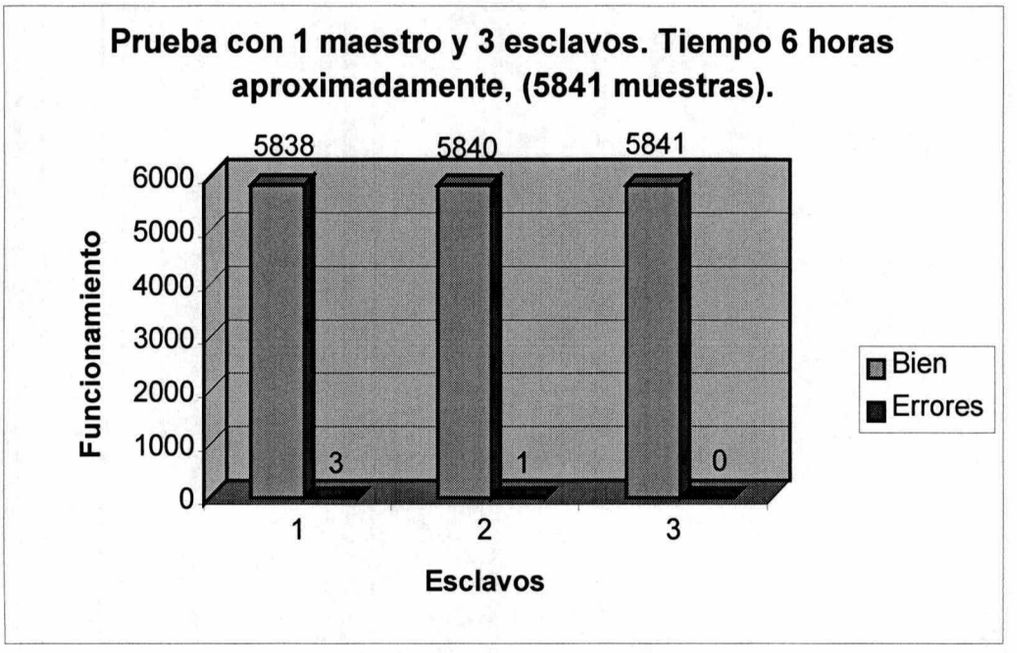


Figura 6.4 Prueba 1. Un maestro con tres esclavos.

Prueba 2

Esta prueba se realizó con dos maestro y tres esclavos, el maestro 1 tiene los esclavos 1 y 3 y el maestro 2 tiene el esclavo 2. La prueba se hizo por un tiempo de 6 horas aproximadamente y se tomaron 3722 muestras. Los resultados de esta prueba se muestran en la figura 6.5.

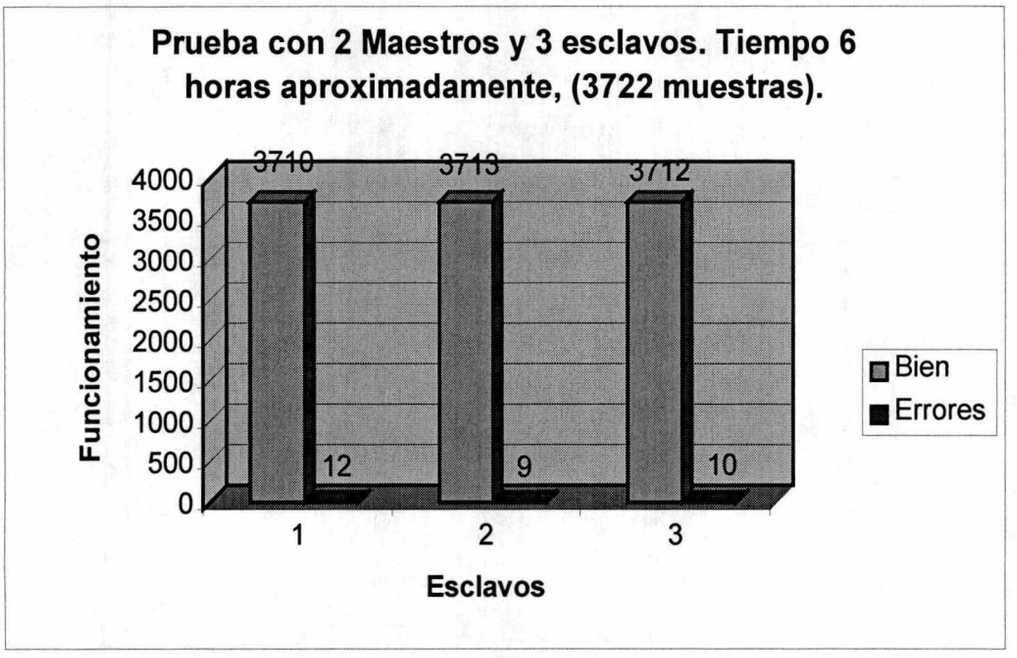


Figura 6.5. Prueba 2. Dos maestros con tres esclavos.

Prueba 3

Esta prueba se realizó con un dos maestros y tres esclavos y con el alambre utilizado antes de que se mejorara el cableado de la red. El maestro 1 tiene los esclavos 1 y 3 y el maestro 2 tiene el esclavo 2. La prueba se hizo por un tiempo de 6 horas aproximadamente y se tomaron 3722 muestras. Los resultados de esta prueba se muestran en la figura 6.5.

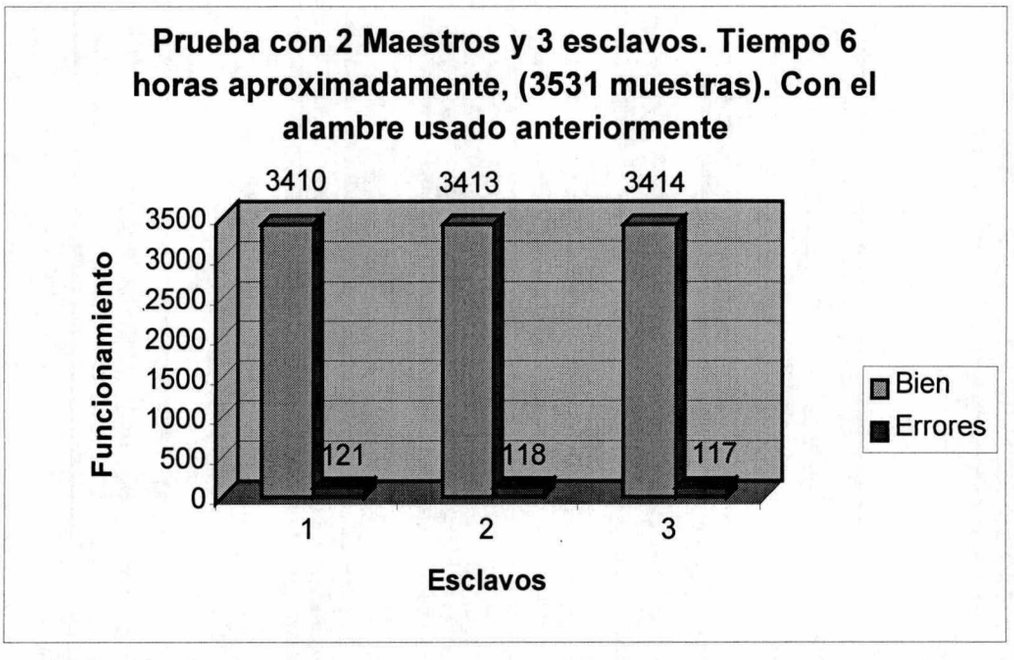


Figura 6.6 Prueba 3. Dos maestros, tres esclavos con el alambre usado anteriormente.

7 CONCLUSIONES Y RECOMENDACIONES A FUTURO

Las redes de comunicación industrial juegan un papel muy importante a la automatización de sistemas de manufactura de las empresas actuales, sin estas sería muy difícil tener un control total de la producción.

Este trabajo se basó en la implantación de uno de los protocolos de comunicación industrial más utilizados, este protocolo es PRFOFIBUS. Terminando se puede concluir que el protocolo de comunicación PROFIBUS, es un protocolo muy versátil y esta bien estructurado. Esto nos ayuda a una fácil implantación de una red basada en este protocolo. Este protocolo además cuenta con la ventaja de estar conformado por normas ya existentes, lo cual es una ventaja en comparación con otros protocolos..

En el trabajo se desarrolló un software en Visual Basic 6.0, este software implementa el protocolo PROFIBUS para comunicarse con dispositivos esclavos que en este caso son microcontroladores Intel 8051 y mostrar en pantalla el estado de las variables que maneja cada microcontrolador. El software cuenta con interfaz de usuario muy amigable y esta hecho de tal manera que para el usuario sea transparente el protocolo. El software también esta hecho para que trabaje en un ambiente multimaestro, en donde cada maestro es administrador de la red en determinado momento y tiene a su cargo diversos esclavos. Se realizó un prototipo para el sistema maestro que convierte las señales de RS-232 a RS-485 y se vio que funciona correctamente.

En el trabajo también se desarrollo un prototipo para los sistemas esclavos, basado en el microcontrolador Atmel AT89S8252 compatible con el Intel 8051. Este prototipo hace el papel de esclavo de la red y funciona correctamente como tal.

