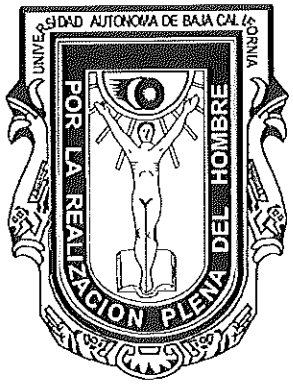
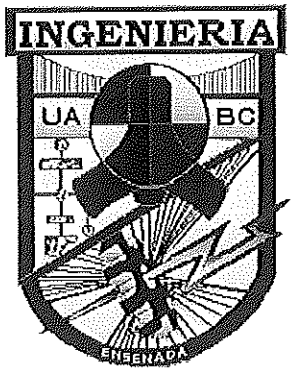
Universidad Autónoma de Baja California

Facultad de Ingeniería
Unidad Ensenada





"Sistema basado en microcontrolador con conexión a la Internet utilizando PPP"

TESIS
QUE PARA OBTENER EL TITULO DE
INGENIERO EN ELECTRÓNICA

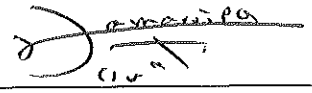
PRESENTA
RENE ALEJANDRO TRENADO RODRÍGUEZ

Ensenada, B.C. Enero del 2005.

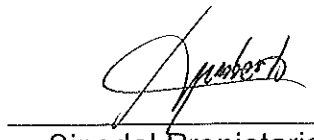
Sistema basado en microcontrolador con conexión a la Internet utilizando PPP

TESIS
QUE PRESENTA:
RENE ALEJANDRO TRENADO RODRÍGUEZ

Aprobada por:



Presidente del Jurado
Dr. Jose de Jesús Zamarripa Topete



Sinodal Propietario
M.C. Humberto Cervantes De Avila

Sinodal Propietario
M.C. Everardo Inzunza González

Lista de Figuras

Lista de Tablas

CAPíTULO I

1.0 INTRODUCCIÓN

En este trabajo de tesis se describe la implementación de uno de los protocolos que comprenden la Internet siendo este uno de los más aceptados y distribuidos en la actual infraestructura de la misma: el protocolo point-to-point o PPP en un sistema basado en microcontrolador de 8-bits de bajo costo como el Motorola/Freescale MC68HC908GP32.

El protocolo PPP se utiliza como plataforma de intercambio de datos entre nuestro sistema basado en un microcontrolador de la familia HC08 y algún otro nodo en la Internet utilizando el User Datagram Protocol / Internet Protocol o UDP/IP.

El código fuente del software de implementación esta escrito en su totalidad en lenguaje de programación C. Además, dicha implementación muestra los grandes beneficios que conlleva el uso de un lenguaje de programación de alto nivel en un sistema basado en microcontrolador. El código ya compilado y enlazado en lenguaje máquina del microcontrolador ocupa menos de 7K de memoria no volátil. Menos de un cuarto de la memoria total del microcontrolador la cual consta de 32K octetos (o bytes) de memoria flash.

1.1 ANTECEDENTES

En la actualidad la Internet forma ya parte integral de nuestras vidas diarias. Millones de personas en todo el mundo se encuentran familiarizadas con los medios para obtener, manejar y transacciones de información a través de la red mundial. Son estas mismas personas las que constantemente alimentan el crecimiento exponencial de la Internet, creando nuevos productos y aplicaciones de consumo dentro de la industria electrónica primordialmente.

La Internet ha entrado a una nueva etapa de crecimiento la cual impacta de manera directa la manera en la que vivimos, en nuestros hogares, o en el trabajo, sin importar distancias ni tiempos. Esto es claramente una tendencia que tendrá un efecto directo en la siguiente evolución de la llamada súper carretera de la información.

Los beneficios no tienen fin. Podríamos imaginar la habilidad de agregar características nuevas a un producto, servicio o aplicación de manera distante, desempeñar el manejo remoto de dispositivos, la integración y unificación de interfaces humanas intuitivas y fáciles de utilizar a cada dispositivo electrónico y poder utilizar dicha interfaz en cualquier lugar del mundo. Mientras los requerimientos de estos dispositivos evolucionen, la integración de la tecnología de Internet a los sistemas actuales y los que están por desarrollarse serán cada vez mas una realidad.

Desafortunadamente, para la mayoría de los dispositivos electrónicos de bajo costo, implementar la tecnología necesaria para lograr una conectividad a la Internet basada en los estándares abiertos no es fácil. Por ejemplo, la gran mayoría de los dispositivos o productos electrónicos del hogar están basados en microcontroladores de muy bajo costo de 8-bits. Ejemplos de estos dispositivos son: televisores, VCRs, lámparas, lavadoras etc. Las probabilidades que dichos sistemas basados en microcontroladores no cuenten por lo menos con un puerto de comunicación al exterior son bastante altas. Mucho menos cuentan con la capacidad y los recursos para poder soportar el protocolo TCP/IP (transmission control protocol/Internet protocol) y algunos otros protocolos que comprenden la Internet sin alterar su función primaria.

La implementación completa de un stack de comunicaciones de TCP/IP requiere una cantidad considerable de recursos de memoria y procesamiento en los sistemas basados en microcontroladores. En la gran mayoría de los casos, agregar esos recursos al sistema implicaría un incremento en el costo y la viabilidad del mismo la cual podría cuestionarse fácilmente.

Sin embargo, actualmente existen diferentes técnicas para permitir que los dispositivos basados en microcontroladores de bajo costo y de recursos limitados se puedan conectar a la Internet:

desde implementaciones limitadas de TCP/IP, implementaciones de TCP/IP completamente en silicio y servidores que actúan como intermediarios. Cada método presenta sus ventajas y desventajas. Este documento explora solamente una.

1.2 OBJETIVO

Este trabajo de Tesis demostrara que un microcontrolador de 8-bits puede conectarse a la Internet utilizando el protocolo PPP.

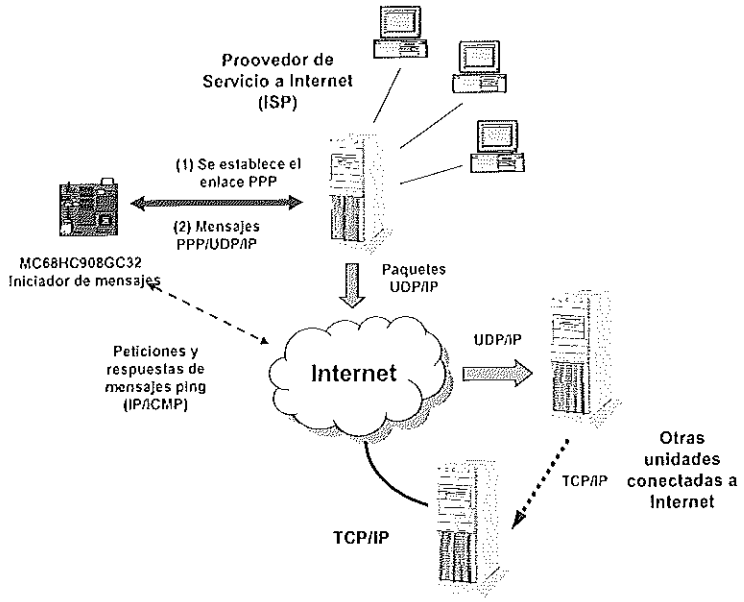


FIGURA 1. Esquema de implementación de esta Tesis del sistema basado en un microcontrolador.

CAPíTULO II

REVISIÓN BIBLIOGRÁFICA

2.0 INTRODUCCIÓN AL FUNCIONAMIENTO DE LA INTERNET

La Internet puede percibirse simplemente como una red de diferentes redes (o red de redes) operando a través de un mecanismo que permite la conexión e interacción entre ellas. Este mecanismo se basa en el Protocolo de Internet al cual comúnmente se le refiere simplemente como protocolo IP.

Para entender como opera la Internet, primero consideremos como funciona una red de área local o LAN. Una LAN es básicamente un grupo de dispositivos electrónicos (o nodos o hosts) conectados entre si los cuales se agrupan manteniendo una relativa proximidad física entre si a través de un medio físico que compartido. Un nodo o host es esencialmente cualquier entidad en la red de área local capaz de recibir y transmitir paquetes de información a otros miembros de la red. Sin importar la tecnología utilizada en la red todos los nodos comparten un medio físico común. Por encima de este medio común, un protocolo de comunicaciones lógicas aceptado permite que los nodos de la red puedan intercambiar información entre ellos.

Una LAN es bastante práctica especialmente cuando el número de nodos que la conforman es relativamente pequeño. Mientras más grande sea el número de nodos o hosts conectados a la LAN más grande será el tráfico de datos que el conducto compartido podría soportar en un momento dado.

Como un ejemplo de una LAN, podríamos considerar el siguiente escenario: Una compañía X estableció una única LAN en todos sus departamentos. Recursos Humanos (RH) trabaja arduamente en la nomina semanal mientras el departamento de producción esta programando el critico plan de manufactura para el próximo perdió mientras tanto, el departamento de ingeniería esta probando el siguiente producto a lanzar al mercado. No hace ningún sentido que el personal de Recursos Humanos experimente grandes retardos en el calculo de nomina debido a la cantidad de trafico existente en la red causados por compartir el mismo canal de comunicación con los demás departamentos de ingeniería y/o producción. Al final del día, nadie recibirá su compensación por que Recursos Humanos no termino el procesamiento de la nomina a tiempo.

Una solución a este hipotético escenario es la de separar la LAN de la compañía en diferentes secciones de red; una para cada departamento. Entonces, en vez de tener solo una LAN, la compañía podría tener tres, y el tráfico de datos podría reducirse solamente al tipo de tráfico que maneja cada departamento. Aplicando este esquema el problema podría ya estar resuelto, sin embargo también es indispensable que cada una de las tres redes o LANs deban estar conectadas entre si de tal manera que los diferentes departamentos puedan compartir información especifica entre si. Para interconectar dos o mas redes se necesitaría un dispositivo u nodo especial de tal manera que este se encuentre conectado a dos o mas redes de manera simultanea y que además sea capaz de transferir información de una a la otra y viceversa. Una máquina con estas características se llama Router o enrutador. La siguiente figura muestra como un enrutador interconecta dos redes.

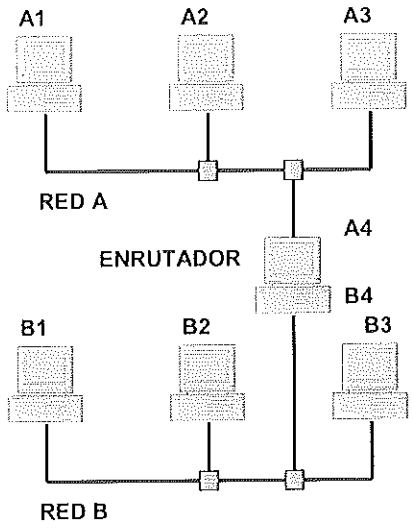


FIGURA 2. En una red un enrutador es un miembro de dos redes simultáneamente.

Un enrutador escucha el tráfico de datos en la red A y de la Red B de manera simultanea. El enrutador detectara cualquier transmisión con la intención de viajar de una red a la otra y re-transmitirá dicha información en la red correspondiente. De acuerdo a la figura, se asume que el enrutador esta conectado a ambas redes a la vez. Este modelo funciona bien en un ambiente de oficina por ejemplo donde los nodos o hosts se encuentran físicamente cercanos el uno del otro. Este esquema cambia un poco cuando definimos un Wide área network or WAN.

En una WAN, las conexiones son típicamente punto-a-punto. Esto significa que solamente una computadora o nodo se conecta a otra ubicada en una localidad distante. En este esquema, un

conducto común se comparte entre los dos nodos o hosts en vez de que este se comparta con muchas computadoras. Considere la siguiente figura #3:

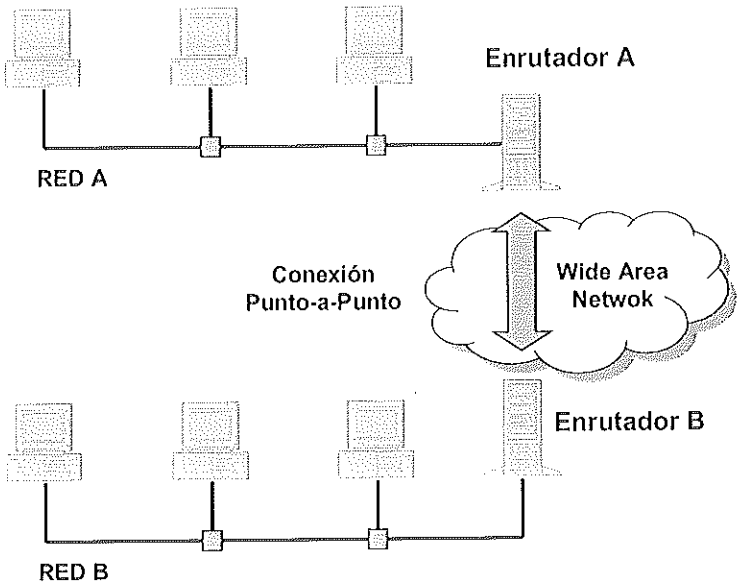


FIGURA 3. Conectando dos redes distantes entre si.

