



INSTITUTO POLITÉCNICO NACIONAL



**CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
(C.I.C)**

**DISEÑO Y CONSTRUCCIÓN DE UN MICRO WEB SERVER PARA
MONITOREO DE SEÑALES ANALÓGICAS Y DIGITALES**

TESIS

**QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON
ESPECIALIDAD EN SISTEMAS DIGITALES**

PRESENTA

ING. GENARO SOLA ORTIZ

DIRECTOR: M. EN C. AMADEO JOSÉ ARGÜELLES CRUZ

México, D.F.

Octubre del 2003

A mis padres Dolores y Benito por el amor, apoyo y confianza que me otorgaron, por el amor a la verdad y a la superación,

A mis hermanos por la paciencia y ayuda.

A ti pequeña Haydeé por el amor y el deseo de superación que supiste sembrar en mí.

A mis amigos por todo el tiempo que sin obligación me brindaron.

AGRADECIMIENTOS

Al Centro de Investigación en Computación por darme la oportunidad llevar a cabo mis estudios de posgrado y por permitirme utilizar su infraestructura e instalaciones.

Al Instituto Politécnico Nacional, por las oportunidades de desarrollo que me fueron otorgadas.

A mi asesor y director de tesis, M. en C. Amadeo José Argüelles Cruz, por la dirección del presente desarrollo, por los consejos profesionales y personales, por las pláticas interminables o aquellas discusiones sobre tópicos de investigación y desarrollo.

A todos los profesores del C.I.C. que compartieron sus conocimientos sin otro fin que el de la superación de sus alumnos y del instituto. Al profesor M. en C. Pablo Manrique Ramírez, por el apoyo incondicional y disposición de los recursos materiales necesarios.

ÍNDICE

RESUMEN.....	I
ABSTRACT	II
ÍNDICE.....	III
ÍNDICE DE FIGURAS	V
ÍNDICE DE TABLAS	VI
GLOSARIO.....	VII
INTRODUCCIÓN.....	1
INTERNET.....	4
Principios.....	4
Redes LAN	5
Métodos de transmisión LAN	6
Topologías LAN.....	7
Dispositivos LAN.....	8
Protocolos Lan y el modelo de referencia OSI.....	10
Ethernet	11
Ethernet e IEEE 802.3	12
Diferencias de servicios entre Ethernet e IEEE 802.3	13
Comparación de diferentes especificaciones de capa física IEEE 802.3.....	13
TCP/IP	14
IP. El protocolo Internet.....	15
La Cabecera IP	15
ICMP	19
TCP	21
Máquina de estados TCP	23
Conexión y terminación.....	24
TCP/IP y OSI.....	25
Ethernet/IP/TCP.	26
Control de direcciones Internet.	28
ARP(Address Resolution Protocol).	29
DESARROLLO.....	31
Descripción del problema	31

Objetivos.....	32
Justificación	32
Dispositivos existentes	33
Sistema Propuesto	35
RTL8019AS.....	35
Configuración y arranque.....	40
IMPLEMENTACIÓN	43
Diseño del dispositivo.....	44
Sistema de control	44
Conexión Ethernet.....	45
Canales de comunicación.....	46
Canal RS232	46
Canal I ² C	46
Sistema de Memoria	47
Sensor de temperatura.....	47
Empleo del sistema y restricciones.....	48
Pruebas Realizadas	49
CONCLUSIONES.....	52
TRABAJOS FUTUROS.....	53
BIBLIOGRAFÍA.....	54
ANEXOS.....	56
ANEXO A.- Código ensamblador	56
ANEXO B. Capa de Soldadura.	92
ANEXO C. Capa de componentes.....	93
ANEXO D. Diagramas Electrónicos	94
ANEXO E. Código Java para el control del puerto digital	95
ANEXO F. Código Java para lectura de la temperatura	96
ANEXO F. Código en Lenguaje C para codificar los archivos en formato hexadecimal para almacenar en memoria permanente	97

Índice de Figuras

Figura 1. Internet, comunicacion global.	4
Figura 2. Red de computadoras.....	5
Figura 3. Red de computadoras LAN.....	5
Figura 4. Representación de Red WAN.....	6
Figura 5. Topología canal o lineal.....	7
Figura 6. Topología Anillo.....	7
Figura 7. Topología de estrella.....	7
Figura 8. Repetidor fibra óptica.....	8
Figura 9. Concentrador.....	8
Figura 10. Puente entre TCP/IP, AppleTalk, DecNet, Netbeui y Ethernet.....	9
Figura 11. Conmutadores de paquetes (switches).....	9
Figura 12. Ruteador.....	10
Figura 12. Especificación LAN dentro de OSI.....	10
Figura 13. Conectores Ethernet (Coaxial, Par trenzado y fibra óptica).....	11
Figura 15. Modelo IP.....	14
Figura 16. Servicios TCP/IP en OSI.....	25
Figura 17. Capas de aplicación y transporte para diferentes servicios IP.....	25
Figura 18. Encapsulación de paquetes.....	26
Figura 19. Demultiplexado de un paquete.....	27
Figura 20. Diagrama a bloques del sistema.....	35
Figura 21. Diagrama del controlador RTL8019AS.....	36
Figura 22. Diagrama de tiempos de lectura escritura RTL.....	37
Figura 23. Diagrama de memoria DMA RTL.....	38
Figura 24. Diagrama Interno FL1012.....	39
Figura 25. Diagrama de flujo del proyecto.....	40
Figura 26. Conexión TCP/IP.....	42
Figura 27. Diagrama de conexiones AVR.....	44
Figura 28. Diagrama de conexiones RTL8019AS.....	45
Figura 29. Canal RS232.....	46
Figura 30. Canal I ² C implementado.....	46
Figura 31. Diagrama del sistema de memoria I ² C.....	47
Figura 32. Sensor de temperatura LM75.....	47
Figura 33. Respuesta para una petición de eco (ping) de 32 bytes de datos.....	49
Figura 35. Pantalla de salida del visualizador.....	51
Figura 36. Capa de Soldadura.....	92
Figura 37. Capa de componentes.....	93
Figura 38. Diagrama esquemático del sistema WEB SERVER.....	94

Índice de Tablas

Tabla 1. Estructura del paquete Ethernet	11
Tabla 2. IEEE802.3 Diferentes especificaciones	13
Tabla 3. Encabezado IP.	15
Tabla 4. Precedencia y Tipo de servicio.	16
Tabla 5. Fragmentación de un paquete.	17
Tabla 6. Proceso de fragmentación de un paquete fragmentado.....	17
Tabla 7. Protocolos superiores a IP.	18
Tabla 8. Octeto de opciones IP.....	18
Tabla 9. Encabezado ICMP.	19
Tabla 10. Mensajes ICMP.	20
Tabla 11. EncabezadoTCP.....	21
Tabla 12. Tipos de direcciones en Internet	29
Tabla 13. Encabezado ARP.	30
Tabla 14. Comparación entre dispositivos existentes.....	34
Tabla 15. Registros generales RTL8019AS.....	36
Tabla 16. Modo de operación, Command Register (CR) RTL8019AS.	37

DISEÑO Y CONSTRUCCIÓN DE UN MICRO WEB SERVER PARA MONITOREO DE SEÑALES ANALÓGICAS Y DIGITALES

Genaro Sola Ortiz

RESUMEN

El constante crecimiento de los usuarios en el mundo de Internet y la tendencia de la centralización de operaciones en Internet requieren de sistemas de bajo costo que utilicen la infraestructura de red disponible y que provean facilidades de control de sistemas remotos. Como una solución a los problemas de monitoreo de estaciones remotas, se diseñó un sistema mínimo que permita el acceso remoto vía Internet para el control y monitoreo de señales analógicas y digitales. Este trabajo describe el desarrollo de dicho sistema que se conecta a una roseta RJ45 con acceso a una red tipo Ethernet para proporcionar servicios básicos de TCP/IP, lo suficientemente robusto para implementar servicios adicionales sin modificaciones sustantivas en el núcleo principal de programa de conexión. Esto se logra diseñando el núcleo independiente a las aplicaciones que son necesarias utilizar e inclusive a la plataforma de sistema operativo seleccionada. Para la visualización de la información en un navegador, se propone un sistema de servicio de archivos que cumplen con los estándares HTML V2.0 y código de JAVA. Mediante este código se restringe la cantidad de conexiones permitidas al dispositivo, es el responsable de la lectura de sensores y dispositivos o del cambio de estado de los actuadores conectados al sistema. Los canales de comunicación utilizados para el manejo de sensores e información, son RS232, I²C, y un puerto digital de 8 bits. Se proporciona un modo de programación al sistema, para que el cambio en el núcleo del programa sea cuestión de habilitar un interruptor de programación. El puerto RS232 sirve para introducir información que puede ser publicada dentro de la página WEB generada, éste puerto sirve también para introducir los archivos generados HTML y JAVA al sistema de almacenamiento permanente. Todo esto permite un sistema versátil y muy robusto que en un futuro podrá ser habilitado para manejar grandes volúmenes de información, mediante la conexión de algún dispositivo como puede ser un disco duro, un lector de CDROM o inclusive un lector DVDROM.

CONSTRUCTION AND DESIGN OF A MICRO WEB SERVER FOR MONITORING ANALOG AND DIGITAL SIGNALS

Genaro Sola Ortiz

ABSTRACT

The constant worldwide growth of internet users and the centralization trend of internet operations require low cost systems that employing the existing net infrastructure provide remote control capabilities. As a solution to remote monitoring stations problems a system that allows remote access, control and monitoring of analog and digital signals via the internet was designed. This work describes the development of such system that connects to a RJ45 jack with access to an Ethernet like net to provide basic TCP/IP services robust enough to implement additional services without important modifications to the connection program main core. This is achieved designing a core that's independent of the needed applications or even the selected operating system platform. For the display of the information in a browser, a file system service that complies with HTML V2.0 standards and JAVA code, is proposed. This code restricts the amount of allowed connections to the device, it's in charge of reading the sensors, connected devices and changes in the state of the actuators connected to the system. The communication channels used for the management of information and sensors are RS232, I²C and an 8 bit digital port. A system programming mode is provided so that changes to the program core are made enabling a switch. The RS232 port is used to enter the information to be published in the generated WEB page, this port is also used to enter the HTML and JAVA generated files in the permanent storage system. All this allows a versatile and robust system that in the future could be enabled to manage great volumes of information by the means of any device connection as a hard drive, CDROM reader or even a DVDROM reader.

INTRODUCCIÓN

El constante desarrollo tecnológico en los sistemas de cómputo, ha permitido que el término de Internet sea cada vez más común, al grado que se considera que dentro de unos 10 años existirá en todos los hogares al menos una conexión a Internet, ya sea por medio de una computadora personal o utilizando dispositivos dedicados para dicha conexión.

Las principales barreras que presentaban la calidad y la velocidad de conexión de los sistemas remotos, han sido salvadas, y pareciera ser que no existe un límite, por lo que muchas empresas pronostican que Internet será el medio de comunicación masivo de mayor uso dentro de unos cuantos años.

Hoy en día se puede revisar cuentas bancarias, realizar compras, intercambiar información, acceder a bibliotecas en otros países, o incluso realizar visitas virtuales a lugares tan alejados como el polo norte o navegar en el espacio mismo. Todo ello gracias a Internet.

Se puede asegurar que Internet se convertirá, si no lo es ya, en el sistema mundial de comunicación y prestador de servicios. Es por ello que muchas compañías han comenzado a migrar sus sistemas y han desarrollado nuevas y novedosas aplicaciones para Internet.

Los sistemas de control no pueden quedar excluidos de esta tendencia global. Hoy en día se habla de refrigeradores que, basándose en un sistema de inventario de comestibles, pueden ser capaces de elaborar una petición electrónica de consumibles faltantes, en tiendas departamentales virtuales que cuentan con dicho servicio¹.

La tendencia se muestra clara, centralizar prácticamente cualquier servicio y sistema de control en Internet; imagínese, por ejemplo, estar fuera de casa y poder encender las luces de su casa, encender el sistema de riego de su jardín, ver si esta encendida la televisión, un juego de video, o inclusive ver mediante una cámara digital, algunas de las recamaras de su domicilio. Para las industrias, el tener un sistema de control de procesos, con el que puede saber exactamente el número de artículos producidos de un determinado producto, poder obtener las condiciones de temperatura de bodegas, gasto de combustible de algún equipo o maquinaria; sin importar, donde se encuentre, siendo solamente necesario estar conectado a Internet, puede representar un gran ahorro de tiempo y dinero.

Es por estas y muchas otras razones que algunas empresas, se dedican a proporcionar soluciones de control y monitoreo de sistemas remotos.

Muchos de estos desarrollos dependen, además de los dispositivos de control, de una computadora personal que se encuentre conectada a Internet. Esta computadora es la encargada de realizar los enlaces de

¹ LG Appliances <http://www.lgappliances.com/demo.html>

conexión a Internet y mediante el uso de puertos de entrada y salida de la computadora, se pueden controlar cualquier tipo de actuadores.

El presente trabajo intenta cubrir las necesidades de monitoreo y control en lugares donde ya se cuenta con una infraestructura de conexión a Internet por medio de una red Ethernet. No necesita de una computadora personal para realizar el enlace vía TCP/IP, o para establecer los métodos de control y monitoreo. El sistema en sí es un pequeño sistema de cómputo dedicado, que tiene incrustado toda la programación necesaria para realizar la conexión a Internet y manejar los dispositivos analógicos y digitales así como el código necesario para publicar paginas WEB y ejecutar programas en Java.

Las principales ventajas que representa el dispositivo con respecto a los sistemas basados en computadoras personales son los siguientes.

- Bajo costo (aproximadamente 200 pesos).
- Espacio reducido (20 X 15 X 2 cm).
- Fácil de transportar y reconfigurar.
- Puertos de entrada y salida.
- Conexión a red RJ45 10 baseT.
- Canales dedicados de intercambio de comunicación I²C y RS232, 2 canales de 8Bits, y un comparador analógico.

A pesar de la gran versatilidad del dispositivo, es indudable que existen algunos límites para el uso de este sistema.

- 8Kb de memoria de programa tipo Flash, memoria RAM de 512 bytes, 256Kb de memoria EEPROM.
- Si es necesario cambiar alguna de las variables a medir, el sistema tendrá que ser reconfigurado a nivel de lenguaje ensamblador, actividad que tendría que hacerse con personal especializado.
- La programación de la memoria Flash se realiza con un programa independiente al código del sistema.
- El cambio de las direcciones MAC del sistema están incrustadas dentro de una memoria con protocolo 3 wire, que no puede ser programada directamente por el microcontrolador ésta debe ser realizada externamente.

Algunas de las desventajas que se presentan pueden solucionarse generado un compilador para el microcontrolador AVR, que sea capaz de traducir el código HTML que generan programas de alto nivel, e incluir una librería especial generada en Java o C++, para el manejo de puertos de entrada y salida.

Un compilador solucionaría muchos de los problemas presentados aquí. Es oportuno mencionar que este trabajo de tesis forma solamente la base necesaria para el desarrollo de nuevos sistemas independientes, con mayor poder para el intercambio de información con mucho futuro, ya que se plantea el manejo de información masiva donde es factible el control de discos duros, unidades de CDROM, DVDROM, cámaras digitales, servicios de correo electrónico, Ftp, o incrementar los servicios de TCP/IP.

En el capítulo uno se establece el panorama general de Internet desde sus inicios y cambios que ha tenido al paso del tiempo, así como el posible desarrollo y tendencia de este sistema de intercambio de información. Se mencionan los principales servicios del protocolo TCP/IP, sobresaltando aquellos que son proporcionados por el sistema mínimo de conexión a Internet propuesto. Se establece un pequeño marco teórico de los diferentes tipos de servicios y enlaces a Internet, mencionando las principales diferencias entre estos, intentado justificar la elección del sistema Ethernet como sistema base de interconexión a Internet.

En el capítulo dos se establecen las principales directivas que fueron seguidas para desarrollar el sistema mínimo de interconexión. Realizando comparaciones entre los sistemas existentes y mencionado las principales características de cada uno de ellos, así como sus desventajas frente el sistema desarrollado, se establecen las características innovadoras del sistema propuesto.

En el capítulo tres se describe el procedimiento de construcción y las características de ingeniería que se encuentran en el sistema. Se explica el uso de cada uno de los puertos del sistema, ya sean de configuración o de comunicación, así como la manera en que pueden ser reconfigurados y programados. Se describe la forma en que el sistema se debe utilizar y se establecen los límites del sistema. Se detallan las pruebas a las que fue sometido el sistema para establecer algunas características de velocidad de comunicación y saturación. En base a estas mediciones se da un panorama general de rendimiento obtenido con respecto a sistemas existentes.

El capítulo cuatro presenta las conclusiones obtenidas del presente trabajo, se detallan los logros obtenidos y las modificaciones en código de programa y de diseño, necesarias para mejorar el presente diseño. Se delinean los posibles trabajos, que a partir de este diseño pueden ser desarrollados, para nuevos proyectos de tesis de maestría o inclusive doctorado.

DISEÑO Y CONSTRUCCIÓN DE UN MICRO WEB SERVER PARA MONITOREO DE SEÑALES ANALÓGICAS Y DIGITALES

Genaro Sola Ortiz

RESUMEN

El constante crecimiento de los usuarios en el mundo de Internet y la tendencia de la centralización de operaciones en Internet requieren de sistemas de bajo costo que utilicen la infraestructura de red disponible y que provean facilidades de control de sistemas remotos. Como una solución a los problemas de monitoreo de estaciones remotas, se diseñó un sistema mínimo que permita el acceso remoto vía Internet para el control y monitoreo de señales analógicas y digitales. Este trabajo describe el desarrollo de dicho sistema que se conecta a una roseta RJ45 con acceso a una red tipo Ethernet para proporcionar servicios básicos de TCP/IP, lo suficientemente robusto para implementar servicios adicionales sin modificaciones sustantivas en el núcleo principal de programa de conexión. Esto se logra diseñando el núcleo independiente a las aplicaciones que son necesarias utilizar e inclusive a la plataforma de sistema operativo seleccionada. Para la visualización de la información en un navegador, se propone un sistema de servicio de archivos que cumplen con los estándares HTML V2.0 y código de JAVA. Mediante este código se restringe la cantidad de conexiones permitidas al dispositivo, es el responsable de la lectura de sensores y dispositivos o del cambio de estado de los actuadores conectados al sistema. Los canales de comunicación utilizados para el manejo de sensores e información, son RS232, I²C, y un puerto digital de 8 bits. Se proporciona un modo de programación al sistema, para que el cambio en el núcleo del programa sea cuestión de habilitar un interruptor de programación. El puerto RS232 sirve para introducir información que puede ser publicada dentro de la página WEB generada, éste puerto sirve también para introducir los archivos generados HTML y JAVA al sistema de almacenamiento permanente. Todo esto permite un sistema versátil y muy robusto que en un futuro podrá ser habilitado para manejar grandes volúmenes de información, mediante la conexión de algún dispositivo como puede ser un disco duro, un lector de CDROM o inclusive un lector DVDROM.

CONSTRUCTION AND DESIGN OF A MICRO WEB SERVER FOR MONITORING ANALOG AND DIGITAL SIGNALS

Genaro Sola Ortiz

ABSTRACT

The constant worldwide growth of internet users and the centralization trend of internet operations require low cost systems that employing the existing net infrastructure provide remote control capabilities. As a solution to remote monitoring stations problems a system that allows remote access, control and monitoring of analog and digital signals via the internet was designed. This work describes the development of such system that connects to a RJ45 jack with access to an Ethernet like net to provide basic TCP/IP services robust enough to implement additional services without important modifications to the connection program main core. This is achieved designing a core that's independent of the needed applications or even the selected operating system platform. For the display of the information in a browser, a file system service that complies with HTML V2.0 standards and JAVA code, is proposed. This code restricts the amount of allowed connections to the device, it's in charge of reading the sensors, connected devices and changes in the state of the actuators connected to the system. The communication channels used for the management of information and sensors are RS232, I²C and an 8 bit digital port. A system programming mode is provided so that changes to the program core are made enabling a switch. The RS232 port is used to enter the information to be published in the generated WEB page, this port is also used to enter the HTML and JAVA generated files in the permanent storage system. All this allows a versatile and robust system that in the future could be enabled to manage great volumes of information by the means of any device connection as a hard drive, CDROM reader or even a DVDROM reader.

INTRODUCCIÓN

El constante desarrollo tecnológico en los sistemas de cómputo, ha permitido que el término de Internet sea cada vez más común, al grado que se considera que dentro de unos 10 años existirá en todos los hogares al menos una conexión a Internet, ya sea por medio de una computadora personal o utilizando dispositivos dedicados para dicha conexión.

Las principales barreras que presentaban la calidad y la velocidad de conexión de los sistemas remotos, han sido salvadas, y pareciera ser que no existe un límite, por lo que muchas empresas pronostican que Internet será el medio de comunicación masivo de mayor uso dentro de unos cuantos años.

Hoy en día se puede revisar cuentas bancarias, realizar compras, intercambiar información, acceder a bibliotecas en otros países, o incluso realizar visitas virtuales a lugares tan alejados como el polo norte o navegar en el espacio mismo. Todo ello gracias a Internet.

Se puede asegurar que Internet se convertirá, si no lo es ya, en el sistema mundial de comunicación y prestador de servicios. Es por ello que muchas compañías han comenzado a migrar sus sistemas y han desarrollado nuevas y novedosas aplicaciones para Internet.

Los sistemas de control no pueden quedar excluidos de esta tendencia global. Hoy en día se habla de refrigeradores que, basándose en un sistema de inventario de comestibles, pueden ser capaces de elaborar una petición electrónica de consumibles faltantes, en tiendas departamentales virtuales que cuentan con dicho servicio¹.

La tendencia se muestra clara, centralizar prácticamente cualquier servicio y sistema de control en Internet; imagínese, por ejemplo, estar fuera de casa y poder encender las luces de su casa, encender el sistema de riego de su jardín, ver si esta encendida la televisión, un juego de video, o inclusive ver mediante una cámara digital, algunas de las recamaras de su domicilio. Para las industrias, el tener un sistema de control de procesos, con el que puede saber exactamente el número de artículos producidos de un determinado producto, poder obtener las condiciones de temperatura de bodegas, gasto de combustible de algún equipo o maquinaria; sin importar, donde se encuentre, siendo solamente necesario estar conectado a Internet, puede representar un gran ahorro de tiempo y dinero.

Es por estas y muchas otras razones que algunas empresas, se dedican a proporcionar soluciones de control y monitoreo de sistemas remotos.

Muchos de estos desarrollos dependen, además de los dispositivos de control, de una computadora personal que se encuentre conectada a Internet. Esta computadora es la encargada de realizar los enlaces de

¹ LG Appliances <http://www.lgappliances.com/demo.html>

conexión a Internet y mediante el uso de puertos de entrada y salida de la computadora, se pueden controlar cualquier tipo de actuadores.

El presente trabajo intenta cubrir las necesidades de monitoreo y control en lugares donde ya se cuenta con una infraestructura de conexión a Internet por medio de una red Ethernet. No necesita de una computadora personal para realizar el enlace vía TCP/IP, o para establecer los métodos de control y monitoreo. El sistema en sí es un pequeño sistema de cómputo dedicado, que tiene incrustado toda la programación necesaria para realizar la conexión a Internet y manejar los dispositivos analógicos y digitales así como el código necesario para publicar paginas WEB y ejecutar programas en Java.

Las principales ventajas que representa el dispositivo con respecto a los sistemas basados en computadoras personales son los siguientes.

- Bajo costo (aproximadamente 200 pesos).
- Espacio reducido (20 X 15 X 2 cm).
- Fácil de transportar y reconfigurar.
- Puertos de entrada y salida.
- Conexión a red RJ45 10 baseT.
- Canales dedicados de intercambio de comunicación I²C y RS232, 2 canales de 8Bits, y un comparador analógico.

A pesar de la gran versatilidad del dispositivo, es indudable que existen algunos límites para el uso de este sistema.

- 8Kb de memoria de programa tipo Flash, memoria RAM de 512 bytes, 256Kb de memoria EEPROM.
- Si es necesario cambiar alguna de las variables a medir, el sistema tendrá que ser reconfigurado a nivel de lenguaje ensamblador, actividad que tendría que hacerse con personal especializado.
- La programación de la memoria Flash se realiza con un programa independiente al código del sistema.
- El cambio de las direcciones MAC del sistema están incrustadas dentro de una memoria con protocolo 3 wire, que no puede ser programada directamente por el microcontrolador ésta debe ser realizada externamente.

Algunas de las desventajas que se presentan pueden solucionarse generado un compilador para el microcontrolador AVR, que sea capaz de traducir el código HTML que generan programas de alto nivel, e incluir una librería especial generada en Java o C++, para el manejo de puertos de entrada y salida.

Un compilador solucionaría muchos de los problemas presentados aquí. Es oportuno mencionar que este trabajo de tesis forma solamente la base necesaria para el desarrollo de nuevos sistemas independientes, con mayor poder para el intercambio de información con mucho futuro, ya que se plantea el manejo de información masiva donde es factible el control de discos duros, unidades de CDROM, DVDROM, cámaras digitales, servicios de correo electrónico, Ftp, o incrementar los servicios de TCP/IP.

En el capítulo uno se establece el panorama general de Internet desde sus inicios y cambios que ha tenido al paso del tiempo, así como el posible desarrollo y tendencia de este sistema de intercambio de información. Se mencionan los principales servicios del protocolo TCP/IP, sobresaltando aquellos que son proporcionados por el sistema mínimo de conexión a Internet propuesto. Se establece un pequeño marco teórico de los diferentes tipos de servicios y enlaces a Internet, mencionando las principales diferencias entre estos, intentado justificar la elección del sistema Ethernet como sistema base de interconexión a Internet.

En el capítulo dos se establecen las principales directivas que fueron seguidas para desarrollar el sistema mínimo de interconexión. Realizando comparaciones entre los sistemas existentes y mencionado las principales características de cada uno de ellos, así como sus desventajas frente el sistema desarrollado, se establecen las características innovadoras del sistema propuesto.

En el capítulo tres se describe el procedimiento de construcción y las características de ingeniería que se encuentran en el sistema. Se explica el uso de cada uno de los puertos del sistema, ya sean de configuración o de comunicación, así como la manera en que pueden ser reconfigurados y programados. Se describe la forma en que el sistema se debe utilizar y se establecen los límites del sistema. Se detallan las pruebas a las que fue sometido el sistema para establecer algunas características de velocidad de comunicación y saturación. En base a estas mediciones se da un panorama general de rendimiento obtenido con respecto a sistemas existentes.

El capítulo cuatro presenta las conclusiones obtenidas del presente trabajo, se detallan los logros obtenidos y las modificaciones en código de programa y de diseño, necesarias para mejorar el presente diseño. Se delinean los posibles trabajos, que a partir de este diseño pueden ser desarrollados, para nuevos proyectos de tesis de maestría o inclusive doctorado.

Internet

Principios.

Internet no apareció de un día a otro en la vida de los usuarios de computadoras. En los principios, las redes de computadoras se concebían como más de una computadora conectadas entre sí, compartiendo recursos, archivos, programas, o espacio de almacenamiento. Las redes de computadoras podían solamente consultar información que se encontraba dentro del área de trabajo. Existían entonces muchas redes de computadoras independientes entre sí, y cada una de ellas contenía sus propios recursos, a saber, impresoras, enciclopedias en CDRom, escáner, etc.

La tendencia era unir estas redes de computadoras independientes, para generar una red cada vez mas grande y con mayores recursos, ya que cada red de computadoras, compartía algunos recursos o algún tipo de información.

Nadie pensó en esos momentos en el tamaño de esta red de redes, ni mucho menos se pensaba en establecer un protocolo estándar de comunicación de datos. Cada red de computadoras, era establecida en base a las necesidades y posibilidades de cada grupo de trabajo. Incluso las computadoras que estaban conectadas entre sí debían ser lo más similares posibles, para que los programas generados para compartir información, pudieran trasladarse transparentemente de una máquina a otra con el mínimo de modificaciones. Las grandes compañías de computadoras generaban su propia tecnología de comunicaciones en red, exclusiva para el tipo de computadoras que ellas mismas producían. Se establecían inclusive protocolos de comunicación, que se encontraban optimizados para la arquitectura de la red y la tecnología propia de cada compañía. Por esta razón, existían diversos protocolos de comunicación así como diversas maneras de conexión de computadoras.

Actualmente dentro de Internet existen millones de computadoras con diferentes tipos de tecnologías de conexión y diferentes sistemas operativos. Y se considera que para el año 2005, el número de usuarios conectados a esta "red de redes" sea de unos doscientos millones de enlaces.

En términos generales, Internet se puede definir como una red de redes de computadoras, compartiendo muchos tipos de información, servicios y recursos.

Para tener una visión global de Internet, es necesario definir algunos conceptos, que son utilizados en el entorno de Internet.

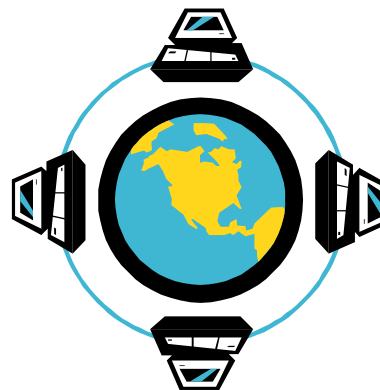


Figura 1. Internet, comunicacion global.

Redes LAN

Una limitante que surge cuando una red de comunicaciones se intenta expandir, es la longitud máxima de cable que puede existir entre una máquina y otra. Aunque esta aseveración tropieza con la necesidad de conexión global de computadoras, es una realidad.

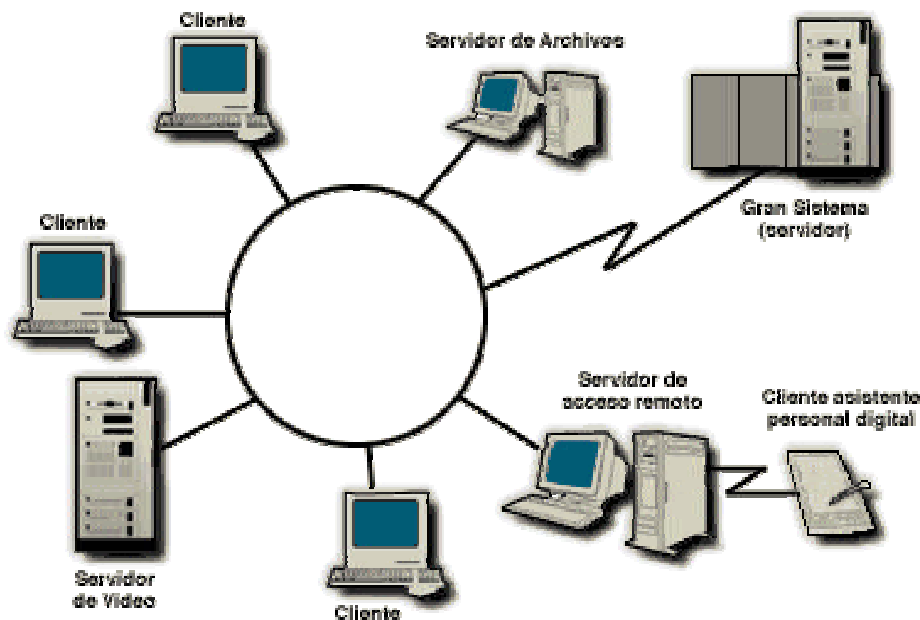


Figura 2. Red de computadoras.

Dependiendo de la tecnología utilizada para conectar las máquinas, se establecen las longitudes máximas de cableado. Cuando se necesita conectar dos computadoras que se encuentran físicamente muy alejadas la una de la otra, se necesitan introducir elementos que permitan levantar las señales, sin que se aumente el nivel de ruido de la señal original transmitida.

Realmente no existe una división específica de cuando una red es llamada local (LAN) y cuando es llamada red amplia (WAN). Pero se intentará establecer algunas condiciones generalmente aceptadas.