Realizada la prueba se vio que los maestros se comunican con cada uno de sus esclavos y pasan el frame al maestro siguiente en el tiempo asignado, muestran la información de las variables que manejan los esclavos, si ocurre un error lo notifican al instante, respetan los tiempos asignados a cada timer. Los esclavos interpretan adecuadamente el frame, responden cuando se les solicita información, leen adecuadamente los sensores y realizan todas las actividades esperadas.

En la prueba se hizo una muestra estadística y se vio que con el cambio de cable mejoró sustancialmente el funcionamiento de la red, por lo tanto es muy importante seleccionar un cable adecuado para este tipo de redes. También se vio que cuando se hizo la prueba con dos maestros hubo más errores de comunicación que con un solo maestro, esto puede ser por que se están presentando errores en la toma del token de los maestros o también puede ser que entre más estaciones se conecten a la red habrá mas probabilidad de errores. Otra fuente de error puede ser directamente de los esclavos, ya que en determinados momentos se sobrecalienta el circuito integrado 75176 que es una parte muy importante de la interfaz de comunicación.

También puede ser probable que ciertos errores estén ocurriendo por demoras en el funcionamiento del software, ya que los lenguajes de programación con ambientes visuales como Visual Basic en ocasiones ocupan muchos recursos del computador sobre todo de memoria y tienden a bloquear o producir demoras en el funcionamiento de éste, y estas demoras pueden causar problemas de comunicación en el sistema maestro.

Para trabajos futuros se recomienda la utilización de un cable y conectores más adecuados ya que se vio que mejoran notablemente el funcionamiento de la red.

También se recomienda mejorar el dispositivo esclavo para que pueda operar como maestro en caso de que éste tenga problemas y quede fuera de la red. Es bueno tener en cuenta que Siemens ya esta trabajando en esclavos con estas características.

Siemens también esta mejorando el protocolo para que se comunique de manera transparente con otro tipo de redes (como Ethernet) y para que pueda trabajar por Internet. Sería muy interesante poder adaptar este proyecto a ese tipo de infraestructura. Una opción muy interesante para hacer lo anterior en este proyecto sin mayores cambios, sería utilizar los maestros como puentes y enlazarlos a cualquier otro tipo de red valiéndonos de la capacidad que tienen Windows y Visual Basic para enlazarse a Internet o a otro tipo de red. De esta forma los maestros seguirán operando normalmente con el software ya hecho y habría otra aplicación que estaría tomando esta información por medio de la base de datos y la estaría compartiendo con otras redes o con Internet.

Finalmente pienso que esta es un área muy interesante e importante para las empresas del mundo actual y creo que se debe de seguir trabajando en este tipo de proyectos para poder llegar a desarrollar un producto con un margen de error aceptable y con muy buenas cualidades para la implantación de sistemas de manufactura.

REFERENCIAS Y BIBLIOGRAFÍA

[1] FRED HALSALL. Data communications, computer networks and open systems. Addison-Wesley. 1992.

[2] HOOPER, et al. Diseño de redes locales. Addison-Wesley Iberoamericana. 1989

[3] JAMES MARTIN, et al. Local area networks, Architectures and implementations. Prentice hall. 1994.

[4] U. REMBOLD, et al. Compute integrated manufacturing and engineering. Addisson Wesley. 1993. Pag. 258-292.

[5] VINCENT C. JONES. MAP/TOP Networking. Mc Graw Hill. 1987. Pag. 1-23

[6] ANSI/IEEE ESTANDAR 802.4. Local Area Networks, Token passing bus access method. IEEE. 1985.

[7] ANSI/IEEE ESTANDAR 802.5. Local Area Networks, Token ring access method. IEEE. 1985.

[8] JOSEP BALCELLS, JOSE LUIS ROMERAL. Autómatas programables. Alfaomega Maracombo. 1997. Pag 271-384.

[9] PROFIBUS, Technical description. Order No. 4.002

[10] www.fieldbus.neta

[11] http://vigna.cimsi.cim.ch/tai/BCD/in/BCD1_2.html

[12] NATIONAL INSTRUMENTS. AutomationView. Volumen 4. Número 2. Verano 1999. Pag. 4-6.

[13] SMAR, Fieldbus Tutorial.

[14] NATIONAL INSTRUMENTS. AutomationView. Volumen 5. Número 3. Otoño 2000. Pag. 4-5.

[15] NATIONAL INSTRUMENTS. AutomationView. Volumen 4. Número 3. Otoño 1999. Pag. 4-5.

[16] ROGER S. PRESSMAN. Ingeniería del Software. Tercera edición. McGraw-Hill. 1993.

[17] EDWARD YOURDON. Análisis Estructurado Moderno. Prentice hall. 1993.

[18] RICHARD BARKER, CLIFF LONGMAN. CASE*METHOD, Function and Process Modelling, ORACLE. ADDISON WESLEY. 1992.

[19] RICHARD BARKER. CASE*METHOD, Entity Relationship Modelling, ORACLE. ADDISON WESLEY. 1992.

[20] ALEJANDRO VEGA S. Manual del microcontrolador 8051.

[21] ATMEL. AT89S8252, 8-Bit microcontroller with 8K bytes flash memory.

ANEXO 1 DICCIONARIO DE TERMINOS USADOS EN EL PROYECTO

ANSI: Amarican National Standars Institute

Baud: Medida de la velocidad de transmisión, equivale al número de bytes por segundo transmitidos.

Bus: Un canal de conexión. Típicamente es un cableado, con uno o más conductores.

CCITT: International Telegraph and Telephone Consultative Comite.

CIM: Computer Integrated Manufacturing.

EPROM: Erasable programmable read-only memory.

EEPROM: Electrical Erasable programmable read-only memory.

Ethernet: Popular red de área local diseñada y hecha por Xerox Corp.

Frame: Grupo de bits enviados serialmente sobre un canal de comunicación de acuerdo a unos parámetros establecidos en el protocolo.

IEEE: Institute of Electrical and Electronics Engineers.

Latencia: Intervalo de tiempo en que una estación espera para transmitir.

Mensaje: Información que contiene datos y un formato ordenado, son enviados en las redes de comunicación.

PDU: Protocol Data Unit. Paquete de información intercambiado entre dos capas de una red.

Protocolo: Reglas que definen el formato, tiempos, secuencia y control de errores del intercambio de datos de una rede de comunicación.

UART: Universal asynchronous receiver/transmitter.

