ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

SOFTWARE DE COMUNICACIÓN PARA SIMULACIÓN DE MEDICIÓN Y CONTROL A TRAVÉS DE LA RED (IP)

PROYECTO DE GRADUACIÓN

PREVIO A LA OBTENCIÓN DEL TITULO DE

INGENIERO EN ELECTRICIDAD

ESPECIALIZACIÓN ELECTRÓNICA

PRESENTADO POR:

MANUEL UVIDIA HERNÁNDEZ

FRANCISCO PACHECO CASTRO

LEONARDO BARONA VALENCIA

AGRADECIMIENTO

A todas las personas que de uno u otro modo proporcionaron información para la realización de este trabajo y especialmente a todos nuestros amigos por su invaluable ayuda.

DEDICATORIA

Dedico este trabajo a mis padres que en todo momento me brindaron su apoyo y comprensión.

Leonardo

A Dios, a mis padres, a mi familia, a mis amigos y especialmente a mi abuela que en todo momento me brindaron su comprensión y apoyo.

Francisco

Dedico este proyecto a Dios, a mis padres y abuela, que nunca me dejaron desfallecer y en todo momento me brindaron su comprensión y apoyo.

Manuel

TRIBUNAL DE GRADUACIÓN

……	……..
Ing. Carlos Monsalve	Ing. Boris Ramos.
SUBDECANO DE LA FIEC	**DIRECTOR DE TÓPICO**
……...……	…….
Ing. Erick Ricaurte	Ing. Alberto Manzur
VOCAL	**VOCAL**

INTRODUCCIÓN

La Internet está cambiando la manera en que usted trabaja. Como tecnología, usted está adoptando el uso de la Internet más rápidamente que ninguna otra tecnología en la historia, incluyendo la PC misma.

La presencia universal de la Internet y las redes de PC, nos guía a una nueva era de conectividad de información y compartimiento de datos que proporciona innovadoras formas de adquirir, analizar y presentar mediciones.

Las redes son intrínsecamente flexibles y, por lo tanto, las posibles permutaciones para crear una solución para medición a través de la red son aparentemente infinitas.

La revolución en medición transforma la industria de prueba y medición. La PC, junto con las innovaciones tecnológicas que han impulsado a la industria de la computación, ha revolucionado los mercados y procesos que la mayoría de las compañías utilizan para llevar a cabo sus transacciones.

Mientras que la tecnología de medición ha mejorado durante el mismo período de tiempo, la PC en si misma ha tenido el mayor impacto en la forma en que tomamos mediciones hoy en día, más que ninguna otra tecnología.

La mayoría de los avances han ocurrido en la lógica de alta densidad y alta velocidad, impulsados por el despliegue de audio, vídeo y memoria de las PC. Durante los últimos 20 años, ha proporcionado la base para generar una revolución en la forma de tomar mediciones, transformando los sistemas de prueba y medidas y los sistemas de automatización industrial, de instrumentos autónomos y frecuentemente incompatibles, en sistemas de automatización y medición de alto rendimiento. En efecto, la industria de la computación ha auxiliado en la creación del sistema de medición digital.

Mucho hemos oído hablar sobre la "instrumentación virtual" y sus beneficios. El concepto de instrumentación virtual nace a partir del uso del computador personal (PC) como "instrumento" de medición de tales señales como:

- **temperatura**
- **presión**
- **caudal**
- **etc.**

Es decir, el PC comienza a ser utilizado para realizar mediciones de fenómenos físicos representados en señales de corriente (Ej. 4-20mA) y/o voltaje (Ej. 0-5Vdc).

Sin embargo, el concepto de "instrumentación virtual" va más allá de la simple medición de corriente o voltaje, sino que también involucra el procesamiento, análisis, almacenamiento, distribución y despliegue de los datos e información relacionados con la medición de una o varias señales específicas. Es decir, el instrumento virtual no se conforma con la adquisición de la señal, sino que también involucra la interfaz hombre-máquina, las funciones de análisis y procesamiento de señales, las rutinas de almacenamiento de datos y la comunicación con otros equipos.

El instrumento virtual es definido entonces como una capa de software y hardware que se le agrega a un PC en tal forma que permite a los usuarios interactuar con la computadora como si estuviesen utilizando su propio instrumento electrónico "hecho a la medida".

Un instrumento virtual puede realizar las tres (3) funciones básicas de un instrumento convencional: adquisición, análisis y presentación de datos. Sin embargo, el instrumento virtual me permite personalizar el instrumento, y agregarle mucha más funcionalidad sin incurrir en costos adicionales. ¿Quiere conectividad de su instrumento con Ethernet?. ¿Quiere almacenar sus datos en una tabla o archivo compatible con MS Excel?. La respuesta a todas estas preguntas es SI PUEDE HACERLO; y mejor aún, lo puede hacer usted mismo.

El instrumento virtual se apalanca en la flexibilidad y poder del PC, y mediante el software que lo acompaña, el nivel de adaptabilidad y personalización del instrumento virtual es casi ilimitado. ¿Porqué limitarse entonces?

RESUMEN

Debido a los continuos cambios en el área tecnológica a la cual nos vemos abocados día a día, ha sido de nuestro interés presentar un proyecto que conjugue lo aprendido e induzca al bienestar de las personas.

Nuestro objetivo primordial es diseñar un sistema que no modifique de manera trascendente lo que existe, sino que por lo demás se adapte y pueda crecer de manera paralela con la industria y en el hogar.

Siempre es prioritario que la operación del sistema sea lo más sencillo posible a la vez que ofrezca seguridad, pero aquí se encuentra la interrogante más fuerte de nuestra existencia. ¿Cuán seguro es un sistema?.

Nuestro mundo esta lleno de señales físicas de todo tipo (eléctricas, mecánicas, etc.); por ello dedicamos un capitulo a las señales de entrada, adquisición de datos y su tratamiento tanto a señales analógicas como digitales. También se podrá encontrar ejemplos muy ilustrativos.

Dentro de los requerimientos de Hardware y Software nuestra idea se fundamenta en utilizar redes LAN y conexión vía Internet con un servidor. Nuestra intención no es crear nuestro propio protocolo de comunicación, sino más bien, utilizar TCP/IP a fin de hacerlo práctico y funcional.

La forma en que las señales analógicas y digitales son convertidas y tratadas se presenta en el *Capítulo Fundamentos de Medición y Control.* La adquisición de datos hacia el computador a través del puerto paralelo se realiza con Visual Basic, en el *Capítulo 3* se encontrará la librería utilizada para el manejo del puerto paralelo; y un diseño de bloques de datos para el monitoreo y control de los dispositivos.

Para la transmisión de datos a través de Internet *(Capitulo 4)* se utiliza el lenguaje ASP (*Active Server Pages*), a fin de poder enviar y recibir los datos de clientes remotos sin demasiados requerimientos de Software, a la vez que nos permite realizar una interface que sea más amigable para el usuario.

En el capitulo final se presenta el análisis de costos para el montaje del proyecto (desarrollo de software) el cual creemos que es muy importante ya que de esta manera tenemos un punto de partida que nos ayudaría a realizar proyectos de similares condiciones. Además, presentamos los beneficios que se obtendrían de completar un sistema como el expuesto para tener una idea del alcance tecnológico que podríamos obtener al desarrollar dicho proyecto.

Nuestra idea no es hacer un mundo en el cual el trabajo sea complicado, sino hacerlo un poco más fácil con el fin de ocupar nuestro tiempo y conocimiento para que nuestras vidas sean cada vez mejor.

- Realizar el software de comunicación que permita simular la adquisición, manipulación de datos y operación de los dispositivos de una industria, utilizando una aplicación desarrollada con Visual Basic los cuales deben ser transmitidos a través de la red (IP) usando ASP (Active Server Pages) con el fin de visualizar los datos en clientes remotos.

- Edificar un formato de bloques de datos que sirva para enviar y recibir información de control y Monitoreo de los dispositivos de la industria al servidor.

- Evaluación de costo–beneficio, para su posible realización.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

I. FUNDAMENTOS DE MEDICIÓN Y CONTROL

1.1 Introducción al control industrial.

Dentro de los requerimientos de la industria actual en el campo de medición y control está la verificación de voltaje, corriente, número de revoluciones por minuto (RPM), prendido de motores de alta y baja tensión, entre otros. En los sistemas de monitoreo se usan sensores de temperatura y presión, junto con convertidor analógico / digital.

Entre las variables mas usadas dentro del Monitoreo, medición y control industrial tenemos: Temperatura, Presión, Voltaje, Corriente, Iluminación, Movimiento, Tamaño, Conteo.

A continuación presentamos una tabla con las acciones para cada dispositivo y su presentación final.

Acción Primaria	Dispositivo	Presentación Final
Sensar	Temperatura	Valor entre 0 – 100ºC
	Presión	Valor entre 15 – 102 Kpa
	Voltaje	Valor 120 - 240 - 380 V.
	Corriente	Valor pre-seteado
Prender	Iluminación	Prendido - Apagado
	Ventilación (a/c)	Prendido - Apagado
	Motores y Bombas	Prendido - Apagado
	Electro Válvulas	Prendido - Apagado
Puertas Eléctricas		Abierta - Cerrada
Monitoreo de Movimiento	Dirección de paso	Valores pre-seteados
	Movimiento	Valores pre-seteados
	Tamaño	Valores pre-seteados
	Conteo	Valor en unidades / RPM

Tabla 1.1 Acciones para cada dispositivo utilizado y su presentación final.

1.2. Instrumentación virtual

1.2.1. ¿Qué es instrumentación virtual?.

Mucho hemos oído hablar sobre la "instrumentación virtual" y sus beneficios. El concepto de instrumentación virtual nace a partir del uso del computador personal (PC) como "instrumento" de medición de tales señales como:

- temperatura
- presión
- caudal
- etc.

Es decir, el PC comienza a ser utilizado para realizar mediciones de fenómenos físicos representados en señales de corriente (Ej. 4-20mA) y/o voltaje (Ej. (0-5Vdc). Sin embargo, el concepto de "instrumentación virtual" va más allá de la simple medición de corriente o voltaje, sino que también involucra el procesamiento, análisis, almacenamiento, distribución y despliegue de los datos e información relacionados con la medición de una o varias señales específicas. Es decir, el instrumento virtual no se conforma con la adquisición de la señal, sino que también involucra la interfaz hombre-máquina, las funciones de análisis y procesamiento de señales, las rutinas de almacenamiento de datos y la comunicación con otros equipos.

El instrumento virtual es definido entonces como una capa de software y hardware que se le agrega a un PC en tal forma que permite a los usuarios interactuar con la computadora como si estuviesen utilizando su propio instrumento electrónico "hecho a la medida".

1.2.2. ¿Cómo construir un instrumento virtual?.

Para construir un instrumento virtual, sólo requerimos de un PC, una tarjeta de adquisición de datos con acondicionamiento de señales (PCMCIA, ISA, XT, PCI, etc.) y el software apropiado, los tres (3) elementos clave en la conformación de un instrumento virtual, teniendo un chasis de acondicionamiento de señales como elemento opcional.

Decimos que el "acondicionamiento de señales" es opcional, porque dependiendo de cada señal y/o aplicación, se puede o no requerir amplificación, atenuación, filtraje, aislamiento, etc., de cada señal. Si la señal está en el rango de los +/- 5Vdc y no se requiere de aislamiento o filtraje, la misma puede ser conectada directamente la tarjeta de adquisición de datos.

En el instrumento virtual, el software es la clave del sistema, a diferencia del instrumento tradicional, donde la clave es el hardware. Con el sistema indicado anteriormente, podríamos construir un osciloscopio "personalizado", con la interfaz gráfica que uno desee, agregándole inclusive más

funcionalidad. Sin embargo, este mismo sistema puede también ser utilizado en la medición de temperatura, o en el control de arranque / parada de una bomba centrífuga. Es allí donde radica uno de los principales beneficios del instrumento virtual, su flexibilidad. Este instrumento virtual no sólo me permite visualizar la onda, sino que a la vez me permite graficar su espectro de potencia en forma simultánea. ¿Podría hacer algo así con un instrumento convencional?.

1.2.3.Comparación del instrumento virtual versus el tradicional.

Para finalizar, la siguiente tabla nos indica algunas de las principales diferencias entre el instrumento convencional o tradicional, y el instrumento virtual:

Instrumento Tradicional	Instrumento Virtual
Definido por el fabricante	Definido por el usuario
Funcionalidad específica, con conectividad limitada.	Funcionalidad ilimitada, orientado a aplicaciones, conectividad amplia.
Hardware es la clave.	Software es la clave
Alto costo / función	Bajo costo / función, variedad de funciones, reusable.
Arquitectura "cerrada"	Arquitectura "abierta".
Lenta incorporación de nuevas tecnología.	Rápida incorporación de nuevas tecnologías, gracias a la plataforma PC.
Bajas economías de escala, alto costo de mantenimiento.	Altas economías de escala, bajos costos de mantenimiento.

Tabla 1.2 Instrumentos Tradicionales vs. Virtuales

La flexibilidad, el bajo costo de mantenimiento, la rehusabilidad, la personalización de cada instrumento, la rápida incorporación de nuevas tecnologías, el bajo costo por función, el bajo costo por canal, etc., son algunos de los beneficios que ofrece la instrumentación virtual.

La instrumentación virtual puede también ser implementada en equipos móviles (laptops), equipos distribuidos en campo (RS-485), equipos a distancia (conectados vía radio, Internet, etc.), o equipos industriales (NEMA 4X, etc.). Existe una tarjeta de adquisición de datos para casi cualquier bus o canal de comunicación en PCs (ISA, PCI, USB, serial RS-232/485, paralelo EPP, PCMCIA, CompactPCI, PCI, etc.), y existe un driver para casi cualquier sistema operativo (WIN 3.1/95/NT, DOS, Unix, MAC OS, etc.).

1.3. Desarrollo de sistemas de adquisición de datos y control.

1.3.1. Uso de tecnologías estándar para adquisición de datos y control.

Hoy en día, más y más compañías en la industria de la manufactura usan computadoras personales en sus plantas y laboratorios para probar sus productos, tomar mediciones y automatizar procesos. Al crear sistemas basados en una PC, los usuarios están aprovechando las ventajas de las tecnologías de la computación mas recientes como tarjetas de adquisición de

datos (DAQ) insertables, almacenamiento de datos a disco duro, Active X e Internet.

A medida que ingenieros y científicos adoptan la computadora para resolver un mayor número de aplicaciones, estas se han vuelto más exigentes; tal es el caso de los sistemas en tiempo real. Si bien es cierto que un sistema de adquisición de datos y control en tiempo real no es fácil de lograr con una computadora personal, lo ideal sería contar con las mismas ventajas que ofrece una PC (interfaz gráfica, sistemas abiertos, conectividad, bajo costo) y la habilidad de crear aplicaciones lo suficientemente confiables para desempañar el control de un sistema crítico.

1.3.2. Control en tiempo real.

"Tiempo Real" es uno de los términos mas comúnmente usados en la industria, pero su definición es ambigua. La mayoría de los ingenieros están de acuerdo en que Tiempo Real significa "con retrasos aceptables". El término *Tiempo Real duro* comúnmente se utiliza para definir a un sistema que debe ejecutarse sin falla y cumplir con los requerimientos de tiempo real en todo momento.

El error más común es pensar que Tiempo Real significa en realidad, rápido; cuando de hecho, muchas aplicaciones de adquisición de datos y control

tienen ciclos muy lentos. Los controladores de temperatura, por ejemplo, comúnmente hacen un muestreo y controlan la temperatura un par de veces por segundo. Así que para que el controlador de temperatura sea estable, debe ejecutar los lazos de control en el orden de un par por segundo. Es el grado de inseguridad con cada tiempo de ciclo del lazo de control el que define los requerimientos de Tiempo Real de un sistema.

1.3.3 Retos al crear un sistema de tiempo real.

La tecnología Windows para desarrollar sistemas de Tiempo Real aún representa retos. En sistemas operativos de Tiempo Real, las interrupciones y eventos son jerarquizados y los eventos con la mayor prioridad se ejecutan antes que los eventos de prioridad menor. En Windows NT/98/95 los drivers o manejadores de dispositivos responden a interrupciones; aunque la tecnología multihilo ha incrementado la confiabilidad y manejo de interrupciones, aún hay situaciones cuando el driver no tiene prioridad sobre otros eventos menos críticos. Puesto que Windows no puede garantizar que va a responder a un evento siempre dentro de un mismo tiempo, a esto se le llama determinístico o no determinístico.

Para resolver la respuesta no determinística de Windows, muchos lazos de control se desarrollan en hardware especializado. Hoy en día existen controladores para diferentes plataformas como VME, PCI o CompacPCI.

Algunos controladores son fijos o tienen poca flexibilidad de programación, otros usan sistemas cerrados y patentados. Aprender nuevos sistemas operativos, software propietario y hardware especializado toma tiempo y muchas veces es difícil la integración a sistemas que no son de Tiempo Real.

1.4 Sistemas de adquisición de datos analógicos y digitales

1.4.1 Sistemas de instrumentación.

Los sistemas de adquisición de datos se utilizan para medir y registrar señales obtenidas básicamente de dos maneras: a) aquellas que se originan a partir de la medición directa de cantidades eléctricas, que pueden incluir voltajes de cd y ca, frecuencia o resistencia; suele hallarse en las áreas de prueba de componentes electrónicos, estudios ambientales y trabajos de control de calidad. b) Señales que se originan a partir de transductores, como galgas extensiométricas y termopares.

Los sistemas de instrumentación se pueden clasificar en dos clases principales: analógicos y digitales. Los sistemas analógicos tratan en forma analógica la información de mediciones. Un sistema analógico se puede definir como una función continua, con una gráfica de voltaje contra tiempo, o desplazamiento contra presión. Los sistemas digitales manejan la información en forma digital. Una cantidad digital puede consistir en un

número de pulsos discretos y discontinuos cuya relación de tiempo contiene información referente a la magnitud o naturaleza de la cantidad.

1.4.2 Sistema de adquisición de datos analógicos.

Un sistema de adquisición de datos analógicos consta de algunos o todos los elementos siguientes:

a) ***Transductores*** para la transformación de parámetros físicos en señales eléctricas.

b) ***Acondicionadores*** *de señales* para la amplificación modificación o selección de ciertas partes de estas señales.

c) ***Dispositivo de presentación visual*** para el monitoreo continuo de señales de entrada. Estos dispositivos pueden incluir osciloscopio de varios canales o de un solo canal, osciloscopio de almacenamiento, panel de medidores, desplegados numéricos, etcétera.

d) ***Instrumento de registro de gráficas*** para obtener un registro permanente de los datos de entrada.

e) ***Instrumentación de cinta magnética*** para guardar los datos de entrada, conservar su forma eléctrica original y reproducirlos posteriormente para un análisis mas detallado.

1.4.3 Sistema de adquisición de datos digitales.

Un sistema de adquisición de datos digitales puede incluir algunos o todos los elementos que se muestran en la figura 1.0. Las operaciones esenciales dentro de un sistema digital incluyen: manipulación de señales analógicas, medición, conversión y manejos de datos digitales, y programación y control interno. La función de cada elemento del sistema de la figura siguiente se describe a continuación:

a) ***Transductor*****.** Transforma parámetros físicos en señales eléctricas aceptables para el sistema de adquisición.

b) ***Acondicionadores de señales.*** Para la amplificación modificación o selección de ciertas partes de estas señales.

c) ***Explorador o Multiplexor*****.** Acepta múltiples entradas analógicas y las conecta secuencialmente a un instrumento de medición.

d) ***Convertidor de señal.*** Transforma la señal analógica en una forma aceptable para el convertidor analógico – digital. Un ejemplo de este dispositivo es un amplificador de voltajes de bajo nivel generados por termopares o galgas extensiométricas.

e) ***Convertidor analógico – digital.*** Convierte el voltaje analógico a su forma digital equivalente. La salida del convertidor A/D se puede desplegar visualmente y estar disponible como voltaje en pasos discretos para procesamiento posterior o grabación en un registrador digital.

f) ***Equipo Auxiliar.*** Esta sección contiene instrumento para funciones de programación de sistemas y procesamiento digital de datos.

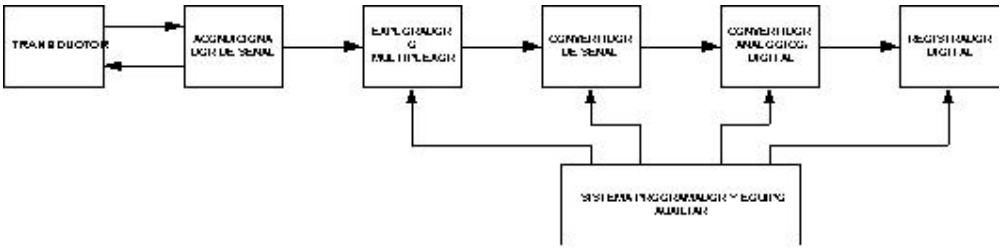

Figura 1.0. Elementos de un sistema de adquisición de datos digitales.

1.4.4 Interface de transductores a sistemas de medición y control electrónico.

Los voltajes y corriente de salida de muchos transductores son señales muy pequeñas. Además de los bajos niveles, a menudo es necesario transmitir la salida del transductor a cierta distancia hacia el equipo de colección de datos y control. En el arreglo de problemas, sobre todo en el ambiente industrial donde hay mucha maquinaria eléctrica, el ruido eléctrico puede causar serias dificultades en circuitos de bajo nivel. Estos ruidos pueden ser radiados, como un campo electromagnético o inducidos en el cableado de la planta, como circuito a tierra, y picos producidos por la fuente de alimentación de ca.

A pesar de las fuentes de ruido, las señales de bajo nivel se deben transmitir con cuidado de un lugar a otro. Un método efectivo para combatir el ruido es incrementar la intensidad de las señales de bajo nivel ante de su transmisión a través de los alambres. Esto se realiza con un amplificador llamado amplificador de instrumentación.

1.5. Arquitectura de diseño para aplicación de medición y automatización

1.5.1 Componentes de medición del programador.

Los ingenieros prefieren construir sus aplicaciones desde cero, sin embargo, conforme evoluciona la teoría de las aplicaciones convencionales, la arquitectura de diseño que empieza desde cero ha probado que requiere de mucho tiempo y es demasiado costosa.

Las arquitecturas basadas en componentes representan un paradigma, donde se hace una transición de aplicaciones monolíticas individuales a bloques básicos o cimiento de alto nivel que los programadores utilizan para diseño de mediciones. Los componentes ofrecen una función para crear nuevas tecnologías sin perder la eficiencia del diseño.

La complejidad de los sistemas de medición obliga a los ingenieros a convertirse en arquitectos del software. Mientras que los bloques básicos de un sistema de medición son sencillos, la tecnología ha agregado capas de sofisticación adicionales al mundo de la programación; esto presenta un reto para la comunidad de desarrollo de software para la medición y automatización.

Al desarrollar una aplicación para medición, los programadores pueden establecer una interfaz con PC con una de muchas fuentes de datos, incluyendo instrumentos autónomos, instrumentos insertables, tarjeta de adquisición de datos, tarjetas para control de movimiento y adquisición de imágenes y E/S distribuidas a lo largo de una red.

Idealmente al crear un sistema de medición independiente del hardware puede eliminar la dependencia de un dispositivo de adquisición especifico y prolongar la vida de su sistema de prueba; sin embargo, la independencia del hardware requiere una cantidad significativa de diseño del software al escribir un programa desde cero.

Además, los programadores enfrentan el reto de la presentación de datos. Cuando utilizan una arquitectura basada en componente, los programadores ya no están limitados a desplegar formas de onda en un monitor al presentar sus datos. Conforme más y más sistemas de medición se conectan a la internet, los programadores están distribuyendo datos a otras computadoras, conectándose a base de datos locales o remotos y presentando datos en muchos formatos diferentes, como HTML. Con la complejidad que representa el despliegue y análisis de datos, los ingenieros no deberían (y no pueden) ser responsables del desarrollo de los cimientos de un sistema de medición.

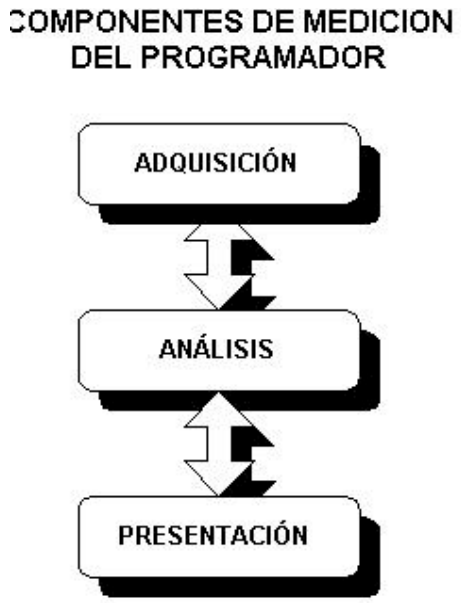

Figura1.1 Componentes de medición del programador

1.5.2 Bloques básicos de la medición.

Como fundamento para incorporar nuevas tecnologías, la arquitectura basada en componentes representa una evolución de los sistemas de medición. Los componentes son la base sobre la cual los programadores pueden construir sus aplicaciones. Los programadores pueden adaptar los componentes a una diversidad de tecnología basada en una interfaz para programación de aplicaciones estándar (API). Lo que da como resultado un diseño del sistema de medición más rico. Además, los programadores ahorran tiempo con este método por medio de la reutilización de código. Al utilizarse componentes que son originarios del sistema de desarrollo, los programadores pueden acortar el tiempo de aprendizaje y aprovechar la facilidad de depuración.

1.5.3. ¿Cómo diseñar componentes de medición ?.

El implementar una arquitectura basada en componentes comprende, desde la construcción de los cimientos o bloques básicos, hasta los niveles mas alto del sistema de medición. Cada uno de los cimientos esta presente en cada componente de medición el bloque de adquisición se integra al sistema operativo por una vía de inteligencia muy elevada para entregar información a las aplicaciones de medición. El bloque de análisis convierte los datos sin procesar que regresan del dispositivo de adquisición en información significativa. La presentación distribuye, despliega y almacena la programación para que los programadores puedan tomar decisiones bien fundamentadas.

Los bloques básicos de un componente de medición son solo el principio. Un programador puede utilizar cada aplicación que sea creada como componente dentro de un sistema de medición mayor, lo cual constituye un valioso elemento para los equipos de desarrollo. El siguiente ejemplo desglosa los beneficios que proporciona la reutilización de códigos a través de una arquitectura basada en componentes; facilita equipos de desarrollo de mayor tamaño, fomenta la cooperación y permite la creación de soluciones a nivel corporativo.

1.5.4 Ejemplo de Arquitectura basada en componentes.

La arquitectura de diseños basada en componentes alejan al programador del hardware y del lenguaje de programación, lo cual le da una aplicación completamente enfocada en el dispositivo a prueba.

Comencemos con un programador (Manuel) al cual se le ha asignado la tarea de adquirir, analizar desplegar y distribuir el sonido de un fenómeno físico a través de la Internet. Manuel elige el entorno del desarrollo Visual Basic. Sin utilizar ningún componente insertable disponible, Manuel debe establecer una interfaz con el manejador de bajo nivel para su dispositivo de adquisición, crear un algoritmo numérico para analizar su señal de entrada, crear una gráfica especializada para desplegar los datos y aprender TCP/IP o HTML para distribuir los datos a través de Internet. Ciertas tecnologías como las de National Instruments tienen componentes de medición disponibles y de esta manera se puede reducir su tiempo de desarrollo en más de un factor de cuatro. Ya no necesita crear herramientas especiales o aprender un nuevo protocolo o lenguaje de programación. Los componentes ofrecen una función para incorporar nuevas tecnologías sin perder la eficiencia del diseño.

1.5.5 Plano de la Arquitectura basada en componentes.

La revolución en medición representa un cambio crucial de paradigma en la manera en que los programadores desarrollan sistema de medición y, por lo tanto, un cambio de paradigma en lo referente al propósito del software, donde se hace una transición, pasando de los diseños individuales hechos desde cero a los bloques básicos de alto nivel utilizados en las aplicaciones para medición. Pronto, cualquier persona tendrá la oportunidad de desarrollar componentes para mediciones reutilizables de alto nivel, en cualquier lenguaje en forma sencilla y consistente, creando soluciones que sean utilizadas a nivel corporativo y beneficiándose de herramientas de ambiente de desarrollo originales y de la reutilización de código.

1.5.6 Mediciones a través de la red.

El uso de Internet esta adoptado más rápido que cualquier otra tecnología. La presencia universal de Internet y lar redes de PC nos guía a una nueva era de conectividad de información y compartimiento de datos que proporciona innovadoras formas de adquirir, analizar y presentar mediciones.

A continuación mostramos de forma simple las posibles soluciones para mediciones a través de la red en tres tipos básicos y recomendamos la tecnología más apropiada para crear cada solución.

- **Sistema de medición remoto:** un nodo PC que extrae mediciones a través de una red de uno o más nodos de medición remota.

- **Sistema de publicación de mediciones:** un nodo de medición que publica datos a través de una red a uno o más nodos PC remotos.

- **Sistema de medición empresarial:** múltiples nodos PC y de medición remota que comparten información a través de una red.

1.5.6.1 Sistema de medición remota.

En este caso, la meta es desplegar uno o más nodos de medición remota que envían mediciones al PC local. Los nodos remotos son, ya sean inteligentes o no inteligentes. Algunos ejemplos de aplicaciones incluyen distribuir E/S a lo ancho de una planta o proporcionar capacidad de entrada remota a un sistema para controlar o investigar fallas del sistema de medición.

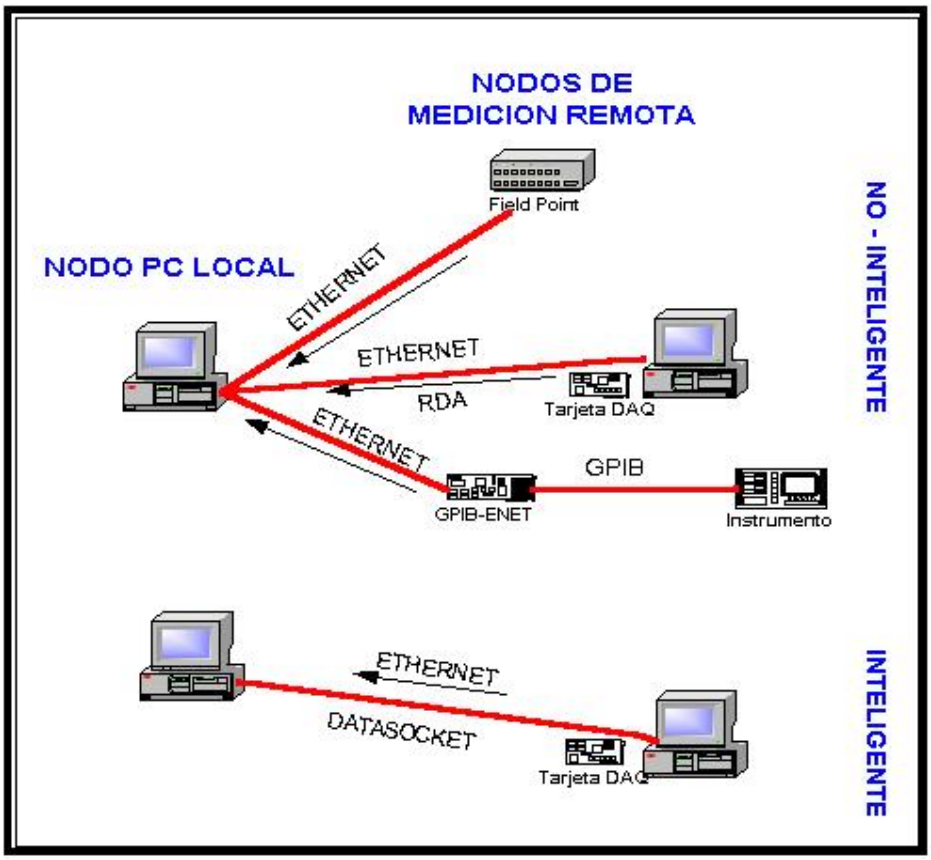

Figura 1.2 Sistema de medición Remota

1.5.6.2 Sistema de publicación de mediciones.

Este escenario es casi lo opuesto del mostrado anteriormente, ahora la meta es tener un nodo de medición que distribuya mediciones a uno o más nodos PC. Algunos ejemplos de aplicaciones incluyen la distribución de mediciones a lo ancho de su empresa o a la implementación de un laboratorio de aprendizaje a distancia en una universidad.