LAN. Es una red de trabajo de alta velocidad tolerante a fallos, que cubre un área geográfica relativamente pequeña. Típicamente conecta estaciones de trabajo, computadoras personales, impresoras y otros dispositivos.



Figura 3. Red de computadoras LAN.

WAN.- Es una red de trabajo, que cubre un área geográfica grande, y se considera que conecta una red de trabajo con otra red.



Figura 4. Representación de Red WAN.

Las LANs ofrecen a los usuarios de computadoras muchas ventajas, incluyendo acceso a dispositivos compartidos y aplicaciones, intercambio de archivos entre los usuarios, y comunicación entre los usuarios vía correo electrónico local y otras aplicaciones. En sus primeros días, las redes LAN utilizaban un solo cable por medio del cual todas las computadoras dentro del área se encontraban conectadas. Para poder diferenciar y controlar el flujo de información de una máquina determinada a otra, se establecieron protocolos para la detección de colisiones y disponibilidad de cable o medio.

Una colisión se presenta cuando dos sistemas desean hacer uso al mismo tiempo del medio, los sistemas detectan el estado de colisión y detienen el proceso de envío o recepción de información, durante un tiempo establecido para intentar de nuevo la conexión al medio de comunicación. Dentro del esquema de interconexión fue necesario establecer una serie de protocolos y modelos de comunicación que se tenían que seguir para que la información pudiera llegar hasta el destinatario. Es decir, que además de la información que se quería compartir, era también necesario indicar el destinatario, el tipo de dato que se estaba transmitiendo e inclusive alguna información sobre el medio sobre el que se estaba transmitiendo.

Métodos de transmisión LAN

La transmisión de datos LAN caen dentro de tres clasificaciones: *unicast*, *multicast* y *broadcast*. En cada tipo de transmisión, un paquete sencillo es enviado hacia uno o varios nodos.

En transmisión *unicast*, un paquete es enviado desde origen hacia el destino sobre la red. Primero el nodo origen dirige el paquete usando la dirección del nodo destino. El paquete es enviado entonces sobre la red, y finalmente, la red pasa el paquete hacia su destino.

Una transmisión *multicast* consiste de un paquete de datos que es copiado y enviado a un subgrupo específico de nodos sobre la red. Primero el nodo origen direcciona el paquete, usando la dirección de multicast. El paquete es entonces enviado sobre la red, la cual hace copias del paquete y las envía a cada nodo que forma parte de la dirección de multicast.

Una transmisión de *broadcast* consiste de un paquete de datos que es copiado y enviado a todos los nodos sobre la red. En este tipo de transmisión, el nodo origen direcciona el paquete usando la dirección broadcast. El paquete es entonces enviado sobre la red, la cual hace copias del paquete y envía una copia a cada nodo sobre la red.

Topologías LAN.

La topología LAN define la manera en la cual los dispositivos de la red son organizados. Existen cuatro topologías LAN comunes: Canal, Anillo, estrella y árbol. Estas topologías son arquitecturas lógicas, pero los dispositivos actuales no necesitan estar físicamente organizados en estas configuraciones. Los canales lógicos y topologías de anillo, por ejemplo son comúnmente organizadas físicamente como estrellas.

Una topología de canal es una arquitectura lineal LAN en la cual las transmisiones de las estaciones de la red propagan la longitud del medio y son recibidas por todas las otras estaciones. Una de las implementaciones LAN mas ampliamente usadas son: Ethernet/IEEE 802.3, incluyendo 100BaseT.

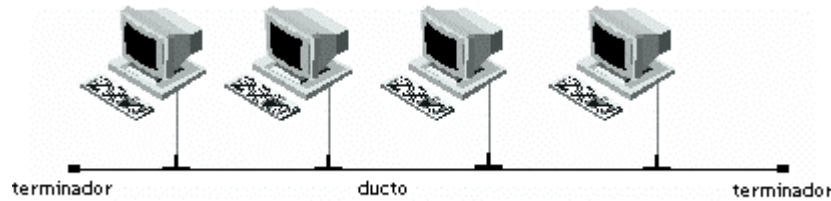


Figura 5. Topología canal o lineal.

Una topología de anillo es una arquitectura de LAN que consiste de una serie de dispositivos conectados uno a otro mediante enlaces de transmisión unidireccional para un ciclo cerrado sencillo. Redes Token Ring /IEEE 802.5 y FDDI implementan una topología de anillo.

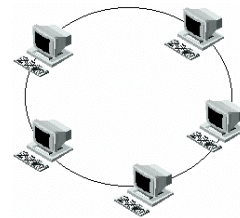


Figura 6. Topología Anillo.

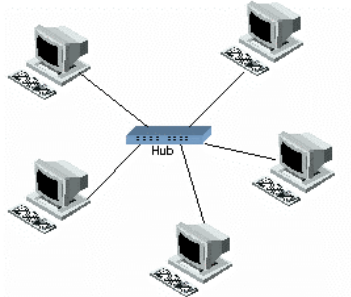


Figura 7. Topología de estrella.

Una topología de estrella es una arquitectura de LAN en la cual los puntos finales de una red son conectados en un concentrador central, o interruptor para enlaces dedicados. Las topologías de canal y anillo a menudo son implementadas en topología de estrella. Una topología de árbol es una arquitectura LAN que es idéntica a una topología de canal excepto que se ramifica con nodos múltiples como sea posible.

Dispositivos LAN

Los dispositivos comúnmente usados en una Lan incluyen repetidores, concentradores, extensores, puentes, conmutadores de paquetes (switchers) y ruteadores.

Un repetidor (o generador) es un dispositivo electrónico que opera sólo en la capa física del modelo OSI (capa 1). Permite sólo extender la cobertura física de una red, pero no cambia la funcionalidad de la misma. Las señales que son recibidas desde un segmento de la red se amplifican, sincronizan y retransmiten hacia otros segmentos de la red. Regeneran una señal a niveles más óptimos, es decir, cuando un repetidor toma una señal muy débil, crea una copia bit por bit de la señal original. La posición de un repetidor es vital, éste se debe poner antes de que la señal se debilite. En el caso de una red local (LAN) la cobertura máxima del cable UTP es 100 metros. Si es necesario ubicar sistemas que se encuentren mas alejados, será pues necesario ubicar un repetidor y teóricamente se podrán alcanzar otros 100 metros a partir de este punto. Existen también regeneradores ópticos conocidos como EDFA (Erbium-Doped Fiber Amplifier) los cuales permiten extender la distancia de un haz de luz sobre una fibra óptica hasta 125 millas.

Los repetidores son incapaces de desarrollar filtrados complejos y otros procesamientos de tráfico; en suma, todas las señales eléctricas, incluyendo ruido eléctrico y otros errores son repetidos y amplificados. El número total de repetidores y segmentos de red que pueden ser conectados esta limitado debido a la sincronización y otros usos.



Figura 8. Repetidor fibra óptica.

Un concentrador es un dispositivo de capa física que conecta múltiples estaciones de usuarios, cada una con un cable dedicado. Las interconexiones eléctricas son establecidas dentro del concentrador. Son usados para crear una red física de estrella mientras que mantenemos la configuración de canal lógico o anillo



de la LAN. Existen concentradores pasivos y activos. Los pasivos sólo interconectan dispositivos, mientras que los activos además regeneran las señales recibidas, como si fuera un repetidor. Un concentrador activo entonces, puede ser llamado como un repetidor multipuertos.

Figura 9. Concentrador.

Un extensor de LAN es un interruptor multicapa de acceso remoto, que conecta un ruteador anfitrión. Los extensores de LAN envían el trafico de todos los protocolos estándar en la capa de red (como IP, IPX, y Apple Talk), y filtran el trafico basado sobre la dirección MAC o el tipo de protocolo de capa de red. Estos extensores de cualquier manera, no son capaces de segmentar el tráfico o crear firewalls de seguridad.

Los puentes operan tanto en la capa física como en la de enlace de datos del modelo de referencia OSI. Pueden dividir una red muy grande en pequeños segmentos. Pero también pueden unir dos redes separadas. Los puentes pueden hacer filtraje para controlar el tráfico en una red. Como un puente opera en la capa de enlace de datos, da acceso a todas las direcciones físicas y a todas las estaciones conectadas a él.

Un puente compara las direcciones contenidas en los paquetes a transmitir contra una tabla de direcciones determinada, estableciendo el segmento al que pertenece una estación determinada para enviarle ese paquete. Un puente también es capaz de conectar dos LANs que usan diferente protocolo (e.g. Ethernet y Token Ring). Esto es posible haciendo conversión de protocolos de un formato a otro.



Figura 10. Puente entre TCP/IP, AppleTalk, DecNet, Netbeui y Ethernet.

Los conmutadores son otro dispositivo de interconexión de capa 2 que puede son usados para mantener el ancho de banda en la red utilizando segmentación. Se configuran para reenviar paquetes a un segmento particular utilizando las direcciones de hardware (MAC).

Debido a que el proceso de envío de paquetes se realiza por hardware el envío de paquetes es más rápido que en un puente.

Los conmutadores pueden ser clasificados en "store-and-forward" y "cut-through", por la manera en que reenvían paquetes al segmento apropiado.

Los conmutadores que emplean la técnica store-and-forward procesan el paquete incluyendo el campo CRC y el direccionamiento del paquete. El paquete es almacenado temporalmente antes de que sea enviado al segmento apropiado. Este tipo de técnica elimina el número de paquetes dañados que son enviados a la red.

Los conmutadores que usan la técnica cut-through son más rápidos debido a que estos envían los paquetes tan pronto la dirección MAC es leída.

Existen conmutadores de paquetes de capa 3 y 4, es decir hacen las funciones que los de capa 2, pero además realizan funciones de enrutamiento (capa 3) y conmutación de voz (capa 4).



Figura 11. Conmutadores de paquetes (switches).

Los ruteadores operan en la capa de red (así como Enlace de Datos y capa física) del modelo OSI. Organizan una red grande en términos de segmentos lógicos. Cada segmento de red es asignado a una dirección así que cada paquete tiene tanto dirección-destino como dirección-fuente.

Éstos son más inteligentes que los puentes (bridges), no sólo construyen tablas de enrutamiento, sino que además utilizan algoritmos para determinar la mejor ruta posible para una transmisión. Los protocolos usados para enviar datos a través de un ruteador deben ser específicamente diseñados para soportar funciones de enrutamiento. IP (Arpanet), IPX (Novell) y DDP (Appletalk Network layer protocol) son protocolos de transporte ruteables. Por ejemplo NetBEUI no es un protocolo ruteable.

Los ruteadores pueden ser de dos tipos:

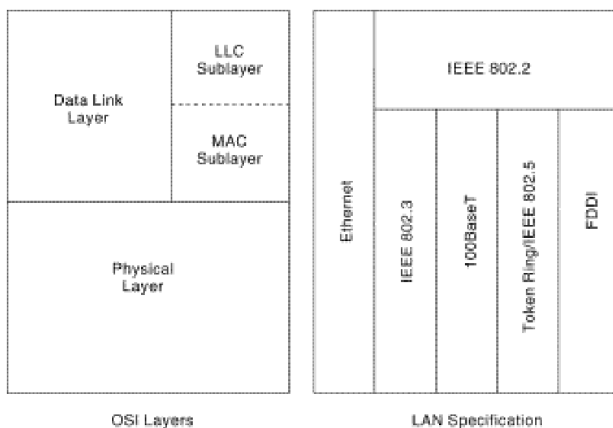
- Ruteadores estáticos: estos ruteadores no determinan rutas. En vez de eso, se debe de configurar la tabla de enrutamiento, especificando las rutas potenciales para los paquetes.
- Ruteadores dinámicos: Estos ruteadores tienen la capacidad de determinar rutas (y encontrar la ruta más óptima) basados en la información de los paquetes y en la información obtenida de los otros ruteadores.



Figura 12. Ruteador

Protocolos Lan y el modelo de referencia OSI.

Los protocolos LAN funcionan en las dos capas mas bajas del modelo OSI (entre la capa física y la capa de enlace de datos). Los protocolos LAN, típicamente usan uno de dos métodos para acceder a la red física media: carrier sense multiple access collision detect (CSMA/CD) y token passing.



En el esquema CSMA/CD de acceso medio, los servicios de red compiten por el uso de la red física media. CSMA/CD es por consiguiente algunas veces llamada “acceso competido”. Ejemplos de LANs que usan el esquema CSMA/CD de acceso medio son redes Ethernet/IEEE 802.3, incluyendo 100BaseT. En el esquema token passing de acceso medio, los dispositivos de red, accesan el medio físico sobre la posesión de un “token”, como lo utiliza token ring/IEEE 802.5 y FDDI.

Figura 12. Especificación LAN dentro de OSI.

Ethernet

El término Ethernet se refiere a la familia de implementaciones de redes área local (Local Network Area LAN) que incluyen tres principales categorías.

- Especificaciones LAN Ethernet y IEEE 802.3 que operan a 10Mbps sobre cable coaxial.
- Especificación LAN sencilla 100Mbps Ethernet, también conocida como “Fast Ethernet”, que opera a velocidades de 100Mbps sobre par trenzado.
- Especificación LAN sencilla 1000Mbps, también conocida como “Gigabit Ethernet”, que opera a 1000Mbps (1Gbps) sobre cables de fibra y par trenzado.



Figura 13. Conectores Ethernet (Coaxial, Par trenzado y fibra óptica).

Ethernet ha sobrevivido como una media esencial tecnológica debido a su tremenda flexibilidad y su relativa simplicidad para implementar y entenderse. A pesar de que otras tecnologías han intentado surgir como reemplazos, los administradores de redes han convertido a Ethernet y a sus derivados como las soluciones efectivas para un amplio rango de requerimientos de implementaciones de campo.

Algunos críticos descartan a Ethernet como una tecnología escalable, pero su esquema de transmisión continua siendo una de las principales características del transporte de datos contemporáneo para aplicaciones de campo².

Un paquete proveniente de un red Ethernet, tiene un formato determinado, un encabezado de 14 octetos de longitud, una trama de datos y una trama al final del paquete de 4 bytes (Frame Check Sequence). El formato del paquete Ethernet se muestra a continuación:

Preámbulo (8)	Dirección destino (6)	Dirección origen (6)	Tipo de servicio (2)	Datos (46 - 1500)	Secuencia de verificación datagrama (4)
---------------	-----------------------	----------------------	----------------------	-------------------	---

Tabla 1. Estructura del paquete Ethernet.

El preámbulo de un paquete Ethernet, es una trama de ceros y unos alternados, que indican la presencia de datos en la estación receptora. El último octeto es el indicador del inicio del paquete de datos Ethernet, para IEEE803.2 este octeto es denominado SOF (Start of Frame), y deben estar definidos los dos últimos bits de éste (0000 0011).

Las direcciones origen y destino, especifican las direcciones MAC, de los sistemas que se están intentando comunicar.

² Introduction to LAN protocols, CISCO Systems Inc. Copyright 1989-1999©

El tipo de servicio especifica el protocolo de un nivel mas alto, que recibirá los datos después de que el procesamiento Ethernet sea concluido. Después de que la capa física y enlace procesen los datos, éstos son enviados a un protocolo de mayor nivel, que ha sido definido por el campo “tipo de servicio”.

El campo de datos tiene una longitud mínima de 46 octetos y una máxima de 1500 octetos. En ocasiones el número de datos dentro de un paquete no es mayor o igual al especificado (46 octetos), por lo que es necesario rellenar con ceros los octetos faltantes, hasta llegar al mínimo permitido. Es importante señalar que dentro de esta sección de datos, pueden viajar otros paquetes de datos, que se utilizan en niveles superiores, tales como IP, ARP o RARP.

El Frame Check Sequence, contiene una serie de octetos (4) que contienen un código CRC, creado por el dispositivo que envía la información y después recalculado por el dispositivo que la recibe. Si la información que llega a una estación, no cumple con el código CRC generado en la estación transmisora, el paquete es totalmente desechado. Esto previene que datos potencialmente dañados, prosigan hacia capas superiores, reduciendo la cantidad de paquetes transmitidos en la red que muy probablemente serán descartados por otras capas.

El uso de un CRC dentro de la trama del paquete, no garantiza que no se haya dañado durante la transmisión pero su uso es generalizado, debido a que reduce la cantidad de datos potencialmente dañados dentro del medio de red y a que el cálculo de éste código no es muy complejo.

Ethernet e IEEE 802.3

Ethernet esta basada sobre la especificación LAN inventada por Xerox Corporation que opera a 10Mbps usando detección de acarreo de múltiple acceso / detección de colisión (CSMA/CD) para ejecutarse sobre cable coaxial. Ethernet fue creada por Xerox en los años 70, pero el término es ahora a menudo usado para referirse a todas las redes locales CSMA/CD .

Ethernet fue diseñado para servir en redes con requerimiento de tráfico ocasional o esporádicamente pesado, la especificación IEEE 802.3 fue desarrollada en 1980 basada sobre la tecnología original Ethernet. La versión Ethernet 2.0 fue conjuntamente desarrollada por Digital Equipment Corporation, Intel Corporation y Xerox Corporation. Compatible con IEEE 802.3.

Ethernet e IEEE 802.3 son usualmente implementadas en una tarjeta de interfase o dentro de un circuito sobre un PCB. Las convenciones de cableado Ethernet especifican el uso de un transmisor para adjuntar un cable hacia el medio físico de red. El transmisor desarrolla muchas de las funciones de la capa física, incluyendo detección de colisión.

IEEE 802.3 proporciona una variedad de opciones de cableado, una de las cuales es una especificación referida a 10Base5. Esta especificación esta cercana a Ethernet. El cable de conexión esta referido a una unidad de interfase adicionada (attachment unit interface AUI), y el dispositivo adjunto a la red es llamado unidad conexión del medio (Media Attachment Unit MAU), en lugar de transmisor.

En el entorno Ethernet de broadcast, todas las estaciones ven todos los paquetes colocados sobre la red. Siguiendo cualquier transmisión, cada estación examina cada paquete para determinar que estación es el destino. Paquetes identificados por una estación de trabajo, son pasados hacia una capa de protocolo mayor.

Bajo el proceso de Ethernet CSMA/CD medio acceso, cualquier estación sobre la LAN CSMA/CD puede acceder a la red en cualquier momento. Antes de enviar los datos, las estaciones CSMA/CD escuchan el tráfico sobre la red. Una estación en espera para enviar datos, aguarda hasta que se detecta que no existe tráfico antes de transmitirlos.

Ethernet permite a cualquier estación en la red transmitir en cualquier momento en que la red este en calma.

Diferencias de servicios entre Ethernet e IEEE 802.3

A pesar de que Ethernet e IEEE 802.3 son muy similares en muchos aspectos, ciertos servicios difieren entre las dos especificaciones. Ethernet proporciona el servicio correspondiente a las capas 1 y 2 del modelo referencia OSI, y IEEE 802.3 especifica la capa física (capa 1) y la parte de acceso al canal de la capa de enlace (capa 2). De hecho, IEEE 802.3 no define un protocolo lógico de control de enlace pero hace diferencias específicas entre las capas físicas, mientras que Ethernet define solamente una. Cada protocolo IEEE 802.3 capa física tiene un nombre en tres partes que resume sus características. La especificación de los componentes en la convención de nombres corresponde a la velocidad de la LAN, método de señalización, y el tipo de medio físico.

Comparación de diferentes especificaciones de capa física IEEE 802.3

Característica	Valor Ethernet	Valores IEEE 802.3				
		10Base5	10Base2	10BaseT	10BaseFL	100BaseT
Velocidad de transmisión (Mbps)	10	10	10	10	10	100
Método de señalización	Baseband	Baseband	Baseband	Baseband	Baseband	Baseband
Longitud Máxima de segmento (m)	500	500	185	100	2,000	1000
Medio	Coaxial 50	Coaxial 50	Coaxial 50	Cable par trenzado sin blindar	Fibra Óptica	Cable par trenzado sin blindar
Topología	Canal	Canal	Canal	Estrella	Punto a punto	Canal

Tabla 2. IEE802.3 Diferentes especificaciones

TCP/IP

El crecimiento exponencial de los usuarios de las redes y la necesidad de intercambio de información entre estos usuarios, obligó a establecer un estándar de comunicaciones que fuese independiente de la tecnología de interconexión, tipo de computadoras, velocidad de conexión o sistema operativo.

El estándar de comunicación entre sistemas de computadoras ha sido y seguramente será durante muchos años el sistema de TCP/IP (conjunto de protocolos Internet). Esto es debido a que TCP/IP se ha diseñado para trabajar en el intercambio de información entre computadoras o entre sistemas de computadoras, sin importar el medio de interconexión, la velocidad de transferencia, la topología de red, el sistema operativo, la arquitectura de la computadora e incluso los programas para dichos intercambios.

TCP/IP, Transmission Control Protocol//Internet Protocol. Cuando la gente menciona TCP/IP, se esta refiriendo al protocolo de pila. TCP/IP es actualmente una grupo de protocolos. Cada protocolo desarrolla una función específica.

La creciente aceptación de TCP/IP puede ser atribuida a diferentes variables, TCP/IP ha estado desde alrededor de principios de los años 70, originalmente desarrollado por el Departamento de la Defensa (DoD Departamento of Defence, Advanced Research Project Agency's Network ARPANET) como el protocolo para correr sobre paquetes intercambiables dentro del área de la red.

Cerca de los años 80, fue distribuido como una parte complementaria de UNIX Berkeley Version 4.2. Debido a que proporcionaba capacidades sobre Internet, soporte de ruteo y amplia disponibilidad, por lo que fue aceptado como estándar para la interconexión de entornos heterogéneos para distribuidores múltiples.

Cuando una organización usa TCP/IP como su protocolo de pila, puede usarlo exclusivamente sobre su propia Internet privada (INTRANET), o en el amplio mundo de Internet.

Internet es una colección de REDES de TRABAJO y puertas de enlace que usan la “suite” TCP/IP.

El protocolo TCP/IP tiene que estar a un nivel superior del tipo de red empleado y funcionar de forma transparente en cualquier tipo de red y en nivel inferior de los programas de aplicación (páginas WEB, correo electrónico) particulares de cada sistema operativo. Todo esto nos sugiere el siguiente modelo de referencia:

Capa de Aplicación (HTTP, SMPT, FTP, TELNET)
Capa de transporte (UDP, TCP)
Capa de Red (IP)
Capa de acceso a la Red (Ethernet, Token Ring, Apple Talk)
Capa física (cable coaxial, par trenzado, AUI)

Figura 15. Modelo IP.

Es conveniente separar el protocolo TCP/IP para su análisis, esto se revisara por separado algunas secciones TCP e IP.

IP. El protocolo Internet

La Cabecera IP

La cabecera de Internet contiene además de las direcciones origen y destino, una serie de parámetros que son esenciales para la operación de los ruteadores. Observe el siguiente datagrama.

4-bits Versión	4-bits longitud	8-bits tipo de servicio (TOS type of service)	16-bits longitud total (en bytes)	
16-bits Identificación			3-bits banderas	13-bits desplazamiento fragmentación
8-bits tiempo de vida (TTL Time to Live)		8-bit Protocolo	16-bits verificación del encabezado	
32 bits Dirección IP de origen				
32 bits Dirección IP de destino				
opciones (si existen)				
datos				

Tabla 3. Encabezado IP.

La cabecera comprende varios campos fijos, presentes en cada paquete, así como varias opciones. Los campos son generalmente ordenados en grupos de 4 octetos es decir 32 bits de datos; en cada palabra de 32 bits, se transmite primero el bit mas significativo. El primer campo es el identificador de la versión IP actual, generalmente se usa la versión 4 de IP, versión 5 es usada para algunos entornos de “tiempo real”, las versiones 6 a la 8 de “nueva generación” para usos futuros.

El segundo campo comprende la longitud del encabezado expresado en número de palabras de 32 bits, el mínimo de palabras que se puede utilizar es precisamente el mínimo número de palabras para un encabezado IP, es decir 5 palabras (longitud de 20 octetos). La longitud máxima de un encabezado IP es de 15 palabras (60 Octetos), incluyendo el encabezado. Si se reservan 20 octetos para el encabezado IP, se puede concluir que un paquete IP puede utilizar 40 octetos de opciones.

El tipo de servicio define la precedencia del paquete y el tipo de enrutamiento deseado. Como se puede apreciar este campo en realidad comprende dos subcampos: la precedencia y el tipo de servicio. La precedencia es una indicación de prioridad. De acuerdo a estas consideraciones 111-Control de Red, 110 Control interno de red, 101 ECP critico, 100 Control rápido, 011 Rápido, 010 Inmediato, 001 Prioritario, 000 Rutinario.

La precedencia no afecta al enrutamiento pero puede retrasarlo si existen varios paquetes dentro de un canal que esperan ser transmitidos; en teoría el paquete con mayor prioridad debe ser transmitido.

0	1	2	3	4	5	6	7
PRECEDENCIA			Tipo de Servicio				
			D	T	R	C	

Tabla 4. Precedencia y Tipo de servicio.

El tipo de servicio es una indicación de enrutamiento, esto quiere decir que para alcanzar un destino pueden existir diversos tipos de enlace: satelitales, radio frecuencia, líneas telefónicas, etc. El tipo de servicio puede definir de alguna manera el tipo de enlace requerido en base a el ancho de banda designado y el costo del enlace. Esto significa que un enlace satelital es sin lugar a dudas el más rápido, pero de igual manera es el mas caro. Si se define el bit 3 “D” significa que deseamos un enlace con bajos tiempos de retardo. Definir el bit 4 “T” indica seleccionar la ruta con el ancho de banda mas alto evitando cualquier tipo de enlace telefónico. Bit 5 “R”, pide alta confiabilidad del enlace, quizá evitando enlaces via radio que pueden ser muy ruidosos y no confiables. Bit 6 “C”, define la ruta mas barata de enlace. En la mayoría de los protocolos no se define mas de un bit para tipo de servicio y la ruta se calcula usando un enrutamiento normal (ningún bit definido), la ruta mas corta, el mayor ancho de banda o la ruta mas confiable, note que las combinaciones de C y T por ejemplo no están definidas, de hecho esto es ilegal.

La longitud total es el número de octetos contenidos en el paquete (incluyendo los 20 octetos del encabezado IP), este campo tiene una longitud de 16 bits. Un paquete IP puede tener una longitud de $2^{16}=65535$ Octetos. Una nota importante es que la resta de la longitud total del paquete IP y la longitud del encabezado por 4 (Total Length - IHL*4), nos da la información exacta de la longitud de los datos presentes en el paquete IP. Si esta resta es cero, podemos afirmar que no existen datos dentro del paquete IP.

Los campos identificación, banderas y el Desplazamiento de fragmentación, son usados para el procedimiento de fragmentación y procedimiento de reensamble. La fragmentación de paquetes es un método que permite dividir paquetes de información en pequeños trozos. La fragmentación es necesaria, por que a través del camino que recorre nuestro paquete puede encontrar diversos medio de transmisión, algunos con mayor o menor ancho de banda. La cantidad de bits que pueden transmitirse por segundo en algún medio en específico, se conoce como MTU (Media Transmisión Unit). Durante la división de un paquete en trozos, un campo que siempre se ha de conservar durante la fragmentación, es el campo de identificación. El campo de longitud se verá reducido de acuerdo al tamaño máximo de transmisión o MTU del medio en turno.

El campo de banderas DF y MF definen el estado de la fragmentación, esto es si el bit DF está definido esto significa que el paquete no se puede fragmentar (Don't fragment). Si el paquete llega a un medio con un MTU mas bajo que el tamaño del paquete, este simplemente se eliminará y el ruteador enviará un mensaje de error tipo ICMP. La bandera MF (Most Fields), indica que el paquete actual es un fragmento de un paquete mayor. El campo de offset define el desplazamiento del fragmento dentro del paquete original.

Lo anterior puede ejemplificarse suponiendo que un paquete de longitud 9020 llega a un medio con MTU de 3000, el paquete se fragmentará de acuerdo a la siguiente tabla.

Paquete de entrada	Campos del encabezado IP					Campo de datos
	Id=X	L=9020	DF=0	MF=0	Offset=0	1---12---23---3
Frag. 1	Id=X	L=3020	DF=0	MF=1	Offset=0	1---1
Frag. 2	Id=X	L=3020	DF=0	MF=1	Offset=3000	2---2
Frag. 3	Id=X	L=3020	DF=0	MF=1	Offset=6000	3---3

Tabla 5. Fragmentación de un paquete.

Note que si se suma la longitud de los fragmentos, el resultado no es la longitud original del paquete ($3020+3020+3020=9060 \neq 9020$); esto se debe a que cada fragmento deberá contener el encabezado IP original, con las respectivas modificaciones en los campos checksum, longitud, precedencia y servicio.

Un fragmento puede a su vez subdividirse en nuevos fragmentos. Supongamos que los fragmentos que se enviaron anteriormente, llegan a un medio con un MTU de 1000 que es mas bajo que la longitud del fragmento (3000).

Frag 2	Campos del encabezado IP					Campo de datos
	Id=X	L=3020	DF=0	MF=1	Offset=3000	2---2
Frag. 2.1	Id=X	L=1020	DF=0	MF=1	Offset=3000	2--
Frag. 2.2	Id=X	L=1020	DF=0	MF=1	Offset=4000	---
Frag. 2.3	Id=X	L=1020	DF=0	MF=1	Offset=5000	--2

Tabla 6. Proceso de fragmentación de un paquete fragmentado.

El tiempo de vida (TTL.- Time to Live) fue definido inicialmente como un indicador del tiempo en segundos máximo que existirá un paquete en la red.

La RFC-791 especifica que los ruteadores deben siempre decrementar el TTL cuando se retrasa un paquete, el decremento es de uno si el tiempo de espera en la cola y el siguiente salto, es menor de un segundo, o decrementar en el número de segundos estimado de espera.

Pero la mayoría de los ruteadores decrementan siempre en uno el TTL. Si el valor de TTL=0 el paquete es destruido, y no retrasado.

El campo de *protocolo* se utiliza para determinar el programa de mayor nivel al que le serán enviados los datos del paquete.

La tabla 7 nos da una idea de los diferentes tipos de protocolos manejados dentro del paquete IP.

Decimal	Protocolo	Detalle
0		Reservado
1	ICMP	Internet Control Message
2	IGMP	Internet Group Management
3	GGP	Gateway-to-Gateway
4	IP	IP dentro de encapsulación IP
5	ST	Stream
6	TCP	Transmission Control
8	EGP	Exterior Gateway
17	UDP	User Datagram
29	ISO-TP4	ISO Transport Protocol Class 4
38	IDPR-CMTP	IDPR Control Messenger Transport
80	ISO-IP	ISO Internet Protocol (CLNP)
88	IGRP	IGRP
89	OSPF	Open Shortest Path First
255		Reserver

Tabla 7. Protocolos superiores a IP.

EL checksum, es un indicador, de que la información contenida dentro del paquete se ha recibido sin errores. Este se calcula como el complemento a uno de la suma de 16 bits de los complementos a uno de todas las palabras de 16 bits en el encabezado. Esta no es una fuerte protección ya que permitiría la permutación de palabras de 16 bits y la adición de palabras con valor cero. La experimentación muestra que existe un adecuado balance entre fácil computo y eficiencia de detección de errores. Este mismo algoritmo es usado en otros protocolos como UDP, TCP, ICMP, u OSPF. Si el checksum falla todo el paquete es descartado. Si un solo bit cambia (por ejemplo el decremento del TTL) el calculo del checksum debe realizarse nuevamente antes de cualquier transmisión.

Los dos siguientes campos son la dirección de hardware de origen y destino, que ya se han detallado anteriormente.

Un paquete IP puede o no contener opciones; cuando las contiene, éstas pueden contener varios parámetros opcionales, concatenados unos con otros. Cada parámetro es identificado por un octeto de “tipo de opción”, que a su vez contiene varios campos.

0	1	2	3	4	5	6	7
C	Clase		Número				

Tabla 8. Octeto de opciones IP.

La bandera Copia (c) se define cuando la opción debe ser copiada en cada sub-paquete durante un proceso de fragmentación. De no ser así, solamente se copiará en el primer fragmento. Las opciones se agrupan por clases y estas a su vez pueden ser de “control” o de “medición e inspección”. Las opciones son raramente usadas hoy en día, la mayoría de ellas son obsoletas.