La Internet no es muy diferente a este escenario. De hecho, la Internet es una colección de LANs y WANs conectadas entre si por enrutadores operando en todo el mundo. Es por esto que la Internet se dice esta compuesta por dos tipos diferentes de máquinas: nodos o host y enrutadores o Routers procesando un número de protocolos de comunicación ampliamente aceptados los cuales ya conforman un estándar.

De acuerdo a lo anterior expuesto y a la figura 2, bien podríamos asumir el hecho que la Red B pudiese estar conectada a una Red C y esta a su vez a otra Red D ubicada en Asia esta a otra red denominada Red E y así sucesivamente. Estas redes interconectadas entre si a través de enrutadores forman una internet (con i minúscula). Cuando diferentes internets en todo el mundo se conectan entre si tenemos como resultado la Internet (con I mayúscula).

Lo que hace posible que diferentes sistemas o computadoras (y a su vez diferentes sistemas y plataformas) operen entre si es una colección de estándares y protocolos de redes comúnmente conocidos como Protocolos de Internet.

Como la gran mayoría de las implementaciones de software de comunicaciones, los protocolos de Internet están modelados por capas. El modelo basado en capas de un software de comunicaciones se le denomina comúnmente stack (o pila). Los protocolos de Internet se pueden modelar con un modelo de cinco capas como se muestra en la figura 4.

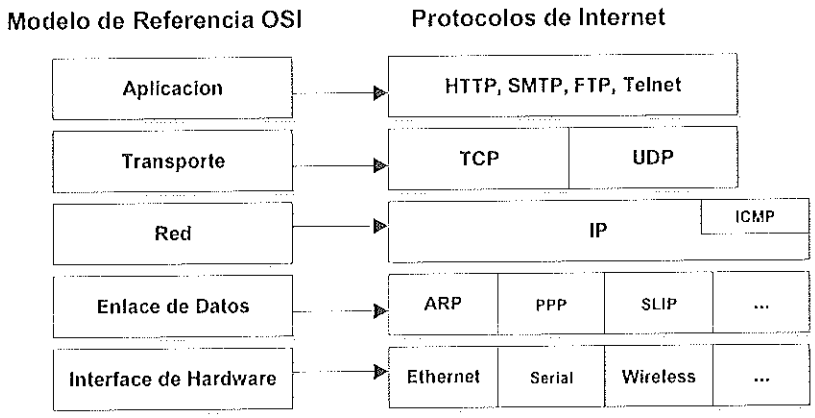


FIGURA 4. El modelo de referencia OSI y su relación con el stack de los protocolos de Internet.

En el stack del protocolo de Internet, cada capa agrega un encabezado (header) y/o una cola (trailer) cuando la información viaja hacia abajo del stack (transmisión). Por ejemplo, si una aplicación que utiliza el HyperText Transfer Protocol (HTTP), como un Web-Browser, quiere enviar un comando HTTP a un nodo remoto en la Internet, la capa de TCP agregara un encabeza dirigido a su capa contraparte TCP ubicada en el nodo remoto. La capa TCP después moverá el comando HTTP hacia abajo del stack o sea a la capa IP. En su turno, esta capa agregara un encabezado especifico de IP encapsulando los datos TCP y HTTP a la vez con información especifica para su contraparte IP en el nodo remoto y así sucesivamente tal y como se muestra en la figura 5.

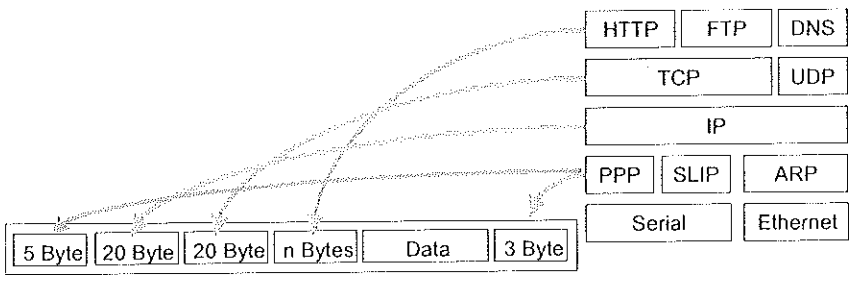


Figura 5. Los encabezados y colas agregados a un mensaje HTTP cuando viaja hacia abajo del stack (cuando se envía el mensaje).

De todos los protocolos que comprenden la familia de protocolos de la Internet el más fundamental es el Protocolo de Internet o IP. Siendo el mejor punto para comenzar en nuestra tarea de entendimiento de la Internet, una breve descripción del protocolo de Internet IP se incluye en la siguiente sección.

2.1 El PROTOCOLO DE INTERNET (IP)

El protocolo de Internet (IP) es el protocolo que hace posible la transmisión de bloques de datos denominados datagrams, de un host o nodo a otro a través de la Internet.

Las funciones principales del Protocolo de Internet son:

1 .- Encontrar una ruta para cada datagram y llevarlo a su destino en una internet.

2.- Fragmentar y reensamblar los paquetes IP

3.- Remover viejos paquetes IP de la red.

El protocolo IP define los datagrams como un bloque de datos el cual presenta un encabezado los cuales conforman las unidades fundamentales de la comunicación en internet. El encabezado contiene la dirección numérica de tanto la fuente como el destino de los dos dispositivos conectados a la Internet. Este número es por lo general referido como dirección IP. La dirección IP identifica de manera única cada uno de los hosts o nodos conectados a la Internet y son utilizadas por los enrutadores para dirigir el tráfico de datagrams a sus destinos. Por lo general, a los enrutadores no les importa la carga (datos) dentro de cada datagram ya que su trabajo es simplemente enrutar los datagrams a su destino lo mas rápido posible.

Los enrutadores son máquinas especiales que operan específicamente a nivel de IP. Desde el punto de vista de las redes o LANs un enrutador es simplemente otra máquina conectada a la red. Desde el punto de vista del usuario, un enrutador es prácticamente invisible. El usuario y las

capas más altas del stack solo ven una gran internet o una gran red donde cada dispositivo conectado cuenta con un identificador único de 32-bits. Estos son los beneficios que trae consigo el protocolo IP.

La fragmentación es otra de las tareas desempeñadas por el protocolo IP. La fragmentación es necesaria cuando un paquete o datagram es demasiado grande como para ser transmitido a través de la interfase de red por debajo de la capa de IP. El protocolo IP divide el paquete en fragmentos más pequeños antes de enviarlos por el adaptador de red. Cuando un paquete fragmentado llega a su contraparte IP, el protocolo IP debe reensamblar el paquete completo antes de pasarlo hacia arriba a la siguiente capa del stack.
Una implementación completa del protocolo IP debería incluir todas estas características tal y como la fragmentación y el reensamble. Sin embargo, la implementación de estos requiere de mucho mas ancho de banda de CPU y más recursos de memoria RAM y ROM que las que un simple microcontrolador de bajo costo podría contener sin mencionar la complejidad implícita a la implementación en software. Es por esta razón, que este documento no describe una implementación de fragmentación y reensamble y tales características se omiten en la implementación final.

Si por alguna razón el nodo remoto envía un paquete fragmentado a nuestro microcontrolador la capa de implementación de PPP en el software simplemente lo ignorara para descartarlo silenciosamente.

El protocolo IP implementa un mecanismo para remover viejos paquetes de la red. En cada uno de los encabezados de los paquetes IP existe un campo que se llama Time-To-Live el cual indica el numero máximo de enrutadores por los cuales puede viajar este paquete antes de ser desechado. Esto es debido a que un paquete que no puede ser enrutado puede rebotar y viajar por toda la Internet consumiendo valuable ancho de banda para siempre.

La mejor manera para entender el protocolo IP se presenta en la figura 6 la cual muestra el formato del paquete en si.

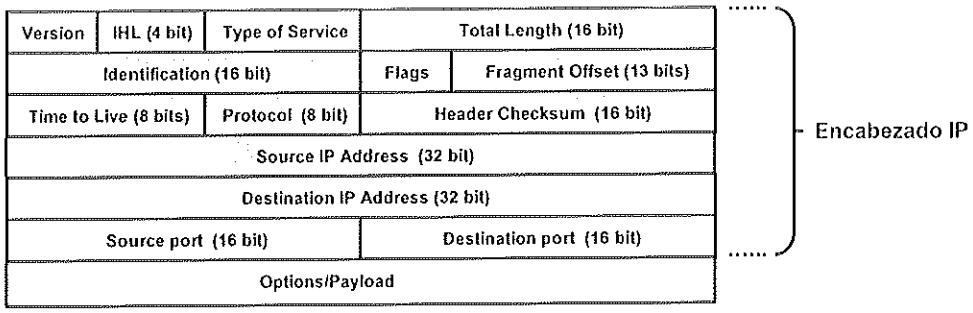


FIGURA 6. El formato de un encabezado IP

Una breve descripción de cada uno de los campos presentes en el encabezado de un paquete IP se muestra en la Tabla 1.

TABLA 1. Descripción de los campos de un paquete (datagrama) IP.

Campo	Descripción
Version	Indica el formato del encabezado de Internet. Actualmente solo dos valores son validos en este campo: 4 (al estándar IP actual) y 6 para la futura versión 6.
IHL (IP Header Length)	El tamaño del encabezado medido en palabras de 32-bits, este valor es comúnmente 5.
Type of service	Especifica parámetros de confiabilidad, precedencia, retardo y velocidad.
Total Length	Tamaño total del paquete (datos y encabezados) medido en bytes.
Identification	Un identificador asignado por el originador del paquete para facilitar el reensamble del mismo en caso de ser necesario.
Flags (3-bits)	Un bit indica fragmentación del paquete mientras el otro es el bit de "No fragmentar", este especifica si el paquete puede ser fragmentado o no. El ultimo bit esta reservado.
Fragment Offset	Indica la porción del fragmento..
Time to Live	Indica el tiempo máximo que este paquete puede existir en la Internet.
Protocol	Indica el protocolo de la siguiente capa el cual se espera reciba los datos cargados por este paquete.
Header Checksum	El checksum de solamente el encabezado.
Source Address	La dirección IP del originador de este paquete.

Destination Address	La dirección IP destino.
Options	El campo de opciones es un campo variable además de opcional. Pudiesen ser cero opciones (lo más común) o más. La implementación del protocolo IP de esta Tesis NO soporta estas opciones.
Padding	Si existen opciones este campo se asegura que el encabezado IP cuente con un tamaño múltiplo de 32-bits.
Data	Los datos cargados por este paquete.

Un ejemplo de un datagram IP se muestra en la figura 7. Note como el paquete IP carga información ICMP de una petición ping de un nodo 192.168.55.2 al 192.168.55.1.

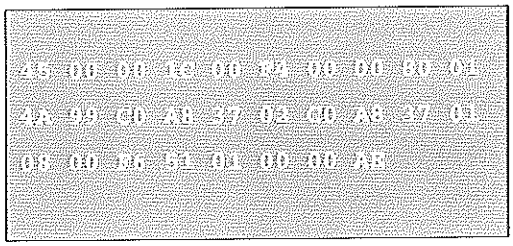


FIGURA 7. Ejemplo de un datagrama IP transportando información de ICMP.

La implementación del protocolo IP utilizada en este documento no utiliza la gran mayoría de los campos del encabezado. Para cada paquete que se recibe, la implementación verifica primero la versión del encabezado y el tamaño del mismo para evitar encabezados mayores a 20 bytes (tamaño típico) y paquetes IP con versión diferente a la versión 4. Los checksums de los paquetes IP no se verifican ya que una verificación más robusta se realiza a todo el paquete IP al nivel de PPP. Esta verificación se llama frame check sequence o FCS.

El protocolo IP no provee un mecanismo para detectar si un paquete efectivamente ha alcanzado su destino. No le interesa si el paquete enviado se pierde, se duplica o se corrompe. El protocolo IP confía en que los protocolos de más alto nivel se aseguren que la transmisión sea confiable. Ese es precisamente el trabajo de la siguiente capa en el stack de Internet: la capa de transporte. La cual en el caso de TCP/IP incluye tanto UDP como TCP.

2.2 USER DATAGRAM PROTOCOL (PROTOCOLO UDP)

UDP son las siglas de user datagram protocol (por sus siglas en ingles), un protocolo estándar con numero asignado 17 (0x11 hexadecimal) como se describe en el RFC 790 (Request for Comments). Su estatus es "recomendado", pero casi todas las implementaciones de TCP/IP comerciales y en uso en la actualidad incluyen UDP. Podemos pensar en UDP como una interfase de aplicación para la capa IP ya que las aplicaciones que utilizan TCP/IP nunca utilizan IP directamente. La capa UDP puede considerarse como una capa extremadamente delgada con solo 8 bytes de encabezado y como consecuencia tiene muy poco overhead. UDP requiere que la aplicación que utilice sus servicios se responsabilice de cualquier error y su posterior manejo, retransmisiones de paquetes, secuencia correcta etc. Esto se muestra en la figura 8.

