ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DISEÑO Y CONSTRUCCIÓN DE UN PROTOTIPO DE UN SERVIDOR WEB BASADO EN UN MICROCONTROLADOR 8x51

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

RIVADENEIRA HINOJOSA NURIA KARINA

DIRECTOR: ING. FERNANDO FLORES

Quito, Julio 2004

DECLARACIÓN

Yo, Nuria Karina Rivadeneira Hinojosa, declaro que el trabajo aquí descrito se de autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que ha consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Karina Rivadeneira H.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Nuria Karina Rivadeneira Hinojosa, bajo mi supervisión.

ING. FERNANDO FLORES
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

En primer lugar quiero agradecer a Dios, por haberme guiado y ayudado a lo largo de estos años.

A mis padres y hermanos que han estado a mi lado apoyándome y animándome en todo momento, brindándome su amor, confianza constantemente.

A mis tíos, con quienes he vivido durante esta etapa de mi vida, por su apoyo, ayuda, cariño y solidaridad conmigo durante los duros momentos de esta fase como estudiante.

A mis amigos, por su amistad sincera, comprensión, solidaridad y cariño, con quienes aprendí a mirar el mundo desde otra perspectiva.

Al Ing. Fernando Flores por la ayuda que me ha brindado durante el desarrollo de este proyecto, por su tiempo y porque más que un profesor es el mejor de los compañeros.

Karina

DEDICATORIA

A mis padres y hermanos quienes son el motor que me impulsa a seguir adelante, a mis tíos que con su ayuda han hecho posible la culminación de esta etapa.

Karina

CONTENIDO

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad las redes se han convertido en elementos fundamentales para la comunicación. Las redes forman la columna vertebral de la comunicación en muchas empresas y negocios, se las usa para compartir información entre agrupaciones educativas, comerciales, gubernamentales, científicas, etc. Esta información puede ser de varias clases tales como: notas, documentos, datos para ser procesados por otra computadora, archivos enviados a colegas y algunas otras formas exóticas de datos.

La mayoría de estas redes se instalaron entre los años 60s y 70s, cuando el diseño de una red de computadoras era una "manifestación de arte" y la comunicación entre computadoras era una comunicación sofisticada, por lo que para conseguirlo era necesario realizar algunos estudios e investigaciones. Como resultado de esto se tienen múltiples modelos de redes.

Pero en los últimos años, los sistemas de comunicación a través de computadoras así como el Internet han experimentado un rápido crecimiento y juegan un papel importante en nuestra vida cotidiana, tanto en el trabajo como en la educación y en el hogar. Cada día que pasa se logra tener nodos, servidores y otros elementos que conforman las redes de tamaño más pequeño y con muchas mejoras para obtener una mayor cantidad de servicios.

Además, la tecnología de la miniaturización ha cobrado fuerza en el desarrollo de pequeños sistemas o dispositivos electrónicos que son capaces de realizar complejas tareas. Usando este conocimiento, muchas aplicaciones son posibles. Imagine que se puede controlar estos pequeños dispositivos utilizando un navegador de Internet normal, visualizar y transferir el estado de sensores o en forma automática generar y enviar un correo electrónico cuando ocurren eventos especiales, por ejemplo para propósitos de seguridad.

PRESENTACIÓN

El presente proyecto de titulación contiene el estudio de una novedosa tecnología para el diseño y la implementación de servidores Web. La construcción del servidor se ha realizado en base de algunos elementos que han sido creados y desarrollados específicamente para este tipo de aplicaciones.

Este trabajo se divide en cuatro capítulos y en varios anexos en los cuales se explica la teoría y las características de los elementos empleados para el desarrollo del presente proyecto.

En el capitulo I se realiza el estudio de las redes TCP/IP y de algunos protocolos que pueden ser usados por la capa Aplicación en las diferentes redes de datos, por lo tanto este capítulo constituye la base teórica sobre la cual se va ha desarrollar el proyecto.

En el capítulo II, se describe el diseño del hardware y del software, para lo cual se hace un estudio de los diferentes elementos utilizados para la implementación del prototipo, también contiene una rápida descripción de la herramienta empleada para el desarrollo del software y la explicación de las rutinas principales del programa implementado.

En el capítulo III se presentan las pruebas necesarias para comprobar el correcto funcionamiento y operación del servidor Web.

En el capítulo IV se muestran las conclusiones y recomendaciones que se han obtenido a lo largo del desarrollo de este proyecto.

Finalmente se incluyen las hojas de datos proporcionadas por los fabricantes de los elementos utilizados, los esquemáticos del servidor Web y el código fuente del software desarrollado.

CAPÍTULO 1

FUNDAMENTOS TEÓRICOS

1.1 INTRODUCCIÓN

El Internet es un conjunto de redes de computadoras interconectadas entre sí, es una red de redes. La palabra Internet o también internetwork se lo puede entender de manera simple como *interconexión de redes*.

Desde sus inicios hasta la actualidad, el Internet ha estado en constante expansión y evolución, cada vez se puede implementar mayor cantidad de servicios y aplicaciones que son utilizados por miles de personas alrededor del mundo, ya que el Internet es una red pública que conecta redes de organismos oficiales, educativos y empresariales. También existen sistemas de redes más pequeñas denominados Intranet, generalmente para el uso de una única organización.

"La tecnología de Internet es una precursora de la llamada superautopista de la información"[1], "la aplicación más difundida en el Internet es el WWW *(World Wide Web)*, que le permite a un usuario acceder a millones de páginas de la más variada y actualizada información en forma inmediata e inclusive realizar varios tipos de negocios a través de la red en cualquier parte del mundo"[2], también existen otras aplicaciones que en la actualidad tienen gran demanda, como por ejemplo el correo electrónico que constituye el servicio postal de mayor rapidez, así como el *chat* que le permite a una persona mantener una conversación con otro usuario de la red, eliminando de esta manera la barrera de la distancia.

Para el usuario la existencia de estas múltiples redes interconectadas entre sí es totalmente transparente, ya que el software de red está basado en un conjunto de protocolos conocido como TCP/IP, el cual se encarga de ocultar la existencia de las múltiples subredes que conforman esta gran red virtual.

---

[1] Enciclopedia Microsoft Encarta 2002

[2] Barreto, Alexis, "ESTUDIO Y ANÁLISIS DE LAS DISTINTAS TECNOLOGÍAS DE ACCESO QUE UN PROVEEDOR DE SERVICIOS DE INTERNET PUEDE IMPLEMENTAR EN ECUADOR", Tesis previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional, Quito – Ecuador, Diciembre de 1999.

Para entender mejor como es que un usuario puede navegar en la red de redes y acceder a toda la información disponible sin importar en que red se encuentra dicha información, más adelante se explicará en que consiste ésta arquitectura de red y cuales son los principales protocolos que le permiten a un usuario acceder y hacer uso de algunas de las aplicaciones que se mencionaron en párrafos anteriores.

1.2 MODELO DE REFERENCIA TCP/IP

Es muy común describir al modelo de referencia TCP/IP como un modelo estratificado en capas, una sobre otra, en donde cada capa proporciona servicios a la capa superior y obtiene servicios de la capa inferior para realizar sus funciones y proporcionar sus servicios.

Este modelo de estratificación de capas se basa en la comunicación virtual entre capas correspondientes en máquinas diferentes, mediante un conjunto de reglas y procedimientos denominados protocolos. A los procesos que se ejecutan en capas correspondientes en máquinas diferentes se los denominan procesos pares.

Una de las ventajas del modelo estratificado de capas es el hecho de que permite sustituir la forma de realizar un servicio en una capa sin necesidad de alterar el resto de los servicios en otras capas, esto es posible gracias a que los detalles operacionales de las capas más bajas están ocultos de las capas más altas, esto simplifica también el plan de mantenimiento a nivel de software.

Internet se fundamenta en la arquitectura TCP/IP, el cual es un modelo estratificado basado en cuatro capas y que en forma general ofrece tres tipos de servicios relacionados entre sí:

- Un servicio no confiable de entrega de paquetes sin conexión, lo que le permite al conjunto TCP/IP adaptarse a un número importante de

tecnologías de red. Sobre este ofrece el servicio de transporte extremo a extremo, que es la base para los servicios de aplicación.

- El segundo es un servicio confiable de entrega de paquetes sin conexión, no usa una conexión lógica, útil sobre canales no confiables tales como los sistemas inalámbricos.

- El tercero es un servicio confiable de entrega de paquetes con conexión, ofrece seguridad en la entrega de paquetes, se asegura de que el mensaje verdaderamente llegue a su destino. Tiene tres etapas que son: establecimiento de conexión, intercambio de datos y liberación de la conexión.

El modelo de referencia normalmente utilizado para describir la arquitectura de Internet, es un subconjunto del modelo de referencia ISO – OSI estratificado en siete capas.

La arquitectura TCP/IP esta formada por cuatro capas: Aplicación, Transporte, Internet e Interfaz de red. A continuación se muestra la estratificación de las capas de Internet y su relación con el modelo ISO – OSI.

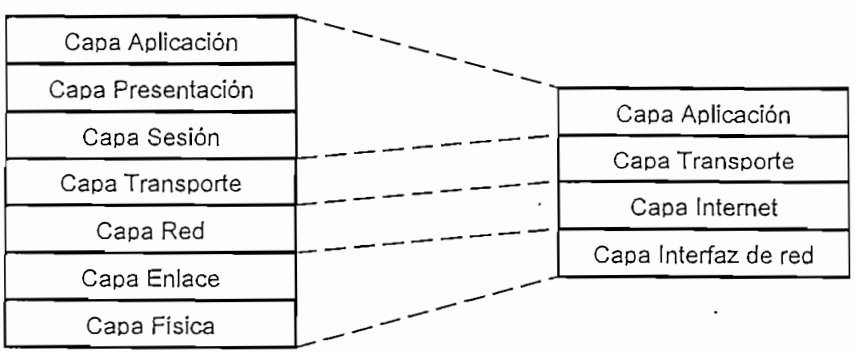


Figura 1.1 ISO – OSI vs Modelo de Referencia para Internet

1.3 CAPAS DE LA ARQUITECTURA TCP/IP

Capa Aplicación: Contiene muchos protocolos definidos por diferentes aplicaciones para proporcionar sus servicios. Por ejemplo: Telnet (Acceso Remoto), FTP (Transferencia de Archivos), SMTP, POP (Correo Electrónico), HTTP (Transferencia de archivos de hipertexto) que constituye la base del WWW (*World Wide Web*),

Capa Transporte: Se encarga de realizar una comunicación extremo a extremo entre capas equivalentes de máquinas diferentes, en esta capa se definen dos tipos de protocolos: TCP (*Transmission Control Protocol*) que es un protocolo orientado a conexión, se asegura que los datos lleguen hacia su destino sin errores y UDP (*User Datagram Protocol*) que es un protocolo no orientado a conexión, envía los datos en forma de paquetes llamados *datagramas* y no garantiza la entrega del paquete, peor aún la entrega de datos sin error.

Capa Internet: También conocida como capa red, se encarga del movimiento de paquetes provenientes de la capa transporte de una máquina a otra a través de la red; es decir, se encarga de entregar y enrutar los paquetes de datos entre los nodos de la red. En esta capa se definen los protocolos: IP (*Internet Protocol*), ICMP (*Internet Control Message Protocol*), IGMP (*Internet Group Management Protocol*).

Capa Interfaz de Red: Es la capa que está en verdadera comunicación con el hardware de la red. En esta capa no se especifica ningún protocolo, ni define algún medio físico en el cual se debe implementar esta arquitectura.

Esta característica representa una gran ventaja ya que se tiene la libertad de elegir el medio de transmisión (cobre, fibra óptica, ondas de radio, satélites, etc.), permitiendo la interconexión de un gran variedad de equipos, de igual manera ocurre con la selección de la tecnología de red (Ethernet, Token Ring, ATM, Frame Relay, FDDI, ISDN, etc.),

Esta capa es la encargada de transmitir los datos o datagramas IP y entregarlos a la capa Internet encapsulados dentro de una trama. La trama es la unidad de intercambio de información a nivel de la capa interfaz de red. El formato de la trama depende del tipo de red física implementada. A este nivel se utilizan las direcciones físicas de las tarjetas de red para la transmisión de datos.

En esta capa se utilizan dos protocolos de bajo nivel: ARP (*Address Resolution Protocol*) y RARP (*Reverse Address Resolution Protocol*) que son los encargados de realizar la conversión de las direcciones, de dirección IP a dirección física y viceversa en el intercambio de información entre la capa interfaz de red y la capa Internet. También utiliza otros protocolos como son SLIP (*Serial Line IP*) y PPP (*Point to Point Protocol*) entre otros.

En la transmisión de datos utilizando la arquitectura TCP/IP, cada capa añade información de control, cabecera o *trailers* a la información enviada por la capa superior, a este proceso se lo llama encapsulación de datos.

En el destino, ocurre el proceso inverso, cada capa retira la información añadida por la capa correspondiente antes de procesar la información y enviarla a la capa superior hasta llegar al usuario.

1.4 PROTOCOLOS DE CAPA INTERFAZ DE RED

Aunque cada máquina de Internet tiene una dirección IP, esta no puede ser utilizada para la transmisión de datos a nivel físico, ya que el hardware del que se dispone para realizar la conexión física del *host* origen al *host* destino utilizando una red, no entiende las direcciones IP utilizadas en Internet. Los *hosts* en una red están conectados a una tarjeta de interfaz que solo entiende direcciones físicas conocidas como direcciones MAC (*Medium Access Control*), estas son las direcciones utilizadas a nivel físico para la interconexión de equipos por medio de una red.

1.4.1 ARP PROTOCOLO DE RESOLUCIÓN DE DIRECCIONES

El protocolo ARP (*Address Resolution Protocol*) "permite a un anfitrión conocer la dirección física de otro anfitrión dentro de la misma red con tan solo proporcionar la dirección IP de su objetivo",[5] para esto ARP utiliza una tabla, por medio de la cual realiza la transformación. Esta tabla contiene la dirección física correspondiente para cada dirección IP, esta tabla se almacena en una memoria intermedia en cada *host*.

Si el *host origen* no tiene este dato en su tabla ARP, debe enviar una solicitud ARP en una trama que contiene la dirección IP del *host destino* a toda la red como un paquete de difusión (*broadcast*) y esperar la respuesta. Si el *host destino* reconoce su dirección IP que esta especificada en el pedido, enviará la respuesta ARP indicando cual es su dirección física únicamente al *host* que hizo la solicitud (*unicast*).

Una vez que el *host* ha recibido la respuesta, almacena tanto la dirección física como la dirección IP en su tabla ARP, de esta manera todos los datagramas siguientes que se envíen a dirección IP de destino, podrán ser traducidos a su respectiva dirección física utilizando la tabla almacenada por el *host* origen.

Cuando se realiza el envío de la solicitud ARP en difusión, todos los *host* actualizan en sus tablas la dirección IP y la dirección física del *host* que realiza dicho pedido, ya que esta solicitud además de llevar la dirección IP del *host* que esta siendo buscado, también lleva la dirección de red y la dirección física de quien originó el mensaje.

De igual manera cuando un *host* ha cambiado su dirección física o su dirección de red, envía un paquete de difusión informando a todos los equipos de la red del cambio efectuado junto con sus nuevas direcciones, para que estos actualicen sus tablas de direccionamiento ARP.

---

[5] Comer, Douglas, "REDES GLOBALES DE INFORMACION CON INTERNET Y TCP/IP, PRINCIPIOS BASICOS Y ARQUITECTURA", Editorial Prentice – Hall, Tercera Edición, México, 1996.

Este protocolo no tiene un encabezado con un formato fijo por lo que es útil para ser utilizado por varias tecnologías de red, pero únicamente por aquellas que soportan difusión.

1.4.2 RARP PROTOCOLO DE RESOLUCIÓN REVERSA DE DIRECCIONES

RARP (*Reverse Address Resolution Protocol*) es un protocolo muy similar a ARP, con la diferencia que en esta ocasión se conoce la dirección física del *host* destino y lo que se desea averiguar es la dirección de red o dirección IP.

Difiere del protocolo ARP porque en esta ocasión se trata de máquinas sin disco de almacenamiento para este tipo de información, por lo que utiliza un servidor RARP para obtener su dirección IP; es decir, es una adaptación del protocolo ARP.

El servidor RARP debe tener una base de datos que asocie las direcciones físicas con las direcciones de red IP, el principio de funcionamiento de este protocolo es que una estación sin disco pero siempre conectada a un servidor RARP lea la dirección física de su propia tarjeta de interfaz de red y envíe la solicitud RARP dentro de una trama que se difunde por toda la red, haciendo un *broadcast* utilizando la dirección física de difusión. El servidor RARP envía una respuesta a este pedido con la dirección IP que corresponde a este sistema sin disco, dicha respuesta viaja en forma de trama utilizando como dirección de destino la dirección física del *host* que envió la solicitud, esta respuesta llega únicamente al solicitante; es decir, viaja en forma *unicast.*

1.4.3 SLIP LÍNEA SERIAL IP

SLIP (*Serial Line IP*) es el más viejo de los protocolos, fue creado en 1984 para conectar estaciones de trabajo SUN a Internet por medio de una línea de discado usando un MODEM.

La familia de protocolos TCP/IP corre sobre una variedad de redes, tales como: LANS IEEE 802.3, 802.4, 802.5, X25, enlaces satelitales, enlaces seriales, etc. Se han definido muchos estándares para la encapsulación de paquetes IP para muchas de estas redes, pero no existe ninguna norma para las líneas seriales. SLIP es actualmente una norma de facto utilizada en las conexiones punto a punto que corren sobre TCP/IP, aunque no es una norma de Internet está documentado por el RFC 1055 y es muy sencillo.

La estación envía paquetes IP a través de la línea serial, con un *byte* especial al final para indicar el fin de la trama, utiliza la técnica de "bandera y relleno de caracteres" en el caso de que el *byte* especial se repita dentro del paquete IP, algunas implementaciones de SLIP agregan un *byte* indicador tanto al principio como al final de cada paquete IP enviado.

Las versiones más recientes de SLIP efectúan cierta compresión de encabezado TCP e IP, que consiste en aprovechar el envío de paquetes consecutivos ya que estos muchas veces tienen campos en común que se repiten en cada uno, se omite estos campos que son iguales en todos los paquetes y los campos que sí difieren se envían como incremento del valor previo.

Aunque aun se utiliza ampliamente, SLIP tiene algunos problemas serios entre ellos se pueden enumerar los siguientes:

- No efectúa detección o corrección de errores.

- Cada lado debe reconocer por adelantado la dirección IP del otro, ninguna de las dos direcciones puede asignarse dinámicamente durante el establecimiento del enlace.

- SLIP no proporciona ninguna forma de verificación de autenticidad.

- No es un estándar aprobado por Internet, por lo que existen muchas versiones diferentes e incompatibles.

1.4.4 PPP PROTOCOLO PUNTO A PUNTO

PPP (*Point to Point Protocol*) fue diseñado para solucionar todos los problemas que tiene el protocolo SLIP con la finalidad de que se convirtiera en un estándar oficial de Internet, es un protocolo de enlace de datos para líneas punto a punto.

El protocolo PPP realiza detección de errores, reconoce múltiples protocolos y permite la negociación de direcciones IP en el momento de la conexión, permite además la verificación de autenticidad, por lo que presenta muchas mejoras con respecto a SLIP.

Este protocolo proporciona tres características:

- Un método para encapsular datagramas sobre líneas seriales, delimitando el inicio y fin de cada trama. El formato de trama también maneja detección de errores.

- Un protocolo de control de enlace LCP (*Link Control Protocol*) para establecer, configurar, probar, mantener y terminar la conexión del enlace de datos.

- Una familia de protocolos de control de red NCP (*Network Control Protocol*) para establecer y configurar diferentes protocolos y opciones de la capa Red.

Puede implementarse en cualquier tipo de interfaz serial DTE/DCE, su único requerimiento es el de implementarse sobre un circuito full duplex, dedicado o conmutado, para el acceso a Internet utiliza un MODEM sobre una línea telefónica *dial up* o dedicada y una conexión entre ruteadores, son ejemplos típicos de enlaces seriales en los que se puede emplear PPP.

1.4.4.1 LCP Link Control Protocol:

PPP utiliza este protocolo para negociar sus posibilidades durante el establecimiento de la conexión.

Los mensajes LCP se transportan en tramas PPP y contienen opciones de configuración para la conexión. Entre otras cosas LCP es el responsable de establecer el enlace, negociar opciones de encapsulación, negociar le tamaño de los paquetes que se van a transmitir, configurar el protocolo de autenticación que se utilizará en la fase de autenticación, determinar cuando el enlace está funcionando adecuadamente y cuando ha fallado, detectar errores de configuración de enlace y terminar el enlace.

LCP define tres tipos diferentes de paquetes:

- Paquetes utilizados para establecer y configurar un enlace.
- Paquetes utilizados para terminar un enlace.
- Paquetes para mantener y administrar un enlace.

1.4.4.2 NCP Network Control Protocol:

PPP define una familia de protocolos de Control de Red encargados cada uno de ellos de establecer y configurar un protocolo distinto de capa Red, a fin de que estos puedan enviar y recibir datagramas sobre un enlace serial.

El NCP para el caso en que IP (*Internet Protocol*) sea utilizado como protocolo de red, se denomina IPCP (*Internet Protocol Control Protocol*) y se encarga de habilitar, configurar y deshabilitar el protocolo IP en los dos extremos de un enlace punto a punto.

1.5 PROTOCOLOS DE CAPA INTERNET

La capa Internet es la que está relacionada con el encaminamiento de los datos del computador origen al computador destino a través de una o más redes conectadas por dispositivos de encaminamiento, se necesita utilizar un protocolo estandarizado para realizar la interconexión de los diferentes tipos de redes, entre los protocolos que se utilizan para alcanzar dicho objetivo se tiene:

1.5.1 IP PROTOCOLO INTERNET.

El protocolo Internet IP es parte del conjunto de protocolos TCP/IP y es el protocolo de interconexión entre redes más utilizado, define el mecanismo de entrega de paquetes sin conexión y con el mejor esfuerzo, el servicio se conoce como no confiable porque la entrega no está garantizada y los paquetes se pueden perder, duplicar, retrasar o entregar sin orden. El protocolo IP es una parte fundamental en el diseño de redes TCP/IP.

El término sin conexión se refiere a que cada paquete es tratado de forma independiente a todos los demás. Una secuencia de paquetes viajan por rutas diferentes, algunos de ellos llegan a su destino mientras que otros pueden perderse.

El protocolo IP proporciona tres definiciones importantes:

- Define la unidad básica para la transferencia de datos utilizada a través de una red de redes TCP/IP.

- El software IP realiza la función de ruteo, seleccionando la ruta por la que los datos serán enviados; es decir, IP es un protocolo de ruteo.

- Aporta especificaciones formales para el formato de los datos y el ruteo, ya que IP incluye un conjunto de reglas que le dan forma a la idea de entrega de paquetes no confiable.

Los servicios que se proporcionan a las capas adyacentes utilizando el protocolo IP se expresan en función de primitivas y parámetros. Las primitivas se utilizan para especificar las funciones que se van a ofrecer y los parámetros se utilizan para pasar datos e información de control.

Esta red de redes utiliza el término *datagrama IP* para identificar a la unidad de transferencia de datos básica; es muy similar a una trama utilizada en la red

física. El datagrama físicamente se divide en dos partes: el encabezado del datagrama y el área de datos.

La diferencia entre el encabezado de una trama y el encabezado del datagrama es el tipo de direcciones, ya que el encabezado del datagrama contiene tanto las direcciones IP de la fuente como del destino, además de un campo que identifica el contenido del datagrama, mientras que el encabezado de la trama contiene únicamente direcciones físicas.

Cabe destacar que la cabecera IP tiene una parte fija de 20 bytes de longitud y una parte opcional de longitud variable. En teoría los datagramas IP pueden tener una longitud de 64 Kbytes pero en la práctica son de 1500 bytes

1.5.1.1 Formato del datagrama IP.

Un datagrama IP está formado por dos partes: cabecera y campo de datos.
En el campo de datos se encapsulan los paquetes TCP, UDP, ICMP, IGMP.
Los campos que conforman el datagrama IP son los siguientes:

- **Versión (4 bits):** Indica el número de la versión del protocolo utilizado. Actualmente se usa la versión 4 (IPv4).

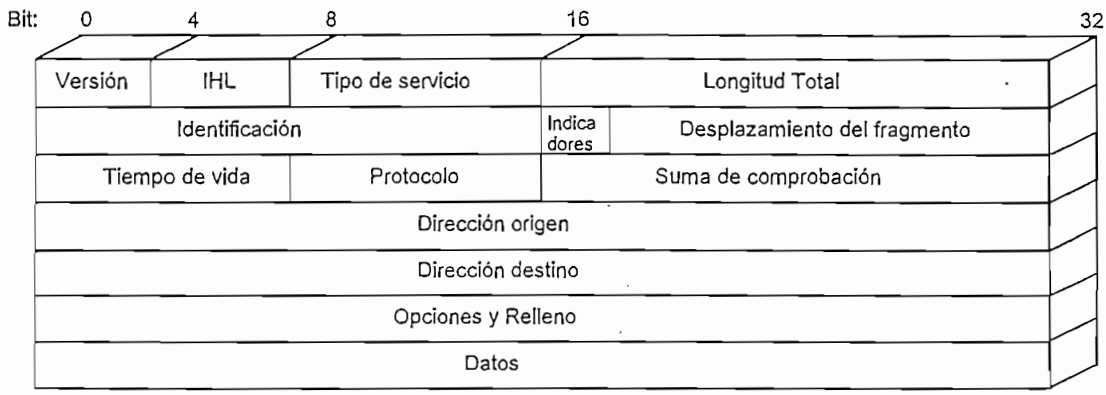


Figura 1.2 Formato de un datagrama IPv4

- **Longitud de cabecera Internet (*IHL, Internet Header Length*) (4 bits):** Define la longitud de la cabecera expresada en unidades de 32 bits. El

valor mínimo es de cinco unidades, que corresponden a la longitud de cabecera mínima de 20 octetos.

- **Tipo de servicio (8 bits)**: Especifica los parámetros de seguridad, prioridad, retardo y rendimiento.

- **Longitud total (16 bits)**: Define la longitud total del datagrama (cabecera y datos) medida en octetos.

- **Identificador (16 bits)**: Contiene un número entero que permite identificar un datagrama, si este es fragmentado, cada uno de estos fragmentos tendrá la misma identificación. Por tanto el identificador debe ser único para la dirección origen del datagrama, la dirección destino y el protocolo usuario durante el tiempo que el datagrama permanezca en la red.

- **Indicadores (3 bits)**: El primer bit de este campo es de uso reservado, mientras que los siguientes dos bits son:

  DF que indica si un datagrama puede (DF = 0) o no (DF = 1) ser fragmentado.

  MF que indica si un fragmento es (MF = 0) o no (MF = 1) el final de un datagrama.

- **Desplazamiento de un fragmento (13 bits)**: Indica el lugar en el que se sitúa el fragmento dentro del datagrama original, medido en unidades de 64 bits, lo que significa que todos los fragmentos a excepción del último tienen una longitud múltiplo de 64 bits.

- **Tiempo de vida (8 bits)**: Indica el tiempo en segundos que un datagrama puede permanecer en la red. Cada dispositivo que procesa el datagrama tiene que decrementar este campo como mínimo en una unidad, de esta forma se puede concluir que el tiempo de vida es similar a la cuenta de un número limitado de saltos que el datagrama puede realizar en la red.

- **Protocolo (8 bits):** Contiene el código numérico de un protocolo de capa superior que en el destino deberá recibir los paquetes de datos almacenados en el campo de datos del datagrama.

- **Suma de comprobación de cabecera (16 bits):** Es un código de detección de errores aplicado solo a la cabecera del datagrama, detecta únicamente los errores que ocurren durante la transmisión del paquete por la red.

- **Dirección origen (32 bits):** Contiene la dirección IP de la máquina origen.

- **Dirección destino (32 bits):** Contiene la dirección IP de la máquina destino.

- **Opciones (variable):** Es un campo opcional y de longitud variable que permite implementar pruebas y control de la red.

- **Relleno (variable):** Se usa para asegurar que la cabecera del datagrama tiene una longitud múltiplo de 32 bits.

- **Datos (variable):** El campo de datos debe tener una longitud múltiplo de 8 bits. La máxima longitud de un datagrama (cabecera más campo de datos) es de 65535 octetos o 64 kbytes.

1.5.1.2 Direcciones IP.

Todas las direcciones IP tienen 32 bits de longitud y son las que van encapsuladas en los campos dirección origen y dirección destino de los paquetes IP. Cada *host* y enrutador de Internet tienen una dirección IP, la dirección está codificada de tal forma que permite especificar la red y la computadora. Este esquema de codificación proporciona flexibilidad al asignar las direcciones a las computadoras y permite una mezcla de tamaños de red en un conjunto de redes.

En base a lo expuesto anteriormente, existen cinco clases de redes que se las pueden asociar con las siguientes condiciones:

- **Clase A:** Pocas redes, cada una con capacidad para muchas computadoras. Todas las direcciones de clase A empiezan con un 0 binario, los siguientes 7 bits identifican a la red y los 24 bits restantes identifican al *host* o computadora.

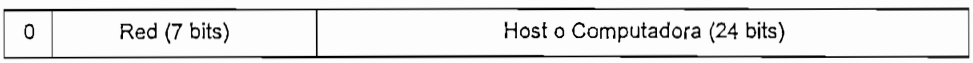


Figura 1.3 Dirección clase A

- **Clase B:** Un número medio de redes, cada una con un número medio de computadoras. Todas las direcciones de clase B empiezan con número binario 10, los 14 bits siguientes identifican a la red y los 16 bits restantes identifican al *host* o computadora.

Figura 1.4 Dirección clase B

- **Clase C:** Muchas redes cada una con pocas computadoras. Todas las direcciones de clase C empiezan con número binario 110, los 21 bits siguientes identifican a la red y los 8 bits restantes identifican al *host* o computadora.

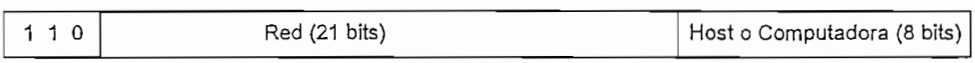


Figura 1.5 Dirección clase C

- **Clase D:** Son usadas especialmente para propósitos de *multicast*.

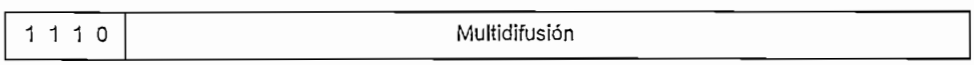


Figura 1.6 Dirección clase D

- **Clase E:** Se usan para propósitos de investigación.

1 1 1 1 0	Para uso futuro e investigación

Figura 1.7 Dirección clase E

Si el número de red o el número de *host* es 255; es decir, todos los bits tienen el valor 1, se trata de una dirección de *broadcast* (toda la red).
Si el número de *host* o el número de red es 0; es decir, todos los bits tienen en valor 0, se refiere a la red actual.

La dirección IP 127.x.y.z perteneciente a la clase A es utilizada para funciones de *loopback*, para probar la comunicación de procesos internos en una máquina local.

A continuación se presenta un resumen de todos los valores posibles de direcciones que se puede tener en cada clase:

Clase	Rango
A	1.0.0.0 a 127.255.255.255
B	128.0.0.0 a 191.255.255.255
C	192.0.0.0 a 223.255.255.255
D	224.0.0.0 a 239.255.255.255
E	240.0.0.0 a 247.255.255.255

Figura 1.8 Rango de direcciones IP

Dentro de este rango de direcciones IP, también existen direcciones que son reservadas para construir las redes llamadas Intranet. Una Intranet no es más que una red TCP/IP privada, implementada por las empresas públicas o privadas. Ninguna máquina que esté conectada a Internet puede poseer alguna de estas direcciones.

Las direcciones IP reservadas para Intranet en cada clase son las siguientes:

- Para las redes clase A, desde 10.0.0.0 hasta 10.255.255.255, en total 1 red.

- Para las redes clase B, desde 172.16.0.0 hasta 172.31.255.255, en total 16 redes.
- Para las redes clase C, desde 192.168.0.0, hasta 192.168.255.255, en total 256 redes.

También existen las subredes que se consiguen al dividir las redes, para ello se utiliza una herramienta llamada *mask* (máscara), la misma que también tiene valores estándar para las diferentes clases de redes.

1.5.2 ICMP PROTOCOLO DE MENSAJES DE ERROR DE INTERNET.

El protocolo *ICMP* (*Internet Control Message Protocol*) es el encargado de proporcionar la información sobre los problemas que ocurren entorno a la comunicación, ya que IP es el protocolo del mejor esfuerzo y ningún sistema funciona bien todo el tiempo, este protocolo permite que los ruteadores envíen mensajes de error o de control hacia otros ruteadores. También se encarga de establecer la comunicación entre el software de IP de una máquina y el mismo software en otra.

Algunas de las razones por las que se usa este protocolo son las siguientes:

- Un datagrama no puede alcanzar su destino.
- Una máquina esta desconectada ya sea en forma temporal o permanente de la red.
- El contador del tiempo de vida del datagrama IP ha expirado.
- Los ruteadores intermedios de la red se han congestionado tanto que no pueden procesar la información entrante.
- El dispositivo de encaminamiento no tiene la capacidad suficiente para almacenar temporalmente un datagrama antes de reenviarlo, cuando la ruta más corta para ese mensaje esté descongestionada.

Aunque ICMP está al mismo nivel que IP, normalmente se considera a este protocolo como un usuario de IP debido a que el destino final de un mensaje

ICMP no es una aplicación o un usuario en la máquina destino sino el software del protocolo Internet en dicha máquina.

Al igual que el resto del tráfico, el mensaje ICMP viaja a través de la red encapsulado en un datagrama IP, por lo que no se garantiza su entrega y su uso no se puede considerar seguro.

En conclusión, técnicamente ICMP es un mecanismo de reporte de errores que permite que los ruteadores encuentren el error y lo reporten a la fuente original, aunque uno de sus deberes es también sugerir las posibles acciones, no específica del todo cual sería la acción para cada error.

1.5.3 IGMP PROTOCOLO DE GESTIÓN DE GRUPO INTERNET.

El protocolo *IGMP* (*Internet Group Management Protocol*) es utilizado tanto por los dispositivos de encaminamiento, así como por las computadoras para intercambiar información entre múltiples miembros de grupos, esta técnica se la conoce como *multicast.*

La multidifusión permite que cada máquina elija si quiere participar en ella. Por lo general se utilizan las direcciones de red clase D para este tipo de comunicación, al igual que ICMP, este protocolo también utiliza al datagrama IP como medio de transporte, por lo que el mensaje es entregado a todos los miembros del grupo en forma no confiable y con el método del mejor esfuerzo.

Cada grupo de multidifusión tiene una dirección multidifusión única, si este es un grupo permanente, no desaparecerá aunque no tenga ningún equipo conectado a esta dirección, si el grupo es temporal las direcciones se pueden asignar en forma dinámica, esta clase de grupo solamente existen cuando poseen miembros.

Este protocolo le permite a un *host* informar a un ruteador *multicast* el deseo de unirse a un grupo *multicast* con la finalidad de recibir todos los *mensajes enviados* a ese grupo, este *host* puede unirse y abandonar el grupo en forma dinámica.

El protocolo IGMP permite que los ruteadores *multicast* sepan a que grupos *multicast* están afiliados algunos *host* de la red en la cual el ruteador está conectado, de esta forma sabe si el mensaje que desea enviar debe difundirlo a toda la red o simplemente descartarlo ya que no existe ningún *host* con la dirección del mensaje en su red o que está afiliado a ese grupo.

1.6 PROTOCOLOS DE CAPA TRANSPORTE

En la arquitectura TCP/IP los protocolos de capa Transporte se sitúan sobre los protocolos de capa Internet, proporcionando los servicios necesarios para los protocolos de capa Aplicación. Todos los protocolos de esta capa brindan un servicio de conexión extremo a extremo, de forma que el envío de mensajes es transparente para el usuario.

En la capa Transporte son posibles dos tipos de servicios básicos: orientado a conexión y no orientado a conexión.

Un servicio orientado a conexión se encarga de realizar el establecimiento, mantenimiento y liberación de la conexión lógica entre usuarios, este tipo de servicio es el más utilizado en la actualidad y tiene una gran variedad de aplicaciones. Esta característica implica generalmente que el servicio es seguro. Un protocolo que cumple con las características antes mencionadas es un protocolo complejo y se llama *TCP* (*Transmission Control Protocol*).

En la capa Transporte también existe un protocolo que proporciona un servicio no orientado a conexión para la capa Aplicación, a este protocolo se lo llama *UDP* (*User Datagram Protocol*), es básicamente un servicio no seguro, por esta razón la cantidad de posibles aplicaciones se reduce notablemente, sin embargo este tipo de servicio puede ser el más apropiado bajo algunos contextos, ya que para algunas aplicaciones el tiempo que se necesita para realizar el establecimiento y la liberación de la conexión no se justifican.

1.6.1 TCP PROTOCOLO DE CONTROL DE TRANSMISIÓN.

Al protocolo *TCP* (*Transmission Control Protocol*) se lo puede definir como entrega de flujo confiable, lo que significa que se garantiza la comunicación extremo a extremo, libre de error y en una secuencia correcta, brindando así muchas facilidades a los programas implementados en la capa Aplicación que necesiten este tipo de servicio.

A pesar de que en niveles más bajos, la arquitectura TCP/IP proporciona una entrega de paquetes no confiable y se corre el riesgo de que estos puedan perderse, duplicarse o destruirse, TCP es el encargado de ocultar estas imperfecciones asumiendo confiabilidad en los protocolos de bajo nivel como es el protocolo IP.

El servicio de entrega confiable proporcionado por TCP se puede caracterizar por las siguientes funciones:

- ***Transferencia continua del flujo de datos:*** Desde el punto de vista de la aplicación TCP transfiere los datos en forma continua como un flujo de bytes a través de la red. Los protocolos de la capa aplicación se limitan a entregar la información que desean enviar a la capa transporte y el protocolo TCP se encarga de fragmentar los datos y encapsularlos en los datagramas IP para la transmisión de los mismos.

  TCP se encarga de negociar el tamaño de los paquetes que van a ser transmitidos entre los extremos, una vez que los paquetes de datos han llegado a su destino a través del protocolo IP, TCP se encarga de reconstruir el flujo de datos original a partir de los datos recibidos antes de enviarlos al programa de aplicación respectivo para su procesamiento.

- ***Confiabilidad:*** TCP asigna un número de secuencia a cada byte transmitido y espera un acuse de recibo positivo (*ACK Acknowledgment*) por parte del receptor, el transmisor guarda un registro de cada paquete transmitido y espera el acuse de recibo antes de enviar el siguiente

paquete. Si este acuse de recibo no llega en un intervalo de tiempo determinado, los datos son retransmitidos. Utiliza para esto la técnica de la ventana deslizante, con la finalidad de aprovechar el ancho de banda.

Muchas veces por la retransmisión los paquetes de datos pueden duplicarse, entonces el destino debe analizar el número de secuencia de cada paquete que recibe y si este es repetido deberá ser descartado.

- ***Control de flujo:*** Al enviar el acuse de recibo el receptor en el *ACK* envía el número de secuencia que espera recibir, además del número de bytes que está en capacidad de almacenar, en base a esta información la máquina de origen utiliza TCP para segmentar sus mensajes para la próxima transmisión con la finalidad de no causar congestión y pérdida de datos, estos datos van almacenados en el paquete *ACK* en forma de un número de secuencia, indicando así el tamaño de la ventana del *host* receptor.

- ***Multiplexación:*** La multiplexación se logra a través del uso de números de puertos de protocolo para identificar el destino final dentro de una máquina. TCP permite multiplexar varias conexiones de transporte utilizadas por la capa aplicación ejecutándose simultáneamente en una misma máquina.

  La conexión consiste en un circuito virtual entre dos programas de aplicación, por que resulta natural asumir al programa de aplicación como punto extremo de la conexión. Un punto extremo no es más que un par de números enteros, uno de ellos es la dirección IP de la máquina y el otro es el puerto TCP en la misma máquina.

- ***Circuitos virtuales:*** TCP es un protocolo orientado a conexión, esto significa que se encarga de establecer, mantener y finalizar una conexión utilizada para la comunicación entre procesos de usuario.

La confiabilidad y el control de flujo son mecanismos requeridos por TCP que inicializan y mantienen cierta información de cada flujo de datos. La combinación de estas condiciones incluyendo los puntos extremos, los números de secuencia, el tamaño de las ventanas forman los llamados circuitos virtuales. Cada conexión es única e identificada por un par de puntos extremos usados para el proceso de envío y recepción de información.

- ***Comunicación Full Dúplex:*** TCP mantiene un flujo de datos bidireccionales utilizando circuitos virtuales.

1.6.2 UDP PROTOCOLO DE DATAGRAMAS DE USUARIO.

El protocolo *UDP* (*User Datagram Protocol*) es un protocolo no orientado a conexión, no confiable; es decir, la entrega y protección contra duplicación no está garantizada.

UDP proporciona el mecanismo utilizado por los programas de aplicación para enviar datagramas a otros programas de aplicación, no emplea acuses de recibo para asegurarse que lleguen los mensajes, no ordena los mensajes ni controla la velocidad a la que fluye la información entre las máquinas, los paquetes pueden llegar más rápido de lo que el receptor los puede procesar; por lo tanto, los mensajes pueden perderse, duplicarse o llegar sin orden. Al igual que TCP, el protocolo UDP se encapsula en un paquete IP para viajar a su destino.

Pero tanto a nivel de capa Transporte y de niveles superiores existe una justificación para un servicio no orientado a conexión. Existen casos en los que el establecimiento y mantenimiento de la conexión no están justificados como por ejemplo: recolección de datos como muestreo, aplicaciones en tiempo real, entre otros.

Entre las aplicaciones estandarizadas que utilizan este protocolo están: TFTP (*Protocolo trivial de transferencia de archivos*), DNS (*Sistema de dominio de*

nombres), RPC (*Llamada a procedimiento remoto*) y SNMP (Protocolo sencillo de administración de red).

1.7 PROTOCOLOS DE CAPA APLICACIÓN

En esta capa se encuentran los protocolos que le permiten al usuario ejecutar las aplicaciones, las mismas que pueden ser desde un programa escrito por un usuario o una de las aplicaciones estandarizadas por el conjunto de protocolos de la arquitectura TCP/IP. Las aplicaciones más conocidas son: DNS, TELNET, FTP, TFTP, SMTP, HTTP, etc.

Cualquiera de las aplicaciones anteriormente mencionadas tienen que interactuar con los protocolos implementados en la capa transporte, para que esta se encargue de enviar y recibir los datos en forma de mensajes o de flujos continuos de bits, según lo requiera la aplicación.

La mayoría de aplicaciones utilizan el modelo cliente – servidor al momento de establecer una comunicación.

El modelo cliente – servidor es una arquitectura de hardware y software adecuada para el proceso distribuido, en el que la comunicación se establece de uno a varios. El proceso cliente es el que solicita un servicio, mientras que el proceso servidor es capaz de proporcionar un servicio. Un proceso cliente se puede comunicar con varios procesos servidores y un servidor se puede comunicar con varios clientes. En el caso de los servicios más sencillos, cada petición llega en un solo datagrama IP y el servidor devuelve una respuesta en otro datagrama.

Los servidores pueden ejecutar tareas simples o complejas y se suelen implantar como aplicaciones de programas, la ventaja que se obtiene es que se puede ejecutar en cualquier sistema que soporte la comunicación TCP/IP. A continuación se detallan algunas de las aplicaciones más conocidas en Internet.

1.7.1 DNS SISTEMA DE DOMINIO DE NOMBRES.

Las primeras configuraciones de Internet obligaban a los usuarios a que usaran direcciones IP solo numéricas, pero rápidamente esto evolucionó para poder usar nombres de *host* simbólicos que sean fáciles de recordar al momento de referirse a una máquina en Internet.

El DNS tiene dos aspectos conceptuales independientes: El primero especifica la sintaxis del nombre y las reglas para delegar autoridad de acuerdo a los nombres. El segundo especifica la implementación de un sistema de nombres distribuido, con la finalidad de transformar eficientemente los nombres en direcciones.

En esencia el uso de DNS, es una invención de un esquema de nombres jerárquico basado en dominio y en una base de datos distribuida para implementar este esquema de nombres. Se usa principalmente para relacionar las direcciones de *host* y destinos de correo electrónico con las direcciones IP, pero también puede emplearse para otros fines.

Para poder asignar un nombre a cada *host* que lo identifique en la red, en las redes TCP/IP se usa una estructura de nombres jerárquica que se asigna de acuerdo a la estructura de la organización, la misma que puede ser en base a su red física o a cualquier otra forma que se desee para facilitar la administración. A este esquema jerárquico basado en un árbol invertido se lo llama espacio de dominio de nombres y parte de este esquema se muestra en la siguiente figura:

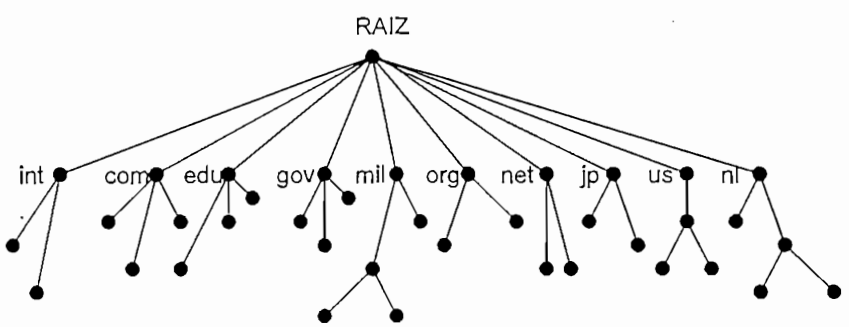


Figura 1.9 Parte del espacio de dominio de nombres de Internet.

Cada dominio se divide en subdominios y estos nuevamente, etc. Los dominios de nivel superior pueden ser de dos clases: genéricos y de país. Los componentes en la jerarquía se separan por puntos, además no hacen distinción entre mayúsculas o minúsculas. Los nombres de componentes pueden ser hasta de 63 caracteres de longitud y los nombres de trayectoria completa no deben exceder los 255 caracteres. Los nombres de dominio pueden ser absolutos y relativos. Un nombre de dominio absoluto termina con un punto (*por ejemplo: eng.su.com.*) mientras que un nombre de dominio relativo no.

Cada dominio controla el modo de asignación de los dominios que están debajo de él. Para crear un dominio nuevo se requiere el permiso del dominio de mayor jerarquía en el que se incluirá el nuevo dominio, con la finalidad de evitar conflictos de nombres. Cada dominio puede llevar un registro de todos sus subdominios.

A continuación se enumeran algunos dominios genéricos o etiquetas especiales que representan los distintos tipos de organizaciones conectadas a Internet:

- com: Organizaciones Comerciales
- edu: Instituciones Educativas
- gov: Instituciones Gubernamentales
- int: Organizaciones Internacionales
- mil: Instituciones Militares
- net: Proveedores de Servicio de Internet
- org: Organizaciones no lucrativas

El modo en que se usa DNS es el siguiente: Para relacionar un nombre con una dirección IP, un programa de aplicación llama a un procedimiento denominado *resolvedor* al cual se le transfiere el nombre como un parámetro, este se encarga de enviar un paquete UDP a un servidor DNS local, el servidor busca el nombre y devuelve la dirección IP correspondiente al *resolvedor*, quien se encarga de enviarla al solicitante. Una vez que el solicitante tiene la dirección IP, está en condiciones de establecer la conexión TCP con el destino o enviarle paquetes UDP.

Este sistema está distribuido en un conjunto de servidores que opera en varias localidades en forma conjunta para resolver el problema de la asociación de nombres en direcciones, resulta eficiente ya que la gran mayoría de nombres se pueden asociar en forma local y muy pocos son los que requieren tráfico en la red.

Un servidor de nombres no es más que un programa servidor que ofrece la asociación nombre – dirección agrupando los nombres de dominio con direcciones IP, normalmente este software está cargado en una máquina que se dedica exclusivamente a brindar este servicio, por lo que a esta máquina se le asigna el nombre de servidor.

Un servidor de nombres no puede contener toda la base de datos de DNS y responder a todas las consultas que recibe por muy alta que sea su capacidad ya que correría el riesgo de sobrecargarse y ser inservible, por tal razón generalmente los dominios están repartidos en diferentes servidores, los mismos que poseen respaldos en caso de que hubieren fallas y además como precaución e inclusive como sistema de administración de estos recursos se ha dividido al espacio de nombres DNS en zonas no traslapadas.