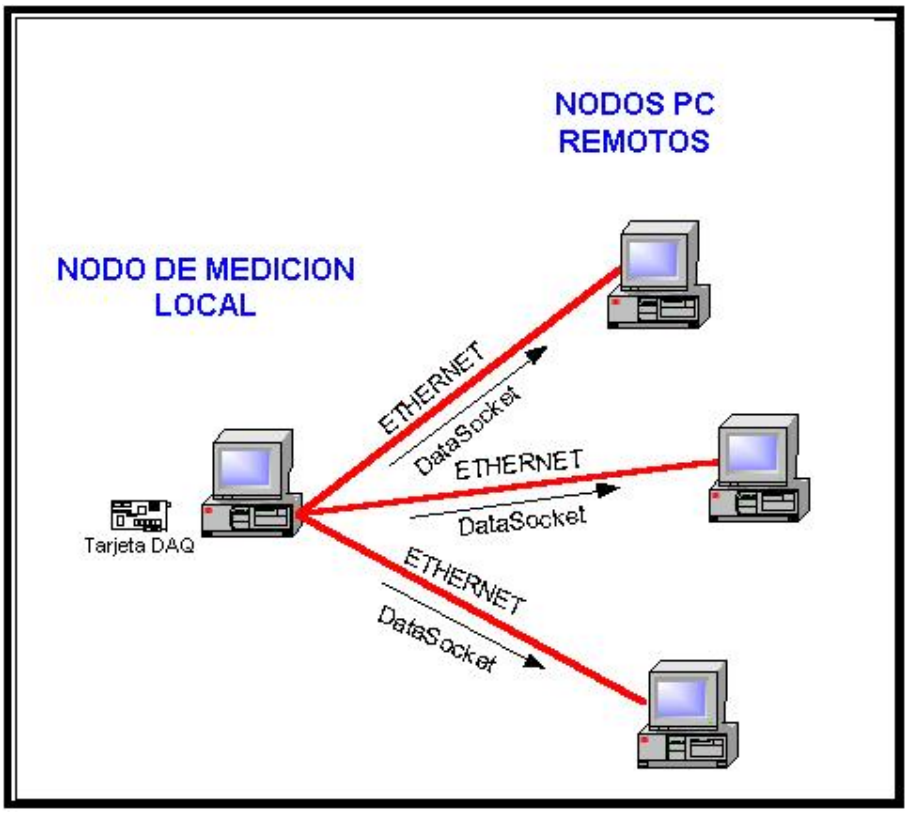

Figura 1.3 Sistema de publicación de mediciones

1.5.6.3 Sistema de medición empresarial

Este escenario es una combinación de los dos primeros, en este caso, la meta es proporcionar acceso a cualquier medición desde cualquier nodo PC o de medición. Los nodos de medición pueden ser de combinación de nodos inteligentes y no inteligentes.

La aplicación más común de este modelo es un sistema de adquisición de datos y control industrial en un piso de producción de una planta que integra el hardware de medición con interfaces hombre-máquina (HMI)/SCADA y otros paquetes de software industrial.

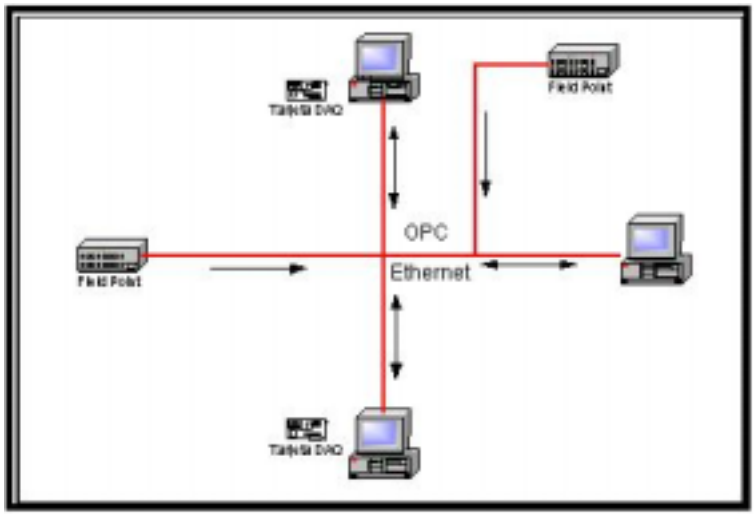

Figura 1.4 Sistema de medición empresarial

1.6. Sensores y transductores

1.6.1 Transductores como elementos de entrada a sistemas de instrumentación.

Un sistema de instrumentación electrónico consiste de varios componentes que se utiliza para realizar una medición y registrar el resultado. Por lo general consta de tres elementos principales: Un dispositivo de entrada, un

acondicionador de señal o dispositivo de procesamiento y un dispositivo de salida.

El dispositivo de entrada recibe la cantidad por medir y envía una señal eléctrica proporcional al dispositivo acondicionador de señal. Aquí la señal se amplifica, se filtra o se modifica en un formato para el dispositivo de salida. Este puede ser un simple medidor indicador, un osciloscopio o un registrador para presentación visual. Puede ser un registrador de cinta magnética para el almacenamiento temporal o permanente de los datos de entrada, o puede ser una computadora digital para manipulación de los datos o proceso de control. El tipo de sistema depende de qué se va a medir y de qué manera se van a presentar los resultados.

1.6.2. ¿Qué es un transductor?.

La variable de entrada de la mayoría de los sistemas de instrumentación es no eléctrica. Con el fin de utilizar métodos eléctricos y técnicas de medición, manipulación o control, las cantidades no eléctricas se convierten en una señal eléctrica por medio de un dispositivo llamado sensor o transductor, términos que se suelen emplear como sinónimos aunque el transductor engloba algo más amplio. Una definición establece que “el transductor es un dispositivo que al ser afectado por la energía de un sistema de transmisión, proporciona energía en la misma forma o en otra a un segundo sistema de

transmisión". Esta transmisión de energía puede ser eléctrica, mecánica, química, óptica (radiante) o térmica.

Esta amplia definición de un transductor incluye, por ejemplo, dispositivos que convierten fuerza o desplazamiento mecánico en una señal eléctrica. Estos dispositivos forman un grupo muy importante y numerosos de transductores que se encuentran en el área de instrumentación industrial y compete al ingeniero de instrumentación conocer este tipo de conversión de energía. Muchos otros parámetros físicos (calor, intensidad luminosa, humedad) se pueden convertir en energía eléctrica por medio de transductores. Estos dispositivos proporcionan una señal de salida cuando son estimulados por una entrada no mecánica; un termistor reacciona a variaciones de temperatura, una fotocelda a los cambios de intensidad luminosa, un haz electrónico a los efectos de un campo magnético, etc.

En todos los casos, la salida eléctrica se mide mediante métodos estándares dejando la magnitud de la cantidad de entrada en términos de una medida eléctrica analógica.

1.6.3. Estructura de los transductores.

Si nos limitamos a los transductores que se emplean para conectar a autómatas programables, a través de las interfaces adecuadas, podemos distinguir las siguientes partes que los componen:

- **Elemento sensor o captador.** Convierte las variaciones de una magnitud física en variaciones de una magnitud eléctrica (señal).

- **Tratamiento de la señal**. Si existe, realiza la función de modificar la señal obtenida para obtener una señal adecuada (filtrado, amplificación, etc.).

- **Etapa de salida**. Comprende los circuitos necesarios para poder adaptar la señal al nivel requerido para la carga exterior.

1.6.4. Clasificación de transductores.

Los transductores se pueden clasificar según su aplicación, método de conversión de energía, naturaleza de la señal de salida, etc. Por lo general todas estas clasificaciones terminan en áreas que se superponen. Una distinción y clasificación estricta de los tipos de transductores es difícil. El cuadro siguiente muestra una clasificación de transductores de acuerdo con los principios eléctricos en que se basan. La primer parte del cuadro, lista transductores que requieren potencia externa. Estos son los transductores pasivos, los cuales producen una variación en algún parámetro eléctrico,

como resistencia, capacitancia, etc., que se puede medir como una variación de voltaje o corriente.

La segunda categoría corresponde a transductores del tipo de autogeneración, que generan un voltaje o corriente analógica cuando son estimulados por medio de alguna forma física de energía. Los transductores de autogeneración no requieren potencia externa. Aunque sería casi imposible clasificar todos los sensores y mediciones, los dispositivos descritos en el siguiente cuadro representan un buen número de transductores disponibles en el comercio para aplicaciones de ingeniería instrumental.

1.6.5. ¿Cómo seleccionar un transductor ?.

En un sistema de medición el transductor es el elemento de entrada con la importante función de transformar algunas cantidades físicas en una señal eléctrica proporcional. La selección del transductor apropiado es, por consiguiente, el primero y tal vez el paso más importante en la obtención de resultados exactos. Un número de preguntas elementales se deben hacer antes de seleccionar un transductor, por ejemplo,

¿Cuál es la cantidad física por medir?

¿ Cuál principio de transductor es el mejor para medir esta cantidad?

¿ Qué exactitud se requiere en esta medición?

La primera se contesta determinando el tipo y rango de la medición. Para una respuesta apropiada a la segunda se requiere que las características de entrada y salida del transductor sean compatibles con el sistema de medición y registro. En la mayoría de los casos, estas dos interrogantes se responden fácilmente, al decir que el transductor apropiado se selecciona por la adición de una tolerancia para la exactitud. En la práctica esto rara vez es posible debido a la complejidad de los diversos parámetros del transductor que afectan la exactitud. Los requerimientos de exactitud del sistema total determinaron el grado con el cual los factores individuales contribuyen a la exactitud que debe ser considerada. Algunos de estos factores son:

a) **Parámetros fundamentales del transductor:** tipo y rango de la medición, sensibilidad, excitación.
b) **Condiciones de ambiente:** Conexiones eléctricas y mecánicas, condiciones de montaje, resistencia a la corrosión.
c) **Condiciones de ambiente:** efectos de la no-linealidad, efectos de histéresis, respuesta en frecuencia, resolución.
d) **Condiciones ambientales:** efectos de la temperatura, aceleración, golpes y vibraciones.

e) **Compatibilidad con el equipo asociado:** condiciones de balance de peso, tolerancia de la sensibilidad, acoplamiento de impedancias, resistencia de aislamiento.

Las categorías a) y b) comprenden características eléctricas y mecánicas básicas del transductor. La exactitud de éste, componente independiente, está contenida en las categorías c) y d). La categoría e) considera la compatibilidad del transductor con el equipo asociado al sistema.

1.6.6. Métodos para mejorar la exactitud global del sistema o reducir el error en la medición del transductor.

El error de medición total en un sistema activado por transductor se puede reducir para que esté dentro del rango de exactitud requerido, por medio de las siguientes técnicas:

- Usando un método de calibración de sistema con correcciones efectuadas en la reducción de datos.
- Monitoreo simultáneo del ambiente, con la consecuente corrección de datos.
- Control artificial del ambiente para minimizar los posibles errores.

Algunos errores individuales son previsibles y el sistema se puede estimar para eliminarlos. Cuando se calibra todo el sistema, estos datos de calibración sirven para corregir datos registrados. Los errores ambientales se corrigen reduciendo los datos si los efectos ambientales se registran al mismo tiempo que los datos reales. Entonces los datos se corrigen aplicando las características ambientales conocidas de los transductores. Estas dos técnicas incrementan de manera significativa la exactitud del sistema.

Otro método para mejorar la exactitud global del sistema es el control artificial del ambiente del transductor. Si se puede conservar sin cambio el ambiente del transductor estos errores se reducen a cero. Dicho tipo de control puede requerir al mover físicamente el transductor a una posición más favorable o aislarlo del medio ambiente mediante una cubierta a prueba de calor, aislamiento de vibraciones, o medios similares.

1.6.7. Tipos de transductores con galgas extensiométricas

1.6.7.1 Galgas extensiométricas.

La galga extensiométrica es un ejemplo de transductor pasivo que convierte un desplazamiento mecánico en un cambio de resistencia. Una galga extensiométrica es un dispositivo delgado, como una oblea, que se puede unir (soldar) a una variedad de materiales con el fin de medir los esfuerzos

aplicados. Las galgas extensiométricas metálicas se fabrican con alambres resistentes de diámetros muy pequeños. La resistencia del alambre o de la lámina delgada cambia con la longitud a medida que el material al cual está soldada sufre tensiones o compresiones. Este cambio en la resistencia es proporcional a la tensión aplicada y se mide con un puente de Wheatstone adaptado especialmente.

Para las aplicaciones de mediciones de tensión mecánica es deseable una alta sensibilidad. Un factor de galga alto significa un cambio de resistencia relativamente grande el cual se mide con mas facilidad que un cambio pequeño en la resistencia.

1.6.7.2 Elementos sensores metálicos.

Las galgas extensiométricas metálicas se forman de alambres de resistencia delgados o grabados en hojas muy finas de metal. Por lo general los alambres de la galga son pequeños, están sujetos a un mínimo de fugas y se pueden utilizar en aplicaciones para altas temperaturas. Los elementos de las laminas son más grandes y estables que las galgas de alambres. Es factible utilizarlos en condiciones extremas de temperatura o en condiciones prolongadas de carga y pueden disipar con facilidad el calor autoinducido.

Se han elaborado varios materiales resistentes para emplearlos en galgas de alambre y lámina; como ejemplo tenemos una aleación níquel-cobre con un coeficiente de temperatura bajo. Se encuentra en galgas que se utilizan en la medición de tensiones dinámicas, donde los niveles de tensión alternante no excedan +/- 1500 µcm./cm. Los límites de operación de temperatura son de 10 ^{o}C a 200 ^{o}C.

1.6.7.3.Galga extensiométrica desoldada.

La galga extensiométrica desoldada consiste en un marco estacionario y una armadura que está colocada en el centro del marco. La armadura sólo se puede mover en una dirección. El desplazamiento está limitado por cuatro filamentos de alambre sensible a la tensión, devanado entre aisladores rígidos montados en el marco y en la armadura.

El transductor de galga extensiométrica desoldada se puede construir en una gran variedad de configuraciones, según el uso requerido. Su aplicación principal es en un transductor de desplazamiento, se puede atar una terminal a la armadura para medir el desplazamiento directamente.

Este transductor puede ser un detector de presión, puede ser usado como un fuelle, la fuerza en el extremo de este se transmite por medio de una terminal a la armadura y la unidad funciona como dinamómetro. Si se aplica presión a

un lado del fuelle y se abre el otro extremo a la atmósfera, se puede leer presión manométrica. Si el fuelle está al vacío y sellado se mide presión absoluta.

1.6.8 Tipos de transductores de desplazamiento.

El concepto de convertir una fuerza aplicada en un desplazamiento es la base para muchos tipos de transductores. Los elementos mecánicos que son usados para convertir la fuerza aplicada en desplazamiento se llaman dispositivos sumadores de fuerza. Los dispositivos sumadores de fuerza que se utilizan son:

a) Diafragma, plano o corrugado

b) Fuelles

c) Tubo de Bourdon, circular o enrollado

d) Tubo recto

e) Masa en cantilever, es suspensión simple o doble

f) Par de pivote

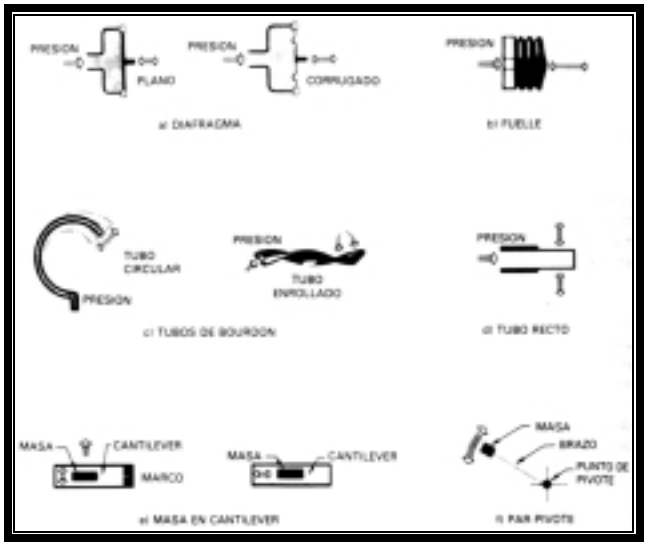

Figura 1.5 Dispositivos Sumadores de Fuerza

El desplazamiento creado por la acción del dispositivo sumador de fuerza se convierte en un cambio de algún parámetro eléctrico. Los principios eléctricos más usados en la medición de desplazamiento son:

a) Capacitivo

b) Inductivo

c) Transformador diferencial

d) Ionización

e) Oscilación

f) Fotoeléctrico

g) Piezoeléctrico

h) Potenciométrico

i) Velocidad

1.6.8.1 Transductor capacitativo.

Puesto que la capacitancia es inversamente proporcional al espaciamiento entre las placas paralelas, cualquier variación de la distancia entre las placas origina una correspondiente variación en la capacitancia. Este principio se aplica al transductor capacitivo. Una fuerza, aplicada a un diagrama que funciona como placa de un capacitor simple, cambia la distancia entre el diafragma y la placa estática. El cambio resultante es en capacitancia que se puede medir con un puente de CA, pero usualmente se mide con un circuito oscilador. El transductor, como parte del circuito oscilador, ocasiona un cambio en la frecuencia del oscilador. Este cambio de frecuencia es una medida de la magnitud de la fuerza aplicada.

El transductor capacitivo tiene una excelente respuesta en frecuencia y puede medir fenómenos estáticos como dinámicos. Sus desventajas son sensibilidad a variaciones de temperatura y la posibilidad de que se presenten señales erráticas o distorsionadas debido a terminales de gran longitud. También la instrumentación que recibe la señal puede ser grande y compleja, y a menudo incluye un segundo oscilador de frecuencia fija para propósitos heterodinos. La diferencia de frecuencia producida, se puede leer por medio de un dispositivo de salida apropiado como un contador electrónico.

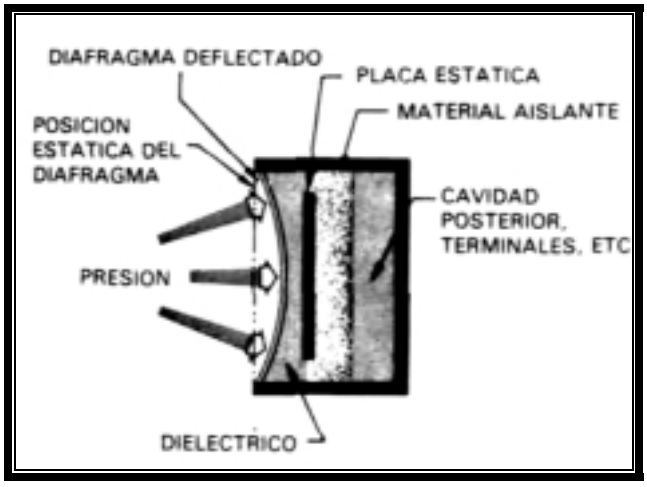

Figura 1.6 Transductor capacitivo

1.6.8.2 Transductor inductivo.

En el transductor inductivo la medición de fuerza se logra por medio del cambio en la razón de inductancia de un par de bobinas o mediante el cambio de inductancia en una sola bobina. En cada caso, la armadura ferromagnética se desplaza mediante la fuerza por medir, variando la reluctancia del circuito magnético. El cambio resultante en inductancia es una medición de la magnitud de la fuerza aplicada.

La bobina sirve como un componente de un oscilador LC cuya frecuencia variará con la fuerza aplicada. Este tipo de transductor es muy utilizado en sistemas de telemetría, con una sola bobina que modula la frecuencia de un oscilador local.

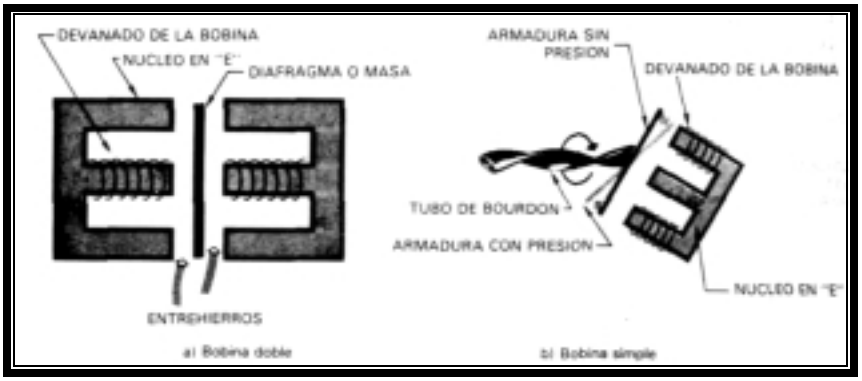

Figura 1.7 Transductor inductivo.

1.6.8.3 Transductor transformador diferencial variable.

El transformador diferencial es un transductor que mide fuerza en términos del desplazamiento del núcleo magnético de un transformador. El transformador consta de un solo devanado primario y dos devanados secundarios, los cuales se colocan a cada lado del primario (ver figura). Los secundarios tienen un número igual de vueltas pero están conectados en oposición y en serie, de forma que las fems inducidas en las bobinas se oponen entre sí. La posición del núcleo móvil determina el eslabonamiento de flujo entre el devanado primario excitado con C.A. y cada uno de los devanados secundarios.

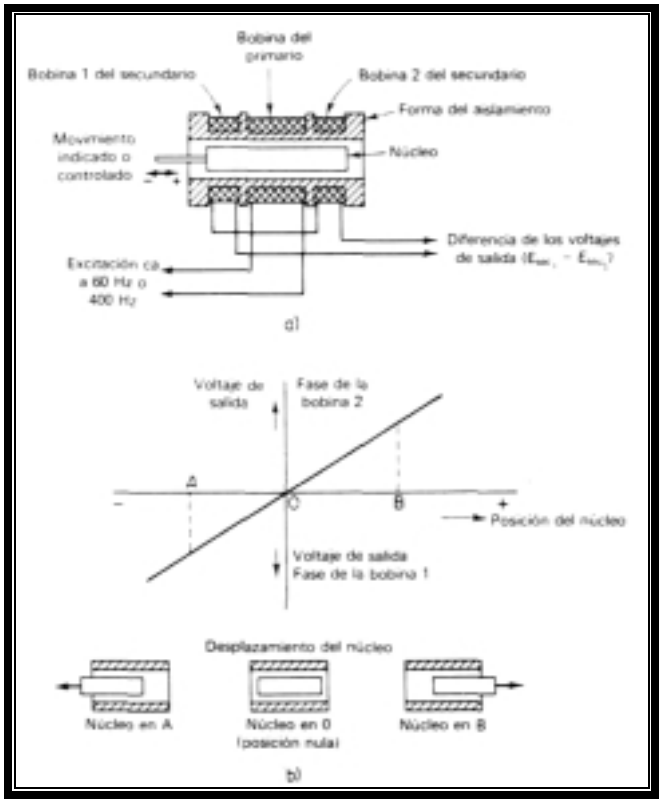

Figura 1.8 Transformador diferencial variable lineal: a) componentes esenciales; b) las posiciones relativas de los núcleos generan los voltajes de salida indicados

1.6.8.4 Transductor de oscilación.

En esta clase de transductores el elemento sumador de fuerza cambia la capacitancia o inductancia en un circuito oscilador LC. La frecuencia se afecta por un cambio en la inductancia de la bobina. La estabilidad del oscilador debe ser excelente para detectar cambios de frecuencia del oscilador causados por la fuerza aplicada desde el exterior.

Este transductor mide tanto fenómenos estáticos como dinámicos y es conveniente para aplicaciones en telemetría. Está restringido a aplicaciones de baja exactitud por su rango de temperatura limitado, pobre estabilidad térmica y baja exactitud.

1.6.8.5. Transductor piezoeléctrico.

Los materiales cristalinos asimétricos, como el cuarzo, la sal de Rochelle y el titanio de bario, producen una fem. cuando están expuestos a un esfuerzo. Esta propiedad se utiliza en transductores piezoeléctricos, donde un cristal se ubica entre una base sólida y un elemento sumador de fuerza. Una fuerza aplicada desde el exterior entra en el transductor a través de su apertura de presión aplicando presión a la parte superior del cristal. Esta produce una fem a través del cristal proporcional a la magnitud de la presión aplicada.

Ya que el transductor tiene una respuesta muy buena a alta frecuencia, su uso principal es en los acelerómetros de alta frecuencia. En esta aplicación el voltaje de salida es de 1 a 30 mV por g de aceleración. El dispositivo no necesita fuente de potencia externa y, por consiguiente, es del tipo de autogeneración. La desventaja principal de este es que puede medir condiciones estáticas.

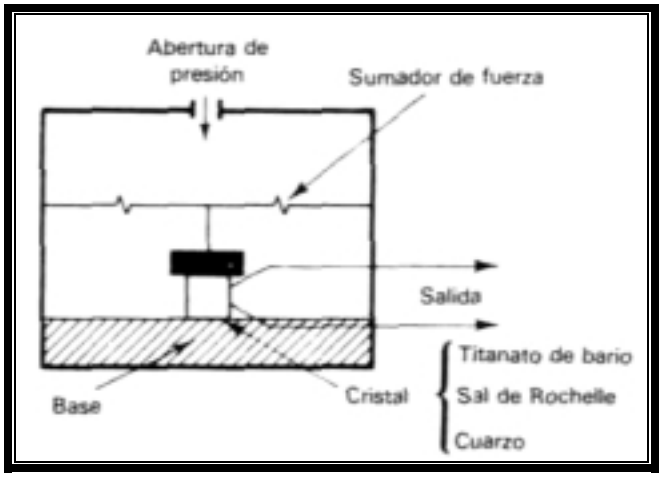

Figura 1.9. Elementos de un transductor piezoeléctrico.

1.6.8.6 Transductor potenciómetro.

Un transductor potenciómetro es un dispositivo electromecánico que contiene un elemento resistivo en contacto con un cursor móvil. El movimiento del cursor resulta en un cambio de resistencia que puede ser lineal, logarítmico, exponencial, etc., según la forma en la cual se devana el alambre de la resistencia. En algunos casos se usan depósitos de carbón, película de platino y otras técnicas para proporcionar el elemento resistivo.

El principio Potenciométrico se usa ampliamente a pesar de sus muchas limitaciones. Su eficiencia eléctrica es muy alta y proporciona una salida suficiente que permite operaciones de control sin mas amplificación. El dispositivo se puede excitar con C.A. o C.D. y, por tanto, sirve para una amplia gama de funciones. Debido a la fricción mecánica del cursor contra el

elemento resistivo, su vida está limitada y puede presentar ruido cuando el elemento esté desgastado. Se requieren desplazamientos grandes para mover el cursor a lo largo de toda la superficie de trabajo del potenciómetro.

1.6.8.7 Transductor de velocidad.

En esencia, el transductor de velocidad consiste en una bobina móvil suspendida en el campo magnético de un imán permanente Se genera un voltaje por el movimiento de la bobina en el campo. La salida es proporcional a la velocidad de la bobina. Este tipo de detector se suele utilizar para medir velocidades desarrolladas en formas lineales, senoidales o aleatorias. El amortiguamiento se obtiene por medios eléctricos, lo que asegura una alta estabilidad en condiciones de temperatura variable.

1.6.9 Sensores para mediciones de temperatura

1.6.9.1 Termómetros de resistencia.

Los detectores resistencia – temperatura, o termómetros de resistencia, emplean un elemento sensible de alambre de platino, cobre o níquel extremadamente puro que proporcionan un valor de resistencia definido para cada temperatura dentro de su rango. Casi todos los conductores metálicos tienen un coeficiente de temperatura de resistencia positiva, de manera que

la resistencia se incrementa con el aumento de temperatura. Algunos materiales, como el carbón y el germanio, tienen coeficiente de temperatura de resistencia negativa; esto significa que la resistencia decrece con el incremento de la temperatura.

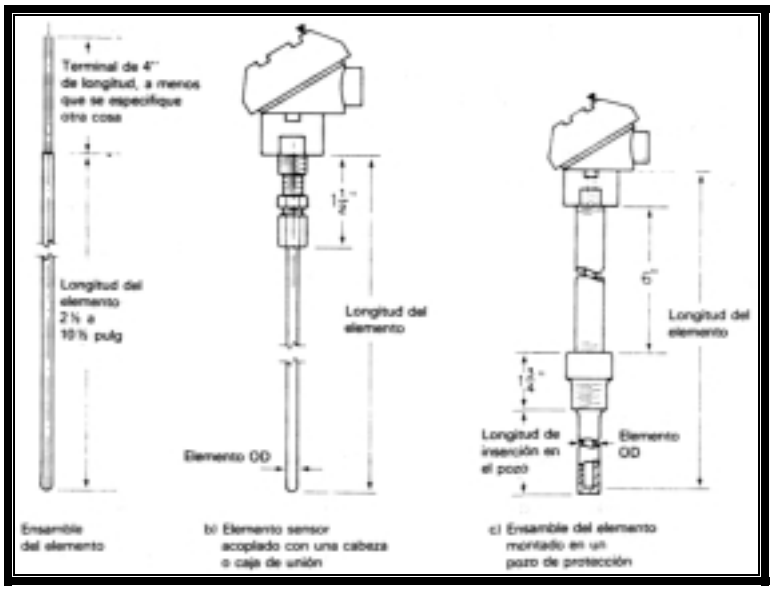

Figura 1.10. Termómetros de resistencia

1.6.9.2 Termopares.

En 1821 Thomas Seebeck descubrió que cuando dos metales disímiles están en contacto, se genera un voltaje cuando éste es función de la temperatura. El dispositivo, formado por dos metales disímiles unidos, se llama termopar y el voltaje se denomina voltaje Seebeck, en honor de su descubridor.

El método clásico para medir voltajes de termopares fue con potenciómetro. Este era un dispositivo mecánico que ya no se usa. Se utilizan dispositivos electrónicos para medir voltajes de termopares y convertir el voltaje Seebeck a temperatura y compensar por la unión de referencia.

1.6.9.3 Fuentes de error en los termopares.

Unión abierta. Hay muchas fuentes de error de una unión abierta y algunas ya se expusieron. En términos generales, el error introducido por una unión abierta es de tal magnitud que se localiza fácil. Esta se identifica mediante la simple medición de la resistencia del termopar.

Descalibración. Este error es una falla potencialmente grave, ya que puede ocasionar más errores sutiles que pueden escapar a su detección. La descalibración se debe a las alteraciones de las características del alambre del termopar, con lo que el voltaje Seebeck cambia. Puede deberse al someter el alambre a temperaturas demasiado altas, difusión de partículas atmosféricas dentro del alambre o por trabajarlo en frío. El último efecto pudo ser causado por tensión del alambre al jalarlo por un conducto largo.

Degradación de aislamiento El termopar a menudo se usa a temperaturas muy altas. En algunos casos el aislamiento se puede romper causando una resistencia de fuga significativa con el consecuente error en la medición del

voltaje Seebeck. Además, los elementos químicos en el aislamiento se pueden difundir en el alambre del termopar y ocasionar descalibración.

Acción galvánica Los elementos químicos que entraron en contacto con el alambre del termopar pueden causar acción galvánica. El voltaje resultante puede ser tanto como 100 veces más el efecto del voltaje Seebeck, lo que origina errores extremos.

Conducción térmica El alambre del termopar desvía energía calorífica de la fuente por medir. Para medir cuerpos de masa pequeña, se puede utilizar un termopar de diámetro pequeño. Sin embargo, el alambre de diámetro más pequeño es mas sensible a los efectos descritos. Si no se logra un compromiso razonable entre los efectos de degradación del alambre de termopar pequeño, las pérdidas de energía térmica con el error de temperatura resultante no se puede encontrar, se puede utilizar un alambre de extensión en el termopar. Esto permite que el termopar se elabore con alambre de diámetro pequeño; mientras que el alambre de extensión que cubre la mayoría de la distancia de conexión es de diámetro mucho mayor y no es susceptible a los efectos de degradación.

1.6.9.4 Características de los termistores.

Los termistores, o resistores térmicos, son dispositivos semiconductores que se comportan como resistencias con un coeficiente de temperatura de resistencia alto y, generalmente negativo. En algunos casos, la resistencia de un termistor a temperatura ambiente puede disminuir hasta un 6% por cada $1^{o}C$ que se eleve la temperatura. Dada esta alta sensibilidad al cambio de temperatura hacen al termistor muy conveniente para mediciones, control y compensar con precisión la temperatura. El uso de termistores está muy difundido en tales aplicaciones, en especial en el rango más bajo de temperatura de - 100 ^{o}C a 300 ^{o}C.

Tres características importantes del termistor lo hacen extremadamente útil en aplicaciones de medición y control: a) resistencia-temperatura, b) voltaje-corriente y c) corriente-tiempo.

1.6.9.5 Aplicaciones del termistor.

Aún cuando los termistores son más conocidos por sus funciones en la medición y control de la temperatura, tienen una gran variedad de aplicaciones, algunas de las cuales se describen a continuación.

El cambio relativamente grande en la resistencia por grado de temperatura en el termistor (llamado sensibilidad) lo convierte en una alternativa adecuada como transductor de temperatura. Un termistor industrial típico con una resistencia de 2000-Ω a 25^{o}C y un coeficiente de temperatura de 2.9%/^{o}C presentará un cambio de resistencia de 78 Ω/^{o}C por cambio en la temperatura.

Cuando este termistor se conecta en un circuito en serie simple consistente de una batería y un microamperímetro, cualquier variación en la temperatura ocasiona un cambio en la resistencia del termistor y un cambio correspondiente en la corriente del circuito. Es factible calibrar el medidor directamente en términos de temperatura, puede ser capaz de discriminar variaciones de temperatura de 0.1^{o}C.. El termistor de 4-kΩ fácilmente indica un cambio de temperatura tan pequeño como de 0.005^{o}C.