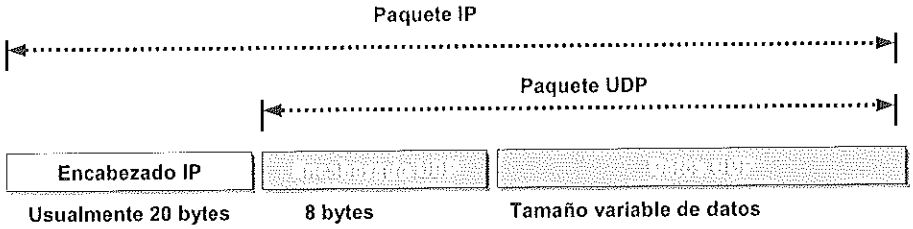


FIGURA 8. UDP visto como una interfase de aplicación a IP.

UDP no provee ningún medio para recuperarse de algún error en la transmisión o recepción. Por ejemplo, la perdida de un paquete implicaría un mecanismo de re-transmisión por parte de la implementación de alto nivel en la capa de aplicación. A diferencia de TCP, UDP no puede recuperarse de un error por si solo, por lo tanto con respecto a TCP UDP es un protocolo no confiable. Por no confiable debe entenderse que UDP no utiliza ningún mecanismo de reconocimiento (acknowledgment) cuando un paquete llega a su destino, ni tampoco re-organiza los datos que recibe fuera de orden o secuencia, ni tampoco proporciona alguna forma de retroalimentación en cuanto a que velocidad debería fluir la información entre nodos o hosts. En otras palabras, los mensajes UDP pueden duplicarse, perderse o llegar fuera de secuencia. Esto implica que es responsabilidad de la capa de la aplicación proporcionar mecanismos para hacer una transferencia de datos confiable y evitar cualquier error o problema como los expuestos anteriormente.

UDP se utiliza principalmente para transmitir video y audio en vivo (en tiempo real), donde si los datos se duplican, se pierden a arriban fuera de secuencia esto no representara un gran problema y la ventaja de tener un protocolo de transporte con tan bajo **overhead** se vuelve evidente.

El encabezado UDP refleja su simplicidad en su diagrama tal y como se muestra en la figura 9

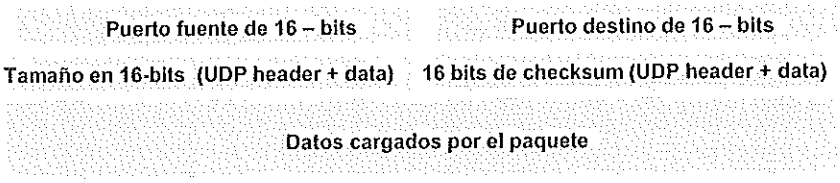


FIGURA 9. El formato de un paquete UDP.

UDP simplemente funciona como un multiplexor/demultiplexor para enviar y recibir datagrams utilizando el concepto de puertos para direccional la información a los diferentes servicios localizados en ambos extremos de la conversación. Nótese como el formato del encabezado UDP especifica dos campos para puertos; uno es para el puerto fuente y otro para el destino.

Un puerto es simplemente un número de 16-bits utilizados por el protocolo de transporte UDP o TCP para identificar cual protocolo de alto nivel o programa de aplicación espera recibir dicha información. En una conexión TCP, por ejemplo, un puerto conocido es el puerto 80. Los servidores HTTP esperan peticiones de sus clientes a través de TCP utilizando este puerto.

Las aplicaciones estándar que utilizan UDP incluyen: Trivial Transfer Protocol o TFTP, Domain Name System (DNS) name server, Remote Procedure Call (RPC), Network File System o NFS, Simple Network Management Protocol o SNMP y Lightweight Directory Access Protocol (LDAP).

Un ejemplo de un paquete UDP/IP que contiene la leyenda "Hello World!" se muestra a continuación en la Figura 10. El paquete es enviado por el nodo o host con dirección IP fuente 192.168.55.2 a la dirección destino 192.168.55.1. El puerto fuente es el 1020 mientras que el destino es el puerto 11222.

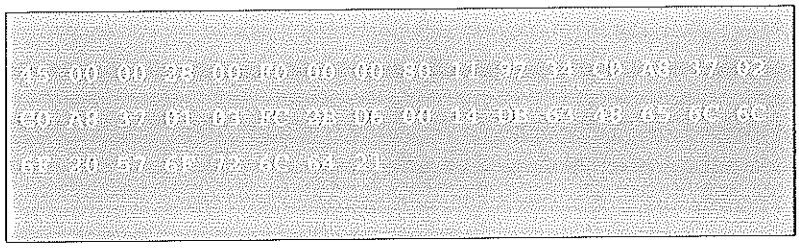


FIGURA 10. Un paquete UDP transportando el mensaje "Hello World!".

2.3 INTERNET CONTROL MESSAGE PROTOCOL (ICMP)

El Internet Control Message Protocol o ICMP se utiliza para proporcionar retroalimentación acerca de posibles problemas potenciales en el ambiente de comunicación basada en el protocolo IP tal y como se describe en el RFC 792 el cual describe a este protocolo. ICMP proporciona un mecanismo para detectar cuando una parte de la Internet a la cual enviamos información se encuentra activa o no.

ICMP siempre va por encima de un paquete IP o lo que es lo mismo siempre es transportado dentro de un paquete IP. Los paquetes o datagramas ICMP tienen el número asignado 1 como numero de protocolo. El campo de datos del paquete IP contendrá el mensaje ICMP mientras que el campo de protocolo incluirá un 1. Un ejemplo de un paquete ICMP se muestra a continuación en la figura 11.

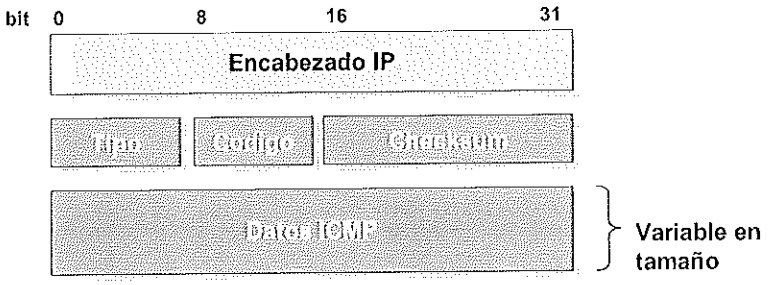
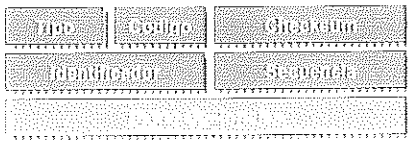


FIGURA 11. El formato de un mensaje ICMP.

El formato de un mensaje ICMP se compone de solo de cuatro campos principalmente. Las implementaciones de este protocolo deben verificar los campos de tipo y código para determinar la naturaleza del mensaje. Por ejemplo, un campo de tipo 8 requiere que el destinatario responda

con un mensaje de eco. El originador de este mensaje ICMP pede entonces determinar si el nodo u host es alcanzable o no. Este tipo de mensaje es probablemente el más popular de ICMP utilizado en la actualidad al cual comúnmente se le denomina "ping" (descrito a continuación). Después del campo de tipo se encuentra el campo de código. El checksum que le sigue se calcula utilizando todo el paquete ICMP sin considerar el encabezado IP.

Esta tesis implementa ICMP para enviar y recibir mensajes "ping". El formato de un mensaje ping se muestra a continuación (oficialmente un ping se llama echo request o petición de eco en español).



Tipo

8 - ECHO REQUEST

0 - ECHO REPLY

Codigo

Siempre 0 para mensajes ECHO

Identificador y Secuencia

Dos campos de 16-bits que facilitan la relación entre los mensajes de petición de eco y sus respuestas.

Datos

Estos datos son opcionales para el originador del mensaje eco (ping) sin embargo el receptor debe replicar esta información en su mensaje de respuesta.

FIGURA 12. Formato del mensaje de Echo y Echo Reply como parte de la especificación ICMP.

Una vez que el transmisor establezca el valor del campo tipo en 8 (echo request) y el código en 0 este debe inicializar el identificador y número de secuencia antes de ejecutar el mensaje ping. Estos campos se utilizan cuando múltiples mensajes ping son transmitidos de manera secuencial. Si se desea, el originador del mensaje puede agregar información opcional al paquete ICMP. El número máximo de datos a transmitir en un mensaje ICMP no debe ser mayor que 64K bytes. Debido a que esta limitante también se aplica a los paquetes de petición, la implementación de ICMP de esta tesis simplemente descarta silenciosamente dichos paquetes.

2.4 MARCANDO A UN PROVEEDOR DE SERVICIO DE INTERNET O ISP

Una vez conectado a la Internet, un sistema puede enviar paquetes de información a otros nodos o hosts en la red siempre y cuando estén activos o on-line sin importar la ubicación física del destinatario. Ese es precisamente el principal trabajo del protocolo IP y de la infraestructura de las internets, de los enrutadores, gateways y demás nodos los cuales conforman la Internet. Cada vez que un sistema requiera de conexión a Internet este deberá contar con una interfase física que le permita lograr la conexión. Una de las maneras mas populares para establecer una conexión a la Internet es la de utilizar un MODEM conectado a una línea telefónica con la ayuda de un proveedor de servicio de Internet o ISP. Un ISP es una compañía que provee acceso a la Internet además de otros servicios como los de Web hosting, web building etc. Un ISP cuenta necesariamente con el equipo de telecomunicaciones necesario para tener un punto de acceso a la Internet con una única dirección IP. Esto se presenta en la figura 13.

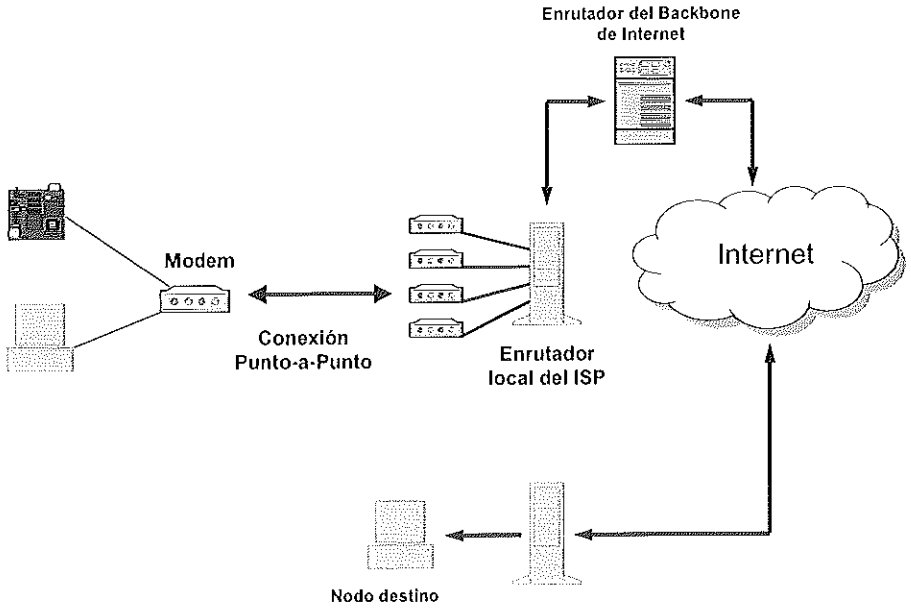


FIGURA 13. Conexión por MODEM a un proveedor de servicio de Internet o ISP.

Un nodo o host marca el número telefónico del ISP con la ayuda de un MODEM. Después que el usuario entra su nombre de usuario y clave de acceso (password) el proceso de validación se concluye. El ISP asigna una dirección IP única al nodo originador de la llamada. Esta dirección IP

fija comúnmente se le denomina como point of presence o POP en el léxico de PPP. Ya que cada host que marque al ISP contara con un POP, este nodo formara parte ahora de la red del ISP y estará a su vez conectada a Internet a través del enrutador o enrutadores presentes en dicha red del ISP. Es en este momento que se dice que el host estará conectado a la Internet.

Cuando un nodo u hosts envía un paquete IP a la Internet, el host no sabe en ningún momento donde se encuentra el nodo remoto (el destino); el simplemente conoce su dirección IP. Cuando un paquete IP alcanza al enrutador del ISP, este tratara de resolver la dirección IP destino dentro de la misma red del ISP. Este paso se repetirá sucesivamente en cada enrutador por donde pase el paquete. Si la dirección no puede resolverse en la red local el enrutador correspondiente lo enviara a otro enrutador en otra red y así sucesivamente.