ICMP

El Servicio ICMP (Internet Control Message Protocol), se establece para el intercambio de información sobre las dificultades de ruteo con paquetes IP o intercambios simples como son las peticiones de eco y respuesta de eco.

El formato del encabezado ICMP, se muestra a continuación:

tipo (0 u 8)	código (0)	suma de verificación (checksum)
Identificador		número de secuencia
datos opcionales		

Tabla 9. Encabezado ICMP.

El primer octeto del encabezado ICMP, corresponde con el tipo de servicio requerido: Una aplicación generalmente utiliza el valor en el tipo de servicio 8 para un servicio de petición de eco, y cero para respuesta de eco dependiendo del tipo de servicio ICMP requerido, en ocasiones es necesario establecer un código adicional para el servicio, en el caso de eco y respuesta de eco, el campo code, deberá estar definido en cero.

El campo de checksum se calcula de igual manera que los checksum de los encabezados TCP e IP. Los campos de identificador y número de secuencia, para el caso de una petición de eco, son generados al azar, y sirven para identificar al paquete de información, para la computadora anfitrión y huésped.

En el caso de un paquete de respuesta los números de secuencia e identificador, deben corresponder con los que fueron enviados por la máquina que realizó la petición de eco.

La máquina que emite una petición de eco puede enviar en la sección de datos, información en cierta secuencia junto con la petición de eco. El sistema que recibe la petición, debe incrustar esa secuencia de datos en su respuesta en la misma secuencia en que fueron obtenidos.

La respuesta a la petición debe contener además de los mismos números en los campos de identificador y secuencia, los datos emitido con la petición original, si un solo dato de estos se pierde o se altera durante el trayecto, todo el paquete es rechazado y se considera que no existió respuesta adecuada para la petición, por lo que es necesario remitir una nueva petición.

ICMP es utilizado principalmente para enviar información, acerca de el éxito o falla de los paquetes de información enviados a través de la red. Cuando un paquete de información no llega a su destino, los ruteadores se encargan de transmitir un mensaje ICMP, incluyendo el tipo de error que se generó en la red. Por ejemplo, “no se puede alcanzar la red de destino”, “host no existente”, o “puerto no se puede alcanzar”.

Cuando se genera la petición de eco se inicializa un contador, que mide el tiempo de respuesta de dicha petición. Sabemos que cualquier paquete en tránsito de red, tiene un tiempo de vida establecido. Si la respuesta excede este tiempo de vida, el paquete es descartado por la red y ésta envía un mensaje ICMP de error.

El sistema que origina la petición, al recibir el mensaje de error, considera que no existió respuesta por parte del destino y se considera perdido el paquete. En algunas ocasiones, el mensaje ICMP de error generado por la red no llega al origen, para ello se utiliza el contador que se inicializo al principio de la transmisión, cuando este contador sobrepasa el tiempo de vida del paquete y no existe mensaje ICMP de parte de la red o respuesta del destino, se considera que el paquete en tránsito no tiene respuesta, el paquete es desechado por el dispositivo.

Un resumen de mensajes ICMP, se puede observar en la siguiente tabla.

Tipo	Código	Descripción
0	0	Respuesta de Eco
3		Destino inalcanzable
3	0	Red inalcanzable
3	1	Host inalcanzable
3	2	Protocolo inalcanzable
3	3	Puerto inalcanzable
3	4	Necesaria fragmentación y DF definida
3	5	Falla en Ruta origen
4		Saturación en origen
5		Redirigido
5	0	Datagramas redirigidos por el area de trabajo
5	1	Datagramas redirigidos por el Host
5	2	Datagramas redirigidos por el tipo de servicio y red de trabajo
5	3	Datagramas redireccionados por el tipo de servicio y respuesta del Host
8	0	Eco
11		Tiempo Excedido
11	0	Tiempo de vida excedido en tránsito
11	1	Tiempo de reensable de fragmento excedido
12		Problema de parámetro
13		Marca de tiempo
14		Respuesta marca de tiempo
15		Información de petición
16		Información de respuesta

Tabla 10. Mensajes ICMP.

Como se puede observar los mensajes ICMP, son cortos y en la mayoría de los casos solamente es necesario revisar los campos tipo y código.

TCP

El Protocolo TCP (Transmisión Control Protocol) proporciona un flujo confiable de servicio de entrega y conexiones virtuales con retransmisión de paquetes, cuando sea necesario. La estructura del encabezado TCP es la siguiente:

Puerto de origen					Puerto de destino				
número de secuencia									
número de reconocimiento (acknowledgement)									
desplazamiento		Reservado	U	A	P	R	S	F	Ventana
Suma de verificación					puntero Urgente				
Opciones + relleno (<i>padding</i>)									

Tabla 11. EncabezadoTCP

Puerto origen.- Número de puerto de origen.

Puerto Destino.- Número de puerto destino.

Número de secuencia.- El número de secuencia del primer octeto de datos en este segmento (excepto cuando la bandera SYN este presente). Si SYN esta presente, el número de secuencia es el número inicial de secuencia (ISN) y el primer octeto de datos es ISN+1.

Número de reconocimiento.- Si el bit de control ACK esta definido, este campo contiene el valor del siguiente número de secuencia con el cual el origen del segmento esperará recibir. Una vez que la conexión sea establecida, este valor siempre es enviado.

Desplazamiento (4 bits).- El número de palabras de 32 bits en el encabezado TCP, el cual indica donde los datos inician. El encabezado TCP tiene una longitud que debe ser un número múltiplo de 32 bits.

Reservado (6 bits).- Reservados para usos futuros. Debe ser cero.

Bits de control (6bits).-

- U (URGENT).- Campo puntero urgente.
- A (ACK).- Campo de reconocimiento.
- P (PSH).- Función Push.
- R (RST).- Resetear conexión.
- S (SYN).- Sincronizar números de secuencia.
- F (FIN).- No mas datos del Emisor.

Ventana(16 bits).- El número de octetos de datos con los cuales el emisor de este segmento esta dispuesto a aceptar, comenzando con el octeto indicado en el campo de reconocimiento (Acknowledgment).

Suma de verificación (16 bits).- Es el complemento a uno de la suma de los complementos a uno de todas las palabras de 16 bits en el encabezado y texto. Si un segmento contiene un número impar de octetos en el encabezado se debe adicionar un octeto conteniendo exclusivamente ceros y debe ser sumado. Obviamente el octeto relleno de ceros no será transmitido como parte de este segmento.

Puntero Urgente (16 bits).- Este campo comunica el valor actual del puntero como un desplazamiento positivo desde el número de secuencia del siguiente octeto de datos urgentes. Este campo solamente puede ser interpretado en segmentos en los cuales el bit de control URG ha sido definido.

Options.- Pueden ser transmitidas al final del encabezado TCP y siempre tienen una longitud múltiplo de 8 bits. Todas las opciones deben ser consideradas en el campo de checksum. Una opción puede comenzar en los límites de cualquier octeto. Hay dos formatos posibles para una opción:

- Un solo octeto de tipo de opción.
- Un octeto de tipo opción, un octeto de longitud de opción

La opción de longitud incluye el tipo de opción y la longitud de opción.

Datos.- El grupo de octetos que serán enviados a un protocolo de mayor nivel. Dentro de este campo, pueden existir nuevos paquetes de información para alcanzar otros protocolos o solamente ser datos que serán procesados dentro de TCP.

Máquina de estados TCP

Se ha mencionado que TCP es un protocolo orientado a conexiones, que se abren o cierran, basándose en las necesidades de comunicación.

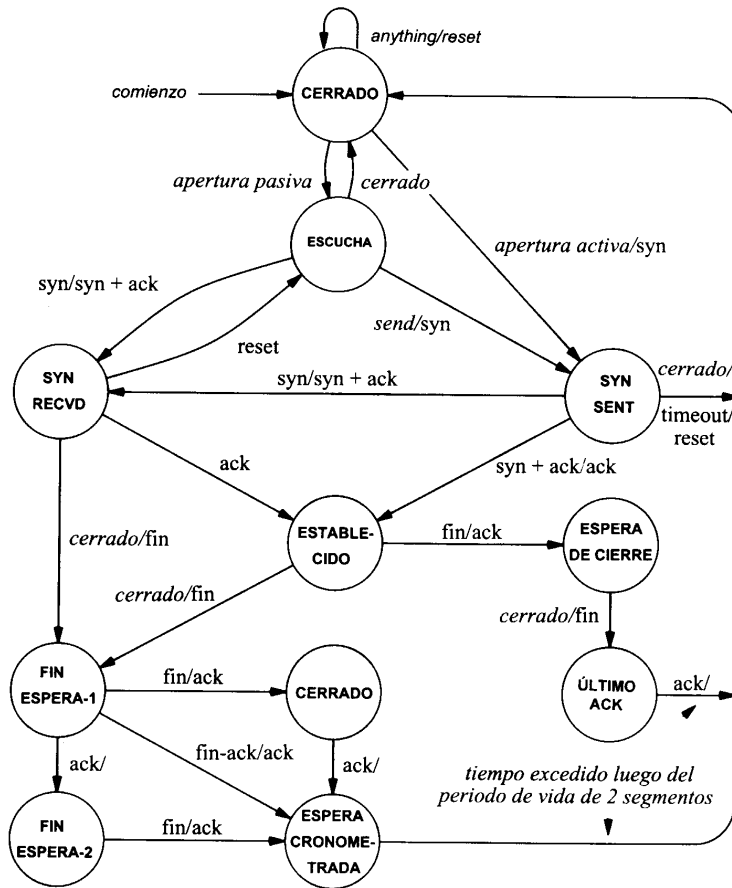


Figura 14. Máquina de estados del protocolo TCP/IP

El estado de la conexión entre dos elementos de la red está definido en el paquete IP, dentro del campo de banderas (flags). En este campo se establecen todos los estados de la conexión y permiten a la aplicación o al usuario terminar o establecer una conexión.

Para establecer una conexión, el programa de aplicación deberá emitir un comando de apertura pasiva (para esperar una conexión desde una máquina) o un comando apertura abierta (para iniciar conexión). El comando *active open* obliga a que se de una transición del estado CLOSED al estado SYN SENT. Cuando TCP continúa con la transacción, emite un segmento SYN. Cuando el otro segmento devuelve un segmento que contiene un SYN, más un ACK, el TCP cambia al estado ESTABLISHED y comienza la transferencia de datos.

El estado TIMED WAIT revela como TCP maneja algunos de los problemas que se presentan con la entrega no confiable. El TCP conserva la noción de máximo tiempo de vida del segmento, el tiempo máximo en que un segmento puede mantenerse activo en una red de redes.

TCP establece el inicio y terminación de las conexiones, establece un autómata o máquina de estados, para conocer el estado de la conexión. También proporciona herramientas adecuadas para la terminación prematura de dicha conexión o la terminación en base a petición de los usuarios.

La figura 14 muestra completamente la máquina de estados del protocolo TCP/IP.

Un sistema que pretende proporcionar un servicio TCP/IP deberá tener en cuenta todos y cada uno de los estados de esta máquina. Pero no necesariamente debe establecerlos dentro de la aplicación desarrollada.

Conexión y terminación

Se ha mencionado que TCP es un protocolo orientado a las conexiones, y antes de que se pueda enviar cualquier tipo de información, es necesario que una conexión sea establecida entre las máquinas que desean comunicarse.

Resumiendo el protocolo necesario para la apertura de una conexión, se intentará describir en tres pasos:

1.- La máquina que desea establecer la conexión, debe generar un número de secuencia aleatorio, que se enviará a la máquina destino, dentro del paquete TCP. En este paquete, la bandera de control SYN, deberá ser activada para notificar que se desea establecer una conexión por primera vez. Es trabajo de la máquina que desea establecer la conexión, especificar el número de puerto (servicio) que desea utilizar.

2.- El servidor responde con su propio segmento SYN, es decir número de secuencia y de reconocimiento (ACK) generado por el destino. Como parte de la respuesta, el destino envía el número de secuencia enviado por el origen pero incrementado en uno.

3.- El cliente debe reconocer el número de secuencia enviado por el servidor (el número que envía el servidor, deberá ser el mismo que él envió incrementado en uno). Una vez verificado el número de reconocimiento, se responderá con el número de ACK enviado por el servidor incrementado en uno.

Es recomendable mencionar que los números iniciales de secuencia generados por las máquinas que desean comunicarse, sean aleatorios, esto es debido a que TCP permite establecer varios hilos de comunicación al mismo tiempo.

Estos números de secuencia, establecen dentro del mundo de Internet los llamados *sockets*³, que pueden permanecer abiertos durante una conexión y ser descartados en cuanto el sistema o el usuario determine terminar la conexión (descartar el socket).

Los sockets permiten a las aplicaciones manejar el flujo de información, como si fueran puertos de entrada/salida de archivos. Cuando el flujo de información entre los sistemas es continuo, se considera que el flujo de información se encuentra establecida en un servicio orientado a conexiones. En caso de que sea necesario establecer un socket y destruirlo cuando se termina la transferencia se habla de un socket orientado a datagramas, que normalmente se utilizan en el protocolo UDP.

UDP a diferencia de TCP, es un servicio sin conexiones y no garantiza que los paquetes llegarán en alguna forma en particular, o sea que pueden perderse, duplicarse o incluso llegar en desorden.

³ *Socket*. Conexión establecida entre dos sistemas conectados a través de Internet.

TCP/IP y OSI

La siguiente figura muestra como los servicios TCP/IP son utilizados dentro del modelo OSI.

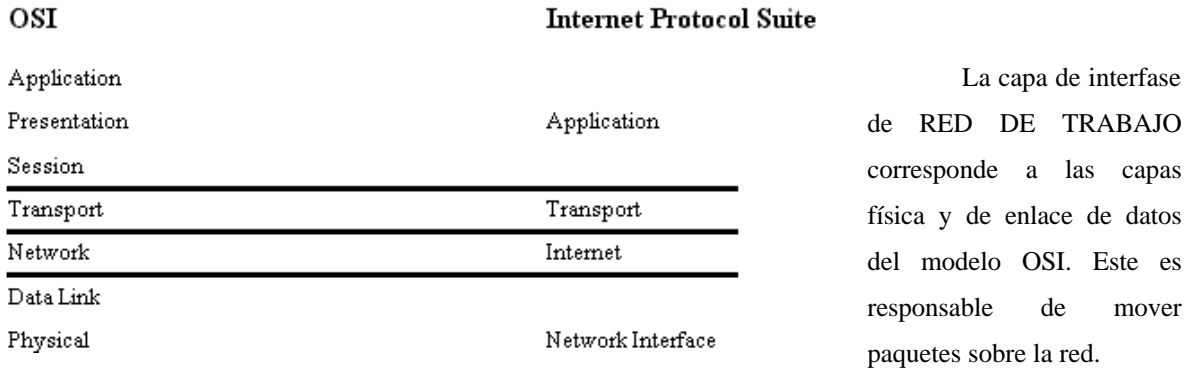


Figura 16. Servicios TCP/IP en OSI

La capa de Internet corresponde con la capa de RED DE TRABAJO del modelo OSI. La capa de transporte es responsable de proporcionar conexión orientada y conexión bajo comunicación entre dos HOSTs. La capa de aplicación direcciona aplicaciones como TELNET, FTP y SNMP.

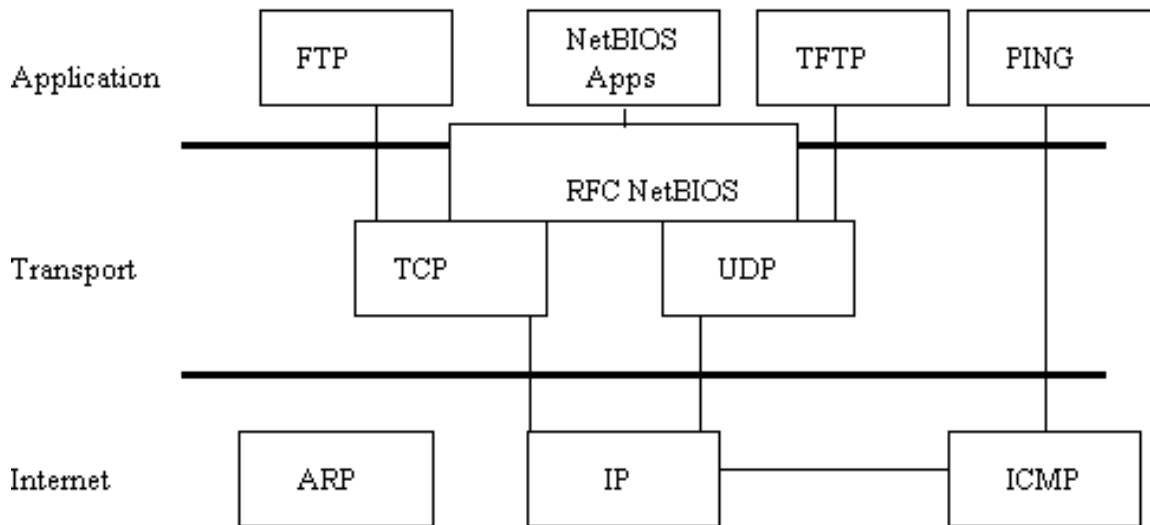


Figura 17. Capas de aplicación y transporte para diferentes servicios IP.

Ethernet/IP/TCP.

Un paquete de datos proveniente de una red Ethernet, puede derivarse en diferentes protocolos de nivel superior. Como ya se ha especificado el paquete Ethernet puede ser un dirigido al protocolo IP, y el paquete de datos IP dirigido a su vez a TCP, y el TCP hacia alguna aplicación específica. La siguiente figura muestra esta encapsulación de datos y su paso desde la red Ethernet hasta la aplicación del usuario.

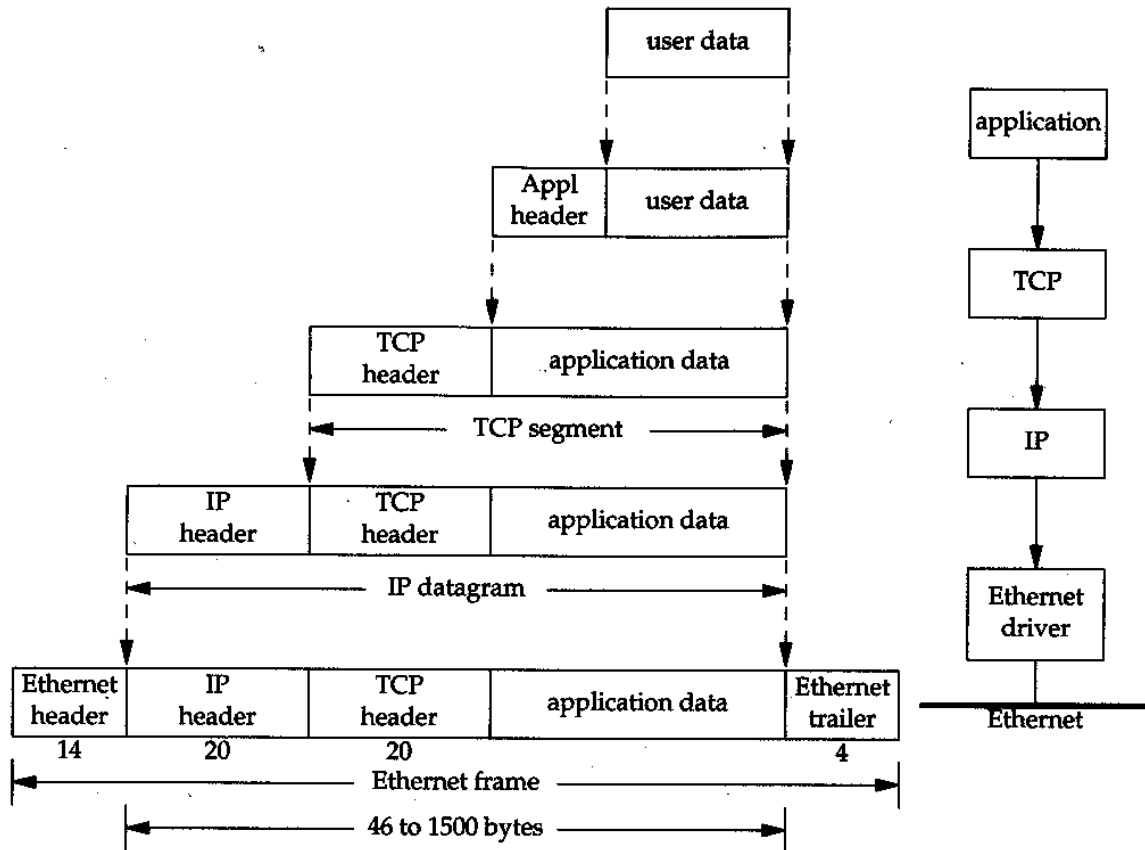


Figura 18. Encapsulación de paquetes.

Observando la figura 18, de manera ascendente, se puede apreciar la forma en que los datos de un protocolo inferior, contienen a los encabezados y datos de capas superiores. En este caso Ethernet/IP/TCP/Aplicación usuario. Cada uno de los encabezados, contiene la información necesaria de control y direccionamiento hasta llegar a la sección donde los datos llegan a la aplicación de usuario.

Como se ha mencionado, el paquete Ethernet puede ser direccionado, no solamente hacia un protocolo IP, como se aprecia en la figura 19. Un paquete Ethernet puede ser enviado hacia un protocolo superior ARP, RARP o IP. Si el protocolo superior es ARP o RARP, los datos son aquí mismo procesados y enviados de nuevo a la red.

Si el paquete requiere de un procesamiento IP, el paquete puede a su vez dividirse en ICMP, IGMP, TCP y UDP. Para ICMP e IGMP los paquetes se procesan y la información pertinente enviada a la red. TCP y UDP requieren un procesamiento adicional.

Aunque ambos llevan datos hasta aplicaciones de usuario, ambos protocolos requieren de procesamientos totalmente diferentes. Se ha mencionado que TCP es un protocolo orientado a conexiones, por lo que antes de que la aplicación de usuario comience a recibir datos, será necesario establecer una conexión, como se ha indicado anteriormente.

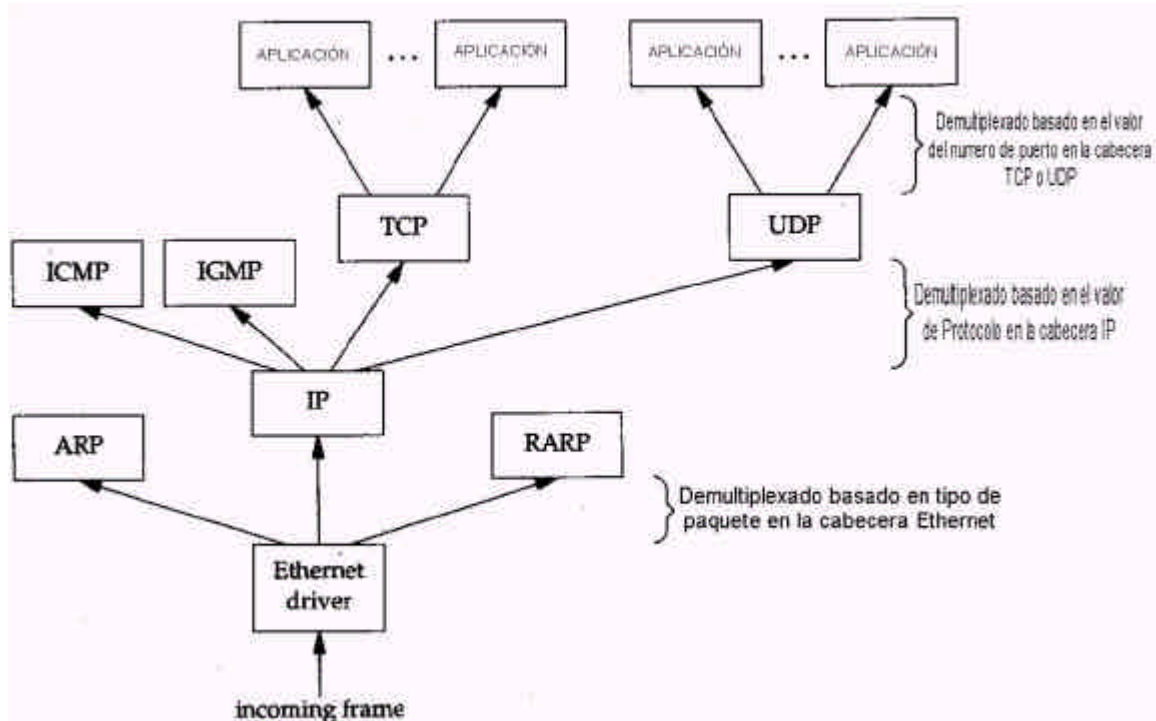


Figura 19. Demultiplexado de un paquete.

Un paquete de entrada, es enviado al controlador Ethernet, que verifica que los datos lleguen sin errores antes de pasarlo a la capa superior. Pasada la revisión de errores, el paquete puede ser enviado a un controlador ARP, RARP o IP, dependiendo del tipo de servicio requerido.

Si el paquete de entrada requiere servicio IP, este a su vez puede derivarse hacia varios protocolos ICMP, IGMP, TCP o UDP. Para el caso de un paquete TCP, este se pasa a un nuevo proceso, que verifica su integridad y extrae los datos necesarios para la aplicación destino.

Control de direcciones Internet.

Cada computadora dentro de un entorno de red debe tener un identificador único que le permita recibir información proveniente de la red. El sistema de direccionamiento IP se encuentra basado en un sistema de direcciones de 32 bits, es decir, si una computadora desea enviar información a otra computadora, solo será necesario especificar esta dirección.

Estos 32 bits de direccionamiento se agrupan grupos de 8, para facilitar su manejo; es decir, cuatro números, expresados en formato decimal, por ejemplo 148.204.103.2 (servidor DNS del IPN). Para los usuarios, es difícil recordar varios grupos de direcciones, no es fácil, por lo que normalmente estas direcciones se especifican con palabras, todas ellas separadas por puntos, como ejemplo “apollo.telecom.ipn.mx” que es el nombre relacionado a la dirección de un servidor DNS del IPN.

Las dos formas de especificar la dirección de una máquina, ya sea con números o con palabras, es válida, pero téngase en cuenta que una computadora solo entiende números. Cuando una dirección de una computadora es especificada por medio de letras en vez de números, una máquina dentro del entorno de red, es la encargada de traducir la dirección especificada con palabras en una dirección especificada en números.

Esta máquina tiene el nombre de DNS (Domain Name Server), y en ella, reside una base de datos donde se encuentran relacionadas, un grupo de direcciones con sus respectivos nombres IP. Una máquina podrá ser direccionada utilizando palabras en lugar de números, si dentro del DNS, se encuentra especificada dicha dirección numérica y alfabética. Es importante mencionar que en el DNS, solamente existen las direcciones IP relacionadas con sus nombres IP, de las máquinas que conforman una red y las direcciones de otros DNS.

Cuando se quiere conocer el número IP de una computadora a partir de su nombre, pregunta al DNS local. Si la información no existe dentro de la base de datos del DNS, éste pregunta a los DNS que se encuentran en su base de datos, por dicha relación. Si la relación nombre/número IP existe, algún DNS responderá enviando dicha relación al servidor o computadora que realizó la petición.

Los números que pueden ser asignados a una computadora, no pueden ser aleatorios, ya que este sistema de direcciones, debe ser único para cada computadora que se encuentra conectada a una red. Existe una entidad normativa que se encarga de asignar rangos de direcciones disponibles, basándose en una serie de criterios.

Primero que nada el sistema de direccionamiento, se asigna de tal manera que el ruteo de la información, sea eficiente. Cada dirección IP nos muestra información acerca del tipo de red y del anfitrión o computadora que desea transmitir/recibir información. Este par indica el tipo de red y el identificador del anfitrión. La cantidad de bytes asignados para el netId y el hostId, es variable y depende de el tamaño de la red.

Obsérvese la tabla siguiente:

	0	1	2	3	4	8	16	24	31	
Tipo A	0	NETID				HOSTID				
Tipo B	1	0	NETID				HOSTID			
Tipo C	1	1	0	NETID				HOSTID		
Tipo D	1	1	1	0	Dirección de Multidifusión					
Tipo E	1	1	1	1	0	Reservado para uso posterior				

Tabla 12. Tipos de direcciones en Internet

A los tipos de direcciones A, B y C se le conoce como tipos primarios de direcciones IP. El tipo de direcciones A, puede ser fácilmente detectada, por que todas sus direcciones tienen un cero en el bit 0 del grupo de dirección.

Para redes tipo A, pueden tener teóricamente hasta $2^{24} = 16,777,216$ anfitriones y como mínimo deberán tener $2^{16}+1 = 65,537$ anfitriones. Las redes Tipo B contendrán entre 65,536 y 257 anfitriones, las redes Tipo C contienen entre 256 y 1 anfitriones. Este arreglo de direcciones facilita a los ruteadores la fácil extracción de información del campo NETID para una mayor velocidad en el direccionamiento de información.

Cuando el campo de HostId es igual a cero, la dirección se encuentre referenciada a la red misma y no a algún anfitrión. Existe una dirección dentro de este esquema, que permite referirse a todos los anfitriones de la red, a esta dirección se el conoce como dirección de difusión. Esta dirección se puede reconocer por que todos los bits del campo HostID están a uno.

ARP(Address Resolution Protocol).

El protocolo de resolución de direcciones, es el encargado de transformar las direcciones de Internet en direcciones Físicas o de acceso medio (MAC). El protocolo se presenta como una solución para diversos sistemas de redes o tecnologías (Ethernet, ProNet, TokenRing). Es importante tener en cuenta que únicamente dentro del sistema de Internet se utilizan direcciones de 32 bits, mientras que las direcciones físicas en Ethernet utilizan un esquema de 48 bits. Los fabricantes de tarjetas Ethernet, normalmente graban permanentemente estas direcciones dentro de una memoria.

El sistema de resolución de direcciones relaciona las direcciones Ethernet (físicas) con las direcciones IP, mediante el uso de mensajes de difusión, este consiste en realizar una petición al sistema remoto mediante la difusión de su dirección IP, solicitando que la máquina con dirección IP especificada, responda enviando su correspondiente dirección física. Hay que notar que en un mensaje de difusión todas las máquinas pertenecientes a la red local reciben el mensaje, pero solamente una de ellas debe responder con su dirección física.

Este procedimiento llamado ARP dinámico, genera una pequeña carga de trabajo adicional a la red, debido a los mensajes de difusión deben alcanzar todas las máquinas del entorno.

Es el costo a pagar por utilizar este sistema de difusión. Una vez que se obtiene una dirección física de una máquina específica, la relación entre dirección física y dirección de Internet, debe ser guardada en alguna memoria intermedia, por si es necesario volver a mandar información al sistema remoto. En ocasiones los mensajes de difusión ARP pueden, por diversas causas, no ser recibidos por la máquina que posee la dirección IP y que debería contestar este mensaje de difusión. Esto provoca que se generen nuevos mensajes de difusión, en espera de la contestación del sistema buscado.

Dentro del protocolo ARP, cuando un sistema requiere la dirección física de un sistema remoto, genera un paquete de información donde se encuentra incluida la dirección IP que se desea alcanzar, su propia dirección IP y física, por si el sistema remoto necesita regresar información al sistema que inició la comunicación. De esta manera se evita que el sistema remoto emita un nuevo mensaje de difusión para localizar al sistema que inició la transmisión de información. Note que la máquina que solicita una dirección física, envía su solicitud mediante difusión, todas las máquinas que están en la red reciben el mensaje que solo una de ellas responde. La dirección física de la máquina que envió el mensaje de difusión junto con su dirección IP, es ahora conocida por todas las máquinas de la red. Esta combinación de direcciones es almacenada en la memoria intermedia de cada una de las máquinas del entorno de red.