Esta alta sensibilidad, junto con la resistencia relativamente alta del termistor que se puede seleccionar (por ejemplo, 100 Ω), hace que el dispositivo sea ideal para mediciones o control remoto, ya que son despreciables los cambios de resistencia en líneas de transmisión o de contacto debidos a los efectos de la temperatura ambiente.

1.6.10 Dispositivos fotosensibles.

Los elementos fotosensibles son herramientas versátiles para detectar energía radiante a luz. Exceden la sensibilidad del ojo humano para todos los colores del espectro y operan aun en las regiones ultravioleta e infrarrojas. El dispositivo fotosensible ha encontrado uso práctico en muchas aplicaciones de ingeniería. Esta sección trata los siguientes dispositivos y sus aplicaciones:

a) Fototubos de tipo al vacío, usando más ventajosamente en aplicaciones que requieren la observación de pulsos de luz de corta duración, o luz modulada a frecuencias relativamente altas.

b) Fototubos de tipo gas, se emplean en la industria cinematográfica como sensores del sonido en la cinta.

c) Fototubos multiplicadores, con amplia capacidad de amplificación, se usan en mediciones fotoeléctricas y dispositivos de control, así como en contadores de centelleo.

d) Celdas fotoconductivas, también conocidas como fotorresistores o resistores dependientes de luz, encuentran un amplio uso en aplicaciones de control en la industria y en los laboratorios.

e) Celdas fotovoltaicas, son disponibles de unión de semiconductores utilizados para convertir energía radiante en potencia eléctrica. Un fino ejemplo es la celda solar que se usa en ingeniería espacial.

1.6.10.1 Fototubo al vacío.

El fotocátodo emite electrones cuando lo estimula la energía radiante incidente. El fotocátodo más importante que se utiliza en fototubos al vacío es la superficie de cesioantimonio, el cual se caracteriza por alta sensibilidad en el espectro visible. El tipo de vidrio empleado en la cubierta de vidrio determina principalmente la sensibilidad del dispositivo a otras longitudes de onda. Normalmente el vidrio corta la radiación transmitida en la región ultravioleta.

1.6.10.2 Fototubo lleno de gas.

El Fototubo lleno de gas tiene la misma construcción general que el fototubo al vacío, salvo que la cubierta contiene gas inerte (normalmente argón) a muy baja presión. Los electrones se emiten desde el cátodo por la acción fotoeléctrica y son acelerados a través del gas por el voltaje aplicado al ánodo. Si la energía de los electrones excede el potencial de ionización del gas (15.7 V para el argón), la colisión de un electrón con una molécula de gas puede resultar en ionización, esto es, la creación de un ion positivo y un segundo electrón. A medida que el voltaje se incrementa por arriba del potencial de ionización, la corriente colectada por el ánodo aumenta debido al mayor número de colisiones entre fotoelectrónes y las moléculas de gas.

Si el voltaje del ánodo se eleva a un valor aún mas alto, la corriente puede ser incontrolable; todas las moléculas del gas se ionizan y entonces el tubo exhibe una descarga resplandeciente. Esto se debe evitar ya que pueden ocurrir daños permanentes en el fototubo.

1.6.10.3 Fototubos multiplicadores.

Para detectar niveles muy bajos de luz, es necesaria una amplificación especial de la fotocorriente en la mayoría de las aplicaciones. El fototubo multiplicador, o fotomultiplicador, utiliza una emisión secundaria para proporcionar una amplificación de corriente en un factor de ion y entonces se convierte en un detector muy útil para niveles de luz muy bajos.

En un fotomultiplicador los electrones emitidos por el fotocátodo se dirigen electrostáticamente hacia una superficie de emisión secundaria, llamada dínodo.

1.6.10.4 Celdas fotoconductivas.

Las celdas fotoconductivas son elementos cuya conductividad es una función de la radiación electromagnética incidente. Muchos materiales son fotoconductivos en algún grado, pero los más importantes en el ámbito comercial son sulfito de cadmio, germanio y silicón. La respuesta espectral

de la celda de sulfito de cadmio es casi igual a la del ojo humano, por lo que se puede usar en aplicaciones donde la visión humana es importante, como el control del alumbrado urbano o el control del diafragma automático de las cámaras fotográficas.

1.6.10.5 Celdas fotovoltaicas.

Las celdas fotovoltaicas tienen varias aplicaciones. La celda solar de silicón convierte la energía radiante del Sol en potencia eléctrica. La celda solar consiste de una película delgada de un solo cristal de silicio tipo p, hasta de $2cm^2$, con una capa muy delgada (0.5 micrón) de material tipo n difundido en ella. La eficiencia de la conversión depende del contenido espectral y de la intensidad de la iluminación.

Se pueden utilizar dispositivos fotovoltaicos de silicón de unidades múltiples para detectar luz en aplicaciones como la lectura de tarjetas perforadas en la industria de procesamientos de datos.

Las celdas de germanio dopado con oro con características de respuesta espectral controlada actúan como dispositivos fotovoltaicos en la región infrarroja del espectro y sirven como detectores infrarrojos.

1.7. Controlador lógico programable

1.7.1 Definición de autómata programable.

Se entiende por controlador lógico programable (PLC), o autómata programable, a toda máquina electrónica diseñada para controlar en tiempo real y en medio industrial procesos secuenciales. Esta definición se está quedando un poco desfasada, ya que han aparecido los micro-plc's, destinados a pequeñas necesidades y al alcance de cualquier persona.

1.7.2 Campos de aplicación.

Un autómata programable suele emplearse en procesos industriales que tengan una o varias de las siguientes necesidades:

- Espacio reducido.
- Procesos de producción periódicamente cambiantes.
- Procesos secuenciales.
- Maquinaria de procesos variables.
- Instalaciones de procesos complejos y amplios.
- Chequeo de programación centralizada de las partes del proceso.
- Aplicaciones generales:

- Maniobra de máquinas.
- Maniobra de instalaciones.
- Señalización y control.

Tal y como dijimos anteriormente, esto se refiere a los autómatas programables industriales, dejando de lado los pequeños autómatas para uso más personal (que se pueden emplear, incluso, para automatizar procesos en el hogar, como la puerta de una cochera o las luces de la casa).

1.7.3 Ventajas e inconvenientes de los PLC's.

Entre las ventajas tenemos:

- Menor tiempo de elaboración de proyectos.
- Posibilidad de añadir modificaciones sin costo añadido en otros componentes.
- Mínimo espacio de ocupación.
- Menor costo de mano de obra.
- Mantenimiento económico.
- Posibilidad de gobernar varias máquinas con el mismo autómata.
- Menor tiempo de puesta en funcionamiento.

Si el autómata queda pequeño para el proceso industrial puede seguir siendo de utilidad en otras máquinas o sistemas de producción.

Y entre los inconvenientes:

- Adiestramiento de técnicos.
- Costo.

Hoy en día los inconvenientes se han hecho nulos, ya que todas las carreras de ingeniería incluyen la automatización como una de sus asignaturas. En cuanto al costo tampoco hay problema, ya que hay autómatas para todas las necesidades y a precios muy bajos.

1.7.4 Estructura externa.

Todos los autómatas programables, poseen una de las siguientes estructuras:

- **Compacta:** en un solo bloque están todos lo elementos.
- **Modular:**
- **Estructura americana:** separa las E/S del resto del autómata.
- **Estructura europea:** cada módulo es una función (fuente de alimentación, CPU, E/S, etc.).

Exteriormente nos encontraremos con cajas que contienen una de estas estructuras, las cuales poseen indicadores y conectores en función del modelo y fabricante. Para el caso de una estructura modular se dispone de la posibilidad de fijar los distintos módulos en railes normalizados, para que el conjunto sea compacto y resistente.

Los micro-autómatas suelen venir sin caja, en formato kit, ya que su empleo no es determinado y se suele incluir dentro de un conjunto más grande de control o dentro de la misma maquinaria que se debe controlar.

1.7.5 Estructura interna.

Los elementos esenciales, que todo autómata programable posee como mínimo, son:

- **Sección de entradas:** se trata de líneas de entrada, las cuales pueden ser de tipo digital o analógico. En ambos casos tenemos unos rangos de tensión característicos, los cuales se encuentran en las hojas de características del fabricante. A estas líneas conectaremos los sensores.

- **Sección de salidas:** son una serie de líneas de salida, que también pueden ser de carácter digital o analógico. A estas líneas conectaremos los actuadores.

- **Unidad central de proceso (CPU):** se encarga de procesar el programa de usuario que le introduciremos. Para ello disponemos de diversas zonas de memoria, registros, e instrucciones de programa.

Adicionalmente, en determinados modelos más avanzados, podemos disponer de funciones ya integradas en la CPU; como reguladores PID, control de posición, etc. Tanto las entradas como las salidas están aisladas de la CPU según el tipo de autómata que utilicemos. Normalmente se suelen emplear optoacopladores en las entradas y relés/optoacopladores en las salidas.

Aparte de estos elementos podemos disponer de los siguientes:

- **Unidad de alimentación** (algunas CPU la llevan incluida).
- **Unidad o consola de programación:** que nos permitirá introducir, modificar y supervisar el programa de usuario.
- **Dispositivos periféricos:** como nuevas unidades de E/S, más memoria, unidades de comunicación en red, etc.

- **Interfaces:** facilitan la comunicación del autómata mediante enlace serie con otros dispositivos (como un PC).

En los siguientes puntos comentaremos la estructura de cada elemento.

1.7.5.1 Memoria.

Dentro de la CPU vamos a disponer de un área de memoria, la cual emplearemos para diversas funciones:

- **Memoria del programa de usuario:** aquí introduciremos el programa que el autómata va a ejecutar cíclicamente.

- **Memoria de la tabla de datos:** se suele subdividir en zonas según el tipo de datos (como marcas de memoria, temporizadores, contadores, etc.).

- **Memoria del sistema:** aquí se encuentra el programa en código máquina que monitoriza el sistema (programa del sistema o firmware). Este programa es ejecutado directamente por el microprocesador o microcontrolador que posea el autómata.

- **Memoria de almacenamiento:** se trata de memoria externa que empleamos para almacenar el programa de usuario, y en ciertos casos

parte de la memoria de la tabla de datos. Suele ser de uno de los siguientes tipos: EPROM, EEPROM, o FLASH.

1.7.5.2 CPU.

La CPU es el corazón del autómata programable. Es la encargada de ejecutar el programa de usuario mediante el programa del sistema (es decir, el programa de usuario es interpretado por el programa del sistema). Sus funciones son:

- **Vigilar** que el tiempo de ejecución del programa de usuario no excede un determinado tiempo máximo (tiempo de ciclo máximo). A esta función se le suele denominar Watchdog (perro guardián).

- **Ejecutar** el programa de usuario.

- **Crear** una imagen de las entradas, ya que el programa de usuario no debe acceder directamente a dichas entradas.

- **Renovar** el estado de las salidas en función de la imagen de las mismas obtenida al final del ciclo de ejecución del programa de usuario.

- **Chequeo** del sistema. Para ello el autómata va a poseer un ciclo de trabajo, que ejecutará de forma continua:

Para ello el autómata va a poseer un ciclo de trabajo, que ejecutará de forma continua de la siguiente forma

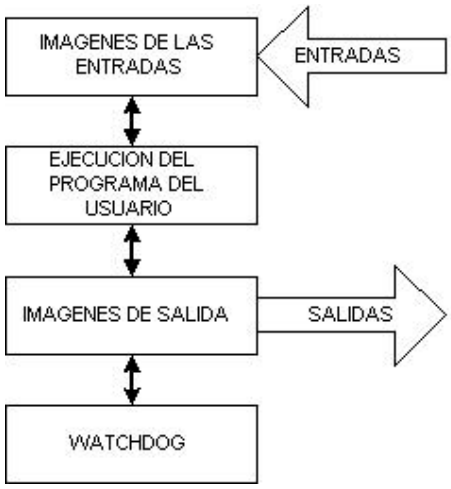

Figura 1.11 Ciclo de trabajo de la CPU.

1.7.5.3 Unidades de E/S.

Generalmente vamos a disponer de dos tipos de E/S:

-Digital.

-Analógica.

Las E/S digitales se basan en el principio de todo o nada, es decir o no conducen señal alguna o poseen un nivel mínimo de tensión. Estas E/S se manejan a nivel de bit dentro del programa de usuario.

Las E/S analógicas pueden poseer cualquier valor dentro de un rango determinado especificado por el fabricante. Se basan en conversores A/D y D/A aislados de la CPU (ópticamente o por etapa de potencia). Estas señales se manejan a nivel de byte o palabra (8/16 bits) dentro del programa de usuario.

Las E/S son leídas y escritas dependiendo del modelo y del fabricante, es decir pueden estar incluidas sus imágenes dentro del área de memoria o ser manejadas a través de instrucciones específicas de E/S.

1.7.5.4 Interfaces.

Todo autómata, salvo casos excepcionales, posee la virtud de poder comunicarse con otros dispositivos (como un PC). Lo normal es que posea una E/S serie del tipo RS-232 / RS-422. A través de esta línea se pueden manejar todas las características internas del autómata, incluida la programación del mismo, y suele emplearse para monitorización del proceso en otro lugar separado.

1.7.5.5.Equipos o unidades de programación.

El autómata debe disponer de alguna forma de programación, la cual se suele realizar empleando alguno de los siguientes elementos:

- **Unidad de programación:** suele ser en forma de calculadora. Es la forma más simple de programar el autómata, y se suele reservar para pequeñas modificaciones del programa o la lectura de datos en el lugar de colocación del autómata.

- **Consola de programación:** es un terminal a modo de ordenador que proporciona una forma más cómoda de realizar el programa de usuario y observar parámetros internos del autómata. Desfasado actualmente.

- **PC:** es el modo más potente y empleado en la actualidad. Permite programar desde un ordenador personal estándar, con todo lo que ello supone: herramientas más potentes, posibilidad de almacenamiento en soporte magnético, impresión, transferencia de datos, monitorización mediante software SCADA, etc.

Para cada caso el fabricante proporciona lo necesario, bien el equipo o el software / cables adecuados. Cada equipo, dependiendo del modelo y fabricante, puede poseer una conexión a uno o varios de los elementos

anteriores. En el caso de los micro-plc se escoge la programación por PC o por unidad de programación integrada en la propia CPU.

1.7.5.6. Dispositivos periféricos.

El autómata programable, en la mayoría de los casos, puede ser ampliable. Las ampliaciones abarcan un gran abanico de posibilidades, que van desde las redes internas (LAN, etc.), módulos auxiliares de E/S, memoria adicional, hasta la conexión con otros autómatas del mismo modelo. Cada fabricante facilita las posibilidades de ampliación de sus modelos, los cuales pueden variar incluso entre modelos de la misma serie.

1.8. Tipos de transductores

TRANSDUCTORES PASIVOS (CON POTENCIA EXTERNA)			
Parámetro eléctrico y clase de transductor	**Principio de operación y naturaleza del dispositivo**	**Aplicación típica**	
Dispositivo potenciómetro	El posicionamiento de un cursos por medio de una fuerza externa varía la resistencia en un potenciómetro o un circuito puente	Presión, desplazamiento	RESISTENCIA
Galga extensiómetrica resistiva	La resistencia de un alambre o semiconductor cambia según la elongación o compresión debida a esfuerzos aplicados externamente	Fuerza, par, desplazamiento	
Medidor de alambre caliente o medidor Pirani	La resistencia de un elemento caliente varía enfriándolo con flujo de gas	Flujo de gas, presión de gas	
Termómetro de resistencia	La resistencia de un alambre de metal puro con un coeficiente de temperatura de resistencia positivo grande varía con la temperatura.	Temperatura, calor radiante	
Termistor	La resistencia de ciertos óxidos de metal con coeficiente de temperatura de resistencia negativo cambia con la temperatura.	Temperatura	

Tabla 1.3 Transductores Pasivos (con potencia externa)

Parámetro eléctrico y clase de transductor	Principio de operación y naturaleza del dispositivo	Aplicación típica	
Higrómetro de resistencia	La resistencia de una cinta conductiva se altera con el contenido de humedad	Humedad relativa	RESISTENCIA
Celda fotoconductiva	La resistencia de una celda como un elemento del circuito se modifica con la luz incidente	Relevador fotosensible	
Transductor de circuito magnético	Los cambios del circuito magnético modifican la autoinductancia o inductancia mutua de una bobina	Presión, desplazamiento	INDUCTANCIA
Detector de reluctancia	La reluctancia de un circuito magnético varía al cambiar la posición del núcleo de hierro de una bobina	Presión, desplazamiento vibración, posición	
Transformador diferencial	El voltaje diferencial de dos devanados secundarios de un transformador varía al mover el núcleo magnético por medio de una fuerza aplicada desde el exterior	Presión, fuerza, desplazamiento, posición	
Medidor de corriente parásita	La inductancia de una bobina se altera por la proximidad de una placa con corrientes parásitas inducidas	Desplazamiento, espesor	

Tabla 1.4 Transductores Pasivos (con potencia externa). Continuación

Parámetro eléctrico y clase de transductor	Principio de operación y naturaleza del dispositivo	Aplicación típica	
Medidor de presión de capacitancia variable	Una fuerza aplicada externamente varía la distancia entre dos placas paralelas	Desplazamiento, presión	CAPACITANCIA
Micrófono de capacitor	La presión del sonido altera la capacitancia entre una placa fija y un diafragma móvil	Voz, música y ruido	
Medidor dieléctrico	La capacitancia varía por cambios en el dieléctrico.	Nivel de líquidos, espesor	
Detector por efecto de Hall	Se genera una diferencia de potencial a través de una placa semiconductora (de germanio) cuando un flujo magnético interactúa con una corriente aplicada	Flujo magnético, corriente	VOLTAJE Y CORRIENTE
Cámara de ionización	Se induce flujo de electrones mediante la ionización de un gas debido a radiación radiactiva	Conteo de partículas, radiación	
Celda fotoemisiva	Hay una emisión de electrones debida a la radiación incidente en una superficie fotoemisiva	Luz y radiación	
Tubo fotomultiplicador	La emisión de electrone secundarios es debida a la radiación incidente sobre un cátodo fotosensible	Luz y radiación, relevadores, fotosensibles	

Tabla 1.5 Transductores Pasivos (con potencia externa). Continuación

TRANSDUCTORES DE AUTOGENERACIÓN (SIN POTENCIA EXTERNA)		
Parámetro eléctrico y clase de transductor	**Principio de operación y naturaleza del dispositivo**	**Aplicación típica**
Termopar y termopila	Se genera una fem por la unión de dos metales disímiles o semiconductores cuando la unión se calienta	Temperatura, flujo de calor, radiación
Generador de bobina móvil	El movimiento de una bobina en un campo magnético genera un voltaje	Velocidad, vibración
Detector piezoeléctrico	Se genera una fem cuando la fuerza externa se aplica a ciertos materiales cristalinos, como el cuarzo	Sonido, vibración, aceleración, cambios de presión
Celda fotovoltaica	Se genera voltaje en un dispositivo de unión semiconductora cuando la energía radiante estimula la celda	Medidor de luz, celda solar

Tabla 1.6 Transductores de Autogeneración (sin potencia externa)

1.9. Criterios de evaluación y selección de un sistema de supervisión y control

Figura1.12 Foto tomada del informativo de National Instruments (NI).

Paso1: Identifique sus tipos de señales de E/S.

Paso2: Seleccione un método de acondicionamiento de señal.

Paso3: Seleccionar el dispositivo de E/S apropiado para la adquisición de datos.

Paso4: Seleccione los cables apropiados para su tarjeta y accesorios de acondicionamiento de señal.

Paso5: Seleccione su método de programación del software.

II. FUNDAMENTOS DE TCP / IP

2.1 Introducción

La gran rapidez con la que Internet ha crecido y se ha popularizado en los últimos años ha supuesto una revolución muy importante en el mundo de las comunicaciones, llegando a causar cambios en muchos aspectos de la sociedad. Lo que se conoce hoy como Internet es en realidad un conjunto de redes independientes (de área local y área extensa) que se encuentran conectadas entre sí, permitiendo el intercambio de datos y constituyendo por lo tanto una red mundial que resulta el medio idóneo para el intercambio de información, distribución de datos de todo tipo e interacción personal con otras personas.

Poca gente sabe lo que es TCP/IP aunque todos lo emplean indirectamente y lo confunden con un solo protocolo cuando en realidad son varios, de entre los cuales destaca y es el más importante el protocolo IP. Bajo este nombre (TCP/IP) se esconde uno de los protocolos mas usados del mundo, debido a que es el más usado por Internet y esta muy extendido en el sistema operativo UNIX.

Internet tiene su origen en la red informática ARPANET que fue un proyecto de la DARPA que en 1973 inició un programa de investigación de tecnologías de comunicación para transmitir paquetes de información e interconectar redes de diferentes características.

De este proyecto surgieron dos redes: Una de investigación, ARPANET, y una de uso exclusivamente militar, MILNET. Para comunicar las redes, se desarrollaron varios protocolos: El protocolo de Internet y los protocolos de control de transmisión. Posteriormente estos protocolos se englobaron en el conjunto de protocolos TCP/IP.

La red continuó extendiéndose por todo el país con gran rapidez, conectando a universidades e instituciones de investigación y educación, organizaciones gubernamentales o no gubernamentales, y redes privadas y comerciales. De esta manera continuó su desarrollo durante los años 80 extendiéndose

internacionalmente, pero ha sido en los 90 cuando Internet se ha convertido en un nuevo y revolucionario medio de comunicación a escala mundial.

Los nuevos medios desarrollados hacen el acceso a Internet mucho más sencillo y agradable para cualquier usuario y esto ha influido notablemente en esta expansión, convirtiendo a Internet en la gran red mundial.

Algunos de los motivos de la popularidad del TCP/IP son:

- Independencia del fabricante.
- Soporta múltiples tecnologías.
- Puede funcionar en máquinas de cualquier tamaño (multiplataforma).
- Estandar de EEUU desde 1983.
- Su destino esta ligado a la INTERNET.

La arquitectura de un sistema en TCP/IP tiene una serie de metas:

- La independencia de la tecnología usada en la conexión a bajo nivel y la arquitectura del ordenador.
- Conectividad Universal a través de la red.
- Reconocimientos de extremo a extremo.
- Protocolos estandarizados.

2.2. Interconectividad: conceptos, arquitectura y protocolos.

2.2.1 Interconectividad para un mundo heterogéneo.

En esta parte analizaremos el concepto fundamental de las interconexiones entre computadoras, la tecnología de la interconectividad para enlazar varias redes y formar un sistema de comunicación uniforme.

En una organización mundial con diversas necesidades de interconectividad se requieren varias redes físicas, y cada cual seleccionará la más adecuada para su tarea. No hay una sola tecnología de conectividad que sea la mejor para todas las necesidades.

El problema principal en la década de los setenta era que una computadora conectada a una red dada solo puede conectarse con las computadoras conectadas a la misma red, por lo que se opto por el servicio universal. El sistema de comunicación que da un servicio universal permite la comunicación de cualquier cantidad de pares de computadoras. El servicio universal es recomendable porque aumenta la productividad individual, pero las incompatibilidades entre el hardware de red y el direccionamiento físico impiden que las organizaciones construyan una red en puente que incluyan varias tecnologías.

A pesar de las incompatibilidades entre las tecnologías de red se diseño un esquema que ofrece servicio universal entre redes diferentes. Llamado interconectividad, el esquema utiliza tanto hardware como software. El software de todas las computadoras conectadas ofrece el servicio universal. El sistema de redes físicas interconectadas se llama interred.

2.2.2. Conexión física de redes mediante enrutadores.

El componente básico para conectar redes heterogéneas es el enrutador. El enrutador es un computador de propósito especial dedicado a interconectar redes, tienen procesador y memorias convencionales así como interfaces de E/S para todas las redes a las que se conecta. Este puede interconectar redes de diferentes tecnologías, incluyendo diferentes medios, esquemas de direccionamiento físico y formatos de cuadros.

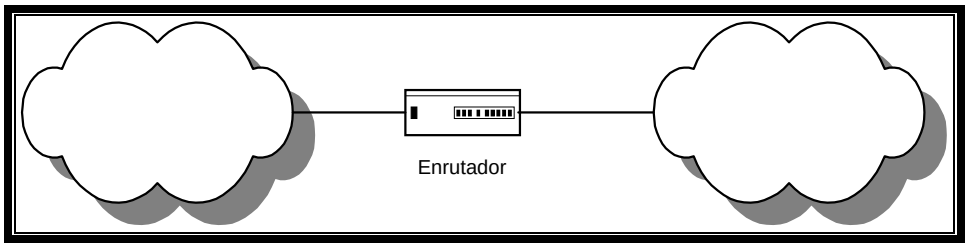

Figura 2.1. Dos redes físicas conectadas mediante un enrutador, que tiene una interfaz independiente para cada conexión de red. Las computadoras pueden conectarse a cualesquiera de las redes.

2.2.3. Arquitectura de las interredes

Una interred consta de un grupo de redes interconectadas mediante enrutadores. El esquema de interred permite que cada organización seleccione la cantidad y los tipos de red, los enrutadores para conectarlas y la topología de interconexión. En el siguiente ejemplo se muestra el uso de dos enrutadores para conectar siete redes físicas arbitrarias y formar una interred.

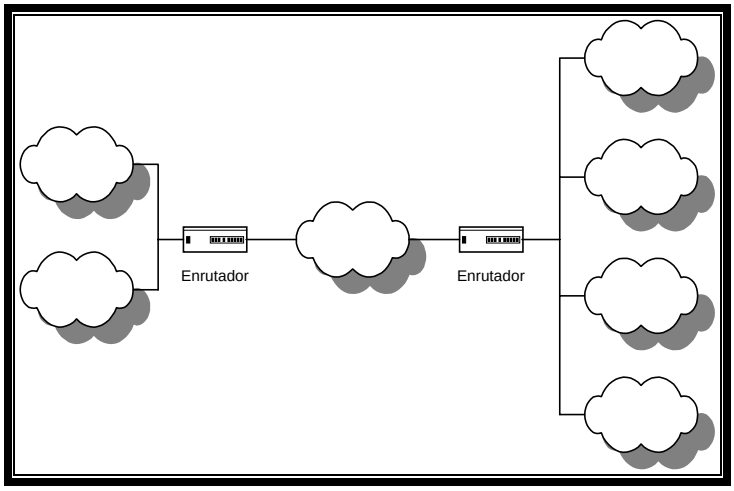

Figura 2.2 Interred formada por tres enrutadores que conectan siete redes físicas. Las redes pueden ser cualquier combinación de LAN y WAN.

2.2.4. Red Virtual.

Decimos que una interred es un sistema de red virtual porque el sistema de comunicación es una abstracción. Esto quiere decir que, aun teniendo una buena combinación de hardware y software no existe la red uniforme.

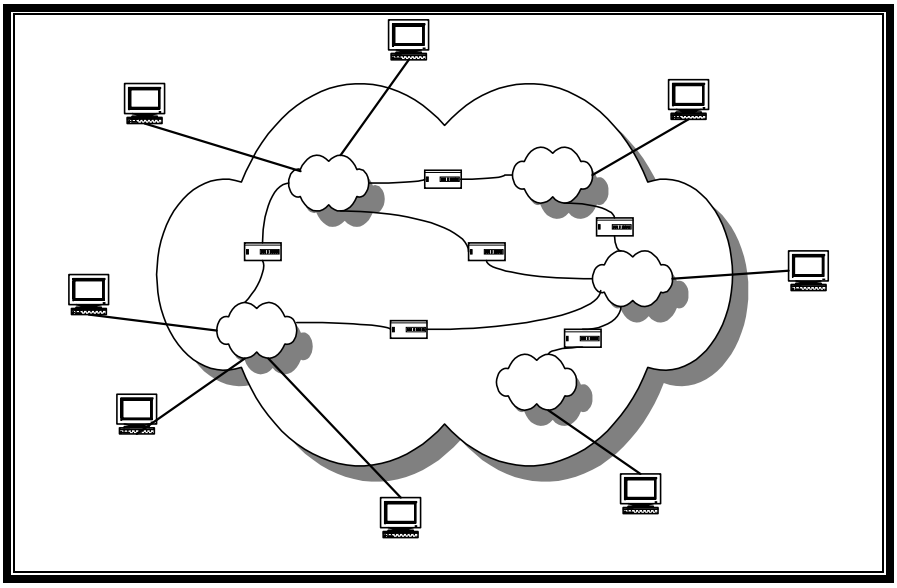

Figura 2.3. Estructura física en la que una computadora se conecta a una red física y los enrutadores interconectan las redes.

2.2.5. Protocolos de interconectividad.

Hay muchos estándares de protocolos de comunicación pero para el diseño de estos se toman en cuenta las siguientes reglas:

- Utilizar estándares existentes de protocolo siempre que dichos estándares se puedan aplicar.
- Inventar nuevos protocolos sólo cuando los estándares existentes no sean suficientes.
- Estar preparado a utilizar nuevos estándares cuando estén disponibles y proporcionen una funcionalidad equivalente.

Se han adaptados muchos protocolos para la interconectividad pero el más usado es el TCP/IP ya que esta ha permitido una Internet global. El software de protocolo TCP/IP funciona bien y maneja interredes grandes.

2.2.6. El Modelo OSI.

El modelo OSI (Open System Interconection) es utilizado por prácticamente la totalidad de las redes del mundo. Este modelo fue creado por la ISO (Organización Internacional de Normalización), y consiste en siete niveles o capas donde cada una de ellas define las funciones que deben proporcionar los protocolos con el propósito de intercambiar información entre varios sistemas.

Esta clasificación permite que cada protocolo se desarrolle con una finalidad determinada, lo cual simplifica el proceso de desarrollo e implementación. Cada nivel depende de los que están por debajo de él, y a su vez proporciona alguna funcionalidad a los niveles superiores.

Los siete niveles del modelo OSI son los siguientes:

Aplicación	El nivel de aplicación es el destino final de los datos donde se proporcionan los servicios al usuario.
Presentación	Se convierten e interpretan los datos que se utilizarán en el nivel de aplicación.
Sesión	Encargado de ciertos aspectos de la comunicación como el control de los tiempos.
Transporte	Transporta la información de una manera fiable para que llegue correctamente a su destino.
Red	Nivel encargado de encaminar los datos hacia su destino eligiendo la ruta más efectiva.
Enlace	Enlace de datos. Controla el flujo de los mismos, la sincronización y los errores que puedan producirse.
Físico	Se encarga de los aspectos físicos de la conexión, tales como el medio de transmisión o el hardware.

Tabla 2.1 Clasificación de las capas del modelo OSI.

2.2.7. Arquitectura cliente-servidor.

La arquitectura cliente-servidor es una forma específica de diseño de aplicaciones, aunque también se conoce con este nombre a los ordenadores en los que estas aplicaciones son ejecutadas. Por un lado, el cliente es el ordenador que se encarga de efectuar una petición o solicitar un servicio. El cliente no posee control sobre los recursos, sino que es el servidor el encargado de manejarlos. Por otro lado, el ordenador remoto que actúa como servidor evalúa la petición del cliente y decide aceptarla o rechazarla consecuentemente.

Una vez que el servidor acepta el pedido la información requerida es suministrada al cliente que efectuó la petición, siendo este último el responsable de proporcionar los datos al usuario con el formato adecuado. Finalmente debemos precisar que cliente y servidor no tienen que estar necesariamente en ordenadores separados, sino que pueden ser programas diferentes que se ejecuten en el mismo ordenador.

2.2.8. El protocolo TCP/IP.

TCP/IP es el protocolo común utilizado por todos los ordenadores conectados a Internet, de manera que éstos puedan comunicarse entre sí. Hay que tener en cuenta que en Internet se encuentran conectados ordenadores de clases muy diferentes y con hardware y software incompatibles en muchos casos, además de todos los medios y formas posibles de conexión. Aquí se encuentra una de las grandes ventajas del TCP/IP, pues este protocolo se encargará de que la comunicación entre todos sea posible. TCP/IP es compatible con cualquier sistema operativo y con cualquier tipo de hardware.

TCP/IP no es un único protocolo, sino que es en realidad lo que se conoce con este nombre es un conjunto de protocolos que cubren los distintos niveles del modelo OSI. Los dos protocolos más importantes son el TCP (*Transmission Control Protocol*) y el IP (*Internet Protocol*), que son los que

dan nombre al conjunto. En Internet se diferencian cuatro niveles o capas en las que se agrupan los protocolos, y que se relacionan con los niveles OSI de la siguiente manera:

- **Aplicación:** Corresponde a los niveles OSI de aplicación, presentación y sesión. Aquí se incluyen protocolos destinados a proporcionar servicios, tales como correo electrónico (SMTP), transferencia de ficheros (FTP), conexión remota (TELNET) y otros más recientes como el protocolo HTTP (*Hypertext Transfer Protocol*).

- **Transporte:** Coincide con el nivel de transporte del modelo OSI. Los protocolos de este nivel, tales como TCP y UDP, se encargan de manejar los datos y proporcionar la fiabilidad necesaria en el transporte de los mismos.

- **Internet:** Es el nivel de red del modelo OSI. Incluye al protocolo IP, que se encarga de enviar los paquetes de información a sus destinos correspondientes. Es utilizado con esta finalidad por los protocolos del nivel de transporte.