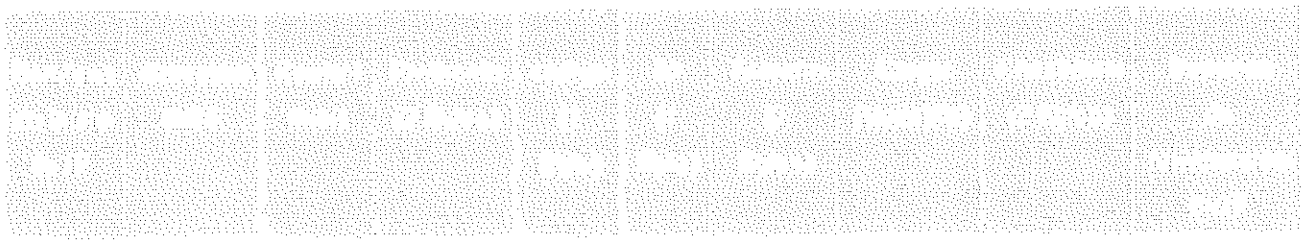
2.5 EL PROTOCOLO DE CONEXIÓN PUNTO-A-PUNTO (PPP)

El protocolo de point-to-point o PPP es el protocolo predominante utilizado en la actualidad en enlaces de tipo seriales. PPP es un conjunto o grupo de protocolos estándar adoptados ampliamente por la industria el cual permite que dos nodos u hosts operen entre si en un ambiente de múltiples proveedores utilizando enlaces seriales tal y como el RS-232.

De acuerdo al RFC 1662, PPP utiliza un encapsulado tipo HDLC proporcionando campos de control y direccionamiento; para PPP estos campos corresponden a las constantes 0xFF y 0x03. Para un enlace con interfase RS232, PPP puede verse como un enlace asíncrono con un stop bit y 8-bits de tamaño de palabra, no paridad y ningún requerimiento especial que especifique una tasa de transmisión.

El único requerimiento impuesto por PPP es el que se emplee un circuito full-duplex sin requerir el uso de señales de control extras tales como RTS (request-to-send) o CTS (clear-to-send). Debido a que ninguna de estas señales se utiliza, la capa física puede desacoplarse de la capa de enlace de datos escondiendo con esto mucho de los detalles del transporte físico.

El formato de un paquete PPP se muestra en la figura 14.



0xC021	Link Control Protocol (LCP)
0xC023	Password Authentication Protocol (PAP)
0xC223	Challenge Handshake Authentication Protocol (CHAP)
0x8021	Internet Protocol Control Protocol (IPCP)
0x0021	Internet Protocol

FIGURA 14. Formato y números de protocolo de un paquete PPP.

2.6 CODIFICACIÓN DE UN PAQUETE PPP

Cada paquete PPP comienza y termia con la bandera 0x7F en hexadecimal. Debido a que esta es una bandera especial, ninguna otra instancia de este valor debe utilizarse dentro del paquete. Para evitar esta confusión, este caracter y algunos otros caracteres especiales ASCII dentro de un paquete deben codificarse de manera distinta. A este proceso se le conoce como "escaping" en ingles. El caracter de control para codificar un caracter especial es el 0x7D seguido por el resultado de una operación XOR del caracter de control con la constante 0x20. Esto también se aplica al caracter de control especial 0x7D (el indicador de escape). La secuencia de escape debe aplicarse a todos los bytes en un paquete PPP sin contar únicamente los indicadores de inicio y final de paquete. Después de la bandera de inicio, dos constantes HDLC le siguen: 0xFF y 0x03. El campo de protocolo siempre es de dos bytes de tamaño indicando que tipo de información esta contenida en el resto del paquete y como debería ser procesada. Para efectos prácticos, la implementación de esta tesis manejara los campos de código, ID y tamaño como campos separados del resto del paquete pero oficialmente estos dos son parte integral del mismo.

El campo código es el tipo de paquete a negociar para los protocolos LCP, PAP, IPCP y CHAP los cuales forman parte de la especificación PPP. Para paquetes IP este valor es usualmente 0x45 (cuando el encabezado IP no incluye opciones lo cual es valido la mayor parte del tiempo).

El ID debe ser único para cada paquete que debe ser negociado y la respuesta debe incluir el mismo paquete para relacionar ambas peticiones a través de este numero de referencia. Una excepción a esta regla se da cuando un paquete PPP encapsula a otro paquete IP es decir el paquete PPP transporta a otro IP. En este caso y para fines prácticos, el ID usualmente corresponderá al campo type-of-service de un encabezado IP. El tamaño de la trama es variable y depende del resultado de las opciones de negociación de una petición o respuesta. En el caso de un paquete IP, el campo de tamaño es corresponde completamente con el campo de tamaño presente en el paquete PPP.

Después de los campos anteriormente descritos, el paquete contiene el resto de las opciones a negociar o en su defecto el resto del paquete IP. Finalmente, una secuencia de verificación de dos bytes denominado "frame check sequence" o FCS se agrega al final del paquete PPP. Este FCS se computa tomando en consideración el resto del paquete sin codificar con la ayuda de una tabla definida en el RFC 1662.

Durante una sesión PPP, no existe distinción entre quien es el servidor y quien el cliente. Ambas extremos del enlace pueden negociar opciones de manera equitativa. Sin embargo, para fines prácticos, este documento define un servidor PPP como el extremo del enlace que es manejado por el ISP y un cliente PPP como el nodo del enlace que inicia la conexión. Otra manera de definir un servidor PPP es aquel que requiere de un nombre de usuario y password para establecer el enlace.

Usualmente, las sesiones PPP se originan por un cliente marcando por teléfono a un ISP. Para comenzar con la sesión, el cliente PPP debe establecer, mantener y terminar la conexión física con el ISP.

Este proceso se muestra a continuación en la figura 15.

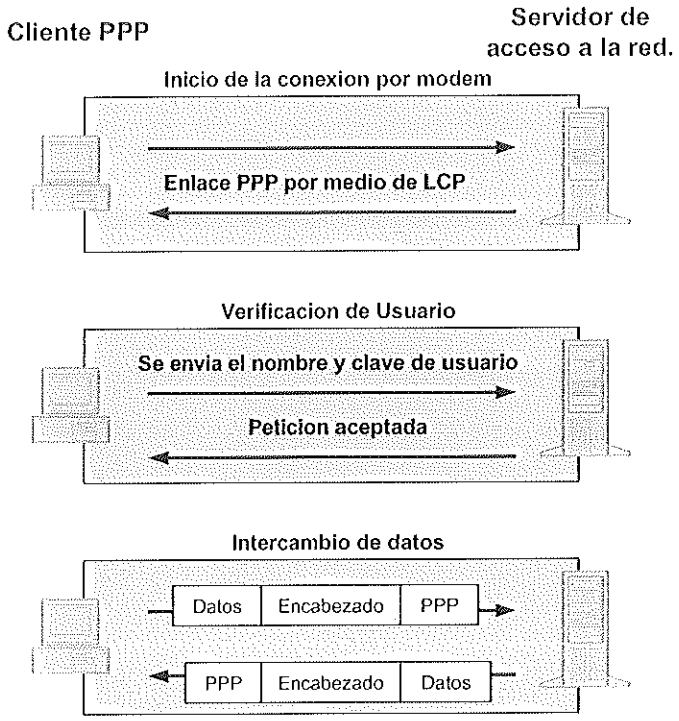


FIGURA 15. Creación de un enlace PPP con un ISP.

Un vistazo mas detallado de la secuencia de acceso a un ISP utilizando PPP involucra los siguientes pasos:

1. Negociaciones LCP – Establecen y configuran el enlace y los parámetros de relacionados con la trama PPP. Tamaño máximo de la trama por ejemplo.

2. Negociar los protocolos de autenticación – Los protocolos de autenticación definidos por PPP son el Challenge Authentication Protocol (CHAP) y el Password Authentication Protocol o PAP. El nivel de seguridad de estos protocolos van desde passwords encriptados a passwords transmitidos en ASCII. La implementación de esta tesis soporta PAP solamente.

3. Negociar los protocolos de control de red (NCP) – Los NCPs son utilizados para establecer y configurar diferentes parámetros del protocolo de red tal y como IP. Esto incluye la negociación del uso de compresión de encabezados y asignación de direcciones IP.

Antes que un enlace PPP se considere listo para ser utilizado por los protocolos de red (en nuestro caso IP) una secuencia de eventos deben ocurrir. El protocolo LCP es quien proporciona el método para establecer, configurar, mantener y terminar la conexión PPP.

El LCP desempeña las siguientes fases:

1. Establecimiento del enlace y negociación de configuración (fase LCP) – Es en esta fase que los paquetes de control de enlace se intercambian y la configuración de las opciones de enlace son negociadas. Una vez que ambos lados de la negociación acuerdan las opciones acordadas el enlace se dice estará abierto, pero no necesariamente listo para que los protocolos de red (network-layer protocols) sean utilizados abiertamente.

2. autenticación (fase PAP o CHAP) – Esta fase es opcional. Cada extremo del enlace pasa por la autenticación con el otro lado del enlace utilizando los métodos de autenticación acordados en la fase anterior (fase LCP).

3. negociación de la configuración del protocolo de red a utilizarse (fase IPCP) – Una vez que la fase LCP haya sido completada, los protocolos de red pueden configurarse de manera separada por el NCP apropiado.

4. Terminación del enlace – El LCP puede terminar el enlace PPP en cualquier momento. Esto por lo general es ocasionado por la petición de un usuario humano pero igual pudiese ocurrir debido a un evento físico.

El protocolo de control de enlace o LCP se utiliza para establecer la conexión a través del intercambio de paquetes de configuración. Las negociaciones LCP con las primeras que ocurren durante el inicio de una sesión PPP.

El mecanismo de negociación utilizado por el LCP en PPP se basa en el uso de códigos de paquetes tal y como se describe en la Tabla 2.

TABLA 2.- identificadores de paquetes PPP.

0	Vendor specific	RFC2153	Extensiones definidas por el proveedor del equipo.
1	Configure-Request	RFC1661	Opciones de configuración que el transmisor pretende negociar.
2	Configure-Ack	RFC1661	Opciones de configuración que el transmisor ha aceptado.
3	Configure-Nak	RFC1661	Opciones de configuración NO aceptadas del paquete de Configure-Request, los valores aceptables son incluidos.
4	Configure-Reject	RFC1661	Opciones que no se reconocen o no se pueden aceptar ni negociar de ninguna manera.
5	Terminate-Request	RFC1661	Termina este enlace.
6	Terminate-Ack	RFC1661	Aceptación de la Terminación de enlace.
7	Code-Reject	RFC1661	Recepción de un paquete LCP con un código desconocido.
8	Protocol-Reje1ct	RFC1661	Recepción de un paquete PPP con campo de protocolo desconocido.
9	Echo-Request	RFC1661	Mecanismo para iniciar un lazo de retorno.
10	Echo-Reply	RFC1661	Respuesta a una petición de tipo eco.
11	Discard-Request	RFC1661	Desechar este mensaje para prueba o depuración del sistema.

La figura #16 muestra un ejemplo del primer paquete LCP transmitido por un ISP

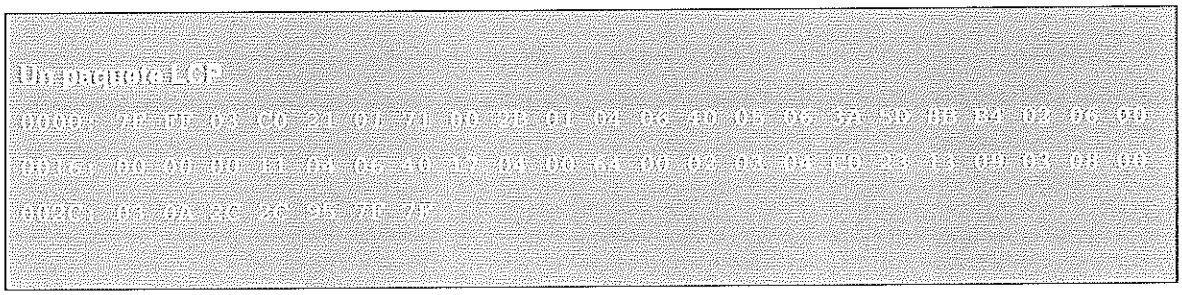


FIGURA 16.- El primer paquete LCP transmitido por un ISP. Nótese como el paquete aun no cuenta con la secuencia de escape para los caracteres especiales.

Una descripción de los datos LCP se muestran en la tabla #3.

TABLA 3.- Ejemplo y descripción de información presente en un paquete LCP.

Tipo de campo	Valor en hexadecimal	Significado
Framing	7F	Comienzo del paquete.
	FF 03	Framing.
Protocolo	C0 21	Protocolo LCP.
código de negociación	01	REQ – Opciones que se pretenden negociar.
Identificador (ID)	71	identificador de este paquete.
Tamaño del paquete	00 2B	Tamaño de los datos cargados por este paquete comenzando desde el código de negociación.
Opciones	01	**Opción 1, Maximum-Receive-Unit.**
	04	Tamaño de la opción 1, 4 Bytes.
	06 40	Valore de la opción sugerido, MRU = 1600.
	05	**Opción 5, Magic number.**
	06	Tamaño de la opción 5, 6 Bytes.
	3A 5D 8B B4	Valor del numero mágico.
	02	**Opción 2, Async-Control-Character-Map.**
	06	Tamaño de la opción 2.
	00 00 00 00	No codificar ningún carácter.
	11	**Opción 17, Multilink-MRRU.**
	04	Tamaño de la opción 11.
	06 40	Valor.
	17	**Opción 23, Link Discriminator for BACP.**
	04	Tamaño de la opción 17.
	00 64	Valor.
	00	**Opción 0, Vendor Specific.**
	02	Tamaño de la opción 0.
	03	**Opción 3, Authentication-Protocol.**
	04	Tamaño de la opción 3.
	C0 23	Valor para sugerir PAP.
	13	**Opción 19, Multilink-Endpoint-Discriminator.**
	09	Tamaño de la opción 13.
	03 08 00 03 0A 2C 2C	Valor de la opción 13.
Checksum	95 7F	El checksum de este paquete.