Cada zona contiene una parte del árbol y por tanto servidores de nombres que contienen la información de autorización correspondiente a esa zona, aunque existen servidores que pueden tener autoridad sobre varias zonas. Normalmente una zona esta formada por un servidor de nombres primario es decir que la información la obtiene de un archivo en su disco y uno o más servidores secundarios que obtienen la información del servidor de nombres primario.

Los límites en cada zona son establecidos en base a la capacidad de administración de la misma, así como también del número de servidores y de su ubicación. Una zona está formada por un único dominio o por el dominio con sus correspondientes subdominios.

1.7.2 TELNET.

TELNET, también conocido como *Protocolo de Terminal de Red* es el protocolo de conexión a otro ordenador, de hecho la mayoría de aplicaciones se basan en este protocolo, ya que facilita la posibilidad de conexión remota, mediante la cual un usuario desde una computadora personal se conecta a un servidor o a una computadora remota y trabaja como si estuviera conectado directamente a ese equipo.

Con este protocolo un usuario puede acceder en forma remota al disco duro, manejar archivos, ejecutar aplicaciones, acceder a datos, imprimir un documento en el destino remoto, modificar las configuraciones, etc. Aunque TELNET no es sofisticado en comparación con otros protocolos existentes, es ampliamente utilizado.

"El usuario inicia una sesión remota especificando la dirección IP *(194.106.2.15)* o el nombre de dominio *(máquina.remota.es)* de la computadora a la cual se desea conectar e inclusive se puede indicar el número de puerto si es necesario, aunque por lo general el protocolo TELNET utiliza el puerto 23 para atender las peticiones de sus clientes, además del protocolo TCP como transporte, con información de control TELNET intercalada."[3]

Una vez establecida la conexión el servidor TELNET solicita el nombre de usuario y el password para determinar si el cliente tiene acceso a la información en la máquina remota a la cual desea acceder, a continuación transfiere cada caracter tecleado por el cliente directamente al otro sistema y muestra los datos que se obtienen de la máquina remota en la pantalla del usuario.

El protocolo TELNET se basa en tres ideas principales:

- Un Terminal Virtual de Red NTV *(Network Terminal Virtual).*
- Un principio de opciones negociables
- Una visión simétrica de terminales y procesos.

{3} RFC 854, Especificación del Protocolo Telnet, Agosto del 2001.

"Dada la gran variedad de hardware de visualización existente en el mercado y para evitar problemas de compatibilidad el protocolo TELNET define un *Terminal Virtual de Red (NTV)* que es un dispositivo imaginario que proporciona una representación intermedia de un terminal, con una serie de funcionalidades que deben cumplir todos los sistemas, ya que este protocolo se supone que se inicia y se finaliza en un NTV, es decir; que permite la interoperabilidad entre computadoras reales basadas en diferentes sistemas operativos, cada uno de los cuales maneja un formato distinto de caracteres, especialmente los de control. La información enviada por el usuario es traducida al formato del NTV y enviada al servidor, el cual traduce los datos del NTV al formato que dicho servidor utiliza. De igual manera ocurre entre la comunicación servidor – cliente."[3]

El *principio de opciones negociables* tiene en cuenta el hecho de que muchos ordenadores querrán proporcionar servicios adicionales a los disponibles en un NVT y muchos usuarios tendrán terminales sofisticados y querrán disponer de todos los servicios posibles en lugar de los mínimos. Hay "opciones" independientes del protocolo TELNET pero estructuradas dentro de él que se podrían usar y que permiten a un usuario y a un servidor ponerse de acuerdo para usar convenciones más elaboradas (o tal vez solo diferentes) para sus conexiones TELNET. Entre esas opciones se podrían incluir el cambio del juego de caracteres, el modo de eco, etc.

En la medida de lo posible, el protocolo TELNET se ha hecho simétrico, una *visión simétrica de terminales y procesos* entre el servidor y el usuario, es ideal para adaptarse a conexiones usuario – usuario y servidor – servidor, además permite que un cliente pueda ser un usuario con un teclado o un programa que envía los caracteres al servidor.

En la siguiente figura se ilustra la forma en que los programas de aplicación comunican a un cliente con un servidor de TELNET.

[3] RFC 854, Especificación del Protocolo Telnet, Agosto del 2001.

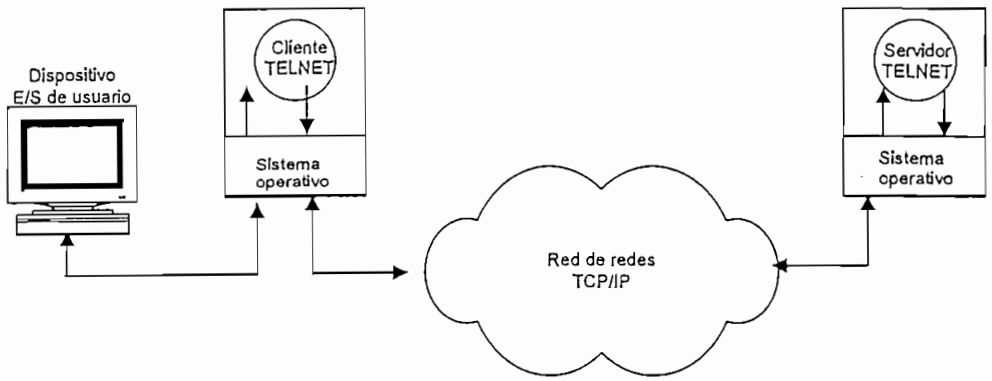


Figura 1.10. Esquema de funcionamiento

Como se muestra en la figura cuando un usuario inicia una comunicación TELNET, un programa de aplicación de la máquina del usuario se convierte en el cliente. El cliente establece una conexión TCP con el servidor por medio de la cual se comunicarán.

Una vez que se ha establecido la conexión, el cliente acepta los pulsos del teclado del usuario y los envía al servidor, al mismo tiempo que el servidor acepta estos caracteres también envía al usuario la información solicitada y la despliega en la pantalla.

"Este protocolo es útil para depurar e incluso implementar completamente otros protocolos como por ejemplo POP3, SMTP, IMAP4 y FTP, no obstante, TELNET no suele ser una aplicación típica disponible en los paquetes de software distribuidos por los proveedores de acceso a Internet."[3]

1.7.3 FTP PROTOCOLO DE TRANSFERENCIA DE ARCHIVOS

El protocolo *FTP (File Transfer Protocol)*, permite a un usuario en cualquier computadora *(cliente)* traer archivos desde cualquier otra computadora *(servidor)* o enviar archivos a cualquier otra computadora en una red TCP/IP como Internet, además permite controlar el acceso de los usuarios. Este protocolo tiene más de dos décadas de existencia con muy buena acogida.

---

[3] RFC 854, Especificación del Protocolo Telnet, Agosto del 2001.

Cuando un usuario solicita la transferencia de un archivo, el protocolo FTP establece una conexión TCP con el sistema destino para intercambiar mensajes de control. Esta conexión permite al usuario transmitir su identificación de usuario y contraseña, además de la identificación del archivo junto con las acciones que se desean realizar sobre el mismo.

Para proporcionar acceso a los archivos públicos, muchas de las localidades TCP/IP permiten el FTP "*anónimo*", lo que significa que el cliente no necesita una cuenta o clave de acceso, únicamente tiene que especificar un nombre de conexión *anónimo* y una clave de acceso de *invitado*. De esta manera el servidor discrima el tipo de usuario y el acceso se restringe únicamente a los archivos públicos.

FTP usa TCP como protocolo de transporte para brindar una conexión extremo a extremo confiable y establece dos tipos de conexiones. La primera es una conexión de control que utiliza el protocolo TELNET, en esta conexión se utiliza el concepto del terminal virtual de red evitando así preocuparse del formato de los datos de control que necesita enviar. Una vez que el archivo se ha especificado y su transferencia ha sido aceptada, se establece una segunda conexión TCP a través de la cual se materializará la transferencia. El archivo se transmite a través de la segunda conexión, sin necesidad de enviar conexión extra o de agregar cabeceras generadas por la capa aplicación al paquete de datos. Cuando la transferencia ha terminado, se utiliza la conexión de control para indicar el fin, además esta misma conexión estará disponible para aceptar nuevas órdenes de transferencia.

Por lo general, el cliente y el servidor crean un proceso separado para manejar la transferencia de datos y la operación de control, las dos conexiones permanecen activas mientras el usuario continúa con la sesión FTP. Sin embargo FTP establece una nueva conexión de transferencia de datos para cada transferencia de archivos; es decir, las conexiones de transferencias de datos y los procesos de transferencias de datos pueden crearse de manera dinámica cuando se necesitan, mientras que la conexión de control continúa a través de una sesión.

Una vez que la conexión de control desaparece, la sesión se da por terminada y el software a ambos extremos se encarga de terminar todos los procesos de transferencia de datos.

La conexión de control utiliza en el servidor el puerto TCP número 21 y en el cliente un puerto libre, mientras que en la transferencia de datos el servidor utiliza el puerto TCP número 20 y el puerto del cliente es asignado en forma dinámica mediante un acuerdo entre los extremos, para esto usa la conexión de control asegurando la transferencia correcta con el cliente.

El protocolo FTP no permite realizar negociación de opciones como lo hace TELNET, solamente emplea la definición básica de *NTV*, facilitando de esta manera la administración de la conexión de control, siendo esta más sencilla que la conexión estándar de TELNET.

Cuando se usa FTP, el usuario realizará todas o algunas de las siguientes operaciones:

- Conexión a un *host* remoto.
- Escoger un directorio.
- Lista de archivos disponibles para la transferencia.
- Definir el modo de transferencia.
- Copiar los archivos desde el *host* remoto
- Desconectarse del *host* remoto.

1.7.4 TFTP PROTOCOLO TRIVIAL DE TRANSFERENCIA DE ARCHIVOS

TFTP (Trivial File Transfer Protocol), proporciona un servicio económico y poco sofisticado, diseñado para aplicaciones que no necesitan interacciones complejas entre el cliente y el servidor.

A diferencia del protocolo FTP, TFTP restringe las operaciones a una fácil transferencia de archivos sin necesidad de autenticación, por lo que inclusive en tamaño el software de TFTP resulta mucho más pequeño que el de FTP, siendo esta característica muy importante para muchas aplicaciones.

Este protocolo utiliza un servicio de transporte no confiable como es UDP o cualquier otro protocolo que le proporcione la característica antes mencionada, además utiliza tiempos límites de transmisión para asegurar que los datos lleguen a su destino.

El cliente al iniciar una transferencia con el servidor, envía un paquete de solicitud indicando si este es de lectura o de escritura al puerto UDP 69, el mismo que ha sido asignado para el servidor TFTP, el cliente puede utilizar cualquier puerto que este disponible. Además de indicar el tipo de paquete de solicitud, también contiene el nombre del mismo y el formato de los datos que van en él.

Una vez que el servidor ha aceptado la solicitud, se establece la conexión y se continua con la transferencia de los paquetes de datos, cada archivo tiene un tamaño fijo (*512 bytes*), bien identificados ya que cada bloque posee un número para el efecto.

El *host* origen envía el primer paquete de datos, se bloquea y espera un acuse de recibo por cada bloque antes de enviar el siguiente, el *host* destino o servidor transmite un acuse de recibo por cada bloque que llega, en él se envía el número que identifica al paquete que fue recibido correctamente. Para indicar que se trata del final del archivo se envía un paquete de menos de 512 bytes. Si se envía un paquete de error en lugar de los datos o del acuse de recibo, este finaliza la transferencia automáticamente.

Para comunicarse con el *host* destino se debe especificar la dirección IP del mismo, además del número de puerto de protocolo UDP del cliente o *host* origen con la finalidad de identificar las operaciones subsiguientes, de esta manera ni los

mensajes de datos o los acuses de recibo deben especificar el nombre del archivo.

Para el proceso de retrasmisión de datos, cada uno de los extremos establece un tiempo límite, así, si en el origen se excede este tiempo, se vuelve a transmitir el último paquete de datos, lo mismo ocurre en el destino con los mensajes de acuse de recibo, de esta manera se asegura que la transmisión no ha fallado y que no se tiene paquetes perdidos.

1.7.5 SMTP PROTOCOLO SENCILLO DE TRANSFERENCIA DE CORREO.

El *SMTP (Simple Mail Transfer Protocol)* un protocolo *sencillo de transferencia de correo*, proporciona una función básica de correo electrónico. Se encarga de transferir los mensajes en forma confiable entre servidores de correo o entre computadoras remotas.

Entre las propiedades del SMTP cabe destacar la utilización de listas mensajería, la gestión de acuses de recibo y el reenvío de mensajes. Este protocolo no especifica como se crean los mensajes, para conseguir este objetivo se necesita de la ayuda de un programa de correo electrónico nativo o un editor local. Una vez que se ha creado el mensaje, SMTP lo acepta, lo guarda hasta que sea transmitido y sea copiado exitosamente haciendo uso del protocolo TCP para enviarlo al módulo SMTP en la computadora remota. En el receptor, el módulo SMTP utilizará una aplicación de correo electrónico local para almacenar el mensaje recibido en el buzón de correo del usuario destino.

Este protocolo utiliza un método diferente a *store and forward* empleado por la mayoría de los protocolos en sistemas de envío de correo. El paquete de datos debe cruzar por varios servidores intermedios dentro de la red hasta llegar a su destino final.

El correo electrónico se basa en el modelo cliente – servidor. El cliente es el *host*, que se encarga de enviar el mensaje y el servidor es quien recibe el mensaje.

Cuando un usuario conectado a un servidor de correo desea enviar un mensaje, establece una conexión TCP con el puerto 25 del servidor de correo local para enviar su mensaje y espera una respuesta. El servidor local envía una línea de texto mediante la cual se identifica e indica si está o no preparado para recibir el correo. Si no lo está, el cliente libera la conexión y lo intenta después.

En el caso que el servidor este dispuesto a aceptar el correo electrónico, el cliente indica de quien viene el mensaje y a quien está dirigido. Si existe tal destino, el servidor autoriza al cliente el envío, una vez que el mensaje ha sido recibido el servidor envía el acuse de recibo, por lo general no se hace control de errores ya que TCP proporciona un servicio orientado a conexión y confiable.

SMTP utiliza un conjunto de comandos definidos para el diálogo cliente – servidor. El cliente por lo general utiliza comandos de cuatro letras, que tienen un significado especial para el servidor, mientras que el servidor envía mensajes formados por dos partes: un código numérico utilizado por el proceso del cliente y una parte literal utilizada para información del usuario, la misma que no es procesada. Todos los comandos, respuesta y datos que se intercambian son líneas de texto ASCII. SMTP utiliza una secuencia de pasos con la finalidad de transferir un mensaje entre un cliente y el servidor, esta secuencia es la siguiente:

1. "El cliente establece la conexión TCP con el servidor y espera a que el servidor le envíe el mensaje *220 Service ready message,* indicando que esta listo para iniciar la transferencia de correo o el mensaje *421 Service not available* cuando el destino esta deshabilitado y no puede continuar con el proceso." [6]

2. Una vez que se ha recibido el mensaje *220*, el cliente envía el comando *HELO* identificándose junto con el nombre de dominio al servidor.

3. El servidor envía un mensaje identificándose con su nombre de dominio.

---

[6] Murhammer Martin W., Bretz Stefan, Pugh Larry R., Suzuki Kazunari, Wood David H., "TCP/IP TUTORIAL AND TECHNICAL OVERWIEV", IBM, Sexta Edición, Octubre 1998, www.redbooks.ibm.com

4. El cliente envía el comando *MAIL* que proporciona la identificación del emisor y el campo *FROM* que contiene la dirección a la que se deben reportar los errores.

5. El servidor prepara su estructura de datos para recibir un nuevo mensaje de correo y responde al comando *MAIL* con el comando *250 OK* que significa que está listo para recibir el mensaje.

6. El cliente envía una serie de comandos *RCPT* que identifican a los recipientes del mensaje de correo.

7. El servidor envía un acuse de recibo por cada comando *RCPT*, el acuse de recibo es un *250 OK* en el caso de que no existan errores, pero si esto sucede, envía un mensaje de error *550 No Duch user here*.

8. Una vez que el servidor ha reconocido todos los mensajes *RCPT* el cliente envía el comando *DATA* indicando al servidor que está listo para transmitir un mensaje de correo completo.

9. El servidor responde con un mensaje *354 Start mail input* seguido de una secuencia de caracteres que el cliente deberá utilizar para dar por terminado el mensaje de correo. Normalmente utiliza el comando *CR LF*.

10. Cuando la transmisión de mensajes ha terminado, el cliente envía el comando *QUIT* indicando que desea terminar la sesión.

11. El servidor emite el comando *221* que significa que está de acuerdo en terminar la comunicación.

Entonces ambos lados han cerrado la conexión TCP. Estos no son los únicos comandos existentes pero sí son los más usados cuando se trata de una transmisión simple. Hay muchos otros comandos que inclusive permiten realizar funciones más complejas.

1.7.6 SNMP PROTOCOLO SIMPLE DE ADMINISTRACION DE RED

"*SNMP* (*Simple Network Monitoring Protocol*) es un protocolo estándar utilizado para monitorear servidores, ruteadores y las redes a las que están conectados"[5], aunque este protocolo opera a nivel de capa Aplicación, su comunicación se realiza mediante los protocolos de capa Transporte.

El modelo *SNMP* de una red administrada consta de cuatro componentes:

- Nodos administrados.
- Estaciones administradas.
- Información de administración.
- Un protocolo de administración.

Los nodos administrados pueden ser *hosts*, enrutadores, puentes, impresoras u otros dispositivos capaces de mostrar información de estado al mundo exterior. Para poder ser administrado por *SNMP* este nodo debe tener la capacidad de ejecutar un proceso de administración llamado *agente SNMP*. Todas las computadoras cumplen con este requisito al igual que una gran cantidad de dispositivos periféricos diseñados para el uso en redes.

Para la administración de la red se usan estaciones administradoras, que son computadoras de propósito general que ejecutan un software de administración especial, esta estación contiene uno o más procesos que se comunican con los agentes a través de la red emitiendo comandos y recibiendo respuestas. Muchas de estas estaciones administradoras poseen una interfaz gráfica de usuario que le permite al administrador de la red inspeccionar el estado de la red y emprender acciones cuando sea necesario.

El protocolo *SNMP* a diferencia de muchos de los protocolos comunes de administración de red existentes que definen un extenso conjunto de comandos para la administración de red, reúne todas las operaciones en el paradigma *obtener – almacenar*; es decir, tiene tan solo dos comandos que le permiten al

[5] Comer, Douglas, "REDES GLOBALES DE INFORMACION CON INTERNET Y TCP/IP, PRINCIPIOS BASICOS Y ARQUITECTURA", Editorial Prentice – Hall, Tercera Edición, México, 1996.

administrador buscar y obtener un valor desde un elemento de datos o almacenar un valor en un elemento de datos, todas las demás operaciones están definidas como consecuencia de las dos anteriores.

La mayor ventaja al usar este paradigma es la estabilidad, simplicidad y flexibilidad. Se dice que es en especial estable ya que mantiene fijas sus definiciones a pesar de que se añaden nuevos elementos al administrador de información base o *MIB* (*Management Information Base*) y se definen nuevas operaciones como consecuencia del almacenamiento.

El *MIB* no es más que un estándar que especifica los elementos de los datos que manejan los servidores o los ruteadores, los mismos que deben ser conservados, también indica las operaciones que pueden realizar cada uno.

El *SNMP* es simple en lo que su implementación se refiere, es fácil de entender y depurar, no es complejo al momento de manejar casos especiales para los comandos y es flexible ya que se puede adaptar a comandos arbitrarios dentro de una estructura.

Algunos de los comandos que maneja *SNMP* son:

get-request	Obtener un valor desde una variable específica.
get-next-request	Obtener un valor sin conocer su nombre exacto.
get-response	Replicar a una operación fetch.
set-request	Almacenar un valor en una variable específica.
trap	Réplica activada por un evento.

El formato de los mensajes *SNMP* a diferencia de la mayor parte de protocolos que se usan en las redes TCP/IP no tienen campos fijos pero contienen tres partes principales: una versión del protocolo, un identificador *community* de *SNMP* que se utiliza para reunir a los ruteadores administrados por un solo administrador y por último el área *data* o área de datos, esta área se divide en unidades de datos de protocolo *PDU's* (*Protocol Data Units*). Un *PDU* puede ser una solicitud enviada por un cliente o una respuesta enviada por un servidor.

1.7.7 WWW WORLD WIDE WEB

El WWW es una de las aplicaciones más importantes del Internet, es un mecanismo proveedor de información electrónica o documentos conocidos como páginas Web para usuarios conectados a Internet. El acceso a cada sitio Web se canaliza a través del URL o identificador único de cada página de contenidos.

Este sistema permite a los usuarios el acceso a una gran cantidad de información: leer publicaciones periódicas, buscar referencias en bibliotecas, realizar paseos virtuales, compras electrónicas, audiciones de conciertos, buscar trabajo y otras muchas funciones. Gracias a la forma en que está organizada la World Wide Web, los usuarios pueden saltar de un recurso a otro con facilidad debido a los enlaces a otras páginas conocidos como hipervínculos. Las páginas Web están almacenadas en servidores conocidos como servidores Web y las conexiones entre los servidores que contienen la información se hacen de forma automática y transparente para el usuario, pues el medio admite las funciones de hipertexto e hipermedia.

Existen múltiples enlaces Web por todo el mundo, que forman una base de información a gran escala en formato multimedia, aunque todavía los contenidos se encuentran mayoritariamente en inglés. Cada vez más compañías implantan redes corporativas, conocidas con el nombre de Intranets, que están basadas en esta tecnología pero a menor escala.

"Generalmente se utilizan los términos Web e Internet como sinónimos, pero en verdad no son la misma cosa. El Web es un subconjunto de la red Internet, una colección de documentos de hipertexto e hipermedia enlazados entre sí. El Internet es la colección de máquinas y enlaces que permiten el almacenamiento e intercambio de información".[2] Las páginas Web pueden estar escritas en *HTML* (*Hypertext Markup Language*), *DHTML* o *XML* (*Extended Markup Language*), lenguajes de marcado de hipertexto.

---

[2] Barreto, Alexis, "ESTUDIO Y ANÁLISIS DE LAS DISTINTAS TECNOLOGÍAS DE ACCESO QUE UN PROVEEDOR DE SERVICIOS DE INTERNET PUEDE IMPLEMENTAR EN ECUADOR", Tesis previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional, Quito – Ecuador, Diciembre de 1999.

Los usuarios visualizan estos datos mediante una aplicación conocida como explorador o *browser.* El explorador muestra en pantalla una página con el texto, las imágenes, los sonidos y animaciones relativas al tema que previamente ha sido seleccionado. El navegante puede interactuar con el sistema señalando con el *mouse* (ratón) aquellos elementos que están en capacidad de desplegar mayor información, pues dichos objetos están vinculados a otras páginas Web de ese servidor u otros; es decir son *hipervínculos.*

1.7.7.1 Web Browsers

Un *browser* no es más que una aplicación que proporciona acceso a un servidor Web. Dependiendo de la aplicación la estructura y capacidad del *browser* pueden variar. Un Web *browser* como mínimo consta de un intérprete *HTML* y un cliente *HTTP* el cual es usado para acceder a páginas Web *HTML.* Además de este requisito básico, muchos *browsers* también soportan *FTP, NNTP, e-mail* entre otros con una interfaz básica fácil de manejar.

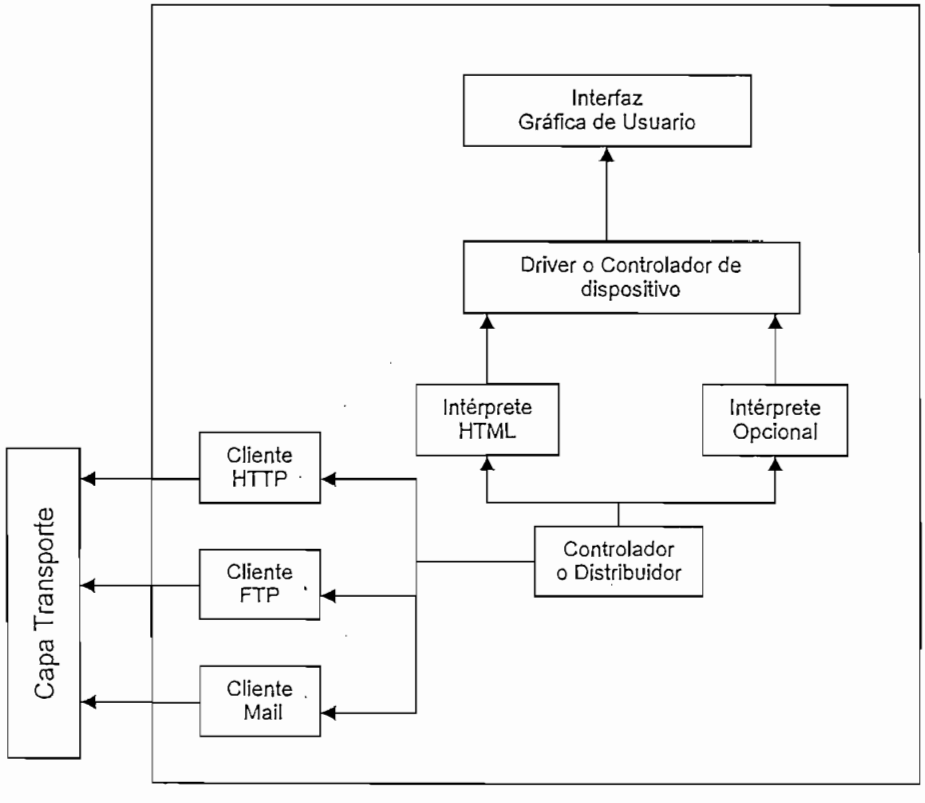


Figura 1.11 Estructura básica de un Web Browser.

Como otros medios de Internet, la Web también usa el modelo cliente – servidor. El Web *browser* es el componente del cliente y es el responsable de la estructura y despliegue de la información actuando en forma recíproca con el usuario e invocando funciones externas tales como: TELNET u otro tipo de funciones para otro tipo de datos externos que los Web *browsers* no soportan en forma directa.

Como ejemplos de Web *browsers* podemos mencionar: *Microsoft Internet Explorer, Netscape Navigator, Sun Hot Java*, etc.

1.7.7.2 Web Server

Los servidores Web son los responsables de proporcionar la información solicitada por los Web *browsers* a otras computadoras, esta información puede ser un archivo almacenado y recuperado desde el disco local del servidor o puede ser generado por un programa de aplicación llamado por el servidor para realizar una función específica. Para ello el servidor debe estar conectado a Internet y debe tener asignada una dirección IP.

El WWW utiliza tres componentes fundamentales:

- ***URL*** *Universal Resource Locutor:* Permite identificar en forma única un documento en el Web.
- ***HTTP*** *Hipertext Transfer Protocol:* Es el protocolo encargado de las comunicaciones entre servidores y navegadores de Web.
- ***HTML*** *Hipertext Markup Language:* Define el formato de los documentos Web y la forma en que un documento debe ser presentado ante un usuario.

1.7.7.3 URL Localizador Uniforme de Recursos

El *URL* (*Uniform Resource Locator*) sirve como nombre mundial de una página Web. Cada página tiene asignado un nombre único, a través del *URL* se puede saber el nombre de la página, donde está ubicada dicha página y como se puede acceder a ella.

Por tanto el *URL* consta de tres partes: el protocolo o esquema, el nombre *DNS* de la máquina en la que se encuentra la página y el nombre local que indica de

manera única la página específica. En el ejemplo siguiente se indican cuales son las partes antes mencionadas:

http://www.cs.vu.nl/welcome.html

Este *URL* consta de las siguientes partes: el protocolo (*http*), el nombre *DNS* del *host* (*www.cs.vu.nl*) y el nombre del archivo (*welcome.html*), con la puntuación respectiva que sirve para separar cada una de las partes.

Como se había mencionado anteriormente, las páginas Web tienen apuntadores a otras páginas Web usando el hipertexto o un icono, para acceder a esta nueva página, debe darse un *clic* y se despliega la página que tiene la información relacionada. Este vínculo conoce el *URL* de la página a la que se puede pasar con esta acción.

Al seleccionarse este texto, lo primero que se busca es el nombre del *host* usando *DNS*, una vez que se tiene la dirección IP del *host* se establece una conexión TCP con él y se envía por esta conexión el nombre del archivo usando el protocolo especificado.

La parte de protocolo se refiere al tipo de protocolo utilizado para acceder a este recurso, por lo general el más utilizado para la transferencia de archivos Web es el protocolo *HTTP*. Sin embargo un *browser* o navegador Web puede ser utilizado para acceder a varios recursos. Cada protocolo tiene asignado un nombre que permite al *browser* identificar el tipo de protocolo que debe usar para transferir un recurso.

Los *URL* se han diseñado no solo para permitir a los usuarios navegar por la Web sino también para entenderse con otros tipos de servicios como *FTP, Telnet,* etc., haciendo innecesario el uso de un programa especializado con una interfaz gráfica de usuario diferente para cada uno de estos servicios, integrando de esta forma casi todos los accesos a Internet en un solo programa, el visualizador de la Web.

Los nombres de *URL* más utilizados son:

Nombre	Usado para:
http	Hipertexto (HTML)
ftp	FTP
gopher	Gopher
mailto	Dirección de correo electrónico
news	Grupo de noticias o artículo de noticias
nntp	Artículo de noticias
telnet	Acceso remoto
wais	Wais
File	Nombre de archivo en un *host* local

Figura 1.12 Nombres de algunos URL más comunes

1.7.7.4 HTTP Protocolo de Transferencia de Hipertexto

El *HTTP* (*HyperText Transfer Protocol*) es el protocolo estándar de transferencia de la Web; es decir, es el encargado de hacer llegar las diferentes páginas desde los servidores remotos al equipo del usuario y se puede utilizar en cualquier aplicación cliente – servidor que suponga la utilización de hipertexto.

Este es un protocolo de capa aplicación orientado a conexión que utiliza TCP como protocolo de transporte y define un conjunto de reglas para poder intercambiar recursos tales como: archivos de texto, imágenes, sonido, video y otros archivos de multimedia o hipermedia a través del WWW, a pesar de que su nombre se presta a confusiones ya que no es un protocolo para transferir hipertexto.

A continuación se definen algunos de los términos utilizados en las definiciones de *HTTP*:

Hipertexto: Es un método de presentación de información en el que el texto, las imágenes, los sonidos y las acciones están unidos mediante una red compleja y no secuencial de asociaciones que permite al usuario examinar los distintos temas, independientemente del orden de presentación de los mismos.

Normalmente es el autor el que establece los enlaces de un documento hipertexto en función de la intención del mismo. El término hipertexto fue creado por Ted Nelson en 1965, con el fin de describir los documentos que se presentan en un ordenador o computadora, expresando la estructura no lineal de las ideas, al contrario de la estructura lineal de los libros, las películas y el habla.

Hipermedia: El término hipermedia es prácticamente un sinónimo de hipertexto, pero recalca los componentes no textuales del hipertexto, como animaciones, sonido y video. Hipermedia significa integración de gráficos, sonido y video en cualquier combinación para formar un sistema de almacenamiento y recuperación de información relacionada y de control de referencias cruzadas. La hipermedia y especialmente en el formato interactivo en el que el usuario controla las opciones, se estructuran alrededor de la idea de ofrecer un entorno de trabajo y de aprendizaje similar al pensamiento humano.

Un entorno de este tipo debe permitir al usuario establecer asociaciones entre los distintos temas, en lugar de desplazarse secuencialmente de uno en uno, como ocurre en las listas alfabéticas. Por ello, los temas hipermedia están vinculados entre sí para permitir al usuario saltar de un concepto a otro relacionado para buscar más información. Por ejemplo, una presentación hipermedia acerca de navegación puede incluir enlaces a temas como la astronomía, la migración de las aves, la geografía, los satélites y el radar. Si la información se encuentra primordialmente en forma de texto, el producto es de hipertexto. Si por el contrario se incluyen videos, música, animación u otros elementos, se habla de un producto hipermedia.

Usuario: Es el cliente que inicia una petición. Entre estos se pueden incluir a los navegadores o *browsers*, editores y otras herramientas de usuario final.

Servidor original: Es un programa de aplicación que acepta conexiones para servir peticiones mediante respuestas, en él reside un recurso dado o se va a crear el recurso.

Recurso: Es un objeto de datos o un servicio de red que puede ser identificado por un *URL*.

Cliente: Es un programa de aplicación que establece conexiones con el propósito de enviar peticiones.

Proxy: Es un programa intermedio que actúa como un servidor o como un cliente con el objeto de hacer las peticiones de parte de otros clientes, las mismas que pueden ser enviadas directamente o pueden ser traducidas a otros servidores, ya que el *proxy* debe interpretar y si es necesario reescribir el mensaje antes de reenviarlo.

Gateway: Es un servidor que actúa como intermediario para otros servidores, a diferencia del *proxy* el *gateway* recibe peticiones como si fuera el servidor original del recurso solicitado, por lo que el cliente puede no estar seguro si esta conectado con un *gateway*. Normalmente son utilizados como traductores de protocolos para acceder a recursos en sistemas que no siguen *HTTP*.

Túnel: A diferencia del *proxy* y el *gateway*, el túnel no realiza operaciones con las solicitudes y respuestas de *HTTP*, simplemente es un punto de retransmisión entre dos conexiones TCP. Los mensajes son transmitidos sin sufrir ningún cambio como si existiera una única conexión TCP entre el usuario y el servidor origen. Los túneles son utilizados cuando existen sistemas intermediarios entre el usuario y el servidor, pero no es necesario para este sistema comprender el contenido de los mensajes.

HTTP está basado en el principio pregunta – respuesta. El uso más común es entre un cliente que ejecuta un programa de aplicación llamado *browser* o navegador y establece una conexión con un servidor Web. Para esto *HTTP* hace uso de una conexión TCP por seguridad, sin embargo cada transacción es tratada en forma independiente y una vez que esta se completa, se cierra. Por lo que se puede resumir una transacción *HTTP* en cuatro pasos:

1. El *browser* o navegador abre una conexión.
2. El *browser* envía una demanda al servidor.
3. El servidor envía una contestación al *browser*.
4. La conexión está cerrada.

El puerto predeterminado para este tipo de conexión es el puerto TCP 80, pero siempre existe la posibilidad de usar algún otro puerto, sin embargo *HTTP* puede ejecutarse sobre otro tipo de protocolo de transporte en Internet o en cualquier otro tipo de redes, pero este protocolo debe proporcionar la característica de un transporte fiable.

En términos simples se puede decir que *HTTP* es un protocolo sin estados, es decir que no guarda ninguna huella de conexión. Por ejemplo para abrir una página que contiene dos gráficos, el *browser* abrirá tres conexiones TCP, una para cada gráfico y otra para la página, manejando todas estas conexiones en forma simultánea.

Otra característica importante es que es flexible en cuanto a los formatos que puede tratar, por ejemplo cuando un cliente emite una solicitud con una lista de prioridades de formatos con los que puede tratar a un servidor, este responde con el formato adecuado.

En la mayoría de los casos la comunicación *HTTP* la empieza el cliente o usuario, que solicita un recurso al servidor origen. El caso más simple es cuando se establece una única conexión entre el usuario y el servidor origen, tal como se muestra en la siguiente figura:

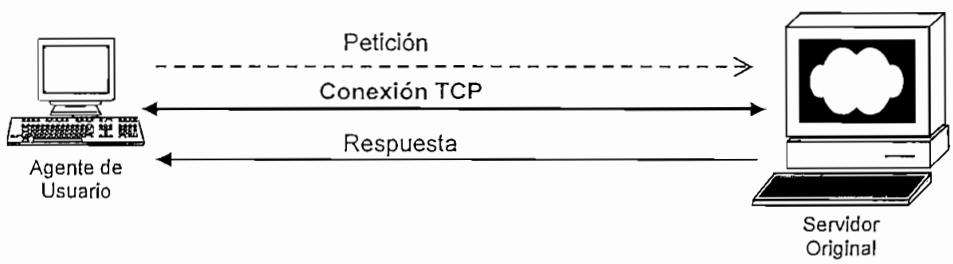


Figura 1.13 Conexión única cliente – servidor

En algunos casos no existe una conexión directa entre el usuario y el servidor origen. Existen uno o más intermediarios entre el usuario y el servidor origen tales como: *proxy, gateway* o túnel. Las peticiones y las respuestas son evaluadas por los intermediarios y enviadas al siguiente destino intermediario y así sucesivamente hasta alcanzar su destino final en la cadena pregunta – respuesta tal como se muestra en la siguiente figura:

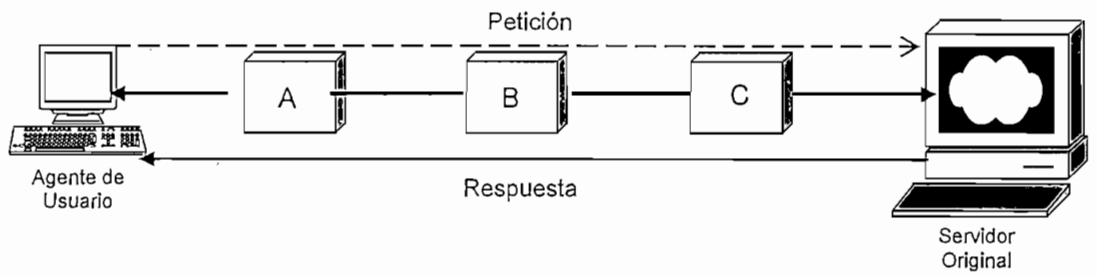


Figura 1.14 Conexión cliente – servidor con intermediarios

Cuando una petición proviene de un *proxy*, este vuelve a escribir todo o parte del mensaje y lo reenvía al siguiente destino. Un *gateway* recibe el mensaje y lo envía a los protocolos subyacentes con un formato adecuado. Puesto que un túnel no puede entender en contenido del mensaje se limita solamente a ser un punto de retransmisión entre dos conexiones.

En la actualidad existen tres versiones del protocolo *HTTP*: HTTP 0.9, HTTP 1.0 y HTTP 1.1. HTTP 0.9 fue la primera versión que apareció en el año 1991, en la actualidad ya no se la usa. HTTP 1.0 es el protocolo más utilizado hoy en día aunque presenta cierto tipo de inconvenientes, uno de ellos, quizá el más grave es que debe abrir una conexión por cada recurso que debe transferir inclusive cuando todos los recursos se encuentran en el mismo servidor remoto. Esto representa un alto uso de recursos por lo que la eficiencia es baja cuando se trata de presentar una página Web completa. Tiene especial incidencia cuando se trata de tráfico en Internet bajo una alta demanda.

La versión HTTP 1.1, es compatible con las anteriores, pero con muchas más opciones, cabeceras, características, etc., permite una mayor velocidad para presentar la página Web al usuario y reduce el tráfico existente en la Web,

aunque no se lo utiliza ampliamente, se espera la implementación y uso masivo de esta versión.

HTTP 1.1 provee una conexión persistente que permite realizar múltiples pedidos a un servidor, los cuales son almacenados y colocados en fila de espera en el *buffer* de salida del servidor. La capa TCP también puede poner múltiples pedidos y respuestas a pedidos dentro de un segmento TCP que se envía a la capa IP para que sea transmitido en forma de datagrama. Gracias a que el número de pedidos disminuye, el número de paquetes que fluye a través de Internet disminuye por lo que el tráfico se reduce y mejora el desempeño.

Esta versión también permite realizar la compresión de archivos HTML por lo que el volumen de información disminuye en forma sustancial al igual que la cantidad de datos que son transmitidos por Internet, además de esta ventaja, quizá la más importante es que le permite a un mismo servidor tener varios nombres de dominio compartiendo la misma dirección IP, lo que simplifica el procesamiento de servidores Web que administran simultáneamente varios sitios Web que son comúnmente conocidos como *virtual hosting*.

1.7.7.5 HTML Lenguaje de Marcaje de Hipertexto.

HTML (*HyperText Markup Language*) es un lenguaje de marcación, que sirve para describir la manera en que debe formarse un documento, una página Web está escrita en este lenguaje. "*HTML* permite a un creador de páginas Web:

- Publicar documentos con títulos, texto, tablas, listas, fotos, etc.
- Recuperar información a través de enlaces de hipertexto, haciendo *clic* sobre él.
- Diseñar formularios para ser utilizados en transacciones de servicios remotos para uso en búsquedas de información, hacer reservaciones, ordenar productos, etc. Incluir hojas de cálculo, vídeo clips, audio y otras publicaciones directamente en sus documentos."[2]

[2] Barreto Alexis, "ESTUDIO Y ANÁLISIS DE LAS DISTINTAS TECNOLOGÍAS DE ACCESO QUE UN PROVEEDOR DE SERVICIOS DE INTERNET PUEDE IMPLEMENTAR EN ECUADOR", Tesis previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional, Quito – Ecuador, Diciembre de 1999.

Este lenguaje es una de las mayores atracciones de la Web, tiene una arquitectura de etiquetas que debe ser entendido por todos los *browsers* o navegadores Web y los servidores Web, aunque esto dependerá de la versión del lenguaje HTML utilizado, ya que las nuevas versiones no entienden algunos comandos de las versiones anteriores.

Estas etiquetas son dispositivos independientes y describen elementos básicos de un documento Web como títulos, párrafos, textos llanos y listas. Tiene también etiquetas más sofisticadas para crear elementos interactivos. Puesto que *HTML* apoya el hipertexto, esto permite que se creen enlaces a otros documentos *HTML*, estos documentos pueden estar en la misma máquina como el documento original o pueden estar en cualquier otra máquina perteneciente a la misma o a otra red en cualquier parte del mundo

"*HTML* ha sido diseñado con la visión de que todo tipo de dispositivo sea capaz de utilizar información de la Web: PC's con monitores de distinta resolución y cantidad de colores, teléfonos celulares, dispositivos de mano (*hand – held*), computadoras con gran y pequeño ancho de banda, etc."[2]

Al igual que el *HTTP* el *HTML* ha estado sometido a constantes cambios, cuando el *Mosaic* era el único visualizador, el lenguaje que interpretaba, el *HTML* 1.0 era el estándar de facto.

El *HTML* 1.0 básicamente era de un solo sentido, los usuarios podían llamar las páginas de los proveedores de información, pero es difícil enviar información en el otro sentido. A medida que las organizaciones comerciales empezaron a usar la Web la demanda de tráfico en dos vías creció también. Por ejemplo, muchas empresas querían que sus clientes llenaran solicitudes de pedidos a través de la Web y de esta manera recolectar los datos necesarios al llenar un formulario, esto

---

[2] Barreto, Alexis, "ESTUDIO Y ANÁLISIS DE LAS DISTINTAS TECNOLOGÍAS DE ACCESO QUE UN PROVEEDOR DE SERVICIOS DE INTERNET PUEDE IMPLEMENTAR EN ECUADOR", Tesis previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional, Quito – Ecuador, Diciembre de 1999.

no era posible con esta versión de *HTML* por lo que surge una nueva versión que incluye formas.

Cuando aparecieron nuevos visualizadores hubo la necesidad de crear un estándar Internet, así aparece *HTML* 2.0 que incluye formas que contiene marcos o botones que permiten a los usuarios proporcionar información o tomar decisiones, luego devuelven la información al dueño de la página. Las formas más comunes son campos en blanco para aceptar texto del usuario, mapas activos y botones.

El *HTML* 3.0 nace como producto de una investigación que tiene como finalidad agregar nuevas opciones y facilidades a la versión 2.0, esta nueva versión incluye tablas, barra de herramientas, fórmulas matemáticas, hojas de estilos avanzados y algunas otras cosas.

CAPÍTULO 2

DISEÑO Y CONSTRUCCIÓN DEL PROTOTIPO

2.1 INTRODUCCIÓN

Un navegador Web como por ejemplo *Microsoft Internet Explorer* es una aplicación del cliente que utiliza *HTTP* para pedir páginas Web ya sea a un servidor en Internet o en una red local.

Los servidores no necesariamente tienen que ser PC's o algún otro tipo de computadora grande, ya que un sistema embebido pequeño con una cantidad de memoria limitada puede ser un servidor de páginas que contengan texto con imágenes simples o que desplieguen páginas con datos en tiempo real y que permitan interactuar al usuario.

El navegador le proporciona al usuario un interfaz que le permite acceder y desplegar las páginas que necesite. Las páginas creadas para sistemas embebidos pueden mostrar contenidos dinámicos que cambian cada vez que la página sea requerida, como ocurre en las páginas que muestran el número de visitantes que han recuperado la página, información de fecha y hora, etc. Algunas páginas pueden ser habilitadas para proporcionar datos al servidor, los mismos que una vez que han sido procesados quizá puedan ser devueltos a la página como resultados, es el caso de los traductores en línea que están disponibles en Internet.

Normalmente los computadores que solicitan páginas Web, muestran la información en pantalla completa, pero para algunas aplicaciones un sistema embebido que también es un cliente HTTP puede presentar una capacidad limitada al momento de desplegar información. Si las páginas solicitadas son muy simples, contienen solo texto puede ser suficiente con tan solo mostrar y desplegar unas cuantas líneas. Un sistema embebido podría recibir y procesar la información de una página Web que contenga grandes volúmenes de información sin desplegar la página en un navegador en lo absoluto.

Un servidor Web con conexión a Internet puede responder a cualquier solicitud desde cualquier navegador, también puede ser programado para responder a las

demandas de ciertas direcciones IP específicas. Si el servidor se encuentra en una red local, pueden acceder a la información que en él se encuentre disponible algunas de las máquinas o todas las que pertenecen a la red. Un sistema embebido funciona como un servidor Web y normalmente consta de:

- Memoria no volátil para almacenar las páginas que caracterizan un servidor.
- Soporte para TCP/IP. Requerimiento para páginas Web y para páginas que viajan como respuesta en una parte de los datos del segmento TCP.
- Soporte para HTTP. El servidor debe entender y responder las solicitudes recibidas por las páginas Web. El estándar HTTP especifica los formatos para las demandas y respuestas.
- Una red local con conexión a Internet. Para publicar páginas Web en Internet el servidor debe tener una conexión a Internet.

Todo esto que se ha mencionado anteriormente se conoce con el nombre de sistema embebido. Pero ¿qué es un sistema embebido? Un sistema embebido es aquel que tiene una computadora inteligente que se dedica a realizar tareas simples o un grupo de tareas que están relacionadas. Se los llama sistemas embebidos porque el código del programa es parte integral o esta incluido dentro del dispositivo.

Por muchos años los sistemas embebidos y las redes Ethernet han existido como dos palabras separadas. Ethernet estaba limitada solamente para computadoras de escritorio y otro tipo de computadoras grandes. Los sistemas embebidos que necesitaban intercambiar información con otros computadores estaban limitados por interfaces de baja velocidad, rango limitado o falta de protocolos de aplicación estándar.

Pero los desarrollos de la tecnología y el mercado hacen posible ahora que los sistemas embebidos puedan comunicarse con una Ethernet local que conecta una red de computadoras así como en el Internet. Un sistema embebido puede hacer que una red de computadoras sea más fácil de monitorear y controlar.

2.2 ELEMENTOS DE LA RED [12]

Muchas de las redes de computadoras locales siguieron uno de los estándares más populares conocido como Ethernet. Estas redes son de alta capacidad y flexibilidad en cuanto al tipo de equipos que pueden interconectar. Muchos de los productos que se han diseñado para ser usados en redes de computadoras tienen soporte o son construidos para Ethernet.

Todas las redes de computadoras tienen una cosa en común, componentes físicos que las habilitan en la red para intercambiar datos y en todas las redes las computadoras deben acordar como van a compartir la información y cual es el camino que las conecta para asegurarse que la información llegue a su destino final.

Entre los principales componentes de la red tenemos:

- Dos o más computadoras que necesiten comunicarse la una con la otra.
- Un interfaz físico definido. Las redes Ethernet tienen especificados estos estándares.
- Cables o transceivers inalámbricos para conectar a las computadoras.

Las computadoras de la red deben también estar de acuerdo en los siguientes aspectos al compartir la red:

- Reglas para decidir cuando una computadora puede transmitir en la red, en especial cuando múltiples computadoras van a compartir el medio de transmisión.
- La forma de identificar el destino de la transmisión.
- Un formato definido para la información enviada a través de la red para que la computadora que la recibe pueda entenderla y hacer uso de ella.