- **Enlace:** Los niveles OSI correspondientes son el de enlace y el nivel físico. Los protocolos que pertenecen a este nivel son los encargados de

la transmisión a través del medio físico al que se encuentra conectado cada *host*, como puede ser una línea punto a punto o una red *Ethernet*.

El TCP/IP necesita funcionar sobre algún tipo de red o de medio físico que proporcione sus propios protocolos para el nivel de enlace de Internet. Por este motivo hay que tener en cuenta que los protocolos utilizados en este nivel pueden ser muy diversos y no forman parte del conjunto TCP/IP. Sin embargo, esto no debe ser problemático puesto que una de las funciones y ventajas principales del TCP/IP es proporcionar una abstracción del medio de forma que sea posible el intercambio de información entre medios diferentes y tecnologías que inicialmente son incompatibles.

Para transmitir información a través de TCP/IP, ésta debe ser dividida en unidades de menor tamaño. Esto proporciona grandes ventajas en el manejo de los datos que se transfieren y, por otro lado, esto es algo común en cualquier protocolo de comunicaciones. En TCP/IP cada una de estas unidades de información recibe el nombre de "datagrama" (*datagram*), y son conjuntos de datos que se envían como mensajes independientes

2.3. IP: Direcciones de protocolos de interred.

2.3.1. IP (Internet Protocol).

El IP es un protocolo que pertenece al nivel de red, por lo tanto, es utilizado por los protocolos del nivel de transporte como TCP para encaminar los datos hacia su destino. IP tiene únicamente la misión de encaminar el datagrama, sin comprobar la integridad de la información que contiene. Para ello se utiliza una nueva cabecera que se antepone al datagrama que se está tratando. Suponiendo que el protocolo TCP ha sido el encargado de manejar el datagrama antes de pasarlo al IP, la estructura del mensaje una vez tratado quedaría así:

Figura 2.4. Estructura de un mensaje

2.3.2. La cabecera IP.

La cabecera IP tiene un tamaño de 160 bits y está formada por varios campos de distinto significado.

Estos campos son:

- **Versión:** Número de versión del protocolo IP utilizado. Tendrá que tener el valor 4. *Tamaño: 4 bits.*
- **Longitud de la cabecera:** (*Internet Header Length, IHL*) Especifica la longitud de la cabecera expresada en el número de grupos de 32 bits que contiene. *Tamaño: 4 bits.*
- **Tipo de servicio:** El tipo o calidad de servicio se utiliza para indicar la prioridad o importancia de los datos que se envían, lo que condicionará la forma en que éstos serán tratados durante la transmisión. *Tamaño: 8 bits.*
- **Longitud total:** Es la longitud en bytes del datagrama completo, incluyendo la cabecera y los datos. Como este campo utiliza 16 bits, el tamaño máximo del datagrama no podrá superar los 65.535 bytes, aunque en la práctica este valor será mucho más pequeño. *Tamaño: 16 bits.*
- **Identificación:** Valor de identificación que se utiliza para facilitar el ensamblaje de los fragmentos del datagrama. *Tamaño: 16 bits.*
- **Flags:** Indicadores utilizados en la fragmentación. *Tamaño: 3 bits.*
- **Fragmentación:** Contiene un valor (*offset*) para poder ensamblar los datagramas que se hayan fragmentado. Está expresado en número de grupos de 8 bytes (64 bits), comenzando con el valor cero para el primer fragmento. *Tamaño: 16 bits.*

- **Límite de existencia:** Contiene un número que disminuye cada vez que el paquete pasa por un sistema. Si este número llega a cero, el paquete será descartado. Esto es necesario por razones de seguridad para evitar un bucle infinito, ya que aunque es bastante improbable que esto suceda en una red correctamente diseñada, no debe descuidarse esta posibilidad. *Tamaño: 8 bits.*
- **Protocolo:** El número utilizado en este campo sirve para indicar a qué protocolo pertenece el datagrama que se encuentra a continuación de la cabecera IP, de manera que pueda ser tratado correctamente cuando llegue a su destino. *Tamaño: 8 bits.*
- **Comprobación:** El campo de comprobación (*checksum*) es necesario para verificar que los datos contenidos en la cabecera IP son correctos. Por razones de eficiencia este campo no puede utilizarse para comprobar los datos incluidos a continuación, sino que estos datos de usuario se comprobarán posteriormente a partir del campo de comprobación de la cabecera siguiente, y que corresponde al nivel de transporte. Este campo debe calcularse de nuevo cuando cambia alguna opción de la cabecera, como puede ser el límite de existencia. *Tamaño: 16 bits.*
- **Dirección de origen:** Contiene la dirección del *host* que envía el paquete. *Tamaño: 32 bits.*
- **Dirección de destino:** Esta dirección es la del host que recibirá la información. Los routers o gateways intermedios deben conocerla para dirigir correctamente el paquete. *Tamaño: 32 bit*s.

2.3.3. La dirección de Internet

Una dirección de internet (dirección IP) es un número de 32 bits asignado a un host y usado para todas las comunicaciones con él. Normalmente suele representarse con cuatro cifras de 8 bits separadas por puntos. A cada computadora se le asigna una dirección única y las asignaciones de números de red deben coordinarse globalmente, los sufijos pueden asignarse de manera local, sin coordinación global. Se han establecido 3 clases diferentes de direcciones, las cuales se representan mediante rango de valores.

2.3.4. Clases de direcciones IP.

El IP divide las direcciones de host en tres clases primarias. La clase de una dirección determina el limite entre el prefijo de red y el sufijo de host. Las tres clases primarias son: Clase A, B y C.

2.3.4.1 Clase A

Son las que en su primer byte tienen un valor comprendido entre 1 y 126, incluyendo ambos valores. Estas direcciones utilizan únicamente este primer byte para identificar la red, quedando los otros tres bytes disponibles para cada uno de los *hosts* que pertenezcan a esta misma red. Esto significa que

podrán existir más de dieciséis millones de ordenadores en cada una de las redes de esta clase. Este tipo de direcciones es usado por redes muy extensas, pero hay que tener en cuenta que sólo puede haber 126 redes de este tamaño. ARPAnet es una de ellas, existiendo además algunas grandes redes comerciales, aunque son pocas las organizaciones que obtienen una dirección de "clase A". Lo normal para las grandes organizaciones es que utilicen una o varias redes de "clase B".

Figura 2.5. Clase A

2.3.4.2. Clase B

Estas direcciones utilizan en su primer byte un valor comprendido entre 128 y 191, incluyendo ambos. En este caso el identificador de la red se obtiene de los dos primeros bytes de la dirección, teniendo que ser un valor entre 128.1 y 191.254 (no es posible utilizar los valores 0 y 255 por tener un significado especial). Los dos últimos bytes de la dirección constituyen el identificador del *host* permitiendo, por consiguiente, un número máximo de 64516 ordenadores en la misma red. Este tipo de direcciones tendría que ser suficiente para la gran mayoría de las organizaciones grandes.

En caso de que el número de ordenadores que se necesita conectar fuese mayor, sería posible obtener más de una dirección de "clase B", evitando de esta forma el uso de una de "clase A".

Figura 2.6. Clase B

2.3.4.3. Clase C

En este caso el valor del primer byte tendrá que estar comprendido entre 192 y 223, incluyendo ambos valores. Este tercer tipo de direcciones utiliza los tres primeros bytes para el número de la red, con un rango desde 192.1.1 hasta 223.254.254. De esta manera queda libre un byte para el *host*, lo que permite que se conecten un máximo de 254 ordenadores en cada red. Estas direcciones permiten un menor número de *host* que las anteriores, aunque son las más numerosas pudiendo existir un gran número redes de este tipo (más de dos millones).

Figura 2.7. Clase C.

2.3.4.4. Clases D y E.

La clase D se reserva todas las direcciones para multidestino (multicast), es decir, un ordenador transmite un mensaje a un grupo especifico de ordenadores de esta clase. Las direcciones estarán comprendidas entre 224.0.0.0. y 239.255.255.255.

Figura 2.8. Clase D.

La Clase E se utiliza exclusivamente para fines experimentales. Las direcciones están comprendidas entre 240.0.0.0 y 247.255.255.255.

Figura 2.9. Clase E.

2.3.5. Computo de la clase de una dirección y notación decimal con puntos.

Las direcciones IP son autoidentificables porque la clase de una dirección puede calcularse a partir de la dirección misma. A continuación se muestra una tabla para calcular la clase de una dirección.

PRIMEROS 4 BITS DE LA DIRECCIÓN	ÍNDICE DE LA TABLA (EN DECIMAL)	CLASE DE DIRECCIÓN
0000	**0**	**A**
0001	**1**	**A**
0010	**2**	**A**
0011	**3**	**A**
0100	**4**	**A**
0101	**5**	**A**
0110	**6**	**A**
0111	**7**	**A**
1000	**8**	**B**
1001	**9**	**B**
1010	**10**	**B**
1011	**11**	**B**
1100	**12**	**C**
1101	**13**	**C**
1110	**14**	**D**
1111	**15**	**E**

Tabla 2.2 Tabla para calcular una clase de dirección IP.

La notación decimal con puntos es una forma sintáctica usada por el software IP para expresar números binarios de 32 bits. La notación decimal con puntos representa los octetos en decimal, con puntos para separarlos.

Número binario de 32 bits	Decimal con puntos
10000001 00110100 00000110 00000000	**129.52.6.0**
10000000 10000000 11111111 00000000	**128.128.255.0**

Tabla 2.3 Ejemplo de direcciones IP.

2.3.6. Autoridad para asignar direcciones

En una interred, cada prefijo de red debe ser único. Para las asignaciones de direcciones los proveedores coordinan con una organización central, la

Autoridad de Números asignados en Internet (IANA), para asegurar que cada prefijo de red sea único en toda la Internet. Solamente es esencial para esta autoridad asignar direcciones IP para redes que estén conectadas a la red global Internet. Una corporación individual puede tener la responsabilidad de asignar direcciones únicas de red dentro de su red de redes TCP/IP, siempre y cuando nunca conecte esa red de redes al mundo exterior.

La Autoridad Internet asigna un número tipo C a una red con menos de 255 computadoras conectadas a ellas; se reserva los números tipo B para una organización que tiene una red más grande. Por último para obtener un número tipo A una organización debe tener una red con más de 65535 anfitriones conectados.

2.3.7. Ejemplo de un direccionamiento IP.

Consideremos una organización que va a interconectar 4 redes físicas. La organización debe adquirir enrutadores para interconectar las 4 redes y luego asignar las direcciones IP. Para comenzar, la organización escoge un prefijo único para cada red. El prefijo es un número de la clase A, B o C dependiendo del tamaño de la red.

En la siguiente figura se muestra un ejemplo. La dirección IP asignada a un host siempre comienza con el prefijo asignado a la red del host. Los sufijos, asignados por el administrador de red pueden ser arbitrarios.

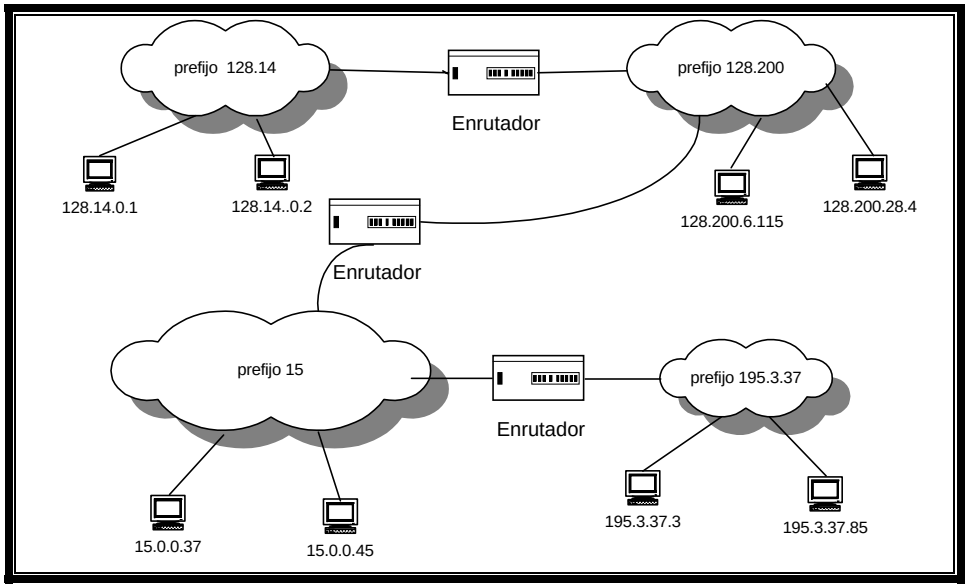

Figura 2.10. Ejemplo de interred privada con direcciones IP asignadas a los host. El tamaño de la red determina la clase de dirección asignada.

2.3.8. Direcciones IP especiales.

El IP define un grupo de Direcciones IP especiales que son reservadas. Es decir estas nunca se asignan a hosts.

Estas direcciones se las puede clasificar en:

- Dirección de red
- Dirección de difusión dirigida
- Dirección de difusión limitada.
- Dirección de esta computadora
- Dirección de retrociclo.

Prefijo	Sufijo	Tipo de dirección	Propósito
Ceros	ceros	esta computadora	se usa durante el arranque
Red	ceros	red	identifica una red
Red	unos	difusión dirigida	difusión en una red especificada
Unos	unos	difusión limitada	difusión en la red local
127	cualesquiera	retrociclo	pruebas de aplicaciones de red

Tabla 2.4 grupo de Direcciones IP especiales que son reservadas.

2.3.9. Enrutadores y principio de direccionamiento del IP.

A todos los enrutadores deben tener también asignarse direcciones IP. De hecho se asigna a cada enrutador 2 o más direcciones IP ya que tienen conexiones a varias redes físicas y cada dirección IP tiene un prefijo que especifica la red física.

Las direcciones IP no identifican computadoras especificas, sino que identifica una conexión entre una computadora y una red. A las computadoras con varias conexiones de red (por ejemplo, enrutadores) debe asignarse una dirección IP para cada conexión.

2.4. Ligas de direcciones de protocolo (ARP)

2.4.1.Direcciones de protocolos y entrega de paquetes.

Las direcciones del protocolo son abstracciones ofrecidas por el software; el hardware de red no sabe como localizar una computadora a partir de su dirección de protocolo. Antes de enviar los paquetes, hay que traducir la dirección de protocolo del siguiente salto en una dirección de hardware equivalente. Esta traducción se conoce como resolución de dirección y se dice que esta resuelta a la dirección de hardware correcta. Los host y enrutadores se sirven de la resolución de dirección cuando necesitan transmitir a otra computadora de la misma red. Las computadoras nunca resuelven la dirección de computadoras conectadas a redes remotas.

2.4.2. Técnicas de resolución de dirección

Los pasos básicos para la resolución de dirección se agrupan en 3 categorías:

TÉCNICAS DE RESOLUCIÓN DE DIRECCIÓN	OBSERVACIONES
Búsqueda en tabla.	Alto Costo de Mantenimiento
Cálculo de forma cerrada o aplicación de algoritmos.	Puede no lograrse una no homogénea distribución de direcciones. Remota posibilidad de duplicación de direcciones. Dificultad de elegir el algoritmo más eficiente
Intercambio de mensajes.	Se consulta mediante un solo mensaje que se emite a todos los equipos de red por el poseedor de cierta dirección IP.

Tabla 2.5 pasos básicos para la resolución de dirección

2.4.3. Protocolo de Resolución de direcciones ARP

Este protocolo le permite a un equipo obtener la dirección física de un equipo destino, ubicado en la misma red física, proporcionando solamente la dirección IP de destino.

La dirección IP y física de la computadora que consulta es incluida en cada emisión general ARP, el equipo que contesta toma esta información y actualiza su tabla de conversión.

ARP es un protocolo de bajo nivel que oculta el direccionamiento de la red en las capas inferiores, permitiendo asignar, a nuestra elección, direcciones IP a los equipos de una red física.

2.4.4. Entrega de mensajes ARP.

La norma ARP especifica que un mensaje ARP se puede colocar en un cuadro y difundir a todas las computadoras de red. Todas las computadoras mencionadas en la solicitud transmiten una respuesta; las demás procesan y descartan la solicitud sin transmitir respuestas.

Hay que recalcar que cuando una computadora transmite una respuesta ARP, ésta no se difunde, sino que se ponen en un cuadro y se envía directamente a la computadora que emitió la solicitud.

2.4.5. Formato de mensajes ARP

El formato de mensajes ARP no es fijo, lo que le permite ser usado por otros protocolos de alto nivel. No hay un tamaño fijo para los campos de dirección de hardware porque podrían inventarse nuevas tecnologías de red con direcciones mayores que el tamaño seleccionado. Aunque el formato de mensajes ARP es lo bastante general para permitir direcciones de protocolos y de hardware arbitrarios, el ARP casi siempre se usa para ligar direcciones IP de 32 bits a direcciones Ethernet de 48 bits.

2.4.6. Transmisión de un mensaje ARP

Cuando se transmite un mensaje ARP de una computadora a otra, éste viaja en un cuadro de hardware. El mensaje ARP se trata como datos en transportación. El hardware de red no sabe nada acerca del formato de mensaje ARP ni examina el contenido de los campos.

En términos técnicos, la colocación de un mensaje en un cuadro se llama encapsulamiento; el ARP se encapsula directamente en un cuadro de hardware. En la figura siguiente se ilustra el concepto.

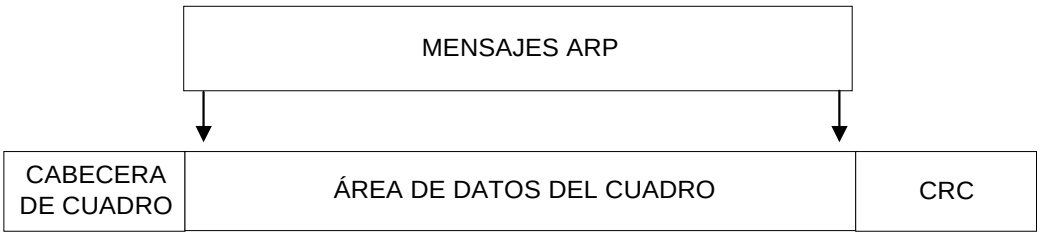

Figura 2.11. Ilustración de un mensaje ARP encapsulado en un cuadro Ethernet. El mensaje ARP viaja en el área de datos del cuadro, el hardware de red ni interpreta ni modifica el contenido del mensaje ARP.

2.4.7. Identificación de los cuadros ARP

El campo de tipo de la cabecera de cuadro especifica que el cuadro contiene un mensaje ARP, de esta manera la computadora sabe si un cuadro que llega tiene un mensaje ARP. El transmisor es el que asigna el número

adecuado al campo de tipo antes de transmitir el cuadro, y el receptor examina el mismo campo de cada cuadro que llega.

2.4.8. Procesamiento de un mensaje ARP de entrada

Al llegar un mensaje ARP, el protocolo especifica que el receptor debe ejecutar dos pasos básicos.

En el primero, el receptor extrae la liga de dirección del transmisor y comprueba si está presente una liga en el caché. De ser así, usa la liga del mensaje ARP de entrada para reemplazar la liga anterior. La actualización de una liga almacenada es una mejora especialmente útil en los casos en los que ha cambiado la dirección de hardware del transmisor.

En el segundo paso, el receptor examina el campo OPERACIÓN del mensaje para decidir si el mensaje es una solicitud o una respuesta. Si es una respuesta, el receptor debe haber emitido previamente una solicitud y, por lo tanto, está esperándola liga. Si es una solicitud, el receptor compara el campo DIR P OBJETIVO con la dirección local de protocolo. Si son iguales, la computadora es el objetivo de la solicitud y debe enviar una respuesta ARP. Para formar la respuesta, comienza por el mensaje de entrada, invierte las ligas del transmisor y del objetivo, inserta su dirección de hardware en el campo DIR H TRANSMISOR y cambia el campo de OPERACIÓN a 2.

2.4.9. Capas, resolución de dirección, direcciones de protocolo

El software de resolución de dirección esconde las direcciones físicas, lo que permite que el software de las capas superiores use direcciones de protocolo. De esta forma, se impone un límite conceptual importante entre la capa de interfaz de red y las capas superiores: las aplicaciones, así como los protocolos de las capas superiores, se construyen para usar sólo direcciones de protocolo.

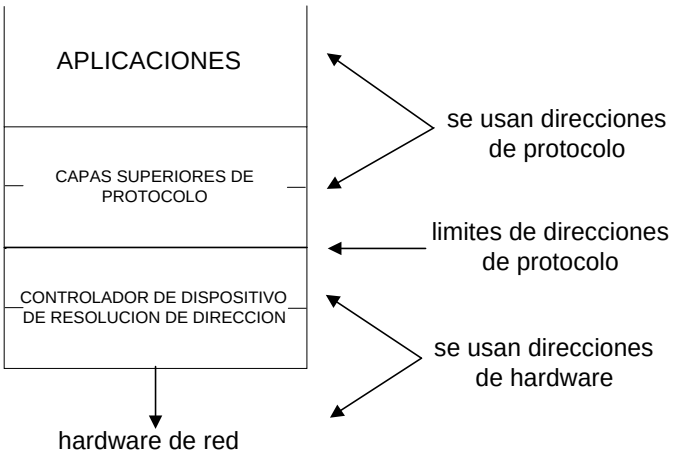

Figura 2.12. Software de protocolo en capas y limite conceptual entre la capa de interfaz de red y las capas superiores. El software por encima del limite usa direcciones de protocolo; el software por debajo traduce las direcciones de protocolo en direcciones físicas equivalentes.

2.5. Datagramas IP y reenvío de datagramas

2.5.1. Servicio sin conexiones.

La meta de la interconectividad es ofrecer un sistema de comunicación que permita la transmisión de datos de una computadora a otra. En una interred bien diseñada, los programas de aplicación transmiten y reciben sin darse cuenta de las redes físicas, pues no saben detalles de la red local a la que se conectan las computadoras ni la interconexión entre ambas.

Los diseñadores deben decidir los servicios de comunicación que ofrecerá el protocolo de interred, así como el modo de prestar con eficacia estos servicios. En particular, deben decidir si ofrecer servicio orientado a conexión, servicio sin conexiones o ambos.

2.5.2. Paquetes virtuales.

El servicio de interred sin conexiones es una extensión de la conmutación de paquetes que permite que el transmisor envíe paquetes de datos por una interred. Cada paquete viaja en forma independiente y tiene información que identifica el destinatario.

Debido a que es posible conectar redes heterogéneas, los enrutadores no pueden transmitir cuadros de una red a otra. Para acomodarse a la heterogeneidad, las interredes deben definir un formato de paquete independiente del hardware.

2.5.3. Datagrama IP.

Los protocolos TCP/IP usan el nombre datagrama IP para referirse a los paquetes de interred. Los datagramas IP tienen el mismo formato general que los cuadros de hardware, el datagrama comienza por una cabecera seguida de un área de datos.

Figura 2.13 Forma general de los datagramas IP, con una cabecera seguida de datos. La cabecera tiene información que controla cuándo y dónde se envía el datagrama.

La cantidad de datos contenidos en el datagrama no es fija. El transmisor determina la cantidad adecuada al propósito. La aplicación que transmite los datos determina el tamaño de los datagramas. El permitir que varíe el tamaño de los datagramas hace que el IP sea adaptable a varias aplicaciones.

En la versión actual del IP (versión 4), los datagramas puede contener desde un octeto hasta 64K octetos de datos, incluyendo la cabecera. En la mayor parte de los datagramas, la cabecera es mucho menor que el área de datos.

En resumen, los paquetes enviados por una interred TCP/IP se llaman datagramas IP. Cada datagrama consta de una cabecera seguida de datos. Las direcciones fuente y destino de la cabecera del datagrama son direcciones IP.

2.5.4. Reenvío de datagramas IP.

Hemos dicho que los datagramas atraviesan la interred siguiendo una trayectoria de su fuente inicial por los enrutadores al destino. Los enrutadores de la trayectoria reciben el datagrama, extraen de la cabecera la dirección destino y la utilizan para determinar el siguiente salto del datagrama. El enrutador reenvía el datagrama al siguiente salto, sea el destino u otro enrutador.

Para hacer eficiente la selección del siguiente salto y permitir que se entienda el cálculo, cada enrutador IP guarda información en una tabla de enrutamiento. La tabla de enrutamiento debe inicializarse al arrancar el enrutador y actualizarse si cambia la topología o falla el hardware.

En concepto, la tabla de enrutamiento tiene un grupo de entradas que especifican destino y siguientes saltos para llegar a ellos. En la siguiente figura se muestra el contenido de la tabla de enrutamiento de uno de los tres enrutadores que interconectan cuatro redes de una interred pequeña.

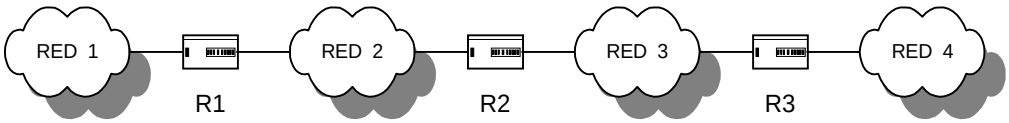

Figura 2.14 Interred con tres enrutadores que conectan cuatro redes físicas y tabla de enrutamiento conceptual del enrutador R2.

Destino	Siguiente Salto
Red 1	R1
Red 2	entrega directa
Red 3	entrega directa
Red 4	R3

Tabla 2.6. Indica una red destino y el siguiente salto de la ruta a esa red.

Cada destino listado en la tabla de enrutamiento es una red, no un host. La distinción es importante porque una interred puede tener más de mil veces más host que redes. Así, el usar las redes como destinos mantienen pequeñas las tablas de enrutamiento.

Debido a que cada destino de una tabla de enrutamiento corresponde a una red, la cantidad de entradas de una tabla de enrutamiento es proporcional a la cantidad de redes de la interred.

2.5.5. Direcciones IP y entradas de tablas de enrutamiento.

En la práctica, una tabla de enrutamiento IP es algo más complicada que la figura anterior. Primero, el campo destino de cada entrada tiene el prefijo de la red destino. Segundo, un campo adicional tiene una máscara de dirección que indica los bits del destino que corresponden al prefijo de red. Tercero, se usa una dirección IP cuando el campo de siguiente salto indica un enrutador. En la siguiente figura se ilustra la apariencia de la tabla de enrutamiento de la figura anterior.

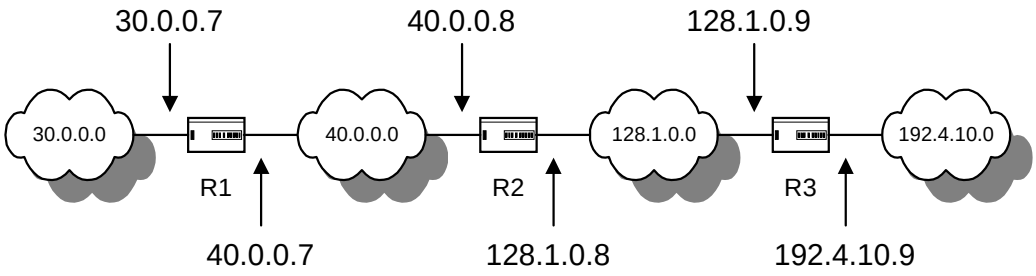

Figura 2.15 Interred de cuatro redes y tres enrutadores con una dirección IP asignada a cada interfaz de enrutador

Destino	Mascara	Siguiente Salto
30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	entrega directa
128.1.0.0	255.255.0.0	entrega directa
192.4.10.0	255.255.255.0	128.1.0.9

Tabla 2.7 Enrutamiento que indica un destino, una máscara y el siguiente salto para llegar al destino.

Las primeras dos redes de la figura tienen prefijos de clase A, la tercera de clase B y la cuarta de clase C. Cada enrutador tiene dos direcciones IP, una para cada interfaz. Por ejemplo, al enrutador que conecta la red 30.0.0.0 a la red 40.0.0.0 han sido asignadas las direcciones 30.0.0.7 y 40.0.0.7.

Aunque se ha asignado el mismo sufijo de host a ambas interfaces del enrutador, el IP no requiere uniformidad, y el administrador tiene la libertad de asignar diferentes números a las interfaces.

2.5.6. Campo de máscara y reenvío de datagramas.

El uso de una tabla de enrutamiento para seleccionar el siguiente salto de un datagrama se llama enrutamiento o reenvío. El campo de máscara de la red en tabla de enrutamiento se usa para extraer la parte de red de la dirección. Para entender el uso de la máscara, imagine que el software de enrutamiento recibe un datagrama para reenviar. También suponga que el datagrama tiene una dirección destino IP, D. El software de enrutamiento debe encontrar una entrada en la tabla de enrutamiento que indique un siguiente salto para D. Para ello, el software examina las entradas de la tabla, emplea la máscara para extraer el prefijo de la dirección D y compara el resultado con el campo destino. Si ambos son iguales se reenviará el datagrama al siguiente salto.

2.5.7. Direcciones de destino y de siguiente salto.

El campo de DIRECCION IP DESTINO del datagrama contiene la dirección del destino final. Cuando un enrutador recibe un datagrama, extrae la dirección destino D y la usa para calcular la dirección del siguiente destino enrutador al que se enviará, N. Aunque el datagrama se envía directamente a la dirección N, la cabecera del datagrama retiene la dirección destino D.

En otras palabras, la dirección destino de una cabecera de datagrama siempre se refiere al destino final. Cuando un enrutador reenvía el datagrama a otro enrutador, la dirección del siguiente salto no aparece en la cabecera del datagrama.

Todas las rutas se calculan usando direcciones IP. Tras calcular la dirección del siguiente salto, N, el software IP usa las ligas de dirección para traducir N ala dirección de hardware equivalente para la transmisión.

2.5.8. Entrega del mejor esfuerzo.

Además de definir el formato de los datagramas de interred, el IP define la semántica de la comunicación y emplea el término mejor esfuerzo para describir el servicio ofrecido. En esencia, la norma indica que, aunque el IP hace su mejor esfuerzo por entregar los datagramas, este puede fallar.

Debido a que el IP se diseñó para operar sobre todo tipo de hardware de red, el hardware puede fallar. Como resultado, los datagramas IP pueden perderse, duplicarse o entregarse fuera de orden o con datos corruptos. Se necesitan capas superiores de protocolo para manejar estos errores.

2.5.9. Formato de cabecera de datagrama IP.

En la siguiente figura se muestran los campos de una cabecera de datagrama IP, incluyendo la DIRECCIÓN IP FUENTE, que tiene la dirección de internet del transmisor, la DIRECCIÓN IP DESTINO, que es la dirección internet del destinatario, y el campo de TIPO, que especifica el tipo de datos.

<table>
<tr><td>0</td><td>4</td><td>8</td><td></td><td>16</td><td>19</td><td>24</td><td>31</td></tr>
<tr><td>VERS</td><td>LONG C</td><td colspan="2">TIPO SERVICIO</td><td colspan="4">LONGITUD TOTAL</td></tr>
<tr><td colspan="4">IDENTIFICACIÓN</td><td>BANDERAS</td><td colspan="3">DESPLAZAMIENTO DE FRAGMENTO</td></tr>
<tr><td colspan="3">TIEMPO DE VIDA</td><td>TIPO</td><td colspan="4">CIFRA DE COMPROBACIÓN DE CABECERA</td></tr>
<tr><td colspan="8">DIRECCIÓN IP FUENTE</td></tr>
<tr><td colspan="8">DIRECCIÓN IP DESTINO</td></tr>
<tr><td colspan="6">OPCIONES IP (PUEDEN OMITIRSE)</td><td colspan="2">RELLENO</td></tr>
<tr><td colspan="8">COMIENZO DE DATOS</td></tr>
</table>

Figura 2.16. Campos de la cabecera de datagrama IP. Tanto la fuente como el destino son direcciones de Internet.

Los campos de la cabecera de datagrama IP tienen tamaños fijos. El datagrama comienza con un número de versión de protocolo de cuatro bits y

una longitud de cabecera de cuatro bits que indica el número de cantidades de 32 bits de la cabecera. El campo de TIPO DE SERVICIO contiene una cifra que indica si el transmisor prefiere que el datagrama viaje por una ruta con retardo mínimo o una ruta de máximo rendimiento; el enrutador que conoce varias rutas al destino puede aprovecharla para escoger la ruta. El campo de LONGITUD TOTAL tiene un entero de 16 bits que indica la cantidad de octetos del datagrama, incluyendo la cabecera y los datos.

El campo de TIEMPO DE VIDA sirve para que los datagramas no viajen eternamente por una trayectoria en ciclo ya que estas trayectorias pueden aparecer cuando falla el software o el administrador configura mal las rutas. El transmisor inicia el campo de TIEMPO DE VIDA con un valor entero entre 1 y 255. Cada enrutador que maneja el datagrama disminuye en 1 el TIEMPODE VIDA. Si el contador llega a cero, el datagrama se descarta y se regresa un mensaje de error a la fuente.