Framing	7F	Fin del paquete.

Las opciones LCP más comunes durante la conexión inicial son el maximum-receive-unit, protocol-field-compression, magic-number, authentication-protocol and async-control-charcter-map todos se describen en el RFC 1661 y RFC 1662. La implementación en este trabajo trata de establecer las condiciones iniciales de negociación. Primero trata de utilizar la configuración por defecto (default settings) propuestos por el ISP y comienza desde allí. Sin embargo, diferentes implementaciones deberán cambiar la máquina de estados definida dentro de la rutina de *HandleLOPOptions* para procesar opciones de LCP distintas.

2.7 PASSWORD AUTHENTICATION PROTOCOL (PAP)

El protocolo de autenticación de clave de acceso o PAP (password authentication protocol en ingles) se define en el RFC 1334. La principal intención de protocolo PAP es la de ser utilizada primariamente por nodos y hosts que se conectan a un servidor PPP comúnmente a través de líneas telefónicas, pero de igual manera podría aplicarse a líneas punto-a-punto dedicadas. El formato de un paquete de petición de autenticación se muestra en la figura #17.

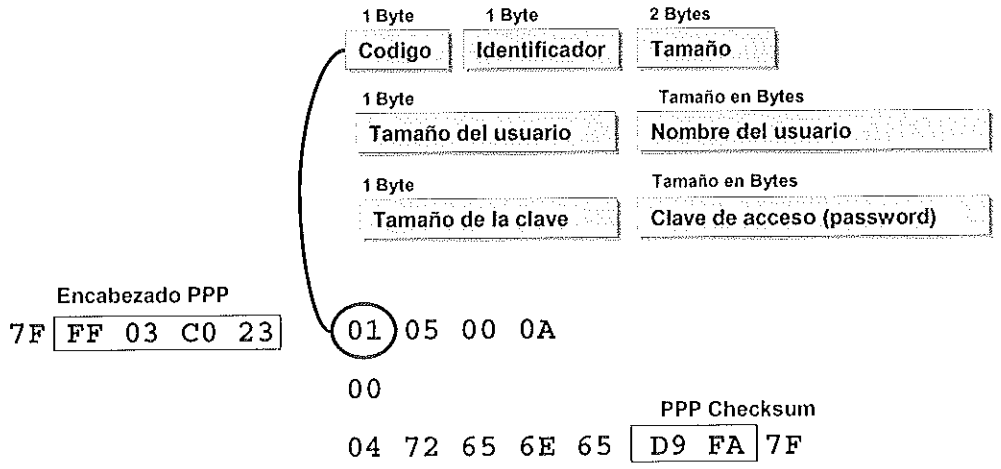


FIGURA 17. Formato y ejemplo de un paquete PAP. El nombre del usuario = "", Password = "rene".

Después que el host PPP ha sido autenticado, el siguiente paso es negociar las opciones del protocolo de red (network-layer protocol). El protocolo de Internet Protocol Control Protocol o IPCP se utiliza para este fin. Las opciones tales como la dirección IP, compresión de encabezado IP, servidores DNS etc. son negociados utilizando IPCP.

El formato de un paquete IPCP es muy similar al del LCP: 1 byte de código de negociación seguido por un identificador, tamaño y las opciones. Una vez que el protocolo IP haya sido configurado, los datagrams de cada red pueden ser enviados en ambas direcciones a través del enlace PPP. En otras palabras la conexión IP se vuelve activa.

IPCP es descrito por el RFC 1332.

2.8 NEGOCIACIONES PPP

En la implementación de PPP en este trabajo de tesis, todas las negociaciones del protocolo LCP se realizan en una máquina de estados dentro del modulo en C PPP.C. Cuando el primer paquete LCP nos llega desde el ISP, la máquina de estados responde con un código NAK en un paquete con las mismas opciones enviadas inicialmente por el ISP. Esto hará que el ISP conteste con una petición acerca del protocolo de autenticación a utilizarse. El diagrama de flujo de la negociación se muestra a continuación en la figura 18.

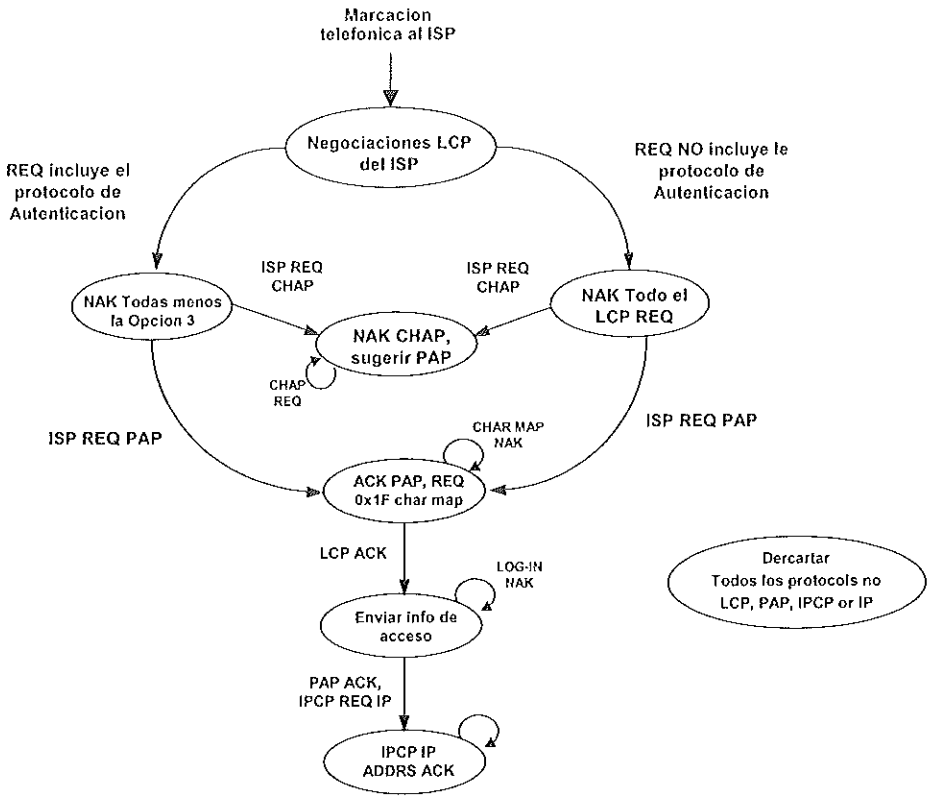


FIGURA 18. Flujo normal de negociaciones PPP.

Un despliegue en hexadecimal de la secuencia de negociación del LCP, PAP e IPCP se muestra en la figura 19. Este despliegue se salvo de una sesión entre un ISP real (Telnor) y la aplicación basada en el microcontrolador M68HC(S)08. Primero, se muestran las negociaciones LCP.v

(1) Primer paquete LCP enviado por el ISP

```
FF 03 C0 21 01 01 00 30 02 06 00 0A 00 00 03 05 C2 23 80 05 06 00 77 BB
67 07 02 08 02 11 04 05 DC 13 13 01 20 B6 60 C1 67 BB 77 00 C0 DC 5E C1
F5 10 00 00 67 40
```

(3) El ISP entonces se ve forzado a negociar el protocolo de acceso (ya sea CHAP o PAP de acuerdo al mensaje NAK previo)

```
FF 03 C0 21 01 02 00 09 03 05 C2 23 80 2A CA
```

(5) El ISP acepta nuestra respuesta con enviándonos un nuevo paquete tipo REQ, esta vez incluye la opción PAP

```
FF 03 C0 21 01 03 00 08 03 04 C0 23 F6 74
```

(8) El ISP acepta codificar todos los caracteres especiales

```
FF 03 C0 21 02 04 00 0A 02 06 FF FF FF FF B0 8E
```

FIGURA 19. Negociaciones LCP con un ISP.

Después que el ISP acuerda el uso del protocolo PAP durante la fase de negociación LCP, el M68HC08 debe enviar el user name y password de una cuenta valida en el ISP para poder acceder a la Internet. Estén proceso se ilustra en la figura 20.

(10) Si la información es correcta, el ISP Acepta el nombre de usuario y su clave

```
FF 03 C0 23 02 05 00 05 00 67 49
```

FIGURA 20. Secuencia PAP.

Ahora que el M68HC08 ya a sido aceptado por el ISP, el siguiente paso es el de configurar los protocolos de red a ser utilizados dentro de la red del ISP. Ya que nos encontramos negociando con un proveedor de servicio de Internet, el protocolo IPCP será utilizado para nosotros poder negociar las opciones del protocolo IP. La negociación de IPCP y PAP se muestran en la figura 21.

(11) El ISP envía una petición de tipo REQ para negociar las opciones IPCP

```
FF 03 80 21 01 01 00 10 02 06 00 2D 0F 01 03 06 C0 A8 37 01 C2 81
```

(13) El ISP envía su respuesta, esta vez debido a el previo NAK enviado, incluye solamente la dirección IP asignada

```
FF 03 80 21 01 02 00 0A 03 06 C8 26 16 02 A4 17
```

(16) El ISP entonces se ve forzado a NO aceptar con un paquete NAK que contiene la dirección IP pre-asignada

```
FF 03 80 21 03 03 00 0A 03 06 C8 38 6F 42 41 F2
```

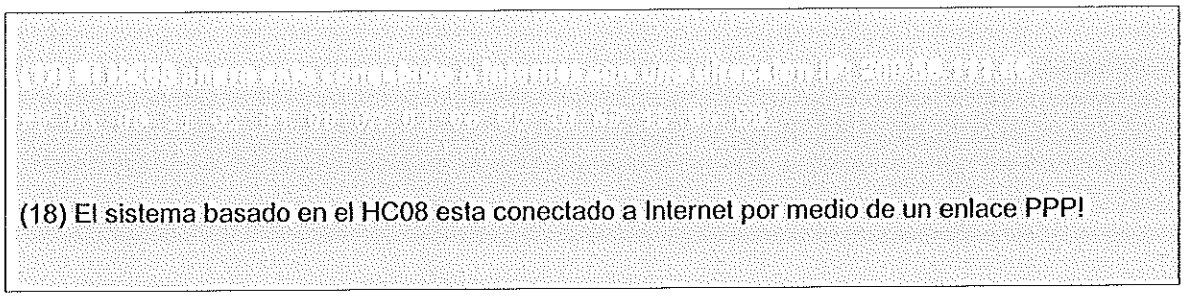


FIGURA 21. Negociaciones IPCP entre el ISP y el microcontrolador MC68HC908GP32.

CAPíTULO III

PROCEDIMIENTO

3.1 MÉTODO E IMPLEMENTACIÓN DE LA APLICACIÓN BASADA EN MICROCONTROLADOR DE UDP/IP SOBRE PPP

Esta trabajo demuestra como un microcontrolador como el MC68HC908GP32 de Freescale Semiconductors puede conectarse a la Internet y aun mantener recursos en el dispositivo para desempeñar alguna otra tarea básica como la de monitoreo remoto y/o control.

La aplicación consiste en un sistema basado en el MC68HC908GP32 que monitorea una variable física externa utilizando el convertidor análogo-digital incluido en el dispositivo vía uno de los 8 canales con los que cuenta el AD.

En el caso de que el AD arroje una lectura especifica u algún otro evento ocurra (un valor de nivel fijo que ha sido alcanzado por ejemplo), el sistema basado en el microcontrolador MC68HC908GP32 enviara una notificación a algún dispositivo en Internet utilizando una dirección IP conocida y pre-compilada en el código ejecutable previamente a través de UDP/IP. Esta dirección IP bien podría ser la de un servidor Proxy en la red o una terminal UDP/IP trabajando como una aplicación especifica o a algún applet de Java o control Active X embebido en una pagina web.

3.2 ESQUEMA A BLOQUES DEL SISTEMA UTILIZADO DURANTE EL DESARROLLO Y PRUEBA

El diagrama a bloques de la aplicación se muestra en la figura 23. El MC68HC908GP32 actúa como un iniciador de mensaje. El microcontrolador espera hasta que ciertas condiciones de programa ocurran. Una condición pre-definida podría ser un sistema de seguridad anunciando alguna intrusión o un sistema de aire acondicionado el cual logra alcanzar una temperatura pre-establecida etc. El sistema primero marcara por teléfono al ISP para establecer el enlace PPP (1). El ISP hará la autenticación del sistema y procederá a asignarle una dirección IP única. Después de esto, el MC68HC908GP32 estará listo para enviar una notificación a la Internet por PPP/UDP/IP (2). Esto se ilustra en la figura 22.