ANEXO 2 CÓDIGO DEL SISTEMA MAESTRO

```
Public fMainForm As frmMain
'Declaración de objetos de la base de datos
Public Db As Database
Public rstimers As Recordset
Public rsesclavos As Recordset
Public RsMaestros As Recordset
Public RsMisEsclavos As Recordset
Public RsObjetos As Recordset
Public RsDOE As Recordset
Public RsTemporalObjetos As Recordset

'Definicion de los timers
Public Bus_idle_timer As Double
Public Claim_timer As Double
Public Response_window_timer As Double
Public Contention_timer As Double
Public Token_pass_timer As Double
Public Token_rotation_timer As Double
Public Token_hold_timer As Double
'Definición de variables usadas en la norma, se definen en ingles por
'compatibilidad con la norma
Public In_ring As Boolean 'Indica si el frame recibido venia para este maestro
Public NS_know As Boolean
Public PS_Know As Boolean
Public Lowest_address As Boolean
Public Sole_active_station As Boolean 'Error en la recibida del frame
Public Just_had_token As Boolean 'Es verdadero cuando la estación acaba de enviar token, y falso cuando recibe
Public Rx_protocolo_frame As Boolean 'Bandera que indica que se recibio un frame token
Public Rx_data_frame As Boolean 'indica que se recibio un frame de datos
'Constantes del tipo de frame recibido o enviado
Public Const Token = 8 'Indica que el valor de FC es un token
Public Const Pedir_token = 0
Public Const Poner_sucesor = 12
Public Const Quien_sigue = 3
Public Const Frame_sin_respuesta = 71
Public Const Frame_con_respuesta = 79
Public Const Frame_de_respuesta = 87
Public Const Pasar_Token = 48
Public Const Token_Recibido = 49
```

```
Public PS As String 'Dirección de la estación anterior
Public NS As String 'Dirección de la proxima estación
Public Noise_burst As Boolean 'Es true cuando se recibe frame erroneo
Public Bus_quiet As Boolean 'Es true cuando no se esta recibiendo datos
Public Estado As String 'proximo estado a ejecutar
Public Tipo_De_Frame_Enviado As Integer

Public Ruta As String
Public N As Integer
'Definición de las variables del frame
Public SD As String 'Delimitador de inicio (8 bits)
Public FC As String 'Control del frame (8 bits)
Public DA As String 'Dirección destino (2-6 octetos bits)
Public SA As String 'Dirección de precedencia (2-6 octetos)
Public INFO As String 'Información (0 o mas octetos
Public FCS As String 'Secuencia de comprobación de frame (4 octetos)
Public ED As String 'Delimitador final  (8 bits)
Public numero
Public recibido As String
Public N_Retransmisiones As Integer 'especifica el numero de veces que se retransmitie un frame
Public N_Retransmisiones_Sucesor As Integer
'Declaración de los registros de prioridad
Public Pr As Integer
Public Rr As Integer
Public Pm As String 'Prioridad del frame que se va a enviar
Public PDU As Boolean 'Es 1 cuando hay PDUs para transmitir
Public Frame_Recibido As String 'ultimo frame recibido
Public Frame_Nuevo As String 'frame que esta llegando
Public Maestro As String 'Identificador del maestro. TS
Public N_esclavos As String 'Numero de esclavos de este maestro
Public Frame_para_enviar As String 'frame a enviar
Public Longitud_Frame_Recibido As Integer 'Longitud de todo el frame
Public Longitud_Info_Recibida As Integer 'Longitud de Información del frame
Public Frames_Recibidos(31, 2) As String 'Frames recibidos de todos los esclavos
Public Numero_Frames_Recibidos As Integer
Public Mis_esclavos(31, 5) As String 'Matriz con la inforamción de los esclavos
'los paremetros son Indice, nombre, Numero de objetos que maneja, info recien recibida
',las varaibles que maneja y valores a enviar (puede ser p o otro diferente)
Public Longitud_Frame_Esperado As Integer
Public Esclavo_Actual As Integer 'Indice que me indica el esclavo actual de la matriz
Public PDUs_En_Cola(100, 1) As String 'Cola de PDU que voy a enviar, es el frame y indice a quien va dirigido
Public Indice_PDU As Integer 'Frame que envie
Public YoInicio As Boolean 'Indica si inica usando la red
Public Funcionando As Boolean 'Indica el sitema funcione o no
Sub Inicio()
   'Este procedimiento inicia todas las variables del sistema
   explorer.RichTextBox1.LoadFile ("c:\windows\conexion.txt")
   Ruta = Trim(explorer.RichTextBox1.Text)
   Dim k As Integer
   Dim rs4 As Recordset
   Set Db = OpenDatabase(Ruta & "espia.mdb")
   Set rstimers = Db.OpenRecordset("Timers")
   Set rsesclavos = Db.OpenRecordset("esclavos")
   Set RsMaestros = Db.OpenRecordset("maestros")
   Set rs4 = Db.OpenRecordset("select * from maestros where SoyYo = true")

   Set RsObjetos = Db.OpenRecordset("Objetos", dbOpenDynaset)
   Set RsDOE = Db.OpenRecordset("DiccionarioObjetosEsclavos", dbOpenDynaset)
```

```
  Set RsTemporalObjetos = Db.OpenRecordset("TemporalObjetos", dbOpenDynaset)
  'numero de estaciones
  Indice_PDU = 0 'Inicio el indice de la cola de PDUs
  If Not rsesclavos.EOF Then
    rsesclavos.MoveLast
  End If
  If Not RsMaestros.EOF Then
    RsMaestros.MoveLast
  End If
  If Not rsesclavos.EOF Then
    N = rsesclavos.RecordCount + RsMaestros.RecordCount
  Else
    N = RsMaestros.RecordCount
  End If
  'CAptura de el ID de este maestro
  Maestro = rs4.Fields("Indice").Value
  If rs4!sucesor <> "" Then
    NS_know = True
    NS = rs4!sucesor
  Else
    NS_know = False
  End If
  If rs4!antecesor <> "" Then
    PS_Know = True
    PS = rs4!antecesor
  Else
    PS_Know = False
  End If

  'recordset con los esclavos de este maestro
  Set RsMisEsclavos = Db.OpenRecordset("SELECT * FROM esclavos where maestro = '" & Trim(Maestro) & "'
order by indice;", dbOpenDynaset)
  If Not RsMisEsclavos.EOF Then
    RsMisEsclavos.MoveLast
    N_esclavos = RsMisEsclavos.RecordCount
  Else
    N_esclavos = 0
  End If
  'Inicialización de matrix con los esclavos
  If Not RsMisEsclavos.EOF Then
    RsMisEsclavos.MoveFirst
    For k = 0 To (N_esclavos - 1)
      Mis_esclavos(k, 0) = RsMisEsclavos.Fields("Indice").Value
      Mis_esclavos(k, 1) = RsMisEsclavos.Fields("Nombre").Value
      Set RsDOE = Db.OpenRecordset("select * from DiccionarioObjetosEsclavos where Esclavo = '" &
RsMisEsclavos.Fields("Indice").Value & "'", dbOpenDynaset)
      Mis_esclavos(k, 4) = ""
      Do While Not RsDOE.EOF
        'guardo las vaiables que maneja el esclavo
        Mis_esclavos(k, 4) = Mis_esclavos(k, 4) + RsDOE!NombreObjeto + ";"
        RsDOE.MoveNext
      Loop
      'Almaceno el número de variables que manejo
      Mis_esclavos(k, 2) = Str(RsDOE.RecordCount)
      Mis_esclavos(k, 5) = ""
      For i = 1 To Mis_esclavos(k, 2) Step 1
        'Envio una "p" por cada variable, diciendole al esclavo que me
        'envie el valor de sus objetos
        Mis_esclavos(k, 5) = Mis_esclavos(k, 5) + "p"
```

```
        Next i
        RsMisEsclavos.MoveNext
      Next k
    End If
    Esclavo_Actual = -1 'indice de la matrix de esclavos, la inicio con el esclavo -1, para evitar problemas en
use_token.
    'Inicio de los timers
    Bus_idle_timer = rstimers.Fields("valor").Value
    rstimers.MoveNext
    Claim_timer = rstimers.Fields("valor").Value
    rstimers.MoveNext
    Response_window_timer = rstimers.Fields("valor").Value
    rstimers.MoveNext
    Contention_timer = rstimers.Fields("valor").Value
    rstimers.MoveNext
    Token_pass_timer = rstimers.Fields("valor").Value
    rstimers.MoveNext
    Token_rotation_timer = rstimers.Fields("valor").Value
    rstimers.MoveNext
    Token_hold_timer = rstimers.Fields("valor").Value

    'Inicio de los timers de el formulario principal
    explorer.Bus_idle_timer.Interval = Bus_idle_timer
    explorer.Claim_timer.Interval = Claim_timer
    explorer.Response_window_timer.Interval = Response_window_timer
    explorer.Contention_timer.Interval = Contention_timer
    explorer.Toke_pass_timer.Interval = Token_pass_timer
    explorer.Token_rotation_timer.Interval = Token_rotation_timer
    explorer.Token_hold_timer.Interval = Token_hold_timer

  End Sub

  Sub Main()
    'función principal del sistema, es la que se ejecuta primero
    frmSplash.Show
    frmSplash.Refresh
    Set fMainForm = New frmMain
    Load fMainForm
    Unload frmSplash
    fMainForm.Show
    Funcionando = False
    Control_De_Comunicacion 'llama al procedimiento que controla los estados
  End Sub
  Sub Control_De_Comunicacion()
    Dim Salir As Boolean