El campo de CIFRA DE COMPROBACIÓN DE CABECERA asegura que los bits de la cabecera no cambien durante el tránsito. El transmisor calcula la suma de complemento de uno de todas las cantidades de 16 bits de la cabecera, excluyendo el campo de cifra de comprobación misma, y luego almacena el complemento de uno de la suma en el campo de CIFRA DE COMPROBACIÓN DE CABECERA. El receptor calcula la misma suma de

16 bits de los datos de la cabecera, incluyendo el campo de cifra de comprobación. Si la cifra de comprobación es correcta, el resultado es cero.

Para mantener pequeños los datagramas, el IP define un grupo de opciones que puede estar presente. Cuando un datagrama IP no lleva opciones, el campo etiquetado LONG C contiene 5 y la cabecera termina tras el campo DIRECCIÓN DESTINO. Dado que la longitud de la cabecera se especifica en múltiplos de 32 bits, si las opciones no terminan en un límite de 32 bits, se agrega un relleno con bits cero para hacer que la cabecera sea un múltiplo exacto.

2.6. Encapsulamiento IP, Fragmentación y Reensamble

2.6.1. Transmisión de datagramas y cuadros.

Cuando un host o enrutador maneja un datagrama, el software IP selecciona primero el siguiente salto al que lo enviará, N, y luego lo transmite allí por la red. Desgraciadamente, el hardware de red no entiende el formato de datagrama ni el direccionamiento de Internet. En cambio, cada tecnología de red define un formato de cuadro y un esquema de direccionamiento físico; el hardware sólo acepta y entrega paquetes que se apegan al formato de cuadro y al esquema de direccionamiento de hardware especificado.

2.6.2. Encapsulamiento

¿Cómo puede transmitirse un datagrama por una red que no entiende su formato?. La respuesta en una técnica conocida como Encapsulamiento. Cuando se encapsula un datagrama IP en un cuadro, el datagrama completo se pone en el área del cuadro. El hardware de red trata los cuadros con datagramas igual que cualquier otro datagrama. De hecho el hardware no examina ni cambia el contenido del área de datos del cuadro. En la siguiente figura se ilustra el concepto.

Figura 2.17 Datagrama IP encapsulado en un cuadro de hardware. El datagrama completo reside en el área de datos del cuadro. En la práctica, el formato de cuadro de algunas tecnologías incluye también una cola de cuadro.

¿Cómo sabe el receptor que el área de datos de un cuadro de entrada contiene un datagrama IP u otros datos?. El transmisor y el receptor deben acordar el número usado en el campo de tipo de cuadro. Al colocar el datagrama en el cuadro, el transmisor asigna al campo de tipo de cuadro el valor especial reservado para IP. Al llegar un cuadro con el número especial en el campo de tipo, el receptor sabe que el área de datos contiene un datagrama IP.

En resumen: Se encapsula un datagrama en un cuadro para transmitirlo por una red. La dirección destino del cuadro es la dirección del siguiente salto del datagrama; la dirección se obtiene traduciendo la dirección IP del siguiente salto a una dirección física equivalente.

2.6.3. Transmisión por una interred.

El encapsulamiento se aplica a una sola transmisión por vez. Tras seleccionar el siguiente salto, el transmisor encapsula el datagrama en un cuadro y transmite el resultado por la red al siguiente salto. Al llegar ahí, el receptor remueve el datagrama IP y descarta el cuadro. Si el datagrama debe reenviarse por otra red, se crea un cuadro nuevo. En la siguiente figura se ilustra el encapsulamiento y el desencapsulamiento de un datagrama al ir de fuente a destino por tres redes y dos enrutadores. Cada red puede usar una tecnología de hardware diferente, lo que significa que los formatos de cuadro pueden ser diferentes.

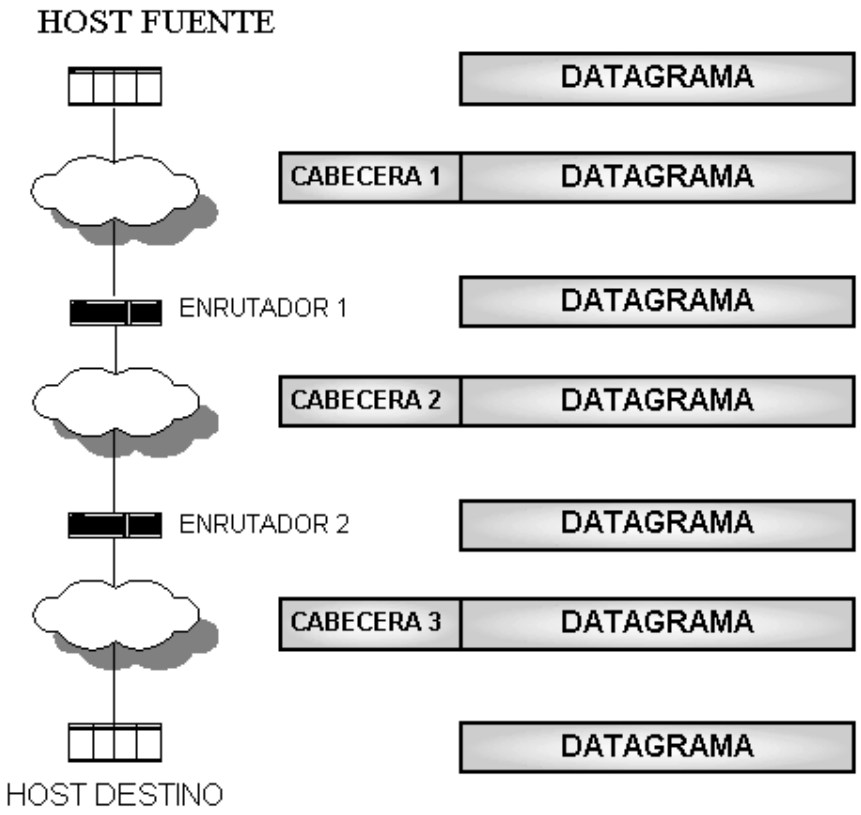

Figura 2.18 Datagrama IP, según aparece en cada paso durante su viaje por una interred.

Cuando el datagrama llega en un cuadro de red, el receptor lo extrae del área de datos del cuadro y descarta la cabecera del cuadro.

2.6.4. MTU, tamaño de datagrama y encapsulamiento.

Las tecnologías de hardware especifican el máximo de datos que puede llevar un cuadro. El límite se conoce como unidad máxima de transmisión (MTU). No hay excepciones al límite MTU. Por ello, los datagramas deben

se iguales o menor a la MTU de la red, o no es posible encapsulados para su transmisión. Cabe indicar que el hardware no se diseño para aceptar cuadros con más datos que los permitidos por la MTU.

Los enrutadores IP usan una técnica conocida como fragmentación para resolver el problema de las MTU heterogéneas. Cuando un datagrama es mayor que la MTU de la red por la que debe transmitirse, el enrutador divide el datagrama en partes más pequeñas llamadas fragmentos y los envía independientemente.

Ningún datagrama puede ser más grande que la MTU de la red por la que se envía. Cuando recibe un datagrama mayor que la MTU de la red por la que debe transmitirse, el enrutador lo divide en partes más pequeñas llamadas fragmentos. Cada fragmento usa el formato de datagrama IP, pero lleva sólo parte de los datos.

2.6.5. Reensamble

Se denomina reensamble al proceso de recreación del datagrama original a partir de fragmentos. Ya que cada fragmento comienza con una copia de la cabecera del datagrama original, todos tienen la misma dirección destino que el datagrama del que provienen, Además, el fragmento que lleva la parte final de los datos tiene establecidos un bit adicional de la cabecera. Por lo

tanto, el receptor a cargo del reensamble puede saber si han llegado bien todos los fragmentos.

2.6.6. Identificación del datagrama y pérdidas de fragmentos.

Recuerde que el IP no garantiza la entrega. Por lo tanto, pueden perderse fragmentos o llegar en desorden. Además, si una fuente envía varios datagramas a un mismo destino, los fragmentos de cada cual pueden llegar en orden arbitrario.

¿Cómo reensambla el software IP los fragmentos que llegan en desorden? El transmisor pone un número de identificación en el campo de IDENTIFICACIÓN de cada datagrama de salida. Al fragmentar el datagrama, el enrutador copia el número de identificación a cada fragmento. El receptor emplea el número de identificación y la dirección IP fuente del fragmento de entrada para determinar el datagrama al que pertenece. Además, el campo de DESPLAZAMIENTO DE FRAGMENTO indica al receptor cómo ordenar los fragmentos de los datagramas.

Recuerde que el IP no garantiza la entrega de datagramas, de modo que si la red desecha paquetes, puede perderse un datagrama o fragmento encapsulado. Al llegar todos los fragmentos, puede reensamblarse el datagrama; sin embargo, hay problemas cuando llegan varios fragmentos y

otros se retardan o pierden. Aunque el datagrama no puede reensamblarse, el receptor debe guardarlos fragmentos para el caso de que sólo estén retardados los faltantes.

2.7. Mecanismo de reporte de errores.

2.7.1 ¿Cómo se detectan los errores?.

Aunque el IP usa semántica de entrega del mejor esfuerzo, este incluye mecanismos de detección de errores y de reporte. Además de la cifra de comprobación de cabecera para detectar errores de transmisión, las implantaciones IP incluyen un sistema de reporte de errores conocido como protocolo de mensaje de control de interred (ICMP).

2.7.2. El protocolo de mensaje de control de interred. (ICMP)

El ICMP incluye mensajes informativos y mensajes para reportar errores. Los enrutadores transmiten mensajes ICMP de error a la fuente de un datagrama que causa problemas.

Es posible emplear los mensajes de error ICMP para probar una interred y obtener información. El programa ping usa mensajes ICMP de solicitud y respuesta de contestación para determinar si es asequible el destino. Otro

programa como el traceroute utiliza mensajes ICMP de tiempo excedido para encontrar la secuencia de enrutadores a distancia1, 2, 3, etc. De la trayectoria a un destino. Por último, los host pueden usar mensajes ICMP de fragmentación requerida para determinar la MTU de trayectoria de un destino.

2.8. TCP: Servicio de transportación confiable

2.8.1 Protocolo de control de transmisión.

Para que los programadores apliquen las técnicas convencionales durante la creación de aplicaciones que usan interredes, el software de estas debe tener la misma semántica que los sistemas de computo convencionales, lo que significa que debe de garantizar una comunicación rápida y confiable. Los datos deben entregarse en el orden de transmisión, sin pérdidas ni duplicaciones.

Los protocolos de transportación ofrecen confiabilidad, lo cual es fundamental para muchas aplicaciones. En el grupo de protocolos TCP/IP, el protocolo de control de transmisión (TCP) es el protocolo de nivel de transportación que ofrece confiabilidad.

2.8.2. Servicio ofrecido por el TCP a las aplicaciones.

Desde el punto de vista de los programas de aplicación, el servicio que ofrece el TCP tiene siete características principales:

- Orientación a conexión
- Comunicación punto a punto.
- Confiabilidad total
- Comunicación duplex integral.
- Interfaz de flujo.
- Arranque confiable de conexión.

El TCP ofrece un servicio de transportación duplex integral completamente confiable sin duplicación ni pérdidas, orientado a conexión, que permite que dos programas de aplicación inicien una conexión, transmitan en ambas direcciones y terminen la conexión. Cada conexión TCP inicia confiablemente y termina sin complicaciones, luego de entregar todos los datos.

2.8.3. Pérdida de paquete y retransmisión.

Para lograr la confiabilidad del TCP se utilizan varias técnicas para manejar partes del problema. Al transmitir el TCP, las pérdidas de paquetes se

compensa con un esquema de retransmisión. Participan ambos lados de la comunicación.

Cuando el TCP recibe datos, devuelve un acuse de recibo (ACK) al transmisor. Cuando los envía, inicia un cronómetro. Si expira antes de llegar el acuse de recibo, expide de nuevo los datos. En la siguiente figura se ilustra la retransmisión.

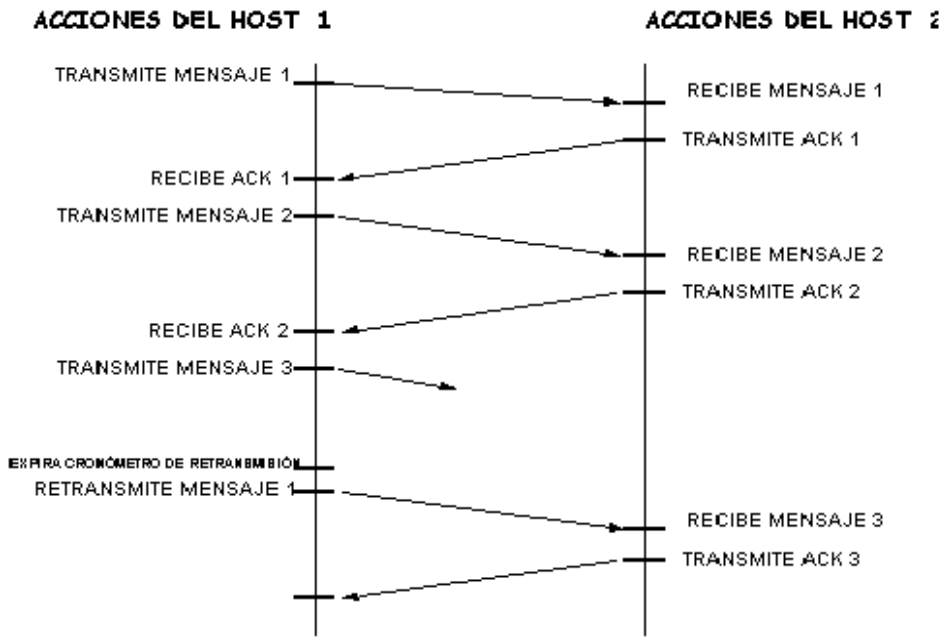

Figura 2.19. Ejemplo de retransmisión. Los elementos de la izquierda corresponden a las acciones del transmisor y los de la derecha a las del receptor. El tiempo discurre hacia abajo. El transmisor retransmite los datos perdidos.

El retardo para que lleguen los datos al destino y regrese un acuse de recibo depende del tráfico en la interred así como la distancia al destino. Dado que el TCP permite que se comunique a la vez varias aplicaciones con varios destinos, y debido a que las condiciones de tráfico influyen en el retardo, el TCP debe manejar retardo que cambian con rapidez.

III. ADQUISICIÓN DE DATOS Y CONTROL

3.1 Introducción

En capítulos anteriores se expuso la forma de adquirir datos de un medio físico cualquiera, en este capítulo encontrará herramientas importantes para introducir esos datos de forma digital hacia el computador, mediante el uso del puerto paralelo. Se encontrarán referencias y librerías para el lenguaje Visual Basic muy útiles, desarrollos esquemáticos para este fin.

En casi todo el mundo las computadoras son dispositivos sumamente útiles, hay un sinnúmero de maneras de comunicarnos con ellas. Una de las tantas maneras de comunicarnos es el puerto paralelo, el cual es sencillo trabajar y nos podemos enganchar a cualquier proyecto que se desee controlar con la

computadora. El truco consiste saber en como trabajan. En nuestro proyecto se presentará un diseño esquemático para la adquisición y emisión de datos mediante el control del puerto paralelo.

En el capitulo 5 se encontrarán las presentaciones del programa desarrollado en Visual Basic para monitoreo y control.

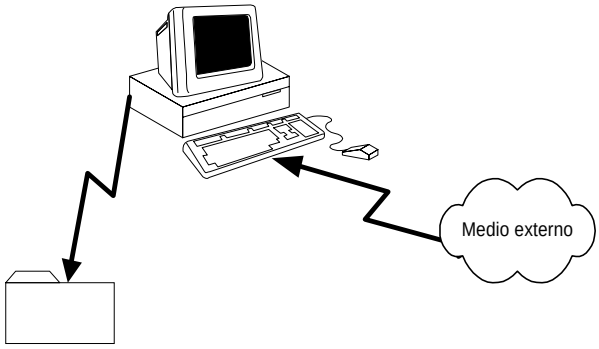

Figura 3.1. Introducción al manejo de puertos

3.2 Introducción al puerto paralelo

El puerto paralelo es el más común de los puertos utilizados para crear proyectos mediante una interfaz (computadora - medio en general). Este puerto es capaz de manejar 9 bits de entradas o pines y 12 bits de salidas, algunos pueden actuar de manera independiente, es decir, pueden utilizarse tanto para adquisición de datos como para su envío. También hay que tener precaución con algunos pines que son de corte a tierra y otros que son de colector abierto.

El puerto está compuesto de 4 líneas de control, 5 de estado y 8 líneas de datos.

Líneas o pines	
2 – 3 – 4 – 5 – 6 – 7 – 8 – 9	**Líneas de** datos
10 – 11 – 12 –13 – 15	**Líneas de** estado
1 – 14 – 16 – 17	**Líneas de** control

Tabla 3.1 Distribución de los pines del puerto paralelo

3.3 Tipos de conectores.

Los puertos paralelos fueron estandarizados en 1994 bajo IEEE1284. Bajo esta estandarización los tipos de conectores son D-type 25 pines y Centronic 36 pines. En la tabla se puede observar las asignaciones del conector de puerto paralelo D-25 con su correspondiente pin al Centronic 36.

La palabra "n" enfrente del nombre de la señal denota que ésta señal es activa en bajo. Por ejemplo: **“n-error”** Si en su impresora ocurre un error, entonces ésta línea es baja. Esta línea es alta mientras la impresora esté en funcionamiento correctamente. La salida del puerto paralelo es normalmente niveles lógicos TTL.

Pin D-Type 25	Pin Centronics	Señal SPP	Dirección In/out	Registro	Inversión de Hardware
1	1	**N** Habilitación	In/Out	Control	Si
2	2	Dato 0	Out	Dato	
3	3	Dato 1	Out	Dato	
4	4	Dato 2	Out	Dato	
5	5	Dato 3	Out	Dato	
6	6	Dato 4	Out	Dato	
7	7	Dato 5	Out	Dato	
8	8	Dato 6	Out	Dato	
9	9	Dato 7	Out	Dato	
10	10	**N** confirmación	In	Estado	
11	11	Ocupado	In	Estado	Si
12	12	Papel vacío	In	Estado	
13	13	Selección	In	Estado	
14	14	Avance Automatic	In/Out	Control	Si
15	32	N error	In	Estado	
16	31	**N** Iniciación	In/Out	Control	
17	36	**N** Selección	In/Out	Control	Si
18 - 25	19-30	Tierra	Gnd		

Tabla 3.2. En la tabla se pueden observar las asignaciones del conector de puerto paralelo D-25 con su correspondiente pin al Centronic 36.

La mayoría de los puertos paralelos de la parte posterior de las computadoras poseen un conector D-25. El conector es normalmente hembra (tiene orificio en lugar de pines) para distinguirlo de los conectores series que son normalmente machos y que también posee la computadora.

3.4 Dirección de puerto paralelo

Las direcciones de puerto paralelo (**LPT1, LPT2** y **LPT3**) cambian con la casa fabricante tanto del BIOS, como de la tarjeta madre. Al decir las notaciones **LPT1**, **LPT2** y **LPT3** son:

- **LPT1**. Puerto paralelo 1 que se utiliza para envío de información del computador al medio.
- **LPT2**. Puerto paralelo 2 que se utiliza para recolección de datos del medio al computador.
- **LPT3**. Puerto paralelo 3 que se utiliza de manera bi-direccional (del computador al medio y viceversa)

El puerto paralelo tiene básicamente 3 direcciones. En general en algunas computadoras las direcciones pre-asignadas en el bios son las siguientes

Address	Notes:
3BCh - 3BFh	Usada para Puerto Paralelo el cual estará incorporada su dirección al uso de tarjetas de vídeo
378h - 37Fh	Usual Dirección para **LPT 1**
278h - 27Fh	Usual Dirección para **LPT 2**

Tabla 3.3 Direcciones del puerto paralelo

LPT1 es normalmente asignada con una dirección base de **378H.**

LPT2 es normalmente asignada con una dirección base de **278H.**

LPT3 es normalmente asignada con una dirección base de **3BCH.**

3.5 ¿Cómo obtener la dirección del puerto paralelo?

1.- En el setup del bios se puede observar las direcciones

2.- En el arranque del sistema de inicio

3.- Software sencillo en C.

3.5.1 En el setup del bios

En el setup del bios se puede observar las siguientes direcciones:

Start Address	Function
0000:0408	**LPT1's Base Address**
0000:040A	**LPT2's Base Address**
0000:040C	**LPT3's Base Address**

Tabla 3.4 Direcciones en el bios.

3.5.2 Software sencillo en C.

```
#include <stdio.h>
#include <dos.h>
void main(void)
{
 unsigned int far *ptraddr;  /* Pointer to location of Port Addresses */
 unsigned int address;       /* Address of Port */
 int a;
 ptraddr=(unsigned int far *)0x00000408;
```

```
for (a = 0; a < 3; a++)
  {
   address = *ptraddr;
   if (address == 0)
            printf("No port found for LPT%d \n",a+1);
   else
            printf("Address assigned to LPT%d is %Xh\n",a+1,address);
   *ptraddr++;
  } }
```

3.6 Configuración de los pines del puerto de impresora

- **LPT1**

Offset	Nombre	Lectura/ escritura	Bit No.	Propiedades	
Base + 0	Puerto de Datos	Escritura	Bit 7	Dato 7	9
			Bit 6	Dato 6	8
			Bit 5	Dato 5	7
			Bit 4	Dato 4	6
			Bit 3	Dato 3	5
			Bit 2	Dato 2	4
			Bit 1	Dato 1	3
			Bit 0	Dato 0	2

Tabla 3.5 pines LPT1.

Los pines 2 - 9 son datos de envío (escritura solamente)

- **LPT2**

Offset	Nombre	Lectura / Escritura	Bit No.	Propiedades	
Base + 1	Puerto de Estado	Lectura	Bit 7	Ocupado	11
			Bit 6	Confirmación	10
			Bit 5	Sin papel	12
			Bit 4	Entrada de Selección	13
			Bit 3	Error	15
			Bit 2	IRQ (Not)	
			Bit 1	Reservado	
			Bit 0	Reservado	

Tabla 3.6 pines LPT2.

Los pines 10 - 11 - 12 - 13 - 15 son los datos de entrada (lectora solamente)

- **LPT3**

Offset	Nombre	Lectura / Escritura	Bit No.	Propiedades	
Base + 2	Puerto de Control	**Lectura / Escritura**	Bit 7	Sin uso	
			Bit 6	Sin uso	
			Bit 5	Habilitación del puerto Bi-Direccional	
			Bit 4	Enable IRQ Via Ack Line	
			Bit 3	Select Printer	17
			Bit 2	Inicializar Impresora (Reset)	16
			Bit 1	Auto-llenado de línea	14
			Bit 0	Habilitación	1

Tabla 3.7 pines LPT3.

Los pines 1 - 14 - 16 - 17 son los datos de escritura o lectura. Los pines tienen una entrada de colector abierto, en esta entrada se debe añadir resistencias de 4,7 KΩ.

Los pines de 18 - 25 son con corte a tierra.

3.7 Formato de bloques de datos para enviar y recibir información de control y monitoreo.

3.7.1 Bloques de datos de **MONITOREO**.

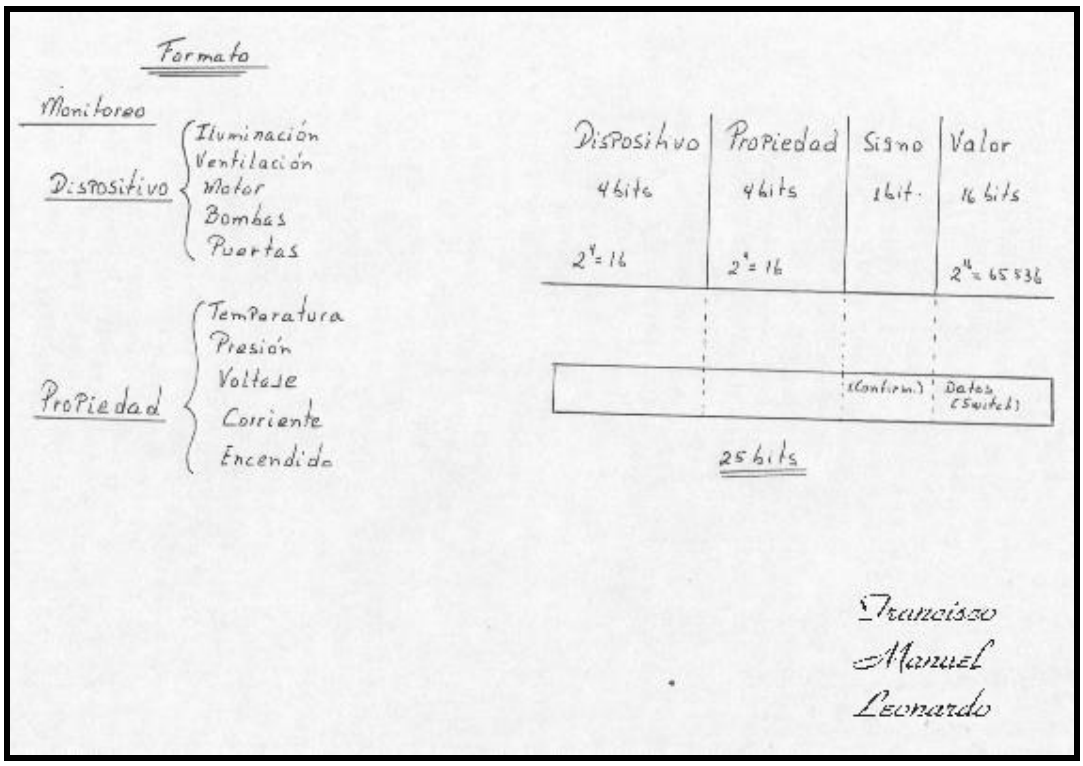

Figura 3.2. Formato de bloques de datos (Monitoreo).

En el formato de monitoreo, para una mejor visualización de los diferentes elementos de la industria se lo ha dividido en ***dispositivo (4 bits), propiedad (4 bits), signo (1 bit) y datos (16 bits).*** Todo esto forma un bloque de 25 bits para enviar y recibir información del funcionamiento de los dispositivos, el bit de signo nos indica la existencia o no de uno de los dispositivos.

El formato tiene una particularidad muy importante de estandarizar los diseños que se hagan y debe permitir sobre todo un crecimiento en la medida que aparezcan nuevos dispositivos a monitorear.

3.7.2 Bloques de datos de **CONTROL**.

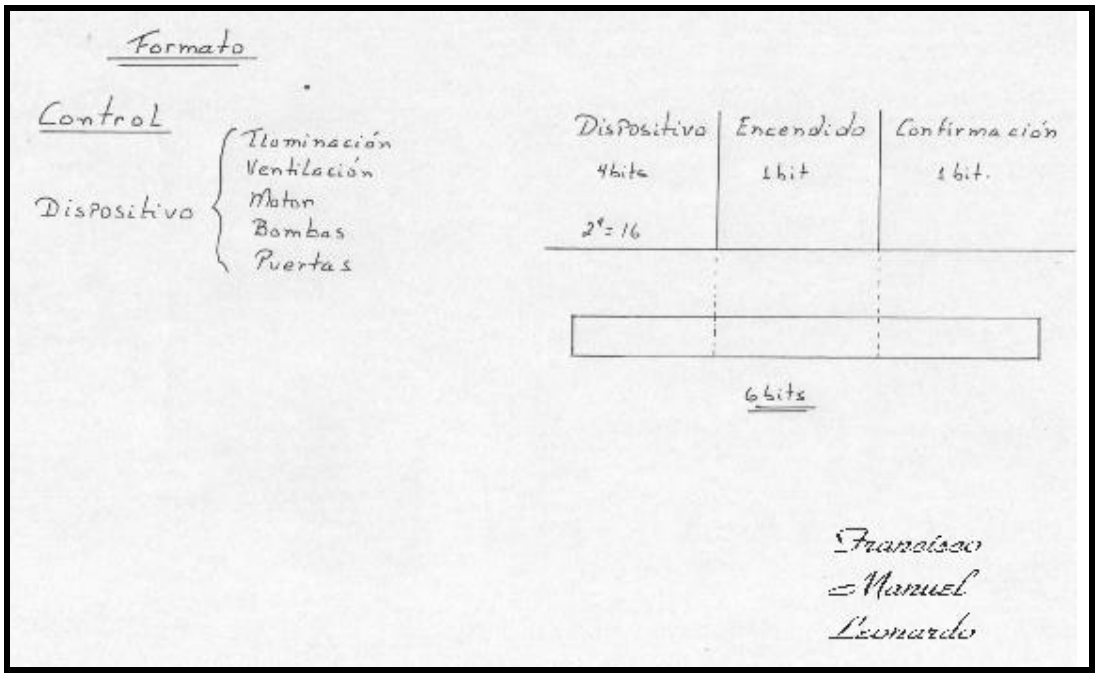

Figura 3.3. Formato de bloques de datos (Control).

Con este bloque ***(6 bits)***, se controlan los ***dispositivos (4 bits)***, con ***acciones (2 bits)*** como por ejemplo el prendido y apagado de motores o luces, pero para esto debe tenerse una estandarización de cómo la interfaz creada podrá saber que acción realizará.

El formato permite un crecimiento en la medida que aparezcan nuevos dispositivos a controlar (hasta $2^4 = 16$).

3.8 Desarrollo esquemático para la recolección y envío de datos mediante el puerto paralelo de la computadora.

3.8.1 Monitoreo (Recolección de datos).

- **LPT1** Se envían 8 Bits señales en alto**.**

- **Borrar registros** Con los 8 bits a través de una puerta NAND se limpian los registros que serán utilizados para **Dispositivo** y **Propiedad**.

- **LPT1** Se envían 8 Bits señales en bajo.

- **LPT1** Se envían datos de **Dispositivo** con los pines 2 - 5 (bits 0 - 3).

- **LPT1** Se genera pulso para cargar registro de **Dispositivo** pin 9 (bit 7) "reloj".

- **LPT1** Se envían datos de **Propiedad** con los pines 2 - 5 (bits 0 - 3).

- **LPT1** Se genera pulso en bajo a través un inversor para cargar registro de **Propiedad** pin 9 (bit 7) "reloj".

- **Borrar registros** Con los 3 bits a través de una puerta NAND los pines 6-7-8 (bits 4-5-6) se limpian los registros que serán utilizados antes de los MUX y se cargan con los valores que registren en el monitoreo de los dispositivos.

- **LPT1** Se envía control para los MUX a fin de obtener los primeros 4 bits con los pines 6 7 8 (bits 4 5 6).

- **LPT2** Se recogen los datos que se generen a nuestra petición de monitoreo a través de un MUX para multiplexar los datos y adecuarlos a los 5 bits que se pueden obtener. El quinto bit servirá para saber si existe o no el dispositivo.

- **LPT1** Se envía control a los MUX para obtener los siguientes 4 bits hasta completar 20 bits, pero solo tendremos 16 bits para nuestro uso.

- **Base de datos** Se procede a guardar en la base de datos los valores recogidos y ver si los valores están dentro de lo permitido.

- **LPT1** Se procede a la recolección de otro **Dispositivo** y **Propiedad** desde el inicio.

3.8.2 Control (Envío de datos).

- **LPT1** Se envían 8 Bits señales en alto.

- **Borrar registros** Con los 8 bits a través de una puerta NAND se limpian los registros que serán utilizados para **Dispositivo** a controlar.

- **LPT1** Se envían 8 Bits señales en bajo.

- **LPT1** Se envían datos de **Dispositivo** con los pines 2 - 5 (bits 0 - 3).

- **LPT1** Se genera pulso para cargar registro de Dispositivo pin 9 (bit 7) "reloj".

- **LPT1** Se genera un bit a través de una puerta AND con los pines 7-8 (bits 6-5) para encender o apagar algún dispositivo que se requiera.

3.9 Librería utilizada para el control del puerto paralelo.

Debido a que Visual Basic no tiene instrucciones de entrada y salida que permita controlar el puerto paralelo, usamos la librería ***Win95io.DLL*** que se tiene que colocar en ***C:\windows\system32***. Esta librería proporciona dos procedimientos y dos funciones:

```
Declare Sub vbOut Lib "WIN95IO.DLL" (ByVal nPort As Integer, ByVal nData As Integer)
Declare Sub vbOutw Lib "WIN95IO.DLL" (ByVal nPort As Integer, ByVal nData As Integer)
Declare Function vbInp Lib "WIN95IO.DLL" (ByVal nPort As Integer) As Integer
Declare Function vbInpw Lib "WIN95IO.DLL" (ByVal nPort As Integer) As Integer
```

De los cuales solo usaremos el procedimiento ***vbOut*** que se utiliza para sacar datos por la dirección de memoria 378 hexadecimal y la función ***vbInp*** para leer los datos por la dirección de memoria 379 hexadecimal.

Esta librería se la puede obtener de manera gratuita en http://www.softcircuits.com proporcionada por ***SoftCircuits Programming***, para lo cual agregamos la descripción que se adjunta con la librería en el Anexo A.

IV. SOFTWARE UTILIZADO (ASP) Y PRESENTACIÓN

4.1 Introducción

En este capítulo se pondrá especial atención al lenguaje de programación utilizado para conectar el sistema en la red de internet (www), se expondrá los inicios, ventajas y desventajas de ASP (Active Server Pages). También se ilustran las presentaciones que fueron desarrolladas y algunos conceptos especiales de carácter genérico.