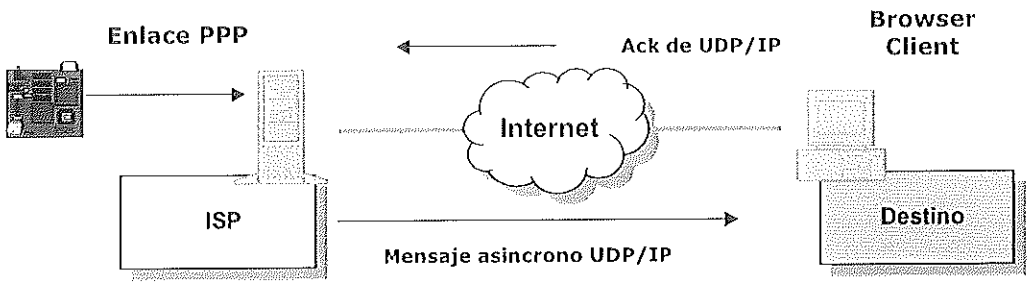


FIGURA 22 – Esquema a bloques de la implementación.

Una vez en la Internet, un mensaje podría viajar virtualmente a cualquier lugar del planeta. Con un poco de esfuerzo, el paquete UDP podría publicarse por medio de una aplicación específica corriendo en una PC u servidor.

3.3 OPERACIÓN DEL SOFTWARE

La implementación del software se ha dividido en una serie de módulos escritos en lenguaje C. La reutilización de código y la expansión del software fueron dos de los objetivos de mantener y separar el software de manera modular; de tal manera que los programadores de C pueden tomar prestado, modificar o re-usar cualquiera de los módulos de código fuente para cumplir con sus necesidades de aplicación específicas. De hecho el código puede portarse a otros dispositivos de la familia HC(S)08 u otras plataformas u microcontroladores.

Inclusive, la lista de módulos podría compilarse a manera de bibliotecas las cuales podrían integrarse a manera de código objeto por el linker (enlazador) durante el proceso de desarrollo.

Los módulos en C que se definen son:

- Main.c.
- CommDrv.c.
- ModemDrv.c.
- PPP.c.
- SLIP.c.
- UDP.c.
- IP.c.
- ICMP.c.

- Delay.c.

El software consiste principalmente de una rutina principal o sea la función estándar de C main() la cual se subdivide a su vez en dos porciones de código: La primera inicializa el puerto de comunicaciones y todos los demás módulos del sistema. La segunda porción es el lazo infinito el cual llama a las funciones ModemEntry () y PPPEntry (). Esto es necesario para mantener las líneas del MODEM activas así como las negociaciones PPP respectivamente.

El primer modulo que se necesita inspeccionar es el CommDrv.C. Este modulo es responsable por la operación correcta del sistema de comunicaciones seriales presente en el Microcontrolador. El modulo incluye un método pseudo-estándar de acceder el puerto serie en el hardware. Para la aplicación, el puerto serie puede verse como un conjunto de funciones que representan y encapsulan la funcionalidad del puerto serie presente en hardware. Para la aplicación, este API se aprecia como una lista de funciones como:

- OpenComm ( ).
- CloseComm( ).
- WriteComm ( ).
- Etc.

La intención de tal implementación es la de perseguir un cierto nivel de abstracción del hardware visto desde la aplicación. La abstracción de hardware nos traen un sin fin de beneficios. Por ejemplo, la re-utilización del código y el fácil mantenimiento del mismo son entre otros algunas de las justificaciones que respaldan su uso. Cuando el hardware cambia, la abstracción cambia en algunas partes de su código; los cambios son casi transparentes para el resto del código de la aplicación.

Nota.- El MC68HC908GP32 define una tabla de vectores de interrupción en la parte mas alta de la memoria flash en las direcciones 0xFFDC a 0xFFFF tal y como se ilustra en la tabla 4. En este espacio de memoria, necesitamos guardar cada una de los localidades de memoria donde se encuentran cada rutina de servicio de interrupción (ISRs) utilizadas por el microcontrolador.

TABLA 4 – Tabla de los vectores de interrupción del MC68HC908GP32.

Vector	Dirección	Fuente del Vector
17	0xFFDC	Time Base Module Vector.

16	0xFFDE	Analog to Digital Conversion Complete.
15	0xFFE0	Keyboard Scan Vector.
14	0xFFE2	Serial Communications Transmit Vector.
13	0xFFE4	Serial Communications Receive Vector.
12	0xFFE6	Serial Communications Error Vector.
11	0xFFE8	SPI Transmit Vector.
10	0xFFEA	SPI Receive Vector.
9	0xFFEC	Timer Interface Module 2 Overflow Vector.
8	0xFFEE	Timer Interface Module 2 Channel 1 Vector.
7	0xFFF0	Timer Interface Module 2 Channel 0 Vector.
6	0xFFF2	Timer Interface Module 1 Overflow Vector.
5	0xFFF4	Timer Interface Module 1 Channel 1 Vector.
4	0xFFF6	Timer Interface Module 1 Channel 0 Vector.
3	0xFFF8	PLL Vector.
2	0xFFFA	IRQ Vector.
1	0xFFFC	Software Interrupt Vector.
-	0xFFFE	Reset.

El modulo CommDrv.C define una subrutina de servicio de interrupción o ISR la cual se ejecuta cada vez que el SCI del microcontrolador recibe un caracter nuevo. Sin embargo, este ISR se compila por el compilador para generar el código objeto que el Linker posteriormente ubicara en la tabla de vectores de interrupción. Esto significa que la dirección del ISR se guarda en la tabla de vectores al momento de enlazar el código. Debido a que el puerto serie en esta implementación se comparte entre los módulos PPP, SLIP y CommDrv para desempeñar diferentes tareas durante el tiempo de ejecución una manera para poderlo compartir se empleo a nivel del modulo CommDrv.c.

Por ejemplo, cuando el MCU marca por teléfono al ISP utilizando un MODEM, después que el ISP contesta, la negociación PPP debe ejecutarse es por esto que el modulo PPP en este punto tendría que controlar el ISR del SCI. En otras palabras ModemDrv.C y PPP.C dependen del ISR proveído por CommDrv.C.

Una manera de lograr esta flexibilidad en el ISR es la de hacer que el ISR en CommDrv.C apunte a un ISR virtual a través de un apuntador a una función. Utilizando este método, el programador y su programa tienen el control total del el código que procese el flujo de caracteres que se reciben a través del puerto serie.

De hecho, el cuerpo de la subrutina de interrupción en CommDrv.C se torna simple y se muestra a continuación en la figura 23.

LISTADO 1

```
static void CommDrvDefaultProc (register BYTE value) { (void) value; };
static void (* EvtProcedure) (register BYTE value) = CommDrvDefaultProc;

//////////////// Rutina de Servicio de interrupción ////////////////
void @interrupt UartRxISR (void) {
        SCS1;                   // Limpiar la bandera de interrupción
        EvtProcedure (SCDR);    // Pasar control a rutina de ISR apuntada por EventProc
}
```

Figura 23. Cuerpo de la subrutina de servicio de interrupción del SCI.

El CPU del MC68HC908GP32 cuenta con un número de modos de direccionamiento en comparación con otros MCUs de 8-bits en el mercado. La definición del ISR en el modulo CommDrv utiliza un poderoso modo de direccionamiento indexado. La instrucción JSR puede saltar a una subrutina apuntada por el par de registros H:X. Al ejecutarse JSR, el program counter saltara a la dirección efectiva de 16-bits apuntada por los registros índice.

Pero como siempre, hay un precio que pagar por esta flexibilidad, y, en este caso, perdemos algunos ciclos de CPU. El mínimo código en ensamblador necesario para representar el código del LISTADO 1 se muestra en la siguiente figura 24.

```
PSH   H
LDA   SCS1   ; Leer el contenido del registro SCS1
LDA   SCDR   ; Guardar el caracter recibido en el acumulador
LDHX  0x45   ; Cargar la dirección efectiva del pseudo-ISR
JSR   ,X     ; Saltar al manejador de evento correspondiente
PUL   H
RTI          ; Regresar de la interrupción
```

FIGURA 24 – código en ensamblador mínimo para representar el listado 1.

Si podemos hacer que el compilador de C posicione EvtProcedure(BYTE value) en alguna dirección de la pagina cero en RAM, podríamos obtener resultados muy similares a los mostrados en la figura 25 pero esto dependerá principalmente en el compilador y la configuración del mismo durante el proceso de desarrollo.

El apuntador *EvtProcedure se inicializa de manera estática por el siguiente constructor:

```
static void (* EvtProcedure) (register BYTE value) =
CommDrvDefaultProc;
```

CommDrvDefaultProc () es una función privada definida como CommDrv la cual hace nada ya que simplemente inicializa el apuntador *EvtProcedure tal y como se muestra a continuación

```
static void CommDrvDefaultProc (register BYTE value) { (void) value; };
```

Al utilizar la función CommEventProc(), la aplicación puede mutar el comportamiento del ISR del SCI a voluntad tal y como se muestra en la implementación del software en esta nota de aplicación.

3.4 LA INTERFASE DE OPERACIÓN DEL MODEM

La implementación de PPP/UDP/IP de esta tesis se construyo al rededor de un MODEM externo "Hayes-compatible". En el pasado, cuando un modem de alta velocidad se decía era aquel que funcionaba a 9600 bauds una compañía llamada Hayes Microcomputer Products Inc. desarrollador un modem que fue ampliamente recibido por la comunidad de usuarios de microcomputadoras. Los comandos seriales utilizados por estos modems se convirtieron en el estándar de-facto en la industria. Dada su popularidad y por cuestiones de compatibilidad, en estos días la gran mayoría de los modems son "Hayes" compatibles.

3.5 OPERACIÓN DE UN HAYES-COMPATIBLE MODEM

Un modem Hayes siempre opera en uno de los dos estados posibles:

- Modo de Comandos
- Estado on-line

Cuando el modem esta en modo de comandos, las instrucciones recibidas a través de su puerto serie son interpretadas como comandos "Hayes". Por ejemplo, uno podría darle una instrucción al modem para que marque un número telefónico o ignore llamadas entrantes a través de simples comandos ASCII. Estos comandos son procesados por el modem y nunca transmitidos por la línea telefónica.

En el estado On-Line una vez que la conexión telefónica se logra con otro modem en el otro extremo de la línea (por ejemplo un ISP), el modem local entra a un estado de on-line y de ninguna manera procesa o interpreta la información que recibe a través de su puerto, por el contrario, todo lo que enviemos al puerto serie del modem será transmitido al modem remoto. Si el modem remoto cuelga, o por alguna otra razón la conexión telefónica se pierde el modem regresara a su estado de modo de comandos.

Cuando un modem "Hayes" recibe un comando (en modo comando por supuesto), este retorna un código ASCII como resultado. Este código puede ser en la forma de una cadena de caracteres o un código numérico. Un código numérico es más simple que una cadena de caracteres inclusive para los sistemas embebidos, pero si quisiésemos controlar el modem a través de una Terminal y un teclado, el modo verbal o mensajes de texto serian más preferibles

que los códigos numéricos. Nosotros podemos establecer el tipo de código de resultado utilizando un comando Hayes.

Tabla 5. Sumario de los códigos de resultado.

Número	Equivalente Verbal	Descripción
0	OK	El comando fue ejecutado.
1	CONNECT	La conexión telefónica se estableció.
2	RING	Señal Ring (timbre) detectada.
3	NO CARRIER	línea telefónica perdida o sin señal.
4	ERROR	Comando invalido, checksum, error en la línea de comando, o comando demasiado largo.
5	CONNECT 1200	conexión establecida a 1200 bps.
6	NO DIALTONE	Señal para marcar no detectada.
7	BUSY	línea ocupada en el otro extremo.
8	NO ANSWER	No hay respuesta cuando se marca al modem remoto.
9	CONNECT 2400	conexión establecida a 2400 bps.

Todos los comandos comienzan con el prefijo AT, a menos que se especifique lo contrario. La gran mayoría de los comandos se dan en una sola cadena de caracteres (o en una sola línea). Los comandos Hayes proporcionan un grupo de comandos bastante comprensibles para configurar un modem, marcar números telefónicos y contestar llamadas entrantes. Este documento muestra en su implementación la manera de como iniciar llamadas solamente, pero hacer que el software conteste las llamadas entrantes no debería de ser tan difícil si se utilizan los comandos apropiados hacia y desde el modem.

Nota.- A pesar que el termino "Hayes compatible" es un termino utilizado en este documento, en realidad no existe un estándar que defina la interfase de comunicación con un modem. No todos los modems Hayes funcionan de la misma manera.

El software de implementación de este trabajo asume la configuración y comportamiento del modem tal y como de enlista a continuación en la Tabla 6:

TABLA 6. Configuración del modem utilizada en la implementación de esta tesis.

Requerimiento	Comando Hayes Estándar
No desempeñar eco de caracteres en modo comando.	ATE0
El Modem retorna códigos de resultado.	ATQ0
Resultados en modo verbal.	ATV1
No desconectarse por un "long space".	ATY0
Monitorear la presencia de una portadora de datos.	AT&C1
Colgar y asumir modo de comandos cuando una transición de on-a-off ocurre en la línea DTR.	AT&D2

Desde el punto de vista del sistema basado en el MC68HC908GP32, el modem externo (Hayes compatible) es simplemente un dispositivo serial conectado al puerto SCI del Microcontrolador. Desde el punto de vista del software, el código fuente para manejar el modem utiliza los servicios proveídos por el modulo CommDrv. Las conexiones necesarias para conectar el modem con el sistema incluyen la tierra del sistema (GND), la línea de transmisión (TX) y recepción (RX).