    Estado_Offline 'Invoca el estado offline para inicializar el sistema
    Salir = False
    Do While Not Salir
      DoEvents
      If Funcionando Then 'procedimiento que esta llamando constantemente los estados
        Seleccionar_Estado
      End If
    Loop

  End Sub
  Sub Seleccionar_Estado()
    'procedimiento para seleccionar el estado siguiente
      Select Case Estado
      Case "use_token"
```

```
        explorer.Token_hold_timer.Enabled = False
        explorer.Token_hold_timer.Enabled = True
        explorer.Token_hold_timer.Tag = ""
        In_ring = True
        Estado_Use_Token 'Va al estado use token
      Case "offline"
        Estado_Offline
      Case "claim"
        Estado_Claim_Token
      Case "idle"
        explorer.Bus_idle_timer.Enabled = False
        explorer.Bus_idle_timer.Enabled = True
        explorer.Bus_idle_timer.Tag = ""
        Estado_Idle
      Case "check_access_class"
        Estado_Check_Access_Class
      Case "await_IFM_response"
        explorer.Recibir.Enabled = True 'activo recibir datos
        explorer.Response_window_timer.Tag = ""
        explorer.Response_window_timer.Enabled = False
        explorer.Response_window_timer.Enabled = True 'activo el timer de ventana de respuesta
        Estado_Await_IFM_Response 'Estado para esperar respuesta de la otra estación
      Case "pass_token"
        Estado_Pass_Token
      Case "check_token_pass"
        Estado_Check_Token_Pass
      End Select
End Sub
Sub Estado_Offline()
  'Este estado se encarga de la inicialización de variables, timers...
  Lowest_address = False
  Sole_active_station = False
  Bus_quiet = True
  Just_had_token = False
  In_ring = True
  N_Retransmisiones = 2 'indico a 2 el numero que se retransmitira un frame si falla
  N_Retransmisiones_Sucesor = 2 'retransmisiones al sucesor en caso de que falle
  Rx_protocolo_frame = False 'indico que no ha llegado ningún frame
  Estado = "idle"
End Sub

Sub Estado_Idle()
  'En este estado la estación esta escuchando
  Dim Salir As Boolean
  Dim i As Integer
  Salir = False
  Do While Not Salir
    DoEvents
    If Rx_protocolo_frame Then 'llego un frame
      If (SA = PS) And (DA = Maestro) And FC = Pasar_Token And In_ring Then
        'procedimiento cuando se recibe un token
        Sole_active_station = False 'No hay error de recepción
        Rx_protocolo_frame = False
        Bus_quiet = True
        Salir = True
        'MsgBox "YO ACABO DE RECIBIR EL TOKEN"
        Call Armar_Frame(PS, Token_Recibido, "")
        Enviar_Datos (Frame_para_enviar)
        Estado = "use_token"
```

```
        ElseIf Noise_burst Then
          Rx_protocolo_frame = False
          Bus_quiet = True
          explorer.Bus_idle_timer.Enabled = False
          explorer.Bus_idle_timer.Enabled = True
          explorer.Bus_idle_timer.Tag = ""
          'Noise_burst = False
        End If
      ElseIf explorer.Bus_idle_timer.Tag = "termine" Then
         explorer.Bus_idle_timer.Tag = ""
         If Bus_quiet And YoInicio Then
          'No he recibido el token y ya es tiempo.
          'paso a claim para resolver el problema
          explorer.Bus_idle_timer.Tag = ""
          explorer.Bus_idle_timer.Enabled = False
          explorer.Bus_idle_timer.Enabled = True
          Estado = "claim"
          Salir = True
        End If
      End If

  Loop

End Sub
Sub Estado_Demand_In()
  'Cuando se recibe un frame de solocitud sucesor se entra a este estado
End Sub
Sub Estado_Claim_Token()
  Estado = "use_token"
End Sub

Sub Estado_Use_Token()
  'Se entra a este estado cuando se recibe el token, desde este estado se envian
  'frames a los esclavos
  Dim Salir As Boolean
  Do While Not Salir
    DoEvents
    If explorer.Token_hold_timer.Tag <> "termine" Then
      'Envio frames a cada esclavo
      'despues de envair cada frame reinicio el timer de espera de respuesta
      Esclavo_Actual = Esclavo_Actual + 1
      If Esclavo_Actual > (N_esclavos - 1) Then
        'Ya se cominico con todos lo esclavos
        Salir = True
        Estado = "check_access_class"
        Esclavo_Actual = -1
      Else
        'Comunicación con el esclavo siguiente
        Tipo_De_Frame_Enviado = Frame_con_respuesta
         'Armo el frame con el esclavo actual, y con "ppp", diciendole que solo pido sus variables
        Call Armar_Frame(Mis_esclavos(Esclavo_Actual, 0), Frame_con_respuesta,
Mis_esclavos(Esclavo_Actual, 5))
         'explorer.Recibir.Enabled = False
         Enviar_Datos (Frame_para_enviar)
         'Despues de enviar frame paso al estado esperar respuesta.
         Bus_quiet = True 'indica que no se ha recibido respuesta
         Estado = "await_IFM_response"
         Salir = True
      End If
```

```
    Else
      explorer.Token_hold_timer.Tag = ""
      explorer.Token_hold_timer.Enabled = False
      Salir = True
      Estado = "check_access_class"
    End If

  Loop
End Sub
Sub Estado_Await_IFM_Response()
  'en este estado se espera respuesta de la estación a la que se le envio el frame
  Dim Salir As Boolean
  Salir = False
  Do While Not Salir
    DoEvents
    If Tipo_De_Frame_Enviado = Frame_con_respuesta Then
      If explorer.Response_window_timer.Tag = "termine" Then 'termino tiempo de espera
        If Bus_quiet Then 'indica que no he recibido datos
          If (N_Retransmisiones > 0) Then
            'No se ha recibido respuesta y se reenviaran datos
            N_Retransmisiones = N_Retransmisiones - 1
'           MsgBox "voy a reenviar datos"
            Enviar_Datos (Frame_para_enviar)
            explorer.Response_window_timer.Tag = ""
            explorer.Response_window_timer.Enabled = False
            explorer.Response_window_timer.Enabled = True
          Else
            'No se ha recibido respuesta y ya se reenvio n numero de veces
            Salir = True
            Estado = "use_token"
            'reportar el error de comunicación
            Mis_esclavos(Esclavo_Actual, 3) = ""
            Error_De_Comunicacion (Mis_esclavos(Esclavo_Actual, 0))
            Actualizar_Esclavo_Virtual (Mis_esclavos(Esclavo_Actual, 0))
            N_Retransmisiones = 2
          End If
        Else
          If Noise_burst Then
            'el frame venia con ruido o defectuoso,
            'se vuelve a enviar
            If (N_Retransmisiones > 0) Then
              'No se ha recibido respuesta y se reenviaran datos
              N_Retransmisiones = N_Retransmisiones - 1
'             MsgBox "voy a reenviar datos"
              Enviar_Datos (Frame_para_enviar)
              explorer.Response_window_timer.Tag = ""
              explorer.Response_window_timer.Enabled = False
              explorer.Response_window_timer.Enabled = True
            Else
              'No se ha recibido respuesta y ya se reenvio n numero de veces
              Salir = True
              Estado = "use_token"
              'reportar el error de comunicación
              Mis_esclavos(Esclavo_Actual, 3) = ""
              Error_De_Comunicacion (Mis_esclavos(Esclavo_Actual, 0))
              Actualizar_Esclavo_Virtual (Mis_esclavos(Esclavo_Actual, 0))
              N_Retransmisiones = 2
            End If
            Noise_burst = False
```

```
            'reportar que el frame enviado no era el deseado
            Salir = True
          Else
            If (INFO <> "ppp") And (INFO <> "") Then
              'si se recibio respuesta correcta
              N_Retransmisiones = 2
              Mis_esclavos(Esclavo_Actual, 3) = INFO
              Bus_quiet = True
              Rx_protocolo_frame = False
              Estado = "use_token"
              Salir = True
              'reportar la comunicación correcta
              Comunicacion_Correcta (Mis_esclavos(Esclavo_Actual, 0))
              'actualizo los datos en la pantalla y la base de datos
              'MsgBox "RECIBI CORRECTAMENTE"
              Actualizar_Esclavo_Virtual (Mis_esclavos(Esclavo_Actual, 0))
              Mis_esclavos(Esclavo_Actual, 5) = ""
              'Actualizo las variables de envio con "ppp..."
              For i = 1 To Mis_esclavos(Esclavo_Actual, 2) Step 1
                Mis_esclavos(Esclavo_Actual, 5) = Mis_esclavos(Esclavo_Actual, 5) + "p"
              Next i
            Else
              Bus_quiet = True