TIPO DE <i>HARDWARE</i>		TIPO DE PROTOCOLO
HLEN	PLEN	OPERACIÓN
EMISOR <i>HA</i> (octetos 0-3)		
EMISOR <i>HA</i> (octetos 4-5)		IP EMISOR (octetos 0-1)
IP EMISOR(octetos 2-3)		DESTINO <i>HA</i> (octetos 0-1)
DESTINO <i>HA</i> (octetos 2-5)		
IP DESTINO (octetos 0-3)		

Tabla 13. Encabezado ARP.

TIPO DE HARDWARE.- Especifica un tipo de interfaz de hardware para el que el transmisor busca respuesta; si el valor es uno se define una red Ethernet.

TIPO DE PROTOCOLO.- Especifica el tipo de dirección de protocolo de alto nivel que proporcionó el transmisor, para IP (0x0800).

HLEN y *PLEN*.- Especifican la longitud de la dirección de hardware y de protocolo de alto nivel.

OPERACIÓN.- especifica el tipo de petición que se realiza en el entorno de red ARP (1) para una solicitud de dirección física, ARP (2), para respuesta que no es transmitida por difusión y RARP (3) para intercambio de mensajes ARP.

EMISOR HA.- Seis octetos para especificar la dirección física de que envía el mensaje de difusión.

IP EMISOR.- Cuatro octetos donde debe estar contenida la dirección IP del que envía el mensaje.

DESTINO HA.- Seis octetos para la dirección de hardware o física destino.

IP DESTINO .- Cuatro octetos para la dirección IP de destino, que puede ser o no conocida.

Desarrollo

Descripción del problema

Hoy en día existen cientos de sistemas basados en computadoras personales, que tienen la finalidad de presentar o “publicar” información en Internet, estos pueden ir desde programas de software que tienen la función de generar un enlace constante con Internet para la publicación de información: (Microsoft Personal Web Server, Linux Apache, IIS WinNT/2000, etc.).

Los programas mencionados anteriormente, pueden publicar información, pero raramente podrían controlar o monitorear algún dispositivo conectado a la computadora personal, es decir, que difícilmente podrían mostrar la temperatura ambiente de alguna habitación cercana al lugar donde se encuentra la computadora o el mismo lugar donde se encuentra. Apagar o prender un foco o un sistema de riego, es una tarea titánica, cuando se utilizan sistemas que están dedicados a la publicación de información y no a el monitoreo o control de dispositivos.

Existen sistemas externos que pueden ser conectados a una computadora personal vía algún puerto de comunicaciones (RS232, Puerto Paralelo, USB), que tienen la tarea principal de monitorear y controlar dispositivos. Por mencionar algunos disponibles, COP8.COM de National Instruments, Rabbit Semiconductor SDK, que son sistemas de fácil implementación que permiten publicar información en Internet de una manera sencilla e intuitiva. Prácticamente cualquier persona con conocimientos básicos de programación, Internet y HTML, puede utilizarlos.

Estos sistemas tienen un gran inconveniente, aun siguen utilizando una computadora que proporciona el enlace a Internet y desarrolla la mayor parte del procesamiento de la información de los sensores y actuadores.

Se debe tener en cuenta el costo de un programa capaz de intercambiar información entre la computadora y el dispositivo que colecta las variables de entorno; el programa debe controlar el acceso a los periféricos y componer la pagina WEB para su publicación.

Sumando todos los costos anteriores, es claro que tener un sistema que se encargue de monitorear y controlar algunas variables de entorno en un lugar definido, es demasiado elevado. Más aún si se considera tener varios sistema dedicados.

Objetivos

GENERALES

Diseñar un sistema mínimo basado en un microcontrolador, que proporcione servicios TCP/IP, que permita controlar y monitorear el estado de diversos sensores y/o actuadores desde un sistema remoto.

ESPECÍFICOS

- Proporcionar conexión a Internet mediante una red Ethernet RJ45.
- Diseño del núcleo del sistema operativo robusto, sistemas de comunicación e interconexión con Internet para el control del dispositivo.
- El núcleo del sistema permitirá el cambio de la página web y de los elementos incrustados en ella (imágenes, código JAVA)
- Enlace a Ethernet y control de puertos, independiente de una computadora personal.
- Implementar puertos de comunicación I²C, RS-232 y puertos digitales en el servidor.
- Desarrollar página HTML y código JAVA para el control y monitoreo de puertos de comunicación.

Justificación

Tener un sistema de monitoreo y control con una computadora personal dedicada y un enlace a Internet exclusivo para el control y monitoreo resultaría absurdo. El presente desarrollo, debe resolver problemas que están vinculados con el uso de una computadora personal dedicada, la no dependencia del sistema operativo y la de un programa específico para publicar información.

El desarrollo debe ser una solución de bajo costo para el control y monitoreo de dispositivos remotos conectados a una red tipo Ethernet. Que permita tener uno o varios sistemas en una sección dentro de un instituto o inclusive siendo un poco ambiciosos, varios sistemas en una casa habitación.

Siendo ésta la base para futuros desarrollos, debe ser flexible y genérico para permitir el control y manejo de sistemas masivos de almacenamiento de información: discos duros, unidades de CDROM, cintas magnéticas, DVDROM.

Dispositivos existentes

Actualmente han surgido un sin número de dispositivos que proporcionan servicios de Internet, todos ellos, utilizando la menor cantidad de componentes electrónicos, que obviamente tienen ciertas limitaciones, comparados con el poder de una computadora.

La ventaja de utilizar estos mini sistemas, radica en el costo del dispositivo y la velocidad en el manejo de la información que se transmite por medio de la red de comunicaciones. Todo ello gracias a ser sistemas dedicados para transmisión de información en Internet.

Los dispositivos que actualmente se encuentran en el mercado difieren en su tamaño, y las ventajas para generar aplicaciones de Internet. Algunos de ellos proporcionan ambientes totalmente gráficos orientados a objetos, donde el conocimiento que requiere el usuario, es casi nulo. Pero aunado a este mínimo esfuerzo, se tiene muy poca versatilidad para el control y desarrollo de aplicaciones personalizadas.

Algunos dispositivos permiten generar las interfaces gráficas en ambientes gráficos para compiladores Java y Microsoft Visual Java. Otros requieren de grandes modificaciones en el núcleo de los programas y re-configuración del sistema, todo basado en el costo del dispositivo. Algunos de ellos tienen una gran cantidad de memoria y procesadores de alto desempeño para el manejo de los periféricos.

Los dispositivos que ofrecen algunas características similares al trabajo presentado son:

- Tini (IButton).- Procesador DS80C390, con aplicaciones Java , Dos puertos seriales, dos puertos 1-Wire, dos puertos CAN, bus serial 2-wire, reloj de tiempo real entradas salidas digitales, 512Kb/1Mb memoria SRAM no volátil . Desarrollo de aplicaciones en Java.
- PicDem Microchip. Basado en un procesador Microchip PIC16F877, soporta interfases Ethernet y RS-232 para comunicaciones, 6 pines para conexión de dispositivos externos, posibilidad de adicionar un MODEM y una pantalla líquida de 16 X 2 . Librerías de programación, basadas en ensamblador nativo de Microchip.
- PicoWeb. Basado en Procesador Atmel AT90S8515 y una interface Ethernet de Realtek RTL8019AS, Conector DB25 para configuración, almacenamiento de datos y conexión de dispositivos externos digitales. Memoria serial de 32Kb para el almacenamiento de páginas Web e imágenes. Software para programación, librerías para programación del sistema en ensamblador de Atmel.
- Rabbit semiconductors Processor. Grupo de instrucciones familiares con el Z80/180, con un grupo de librerías para manejar TCP/IP
- Existen también una cantidad considerable de compañías que proporcionan procesadores dedicados para el manejo de protocolo TCP/IP, como son Interniche, Segger Microcontroler RTIP, Seiko TCPIP Stack.

Microcontrolador		Memoria para aplicaciones	Puertos Digitales	Protocolos soportados	Puertos de comunicación	Costo (pesos)
Micro WEB server	ATMEL AT90S8515	256Kb	16	HTTP TCP/IP	(1) RJ45 (1) RS232 (1) I2C (1) ISP	\$1,200
Tini	DS80C390	512 Kb	64	HTTP TCP/IP	(1) RJ45 (2) RS232 (1) 2-Wire (2) CAN (2) 1-Wire	\$4,000
PicDem	Microchip PIC16F877	8 Kb	6	HTTP TCP/IP	(1) RJ45 (1) RS232	\$3,900
PicoWeb	Atmel AT90S8515	32 Kb	8	HTTP TCP/IP	(1) Conector DB25 para Programación	\$2,285
Rabbit semiconductors Processor	Rabbit Semiconductors 2000	256 Kb	56	HTTP TCP/IP	(1) RJ45 (6) Puertos Seriales	\$5,600

Tabla 14. Comparción entre dispositivos existentes.

Sistema Propuesto

El sistema propuesto cuenta con dos puertos Digitales de 8 bits, un puerto I²C para conectar sensores o sistemas de comunicación adicionales, pantalla LCD, teclado, o bien para introducir nuevas variables al sistema (sensores de temperatura, convertidores de voltaje a señales digitales, sensores de presión, etc.). Puerto de comunicaciones RS232 estándar, para introducir datos desde dispositivos que cuenten con este puerto estándar (osciloscopios, analizadores de señales lógicas, etc).

Se plantea un sistema controlado por un microcontrolador (en este caso un AVR AT90S8015 de Atmel), que se encarga de controlar el flujo de información, de la red hacia los dispositivos de control y de los dispositivos hacia la red. Establece también los protocolos de comunicación necesarios.

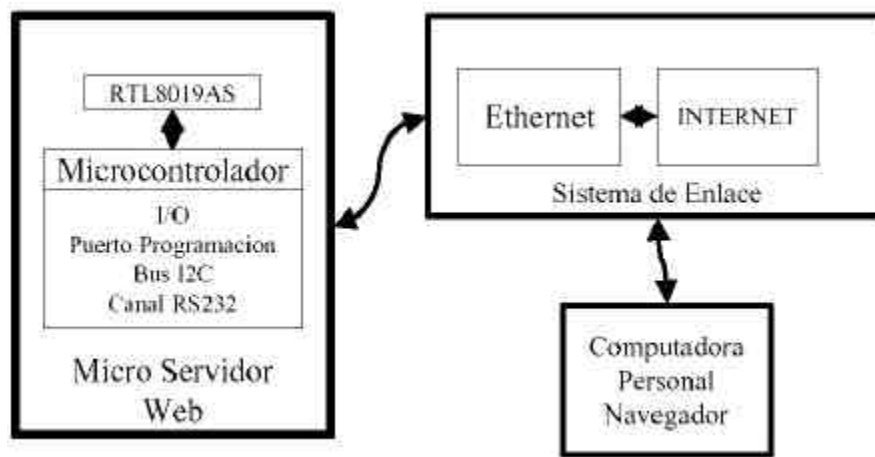


Figura 20. Diagrama a bloques del sistema.

RTL8019AS

El controlador Ethernet tiene la tarea de enviar y extraer datos de la red. Este controlador esta diseñado para controlar el flujo de información de la red al dispositivo o viceversa.

Cuenta con un canal de datos de 16 bits, un canal de direcciones de 20 bits, siete interrupciones por hardware, tres modos de conexión, Ethernet, AUI y BNC.

El controlador esta diseñado para trabajar bajo un entorno PNP (Plug and Play), para configuración de direcciones del dispositivo y transferencias DMA de datos configuradas por Software.

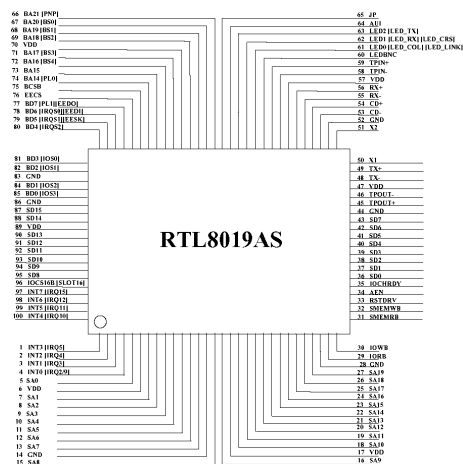


Figura 21. Diagrama del controlador RTL8019AS.

Internamente consta de un grupo de 64 registros de 8 bits, agrupados en 4 paginas. Estas páginas son cambiadas, escribiendo en la dirección 00H (CR, Control Register) del controlador, donde con los bits PS0 y PS1, especifican el número de página a utilizar y por lo tanto, los registros que serán configurados.

De manera general, se presentan los registros internos del controlador RTL.

No (Hex)	Page0		Page1	Page2	Page3	
	[R]	[W]	[R/W]	[R]	[R]	[W]
00	CR	CR	CR	CR	CR	CR
01	CLDA0	PSTART	PAR0	PSTART	9346CR	9346CR
02	CLDA1	PSTOP	PAR1	PSTOP	BPAGE	BPAGE
03	BNRY	BNRY	PAR2	-	CONFIG0	-
04	TSR	TPSR	PAR3	TPSR	CONFIG1	CONFIG1
05	NCR	TBCR0	PAR4	-	CONFIG2	CONFIG2
06	FIFO	TBCR1	PAR5	-	CONFIG3	CONFIG3
07	ISR	ISR	CURR	-	-	TEST
08	CRDA0	RSAR0	MAR0	-	CSNSAV	-
09	CRDA1	RSAR1	MAR1	-	-	HLTCLK
0A	8019ID0	RBCR0	MAR2	-	-	-
0B	8019ID1	RBCR1	MAR3	-	INTR	-
0C	RSR	RCR	MAR4	RCR	-	FMWP
0D	CNTR0	TCR	MAR5	TCR	CONFIG4	-
0E	CNTR1	DCR	MAR6	DCR	-	-
0F	CNTR2	IMR	MAR7	IMR	-	-
10-17	Remote DMA Port					
18-1F	Reset Port					

Tabla 15. Registros generales RTL8019AS.

Cada uno de estos registros de 8 bits, se puede, configurar bit a bit, para la transferencia y control de datos. Estos registros son modificados y configurados, accediendo a la dirección de memoria establecida por la columna “No. (Hex)”, a la que aún es necesario sumar la dirección base en la que se encuentre el controlador Ethernet.

El puerto DMA se puede acceder desde la dirección BASE+10h hasta BASE+17H, dependiendo de la configuración de paginas DMA del controlador. Los registros de la pagina 0, 1 y 2 son los estándar para cualquier tipo de controlador NE2000, y es en la página 3 donde residen los registros de propósito general, específicos del controlador RTL.

Uno de los registros principales del controlador RTL, es el registro CR (Command Register), donde se puede configurar la página de registros a acceder, el modo de operación de la DMA, el inicio de la transmisión de un paquete de información o detener el acceso a la DMA para los datos de entrada. Este registro se configura de la siguiente manera.

PS1		PS0	RD2	RD1	RD0	TPX	STA	STP
PS1	PS0	Descripción						
0	0	Selecciona Página de registros 0						
0	1	Selecciona Página de registros 1						
1	0	Selecciona Página de registros 2						
1	1	Selecciona Página de registros 3						
RD2	RD1	RD0	Descripción					
0	0	0	No permitido					
0	0	1	Lectura Remota					
0	1	0	Escritura Remota					
0	1	1	Enviar paquete					
1	X	X	Abortar/Completar DMA					
STA		STP	Descripción					
1		0	Comando de inicio, se habilita la DMA para la recepción de paquetes de información.					
0		1	Se detiene la recepción de paquetes					

Tabla 16. Modo de operación, Command Register (CR) RTL8019AS.

TPX.- Se define para transmitir un paquete de información, cuando ésta ya se encuentra dentro de la DMA. Cuando la transmisión es completada, este bit es definido nuevamente a cero.

A nivel de Hardware, las operaciones de lectura/escritura de información del controlador RTL al procesador, se realizan mediante un protocolo de comunicaciones, especificado a continuación.

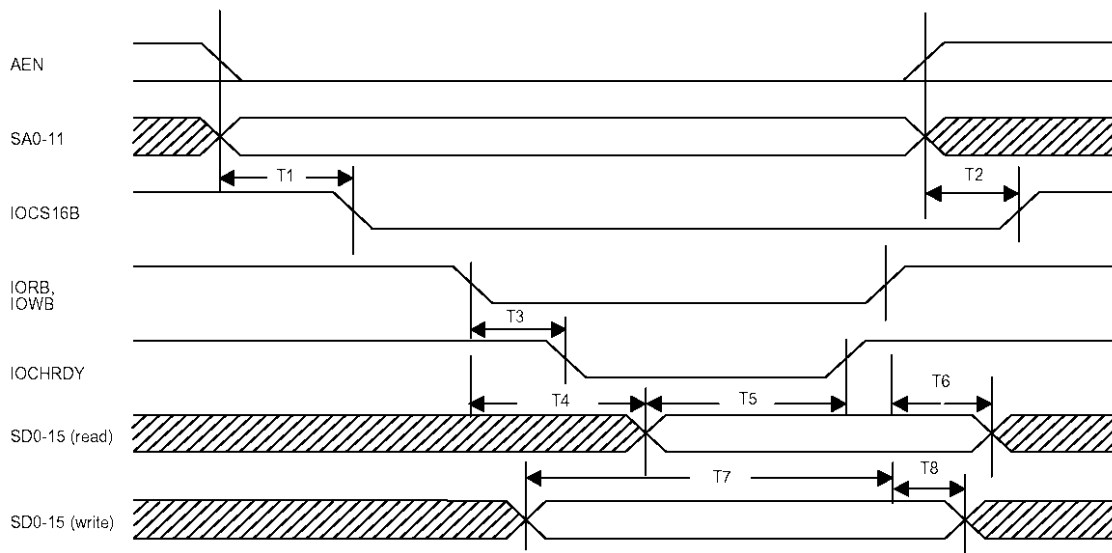


Figura 22. Diagrama de tiempos de lectura escritura RTL.

No todas las líneas deben ser cambiadas para escribir/leer, algunas de ellas puede estar definidas por hardware, para el presente desarrollo, las líneas de control AEN, IOCS16B, IOCHRDY, se definieron vía hardware.

El controlador RTL contiene una memoria interna DMA de 16K para el almacenamiento de paquetes de información que serán enviadas a la red o que serán leídos desde ella. Esta memoria, esta dividida en páginas de 256 bytes. La longitud total de la memoria DMA, esta definida por los registros de control Page Start Address y Page Stop Address.

Como se puede observar en la figura 23, la memoria tiene un comportamiento FIFO (First In First Out) de anillo, esto quiere decir que el primer paquete que llego de la red, será el primer paquete que saldrá de la memoria. El arreglo de anillo especifica que los paquetes son almacenados en posiciones consecutivas de la memoria, pero cuando se alcanza el límite máximo de la memoria, esta regresa al inicio del bloque de memoria. Es por ello, que los datos que permanecen mucho tiempo dentro del sistema de memoria pueden ser borrados por un paquete entrante. El controlador RTL, no envía ninguna información, cuando se alcanza el máximo de la memoria, ni tampoco informa cuando un paquete que no ha sido leído es borrado por un nuevo paquete de entrada. Por lo tanto para evitar la pérdida de paquetes de información, los paquetes deben ser extraídos a una velocidad mucho mayor que la velocidad con la que llegan del sistema de red.

En la figura 23, se muestra el funcionamiento de la memoria FIFO.

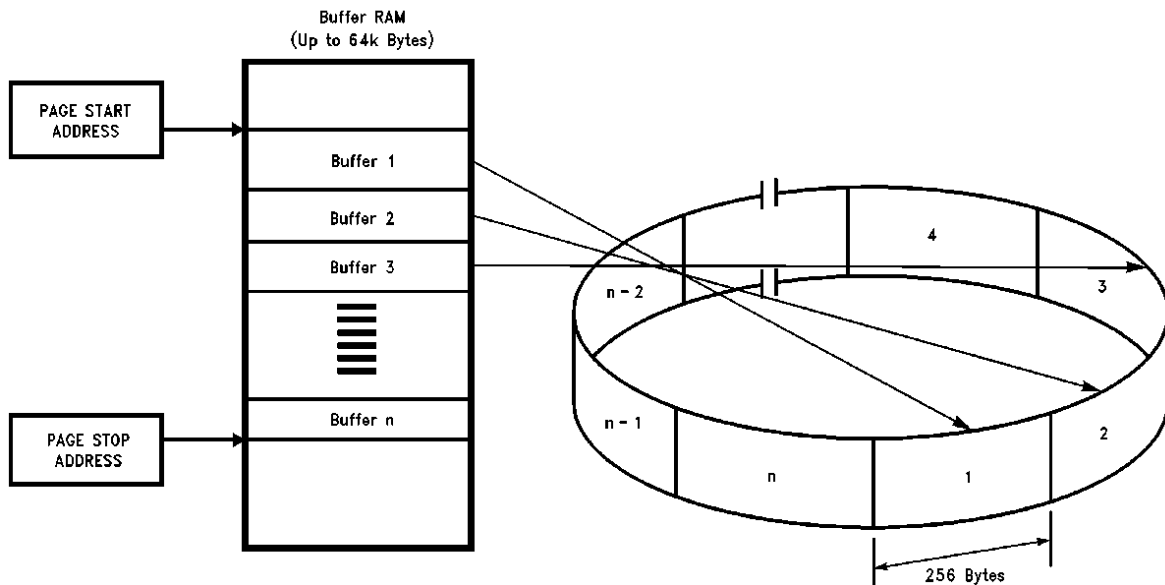


Figura 23. Diagrama de memoria DMA RTL.

Una región de memoria es reservada para almacenar temporalmente los paquetes que serán enviados a la red, el valor de memoria esta definido por la constante XMT_BUF_START=\$5400, valor que debe ser almacenado en el registro de control TPSR.

La DMA del RTL8019AS, funciona en la localidad 10H del controlador, y funciona como se ha mencionado, de manera FIFO, el primer bloque de datos recibido será el primero en ser enviado a la DMA serial del controlador, cuando este dato es leído, el “anillo” coloca el segundo dato en la posición de la DMA, por lo que será el siguiente dato a leer.

El controlador Ethernet, interactúa con el microcontrolador AVR, mediante un bus de 8 bits para lectura y escritura de datos (SD0-SD7). Un canal de 5 bits de direcciones, dos bits para el proceso de lectura escritura y un bit para especificar en tiempo de ejecución, un regreso al estado inicial (RESET). Aunque el controlador permite un canal de datos de 16 bits, para el presente trabajo y por la configuración AVR-RTL, se considera adecuado, utilizar un canal de datos de 8 bits.

El direccionamiento del RTL, se realiza mediante un canal de 20 direcciones, de las cuales, SA19 al SA5, son alambradas por “hardware”, para especificar la dirección base del controlador (300H). Del SA4 al SA0 son configuradas por el controlador para acceso a registros y a la memoria DMA (Rango de memoria 300H–31FH). La configuración anterior, le permite al microcontrolador AVR, manejar completamente al RTL, con solamente 5 líneas de dirección (SA4-SA0).

Los canales de I²C, RS232 y los 8 bits digitales son manejados por el AVR e introducidos al sistema de red por el RTL.

La conexión del sistema con la red Ethernet, se realiza físicamente por los canales TPOUT+, TPOUT-, TPIN+ y TPIN-, que son conectados con un filtro transmisor/receptor 20F0001N, que es el encargado de acoplar el sistema digital del MicroWeb Server, con los canales de red, al mismo tiempo que limita la cantidad de ruido de la red y evita lleguen descargas de alto voltaje al diseño.

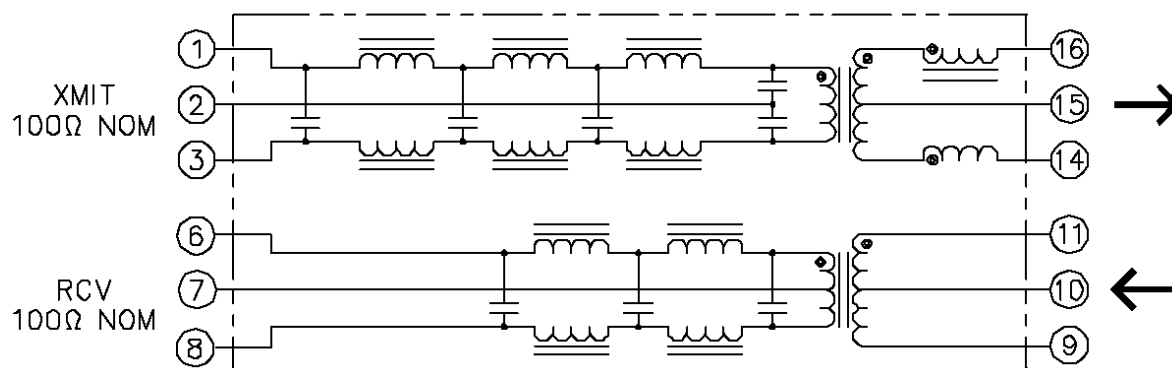


Figura 24. Diagrama Interno FL1012.

Los canales de salida (16 al 9) del FL1012, son conectados directamente a un conector RJ45 y de ahí al sistema de red.

Configuración y arranque.

El control del sistema se realiza, con un procesador ATMEL AVR AT90S8015, éste dispositivo es el encargado de controlar el flujo de información entre dispositivos de adquisición de datos y enlace a la red Ethernet.

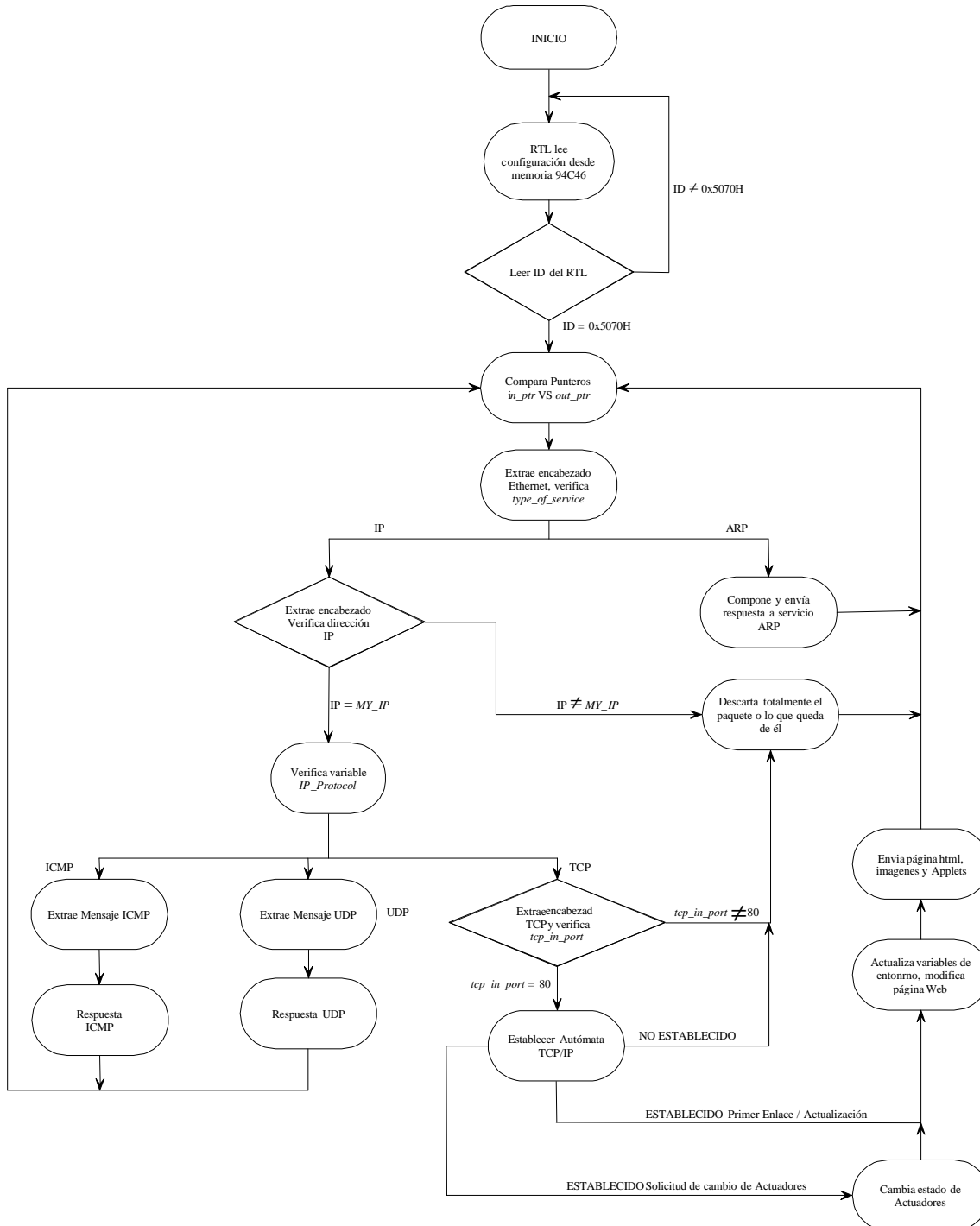


Figura 25. Diagrama de flujo del proyecto

El primer paso para el arranque del sistema, es configurar el controlador RTL para la adquisición de datos provenientes del entorno de red. Esta configuración, es leída de la memoria 24C256, donde se obtienen los parámetros de configuración de arranque del RTL. Esta memoria se encuentra conectada al microcontrolador por medio de un canal I2C.

Durante esta operación, se definen parámetros como son el tamaño de las páginas para el almacenamiento de los paquetes Ethernet provenientes de la red, así como las direcciones de inicio y fin de dichas páginas. Se establece el modo de Multicast y broadcast como activos. Se establece la dirección física del dispositivo dentro de los registros del RTL.

Una vez configurado el RTL, se define el controlador Ethernet en modo de escucha, para recibir información. Existen tres punteros a direcciones de memoria de la DMA (in_ptr, out_ptr y next_ptr). El apuntador in_ptr, define la dirección de la página de inicio del primer bloque de datos recibido. El apuntador out_ptr, apunta a la dirección de la página del último bloque de datos recibido. El apuntador next_ptr, se utiliza para almacenar el siguiente paquete de información a procesar

El uso de los apuntadores in_ptr y out_ptr, permite establecer si existen datos nuevos en la memoria del RTL. Si la dirección de memoria contenida en estos apuntadores es la misma, no existen paquetes Ethernet.

Cuando existe una diferencia entre estos valores, los datos contenidos en la dirección de memoria almacenada en el puntero out_ptr, deben ser leídos, procesados o descartados, en el caso de no contener información necesaria para el sistema. El primer dato que es extraído de la DMA del sistema, corresponde a la dirección de memoria del siguiente grupo de datos a los que se tendrá que hacer referencia, una vez que el paquete en out_ptr sea procesado. El puntero next_ptr, se utiliza para almacenar la dirección de memoria del siguiente paquete a procesar, que no necesariamente es el ultimo paquete que arribó al sistema (in_ptr).

El primer bloque de datos procesado, corresponde con el encabezado Ethernet de 14 octetos de longitud. Los primeros 6 que corresponden a la dirección de destino, son por el momento descartados. Los siguientes 6 octetos corresponden a la dirección física (MAC) del sistema que originó la petición, es almacenada temporalmente en la variable de entorno *source_address*, el tipo de servicio requerido se almacena en la variable *type_of_service*.