El modem cuenta con un número de señales de hardware estándar de soporte. Solo dos de estas líneas son utilizadas por el sistema expuesto en esta tesis: carrier detect (CD) y data terminal ready (DTR). Estas dos líneas en conjunto con las tres anteriores hacen un total de cinco líneas las necesarias para manejar el modem del sistema tal y como se muestra en la Tabla 7.

TABLA 7. Conexiones del microcontrolador al conector serial DB-9.

Numero de Pin	Nombre	Descripción	Nombre Pin
1	CD	Carrier Detect	PUERTO D pin 1.
2	RxD	línea de Recepcion	Receptor SCI.
3	TxD	línea de Transmision	Transmisor SCI.
4	DTR	Data Terminal Ready	PUERTO D pin 0.
5	GND	Tierra	Tierra del sistema.

Debe notarse como el puerto serie (SCI) del microcontrolador maneja las líneas de transmisión y recepción del modem de manera directa mientras que dos líneas extras de GPIO (puertos de entrada y salida de uso general) manejan las señales de DTR (data terminal ready) y CD (carrier

detect). La señal DTR se requiere para terminar una llamada cuando el modem se encuentra en el estado On-Line y regresar al modo de comandos cuando ocurre una transición de caída (de on a off). La señal CD se monitorea por el software para detectar si el modem se encuentra en modo de comandos (CD = 1) o en modo On-Line (CD = 0).

El manejador del modem utiliza los mismos servicios del puerto serie del microcontrolador. Debido a esto, la implementación del manejador del modem cuenta con su propia rutina de servicio de interrupción para procesar cada uno de los caracteres que se reciben desde el modem. Una vez que el modem entra en modo On-Line, la rutina de servicio de interrupción del modem se remueve del ISR del SCI. Esto permitirá posteriormente la instalación del manejador de interrupciones correspondiente al enlace punto-a-punto (SLIP o PPP) durante la ejecución del programa.

La rutina de servicio de interrupción del modem simplemente agrega los caracteres que recibe del modem en una cola o queue. Por defecto el número máximo de caracteres que pueden guardarse en la cola son 32 bytes. Esta cola de caracteres se desempeña como una FIFO (first-in first-out buffer) y la mayoría de las funciones de interfase del modem la utilizan. Una cola de caracteres como la utilizada por el manejador del modem cuenta con dos apuntadores: uno apunta a la siguiente dirección libre para agregar caracteres y en otro apunta al primer caracter a remover de la cola. Esta operación se describe en la figura 25.

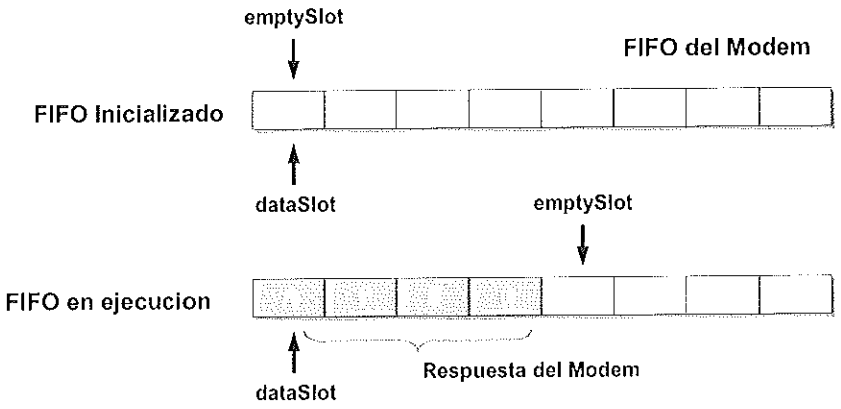


FIGURA 25. Implementación de una lista tipo FIFO en el manejador del Modem.

La figura 25 muestra ambos apuntadores internos a la implementación FIFO. Al momento de inicialización, ambos apuntadores se igualan a cero, esto indica que la cola se encuentra vacía. Una vez que un caracter se recibe a través del puerto serie, este se guarda en la localidad

apuntada por el apuntador *emptySlot* antes de que este se incremente en uno. La figura también muestra ATE0 como respuesta del modem la cual se guardo previamente en la cola a través del proceso previamente descrito. Nótese como ahora emtySlot apunta a la siguiente localidad disponible en la cola FIFO. El apuntador dataSlot presenta un comportamiento muy similar al de emptySlot. Para leer un caracter de la cola FIFO, la aplicación invoca la función ModemGetch() para leer la letra "A" la cual es apuntada por el apuntador dataSlot, después este se incrementa por uno. En este punto, dataSlot apunta ahora a la letra "T". Este proceso se repite para cada caracter leído de la cola FIFO.

El código para agregar un caracter a la cola FIFO se ilustra en el siguiente listado de la figura 26.

LISTADO 2

```
#define MODEM_BUFFER_SIZE 32    // Tamaño del buffer del modem

volatile BYTE mDataSlot = 0;    // Apunta al siguiente caracter disponible
volatile BYTE mEmptySlot = 0;   // Apunta a la siguiente ranura libre en el FIFO

static BYTE *ModemBuffer;       // Apuntador al buffer del Modem

void ProcModemReceive (BYTE c) {
        ModemBuffer [mEmptySlot++] = c;          // Guarda el caracter en el FIFO
        if (mEmptySlot > MODEM_BUFFER_SIZE) {    // Revisar posible sobrecarga
                mEmptySlot = 0;                  // La FIFO es circular
  }
}
```

Figura 26. Código para capturar un caracter en la FIFO del modem.

El listado de código muestra el servicio de interrupción del modem el cual debe ser llamado desde el ISR del manejador del SCI.

El método descrito permite una gran flexibilidad en el manejo de la cola FIFO. Por ejemplo, para determinar el número de caracteres que se encuentran guardados en la cola FIFO el software

solo necesita restar los apuntadores dataSlot de emptySlot. Otro ejemplo de flexibilidad es la habilidad para desechar todos los caracteres de la cola, para esto, simplemente se requiere de igualas dataSlot a emptySlot (dataSlot = emptySlot).

El código para extraer un caracter de la cola FIFO se muestra en la figura 27.

LISTADO 3

```
BYTE ModemGetch (void) {
BYTE c = 0;
  if (mDataSlot != mEmptySlot) {
      c = ModemBuffer [mDataSlot];
      mDataSlot++;
      if (mDataSlot > MODEM_BUFFER_SIZE) mDataSlot = 0;
                          return(c);
  }
  else {
    return (BYTE)0x00;
        }
}
```

Figura 27. Código para remover un caracter de la FIFO del modem.

Otras dos funciones de igual importancia de definen dentro del modulo manejador del modem: las funciones Transmit() y WaitFor(). La primera transmite información hacia el modem mientras que la segunda bloques el flujo de ejecución del programa hasta recibir un caracter o cadena de caracteres en particular. Esta función retorna cuando una recepción valida se cumple o cuando el tiempo de espera expira. La combinación de ambas funciones proporcionan el soporte necesario para ejecutar los complejos scripts requeridos para las sesiones SLIP. Obviamente, esos scripts se ejecutarían desde memoria ROM donde residirían de manera permanente como código compilado con todas las desventajas que eso implica.

3.6 EL MODULO PPP

La implementación en software del modulo PPP proporciona todos los mecanismos necesarios para las negociaciones LCP, PAP y IPCP. Estas negociaciones se realizan a través de una máquina de estados fija en memoria la cual se encuentra definida dentro de la rutina PPPEntry(). Esta máquina de estados es básicamente una máquina pasiva; esta construye los paquetes PPP de respuestas basados en el contenido de los paquetes previamente recibidos. Este nos permite hacer que las negociaciones siempre se desenvuelvan de la manera en que el software lo desee.

El modulo PPP define dos buffers en memoria RAM: el buffer InBuffer[ ] y el OutBuffer[ ]. Por defecto, cada buffer es de 88 bytes de tamaño. El buffer InBuffer guarda los paquetes PPP que se reciben del modem mientras que el OutBuffer guarda los paquetes a transmitir.

A pesar que ambos buffers se definen dentro del modulo PPP estos son globales, ya que su uso se extiende a otros componentes del stack. Cada modulo debe definir la estructura del arreglo de datos que espera observar dentro de los buffers de entrada y salida.

La Figura 28 muestra un mensaje de respuesta a un comando de ECHO (ping) el cual fue procesado por la rutina PPPEntry(). El paquete se encuentra guardado en el arreglo OutBuffer[ ]. Esta función a su vez ejecuta el manejador de paquetes IP el cual pasa un apuntador ip_in al manejador de paquetes ICMP. Dentro de este ultimo manejador, la información ICMP se accesa utilizado el campo "Payload" resultado de una operación "cast" de ICMPDatagram a el OutBuffer. ICMPDatagram es una estructura definida en ICMP.H.

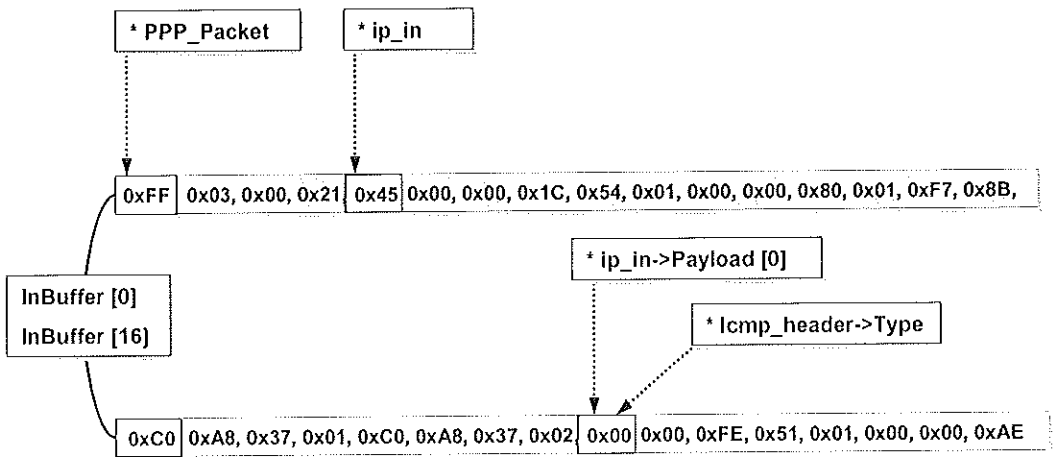


FIGURA 28. Como el arreglo InBuffer[ ] es compartido a través de los diferentes módulos. La respuesta a un mensaje "ping" utilizando PPP, IP e ICMP se muestra en esta figura.

Para llenar el arreglo InBuffer[ ], cada vez que un caracter se recibe a través del puerto serie del microcontrolador, la rutina de servicio de interrupción del SCI debe pasar ese caracter a manera de argumento a la función ProcPPPReceive(). Esta función se encarga de decodificar, completar y validar un paquete completo.

El diagrama de la figura 29 ilustra este proceso.

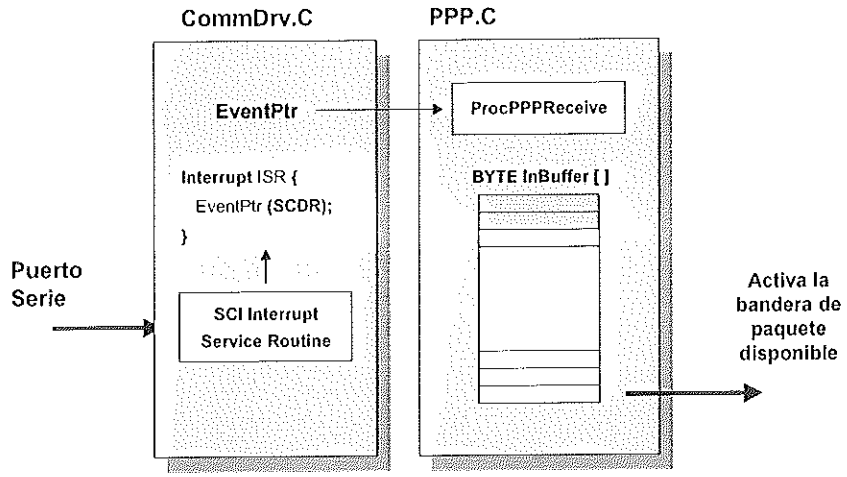


FIGURA 29. El modulo PPP guarda los paquetes recibidos en el arreglo InBuffer[ ].

ProcPPPReceive() actúa para cada caracter recibido. Debido a que la única manera existente para comunicar una rutina de servicio de interrupción con el flujo normal de programa es a través de variables globales, el modulo PPP define una bandera global denominada *PPPStatus*. Cuando un paquete PPP completo dentro del arreglo InBuffer[ ] se encuentra listo para ser procesado, la función *ProcPPPReceive* pone la bandera global IsFrame en uno. Esta bandera es a su vez monitoreada por la función *PPPEntry()* en el ciclo principal del programa.