[12] Axelson Jan, "DESIGNING AND PROGRAMMING SMALL DEVICES FOR NETWORKING", www.dedicatedteacher.com/estore, 2003

2.3 ETHERNET [12]

En una red Ethernet, el interfaz de red es una tarjeta que utiliza un controlador. Este controlador de Ethernet contiene el código del programa que administra la comunicación entre este y el nivel más alto de la pila de protocolos.

Para enviar un datagrama IP sobre una red Ethernet, las capas IP pasan el datagrama hacia el controlador de Ethernet, el dispositivo informa al procesador que hay una trama que contiene un datagrama y está formada por el campo de direccionamiento, el campo de verificación de error de la información.

Cuando se recibe un datagrama IP desde la red, el controlador de Ethernet debe la dirección de destino es correcta o si se trata de de una dirección *multicast* o *broadcast* que debe estar configurada en el controlador para que este pueda aceptar. Si la dirección destino es aceptada, el controlador realiza el chequeo de errores y luego pasa el datagrama o una indicación de error a la capa IP.

Ethernet no es la única forma de conectar una red de computadoras con dispositivos embebidos, pero es la opción más popular. Es posible unir y usar una red Ethernet sin conocer mucho acerca de su funcionamiento interno. Hay hardware y software que fueron construidos para Ethernet y que pueden aportar con mayores detalles, esto y un poco de conocimiento sobre este tipo de red ayudan para la selección de los componentes idóneos para la misma. Esto facilita la escritura del software necesario para el intercambio de datos sobre la red, proporcionando una solución a cualquier problema que se pueda presentar.

Ventajas:

Existen muchas razones por las que Ethernet es popular y es la red más usada para aplicaciones de sistemas embebidos y la comunicación con otros computadores.

---

[12] Axelson Jan, "DESIGNING AND PROGRAMMING SMALL DEVICES FOR NETWORKING", www.dedicatedteacher.com/estore, 2003

- Ethernet es lo suficientemente versátil para satisfacer varios propósitos. Probablemente la aplicación más conocida es la unión de computadoras de escritorio en oficinas, pero este no es el único uso. Ethernet puede transferir cualquier clase de datos desde mensajes cortos hasta archivos muy grandes. Este tipo de red no requiere de una gran computadora con alta velocidad de procesamiento que contenga un controlador de Ethernet, ya que usando un microcontrolador de 8 bits se puede comunicar con una red Ethernet.

- Es fácil de usar, debido a que casi todo el trabajo está hecho. Todas las computadoras en la red siguen las especificaciones de Ethernet para interconexión, administración del tráfico en la red y el intercambio de datos.

- Los componentes para Ethernet son populares, fáciles de encontrar y sobre todo son baratos Muchas computadoras y otros equipos de escritorio son construidos con soporte para Ethernet o a lo sumo requieren una tarjeta de expansión que provea el interfaz necesario. Tanto *Windows* como otros sistemas operativos incluyen en el software el soporte para redes Ethernet.

- Diseños con sistemas embebidos tienen una buena selección de módulos con capacidad Ethernet. Muchos módulos incluyen el CPU mientras que otros contienen únicamente el controlador de red y el interfaz para poder conectar el CPU o se puede seleccionar cada uno como por ejemplo el CPU, el controlador de Ethernet y los otros componentes relacionados.

- Los costos son razonables porque Ethernet y TCP/IP son populares, tanto hardware como software están disponibles a bajo costo e inclusive a veces pueden ser gratuitos. Hardware que soporte tanto Ethernet como TCP/IP existe y es fácil de añadir a las computadoras de todo tipo, incluyendo los desarrollados para sistemas embebidos.

- Ethernet es rápido, soporta velocidades que van desde los 10 Mbps hasta los 10 Gbps. 10 Mbps es adecuado para muchos sistemas embebidos.

- Puede cubrir cortas y largas distancias en función del tipo de cable que se utilice para realizar las conexiones ya sea entre computadoras, repetidores, *hub* o *switch*,

Limitantes:

A pesar de que Ethernet es la respuesta que todos los sistemas embebidos necesitan. Para algunos sistemas existen formas más simples, menos caras o inclusive más apropiadas para conectar estos sistemas.

- Ethernet solo, no garantiza la transferencia en tiempo real o transferencias que puedan ocurrir con el menor retardo posible, en un cierto tiempo o cada cierto intervalo, porque un dispositivo tiene que esperar para transmitir en la red y no sabe cuando exactamente podrá hacerlo. Aunque generalmente las trasmisiones Ethernet tienen un mínimo de retraso a menos que la red esté extremadamente ocupada.

- Ethernet no es muy eficiente cuando se trata de transmitir pequeñas cantidades de datos. Todos los datos en Ethernet viajan en estructuras llamadas tramas. Cada trama debe tener entre 46 y 1500 bytes de datos, junto con los datos cada trama incluye bytes para sincronización, direccionamiento, control de errores y otra información identificada, tal como se muestra en la figura siguiente:

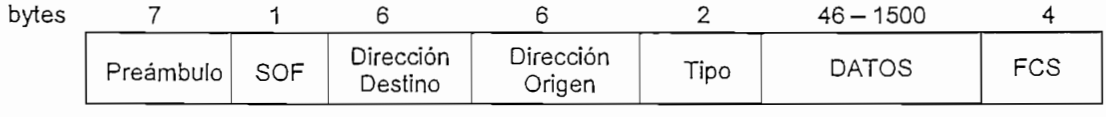


Figura 2.1 Trama Ethernet

2.4 SISTEMAS EMBEBIDOS [10]

Hay un consenso general en cuanto a la idea de que con el pasar de los años existirán cada vez más dispositivos que utilicen sistemas embebidos conectados a Internet, la predicción es tal que se asume que para el año 2010 un 95% de los dispositivos conectados a Internet no serán computadoras, sino este tipo de sistemas.

Existen algunos productos que son ya soluciones completas que están provistas por los dos elementos que son hardware y el código del programa para las comunicaciones de Ethernet e Internet. El hardware típico incluye en su tabla de circuito un CPU, un controlador de Ethernet y los componentes relacionados. El código del programa incluye soporte para Ethernet, TCP/IP y otros protocolos de Internet.

Más allá de los componentes básicos mencionados anteriormente, las opciones varían. Diferentes productos usan diferentes CPU's, el tipo y la cantidad de memoria y los puertos de entrada y salida (I/O) también varían. Un producto puede ser programado utilizando *Assembler, C, Java,* o una combinación de estos lenguajes.

Una de las soluciones más populares es aquella que utiliza un microcontrolador de 8 bits como por ejemplo: Rabbit 2000, AVR o PIC y un controlador Ethernet provisto con una dirección MAC como es el CS8900A o el RTL8201BL manejado por un puerto que trabaja en modo de 8/16 bits. Normalmente para la pila de protocolos TCP/IP se utiliza el lenguaje C como alternativa para el desarrollo del software, ya que todos estos microcontroladores pueden ser programados utilizando este lenguaje, pero al momento de escoger cual es el más adecuado para cierto tipo de aplicación, se debe hacer en base a confiabilidad, funcionalidad, velocidad, etc.

---

[10] Beyond Logic, IP & ETHERNET INTERFACES, http://www.beyondlogic.org/etherip/ip.htm, Marzo 2004

Para evitar los problemas que puedan presentarse al manejar elevados volúmenes de paquetes de *broadcast* que son transmitidos por la red, se puede aplicar como posible solución la utilización de un procesador que sea más grande y veloz, como es el caso de los dispositivos embebidos utilizados bajo Linux tales como los basados en *Coldfire, Dragon Ballo, ARM*, los mismos que son los suficientemente fuertes para trabajar en un ancho de banda conveniente evitando de esta manera ser susceptibles al ataque por parte de personas indeseadas en la administración de la red.

Otra de las soluciones que se presenta para ser implementada bajo la pila de los protocolos TCP/IP es el que usa un procesador *Seiko S-7600A*, el mismo que posee un controlador PPP, lo que significa que puede ser conectado a un MODEM, brindando de esta forma la facilidad de conectarse a las redes Ethernet o navegar por Internet desde su casa en cualquier momento, las 24 horas del día.

Este procesador puede enviar grandes volúmenes de información encapsulados automáticamente en un datagrama TCP o UDP utilizando IP. Este tipo de solución actualmente no se encuentra disponible ya que *Seiko* ha discontinuado lo que es el Internet embebido, sin embargo el concepto aun se mantiene vivo.

Ipsil tomando el concepto anterior, almacena datos en su *IPμ8932* y lo combina con un servidor Web, MAC Ethernet y un controlador TCP/IP en un solo integrado, esto permite que un solo integrado con 20 entradas digitales o analógicas muestren páginas Web sin la necesidad de un microcontrolador. Pero si se necesita implementar algo más complejo se puede añadir un microcontrolador externo compatible.

Lantronix también presenta una solución novedosa al formar una nueva categoría que encapsula conectividad, comunicación y mucho más en la familia de productos *XPORT*, la misma que hace posible un mundo donde con un solo dispositivo se puede acceder a Internet.

Figura 2.2 Servidor Web con sistema embebido, LANTRONIX

El servidor *XPORT* con sistema embebido, es una solución completa que se habilita haciendo uso de un complejo paquete que utiliza un conector RJ-45. Utilizando este producto varios fabricantes podrían comercializar sus redes con todas las capacidades con las que fueron construidas. La demanda por dispositivos de conectividad a Internet aumenta exponencialmente, *Lantronix* por medio de este pequeño dispositivo elimina toda la complejidad que demandaría otro fabricante para incorporar una solución como esta. El software que le permite tanto la conexión a Internet como a Ethernet va dentro de este pequeño sistema embebido.

Este pequeño dispositivo incorpora todas las características esenciales para la conectividad de redes, incluyendo conexión a Ethernet 10BaseT y 100BaseTX, un sistema operativo probado, un servidor Web con sistema embebido, la pila completa de protocolos TCP/IP y 128 bits AES para encriptación de información por seguridad.

Atmel ha desarrollado un servidor Web con sistema embebido el *AVR460*, proyectándose hacia lo que sucederá en el futuro, en donde casas inteligentes necesitarán conectarse a Internet o con otros dispositivos en red, para lo cual es fundamental la utilización de un microcontrolador. El *AVR* incluye un servidor completo con capacidad de soporte para todos los protocolos TCP/IP y con conexión a Ethernet, también incluye apoyo para envío de correo y el software necesario para la configuración del servidor en la red.

El servidor Web *AVR* ha sido diseñado para ser integrado a cualquier equipo digital, se puede conectar a cualquier interfaz Ethernet y se puede comunicar utilizando cualquier tipo de *browser*. Con un sistema como este, una computadora en casa conectada a una red puede controlar todos los dispositivos de la red e inclusive recibir cualquier tipo de requerimiento desde otra computadora en Internet. El servidor Web está identificado por una única dirección IP y puede ser controlado desde cualquier parte del mundo con tan solo una autorización y una orden.

Otra de las soluciones novedosas desarrolladas por Atmel es el *Web Hardwired TCP/IP*, esta es una solución desarrollada con WIZnet para un microcontrolador 80C51, esta solución tiene todo el hardware necesario para el desarrollo de cualquier aplicación, no necesita ningún desarrollo adicional y permite una conexión fácil de cualquier aplicación en Internet.

El Kit de desarrollo Web LAN51H está diseñado para probar conectividad a Internet con un Atmel 80C51 que tiene una memoria Flash y el WIZnet IIM7010A con un único sistema de implementación de algunos de los protocolos TCP/IP que están incluidos dentro de un dispositivo.

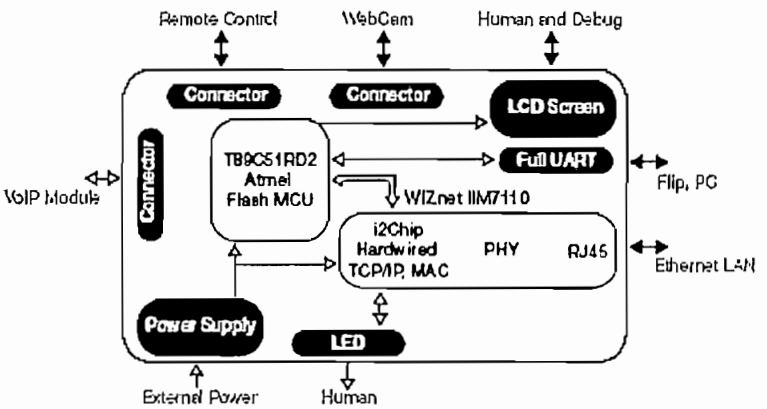


Figura 2.3 Servidor Web LAN51H

El WIZnet IIM7010A es un módulo que incluye el W3100A, éste es el dispositivo que contiene como parte del hardware el software que administra algunos de los protocolos TCP/IP, un controlador Ethernet PHY, este controlador pertenece a la

familia Realtek (RTL8201BL) y de un adaptador de línea con un conector RJ-45 que se identifica con una dirección MAC.

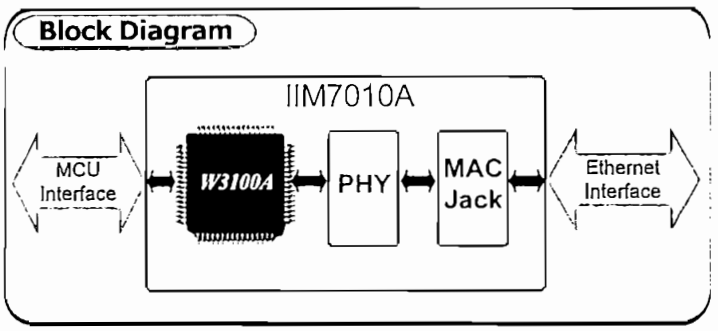


Figura 2.4 Componentes del IIM7010A

Esta es una solución ideal para los usuarios que les interesa desarrollar una aplicación y habilitarla rápidamente en Internet.

El hecho de que todas estas soluciones manejen el conjunto de protocolos TCP/IP, da lugar a que cualquiera de ellas sin importar el tamaño interactué con sistemas operativos diferentes para poder comunicarse entre sí o con otros dispositivos de red.

2.5 COMPONENTES DE LOS SISTEMAS EMBEBIDOS

Cuando se utilizan sistemas embebidos para redes, se deben tomar algunas decisiones acerca de los dispositivos de hardware y el código de programación para controlar este hardware.

En algunos casos se puede unir un controlador Ethernet a un CPU y se escribe el código que soporte la comunicación y los protocolos de Internet usados por los dispositivos, también se puede ahorrar tiempo utilizando un módulo que ya contiene un CPU unido a Ethernet y el apoyo de software necesario para que soporte las comunicaciones o se puede optar por un camino intermedio que sería

el de escoger algunas herramientas de software que están provistas para este tipo de diseños y circuitos.

En la figura siguiente se muestra en diagrama de bloques el diseño aproximado de este prototipo:

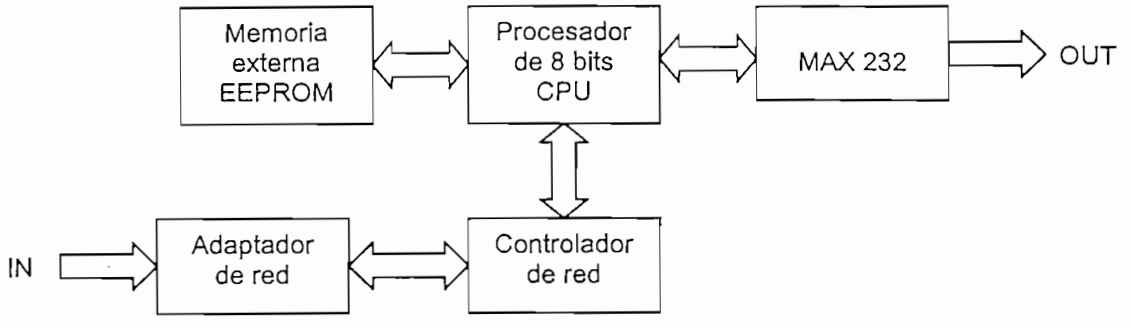


Figura 2.5 Diseño aproximado del Prototipo

2.5.1 CARACTERÍSTICAS DE LOS CONTROLADORES DE ETHERNET

Un sistema embebido que soporta Ethernet requiere un controlador de red y el hardware que le proporcione un interfaz de Ethernet. Muchos controladores de Ethernet están diseñados para el uso en computadoras de escritorio e incluyen soporte para buses estándar de PC y funciones plug and play. Los típicos sistemas embebidos no necesitan de todas las capacidades de los controladores Ethernet para computadoras. Muy pocos de los antiguos controladores de red existentes tienen vida útil en este tipo de sistemas embebidos. Existen controladores diseñados recientemente que son específicos para estos sistemas y que ya se encuentran disponibles.

Las típicas comunicaciones Ethernet son manejadas por una combinación de un controlador Ethernet y un código de configuración del dispositivo que se comunica con el procesador. En la pila de protocolos de red, el controlador Ethernet se comunica con el hardware de interfaz Ethernet y con la capa IP o la capa Aplicación. Muchos sistemas embebidos usan IP con TCP o UDP, pero para

algunas aplicaciones el controlador Ethernet puede comunicarse directamente con la capa Aplicación.

El circuito integrado de la unidad de control maneja muchos de las tareas del envío y recepción de tramas Ethernet. Para el envío de una trama, un controlador realiza los siguientes pasos:

- Recibe el mensaje por el nivel más alto de software y lo envía a su destino.
- Calcula la trama Ethernet y verifica la secuencia.
- Verifica el campo de datos, direccionamiento y otra información en los campos de la trama.
- Transmite la trama cuando la red está inactiva.
- Detecta colisiones, cancela cualquier trama transmitida con colisión y reintenta de acuerdo al protocolo especificado por el estándar IEEE 802.3.
- Provee de una indicación de éxito o falla de la transmisión.

Recibiendo una trama, un controlador realiza lo siguiente:

- Detecta y sincroniza la recepción de nuevas tramas.
- Ignora cualquier trama que sea de tamaño menor al mínimo.
- Ignora cualquier trama que no contenga la dirección de la interfaz o una dirección válida de *multicast* o *broadcast* en el campo de dirección destino.
- Calcula la trama, chequea el valor de la secuencia, compara el resultado con el valor recibido e indica un error si éstos no se igualan.
- Prepara los datos recibidos en la trama y cualquier otra información disponible en la computadora del receptor.
- El más alto nivel del software lee el mensaje y ejecuta la acción que se solicita.

2.5.2 CONTROLADOR BÁSICO ETHERNET.

En una Ethernet que interconecte sistemas embebidos, un CPU administra la comunicación con el controlador Ethernet. El requerimiento mínimo para el CPU es un microcontrolador con un bus externo de datos de 8 bits.
Algunos de los controladores que han sido populares para sistemas embebidos son diseñados para uso o expansión de tarjetas para bus ISA de las primeras computadoras.

Una abreviación del término para un interfaz controlador de red es NIC, este término también puede hacer referencia a una tarjeta de expansión que contiene un interfaz controlador de red.

2.5.2.1 Realtek RTL8019AS [15]

Es un controlador para Ethernet, Full Dúplex y con funciones Plug and Play. Entre sus principales características están:

- 100 pines QFP (Quad Flat Pack).
- Software compatible con RLT8019
- Soporta el modo auto-detectable
- Compatible para Ethernet II y IEEE 802.3 como 10Base5, 10Base2 y 10BaseT.
- Apoyo para Ethernet con función Full Duplex para doble canal de banda ancha.
- Soporte de tres niveles de operación bajo ciertas modalidaes:
  - Descanso.
  - Bajo consumo de potencia con funcionamiento del reloj interno.
  - Bajo consumo de potencia con reloj interno detenido.
- Soporta UTP, AUI y BNC auto detectable.
- Soporte de memoria flash de lectura/escritura.
- Construido con 16Kbytes de SRAM (buffer).
- Fuente de polarización única de +5V dc.

[15] RTL8019AS, Hoja de datos del fabricante, www.connectone.com

- Soporta 8 líneas de interrupción.
- Soporte para 16 líneas de direccionamiento de entrada/salida.
- Requiere una memoria EEPROM 9346 (64 x 16 bits) que almacena los recursos de configuración y parámetros de identificación.
- Capacidad para programar la memoria EEPROM 9346 directamente.
- Soporte para 4 leds de diagnóstico con opción de programación.

Este dispositivo es un controlador Ethernet con una alta integración y una solución que implementa un adaptador Plug and Play compatible con transmisión full dúplex y caracterizado por un bajo consumo de potencia.

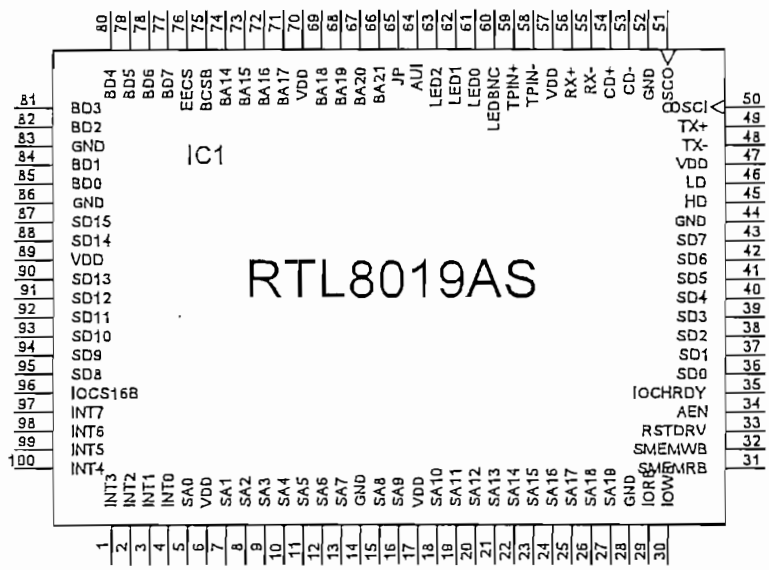


Figura 2.6 Distribución de pines del controlador de red RTL8019AS

La función full dúplex habilita simultáneamente la transmisión y recepción sobre un par trenzado utilizado por Ethernet, está característica no solo incrementa el ancho de banda de 10MHz a 20MHz sino que también evita la acción que degrada a los canales que compiten por el acceso al canal al utilizar el protocolo CSMA/CD.

Para las redes Ethernet el circuito integrado puede conectarse utilizando un terminal RJ-45 aunque también tiene la posibilidad de utilizar un AUI o un BNC.

Los 16Kbytes de memoria SRAM del controlador, contienen paquetes que esperan ser transmitidos a la red y paquetes que son recibidos desde la red. Cuando se usan 8 bits de datos, solamente 8Kbytes del buffer están disponibles para estas acciones.

A continuación se muestra una conexión típica utilizando el controlador de red RTL8019AS para una red Ethernet.

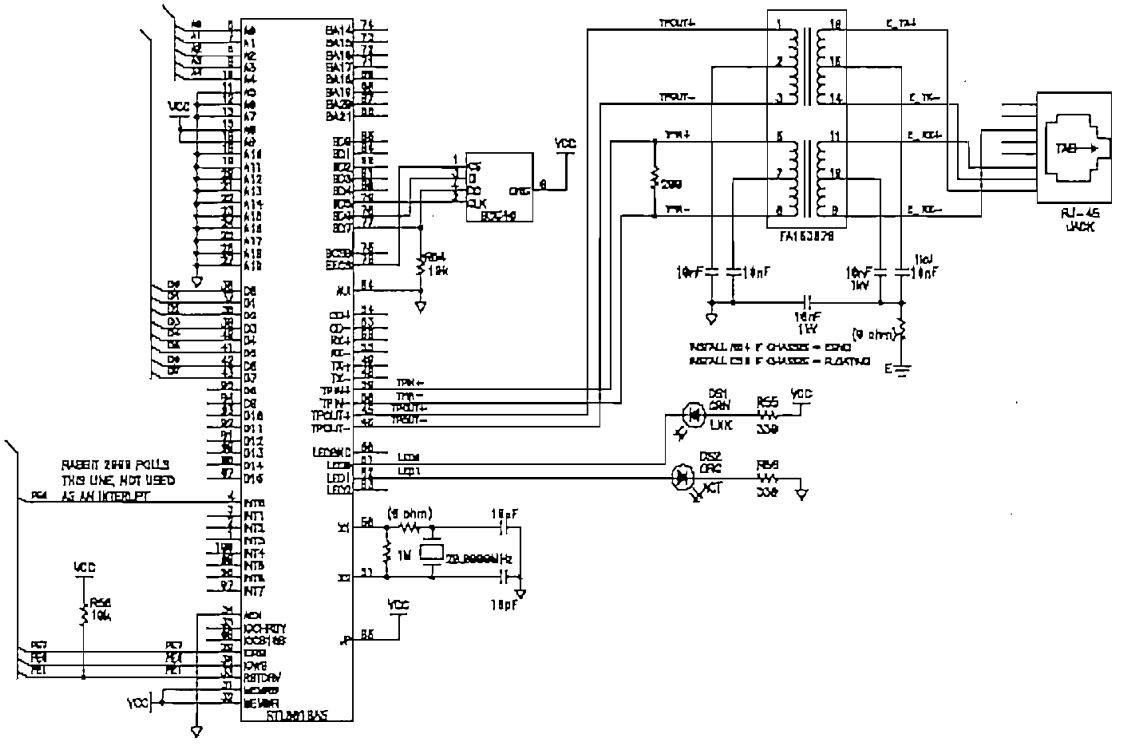


Figura 2.7 Aplicación Típica Ethernet usando un RTL8019AS con interfaz de 10Mbps. [12].

2.5.2.2 Cirrus Logic CS8900A [17]

Es un viejo controlador basado en ISA que es conveniente para algunos sistemas embebidos, similar al controlador Realtek, posee las siguientes características:

- Máxima corriente de consumo 55mA con 5V.
- Opera con una fuente de alimentación de 3V o de 5V

[12] Axelson Jan, "DESIGNING AND PROGRAMMING SMALL DEVICES FOR NETWORKING", www.dedicatedteacher.com/estore, 2003

[17] CS8900A, Hoja de datos del fabricante, www.embeddedethernet.com

- Operación full duplex.
- Puerto con filtros análogos para redes 10Base-T.
- Puerto AUI para redes 10Base2, 10Base5 y 10Base-F.
- Características programables para recepción.
- Características programables para transmisión.
- Led indicador del estado de la actividad de la red LAN.
- Modo de parada/espera.

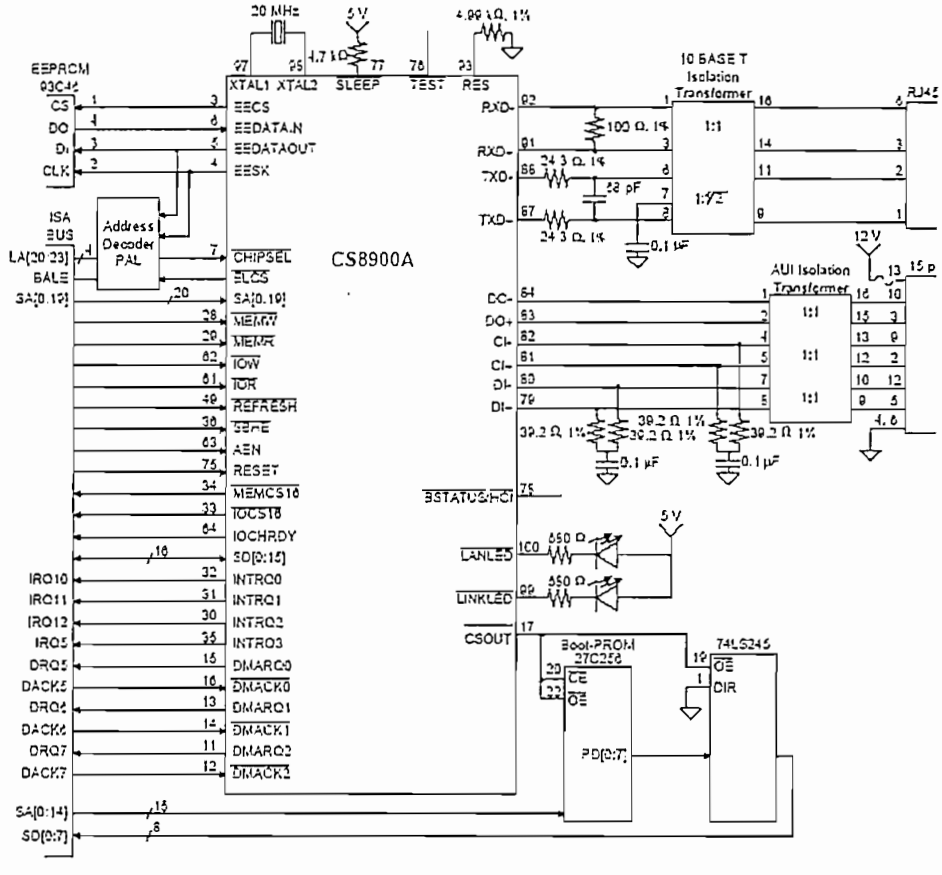


Figura 2.8 Aplicación Típica Ethernet usando un CS8900 con interfaz de 10Mbps. [12]

Se puede conectar por medio de un RJ-45, tiene un puerto AUI que habilita la conexión de otros 10Mbps. Hay versiones que necesitan 5V y 3V de suministro

[12] Axelson Jan, "DESIGNING AND PROGRAMMING SMALL DEVICES FOR NETWORKING", www.dedicatedteacher.com/estore, 2003

de energía. Es un integrado de 100 pines, tiene una interfaz serial para manejar una EEPROM que almacena las direcciones y otros datos de configuración.
Tiene una memoria de 4 Kbytes, los primeros 350 bytes contienen los contenidos de los registros para la configuración del interfaz del bus, proporciona el estado y la información de control, inicializa la transmisión y filtra las direcciones. El resto de la memoria almacena las tramas recibidas y espera por la tramas para transmitir. La cantidad de memoria asignada para cada dirección puede variar dependiendo del tráfico.

Este controlador tiene 20 líneas de direccionamiento, aunque soporta 8 y 16 bits de datos, tiene severas limitaciones cuando opera con 8 bits de datos, ya que con este tipo de configuración el controlador no apoya las interrupciones. El CPU debe revisar el integrado para detectar cuando una trama que ha sido recibida está disponible, cuando una trama ha terminado de ser transmitida o cuando ha ocurrido un error.

2.5.2.3 Realtek RTL8201BL [23]

Este un controlador de red que tiene la capacidad de interactuar directamente con un dispositivo de capa MAC.

Este controlador se caracteriza por los siguientes rasgos:

- Soporta una operación de 10/100 Mbps
- Soporta operación half y full dúplex
- Puede interactuar con la red por medio de un cable trenzado o por fibra óptica cumpliendo las normas IEEE 802.3 e IEEE 802.3u
- Velocidad par la transmisión full dúplex puede ser negociable
- Funcionamiento con 3.3 V con una señal de entrada/salida de 5V.
- Cristal de oscilación de 25MHz
- Habilitado control de flujo
- Encapsulado LQFP de 48 pines

[23] RTL8201BL, Hoja de datos del fabricante, www.connectone.com

Tiene la posibilidad de ser utilizado como un interfaz adaptador de red, MAU, Hub Ethernet, Switch Ethernet, adicionalmente puede ser usado en cualquier sistema embebido que necesite conectarse a la red a través de un par trenzado con conexión 10/100Base–T o de fibra óptica con una conexión del tipo 100base – FX.

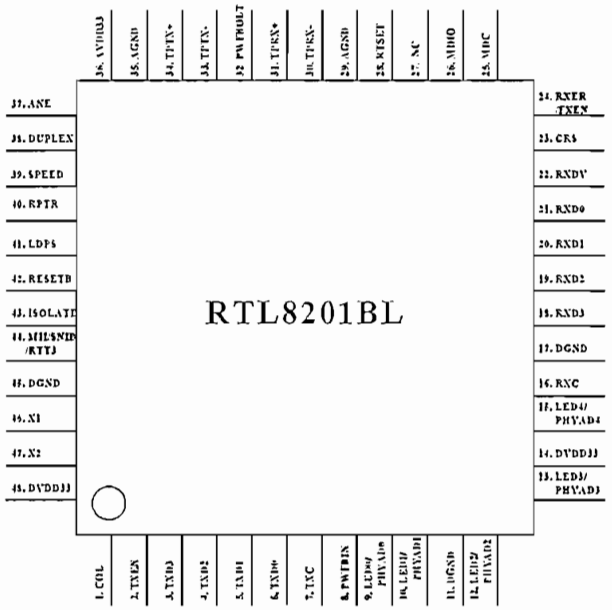


Figura 2.9 Distribución de pines del controlador RTL8201BL

2.5.3 PROCESADOR.

El procesador es el circuito integrado encargado de interpretar y ejecutar las instrucciones; es decir, es un microprocesador fabricado en un chip que contiene millones de componentes electrónicos.

El microprocesador está formado por una unidad aritmético – lógica que realiza cálculos, comparaciones y toma decisiones lógicas, para esto utiliza una serie de registros en donde almacena la información temporalmente y una unidad de control que es la encargada de interpretar y ejecutar las instrucciones.

Cuando se ejecuta un programa, el registro de la CPU llamado contador del programa lleva la cuenta de la dirección de la siguiente instrucción *para garantizar*

que estas se ejecuten en la secuencia adecuada. La unidad de control de la CPU coordina y temporiza las funciones de la misma, tras lo cual recupera la siguiente instrucción de la memoria. En una secuencia típica, la CPU localiza la instrucción en el dispositivo de almacenamiento correspondiente.

Existen varios tipos de procesadores que se pueden utilizar para los sistemas embebidos y los distintos fabricantes ofrecen algunos tipos de soluciones con diferentes características, las mismas que son beneficiosas al momento de tomar una decisión en función de la aplicación que se va a implementar.

2.5.3.1. Procesador Atmel AT89S8252 [18]

Este es un procesador compatible con Intel x51, y tiene las siguientes características:

- CPU de 8 bits
- Internamente contiene un procesador booleano; es decir, puede trabajar con estructura de datos a nivel de bit normalmente utilizado en aplicaciones de control.
- Posee 4 pórticos de 8 bits bidireccionales.
- 8Kbytes de memoria FLASH reprogramable con duración de 1000 ciclos lectura/escritura.
- 2 Kbytes de memoria EEPROM, con duración de 1000 ciclos de lectura/escritura.
- Rango de operación de 4 a 6 V.
- RAM interna 256 x 8 bits.
- 32 líneas programables de entrada y salida.
- Tres temporizadores/contadores de 16bits.
- Posee además dos niveles de interrupciones
- Puerto serial full duplex.

[18] AT89S8282, Hoja de datos de fabricante, www.atmel.com/atmel/acrobat/doc0401.pdf

El AT89S8252 es un microordenador de 8 bits con 8Kbytes de memoria flash programable que puede ser borrada, con alto rendimiento y 2Kbytes de memoria EEPROM. Este dispositivo usa una memoria con tecnología de de alta densidad de integración no volátil y es compatible con la familia 80C51. El circuito integrado permite programar nuevamente la memoria del sistema a través de un SPI (Serial Port Interface) interfaz serial o utilizando un programador de memoria no volátil convencional.

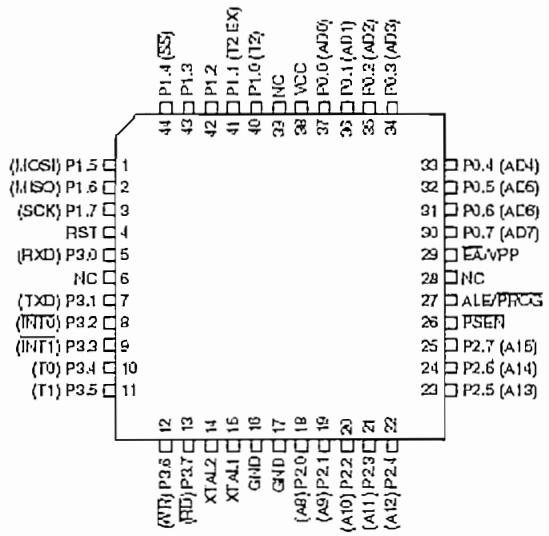


Figura 2.10 Distribución de pines del procesador Atmel AT89S8252

En el modo de bajo consumo de energía deja de trabajar el CPU mientras permite que la memoria RAM, temporizadores/contadores, el puerto serial y las interrupciones del sistema continuen trabajando. Guarda el contenido de la memoria RAM pero detiene el oscilador desactivando de esta manera todas las otras funciones de los otros integrados hasta que la próxima interrupción de hardware reestablezca el sistema.

2.5.3.2. Procesador Rabbit RCM3200 [19]

El RCM3200 es un módulo microprocesador ideal para diseños en los que se desea rapidez en el desarrollo e implementación de sistemas embebidos con

[19] RCM3200, Hoja de datos del fabricante, www.keentech.com.cn/product/Rabbit/documents/rcm3200.pdf

integración completa de conectividad con Ethernet 10/100Base-T. Entre sus principales características se tiene:

- Un microprocesador Rabbit 3000 que corre a 44.2 MHz.
- Puerto Ethernet 10/100Base – T con conector RJ-45.
- Memoria FLASH de 512K.
- Memoria SRAM de 512K programable
- Memoria SRAM con 256K para datos.
- 3.3 V para funcionamiento con 255 mA.
- Encapsulado de 52 pines digital.
- 6 puertos seriales.

Tiene habilitado un interfaz esclavo con un puerto para realizar conexiones maestro – esclavo por medio del cual se puede unir a otro procesador que este basado en el mismo sistema junto con un bus I/O que se puede configurar para 8 líneas de datos y 6 líneas de direccionamiento. El puerto 10 / 100 Base–T para Ethernet permite una comunicación a tiempo real con la Web.

2.5.3.3. Procesador Atmel AT89C51RD2. [20]

Es un procesador compatible con Intel x51 y x52 de 8 bits. Entre sus principales características están:

- 6 puertos de I/O de 8 bits (para las versiones de 64 y 68 pines).
- 4 puertos de I/O de 8 bits (para la versión de 44 pines).
- Tres temporizadores/contadores de 16 bits.
- En el modo estandar
  40 MHz (Vcc 2.7V a 5.5V)
  60 MHz (Vcc 4.5V a 5.5V)
- En el modo X2
  20 MHz (Vcc 2.7V a 5.5V)
  30 MHz (Vcc 4.5V a 5.5V)

[20] AT89C51RD2, Hoja de datos del fabricante, www.atmel.com/dyn/general/tech_doc.asp?doc_id=8970

- 64Kbytes de memoria FLASH programable y de datos con 100K de ciclos de escritura.
- 1Kb de memoria RAM expandida.
- Doble puntero de datos.
- Interfaz maestro – esclavo.
- Puerto de Reset asincrónico.
- Rango de suministro de energía entre 2.7V a 5.5V.

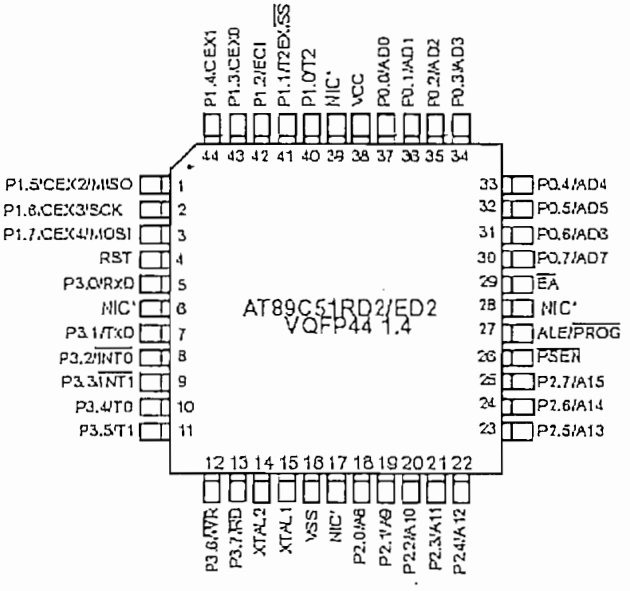


Figura 2.11 Distribución de pines del procesador Atmel AT89C51RD2

Los 64 Kbytes de memoria FLASH son utilizados tanto para el código fuente como para los parámetros y puede ser programada en modo paralelo o en modo serial con el puerto *ISP (Interface Serial Port)* o mediante software, el voltaje de programación es generado en forma interna por medio de un pin y es el valor de Vcc normal.

El diseño del AT89C51RD2 permite reducir el consumo de potencia del sistema disminuyendo la frecuencia de reloj a cualquier valor menor, incluso DC sin que por esto ocurra pérdida alguna de datos.

Todas estas características y otras adicionales hacen que este *procesador* sea más poderoso y por tanto muy útil para aplicaciones en las que se necesitan un

modulador por ancho de pulso (PWM), alta velocidad en los puertos de entrada y salida, lectores de tarjetas inteligentes, etc.

2.5.3.4. Procesador Cygnal C8051F005 [21]

Las principales características de este procesador son:

- Procesador de 12 bits.
- Dos convertidores digitales analógicos de 12 bits.
- Dos comparadores analógicos.
- Programable a una velocidad superior a los 100Kbps con una frecuencia de 25MHz aproximadamente.
- 2304 bytes de memoria RAM interna para datos.
- 32 Kbytes de memoria FLASH programable distribuidos en sectores de 512 bytes.
- 4 puertos de entrada salida (I/O).
- 4 contadores/temporizadores de 16bits para propósitos generales en serie con 5 módulos capturador/comparador.
- Oscilador interno programable de 2 a 16 MHz.
- Cristal como oscilador externo.
- Opera con una fuente de polarización entre 2.7V y 3.6V.
- Corriente de operación típica de 10mA a 20MHz.
- Encapsulado de 48 pines en el modo TQFP.

Este dispositivo posee algunas características importantes, lo que le permite mejorar su actuación global y facilita el uso de aplicaciones externas. Un claro ejemplo de lo mencionado anteriormente es el manejo de interrupciones extendidas que le proporciona 21 fuentes de interrupciones permitiendo a numerosos periféricos analógicos y digitales interrumpir al controlador, el manejo de interrupciones implica menos trabajo por parte del procesador haciéndolo de esta manera más rápido y eficaz.

[21] C8051F005, Hoja de datos del fabricante, www.circuitcellar.com/library/print/0902/brady

Las fuentes de interrupción externas son muy útiles cuando se desarrollan o se construyen sistemas multitarea en tiempo real.

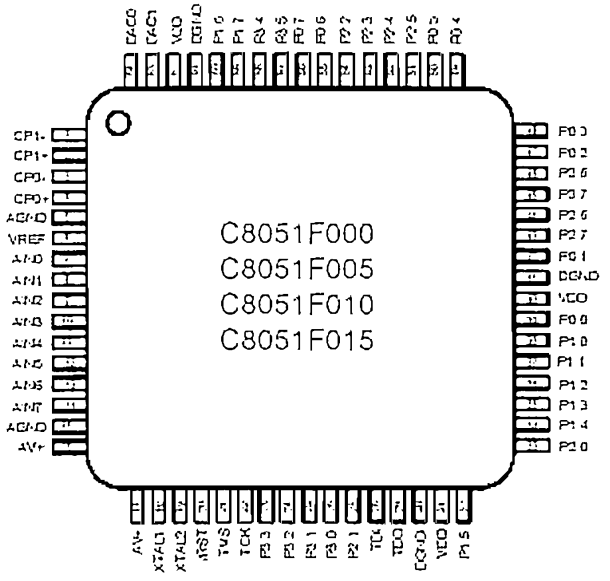


Figura 2.12 Distribución de pines del procesador Cygnal C8051F005

2.5.3.5. Procesador Philips P89C51RD2 [16]

Las principales características de este dispositivo son:

- CPU 80C51.
- Memoria FLASH de 64Kb programable con capacidad ISP (In – System Programming) y IAP (In – Application Programming).
- Permite usar la memoria ROM con rutinas programadas a bajo nivel en la memoria FLASH para transmitir vía UART.
- 6 relojes para el ciclo de operación por máquina.
- 12 relojes para el ciclo de operación por máquina (opcionales).
- Velocidad de 20 MHz con 6 relojes para el ciclo de operación (desempeño equivalente a 40 MHz). Velocidad de 33 MHz con 12 relojes para el ciclo de máquina.
- Memoria RAM expansible externamente a 64Kb.
- 4 interrupciones de prioridad nivelada.

[16] P89C51RD2, Hoja de datos del fabricante, www.semiconductors.philips.com

- 7 fuentes de interrupción.
- 4 puertos de entrada/salida (I/O) de 8 bits.
- Puerto serial mejorado UART full duplex que permite:
  - Detección de error de trama.
  - Reconocimiento automático de direcciones
- Control del modo de potencia que permite:
  - Detener y activar el reloj
  - Modo de espera
  - Modo de bajo consumo de potencia
- Salida para reloj programable.
- Registro DPTR secundario.
- Puerto para inicialización asincrónico
- Bajo Interferencia electromagnética.
- Conjunto de contadores programables que permiten:
  - Modulación por ancho de pulso (PWM)
  - Capturar y comparar datos.

Este dispositivo es un microcontrolador de 8 bits que contiene una memoria FLASH no volátil de 64 Kbytes programable, permite trabajar en forma paralela la programación en serie del sistema y la aplicación. La facilidad ISP (In System Programming) que presenta este módulo permite que el usuario pueda transmitir un nuevo código mientras el controlador está situado en el programa de la aplicación.

Admite borrar y reprogramar la memoria FLASH utilizando subrutinas que están contenidas en la memoria ROM utilizando el puerto serial, evitando el uso del programador de memorias.

Este dispositivo ejecuta un ciclo de máquina en 6 ciclos de reloj, por tanto proporciona dos veces la velocidad normal de un 8051 convencional.

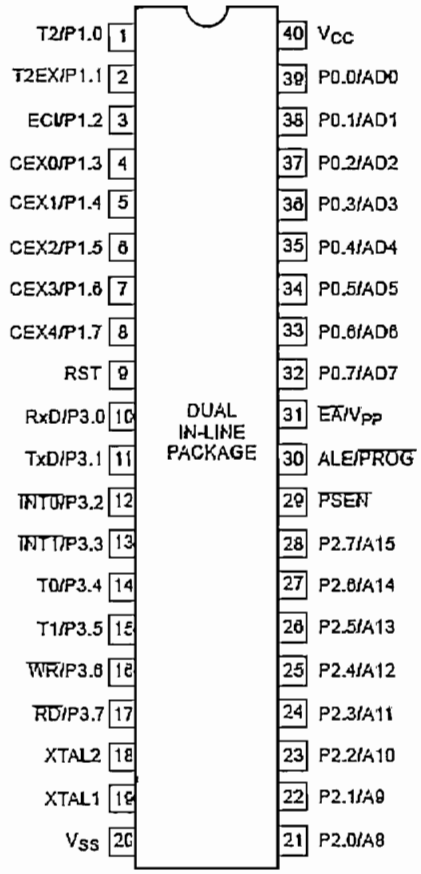


Figura 2.13 Distribución de pines del procesador Philips P89C51RD2

2.5.4 MAX 232. [25]

El Max232 dispone internamente de 4 conversores de niveles TTL al bus estándar RS232 y viceversa (2 para transmisión y 2 para recepción); es decir, cambia los niveles de voltaje de TTL a los del estándar RS232 cuando se hace una transmisión y cambia los niveles de RS232 a los niveles TTL cuando se hace una recepción, necesarios para la comunicación serial, con lo que se puede manejar 4 señales del puerto serial del computador, por lo general las más utilizadas son: TX, RX, RTS, CTS. Para su correcta operación se utiliza 4 capacitores externos, que van conectados como se muestra en la figura siguiente:

Los voltajes de funcionamiento son los que se indican a continuación:

- Rango del voltaje de entrada Vcc -0.3V a 6V.

[24] MAX232, Hoja de datos del fabricante, http://pdfserv.maxim-ic.com/arpdf/MAX220-MAX249.pdf

- Voltaje positivo de salida Vs+ 0.3V a 15V.
- Voltaje negativo de salida Vs- va desde -0.3V a -15V.