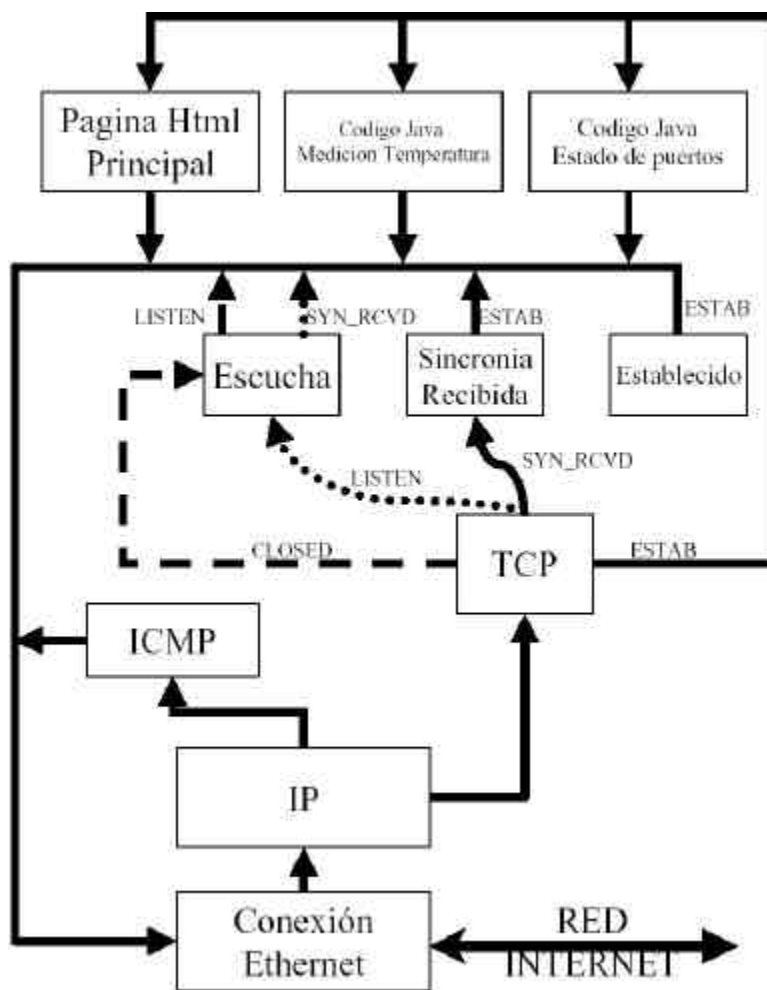
El tipo de servicio requerido por el emisor del paquete, debe ser 0x0800 (IP) o 0x0806 (ARP) para que el MicroWeb Server pueda atenderlo, en caso contrario, todo el paquete es descartado y se continua con el siguiente paquete de datos.

Si la petición corresponde con un servicio ARP, se extrae del paquete la dirección física, IP del sistema origen e IP destino. Lo anterior es necesario, por que aunque sabemos que el paquete corresponde a un servicio ARP, no tenemos la seguridad de que el MicroWeb deba responder. Solamente si la dirección IP destino concuerda con la dirección del servidor Web, el servicio ARP será procesado.

El paquete ARP debe ser generado de acuerdo con el esquema ARP de respuesta anteriormente descrito, para ser enviado a través de la red al sistema que originó la petición. Recuerde que las peticiones ARP se transmiten vía broadcast (a todos los sistemas de la red).

Los servicios IP (0x0800), que pueden ser manejados por el sistema (MicroWeb Server) son: TCP (0x06) e ICMP (0x01). El controlador obtiene de la DMA el encabezado IP sin las opciones, antes de decidir que tipo de servicio IP se atenderá. Si el paquete de datos contiene una dirección IP diferente del sistema, es obvio que no esta dirigido al éste, por lo que el paquete es descartado y se debe procesar el siguiente paquete.

Si se ha comprobado que el paquete corresponde con la dirección IP del sistema y que se requiere un servicio ICMP, aun es necesario comprobar que el paquete contenga una petición de eco (0x08), que por el



momento es el único servicio ICMP que será atendido. Si el paquete cumple con las especificaciones, se obtiene la información ICMP del paquete (icmp_code, icmp_chksum, icmp_identifier, icmp_séquence, e icmp_data) para poder responder adecuadamente.

La respuesta para un paquete TCP es un poco mas compleja que un servicio ICMP.

Como se ha mencionado anteriormente, TCP debe establecer una conexión (*socket*) antes de poder transmitir información. Para establecer la respuesta a un paquete TCP, es importante considerar las banderas TCP (flags) y el estado en el que se encuentra la conexión (*html_socket*).

Figura 26. Conexión TCP/IP.

Implementación

En base a las anteriores aseveraciones, se plantea el siguiente desarrollo.

Un sistema mínimo basado en un microcontrolador que proporcione los servicios de enlace a una red Ethernet, que sea capaz de tomar datos de sensores acoplados al sistema, y que pueda modificar el estado de algunas variables definidas. Este estado, deberá ser presentado en una interfase de computadora conectada a Internet, vía un navegador (Internet Explore, Netscape, Mozilla, etc.) en cualquier parte del mundo. Esta interfase debe contener los elementos necesarios, para modificar el estado de alguna de estas variables de entorno.

El sistema contará con puertos de comunicación para programación y configuración. Debe contar con al menos un canal de datos estándar para adicionar nuevos sensores o actuadores, para futuras expansiones. Dispositivos de almacenamiento permanente para los archivos, imágenes y programas.

El diseño del dispositivo se basa en 4 grupos principales:

- Sistema de control.
- Conexión Ethernet
- Canales de comunicación (I²C, RS232)
- Sistema de memoria
- Sensores externos

La parte de programación estará dividida en tres secciones

- Programación del controlador Ethernet
- Programación de la pila TCP/IP.
- Programación del sistema de intercambio de archivos.
- Programación de Applets de Java y código Html.
- Programación de memoria 24LC256

A continuación se desarrollarán a fondo cada una de las secciones del diseño del dispositivo y la programación del sistema, dentro del anexo A se encuentra el código en ensamblador.

Diseño del dispositivo

Sistema de control

El núcleo del sistema reside en un microcontrolador ATMEL AVR AT90S8515, que se encarga de controlar el flujo de información de la red hacia el sistema y del sistema a la red. Lee y cambia las variables de estado de diferentes dispositivos (canal de temperatura y canales digitales) a petición del usuario que se conecta al sistema.

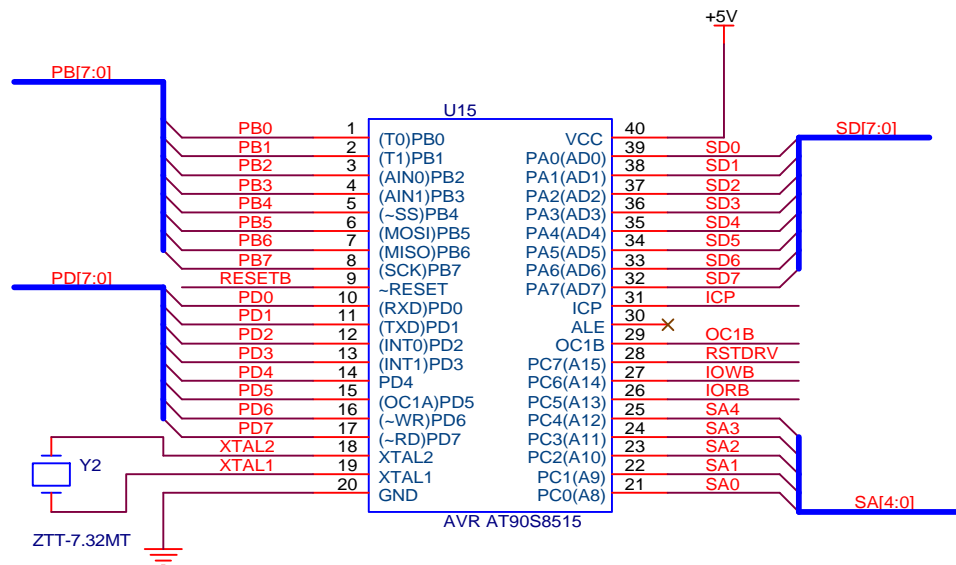


Figura 27. Diagrama de conexiones AVR.

El canal SD[7:0] es definido como el canal de datos que serán intercambiados entre los controladores RTL y AVR. El control del RTL es realizado por las líneas de control RSTDRV, IORB, IOWB y SA[4:0].

El canal PB[7:0], se utiliza para enviar o recibir datos digitales, puede ser considerado como un canal de 8 bits de propósito común. Este canal es utilizado durante el proceso de programación del microcontrolador AVR (específicamente los canales PB5, PB6, PB7 y RESETB), mediante el canal SPI (Serial Programming Interface). Una vez realizada la programación, las líneas quedan libres y pueden ser utilizadas libremente.

El canal PD[7:0], tiene el mismo propósito que el canal PB. Para el presente desarrollo se planteó utilizarlo para introducir las variables del sistema y comunicación. Específicamente, los canales PD0 y PD1 se utilizan para intercambio de información RS232. PD6 y PD7 controlan el flujo de información de dispositivos con protocolo de comunicaciones I²C (sensor de temperatura).

Conexión Ethernet

El diseño de conexión a una red tipo Ethernet se realiza con el controlador RTL8919AS, como se ha mencionado. Este dispositivo cuenta con un canal de direcciones de 20 bits (SA[19:0]). Se establece la dirección de base del controlador en 0x0300h, mediante la conexión física de las terminales SA[19:5], las líneas restantes necesarias son SA[4:0] que son manejadas por el controlador AVR.

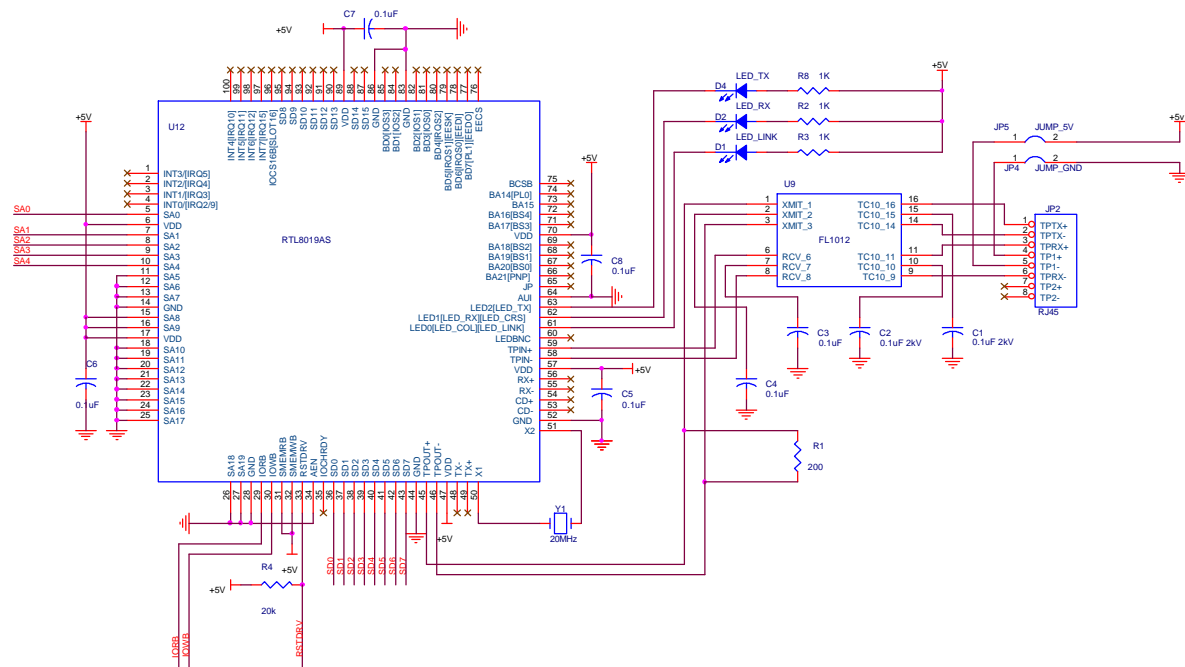


Figura 28. Diagrama de conexiones RTL8019AS.

El control del puerto de lectura/escritura del controlador es establecido por el manejo de las líneas IORB e IOWB en conjunto con los canales SA[4:0], y de acuerdo al protocolo de E/S definido por el controlador RTL.

La información que sale del controlador a la red o viceversa, es acoplada con filtro FL1012 desde las terminales TPIN-/TPIN+ y TPOUT-/TPOUT+ y después al conector RJ45 al medio físico de la red Ethernet.

Los leds se programan para indicar estados del controlador, transmisión, recepción o enlace. Estos leds tienen un comportamiento que se define cuando se inicializa el controlador RTL.

Canales de comunicación

Canal RS232

El canal RS232 es utilizado para enviar archivos hacia la memoria I²C del sistema, mediante un programa almacenado en el microcontrolador. Este canal puede utilizarse también para otras aplicaciones.

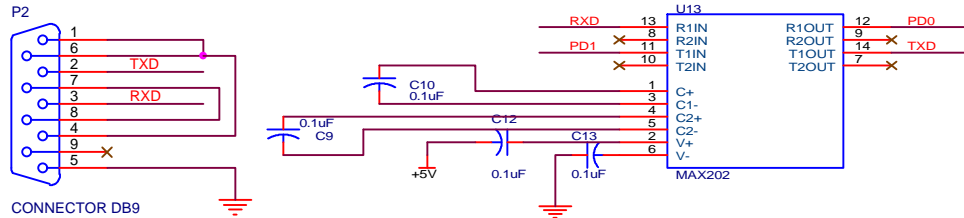


Figura 29. Canal RS232

Una vez programada la memoria, el canal RS232 queda libre en “tiempo de ejecución” y es posible realizar transferencia de datos mediante este puerto. Es importante señalar que la pagina Web no tiene programada la visualización de la información obtenida por este puerto, pero puede ser programada de acuerdo a aplicaciones específicas.

Canal I²C

Se seleccionó un canal de datos I²C, debido a que únicamente necesita de dos líneas de comunicación y puede manejar hasta 112 direcciones bajo el esquema de direcciones de 7 bits y 1024 en direccionamiento de 10 bits. La cantidad de dispositivos que trabajan con el esquema de este bus, se ha incrementado de manera exponencial, hoy en día podemos encontrar sensores, memorias, microcontroladores, pantallas de cristal líquido, procesadores de video, convertidores analógico/digital o digital/analógico, etc. En comparación con un bus CAN, el costo de implementación es mucho menor, así como el costo de los dispositivos. El bus CAN puede operar en entornos con altos niveles de ruido electromagnético, pero esto es una característica específica que no es estrictamente necesaria para el el MicroWeb Server.

La implementación del canal I²C es relativamente sencilla, el control del bus reside en el microcontrolador AVR, mediante dos líneas que definen al canal: SCL y SDA. La secuencia en el cambio de los niveles de voltaje en estas líneas define el dispositivo y las funciones a utilizar.

El nivel de voltaje en los canales SDA y SCL, es definido mediante el sistema de colector abierto del microcontrolador AVR. Cuando el puerto se encuentra definido como 0, indica que el puerto se encuentra conectado directamente a tierra y un valor de uno significa conexión con 5 volts. Lo anterior explica el uso de las resistencias de descarga para cada uno de los dispositivos I²C, que en realidad trabajan como un sistema de pull up.

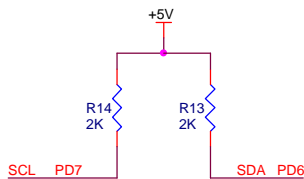


Figura 30. Canal I²C implementado.

Sistema de Memoria

El sistema de almacenamiento del sistema reside en una memoria I²C ATMEL 26C256, que puede almacenar 32,768 bytes de información (32K X 8 = 256K). Aquí se almacena los archivos necesarios para establecer la página WEB, es decir archivos html, applets de Java e imágenes.

La transferencia de información esta controlada por las líneas SCL/PD7 y SDA/PD6. El puente JUMPER_3 sirve para proteger a la memoria de escrituras por errores en la transmisión del protocolo I²C. Dirección del dispositivo 0xA0h y 0xA1h.

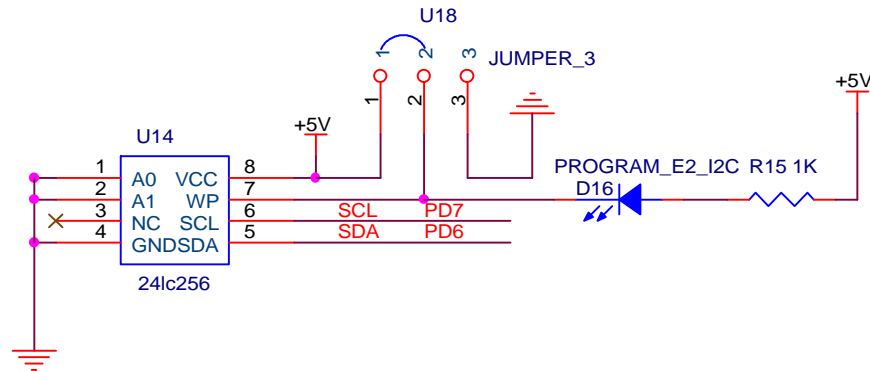


Figura 31. Diagrama del sistema de memoria I²C.

Sensor de temperatura.

El sensor de temperatura seleccionado es el LM75 ya que cumple con el protocolo de comunicaciones I²C y obtiene lecturas en grados centígrados. La dirección base de este dispositivo es 0x90 y 0x91. Los rangos de temperatura van de 125°C a -55 grados teniendo algunas secciones de no linealidad.

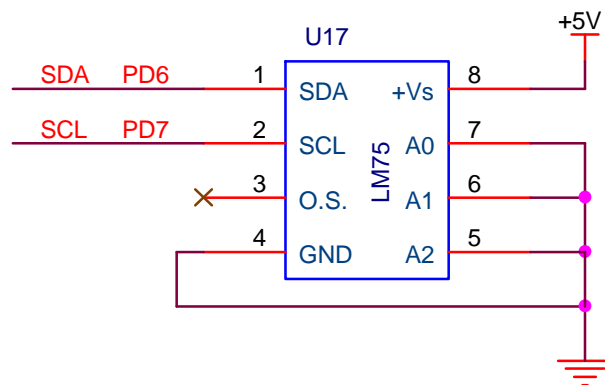


Figura 32. Sensor de temperatura LM75

Empleo del sistema y restricciones


El sistema es conectado a una red tipo Ethernet, mediante un cable RJ45, la dirección IP del sistema esta definida por una variable de entorno denominada *MY_IP*, la cual es grabada dentro de la memoria E2PROM del controlador, es decir, reside en el código ensamblador. Los archivos de imágenes, código html, y applets JAVA, son almacenados en la memoria 24LC256, mediante el uso de dos programas, *eepromi2c.asm* y *pro_memi2c.c*.

Los archivos generados en programas de alto nivel, como son applet de Java, imágenes GIF y código html, son enviados utilizando el puerto de comunicación RS232. Primeramente se seleccionan los archivos que serán enviados, hasta un máximo de 10.

El programa no envía únicamente los archivos, los procesa determina su longitud y calcula un CRC que es necesario dentro del proceso de transmisión. El programa *eepromi2c.asm*, adquiere la información del puerto de comunicaciones RS-232, la traduce dentro del microcontrolador AVR al protocolo I²C y la envía a la memoria i2c.

Este programa supone que el primer dato que entra por el puerto de comunicaciones, estará alojado en la dirección 0x000 de la memoria, ya que el programa desarrollado en Lenguaje C, se encarga de organizar los archivos de manera secuencial. Es importante recordar que el conector denominado *PROGRAM_E2_I2C*, debe definirse para tal propósito.

Los archivos almacenados en la memoria tienen el siguiente formato, longitud del archivo 2 bytes, checksum 2 (bytes), y bytes de información del archivo. Para ejemplificar el proceso anterior observemos el

formato de la imagen *long_F5_device_on.gif*  que define el estado de apagado de un actuador del sistema.

```
long_F5_device_off:
0x00,0xC3
checksum_F5_device_off:
0xB1,0xB6
F5_device_off:
0x47,0x49,0x46,0x38,0x39,0x61,0x1E,0x00,0x21,0x00,0xB3,0x00,0x00,0x00,0x00,0x00
0x9C,0x9C,0xCE,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x21,0xF9,0x04
0x01,0x00,0x00,0x02,0x00,0x2C,0x00,0x00,0x00,0x00,0x1E,0x00,0x21,0x00,0x00,0x04
0x5A,0x50,0xC8,0x49,0xAB,0xBD,0x38,0xEB,0xCD,0xBB,0xFF,0x60,0x88,0x01,0x64,0x59
0x8A,0x92,0xA9,0x92,0xE2,0xEA,0x82,0x6E,0xEC,0xC5,0x74,0x47,0xCB,0x1B,0x7B,0xAB
0xB6,0xB0,0x9B,0xB6,0xDF,0x69,0x26,0x04,0x7C,0x8C,0xBF,0x10,0xB2,0xA6,0xBC,0xA1
0x7C,0xB8,0x27,0x14,0x28,0x4D,0xF1,0xAA,0xD3,0x61,0x75,0x85,0xB5,0x6A,0x51,0xB4
0xC0,0xD3,0x28,0x08,0x04,0xA8,0x63,0xB3,0x49,0x8C,0x55,0x03,0xCC,0x5B,0xC9,0x99
0xD5,0x2D,0xCF,0xEB,0x13,0x72,0xF7,0x8B,0xF7,0x44,0x00,0x00,0x3B,0x00
```

Los dos primeros bytes (0x00C3 = 195) definen la longitud del archivo, los siguientes dos (0xB1B6) establecen la suma de chequeo del archivo y después los bytes que definen el archivo de imagen.

Pruebas Realizadas

Una vez conectado el sistema a la red Ethernet, debe observarse que el led indicador de enlace (link), se encuentre parpadeando, como indicativo de enlace a la red. Para el sistema remoto, el primer paso para revisar que el sistema se encuentre enviando datos a la red, es enviar una petición de eco, mediante el uso del programa “ping”.

La petición de eco debe realizarse con un grupo de datos de 32 bytes con la secuencia predeterminada “abcdefghijklmnopqrstuvwabcdefghi”, esta secuencia de datos, se encuentra implícita dentro del programa ping en muchas plataformas de sistemas operativos.

Para el sistema Windows, solo es necesario ejecutar el programa desde la línea de comandos “ping -t 148.204.45.64”. El uso de la opción -t, que significa que se enviarán paquetes de datos a la dirección especificada, hasta que el usuario termine la petición (CTRL.+ C). El resultado de esta petición, es mostrado en la figura 33.

```

MS-DOS PING
Auto [v] [?] [P] [C] [V] [F] [A]
C:\WINDOWS>ping -t 148.204.45.64
Haciendo ping a 148.204.45.64 con 32 bytes de datos:

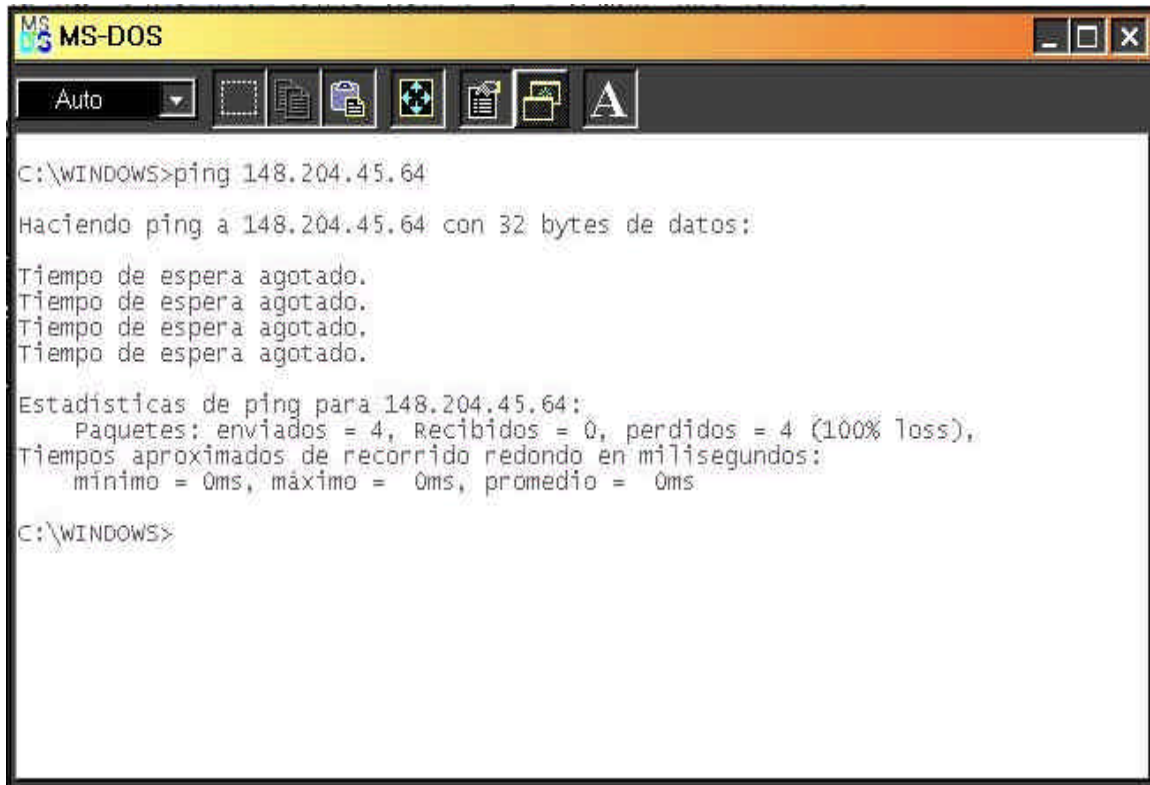
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=2ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255
Respuesta desde 148.204.45.64: bytes=32 tiempo=1ms TDV=255

```

Figura 33. Respuesta para una petición de eco (ping) de 32 bytes de datos.

Si se hubiera mandado solamente el comando “ping 148.204.45.64”, solo se enviarán 4 peticiones de eco.

Cada petición de eco tiene una respuesta como las que se muestra anteriormente o bien puede ser que el sistema no encuentre la ruta especificada, por lo que enviará un error de eco, como se muestra en la figura 34.



```

MS-DOS
Auto
C:\WINDOWS>ping 148.204.45.64

Haciendo ping a 148.204.45.64 con 32 bytes de datos:

Tiempo de espera agotado.
Tiempo de espera agotado.
Tiempo de espera agotado.
Tiempo de espera agotado.

Estadísticas de ping para 148.204.45.64:
    Paquetes: enviados = 4, Recibidos = 0, perdidos = 4 (100% loss),
    Tiempos aproximados de recorrido redondo en milisegundos:
        mínimo = 0ms, máximo = 0ms, promedio = 0ms

C:\WINDOWS>
  
```

Figura 34. Respuesta al comando “ping” cuando el sistema no se encuentra enlazado

Para cada petición de eco existe una respuesta donde es importante destacar el campo especificado por “tiempo”, que es el tiempo que le tomo a la petición de eco llegar al destino y regresar.

El tiempo de respuesta para el sistema actual dentro de una LAN Ethernte 10Mbs, es de menos de un milisegundo promedio.

Cuando se ha verificado la conexión con el sistema remoto, es posible lanzar la primera petición http, utilizando algún navegador, como Internet Explorer, Netscape Navigator, Mozilla, etc. El navegador seleccionado debe reconocer comandos Html 1.0 y Java Script 2.0 o posteriores.

La figura 35 muestra la interfaz de salida utilizando Internet Explorer 5.0.



Figura 35. Pantalla de salida del visualizador.

Debe recordarse que los datos presentados son actualizados por petición. La temperatura mostrada solo se actualiza cuando el sistema realiza una petición de “cambio de estado”/”Actualización”. Si se desea cambiar el estado de alguno de los actuadores, deberá marcarse dentro de las casillas de verificación el estado deseado y después presionar el botón de “cambia estado”.

Para probar el cambio de estado de los actuadores, se acopla un modulo de leds, en el puerto de expansión PORTB del sistema. Aunque es obvio que incrustando un sistema de potencia al modulo, pueden controlarse cualquier tipo de dispositivos.

Conclusiones

Como resultado de la presente investigación, se ha logrado el desarrollo de un sistema que proporciona servicios básicos TCP/IP que controla y monitorea variables de estado desde un sistema remoto, haciendo uso de la red de redes (INTERNET).

El núcleo central del sistema del micro servidor controla los accesos al medio físico de red, enviando y recibiendo paquetes de información. Este núcleo central es el encargado de obtener el estado de las variables de entorno y de adecuarlas dentro de una página WEB.

Los puertos de comunicación se encuentran disponibles para adicionar otros dispositivos que cumplan con los estándares de dichos puertos (I²C y RS232).

El sistema permite cambiar la página WEB, sin necesidad de modificar el núcleo central del sistema controlado. Esto es posible gracias a que el sistema funciona mediante el intercambio de archivos. Solamente es necesario cambiar el contenido de los archivos residentes en la memoria permanente del sistema, para obtener un nuevo entorno en la página WEB. Incluso pueden adicionarse nuevos archivos siempre y cuando estén contenidos en el código HTML de la página web; conservando el formato para dichos archivos (Una F de File y un número, F8, F20, F43, etc.).

La propuesta de un laboratorio a distancia, es factible con este diseño, ya que existen muchos dispositivos, como osciloscopios, analizadores de señales, cámaras de video, que con pequeñas modificaciones pueden acoplarse a este sistema, proporcionando análisis de señales desde lugares remotos. La conexión a este u otros dispositivos, permiten a instituciones educativas de bajos recursos, utilizar aparatos electrónicos que no existen físicamente en dicha institución.

Las metas propuestas, son sin duda alcanzadas, y las propuestas para nuevos diseños y mejoras futuras harán que el sistema crezca desempeño.

La versatilidad del sistema, le permitiría estar presente en una gran cantidad de aplicaciones. Centralizar o divulgar información para análisis en grupo. Y lo mas importante de todo; éste es un sistema dedicado de publicación de información o control de dispositivos. No necesita de una computadora para la transmisión de información, por lo que el costo se ve drásticamente reducido. La velocidad del sistema dedicado, supera a sistemas basados en PC de alto desempeño, ya que no necesita atender otro tipo de operaciones que no estén involucradas con la transmisión TCP/IP.

TRABAJOS FUTUROS

Uno de los principales trabajos a futuro, es el desarrollo de un compilador visual, en el que se pueda seleccionar los archivos (incluyendo la página WEB), para que sean almacenados directamente en la memoria permanente del sistema, utilizando el puerto SPI del sistema. Este compilador tendrá, además, la tarea de modificar el núcleo principal donde se encuentran las direcciones de memoria de inicio de cada uno de los archivos necesarios.

El sistema maneja una cantidad relativamente pequeña de memoria, pero haciendo pequeñas modificaciones en el núcleo central del programa es posible manejar sistemas de almacenamiento masivo, que pudieran ser desde discos duros o unidades de CD-ROM e inclusive unidades de almacenamiento DVD-ROM.

Las imágenes obtenidas desde algún equipo médico, como tomógrafos o sistemas de resonancia magnética, pueden ser enviadas a una red Ethernet o TCP/IP, para ser almacenadas para futuros análisis o enviadas a sistemas con mayor capacidad de procesamiento para análisis a detalle. Pueden ser consultadas por un grupo de especialistas que no necesitan estar en el lugar donde se toman dichas placas, ni si quiera en el mismo hospital o en el mismo país.

El monitoreo de áreas de trabajo o estudio, es un tópico, que fácilmente es alcanzado, con la adición de una cámara de video al sistema. La persona que tenga los privilegios suficientes para acceder a dichos dispositivos, podrá monitorear remotamente la actividad de dicha área o áreas inclusive. Recuerde que el sistema no maneja un flujo de video constante, puede manejar un sistema de secuencias de imágenes, por lo que una mejora consiste en implementar un servidor de flujo de video y/o audio (*streamcast transmisores de flujos de audio y/o video*).

Hay que recordar que el sistema diseñado, forma la base para que otros dispositivos puedan acoplarse utilizando los puertos de comunicación, con un mínimo de cambios en el núcleo del sistema. Los planteamientos para futuros desarrollos pudieran parecer ambiciosos, pero las consecuentes versiones del sistema tendrán que mejorar las especificaciones de diseño, almacenamiento y rapidez actual, para permitir una mayor versatilidad del sistema y el fácil acoplamiento de nuevos dispositivos.

Anexos

Los listados de código que se presentan a continuación detallan cada uno de los procesos que son realizados dentro del microprocesador para realizar la interconexión del dispositivo con el entorno denominado Internet.