El codigo fuente de la figura 30 muestra el cuerpo de la función *PPPEntry*.

```
void PPPEntry (void) {

  if (PPPStatus & IsFrame) {    /* Hay un paquete PPP disponible para ser procesado? */

        switch (*(WORD *)(&InBuffer [2])) {  /* Procesar el protocolo especifico */

                case 0xC021:              /* Manejador LCP */
                        HandleLCPOptions ();
                break;

                case 0xC023:              /* Manejador PAP */
              HanldePAPPackets ();
                break;

                case 0x8021:   /* Manejador IPCP */
                   HandleIPCPOptions ();
                break;

                case 0x0021:              /* Manejador del protocolo IP */
                   IPHandler ((IPDatagram *)&InBuffer [4]);
                break;

                default:
                break;
      };
      PPPStatus &= ~IsFrame;  /* Desechar este paquete PPP */
      PPPStatus |= ReSync;     /* Re-sincronizar el encapsulador PPP */
   }
}
```

FIGURA 30. El cuerpo de la función PPPEntry,

Cuando un paquete de recepción PPP es validado, PPPEntry() verifica el campo del protocolo para después invocar su manejador correspondiente. Si se desease implementar nuevos manejadores de protocolos estos deberán ponerse dentro de la sentencia switch.

La bandera IsFlag se limpia automáticamente al final del procesamiento del paquete. Esto es necesario para evitar sobre-escrituras al mismo cuando un paquete nuevo comienza a recibirse aun antes de terminar de procesar el paquete previo. Limpiando la bandera IsFrame le dice a la rutina ProcPPPReceive que esta puede comenzar a recibir un nuevo paquete. Para esto, esta ultima rutina debe verificar la recepción de un caracter 0x7F (el comienzo de un paquete PPP). La bandera ReSync es la que se utiliza para instruir al receptor de paquetes PPP que debe esperar hasta recibir un nuevo paquete PPP.

3.7 IMPLEMENTACIÓN DEL PROTOCOLO DE INTERNET

Los paquetes PPP que encapsulen a uno IP se procesan de acuerdo al manejador presente dentro de la sentencia switch de la función PPPEntry(). En realidad el manejador de paquetes IP es bastante simple: solo se verifica la dirección IP destino para determinar si el paquete que recibimos es igual a la dirección IP de nuestro sistema. Despues el manejador simplemente pasa el paquete al protocolo de la siguiente capa tal y como se muestra en la figura 31.

```
void IPHandler (IPDatagram *ip) {

    /* Comparar nuestra direccion IP con la de paquete recibido */
        if (!IPCompare ((BYTE *)&ip->SourceAddress[0]) {
            /* Un paquete mal enrutado o fue recibido */
        }
        else
                switch (ip->Protocol) {

                        case UDP:   /* Invocar el manejador UDP */
                                UDP_Handler ((UDPDatagram *)&ip->SourceAddress [0]);
                        break;
```

```
                    case TCP:    /* Caso para un segmento TCP */
                    break;

                    case ICMP:   /* Llamar al manejador de mensajes ICMP */
                            IcmpHandler ((IPDatagram *)ip);
                    break;

                    default:     /* Protocolo de transporte no soportado */
                    break;
        }
}
```

Figura 31. El manejador de los paquetes IP.

Durante la etapa de inicialización del programa la función IPInit() debe invocarse para así inicializar los buffers de entrada y salida. Los apuntadores ip_in e ip_out son globales, de tal manera que otros módulos de programa puedan utilizarlos para construir o verificar los paquetes IP. Por ejemplo, algunos mensajes ICMP necesitan acceder el campo de TTL del encabezado del paquete IP, o en el caso de TCP o UDP la implementación del manejador necesita acceder la dirección IP fuente y destino para calcular el checksum del pseudo-header.

El manejador de paquetes IP verifica el campo de protocolo contenido en el encabezado IP del paquete para así invocar el manejador correspondiente. Ya que esta tesis solo cubre la funcionalidad de UDP e ICMP solo estos dos protocolos se incluyen dentro del manejador.

En el caso de que un paquete ICMP se reciba, este es el código que se ejecuta, figura 32:

```
switch (ip->Payload [0]) {

        case ECHO:
                Move ((BYTE *)ip, (BYTE *)ip_out, ip->Length);  /* Move ping datagram to output
buffer */
```

```
            /* Intercambiar las direcciones IP fuente y destino en el Output Buffer */
            ip_out->DestAddress [0] = ip->SourceAddress [0];
            ip_out->DestAddress [1] = ip->SourceAddress [1];
            ip_out->DestAddress [2] = ip->SourceAddress [2];
            ip_out->DestAddress [3] = ip->SourceAddress [3];

            ip_out->SourceAddress [0] = ip->DestAddress [0];
            ip_out->SourceAddress [1] = ip->DestAddress [1];
            ip_out->SourceAddress [2] = ip->DestAddress [2];
            ip_out->SourceAddress [3] = ip->DestAddress [3];

            ip_out->Payload [0] = ECHO_REPLY;    /* Este es una respuesta a un ping */
            ip_out->Payload [1] = 0;             /* Pone el codigo ICMP en  0 */
            ip_out->Payload [2] = 0;             /* Inicia el checksum a 0 */
            ip_out->Payload [3] = 0;             /* durante el calculo del mismo */
            /* Calcula el checksum */
            Value = IPCheckSum ((BYTE *)&ip_out->Payload[0], (ip->Length - 20) >> 1);

            ip_out->Payload [2] = (Value >> 8);  /* Agraga el checksum */
            ip_out->Payload [3] = (Value & 0xFF);
            IPNetSend (ip_out);                  /* Enviar el packet a la capa IP */
        break;

        case ECHO_REPLY:
            // El codigo para manejar respuestas de ping
            // va aqui
        break;

        case TRACEROUTE:
        break;

        default:
        break;
}
```

Figura 32. El manejador de paquetes ICMP.

Un mensaje de tipo ECHO proveniente de un nodo o host remoto se conoce comúnmente como una petición "ping". El manejador de ICMP procesa dichos mensaje de manera muy simple. El manejador simplemente intercambia las direcciones IP fuente y destino además de cambiar el tipo de mensaje por el de un ECHO_REPLY. Antes que el nuevo paquete sea retransmitido de regreso al host remoto, se debe calcular el checksum de la información ICMP.

La implementación de UDP no es tan diferente a la de ICMP. Sin embargo, dado que casi en su totalidad todo el procesamiento de paquetes UDP se realiza a nivel de la aplicación, el modulo UDP soporta el uso de una función de tipo CALLBACK para el procesamiento de datos provenientes del canal UDP.

Cada vez que recibamos un paquete IP a través de la interfase PPP que contenga datos UDP la función CALLBACK especificada por UDPSetCallbackProc() se invoca desde dentro del manejador de UDP. La implementación de UDP especifica una función por defecto. Esta ultima se define dentro del modulo UDP.C como una función de tipo estática. El formato de la función CALLBACK se muestra a continuación:

```
void UDPReceive (BYTE *udp_data, BYTE size_of_data, WORD udp_port)
{
      // Hacer algo aqui
}
```

Ya que en la arquitectura del software presentado en esta tesis no se especifica ni se utiliza ningún tipo de mecanismo de buffering, la información que se pasa a la función CALLBACK a través del apuntador udp_data es en realidad la información que se encuentra físicamente contenida en RAM en el espacio del arreglo InBuffer[ ]. Por esta razón, estos datos deben procesarse tal cual en el preciso momento de la recepción. Además no existe el riesgo de la recursividad ya que el arreglo InBuffer y el colector de paquetes PPP de bloquean de toda operación por la función PPPEntry() por medio de la bandera IsFrame

CAPÍTULO IV

RESULTADOS

El código fuente compilado resultante y sus efectos en los recursos de memoria disponibles en el microcontrolador de muestran de manera numérica a continuación:

TABLA 8. Estadísticas del código compilado resultante.

Segmento	Dirección de comienzo	Dirección de terminación	Tamaño (bytes)
Non-initialized data in zero page RAM	0x0040	0x0044	4
Non-initialized data in RAM	0x0067	0x0128	193
Program code	0xB000	0xC513	5395
Program initialized RAM	0x0045	0x0066	33
Text string and constants	0xC53E	0xC7C8	650
Vector table	0xFFDC	0xFFFF	35
Total RAM			230
Total ROM			6080

CAPíTULO V

CONCLUSIONES

5.0 LA APLICACIÓN FINAL

El sistema basado en el microcontrolador MC68HC908GP32 es capaz de conectarse a la Internet a través de un modem a 9600 bauds y aun mantener un considerable número de recursos para realizar las otras tareas para las cuales fue concebido del sistema. El CPU de la familia M68HC08 cuenta con un poderoso grupo de instrucciones máquina y de modos de direccionamiento para soportar la programación del lenguaje C. El codigo fuente presentado en esta tesis puede optimizarse tanto en tamaño como velocidad utilizando las características únicas del CPU para soportar el lenguaje C (sin mencionar en grado de optimización que podría alcanzarse si se utilizase el lenguaje ensamblador).

En número de aplicaciones donde este tipo de solución de conectividad a Internet son bastas, los recursos aun disponibles dentro del microcontrolador son entre otros dos puertos SPI, dos temporizadores de dos canales, 7 canales del convertidor análogo-digital, un reloj de tiempo real, una interfase a teclado y mas de la mitad de memoria RAM y ROM.

La programación de sistemas con conectividad a Internet puede ser complicada en la mayoría de las ocasiones, especialmente cuando el programador cuenta con poca o ninguna experiencia previa con los aspectos internos del conjunto de protocolos que conforman la especificación de TCP/IP. Esta tesis comprueba a manera de introducción que la conectividad a Internet es no sólo posible si no ya una realidad en la actualidad.

A manera de conclusión, cuando se tiene acceso a las herramientas, utilerías e información apropiadas con un poco de imaginación, obtener el conocimiento necesario para crear aplicaciones compatibles con la Internet pueden lograrse fácilmente.

El software presentado en este trabajo puede utilizarse como referencia para aplicaciones más profesionales y serias. Mejorar el software debería ser una tarea relativamente accesible.

5.1 RECOMENDACIONES

El esquema de manejo de arreglos como buffers podría ser lento e inapropiado para un microcontrolador relativamente pequeño, pero se ha probado a través de esta tesis que el MC68HC908GP32 lo soporta fácilmente. Además, no existe una razón lo suficientemente fuerte para ignorar la técnica de procesamiento de byte-por-byte si así se desea. El CPU podría procesar la información PPP sin la necesidad de guardar los encabezados y colas de los paquetes reduciendo el número localidades de memoria RAM para procesar la información de un paquete.

Lo mismo aplicaría para los paquetes de salida (a transmitir) cuando exista la suficiente información en memoria como para reproducirlos a discreción. Tal vez, este seria el trabajo de una estructura en C llamada SOCKET el cual funcionaria por encima de la implementación PPP y UDP. Solo seria cuestión de sentarse, planear, codificar y experimentar. Un ingeniero con algo de imaginación y un sistema basado en microcontrolador listo para conectarse a la Internet podría ser una combinación bastante interesante.

BIBLIOGRAFÍA

RFC 1071 Computing the Internet checksum. R.T. Braden, D.A. Borman, C. Partridge. Sep-01-1988.

RFC 1122 Requirements for Internet hosts - communication layers. R.T. Braden. Oct-01-1989.

RFC 1123 Requirements for Internet hosts - application and support. R.T. Braden. Oct-01-1989.

RFC 0768 User Datagram Protocol. J. Postel. Aug-28-1980.

RFC 0867 Daytime Protocol. J. Postel. May-01-1983.

RFC 1547 Requirements for an Internet Standard Point-to-Point Protocol. D. Perkins. December 1993.

RFC 0950 Internet Standard Subnetting Procedure. J.C. Mogul, J. Postel. Aug-01-1985.

RFC 1663 PPP Reliable Transmission. D. Rand. July 1994.

RFC 1700 Assigned Numbers. J. Reynolds, J. Postel. October 1994.

RFC 1962 The PPP Compression Control Protocol (CCP). D. Rand. June 1996.

RFC 1055 Nonstandard for transmission of IP datagrams over serial lines: SLIP. J.L. Romkey. Jun-01-1988.

RFC 1144 Compressing TCP/IP headers for low-speed serial links. V. Jacobson. Feb-01-1990.

RFC 1332 The PPP Internet Protocol Control Protocol (IPCP). G. McGregor.May 1992.

RFC 1334 PPP Authentication Protocols. B. Lloyd, W. Simpson. October 1992.

RFC 1570 PPP LCP Extensions. W. Simpson. January 1994.

RFC 1661 The Point-to-Point Protocol (PPP). W. Simpson, Editor. July 1994.

RFC 0791 Internet Protocol. J. Postel. Sep-01-1981.

RFC 0792 Internet Control Message Protocol. J. Postel. Sep-01-1981.

RFC 0793 Transmission Control Protocol. J. Postel. Sep-01-1981.

RFC 1662 PPP in HDLC-like Framing. W. Simpson, Editor. July 1994.

RFC 1989 PPP Link Quality Monitoring. W. Simpson. August 1996.

RFC 1994 PPP Challenge Handshake Authentication Protocol (CHAP). W. Simpson. August 1996.