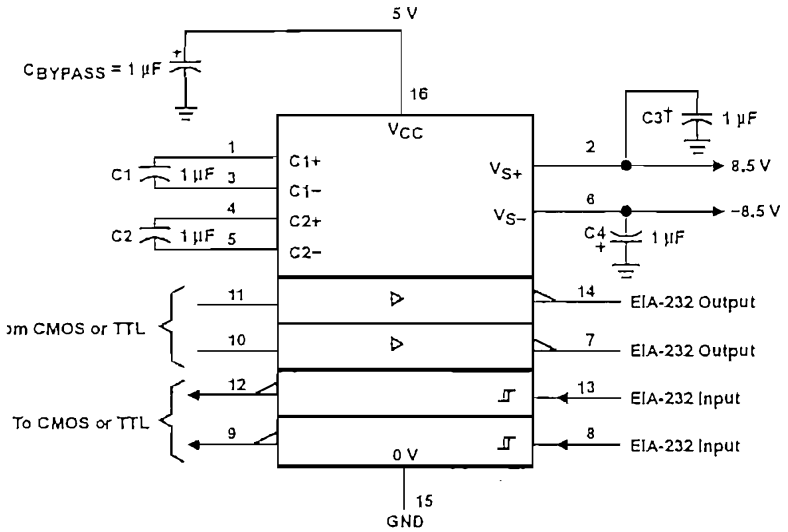


Figura 2.14 Circuito básico con el MAX232

2.5.5 ADAPTADOR DE LINEA LU1T516 [22]

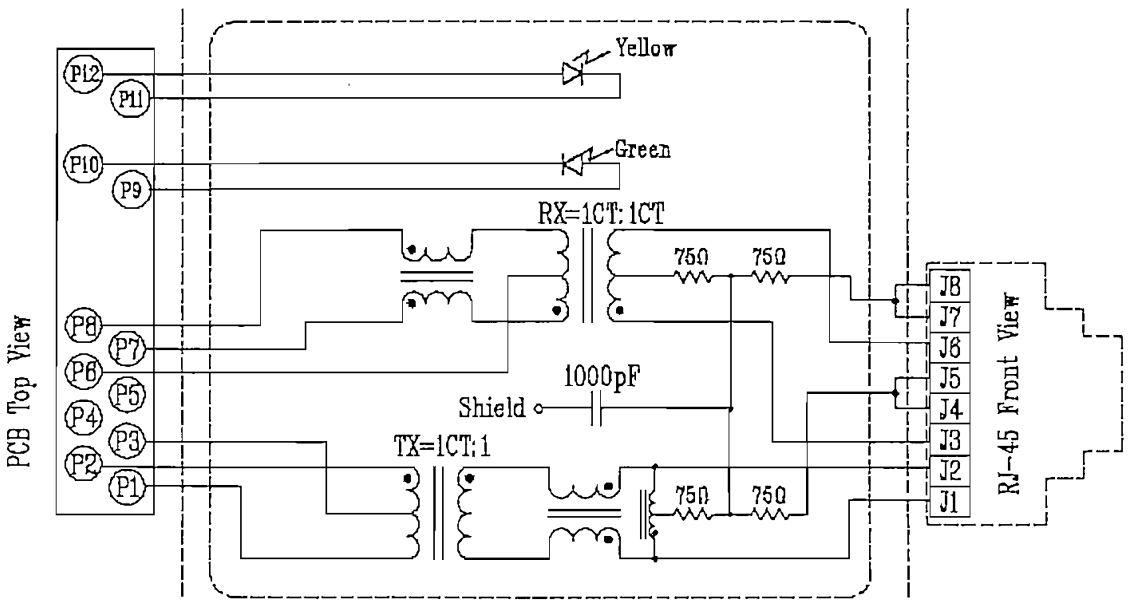


Figura 2.15 Esquemático del adaptador de línea LU1T516

Especificaciones generales:

[22] LU1T516, Hoja de datos del fabricante, www.siteplayer.com/docs

- Conector RJ – 45 integrado con filtro.
- Tamaño igual al RJ – 45 para ahorrar espacio en la tarjeta.
- Reduce la radiación electromagnética, mejora el funcionamiento de EMI (Interferencia electromagnética).
- Diseñado para transmisión 10/100Base – T sobre cable UTP categoría 5.
- Cumple con los estándares IEEE 802.3
- Rango de funcionamiento de temperatura va desde 0°C a 70°C.
- Rango de temperatura de almacenamiento: -25°C a +125°C.

Para el caso en que se usen conectores BNC o AUI, este dispositivo debe ser cambiado por el que corresponda. Para el caso del uso de un conector BNC se puede usar un adaptador de línea DP8392 y si se trata de un conector AUI el adaptador de línea puede ser el Bel9529.

2.5.6 CIRCUITO PARA RESET [28]

Para realizar el reset de todos los dispositivos de la tarjeta en el caso de que fuera necesario cuando exista algún problema de funcionamiento se puede implementar un circuito que consiste en un inversor schmitt trigger 74HC14 y un switch con un circuito RC para el envío de señales al inversor.

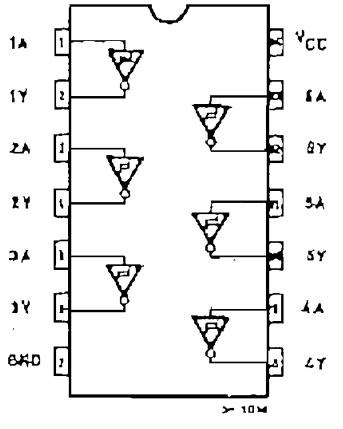


Figura 2.16 74HC14

[28] 74HC14, Hoja de datos del fabricante, http://www.todopic.com.ar

Cada una de las salidas se puede conectar a un dispositivo diferente, con lo que se garantiza que al momento de efectuar el reset, éste se ejecute simultáneamente en todos los componentes que lo requieran.

2.6 DISEÑO DE HARDWARE

En base a la información presentada anteriormente se procede a realizar el diseño del prototipo de un servidor Web basado en un microcontrolador. Las principales características a ser consideradas son las siguientes:

- Un puerto de comunicaciones serial RS232.
- Un puerto Ethernet para cable UTP con un conector RJ-45.
- Un procesador con 64Kb de memoria FLASH programable con capacidad para memoria RAM interna de por lo menos 1Kb y expandible hasta 256Kb con memoria externa.

En base a las consideraciones anteriores se han seleccionado los siguientes elementos:

- Un procesador Atmel AT89C51RD2.
- Un controlador de red de área local (CAN RTL8201BL).
- Un adaptador de línea LU1T516 (Para cable UTP con conector RJ-45).
- Una memoria EEPROM 24C02.
- Un circuito para realizar el PWR (Power reset).
- Un interfaz para el RS232 con un MAX232.
- Circuitería adicional para el correcto funcionamiento de los elementos.

Para este proyecto se ha utilizado un módulo que está provisto del controlador de red RTL8201LB, un adaptador de línea con conector RJ-45 y un dispositivo que tiene integrado como parte de hardware un software básico para manejo de algunos de los protocolos TCP/IP, este dispositivo es el W3100A y sus características se detallan a continuación:

2.6.1. W3100A [29]

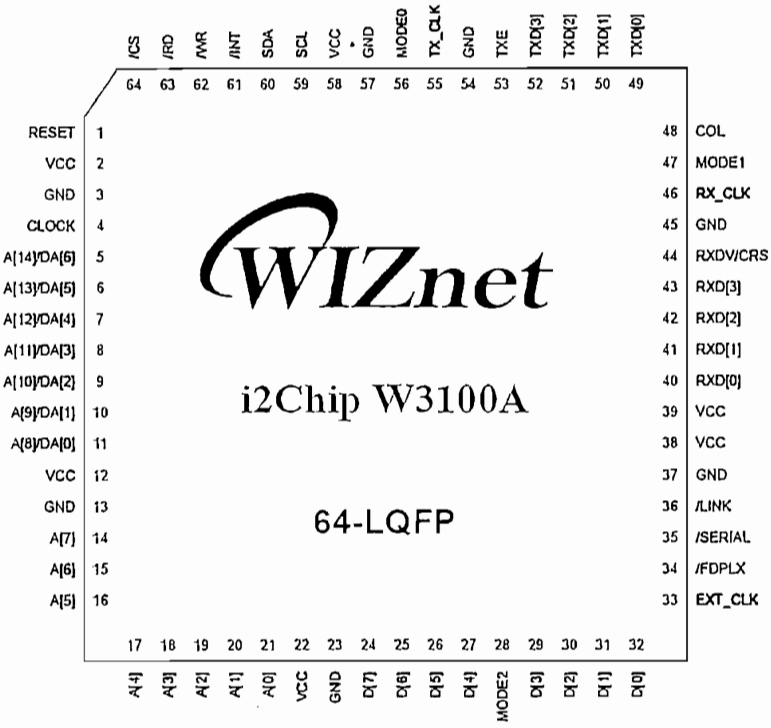


Figura 2.17 Distribución de pines del W3100A

Las características de este dispositivo son:

- Hardware que incluye manejo de protocolos TCP, IP versión 4, UDP, ICMP, ARP.
- Hardware que incluye manejo de protocolos a nivel de Ethernet como MAC y DLC.
- Soporte de 4 conexiones simultáneas independientes.
- El ICMP configurado internamente responde a comandos PING.
- Interfaz I2C.
- Interfaz stándar MII para la comunicación con elementos de capa física.
- Soporte para modo de transmisión full dúplex.
- 16Kbytes de memoria SRAM interna.
- Voltaje de operación de 3.3V, con tolerancia a 5V.

[29] W3100A, Hoja de datos del fabricante, www.atmel.com/dyn/general

- Encapsulado del tipo LQFP pequeño de 64 pines.

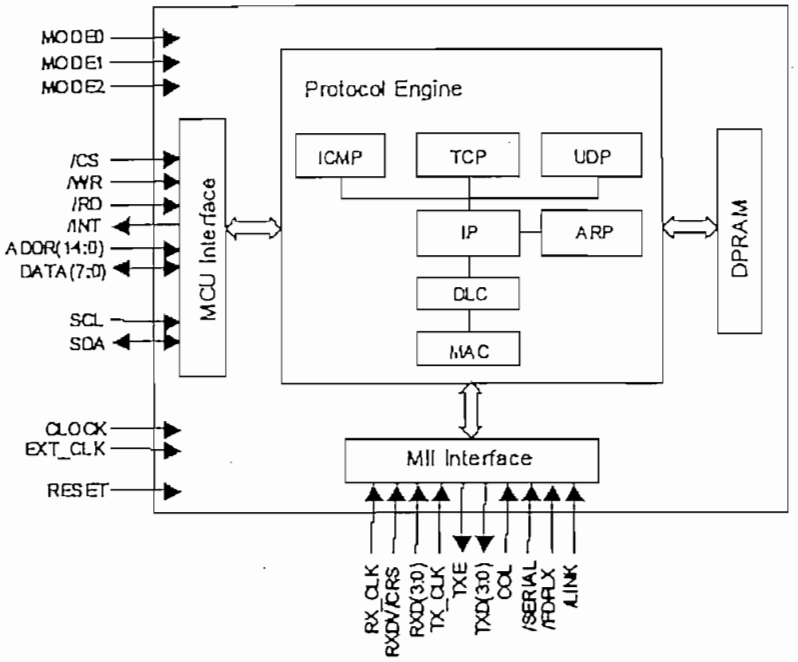


Figura 2.18 Diagrama de bloques interno del W3100A

Este dispositivo además puede ser utilizado para la implementación de otras aplicaciones como son: transmisión de voz sobre IP, manejo de imágenes utilizando una cámara Web, etc.

2.6.2. DESCRIPCIÓN DE LA INTERCONEXIÓN DE LOS ELEMENTOS QUE FORMAN EL PROTOTIPO.

El AT89C51RD2 es un microcontrolador de 8 bits, en el que se almacena el software para el funcionamiento del servidor Web. Contiene un CPU 80C51 que es el encargado del procesamiento de los datos que llegan desde la red.

Para que este procesador pueda interactuar con la red necesita de la ayuda del controlador de red (RTL8201BL). Este dispositivo hace uso de un adaptador de línea (LU1T516) que opera a velocidades de 10/100Mbps correspondientes a la red Ethernet 10/100Base-T y para la conexión física con la red utiliza un conector

RJ–45 para cable UTP. A través de este conector se realizará la transmisión y recepción de los datos y tramas que envíe y reciba el microcontrolador.

A continuación se muestra un esquema con las conexiones en forma detallada del microcontrolador AT89C51RD2 con los demás dispositivos. (Los esquemáticos se han desarrollado utilizando la herramienta Orcad, versión 9.2).

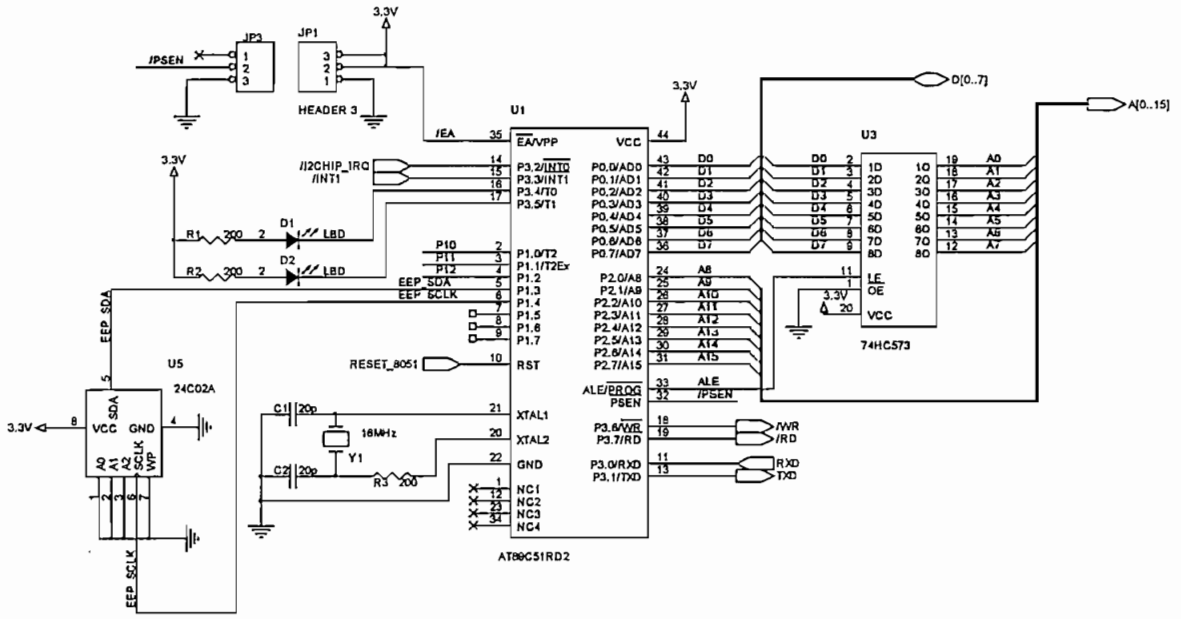


Figura 2.19 Esquemático del servidor Web (Primera parte)

En la figura anterior se puede observar que no todos los pines del microcontrolador están siendo utilizados, para la aplicación solo fueron necesarios algunos de ellos, los que están en contacto con el latch 74HC573, con la memoria EEPROM y con el switch para controlar el modo de programación, además de los indispensables para la configuración básica y el correcto funcionamiento de este dispositivo independientemente de la aplicación para la que sea utilizado.

Para el procesador AT89C51RD2:

- Los pines 44 Vcc y 1 Gnd, deben ir conectados a +3.3V y tierra común respectivamente para la correcta polarización y funcionamiento del microcontrolador.

- En este proyecto se utiliza memoria externa; por tanto, el pin 32 PSEN que es la señal de *strobe* para acceder a la memoria de programa externa va conectado al switch que habilita la función de programación del microcontrolador.

- El pin 33 ALE está conectado al latch 74HC573, que controla al dispositivo W3100A que es el que se encarga del manejo de protocolos TCP/IP y establece la conexión entre el controlador de red y el microcontrolador.

- Los pines 21 XTAL1 y 20 XTAL2, son la entrada y la salida, respectivamente de un amplificador inversor que está conectado a un cristal externo de 16MHz que es la frecuencia ideal para la sincronización de este elemento con los demás dispositivos de la tarjeta. Este cristal requiere de unos capacitores de 20pF conectados a tierra para desacoplamiento, este valor está indicado por el fabricante.

- En el pin 10 RST, si se coloca un valor en alto durante dos ciclos de máquina mientras que el oscilador está trabajando se reestablece el dispositivo. En este caso, esta entrada va conectada a la circuitería de reset descrita anteriormente.

- Los pines 11 P3.0 (RXD) entrada del puerto serial, 13 P3.1 (TXD) salida del puerto serial, están conectados al MAX232, son los encargados de la comunicación con el puerto serial, el mismo que va a interactuar con dispositivos externos, en este caso, por medio de este puerto se realiza la grabación del código del programa en la memoria del microcontrolador.

- Los pines 16 P3.4 (T0) y 17 P3.5 (T1) están conectados a los leds indicadores, los mismos que sirven para saber el estado en el que se encuentra en microcontrolador.

- Para este diseño, del puerto P1 solo se han utilizado los pines 5 y 6 para la comunicación con la memoria EEPROM externa, lo demás quedan disponibles para implementar cualquier otra aplicación.

- Los puertos P0 y P2 forman el bus de direcciones por medio del cual el microcontrolador va a interactuar con dispositivo W3100A que es el intermediario para la comunicación del microcontrolador con el controlador de red. El puerto P0 está conectado directamente al W3100A y al latch 74HC573

Para el controlador de red RTL8201BL:

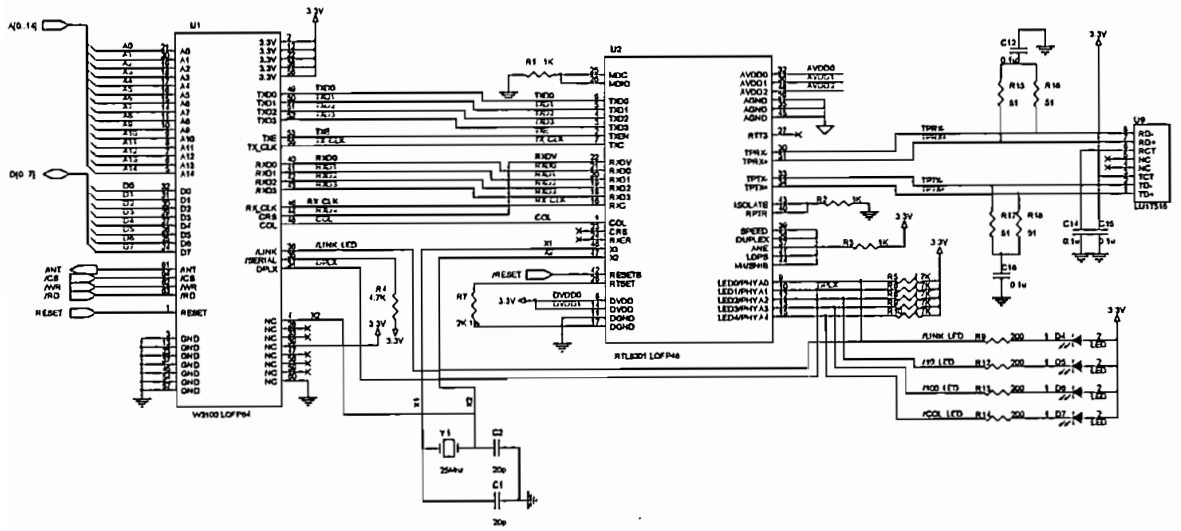


Figura 2.20 Esquemático del servidor Web (Segunda Parte)

- Los pines 14, 36 y 48 deben ir conectados a Vcc o +3.3V y los pines 11, 17, 29, 35 y 45 deben ir conectados a Gnd o tierra común para la correcta polarización del controlador.

- Los pines 25 MDC y 26 MDIO están conectados a una resistencia de 1KΩ. El pin 25 entrega una señal de reloj sincrónico para el MDIO, por medio de este pin se realiza la sincronización para la transferencia y

recepción de la información. El pin 26 es bidireccional y es usado para la administración de la transferencia de información.

- Los pines 2 TXEN, 3, 4, 5, 6 TXE y 7 TXC son los encargados de la sincronización y transmisión de datos entre el controlador del red y el W3100A.

- Los pines 16 RXC, 18, 19, 20, 21 RXD y 22 RXDV son los encargados de la sincronización y de establecer la comunicación para recibir los datos desde el W3100A.

- El pin 1 COL es el encargado de detectar colisiones en la transmisión.

- Los pines 46 X1 y 47 X2 se utilizan para conectar el cristal de 25MHz, este valor de cristal oscilador es el recomendado por el fabricante para el correcto funcionamiento de este controlador de red.

- Los pines 32 AVDD0 y 36 AVDD1 van conectados a un circuito formado por inductancias y capacitores cuyos valores están especificados por el fabricante y sirven para activar el correcto funcionamiento del dispositivo mientras que el pin 48 AVDD2 conectado en un valor alto (3.3 V) sirve para habilitar el dispositivo.

- Los pines 30 TPRX- y 31 TPRX+ son entradas diferenciales que son compartidas por el modo de funcionamiento ya sea 100Base-TX, 100Base-FX o 10Base-T y que están conectadas directamente al adaptador de línea LU1T516 con conector RJ-45.

- Los pines 33 TPTX- y 34 TPTX+ son las salidas diferenciales compartidas por los modos de funcionamiento 100Base-TX, 100Base-FX o 10Base-T y al igual que los pines anteriores también están conectadas al adaptador de línea LU1T516 con conector RJ-45.

- El pin 43 ISOLATE es la entrada que habilita al controlador de red a interactuar con dispositivos de la capa MAC, por lo que va conectado a un valor bajo.

- El pin 40 RPTR está conectado a un valor bajo con la finalidad de activar el modo repetidor, esto significa que puede estar enviando datos en forma constante si el caso lo amerita.

- El pin 39 SPEED se habilita durante la condición de reset, con un valor en alto garantiza que se establecerá una conexión de 100Mbps.

- El pin 38 DUPLEX habilita este modo de operación con un valor en alto durante las condiciones de funcionamiento o bien de reset.

- El pin 37 ANE es que el habilita la autonegociación cuando se conecta a un valor en alto.

- El pin 41 LDPS detecta el estado del enlace antes de habilitar o no la transmisión, si el enlace está caído, no transmite, pero apenas detecta un nivel de señal adecuado habilita la transmisión en modo de 10 o 100 Mbps, esto se refleja en una reducción del consumo de energía.

- El pin 44 MII/SNI cuando está conectado a un valor en alto habilita la transmisión MII (Media Independent Interface), ya sea para el modo 100Base-TX, 100Base-FX o 10Base-T.

- Los pines 9, 10, 12, 13, y 15 sirven para habilitar LEDS que puedan identificar cual es la función que está realizando el controlador de red, para este caso, estos leds no están hablitados.

Para el W3100A.

- Los pines 5 – 11 (A14 – A8) son entradas de direcciones y también son utilizados como dispositivos de direccionamiento cuando se trabaja en el modo I2C.

- Los pines 14 – 21 (A7 – A0) son entradas de direcciones.

- Los pines 24 – 27 (D7 – D4) y 29 – 32 (D3 – D0) son entradas y salidas de datos.

- El pin 61 INT es la salida de interrupciones, que se comunica con el microcontrolador después de haber realizado una transmisión o recepción.

- El pin 64 CS es el CHIP SELECT, esta señal se activa con un nivel en bajo.

- El pin 62 RW y 63 RD que habilitan la lectura y escritura respectivamente están conectados directamente al microcontrolador, se encargan del proceso de lectura y escritura entre este y el W3100A.

- El pin 1 es el reset que va conectado directamente al inversor 74HC14 para el proceso de reinicio en forma manual.

- Los pines 2, 12, 22, 38, 39 y 58 van conectados a +3.3V o Vcc para la correcta polarización del elemento.

- Los pines 3, 13, 23, 37, 45, 54 y 57 van conectados directamente a tierra para la polarización correcta del dispositivo.

- El pin 4 es un reloj primario requerido para funcionamiento interno del W3100A.

- El pin 60 SDA, está conectado a un nivel bajo con la finalidad de habilitar el acceso serial de los datos cuando se utiliza el modo I2C.

- Los pines 49 – 52 (TXD0 – TXD3) son las salidas de los datos y están conectados directamente al controlador de red.

- El pin 53 TXE es una salida que se activa cuando el primer paquete de datos serial es válido y se desactiva cuando el último paquete de datos está fuera. Esta señal se conecta directamente a un dispositivo de capa física, en este caso el controlador de red.

- El pin 55 TX_CLK es la entrada que recibe la señal de reloj de 25MHz al controlador de red.

- Los pines 40 – 43 (RXD0 – RXD3) son las entradas que se encargan de recibir los datos provenientes del controlador de red, estas dependen de la señal proveniente del pin 46 RX_CLK, este pin es una entrada que vuelve a sincronizar la señal que proviene del controlador de red, además indica que una portadora está presente.

- El pin 44 CRS es el encargado de detectar que la portadora.

- El pin 48 COL es una entrada que se activa cuando ha ocurrido una colisión en el modo half dúplex, esta es una señal asincrónica y se pone en valor alto y es ignorada cuando existe una transmisión full dúplex.

- El pin 34 DPLX es una entrada que permite realizar la selección del modo de transmisión, half o full dúplex y se debe colocar un valor de 0_L para activar el dispositivo en modo full dúplex y el valor 1_L para activar el dispositivo en modo half dúplex.

- El pin 35 SERIAL es una entrada por medio de la cual se selecciona el modo de transmisión de datos, con un nivel bajo la velocidad es de

10Mbps, pero si se coloca una resistencia de valor alto en el orden de los KΩ, entonces se puede configurar para que la transmisión sea en el modo MII.

2.7 DISEÑO DE SOFTWARE [25]

Para el desarrollo de software de este proyecto se necesita de la ayuda de un ensamblador, esto no es más que una herramienta del software diseñada para simplificar la tarea de escritura de programas para computadoras, se encarga de traducir el código simbólico en un código ejecutable. Entonces este código puede almacenarse en un microcontrolador para ser ejecutado.

Cuando se ensamblan un conjunto de programas lo que se hace es traducirlos en un idioma mediante el cual se pueda instruir al CPU para que este ejecute eficazmente las funciones que se desea realizar; por tanto al momento de escribir un programa se debe estar totalmente familiarizado con la arquitectura del microcontrolador y el lenguaje que puede interpretar.

Un programa hecho en un ensamblador consta de tres partes:

- Instrucciones de Máquina
- Directivas del ensamblador
- Control del ensamblador

Una *instrucción de máquina* es un código que puede ser ejecutado por la máquina. Las *directivas del ensamblador* se usan para definir la estructura del programa, los símbolos y generar código no ejecutable como datos, mensajes, etc. El *control del ensamblador* controla los modos del ensamblado y dirige el flujo del ensamblado.

Muchos programas son demasiado largos o complejos como para escribirlos como una sola unidad. El programar se vuelve mucho más simple cuando el

[25] Keil Software, "MACRO ASSEMBLER AND UTILITIES", User's Guide, www.keil.com, Febrero 2001

código es dividido en pequeñas unidades funcionales o módulos, además este tipo de programas son más fáciles de codificar, depurar y realizar cambios o actualizaciones.

Los beneficios de la programación por módulos son los siguientes:

Desarrollo de programa eficiente: Pueden desarrollarse programas más rápidamente, subsecuentemente los subprogramas pequeños son más fáciles de entender, diseñar y probar en comparación con los que están formados por un solo módulo. Una vez que se han definido las entradas y el rendimiento de cada módulo, el programador puede proporcionar la entrada requerida y verificar la exactitud de cada módulo examinando el rendimiento.

Una vez que esto ha terminado, los módulos separados se unen y son ensamblados como un solo módulo para el programa ejecutable, finalmente el módulo completo puede ser sometido a la última prueba.

Uso múltiple de subprogramas: El código escrito para un programa es a menudo útil para otros, la programación modular permite guardar estas instrucciones para ser utilizadas en el futuro. El código puede ser reutilizado, se pueden unir con otros módulos que estén previamente almacenados y que cumplan con los requisitos de entrada y salida. También se tiene la facilidad de guardar estas rutinas para ser usadas solamente con ciertos programas; es decir, que no estén disponibles para todos.

Facilidad de depuración y modificación: Los programas modulares son por lo general más fáciles de depurar que otro tipo de programas, debido a que las interfaces para cada módulo están bien definidas, se pueden aislar los problemas que presentan cada módulo en forma específica.

Una vez que se ha detectado el problema en el módulo defectuoso es más fácil arreglarlo y modificarlo, gracias a la programación modular que simplifica el

trabajo. Se pueden realizar estas modificaciones y depurar solamente dicho módulo teniendo la certeza que el resto del programa no cambiará

2.7.1 INTRODUCCIÓN A LA HERRAMIENTA DE DESARROLLO KEIL [26]

Keil es un software que tiene herramientas de desarrollo y se usan para compilar y codificar en un lenguaje de programación. *Keil* soporta dos lenguajes: *Assembler y C.*

Los archivos fuente creados pasan por el compilador C51 o el ensamblador A51 dependiendo del tipo de lenguaje que se está usando.

El compilador C51 es una aplicación que maneja el lenguaje ANSI - C, además incluye varias herramientas que son específicas para ayudar al desarrollo de software para la arquitectura 8051, mientras que el ensamblador A51 apoya el juego completo de instrucciones para programación en bajo nivel de este tipo de arquitecturas.

Keil se encarga de ensamblar los códigos fuente, enlaza y localiza los objetos, módulos y librerías, crea archivos HEX y pone a punto el programa desarrollado para cada tarjeta. Este software permite escoger el tipo de procesador que se va usar antes de empezar con el desarrollo del programa; es decir, se adapta a cada uno de los diseños que se desea crear.

Keil ha abierto numerosas herramientas que ofrecen ciertas características y ventajas para el desarrollo de aplicaciones en sistemas embebidos. Es una plataforma de desarrollo de software bajo *Windows* basado en la combinación de un editor robusto y un administrador de proyecto, lo que brinda muchas facilidades, entre ellas se tiene:

- Editor de código fuente completo

[26] Keil Software, "Cx51 COMPILER", User's Guide, www.keil.com, Septiembre 2001.

- Base de datos del dispositivo para configurar el ambiente de desarrollo de herramientas.
- Un administrador de proyecto que le permite crear y mantener los proyectos que se van desarrollando.
- Facilidad de desarrollo integrada para ensamblar, compilar, depurar y unir las aplicaciones para sistemas embebidos.
- Diálogos para todos los ambientes de herramientas para desarrollo.
- Verdadera corrección de errores integrado con un simulador de CPU periférico de gran velocidad.
- Vínculos a los manuales, herramientas de desarrollo, hojas de datos de los dispositivos a ser utilizados y guías de desarrollo.

2.7.1.1 Compilador C51

El compilador C51 del *Keil* es un compilador ANSI 51 que se escribió específicamente para generar un código rápido y compacto para la familia de microcontroladores 8051. El compilador C51 genera un código objeto que se asemeja a la eficacia y velocidad de la programación en *Assembler*.

Al utilizar un lenguaje de alto nivel como es el lenguaje C se tiene muchas ventajas sobre el lenguaje *Assembler* ya que no se requiere conocer todo el juego de instrucciones que se necesita para usar el ensamblador, tan solo se requiere un conocimiento básico de la estructura de memoria del CPU 8051, adicionalmente existen muchas funciones que están integradas en las librerías del compilador. Detalles tales como la asignación de registros y el direccionamiento de los diferentes tipos de memoria y de datos son manejados por el compilador.

Los programas mantienen una estructura formal (estructura impuesta por el lenguaje de programación C) pero se puede dividir en funciones separadas. Esto contribuye a la reutilización del código fuente así como a una mejor estructura de aplicación global gracias a la habilidad de combinar estas rutinas inconstantes con rutinas específicas mejorando la legibilidad del programa, sin mencionar el ahorro en cuanto a tiempo al momento de desarrollar el programa.

2.7.1.2 Ensamblador A51

El A51 es un macro ensamblador de la familia de microcontroladores 8051. Se encarga de traducir el lenguaje mnemónico simbólico a un lenguaje ejecutable de máquina. El ensamblador A51 maneja el juego de instrucciones existentes para un microcontrolador 8051, usando la velocidad máxima, tamaño de código pequeño y el hardware exacto, además del control que es esencial. Las facilidades que brinda este ensamblador permiten ahorrar tiempo en el desarrollo y mantenimiento del programa porque las rutinas que son comunes necesitan ser desarrolladas una sola vez.

2.7.1.3 Creación de un proyecto usando Keil.

Para crear un proyecto usando *Keil* debe seguir los siguientes pasos:

- En el menú ***Project*** seleccione la opción ***New Project***
- Inmediatamente le pedirá que guarde este nuevo proyecto con un nombre, el que usted desee asignarle.
- En la pantalla que aparece a continuación, debe escoger el tipo de CPU con el que va a trabajar.
- En el menú ***File*** se escoge ***New*** para obtener un nuevo archivo de texto para empezar con la programación.
- En el mismo menú está la opción guardar para designarle un nombre a este archivo de texto y salvar los cambios realizados.

Se pueden implementar cuantos archivos de texto sean necesarios, cada uno de estos corresponde a un módulo del programa (que podría corresponder a lenguaje C o *Assembler*), para ensamblar estos módulos como uno solo se deben realizar los siguientes pasos:

- En la ventana de la izquierda cuando se ha seleccionado un nuevo proyecto aparece una carpeta llamada ***Target 1***, debajo está una subcarpeta llamada ***Source Group 1***, haciendo clic con el botón secundario del *mouse* se escoge la opción ***Add Files to group "Source***

Group 1" y se agregan los archivos de texto o módulos que se han creado al desarrollar el programa.

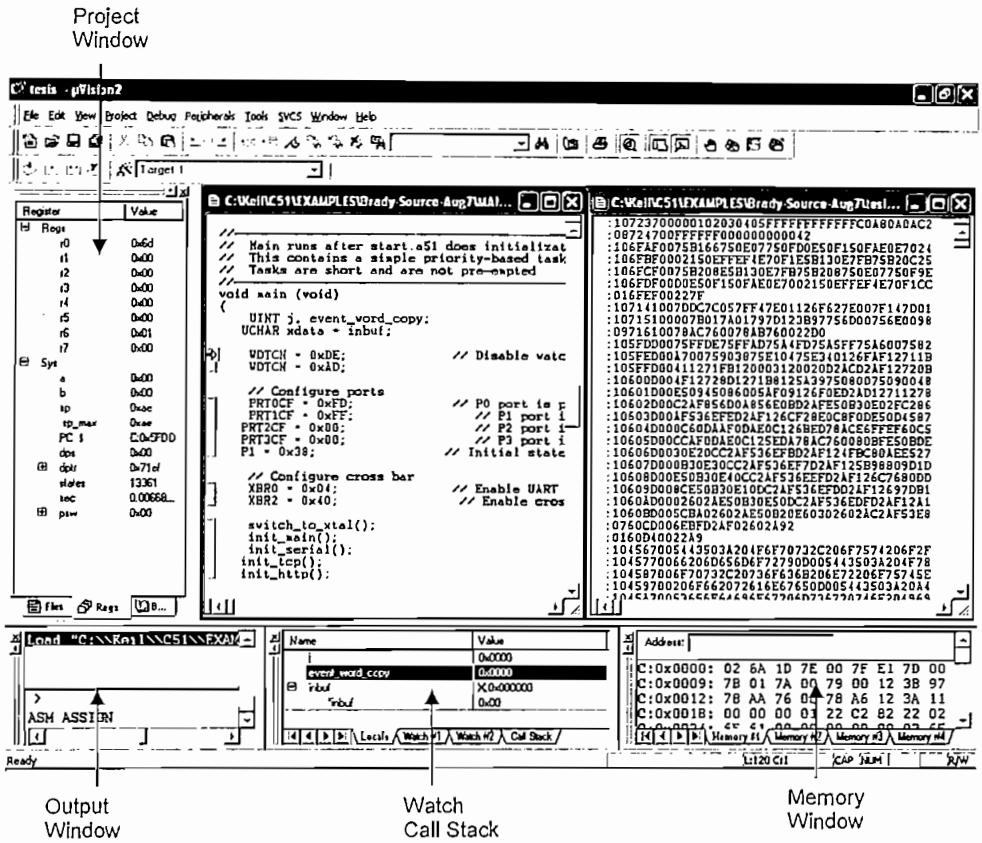


Figura 2.21 Ambiente de trabajo de Keil.

- Una vez que están todos los archivos necesarios se procede a construir la tarjeta, en el menú ***Project*** se escoge la opción ***Build Target*** y en la opción ***Options for Target*** se puede configurar la alternativa para crear el archivo HEX que se grabará en el microcontrolador. Este archivo se creará únicamente si no existen errores de sintaxis, en el caso de que esto suceda en la parte inferior de la pantalla llamada ***Output Window*** se muestran los mensajes de error existentes.

- En el menú ***Debug*** está la opción ***Start/Stop Debug Session***, la misma que nos permite realizar la simulación para ver lo que ocurre en el CPU cuando se implementa este programa.

- En la pantalla de la izquierda, ahora se tienen los estados de los registros en lugar de los archivos que se ensamblaron para el proyecto.

2.7.2 LIGERA REVISIÓN DEL LENGUAJE C [26]

El lenguaje de programación C es de propósito general, proporciona eficacia en cuanto al código, elementos de programación estructurada y gran juego de operadores.

Este lenguaje de programación no es un idioma grande diseñado para una aplicación particular, su generalidad combinada con la ausencia de restricciones hace que este lenguaje sea una solución de programación conveniente y eficaz para una gran variedad de tareas que se realizan desarrollando software. Muchas aplicaciones pueden resolverse fácilmente empleando el lenguaje C en lugar de otros lenguajes de programación especializados.

A pesar de todas las ventajas mencionadas anteriormente, para poder adaptar este lenguaje de programación al desarrollo de software que pueda ser aplicado a los sistemas embebidos como es el caso de esta aplicación, varias de las librerías han sido alteradas, modificadas y reforzadas con la finalidad de satisfacer los requerimientos de los microcontroladores x51, mantener la flexibilidad del lenguaje C y la eficacia y velocidad del código *Assembler.*

La familia de los microcontroladores 8051 es una de las arquitecturas que más crecimiento ha experimentado en los últimos años, existen más de 400 modelos creados por los diferentes fabricantes, inclusive se ha mejorado su capacidad de almacenamiento a varios *Mbytes* con la finalidad de poder emplear este tipo de dispositivos en aplicaciones mucho más complejas. Por eso las modificaciones realizadas al lenguaje C y acogidas por la versión de compilador Cx51 de *Keil* ofrecen una solución flexible adaptable a cualquier modelo de procesador 8051 como Philips, Atmel, Intel, Dallas, etc.

---

[26] Keil Software, "Cx51 COMPILER", User's Guide, www.keil.com, Septiembre 2001.

Sin embargo se mantienen varias de las extensiones del lenguaje C normal de ANSI para dar soporte a la arquitectura 8051. Se incluyen extensiones para:

- Áreas de memoria.
- Tipos de memoria
- Modelos de memoria
- Especificaciones del tipo de memoria.
- Especificaciones del tipo de datos variables.
- Variables tipo bit y bits direccionables de datos.
- Registros de funciones especiales.
- Indicadores o punteros.

2.7.2.1 Características

El compilador Cx51 difiere en muy pocos aspectos en relación al lenguaje C ANSI normal. Estas diferencias pueden agruparse en las diferencias relacionadas con el compilador y las librerías. Las diferencias relacionadas con el compilador son:

Caracteres extensos: Los caracteres extensos de 16 bits no son soportados por el compilador Cx51. ANSI proporciona soporte para caracteres extensos que pueden ser usados en el futuro a nivel internacional.

Llamadas a funciones recursivas: Las llamadas a funciones recursivas no son soportadas por defecto, las funciones que son recursivas deben declararse utilizando funciones de reentrada. Las funciones de reentrada pueden llamarse recursivas porque los datos locales y los parámetros son almacenados en el *stack* de reentrada.

En las librerías que forman parte del compilador Cx51 existen más de 100 funciones predefinidas que sirven para el desarrollo exclusivo de software para arquitecturas 8051, estas funciones hacen que el desarrollo de software para sistemas embebidos sea mucho más fácil proporcionando rutinas que realizan tareas comunes como *strings,* manipulación del *buffer*, conversión de datos,

operaciones matemáticas de punto flotante, etc. Generalmente estas funciones son parte de las librerías normales del lenguaje C de ANSI.

Algunas de las funciones mencionadas en el párrafo anterior difieren ligeramente para aprovechar las ventajas de las características encontradas en la arquitectura 8051, en donde los diferentes tipos de funciones de retorno se ajustan a los diversos tipos de argumentos para ser factible el uso de datos más pequeños (bits). Estas alteraciones a las librerías del lenguaje C normal proporcionan un máximo desempeño a la vez que reducen el tamaño de programa.

Las librerías del ANSI C estándar incluyen muchas rutinas que también están en el compilador Cx51; sin embargo, muchas de ellas no pueden ser utilizadas para el desarrollo de aplicaciones de sistemas embebidos y se excluyen de las librerías del Cx51.

A continuación se mencionan las rutinas de las librerías del ANSI estándar que están incluidas en el compilador Cx51:

abs	fmod	labs
acos	free	log
asin	getchar	log10
atan	gets	longjmp
atan2	isalnum	malloc
atof	isalpha	memchr
atoi	iscntrl	memcmp
atol	isdigit	memcpy
calloc	isgraph	memmove
ceil	islower	memset
cos	isprint	modf
cosh	ispunct	pow
exp	isspace	printf
fabs	isupper	putchar
floor	isxdigit	puts

rand
realloc
scanf
setjmp
sin
sinh
sprintf
sqrt
srand
sscanf
strcat
strchr
strcmp
strcpy
strcspn
strlen
strncat
strncmp
strncpy
strpbrk
strrchr
strspn
strstr
strtod
strtol
strtoul
tan
tanh
tolower
toupper
va_arg
va_end
va_start
vprintf
vsprintf

Las siguientes son rutinas del ANSI estándar que no están incluidas en las librerías del Cx51.

abort
asctime
atexit
bsearch
clearerr
clock
ctime
difftime
div
exit
fclose
feof
ferror
fflush
fgetc
fgetpos
fgets
fopen
fprintf
fputc
fputs
fread
freopen
frexp
fscanf
fseek
fsetpos
ftell
fwrite
getc
getenv
gmtime
ldexp
ldiv
localeconv
localtime
mblen
mbstowcs
mbtowc
mktime
perror
putc
qsort
raise
remove
rename
rewind
setbuf

setlocale	strtok	ungetc
setvbuf	strxfrm	vfprintf
signal	system	wcstombs
strcoll	time	wctomb
strerror	tmpfile	
strftime	tmpnam	

Las siguientes rutinas son las que no se encuentran en el ANSI estándar pero están incluidas en las librerías del Cx51.

acos517	_iror_	strpos
asin517	log10517	strrpbrk
atan517	log517	strrpos
atof517	_lrol_	strtod517
cabs	_lror_	tan517
chkfloat	memccpy	_testbit_
cos517	_nop_	toascii
crol	printf517	toint
cror	scanf517	_tolower
exp517	sin517	_toupper
_getkey	sprintf517	ungetchar
init_mempool	sqrt517	
irol	sscanf517	

2.7.2.2 Tipos de memoria

La arquitectura 8051 soporta varias áreas de memoria físicamente separadas y espacios de memoria para datos y programas. Cada área de memoria ofrece ciertas ventajas y desventajas. Esta variedad del espacio de memoria es muy diferente a la mayoría de maimframes, miniordenadores y arquitecturas de microordenadores donde el programa, los datos y las constantes son almacenados en el mismo espacio físico de memoria dentro la computadora.

El modelo de memoria determina el espacio de memoria predefinido para ser usado por funciones, argumentos, variables automáticas y declaraciones. Existen tres modelos de memoria: pequeño, compacto y grande.

Modelo pequeño: Todas las variables residen en la memoria de datos interna de la arquitectura 8051. En este modelo de memoria el acceso variable es muy eficaz. Todos los objetos así como el *snack* de protocolos deben encajar en la RAM interna.

Modelo compacto: Usando el modelo compacto, todas las variables residen en una página externa de la memoria de datos. Este modelo de memoria puede acomodar un máximo de 256bytes de variables. La limitación se debe a la utilización del esquema de direccionamiento indirecto a través de los registros R0 y R1 (@R0 y @R1). El acceso variable a este modelo de memoria no es rápido debido a que es menos eficaz que el modelo pequeño.

Modelo grande: En el modelo grande todas las variables residen en la memoria de datos externa (hasta de 64 kbytes). El puntero de datos (DPTR) es usado para el direccionamiento. El acceso a la memoria que se hace a través del puntero de datos es ineficiente especialmente con variables que tienen una longitud de dos o más bytes. Este mecanismo de acceso de datos genera más código que los modelos pequeño y compacto.

2.7.2.3 Limitaciones

El compilador Cx51 incluye entre otras, algunas de las siguientes limitaciones:

- Los nombres de variables pueden tener hasta 256 caracteres de longitud.
- El número máximo de funciones anidadas en una lista de parámetros es 10.
- El número máximo *include files* anidados es 9.
- La máxima longitud de una línea o una definición de un macro es de 2000 caracteres.
- Una llamada de función puede pasar hasta 32 parámetros.

2.7.3 ELABORACIÓN DEL DIAGRAMA DE BLOQUES GENERAL DEL PROGRAMA.

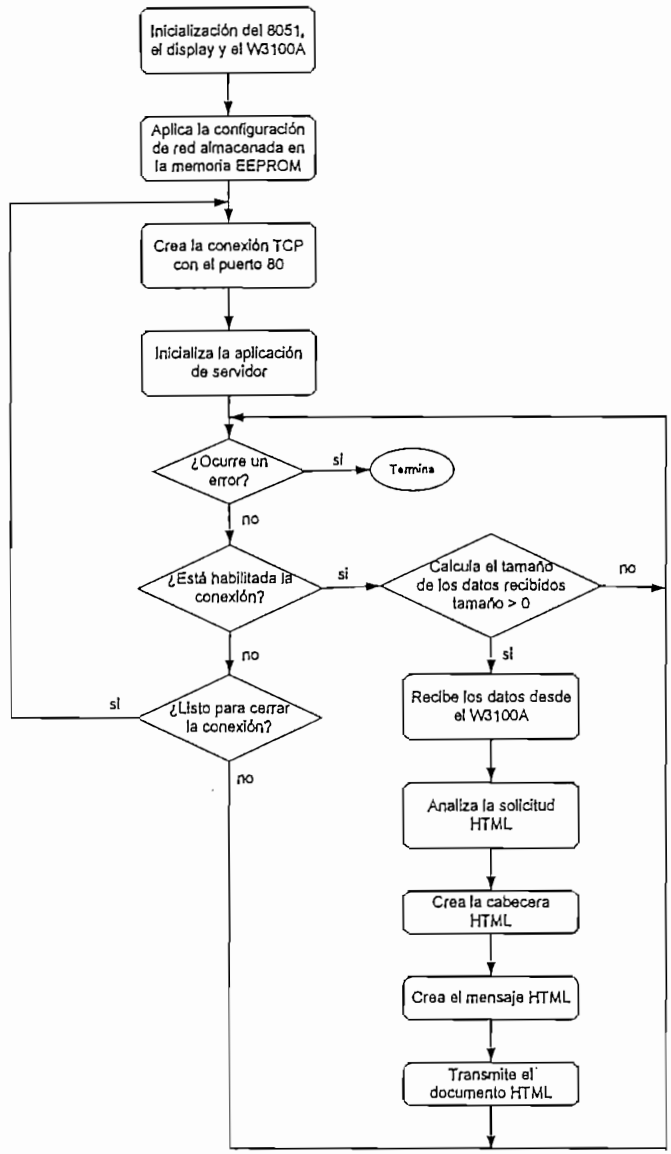


Figura 2. 22 Diagrama de bloques del programa

2.7.4 DESCRPICION DEL PROGRAMA

El software para la implementación del servidor Web, se ha desarrollado por módulos, está es una de las facilidades de la herramienta Keil.

Este software consta de los siguientes módulos:

- Inicio.C

- Control.C
- Eeprom.C
- Función.C
- Puerto.C
- Red.C
- Tesis.C
- Cambio.C
- Screen.C

Inicio: Este módulo describe las funciones del servidor Web, las mismas que son muy simples, además establece la conexión con el puerto 80, esta desarrollado únicamente para presentar páginas con formato *html*.

Control: Contiene las rutinas para la programación del controlador de red, habilita y cierra las conexiones de éste con el medio físico.

Eeprom: Implementa las funciones I2C para habilitar la programación de la memoria utilizando el puerto serie.

Función: Define el interfaz que se maneja utilizando el puerto serie para cambiar las configuraciones de red que se almacenan en la memoria EEPROM.

Puerto: Define las rutinas implementadas para el manejo del puerto serie.

Red: Maneja las rutinas de configuración para el W3100A, obtiene los datos de configuración de la red almacenados en la memoria EEPROM.

Tesis: Se encarga de la edición de la páginas Web que se van a almacenar el el microcontrolador.