En este capitulo encontrará ideas interesantes las cuales fueron plasmadas en el programa.

Es de nuestro interés que el lector pueda mejorar la presentación y envió de información a las personas que la requieran en forma rápida.

4.2 Introducción a Active Server Pages

Active Server Pages (ASP) de Microsoft, es un entorno que procesa las secuencias de comandos incorporadas en archivos tipo texto con la extensión .asp que puede combinar texto, páginas HTML, secuencias de comandos y componentes ActiveX para crear páginas y aplicaciones Web interactivas y dinámicas, ejecutadas en el servidor.

Desaparece por tanto el problema de si el cliente puede o no ejecutar sentencias de comandos, el servidor Web solo envía el resultado en código HTML standard interpretable por cualquier explorador.

4.3 Requisitos previos

Es necesario conocer las herramientas más comunes para la creación de páginas Web como FrontPage, Microsoft Visual InterDev, DreamWeaver, etc. Además, como nociones de lenguajes de programación, sobre todo Basic (MS Basic), y otro poco de SQL, para las consultas de base de datos.

4.4 Software necesario para la ejecución de páginas active server.

Para la implantación de un servidor Web que soporte ASP el software necesario es, si lo que estamos configurando es un servidor de alto rendimiento:

- WINDOWS NT 4.0
- IIS 4.0 (INTERNET INFORMATION SERVER 4.0) Ó IIS3.0 + ASP.EXE

Para desarrollo de redes internas los requerimientos son más sencillos:

- WINDOWS 95 + PERSONAL WEB SERVER 1.0 + ASP.EXE
- WINDOWS 98 + PERSONAL WEB SERVER 4.0

Tanto IIS como Personal Web Server pueden descargarse desde la web de Microsoft.

Nota: Personal Web Server 4.0 esta incluido en algunas de las distribuciones de Windows 98 en el directorio ADD-ONS\PWS.

4.5 Conceptos básicos.

4.5.1 Declaración del lenguaje

Como ocurre en otros lenguajes de programación, en ASP existe una sentencia de declaración opcional del lenguaje a utilizar.

<%@ LANGUAGE="VBScript" %>

Esta declaración se pone al principio del archivo, antes de cualquier otra expresión.

ASP, VBScript y Javascript son lenguajes de programación comunes, luego su sintaxis es implementada en código ASCII, por lo que para poder crear, editar y modificar dicho código sólo es necesario utilizar un simple y común editor de textos, como puede ser el "edit" del DOS, el "Notepad" o el "Wordpad" de los entornos Windows, o cualquiera de los múltiples editores de texto que existen en los entornos de sistemas, así como en los Mac.

4.5.2 Bloques de código y comentarios

En páginas ASP, para introducir bloques de sentencias hay que escribir los símbolos reservados:

```
<% {sentencias} %>
```

donde sentencias pueden ser una o varias expresiones del lenguaje, como se muestra en el siguiente ejemplo:

```
<%
Request("param")
Response.Write(Now)
while not cond do
rem do nothing
loop
%>
```

En este punto queremos llamar la atención del lector sobre el hecho de que las sentencias en VBScript no se separan por punto y coma (;).

Los comentarios de código VBScript se especifican mediante la palabra reservada rem o con el carácter comilla simple (') y tienen un ámbito de una línea. Por ejemplo:

```
<%
rem Esto es un comentario
' que ocupa varias.http://www.Topico.com.ec
rem líneas
%>
```

Y este es un comentario mal construido:

```
<%
rem Esto es un comentario
pero es ya no lo es así que el procesador de ASP
lo interpretará como código, y dará error
%>
```

4.5.3 Tipos de datos de VBScript

VBScript sólo tiene un tipo de datos que se llama **Variant**. Un tipo Variant es una clase especial de tipo de datos que puede contener diferentes tipos de información, dependiendo de cómo se utilice. Como Variant es el único tipo de datos en VBScript, también es el único tipo de datos devuelto por todas las funciones en VBScript.

Más allá de las simples clasificaciones numéricas y de cadena, un tipo Variant puede hacer más distinciones acerca de la naturaleza específica de la información numérica. Por ejemplo, puede tener información numérica que representa una fecha o una hora. Cuando se utiliza con datos de fecha u hora, el resultado se expresa siempre como una fecha o una hora. Por supuesto, también puede tener una amplia variedad de información numérica en el intervalo de tamaños de sencillos valores Boolean, a enormes números de signo flotante. Estas categorías de información diferentes que puede contener un tipo Variant se llaman *subtipos*. La mayoría de las veces sólo se puede establecer el tipo de datos que desea en un tipo Variant y se comporta del modo más adecuado para el tipo de datos que contiene.

La tabla siguiente muestra varios subtipos de datos que puede contener un tipo Variant.

Subtipo	Descripción
Empty	**Variant está sin inicializar. El valor es 0 para variables numéricas o una cadena de longitud cero ("") para variables de cadena.**
Null	**Variant contiene intencionadamente datos no válidos.**
Boolean	**Contiene True o False.**
Byte	**Contiene un entero desde 0 hasta 255.**
Integer	**Contiene un entero desde -32.768 hasta 32.767.**
Currency	**Desde -922.337.203.685.477,5808 hasta 922.337.203.685.477,5807.**
Long	**Contiene un entero -desde 2.147.483.648 hasta 2.147.483.647.**
Single	**Contiene un número de signo flotante y precisión simple desde -3,402823E38 hasta -1,401298E-45 para valores negativos y desde 1,401298E-45 hasta 3,402823E38 para valores negativos.**
Double	**Contiene un número de signo flotante y precisión doble desde -1,79769313486232E308 hasta -4,94065645841247E-324 para valores negativos y desde 4,94065645841247E-324 hasta 1,79769313486232E308 para valores positivos.**
Date (Time)	**Contiene un número que representa una fecha entre el 1 de enero de 100 y el 31 de diciembre de 9999.**
String	**Contiene una cadena de longitud variable que puede contener hasta 2 mil millones de caracteres de longitud.**
Object	**Contiene un objeto.**
Error	**Contiene un número de error.**

Tabla 4.1 Subtipos de datos

4.5.4 Variables de VBScript

Una variable es un marcador de posición útil que hace referencia a una ubicación de la memoria del equipo donde puede almacenar información de programa que puede cambiar durante el tiempo en que se ejecuta la

secuencia de comandos. El lugar donde se ubica la variable en la memoria del equipo no es importante. Lo que es importante es que sólo tiene que hacer referencia a su nombre para ver o cambiar su valor. En VBScript, las variables siempre son de un tipo de datos fundamental, **Variant**.

4.5.5 Declaración de variables

Declare variables explícitamente en la secuencia de comandos utilizando la instrucción **Dim**, la instrucción **Public** y la instrucción **Private**. Por ejemplo:

```
Dim GradosFahrenheit
```

Declare múltiples variables separando cada nombre de variable por una coma. Por ejemplo:

```
Dim Superior, Inferior, Izquierda, Derecha
```

También puede declarar una variable implícitamente por el simple uso de su nombre en alguna parte de la secuencia de comandos. Esto no es conveniente, ya que podría escribir mal el nombre de la variable en uno o más lugares provocando resultados inesperados cuando ejecuta el código. Por este motivo, existe la instrucción **Option Explicit** para solicitar una declaración explícita de todas las variables. La instrucción **Option Explicit** debe ser la primera instrucción de la secuencia de comandos.

4.5.6 Restricciones de nombre

Los nombres de variables siguen las normas estándar de denominación en VBScript. Un nombre de variable:

- Debe comenzar con un carácter alfabético.
- No puede contener un punto.
- No debe superar 255 caracteres.
- Debe ser único en el alcance donde se declara.

4.5.7 Asignación de valores a variables

Los valores se asignan a variables creando una expresión de la siguiente forma: la variable está en la parte izquierda de la expresión y el valor que desea asignar a la misma está a la derecha. Por ejemplo: B = 200

4.5.7.1 Escalares y Matrices

Muchas veces, sólo desea asignar un único valor a una variable que ha declarado. Una variable que contiene un único valor es una variable *escalar*. Otras veces, es útil asignar más de un valor relacionado a una única variable. Entonces puede crear una variable que pueda contener una serie de valores. Esto se llama una variable de ***matriz***. Las variables matriz se declaran de la misma forma que las variables escalares. La diferencia es que una

declaración de una variable de matriz utiliza paréntesis a continuación del nombre de la variable.

También puede declarar una matriz cuyo tamaño cambia durante el tiempo en que se ejecuta la secuencia de comandos. Esto se llama una matriz dinámica.. La matriz se declara inicialmente dentro de un procedimiento utilizando la instrucción **Dim** como con cualquier tipo de matriz o la instrucción **ReDim**. La diferencia es que no se establece un tamaño o un número de dimensiones dentro del paréntesis.

No hay límite en cuanto al número de veces que puede cambiar el tamaño de una matriz dinámica, pero debe saber que si hace una matriz menor de lo que era, pierde los datos de los elementos eliminados.

4.5.7.2 Constantes de VBScript

Una *constante* es un nombre significativo que contiene un número o una cadena que nunca cambia. Actualmente VBScript no tiene constantes definidas por el lenguaje. En VBScript, las constantes se implementan como valores *literales* asignados a nombres de variables.

```
Const MiCadena
MiCadena = "Ésta es mi cadena."
Const MiEdad
```

```
MiEdad = 49
```

Observe que el literal de cadena se escribe entre comillas ("). Las comillas son la mejor y la forma más clara de diferenciar valores de cadena y valores numéricos. Los literales de fecha y hora se pueden representar encerrándolos entre signos de número (#). Por ejemplo:

```
Const FechaLímite
FechaLímite = #1-1-96#
```

4.5.8 Procedimientos de VBScript

En VBScript hay dos tipos de procedimientos; el procedimiento **Sub** y el procedimiento **Function**.

4.5.8.1 Procedimientos Sub

Un procedimiento **Sub** es una serie de instrucciones de VBScript, incluidas entre las instrucciones **Sub** y **End Sub** que ejecutan acciones, pero que no devuelven ningún valor. Un procedimiento **Sub** puede tener argumentos (constantes, variables o expresiones que se transfieren por una llamada a un procedimiento). Si un procedimiento **Sub** no tiene argumentos, su instrucción **Sub** debe incluir dos paréntesis vacíos.

4.5.8.2 Procedimientos Function

Un procedimiento **Function** es una serie de instrucciones VBScript incluidas entre las instrucciones **Function** y **End Function**. Un procedimiento **Function** es similar a un procedimiento **Sub**, pero puede devolver un valor. Un procedimiento **Function** puede tener argumentos (constantes, variables o expresiones que se transfieren por una llamada a un procedimiento). Si un procedimiento **Function** no tiene argumentos, su instrucción **Function** debe incluir un par de paréntesis vacíos. Un procedimiento **Function** devuelve un valor asignando un valor a su nombre en una o más instrucciones del procedimiento. El tipo de datos devuelto por un procedimiento **Function** siempre es un tipo **Variant**.

4.5.8.3 Obtención de datos dentro y fuera de Procedimientos

Cada fragmento de datos se transfiere a los procedimientos utilizando un *argumento*. Los argumentos sirven como marcadores de posición para los datos que desea transferir al procedimiento. Cuando crea un procedimiento utilizando la instrucción **Sub** o la instrucción **Function**, se deben incluir paréntesis después del nombre del procedimiento. Los argumentos se colocan dentro de estos paréntesis, separados por comas.

Puede dar a los argumentos cualquier nombre que sea un nombre de variable válido.

Para obtener datos fuera de un procedimiento, debe utilizar un procedimiento **Function**. Recuerde que un procedimiento **Function** puede devolver un valor, mientras que un procedimiento **Sub** no puede hacerlo.

4.5.8.4 Uso de los procedimientos Sub y Function en código

Para utilizar un procedimiento **Function** en el código, siempre se debe utilizar en la parte derecha de una asignación de variable o en una expresión.

Por ejemplo:

```
Temp = Celsius(fGrados)
```

Para llamar a un procedimiento **Sub** desde otro procedimiento, escriba sólo el nombre del procedimiento junto con los valores de los argumentos necesarios, separados por una coma. La instrucción **Call** no se necesita, pero si la utiliza, debe escribir los argumentos entre paréntesis.

El siguiente ejemplo muestra dos llamadas al procedimiento MiProc. Uno utiliza la instrucción **Call** en el código y el otro no. Ambos procedimientos hacen exactamente lo mismo.

```
Call MiProc(primerarg, segundoarg)
MiProc primerarg, segundoarg
```

Observe que se omiten los paréntesis en la llamada cuando no se utiliza la instrucción **Call**.

4.5.9 Modelos de objetos integrados de ASP

Para utilizar la mayoría de los objetos, primero es necesario crear una *instancia* de los mismos. Sin embargo, Active Server Pages (ASP) incluye cinco objetos que no requieren creación de instancias. En la tabla siguiente se resumen estos *objetos integrados*, las tareas en las que se los emplea.

Objeto	Tarea
Request	**Obtener** información de un usuario.
Response	**Enviar** información a un usuario.
Server	**Controlar** el entorno de ejecución de ASP.
Session	**Almacenar** información sobre una sesión de usuario.
Application	**Compartir** información entre los usuarios de una aplicación.

Tabla 4.2 objetos integrados

4.5.10 Sintaxis de los objetos

Los objetos **Request** y **Response** contienen *colecciones*. Una **colección** es un conjunto de datos relacionados a los que se tiene acceso de la misma forma. También puede tener acceso a la información de una colección mediante la instrucción **For...Each** . El acceso a los objetos desde una secuencia de comandos se realiza utilizando *métodos* y *propiedades*.

4.5.11 Uso de los métodos

Un **método** es un procedimiento que actúa sobre un objeto. La sintaxis general es: **Objeto.Método***parámetros*

Donde *parámetros* puede ser una variante, un dato, una cadena o una dirección URL, dependiendo del método.

4.5.12 Uso de las propiedades

Una propiedad es un atributo con nombre de un objeto. Las propiedades definen características del objeto, como su tamaño, color y ubicación en la pantalla; o bien su estado, por ejemplo habilitado o inhabilitado. La sintaxis general es: **Objeto.Propiedad***parámetros*

Donde *parámetros* puede ser un valor, una cadena o un indicador, dependiendo de la propiedad.

4.5.13 Obtener información de formularios HTML

Un *formulario* HTML es el medio utilizado con más frecuencia para obtener información de un usuario de Web. Los cuadros de texto, botones de opción y casillas de verificación de un formulario, mostrado en una página HTML con un explorador, proporcionan al usuario un método sencillo para enviar información. Cuando el usuario hace clic en el botón Enviar, el explorador envía la información recogida al servidor Web.

Puede usar los archivos .asp para recoger o procesar valores de formularios HTML de tres maneras:

- Un archivo .htm estático puede contener un formulario que envíe sus valores a un archivo .asp.
- Un archivo .asp puede crear un formulario que envíe información a otro archivo .asp.
- Un archivo .asp puede crear un formulario que envíe información a sí mismo, es decir, al archivo .asp que contiene el formulario.

Los primeros dos métodos funcionan de la misma forma que los formularios que interactúan con otros programas de puertas de enlace, excepto en que, con ASP, es posible incluir comandos que lean las elecciones del usuario y respondan a ellas.

La creación de un archivo .asp que contenga una definición de un formulario que envía información al propio archivo es un medio ligeramente más complicado, pero muy eficaz, de trabajar con formularios.

4.5.14 Uso del objeto REQUEST (Obtener información de un usuario).

A menudo deseará obtener información *sobre* un usuario, como por ejemplo el tipo de explorador que utiliza. También puede encontrar conveniente conseguir información *que envíe* el usuario, como la que remite en los formularios. El objeto integrado **Request** de ASP hace que conseguir esta información sea muy sencillo.

El objeto **Request** permite tener acceso a cualquier información incluida con una petición HTTP. Esto comprende los siguientes datos:

- Un conjunto de datos estándar incluido en el juego de variables del servidor.
- Un conjunto de parámetros enviados con el método POST.

- Un conjunto de parámetros de consulta adjuntos al método GET.
- *Cookies* enviados desde un explorador. Los cookies permiten asociar un conjunto de información con un usuario.
- Certificados de cliente.

El objeto **Request** tiene asociadas cinco colecciones:

- QueryString
- Form
- Cookies
- ServerVariables
- ClientCertificate

Puede utilizar la siguiente sintaxis general para tener acceso a la información del objeto **Request**:

Request.*NombreColección***(***variable***)**

Donde *NombreColección* puede ser **QueryString, Form, Cookies, ServerVariables o ClientCertificate**, y *variable* es el nombre de la variable de la colección a la que desea tener acceso.

Puede usar la siguiente sintaxis general para tener acceso a las variables del objeto **Request** sin especificar el nombre de la colección:

Request(*nombrevariable***)**

La búsqueda en las colecciones se hace siguiendo este orden: QueryString**,** **Form**, **Cookies**, **ServerVariables**, **ClientCertificate**. Se obtendrá la primera variable que coincida con *nombrevariable*.

Nota: Si es posible que una página HTML tenga más de una variable con el mismo nombre, asegúrese de indicar el nombre de la colección entre **Request** y el nombre de la variable.

4.5.15 Uso de la colección QUERYSTRING

Aunque podría utilizar la variable de servidor QUERY_STRING para procesar la información de QUERY_STRING de una petición de usuario, ASP cuenta con la colección **QueryString**, que facilita el acceso a tal información. Si el método del formulario es ***POST***, la colección **QueryString** contendrá toda la información pasada como parámetro después del signo de interrogación en la dirección URL. Si el método del formulario es ***GET***, la colección **QueryString** contendrá toda la información enviada en el formulario.

Además, la colección **QueryString** se ocupa automáticamente del caso de varias variables con el mismo nombre. Al analizar una cadena de consulta tal como nombre=Andrés&nombre=Martín&nombre=Patricia, por ejemplo, ASP creará una nueva colección llamada nombre que a su vez contendrá tres valores: Andrés, Martín y Patricia. Cada uno de estos valores tiene un índice entero, con el resultado siguiente:

Referencia	Valor
Request.QueryString("nombre")(1)	Andrés
Request.QueryString("nombre")(2)	Martín
Request.QueryString("nombre")(3)	Patricia

Tabla 4.3 Variables QueryString

Las colecciones creadas de esta forma tienen la propiedad **Count**. La propiedad **Count** describe el número de elementos que contiene una colección. En este ejemplo, el valor de Request.QueryString("nombre").Count es 3, ya que hay tres valores distintos almacenados en la colección nombre.

Si utilizase el método **Response.QueryString** para tener acceso a la variable nombre, el resultado sería una cadena delimitada por comas. En el ejemplo anterior, el valor de Request.QueryString("nombre") sería "Andrés, Martín, Patricia".

4.5.16 Uso de la colección FORM

La colección **Form** contiene todos los valores que introdujo un usuario en un formulario remitido con el método POST.

La colección **Form** trata el caso de múltiples parámetros con el mismo nombre de la misma forma en que lo hace la colección **QueryString**.

4.5.17 Uso de la colección SERVERVARIABLES

La colección **ServerVariables** proporciona información de los encabezados HTTP enviados junto con una petición de usuario, así como determinadas variables de entorno del servidor Web. Puede utilizar esta información para proporcionar respuestas personalizadas a los usuarios.

4.5.18 Envío de información al archivo .ASP originario

Con ASP, tiene la flexibilidad de poder definir en un archivo .asp un formulario que envíe los valores introducidos al mismo archivo; es decir, un formulario que envíe valores al archivo .asp que lo contiene. Cuando un usuario rellena y remite los valores de un formulario, puede usar el objeto **Request** para leerlos. Si recibe un valor no válido, puede enviar un mensaje al usuario, indicando el problema y solicitando un valor distinto.

Si la página enviada al usuario sólo contiene un mensaje, el usuario deberá volver a la página del formulario. Puede evitarle este paso enviando el mensaje y definiendo el formulario de nuevo.

Sin embargo, si envía los mensajes introducidos en el formulario al archivo que lo definió, podrá enviar mensajes informativos junto con el contenido del formulario y de este modo sólo tendrá que definirlo una vez.

4.5.19 Uso del objeto RESPONSE

El objeto integrado **Response** de ASP sirve para controlar la información enviada a un usuario, sirviéndose de los métodos:

- **Response.Write** para enviar información directamente a un explorador.
- **Response.Redirect** para dirigir al usuario a una dirección URL distinta de la solicitada.
- **Response.ContentType** para controlar el tipo del contenido enviado.
- **Response.Cookies** para establecer valores de cookie.
- **Response.Buffer** para almacenar la información en un búfer.

4.5.20 Envío de texto a un usuario

Write es el método más utilizado del objeto **Response**. Puede usar **Write** para enviar información a un usuario desde dentro de delimitadores ASP.

Response.Write *variante*

Donde *variante* puede ser cualquier tipo de datos que admita el lenguaje de secuencia de comandos principal predeterminado.

Ejemplo:

```
<%response.write "<center>Hola mundo</center>" %>
```

obtenemos: Hola mundo

El método **Response.Write** es especialmente útil cuando se desea devolver información al usuario desde dentro de un procedimiento.

No es necesario utilizar **Response.Write** para enviar contenido a un usuario. El contenido no comprendido entre delimitadores se envía directamente al explorador, el cual le da formato y lo muestra como corresponda.

4.5.21 Redirigir un usuario a otra dirección URL

En lugar de enviar un contenido a un usuario, puede remitirlo a otra dirección URL con el método **Redirect**.

Response.Redirect *dirección URL*

Por ejemplo, si desea asegurarse de que los usuarios entren en su aplicación desde una página determinada, puede comprobar si han estado en ella y enviarles allí en caso contrario.

Ejemplo: <%response.redirect "www.renfe.es"%>

El navegador se dirigirá a la URL especificada

4.5.22 Establecer el tipo de contenido HTTP

Puede usar la propiedad **ContentType** del objeto **Response** para establecer la cadena de tipo de contenido HTTP para el contenido enviado al usuario.

La sintaxis general es: **Response.ContentType** = *TipoContenido* donde ***TipoContenido*** es una cadena que describe el tipo del contenido. Si desea una lista completa de los tipos admitidos, consulte la documentación del explorador de Web o la especificación HTTP vigente.

Por ejemplo, si desea enviar el origen (es decir, el archivo .asp a partir del cual se genera una página ASP) a un explorador, establezca "**text/plain"** en ContentType**.**

El explorador mostrará entonces la página como texto, en lugar de interpretarla como HTML.

4.5.23 Almacenar la respuesta en búfer

La configuración predeterminada es no almacenar en búfer todas las páginas ASP. Sin embargo, puede establecer **True** en la propiedad **Buffer** del objeto **Response** para procesar toda la secuencia de comandos de una página antes de enviar nada al usuario: **<% Response.Buffer = True %>**

Puede usar el almacenamiento en búfer para determinar en algún momento del procesamiento de una página que no desea enviar el contenido anterior al usuario. En lugar de ello, puede redirigir el usuario a otra página con el método **Redirect** del objeto **Response**, o borrar el búfer con el método **Clear** de **Response** y enviar un contenido distinto al usuario. En el ejemplo siguiente se utilizan estos dos métodos.

De forma predeterminada, el almacenamiento en búfer está desactivado para todas las páginas ASP de todas las aplicaciones, debido al valor 0 en la clave **BufferingOn** del registro.

4.5.24 Trabajo con componentes Activex Server

Los componentes ActiveX Server, anteriormente conocidos como servidores de Automatización, están diseñados para ejecutarse en un servidor Web como parte de las aplicaciones Web. Los componentes contienen funciones dinámicas comunes, como por ejemplo el acceso a bases de datos, para que usted no tenga que realizarlas una y otra vez.

Normalmente, los componentes se invocan desde archivos .asp. Sin embargo, también puede llamarlos desde otros orígenes, como por ejemplo aplicaciones ISAPI, desde otro componente de servidor, o desde otros lenguajes OLE compatibles.

Active Server Pages (ASP) incluye cinco componentes ActiveX Server:

- **Componente Ad Rotator (Adrotator)** Gira automáticamente los anuncios mostrados en una página de acuerdo con una programación especificada.

- **Componente Capacidades del explorador (*FileSystemObject*)** Determina la capacidad, el tipo y la versión de cada explorador que tiene acceso a su sitio Web.
- **Componente Acceso a archivos** Proporciona acceso a la entrada y salida de un archivo
- **Componente Vínculo de contenidos** Crea tablas de contenido para las páginas Web y las vincula secuencialmente como las páginas de un libro.
- **Componente Acceso a bases de datos** Proporciona acceso a la entrada y salida de un archivo

4.5.25 Crear una instancia de un componente

Puede crear una instancia de un componente ActiveX Server con una sola instrucción. Una vez creada la instancia, puede usar los métodos asociados al componente o establecer y leer sus propiedades.

En la secuencia de comandos siguiente se usa el método **Server.CreateObject** para crear una instancia del componente Funciones del explorador y luego la asigna a la variable bc:

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
```

También puede usar la etiqueta <OBJECT> para crear una instancia de un componente. En este ejemplo se crea una instancia del componente Ad Rotator:

```
<OBJECT RUNAT=Server ID=MiAn PROGID="MSWC.AdRotator">
</OBJECT>
```

4.5.25.1 Componente AdRotator

El componente ADRotator automatiza la rotación de imágenes de anuncio en una página Web. Cada vez que un cliente abre o recarga la página este componente presenta una nueva imagen según las definiciones especificadas en un archivo.

Archivos necesarios:

Archivo Rotator Schedule: es un archivo de texto que contiene la agenda de presentación de los anuncios.

Archivo de redirección: es un archivo .asp que implementa la redirección a la URL anunciante.

CREACIÓN DEL OBJETO AdRotator

<% Set Rotacion=Server.CreateObject("MSWC.AdRotator") %>

Propiedades:

Border

Permite especificar si los anuncios se presentan enmarcados.

<% objeto.border=tamaño %>

Clickable

Permite especificar si los anuncios se presentan como hipervinculos.

<% objeto.clickable= True o False %>

TargetFrame

Permite especificar el marco de destino del hipervinculo.

<% objeto. TargetFrame= nombre del marco destino %>

Metodos:

GetAdvertisement

Recupera el siguiente anuncio del archivo Schedule.

Sintaxis: Objeto.GetAdvertisement (url del archivo Shedule)

4.5.26 Componente Acceso a archivos (FileSystemObject)

El componente FSO nos permite abrir y crear archivos de texto en el servidor.

Este componente consta de 22 métodos, de los cuales podemos seleccionar 2 que son los que nos van a permitir leer o escribir en archivos de texto existentes en el servidor o crear dichos archivos.

Para crear un objeto FSO la sintaxis es la misma que para cualquier otro componente ActiveX

```
<% Set MiFSO=Server.CreateObject("Scripting.FileSystemObject") %>
```

Cuando abrimos o creamos un archivo de texto mediante FSO este nos devuelve una instancia del objeto **TextStream** que es la que representa el archivo físico y con la cual trabajaremos.

Metodos:

CreateTextFile

Crea un archivo físico y devuelve la instancia de TextStream con la cual trabajaremos.

Sintaxis:

<% Set MiArchivo=MiFSO.CreateTextFile("Nombre Archivo",Sobreescribir") %>

Nombre archivo Nombre del archivo a crear.

Sobreescribir

Admite los valores TRUE o FALSE, si el archivo ya existe y el valor dado es TRUE se crea de nuevo, si no , devuelve un error.

OpenTextFile

Abre un archivo físico y devuelve la instancia de TextStream con la cual trabajaremos.

Sintaxis

<% Set MiArchivo=MiFSO.OpenTextFile("Nombre Archivo",modo,crear") %>

Nombre Archivo Nombre del archivo a abrir.

Modo Indica si queremos abrir el archivo para lectura (1) , para escritura (2) o para escribir nuevos registros al final del archivo(8)

Crear Admite los valores TRUE o FALSE, si el archivo no existe y el valor dado es TRUE se crea.

Ejemplo: Apertura de archivo para lectura:

```
<% Set MiArchivo=MiFSO.OpenTextFile("c:\Archivo_nuevo.txt",1,true") %>
```

4.5.27 Componente acceso a bases de datos

El componente **Acceso a bases de datos** utiliza ADO (Objetos de datos ActiveX) para tener acceso a la información almacenada en una base de datos o en otra estructura tabular de datos.

4.5.27.1 Definiendo el nombre de la fuente de datos para conectividad de bases de datos abiertas (DSN ODBC)

- La administración de orígenes de datos **ODBC** (Open Database Connectivity) es una utilidad general de Windows y Windows NT.
- Permite que las aplicaciones accedan a los datos a través usando SQL como lenguaje estándar.
- Se administran a través de la ventana **ODBC (NT)** o **ODBC de 32 bits** (Windows 9x) del *Panel de Control.*
- Se pueden configurar tres diferente fuentes de datos ODBC, la forma más interesante es la de **DSN del Sistema**, que presenta la ventaja de poder ser accedida por cualquier usuario, siendo el tipo usado en aplicaciones **ASP**.

4.5.27.2 Configuración de una fuente de datos

En este ***ejemplo*** declaramos una base de datos Access, el procedimiento es prácticamente igual para cualquier otra tecnología de bases de datos.

Para declarar una base de datos **ODBC** haremos doble click en el icono **Fuentes de Datos ODBC** que encontraremos en el panel de control de Windows (**Inicio->Configuración->Panel de Control**), una vez abierto el programa nos situaremos en la pestaña **DSN de Sistema**

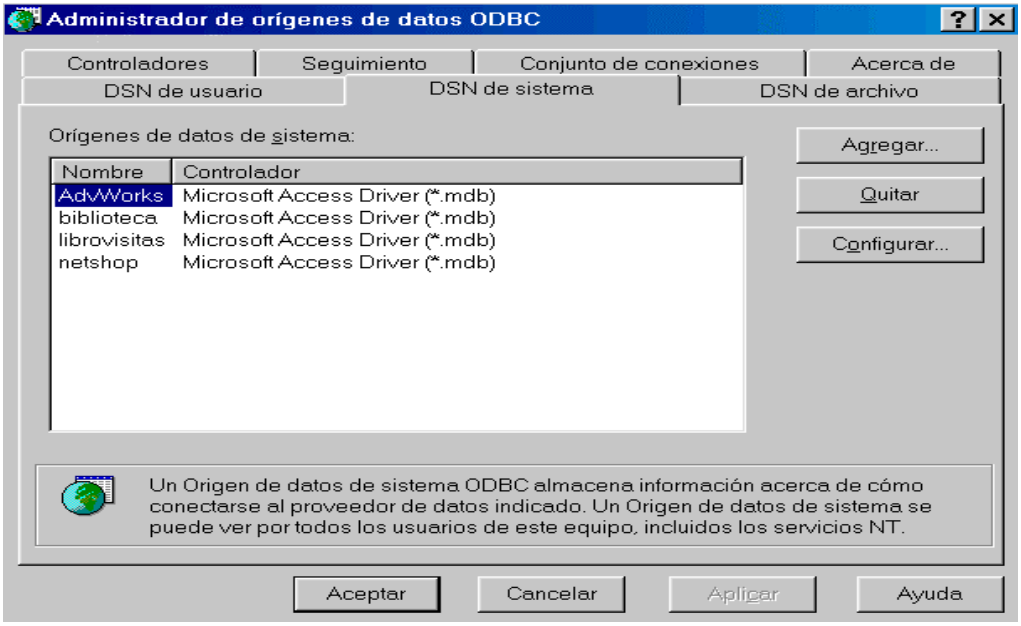

Figura 4.1 Fuentes de datos ODBC (DSN Sistema)

Pulsaremos el botón **AGREGAR** para añadir una nueva fuente de datos, lo que nos llevara a la pantalla de controladores.

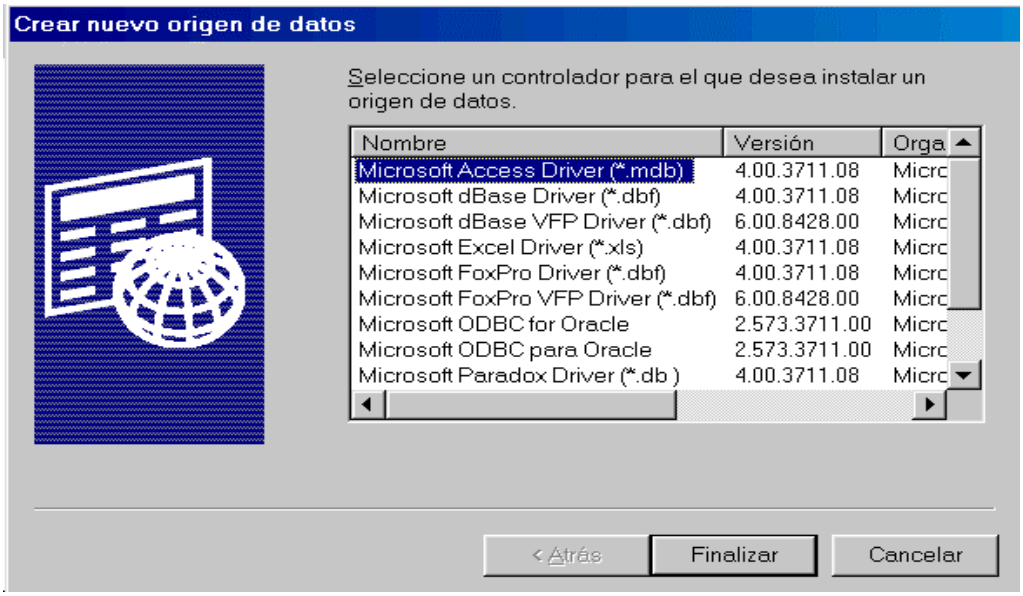

Figura 4.2 Controladores de Bases de datos.

En esta ventana elegiremos el driver adecuado para la tecnología de la base de datos que queremos tratar desde nuestras páginas ASP y pulsaremos finalizar. La siguiente pantalla que se nos muestra nos pedirá la ubicación física de nuestra base de datos y el nombre **ODBC** con el que queremos acceder a ella.

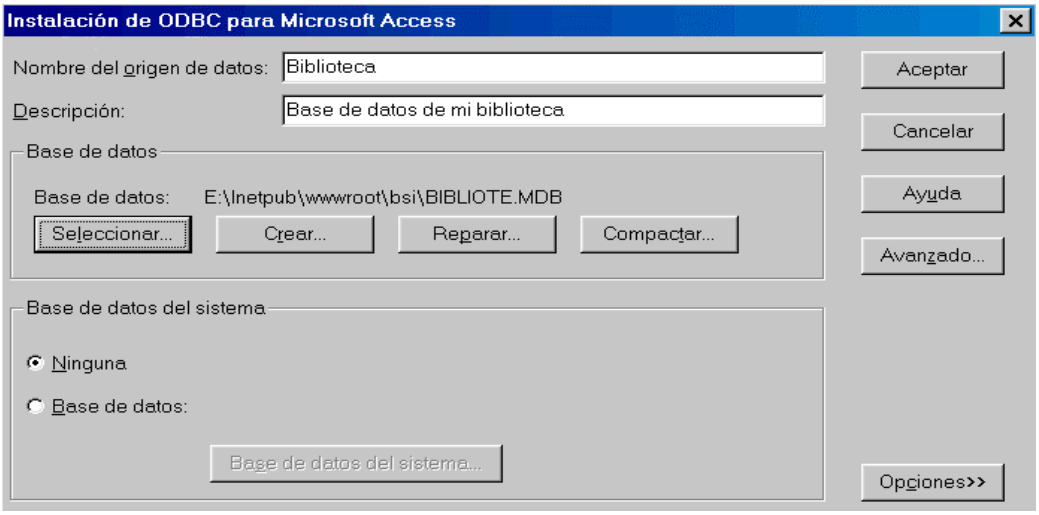

Figura 4.3 Instalación ODBC

- **Nombre de origen de datos:** Es el nombre simbólico de nuestra base de datos, será el que utilicemos desde ASP para referirnos a ella.
- **Descripción:** Es una descripción de la base de datos
- **Base de datos:** Es el camino físico de la base de datos, para establecerlo pulsaremos el botón **SELECCIONAR** y mediante un explorador elegiremos nuestra fuente de datos.

Una vez hecho esto pulsaremos el botón **ACEPTAR** y ya tendremos nuestra base de datos disponible para su uso desde nuestras aplicaciones Web.

NOTA: Si nuestro sistema operativo es NT y declaramos una base de datos ACCESS debemos de tener en cuenta que ACCESS crea en el mismo directorio de la base de datos un archivo con extensión .ldb cuando cualquier

usuario interactúa con la base de datos. Ello nos obliga a dar derechos suficientes al usuario de IIS (IUSR_Nombre del equipo) para manipular el directorio de la base de datos.

4.5.27.3 Activex Data Objects (ADO)

Una de las características más interesante de ASP es su facilidad para el manejo de bases de Datos que residen en el servidor. Esto lo conseguimos mediante el uso de **ADO (ActiveX Data Object)** de una forma fácil, rápida y con un mínimo consumo de recursos del sistema.

ADO usa **ODBC** para el acceso a bases de datos. lo que nos independiza de la tecnología de las mismas; esto implica que podemos cambiar la tecnología de la base de datos y si mantenemos la misma estructura de datos, nuestras aplicaciones desarrolladas con **ADO** pueden seguir funcionando sin cambiar ni una sola línea de código.

Para desarrollo podemos crear nuestras fuentes de datos en Microsoft Access, pero en entornos de producción con gran afluencia de clientes deberemos de usar gestores de bases de datos más potentes, como **Oracle, Microsoft Sql Server, etc**.

ADO esta formado por varios objetos organizados de forma jerárquica (cada uno de ellos con sus métodos y propiedades específicos).

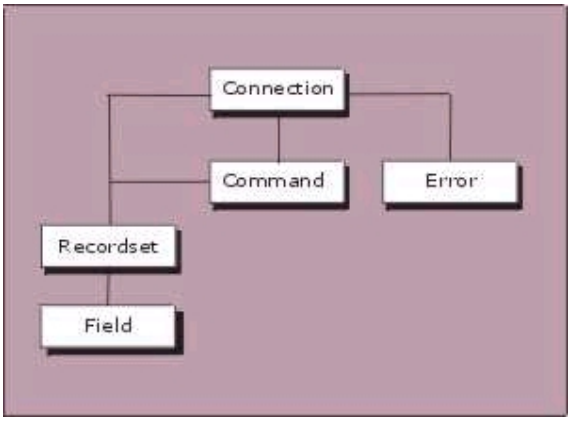

Figura 4.4 ADO (ActiveX Data Object)

4.5.28 Objeto: CONNECTION

Proporciona una conexión a una base de datos **ODBC** desde una página ASP. Esta conexión nos permitirá efectuar las operaciones que deseemos sobre la base de datos.

Es el objeto primario de ADO, ninguno de los otros objetos puede existir si este no es declarado de forma explicita o implícita.

La conexión terminará cuando nosotros la cerremos explícitamente con el método Close o bien cuando termine la ejecución de la página ASP.

Para establecer la conexión lo primero que hacemos es crear el Objeto Connection por medio de la propiedad **CreateObject de objeto Server:**

```
<% Set conexion=Server.CreateObject("ADODB.Connection")%>
```

Una vez establecida la instancia del objeto pasamos a configurarlo mediante sus distintas propiedades y métodos.

4.5.28.1 Propiedades del Objeto CONNECTION:

- ***ConnectionString***

  Especifica la referencia a la base de datos con la cual queremos conectar, conteniendo en una cadena de texto la información necesaria para efectuar esa conexión mediante parejas de valores separadas por ";".

  Los valores que podemos asignar son:

Data Source:	**DSN**=Nombre ODBC de la Base de Datos
Usuario:	**User**=Nombre de Usuario
Password:	**Password**=Password del usuario para la base de datos

Ejemplo:

```
<%
conexion.ConnectionString="DSN=MIOdbc;User=pepe;Password=1234"
%>
```

- ***Mode***

  Especifica los permisos de la conexión.

  Algunos de los valores más habituales que podemos asignar son:

1	Establece permiso solo de **Lectura**
2	Establece permiso solo de **Escritura**
3	Establece permiso de **Lectura/Escritura**

Ejemplo:

<% conexion.Mode=3 %>

4.5.28.2 Métodos del objeto Connection:

- ***Close***

  Cierra el objeto

Ejemplo:

<% conexion.close %>

- ***Execute***

  Ejecuta una sentencia SQL contra la base de datos.

Ejemplo:

<% Set resultado=conexion.execute (Select * from amigos) %>

- ***Open***

  Abre la conexión con los parámetros especificados en las propiedades (arrriba).

Ejemplo:

<% conexion.open %>

4.5.29 Objeto: ERROR

El objeto Error contiene la colección **Errors**, que es la encargada de almacenar los errores que se pudieran producir durante la ejecución de operaciones contra Bases de Datos.

4.5.29.1 Propiedades del Objeto ERROR:

- ***Description***

  Descripción del error.

- ***Number***

  El numero de error.

4.29.2 Métodos del Objeto Error:

- ***Clear***

  Elimina los datos del objeto Error.

Ejemplo:

Examinando los posibles datos de la colección **Errors**

```
Miconexion.open
If Miconexion.Errors.Count > 0 then
For each error in Miconexion.errors then
Response.write Error.Number & " = "& Error.Description
next
End if
```

Nota: **Count** es una propiedad de la colección **Errors**.

4.5.30 Objeto: RECORDSET

El objeto **Recordset** es el interface entre los datos obtenidos de nuestras consultas sobre las tablas y nuestras páginas asp. Representa una tabla organizada en filas (registros) y columnas (campos).

4.5.30.1 Definición del tipo de Cursor

Entendemos como cursor el puntero que nos permite desplazarnos por los registros del recordset. Dependiendo del tipo elegido determinaremos los desplazamientos y cambios realizables en los datos.

El tipo de cursor lo definiremos mediante la propiedad ***CursorType***, los posibles valores son:

Denominación	valor	Características
adOpenForwardOnly	**0**	Es el cursor por defecto, solo nos permite recorrer la tabla de forma secuencial (no se puede volver hacia atrás) y **no permite modificaciones** en los registros. Por el contrario es el de menos consumo. No vemos los cambios realizados en la tabla por otro recordset**.**
adOpenKeyset	**1**	Nos permite movernos en los dos sentidos, **si permite modificaciones** en los registros. Vemos los cambios realizados en la tabla por otro recordset a excepción de las nuevas altas.
adOpenDynamic	**2**	Nos permite movernos en los dos sentidos, **si permite modificaciones** en los registros. Vemos Todos los cambios realizados en la tabla por otro recordset.
adOpenStatic	**3**	Nos permite movernos en los dos sentidos, **no permite modificaciones** en los registros. No vemos los cambios realizados en la tabla por otro recordset.

Tabla 4.4 propiedad CursorType

4.5.30.2 Definición del tipo de Cerrojo

Entendemos como cerrojo el tipo de bloqueo que efectuaremos en la base de datos cuando modifiquemos un **recordset**, a fin de evitar que dos o más usuarios accedan a modificar un mismo registro a la vez.

El tipo de cerrojo lo definiremos mediante la propiedad ***LockType***, los posibles valores son:

Denominación	valor	Características
adLockReadOnly	**1**	Es el defecto; no permite al usuario modificar los datos de la tabla.
dLockPessimistic	**2**	Cuando se abra la tabla nadie mas podrá hacerlo, este modo nos asegura la plena integridad de los datos.
adLockOptimistic	**3**	Cierra la tabla a los demás usuarios cuando se invoque al método Update del objeto recordset; de este modo la Base de datos quedará bloqueada menos tiempo que con el método anterior.

Tabla 4.5 propiedad ***LockType***

4.5.30.3 Moviéndose por los datos del recordset

- **Métodos usados:**

Método	Características
Move *Num_registros*	**Mueve el cursor *Num_registros* hacia abajo si es positivo y hacia arriba si es negativo**
MoveFirst	**Mueve el cursor al primer registro del Recordset**
MoveLast	**Mueve el cursor al ultimo registro del Recordset**
MoveNext	**Mueve el cursor un registro hacia delante**
MovePrevious	**Mueve el cursor un registro hacia atrás**

Tabla 4.6 Metodos recorset

- **Propiedades usadas:**

Propiedades	Características
PageSize	**Establece el número de registros por página del recordset**
	rs.Pagesize=10
AbsolutePage	**Mueve el cursor al primer registro de dicha página (es necesario definir anteriormente el pageSize)**
	rs.AbsolutePage=2
PageCount	**Contiene el numero de páginas del recordset, tomando como base PageSize**
	xx=rs.PageCount
Absoluteposition	**Mueve el cursor al num_registro especificado**
	rs.Absoluteposition=17
RecordCount	**Contiene el numero de registros del recordset; Nota: No funciona con el cursor** adOpenForwardOnly
	xx=rs.recordcount
BOF	**Toma el valor True cuando estamos en el primer registro del recordset**
EOF	**Toma el valor True cuando estamos en el ultimo registro del recordset**

Tabla 4.7 Propiedades recorset

- **Métodos usados para modificar los *datos***

Método	Características
AddNew	**Abre un nuevo registro en el recordset para introducir datos**
	rs.Addnew **rs("codigo")=1234** **rs("titulo")="Todo sobre ASP"** **rs.Update**
Delete	**Elimina el registro actual**
Update	**Actualiza un registro del recordset tras haberlo modificado**
	rs("titulo")="Como hacerse rico en 10 minutos" **rs.Update**

Tabla 4.8 Métodos usados **para modificar los *datos***

- **Métodos usados para *Abrir y cerrar el recordset***

Método	Características
Open *Sql, conexion*	**Abre el recordset y almacena en el resultado de *sql* contra la *conexion***
	set rs=createobject("ADODB.Recordset") **rs.CursorType = 1** **rs.LockType = 3** **Sqltxt="SELECT * FROM libros"** **rs.open Sqltxt, "DSN=Biblioteca"**
Close	**Cierra el recordset**

Tabla 4.9 Métodos usados para *Abrir y cerrar el recordset*

4.5.31 Objeto: COMMAND

Representa un comando SQL que se ejecuta contra la base de datos declarada en el objeto **Connection**.

Si el resultado de ese comando es un conjunto de datos, estos se almacenaran en un objeto de tipo **Recordset**.

4.5.32 Objeto: FIELD

El objeto **Field** representa la información relativa a un campo de un **Recordset**.

Contiene la colección **Fields** que representa todos los campos de la tabla, cada miembro de esa colección es un objeto de tipo Field.

4.5.33 Objeto SERVER

El objeto **Server** nos proporciona acceso a métodos y propiedades del servidor.

4.5.33.1 Propiedades del Objeto SERVER:

- **ScriptTimeout**

Especifica la cantidad máxima de tiempo que puede tardar la ejecución de una secuencia de comandos (Tiempo máximo que puede tardar en ejecutarse una página dada).

Sintaxis

Server.ScriptTimeout= nº de segundos

4.5.33.2 Métodos del Objeto SERVER:

- **CreateObject**

Crea una instancia de un componente **ActiveX** en el servidor.

Sintaxis

Server.CreateObject (**IdProg**)

IdProg es el identificativo del tipo de componente que queremos crear, nos viene suministrado por el fabricante del componente.

4.5.33 Objeto APPLICATION

El objeto **Application** se utiliza para compartir información entre todos los usuarios de una aplicación (entendemos por una aplicación ASP todos los archivos .asp de un directorio virtual y sus subdirectorios. Como varios usuarios pueden compartir un objeto **Application**, existen los métodos **Lock** y **Unlock** para asegurar la integridad del mismo (varios usuarios no puedan modificar una misma propiedad al mismo tiempo).

4.5.34.1 Creación de una variable en APPLICATION

Sintaxis

Application("Nomvariable")= valor

LOCK.- El método **Lock** asegura que sólo un cliente puede modificar o tener acceso a las variables de **Application** al mismo tiempo.

Sintaxis

Application.Lock

UNLOCK.- El método **Unlock** desbloquea el objeto **Application** para que pueda ser modificado por otro cliente después de haberse bloqueado mediante el método **Lock**. Si no se llama a este método de forma explícita, el servidor Web desbloquea el objeto **Application** cuando el archivo .asp termina o transcurre su tiempo de espera.

Sintaxis

Application.Unlock

4.5.35 Objeto SESSION

El objeto **Session** permite almacenar la información necesaria par una sesión de usuario contra nuestra aplicación ASP. Las variables que almacenemos en el objeto **Session** no se pierden al cambiar de página, si no que se mantienen hasta que el cliente sea eliminado por el servidor.

Las variables de **Session** de un cliente solo pueden ser accedidas por ese cliente.

El servidor crea automáticamente el objeto Session cuando un usuario que no tenga actualmente una sesión solicita una pagina Web de la aplicación.

(***Nota****: el servidor elimina un cliente bien cuando desde una pagina ASP se invoca el método* ***Abandon*** *o bien cuando este cliente lleva 20 minutos sin actividad en nuestra aplicación*).

4.5.35.1 Creación de una variable en SESSION

Metodos:

Sintaxis

Sesion("Nomvariable")= valor

Abandon

Destruye todos los objetos y variables almacenados en el objeto **Session.**

4.5.36 Consulta Sencilla utilizando una conexión ODBC

Este ejemplo nos permite listar el resultado de una consulta de selección **"*select*"** elaborada desde un formulario. Suponemos que tenemos declarada en **odbc** una base de datos bajo la DSN de Sistema **biblioteca**.

Esta base de datos contiene una tabla llamada **libros** con los campos **Titulo** y **Autor.**

El formulario nos presenta un formato para escribir el titulo que buscamos, este titulo se lo pasamos como parámetro a nuestra pagina .asp que nos mostrara todos los títulos coincidentes.

Fconsu.htm

```
<html>
<head>
<title>Formulario para consulta sencilla</title>
</head>
<body>
<h3 align="center">Ejemplo de consulta sencilla</h3>
<p> </p>
<form method="POST" action="consultasencilla.asp">
<table border="1" width="100%">
<tr>
<td width="16%">Titulo:</td>
<td width="84%"><input type="text" name="titulo" size="55"></td>
</tr>
<tr>
<td width="16%"><input type="submit" value="Enviar" name="B1"></td>
<td width="84%"><input type="reset" value="Restablecer" name="B2"></td>
</tr>
</table>
</form>
```

Ahora desarrollamos la pagina ASP que recibirá los parámetros enviados por el formulario y asi poder formar la consulta de selección.

Consultasencilla.asp

```
<html>
<head>
<title>Ejemplo de consulta sencilla</title>
</head><body bgcolor="#808000">
<div align="center">
<center>
<table border="1" width="62%" bgcolor="#008000">
<tr>
<td width="100%">
<p align="center"><font color="#FFFFFF">Resultados de su consulta</font></td>
</tr>
</table>
</center>
</div>
<%ctitulo=request.form("titulo")
set rs=createobject("ADODB.Recordset")
sqltxt="select titulo,autor from libros where titulo like '%"&ctitulo&"%'"
rs.open sqltxt,"DSN=biblioteca" %>
<%if rs.eof then%>
<h3 align="center">
<font color="#FF0000">No hay datos que coincidan con su petición </font></h3>
<%else%>
<table border="0" width="100%">
```