              Rx_protocolo_frame = False
              Noise_burst = False
            End If
          End If
        End If
      End If
    ElseIf Tipo_De_Frame_Enviado = Frame_sin_respuesta Then
      Estado = "use_token"
      Salir = True
    End If
  Loop

End Sub
Sub Estado_Check_Access_Class()
  'Estado para pasar el token al siguiente maestro. si existe

  If NS_know Then
    'existe un sucesor y se le pasa el token
    explorer.Toke_pass_timer.Enabled = False
    explorer.Toke_pass_timer.Enabled = True
    explorer.Toke_pass_timer.Tag = ""
    Estado = "pass_token"
  '  MsgBox "estoy en acheck access classs y voy para pass_toekn"
  Else
    'No existe sucesor, sigue con el token
    explorer.Token_hold_timer.Enabled = False
    explorer.Token_hold_timer.Enabled = True
    explorer.Token_hold_timer.Tag = ""
    In_ring = True
    Estado = "use_token"
  End If
End Sub
Sub Estado_Pass_Token()
  'En este estado se le pasa el token al siguiente maestro
  'MsgBox "Estoy en estado passssst token"
  Dim Salir As Boolean
```

```
  Salir = False
  Do While Not Salir
    DoEvents
    If explorer.Toke_pass_timer.Tag <> "termine" Then
      Call Armar_Frame(NS, Pasar_Token, "")
      'explorer.Recibir.Enabled = False
      Enviar_Datos (Frame_para_enviar)
'      MsgBox "envie datos al sucesooooorrrrrr"
      'Despues de enviar frame paso al estado esperar respuesta.
      Bus_quiet = True 'indica que no se ha recibido respuesta
      explorer.Recibir.Enabled = True 'activo recibir datos
      explorer.Response_window_timer.Tag = ""
      explorer.Response_window_timer.Enabled = False
      explorer.Response_window_timer.Enabled = True
      Estado = "check_token_pass"
      Salir = True
      NS_know = True
    Else
      'NS_know = False
      Estado = "idle"
      Salir = True
'      MsgBox "El maestro sucesor no me envio respuesta, estoy en pass_token"
    End If
  Loop
End Sub
Sub Estado_Check_Token_Pass()
  'Estado para esperar la respuesta del siguiente maestro diciendo que tomo el token
  Dim Salir As Boolean
  Salir = False
' MsgBox "Estoy en check token passs y voy a ver si el sucesro respondio"
  Do While Not Salir
    DoEvents
    If explorer.Response_window_timer.Tag = "termine" Then
      If (Bus_quiet = False) Then
        'recibi respuesta, debo verificar si viene bien
        Rx_protocolo_frame = False
        Bus_quiet = True
        If Noise_burst Then
          Noise_burst = False
          If N_Retransmisiones_Sucesor > 0 Then
            'El frame venia con ruido
            Enviar_Datos (Frame_para_enviar)
            N_Retransmisiones_Sucesor = N_Retransmisiones_Sucesor - 1
            'Despues de enviar frame paso al estado esperar respuesta.
            Bus_quiet = True 'indica que no se ha recibido respuesta
            explorer.Recibir.Enabled = True 'activo recibir datos
            explorer.Response_window_timer.Tag = ""
            explorer.Response_window_timer.Enabled = False
            explorer.Response_window_timer.Enabled = True
            'reenvio y me quedo en este estado esperando respuesta
'            MsgBox "El sucesor no me responde, hay ruido "
          Else
            N_Retransmisiones_Sucesor = 2
            Salir = True
            Estado = "idle"
'            MsgBox "ya retrnsmito 2 veces y nada"
            'Notificar que no hubo respuesta del sucesor
          End If
        Else
```

```
            'If FC = Token_Recibido Then
              'El frame venia correcto
              Salir = True
              N_Retransmisiones_Sucesor = 2
              YoInicio = False
              Estado = "idle"
              ' MsgBox "El sucesor respondio OK guwyyyyyyyyyyyyy voy para idle"
            'End If
          End If
        Else
          'No recibi respuesta, intentare de nuevo
          Rx_protocolo_frame = False
          Bus_quiet = True
      '    MsgBox "No he recibido respueta voy a reinttentar"
          'Recibi respuesta
          If N_Retransmisiones_Sucesor > 0 Then
            'El frame venia con ruido
            Enviar_Datos (Frame_para_enviar)
            N_Retransmisiones_Sucesor = N_Retransmisiones_Sucesor - 1
            'Despues de enviar frame paso al estado esperar respuesta.
            Bus_quiet = True 'indica que no se ha recibido respuesta
            explorer.Recibir.Enabled = True 'activo recibir datos
            explorer.Response_window_timer.Tag = ""
            explorer.Response_window_timer.Enabled = False
            explorer.Response_window_timer.Enabled = True
            'reenvio y me quedo en este estado esperando respuesta
      '      MsgBox "El sucesor no me responde, hay ruido "
          Else
            N_Retransmisiones_Sucesor = 2
            Salir = True
            Estado = "idle"
      '      MsgBox "ya retrnsmito 2 veces y nada"
            'Notificar que no hubo respuesta del sucesor
          End If
        End If

      End If
    Loop

End Sub
Public Function Enviar_Datos(datos As String)
    'Función para enviar datos al anillo
    If Not explorer.MSComm1.PortOpen Then
      explorer.MSComm1.PortOpen = True
    End If
    explorer.MSComm1.Output = datos

End Function
Public Function Recibir_Datos() As String
    'Función para recibir datos del anillo
    If Not explorer.MSComm1.PortOpen Then
      explorer.MSComm1.PortOpen = True
    End If
    Recibir_Datos = explorer.MSComm1.Input
End Function
Public Sub Armar_Frame(Estacion As String, Tipo_Frame As Integer, Datos_INFO As String)
    'Este procedimiento construye el frame que se va a enviar a cada esclavo o maestro
    'Se arma un frama para pedirle o enviarle información de sus variables a
    'cada esclavo,en el parametro Datos_INFO, viene la inforamación a enviar
```

```
  'pueden ser valores o "p" que indican que estoy solicitando información
  Dim N_Objetos, i As Integer
  SD = Chr(Numero_Decimal(1, 1, 0, 1, 1, 0, 0, 0)) 'Inicio de frame
  FC = Chr(Tipo_Frame)
  'para frame debe ser 01, para token debe ser 00
  DA = Chr(Numero_Decimal(0, 0, 0, 0, 0, 0, 0, 0)) + Chr(Val(Estacion) + 40) 'Direccion destino
  'La primera parte de DA indica si es una dirección o son varias
  '0 indica una dirección, 1 indica varias, y se adicionan las direcciones de las estaciones
  SA = Chr(Val(Maestro) + 40) 'Dirección fuente
  N_Objetos = Mis_esclavos(eslavo_actual, 2)
  'El campo INFO, tiene un alongitud igual al número de variables que maneja el
  'objeto multiplicado por 2, ya que por cada variable se envia el identificador y el
  'valor, si el valor es "p" significa que se pide información, de lo controrio se
  'envia un valor para esa variable
  INFO = ""
  If (Tipo_Frame = 48) Or (Tipo_Frame = 49) Then
    'Es un frame para pasar o recibir el token
    INFO = "ppp"
  Else
    'Es un frame para esclavo
    For i = 1 To N_Objetos Step 1
      INFO = INFO + Chr(40 + i) 'indice del objeto
      INFO = INFO + Mid(Datos_INFO, i, 1) 'valor a enviar (puede ser p o otro valor)
    Next i
  End If
  FCS = Chr(Numero_Decimal(0, 1, 0, 0, 0, 0, 0, 1)) 'secuencia de con
  ED = Chr(Numero_Decimal(1, 1, 1, 1, 1, 1, 0, 0))
  Frame_para_enviar = SD + FC + DA + SA + INFO + FCS + ED
  Longitud_Frame_Esperado = Len(Frame_para_enviar) 'Longitud que debe tener el frame espeado
  'MsgBox "voy a enviar " + Frame
End Sub
Public Sub Interpretar_Frame_Recibido(frame1 As String)
  On Error GoTo ErrorTrans
  Dim Salir As Boolean
  Dim k, j As Integer
  Longitud_Frame_Recibido = Len(frame1)
  Salir = False
  Longitud_Info_Recibida = Longitud_Frame_Recibido - 7
  'MsgBox "La longitud del frame es :" + Str(Longitud_Frame_Recibido)
  'MsgBox "La longitud del info es :" + Str(Longitud_Info_Recibida)
  'MsgBox "El frame recibido es:" + (frame1)
  If Asc(Mid(frame1, 1, 1)) - 40 <= 40 Then
    SD = Asc(Mid(frame1, 1, 1)) - 40
  Else
    SD = Asc(Mid(frame1, 1, 1))
  End If

    FC = Asc(Mid(frame1, 2, 1))

  If Asc(Mid(frame1, 4, 1)) - 40 <= 40 Then
    DA = Asc(Mid(frame1, 4, 1)) - 40
  Else
    DA = Asc(Mid(frame1, 4, 1))
  End If
  If Asc(Mid(frame1, 5, 1)) - 40 <= 40 Then
    SA = Asc(Mid(frame1, 5, 1)) - 40
  Else
    SA = Asc(Mid(frame1, 5, 1))
  End If
```

```
  INFO = Mid(frame1, 6, Longitud_Info_Recibida)
  If Asc(Mid(frame1, Longitud_Info_Recibida + 6, 1)) - 40 <= 40 Then