Cambio: En este archivo se encarga de realizar las conversiones que sean necesarias, como por ejemplo transformar los caracteres a código hexadecimal y viceversa, etc.

Screen: Implementa las funciones básicas para el manejo del display.

El código fuente de todos los módulos mencionados anteriormente está en el anexo.

2.7.5 REQUERIMIENTOS DE MEMORIA FLASH Y EEPROM

Este microcontrolador tiene una memoria Flash 64Kbytes, en está memoria se almacena el código fuente y la página Web que el servidor va a mostrar.

La información de la trama Ethernet se almacena en la memoria RAM interna del controlador.

En la memoria EEPROM se almacenan los datos correspondientes a las direcciones que maneja el microcontrolador como son: dirección MAC, IP, Máscara y Gateway, entre otros.

2.7.6 MAPA DE MEMORIA

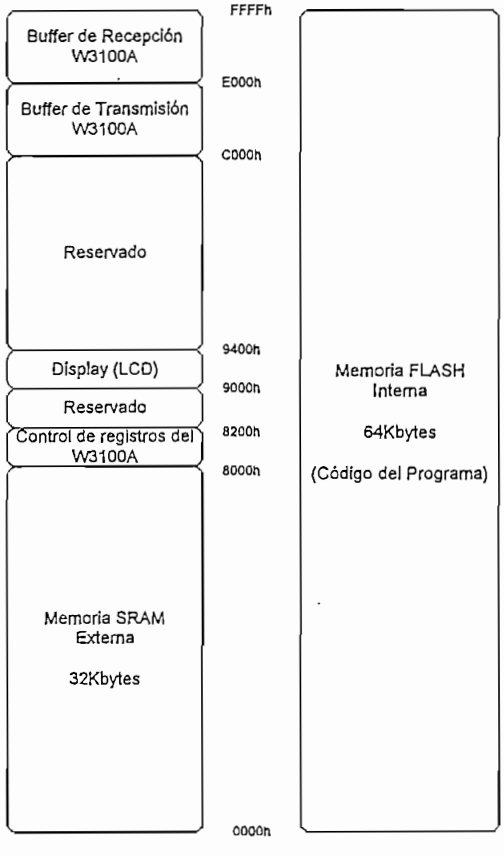


Figura 2. 23 Mapa de Memoria

Limitaciones:

Para este proyecto se ha utilizado un microcontrolador AT89C51RD2, el mismo que por sus características tiene ciertas limitaciones.

- El tamaño de página máximo debe ser menor a 64 Kbytes, ya que se almacena en la flash interna, se debe descontar el espacio necesario para el código ejecutable de la aplicación.

- Se podría utilizar la memoria RAM externa de 32 Kbytes para almacenar la página Web, pero esto limitaría su tamaño y requeriría que la información sea descargada nuevamente cada vez que se suspenda la alimentación eléctrica del prototipo.

2.8 CONSTRUCCION DEL PROTOTIPO

Para la implementación se utiliza el sistema EVB8051 provisto por ATMEL, que ofrece las facilidades necesarias para el desarrollo de la aplicación propuesta en el presente proyecto.

Este proyecto conecta un procesador Atmel AT89C51RD2 (compatible con la familia MCS – 51 de Intel) con un controlador de red Realtek RTL8201BL. El resultado es un económico y práctico interfaz de red, fácil de usar con una Ethernet. Puesto que se usa una arquitectura bien conocida como es la del x51, incluso los negocios de escala pequeña y mediana ahora puede implementar esta clase de dispositivos en sus redes.

Este prototipo tiene varias facilidades:

- Para la programación del microcontrolador no se necesita de un programador, ya que este proceso se lo puede realizar utilizando el puerto serial de la computadora.

- Para cambiar la configuración de red en la memoria EEPROM, se lo puede hacer mediante una conexión utilizando el puerto serie de la computadora y estableciendo la comunicación por medio del Hyper Terminal.
- Para realizar pruebas previas a la puesta en marcha del proyecto, el prototipo se puede conectar a la computadora a su tarjeta de red haciendo uso de un cable UTP cruzado.

En la siguiente figura se muestra un diagrama de bloques con todos los elementos implementados para conocer la estructura del prototipo en forma más detallada.

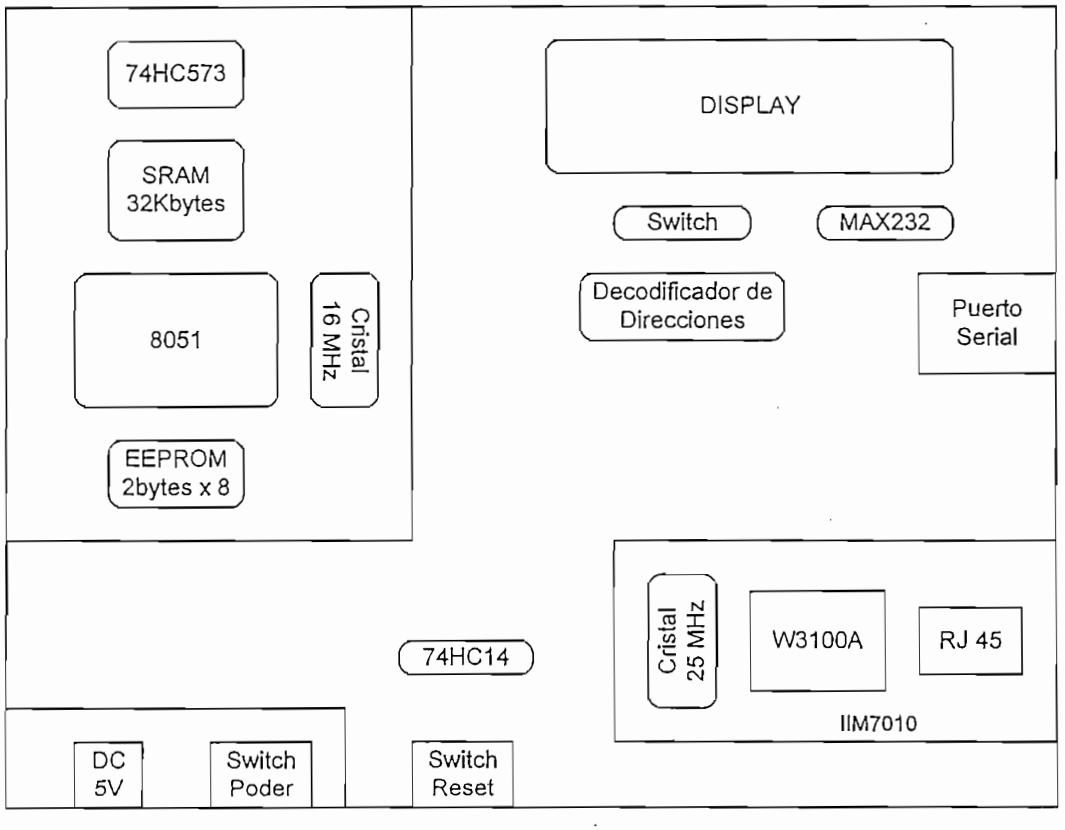


Figura 2. 24 Diagrama de bloques del prototipo

El módulo IIM7010 es el que reúne al controlador de red RTL8201BL en la parte inferior y en la parte superior se encuentran el W3100A y el interfaz RJ-45. Este módulo además de facilitar la conexión del resto de los elementos con la capa

física ya que no necesita un detalle de la configuración de los circuitos en la red, provee una solución a bajo costo.

A continuación se muestra el sistema EVB8051 utilizado:

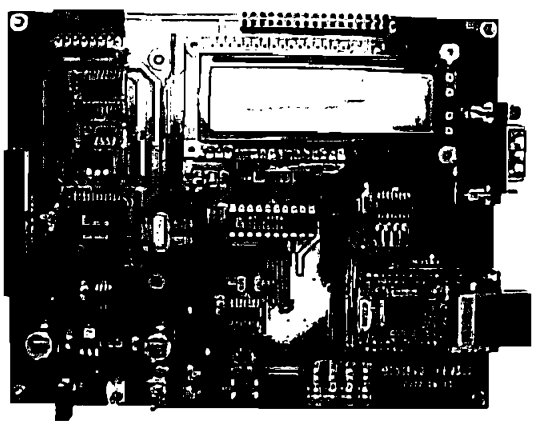


Figura 2. 25 Fotografía del prototipo

El módulo IIM7010A es el siguiente:

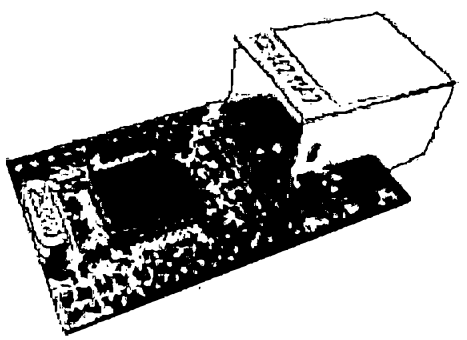


Figura 2. 26 Fotografía del Módulo IIM7010A

2.8.1 HERRAMIENTAS PARA LA PROGRAMACION DEL MICROCONTROLADOR.

Para programar el microcontrolador, se hace uso de una herramienta de software que es proporcionado por el fabricante, este software toma el nombre de FLIP en

la versión 1.8.8, tiene una interfaz muy amigable y es de fácil instalación y utilización.

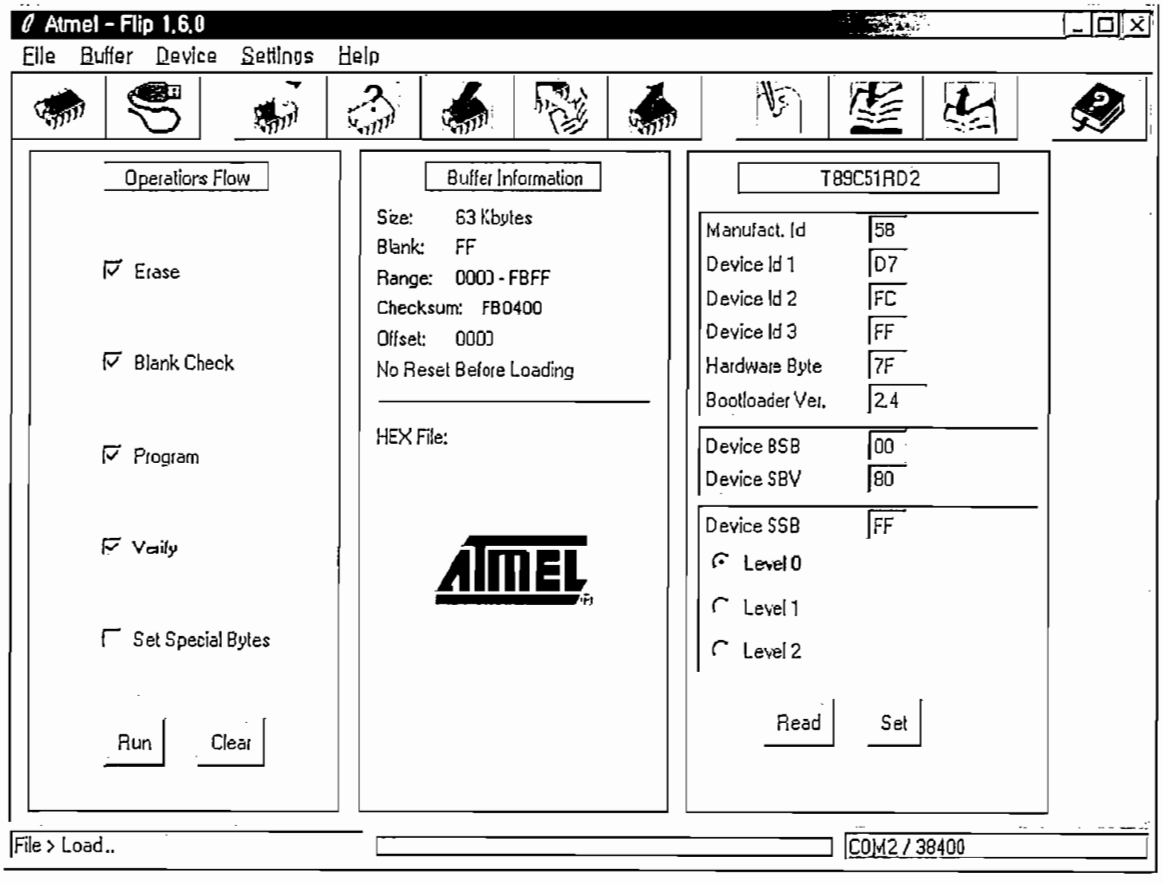


Figura 2. 27 Interfaz FLIP

El FLIP es un software que permite grabar archivos con código hexadecimal en los microcontroladores de la familia ATMEL, eliminando la necesidad de utilizar un programador para estos dispositivos.

Las conexiones que se deben realizar para habilitar estas funciones son:

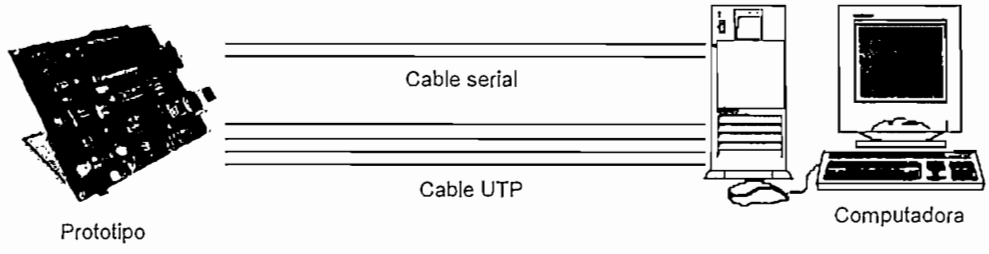


Figura 2. 28 Conexión del prototipo a la computadora

Para programar el microcontrolador, primero se debe escoger cual es el elemento que se va programar, para esto se hace clic en el primer botón en la parte superior desde la derecha, una vez que se ha escogido el integrado correcto, se debe realizar la conexión con el puerto serial de la computadora, para esto se presiona el según botón en la parte superior desde la derecha, se escoge el puerto, la velocidad de conexión está establecida en 38400, el interfaz gráfico que aparece al realizar este paso es el siguiente:

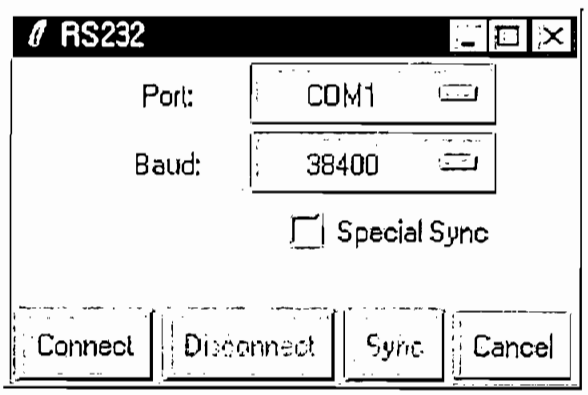


Figura 2. 29 Conexión con el puerto serie

2.8.2 ASIGNACIÓN DE UNA DIRECCION MAC.

El prototipo tiene una dirección MAC dada por el fabricante, sin embargo esta dirección se la puede cambiar haciendo uso del puerto serie.

La dirección que está dada por el fabricante es 00 08 DC 00 00 00, esta dirección ha sido cambiada por otra. La nueva dirección MAC asignada es: 00 A0 4B 02 97 AD.

2.8.3 ASIGNACION DE UNA DIRECCION IP VALIDA

Para el presente proyecto se ha asignado la dirección 192.188.57.195, que corresponde a una dirección válida del grupo de direcciones asignadas a la ex - Facultad de Ingeniería Eléctrica, para permitir la comprobación de funcionamiento del sistema desde cualquier parte.

2.8.4 COSTOS DE VARIOS PROTOTIPOS SIMILARES

Existen muchos tipos de sistemas embebidos, unos tienen mayores facilidades en comparación a otros, debido a que este tipo de sistemas se caracterizan por la variedad de soluciones y sobre todo porque existen algunas que son compatibles con varios tipos de microcontroladores para el desarrollo de diferentes aplicaciones, el precio también cambia, en especial los de aquellos sistemas que brindan la posibilidad de un mayor número de aplicaciones y con alta capacidad de almacenamiento.

Entre los sistemas embebidos más populares que se encuentran el mercado tenemos.

Equipo	Descripción	Precio (USD)
Lantronix	Servidor embebido con leds que indican el estado de la comunicación.	149
Parallax	Sistema embebido, diseñado para el desarrollo de 10 aplicaciones diferentes.	159
Rabbit	RCM2000 Kit de desarrollo de sistemas embebidos, soporte para protocolos TCP/IP.	359
Rabbit	RCM2300. Kit de desarrollo con una mayor capacidad de almacenamiento que el RCM2000	359
Rabbit	RCM3000 Kit completo de desarrollo	379
Radiotronix	Kit de desarrollo de sistemas embebidos para comunicación inalámbrica	349
Arbor	Módulos de sistemas embebidos de fácil integración a las computadoras con procesador Pentium III	875
Charon I	Kit de desarrollo de sistemas embebidos con un procesador 8252	170
Charon II	Kit de desarrollo de sistemas embebidos con un procesador 89C51RD2	210
AVR 460	Kit de desarrollo de Atmel con un procesador 89C51RD2	330
WizNet	Kit de desarrollo de Atmel con un procesador 89C51RD2 con pantalla LCD.	250

CAPÍTULO 3

PRUEBAS DEL PROTOTIPO IMPLEMENTADO

3.1 PRUEBA DE OPERACIÓN

Para comprobar el correcto funcionamiento del prototipo se asigna la dirección válida 192.188.57.195 que corresponde a la máquina fie195.epn.edu.ec (en el Servidor de nombres).

Desde cualquier parte de Internet ejecutamos el programa navegador que en este caso es Internet Explorer y especificamos la dirección:

http://fie195.epn.edu.ec

El resultado es la siguiente pantalla:

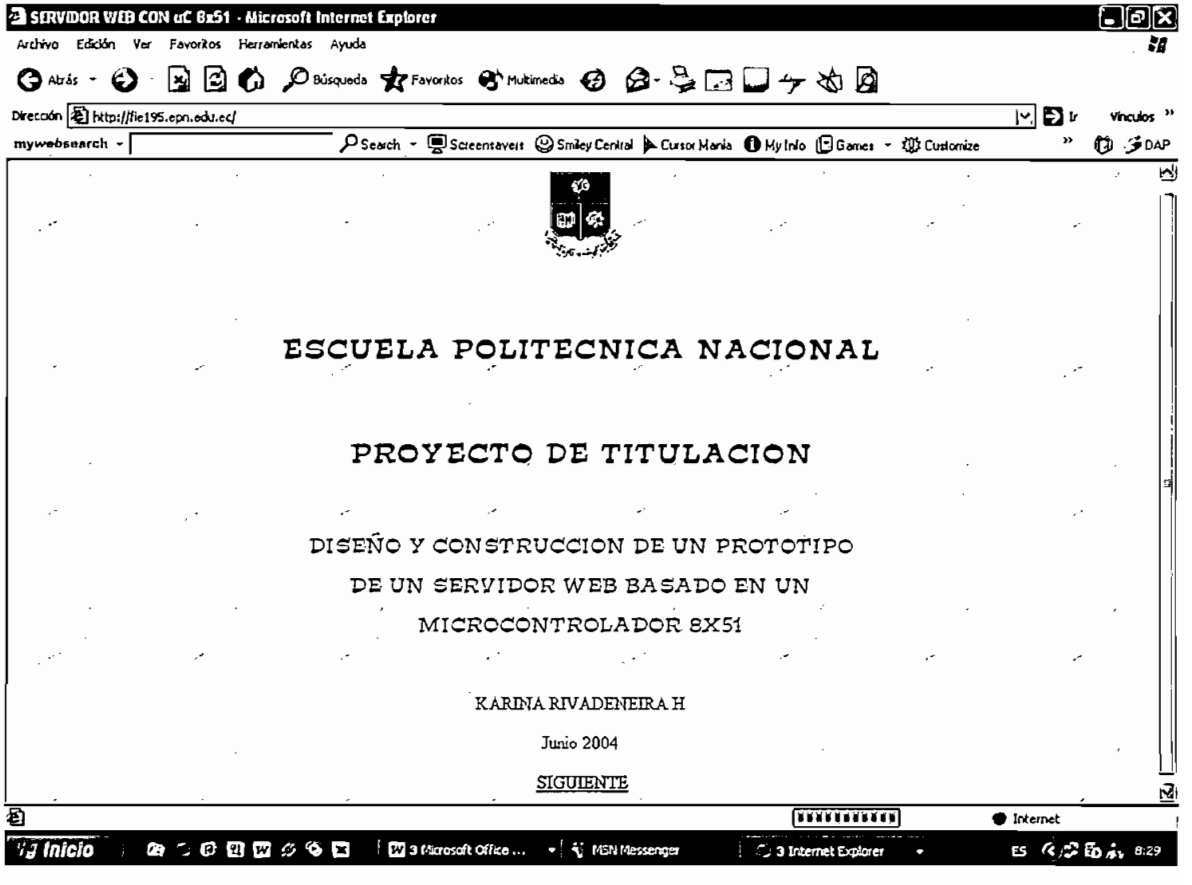


Figura 3.1 Página implementada en el Prototipo

Luego se procede a navegar por los enlaces de la página principal y se ha verificado el correcto funcionamiento del mismo.

3.2 PRUEBA DE PING

Al ejecutar el comando ping a la dirección asignada ***fie195.epn.edu.ec*** con 128 bytes de datos desde la misma red se tiene respuesta inmediata, tal como se detalla en la figura siguiente.

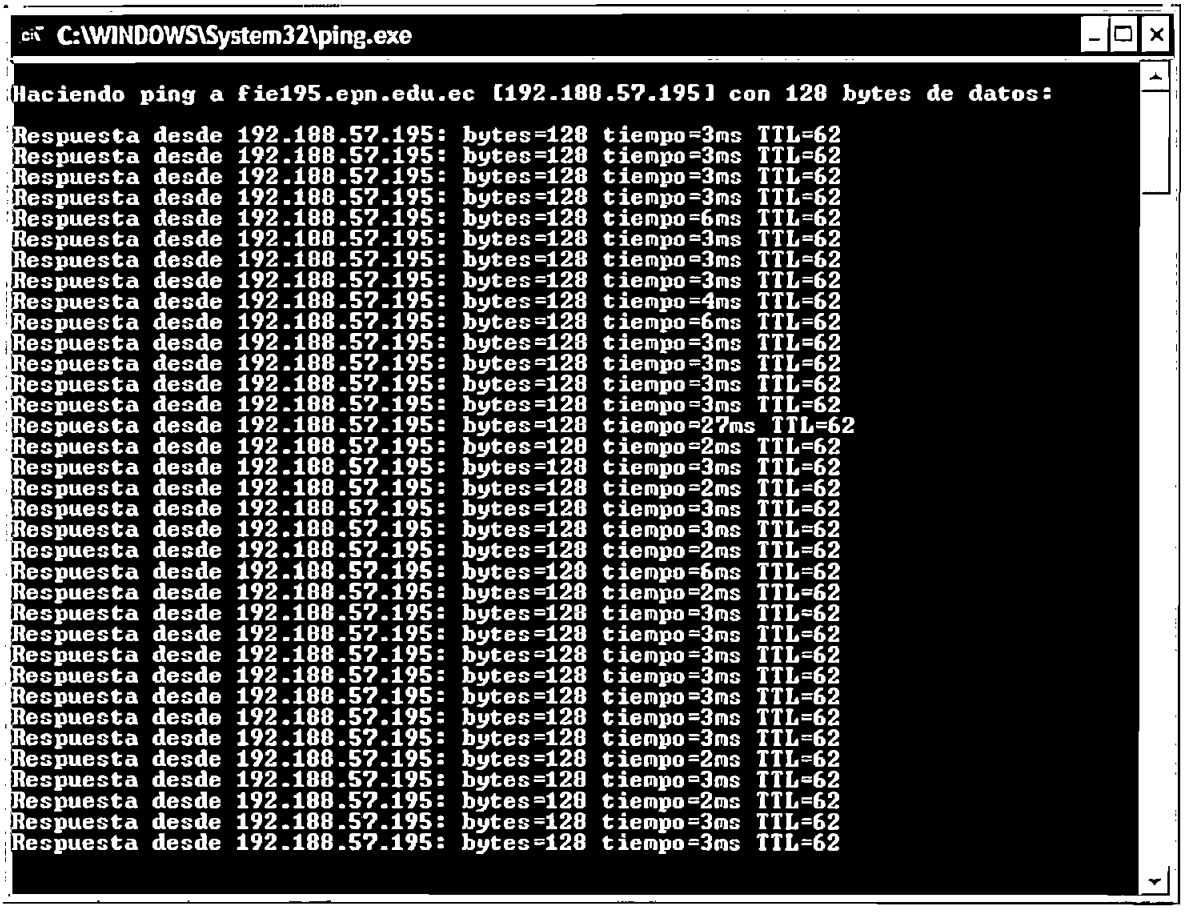


Figura 3.2 Prueba de Ping dentro de la misma red.

Al ejecutar el comando ping a la dirección asignada ***fie195.epn.edu.ec*** con 128 bytes de datos desde una red externa utilizando una conexión vía MODEM con un proveedor de ISP diferente, existe una variación significante en la respuesta, tal como se detalla en la figura siguiente:

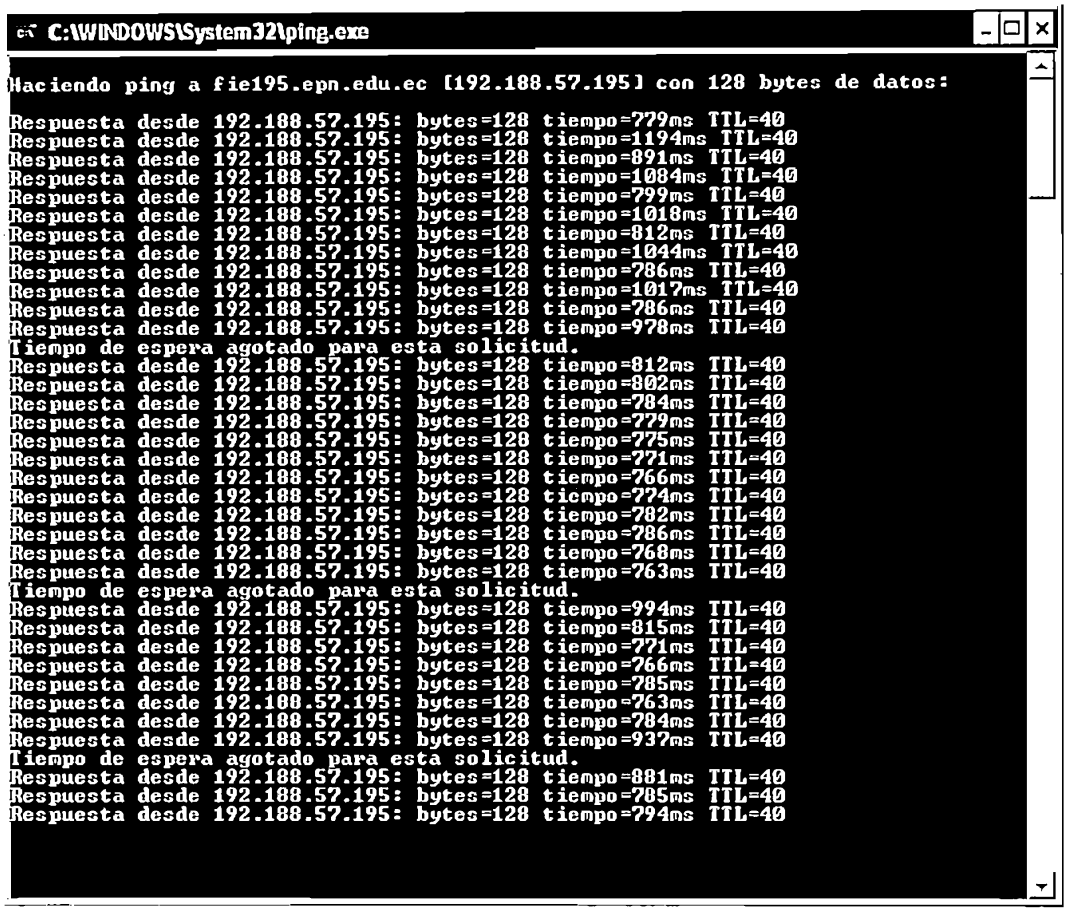


Figura 3.3 Prueba de Ping desde una red diferente

3.3 TIEMPO DE RESPUESTA

Los tiempos de respuesta del servidor Web implementado son bajos cuando se hace conexión desde la misma red, tal como se muestra en la figura 3.2, pero si se establece conexión desde otra red, los tiempos de respuesta son altos e inclusive hay momentos en que el tiempo de espera se agota tal como se puede apreciar en la figura 3.3, esto se debe a sus limitadas capacidades, en lo que a velocidad se refiere, además de las diferentes velocidades de conexión que dependen del servicio que presta el ISP.

3.4 ACCESO SIMULTÁNEO

Al realizar un acceso simultáneo al servidor lo que se desea demostrar es el número de conexiones que el servidor está en capacidad *de soportar*, pero

cuando se hace acceso simultáneo a la página Web que tiene almacenada, la respuesta que se obtiene no es tan cierta, debido a que es muy complicado realizar un acceso exactamente al mismo tiempo desde diferentes máquinas mientras la conexión permanece abierta.

Una vez que la página termina de cargarse, se cierra la conexión, por tanto en estas condiciones no es tan fácil comprobar cuantas conexiones simultáneas está en condiciones de responder, sin embargo si se hacen varios ping, la conexión permanece abierta y esperando la respuesta del servidor, lo que nos da la posibilidad de detectar cuando empieza a tener problemas para satisfacer las solicitudes, en la figura siguiente se tiene el acceso al servidor haciendo la prueba ping simultáneamente con varias conexiones.

En una primera parte se hicieron 9 solicitudes simultáneas, la conexión al momento de satisfacer las 9 solicitudes es aceptable, tuvo muy pocas solicitudes no contestadas y las respuestas que se obtuvieron son las siguientes

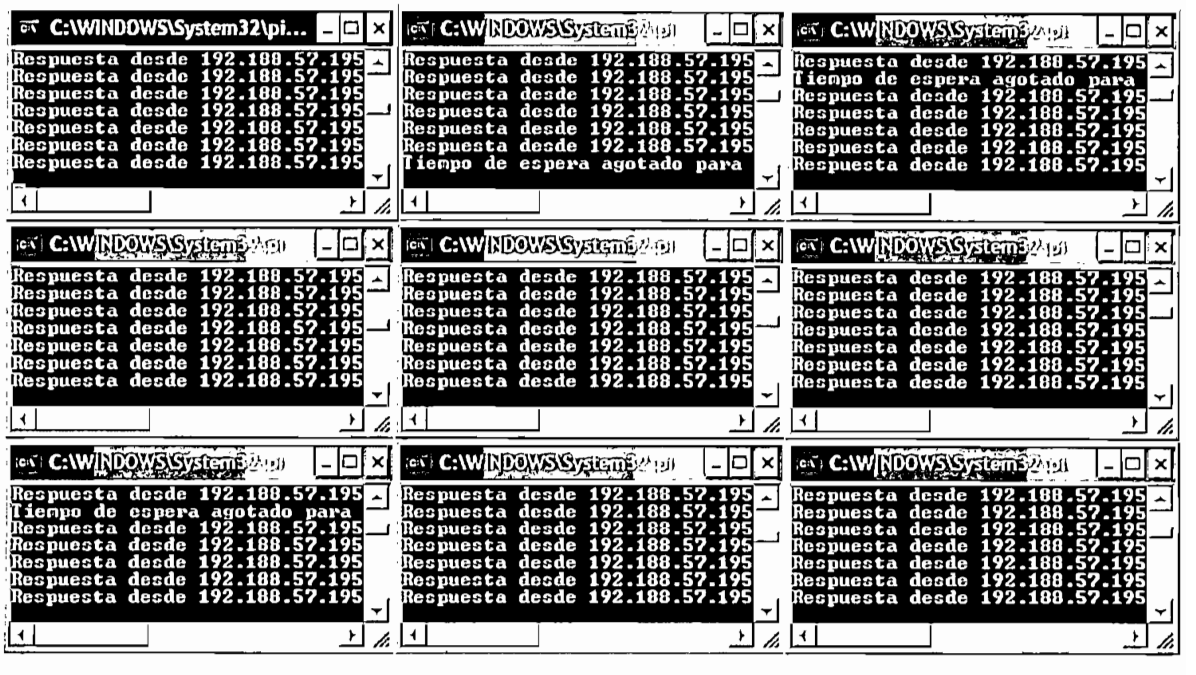


Figura 3.4 9 conexiones simultáneas

En una segunda parte se hicieron 12 solicitudes simultáneas, en ese momento el servidor empezó a presentar problemas para satisfacer a las demandas

realizadas, tal como se muestra en la figura siguiente, el tiempo se espera de solicitudes agotadas se incremento considerablemente.

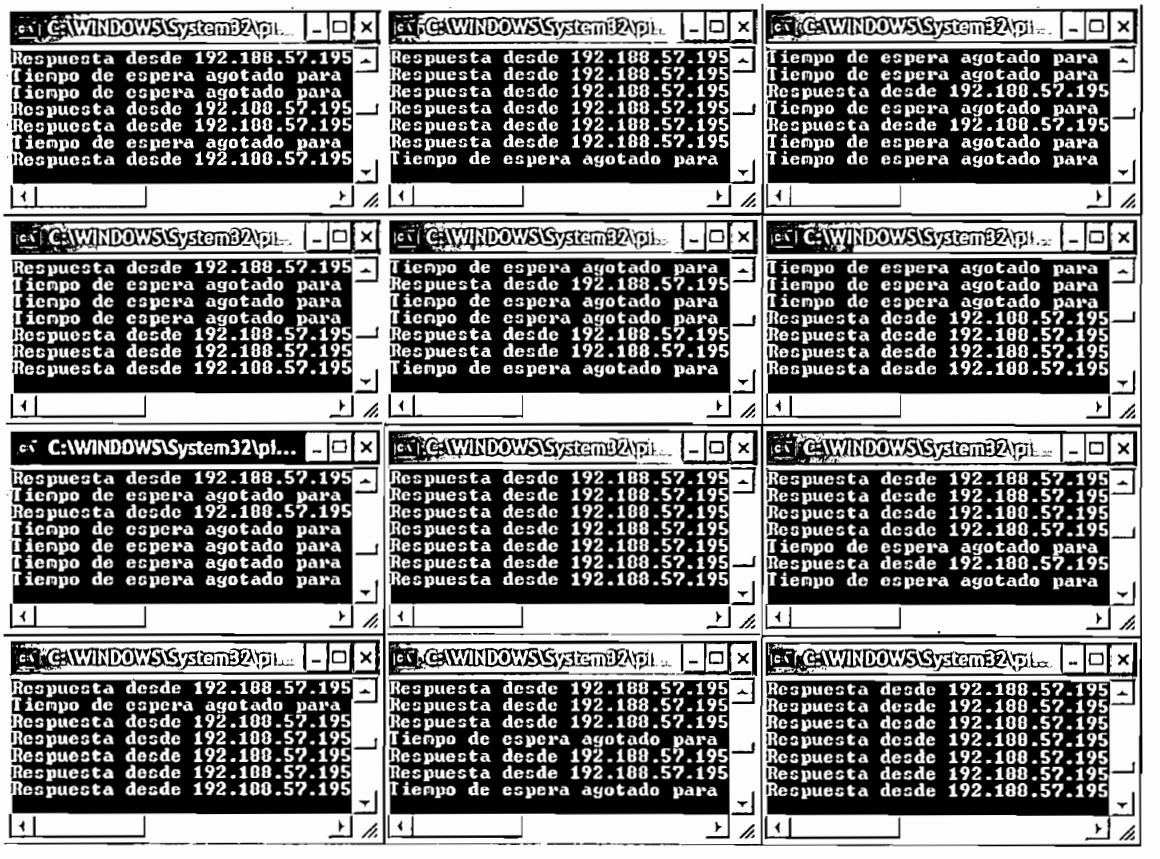


Figura 3.5 12 conexiones simultáneas

3.5 ACCESO DESDE DIFERENTES MAQUINAS EN DIFERENTES REDES

Con la finalidad de comprobar si el prototipo implementado permite el acceso de múltiples usuarios desde cualquier parte, se han hecho varias conexiones desde diferentes máquinas que pertenecen a diferentes redes.

Al establecer la conexión con el servidor, éste ha respondido satisfactoriamente y no presenta ningún problema para satisfacer las demandas. El resultado de esta prueba se lo puede apreciar en las siguientes figuras.

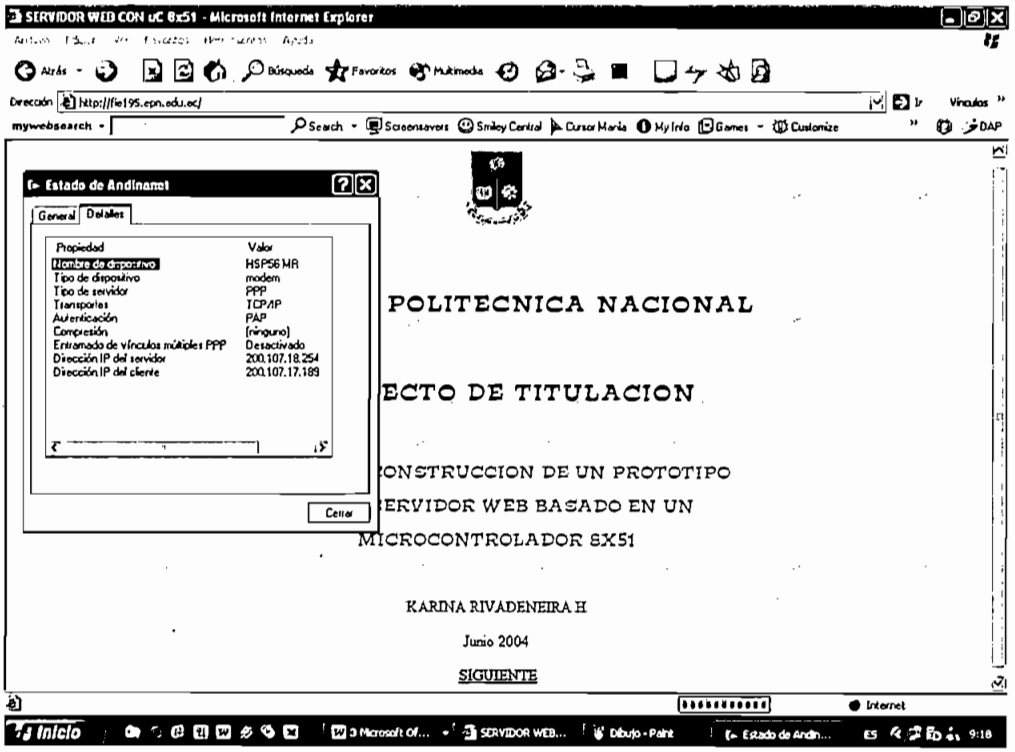


Figura 3.6 Conexión desde una red externa (Andinanet).

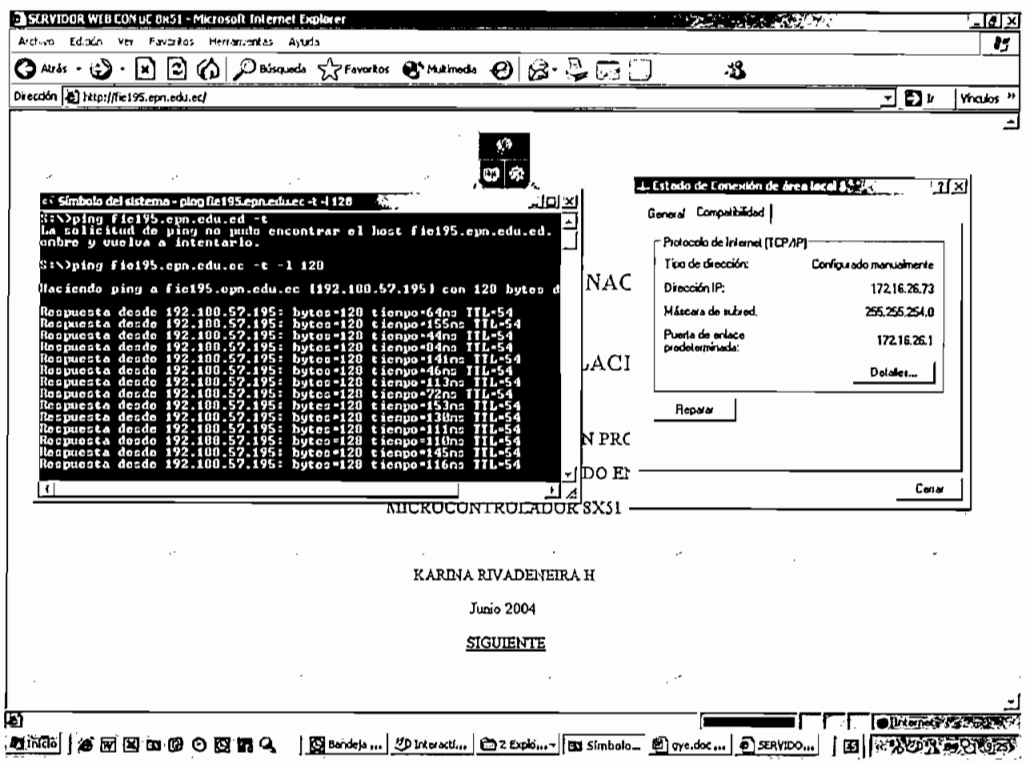


Figura 3.7 Conexión desde una red LAN de una empresa

CAPÍTULO 4

CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- En los últimos años la humanidad ha experimentado uno de los cambios más grandes de su historia con el aparecimiento del Internet, su rápido crecimiento, implementación y desarrollo a tal punto que se ha convertido en el sistema universal de comunicación, publicación de noticias y novedades cambiando notablemente la forma de vida de quienes se interrelacionan con el resto mundo utilizando este nueva forma de comunicación.

- Vivimos en el tiempo de la interacción entre personas que necesitan comunicarse para intercambiar no solo información, conocimiento sino también productos y servicios, compartir experiencias que nos permitan conocer más con menos esfuerzo permitiéndonos ser productivos y competitivos a todo nivel.

- El objetivo de este trabajo es desarrollar un servidor Web que consienta no solo a empresas sino también a personas publicar sus conocimientos, ideas y productos en la "super autopista de la información" con mucha facilidad, permitiéndole de esta manera interactuar con el resto del mundo para obtener más y mejores beneficios en el desempeño de sus actividades.

- Dependiendo del tipo de software que se desarrolle para este tipo de hardware, este no solo puede ser un servidor de páginas Web, también puede ser implementado un servidor de correo, un servidor para intercambio de archivos FTP, un servidor básico TELNET, etc. Las aplicaciones son múltiples.

- Este tipo de servidores con sistemas embebidos están diseñados para adaptarse a las necesidades de todo tipo de usuarios, tanto los que necesitan comunicarse a grandes velocidades utilizando *tecnologías* de red que son diseñadas con este propósito así como también los usuarios que

tan solo disponen de una línea telefónica y de un MODEM analógico para realizar la comunicación y el acceso a Internet.

- Muchos servidores son grandes computadoras que almacenan cientos de archivos y por tanto son mucho más costosas, mientras que un servidor con sistema embebido puede almacenar pequeñas páginas Web o muy pocos archivos, lo cual en cierto modo representa una desventaja, sin embargo existe la posibilidad de que este servidor sea tan solo un administrador de uno o varios dispositivos conectados en red, permitiendo compartir los archivos que tienen almacenados, con lo que fácilmente podría convertirse en un servidor mucho más grande que cumpla con las mismas funciones que una gran computadora pero con un menor costo de implementación.

- En el desarrollo del software existe una parte que es común para cualquier tipo de hardware. Este software común es la implementación de los protocolos para establecer la comunicación, gracias a la utilización de protocolos estandarizados. El uso de estos protocolos hace que la comunicación sea independiente del hardware, sin embargo para el desarrollo del software que determina el tipo de servidor que se está implementando si se debe tomar en cuenta el hardware, ya que las rutinas que se implementan dependerán del tipo de controlador de red y del procesador que se está utilizando.

- El escaso número de direcciones IP representa un problema en Internet. Como solución a este problema se asigna en forma dinámica una dirección IP para el uso compartido por varias computadoras, sin embargo existen muchas soluciones que necesitan una dirección IP estática para su correcto funcionamiento. Un servidor con sistema embebido puede recibir una dirección IP en forma estática o dinámica en caso de que exista un cliente DHCP. La dirección estática se la puede almacenar en la memoria no volátil o dentro de una aplicación y mediante el código de programa se puede recuperar este valor y hacer uso de él cuando el sistema lo necesite

o cuando sea necesario para el correcto funcionamiento de la aplicación desarrollada.

- En una red de área local de medio compartido como es el caso de Ethernet, los servidores en general deben tener enlaces de mayor velocidad que las estaciones de trabajo con la finalidad de eliminar el riesgo de formar cuellos de botella en las conexiones.

- En la actualidad los computadores de propósito general utilizan procesadores de 32 y 64 bits, pero los sistemas embebidos son construidos con un procesador de 8 o 16 bits y la cantidad de memoria en la que se almacena el software también se verá afectado en función del procesador que se escoja.

- Por lo general el software diseñado por un sistema embebido no puede ser utilizado por otro sistema sin que antes se realicen varias modificaciones significativas, esto se debe a la gran variedad de hardware existente ya que las especificaciones varían dependiendo de la aplicación que se desee implementar.

- Una de las diferencias fundamentales entre el software desarrollado para sistemas embebidos y los escritos para otras plataformas de computadoras es que el software de los sistemas embebidos terminan en un lazo infinito y por lo general estos lazos contiene una parte significante de la funcionalidad del programa ya que así se evita realizar el proceso de reset por cada vez que se desee utilizar la aplicación.

- La seguridad en un ambiente de red o de redes es importante y difícil. Es importante pues la información tiene un valor significativo y resulta difícil ya que se debe considerar como y cuando los usuarios y las computadoras pueden confiar uno en el otro, lo que significa que se deben entender los detalles técnicos del hardware y los protocolos de red. En el caso de este servidor Web, la implementación es completamente segura ya que el

software se almacena en la memoria RAM y la única forma de modificar o realizar cambios significativos es mediante una conexión física al servidor, lo que garantiza que los *"jackers"* puedan atacar al servidor.

4.2 RECOMENDACIONES

- Se puede desarrollar una aplicación utilizando esta tarjeta para comunicación inalámbrica cumpliendo con el estándar IEEE 802.11 usando transmisión de radiofrecuencia en la capa física en la banda de 2.4GHz o bien utilizando transmisiones infrarrojas, los dos métodos permiten transmitir datos a 1 o 2Mbps. Si se implementa una comunicación inalámbrica utilizando el estándar IEEE 802.11b con velocidades de transmisión de hasta 11Mbps en la banda de 2.4GHz o el estándar IEEE 802.11a con una velocidad de transmisión de hasta 54Mbps en la banda de 5GHz la vía más rápida de conectar el dispositivo con un interfaz Ethernet para este tipo de redes es utilizando un punto de acceso inalámbrico. Este tipo de sistema embebido puede utilizarse como tal y se puede interconectar con otro interfaz que desea interactuar con la red inalámbrica utilizando un cable.

  Inicialmente la configuración del punto de acceso requiere de una computadora, pero una vez que este ha sido configurado, el administrador de la red puede cambiar la configuración utilizando una página Web para el punto de acceso.

- Se recomienda continuar con el estudio de este tipo de hardware y sobre todo con el desarrollo de posibles aplicaciones como por ejemplo la implementación de un servidor de correo SMTP con la finalidad de permitir a las personas enviar y recibir mensajes sobre una red. El envío de un correo electrónico es una manera muy conveniente para que un sistema embebido intercambie información con otros sistemas o bien se comunique sin intervención humana como ocurre cuando se utiliza un sistema embebido en los sistemas de seguridad ya que se puede programar el

envío de un mensaje cuando ocurre una condición de alarma en forma automática, lo que en la actualidad se conoce como domótica o servicios inteligentes.

- Otra aplicación en la que un sistema embebido es muy útil es la recolección de datos en tiempo real como por ejemplo al implementar un medidor de temperatura que se encargue de tomar datos cada cierto intervalo de tiempo o manualmente a través de una página Web por medio de la cual se puedan manejar los dispositivos conectados a este hardware, de esta forma se podría ejercer control en un cuarto de equipos, en un laboratorio, en un invernadero, etc., en donde el control de las condiciones de temperatura son críticas.