ANEXO A.- Código ensamblador

```

;*****
;
.include "8515def.inc"
;Reset Handle
;*****
;
.listmac

.def    temp =r16
.def    rtemp =r17
.def    address=r22

;**** Constantes I2C E2PROM y LM75****
.equ    SCLP    = 6      ; SCL número de Pin
.equ    SDAP    = 7      ; SDA número de Pin
.equ    b_dir    = 0      ; bit dirección de transferencia
.equ    TWIrd    = 1
.equ    TWIwr    = 0
.equ    add_sen_temp=$90
.equ    E2promAdd=$A0

/* Constantes de dirección de memoria de archivos en I2CE2PROM
.equ    header_http = 0x0000
.equ    long_main_web_page = 0x0016
.equ    long_cic_gif = 0x03C0
.equ    long_poli_gif = 0x0888
.equ    long_F3_class = 0x0A90
.equ    long_F4_devices_class = 0x0FC4
.equ    long_F5_device_off = 0x1530
.equ    long_F6_device_on = 0x15E2
.equ    long_F8_devices = 0x16F0
.equ    long_F9_temp = 0x16F2
.equ    READ=$00
.equ    WRITE=$01
.equ    MY_IP0=$94      ;dirección IP    148
.equ    MY_IP1=$CC      ;                204
.equ    MY_IP2=$2D      ;                45
.equ    MY_IP3=$40      ;                64

;*****NIC Página 0 lectura de registros
.equ    CR=$00
.equ    CLDA0=$01
.equ    CLDA1=$02
.equ    BNDY=$03
.equ    TSR=$04

```

```
.equ    NCR=$05
.equ    FIFO=$06
.equ    ISR=$07      ;Estado de Registro de interrupción
.equ    CRDA0=$08
.equ    CRDA1=$09
.equ    ID0=$0A
.equ    ID1=$0B
.equ    RSR=$0C
.equ    CNTR0=$0D
.equ    CNTR1=$0E
.equ    CNTR2=$0F
```

```
;/*****NIC Página 0 escritura de registros
```

```
.equ    PSTART=$01
.equ    PSTOP=$02
.equ    TPSR=$04
.equ    TBCR0=$05
.equ    TBCR1=$06
.equ    RSAR0=$08
.equ    RSAR1=$09
.equ    RBCR0=$0A
.equ    RBCR1=$0B
.equ    RCR=$0C
.equ    TCR=$0D
.equ    DCR=$0E
.equ    IMR=$0F
```

```
;/*****NIC Registros página 1
```

```
.equ    PAR0=$01
.equ    PAR1=$02
.equ    PAR2=$03
.equ    PAR3=$04
.equ    PAR4=$05
.equ    PAR5=$06
.equ    CURR=$07
.equ    MAR0=$08
.equ    MAR1=$09
.equ    MAR2=$0A
.equ    MAR3=$0B
.equ    MAR4=$0C
.equ    MAR5=$0D
.equ    MAR6=$0E
.equ    MAR7=$0F
```

```
;/*****NIC página 3 registros
```

```
.equ    CR9346=$01
.equ    LEDS=$06
```

```
;/*****NIC Otros registros
```

```
.equ    NIC_DATA=$20
.equ    NIC_RESET=$28
```

```
;/*****NIC Otros registros
```

```
.equ    RCV_BUF_START=$64
.equ    XMT_BUF_START=$84
```

```

.equ    IORB=$5
.equ    IOWB=$6
.equ    RESETDRV=$7
.equ    as_in=$00
.equ    as_out=$FF

;/** *****
;/** *** Definiciones de protocolos          ***
;/** *****

.equ    ARP=$0806
.equ    IP=$0800
.equ    ICMP=$01
.equ    TCP=$06
.equ    UDP=$17
.equ    ECHO=$08
.equ    REPLY=$00
.equ    TCP_FIN=$01
.equ    TCP_SYN=$02
.equ    TCP_RST=$04
.equ    TCP_PSH=$08
.equ    TCP_ACK=$10
.equ    TCP_URG=$20
.equ    LISTEN=$01
.equ    SYN_SENT=$02
.equ    SYN_RCVD=$03
.equ    ESTAB=$04
.equ    FIN_WAIT_1=$05
.equ    FIN_WAIT_2=$06
.equ    CLOSING=$07
.equ    TIME_WAIT=$08
.equ    CLOSE_WAIT=$09
.equ    LAST_ACK=$0A
.equ    CLOSED=$0B

;***** VARIABLES *****

.equ    ID_0=$0060
.equ    ID_1=$0061          ;misma dirección en my_ip solo
                           necesario al inicio

;/* Variables Ethernet
.equ    Phy_Add=$0060 ;Dirección Física SRAM (60 - 65)
.equ    Source_Add=$0066   ;Dirección Origen SRAM (66 - 6B)
.equ    type=$006C         ;(6C - 6D)

;/*Variables de encabezado IP
.equ    version=$006E
.equ    type_service=$006F
.equ    IP_Length=$0070      ;(70 - 71)
.equ    identification=$0072  ;(72 - 73)
.equ    fragment=$0074 ;(74 - 75)
.equ    TTL=$0076
.equ    IP_protocol=$0077
.equ    IP_checksum=$0078     ;(78 - 79)
.equ    source_ip=$007A       ;(7A - 7D)
.equ    dest_ip=$007E         ;(7E - 81)

;/* ICMP - Encabezado ping

```

```

.equ    icmp_type=$0082
.equ    icmp_code=$0083
.equ    icmp_chksm=$0084    ;(84 - 85)
.equ    icmp_identfier=$0086    ;(86 - 87)
.equ    icmp_sequence=$0088    ;(88 - 89)

/* TCP – Encabezado
.equ    tcp_source_port=$008A    ;(8A - 8B)
.equ    tcp_dest_port=$008C    ;(8C - 8D)
.equ    seq=$008E    ;(8E - 91)B5 - B8
.equ    ack=$0092    ;(92 - 95)B9 - BC
.equ    offset=$0096
.equ    flags=$0097
.equ    window=$0098    ;(98 - 99)BF - C0
.equ    my_ip=$009A    ;My_IP en SRAM (9A - 9D)
.equ    in_ptr=$009E
.equ    out_ptr=$009F
.equ    next_ptr=$00A0
.equ    hdr_len=$00A1
.equ    opt_len=$00A2
.equ    chk=$00A3    ;(A3 - A4)CA - CB
.equ    html_socket=$00A5
.equ    tcp_options=$00A6

.equ    kind=$00A7
.equ    max_seg=$00A8 ;(A8 - A9)cF - D0
.equ    xm_ack=$00AA ;(AA - AD)D1 - D4
.equ    xm_seq=$00AE ;(AE - B1)D5 - D8
.equ    port=$00B2    ;(B2 - B3)D9 - DA
.equ    ip_=$00B4    ;(B4 - B7)DB - DE
.equ    tcp_data_len=$00B8    ;(B8 - B9)DF - E0
.equ    page_len=$00BA;(BA - BB)E1 - E2
.equ    file_number=$00BC
.equ    temperatura=$00BD    ;(BD - C2) 6 bytes
.equ    header_http_ram=$00C3 ;(C3 - D8) 22 bytes
.equ    read_mess=$0A0;ram para mensaje http
.equ    icmp_data=$0A0;(ram para icmp_data)32 bytes
.equ    web_page=$0A0 ;Longitud variable

/****Límite RAM 0200, 96 bytes reservados para pila de datos*//////////

```

```

.macro Carga_Z
    ldi    ZH,high(@0<<1);parte alta de dirección en ZH
    ldi    ZL,low(@0<<1) ;parte baja de dirección en ZL
.endmacro

```

```

.macro Carga_X
    ldi    XH,high(@0) ;parte alta de dirección en XH
    ldi    XL,low(@0) ;parte baja de dirección en XL
.endmacro

```

```

.macro Carga_Y
    ldi    YH,high(@0) ;parte alta de dirección en YH
    ldi    YL,low(@0) ;parte baja de dirección en YL
.endmacro

```

```
rjmp RESET
```

```
RESET:
```

```
ldi temp,low(RAMEND)
out SPL,temp
ldi temp,high(RAMEND)
out SPH,temp ;Inicializa apuntador de pila
rcall Init_AVR
Carga_X header_http_ram
Carga_Z header_http
ldi temp,22
rcall load_hader_http
ldi XL,temperatura
ldi XH,temperatura>>8
rcall init_temp
ldi rtemp,MY_IP0 ;Define sistema IP
sts my_ip,rtemp
ldi rtemp,MY_IP1
sts my_ip+1,rtemp
ldi rtemp,MY_IP2
sts my_ip+2,rtemp
ldi rtemp,MY_IP3
sts my_ip+3,rtemp
rcall inicializa
rcall Read_Phy_Add
rcall inicializa
ldi temp,LISTEN
sts html_socket,temp
ldi address,TCR
ldi temp,0x00
rcall escribe
rjmp RESET
ret
```

```
poll_nic:
```

```
ldi address,CR
ldi temp,0x62 ;página 1
rcall escribe
ldi address,CURR ;Obtiene apuntador acutal entrada
rcall lee
sts in_ptr,rtemp
ldi address,CR
ldi temp,0x22
rcall escribe
ldi address,BNDRY
rcall lee
cp rtemp,temp
brne sigue_poll
ret
sigue_poll:
lds XH,out_ptr
clr XL
ldi temp,READ
rcall remote_DMA_setup
ldi address,NIC_DATA
```



```

rcall lee
clr rtemp
rcall lee
sts next_ptr,rtemp
ldi temp,0x08
rcall discard
ldi temp,0x08
ldi XL,Source_Add
ldi XH,Source_Add>>8      ;(6)Source_Add, (2)Type_of_service
rcall DMA_con_read
lds rtemp,type
cpi rtemp,IP>>8
brne fin_poll
lds rtemp,type+1
cpi rtemp,IP
brne ARP_CHECK
rcall IP_FUN
rjmp fin_poll
ARP_CHECK:
cpi rtemp,ARP
brne fin_poll
rcall ARP_FUN
fin_poll:
ldi address,CR
ldi temp,0x22
rcall escribe
ret

; /* Funcion para manejar llamadas IP
IP_FUN:
rcall lee
sts version,rtemp
andi rtemp,0x0F
clc
rol rtemp
rol rtemp
sts hdr_len,rtemp
subi rtemp,0x14      ;resta 20 (0x14)
sts opt_len,rtemp
lds rtemp,version
clc
ror rtemp
clc
ror rtemp
clc
ror rtemp
clc
ror rtemp
sts version,rtemp
ldi temp,19
ldi XL,type_service ;Type_of_service, (2) IP_length, (2) Identification
ldi XH,type_service>>8      ;(2)Fragment,TTL, IP_Protocol,
                                ;(2)Chksum,(4) Source_IP
                                ;(4) Dest_ip
rcall DMA_con_read
lds rtemp,dest_ip

```

```

    cpi  rtemp,MY_IP0
    brne fin_IP
    lds  rtemp,dest_ip+1
    cpi  rtemp,MY_IP1
    brne fin_IP
    lds  rtemp,dest_ip+2
    cpi  rtemp,MY_IP2
    brne fin_IP
    lds  rtemp,dest_ip+3
    cpi  rtemp,MY_IP3
    brne fin_IP
sigue_IP_sigue:
    lds  temp,IP_Protocol
    cpi  temp,ICMP
    brne TCP_CALL
    rcall ICMP_FUN
    rjmp fin_IP
TCP_CALL:
    cpi  temp,TCP
    brne UDP_CALL
    rcall TCP_FUN
    rjmp fin_IP
UDP_CALL:
    cpi  temp,UDP
    brne fin_IP
    rcall UDP_FUN
    rjmp fin_IP
fin_IP:
    ret

UDP_FUN:
    ret

ICMP_FUN:
    rcall lee
    sts  icmp_type,rtemp
    cpi  rtemp,ECHO
    breq ping_fun
    ret
ping_fun:
    ldi  XL,icmp_code
    ldi  XH,icmp_code>>8
    ldi  temp,7
    rcall DMA_con_read
    ldi  XL,icmp_data
    ldi  XH,icmp_data>>8
    ldi  temp,32
    rcall DMA_con_read
    rcall ping_resp
    ret

ping_resp:
    rcall load_eth_head
    ldi  ZL,0x3C      ;número de bits del paquete IP
    ldi  ZH,0x00
    ldi  rtemp,ICMP    ;tipo de servicio de respuesta

```

```

rcall load_IP_header
ldi temp,0x00
rcall escribe
lds temp,icmp_code
rcall escribe
ldi temp,0xAA    /*Calcular Checksum
sts chk,temp
ldi temp,0xA3
sts chk+1,temp
lds XL,icmp_code
ldi XH,0x00      ;type=0 (respuesta de eco)
rcall chk_calc
lds XL,icmp_identfier+1
lds XH,icmp_identfier
rcall chk_calc
lds XL,icmp_sequence+1
lds XH,icmp_sequence
rcall chk_calc
lds temp,chk
com temp
rcall escribe
lds temp,chk+1
com temp
rcall escribe
Carga_X          icmp_identfier
ldi rtemp,4
rcall DMA_con_write
ldi XL,icmp_data
ldi XH,icmp_data>>>8
ldi rtemp,0x20
rcall DMA_con_write
ldi XL,0x4A      ;envia paquete, longitud 74(0x4A)
ldi XH,0x00
rcall send_packet
ret

/* Procedimiento para llamadas ARP
ARP_FUN:
ldi temp,14
rcall discard
ldi XL,source_ip
ldi XH,source_ip>>>8
ldi temp,4
rcall DMA_con_read
ldi temp,0x06
rcall discard
ldi XL,dest_ip
ldi XH,dest_ip>>>8
ldi temp,4
rcall DMA_con_read
lds temp,dest_ip
lds rtemp,my_ip
cp temp,rtemp
brne fin_ARP
lds temp,dest_ip+1
lds rtemp,my_ip+1

```

```

cp    temp,rtemp
brne  fin_ARP
lds   temp,dest_ip+2
lds   rtemp,my_ip+2
cp    temp,rtemp
brne  fin_ARP
lds   temp,dest_ip+3
lds   rtemp,my_ip+3
cp    temp,rtemp
brne  fin_ARP
rcall ARP_response
fin_ARP:
ret

```

;/*Respuesta ARP

ARP_response:

```

rcall load_eth_head
ldi   temp,ARP>>8
rcall escribe
ldi   temp,ARP
rcall escribe
ldi   temp,0x00
rcall escribe
ldi   temp,0x01
rcall escribe
ldi   temp,0x08
rcall escribe
ldi   temp,0x00
rcall escribe
ldi   temp,0x06
rcall escribe
ldi   temp,0x04
rcall escribe
ldi   temp,0x00
rcall escribe
ldi   temp,0x02
rcall escribe
ldi   rtemp,0x06
ldi   XL,Phy_add
ldi   XH,Phy_add>>8
rcall DMA_con_write
ldi   rtemp,0x04
ldi   XL,my_ip
ldi   XH,my_ip>>8
rcall DMA_con_write
ldi   rtemp,0x06
ldi   XL,source_add
ldi   XH,source_add>>8
rcall DMA_con_write
ldi   rtemp,0x04
ldi   XL,source_ip
ldi   XH,source_ip>>8
rcall DMA_con_write
ldi   temp,0x00
ldi   rtemp,18
arp_response_loop:

```

```

rcall escribe
dec rtemp
brne arp_response_loop
ldi XL,0x3C ;envía paquete, longitud 60
ldi XH,0x00
rcall send_packet
ret

```

Read_ID:

```

push temp
push address
push rtemp
ldi address,0x00
ldi temp,0x21
rcall escribe
ldi address,0x0A
rcall lee
st X+,rtemp
ldi address,0x0B
rcall lee
st X,rtemp
ldi address,0x00
ldi temp,0x22
rcall escribe

pop rtemp
pop address
pop temp
ret

```

Lee:

```

push address
out PORTC,address
in rtemp,PINA
in rtemp,PINA
sbi PORTC,IORB
pop address
ret

```

;/****Recibe Dirección en address, Value in temp

;/*

Escribe:

```

push address
push temp
out PORTA,temp ;coloca vaor en puerto A(PORTA)
ldi temp,as_out
ori address,0x60
out PORTC,address
cbi PORTC,IOWB
sbi PORTC,IOWB
ldi temp,as_in
out DDRA,temp ;define PORTA como entrada
pop temp
pop address
ret

```

```
/*Read physical Address
```

```
/*
```

```
Read_Phy_Add:
```

```
    push address
    push temp
    ldi  XL,0x00      ;lee dirección
    ldi  XH,0x00      ;dirección en registro X:0x0000h
    ldi  temp,READ
    rcall Remote_DMA_Setup
    ldi  XL,Phy_Add
    ldi  XH,Phy_add>>8
    ldi  address,NIC_DATA
    ldi  temp,0x06
do_read:
    dec  temp
    rcall lee
    st   X+,rtemp
    rcall lee
    cpi  temp,0x0
    brne do_read
    pop  temp
    pop  address
    ret
```

```
/****** FUNCTIONS *****/
```

```
/* Region de memoria en registro X, temp=0 READ, diferente WRITE
```

```
Remote_DMA_Setup:
```

```
    push rtemp
    push address
    push temp
    ldi  address,CR
    ldi  temp,0x22    ;Página 0 Abortar DMA
    rcall escribe
    ldi  address,RBCR0
    ldi  temp,0xFF     ;Define número máximo de bytes leer/escribir
    rcall escribe
    ldi  address,RBCR1
    ldi  temp,0xFF     ; Define número máximo de bytes leer/escribir
    rcall escribe
    ldi  address,RSAR0
    mov  temp,XL
    rcall escribe
    ldi  address,RSAR1
    mov  temp,XH
    rcall escribe
    pop  temp
    push temp
    ldi  address,CR
    cpi  temp,READ
    breq DMA_Read
    ldi  temp,0x12
    rjmp DMA_Ex
DMA_Read:
    ldi  temp,0x0A
DMA_Ex:
    rcall escribe
```

```

    pop temp
    pop address
    pop rtemp
    ret
;****Init AVR
;*/
Init_AVR:
    ldi rtemp,as_out
    out DDRC,rtemp ;define PORTC como salida
    ldi rtemp,as_in
    out DDRA,rtemp ;define PORTA como entrada
    sbi PORTC,IOWB
    sbi PORTC,IORB
    cbi PORTC,RESETDRV
    ldi rtemp,as_out
    out DDRB,rtemp
    ret

;****Escribe configuración principal, sin valores de entrada
;*/Sin valores de regreso
inicializa:
    ldi address,CR
    ldi temp,0x21 ;página 0, Aborta DMA, Detiene NIC
    rcall escribe
    ldi address,NIC_RESET
    ldi temp,0xFF
    rcall escribe
    ldi address,DCR
    ldi address,RCR
    ldi temp,0x0C ;Monitor apagado Promiscuo apagado
    rcall escribe ;Acepta Llamadas Multicast, Acepta
    ;Broadcast, Rechaza Runts, Rechaza errores

    ldi address,TCR
    ldi temp,0x02 ;Loop Back Interno
    rcall escribe
    ldi address,BNDRY
    ldi temp,RCV_BUF_START ;inicio de buffer en RAM
    rcall escribe
    ldi address,PSTART
    ldi temp,RCV_BUF_START
    rcall escribe
    ldi address,PSTOP
    ldi temp,XMT_BUF_START;8-bit, espacio para transmitir paquetes
    rcall escribe
    ldi address,ISR
    ldi temp,0xFF ;Borra el registro ISR (Interrupt Status Register)
    rcall escribe
    ldi address,IMR
    ldi temp,0x00 ;Defien registro de Interrupción
    rcall escribe
    ldi address,CR
    ldi temp,0x61 ;// Página 1
    rcall escribe
    ldi address,PAR0 ;Define Physical_Address
    lds temp,Phy_Add
    rcall escribe

```

```

ldi  address,PAR1
lds  temp,Phy_Add+1
rcall escribe
ldi  address,PAR2
lds  temp,Phy_Add+2
rcall escribe
ldi  address,PAR3
lds  temp,Phy_Add+3
rcall escribe
ldi  address,PAR4
lds  temp,Phy_Add+4
rcall escribe
ldi  address,PAR5
lds  temp,Phy_Add+5
rcall escribe
ldi  address,CURR
ldi  temp,0x40      ;Pone NIC en modo de inicio
rcall escribe
ldi  address,CR
ldi  temp,0x22
rcall escribe
ret

/*Descarta valores leidos desde la memoria DMA
/*la cantidad de valores descartados, esta especificado en temp
discard:
    push temp
discard_loop:
    dec  temp
    rcall lee
    cpi  temp,0x0
    brne discard_loop
    pop  temp
    ret

/*Compone el encabezado IP
/*sin valores a la entrada*/
load_IP_header:
    push temp
    push rtemp
    ldi  temp,IP>>8
    rcall escribe
    ldi  temp,IP
    rcall escribe
    ldi  temp,0x45
    rcall escribe
    ldi  temp,0x00
    rcall escribe
    mov  temp,ZH      ;Longitud del paquete IP en registro Z IP_packet
    rcall escribe
    mov  temp,ZL
    rcall escribe
    lds  temp,identification
    rcall escribe
    lds  temp,identification+1
    rcall escribe

```



```

ldi temp,0x00 ;Fragmentación
rcall escribe
ldi temp,0x00
rcall escribe
ldi temp,0xFF ;/*TTL
rcall escribe
mov temp,rtemp ;/*IP_send_protocol
rcall escribe
lds temp,my_ip
sts chk,temp ;Primera entrada al Chksum
lds temp,my_ip+1
sts chk+1,temp
lds XH,my_ip+2 ;suma X al checksum
lds XL,my_ip+3
rcall chk_calc
lds XH,source_ip ;suma X al checksum
lds XL,source_ip+1
rcall chk_calc
lds XH,source_ip+2 ;suma X al checksum
lds XL,source_ip+3
rcall chk_calc
ldi XH,0x45 ;suma X al checksum
ldi XL,0x00
rcall chk_calc
mov XH,ZH
mov XL,ZL
rcall chk_calc
lds XH,identification
lds XL,identification+1
rcall chk_calc
ldi XH,0xFF ;TTL
pop rtemp
push rtemp
mov XL,rtemp ;IP_Protocol
rcall chk_calc
lds temp,chk
com temp
rcall escribe
lds temp,chk+1
com temp
rcall escribe
ldi rtemp,0x04
ldi XL,my_ip
ldi XH,my_ip>>8
rcall DMA_con_write
ldi rtemp,0x04
ldi XL,source_ip
ldi XH,source_ip>>8
rcall DMA_con_write
pop rtemp
pop temp
ret
;/*Longitud de paquete a enviar en Registro X
send_packet:
push rtemp

```

```

push temp
ldi temp,0x22
ldi address,CR
rcall escribe
mov temp,XL
ldi address,TBCR0
rcall escribe
mov temp,XH
ldi address,TBCR1
rcall escribe
ldi temp,XMT_BUF_START
ldi address,TPSR
rcall escribe
ldi temp,0x26
ldi address,CR
rcall escribe
pop temp
pop rtemp
ret

;/*compone encabezado Ethernet*/
;/*sin valores a la entrada*/
load_eth_head:
push temp
push rtemp
ldi XH,XMT_BUF_START
ldi XL,XMT_BUF_START>>8
ldi temp,WRITE
rcall remote_DMA_setup
ldi rtemp,0x06
ldi XL,source_add
ldi XH,source_add>>8
rcall DMA_con_write
ldi rtemp,0x06
ldi XL,Phy_add
ldi XH,Phy_add>>8
rcall DMA_con_write
pop rtemp
pop temp
ret

;/*Obtiene datos TCP, del paquete de entrada
;/*Sin valores de entrada
TCP_FUN:
push temp
push rtemp
rcall lee
sts tcp_source_port,rtemp
rcall lee
sts tcp_source_port+1,rtemp
rcall lee
sts tcp_dest_port,rtemp
rcall lee
sts tcp_dest_port+1,rtemp
cpi rtemp,0x50 ;Solamente puerto HTML=80 (0x50)
breq tcp_sigue

```

```

rjmp tcp_fun_fin
tcp_sigue:
ldi  XL,seq
ldi  XH,seq>>8
ldi  temp,8
rcall DMA_con_read
rcall lee
andi rtemp,0xF0
ror  rtemp
ror  rtemp
sts  offset,rtemp
subi rtemp,0x14
sts  tcp_options,rtemp
rcall lee
andi rtemp,0x3F
sts  flags,rtemp
cpi  rtemp,TCP_SYN
brne tcp_win
clr  rtemp          ;max_seg=0
sts  max_seg,rtemp
sts  max_seg+1,rtemp
tcp_win:
rcall lee
sts  window,rtemp
rcall lee
sts  window+1,rtemp
ldi  temp,0x04
rcall discard      ;descarta checksum y urgent pointer
ldi  temp,0x0
sts  file_number,temp ;define número de archivo a enviar
Carga_X          read_mess;almacena en memoria intermedia
ldi  temp,64
rcall clear_ram
Carga_X          read_mess
ldi  temp,64
rcall DMA_con_read
lds  temp,tcp_options
ldi  XL,0x00
loop_options:
rcall lee
sts  kind,rtemp
cpi  rtemp,0x02
brne sigue_options
rcall lee
lds  temp,flags
andi temp,TCP_SYN ;verifica existencia de SYN
cpi  temp,TCP_SYN
brne option_kind_case
lds  YH,max_seg
rcall lee
add  YH,rtemp
lds  YL,max_seg+1 ;incrementa segundos de espera
rcall lee
add  YL,rtemp
sts  max_seg,YL
brcc inc_kind_case

```

```

inc    YH
sts    max_seg,YH
rjmp   inc_kind_case
option_kind_case:
rcall  lee
rcall  lee
inc_kind_case:
adiw   XL,0x03
sigue_options:
inc    XL
lds    rtemp,tcp_options
cp     XL,rtemp
brmi   loop_options
lds    rtemp,read_mess ;comienza lectura de datos TCP
cpi    rtemp,'G'        ;en espera de cadena "GET"
brne   tcp_listen_response
lds    rtemp,read_mess+1
cpi    rtemp,'E'
brne   tcp_listen_response
lds    rtemp,read_mess+2
cpi    rtemp,'T'
brne   tcp_listen_response
lds    rtemp,read_mess+3
cpi    rtemp,''
brne   tcp_listen_response
lds    rtemp,read_mess+4
cpi    rtemp','
brne   tcp_listen_response
lds    rtemp,read_mess+5
cpi    rtemp,''
breq   tcp_listen_response
cpi    rtemp','?'
brne   tcp_listen_file
rcall  review_devices ;verifica estado de dispositivos
rjmp   tcp_listen_response
tcp_listen_file:
rcall  get_file_name ;Archivo requerido, obtiene nombre
tcp_listen_response:
rcall  tcp_response
tcp_fun_fin:
pop    rtemp
pop    temp
ret

; /*Establece el estado del socket_html
; /*SIN VALORES DE ENTRADA
tcp_response:
push   temp
push   rtemp
rcall  tcp_length_calc
lds    temp,html_socket
cpi    temp,LISTEN
brne   TCP_SYN_CHECK
rcall  tcp_listen_case
rjmp   TCP_RESPONSE_END
TCP_SYN_CHECK:

```

```

    cpi    temp,SYN_RCVD
    brne   TCP_ESTAB_CHECK
    rcall  tcp_syn_rcvd_case
    rjmp   TCP_RESPONSE_END
TCP_ESTAB_CHECK:
    cpi    temp,ESTAB
    brne   TCP_RESPONSE_END
    rcall  tcp_estab_case
TCP_RESPONSE_END:
    pop    rtemp
    pop    temp
    ret

/*Calcula longitud de paquete TCP
/*sin valores de entrada
tcp_length_calc:
    push temp
    push rtemp
    ldi    temp,0x00
    sts    tcp_data_len+1,temp
    sts    tcp_data_len,temp
    lds    temp,hdr_len
    lds    rtemp,offset
    add    temp,rtemp
    lds    rtemp,IP_Length+1
    sub    rtemp,temp
    brcc   fin_tcp_leng
    lds    temp,IP_Length
    dec    temp
    sts    tcp_data_len,temp
fin_tcp_leng:
    sts    tcp_data_len+1,rtemp
    pop    rtemp
    pop    temp
    ret

/*Verifica estado de banderas del mensaje
/*espera SYN
tcp_listen_case:
    push temp
    lds    temp,flags
    andi   temp,TCP_SYN
    sbrs   temp,1
    rjmp   TCP_listen_case_end
    lds    temp,flags
    andi   temp,TCP_ACK
    sbrc   temp,4
    rjmp   TCP_listen_case_end
    lds    temp,flags
    andi   temp,TCP_RST
    sbrc   temp,2
    rjmp   TCP_listen_case_end
    lds    temp,flags
    andi   temp,TCP_FIN
    sbrc   temp,0
    rjmp   TCP_listen_case_end

```

```
rcall tcp_listen
TCP_listen_case_end:
pop temp
ret

;/*verifica estado de banderas del mensaje
;verifica SYN_RCV
tcp_syn_rcvd_case:
push rtemp
push temp
lds rtemp,port
lds temp,tcp_source_port
cp temp,rtemp
brne tcp_syn_rcvd_end
lds rtemp,port+1
lds temp,tcp_source_port+1
cp temp,rtemp
brne tcp_syn_rcvd_end
lds rtemp,ip_
lds temp,source_ip
cp temp,rtemp
brne tcp_syn_rcvd_end
lds rtemp,ip_+1
lds temp,source_ip+1
cp temp,rtemp
brne tcp_syn_rcvd_end
lds rtemp,ip_+2
lds temp,source_ip+2
cp temp,rtemp
brne tcp_syn_rcvd_end
lds rtemp,ip_+3
lds temp,source_ip+3
cp temp,rtemp
brne tcp_syn_rcvd_end
lds temp,flags
andi temp,TCP_ACK
sbrs temp,0x04
rjmp tcp_syn_rcvd_end
lds temp,flags
andi temp,TCP_SYN
sbrc temp,0x01
rjmp tcp_syn_rcvd_end
lds temp,flags
andi temp,TCP_RST
sbrc temp,0x02
rjmp tcp_syn_rcvd_end
lds temp,flags
andi temp,TCP_FIN
sbrc temp,0x00
rjmp tcp_syn_rcvd_end
rcall tcp_rcvd
tcp_syn_rcvd_end:
pop temp
pop rtemp
ret
```

```

/*verifica conexión establecida, establece autómata
tcp_estab_case:
    push temp
    push rtemp
    lds temp,flags
    andi temp,TCP_ACK
    sbrs temp,0x04
    rjmp tcp_estab_end
    lds temp,flags
    andi temp,TCP_SYN
    sbrc temp,0x01
    rjmp tcp_estab_end
    lds temp,flags
    andi temp,TCP_RST
    sbrc temp,0x02
    rjmp tcp_estab_end
    lds temp,seq
    lds rtemp,xm_ack
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,seq+1 ;lee numero de secuencia (4 bytes)
    lds rtemp,xm_ack+1
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,seq+2
    lds rtemp,xm_ack+2
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,seq+3
    lds rtemp,xm_ack+3
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,ack ;lee numero reconocimiento (4)
    lds rtemp,xm_seq
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,ack+1
    lds rtemp,xm_seq+1
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,ack+2
    lds rtemp,xm_seq+2
    cpse temp,rtemp
    rjmp tcp_estab_end
    lds temp,ack+3
    lds rtemp,xm_seq+3
    cp temp,rtemp
    brne tcp_estab_end ;no iguales, termina conexión
    lds temp,port
    lds rtemp,tcp_source_port
    cp temp,rtemp
    brne tcp_estab_end
    lds temp,port+1
    lds rtemp,tcp_source_port+1
    cp temp,rtemp ;verifica puerto requerido
    brne tcp_estab_end

```

```

lds temp,ip_
lds rtemp,source_ip ;verifica dirección IP (4)
cp temp,rtemp
brne tcp_estab_end
lds temp,ip_+1
lds rtemp,source_ip+1
cp temp,rtemp
brne tcp_estab_end
lds temp,ip_+2
lds rtemp,source_ip+2
cp temp,rtemp
brne tcp_estab_end
lds temp,ip_+3
lds rtemp,source_ip+3
cp temp,rtemp
brne tcp_estab_end
rcall tcp_estab
tcp_estab_end:
pop rtemp
pop temp
ret