    FCS = Asc(Mid(frame1, Longitud_Info_Recibida + 6, 1)) - 40
  Else
    FCS = Asc(Mid(frame1, Longitud_Info_Recibida + 6, 1))
  End If
  If Asc(Mid(frame1, Longitud_Info_Recibida + 7, 1)) - 40 <= 40 Then
    ED = Asc(Mid(frame1, Longitud_Info_Recibida + 7, 1)) - 40
  Else
    ED = Asc(Mid(frame1, Longitud_Info_Recibida + 7, 1))
  End If

Exit Sub

ErrorTrans:    'Error de transmisión borrar variables
  SD = 0
  FC = 0
  DA = 0
  SA = 0
  INFO = ""
  FCS = 0
  ED = 0
End Sub

Public Function Verificar_Frame_Recibido(Longitud_frame As Integer) As Boolean
  'Verificar si el frame recibido es el esperado
  Dim Escl_Actual As String
  If Esclavo_Actual = -1 Then
    Escl_Actual = 200 'para evitar errores caundo estoy en estado use token
  Else
    Escl_Actual = Mis_esclavos(Esclavo_Actual, 0)
  End If
  If DA = Maestro Then
    'El frame si viene para este maestro
    If SA = Escl_Actual And Estado = "await_IFM_response" Then
      'recibi repsueesta de un esclavo
      In_ring = True
      If Longitud_frame = Longitud_Frame_Esperado Then
        'MsgBox "recibi bien, esclavo actual " + Str(Escl_Actual)
        'El frame es el esperado y esta correcto
        Noise_burst = False
      Else
        'El frame viene mal, se debe volver a pedir
        Noise_burst = True
        'MsgBox "no tiene la longitud esperada, la esperada es: " + Str(Longitud_Frame_Esperado) + " y llego " +
Str(Longitud_frame)
      End If
    ElseIf (SA = NS) And Estado = "check_token_pass" Then
      'recibi respuesta de maestro
      If Longitud_frame = Longitud_Frame_Esperado Then
        'El frame es el esperado y esta correcto
        Noise_burst = False
      Else
        'El frame viene mal, se debe volver a pedir
        Noise_burst = True
        'MsgBox "no tiene la longitud esperada, la esperada es: " + Str(Longitud_Frame_Esperado) + " y llego " +
Str(Longitud_frame)
      End If
    Else
```

```
        'Otra estación le envio un frame a este maestro
        'verificar que tipo de frame es y mirar si otra estación esta
        'asumiendo que tiene el token
        Noise_burst = True
      End If
    Else
      Noise_burst = True
      'MsgBox "ESTE FRAME NO VIENE PARA MI"
      'El frame no viene para este maestro....
      'verificar que tipo de frame es y mirar si otra estación esta
      'asumiendo que tiene el token
    End If
End Function
Public Sub Actualizar_Esclavo_Virtual(Esclavo1 As String)
    On Error Resume Next
    'actualizo la inforamción en el listview1
    Dim i, j, Informacion, Indice, Anterior, Dato, Variables, Acabe
    For i = 0 To N_esclavos - 1 Step 1
      If Mis_esclavos(i, 0) = Esclavo1 Then
        Indice = i
        i = N_esclavos - 1
      End If
    Next i
    Informacion = Mis_esclavos(Indice, 3)
    If Len(Informacion) > 0 Then
      explorer.ListView1.ListItems.Clear
      explorer.ListView1.ColumnHeaders.Clear
      ' Crea una variable de objeto para el objeto ColumnHeader
      explorer.Label8.Caption = "Objetos del dispositivo virtual: (" + "Esclavo " + Esclavo1 + ")"
      Dim clmX As ColumnHeader
      ' Agrega ColumnHeaders. El ancho de las columnas es el ancho
      ' del control dividido por el número de objetos ColumnHeader.
      Set clmX = explorer.ListView1.ColumnHeaders.Add(, , "Objeto", explorer.ListView1.Width / 2)
      Set clmX = explorer.ListView1.ColumnHeaders.Add(, , "Valor", explorer.ListView1.Width / 2)
      explorer.ListView1.BorderStyle = ccFixedSingle ' Establece la propiedad BorderStyle.
      explorer.ListView1.View = lvwReport ' Establece la propiedad View como Report.
      Anterior = 0
      Variables = Mis_esclavos(Indice, 4)
      j = 1
      For i = 2 To Mis_esclavos(Indice, 2) * 2 Step 2 'lo ejecuata el numero de variables que tiene
        Acabe = False
        Dato = ""
        Do While Not Acabe
          'capturo el nombre de la variable, estan separadas por (;).
          If Mid(Variables, j, 1) <> ";" Then
            Dato = Dato + Mid(Variables, j, 1)
          Else
            Acabe = True

          End If
          j = j + 1
        Loop
        Set itmX = explorer.ListView1.ListItems.Add(, , Dato)
        'If Asc(Mid(Informacion, i, 1)) <= 79 Then
        '   itmX.SubItems(1) = Asc(Mid(Informacion, i, 1)) - 40
        'Else
          itmX.SubItems(1) = Asc(Mid(Informacion, i, 1))
        'End If
```

```
    Next i
  Else
    explorer.Label8.Caption = "Objetos del dispositivo virtual: (" + "Esclavo " + Esclavo1 + ")"
    explorer.ListView1.ListItems.Clear
    explorer.ListView1.ColumnHeaders.Clear
  End If
End Sub
Public Function Numero_Decimal(ParamArray num())
  'procedimiento para convertir un número binario a decimal
  'ParamArray se usa para recibir un array variable
  j = 7
  For Each x In num
    Numero_Decimal = Numero_Decimal + (x * (2 ^ j))
    j = j - 1
  Next x
  If Numero_Decimal <= 40 Then 'le sumo cuarenta para no tener incompatibilidad en
  'la transmisión, ya que los caracteres menores de 40, son caracteres de control
     Numero_Decimal = Numero_Decimal + 40
  End If

End Function

Public Function Numero_Binario(numerod As Integer) As String
  'procedimiento para convertir de decimal a binario
  num = numerod
  Dim numeros(7) As String
  For j = 0 To 7
    numeros(j) = Int(num) Mod 2
    num = num / 2
    DoEvents
  Next j
  For j = 7 To 0 Step -1
    Numero_Binario = Numero_Binario + Trim(numeros(j))
    DoEvents
  Next j

End Function
Public Function numero_hex(numeroh As String) As Integer
  'Funcion para convertir un numero hexadecimal a entero
  Dim j As Integer
  Dim num As Integer
  Dim letra As String
  j = Len(Trim(numeroh))
  k = 1
  While j > 0
    letra = Mid(Trim(numeroh), k, 1)
    Select Case letra
      Case "A", "a"
        num = 10
      Case "B", "c"
        num = 11
      Case "C", "c"
        num = 12
      Case "D", "d"
        num = 13
      Case "E", "d"
        num = 14
      Case "F", "f"
        num = 15
```

```
        Case Else
          num = Val(letra)

      End Select
      numero_hex = numero_hex + num * 16 ^ (j - 1)
      j = j - 1
      k = k + 1
      DoEvents
    Wend
End Function

Sub Error_De_Comunicacion(Estacion As String)
    Dim itmX As ListItem
    Estacion = "Esclavo " + Trim(Estacion)
    'Establece la variable al elemento encontrado.
    Set itmX = explorer.lvListView.FindItem(Trim(Estacion), , , lvwPartial)
    explorer.SSTab1.Tab = 1
    explorer.Image4.Picture = explorer.Image3.Picture
    explorer.Label3.ForeColor = &HFF&
    explorer.Label5.ForeColor = &H80000012
    explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , , lvwWholeWord).Index).Icon =
explorer.ImageList2.ListImages.Item(3).Index
    explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SmallIcon = explorer.ImageList1.ListImages.Item(3).Index
    explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SubItems(3) = "Alarma"
    explorer.tvTreeView.Nodes(Estacion).Image = explorer.ImageList1.ListImages.Item(3).Index
    explorer.lvListView.Refresh
    explorer.Label7.Caption = Estacion
    explorer.Label2.Caption = "Mensaje de alarma: La estación " + Estacion + " no responde." + _
    "Posibles causas: Estación se daño, cable de comunicación esta roto, puerto de comunicación" + _
    "esta dañado (esclavo o maestro)"
End Sub
Sub Comunicacion_Correcta(Estacion As String)
    Dim itmX As ListItem
    Estacion = "Esclavo " + Trim(Estacion)
    ' Establece la variable al elemento encontrado.
    Set itmX = explorer.lvListView.FindItem(Trim(Estacion), , , lvwPartial)
    explorer.SSTab1.Tab = 1
    explorer.Image4.Picture = explorer.Image1.Picture
    explorer.Label3.ForeColor = &H80000012
    explorer.Label5.ForeColor = &HC000&
    explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , , lvwWholeWord).Index).Icon =
explorer.ImageList2.ListImages.Item(2).Index
    explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SmallIcon = explorer.ImageList1.ListImages.Item(4).Index
    explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SubItems(3) = "Funcionamiento correcto"
    explorer.tvTreeView.Nodes(Estacion).Image = explorer.ImageList1.ListImages.Item(4).Index
    explorer.lvListView.Refresh
    explorer.Label7.Caption = Estacion
    explorer.Label2.Caption = "Mensaje de alarma: (Todo OK)"
End Sub
```