- Se puede también implementar un servidor o un cliente FTP para el intercambio de archivos pero con opciones limitadas debido a la cantidad de memoria disponible ya que los archivos se pueden almacenar únicamente en localidades específicas de memoria.

BIBLIOGRAFÍA

[1] Enciclopedia Microsoft Encarta 2002.

[2] Barreto, Alexis, "ESTUDIO Y ANÁLISIS DE LAS DISTINTAS TECNOLOGÍAS DE ACCESO QUE UN PROVEEDOR DE SERVICIOS DE INTERNET PUEDE IMPLEMENTAR EN ECUADOR", Tesis previa a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones, Escuela Politécnica Nacional, Quito – Ecuador, Diciembre de 1999.

[3] RFC 854, Especificación del Protocolo Telnet, Agosto del 2001.

[4] RFC 959, Protocolo de Transferencia de Ficheros, Febrero 2000.

[5] Comer, Douglas, "REDES GLOBALES DE INFORMACION CON INTERNET Y TCP/IP, PRINCIPIOS BASICOS Y ARQUITECTURA", Editorial Prentice – Hall, Tercera Edición, México, 1996.

[6] Murhammer Martin W., Bretz Stefan, Pugh Larry R., Suzuki Kazunari, Wood David H., "TCP/IP TUTORIAL AND TECHNICAL OVERWIEV", IBM, Sexta Edición, Octubre 1998, www.redbooks.ibm.com

[7] Stallings William, "COMUNICACIONES Y REDES DE COMPUTADORES", Sexta Edición, Prentice – Hall, Madrid – España, 2000.

[8] Tanenbaum Andrew, "REDES DE COMPUTADORAS", Tercera Edición, Prentice – Hall, México, 1997.

[9] Hidalgo Pablo, "FOLLETO DE TELEMATICA", Escuela Politécnica Nacional, 2003.

[10] Beyond Logic, IP & ETHERNET INTERFACES, http://www.beyondlogic.org/etherip/ip.htm, Marzo 2004

[11] Dannenberg, A., MSP430 INTERNET CONNECTIVITY, Texas Instruments, www.texasinstruments.com, Noviembre 2001

[12] Axelson Jan, "DESIGNING AND PROGRAMMING SMALL DEVICES FOR NETWORKING", www.dedicatedteacher.com/estore, 2003

[13] González Vázquez José Adolfo, "INTRODUCCIÓN A LOS MICROCONTROLADORES, Hardware, Software y aplicaciones", Mc Graw – Hill, España 1992.

[14] http://www.web51.hw-server.com

[15] RTL8019AS, Hoja de datos del fabricante, www.connectone.com

[16] P89C51RD2, Hoja de datos del fabricante, www.semiconductors.philips.com

[17] CS8900A, Hoja de datos del fabricante, www.embeddedethernet.com

[18] AT89S8282, Hoja de datos de fabricante, www.atmel.com/atmel/acrobat/doc0401.pdf

[19] RCM3200, Hoja de datos del fabricante, www.keentech.com.cn/product/Rabbit/documents/rcm3200.pdf

[20] AT89C51RD2, Hoja de datos del fabricante, www.atmel.com/dyn/general/tech_doc.asp?doc_id=8970

[21] C8051F005, Hoja de datos del fabricante, www.circuitcellar.com/library/print/0902/brady

[22] LU1T516, Hoja de datos del fabricante, www.siteplayer.com/docs

[23] RTL8201BL, Hoja de datos del fabricante, www.connectone.com

[24] MAX232, Hoja de datos del fabricante, http://pdfserv.maxim-ic.com/arpdf/MAX220-MAX249.pdf

[25] Keil Software, "MACRO ASSEMBLER AND UTILITIES", User's Guide, www.keil.com, Febrero 2001

[26] Keil Software, "Cx51 COMPILER", User's Guide, www.keil.com, Septiembre 2001.

[27] Barr Michael, "Programming Embedded Systems in C and C++", primera edición, Enero, 1999.

[28] 74HC14, Hoja de datos del fabricante, http://www.todopic.com.ar

[29] W3100A, Hoja de datos del fabricante, www.atmel.com/dyn/general

ANEXOS

ANEXO 1

CONSTRUYA SU PROPIO SERVIDOR WEB 8051
CIRCUIT CELLAR (JIM BRADY)

CIRCUIT CELLAR®
THE MAGAZINE FOR COMPUTER APPLICATIONS

FEATURE ARTICLE

Jim Brady

Build Your Own 8051 Web Server

Building your own web server can be a difficult task, especially if you proceed without proper direction and the right parts for the job. Fortunately, Jim has finished an 8051 server and he's eager to walk you through his project. With this tutorial, you can avoid common difficulties.

This article grew out of an experiment to see how hard it would be to build an 8051 web server and write a minimal TCP/IP stack. It seems like everything is serving web pages these days, so why not an 8051? It was not easy, but it was a fun project. After a few months of studying ARP and TCP, I had something up and running.

In this article, I'll explain how I built an 8051 web server and describe what I learned along the way. I'll also discuss timing and performance. If you want to follow along, download the source code from the *Circuit Cellar* ftp site.

COMPONENTS

I wanted an 8051 with enough RAM to hold a full-sized Ethernet frame of 1.5 KB, and with analog inputs so it could do something useful. The Cygnal parts were my first choice. The C8051F005 is fast, and it has a 12-bit A/D converter and 2.4 KB of RAM. The C8051F005's 32-KB flash memory is large enough for a reasonably sized program plus a few web pages. At first I thought its lack of a conventional bus would be a problem, but it turned out to be no problem at all.

The Cygnal 8051 makes up for being just 8 bits with its speedy 25-MIPS peak performance. The Ethernet controller's RAM adds additional buffering capability for incoming frames, which is key for allowing the CPU time to process a frame while more are received. Browsers running on fast machines can easily fire out two or three Ethernet frames within a millisecond!

For the Ethernet controller, I looked at the Realtek RTL8019AS and the Cirrus Logic CS8900A. The former is inexpensive and NE2000-compatible, but I've used many Cirrus parts over the years, so I went with the CS8900A.

The CS8900A's 4 KB of RAM is enough to hold a number of incoming frames. As with any Ethernet controller, the datasheet for it is long and there are many registers to set up. So, I sat down and read through the datasheet to figure out how to talk to it. By looking at a sample driver, which I downloaded from the Cirrus web site, I was able to create an interface in C, compile it to assembly, and then hand-optimize the assembly code.

So that's it, almost everything in two chips. I added an RS-232 port that runs at 115.2 KB for debugging, even though I found Cygnal's full-speed emulator to be more than adequate.

BENCHMARKING

The first thing I did when I got the Cygnal evaluation board was run the trusty sieve benchmark on it. [1] First,

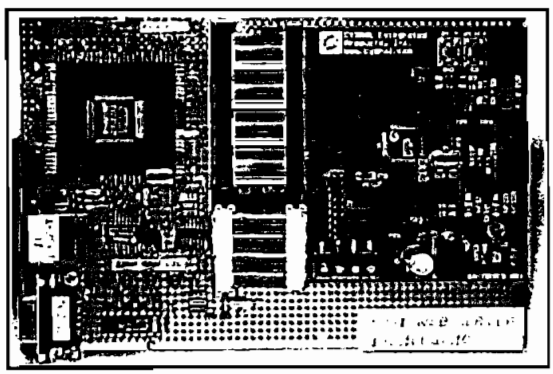


Photo 1—*The Cygnal C8051F005TB sits atop my breadboard at the upper-right. The top ribbon cable connects the 8051 CPU to the CS8900A Ethernet controller while the lower ribbon cable carries analog signals.*

I soldered a 22.1184-MHz crystal onto the board and wrote a function to make the CPU use it instead of the slower on-chip oscillator.

Most of its Cygnal 8051's instructions execute in one or two clocks, as compared to 12 or 24 clocks for standard 8051s. So, I expected good performance, and indeed the C8051F005 runs the sieve benchmark about 19 times faster than a standard 8051. It also ran faster than most 16-bit CPUs I've tested, which is impressive because much of the sieve is 16-bit operations.

To be fair, I should mention that the Cygnal 8051 has the advantage of running entirely out of internal memory. But the results are still representative of what I could expect of the various CPUs in this application. To run the sieve on the C8051F005, I had to scale it down to fit into RAM, and then scale the result to allow comparison to other CPUs. You can see the test results in Table 1.

CPU	Crystal (megahertz)	Sieve—10 loops (seconds)
Cygnal C8051F005	22.1184	0.43
Intel 80C51	11.0952	8.2
Intel 80C196 (16 bit)	18.4320	1.3
Philips XA-S3 (16 bit)	22.1184	1.0

Table 1—*The Cygnal 8051 is much faster than a standard 8051, and even faster than some 16-bit CPUs. The Keil compiler was used for the 8051s, Tasking was used for others, and a large memory model for all.*

BUILDING THE BREADBOARD

Photo 1 shows my breadboard with the Cygnal eval board mounted atop it. The board is part of the Cygnal P/N C8051F005DK package (about $99). It also includes an RS-232-to-JTAG interface to program the 8051's flash memory, IDE, and a full-speed emulator. My breadboard holds the CS8900A, associated Ethernet I/O, thermistor circuit, and RS-232 interface. To hold the 100-lead TQFP CS8900A, I used an RDI/Wainwright solder mount board.

I connected the evaluation board to my breadboard with two ribbon cables, one for analog and the other for digital. In order to keep the ribbon cables short to reduce cross talk, I cut off the prototyping area of the eval board. In addition, I added a 22.1184-MHz crystal and cut the trace that connected the crystal to the I/O connector. I did this to make sure noise could not get into the oscillator.

The Ethernet transformer and associated circuit design in Figure 1 was taken directly from the CS8900A datasheet. The eval board operates from 3.3 V, so I used the 3.3-V version of the CS8900A and the corresponding impedance-matching components for the 10BaseT interface. I kept wire lengths to a minimum in the area between the CS8900A and the RJ45 Ethernet connector.

A thermistor bridge circuit allows the ratiometric measurement of both power supply voltage and temperature. Using a ratio prevents 3.3-V power supply fluctuations from affecting the temperature measurement. After linearizing the thermistor characteristic, the firmware displays temperature on the web page.

ETHERNET I/O

Listing 1 shows the assembly code that reads Ethernet frames from the CS8900A. Less you think this bit-banging approach is inefficient, consider that the Cygnal I/O speed is 40 ns while the maximum access time of the CS8900A is 135 ns. The CS8900A access time imposes the limit, not the 8051. I suspect this method of data transfer is at least as fast as conventional 8-bit bus I/O.

Figure 1 shows the interface between the CPU and Ethernet controller. 8051 port lines P1.0 to P1.2 select the CS8900 address. Only three lines are required because most of the CS8900's registers are indirectly addressed. Pulsing port pins P1.3 and P1.4 generates read and write strobes. 16-bit data is transferred in and out of ports 2 and 3.

My first design used the interrupt output of the CS8900A to interrupt the 8051 when an Ethernet frame arrived. The problem with this is that Cirrus Logic recommends reading everything out of the chip in the interrupt service routine. Because the CS8900A has more RAM than the 8051, I went with polling and only read the most recent frame. This way the CS8900A can queue a number of frames while the 8051 pulls them out

Listing 1—*This code reads an Ethernet frame from the CS8900A. The Cygnal 8051 is so fast that NOPs must be inserted to meet the CS8900A worst-case access time.*

```
;*****************************************************************
; This fragment reads the incoming frame from the CS8900A.
; Call from C as read_frame(UINT xdata * buf, UINT len).
; R6 and R7 point to buf, and the length is in R4 and R5.
;*****************************************************************
RSEG ?PR?_READ_FRAME?CS8900
_READ_FRAME:
MOV   DPL, R7                 //Set up data pointer
MOV   DPH, R6
LOOP:
MOV   P1, #018H               //Take CS8900A CS low, set address
CLR   P1.3                    //Take CS8900A RD strobe low
NOP                           //Allow for CS8900A access time
NOP                           //Each NOP is 45 ns at 22.1184 MHz
NOP
MOV   A, P2                   //Read low byte into port 2
MOVX  @DPTR, A
INC   DPTR
MOV   A, P3                   //Read high byte into port 3
MOVX  @DPTR, A
INC   DPTR                    //Advance data pointer
MOV   P1, #038H               //Take RD strobe, chip select high
DJNZ  R5, LOOP                //Decrement, loop again if needed
DJNZ  R4, LOOP
RET
```

and processes them one at a time. I configured the CS8900A to capture only the frames directed to my MAC address, plus broadcast frames. If the 8051 had to deal with every Ethernet frame on a busy network, it would be in big trouble.

CAN IT BE DONE?

Can a CPU with only 2 KB of RAM really handle large Ethernet messages and the complexities of protocols such as TCP and ARP? Surprisingly, it turns out that even a few hundred bytes would suffice. Small RAM footprints work with TCP because you can tell the other end to limit the message size. TCP can, for example, advertise a maximum segment size of only 100 bytes.

TCP checksums are computed over the entire TCP segment and placed near the beginning of it. This makes it desirable to hold the entire segment in RAM to compute its sum. But here again, if the segment is small, this is not a problem. You can also handle IP processing in a small memory space, as long as you do not try to reassemble fragmented incoming messages. And you do not need to, because the other machine's TCP layer will limit the size of the message it sends. Only UDP will send you large, possibly fragmented messages, and I'm not trying to support that here. To serve a web page, the only other protocol you need to worry about is ARP.

A web server must handle ARP requests because you will receive them when the other end wants to find out your hardware address. You also need to originate your own ARP requests in order to find out the hardware address of the device you are sending to. Of course, you could simply reply back to the same hardware address that sent you the frame, but this would only work for a server (not a client) on a local network. In addition, this would get complicated with multiple simultaneous connections.

ARP is a simple protocol that can cause big problems for a small server. An ARP request must be sent, and a reply received, while a regular message is waiting to be transmitted. One send buffer no longer suffices. You need more space (i.e., enough RAM to hang on to the message-in-waiting), and you must buffer the outgoing and incoming ARP messages. It's a good thing that ARP messages are only 64 bytes long. Figure 2 shows how ARP fits into the flow of things, as well as the flow of an Ethernet frame through the various protocol layers.

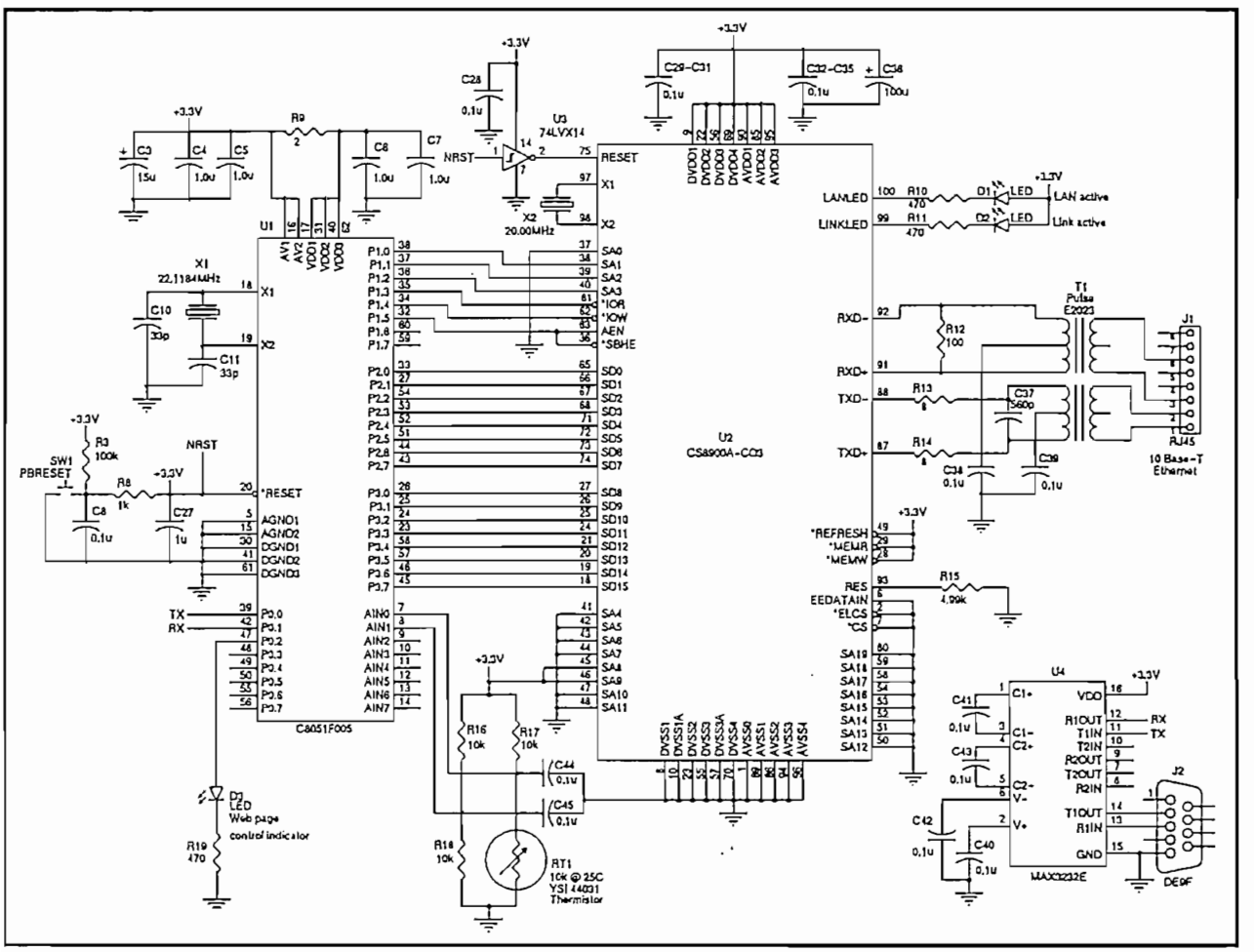


Figure 1—*The left side shows part of the Cygnal C8051F005TB target board. On the right is my CS8900A Ethernet interface, temperature-sensing bridge, and RS-232 port.*

The firmware is written so that the functions in each protocol layer are concerned only with that layer.

WHAT DO YOU NEED?

My web page includes a JPEG image, which the browser loads after the HTML portion. Most of the browsers I tested establish a single connection to load both parts of the page. Netscape 4.7, however, opens two separate connections to the server, and the loading of the HTML page and image overlap to some extent.

Each connection comes from a different port on the browser's machine. To handle this situation, I put all connection-specific information, such as client IP address, client port number, sequence number, ack number, and TCP state into a structure that's indexed by the connection number. Each element of this structure can be thought of as a connection.

When a TCP segment arrives, I check to see if it matches an existing connection and use the state information for that connection. That allows the web server to handle simultaneous connections from the same PC or from multiple PCs.

Beyond ARP, IP, and a TCP state machine to handle simultaneous connections, what do you need, and what can you safely leave out? TCP has a long list of goodies, such as algorithms to estimate the best time-out, avoid congestion, assemble out-of-order segments, and so on. I made a list of about 20 such items. The truth is that I am serving web pages to various browsers and multiple PCs simultaneously, but I've implemented only a few of these capabilities. So, I am content for now, and I have a 20-item to-do list for a rainy day.

EMBEDDED WEB PAGES

My program uses 22 KB out of the 32-KB on-chip flash memory, leaving enough room for my 7-KB web page. Access to on-chip flash memory is fast. For more storage, the obvious choice would be an SPI serial EEPROM. 64-KB devices are inexpensive, and they can be clocked in the 2- to 5-MHz range, depending on the part. Using the built-in SPI port on the C8051F005, set to provide a 2-MHz SPI clock, my 7-KB web page could be transferred into the 8051 in about 30 ms. This would add about 50% to the time needed to serve the page.

Web pages for embedded systems need capabilities that simple static web pages do not, such as dynamic data and two-way capability. Dynamic data is handled by inserting a tag in HTML. A tag number is included to tell the server what variable to insert. Two-way communications is provided using a form, which is a standard HTML construct that allows the browser to send selections back to the web server.

One purpose of a small web server like this is to make a product easy to use. A little eye candy is nice, and web pages for embedded systems would do well to have one or more images to make them attractive.

Photo 2 shows my web page, as presented on an MSIE browser. This page is bidirectional in that it both displays data and has radio buttons to turn an LED on the breadboard on and off. The state of the buttons is sent to the web server in a post message and can be easily extracted. The 0.8-KB HTML portion of the page and 6.2-KB JPEG graphic combine for a total of 7 KB. This page is used as the basis for comparisons later on in the article.

RUN-TIME PROFILING

Web pages are for human consumption, and 100-ms response times appear

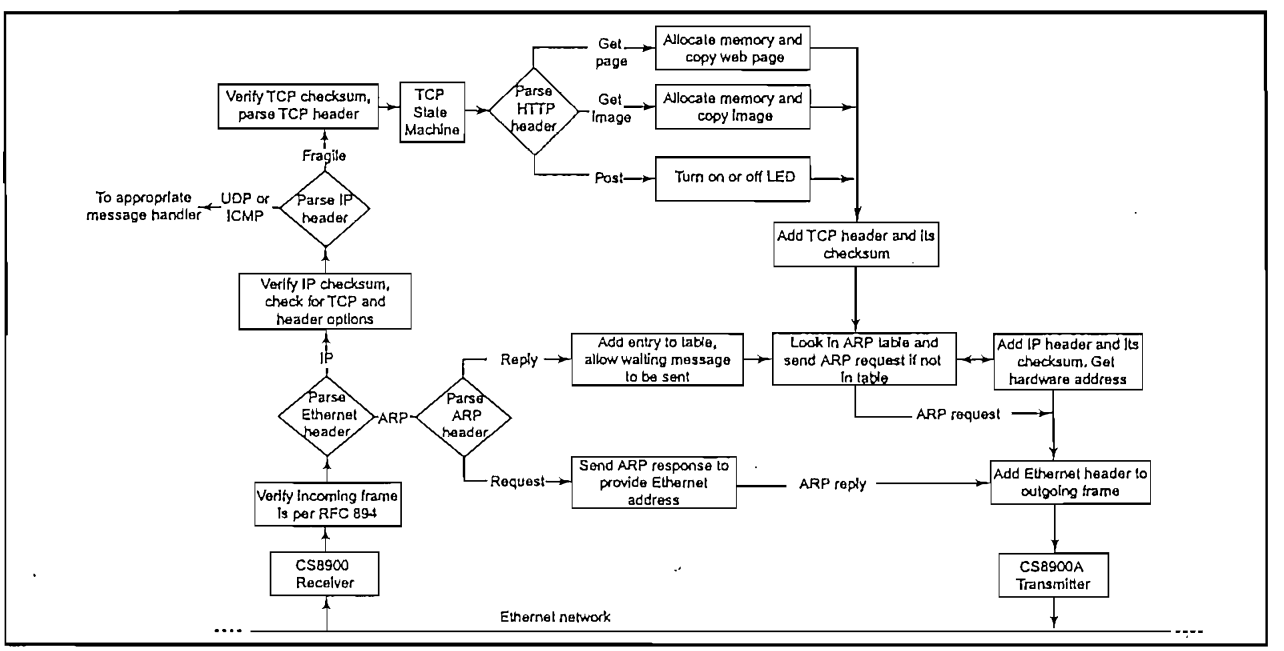


Figure 2—*As a message goes up the left side of the flow chart, two checksums are verified before reaching the TCP state machine. Web page requests generate an HTML page or image, and then two checksums are added going down the right side. ARP message flow is shown in the central portion of the chart.*

Task	Times run	Accumulated time (milliseconds)
Search and replace tags	6	23.8
Copy buffers	8	9.1
Write to CS8900A	11	4.6
Parse incoming HTTP headers	2	4.5
Compute checksums	38	4.4
Read from CS8900A	8	0.6
TCP state machine	8	0.4
Total time		47.4

Table 2—*As you can see, searching for and replacing tags consumes the most time. To serve a single web page, 19 frames are sent and received.*

snappy. Using the Finisar (formerly Shomiti; go to www.finisar.com for more information) Surveyor Lite, I measured the time required for my 8051 web server to serve the 7-KB web page to a 500-MHz Pentium machine running MSIE 5.5. [2] It took 60 ms. In contrast, it takes my 100-MHz Pentium server running Apache about 32 ms to serve the same page. This demonstrates that the response of the 8051 is respectable. For these measurements, I had to clear the browser's page cache each time to make sure my browser was actually transferring the web page rather than just displaying a cached version.

To figure out how much time my web server spent doing various tasks, I added debug code that set a port pin when the CPU began the task and cleared the pin when it completed the task. Because many of the tasks I was interested in run a number of times while serving the web page, I had to add up all the pulse on times. This was hard to do with a oscilloscope, so I used an 82C54 timer chip. The 80C51 port pin output drives the 82C54 gate input. When in the high state, the 82C54 counts transitions from a 1-MHz oscillator. This provides accumulated pulse-width times with 1-µs resolution. I set up another 82C54 counter to count the number of times an event ran. Run times are summarized in Table 2.

The total of 47.4 ms falls short of the 60 ms it takes to transfer the page. When I added up the intervals between the 8051 sending an Ethernet frame and the browser's 500-MHz Pentium responding, I came up with 8.5 ms. This accounts for most of the difference. It's mind boggling, but true, that the 8051 is waiting for a Pentium.

Searching for and replacing tags is the most time-consuming task. My web page uses tags as placeholders for dynamic values, such as temperature. When it serves the page, it searches for these tags and then replaces them with the appropriate value.

It turns out that the `strstr()` function is the time hog. After some investigation, I found this to be true in general of `strstr()`. This makes sense because it has to parse through a lot of text, comparing each letter of the text to the corresponding letter of the search string. It may have many partial matches before it finally finds a complete match. One way to speed up the process would be to tightly limit the search range of `strstr()`. Another approach would be to keep an index of offsets to the tags, but the index would need to be changed each time a page was added or modified.

The second most time-consuming task is copying the web page from flash memory to RAM, using `memcpy()`. Why not just skip this step and copy directly from flash memory to the CS8900A? Again, the tags are the problem; they need to be replaced with actual values, and you can't replace them while in flash memory. Perhaps a faster approach would be to copy directly from flash memory to the CS8900A, looking for tags as you copy. But then you would have a thorny problem with the TCP checksum. It's computed over the entire segment, but must be inserted at the beginning of the segment.

It's interesting to note that the checksum is computed a whopping 38 times to transfer a single web page. This transfer is made up of 19 Ethernet frames, 11 from my 8051 server and eight from the browser. It takes three frames to establish the connection, two frames to transfer the HTML page, eight frames to transfer the image, and six frames that are just acks. For both incoming as well as outgoing frames, two checksums are computed: one for the IP header and the other for the TCP segment, which makes 38 checksums. I was glad I used assembler for the checksum code!

I can't help but wonder how much a 16-bit CPU would speed things up, just by virtue of its being 16 bits. The checksum would certainly run faster because the sum is done over 16-bit chunks. Also, CS8900A I/O is 16 bits. Other tasks, such as memcpy() and strstr(), may need custom library code, because many 16-bit compilers default to doing these operations 1 byte at a time.

Description	Code space
TCP/IP	9.5 KB
Web page including image	7.0 KB
HTTP server	3.8 KB
ARP	2.5 KB
C Library	2.9 KB
UDP	1.4 KB
CS8900A I/O	1.0 KB
RS-232	0.5 KB
Analog	0.3 KB
Priority task switcher	0.3 KB
Total	29.2 KB

Table 3—*Here you can see the footprints of various parts of the code.*

MEMORY USAGE

Of the 2.4 KB of RAM on the 8051, the lower 256 bytes are used for frequently accessed variables. The 2048-byte area of additional on-chip memory is addressed as XDATA memory. Incoming and outgoing message buffers are dynamically allocated from this space. Dynamic allocation is unusual for an 8-bit CPU, but it makes sense here because sometimes the incoming frame is large and the outgoing frame is small (occasionally, it can also be the other way around). At other times, the outgoing frame must be held in memory while an ARP message is sent and received. The firmware uses dynamic allocation using the library functions `malloc()` and `free()`, provided with the Keil C compiler. With this approach, no more RAM is tied up handing Ethernet frames than there needs to be at any given moment.

When it was all said and done, I had consumed 29 KB of the 32-KB flash memory. This reminds me of some-

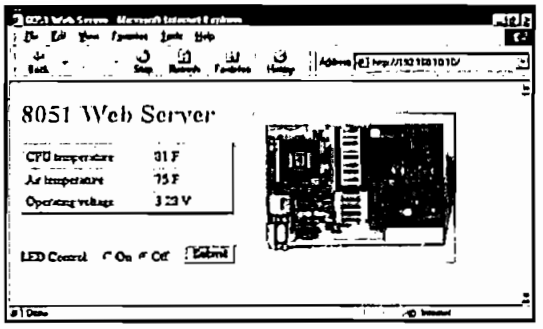


Photo 2—*Seeing is believing! The web page and image served by the 8051 can be seen in the browser window. The on/off buttons control the state of an LED on the breadboard.*

thing I heard once about projects expanding to fill available space. The details are shown in Table 3. The TCP/IP portion is small as stacks go, but remember that I have implemented only a subset of TCP/IP here.

FUTURE DIRECTIONS

All things considered, I'm glad that I picked a part with 2.4 KB of RAM. A CPU with a few hundred bytes could do the job, but I wouldn't go through the trouble. It would be interesting to port the code to a 16-bit DSP chip and compare performance to this fast 8051. Most DSPs have enough RAM. It would also be nice to have a modular-size TCP/IP stack and create an API like the big boys have. This will surely require more than 32 KB, but lo and behold, 64-KB flash memory 8051s are already here.

Jim Brady is an embedded systems engineer living in southern Oregon. He has 20 years of experience designing with microcontrollers and device networks. You may reach him at jimbrady@aol.com

SOFTWARE

To download the source code, go to ftp.circuitcellar.com/pub/Circuit_Cellar/2002/146/.

REFERENCES

[1] J. Gilbreath and G. Gilbreath, "Eratosthenes Revisited: Once More Through the Sieve," *BYTE*, January 1983.

[2] E. A. Hall, *Internet Core Protocols*, O'Reilly & Associates, Inc., Sebastopol, CA, February 2000.

SOURCES

CS8900A Ethernet controller
Cirrus Logic, Inc.
www.cirrus.com

C8051F005 Mixed signal MCU
Cygnal Integrated Products, Inc.
www.cygnal.com

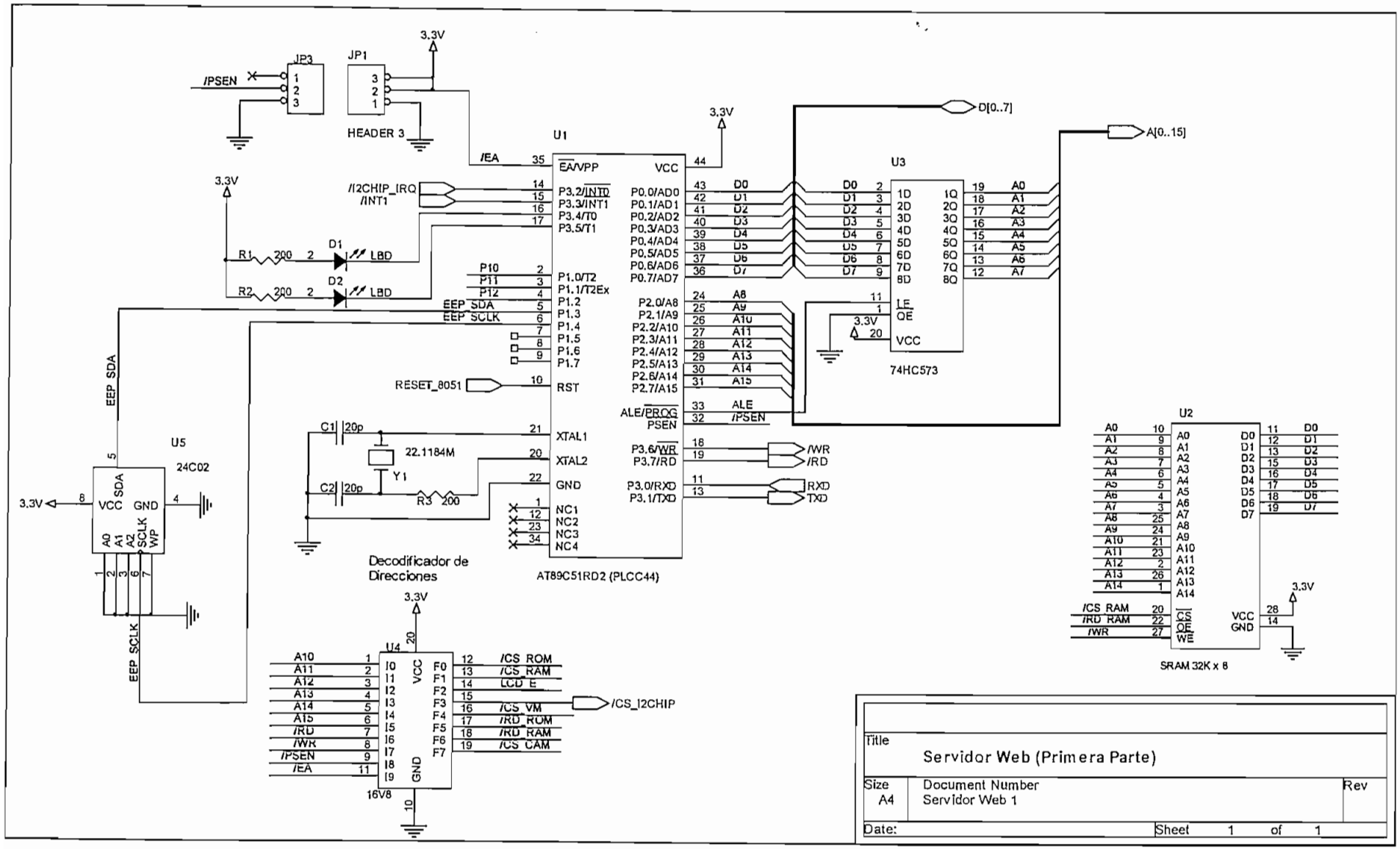
C compiler
Keil Software, Inc.
www.keil.com

RDI/Wainwright solder mount board
RDI/Wainwright
www.rdi-wainwright.com

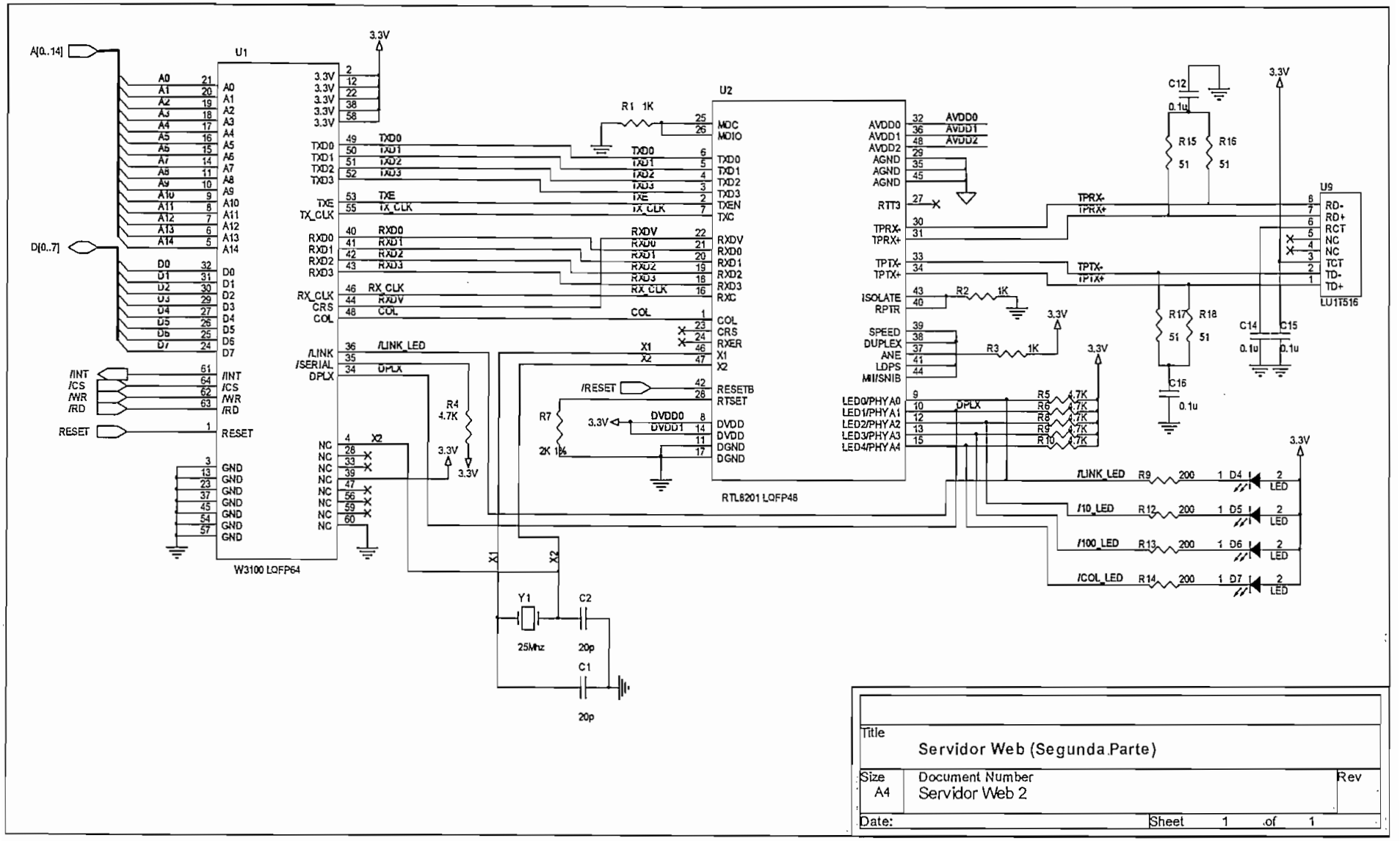
RTL8019AS
Realtek Semiconductor Corp.
www.realtek.com.tw

ANEXO 2

ESQUEMÁTICOS DEL PROTOTIPO



Title	Servidor Web (Primera Parte)	
Size A4	Document Number Servidor Web 1	Rev
Date:	Sheet 1 of 1	


A[0..14]
D[0..7]
U1
W3100 LQFP64
U2
RTL8201 LQFP48
U9
LU1T516
3.3V
/INT
/CS
/WR
/RD
RESET
/LINK_LED
/10_LED
/100_LED
/COL_LED
Y1
25Mhz
20p
C1
C2
R1 1K
R2 1K
R3 1K
R4 4.7K
R5 4.7K
R6 4.7K
R7 2K 1%
R8 4.7K
R9 4.7K
R10 4.7K
R12 200
R13 200
R14 200
R15 51
R16 51
R17 51
R18 51
C12 0.1u
C14 0.1u
C15 0.1u
C16 0.1u
Title
Servidor Web (Segunda Parte)
Size
A4
Document Number
Servidor Web 2
Rev
Date:
Sheet 1 of 1

ANEXO 3

HOJA DE DATOS DEL MICROCONTROLADOR

AT89C51RD2

Features

- 80C52 Compatible
  - 8051 Instruction Compatible
  - Six 8-bit I/O Ports (64 Pins or 68 Pins Versions)
  - Four 8-bit I/O Ports (44 Pins Version)
  - Three 16-bit Timer/Counters
  - 256 Bytes Scratch Pad RAM
  - 9 Interrupt Sources with 4 Priority Levels
- Integrated Power Monitor (POR/PFD) to Supervise Internal Power Supply
- ISP (In-System Programming) Using Standard V_{CC} Power Supply
- Boot ROM Contains Low Level Flash Programming Routines and a Default Serial Loader
- High-speed Architecture
  - 40 MHz in Standard Mode
  - 20 MHz in X2 Mode (6 Clocks/Machine Cycle)
- 64K Bytes On-chip Flash Program/Data Memory
  - Byte and Page (128 Bytes) Erase and Write
  - 100k Write Cycles
- On-chip 1792 bytes Expanded RAM (XRAM)
  - Software Selectable Size (0, 256, 512, 768, 1024, 1792 Bytes)
  - 768 Bytes Selected at Reset for T89C51RD2 Compatibility
- On-chip 2048 Bytes EEPROM Block for Data Storage (AT89C51ED2 Only)
  - 100K Write Cycles
- Dual Data Pointer
- Variable Length MOVX for Slow RAM/Peripherals
- Improved X2 Mode with Independent Selection for CPU and Each Peripheral
- Keyboard Interrupt Interface on Port 1
- SPI Interface (Master/Slave Mode)
- 8-bit Clock Prescaler
- 16-bit Programmable Counter Array
  - High Speed Output
  - Compare/Capture
  - Pulse Width Modulator
  - Watchdog Timer Capabilities
- Asynchronous Port Reset
- Full-duplex Enhanced UART with Dedicated Internal Baud Rate Generator
- Low EMI (Inhibit ALE)
- Hardware Watchdog Timer (One-time Enabled with Reset-Out), Power-off Flag
- Power Control Modes: Idle Mode, Power-down Mode
- Single Range Power Supply: 2.7V to 5.5V
- Industrial Temperature Range (-40 to +85°C)
- Packages: PLCC44, VQFP44, PLCC68, VQFP64

Description

AT89C51RD2/ED2 is high performance CMOS Flash version of the 80C51 CMOS single chip 8-bit microcontroller. It contains a 64-Kbyte Flash memory block for code and for data.

The 64-Kbytes Flash memory can be programmed either in parallel mode or in serial mode with the ISP capability or with software. The programming voltage is internally generated from the standard V_{CC} pin.

The AT89C51RD2/ED2 retains all of the features of the Atmel 80C52 with 256 bytes of internal RAM, a 9-source 4-level interrupt controller and three timer/counters.

The AT89C51ED2 provides 2048 bytes of EEPROM for nonvolatile data storage.

8-bit Flash Microcontroller

AT89C51RD2
AT89C51ED2

Preliminary

4235A–8051–04/03

In addition, the AT89C51RD2/ED2 has a Programmable Counter Array, an XRAM of 1792 bytes, a Hardware Watchdog Timer, SPI interface, Keyboard, a more versatile serial channel that facilitates multiprocessor communication (EUART) and a speed improvement mechanism (X2 Mode).

The fully static design of the AT89C51RD2/ED2 allows to reduce system power consumption by bringing the clock frequency down to any value, including DC, without loss of data.

The AT89C51RD2/ED2 has 2 software-selectable modes of reduced activity and an 8-bit clock prescaler for further reduction in power consumption. In the Idle mode the CPU is frozen while the peripherals and the interrupt system are still operating. In the Power-down mode the RAM is saved and all other functions are inoperative.

The added features of the AT89C51RD2/ED2 make it more powerful for applications that need pulse width modulation, high speed I/O and counting capabilities such as alarms, motor control, corded phones, and smart card readers.

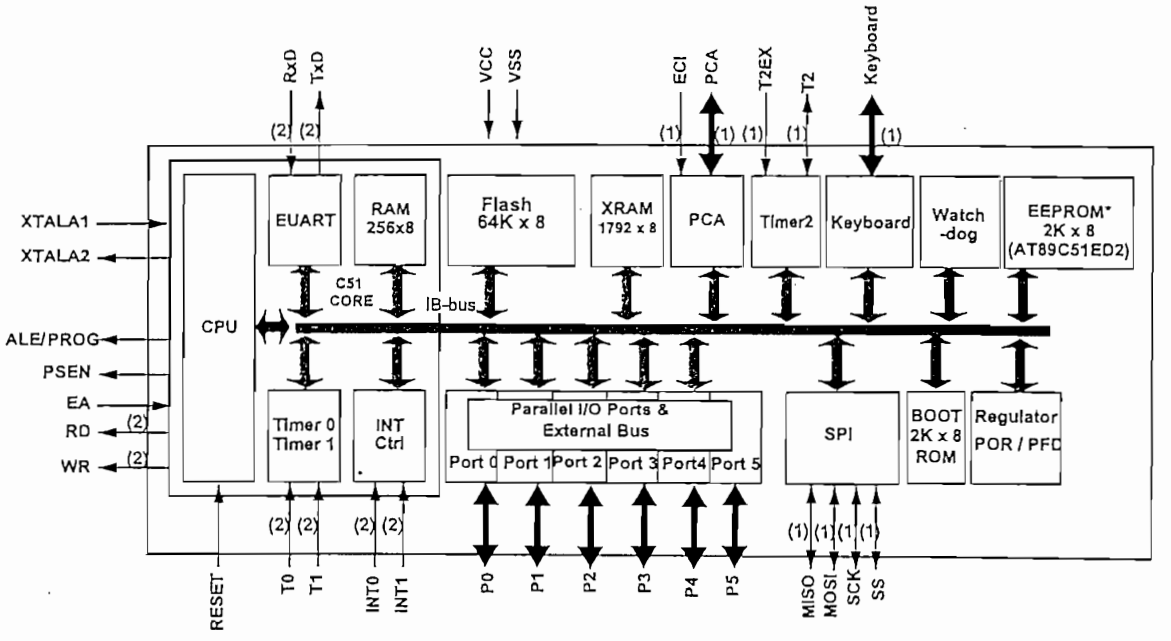
Table 1. Memory Size and I/O Pins

Package	Flash (Bytes)	XRAM (Bytes)	Total RAM (Bytes)	I/O
PLCC44/VQFP44	64K	1792	2048	34
PLCC68/VQFP64[1]	64K	1792	2048	50

Note: 1. For PLCC68 and VQFP64 packages, please contact Atmel sales office for availability.