```

;automata en estado de escucha, obtiene valores del paquete
tcp_listen:

```

push temp
push rtemp
lds temp,seq+3 ;intercambia seq con ack
sts xm_ack+3,temp
lds temp,seq+2
sts xm_ack+2,temp
lds temp,seq+1
sts xm_ack+1,temp
lds temp,seq+0
sts xm_ack+0,temp
lds temp,tcp_source_port ;define puerto de entrada
sts port,temp
lds temp,tcp_source_port+1
sts port+1,temp
lds temp,source_ip ;define IP de entrada
sts ip_,temp
lds temp,source_ip+1
sts ip_+1,temp
lds temp,source_ip+2
sts ip_+2,temp
lds temp,source_ip+3
sts ip_+3,temp
ldi XL,xm_ack ;establece ack y longitud de datos
ldi XH,xm_ack>>8
ldi YL,tcp_data_len
ldi YH,tcp_data_len>>8
rcall suma_registro_16_a_32
ldi XL,xm_ack
ldi XH,xm_ack>>8
rcall inc_reg_32 ;incrementa en 1, registro xm_ack
ldi XL,xm_seq
ldi XH,xm_seq>>8

```



```

rcall inc_reg_32
rcall load_eth_head
ldi  ZL,0x30      ;numero de bits del paquete 48 IP
                    ;20 IPH, 20TCPH, 8 options)

ldi  ZH,0x00
ldi  rtemp,TCP    ;tipo de servicio de respuesta
rcall load_IP_header
ldi  temp,0x07
sts  chk,temp
ldi  temp,0xB8
sts  chk+1,temp
ldi  YL,0x1C      ;longitud de mensaje
ldi  YH,0x00
ldi  ZL,TCP_ACK   ;establece banderas para conexión
ori  ZL,TCP_SYN
ldi  ZH,0x70
rcall load_TCP_header      ;Carga encabezado TCP
ldi  temp,0x02      ;Escribe en DMA secuencia inicial
rcall escribe
ldi  temp,0x04
rcall escribe
ldi  temp,0x05
rcall escribe
ldi  temp,0xB4
rcall escribe
ldi  temp,0x00
rcall escribe
ldi  temp,0x00
rcall escribe
ldi  temp,0x00
rcall escribe
ldi  temp,0x00
rcall escribe
ldi  XL,0x3E      ;enviar paquete longitud 62h
ldi  XH,0x00
rcall send_packet
ldi  XL,xm_seq    ;incrementa en uno numero de SEQ
ldi  XH,xm_seq>>8
rcall inc_reg_32
ldi  temp,SYN_RCVD      ;define SYN_REC en html_socket
sts  html_socket,temp
pop  rtemp
pop  temp
ret

/*Html_socket en estado de recibido, compone mensaje de respuesta
tcp_rcvd:
push rtemp
push temp
ldi  XL,xm_ack      ;ack = ack +data_len
ldi  XH,xm_ack>>8
ldi  YL,tcp_data_len
ldi  YH,tcp_data_len>>8
rcall suma_registro_16_a_32
ldi  temp,ESTAB
sts  html_socket,temp

```

```

pop temp
pop rtemp
ret

```

;Html_socket en establecido, extrae datos finales de paquete

tcp_estab:

```

push temp
push rtemp
rcall load_y_file    ;dirección i2c para archivo
Carga_X page_len
ldi  ZH,0x00
ldi  ZL,0x02
rcall read_E2prom    ;obtiene longitud de archivo
push YL
push YH
ldi  XL,xm_ack        ;incrementa número de secuencia
ldi  XH,xm_ack>>8
ldi  YL,tcp_data_len
ldi  YH,tcp_data_len>>8
rcall suma_registro_16_a_32
rcall load_eth_head
lds  ZL,page_len+1
lds  ZH,page_len
adiw ZH:ZL,0x28        ;0x28 longitud IP(20)+TCP(20)
ldi  rtemp,TCP          ;tipo de servicio de respuesta
rcall load_IP_header
lds  temp,file_number    ;archivo 0-9???
cpi  temp,0x09
breq chk_temperatura ;no archivo, lectura de temp
cpi  temp,0x08
breq chk_devices      ;y estado de puertos
tcp_estab_chk:
pop  YH
pop  YL
Carga_X chk
ldi  ZH,0x00
ldi  ZL,0x02
rcall read_E2prom
rjmp no_temperatura
chk_devices:          ;obtiene estado de puerto digital
Carga_X              chk
pop  YH
pop  YL
ldi  temp,0xDA
st   X+,temp
ldi  temp,0x2F
st   X+,temp
in   temp,PORTB
ldi  XH,0x00
mov  XL,temp
rcall chk_calc
rjmp no_temperatura
chk_temperatura:      ;lectura de sensor térmico
pop  YH
pop  YL
Carga_X              chk

```

```

ldi temp,0xDA
st X+,temp
ldi temp,0x2F
st X+,temp
Carga_X      temperatura
rcall lee_sensor_temp
rcall write_temp
lds XH,temperatura
lds XL,temperatura+1
rcall chk_calc
lds XH,temperatura+2
lds XL,temperatura+3
rcall chk_calc
lds XH,temperatura+4
lds XL,temperatura+5
rcall chk_calc
no_devices:
no_temperatura: ;requiere archivo
push YL
push YH
lds YL,page_len+1
lds YH,page_len
ldi ZL,TCP_ACK
ori ZL,TCP_FIN
ldi ZH,0x05
rcall load_TCP_header
Carga_Y header_http
ldi ZL,22
ldi ZH,0x00
ldi XL,0x00
ldi XH,0x00
rcall read_E2prom
tcp_estab_file:
lds ZL,page_len+1
lds ZH,page_len
clc
subi ZL,22
brcc write_whole_page
subi ZH,1
write_whole_page: ;escribe pagina html completa
lds temp,file_number
cpi temp,0x09
brne write_who_page
Carga_X      temperatura
pop YH
pop YL
ld temp,X+
rcall escribe
ld temp,X+
rcall escribe
ld temp,X+
rcall escribe
ld temp,X+
rcall escribe
ld temp,X+
rcall escribe

```

```

ld    temp,X+
rcall escribe
rjmp send_web_page
write_who_page:    ;escribe datos de archivo
lds   temp,file_number
cpi   temp,0x08
brne  write_who1_page
pop   YH
pop   YL
ldi   temp,0x00
rcall escribe
in    temp,PORTB    ;escribe estado de puertos
rcall escribe
rjmp send_web_page
write_who1_page:
pop   YH
pop   YL
ldi   XH,0x00
ldi   XL,0x00
rcall read_E2prom
send_web_page:
lds   XL,page_len+1
lds   XH,page_len
adiw  XH:XL,0x36    ;0x36(14ETH, 20IP, 20TCP)+page_len
rcall send_packet
ldi   XL,xm_ack
ldi   XH,xm_ack>>8
ldi   YL,page_len
ldi   YH,page_len>>8
rcall suma_registro_16_a_32
ldi   temp,LISTEN
sts   html_socket,temp
pop   rtemp
pop   temp
ret
; /*Recibe longitud de pagina web en el registro Y
; /*Dirección de memoria de programa en Z
write_web_page:
push temp
push rtemp
push XL
push XH
clr   XL
clr   XH
clc
sigue_web_page:
lpm
mov  temp,r0
rcall escribe
adiw ZH:ZL,0x01
adiw XH:XL,0x01
cp   XL,YL
brne sigue_web_page
cp   XH,YH
brne sigue_web_page
pop  XH

```

```

    pop  XL
    pop  rtemp
    pop  temp
    ret
; /*Recibe en X dirección de memoria en Ram del registro de 32 bits
; /*Recibe en Y DMR del registro de 16 bit
; /*Resultado almacenado en localidad de memoria registro de 32 bits
suma_registro_16_a_32:
    push rtemp
    push temp
    push ZH
    push ZL
    push YH
    push YL
    push XH
    push XL
    ld    temp,X+
    ld    temp,X+
    ld    ZH,X+
    ld    ZL,X
    ld    temp,Y+
    ld    rtemp,Y
    ld    YL,-X
    ld    YL,-X
    ld    YL,-X
    ld    YH,X+
    ld    YL,X
    clr   XL
    clc
    add   ZL,rtemp
    adc   ZH,temp
    adc   YL,XL
    adc   YH,XL
    pop   XL
    pop   XH
    st    X+,YH
    st    X+,YL
    st    X+,ZH
    st    X+,ZL
    push  XH
    push  XL
    pop   XL
    pop   XH
    pop   YL
    pop   YH
    pop   ZL
    pop   ZH
    pop   temp
    pop   rtemp
    ret

inc_reg_32:
    push XL
    push XH
    push YL
    push YH

```

```

push ZL
push ZH
push temp
ld  ZH,X+
ld  ZL,X+
ld  YH,X+
ld  YL,X+
clc
inc  YL
brne fin_inc_reg_32
inc  YH
brne fin_inc_reg_32
inc  ZL
brne fin_inc_reg_32
inc  ZH
fin_inc_reg_32:
st  -X,YL
st  -X,YH
st  -X,ZL
st  -X,ZH
pop  temp
pop  ZH
pop  ZL
pop  YH
pop  YL
pop  XH
pop  XL
ret
; /*tcp_chksum almacenado en chk
; /*en registro Y TCP_length
; en registro Z data flags
load_TCP_header:
lds  XH,my_ip      ;suma my_ip a CHK
lds  XL,my_ip+1
rcall chk_calc
lds  XH,my_ip+2
lds  XL,my_ip+3
rcall chk_calc
lds  XH,source_ip  ; suma source_ip a CHK
lds  XL,source_ip+1
rcall chk_calc
lds  XH,source_ip+2
lds  XL,source_ip+3
rcall chk_calc
ldi  XH,0x00
ldi  XL,TCP
rcall chk_calc      ;suma protocolo a CHK
mov  XH,YH
mov  XL,YL
rcall chk_calc
lds  XH,tcp_dest_port
lds  XL,tcp_dest_port+1      ;suma tcp_dest_port a CHK
rcall chk_calc
lds  XH,tcp_source_port
lds  XL,tcp_source_port+1
rcall chk_calc

```

```
lds XH,xm_seq
lds XL,xm_seq+1
rcall chk_calc
lds XH,xm_seq+2
lds XL,xm_seq+3
rcall chk_calc
lds XH,xm_ack
lds XL,xm_ack+1
rcall chk_calc
lds XH,xm_ack+2
lds XL,xm_ack+3
rcall chk_calc
mov XH,ZH      ;data Flags almacenada registro Z
mov XL,ZL
rcall chk_calc
ldi XH,0x05
ldi XL,0xB4
rcall chk_calc
lds temp,tcp_dest_port
rcall escribe
lds temp,tcp_dest_port+1
rcall escribe
lds temp,tcp_source_port
rcall escribe
lds temp,tcp_source_port+1
rcall escribe
lds temp,xm_seq
rcall escribe
lds temp,xm_seq+1
rcall escribe
lds temp,xm_seq+2
rcall escribe
lds temp,xm_seq+3
rcall escribe
lds temp,xm_ack
rcall escribe
lds temp,xm_ack+1
rcall escribe
lds temp,xm_ack+2
rcall escribe
lds temp,xm_ack+3
rcall escribe
mov temp,ZH
rcall escribe
mov temp,ZL
rcall escribe
ldi temp,0x05
rcall escribe
ldi temp,0xb4
rcall escribe
lds temp,chk
com temp
rcall escribe
lds temp,chk+1
com temp
rcall escribe
```

```

    ldi    temp,0x00
    rcall escribe
    rcall escribe
    ret

; /*Lee estado de dispositivos
review_devices:
    push temp
    push rtemp
    ldi    rtemp,0x00
    Carga_X read_mess
    adiw XH:XL,0x06
review_devices_loop:
    ld     temp,X+
    cpi    temp,'D'
    brne  fin_review_devices
    ld     temp,X+
    subi   temp,0x30
    rcall mask_bit
    or     rtemp,temp
    adiw XH:XL,0x04
    rjmp  review_devices_loop
fin_review_devices:
    out    PORTB,rtemp
    pop    rtemp
    pop    temp
    ret

; /*Obtiene el numero archivo a leer
get_file_name:
    push XH
    push XL
    push temp
    Carga_X read_mess
    adiw XH:XL,0x05
    ld     temp,X+
    cpi    temp,'F'
    brne  get_file_name_end
    ld     temp,X
    subi   temp,0x30
    sts    file_number,temp
get_file_name_end:
    pop    temp
    pop    XL
    pop    XH
    ret

mask_bit:
    push rtemp
    ldi    rtemp,0x01
mask_bit_loop:
    cpi    temp,0x00
    breq   mask_bit_end
    rol    rtemp
    dec    temp
    rjmp   mask_bit_loop

```



```

mask_bit_end:
mov temp,rtemp
pop rtemp
ret

/*Recibe en X region de ram desde donde se limpia RAM, Temp
/*cantidad de datos contiguos X ram que seran borrados
clear_ram:
push XH
push XL
push temp
push rtemp
ldi rtemp,0x00
clear_ram_loop:
st X+,rtemp
dec temp
brne clear_ram_loop
pop rtemp
pop temp
pop XL
pop XH
ret

/* recibe en Z dirección de memoria de programa de donde
/* se va a tomar el header_http, en temp longitud de encabezado
/* recibe en X dirección de memoria RAM donde se va a guardar encabezado
load_hader_http:
loop_load_header_http:
lpm
st X+,r0
adiw ZH:ZL,0x01
dec temp
brne loop_load_header_http
ret

TWI_hp_delay:
ldi r20,10
TWI_hp_delay_loop:
dec r20
brne TWI_hp_delay_loop
ret

TWI_qp_delay:
ldi r20,5
TWI_qp_delay_loop:
dec r20
brne TWI_qp_delay_loop
ret

TWI_rep_start:
sbi DDRD,SCLP ; SCL nivel bajo
cbi DDRD,SDAP ; Libera SDA
rcall TWI_hp_delay ; retardo de medio periodo
cbi DDRD,SCLP ; libera SCL
rcall TWI_qp_delay ; retardo de cuarto de periodo

```

```

TWI_start:
    mov r21,r18      ; copia dirección a transmitir
    sbi  DDRD,SDAP    ; SDA a nivel bajo
    rcall TWI_qp_delay ; retardo de cuarto de periodo
TWI_write:
    sec              ; define bandera de acarreo
    rol  r21          ; desplaza en carry saca un bit
    rjmp TWI_write_first
TWI_write_bit:
    lsl  r21          ; si registro transmisión vacío
TWI_write_first:
    breq TWI_get_ack  ; obtener acknowledge
    sbi  DDRD,SCLP    ; SCL a nivel bajo
    brcc TWI_write_low ; si el bit esta en alto
    nop              ; (ecualiza numero de ciclos)
    cbi  DDRD,SDAP    ; libera SDA
    rjmp TWI_write_high
TWI_write_low:
    ;
    sbi  DDRD,SDAP    ; SDA a nivel bajo
    rjmp TWI_write_high ; (ecualiza numero de ciclos)
TWI_write_high:
    rcall TWI_hp_delay ; retardo medio periodo
    cbi  DDRD,SCLP    ; libera SCL
    rcall TWI_hp_delay ; retardo medio periodo
    rjmp TWI_write_bit
TWI_get_ack:
    sbi  DDRD,SCLP    ; SCL a nivel bajo
    cbi  DDRD,SDAP    ; libera SDA
    rcall TWI_hp_delay ; retardo medio periodo
    cbi  DDRD,SCLP    ; libera SCL

TWI_get_ack_wait:
    sbis PIND,SCLP    ; espera a que SCL en alto
    ;(Dado el caso inserta estados de espera)
    rjmp TWI_get_ack_wait
    clc              ; limpia bandera carry
    sbic PIND,SDAP    ; si SDA alto
    sec              ; define bandera carry
    rcall TWI_hp_delay ; retardo medio periodo
    ret

TWI_do_transfer:
    sbrs r18,b_dir     ; si dir = write
    rjmp TWI_write      ; a escribir datos
TWI_read:
    rol  r19           ; almacena acknowledge
    ; (usado por TWI_put_ack)
    ldi  r21,0x01      ; dato = 0x01
TWI_read_bit:
    ; hacer
    sbi  DDRD,SCLP    ; SCL bajo
    rcall TWI_hp_delay ; retardo medio periodo
    cbi  DDRD,SCLP    ; libera SCL
    rcall TWI_hp_delay ; retardo medio periodo
    clc              ; limpia bandera carry
    sbic PIND,SDAP    ; si SDA esta en alto
    sec              ; define bandera decarry

```

```

    rol    r21            ;almacena bit de dato
    brcc   TWI_read_bit   ;mientras no reciba registro lleno
TWI_put_ack:
    sbi    DDRD,SCLP     ; SCL bajo
    ror    r19            ; obtiene bit de estado
    brcc   TWI_put_ack_low ; si bit bajo ir bajo
    cbi    DDRD,SDAP     ;libera SDA
    rjmp   TWI_put_ack_high
TWI_put_ack_low:        ; sino
    sbi    DDRD,SDAP     ; SDA bajo
TWI_put_ack_high:
    rcall  TWI_hp_delay  ; retardo medio periodo
    cbi    DDRD,SCLP     ; libera SCL
TWI_put_ack_wait:
    sbis   PIND,SCLP     ; espera SCL alto
    rjmp   TWI_put_ack_wait
    rcall  TWI_hp_delay  ; retardo medio periodo
    ret

TWI_stop:
    sbi    DDRD,SCLP     ; SCL bajo
    sbi    DDRD,SDAP     ; SDA bajo
    rcall  TWI_hp_delay  ; retardo medio periodo
    cbi    DDRD,SCLP     ; libera SCL
    rcall  TWI_qp_delay  ; retardo cuarto de periodo
    cbi    DDRD,SDAP     ; libera SDA
    rcall  TWI_hp_delay  ; retardo medio periodo
    ret

TWI_init:
    clr    r19            ; limpia registro de estado TWI
                        ;(usado como registro temporal)
    out    PORTD,r19      ; define pins TWI a collector abierto
    cbi    DDRD,SCLP
    cbi    DDRD,SDAP
    ret

;***** Recibe valor de temperatura en temp, direccin de memoria RAM en X
write_temp:
    push temp
    push rtemp
    push XL
    push XH
    ldi    rtemp,'0'
    adiw   XH:XL,0x04
    clc
    ror    temp
    brcc   write_temp_sigue
    ldi    rtemp,'5'
write_temp_sigue:
    st     X,rtemp
    pop    XH
    pop    XL
    push   XL
    push   XH
    ldi    rtemp,' '
    cpi    temp,0x64

```

```

brmi write_temp_sigue_0
subi temp,0x64
ldi rtemp,'1'
write_temp_sigue_0:
st X+,rtemp
cpi temp,0x00
brne write_temp_sigue_1
ldi rtemp,'0'
st X+,rtemp
st X+,rtemp
rjmp write_temp_end
write_temp_sigue_1:
ldi rtemp,0x00
write_temp_loop:
cpi temp,0x0A
brmi write_temp_subi
subi temp,0x0A
inc rtemp
rjmp write_temp_loop
write_temp_subi:
push temp
ldi temp,0x30
add rtemp,temp
st X+,rtemp
pop temp
ldi rtemp,0x30
add temp,rtemp
st X+,temp
write_temp_end:
pop XH
pop XL
pop rtemp
pop temp
ret

```

;/*Recibe en registro X, dirección de memoria ram donde se ;/*inicializala cadena de temperatura
init_temp:

```

push rtemp
push XL
push XH
ldi rtemp,'X'
st X+,rtemp
ldi rtemp,'X'
st X+,rtemp
ldi rtemp,'X'
st X+,rtemp
ldi rtemp',''
st X+,rtemp
ldi rtemp,'X'
st X+,rtemp
ldi rtemp,'C'
st X,rtemp
pop XH
pop XL
pop rtemp
ret

```

;Lee Memoria E2PROM, Si registro X es igual a cero manda valor a
 ;NIC_DATA, X diferente de cero, almacena en RAM en Y se especifica
 ;direccion de inicio E2PROM desde donde se va a leer la cantidad
 ;de bytes especificada por Z

```
read_E2prom:
    push r0
    push r1
    push temp
    push r16
    push r17
    push r18
    push r19
    push r21
    mov r0,ZH
    mov r1,ZL
    clr  ZH
    clr  ZL
    rcall TWI_init
    ldi  r18,E2promAdd+TWIwr ; define dirección de dispositivo
                                ;y escribe
    rcall TWI_start          ; envia condicion de inicio
    mov r21,YH              ; escribe palabra de dirección
    rcall TWI_do_transfer; ejecuta transferencia
    mov r21,YL              ; escribe palabra de dirección
    rcall TWI_do_transfer; ejecuta transferencia
    rcall TWI_stop          ; envia condicion de paro y libera
                                ; el bus

    loop_read_E2prom:
    ldi  r18,E2promAdd+TWIrd ; Define dirección de dispositivo
                                ; y lee
    rcall TWI_rep_start ; envia condicion de inicio y suma
    sec
    rcall TWI_read      ; Ejecuta transferencia (lee)
    cpi  XL,0x00
    brne write_RAM
    cpi  XH,0x00
    brne write_RAM
    mov temp,r21
    rcall escribe
    rjmp continue_E2PROM
write_RAM:
    st  X+,r21
continue_E2PROM:
    rcall TWI_stop      ; Envia condición de paro y libera
                                ; el bus

    adiw YH:YL,0x01
    adiw ZH:ZL,0x01
    cp  r1,ZL
    brne loop_read_E2prom
    cp  r0,ZH
    breq end_read_E2prom
    rjmp loop_read_E2prom
end_read_E2prom:
    pop r21
    pop r19
```

```

    pop r18
    pop r17
    pop r16
    pop temp
    pop r1
    pop r0
    ret
; /* Lee sensor de temperatura I2C, regresando en temp, valor de
; /* temperatura recibe en r18 TWIadr, r20 TWIdelay
lee_sensor_temp:
    push rtemp
    push r18
    push r19
    push r20
    push r21
    rcall TWI_init      ; inicializa interface TWI
    ldi r18,add_sen_temp+TWIrd      ; define dirección dispositivo y
                                ; lee
    rcall TWI_start     ; envia condicion de inicio
    rcall TWI_read      ; Ejecuta Transferencia (lee)
    mov temp,r21
    sec                 ; define no acknowledge (lee si
                                ; esta seguido por condicion de
                                ; paro)
    rcall TWI_read      ; Ejecuta transferencia (lee)
    mov rtemp,r21
    clc
    rol rtemp
    rol temp
    rcall TWI_stop      ; Envia condicion de paro y libera el bus
    pop r21
    pop r20
    pop r19
    pop r18
    pop rtemp
    ret

; /* carga en el registro Y la dirección de memoria E2PROM I2C de acuerdo al valor almacenado
; /* file_number
load_y_file:
    lds temp,file_number
    cpi temp,0x00
    brne load_y_file_1
    Carga_Y long_main_web_page
    rjmp load_y_file_long
load_y_file_1:
    cpi temp,0x01
    brne load_y_file_2
    Carga_Y long_poli_gif
    rjmp load_y_file_long
load_y_file_2:
    cpi temp,0x02
    brne load_y_file_3
    Carga_Y long_cic_gif
    rjmp load_y_file_long
load_y_file_3:

```

```
cpi temp,0x03
brne load_y_file_4
Carga_Y long_F3_class
rjmp load_y_file_long
load_y_file_4:
cpi temp,0x04
brne load_y_file_5
Carga_Y long_F4_devices_class
rjmp load_y_file_long
load_y_file_5:
cpi temp,0x05
brne load_y_file_6
Carga_Y long_F5_device_off
rjmp load_y_file_long
load_y_file_6:
cpi temp,0x06
brne load_y_file_8
Carga_Y long_F6_device_on
rjmp load_y_file_long
load_y_file_8:
cpi temp,0x08
brne load_y_file_9
Carga_Y long_F8_devices
rjmp load_y_file_long
load_y_file_9:
cpi temp,0x09
brne load_y_file_long
Carga_Y long_F9_temp
rjmp load_y_file_long
load_y_file_long:
ret
```

ANEXO B. Capa de Soldadura.

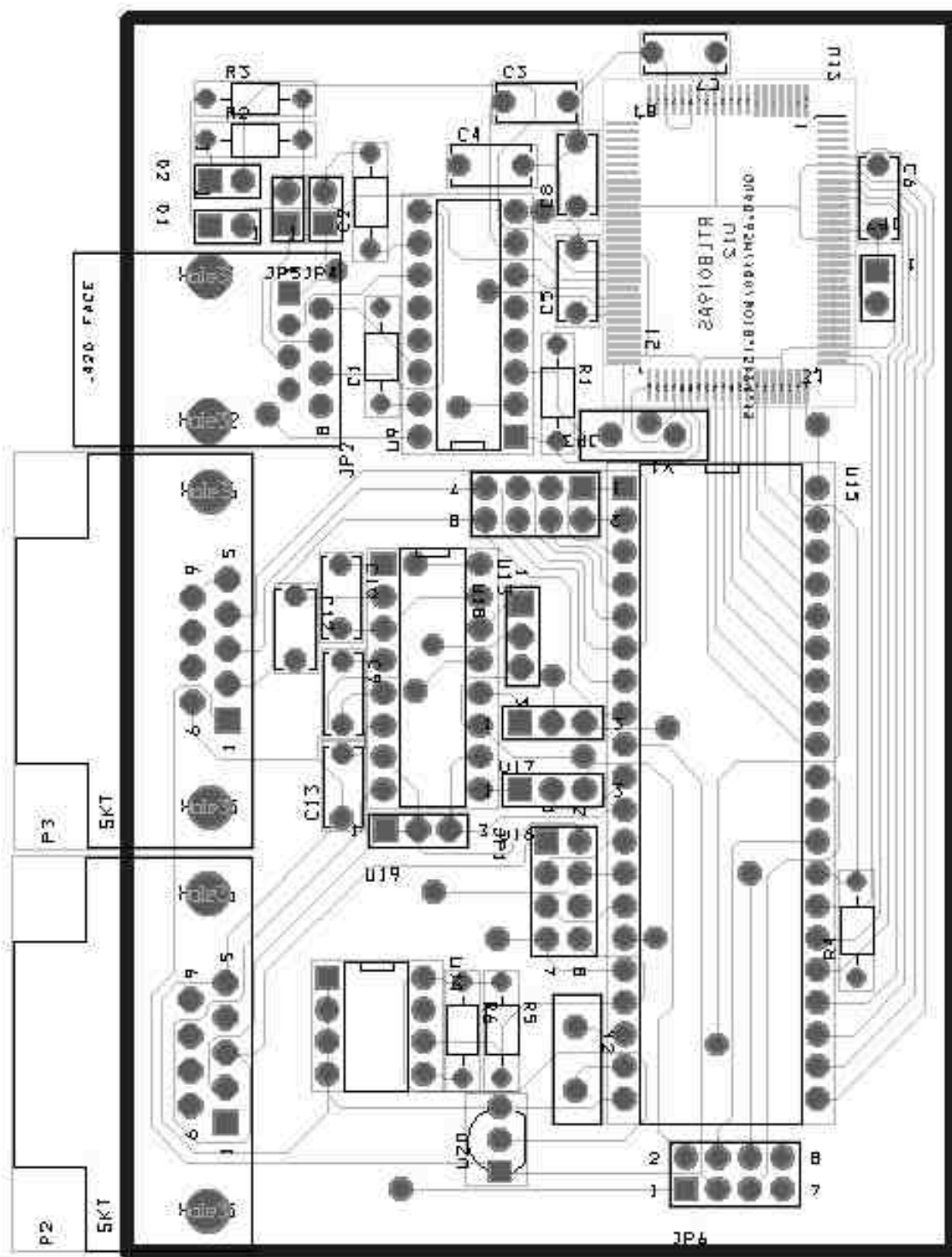


Figura 36. Capa de Soldadura

ANEXO C. Capa de componentes

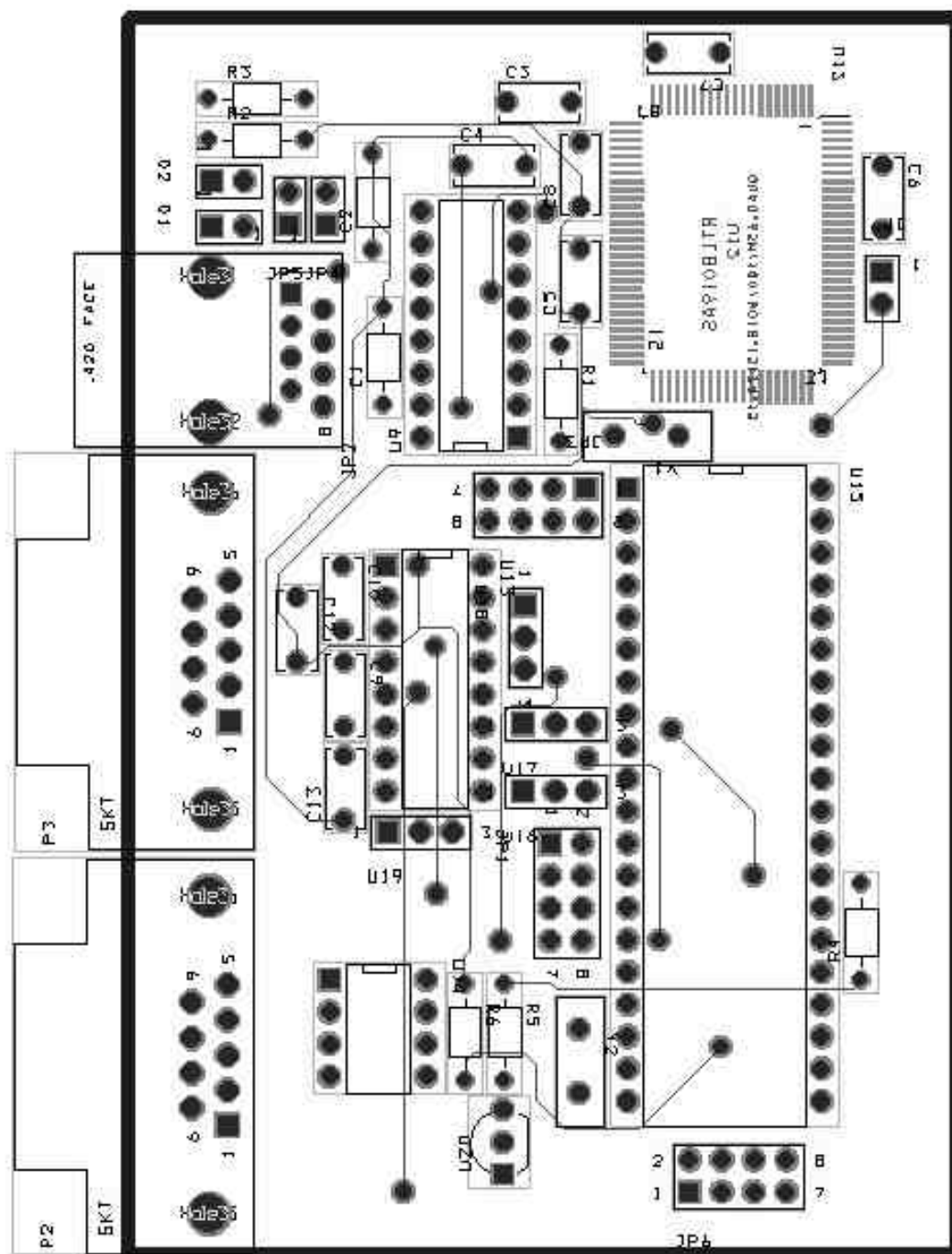
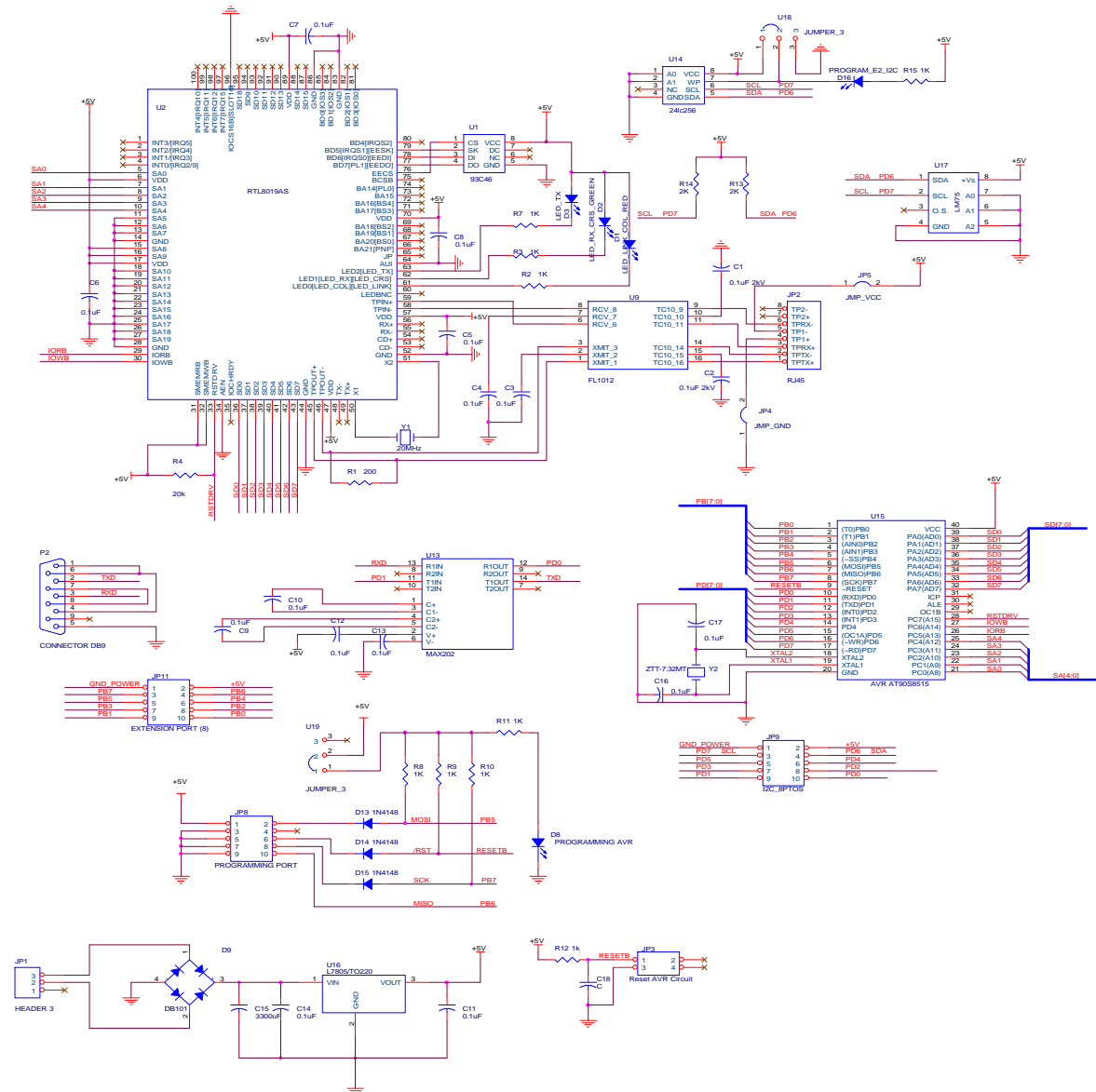


Figura 37. Capa de componentes.