Formulario Explorer

```
Dim mbMoving As Boolean
Const sglSplitLimit = 2200


Private Sub Bus_idle_timer_Timer()
   explorer.Bus_idle_timer.Tag = "termine"
End Sub


Private Sub Combo1_Click()
   'Lleno el combo2 con los objetos del esclavo
   Dim Esclavo, i, Indice, Acabe, Dato, Variables
   For i = 0 To N_esclavos - 1 Step 1
      If Mis_esclavos(i, 1) = Combo1.Text Then
         Indice = i
         i = N_esclavos - 1
      End If
   Next i
   Combo2.Clear
   j = 1
   Variables = Mis_esclavos(Indice, 4)
   For i = 1 To Mis_esclavos(Indice, 2) Step 1
      Acabe = False
      Dato = ""
      Do While Not Acabe
         'capturo el nombre de la variable, estan separadas por (;).
         If Mid(Variables, j, 1) <> ";" Then
            Dato = Dato + Mid(Variables, j, 1)
         Else
            Acabe = True
         End If
         j = j + 1
      Loop
      Combo2.AddItem Dato

   Next i
End Sub

Private Sub Command1_Click()
Estado = "use_token"
End Sub

Private Sub Command3_Click()

   Dim i, Indice, Valor, Pos
   If IsNumeric(Text2.Text) Or (Text2.Text = "p") Then
      Indice = -1
      For i = 0 To N_esclavos - 1 Step 1
         If Mis_esclavos(i, 1) = Combo1.Text Then
            Indice = i
            i = N_esclavos - 1
         End If
      Next i
      If Indice <> -1 Then
         If Combo2.Text <> "" Then
```

```
            Pos = Combo2.ListIndex + 1
            Valor = ""
            For i = 1 To Mis_esclavos(Indice, 2) Step 1
              If i = Pos Then
                If Text2.Text = "p" Then
                  Valor = Valor + "p"
                Else
                  Valor = Valor + Chr(Val(Text2.Text) + 40)
                End If
              Else
                Valor = Valor + Mid(Mis_esclavos(Indice, 5), i, 1)
              End If
            Next i
            Mis_esclavos(Indice, 5) = Valor
            MsgBox "El frame fue actualizado y esta listo para transmitirse"
            Combo2.Text = ""
            Text2.Text = ""
          Else
            MsgBox "Debe seleccionar un objeto", vbCritical, "Error"
          End If
        Else
          MsgBox "Debe seleccionar un esclavo", vbCritical, "Error"
        End If
      Else
        MsgBox "El valor a asignar, no es un número correcto", vbCritical, "Error"
      End If
End Sub

Private Sub Form_Load()
    Inicio 'variables de inicio
    Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
    Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
    Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
    Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)
    Dim myDb As Database, myRs, rs2, rs3, rsObj As Recordset
    Set myDb = DBEngine.Workspaces(0).OpenDatabase(Ruta & "espia.mdb")
    Set myRs = myDb.OpenRecordset("SELECT * FROM esclavos where maestro = '" & Trim(Maestro) & "';",
dbOpenDynaset)
    Set rs2 = myDb.OpenRecordset("SELECT * FROM maestros where indice <> '" & Trim(Maestro) & "';",
dbOpenDynaset)
    Set rs3 = myDb.OpenRecordset("maestros", dbOpenDynaset)
    'Inicializa el control TreeView y crea varios nodos.
    tvTreeView.ImageList = ImageList1
    Dim nodX As Node    ' Crea un árbol.
    While Not rs3.EOF
      Set nodX = tvTreeView.Nodes.Add(, , "Maestro (DPM" & Trim(rs3!Indice) & ")", "Maestro (DPM" &
Trim(rs3!Indice) & ")", 1)
      rs3.MoveNext
    Wend
    While Not myRs.EOF
      Set nodX = tvTreeView.Nodes.Add(1, tvwChild, "Esclavo " & Trim(myRs!Indice), "Esclavo " &
Trim(myRs!Indice), 4)
      Set rsObj = myDb.OpenRecordset("select * from DiccionarioObjetosEsclavos where Esclavo = '" &
myRs.Fields("Indice").Value & "'", dbOpenDynaset)
      Do While Not rsObj.EOF
        'adiciono los objetos del esclavo al árbol
        Set nodX = tvTreeView.Nodes.Add("Esclavo " & Trim(myRs!Indice), tvwChild, Trim(rsObj!NombreObjeto)
+ Trim(myRs!Indice), Trim(rsObj!NombreObjeto), 6)
        rsObj.MoveNext
```

```
    Loop
    Me.Combo1.AddItem "Esclavo " & Trim(myRs!Indice)
    myRs.MoveNext
  Wend

  nodX.EnsureVisible ' Expande el árbol para mostrar todos los nodos.

'LISTVIEW
  ' Crea una variable de objeto para el objeto ColumnHeader.
  Dim clmX As ColumnHeader
  ' Agrega ColumnHeaders. El ancho de las columnas es el ancho
  ' del control dividido por el número de objetos ColumnHeader.
  Set clmX = lvListView.ColumnHeaders. _
  Add(, , "Indice", lvListView.Width / 4)
  Set clmX = lvListView.ColumnHeaders. _
  Add(, , "Nombre", lvListView.Width / 4)
  Set clmX = lvListView.ColumnHeaders. _
  Add(, , "Tipo de proceso", lvListView.Width / 4)
  Set clmX = lvListView.ColumnHeaders. _
  Add(, , "Estado Actual", lvListView.Width / 4)


  lvListView.BorderStyle = ccFixedSingle ' Establece la propiedad BorderStyle.
  lvListView.View = lvwIcon ' Establece la propiedad View como Report.

  ' Agrega una imagen a ImageList1--Icons ImageList.
  Dim imgX As ListImage
  ' Para usar controles ImageList con el control ListView, deberá
  ' asociar un determinado control ImageList a las propiedades Icons
  ' y SmallIcons.
  lvListView.Icons = ImageList2
  lvListView.SmallIcons = ImageList1

  Dim d As Recordset
  On Error Resume Next
  rs2.MoveFirst
  myRs.MoveFirst
  ' Crea una variable para agregar objetos ListItem.
  Dim itmX As ListItem
  While Not myRs.EOF

    Set itmX = lvListView.ListItems.Add(, CStr("Esclavo " + Trim(myRs!Indice)), CStr("Esclavo " +
Trim(myRs!Indice)))
    itmX.Icon = 2   ' Establece un icono de ImageList1.
    itmX.SmallIcon = 4  ' Establece un icono de ImageList2.

    If Not IsNull(myRs!nombre) Then
      itmX.SubItems(1) = CStr(myRs!nombre)
    End If
    If Not IsNull(myRs!TipoDeProceso) Then
      itmX.SubItems(2) = myRs!TipoDeProceso
    End If
      itmX.SubItems(3) = "Funcionamiento correcto"
     myRs.MoveNext  ' Pasa al registro siguiente.
  Wend
  'Para los maestros
  While Not rs2.EOF

    Set itmX = lvListView.ListItems.Add(, , CStr("Maestro (DPM" + rs2!Indice + ")"))
```

```
      itmX.Icon = 1   'Establece un icono de ImageList1.
      itmX.SmallIcon = 1  'Establece un icono de ImageList1.

      If Not IsNull(rs2!nombre) Then
         itmX.SubItems(1) = CStr(rs2!nombre)
      End If
      If Not IsNull(rs2!caracteristicas) Then
         itmX.SubItems(2) = rs2!caracteristicas
      End If
         itmX.SubItems(3) = "Funcionamiento correcto"
      rs2.MoveNext  'Pasa al registro siguiente.
   Wend


End Sub


Private Sub Form_Paint()
   lvListView.View = Val(GetSetting(App.Title, "Settings", "ViewMode", "0"))
End Sub


Private Sub Form_Unload(Cancel As Integer)

   Dim Msg   'Declara la variable.
   'Establece el texto del mensaje.
   Msg = "¿Realmente desea salir de la aplicación?"
   ' Si el usuario hace clic en el botón No, se detiene QueryUnload.
   If MsgBox(Msg, vbQuestion + vbYesNo, Me.Caption) = vbNo Then Cancel = True

   If Not Cancel Then
      Dim i As Integer
      'cerrar todos los subformularios
      For i = Forms.Count - 1 To 1 Step -1
         Unload Forms(i)
      Next
      If Me.WindowState <> vbMinimized Then
         SaveSetting App.Title, "Settings", "MainLeft", Me.Left
         SaveSetting App.Title, "Settings", "MainTop", Me.Top
         SaveSetting App.Title, "Settings", "MainWidth", Me.Width
         SaveSetting App.Title, "Settings", "MainHeight", Me.Height
      End If
      SaveSetting App.Title, "Settings", "ViewMode", lvListView.View
   End If
   'Cerrar los objetos de la base de datos
   rstimers.Close
   rsesclavos.Close
   RsMaestros.Close
   RsMisEsclavos.Close

End Sub



Private Sub Form_Resize()

   On Error Resume Next
   If Me.Width < 3000 Then Me.Width = 3000
   SizeControls imgSplitter.Left
```

```
End Sub


Private Sub imgSplitter_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As Single)
   With imgSplitter
      picSplitter.Move .Left, .Top, .Width \ 2, .Height
   End With
   picSplitter.Visible = True
   mbMoving = True
End Sub


Private Sub imgSplitter_MouseMove(Button As Integer, Shift As Integer, x As Single, Y As Single)
   Dim sglPos As Single

   If mbMoving Then
      sglPos = x + imgSplitter.Left
      If sglPos < sglSplitLimit Then
         picSplitter.Left = sglSplitLimit
      ElseIf sglPos > Me.Width - sglSplitLimit Then
         picSplitter.Left = Me.Width - sglSplitLimit
      Else
         picSplitter.Left = sglPos
      End If
   End If
End Sub


Private Sub imgSplitter_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
   SizeControls picSplitter.Left
   picSplitter.Visible = False
   mbMoving = False
End Sub


Private Sub TabStrip1_Click()

End Sub

Private Sub lvListView_ItemClick(ByVal Item As MSComctlLib.ListItem)
If Mid(Item, 1, 7) = "Esclavo" Then
   'es un esclavo
   If Item.SubItems(3) = "Alarma" Then
      explorer.SSTab1.Tab = 1
      Me.Label1.Caption = Item
      Me.Label7.Caption = Item
      explorer.Image4.Picture = explorer.Image3.Picture
      explorer.Label3.ForeColor = &HFF&
      explorer.Label5.ForeColor = &H80000012
      Item.Icon = explorer.ImageList2.ListImages.Item(3).Index
      Item.SmallIcon = explorer.ImageList1.ListImages.Item(3).Index
      'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , , lvwWholeWord).Index).Icon
= explorer.ImageList2.ListImages.Item(3).Index
      'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SubItems(3) = "Alarma"
      explorer.Label2.Caption = "Mensaje de alarma: La estación " + Item + " no responde." + _
      "Posibles causas: Estación se daño, cable de comunicación esta roto, puerto de comunicación" + _
      "esta dañado (esclavo o maestro)"
   Else
```

```
        explorer.SSTab1.Tab = 1
        Me.Label1.Caption = Item
        Me.Label7.Caption = Item
        explorer.Image4.Picture = explorer.Image1.Picture
        explorer.Label3.ForeColor = &H80000012
        explorer.Label5.ForeColor = &HC000&
        Item.Icon = explorer.ImageList2.ListImages.Item(2).Index
        Item.SmallIcon = explorer.ImageList1.ListImages.Item(4).Index
        'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , , lvwWholeWord).Index).Icon
= explorer.ImageList2.ListImages.Item(3).Index
        'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SubItems(3) = "Alarma"
        explorer.Label2.Caption = "Mensaje de alarma: (Todo OK)"
    End If
    Actualizar_Esclavo_Virtual (Mid(Item, 9, 1))
Else
    'es un maestro
    If Item.SubItems(3) = "Alarma" Then
        explorer.SSTab1.Tab = 1
        Me.Label1.Caption = Item
        Me.Label7.Caption = Item
        explorer.Image4.Picture = explorer.Image3.Picture
        explorer.Label3.ForeColor = &HFF&
        explorer.Label5.ForeColor = &H80000012
        Item.Icon = explorer.ImageList2.ListImages.Item(3).Index
        Item.SmallIcon = explorer.ImageList1.ListImages.Item(3).Index
        'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , , lvwWholeWord).Index).Icon
= explorer.ImageList2.ListImages.Item(3).Index
        'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SubItems(3) = "Alarma"
        explorer.Label2.Caption = "Mensaje de alarma: La estación " + Item + " no responde." + _
        "Posibles causas: Estación se daño, cable de comunicación esta roto, puerto de comunicación" + _
        "esta dañado (esclavo o maestro)"
    Else
        explorer.SSTab1.Tab = 1
        Me.Label1.Caption = Item
        Me.Label7.Caption = Item
        explorer.Image4.Picture = explorer.Image1.Picture
        explorer.Label3.ForeColor = &H80000012
        explorer.Label5.ForeColor = &HC000&
        'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , , lvwWholeWord).Index).Icon
= explorer.ImageList2.ListImages.Item(3).Index
        'explorer.lvListView.ListItems(explorer.lvListView.FindItem(Trim(Estacion), , ,
lvwWholeWord).Index).SubItems(3) = "Alarma"
        explorer.Label2.Caption = "Mensaje de alarma: (Todo OK)"
    End If
    explorer.Label8.Caption = "Objetos del dispositivo virtual: " + Item
    explorer.ListView1.ListItems.Clear
End If

End Sub



Private Sub Recibir_Timer()
'timer para estar recibiendo datos
    Frame_Nuevo = Recibir_Datos()
    Frame_Recibido = Frame_Nuevo
    If Len(Frame_Recibido) > 0 Then
```

```
      Text1.Text = Frame_Recibido
      Rx_protocolo_frame = True
      Bus_quiet = False 'Indica que se recibio frame
      Interpretar_Frame_Recibido (Frame_Recibido) 'funcion para desglosar el mensaje
      Verificar_Frame_Recibido (Len(Frame_Recibido))  'para saber si el frame es para este maestro
   End If

End Sub

Private Sub Response_window_timer_Timer()
   Me.Response_window_timer.Tag = "termine"
End Sub

Private Sub Toke_pass_timer_Timer()
   explorer.Toke_pass_timer.Tag = "termine"
End Sub

Private Sub Token_hold_timer_Timer()
Me.Token_hold_timer.Tag = "termine"
End Sub


Private Sub tvTreeView_DragDrop(Source As Control, x As Single, Y As Single)
   If Source = imgSplitter Then
      SizeControls x
   End If
End Sub


Sub SizeControls(x As Single)
   On Error Resume Next


   'Ajustar anchura
   If x < 1500 Then x = 1500
   If x > (Me.Width - 1500) Then x = Me.Width - 1500
   tvTreeView.Width = x
   Label1.Width = x
   Label2.Width = x - 120
   imgSplitter.Left = x
   SSTab1.Width = x
   lvListView.Left = x + 40
   lvListView.Width = Me.Width - (tvTreeView.Width + 140)
   lblTitle(0).Width = tvTreeView.Width
   lblTitle(1).Left = lvListView.Left + 20
   lblTitle(1).Width = lvListView.Width - 40

   ListView1.Left = x + 40
   ListView1.Width = lvListView.Width
   Label8.Left = x + 40
   Label8.Width = lvListView.Width

   'set the top
 tvTreeView.Top = picTitles.Height

 SSTab1.Top = picTitles.Height


   'set the height
```

```
   If sbStatusBar.Visible Then
      tvTreeView.Height = ((Me.ScaleHeight - (picTitles.Top + picTitles.Height + sbStatusBar.Height))) / 1.2
      Label1.Top = tvTreeView.Height + 300
      SSTab1.Height = Me.ScaleHeight - (picTitles.Top + picTitles.Height + sbStatusBar.Height)


   Else
      tvTreeView.Height = ((Me.ScaleHeight - (picTitles.Top + picTitles.Height))) / 1.2
      Label1.Top = tvTreeView.Height + 300
      SSTab1.Height = Me.ScaleHeight - (picTitles.Top + picTitles.Height)


   End If


   lvListView.Height = tvTreeView.Height - 1000
   Label8.Top = tvTreeView.Height - 420
   ListView1.Top = tvTreeView.Height - 120
   ListView1.Height = Me.ScaleHeight - (picTitles.Top + picTitles.Height) - lvListView.Height - 600

   imgSplitter.Top = tvTreeView.Top
   imgSplitter.Height = SSTab1.Height
End Sub



Private Sub SSTab1_DragDrop(Source As Control, x As Single, Y As Single)

 If Source = imgSplitter Then
      SizeControls x
   End If
End Sub

Private Sub tvTreeView_NodeClick(ByVal Node As MSComctlLib.Node)
Me.Label1.Caption = Node
Me.Label7.Caption = Node
If Mid(Node, 1, 7) = "Esclavo" Then
   Actualizar_Esclavo_Virtual (Mid(Node, 9, 1))
ElseIf Mid(Node, 1, 7) = "Maestro" Then
   explorer.Label8.Caption = "Objetos del dispositivo virtual: " + Node
   explorer.ListView1.ListItems.Clear
End If

End Sub
```