Block Diagram

Figure 1. Block Diagram



(1): Alternate function of Port 1

(2): Alternate function of Port 3

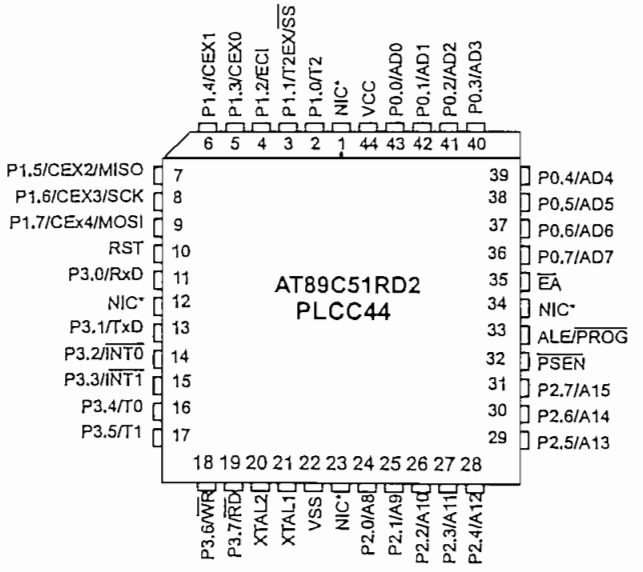
SFR Mapping

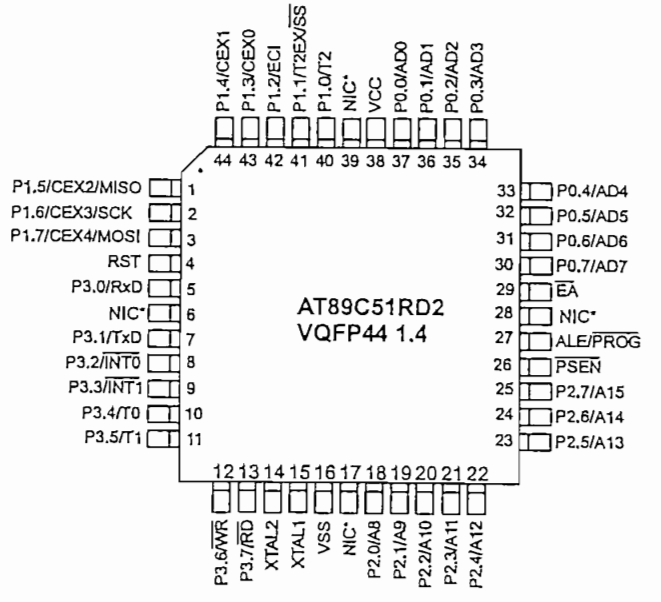
The Special Function Registers (SFRs) of the AT89C51RD2/ED2 fall into the following categories:

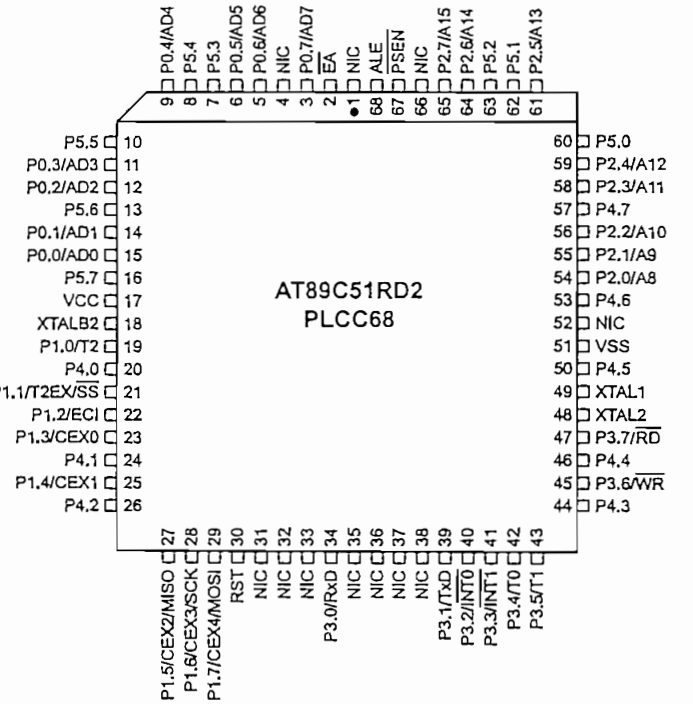
- C51 core registers: ACC, B, DPH, DPL, PSW, SP
- I/O port registers: P0, P1, P2, P3, PI2
- Timer registers: T2CON, T2MOD, TCON, TH0, TH1, TH2, TMOD, TL0, TL1, TL2, RCAP2L, RCAP2H
- Serial I/O port registers: SADDR, SADEN, SBUF, SCON
- PCA (Programmable Counter Array) registers: CCON, CCAPMx, CL, CH, CCAPxH, CCAPxL (x: 0 to 4)
- Power and clock control registers: PCON
- Hardware Watchdog Timer registers: WDTRST, WDTPRG
- Interrupt system registers: IE0, IPL0, IPH0, IE1, IPL1, IPH1
- Keyboard Interface registers: KBE, KBF, KBLS
- SPI registers: SPCON, SPSTR, SPDAT
- BRG (Baud Rate Generator) registers: BRL, BDRCON
- Clock Prescaler register: CKRL
- Others: AUXR, AUXR1, CKCON0, CKCON1

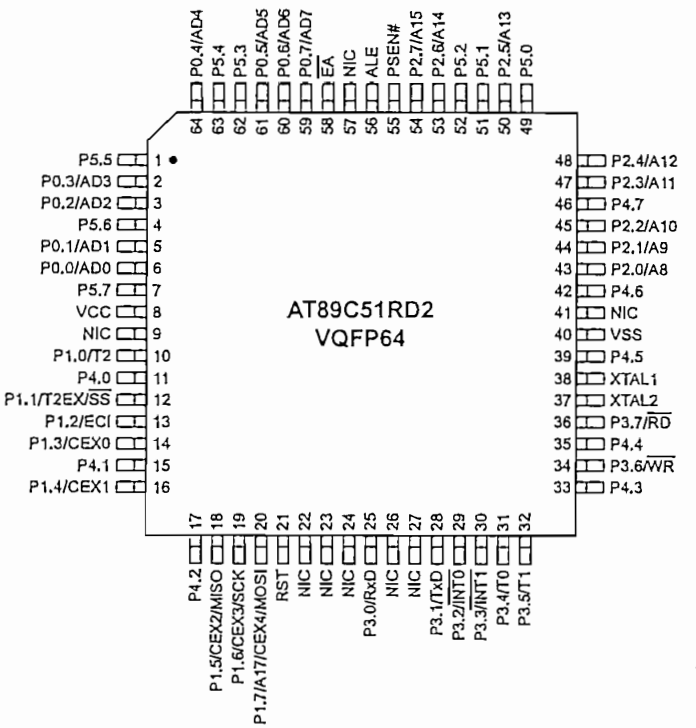
Pin Configurations

Figure 2. Pin Configurations









NIC: Not Internaly Connected

Table 13. Pin Description

Mnemonic	Pin Number PLCC44	VQFP44	PLCC68	VQFP64	Type	Name and Function
V_{SS}	22	16	51	40	I	Ground: 0V reference
V_{CC}	44	38	17	8	I	Power Supply: This is the power supply voltage for normal, idle and power-down operation
P0.0 - P0.7	43 - 36	37 - 30	15, 14, 12, 11, 9,6, 5, 3	6, 5, 3, 2, 64, 61,60,59	I/O	Port 0: Port 0 is an open-drain, bi-directional I/O port. Port 0 pins that have 1s written to them float and can be used as high impedance inputs. Port 0 must be polarized to V_{CC} or V_{SS} in order to prevent any parasitic current consumption. Port 0 is also the multiplexed low-order address and data bus during access to external program and data memory. In this application, it uses strong internal pull-up when emitting 1s. Port 0 also inputs the code bytes during EPROM programming. External pull-ups are required during program verification during which P0 outputs the code bytes.
P1.0 - P1.7	2 - 9	40 - 44 1 - 3	19, 21, 22, 23, 25, 27, 28, 29	10, 12, 13, 14, 16, 18, 19, 20	I/O	Port 1: Port 1 is an 8-bit bi-directional I/O port with internal pull-ups. Port 1 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally pulled low will source current because of the internal pull-ups. Port 1 also receives the low-order address byte during memory programming and verification.
						Alternate functions for AT89C51RD2/ED2 Port 1 include:
	2	40	19	10	I/O	P1.0: Input/Output
					I/O	T2 (P1.0): Timer/Counter 2 external count input/Clockout
	3	41	21	12	I/O	P1.1: Input/Output
					I	T2EX: Timer/Counter 2 Reload/Capture/Direction Control
					I	$\overline{SS}$: SPI Slave Select
	4	42	22	13	I/O	P1.2: Input/Output
					I	ECI: External Clock for the PCA
	5	43	23	14	I/O	P1.3: Input/Output
					I/O	CEX0: Capture/Compare External I/O for PCA module 0
	6	44	25	16	I/O	P1.4: Input/Output
					I/O	CEX1: Capture/Compare External I/O for PCA module 1
	7	1	27	18	I/O	P1.5: Input/Output
					I/O	CEX2: Capture/Compare External I/O for PCA module 2
					I/O	MISO: SPI Master Input Slave Output line
						When SPI is in master mode, MISO receives data from the slave peripheral. When SPI is in slave mode, MISO outputs data to the master controller.
	8	2	28	19	I/O	P1.6: Input/Output
					I/O	CEX3: Capture/Compare External I/O for PCA module 3
					I/O	SCK: SPI Serial Clock
	9	3	29	20	I/O	P1.7: Input/Output:

Table 13. Pin Description (Continued)

Mnemonic	Pin Number				Type	Name and Function
	PLCC44	VQFP44	PLCC68	VQFP64		
					I/O	**CEX4:** Capture/Compare External I/O for PCA module 4
					I/O	**MOSI:** SPI Master Output Slave Input line When SPI is in master mode, MOSI outputs data to the slave peripheral. When SPI is in slave mode, MOSI receives data from the master controller.
XTALA1	21	15	49	38	I	**XTALA 1:** Input to the inverting oscillator amplifier and input to the internal clock generator circuits.
XTALA2	20	14	48	37	O	**XTALA 2:** Output from the inverting oscillator amplifier
P2.0 - P2.7	24 - 31	18 - 25	54, 55, 56, 58, 59, 61, 64, 65	43, 44, 45, 47, 48, 50, 53, 54	I/O	**Port 2:** Port 2 is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally pulled low will source current because of the internal pull-ups. Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @Ri), port 2 emits the contents of the P2 SFR.
P3.0 - P3.7	11, 13 - 19	5, 7 - 13	34, 39, 40, 41, 42, 43, 45, 47	25, 28, 29, 30, 31, 32, 34, 36	I/O	**Port 3:** Port 3 is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current because of the internal pull-ups. Port 3 also serves the special features of the 80C51 family, as listed below.
	11	5	34	25	I	**RXD (P3.0):** Serial input port
	13	7	39	28	O	**TXD (P3.1):** Serial output port
	14	8	40	29	I	$\overline{\text{INT0}}$ **(P3.2):** External interrupt 0
	15	9	41	30	I	$\overline{\text{INT1}}$ **(P3.3):** External interrupt 1
	16	10	42	31	I	**T0 (P3.4):** Timer 0 external input
	17	11	43	32	I	**T1 (P3.5):** Timer 1 external input
	18	12	45	34	O	$\overline{\text{WR}}$ **(P3.6):** External data memory write strobe
	19	13	47	36	O	$\overline{\text{RD}}$ **(P3.7):** External data memory read strobe
P4.0 - P4.7	-	-	20, 24, 26, 44, 46, 50, 53, 57	11, 15, 17,33, 35,39, 42, 46	I/O	**Port 4:** Port 4 is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current because of the internal pull-ups.
P5.0 - P5.7	-	-	60, 62, 63, 7, 8, 10, 13, 16	49, 51, 52, 62, 63, 1, 4, 7	I/O	**Port 5:** Port 5 is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1s written to them are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally pulled low will source current because of the internal pull-ups.
RST	10	4	30	21	I	**Reset:** A high on this pin for two machine cycles while the oscillator is running, resets the device. An internal diffused resistor to V_{SS} permits a power-on reset using only an external capacitor to V_{CC}. This pin is an output when the hardware watchdog forces a system reset.

Table 13. Pin Description (Continued)

Mnemonic	Pin Number				Type	Name and Function
	PLCC44	VQFP44	PLCC68	VQFP64		
ALE/$\overline{PROG}$	33	27	68	56	O (I)	**Address Latch Enable/Program Pulse:** Output pulse for latching the low byte of the address during an access to external memory. In normal operation, ALE is emitted at a constant rate of 1/6 (1/3 in X2 mode) the oscillator frequency, and can be used for external timing or clocking. Note that one ALE pulse is skipped during each access to external data memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming. ALE can be disabled by setting SFR's AUXR.0 bit. With this bit set, ALE will be inactive during internal fetches.
PSEN	32	26	67	55	O	**Program Strobe ENable:** The read strobe to external program memory. When executing code from the external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory. $\overline{PSEN}$ is not activated during fetches from internal program memory.
EA	35	29	2	58	I	**External Access Enable:** $\overline{EA}$ must be externally held low to enable the device to fetch code from external program memory locations 0000H to FFFFH. If security level 1 is programmed, $\overline{EA}$ will be internally latched on Reset.

ANEXO 4

HOJA DE DATOS DEL CONTROLADOR

DE RED RTL8201BL

1. Features

The Realtek RTL8201BL is a Fast Ethernet Phyceiver with selectable MII or SNI interface to the MAC chip. It provides the following features:

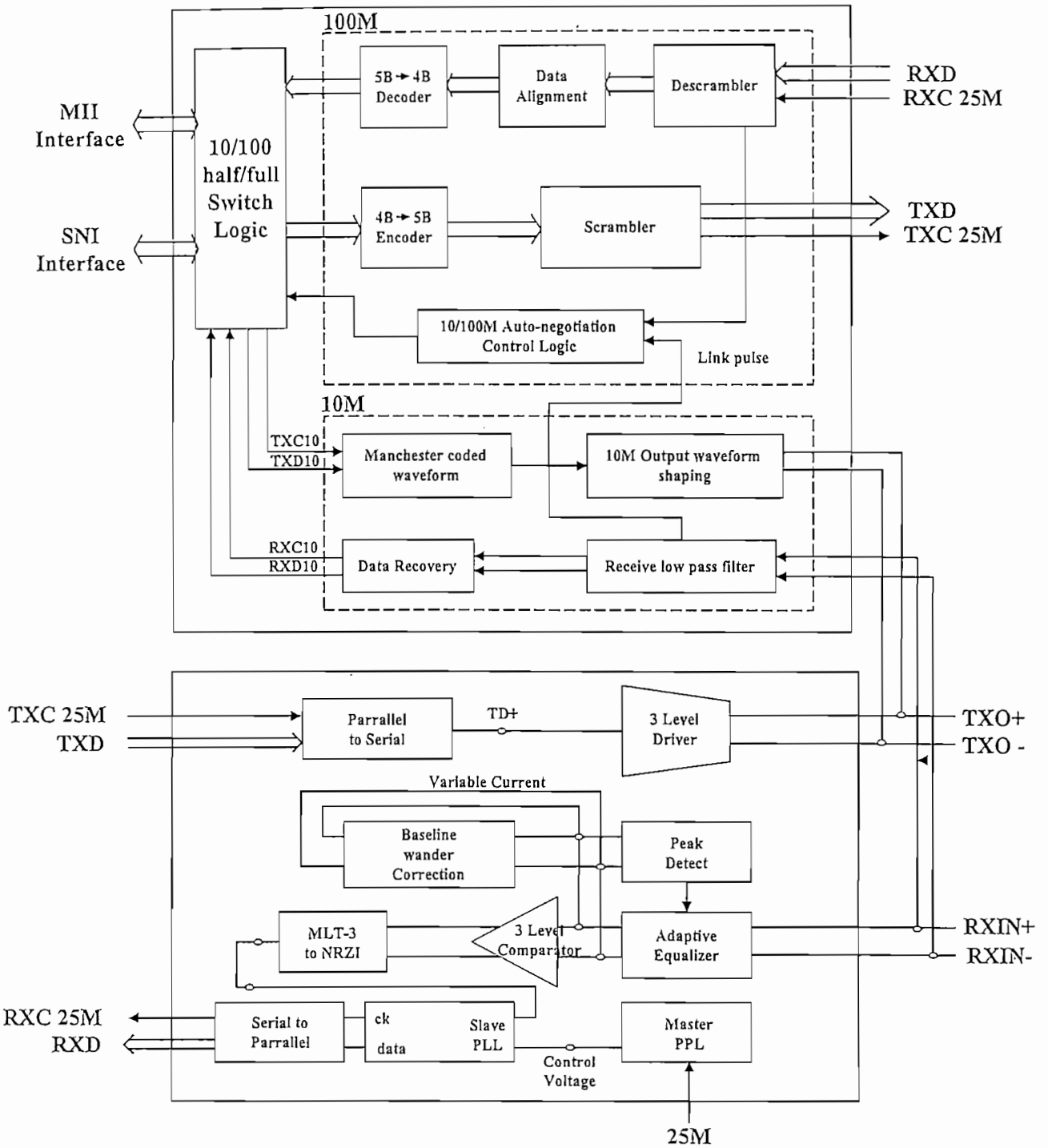
- Supports MII/7-wire SNI (Serial Network Interface) interface
- Supports 10/100Mbps operation
- Supports half/full duplex operation
- Support of twisted pair or Fiber mode output
- IEEE 802.3/802.3u compliant
- Supports IEEE 802.3u clause 28 auto negotiation
- Supports power down mode
- Supports operation under Link Down Power Saving mode
- Supports Base Line Winder (BLW) compensation
- Supports repeater mode
- Speed/duplex/auto negotiation adjustable
- 3.3V operation with 5V IO signal tolerance
- Low operation power consumption and only need single supply 3.3V
- Adaptive Equalization
- 25MHz crystal/oscillator as clock source
- Multiple network status LED support
- Flow control ability support to co-work with MAC (by MDC/MDIO)
- 48 pin LQFP package

2. General Description

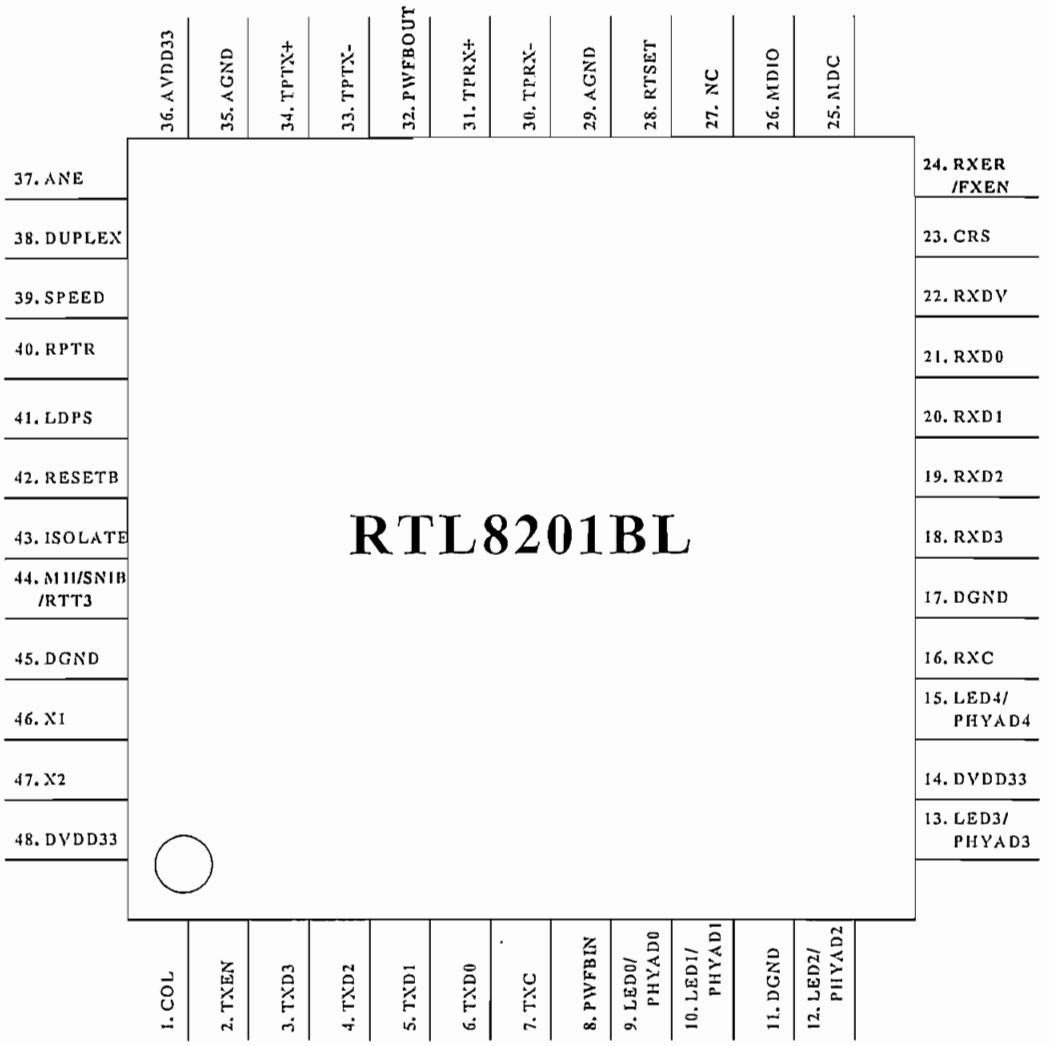
The RTL8201BL is a single-port Phyceiver with an MII (Media Independent Interface)/SNI(Serial Network Interface). It implements all 10/100M Ethernet Physical-layer functions including the Physical Coding Sublayer (PCS), Physical Medium Attachment (PMA), Twisted Pair Physical Medium Dependent Sublayer (TP-PMD), 10Base-Tx Encoder/Decoder and Twisted Pair Media Access Unit (TPMAU). A PECL interface is supported to connect with an external 100Base-FX fiber optical transceiver. The chip is fabricated with an advanced CMOS process to meet low voltage and low power requirements.

The RTL8201BL can be used as a Network Interface Adapter, MAU, CNR, ACR, Ethernet Hub, Ethernet Switch. Additionally, it can be used in any embedded system with an Ethernet MAC that needs a twisted pair physical connection or fiber PECL interface to external 100Base-FX optical transceiver module.

3. Block Diagram



4. Pin Assignments



5. Pin Description

LI: Latched Input in power up or reset
I: Input
P: Power
I/O: Bi-directional input and output
O: Output

5.1 100 Mbps MII & PCS Interface

Symbol	Type	Pin No.	Description
TXC	O	7	**Transmit Clock:** This pin provides a continuous clock as a timing reference for TXD[3:0] and TXEN.
TXEN	I	2	**Transmit Enable:** The input signal indicates the presence of a valid nibble data on TXD[3:0].
TXD[3:0]	I	3, 4, 5, 6	**Transmit Data:** MAC will source TXD[0..3] synchronous with TXC when TXEN is asserted.
RXC	O	16	**Receive Clock:** This pin provides a continuous clock reference for RXDV and RXD[0..3] signals. RXC is 25MHz in the 100Mbps mode and 2.5Mhz in the 10Mbps mode.
COL	O	1	**Collision Detected:** COL is asserted high when a collision is detected on the media.
CRS	O	23	**Carrier Sense:** This pin's signal is asserted high if the media is not in IDEL state.
RXDV	O	22	**Receive Data Valid:** This pin's signal is asserted high when received data is present on the RXD[3:0] lines; the signal is deasserted at the end of the packet. The signal is valid on the rising of the RXC.
RXD[3:0]	O	18, 19, 20, 21	**Receive Data:** These are the four parallel receive data lines aligned on the nibble boundaries driven synchronously to the RXC for reception by the external physical unit (PHY).
RXER/ FXEN	O/LI	24	**Receive error:** if any 5B decode error occurs, such as invalid J/K, T/R, invalid symbol, this pin will go high. **Fiber/UTP Enable:** During power on reset, this pin status is latched to determine at which media mode to operate: 1: Fiber mode 0: UTP mode An internal weak pull low resistor, sets this to the default of UTP mode. It is possible to use an external 5.1KΩ pull high resistor to enable fiber mode. After power on, the pin operates as the Receive Error pin.
MDC	I	25	Management Data Clock: This pin provides a clock synchronous to MDIO, which may be asynchronous to the transmit TXC and receive RXC clocks. The clock rate can be up to 2.5MHz.
MDIO	I/O	26	**Management Data Input/Output:** This pin provides the bi-directional signal used to transfer management information.

5.2 SNI (Serial Network Interface): 10Mbps only

Symbol	Type	Pin No.	Description
COL	O	1	Collision Detect
RXD0	O	21	Received Serial Data
CRS	O	23	Carrier Sense
RXC	O	16	Receive Clock: Resolved from received data
TXD0	I	6	Transmit Serial Data
TXC	O	7	Transmit Clock: Generate by PHY
TXEN	I	2	Transmit Enable: For MAC to indicate transmit operation

5.3 Clock Interface

Symbol	Type	Pin No.	Description
X2	O	47	**25MHz Crystal Output:** This pin provides the 25MHz crystal output. It must be left open when X1 is driven with an external 25MHz oscillator.
X1	I	46	**25MHz Crystal Input:** This pin provides the 25MHz crystal input. If a 25MHz oscillator is used, connect X1 to the oscillator's output. Refer to section 8.3 to obtain clock source specifications.

5.4 100Mbps Network Interface

Symbol	Type	Pin No.	Description
TPTX+ TPTX-	O O	34 33	**Transmit Output:** Differential pair shared by 100Base-TX, 100Base-FX and 10Base-T modes. When configured as 100Base-TX, output is an MLT-3 encoded waveform. When configured as 100Base-FX, the output is pseudo-ECL level.
RTSET	I	28	**Transmit Bias Resistor Connection:** This pin should be pulled to GND by a 5.9KΩ (1%) resistor to define driving current for transmit DAC. The resistance value may be changed, depending on experimental results of the RTL8201BL.
TPRX+ TPRX-	I I	31 30	**Receive Input:** Differential pair shared by 100Base-TX, 100Base-FX, and 10Base-T modes.

5.5 Device Configuration Interface

Symbol	Type	Pin No.	Description
ISOLATE	I	43	Set high to isolate the RTL8201BL from the MAC. This will also isolate the MDC/MDIO management interface. In this mode, the power consumption is minimum. This pin can be directly connected to GND or VCC.
RPTR	I	40	Set high to put the RTL8201BL into repeater mode. This pin can be directly connected to GND or VCC.
SPEED	LI	39	This pin is latched to input during a power on or reset condition. Set high to put the RTL8201BL into 100Mbps operation. This pin can be directly connected to GND or VCC.
DUPLEX	LI	38	This pin is latched to input during a power on or reset condition. Set high to enable full duplex. This pin can be directly connected to GND or VCC.
ANE	LI	37	This pin is latched to input during a power on or reset condition. Set high to enable Auto-negotiation mode, set low to force mode. This pin can be directly connected to GND or VCC.
LDPS	I	41	Set high to put the RTL8201BL into LDPS mode. This pin can be directly connected to GND or VCC. Refer to Section 7.7 for more information.
MII/SNIB/ RTT3(test)	LI/O	44	This pin is latched to input during a power on or reset condition. Pull high to set the RTL8201BL into MII mode operation. Set low for SNI mode. This pin can be directly connected to GND or VCC. In test mode, this pin is an output pin and redefined as RTT3

5.6 LED Interface/PHY Address Config

These five pins are latched into the RTL8201BL during power up reset to configure PHY address [0:4] used for MII management register interface. And then, in normal operation after initial reset, they are used as driving pins for status indication LED. The driving polarity, active low or active high, is determined by each latched status of the PHY address [4:0] during power-up reset. If latched status is High then it will be active low, and if latched status is Low then it will be active high. Refer to Section 7.5 for more information.

Symbol	Type	Pin No.	Description
PHYAD0/ LED0	LI/O	9	**PHY Address [0]** **Link LED:** Active when linked.
PAD1/ LED1	LI/O	10	**PHY Address [1]** **Full Duplex LED:** Active when in Full Duplex operation.
PHYAD2/ LED2	LI/O	12	**PHY Address [2]** **Link 10/ACT LED:** Active when linked in 10Base-T mode, and blinking when transmitting or receiving data.
PHYAD3/ LED3	LI/O	13	**PHY Address [3]** **Link 100/ACT LED:** Active when linked in 100Base-TX and blinking when transmitting or receiving data.
PHYAD4/ LED4	LI/O	15	**PHY Address [4]** **Collision LED:** Active when collisions occur.

5.7 Reset and other pins

Symbol	Type	Pin No.	Description
RESETB	I	42	**RESETB:** Set low to reset the chip. For a complete reset function, this pin must be asserted low for at least 10ms.
PWFBOUT	O	32	**Power Feedback Output:** Be sure to connect a 22uF tantalum capacitor for frequency compensation and a 0.1uF capacitor for noise de-coupling. Then connect this pin through a ferrite bead to PWFBIN(pin8). The connection method is figured in section 7.11.
PWFBIN	I	8	**Power Feedback Input:** see the description of PWFBOUT.
NC		27	Not connection

5.8 Power and Ground pins

Symbol	Type	Pin No.	Description
AVDD33	P	36	**3.3V Analog power input:** 3.3V power supply for analog circuit; should be well decoupled.
AGND	P	29,35	**Analog Ground:** Should be connected to a larger GND plane
DVDD33	P	14,48	**3.3V Digital Power input:** 3.3V power supply for digital circuit.
DGND	P	11,17,45	**Digital Ground:** Should be connected to a larger GND plane.

ANEXO 5

HOJA DE DATOS DEL DISPOSITIVO

W3100A

WIZnet
www.WIZnet.co.kr

i2Chip W3100A

Technical Datasheet v1.3

■ Description

The i2Chip W3100A is an LSI of hardware

■ Features

– Hardware Internet

i2Chip W3100A

■ Description

The i2Chip W3100A is an LSI of hardware protocol stack that provides an easy, low-cost solution for high-speed Internet connectivity for digital devices by allowing simple installation of TCP/IP stack in the hardware.

The W3100A offers system designers a quick, easy way to add Ethernet networking functionality to any product. Implementing this LSI into a system can completely offload Internet connectivity and processing standard protocols from the system, thereby significantly reducing the software development cost.

The W3100A contains TCP/IP Protocol Stacks such as TCP, UDP, IP, ARP and ICMP protocols, as well as Ethernet protocols such as Data Link Control and MAC protocol.

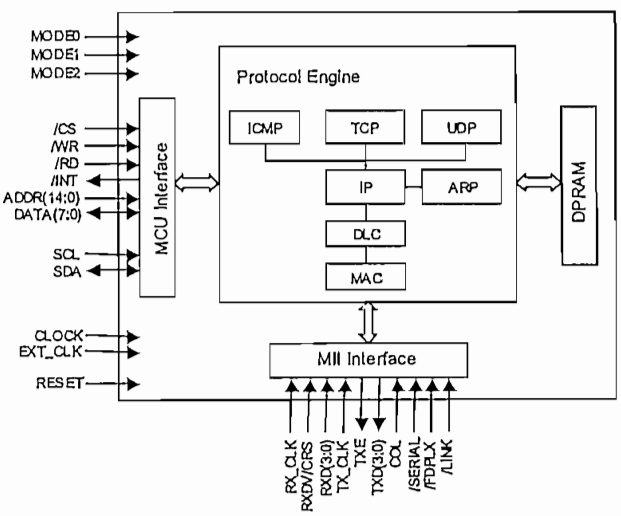
The W3100A offers a socket API (Application Programming Interface) that is similar to the windows socket API. The chip offers Intel and Motorola MCU (8051, i386, 6811 tested) bus interface and I^2C for upper-layer and supports standard MII interface for under-layer Ethernet.

The W3100A can be applied to handheld devices including Internet phones, VoIP SOC chips, Internet MP3 players, handheld medical devices, LAN cards for Web servers, cellular phones and many other non-portable electronic devices such as large consumer electronic products.

■ Features

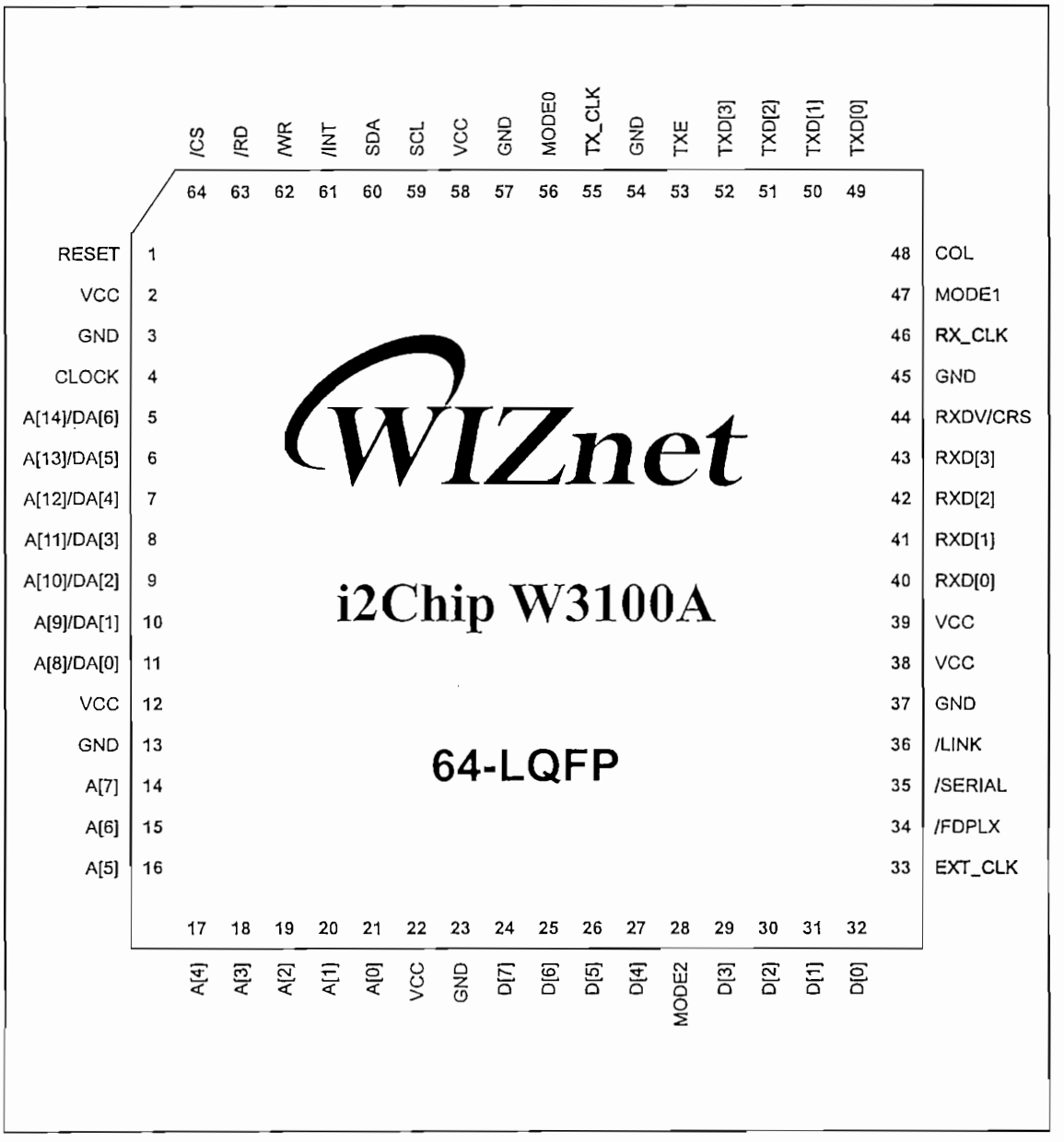
- Hardware Internet protocols included:
  TCP, IP Ver.4, UDP, ICMP, ARP
- Hardware Ethernet protocols included:
  DLC, MAC
- Supports 4 independent connections simultaneously
- Internal ICMP responds to PING commands
- Protocol processing speed: full-duplex 4~5 Mbps
- Intel/Motorola MCU bus Interface
- I^2C Interface
- Standard MII Interface for under-layer physical chip
- Socket API support for easy application programming
- Supports full-duplex mode
- Internal 16Kbytes Dual-port SRAM for data buffer
- 0.35 μm CMOS technology
- Wide operating voltage:
  3.3V internal operation, 5V tolerant 3.3V IOs
- Small 64 Pin LQFP Package

■ Block Diagram



■ Pin Assignment

Figure 1: 64-Pin LQFP Pin Assignments



■ Signal Description

Table 1: W3100A MII Signal Description

PIN#	Signal	I/O	Description
52 51 50 49	TXD[3] TXD[2] TXD[1] TXD[0]	O	**TRANSMIT DATA:** Nibble/Serial NRZ data output to the ENDEC that is valid on the rising edge of TX_CLK. In serial mode, the TXD[0] pin is used as the serial data pin, and TXD[3:1] are ignored.
53	TXE	O	**TRANSMIT ENABLE:** becomes active when the first nibble/serial data of the packet is valid on TXD[3:0] and goes low after the last nibble/serial data of the packet is clocked out of TXD[3:0]. This signal connects directly to the ENDEC (PHY device). This signal is active high.
55	TX_CLK	I	**TRANSMIT CLOCK:** TX_CLK is sourced by the PHY. TX_CLK is 2.5 MHz in 10BASE-T Nibble mode, and 25 MHz in 100BASE-T Nibble mode.
43 42 41 40	RXD[3] RXD[2] RXD[1] RXD[0]	I	**RECEIVE DATA:** Nibble wide receive data (synchronous to RX_CLK) that must be driven on the falling edge of RX_CLK. In serial mode, the RXD[0] pin is used as the data input pin which is also clocked in on the falling edge of RX_CLK. and RXD[3:1] pins become don't cares.
44	RXDV/CRS	I	**CARRIER SENSE:** signal provided by the ENDEC and indicates that carrier is present. This signal is active high.
46	RX_CLK	I	**RECEIVE CLOCK:** Re-synchronized clock from the ENDEC and indicates that carrier is present.
48	COL	I	**COLLISION DETECT:** becomes active when a collision has been detected in Half Duplex modes. This signal is asynchronous, active high and ignored during full-duplex operation.

Table 2: W3100A MCU Interface Signal Description

PIN#	Signal	I/O	Description
5-11	A[14-8] / DA[6-0]	I	**ADDRESS PINS / DEVICE ADDRESS PINS** Used as Address[14 – 8] pin when set in MCU Bus Interface mode. Used as Device address[6 – 0] pin for I^2C Interface when set in I^2C Interface mode.
14-21	A[7-0]	I	**ADDRESS PINS**
24-27 29-32	D[7-4] D[3-0]	I/O	**DATA PINS**
61	/INT	O	**INTERRUPT:** Indicates that the W3100A requires MCU attention after reception or transmission. The interrupt is cleared by writing to the ISR (Interrupt Status Register). All interrupts are maskable by writing IMG (Interrupt Mask Register). This signal is active low.
64	/CS	I	**CHIP SELECT:** This signal is active low.

62	/WR	I	**WRITE ENABLE:** This signal is active low.
63	/RD	I	**READ ENABLE:** This signal is active low.
59	SCL	I	**SCL:** clock used by I^2C when using I^2C Interface mode External Pull high (4.7 kΩ) is required.
60	SDA	I/O	**SDA:** data used by I^2C when using I^2C Interface mode External Pull high (4.7 kΩ) is required.

Table 3: W3100A Miscellaneous Signal Description

PIN#	Signal	I/O	Description
1	RESET	I	**RESET:** Active High input that initializes or reinitializes the W3100A. Asserting this pin will force a reset process to occur which will result in all internal registers reinitializing to their default states as specified for each bit in section x.x, and all strapping options are reinitialized. Refer to section x.x for further detail regarding reset.
4	CLOCK	I	**CLOCK:** primary clock required for internal operation of W3100A. In general, PHY driving clock is shared for saving cost. (25MHz is recommended) Note) Sharing crystal source clock with multiple devices may cause some troubles. In our reference design, we used Realtek's PHY and one crystal for both PHY and W3100A with verification. But for other kind of PHY, please confirm safety prior to decision.
33	EXT_CLK	I	**EXTERNAL CLOCK:** supplementary clock used for MCU I/F of W3100A. In external clocked mode, W3100A uses this clock to interface with MCU, and the access time of W3100A varies upon the frequency of the external clock. Refer to xx for detailed timing diagram. Frequency higher than 25MHz clock rate is granted.
36	/LINK	I	**LINK:** This is the signal generated by Ethernet PHY to indicate the PHY is connected to the Ethernet HUB device or other peer device. This is active low. W3100A can knows the status of physical line connection with this /LINK input. If /LINK is high, W3100A interprets the physical line is disconnected. It results in TCP timeout and connection close. In special PHY case, LINK signal varies in time, which can be grounded.
35	/SERIAL	I	**10BASE-T SERIAL/NIBBLE SELECT:** With the selection of this active low input, transmit and receive data are exchanged serially at a 10 MHz clock rate on the least significant bits of the nibble-wide MII data buses, pins TXD[0] and RXD[0], respectively. This mode is intended for use with the W3100A connected to a PHY using a 10 Mb/s serial interface. There is an internal pull-up resister for this pin. If this pin is left floated externally, then the device will be configured to normal mode. This pin must be externally pulled low (typically x kΩ) in

<table>
<tr><td></td><td></td><td></td><td>order to configure the W3100A for Serial MII operation.</td></tr>
<tr><td>34</td><td>/FDPLX</td><td>I</td><td>FULL/HALF DUPLEX SELECT: This input pin selects Half/Full Duplex operation.
This pin must be externally pulled low (typically x kΩ) in order to configure the W3100A for Full Duplex operation.
0 = Full Duplex
1 = Half Duplex</td></tr>
<tr><td>28, 47, 56</td><td>MODE[2–0]</td><td>I</td><td>MODE SELECT: This input pin selects MCU I/F type and operating mode of W3100A.
Since each pin is positioned as pull-down internally, clock mode – the default mode – is selected when the connection is not made.
<table>
<tr><th>M2</th><th>M1</th><th>M0</th><th></th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Clocked mode</td><td>Mode where MCU Bus signal is analyzed by W3100A by using the clock when MCU Bus I/F is in use.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>External clocked mode</td><td>Mode where MCU Bus signal is analyzed by W3100A by using the external clock when MCU Bus I/F is in use.</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Non-clocked mode</td><td>Mode where MCU bus signal is used directly by W3100A when MCU Bus I/F is in use.</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>I^2C mode</td><td>Mode using I^2C for MCU I/F.</td></tr>
<tr><td>1</td><td>X</td><td>X</td><td>Test mode</td><td>Mode used for testing at the plant. Not to be used by regular users.</td></tr>
</table>
Clocked mode, External clocked mode and Non-clocked mode are used to connect MCU and W3100A when MCU Bus I/F is in use. Choose an appropriate mode and use it by analyzing the MCU bus timing. Refer to timing diagram for each mode for more detail.</td></tr>
</table>

Table 4: W3100A Power Supply Signal Description

PIN#	Signal	I/O	Description
2, 12, 22, 38, 39, 58	VCC		POSITIVE 3.3V SUPPLY PINS
3, 13, 23, 37, 45, 54, 57	GND		NEGATIVE (GROUND) SUPPLY PINS: a decoupling capacitor is recommended to be connected between the Vcc and GND pins

ANEXO 6

HOJA DE DATOS DE LA MEMORIA

EEPROM AT24C02

Features

- Write Protect Pin for Hardware Data Protection
  - Utilizes Different Array Protection Compared to the AT24C02/04/08/16
- Low-voltage and Standard-voltage Operation
  - 2.7 (V_{CC} = 2.7V to 5.5V)
  - 1.8 (V_{CC} = 1.8V to 5.5V)
- Internally Organized 256 x 8 (2K), 512 x 8 (4K), 1024 x 8 (8K) or 2048 x 8 (16K)
- 2-wire Serial Interface
- Schmitt Trigger, Filtered Inputs for Noise Suppression
- Bi-directional Data Transfer Protocol
- 100 kHz (1.8V, 2.5V, 2.7V) and 400 kHz (5V) Clock Rate for AT24C02A, 04A and 08A
- 100 kHz (1.8V) and 400 kHz (2.5V, 2.7V and 5V) Clock Rate for AT24C16A
- 8-byte Page (2K), 16-byte Page (4K, 8K, 16K) Write Modes
- Partial Page Writes are Allowed
- Self-timed Write Cycle (10 ms max)
- High Reliability
  - Endurance: One Million Write Cycles
  - Data Retention: 100 Years
- Automotive Grade and Extended Temperature Devices Available
- 8-lead JEDEC SOIC, 8-pin PDIP, and 8-lead TSSOP Packages

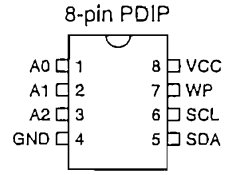
Description

The AT24C02A/04A/08A/16A provides 2048/4096/8192/16384 bits of serial electrically erasable and programmable read only memory (EEPROM) organized as 256/512/1024/2048 words of 8 bits each. The device is optimized for use in many industrial and commercial applications where low power and low voltage operation are essential. The AT24C02A/04A/08A/16A is available in space saving 8-pin PDIP, 8-lead JEDEC SOIC, and 8-lead TSSOP (AT24C02A/04A) packages and is accessed via a 2-wire serial interface. In addition, the entire family is available in 2.7V (2.7V to 5.5V) and 1.8V (1.8V to 5.5V) versions.

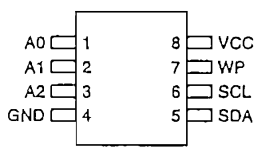
Pin Configurations

Pin Name	Function
A0 - A2	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No-connect

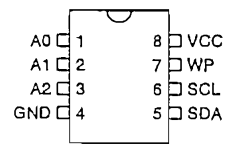
8-pin PDIP



8-lead SOIC



8-lead TSSOP



2-wire Serial EEPROM

2K (256 x 8)

4K (512 x 8)

8K (1024 x 8)

16K (2048 x 8)

AT24C02A
AT24C04A
AT24C08A
AT24C16A

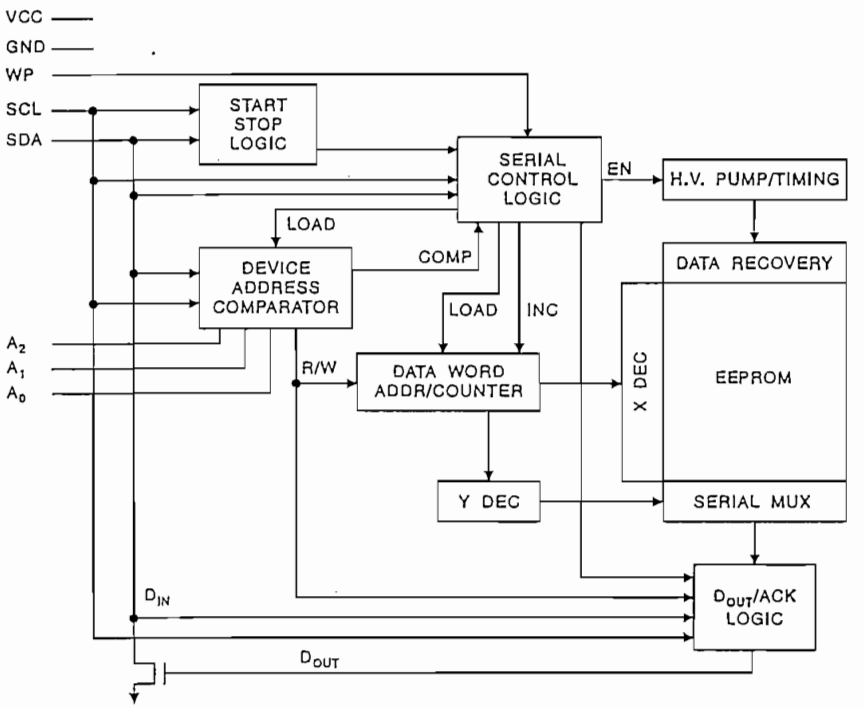
Rev. 0976D–12/01

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.25V
DC Output Current	5.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Block Diagram



Pin Description

SERIAL CLOCK (SCL): The SCL input is used to positive edge clock data into each EEPROM device and negative edge clock data out of each device.

SERIAL DATA (SDA): The SDA pin is bidirectional for serial data transfer. This pin is open-drain driven and may be wire-ORed with any number of other open-drain or open collector devices.

DEVICE/PAGE ADDRESSES (A2, A1, A0): The A2, A1 and A0 pins are device address inputs that must be hard wired for the AT24C02A. As many as eight 2K devices may be addressed on a single bus system (device addressing is discussed in detail under the Device Addressing section).

The AT24C04A uses the A2 and A1 inputs for hard wire addressing and a total of four 4K devices may be addressed on a single bus system. The A0 pin is a no-connect.

The AT24C08A only uses the A2 input for hardwire addressing and a total of two 8K devices may be addressed on a single bus system. The A0 and A1 pins are no-connects.

The AT24C16A does not use the device address pins, which limits the number of devices on a single bus to one. The A0, A1 and A2 pins are no-connects.

WRITE PROTECT (WP): The AT24C02A/04A/08A/16A have a Write Protect pin that provides hardware data protection. The Write Protect pin allows normal read/write operations when connected to ground (GND). When the Write Protect pin is connected to V_{CC}, the write protection feature is enabled and operates as shown in the following table.

WP Pin Status	Part of the Array Protected			
	24C02A	24C04A	24C08A	24C16A
At V_{CC}	Upper Half (1K) Array	Upper Half (2K) Array	Full (8K) Array	Full (16K) Array
At GND	Normal Read/Write Operations			

Memory Organization

AT24C02A, 2K SERIAL EEPROM: Internally organized with 32 pages of 8 bytes each, the 2K requires an 8-bit data word address for random word addressing.

AT24C04A, 4K SERIAL EEPROM: The 4K is internally organized with 32 pages of 16 bytes each. Random word addressing requires a 9-bit data word address.

AT24C08A, 8K SERIAL EEPROM: The 8K is internally organized with 64 pages of 16 bytes each. Random word addressing requires a 10-bit data word address.

AT24C16A, 16K SERIAL EEPROM: The 16K is internally organized with 128 pages of 16 bytes each. Random word addressing requires an 11-bit data word address.

Pin Capacitance

Applicable over recommended operating range from T_A = 25°C, f = 1.0 MHz, V_{CC} = +1.8V.

Symbol	Test Condition	Max	Units	Conditions
$C_{I/O}$	Input/Output Capacitance (SDA)	8	pF	$V_{I/O}$ = 0V
C_{IN}	Input Capacitance (A_0, A_1, A_2, SCL)	6	pF	V_{IN} = 0V

Note: 1. This parameter is characterized and is not 100% tested.