ANEXO D. Diagramas Electrónicos



ANEXO E. Código Java para el control del puerto digital

```
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.net.*;

public class F4 extends Applet{
    DataInputStream datainput;
    URL fileURL;
    private Image device_off;
    private Image device_on;
    int value=0;
    public void init(){
        try{
            fileURL = new URL("http://148.204.45.64/F8");
        }
        catch(MalformedURLException e){}
        device_off=getImage(getDocumentBase(),"F5");
        device_on=getImage(getDocumentBase(),"F6");
    }
    public void start(){
        try{
            datainput = new DataInputStream( fileURL.openStream());
            value= (int)datainput.readChar();
            datainput.close();
        }
        catch (IOException e){}
    }
    public void paint(Graphics g){
        int x=15,y=15;
        for(int i=0;i<8;i++){
            if(((int)Math.pow(2,i))&value)!=0)
                g.drawImage(device_on,x,y,this);
            else
                g.drawImage(device_off,x,y,this);
            x+=35;
        }
    }
}
```

ANEXO F. Código Java para lectura de la temperatura

```
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.net.*;

public class F3 extends Applet{
    DataInputStream datainput;
    URL              fileURL;
    TextArea         Display;

    public void init(){
        try{
            fileURL = new URL("http://148.204.45.64/F9");
        }
        catch(MalformedURLException e){ }
        setBackground(java.awt.Color.white );
        Display = new TextArea("",1,5,TextArea.SCROLLBARS_NONE);
        Display.setEditable(false);
        add(Display);
    }
    public void start(){
        String      texto;
        try{
            datainput = new DataInputStream( fileURL.openStream());
            while(( texto = datainput.readLine())!=null)
                Display.appendText(texto);
            datainput.close();
        }
        catch (IOException e){
            showStatus("Exception: " + e.toString());
        }
    }
}
```

ANEXO F. Código en Lenguaje C para codificar los archivos en formato hexadecimal para almacenar en memoria permanente

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

void main(void){

    char                web_header[]="HTTP/1.0 200 OK\r\nContent-type: image/gif\r\n\r\n";
    char                *pointer_web_page=web_header;
    int                 i,temp,page_len;
    unsigned int        pseudo_chk,temp_chk,imp,header_len;
    char                file_path[]="d:\\genaro\\tesis\\gino_rtl\\";
    char                open[100]="";
    char                file_open[]="imagen.jpg";
    char                file_out[]="avr_html.asm";
    FILE                *fil,*fil_out;
    unsigned char        ttt=0;
    unsigned int         tt=0;

    strcat(open,file_path);
    strcat(open,file_open);
    if ((fil = fopen(open, "rb"))== NULL){
        fprintf(stderr, "Cannot open input file.\n");
    }

    open[0]=0;
    strcat(open,file_path);
    strcat(open,file_out);
    if ((fil_out = fopen(open, "w"))== NULL){
        fprintf(stderr, "Cannot open input file.\n");
    }

    page_len=0;
    while(*pointer_web_page!='\x0'){
        page_len++;
        pointer_web_page++;
    }
    header_len=page_len;

    while(tt!=EOF){
        tt=fgetc(fil);
        page_len++;
    }
    page_len--;

    temp_chk=0;
    pseudo_chk=0;
    for(i=0;i<header_len;i+=2){
        temp_chk=pseudo_chk;
        temp=web_header[i];
        pseudo_chk+=temp<<8;
```

```

        temp=web_header[i+1];
        pseudo_chk+=temp;

        if(temp_chk>pseudo_chk) pseudo_chk++;
    }

    if(page_len%2)
        imp=1;
    else
        imp=0;

    fseek(fil,0,SEEK_SET);
    for(i=header_len;i<page_len-imp;i+=2){
        temp_chk=pseudo_chk;
        tt=fgetc(fil);
        pseudo_chk+=tt<<8;
        tt=fgetc(fil);
        pseudo_chk+=tt;
        if(temp_chk>pseudo_chk) pseudo_chk++;
    }
    if(imp){
        tt=fgetc(fil);
        temp_chk=pseudo_chk;
        pseudo_chk+=tt<<8;
        if(temp_chk>pseudo_chk) pseudo_chk++;
    }

    fseek(fil,0,SEEK_SET);
    pseudo_chk+=20;
    printf("\n%x\n",pseudo_chk);
    printf("%x",page_len);
    while(!kbhit());

    for(i=0;i<(page_len-header_len);i++){
        if(!(i%50)){
            if(i!=0){
                fputc("\",fil_out);
                fputc(0x0D,fil_out);
                fputc(0x0A,fil_out);
            }
            fputc('.',fil_out);
            fputc('d',fil_out);
            fputc('b',fil_out);
            fputc("\t",fil_out);
            fputc("\",fil_out);
        }
    }

    tt=fgetc(fil);
    if(tt!="")
        fputc(tt,fil_out);
    else{
        fputc(0x22,fil_out);
        fputc(0x2C,fil_out);
        fputc('0',fil_out);
    }

```

```
        fputc('x',fil_out);
        fputc('2',fil_out);
        fputc('2',fil_out);
        fputc(0x2C,fil_out);
        fputc(0x22,fil_out);
    }

}
fputc("\",fil_out);

fclose(fil);
fclose(fil_out);
}
```

Bibliografia

[AV: 2002]	AVR RISC Microcontroller Handbook Kuhnel, Claus Newnes Butterworth-Heinemann
[RE: 1998]	Realtek http://www.realtek.com.tw/htm/download/complete.asp Communications Network Ics
[HT: 2001]	HTML 3.2 & CGI John December and Mark Ginsburg Sams.net Publishing
[CR: 2000]	Implementation notes and Docs for an Ethernet solution http://www.cornelius-consult.de/ethernet/ Werner Cornelius Cornelius Consult
[MI: 2002]	Microchip PICDEM.net Board http://www.microchip.com/download/tools/picmicro/demo/pdem18r/39579a.pdf Microchip
[AC: 1996]	Ethernet: Distributed Packet Switching for Local Computer Networks http://www.acm.org/classics/apr96/#abstract <i>Robert M. Metcalfe and David R. Boggs</i> <i>Xerox Palo Alto Research Center</i>
[PC: 2001]	PC Interfacing via Ethernet Electronics World http://www.eix.co.uk/Ethernet/ Eddy Insam
[IR: 2000]	iReady Tecnology http://www.iready.org/docs/
[IC: 2000]	El Bus I²C Dominique Paret Editorial Paraninfo
[TC: 2000]	TCP/IP Illustrated, Vol 1 W. Richard Stevens Addison-Wesley
[RO: 2000]	Routing In The Internet Christian Huitema Prentice Hall
[ON: 2001]	On theBoard Micro electronics systemhouse http://www.techweb.co.jp/onboard/obtry1.html

[PW: 2001]	PicoWeb http://www.picoweb.net/downloads.html Lightner Engineering
[PR: 1995]	Protocol Directory http://www.protocols.com/pbook/tcpip.htm RadCom Academy
[TP: 2000]	TCP/IP PROTOCOL STACK http://www.kprinc.com/1999/m1/pr020.htm SEIKO INSTRUMENTS
[IN: 2000]	TCP/IP: Introduction to TCP/IP Concepts http://www.bizcenter.net/dpec/courses/tc1/tc1title.htm DPEC
[IB: 2000]	I²C Bus http://www.geocities.com/SiliconValley/9504/mic_b000.htm Electronic Pages
[PB: 2002]	TCP/IP Principios básicos, protocolos y arquitecturas Douglas E. Comer Prentice Hall

Glosario

ARCHIE.- Permite la búsqueda de información en los servidores FTP Anónimos. Basado en la arquitectura Cliente / servidor, Archie da nombre a ambos. Los servidores Archie contienen una lista de toda la información que contienen los servidores FTP Anónimos a los que agrupa. Existen muchos clientes Archie: archie, xarchie, e incluso una pasarela Archie desde WWW.

CERN.- Laboratorio de Física de Partículas. Fue el desarrollador del World Wide Web, buscando construir un sistema de hipertexto e hipermedia. Actualmente la iniciativa en el desarrollo, especificaciones y software pertenece al consorcio W3. Consorcio de empresas del Sector Informático y Comunicaciones.

CGI (Common Gateway Interface). Es una interfase para que programas externos (pasarelas) puedan ejecutarse bajo un servidor de información. Actualmente, los servidores de información soportados son servidores HTTP (hypertext Transfer Protocol). Las pasarelas pueden usarse para muchos propósitos, algunos de ellos: Manejo de formas y cuestionarios, Conversión de las páginas principales del sistema a páginas html y presentación del resultado por parte del cliente WWW, interfase con bases de datos WAIS y Archie, y presentación de los resultados en formato html por parte de clientes WWW, Mensajería electrónica (comunicación con los administradores WWW),

FAQ (Frequently Asked Questions). Preguntas más frecuentes a problemas.

FIREWALLS. Pretenden asegurar las redes corporativas frente a entradas no autorizadas. El sistema Firewall se coloca entre la red local e Internet. La regla básica de un Firewall es asegurar que todas las comunicaciones entre la red e Internet se realicen conforme a las políticas de seguridad de la organización o corporación. Además, estos sistemas conllevan características de privacidad, autenticación, etc. Las dos técnicas usadas en la construcción de un "Internet Firewalls" son: Aplicaciones, Filtrado de paquetes.

FTP (File Transfer Protocol). Permite transmitir ficheros sobre Internet entre una máquina local y otra remota.

FTP Anonymous. Los servidores FTP anonymous son grandes cajones de ficheros distribuidos y organizados en directorios. Contienen programas (normalmente de dominio público o shareware), ficheros de imágenes, sonido y video. El medio de acceso y recuperación de la información es FTP (File Transfer Protocol). Para entrar en estos servidores, tecleamos FTP y nombre del servidor. El sistema nos pregunta login, a lo que respondemos con la palabra 'anonymous' y en el password le indicaremos nuestra dirección de correo electrónico. Algunos servidores autentifican esta dirección.

GIF (Graphics Interchange Format). Formato Gráfico desarrollado por CompuServe en 1.987 para resolver el problema del intercambio de imágenes a través de diferentes plataformas. Ha llegado a ser (de hecho) el formato estándar de Internet. El original formato GIF87a soportaba 256 colores (8bits) y compresión de imagen con una variante del algoritmo LZW. Este estándar fué revisado en 1.989, resultando un nuevo estándar llamado GIF89a.

GIF animados. GIF89a permite que varias imágenes puedan ser compiladas dentro de un mismo fichero GIF. Estas imágenes pueden ser ligadas a modo de secuencias (o frames). La visualización de este fichero produce una salida animada. Es posible también actuar sobre el tamaño de cada secuencia, tiempo entre ellas, colores de fondo, inclusión de textos y comentarios y otras características avanzadas.

GNU. La Fundación para el Software Libre (FSF - Free Software Foundation) está dedicada a eliminar las restricciones de uso, copia, modificación y distribución del software. Promueve el desarrollo y uso del software libre en todas las áreas de la computación. Específicamente, la Fundación pone a disposición de todo el mundo un completo e integrado sistema de software llamado GNU. La mayor parte de este sistema está ya siendo utilizado y distribuido. Según la FSF, se puede o no se puede pagar para obtener el software de GNU, pero al menos se tienen dos libertades una vez que se tiene el software: la primera, la libertad de copiar el programa y darlo a amigos y colaboradores, y la segunda, la libertad para cambiar el programa y adaptarlo a las necesidades propias (por acceso a todas las fuentes).

- GOPHER.** Es un sistema de entrega de información distribuida. Utilizando gopher podemos acceder a información local o bien a acceder a servidores de información gopher de todo el mundo.
- Gopher** combina las características de BBS (Bulletin Board Service) y bases de datos, permitiendo establecer una jerarquía de documentos, y permitiendo búsquedas en ellos por palabras o frases clave. Concebido y desarrollado en la Universidad de Minnesota en el año 91 es de libre distribución para fines no comerciales. Gopher soporta directorios, ficheros de texto, ítem de búsqueda, sesiones telnet y tn3270, multimedia y texto formateado (postscript y otros).
- HREF.** Permite especificar una dirección de enlace dentro de un documento HTML. Por ejemplo, la línea en html: El enlace a La Universidad de Cordoba, será presentada como El enlace a La Universidad de Cordoba
- HTML** (Hypertext Markup Language). Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986 (SGML, Standard Generalized Markup Language). HTML sigue un modelo de desarrollo abierto. Cuando una nueva característica es propuesta, es implementada en algunos clientes y probada en algunas aplicaciones. Si la demanda para esta nueva característica es suficiente, otras implementaciones son animadas a seguir esta nueva demanda, y la nueva característica llega a ser ampliamente empleada. En este proceso, el diseño es revisado y quizás modificado o potenciado. Finalmente, cuando existe suficiente experiencia con esta nueva característica, llega a ser parte del conjunto estándar de HTML
- HTML nivel 2** Es esencialmente igual que HTML pero con el añadido del manejo de formas.
- HTML + (o HTML nivel 3).** Es un súper conjunto de HTML diseñado para añadir nuevas características como tablas, figuras y mapas sensitivos, formas para interrogación de bases de datos y cuestionarios, fórmulas matemáticas y mail
- HTTP** (Hypertext Transfer Protocol). Es un protocolo con la ligereza y velocidad necesaria para distribuir y manejar sistemas de información hipermedia. Es un protocolo genérico orientado al objeto, que puede ser usado para muchas tareas como servidor de nombres y sistemas distribuidos orientados al objeto, por extensión de los comandos, o métodos usados. Una característica de HTTP es la independencia en la visualización y representación de los datos, permitiendo a los sistemas ser contruidos independientemente del desarrollo de nuevos avances en la representación de los datos. HTTP ha sido usado por los servidores World Wide Web desde su inicio en 1.990.
- HTTPS.** Primera acepción: Servidor WWW para sistemas Windows NT. Segunda acepción: URL creada por Netscape Communications Corporation para designar documentos que llegan desde un servidor WWW seguro. Esta seguridad es dada por el protocolo SSL (Secure Sockets Layer) basado en la tecnología de encriptación y autenticación desarrollada por la RSA Data Security Inc.
- INTERNET.** Es la red de redes. Nacida como experimento del ministerio de defensa americano, conoce su difusión más amplia en el ámbito científico-universitario. Embrión de las 'superautopistas de la información'. Para convertirse en ellas faltan mayores infraestructuras y anchos de banda. Desde el punto de vista técnico, Internet es un gran conjunto de redes de ordenadores interconectadas (la mayor red mundial : mapa color ps, mapa mono ps, tabla ps, tabla txt). Desde otro punto de vista, Internet es un fenómeno sociocultural. Un usuario desde su consola, tiene acceso a la mayor fuente de información que existe. En cuanto a funcionamiento interno, Internet no se ajusta a ningún tipo de ordenador, tipo de red, tecnología de conexión y medios físicos empleados. Internet no tiene una autoridad central, es descentralizada. Cada red mantiene su independencia y se une cooperativamente al resto respetando una serie de normas de interconexión. La familia de protocolos TCP/IP es la encargada de aglutinar esta diversidad de redes. A principios de 1.992 fué creada la Internet Society (ISOC). Se trata de una sociedad profesional sin ánimo de lucro, formada por organizaciones e individuos de todos los sectores involucrados de una u otra forma en la construcción de Internet (usuarios, proveedores, fabricantes de equipos, administradores, etc.). El principal objetivo es fomentar el crecimiento de la Internet en todos sus aspectos (número de usuarios, nuevas aplicaciones, infraestructuras, etc.).

INTERNET DRAFT. Documentos de trabajo de la Internet Engineering Task Force (IETF). Los borradores Internet Draft tiene una validez máxima de 6 meses. Pueden ser modificados, reemplazados o quedar obsoletos por otros documentos.

IRC (Internet Relay Chat). Escrito por Jarkko Oikarinen (jto@tolsum.oulu.fi) en 1.988. Desde su comienzo en Finlandia, ha sido usado en más de 50 países alrededor del mundo. Fue diseñado para reemplazar al programa 'talk', pero ha llegado a ser mucho más que esto. IRC es un sistema de conversación multiusuario, donde la gente se reúne en canales (lugar virtual, normalmente con un tema de conversación) para hablar en grupo o en privado. IRC consiguió fama internacional durante la guerra del Golfo Pérsico, cuando las noticias llegaban a través de telegramas a todo el mundo, la gente que estaba en irc, recogía estas noticias en un simple canal de irc. IRC trabaja en arquitectura Cliente / servidor. El usuario rueda un programa cliente llamado 'irc', el cual conecta vía red con otro programa servidor. La misión del servidor es pasar los mensajes de usuario a usuario a través de la red irc.

JAVA. Java es un lenguaje orientado a objetos y desarrollado por Sun Microsystem. Comparte similitudes con C, C++ y Objective C. Basándose en otros lenguajes orientados al objeto, Java recoge lo mejor de todos ellos y elimina sus puntos más conflictivos. El principal objetivo de JAVA fue hacer un lenguaje que fuera capaz de ser ejecutado de una forma segura a través de Internet (aunque el código fuera escrito de forma maliciosa). Esta característica requiere la eliminación de muchas contracciones y usos de C y C++. El más importante, es que no existen punteros. Java no puede acceder arbitrariamente a direcciones de memoria. Java es un lenguaje compilado en un código llamado "código-byte" (byte-code). Este código es interpretado "en vuelo" por el interprete Java. Java fue diseñado también para escribir código libre de bugs, esto se consigue en gran parte, eliminando las operaciones de localización de memoria del lenguaje C. Java no es un lenguaje para ser usado solo en el WWW, pero su despegue y utilización se debe al World Wide Web. Hoy día casi todos los browser interpretan código Java

JPEG (Join Photographic Expert Group). Formato gráfico comprimido desarrollado por la 'Join Photographic Expert Group'. El formato JPEG soporta 24 bits por píxel y 8 bits por píxel en imágenes con escala de grises. Realiza un buen trabajo con imágenes realísticas (imágenes escaneadas)

LINK. Enlace, hiperenlace. Ver HREF, TELNET, FTP, GOPHER, HTTP.

LINUX. Linux es una implementación independiente con "espíritu" POSIX (especificación para sistemas operativos). Tiene extensiones System V y BSD, y ha sido escrito completamente basándose en aportaciones. Linux no tiene código propietario. Linux está distribuido libremente bajo "GNU Public License". Actualmente solo trabaja en IBM PC (o compatibles) y con arquitecturas ISA e EISA, y requiere un procesador 386 o superior. El kernel de Linux está escrito por Linux Torvalds (Torvalds@kruuna.helsinki.fi), desde Finlandia y otros voluntarios de otras partes del mundo. La mayoría de los programas que se ejecutan bajo linux son freeware, y muchos de ellos del Proyecto GNU. Linux tiene todas las características que se pueden esperar de un moderno y flexible UNIX. Incluye multitarea real, memoria virtual, librerías compartidas, dirección y manejo propio de memoria y TCP/IP. Usa las características hardware de la familia de procesadores 386 para implementar las capacidades anteriores. En cuanto a software que rueda sobre linux, podemos citar GCC, Emacs, X-Windows, todas las utilidades del Unix estándar, TCP/IP (incluyendo SLIP y PPP) y cientos de programas que cualquiera pueda compilar y portar a esta plataforma. En cuanto a hardware, admite bus local VESA y PCI. No rueda en MCA (MicroChannel, bus propietario de IBM). Existe un proyecto para portar Linux a las máquinas basadas en el 68000 de Motorola (como por ejemplo, Comodore Amiga y Atari) y otro proyecto para portar Linux a la arquitectura PowerPC.

LYNX. Lynx es un cliente para servidores World Wide Web para usuarios UNIX y VMS que se conectan al sistema a través de terminales ascii o emuladores. Soporta terminales VT100 y emuladores de terminal VT100 (como kermi, procomm, etc.). Lynx fue desarrollado por la Universidad de Kansas y es de dominio público para usos no comerciales.

MACINTOSH. Serie de ordenadores de Apple Computer. Posee un sistema operativo basado en ventanas. El entorno es intuitivo, eliminando el teclado de los comandos del sistema. Prácticamente todo puede hacerse a través de menús desplegables y de ratón. A todos los objetos se le asigna una representación gráfica (iconos).

MAIL. El correo electrónico es el servicio más básico, antiguo, y más utilizado dentro de Internet.

La mensajería electrónica es el medio más eficaz y más rápido de comunicación, permite intercambiar además de mensajes, programas, audio, vídeo e imágenes.

MAILING LISTS. Listas de correo o listas de distribución, establecen foros de discusión privados a través de correo electrónico. Las listas de correo están formadas por direcciones e-mail de los usuarios que la componen. Cuando uno de los participantes envía un mensaje a la lista, ésta reenvía una copia del mismo al resto de usuarios de la lista (inscritos en ella).

MBONE. Es un desarrollo de los dos primeros experimentos de transmisión de audio de la IETF (Internet Engineering Task Force) en el que audio y video son transmitidos en tiempo real desde el lugar de reunión de la IETF a destinos a lo largo del mundo. La idea es construir un banco de pruebas semipermanente de multi transmisión IP para soportar las transmisiones de la IETF y mantener una experimentación continua entre reuniones. Este es un esfuerzo de cooperación voluntario. MBONE es una red virtual. Está compuesta por porciones de red física Internet para soportar el enrutamiento de paquetes IP multicast hasta que estas funciones estén integradas en los routers de nueva producción. La red esta compuesta de islas que pueden soportar directamente IP multicast, como 'Ethernet LAN Multicast', enlazados por links virtuales punto-a-punto llamados túneles. Los puntos finales de los túneles son normalmente estaciones de trabajo (máquinas que teniendo sistema operativo soportan IP multicast y ruedan un 'daemon' llamado 'mrouted' para enrutar multicast). El tráfico durante una sesión multicast está entre 100-300 Kbits/sg. La velocidad de 500 Kb/sg. Ha sido vista como un ancho de banda razonable. Normalmente, la señal de audio es transportada entre 32 y 64 Kb/sg. La señal de video necesita al menos 128 Kb/sg.

MICROSOFT WINDOWS. Sistema operativo gráfico de Microsoft basado en ventanas. Es el más popular en entornos PC. Permite el acceso a Internet mediante TCP/IP y Winsockets.

MIRROR. Término usado en Internet para hacer referencia a un FTP, WEB o cualquier otro recurso que es espejo de otro. Estos mirrors se realizan automáticamente y en una frecuencia determinada, y pretenden tener una copia exacta del lugar del que hacen mirror.

MOSAIC. Cliente WWW desarrollado en NCSA para las siguientes plataformas:

- Mosaic para X: Usa X11/Motif. Fue el primer cliente para WEB. Soporta http 1.0. Disponible mediante FTP anonymous en ftp.ncsa.uiuc.edu en el directorio Mosaic.
- Mosaic para MS-Windows: Precisa las librerías de 32 bit (win32). Disponible vía FTP anonymous en ftp.ncsa.uiuc.edu en el directorio PC/Windows/Mosaic
- Mosaic para Macintosh: Creado para este entorno, está disponible vía FTP anonymous en ftp.ncsa.uiuc.edu
- Mosaic para VMS: Usa X11/DEC Windows/Motif. Escrito para el sistema operativo VMS de Digital Equipment Corporation. Está disponible mediante FTP anonymous en ftp.ncsa.uiuc.edu en el directorio Mosaic

NCSA (National Center for Supercomputing Applications). NCSA, de la Universidad de Illinois, desarrolladores de Mosaic, el primer cliente para servidores WWW.

NETSCAPE. Cliente WWW desarrollado por Netscape Communications Corp. Descarga y visualiza las imágenes en forma incremental, permitiendo, mientras, leer el texto (también descargado de forma incremental). Es probablemente el mejor cliente WWW. Soporta acceso directo a news, sin pasarelas, y muchas de las extensiones de HTML.

NEWS. Es el tablón de anuncios electrónico. Permite al usuario participar en grupos de discusión, mediante el envío de mensajes, o bien sólo acceder a estos grupos para obtener información. Los mensajes están clasificados por temas y se integran por grupos (newsgroups). News es un conjunto de Newsgroups distribuidos electrónicamente en todo el mundo. Los grupos pueden estar moderados o no, en el primer caso, el moderador decide que mensajes aparecerán.

PERL. Perl es un lenguaje para manipular textos, ficheros y procesos. Perl proporciona una forma fácil y legible para realizar trabajos que normalmente se realizarían en C o en alguna Shells. Podría decirse que Perl está a caballo entre un lenguaje de alto nivel (tipo C) y una 'Commands shell'. Perl corre en varios sistemas operativos y permite portar las fuentes a diferentes plataformas. No obstante, donde nació y donde más se ha difundido es bajo el sistema operativo UNIX.

PPP (Point-To-Point Protocol). Ver SLIP

SGML (Standard Generalized Markup Language). Es un lenguaje más extenso que HTML. HTML es una aplicación de SGML. Para aprender mas sobre SGML, existe un documento llamado 'A Gentle Introduction to SGML' proporcionado por 'Text Encoding Initiative'.

SLIP (Serial Line Internet Protocol). Junto con PPP (Point-to-Point Protocol) son estándares para transmisión de paquetes IP (Internet Protocol) sobre líneas serie (líneas telefónicas). La información de Internet es empaquetada y transmitida en paquetes IP. Un proveedor de servicio de acceso a Internet puede ofrecer SLIP, PPP o ambos. El ordenador debe usar un software de conexión (normalmente suministrado por el proveedor) que marca el protocolo de conexión con el servidor. PPP es un protocolo más reciente y robusto que SLIP.

- **SLIP dinámico:** Cuando se usa SLIP para conectarse a Internet, el servidor del proveedor de acceso a Internet, identifica al ordenador proporcionándole una dirección IP (por ejemplo 150.214.110.8). Mediante SLIP dinámico, ésta dirección es asignada dinámicamente por el servidor de entre un conjunto de direcciones. Esta dirección es temporal, y dura lo que dure la conexión.
- **SLIP estático:** Cuando se usa SLIP estático, el servidor del proveedor de acceso a Internet asigna una dirección permanente al ordenador para su uso en todas las sesiones.

TCP/IP (Transmission Control Protocol/Internet Protocol). Familia de protocolos que hacen posible la interconexión y tráfico de red en Internet. A ella pertenecen por ejemplo: FTP, SMTP, NNTP, etc. Los dos protocolos más importantes son los que dan nombre a la familia IP y TCP

UNIX. Es un sistema operativo multiusuario y multitarea. Como características más importantes:

- Redireccionamiento de Entradas/Salidas.
- Sistema jerárquico de ficheros. Estructura de árbol invertido (File System).
- Interfase simple e interactiva con el usuario.
- Alta portabilidad al estar escrito en C. Es casi independiente del hardware
- Creación de utilidades fácilmente.

Los componentes básicos del Unix.:

- **Kernel.** Parte del S.O. residente permanentemente en memoria. Dirige los recursos del sistema, memoria, E/S y procesos. Podemos distinguir dos partes: sección de manejo de procesos y sección de manejo de dispositivos.
- **Shell.** Intérprete de comandos. Interpreta y activa los comandos o utilidades introducidos por el usuario.
- Es un programa ordinario (ejecutable) cuya particularidad es que sirve de interfase entre el Kernel y el usuario. Es también un lenguaje de programación, y como tal permite el usar variables, estructuras sintácticas, entradas/salidas etc.
- **Programas.** La shell es un caso especial de programa. Son programas que son partes estándar de Unix (comandos de sistema, daemon y utilidades), programas de usuario (compilados) y shell scripts (comandos y sentencias interpretadas por una shell).

URL (Uniform Resource Locator). Utilizado para especificar un objeto en Internet. Puede ser un fichero, grupo de news, gopher, etc.

VMS. Sistema Operativo propietario de Digital Equipment Corporation (DEC) para sus máquinas VAX

WAIS. Es un sistema de recuperación de información distribuido. Permite al usuario la búsqueda en bases de datos en la red (bases de datos WAIS) usando un interfase fácil de usar. Las bases de datos son en su mayoría colecciones de documentos, aunque pueden contener sonido, imágenes o video. WAIS es capaz de buscar por el contenido de un documento. WAIS usa un modelo Cliente/Servidor.

WEB. Ver WWW

WWW ROBOTS. Son programas que automáticamente atraviesan el universo WWW recogiendo enlaces. La mayoría de los robots siguen un protocolo muy simple, del cual es fácil proteger a los servidores de su acceso (completamente o parcialmente).

WWW (World Wide Web). Servidor de información, desarrollado en el CERN (Laboratorio Europeo de Física de Partículas), buscando construir un sistema distribuido hipermedia e hipertexto. También llamado WEB y W3

X.500. El directorio X.500 es una base de datos distribuida que permite la consulta de datos sobre objetos del mundo real. A través de X.500 se puede buscar información sobre personas, departamentos y organizaciones de todo el mundo. Puede proporcionar direcciones de mensajería electrónica, direcciones postales, teléfonos y números de Fax.