ANEXO 3 CODIGO DEL SISTEMA ESCLAVO

```
;Sistema para los esclavos, el sistema recibe un frame del PC (maestro),
;el frame tiene la estructura: SD + FC + DA + SA + INFO + FCS + ED en donde
;SD=Inicio de frame, FC=Contro del frame, DA=dirección destino, SA=Dirección fuente
;INFO= información (mas de un byte), ED = fin del frame.
;el sistema debe detectar que tipo de frame es (FC), Si el frame va dirigido a
;este esclavo (DA), y validar la inforamción y dar respuesta con el mismo formato
;En la paginas 113 a 116 se especifica que significa cada campo. En la pagina 131 de pruebas
;también se puede ver el funcionamiento

Indice  equ 43          ;identificador del esclavo
NumVar  equ 2           ;número de variables u objetos que maneja el esclavo
TRANSM  equ P1.7        ;activa transmisión y recebción del max485

        org 00H
        ljmp Inicio

        org 23H         ;Interrupción puerto serial
        jb ri,recibe
        clr ti
        setb TRANSM     ;Activa transmisión del esclavo
        cjne @r1,#0h,siga
```

```
        mov r1,#20h
        clr TRANSM    ;VUELVE A ESCUCHAR SOLAMENTE
        reti
siga:   mov a,@r1
        mov sbuf,a
        inc r1
        reti


recibe:
        clr ri        ;reseteo recepción
        mov a,sbuf
        cjne a,#252,final  ;caracter final del frame (ED)
        mov @r1,a
        inc r1
        mov @r1,#0h       ;adiciono un caracter nulo para saber cuando dejar de transmitir

        mov r1,#20h       ; cargo en a el valor de SD (Delimitador de inicio)
        cjne @r1,#216,sal  ; Si SD no es el delimitador de inicio correcto, salgo

        mov r1,#21h       ; cargo en a el valor de FC (frame de control)
        cjne @r1,#79,sal   ; Verifico el valor de FC, si es 79, es un frame con respuesta

        mov r1,#23h       ; cargo en r1 el valor de DA (Dirección destino)
        cjne @r1,#Indice,sal  ;Comparo el DA con el indice del esclavo, Si es igual puedo responder

        ljmp verifi       ;procedimiento para verificar el frame
        reti

final:  mov @r1,a         ;Si no es el delimitador final, almacena el valor enviado
        inc r1
        reti
```

```
sal:   mov r1,#20h     ;reinicia el valor de r1 y sale
       reti

Inicio
       mov 7fh,#'I'    ;dirección donde estan almacenadas la variable 1
       mov 7eh,#'J'    ;dirección donde estan almacenadas la variable 2
       mov 7dh,#'L'    ;dirección donde estan almacenadas la variable 3
       mov TH1,#$FD    ;transmisión a  9600 b/s
       mov TL1,#$FD
       mov TMOD,#$20   ;Timer1 modo 2
       mov TCON,#$00
       setb TR1        ;Activo timer 1
       mov SCON,#$50   ;01010000, modo 1 de transmisión
       mov IE,#$90     ;10010000, habilita interripción serial
       clr TRANSM      ;EL MICRO ESCUCHA SOLAMENTE
       clr ri
       mov r1,#20h

PROGRAM NOP            ;ciclo repetitivo de espera
       NOP
       NOP
       NOP
       ljmp PROGRAM

verifi: ;verificación y frame de respuesta
       ;verifico si el frame es para pedir datos o para actualizar.
       ;si es para pedir datos, los campos donde van los datos lleva
       ;el valor "p", si no llevan el valor que se actualiza
       mov r1,#26h      ;dirección en la que se almacena el valor de la variable 1
       cjne @r1,#'p',ver1
       sjmp ver11

ver1:  mov 7fh,@r1
```

```
ver11:  mov r1,#28h      ;dirección en la que se almacena el valor de la variable 2
        cjne @r1,#'p',ver2
        sjmp ver22

ver2:   mov 7eh,@r1
ver22:  mov r1,#2Ah      ;dirección en la que se almacena el valor de la variable 1
        cjne @r1,#'p',ver3
        sjmp frame

ver3:   mov 7dh,@r1

        ;Procedimiento que armar el frame a enviar
frame:  mov 21h,#71      ;Cambio el FC, a frame sin respuesta
        mov a,23h
        mov 23h,24h      ;Cambio dirección destino
        mov 24h,a        ;Cambio dirección fuente
        mov r1,#26h      ;dirección donde se encuentra la primera variable del objeto que llego
        mov @r1,7fh
        mov r1,#28h      ;dirección donde se encuentra la segunda variable del objeto que llego
        mov @r1,7eh
        mov r1,#2Ah      ;dirección donde se encuentra la tercera variable del objeto que llego
        mov @r1,7dh

        setb TRANSM      ;Activa transmisión del esclavo
        mov r1,#20h      ;empiezo a responder el frame con el caracter de inic SD
        mov a,@r1
        inc r1
        mov sbuf,a
        reti

        END
```