ANEXO 7

HOJA DE DATOS DE LA MEMORIA

SRAM IS62LV256

IS62LV256

ISSI®

32K x 8 LOW VOLTAGE STATIC RAM

JANUARY 2000

FEATURES

- Access time: 45, 70 ns
- Low active power: 70 mW
- Low standby power
  — 45 µW CMOS standby
- Fully static operation: no clock or refresh required
- TTL compatible inputs and outputs
- Single 3.3V power supply

DESCRIPTION

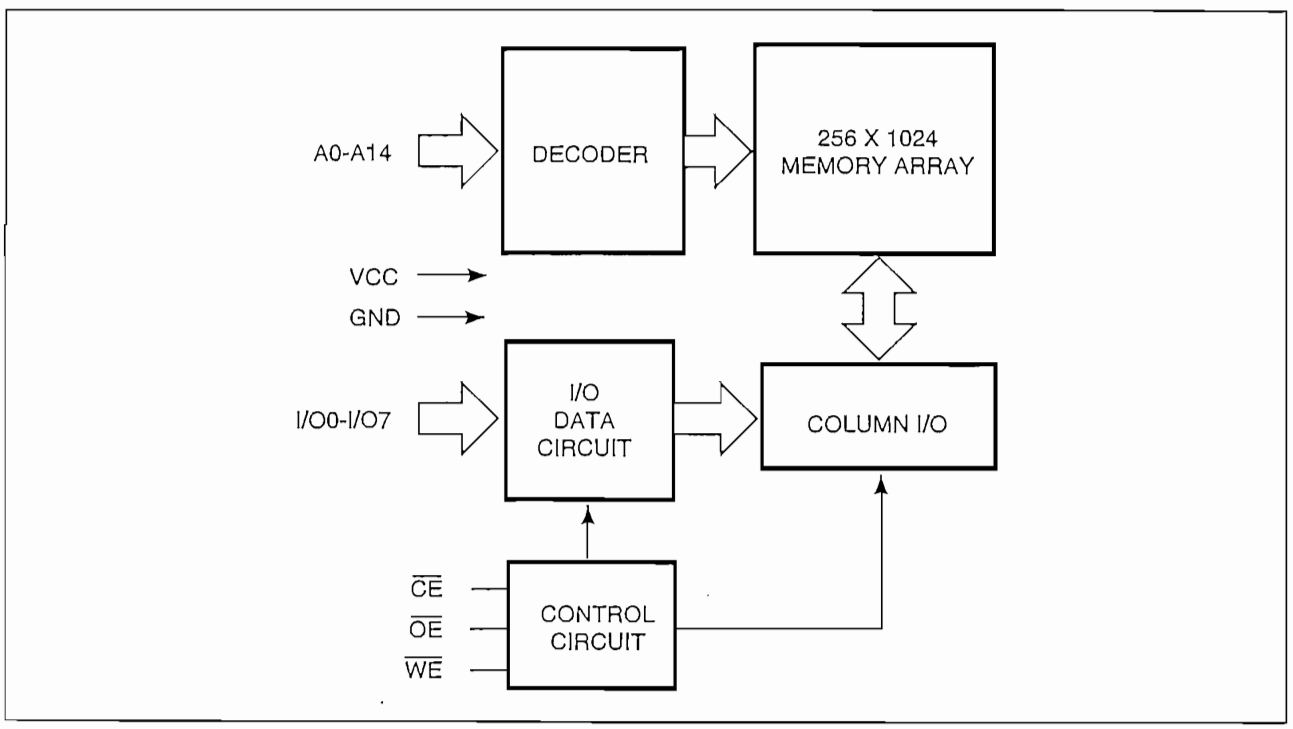
The *ISSI* IS62LV256 is a very high-speed, low power, 32,768-word by 8-bit static RAM. It is fabricated using *ISSI*'s high-performance CMOS double-metal technology.

When $\overline{CE}$ is HIGH (deselected), the device assumes a standby mode at which the power dissipation is reduced to 10 µW (typical) with CMOS input levels.

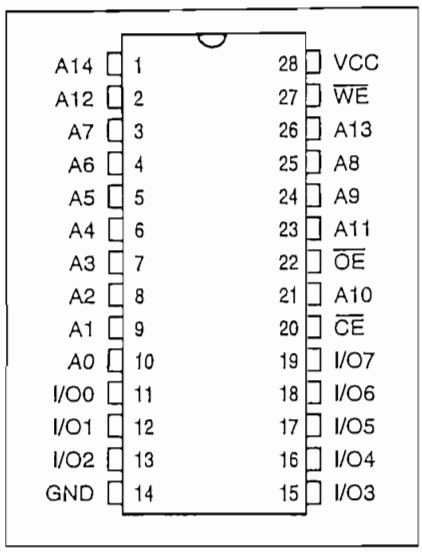
Easy memory expansion is provided by using an active LOW Chip Enable ($\overline{CE}$) input and an active LOW Output Enable ($\overline{OE}$) input. The active LOW Write Enable ($\overline{WE}$) controls both writing and reading of the memory.

The IS62LV256 is pin compatible with other 32K x 8 SRAMs in 300-mil plastic DIP and SOJ, 330-mil plastic SOP, and TSOP (Type I) packages.

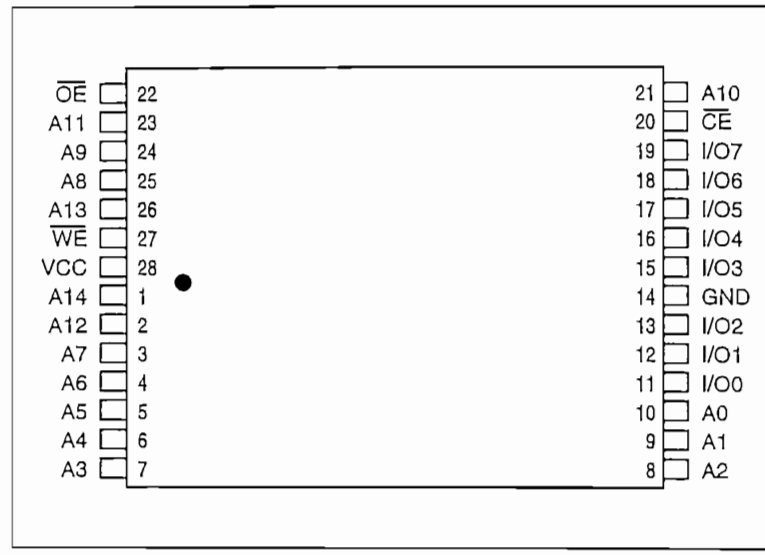
FUNCTIONAL BLOCK DIAGRAM



PIN CONFIGURATION
28-Pin DIP, SOJ and SOP



PIN CONFIGURATION
28-Pin TSOP (Type I)



PIN DESCRIPTIONS

A0-A14	Address Inputs
$\overline{CE}$	Chip Enable Input
$\overline{OE}$	Output Enable Input
$\overline{WE}$	Write Enable Input
I/O0-I/O7	Input/Output
Vcc	Power
GND	Ground

TRUTH TABLE

Mode	$\overline{WE}$	$\overline{CE}$	$\overline{OE}$	I/O Operation	Vcc Current
Not Selected (Power-down)	X	H	X	High-Z	I_{SB1}, I_{SB2}
Output Disabled	H	L	H	High-Z	I_{CC1}, I_{CC2}
Read	H	L	L	D_{OUT}	I_{CC1}, I_{CC2}
Write	L	L	X	D_{IN}	I_{CC1}, I_{CC2}

ABSOLUTE MAXIMUM RATINGS(1)

Symbol	Parameter	Value	Unit
V_{TERM}	Terminal Voltage with Respect to GND	–0.5 to +4.6	V
T_{BIAS}	Temperature Under Bias	–55 to +125	°C
T_{STG}	Storage Temperature	–65 to +150	°C
P_T	Power Dissipation	0.5	W
I_{OUT}	DC Output Current (LOW)	20	mA

Notes:
1. Stress greater than those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

OPERATING RANGE

Range	Ambient Temperature	Vcc
Commercial	0°C to +70°C	3.3V ± 5%
Industrial	–40°C to +85°C	3.3V ± 5%

DC ELECTRICAL CHARACTERISTICS (Over Operating Range)

Symbol	Parameter	Test Conditions		Min.	Max.	Unit
V_{OH}	Output HIGH Voltage	Vcc = Min., I_{OH} = –1.0 mA		2.4	—	V
V_{OL}	Output LOW Voltage	Vcc = Min., I_{OL} = 2.1 mA		—	0.4	V
V_{IH}	Input HIGH Voltage			2.2	Vcc + 0.3	V
V_{IL}	Input LOW Voltage[1]			–0.3	0.8	V
I_{LI}	Input Leakage	GND ≤ V_{IN} ≤ Vcc	Com.	–2	2	µA
			Ind.	–5	5	
I_{LO}	Output Leakage	GND ≤ V_{OUT} ≤ Vcc, Outputs Disabled	Com.	–2	2	µA
			Ind.	–5	5	

Notes:
1. V_{IL} = –3.0V for pulse width less than 10 ns.
2. Not more than one output should be shorted at one time. Duration of the short circuit should not exceed 30 seconds.

POWER SUPPLY CHARACTERISTICS[1] (Over Operating Range)

Symbol	Parameter	Test Conditions		-45 ns Min.	-45 ns Max.	-70 ns Min.	-70 ns Max.	Unit
I_{CC1}	Vcc Operating Supply Current	Vcc = Max., $\overline{CE}$ = V_{IL} I_{OUT} = 0 mA, f = 0	Com. Ind.	— —	20 30	— —	20 30	mA
I_{CC2}	Vcc Dynamic Operating Supply Current	Vcc = Max., $\overline{CE}$ = V_{IL} I_{OUT} = 0 mA, f = f_{MAX}	Com. Ind.	— —	35 45	— —	30 40	mA
I_{SB1}	TTL Standby Current (TTL Inputs)	Vcc = Max., V_{IN} = V_{IH} or V_{IL} $\overline{CE}$ ≥ V_{IH}, f = 0	Com. Ind.	— —	2 5	— —	2 5	mA
I_{SB2}	CMOS Standby Current (CMOS Inputs)	Vcc = Max., $\overline{CE}$ ≥ Vcc – 0.2V, V_{IN} ≥ Vcc – 0.2V, or V_{IN} ≤ 0.2V, f = 0	Com. Ind.	— —	90 200	— —	90 200	µA

Notes:
1. At f = f_{MAX}, address and data inputs are cycling at the maximum frequency, f = 0 means no input lines change.

CAPACITANCE[1,2]

Symbol	Parameter	Conditions	Max.	Unit
C_{IN}	Input Capacitance	V_{IN} = 0V	6	pF
C_{OUT}	Output Capacitance	V_{OUT} = 0V	5	pF

Notes:
1. Tested initially and after any design or process changes that may affect these parameters.
2. Test conditions: T_A = 25°C, f = 1 MHz, Vcc =3.3V.

READ CYCLE SWITCHING CHARACTERISTICS[1] (Over Operating Range)

		-45 ns		-70 ns		
Symbol	Parameter	Min.	Max.	Min.	Max.	Unit
t_{RC}	Read Cycle Time	45	—	70	—	ns
t_{AA}	Address Access Time	—	45	—	70	ns
t_{OHA}	Output Hold Time	2	—	2	—	ns
t_{ACE}	$\overline{CE}$ Access Time	—	45	—	70	ns
t_{DOE}	$\overline{OE}$ Access Time	—	25	—	35	ns
t_{LZOE}[2]	$\overline{OE}$ to Low-Z Output	0	—	0	—	ns
t_{HZOE}[2]	$\overline{OE}$ to High-Z Output	0	20	0	25	ns
t_{LZCE}[2]	$\overline{CE}$ to Low-Z Output	3	—	3	—	ns
t_{HZCE}[2]	$\overline{CE}$ to High-Z Output	0	20	0	25	ns
t_{PU}[3]	$\overline{CE}$ to Power-Up	0	—	0	—	ns
t_{PD}[3]	$\overline{CE}$ to Power-Down	—	30	—	50	ns

Notes:
1. Test conditions assume signal transition times of 5 ns or less, timing reference levels of 1.5V, input pulse levels of 0 to 3.0V and output loading specified in Figure 1a.
2. Tested with the load in Figure 1b. Transition is measured ±500 mV from steady-state voltage. Not 100% tested.
3. Not 100% tested.

ANEXO 8

HOJA DE DATOS DEL ADAPTADOR DE LÍNEA

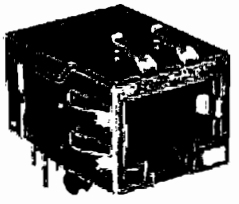
PARA CABLE UTP LU1T516

"LAN-MATE" SERIES

SINGLE RJ45 CONNECTOR MODULE WITH INTEGRATED 10/100 BASE-TX MAGNETICS AND LEDS

P/N: LU1T516-43

Feature



- RJ-45 connector integrated with X'FMR /Impedance resistor/High voltage capacitor .
- Size same as RJ-45 connector to save PCB Board space.
- Reduce EMI radiation, Improve EMI performance.
- Design for 100 BASE transmission over UTP-5 cable.
- Operating temperature range: 0℃ to +70℃.
- Storage temperature range: -25℃ to +85℃
- Recommended panel

Part Number	Standard LED	Forward*V(Max)	(TYP)
3	YELLOW	2.6V	2.1V
4	GREEN	2.6V	2.2V

*with a forward current of 20mA

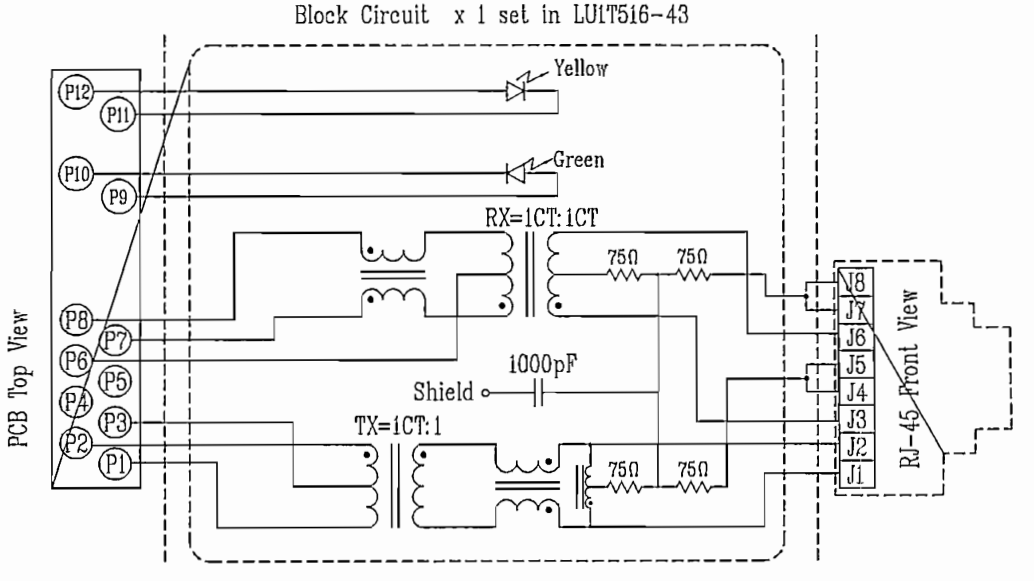
Electrical Specifications @ 25°C

Part Number	Turns Ratio (±5%)		OCL (uH Min) @ 100 KHz/0.1V with 8mA DC Bias	C_{WW} (pF Max)	L.L (uH Max)	Rise Time 10-90% (nS Typ)	HI-POT (Vrms)
	TX	RX					
LU1T516-43	1CT:1	1CT:1CT	350	28	0.4	2.5	1500

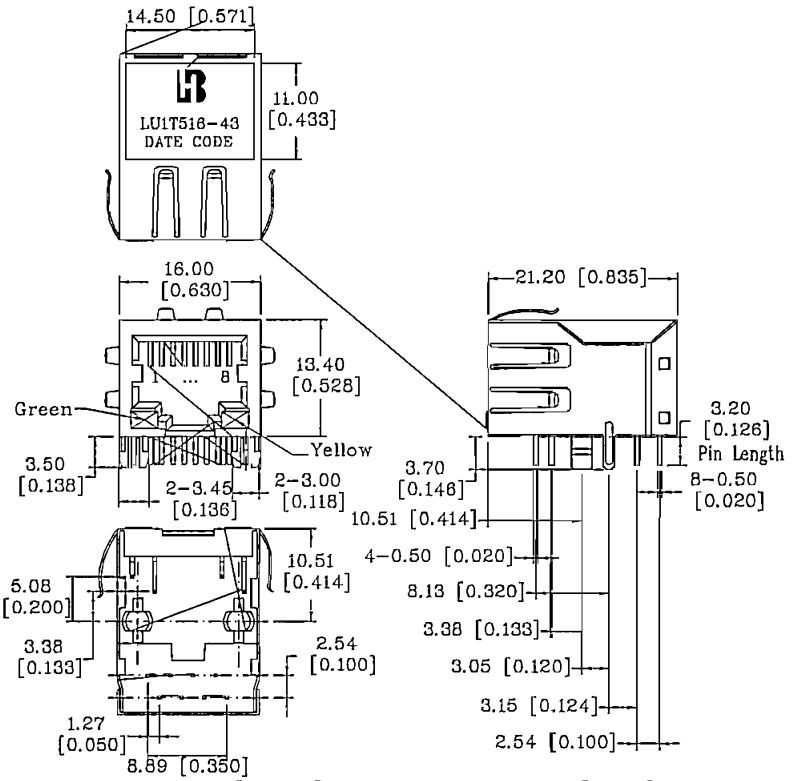
Continue

Part Number	Insertion Loss (dB Max)	Return Loss (dB Min)					Cross talk (dB Min)TX/RX		CMR (dB Min)TX/RX	
	0.3-100MHz	30MHz	40MHz	50MHz	60MHz	80MHz	0.3-60MHz	60-100MHz	1-60MHz	60-125MHz
LU1T516-43	-1.15	-16	-14	-13.5	-13	-10	-45	-35	-37/-35	-33/-28

Schematic



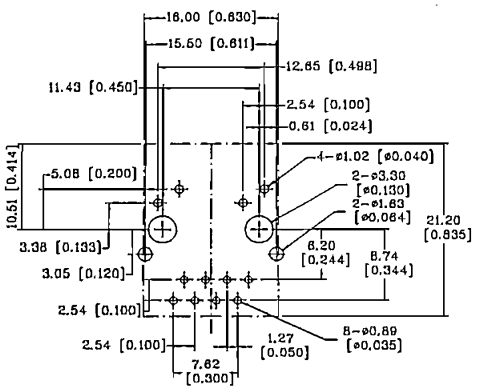
Mechanical



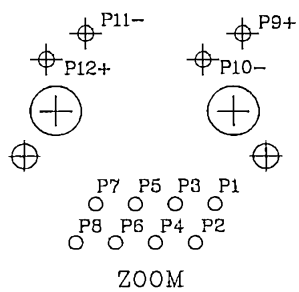
Units: mm[Inches] Tolerances: xx.x0±0.25[0.010]
0.xx±0.05[0.002]

1. Plastic Material	:	PBT 15% Glass Filled(UL 94V-0), Black Housing
2. Metal Shield	:	Brass C2680-H, 20u Nickel Plated, t=0.25mm
3. Spring Wire	:	0.45mm Dia. Phosphor bronze with 6u gold plated
4. Solder Pin	:	0.4x0.4mm, Phosphor bronze with solder Plated
5. Durability	:	300 Mating cycles Min.

PCB Layout Mounting Hole(Top View)



PCB LAYOUT MOUNTING HOLE (TOP VIEW)
Units: mm[Inches] Tolerances: xx.x0±0.25[0.010]
0.xx±0.05[0.002]



ZOOM

ANEXO 9

LIBRERÍAS Y RUTINAS DEL COMPILADOR Cx51

LIBRERÍAS Y RUTINAS DEL COMPILADOR Cx51

Todas las rutinas de las librerías que se mencionan a continuación son implementadas para ser independientes y funcionar utilizando cualquier banco de registros.

Librerías de Archivos: Estas librerías soportan la mayoría de las funciones de llamada del lenguaje C.

C51S.LIB	Modelo de librería pequeña sin aritmética de punto flotante
C51FPS.LIB	Modelo de librería pequeña con aritmética de punto flotante
C51C.LIB	Modelo de librería compacta sin aritmética de punto flotante
C51FPC.LIB	Modelo de librería compacta con aritmética de punto flotante
C51L.LIB	Modelo de librería grande sin aritmética de punto flotante
C51FPL.LIB	Modelo de librería grande con aritmética de punto flotante

Las rutinas de manipulación del *buffer* se usan para trabajar en el *buffer* de la memoria basados en las rutinas de caracter por caracter. Un *buffer* es un arreglo de caracteres similar a un *string*, pero normalmente no termina con un caracter nulo o cero. Todas estas rutinas se llevan a cabo como funciones y los prototipos están incluidos en el archivo STRING.H

Rutinas	Descripción
Memccpy	Copia los bytes de datos desde un *buffer* a otro hasta un caracter especificado o especifica el número de caracteres que tienen que ser copiados
Memchr	Retorna el puntero a la primera ocurrencia de un caracter específico en un *buffer*
Memcmp	Compara un número dado de caracteres desde dos *buffers* diferentes
Memcpy	Copia un número específico de bytes de datos desde un *buffer* a otro

memmove	Copia un número específico de bytes de datos desde un *buffer* a otro
memset	Inicializa un número específico de bytes de datos en un *buffer* a un valor de caracter especificado.

Las rutinas de conversión de caracteres y clasificación permiten probar en forma individual las características de una variedad de caracteres y convertirlos a los diferentes formatos. Todas estas rutinas están incluidas en el archivo CTYPE.H

Rutinas	Descripción
isalnum	Pruebas para un caracter alfanumérico
isalpha	Pruebas para un caracter alfabético
iscntrl	Pruebas para un caracter de control
isdigit	Pruebas para un dígito decimal
isgraph	Pruebas para un caracter imprimible con excepción del espacio
islower	Pruebas para el caracter alfabético en minúscula
isprint	Pruebas para un caracter imprimible
ispunct	Pruebas para un caracter de puntuación
isspace	Pruebas para un caracter de espacio
isupper	Pruebas para un caracter alfabético en mayúscula
isxdigit	Pruebas para un dígito hexadecimal
toascii	Convertidor de un caracter a un código ASCII
toint	Convertidor de un dígito hexadecimal a un valor decimal
tolower	Prueba un caracter y lo convierte en minúscula si es mayúscula
_tolower	Incondicionalmente convierte un caracter en minúscula
toupper	Prueba un caracter y lo convierte en mayúscula si es minúscula
_toupper	Incondicionalmente convierte un caracter en mayúscula

Las rutinas de conversión de datos convierten cadenas de caracteres ASCII a números. Todas estas rutinas se llevan a cabo como funciones y la mayoría son prototipos incluidas en los archivos STDLIB.H y MATH.H

Rutinas	Descripción
Abs	Genera un valor absoluto de un tipo entero
atof	Convierte un string en un float
Atoi	Convierte un string en un int
Atol	Convierte un string en un long
Cabs	Genera un valor absoluto de un tipo carácter
Labs	Genera un valor absoluto de un tipo long
Strtod	Convierte un string en un float
Strtol	Convierte un string en un long
strtoul	Convierte un string en un long sin signo

String: Cadena de tipo caracter.

Float: Es un número real (4 bytes) que puede tener un punto decimal y está comprendido en el rango de:

-3.4×10^{38} a -1.17×10^{-37} para números negativos

1.17×10^{-37} a 3.4×10^{38} para números positivos

Int: Es un número entero cuyo rango de valores depende de la máquina.

Long: Es un número entero de formato largo (4 bytes).

Las rutinas matemáticas realizan cálculos comunes. La mayoría de estas rutinas trabajan con valores de punto flotante, por tanto incluyen librerías que soportan punto flotante. Todas estas rutinas son implementadas como funciones y la mayoría son prototipos que están incluidos en los archivos MATH.H

Rutinas	Descripción
acos	Calcula el arc coseno de un número específico
asin	Calcula el arc seno de un número específico
atan	Calcula el arc tangente de un número específico
atan2	Calcula el arc tangente de una fracción
ceil	Halla el valor del entero más pequeño que es mayor o igual a un número específico
cos	Calcula el coseno de un número específico
cosh	Calcula el coseno hiperbólico de un número específico
exp	Calcula la función exponencial de un número específico
fabs	Encuentra el valor absoluto de un número específico
floor	Halla el valor del entero más grande que es menor o igual que un número específico
fmod	Calcula el resto del punto flotante
log	Calcula el logaritmo natural de un número específico
log10	Calcula el logaritmo en base 10 de un número específico
modf	Genera el entero y los componentes fraccionarios de un número específico
pow	Calcula un valor elevado a una potencia
rand	Genera un número pseudoaleatorio entero al azar
sin	Calcula el seno de un número específico
sinh	Calcula el seno hiperbólico de un número específico
srand	Inicializa el generador de números pseudoaleatorios
sqrt	Calcula la raíz cuadrada de un número específico
tan	Calcula la tangente de un número específico
tanh	Calcula la tangente hiperbólica de un número específico
chkfloat	Chequea el estado de los números float
crol	Desplaza un caracter sin signo a la izquierda de un número de bits especificado
cror	Desplaza un caracter sin signo a la derecha de un número de bits especificado
irol	Desplaza un entero sin signo a la izquierda de un número de bits especificado

iror	Desplaza un entero sin signo a la derecha de un número de bits especificado
lrol	Desplaza un long sin signo a la izquierda de un número de bits especificado
lror	Desplaza un long sin signo a la derecha de un número de bits especificado

Las funciones de asignación de memoria proporcionan medios para especificar y asignar bloques libres de memoria de un espacio disponible. Toda asignación de memoria se lleva a cabo como funciones y son prototipos incluidos como archivos dentro de STDLIB.H

Antes de usar cualquiera de estas funciones para asignar memoria, se debe especificar primero usando la rutina init_*mempool* las condiciones y el tamaño del espacio de memoria que se requiere para satisfacer las demandas.

Las rutinas *calloc* y *malloc* asignan bloques del espacio de memoria. La rutina *calloc* asigna un arreglo con un número específico de elementos de un tamaño dado e inicializa el arreglo con ceros. La rutina *malloc* asigna un número específico de bytes. La rutina *realloc* cambia el tamaño de un bloque asignado mientras que la rutina *free* libera un bloque de memoria asignado previamente al espacio de memoria disponible.

Rutinas	Descripción
calloc	Asigna un lugar para almacenar un arreglo desde el espacio de memoria
free	Libera bloques de memoria que fueron asignados usando calloc, malloc o realloc
init-mempool	Inicializa la localidad y el tamaño del espacio de memoria
malloc	Asigna un bloque del espacio de memoria
realloc	Reasigna un bloque del espacio de memoria

El flujo de rutinas de entrada y salida permiten leer y escribir datos desde el interfaz serial 8051 o un interfaz de entrada/salida (I/O) definido por el usuario. Estas funciones se encuentran almacenadas dentro del directorio LIB.

Rutinas	Descripción
getchar	Lee y reenvía un caracter que usa las rutinas *_getkey* y *putchar*
_getkey	Lee un carácter que usa las funciones del interfaz serial 8051
ges	Lee y reenvía una cadena de caracteres que usan la rutina *getchar*
printf	Escribe datos estructurados que usan la rutina *putchar*
putchar	Escribe un caracter usando el interfaz serial 8051
puts	Escribe una cadena de caracteres y una nueva línea de caracteres que utilizan la rutina *putchar*
scanf	Lee los datos formateados que utilizan la rutina *getchar*
Sprintf	Escribe datos formateados a una cadena
Sscanf	Lee datos formateados desde una cadena
ungetchar	Coloca un caracter una localidad atrás dentro de la entrada del *buffer getchar*
Vprintf	Escribe datos formateados que usan la función *getchar*
Vsprintf	Escribe datos formateados a una cadena

Las rutinas de manipulación de *strings* se ejecutan como funciones y son prototipos que se incluyen dentro del archivo STRING.H. Estos realizan las siguientes funciones:

- Copiar cadenas
- Añaden una cadena al final de otra
- Comparan dos cadenas
- Localizan uno o más caracteres desde una posición especifica de una cadena

Rutinas	Descripción
strcat	Concatena dos cadenas
strchr	Devuelve un puntero de la primera ocurrencia de un caracter especificado de una cadena
strcmp	Compara dos cadenas y devuelve un valor
strcpy	Copia una cadena a otra y devuelve el puntero a la primera
strcspn	Devuelve la posición (subíndice) del primer caracter de una cadena que pertenece al conjunto de caracteres contenidos en otra cadena
strlen	Devuelve la longitud en bytes de la cadena
strncat	Concatena un número específico de caracteres de una cadena a otra
strncmp	Compara dos cadenas en un número específico de caracteres
strncpy	Copia un número específico de caracteres desde una cadena a otra
strpbrk	Devuelve el puntero al primer caracter de una cadena que se iguala a cualquier caracter en una segunda cadena
strpos	Devuelve la posición de la primera ocurrencia de un caracter especificado en una cadena
strrchr	Devuelve el puntero a la última ocurrencia de un caracter especificado en una cadena
strrpbrk	Devuelve el puntero al último caracter en una cadena que se iguala a cualquier caracter en la segunda cadena
strrpos	Devuelve la posición de la última ocurrencia de un caracter especificado en una cadena
strspn	Devuelve la posición del primer caracter en una cadena que no se iguala a cualquier caracter en la segunda cadena
strstr	Devuelve el puntero en una cadena que es idéntica a la segunda sub-cadena

Otras funciones importantes que vale la pena mencionar son:

Rutinas	Descripción
va_arg	Recupera un argumento desde una lista de argumento
va_end	Reestablece un puntero del argumento
va_start	Coloca un puntero al comienzo de un argumento de una lista
setjmp	Guarda la condición actual del stack y dirección del programa
longjmp	Restaura la condición del stack y dirección del programa
nop	Inserta una instrucción NOP del 8051
testbit	Prueba el valor de un bit y lo aclara a cero.

ANEXO 10

CÓDIGO FUENTE DE ALGUNAS DE LAS RUTINAS IMPORTANTES IMPLEMENTADAS

CODIGO FUENTE DEL PROGRAMA IMPLEMENTADO

```
/*
*************************************************************************
                              INICIO.C

Este es un programa que describe las funciones de un servidor Web simple.
Utiliza el puerto 80 y un Web Browser para habilitar el servicio, está
desarrollado únicamente para la presentación de páginas html.

*************************************************************************
*/


#include <reg52.h>                  // Definición de registros para el
microcontrolador 8052
#include "PUERTO.h"
#include "CONTROL.h"
#include "TESIS.h"
#include "SCREEN.h"
#include "EEPROM.h"
#include "RED.h"



#define     MAX_BUF_SIZE      2048        // Tamaño máximo del buffer de
Tx


sfr  CKCON              = 0x8F;           // Definición del registro
CKCON
u_char xdata * rx_buf = 0x7000;           // Posición del buffer de Rx
u_char xdata * tx_buf = 0x7800;           // Posición del buffer de Tx


void Init8052();
void init_sock(u_char i);
void InitNetConfig(void);


// Inicializa el microncontrolador y el controlador de red
void main()
{
      SOCKET i;
      int len;
      u_char state;
      u_char type;

      Init8052();


      PutString("--== Programa  para  Servidor Web ");PutStringLn(" ==--
");

      InitLcd();
      initW3100A();

      InitNetConfig();
```

```
      for (i = 0; i < MAX_SOCK_NUM; i++) init_sock(i);


      while (1)
      {
            i = 0;
            for (i = 0; i < MAX_SOCK_NUM; i++)
            {
                  state = select(i, SEL_CONTROL);

                switch(state)
                {
                case SOCK_ESTABLISHED :
                      if ((len = select(i, SEL_RECV)) > 0)
                      {
                            if (len > MAX_BUF_SIZE) len = MAX_BUF_SIZE;

                            len = recv(i, rx_buf, len);
                            type = ParseReq(rx_buf);
                            len = PrintHeader(tx_buf, type);

                            switch (type)
                            {

                                  case 'c':
                                        len += DoHTML(tx_buf + len);
// Crea la página HTML
                                        len = send(i, tx_buf, len);
// Envía la págia Web
                                        break;
                            }
                      }
                      break;
                case SOCK_CLOSE_WAIT :  // Termina la conexión
                      close(i);
                      break;
                case SOCK_CLOSED :            // Cierra la conexión
                      init_sock(i);
                      break;
                }
            }
      }

}

// Inicia las funciones del 8052
void Init8052(void)
{
      EA = 0;
      CKCON = 0x01;
      IT0 = 0;
      EX0 = 1;
      EX1 = 0;
      EA = 1;
      InitSerial();
      wait_10ms(1);
}

// Reinicia las funciones para desconectar el canal.
```

```
void init_sock(u_char i)
{
      socket(i, SOCK_STREAM, 80, 0);

      NBlisten(i);

}


// Configuración de Red
void InitNetConfig(void)
{
      int i, j;
      char c;
      un_l2cval tip;
      u_char xdata ip[6];
      u_char xdata ipstr[16];
      if(!Check_EEPROM())
      {
            ip[0] = 0x00; ip[1] = 0x01; ip[2] = 0x02; ip[3] = 0x03; ip[4]
= 0x04; ip[5] = 0x05;
            setMACAddr(ip);
            ip[0] = 192; ip[1] = 168; ip[2] = 0; ip[3] = 24;
            setIP(ip);
            ip[3] = 1;
            setgateway(ip);
            ip[0] = 255; ip[1] = 255; ip[2] = 255; ip[3] = 0;
            setsubmask(ip);
      }
      else
      {
            ClrScr();
            GotoXY(0,0);
            Puts(" <CONFIGURACION> ");
            T0 = 0;
            T1 = 1;

            EEP_ReadBytes(GIPADDR,tip.cVal,4);
            if(tip.lVal == 0 || tip.lVal == (0-1))
            {
                  PutString("Iniciando la Configuracion de Red
...");PutStringLn("");PutStringLn("");
                  ip[0] = 0x00; ip[1] = 0x08; ip[2] = 0xDC; ip[3] = 0x00;
ip[4] = 0x00; ip[5] = 0x00;
                  setMACAddr(ip);
                  ip[0] = 192; ip[1] = 168; ip[2] = 0; ip[3] = 2;
                  setIP(ip);
                  ip[3] = 1;
                  setgateway(ip);
                  ip[0] = 255; ip[1] = 255; ip[2] = 255; ip[3] = 0;
                  setsubmask(ip);
                  All_Config_Save();
            }

            PutString("Presione 'C' para cambiar la Configuracion de Red
");
            GotoXY(0,1);
            for(i = 0; i < 16; i++)
            {
                  for( j = 0 ; j < 50 ; j++)
```

```
                {
                        if(IsPressedKey() == 1)
                        {
                                c = GetByte();PutStringLn("");
                                if(c == 'C' || c == 'c')
                                {
                                        Configure();
                                        c = '*';
                                        break;
                                }
                                else if(c== 0x1B)
                                {
                                        c = '*';
                                        break;
                                }
                        }
                        wait_1ms(2);
                }
                if(c == '*') break;
                T0 = !T0;
                T1 = !T1;
                Putch(0xFF);
                PutByte('.');
        }
        PutStringLn("");
        Config_Load();
}
GetNetConfig();
sysinit(0x55,0x55);
ClrScr();
GotoXY(2,0);
Puts("SERVIDOR WEB");
GotoXY(1,1);
GetIPAddress(ip);
inet_ntoa(ip,ipstr); // Muestra la dirección IP en el display
Puts(ipstr);

}
```

```
/*
*************************************************************************
                              TESIS.C

Se encarga de la edición y manejo de páginas Web.
*************************************************************************
*/

#include <reg52.h>      // Defición de registros del microcontrolador
#include "PUERTO.h"
#include "CONTROL.h"
#include "SCREEN.h"
#include "CAMBIO.h"
#include "TESIS.h"



//Esta función se encarga de copiar los datos que se van a transmitir al
buffer
unsigned int WriteBuf(unsigned char xdata *TX_Buf, unsigned char *Stream)
{
      unsigned int Length;

      for (Length = 0; *Stream != '\0'; Length++)
            *TX_Buf++ = *Stream++;

      return Length;
}


//Cabecera HTML para enviar al buffer

unsigned int PrintHeader(unsigned char *TX_Buf, unsigned char
content_type)
{
      unsigned int Length;

      Length = WriteBuf(TX_Buf, "HTTP/1.1 200 OK\r\n");

      switch (content_type) {
            case 't':                              // texto
                  Length += WriteBuf(TX_Buf + Length, "Content-Type:
text/plain\r\n");
                  break;
            case 'g':                              // imagen gif
                  Length += WriteBuf(TX_Buf + Length, "Content-Type:
image/gif\r\n");
                  break;
            case 'j':                              // imagen jpg
                  Length += WriteBuf(TX_Buf + Length, "Content-Type:
image/jpeg\r\n");
                  break;
            case 'h':                              // documento html
                  Length += WriteBuf(TX_Buf + Length, "Content-Type:
text/html\r\n");
                  break;
      }

      return Length;
}
```

```
// Envía el tamaño del documento HTML en código ASCII
unsigned int MSG_Length(unsigned char *TX_Buf, unsigned int DataLength)
{
      unsigned int Length;

      Length = WriteBuf(TX_Buf, "longitud del contenido: ");

      *(TX_Buf + Length++) = D2C(DataLength / 10000);
      *(TX_Buf + Length++) = D2C((DataLength / 1000) % 10);
      *(TX_Buf + Length++) = D2C((DataLength / 100) % 10);
      *(TX_Buf + Length++) = D2C((DataLength / 10) % 10);
      *(TX_Buf + Length++) = D2C(DataLength % 10);

      Length += WriteBuf(TX_Buf + Length, "\r\n\r\n");

      return Length;
}

// Creación del documento HTML

unsigned int DoHTML(unsigned char *TX_Buf)
{
      unsigned int Length;
      unsigned char *Msg;

      Length = MSG_Length(TX_Buf, 1131);

      Msg = "<html><head><title>SERVIDOR WEB CON uC 8x51</title></head>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<body
background=\"http://fie203.epn.edu.ec/web51/font11.jpg\">";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<span style=\"<img
src=\"http://fie203.epn.edu.ec/web51/index.1.1.gif\"v:shapes=\"_x0000_il0
25\"width=\"77\"height=\"89></span></p>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\">  </p>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\"><b><font
color=\"#000080\"face=\"Expo\"size=\"5\">ESCUELA POLITECNICA
NACIONAL</font></b></p>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\">  </p>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\"><b><font face=\"Expo\"
size=\"5\">PROYECTO DE TITULACION</font></b></p>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\">  </p>";
      Length += WriteBuf(TX_Buf + Length, Msg);
```

```
      Msg = "<p align=\"center\" style=\"line-height: 100%\"><font
face=\"Expo\" size=\"4\">DISEÑO Y CONSTRUCCION DE UN PROTOTIPO</font>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\" style=\"line-height: 100%\"><font
face=\"Expo\" size=\"4\">DE UN SERVIDOR WEB BASADO EN UN</font>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\" style=\"line-height: 100%\"><font
face=\"Expo\" size=\"4\">MICROCONTROLADOR 8X51</font>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p align=\"center\" style=\"line-height: 100%\"> <p
align=\"center\" style=\"line-height:100%\">KARINA RIVADENEIRA H<p
align=\"center\" style=\"line-height:100%\">Junio 2004<p
align=\"center\"style=\"line-height: 100%\"> <a
href=\"http://fie203.epn.edu.ec/web51/h1.html\">SIGUIENTE</a><p>";
      Length += WriteBuf(TX_Buf + Length, Msg);

      Msg = "<p> <p align=\"center\"> </p><p
align=\"center\"> </p></body></html>";
      Length += WriteBuf(TX_Buf + Length, Msg);
      return Length;
}

// Analiza los requerimientos del Web Browser

unsigned char ParseReq(unsigned char *Data_Buf)
{
      unsigned char *Pointer;
      unsigned char i = 0;

      Pointer = Data_Buf;

      while(*Pointer++ != 0x2E);

            if (*Pointer =='g') {
                  PutStringLn("GIF");
                  return 'g';

      } else if (*Pointer =='j') {
                  PutStringLn("JPG");
                  return 'j';

      } else if (*Pointer =='h') {
                  PutStringLn("HTML");
                  return 'h';



      }
}
```

```
/*
*************************************************************************
                              TESIS.H

Describe las funciones relacionadas con el servidor Web
*************************************************************************
*/
extern unsigned int WriteBuf(unsigned char xdata *TX_Buf, unsigned char
*Stream);

extern unsigned int PrintHeader(unsigned char *TX_Buf, unsigned char
content_type);

extern unsigned int MSG_Length(unsigned char *TX_Buf, unsigned int
DataLength);

extern unsigned int DoHTML(unsigned char *TX_Buf);

extern unsigned char ParseReq(unsigned char *Data_Buf);

extern UCHAR D2C(char c);
```

ANEXO 11

GLOSARIO

GLOSARIO

ACK ***Reconcimiento o acuse de recibo*** *(Acknowledgment)*
Respuesta enviada por un receptor para indicar que recibió la información que le fue enviada.

ARP ***Protocolo de resolución de direcciones*** *(Address Resolution Protocol)*
Protocolo TCP/IP utilizado para asignar una dirección IP de lato nivel a una dirección de hardware físico de bajo nivel. Se utiliza a través de una sola red física y está limitada a redes que soportan difusión de hardware.

CSMA/CD ***Acceso Múltiple sensible a portadora con detección de colisiones*** *(Carrier Sense Multiple Access with Collision Detection)*
Técnica de control de acceso al medio para medios de transmisión de acceso múltiple. Una estación que desea transmitir, primero detecta el medio y solo transmite si el medio está desocupado, pero si detecta una colisión, entonces cesa la transmisión.

Datagrama
Unidad básica de información que pasa a través de una red de redes TCP/IP.

DCE ***Equipo de comunicación de datos*** *(Data Communications Equipment)*
Término que se aplica al equipo de comunicación que forma una red de conmutación de paquetes para distinguirlos de los computadores o terminales que se conectan a la red.

DHCP ***Protocolo de configuración dinámica de servidor*** *(Dynamic Host Configuration Protocol)*
Protocolo utilizado por un servidor para obtener toda la información de configuración necesaria incluida en una dirección IP.

DNS ***Sistema de nombres de dominio*** *(Domain Name System)*

Sistema de base de datos distribuida en línea y utilizado para transformar nombres de máquinas en direcciones IP que puedan leer los usuarios.

DTE ***Equipo Terminal de datos*** *(Data Terminal Equipment)*
Término aplicado a computadoras y/o terminales para distinguirlas de una red de conmutación de paquetes a la que están conectadas.

EMI ***Interferencia electromagnética*** *(ElectoMagnetic Interference)*
Interferencia originada por campos eléctricos o electromagnéticos y que pueden afectar, en ocasiones, considerablemente a los equipos informáticos, alterando datos o perjudicando el comportamiento de circuitos eléctricos y dispositivos.

FCS ***Secuencia de chequeo de trama*** *(Frame Check Sequence)*
Código de detección de errores insertado como campo en un bloque de datos para transmitirlos. Este código sirve para comprobar errores cuando se reciben los datos

FFDI ***Interfaz para distribución de datos en fibra*** *(Fiber Distribution Data Interface)*
Tecnología de red Token Ring basada en fibra óptica con una tasa de transferencia de 100Mbps utilizando luz con longitud de onda de 1300 nm.

FTP ***Protocolo de transferencia de archivos*** *(File Transfer Protocol)*
Protocolo estándar de alto nivel de TCP/IP que sirve para transferir archivos de una máquina a otra.

Gateway ***Compuerta***
Programa o dispositivo de comunicaciones que transfiere datos entre redes que tienen funciones similares pero operativas diferentes

HOST
Computador o mainframe que hace las veces de nodo central para el intercambio de mensajes en un sistema de correo electrónico. Computadora utilizada para

preparar programas para uso de otra computadora u otro sistema de procesamiento de datos. Entidad que tiene una dirección dentro de una red.

HTML ***Lenguaje de marcado de hipertexto*** *(Hypertext Markup Language)*
Lenguaje en el que se escriben las páginas a las que se accede a través de navegadores WWW. Admite componentes hipertextuales y multimedia.

HTTP Protocolo de transferencia de hipertexto *(Hypertext Transfer Protocol)*
Protocolo usado para la transferencia de documentos WWW

ICMP ***Protocolo de mensajes de control Internet*** *(Internet Control Message Protocol)*
Parte integral del protocolo de Internet (IP) que resuelve errores y controla los mensajes.

IGMP ***Protocolo de administración de grupo Internet*** *(Internet Group Management Protocol)*
Protocolo utilizado por los host para mantener a los ruteadores locales informados de sus miembros y de sus grupos de multidifusión.

IP ***Protocolo Internet*** *(Internet Protocol)*
Define la unidad de información enviada entre sistemas, que proporciona un servicio de entrega de paquetes básico

ISDN ***Red digital de servicios integrados*** *(Integrated Services Digital Network)*
Tecnología en plena evolución que es ofrecida por las compañías telefónicas. ISDN combina servicios de voz y digitales a través de la red en un solo medio, haciendo posible ofrecer a los clientes servicios digitales de datos así como conexiones de voz a través de un solo "cable".

ISO ***Organización internacional para normalizaciones*** *(International Standard Organization)*

Organización de carácter voluntario fundada en 1946 que es responsable de la creación de estándares internacionales en muchas áreas, incluyendo la informática y las comunicaciones

ISP ***Interfaz de puerto serie*** *(Interface Serial Port)*
Interfase estándar para datos transmitidos secuencialmente que no son síncronos con la unidad central de procesamiento.

MAC ***Control de acceso al medio*** *(Médium Access Control)*
Se trata de generar los protocolos de hardware de bajo nivel utilizados para acceder a una red en particular. Este término también se usa como sinónimo de dirección física.

MIB ***Administración de la base de datos*** *(Management Information Base)*
Conjunto de variables que un ruteador mantiene corriendo SNMP. Los administradores pueden obtener o almacenar estas variables.

OSI ***Sistema de Interconexión Abierto*** *(Open System Interconnection)*
Se trata de los protocolos específicamente estándares de ISO, para la interconexión de sistemas de computadoras cooperativos.

PDU ***Unidad de datos de protocolo*** *(Protocol Data Unit)*
Conjunto de datos especificado en un protocolo de una capa dada y que consta de información de control del protocolo de esa capa, posiblemente de datos del usuario de esa capa.

POP ***Protocolo de oficina de correos*** *(Post Office Protocol)*
Protocolo diseñado para permitir a sistemas de usuario individual leer correo electrónico almacenado en un servidor.

PPP ***Protocolo Punto a Punto*** *(Point to Point Protocol)*
Protocolo para comunicaciones entre computadoras mediante una interfaz de serie. Utiliza el protocolo Internet.

RARP ***Protocolo reverso de resolución de direcciones*** *(Reverse Address Resolution Protocol)*

Protocolo TCP/IP que una máquina sin disco utiliza al arrancar para encontrar su dirección IP. La máquina difunde una solicitud que contiene su dirección de hardware físico y un servidor responde enviando a la máquina su dirección IP.

RFC ***Petición de comentarios*** *(Requests for Comments)*

Serie de documentos que describen el conjunto de protocolos de Internet y otros protocolos similares.

RPC ***Procedimiento de llamada remota*** *(Remote Procedure Call)*

Tecnología en la que un programa invoca servicios a través de una red haciendo modificaciones en los procedimientos de llamada.

SLIP ***Protocolo Internet de línea en serie*** *(Serial Line IP)*

Conexión de acceso telefónico a Internet que utiliza el protocolo TCP/IP.

SMTP ***Protocolo sencillo de transferencia de correo*** *(Simple Mail Transfer Protocol)*

Protocolo estándar de TCP/IP para transferir mensaje de correo electrónico de una máquina a otra. Especifica como interactúan dos sistemas de correo y el formato de los mensajes de control que intercambian para transferir el correo.

SNMP ***Protocolo sencillo de gestión de red*** *(Simple Network Management Protocol)*

Protocolo estándar utilizado para monitorear hosts, ruteadores y las redes a las que están conectados.

TCP ***Protocolo de control de transmisión*** *(Transmission Control Protocol)*

Protocolo de nivel de transporte TCP/IP, estándar que proporciona el servicio de flujo confiable full dúplex y del cual dependen muchas aplicaciones.

TFTP ***Protocolo trivial de transferencia de archivos*** *(Trivial File Transfer Protocol)*

Protocolo estándar TCP/IP para transferencia de archivos con capacidad mínima y sobrecarga mínima.

UDP ***Protocolo de datagrama de usuario*** *(User Datagram Protocol)*

Protocolo estándar que TCP/IP permite a un programa de aplicación en una máquina enviar un datagrama hacia el programa de aplicación de otra máquina.

URL ***Localizador uniforme de recursos*** *(Uniform Resource Locutor)*

Cadena que proporciona la localización de una parte de la información.

WWW ***Red extendida por todo el mundo*** *(World Wide Web)*

Servicio de información a gran escala que permite a un usuario buscar información. Ofrece un sistema de hipermedios que pueden almacenar información como texto, gráficos, audio, etc.