```
<tr>
<th width="33%" valign="middle" align="center" bgcolor="#008080"><font
color="#000000">Titulo</font></th>
<th width="33%" valign="middle" align="center" bgcolor="#008080"><font
color="#000000">Autor</font></th>
</tr>
<%do while not rs.eof%>
<tr>
<td width="33%" valign="middle" align="center" bgcolor="#FFFFFF"><font
color="#000000"><%=rs("Titulo")%></font></td>
<td width="33%" valign="middle" align="center" bgcolor="#FFFFFF"><font
color="#000000"><%=rs("Autor")%></font></td>
</tr>
<%rs.movenext
loop
rs.close
Set rs=Nothing
end if%>
</table>
<p> </p>
<p align="center"><a href="fconsu.htm">Otra vez</a></p>
</body></html>
```

V. MANUAL DEL USUARIO.

5.1 Introducción.

Este manual del usuario contiene información del manejo del software, procedimientos de configuración inicial, operación básica y uso de opciones.

Además, encontrará los requerimientos tanto para el computador controlador (Servidor) del sistema como para los usuarios a través de la red (Clientes).

5.2 Software de monitoreo y control.

Esta sección se la hace para fines de ayuda y un correcto funcionamiento del software desarrollado para aplicaciones especificas.

5.2.1 Requerimientos del sistema (Servidor).

Entre los **requerimientos del sistema,** se requerirá establecer una conexión dedicada y tener una dirección de Internet a fin de poder mantener una actualización de datos permanente.

Software:

- **Sistema Operativo:** Windows 95 o 98.
- **Internet Explorer 4.0 o superior**, se incluye en algunas versiones de Windows.
- **Personal Web Server 4.0**, está incluido en algunas distribuciones de Windows 98 en el directorio ADD-ONS\PWS.
- **Librería Win95io.dll**, la cual es de distribución gratuita y se la puede obtener de la dirección **www.softcircuits.com** .

Hardware (se recomienda):

- Dispositivo de adquisición de datos y control modelo **MLF 2.0**
- Computador con microprocesador 586 o superior (Pentium).
- Velocidad de 166 MHZ o superior.
- 32 Mb de Memoria RAM (mínimo)
- Tarjeta de red
- Disco duro de 10 Gb.

- **Cable FML** para conexión de la interfaz computador - dispositivo MLF 2.0

5.2.2 Configuración del sistema (Cliente).

Para el cliente los requerimientos son mínimos.

Software:

- **Sistema Operativo:** Windows 95 o 98.
- **Internet Explorer 4.0 o superior**, se incluye en algunas versiones de Windows.
    - Instalación típica:

      Requiere para instalar: 70 MB

      Requiere para ejecutarse: 55 MB después restaurar el sistema.
    - Instalación completa:

      Requiere para instalar: 111 MB

      Requiere para ejecutarse: 80 MB después restaurar el sistema.

Hardware:

Se requiere:

- Computador con microprocesador 586 o superior (Pentium).
- Velocidad de 166 MHZ o superior.
- Tarjeta de red.
- Acceso a Internet.

5.3 Configuración del sistema (Servidor).

5.3.1 Configuración de PWS (Personal Web Server).

Una vez instalado el PWS lo ejecutamos y aparecerá la siguiente pantalla.

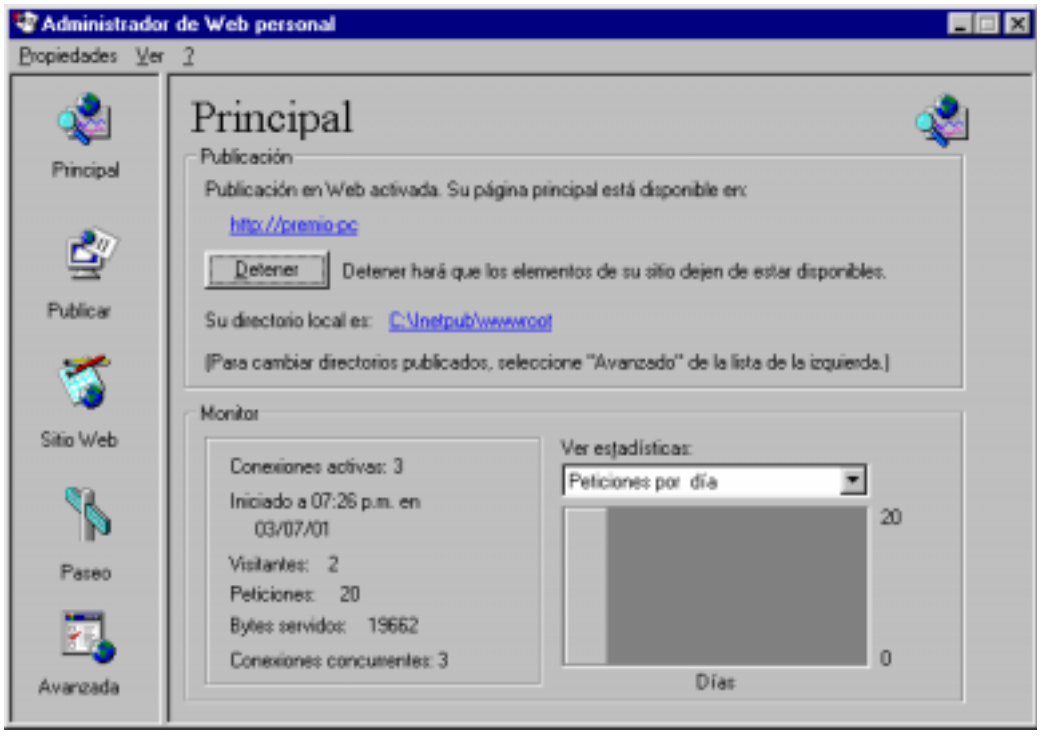

Figura 5.1 Administración de Web Personal (inicial).

Luego de lo cual se seleccionará la opción ***avanzada*** que nos mostrará la siguiente pantalla con el detalle de las direcciones virtuales.

Seleccione ***HOME*** y pulse el botón ***Agregar*** para colocar la carpeta virtual que hará la conexión con la carpeta donde se encuentran sus páginas ASP, que desea publicar

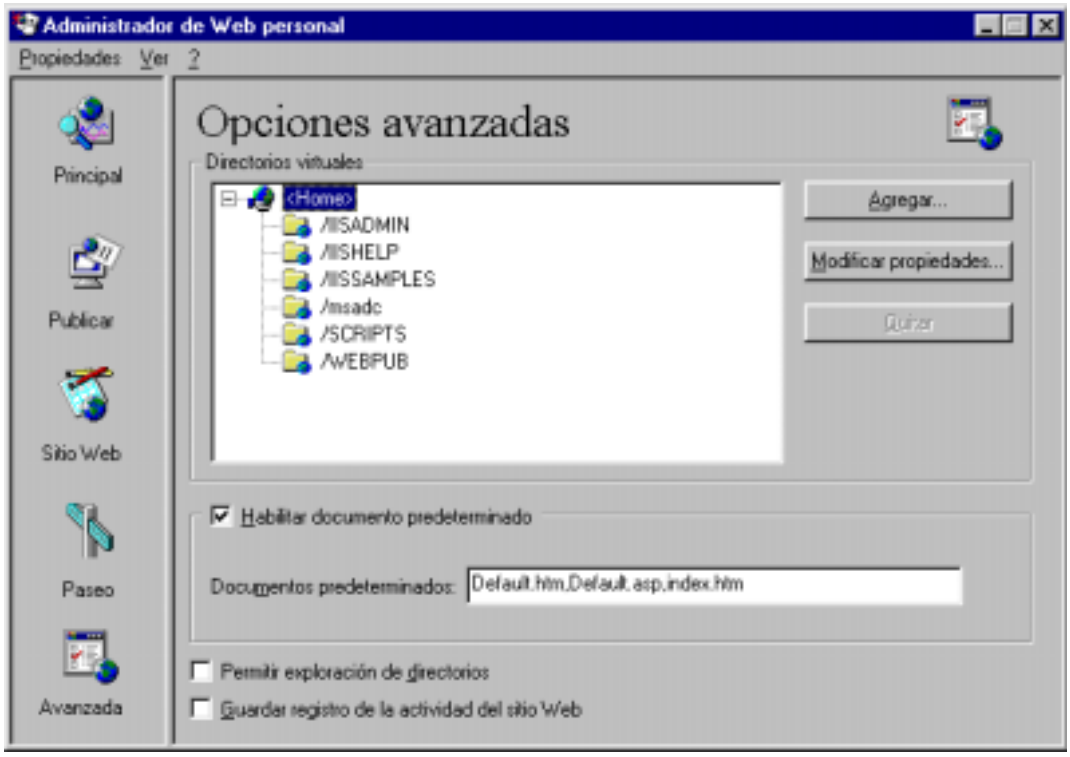

Figura 5.2 Opciones avanzadas.

Presione ***Examinar*** para seleccionar la carpeta donde se encuentre sus páginas ASP.

Luego en ***Alias*** se coloca el nombre de la carpeta que se vaya a utilizar en el proyecto, por ejemplo: "Paginas ASP".

Seleccione la casilla de Ejecución de la selección de Accesos para ejecutar las paginas y presione ***Aceptar***

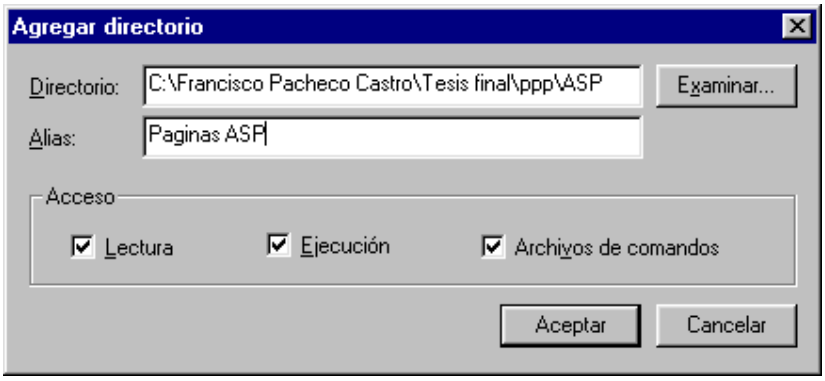

Figura 5.3 Agregar directorio.

Una vez realizado todo estos pasos podrá ver su carpeta anexada como en la siguiente figura.

En documentos predeterminados inserte ***index.htm*** que es su archivo inicial, cierre el Administrador de Web Personal y con todo esto esta listo para ejecutar las paginas ASP generadas.

Figura 5.4 Administrador de Web Personal (final).

En el Internet Explorer digite la siguiente dirección **http://200.10.151.164/proyecto/index.htm,** es la dirección para acceder a nuestro proyecto en la red.

Le aparecerá la pantalla inicial del proyecto.

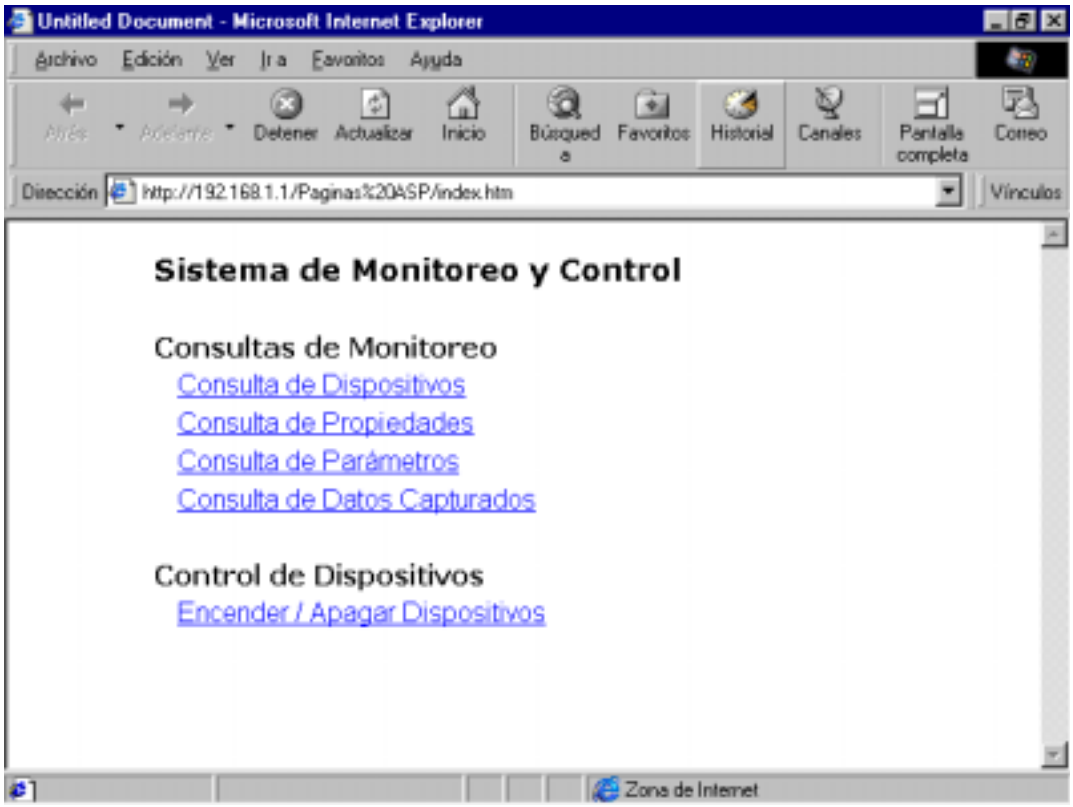

Figura 5.5 Pantalla inicial.

5.3.2 Configuración de Programa Monitoreo y Control.

Como primer paso obtenga la librería **Win95io.dll** y cópiela en la dirección C:// Windows/System32, esta es la librería que necesita el software que haremos mención.

Copie la carpeta en la cual se encuentra el proyecto a su computador, la cual contiene una base de datos que le servirá para poder trabajar.

Configure el controlador de su base de datos vea la sección **4.5.27 Componente Acceso a Base de Datos** para configurar ***monitoreo.mdb*** que se encuentra dentro de la carpeta copiada anteriormente C://ppp/Datos, en nuestro caso. El nombre de origen de datos será ***Paralelo***.

Cargue el archivo ejecutable desarrollado para este fin.

En la primera ventana le pedirá usuario y contraseña, digítelas y pulse ***Aceptar***.

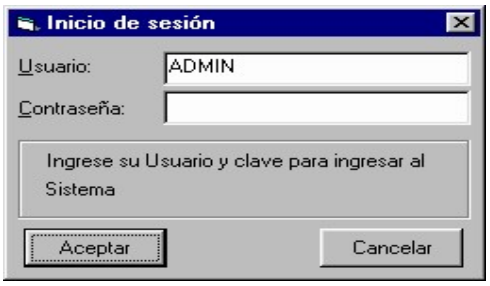

Figura 5.6 Inicio de Sesión.

Seleccione Menu Maestro/Generalidades/Tablas Maestras, para configurar los ***Dispositivos***, ***Propiedades*** y ***Parametros***.

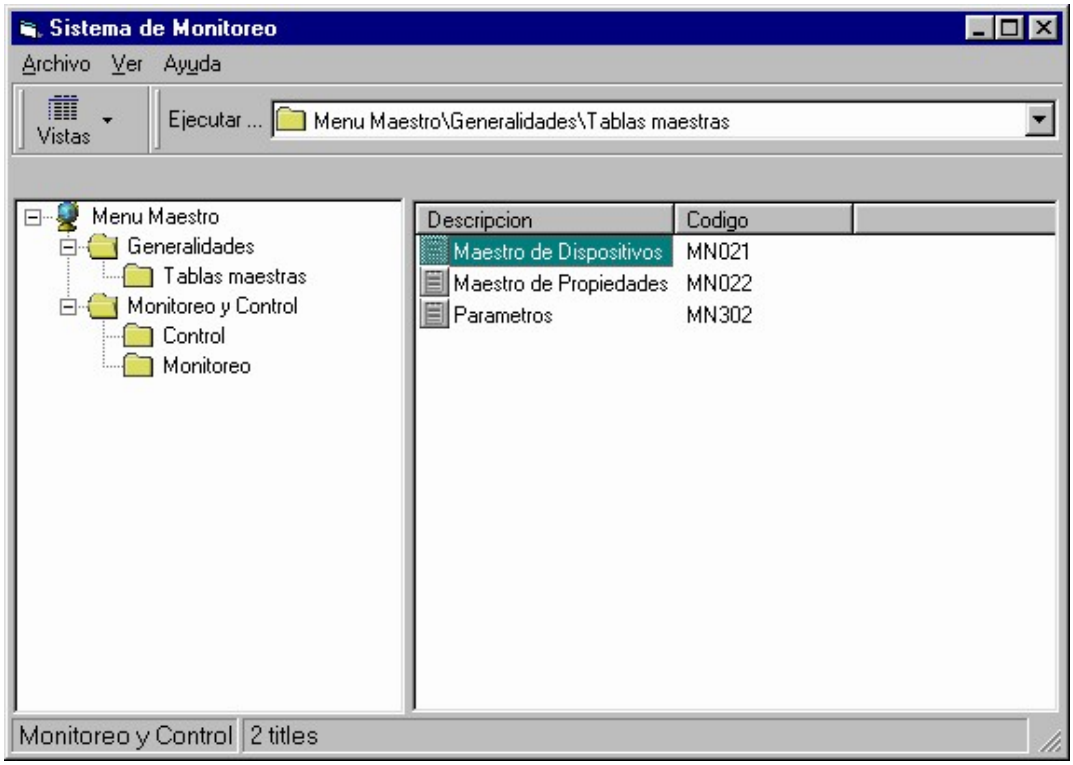

Figura 5.7 Sistema de Monitoreo.

Seleccione el item ***Maestro de Dispositivo*** a fin de configurar los dispositivos que se conecten al computador.

Le aparecerá una pantalla que esta conectada a la base de datos ***monitoreo.mdb***, si no estuviere en blanco elimine los items con el botón ***Eliminar***.

Con el botón ***Agregar*** ingrese los dispositivos que vaya a utilizar. Los Dispositivos los puede ordenar por Código o por Descripción.

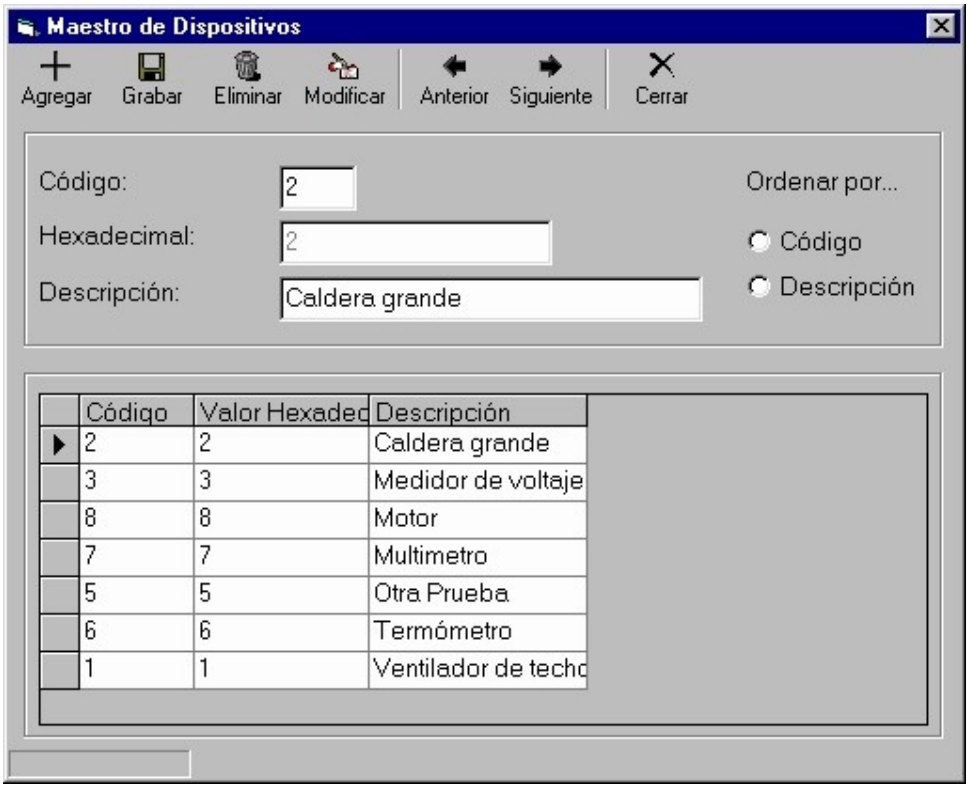

Figura 5.8 Maestra de Dispositivos.

Seleccione el item ***Maestro de Propiedades*** a fin de configurar las propiedades de los dispositivos anteriores, recuerde que cada dispositivo tendrá una o más propiedades según lo que se requiera.

Agregue o elimine según lo que se requiera para determinar las características de los dispositivos a ser conectados.

La pantalla que aparece esta conectada a la base de datos ***monitoreo.mdb***, si no estuviere en blanco elimine los items con el botón ***Eliminar***.

Proceda a llenar las Propiedades.

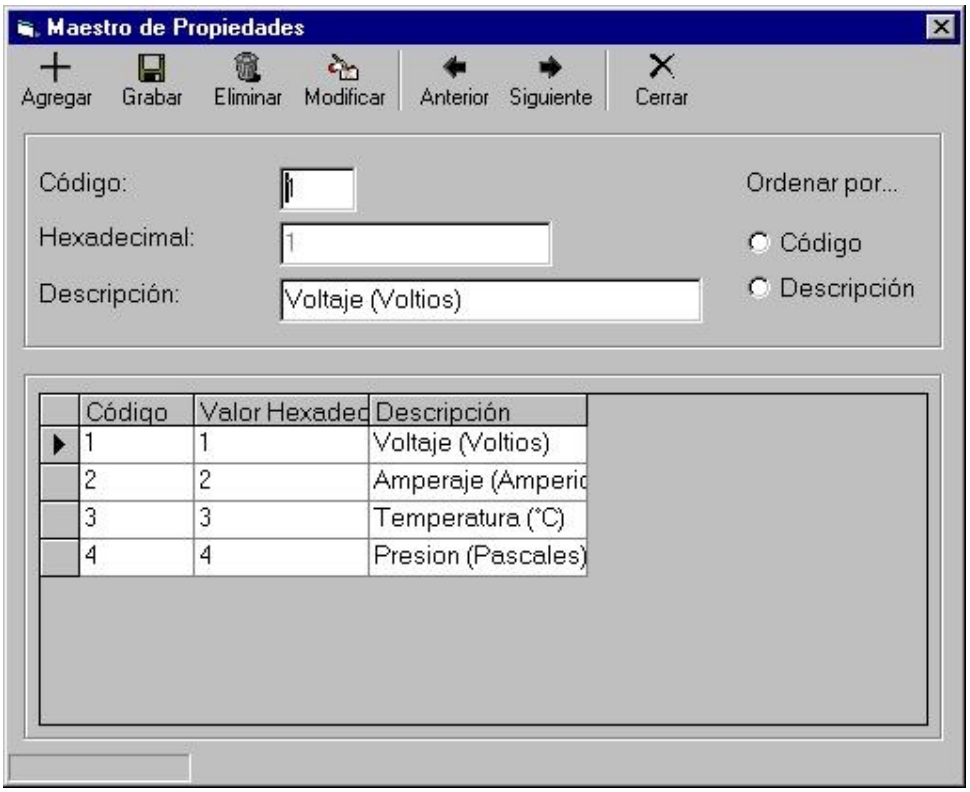

Figura 5.9 Maestro de Propiedades.

En la selección de ***Parametros*** se pondrá los valores mínimo y máximo de los dispositivos a ser conectados.

Seleccione el dispositivo, luego la propiedad y sete los valores mínimos y máximos para los dispositivos que estén conectados.

Guarde los cambios que genere.

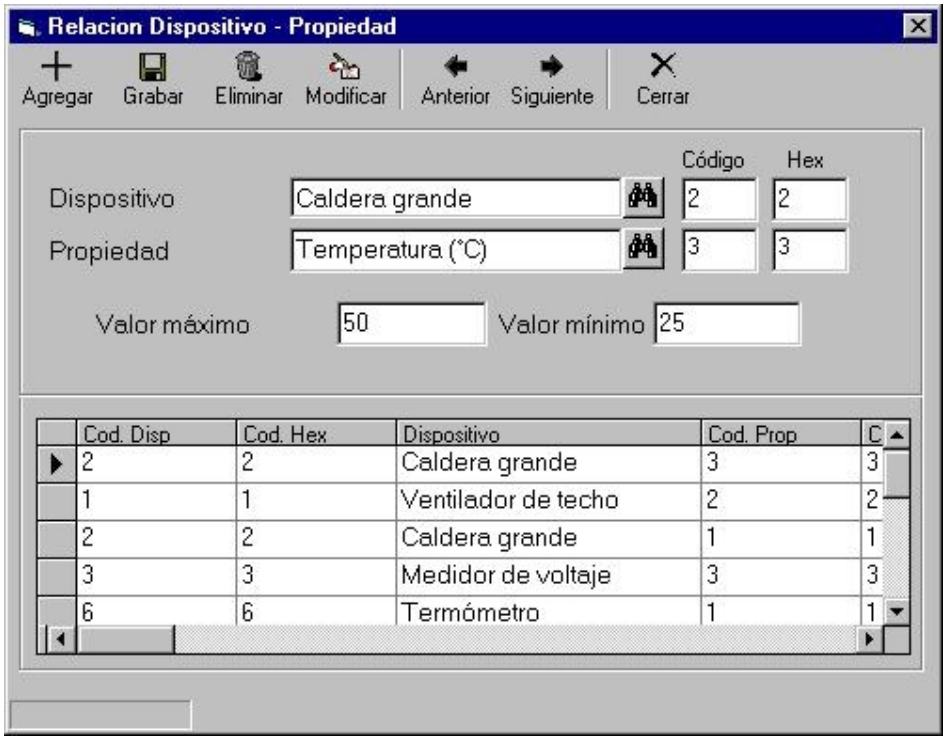

Figura 5.10 Iniciar Parámetros.

5.4 Ejecución del sistema (Servidor).

Para la ejecución del sistema, ingrese a la carpeta Monitoreo y Control de la aplicación desarrollada.

Dentro de la carpeta Monitoreo, haga doble click en el icono Monitoreo que aparece.

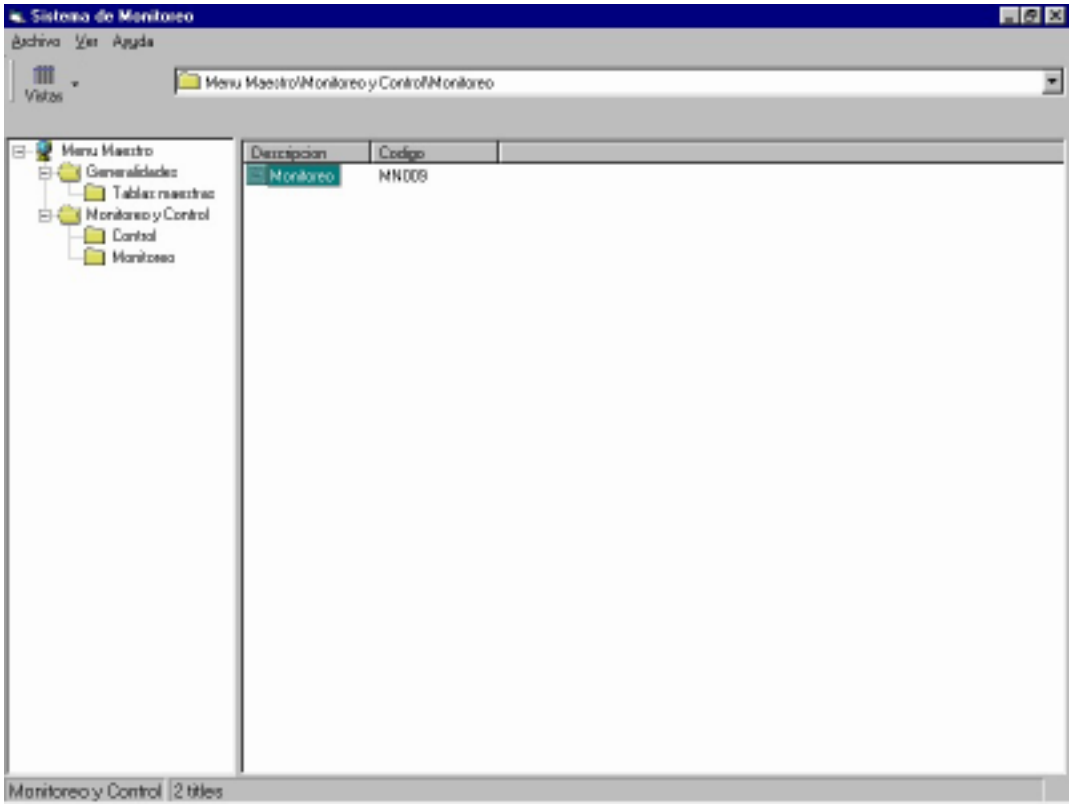

Figura 5.11 Ejecución del Sistema.

En la ejecución del monitoreo predetermine los valores de cada que tiempo censará los equipos.

Existe un boton de monitoreo por paso para corregir cualquier error que se incurrio anteriormente.

Habilite la casilla de ***Grabar Datos*** cuando todos los pasos anterior estén completos y no exista ningún error.

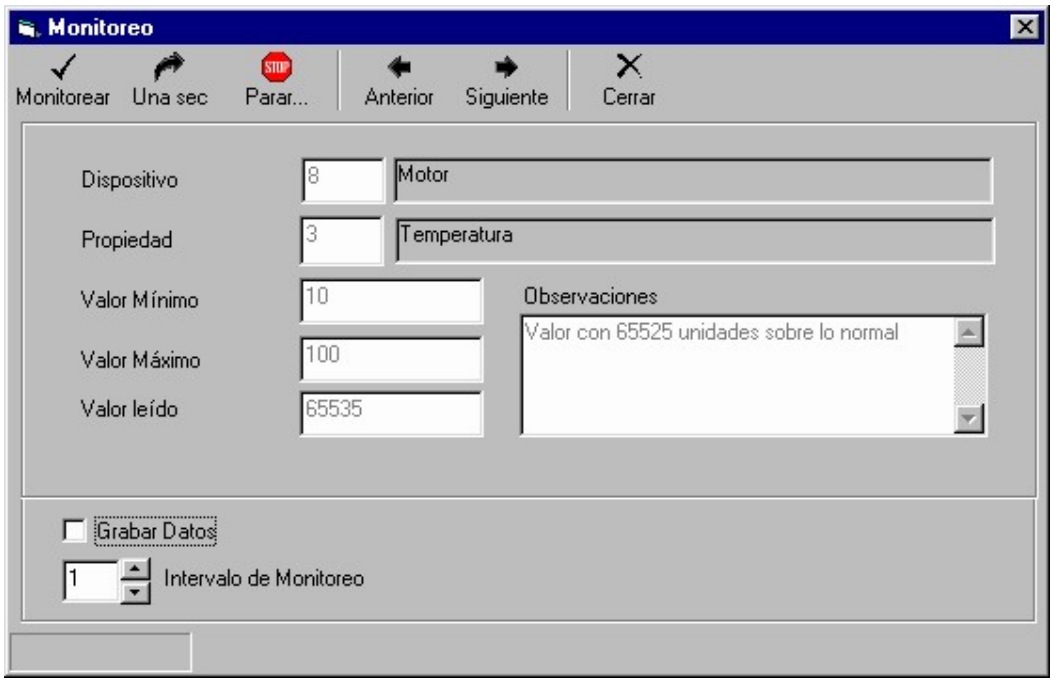

Figura 5.12 Monitoreo.

Para controlar los equipos debe ejecutar el icono Control que estará en la carpeta del mismo nombre.

Seleccione el dispositivo a ser controlado, la acción de prender o apagar y pulse ***Control***.

Para todo esto debe estar conectado al Desarrollo electrónico **MLF 2.0**.

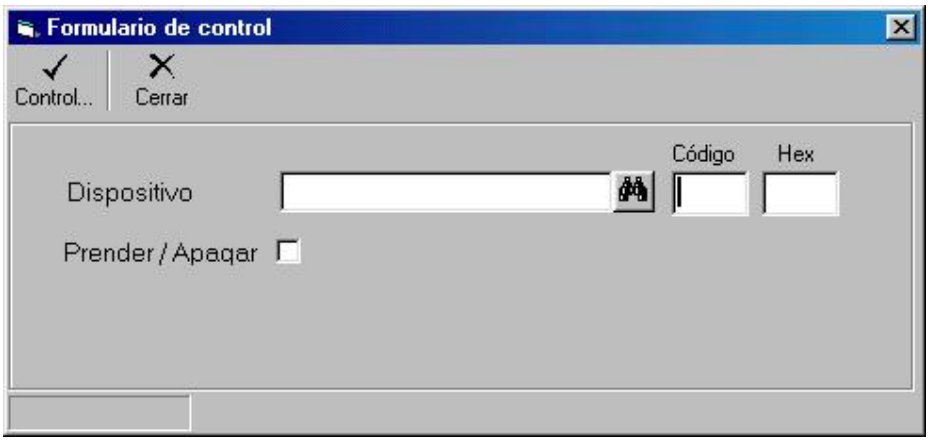

Figura 5.13 Formulario de Control.

VI. ANÁLISIS COSTO-BENEFICIO.

6.1 Desarrollo de software.

En este capítulo se explicarán los procedimientos que se llevaron a cabo para desarrollar el software de Monitoreo y Control.

Después se pondrá a consideración del lector la serie de costos, tiempos de ejecución, así como también los conocimientos básicos para desarrollar esta alternativa.

6.1.1 Licencias.

Se asume que las licencias de Windows 95/98/NT o Windows 2000 que tienen un costo en el mercado ya se han incurrido en ellas mediante la

compra del hardware que se va ha necesitar. Adicionalmente para no tener problemas de orden legal se optaría por comprar el software de **Visual Basic** valorado en aproximadamente 200 dólares.

6.1.2 Requerimientos para desarrollo.

Para la realización de este software fue necesario utilizar:

Máquinas.- Un computador con un costo aproximado en el mercado es de USD 900, pero si se tiene una infraestructura en red mucho mejor. Para el montaje de la oficina se recomienda una PC PENTIUM III por cuanto se requiere de una PC que nos permita crecer o actualizar los programas requeridos en algún momento, además, de recuperar la inversión con un valor de salvamento al final del proyecto.

Diseñadores/Desarrolladores.- Nuestros salarios durante el tiempo de desarrollo e implementación, mensualmente fueron de USD200 medio tiempo durante 5 meses.

Oficina.- El costo de montar una oficina varía desde los gustos en calidad de inmobiliario hasta la situación geográfica de la oficina, con un estimado muy bueno en el centro de Guayaquil es de USD200 (alquiler).

Gastos de telefonía.- Consumo de 50 horas mensuales por concepto de conexión a Internet y llamadas telefónicas es de USD80.

Suscripción a un servidor de ISP's.- Los gastos de telefonía, así como de suscripción a un servidor de ISP's, solo se requerirán en situaciones extraordinarias, para requerir actualizaciones de software o librerías que se necesiten de la red. El precio de este servicio oscila entre 20 y 30 dólares.

6.1.3 Conocimientos básicos.

Debe de tener los siguientes conocimientos:

- Programación orientada a objetos.
- Lenguaje "**VISUAL BASIC**".
- Active Server Pages (ASP).
- Personal Web Server

6.1.4 Tiempo.

El tiempo de desarrollo del software fue de 5 meses (Aproximadamente) a medio tiempo (tiempo acoplado a nuestra disponibilidad).

6.2 Costos de publicación y mantenimiento.

6.2.1 Costos de montaje del proyecto

En la siguiente tabla se muestra los gastos en que se incurrirá para el diseño del software, en un tiempo estimado de 5 meses. Todo esto incluye gastos de PC, oficina y personal involucrado directamente en el proyecto.

COSTO DE MONTAJE DE PROYECTO				
CONCEPTO	TIEMPO	CANTIDAD	COSTO UNITARIO	VALOR TOTAL **Dólares**
Computador	1 sola vez	1	900	900
Alquiler de oficina	mensual	5	200	1000
Consumo telefónico	mensual	5	40	200
Compra de licencias	1 sola vez	1	400	400
Varios*	1sola vez	1	300	300
Diseñadores/Desarrolladores X 3	mensual	5	200	3000
TOTAL				**5800**

* incluye: papelería , mantenimiento, luz de oficina, etc.

Tabla 6.1 Costo de montaje de proyecto.

Se puede hacer un estimado de gasto de montaje al dividir el total para el tiempo que se toma desarrollar el proyecto. Haciendo esto nos da un gasto de **USD$ 1160** (Dólares norteamericanos) por cada mes.

6.2.2 Inversión adicional.

Como el software puede ser ejecutado sobre una red interna de una organización (LAN) y/o en una red (IP), hemos creído conveniente asesorarnos con los costos de los diferentes proveedores de acceso a Internet de nuestro país a fin de que el lector tenga una idea de la inversión adicional que tendrían que hacer las organizaciones para sacarle el mayor provecho al sistema.

La inversión adicional que se haga sin duda será en beneficio de las organizaciones ya que tendrán acceso a la red mundial de computadores y con ello a otros contactos adicionales. Por decirlo de otra manera nuestro sistema es un valor agregado hacia la empresa.

Proveedores de acceso a Internet en nuestro país hay una amplia elección de empresas, por lo que los costos tenderán en un futuro cercano a disminuir, con un amplio beneficio para los consumidores finales y por ende muchas organizaciones estarán en capacidad de hacerse de éste servicio, con lo cual el mercado de venta de programas aplicados o de uso en Internet se amplían considerablemente.

En la siguiente tabla se muestran los costos adicionales para que los usuarios (empresas) obtengan un mejor rendimiento del proyecto.

INVERSIÓN ADICIONAL DEL USUARIO PARA USO DEL SERVICIO DE MANERA CORPORATIVA				
CONCEPTO	TIEMPO	CANTIDAD	COSTO UNITARIO	VALOR TOTAL **Dólares**
Enlace de ultima milla	anual (Portanet)	1	3500	3500
	anual (Satnet)	1	2550	2550
Compra de dominio	anual (Portanet)	1	35*	35
	anual (On-net)	1	30	30
	anual (Satnet)	1	50	50
	anual (Network Solutions)	1	17.5**	17.5
	anual (Ecuanet)	1	50	50
Conexión dedicada	mensual (Satnet de 128K)	1	1500	1500
	mensual (Cyberweb de 128K)	1	1530***	1530
	mensual (On-net de 64K)	1	1000	1000

* adicionalmente se paga por una sola vez 20 dólares a favor del forward
** por dos años por extensiones .com,.net o .org
*** gasto adicional por cargo de activación $1500

Tabla 6.2 Costo de inversión adicional de los usuarios.

6.2.3 Costos de publicación

COSTO DE PUBLICACIÓN DEL PROYECTO				
CONCEPTO	TIEMPO	CANTIDAD	COSTO UNITARIO	VALOR TOTAL **Dólares**
web hosting	**mensual (Portanet)**	**15Mb**	**0.54**	**8**
	Mensual (Ecuanet)	**1Mb**	**20**	**20**
	Mensual (Satnet)	**20Mb**	**0.3**	**6**

Tabla 6.3 Costo de publicación de proyecto.

Los costos de publicación es la mensualidad que se debe pagar para que su página este disponible en el WWW.

6.3 Beneficios

Uno de los beneficios de realizar un software de manera que se pueda compartir colectivamente dentro de una organización son los bajos costos de adquisición de licencias, ya que todos pueden acceder a la información pero con ciertas restricciones, que sin duda cualquier sistema debe proporcionar, sobre todo seguridad y poder discriminar a los usuarios no autorizados en tiempo muy corto.

El poder acceder a la información en tiempo real nos permite hacer una correcta planificación de los recursos tanto materiales como humanos, donde los tiempos de respuesta y adaptación de las organizaciones a los cambios se torna prioritario día con día.

La idea principal de realizar este tema fue tener una visión de cómo poder unir el amplio campo de los lenguajes de programación (en este caso se escogieron Visual Basic y ASP por lo nuevo y versátil), con la electrónica y los dispositivos para medir los fenómenos físicos que nos rodean.

El sistema puede ser implantado en una industria donde el conteo de cierto producto de salida o entrada se necesite conocer en un determinado momento, saber si un equipo deja de funcionar, ver los niveles de producción

en el mes o cuanto producto se despacho al final del día. Todo esto nos conllevará a muy corto plazo a automatizar los procesos y planificar correctamente los recursos.

Otro ejemplo podría ser la seguridad en un lugar determinado, donde existen controles de movimiento, sensores de temperatura, controles de iluminación, etc. A fin de conocer en un corto tiempo de una situación de emergencia y poder tomar las acciones que se requieran para una inmediata solución.

Sin duda alguna, los beneficios obtenidos al realizar este tipo de proyecto es muy grande, esto nos lleva a poder asociar o a conjugar toda una tecnología de punta que día a día se impone más, hasta en nuestros hogares como es la automatización y control en tiempo real.

CONCLUSIONES Y RECOMENDACIONES

Esperamos que este tema nos sirva de mucha ayuda a fin de generar una serie de ideas y productos que sirvan en nuestro país, a fin de dar una diferente alternativa a un costo relativamente razonable y sobre todo no desechar lo que actualmente se tiene en las industrias o en determinado sitio si no mas bien adaptar nuestro sistema al existente con el único objetivo de crear instituciones altamente eficientes.

En el ámbito tecnológico nuestro país no ocupa un gran lugar en el entorno mundial, pero gran cantidad de personas muy capacitadas salen año a año de las universidades que con una correcta orientación pueden hacer la diferencia.

Las universidades deben de destinar recursos para investigación y acercar al estudiante a su ámbito profesional desde el inicio mismo de su carrera, por cuanto son los únicos organismos capacitados e idóneos para la consecución de objetivos que ayuden a la sociedad.

GLOSARIO Y TERMINOLOGÍA

μ .- micro.

°C .- Grados Centígrados.

Active X .- Aplicaciones pequeñas con lenguajes de programación orientadas a objetos.

ADO.- El ActiveX Data Access provee una interface a los proveedores de datos como, Microsoft SQL, Access u Oracle. ADO es utilizado dentro de las páginas ASP, para comunicarse con las bases de datos.

API´s .- Interfaces para programas de aplicación estándar.

Bits .- Señales digitales representadas en unos y ceros.

Byte .- Conformación de grupos de bits.

CA .- Corriente Alterna.

cm .- Centímetros.

Convertidores A/D .- Convierten una señal analógica en digital.

CPU .- Unidad Central de Proceso.

DC .- Corriente Continua.

E/S .- Dispositivos de entrada / salida.

EPROM, EEPROM .- Memorias programables borrables y eléctricamente borrables.

ETHERNET .- Red interna en una organización.

fem .- Fuerza Electromotriz.

GPIB – ENET .- Emulador, para transmisión y recepción de datos de un dispositivo GPIB a una ETHERNET y viceversa.

GPIB .- (General Purpose Interface Bus). Bus de interface de propósito general.

Host .- Computadores que aparecen como servidores de una serie de computadores conectados en red.

HTML .- Formato de archivos utilizados en Internet, conocido con páginas WEB.

Internet Explorer .- Visualizador de Internet de Windows, disponible en la red de Internet.

ISP´s .- Proveedor de servicios de Internet.

LAN .- Redes de Area Local.

Local system .- Sistema local.

OLE-DB.- OLE-DB es una interface a proveedores de datos, como Microsoft SQL Server o Access. Por ejemplo, ADO utiliza un set de interfaces provistas por OLE-DB para comunicarse con Microsoft SQL Server. Nunca se usa directamente OLE-DB en las páginas active server, sino en cambio se utiliza ADO como una interface de más alto nivel hacia OLE-DB.

ODBC.- El Open Database Conectivity fue utilizado como un estándar para comunicarse con base de datos, casi todas las bases de datos en el

mercado, son actualmente compatibles con ODBC, sin embargo Microsoft está gradualmente reemplazando ODBC con OLE-DB.

Mac OS .- Sistema operativo desarrollado por Apple computer para sus computadoras Macintosh.

mV .- Milivoltios.

Netscape .- Visualizador de internet, disponible en la red de Internet.

OPC .- OLE **aplicaciones** para procesos de control.

PC .- Computadora personal.

PENTIUM .- Marca registrada por INTEL para la serie de microprocesadores 586, 686, 786.

PLC .- Controladores Lógicos Programables.

RDA .- Dispositivo de acceso remoto.

Remote system .- Sistema remoto.

Routers .- Dispositivos capaces de enviar / recibir información para ser transmitida a través de la red.

RPM .- Revoluciones por minuto.

RS-232 / 422 .- Protocolo de comunicaciones para computadoras y de dispositivos hacia estas.

SCADA .- Programa de control industrial.

Tarjeta DAQ .- Tarjeta de adquisición de datos.

TCP / IP .- Protocolo de transferencia de comunicaciones o protocolo de Internet.

Visual Basic .- Lenguaje de programación, que permite hacer interfaces gráficas.

WAN .- Redes de acceso mundial.

Windows 95/98/2000/NT .- Sistema operativo desarrollado por Microsoft, para máquinas IBM y compatibles.

Ω .- Ohmios.

ANEXO A

Win95io, Version 1.01

Win95io.DLL is a 32-bit DLL that provides access to port input/output (I/O) under Windows 95. It was written for use with 32-bit Visual Basic since Visual Basic does not have I/O instructions. Note that, due to the Win32 architecture, the DLL may not always behave exactly as expected. This DLL is documented in more detail below.

describe those situations where we believe it is not be useful. SoftCircuits makes no other claims with regards to this software including claims that this documentation is accurate. Use this software and documentation at your own risk.

This example program was provided by:

SoftCircuits Programming

http://www.softcircuits.com

P.O. Box 16262

Irvine, CA 92623

==========

While DOS and Windows 3.x applications may perform port I/O freely, Win32 operating systems assume that port I/O is handled by device drivers. As discussed in Microsoft KB article Q112298, attempting to use the port I/O functions from within an application for Windows NT running in user mode causes a privileged instruction exception to occur. Although Windows 95 takes a similar approach, port access under Windows 95 does NOT cause a protection fault. Thus the reason for writing this DLL.

Although port access under Windows 95 does not cause a protection fault, it may not always perform as expected. For example, if a DOS application is

accessing the same port, or a device driver has control of the same port then a port read or write may be ignored. Further, if you write a 32-bit application that uses this DLL, care must be taken to prevent it from running under NT.

Declarations:

```
Declare Sub vbOut Lib "WIN95IO.DLL" (ByVal nPort As Integer, ByVal nData As Integer)
Declare Sub vbOutw Lib "WIN95IO.DLL" (ByVal nPort As Integer, ByVal nData As Integer)
Declare Function vbInp Lib "WIN95IO.DLL" (ByVal nPort As Integer) As Integer
Declare Function vbInpw Lib "WIN95IO.DLL" (ByVal nPort As Integer) As Integer
```

BIBLIOGRAFÍA.

1. Amplificadores Operacionales y circuitos integrados lineales, Quinta edición (Robert Coughlin, Frederick Driscoll).
2. Autómatas Programables (Josep Balcells y José Luis Romeral)
3. Catálogo de Medición y Automatización Versión Resumida (National Instruments).
4. Diseño electrónico (Savant, Roden, Carpenter).
5. Instrumentación Electrónica Moderna y Técnica de Medición (Willian Cooper y Albert Helfrick).
6. Instrumentation newsletter third Quarter 1999 (National Instruments)
7. Introduction to the Internet Protocols (Charles L. Hedrick)
8. Redes de Computadoras, Internet e Interredes (Douglas E, Comer).
9. Redes globales de información con Internet y TCP/IP (Douglas E. Comer).
10. http://www.ella.swin.net
11. http://www.lawebdelprogramador.com
12. http://www.lptk.com
13. http://www.monografias.com/trabajo5/datint
14. http://www.monografias.com/trabajos/paralelos
15. http://www.mattjustice.com/parport/part_vb.html
16. http://www.softcircuits.com/sw_tools.com
17. http://www.aspfacil.com
18. http://www.lavariable.com